

# Coupled Oscillator Networks for Solving Combinatorial Optimization Problems in CMOS Technology

**Gekoppelte Oszillator-Netzwerke zum Lösen von kombinatorischen Optimierungsproblemen  
in CMOS Technologie**

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)

Genehmigte Dissertation von Markus Sebastian Graber

Tag der Einreichung: 09.09.2024, Tag der Prüfung: 18.11.2024

1. Gutachten: Prof. Dr.-Ing. Klaus Hofmann

2. Gutachten: Prof. Dr. Aida Todri-Sanial

Technische Universität Darmstadt, Darmstadt, 2024



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



Integrierte  
Elektronische  
Systeme

Electrical Engineering and  
Information Technology  
Department

Integrated Electronic  
Systems Lab

Coupled Oscillator Networks for Solving Combinatorial Optimization Problems in CMOS Technology  
Gekoppelte Oszillator-Netzwerke zum Lösen von kombinatorischen Optimierungsproblemen in CMOS Technologie

Accepted doctoral thesis by Markus Sebastian Graber

Date of submission: 09.09.2024

Date of thesis defense: 18.11.2024

Technische Universität Darmstadt, Darmstadt, 2024

Bitte zitieren Sie dieses Dokument als:

URN: urn:nbn:de:tuda-tuprints-288120

URL: <https://tuprints.ulb.tu-darmstadt.de/28812>

Jahr der Veröffentlichung auf TUprints: 2024

Dieses Dokument wird bereitgestellt von tuprints,  
E-Publishing-Service der TU Darmstadt

<https://tuprints.ulb.tu-darmstadt.de>

[tuprints@ulb.tu-darmstadt.de](mailto:tuprints@ulb.tu-darmstadt.de)

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:

Namensnennung – Weitergabe unter gleichen Bedingungen 4.0 International

<https://creativecommons.org/licenses/by-sa/4.0/>

This work is licensed under a Creative Commons License:

Attribution–ShareAlike 4.0 International

<https://creativecommons.org/licenses/by-sa/4.0/>

*«It seems that perfection is reached not when there is nothing left to add, but when there is nothing left to take away», Antoine de Saint-Exupéry*

## **Acknowledgement**

I want to thank Prof. Dr.-Ing. Klaus Hofmann for giving me this opportunity and for his support and guidance over the past years.

I want to thank the following persons for proofreading (parts of) the thesis:

Nico Angeli, Dominic Korner, Andreas Kramer, Maximilian Naneder, Thomas Schrauth, and Beate Throm

I want to thank Tim Lukas Kessel and Malte Nilges for their help with the development of the test & measurement PCBs including its firmware.

I want to thank the German Research Foundation (DFG) for (partially) funding this work in project ‘*COBRA: CMOS Oscillator Based Rapid Annealing Computing*’, grant number 496307198.

---

Large language models were used to enhance the language of this thesis. Grammarly by Grammarly, Inc. was used for spell-checking. ChatGPT by OpenAI and DeepL by DeepL SE were occasionally used to enhance the clarity of the writing. Any suggestions of these tools were carefully reviewed and verified to not alter the original content in any way. No content was generated by any of these tools.

---

## Erklärungen laut Promotionsordnung

### § 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

### § 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

### § 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

### § 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 09.09.2024

---

Markus Graber

---

# Zusammenfassung

---

Das Lösen von Optimierungsproblemen ist wichtig, aber auch zeit- und energieintensiv. Dies gilt insbesondere für einige kombinatorische Optimierungsprobleme, die NP-vollständig sind, sodass die benötigte Zeit zum Lösen exponentiell mit deren Größe ansteigt. Ein neuartiger Ansatz, um solche Probleme zu lösen, sind sogenannte Ising-Maschinen. Dies sind physikalische Systeme, die auf das Lösen von Problemen in der Form des Ising-Modells spezialisiert sind. Anstatt digitale Logikschaltungen, die bei klassischen Prozessoren verwendet werden, nutzen diese andere physikalische Ansätze, die potenziell deutlich energieeffizienter sein könnten. Insbesondere gekoppelte Oszillator-Netzwerke sind ein vielversprechender Ansatz, weil diese von Natur aus einen Zustand anstreben, der dem Grundzustand des Ising-Modells entspricht. Somit können sie Optimierungsprobleme in deren zugrundeliegenden phasendynamischen Verhalten lösen.

Diese Arbeit fokussiert sich auf die Umsetzung solcher gekoppelten Systeme mittels komplementärer Metall-Oxid-Halbleiter (CMOS) Silizium-Technologie. Das Ziel ist es, Lösungen nahe an dem globalen Optimum zu finden, und das schnell, energieeffizient und auf kleinster Fläche. Zuerst wird das Verhalten dieser Systeme modelliert und ein Simulationsansatz für solche großen Oszillator-Netzwerke vorgeschlagen.

Als Zweites wird das Design diskutiert, welches auf einem 4-stufigen Ring-Oszillator basiert. Die Interaktion der Oszillatoren wird durch einen aktiven Koppler, der auf differentiellen Transistorstufen beruht, ermöglicht. Die Stärke der Interaktion wird gemäß dem Optimierungsproblem durch dessen Bias-Strom mittels eines 4-bit Digital-zu-Analog-Wandlers konfiguriert. Zusätzlich werden notwendige Schaltungen für die sub-harmonische Synchronisierung, Phasenmessung und Frequenzkalibration vorgestellt. Leider ist die Zuweisung der Variablen und Koeffizienten des Optimierungsproblems zu den physikalischen Oszillatoren und Kopplern des Hardware-Netzwerkes ein zeitaufwendiger Schritt. Deshalb wird ein flexibles Routing-System verwendet, das es ermöglicht, die Oszillatoren auf dem Chip untereinander nach Bedarf zu verbinden. Insgesamt werden zwei Chip-Generationen entwickelt und analysiert, wobei die zweite Generation auf einer Siliziumfläche von  $4.6 \text{ mm}^2$  in einem 28 nm Technologieknoten insgesamt 1.440 Oszillatoren und 11.724 Koppler bereitstellt.

Als Drittes wird eine umfangreiche experimentelle Analyse vorgestellt. Diese verdeutlicht, dass die Wahl der richtigen Koppelstärke und Synchronisationsstärke entscheidend ist, um die bestmögliche Genauigkeit zu erzielen. Probleme können in nur 950 ns gelöst werden und erreichen im Durchschnitt mehr als 94% der optimalen Lösung. Die Berechnung benötigt lediglich  $320 \mu\text{W}$  pro Oszillator. Zufällige Abweichungen der Schaltungselemente sind eine der größten Schwierigkeiten dieses analogen Rechenkonzepts. Während Frequenzabweichungen durch die Oszillator-Kalibrierung kompensiert werden können, ändern sich auch die einzelnen Kopplerstärken und damit auch das eigentliche Optimierungsproblem, das in der Hardware repräsentiert und damit gelöst wird.

Die Ergebnisse zeigen, dass gekoppelte Oszillator-Systeme schnell und energieeffizient Optimierungsprobleme lösen können. Diese Netzwerke tendieren natürlicherweise dazu, Lösungen nahe dem globalen Optimum zu finden. Allerdings ist die mögliche Auflösung der Koeffizienten in einem derartigen analogen Ansatz begrenzt. Dieser ist aber besonders gut für Anwendungen geeignet, die Lösungen sehr schnell benötigen und dabei nicht-optimale Ergebnisse tolerieren können. Basierend auf dem vorgestellten Design können zukünftig solche Systeme weiter vergrößert werden, um immer schwierigere Probleme zu lösen.

---

# Abstract

---

Solving optimization problems is important across various domains but can be time-consuming and energy-intensive. For example, optimization is essential to find an optimal schedule for an airline or to manage resources in a complex supply chain. Unfortunately, many combinatorial optimization problems can be NP-complete, which causes the runtime to increase exponentially with the problem size. A new approach to compute such problems are Ising machines, which are specialized physical implementations dedicated to solving optimization problems in an Ising model form. Instead of classical processors based on digital logic circuits, they exploit other physical mechanisms for potentially more efficient computation. Coupled oscillator networks are a promising approach, because they naturally strive towards a state, which corresponds to the ground state of the Ising model. Therefore, they can solve optimization problems within their underlying phase dynamics.

This work focuses on implementing such coupled oscillator networks using complementary metal-oxide-semiconductor (CMOS) technology. The goal is to find near-optimal solutions for optimization problems, while being fast, energy-efficient, and compact. First, a behavioral model and a simulation approach for such large oscillator networks are proposed, which provides insight into the design and reduces the simulation time.

Secondly, the design is discussed, which is based on 4-stage differential ring oscillators. The interaction between the oscillators is established by an active coupler circuit consisting of two differential stages. The coupling strength is determined by its bias current, which is provided by a 4-bit digital-to-analog converter. Additional circuits for the sub-harmonic injection locking, phase measurement, and frequency calibration are also considered. The assignment of the variables and coefficients of the optimization problems to the physical oscillators and couplers in the hardware is a time-consuming task. Hence, a flexible routing network is introduced, which can connect any oscillators across the chip on demand. Two chip generations were fabricated. The second generation has a total of 1,440 oscillators and 11,724 couplers with 4-bit weight resolution and is implemented on  $4.6 \text{ mm}^2$  using a 28 nm technology node.

Thirdly, a comprehensive experimental analysis of the presented chips shows that selecting the right coupling strength and sub-harmonic injection strength is important to achieve the best performance. Optimization problems are solved by the chip within 950 ns and reach on average more than 94% of the optimal solution, while consuming  $320 \mu\text{W}$  per oscillator node. However, random manufacturing mismatch is one of the main challenges of such an analog implementation. While a calibration scheme compensates for the frequency variations of the oscillators, the individual variations in the coupling strength slightly alter the hardware-represented optimization problem.

Overall, the results highlight the fast and efficient computing capabilities of coupled oscillator networks. These networks tend to find solutions near the global optimum, but the analog computing principle limits the possible weight resolution that can be represented in hardware. They are particularly well-suited for applications that require rapid computations and can tolerate non-optimal solutions. Based on the presented design, such systems can be further scaled up to address even more challenging optimization problems.

---

# Contents

---

<b>List of Abbreviations</b>	<b>xv</b>
<b>Glossary</b>	<b>xvi</b>
<b>List of Symbols</b>	<b>xvii</b>
<b>List of Schematic Symbols</b>	<b>xviii</b>
<b>1. Introduction</b>	<b>1</b>
1.1. Motivation . . . . .	1
1.2. Research Scope and Design Objective . . . . .	2
1.3. Thesis Outline . . . . .	3
<b>2. The Ising Model and Optimization Problems</b>	<b>5</b>
2.1. Optimization Problems . . . . .	5
2.1.1. P and NP . . . . .	5
2.1.2. Karp's 21 NP-complete problems . . . . .	7
2.1.3. Solvers . . . . .	10
2.2. Ising Model . . . . .	12
2.2.1. Definition . . . . .	13
2.2.2. Applications . . . . .	13
2.2.3. Generalization: Potts Model . . . . .	13
2.2.4. Relevance for Mathematical Optimization . . . . .	14
2.3. Quadratic Unconstrained Binary Optimization . . . . .	14
2.3.1. Definition . . . . .	14
2.3.2. Transformation into the Ising Model . . . . .	15
2.3.3. Application Examples . . . . .	15
2.4. Ising Machines . . . . .	16
2.4.1. Electrical Approaches . . . . .	17
2.4.2. Optical Approaches . . . . .	19
2.4.3. Quantum Approaches . . . . .	21
<b>3. Oscillators and their Ising Machines</b>	<b>22</b>
3.1. Oscillators . . . . .	22
3.1.1. Barkhausen Criterion . . . . .	22
3.1.2. Electrical Oscillators . . . . .	23
3.2. Injection-Locking . . . . .	25
3.2.1. An Everyday Example: Human Sleep-Wake Cycle . . . . .	25
3.2.2. Injection-locking in Electrical Oscillators . . . . .	26



3.2.3. Sub-harmonic Injection Locking . . . . .	26
3.2.4. Super-harmonic Injection Locking . . . . .	27
3.2.5. Applications of Injection Locking . . . . .	27
3.2.6. Coupling . . . . .	28
3.3. Oscillator-based Ising Machines . . . . .	28
3.3.1. Phenomenological Introduction . . . . .	29
3.3.2. Continuous and Discrete Phases . . . . .	29
3.3.3. Theoretical Background on Oscillator-based Ising Machines . . . . .	29
3.3.4. Overall Procedure . . . . .	31
3.4. Conclusion of Oscillators and OIMs . . . . .	32
<b>4. Modeling and Simulation</b>	<b>33</b>
4.1. Kuramoto Model . . . . .	34
4.2. Abstract Model . . . . .	35
4.2.1. Oscillator Models . . . . .	36
4.2.2. ISF-based OIM Model . . . . .	40
4.3. Schematic Simulation Setup . . . . .	45
4.4. Comparison . . . . .	47
4.5. Conclusion of the Modeling and Simulation . . . . .	49
<b>5. System and Circuit Design</b>	<b>50</b>
5.1. System Design . . . . .	50
5.1.1. Overview . . . . .	50
5.1.2. Design Goals . . . . .	51
5.1.3. Network Design & Embedding . . . . .	53
5.1.4. Oscillator Node Overview . . . . .	53
5.1.5. CMOS Technology Node . . . . .	54
5.2. Oscillator . . . . .	55
5.2.1. Oscillator Type . . . . .	55
5.2.2. Single-ended and Differential Signaling . . . . .	56
5.2.3. Frequency . . . . .	56
5.2.4. Number of Stages . . . . .	56
5.2.5. Oscillator Design . . . . .	57
5.2.6. Frequency Calibration . . . . .	58
5.2.7. Output Buffer . . . . .	61
5.2.8. Differential-to-Single Ended Conversion . . . . .	62
5.3. Coupler . . . . .	63
5.3.1. Passive Couplers . . . . .	63
5.3.2. Active Couplers . . . . .	63
5.3.3. Disabled Mode . . . . .	67
5.3.4. Coupling Strength . . . . .	67
5.3.5. Manufacturing Mismatch . . . . .	69
5.4. Sub-Harmonic Injection Locking . . . . .	69
5.4.1. Parameters of the SHIL . . . . .	70
5.4.2. Injection Position . . . . .	70
5.4.3. Injector Circuit . . . . .	71





5.5. Routing . . . . .	72
5.5.1. Signaling . . . . .	72
5.5.2. Routing Network . . . . .	73
5.5.3. Compensation Scheme . . . . .	76
5.6. Spin-Bias Connection . . . . .	78
5.6.1. All-to-all Network Implementations . . . . .	78
5.6.2. Implementation . . . . .	79
5.6.3. Phase Calibration . . . . .	79
5.7. Digital Control & Configuration . . . . .	81
5.7.1. Weight & Configuration Memory . . . . .	82
5.7.2. Phase-to-Digital Converter . . . . .	82
5.8. Physical Design . . . . .	84
5.8.1. Area Breakdown . . . . .	86
5.9. Overview of Firefly & Glowworm . . . . .	86
5.10. Future Features . . . . .	88
5.10.1. Weight Setting DACs . . . . .	88
5.10.2. Bias Calibration & Distribution . . . . .	88
5.10.3. Objective Computing Core . . . . .	89
5.11. Conclusion of the System & Circuit Design . . . . .	89
<b>6. Experimental Setup</b>	<b>91</b>
6.1. Concept . . . . .	91
6.2. Software Flow . . . . .	92
6.2.1. Routing Paths . . . . .	93
6.2.2. Bitstream Generation . . . . .	94
6.2.3. State Extraction . . . . .	94
6.3. Benchmark Problems . . . . .	95
6.4. Embedding . . . . .	97
6.5. Conclusion of the Experimental Setup . . . . .	97
<b>7. Performance Evaluation and Discussion</b>	<b>99</b>
7.1. Fundamental Operation . . . . .	99
7.1.1. Coupling & Locking . . . . .	100
7.1.2. System Level . . . . .	102
7.2. Evaluation Methodology . . . . .	103
7.3. Operating Parameters . . . . .	105
7.3.1. Initial States . . . . .	105
7.3.2. Randomness of the Computing . . . . .	108
7.3.3. Coupling . . . . .	109
7.3.4. SHIL . . . . .	114
7.3.5. Coupling Time . . . . .	121
7.3.6. Frequency Mismatch . . . . .	122
7.4. Spin-Bias Coupling . . . . .	123
7.4.1. Strength Mismatch . . . . .	123
7.4.2. Comparison to Local Connections . . . . .	126
7.5. Routing . . . . .	127
7.5.1. Pairwise Coupling . . . . .	127



- 7.5.2. Pairwise Coupling With Sub-harmonic Locking . . . . . 129
- 7.5.3. Routing Only Connections . . . . . 131
- 7.5.4. Mixed Routing Problems . . . . . 132
- 7.5.5. Routing Conclusion . . . . . 133
- 7.6. Benchmarks . . . . . 133
- 7.7. Comparison with Other Works . . . . . 138
- 7.8. Further Directions . . . . . 140
  - 7.8.1. Post-processing . . . . . 140
  - 7.8.2. Hybrid Approaches . . . . . 141
  - 7.8.3. Application Specific Oscillator-based Solver . . . . . 141
- 7.9. Conclusion of the Evaluation . . . . . 142
  
- 8. Conclusion . . . . . 144**
  
- References . . . . . 146**
  
- List of Own Publications . . . . . 157**
  
- Supervised Theses . . . . . 158**
  
- Supervised Seminars . . . . . 159**
  
- A. Appendix . . . . . 160**
  - A.1. Frequency Calibration . . . . . 160
    - A.1.1. Calibration Algorithm . . . . . 160
    - A.1.2. Calibration Errors . . . . . 160
    - A.1.3. Clock Domain Crossing . . . . . 161
    - A.1.4. Calibration Results . . . . . 162
  
- Chip Gallery . . . . . 162**

---

# List of Figures

---

1.1. Motivation to Use OIMs . . . . .	2
1.2. Outline of this Thesis . . . . .	4
2.1. Overview of Optimization Problem Types . . . . .	6
2.2. The Maximum Cut . . . . .	8
2.3. Planar and Non-Planar Graphs . . . . .	8
2.4. Traveling Salesman Problem Example . . . . .	9
2.5. Linear Spin Example for the Ising Model . . . . .	13
2.6. Classification of Ising Machines . . . . .	17
2.7. Photonic Ising Machine . . . . .	20
2.8. Principle of Coherent Ising Machines . . . . .	20
3.1. Oscillator Structure for the Barkhausen Criterion . . . . .	23
3.2. Example of Typical CMOS Oscillator Types . . . . .	24
3.3. Injection Locking Example . . . . .	26
3.4. Sub-harmonic Injection Locking . . . . .	27
3.5. Exemplary Maximum Cut Problem Using Oscillators . . . . .	29
3.6. Exemplary Maximum Cut Problem of a Ring of Three Oscillators . . . . .	30
4.1. Abstraction Levels of OIM Simulations . . . . .	33
4.2. Kuramoto Model Simulation Example . . . . .	35
4.3. Coupling Behavior Using the Kuaramoto Model . . . . .	36
4.4. ISF Example of a LC Oscillator . . . . .	38
4.5. ISF Example of a Ring Oscillator . . . . .	39
4.6. Non-linearity of the ISF Function . . . . .	40
4.7. Oscillator Setup . . . . .	41
4.8. Extraction for the Coupler Current . . . . .	42
4.9. Oscillator Unrolling for Replica Loads . . . . .	42
4.10.Extracted Coupler Current . . . . .	43
4.11.Comparison of the Proposed Model with the Actual Transistor Circuit . . . . .	44
4.12.Schematic Level Simulation Flow . . . . .	45
4.13.Simulator Step Size and Accuracy . . . . .	47
4.14.Coupling via the Simulators Step Size . . . . .	48
5.1. The Ising Problem and OIM Networks . . . . .	51
5.2. Simplified Overview of an Oscillator Node . . . . .	52
5.3. Overview of an Oscillator Node . . . . .	54
5.4. Schematic of the Oscillator . . . . .	58
5.5. Coupling with Frequency Differences . . . . .	58

5.6. Uncalibrated vs Calibrated Frequencies . . . . .	59
5.7. Frequency Tuning Scheme of Firefly . . . . .	60
5.8. Comparison between the On-Chip and an Ideal Calibration . . . . .	61
5.9. Oscillator Output Buffer . . . . .	62
5.10. Schematic of the Differential-to-Single-Ended Converter . . . . .	62
5.11. Resistively Coupled Oscillators . . . . .	63
5.12. Principle of the Back-to-Back Connected Stages . . . . .	64
5.13. Principle of the Double Differential Stage Couplers . . . . .	64
5.14. Coupler Circuit Used in the Firefly Design . . . . .	65
5.15. Coupler Circuit Used in the Glowworm Design . . . . .	66
5.16. Input Switch Matrix of the Coupler . . . . .	67
5.17. Simulated Impact of the Coupling Strength on the Settling . . . . .	68
5.18. Impact of the Strength on the Dynamic Coupling . . . . .	68
5.19. Scheme of the Injection into the Oscillator . . . . .	71
5.20. Schematic of the Sub-harmonic Injector . . . . .	71
5.21. Schematic of the Routing Receiver . . . . .	72
5.22. Connectivity of the Switch Blocks . . . . .	73
5.23. Routing Block Scheme . . . . .	74
5.24. Delay Issue of the Routing Network . . . . .	75
5.25. Coupling in the Presence of a Delay . . . . .	75
5.26. Pair of Coupled Oscillator with a Delay . . . . .	76
5.27. Oscillator Internal Waveform . . . . .	77
5.28. Delay Compensation Scheme for the Routing . . . . .	77
5.29. System Clocking Overview for Calibration . . . . .	79
5.30. Phase Calibration Principle for Spin-Bias Connections . . . . .	80
5.31. Overview of the Digital Configuration Block . . . . .	81
5.32. Possible Improved Concept Using SRAM . . . . .	82
5.33. Simplified Schematic of the PDC . . . . .	83
5.34. Layout of the Oscillator Tile . . . . .	84
5.35. Layout of the Routing Tile . . . . .	85
5.36. Full-chip Layout . . . . .	85
5.37. Breakdown of the Area of the Building Blocks . . . . .	86
5.38. Die-Shots of Firefly and Glowworm . . . . .	87
6.1. Principle of the Experimental Setup . . . . .	91
6.2. Lab Setup . . . . .	92
6.3. Flow for Benchmarking . . . . .	93
6.4. Exemplary Phase Distribution for State Classification . . . . .	95
6.5. Example of the Benchmark Problem Generation . . . . .	96
6.6. Topology Overview of the Benchmark Problems . . . . .	96
7.1. Transient Behavior of Two Coupled Oscillators . . . . .	100
7.2. Transient Behavior of Anti-Phase Coupling . . . . .	101
7.3. Illustration of an Example Problem . . . . .	103
7.4. Exemplary Computing Runs . . . . .	104
7.5. Control of the Initial Phases . . . . .	105
7.6. Correlation between Neighboring Oscillators vs. Distance . . . . .	106
7.7. Histogram of the Correlation Coefficients between Neighboring Oscillators . . . . .	107

7.8. Spin Decorrelation by Free-Running Oscillators . . . . .	107
7.9. Hamming Distance of the Obtained Solutions . . . . .	109
7.10. Impact of the Coupling Strength . . . . .	110
7.11. Visual Example of the Coupling Strength . . . . .	111
7.12. Example Impact of the Coupler Strength using the 3-node Ring Example . . . . .	112
7.13. Example of the Coupler Mismatch based on the Fight between Couplers . . . . .	113
7.14. SHIL Range for Free-Running Oscillators . . . . .	114
7.15. SHIL in the Presence of Coupling . . . . .	116
7.16. Locking for Higher Order Harmonics . . . . .	117
7.17. Impact of the SHIL Frequency . . . . .	118
7.18. Impact of the SHIL Strength . . . . .	119
7.19. Impact of Very Strong SHIL . . . . .	120
7.20. Ramp-up of the SHIL Signal . . . . .	121
7.21. Impact of the Coupling Time . . . . .	122
7.22. Impact of the Frequency Mismatch . . . . .	123
7.23. Variation and Linearity of the Spin-Bias Connections . . . . .	124
7.24. Mismatch of the Reference Spin Connections . . . . .	125
7.25. Strength comparison between Spin-Bias and Oscillator-to-Oscillator Connections . . . . .	126
7.26. Waveform of Coupled Oscillators with Delay . . . . .	127
7.27. Coupling Success Rate . . . . .	128
7.28. Success Rate and Frequency . . . . .	129
7.29. Exemplary Waveforms of the Locking of an Coupled Oscillator Pair with Delay . . . . .	130
7.30. Direct Comparison Routing vs Local . . . . .	132
7.31. Performance of Mixed Problem with Local and Routing Connections . . . . .	133
7.32. Full Set of Benchmark Problems . . . . .	134
7.33. Comparison between Different Dies . . . . .	136
7.34. Scattering of the Problem Specific Performance per Chip . . . . .	137
A.1. Oscillator Calibration Procedure . . . . .	160
A.2. Frequency Measurement Uncertainty . . . . .	161
A.3. Clock Domain Crossings . . . . .	162
A.4. Frequency Maps after On-Chip Calibration . . . . .	163
A.5. All Tapeouts During my Time at the IES . . . . .	164

---

# List of Tables

---

2.1. Example of Combinations of the Traveling Salesman Problem . . . . .	9
2.2. Exponential Growth of the Traveling Salesman Problem . . . . .	10
4.1. Comparison of OIM Simulation Methods . . . . .	48
5.1. Comparison between Both Chip Generations . . . . .	87
7.1. Anti-phase 3-ring Scenario . . . . .	112
7.2. Comparison of CMOS OIMs . . . . .	139
A.1. Frequency Variations between Dies . . . . .	163

---

## List of Abbreviations

---

AC	Alternating Current
ASIC	Application-Specific Integrated Circuit
CIM	Coherent Ising Machine
CMFB	Common-Mode FeedBack
CMOS	Complementary Metal-Oxide-Semiconductor
COP	Combinatorial Optimization Problem
CPU	Central Processing Unit
D2S	Differential-to-Single-ended Converter
DAC	Digital-to-Analog Converter
DC	Direct Current
DNL	Differential Non-Linearity
FPGA	Field Programmable Gate Array
GPU	Graphics Processing Unit
IC	Integrated Circuit
ISF	Impulse Sensitivity Function
LSB	Least Significant Bit
MEMS	Micro-ElectroMechanical System
MOSFET	Metal-Oxide-Semiconductor Field Effect Transistor
MPW	Multi-Project Wafer
MU-MIMO	Multi-User Multiple-Input-Multiple-Output
NMOS	N-channel Metal-Oxide-Semiconductor field effect transistor
OEPO	OptoElectronic Parametric Oscillator
OIM	Oscillator-based Ising Machine
ONN	Oscillatory Neural Network
PCB	Printed Circuit Board
PDC	Phase-to-Digital Converter
PLL	Phase-Locked Loop
PMOS	P-channel Metal-Oxide-Semiconductor field effect transistor
PPV	Perturbation Projection Vector
PSS	Periodic Steady State
PVT	Process, Voltage, Temperature
PXF	Periodic transfer Function
QUBO	Quadratic Unconstrained Binary Optimization
SB	Switch Block
SHIL	Sub-Harmonic Injection Locking
SoC	System on Chip
SRAM	Static Random-Access Memory
TSP	Traveling Salesman Problem

---

# Glossary

---

## EDA Tools

**Cadence Innovus**© Implementation System by Cadence Design Systems, Inc. A tool for physical implementation of digital circuits.

**Cadence Ocean**© Scripting language for controlling the simulator in products of Cadence Design Systems, Inc. It is closely related to Cadence Skill.

**Cadence Skill**© Scripting language used in products of Cadence Design Systems, Inc.

**Cadence Spectre**© Circuit simulator by Cadence Design Software, Inc. Multiple additions for fast simulation, verification, power, yield, and other analysis types are available.

**SPICE** Simulation Program with Integrated Circuit Emphasis. An open-source circuit simulator developed by the University of California Berkley.

**Verilog-A** A modeling language for analog circuits with continuous time behavior.

## Miscellaneous

**1/f noise** Noise with a power spectral density inversely proportional to the frequency. Usually it is more relevant in the low-frequency spectrum.

**Circadian Rhythm** Natural endogenous oscillation of an organism, with a period of roughly 24 h. For example the sleep-wake rhythm of animals and humans.

**Electroencephalography (EEG)** A method for recording the brain activity by measuring the electrical potential at multiple external electrodes on the scalp.

**FinFET** A 3D topology of a field-effect transistor, where the gate is placed over at least two sides of the channel.

**National Football League (NFL)** Professional American football league.

**Parametron** A resonant circuit proposed by E. Goto, which was used to conduct phase-based computations.

**W/L ratio** Ratio between the width  $W$  and length  $L$  of the transistor channel.

## Optimization

**NP** A complexity class in the field of computational complexity theory. An explanation is given in Section 2.1.1.

**NP-complete** A complexity class for decision problems. An explanation is given in Section 2.1.1.

**NP-hard** A complexity class for decision problems. An explanation is given in Section 2.1.1.

**P** A complexity class in the field of computation complexity theory. Decision problems can be solved within polynomial time by a deterministic Turing machine.

**SAT** Boolean satisfiability problem. An NP-complete optimization problem.

## Thesis Specific

**Anti-phase** Coupling with a phase difference of  $\pi$  ( $180^\circ$ ).

**Firefly** 1<sup>st</sup> generation of oscillator-based Ising machine ASIC developed in this work.

**Glowworm** 2<sup>nd</sup> generation of oscillator-based Ising machine ASIC developed in this work.

**In-phase** Coupling in the identical phase with a phase difference of 0.






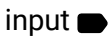
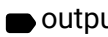



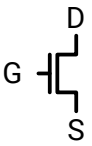
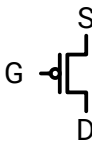
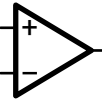







# List of Symbols

---


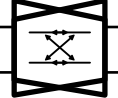

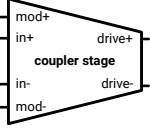



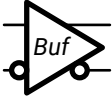
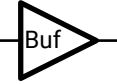
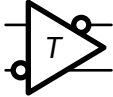



Symbol	Range/Unit	Description
$E$	-	A edge (also called Link) of a mathematical graph. It connects two nodes and has a weight.
$G$	-	A general mathematical graph containing nodes and edges.
$h$	$\mathbb{R}^n$	Weight coefficients of the bias term of the Ising model, introducing a bias for the spins. A single coefficient is $h_i$ .
$H(\sigma)$	$\mathbb{R}$	Hamiltonian of the Ising model indicating the energy of the system, which should be minimized.
$J$	$\mathbb{R}^{n \times n}$	Weight coefficients of the Ising model linking the spins. $J_{ij}$ denotes a single element.
$K_{3,3}$	-	Complete bipartite graph with 6 nodes in 2 sets. See Figure 2.3d.
$K_5$	-	Complete 5 node graph. See Figure 2.3c.
$\mathbf{Q}$	$\mathbb{R}^{n \times n}$	Weight matrix of the quadratic unconstrained binary optimization (QUBO) optimization problem definition. $Q_{ij}$ denotes a single element.
$Q_{ii}$	$\mathbb{R}$	Main diagonal of the weight matrix $\mathbf{Q}$ of the QUBO optimization problem definition.
$Q$	-	Quality factor (Q factor) of an oscillator.
$V$	-	A node (also called Vertex) of a mathematical graph.
$w$	$\mathbb{R}$	The weight of an edge linking two nodes in a graph.
$x$	$\{0, 1\}^n$	Vector of the discrete variables of the QUBO problem formulation. $x_i$ denotes a single element.
$\sigma$	$\{-1, +1\}^n$	Discrete variables for the spin of the Ising model. $\sigma_i$ denotes a single element.
$\phi$	rad or $^\circ$	Phase of an oscillator. Usually, the phase difference between oscillators is considered in this work.
$\omega$	rad s <sup>-1</sup>	Angular frequency of an oscillator. $\omega_0$ denotes the natural frequency of the oscillator.

# List of Schematic Symbols

## Basic Symbols

 Negative Supply	 Positive Supply	 Current Source	 Input Pin	 Output Pin	 Capacitor
 Resistor	 Inductor	 NMOS	 PMOS	 Operational Amplifier	 Digital-to-Analog Converter (DAC)
 Inverter	 Tri-state Buffer	 AND Logic Gate	 Multiplexer	 Clock Divider	 Switch

## Specific Symbols

 Oscillator	 Coupler	 Coupler (simplified symbol)	 Coupler stage	 Coupler stage (simplified symbol)	 Common-Mode-Feedback
 Current Summation	 Differential Buffer	 Differential Buffer (simplified)	 Delay Cell	 Delay Line	
 Sub-harmonic Injector	 Phase-to-Digital Converter (PDC)				

---

# 1. Introduction

---

## 1.1. Motivation

Modern technologies allow us to get information and take action with just a fingertip. For instance, they can suggest the shortest route using public transportation, show available flights to the next vacation destination, provide information about the next game of the favorite sports club, or enable ordering goods directly to the doorstep. Although this is mostly unnoticed by the user, optimization problems are involved in each of these tasks. Finding the shortest route might be simple, but creating timetables for public transportation, cost-effective airline schedules, a suitable match schedule for sports events, or planning a whole logistic chain involves solving complex optimization problems. For example, the seemingly simple problem of planning the National Football League (NFL) match schedule, which involves 17 teams with 256 matches, has over a trillion possible combinations. Before the era of powerful computers, a team of 4 people had 10 weeks to find a sufficient schedule. Nowadays, a high performance computer still takes days to find ‘the best’ schedule that meets all constraints [1]. The schedule of an airline with many flights leads to a much more complex optimization problem with several orders of magnitude more possible combinations. Unfortunately, many optimization problems are NP-complete, so the runtime required to solve such a problem increases exponentially with its size.

The underlying computations within high-performance central processing units (CPUs) and graphics processing units (GPUs) are done by integrated microelectronics. However, the computing power is often the limiting factor and determines which problem can be solved within a feasible time. In 1965, the famous ‘Moore’s Law’ predicted that the number of transistors in an integrated circuit (IC) would double approximately every 1-2 years [2]. As of this writing, a 3 nm technology node is in production and the next 2 nm technology node is expected shortly. However, this trend might eventually come to an end when reaching physical limits such as the size of atoms or the speed of light [3]. Despite the tremendous advances over the last decades, the available computing power still limits the maximum size and complexity of optimization problems that can be solved in a reasonable time. Thus, traditional digital logic circuits might not keep up with the growing demand and complexity of solving optimization problems.

There is growing interest in different (physical) approaches to make solving optimization problems more energy-efficient and faster. The primary goal is to find physical implementations, which naturally seek a ground state. Their natural tendency can be exploited for computing to build a system that represents and thereby solves the optimization problem. Using a device or small circuit, that naturally behaves according to a differential equation could be much more efficient than solving the same differential equation using a whole digital computing core. Hence, such an approach is hoped to be more energy- and time-efficient, than implementing algorithms executed on digital logic. To solve combinatorial optimization problems (COPs), which are often mathematically formulated in the NP-complete Ising model, the research direction of so-called Ising machines is emerging.

The very promising approach of oscillator-based Ising machines (OIMs) has been recently presented [4], [5]. It uses a configurable coupled network of oscillators, which naturally strive towards the ground state. Because this ground state corresponds to the Ising Hamiltonian, it can potentially solve optimization problems following an Ising model form. Figure 1.1 outlines this principle. The application problem is transformed into an Ising model form, which is illustrated as a graph. The oscillator network is configured based on the edges of the graph, so that it represents the optimization problem in hardware. The discrete state of the optimization variables (shown as nodes) is encoded in the phase of the oscillators. After letting the coupled oscillator system naturally strive towards its ground state, the phases are read out and present the discrete states of the computed solution to the optimization problem. Theoretic work and experiments using a printed circuit board (PCB)-based implementation have already been successfully presented. However, such a PCB implementation is a proof-of-concept and not suitable to meet the demands of real-world optimization problems. Providing thousands of densely interconnected nodes would result in impractical large systems. Consequently, a compact implementation as an IC is desired, which can provide enough oscillators for relevant optimization problems. Such an IC could potentially achieve high computing speeds and low power consumption. Additionally, it can easily electrically interface with digital computing systems and could be, in principle, even integrated on the same die or package in the future.

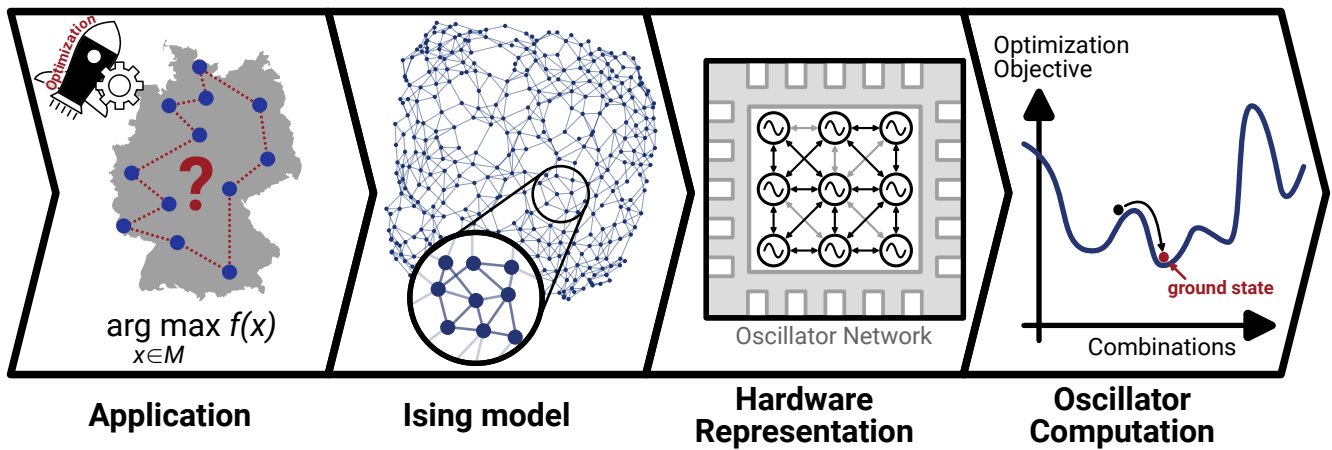


Figure 1.1.: Simplified principle of solving combinatorial optimization problems using configurable coupled oscillator networks. The oscillator interaction within the OIM is configured based on the optimization problem. Only a few nodes of the entire network are shown for better visibility. This coupled oscillator network naturally strives towards a ground state, which corresponds to the ground state of the actual optimization problem.

## 1.2. Research Scope and Design Objective

This thesis presents a new approach to integrating a coupled oscillator network on a single silicon chip. A standard complementary metal-oxide-semiconductor (CMOS) technology is used, which allows for a short time to market with the ability to quickly mass-produce such systems using already available manufacturing capacities. The aim is to enable capable, high-performance OIMs, which can solve combinatorial optimization problems formulated in the Ising machine by finding near-optimal solutions. The main research questions are the following:

## Scientific Research Questions

- **How can oscillators be designed to effectively solve combinatorial optimization problems?**  
For the new approach of OIMs, no design guidelines and very few suitable circuit topologies have been discussed in the literature. Due to the unusual application of oscillators for computing, new design criteria and circuit topologies are needed to meet the unique requirements of OIMs.
- **What are the critical design factors for such systems to find solutions that are as close to the global optimum as possible?**  
Since this concept indirectly solves the optimization problem within its phase dynamics, there is no guarantee that the obtained solution is optimal. Imperfections and random variations will likely result in a sub-optimal solution obtained by this computational approach. Since finding a near-optimal solution is crucial, design factors for successful computing should be identified.
- **How can this approach be scaled up to thousands of oscillators?**  
To keep up with large real-world optimization problems, a coupled oscillator network with sufficient size is needed. Only small prototypes can be made within the scope of such research work. However, a proposed OIM implementation should be able to scale up to any size that still fits on a single die. Thus, challenges when building even larger systems should be identified and potential solutions should be proposed.
- **What are the strengths, weaknesses, and limits of oscillator-based computing?**  
Since the OIMs are a recently emerging trend in computation, the potential of such a novel computing principle should be evaluated. This involves demonstrating the experimental performance of a fabricated system regarding the accuracy, speed, power consumption, and area. Also, the advantages, disadvantages, as well as unique characteristics of this approach should be outlined.

Approaches for the modeling, simulation, system design, circuit design, and physical design of OIMs are discussed. Because this thesis focuses on an actual hardware implementation, the mathematical background of applications, and a software stack to support solving any optimization problem in an Ising form, is outside the scope of the thesis. Only basic algorithms are implemented, which are needed to support the experimental investigation of the performance for solving the Ising model. Furthermore, the requirements of OIMs are considered from an application point of view. Two generations of application-specific integrated circuits (ASICs) were designed, fabricated, and tested in this work to investigate the capabilities of OIMs. The extensive experimental results provide a deep insight into the internal operation and effects of manufacturing mismatch. Last, the computing performance is demonstrated and evaluated using comprehensive benchmarks and the OIM implementation is discussed.

### 1.3. Thesis Outline

The structure of this thesis is outlined in Figure 1.2. Chapter 2 starts with an introduction to combinatorial optimization, the Ising model, and the related quadratic unconstrained binary optimization (QUBO) problems. Additionally, proposed physical implementations of Ising machines are mentioned. The general principle of OIMs is summarized in Chapter 3 after a short summary of the underlying principles of electrical oscillators and injection locking. The modeling and simulation of coupled oscillator networks are discussed in Chapter 4. Available oscillator models were refined for the scope of OIMs and the general simulation approach for

such large networks of coupled oscillators is discussed. The actual system design, starting from the system level down to the transistor level, is presented in Chapter 5. This not only provides details of the oscillator and coupler circuit implementation, but also shows the whole implementation of the two fabricated systems. The experimental setup for silicon performance verification and exploration of the characteristics of OIMs is presented in Chapter 6. A comprehensive experimental analysis is provided in Chapter 7, since a simulation of large networks of coupled oscillators is very time-consuming and behavioral models potentially do not include all physical effects. The chapter investigates the statistical behavior, evaluates suitable operating parameters for the OIM chip, evaluates the flexible routing network, considers manufacturing mismatch, and reviews the potential of OIMs. Finally, this work is concluded in Chapter 8.

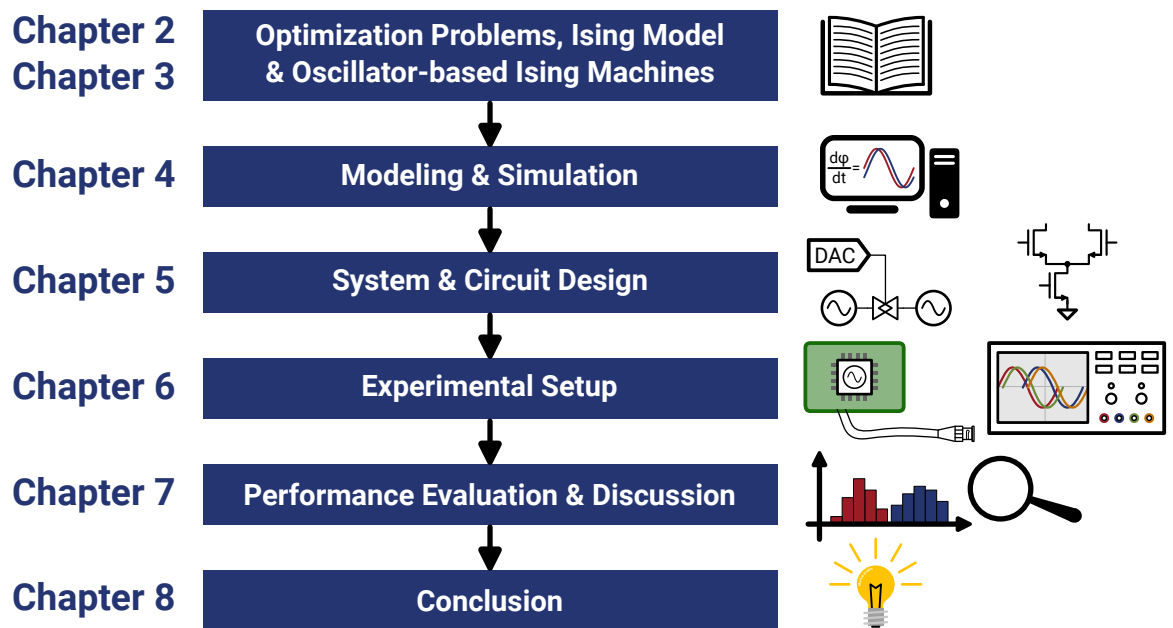


Figure 1.2.: Outline of this thesis.

---

## 2. The Ising Model and Optimization Problems

---

This chapter is a short introduction to the Ising model and mathematical optimization. It provides the necessary fundamentals for the design and evaluation work in this thesis. While the Ising model was originally proposed to describe the spins in a ferromagnetic material, it is much more generally used today. The simple mathematical description of the interaction between spins is a reason for its success, extending its use beyond the ferromagnetic context nowadays. For example, the Ising model is used as a mathematical formulation for combinatorial optimization problems, where binary variables are used. So-called Ising machines are dedicated hardware implementations, which focus on solving optimization problems in the Ising form. Depending on their implementation, they might be limited regarding their problem topology. While any pair of spins could be associated in the mathematical form, a hardware implementation might not provide a physical interaction between any spins. The close to the Ising model related quadratic unconstrained binary optimization (QUBO) problems are also introduced. Additionally, this chapter provides a very short introduction to mathematical combinatorial optimization. The famous traveling salesman problem is presented as motivation due to its intuitive character. The maximum cut problem, which is later used to benchmark and evaluate the design proposed in this thesis is also described. Traditional methods of solving optimization problems are also mentioned to put this work on OIMs into context.

### 2.1. Optimization Problems

Since Ising machines target solving optimization problems, a more general overview is given here. It aims to provide enough background to understand the motivation of this thesis. However, optimization and computational complexity are fields on their own outside the scope of this thesis. A brief overview of different types of optimization problems is shown in Figure 2.1.

#### 2.1.1. P and NP

Although the following definitions are commonly used, the presented definitions are taken from Korte and Vygen [7, Chapter 15.3]. However, an intuitive description is provided here instead of a formal mathematical definition. This fits the scope of this thesis since it is not based on a strict mathematical description, but instead uses this as a motivational aspect.

**Polynomial Time** An algorithm consecutively executes instructions to solve an input problem of size  $n$ . To solve a problem of given size, a certain amount of instructions need to be executed, which depends on the complexity of an algorithm. When the problem size increases, the number of executed instructions also increases. When the increase of instructions has an upper bound by a polynomial of  $n$ , e.g.  $n^3$ , a problem can be solved in polynomial time. If no algorithm exists that can solve a problem in polynomial time, it takes non-polynomial time. The existence of a single algorithm with polynomial time is enough to prove that a

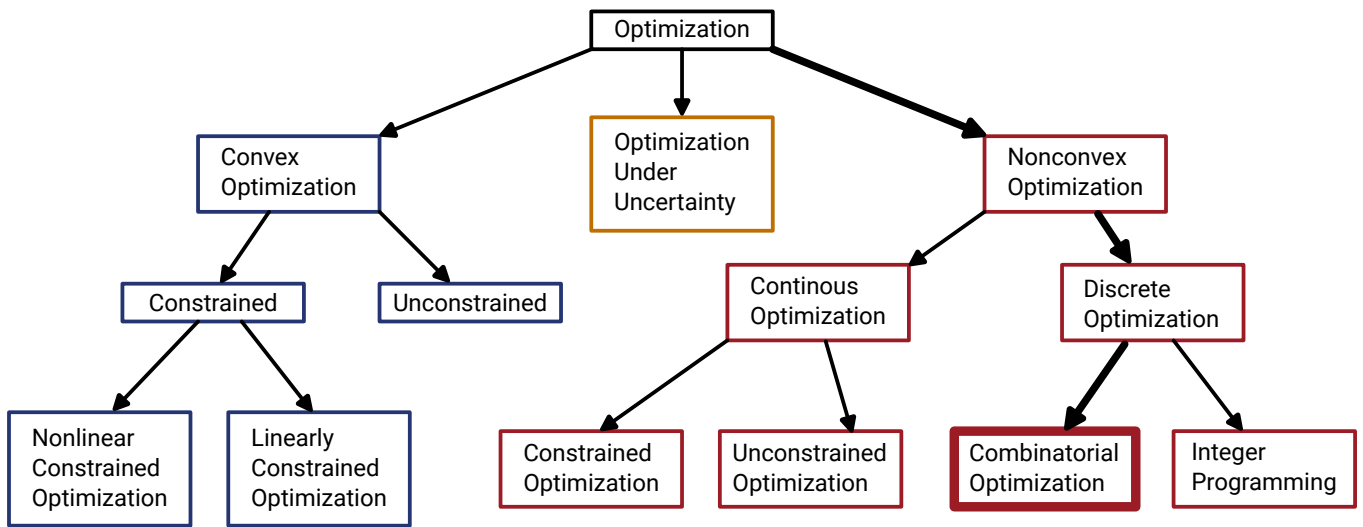


Figure 2.1.: General classification of optimization problems. Subcategories are omitted for simplification reasons. The **bold** path highlights the combinatorial optimization problems relevant for this thesis. This image is adapted and modified from [6].

problem is of polynomial time. A polynomial time transformation into another problem, which can be solved in polynomial time, is also a proof.

**Decision Problem** A decision problem is a problem, which can be either answered with ‘yes’ or ‘no’. In a mathematical context, ‘0’ or ‘1’ is often used instead. For example, this can be asking whether a number is even or odd. Also, this can be asking if a number is prime. A decision problem for a graph can be the existence of an Eulerian path<sup>†1</sup>.

**Complexity Class  $\mathcal{P}$**   $\mathcal{P}$  is the class of all decision problems which can be solved within polynomial time.

**Complexity Class  $\mathcal{NP}$**  A decision problem is in  $\mathcal{P}$  when a proof of the answer can be verified within polynomial time. Such a proof (or certificate) can be a combination, which satisfies the problem question. For example, a question could be if a graph has a Hamiltonian cycle<sup>†2</sup>. When the answer is yes and a Hamiltonian cycle is given, this can be easily verified in polynomial time. All problems in  $\mathcal{NP}$  contain the problems of  $\mathcal{P} \subseteq \mathcal{NP}$ . Hence, a problem in  $\mathcal{NP}$  might be solvable in polynomial time as well. The  $\mathcal{P} \stackrel{?}{=} \mathcal{NP}$  problem is an unsolved problem in computational complexity [8], [9].

**$\mathcal{NP}$ -complete** To be  $\mathcal{NP}$ -complete, every problem in  $\mathcal{NP}$  can be transformed into that problem. As example, Cook proofed the 3-Satisfiable problem as NP-complete [8] . The following list of Karp’s 21 NP-complete can be used as an example as well [10].

<sup>†1</sup>An Eulerian path is a path which visits every edge of the graph exactly once. Nodes can be visited multiple times.

<sup>†2</sup>A Hamiltonian cycle is a path, which visits each node exactly once and starts and ends at adjacent nodes. By adding the edge between those starts and end nodes, a complete cycle would be formed.



---

**$\mathcal{NP}$ -hard** To be  $\mathcal{NP}$ -hard, all problems in  $\mathcal{NP}$ , must be reduceable in polynomial time to it. When assuming that  $\mathcal{P} \neq \mathcal{NP}$ , then no polynomial time algorithm will exist to solve NP-hard problems.

### 2.1.2. Karp's 21 NP-complete problems

A famous set of optimization problems are Karp's 21 NP-complete problems. Those were proven in 1972 by Richard Karp to be NP-complete [10]. This work focuses on the maximum cut problem because it is closely related to the Ising model. A maximum cut optimization problem can be easily mapped (or transformed) into a Ising form, as explained later. Compared to the Ising model, it is more intuitive because a cut of a graph is easy to imagine.

- Satisfiability (SAT)
- 0-1 integer programming
- Clique
- Set packing
- Node cover
- Set covering
- Feedback node set
- Feedback arc set
- Directed Hamilton circuit
- Undirected Hamilton circuit
- Satisfiability with at most 3 literals per clause (3-Sat)
- Chromatic number
- Clique cover
- Exact cover
- Hitting set
- Steiner tree
- 3-dimensional matching
- Knapsack
- Job sequencing
- Partition
- Maximum cut

#### Maximum Cut Problem

The problem consists of a graph  $G = (V, E)$  with nodes  $V$  (also called vertices) and undirected edges  $E$  (also called links) [11]. Each edge has a weight  $w \in \mathbb{R}$ . The graph should be partitioned so that each node is a member of one of two sets. The cut is the sum of edges with weight  $w$  that start in one set and end in the other set. The goal of the maximum cut is to find a partitioning of the graph, whose cut is at least as large as any other cut. In other words, the maximum cut shall be found.

An example using a simple maximum cut problem is shown in Figure 2.2. The graph consists of 5 nodes and 6 edges, as sketched in Figure 2.2a. For simplicity, all edge weights  $w$  are assumed to have an equal value of  $w = 1$  in this example. The two sets are marked as **blue** set and **red** set. Figure 2.2b shows a possible partitioning of the nodes, with nodes 2 and 4 assigned to the **blue** set and the remaining edges to the **red** set. This yields a cut of 3, where the three **green**-marked edges span between both sets. However, this is not the maximum cut for this example. One of the optimal solutions is shown in Figure 2.2c with a cut of 5. A total of four different combinations yield the maximum cut in this example. Nodes  $\{3, 4\}$  and nodes  $\{2, 5\}$  must be grouped together in one set, while node  $\{1\}$  can be in either of the two sets. Additionally, the **blue** and **red** set can be interchanged without affecting the cut. A weighted maximum cut problem is illustrated in Figure 2.2d. The edges now have varying weights, such as 1, 2, or 3, as indicated by the small numbers. Since the cut is the sum of edge weights, which connect the two sets, a maximum cut of 9 is achieved in this example.

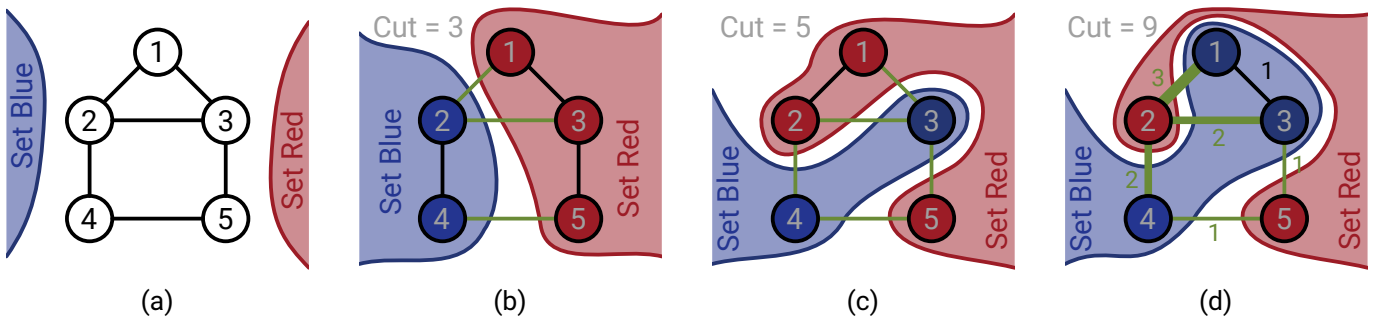


Figure 2.2.: Example of a maximum cut problem. (a) A very simple maximum cut problem with 5 nodes and 6 edges. (b) A non-optimal solution. (c) An optimal solution to the problem. (d) A weighted maximum cut problem with varying edge weights, which are indicated by the small numbers.

**Special Case: Planar Graphs** Although the maximum cut is proven to be NP-complete in general [10], a polynomial time algorithm exists for the special case of planar graphs. The proof by Hardlock transforms a planar maximum cut problem into a maximum weighted matching problem, which can be solved within polynomial time. Since Hardlock’s transformation is of polynomial time, such maximum cut problems can be solved within polynomial time [12]. A graph is planar if it can be drawn in a two-dimensional plane without any intersection between edges. Edges may only touch at their endpoint nodes. An example of a planar graph is shown in Figure 2.3a, opposing to a non-planar graph in Figure 2.3b. A more mathematical definition is provided by Kuratowski’s theorem [13], which states that a graph is planar if it does not contain a subgraph as subdivision of  $K_5$  or  $K_{3,3}$ .  $K_5$ , which is shown in Figure 2.3c, is a graph with 5 nodes, where each node is connected to any other node with an edge.  $K_{3,3}$ , which is shown in Figure 2.3d, is a bipartite graph with two sets of 3 nodes each. Every node is connected to every node of the other set, but no edge exists within the same set. The non-planar graph of Figure 2.3b contains a subdivision of  $K_5$ , which is highlighted in green. The node causing the subdivision is highlighted in red.

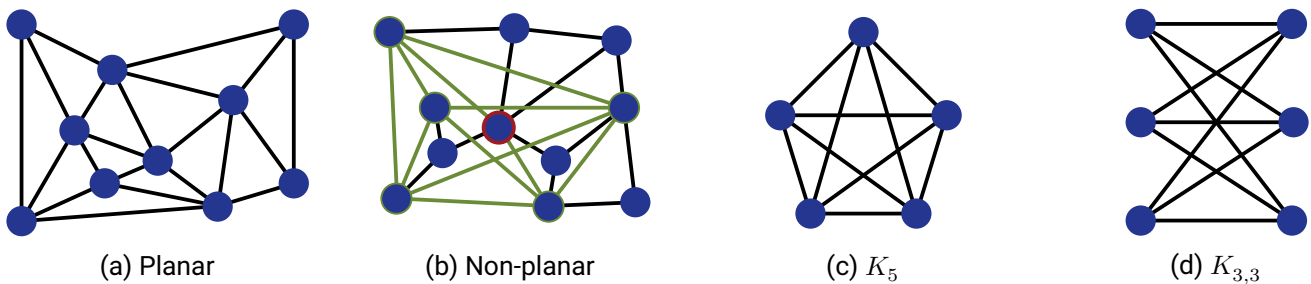


Figure 2.3.: Example of a (a) planar and (b) non-planar graph. (c) The complete 5 node graph  $K_5$ . (d) The complete bipartite graph  $K_{3,3}$  with two sets of 3 nodes each.

**Applications** One application of the maximum cut problem is a via minimization task [14]. The goal is to minimize the number of required vias for an integrated circuit or PCB. A via is here the connection between two conducting layers of metal. Cuts in graphs have been proposed for image segmentation as well [15]. It separates an image into an object and background, based on a user-defined point marking the object and the background.

## Traveling Salesman Problem

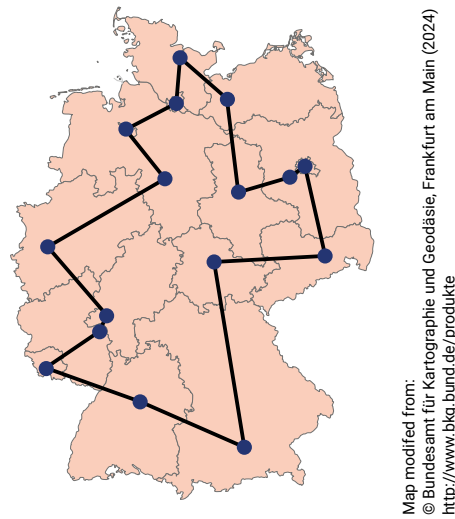


Figure 2.4.: Example of the traveling salesman problem visiting all 16 capital cities of the federal states of Germany. The black path is a possible route, but might not be the shortest.

The traveling salesman problem (TSP) is a famous NP-complete optimization problem. Due to its intuitive understanding, it is used as a brief introduction to combinatorial optimization problems and the exponential growth of possible combinations. Figure 2.4 shows a map with 16 cities. The shortest route should be found, which visits every city exactly once and ends at the starting point. The exponential growth of combinations is shown in Table 2.1. It lists traveling salesman problems for various sizes, chosen by points of interests within Germany. The number of combinations is given in a rounded form to put the focus on the order of magnitude. A theoretical computation time is calculated assuming a hypothetical supercomputer, which has 1000 parallel units, with each unit calculating one combination within one nanosecond. For simplicity, a constant calculation time per problem is assumed ignoring that the needed calculations per combination grow with size as well. For just 16 cities or less, the problem is solved almost immediately in under one second. The computation time is already at 26 cities so long that these cities will likely not exist anymore after checking all combinations. The time needed for the 36 example exceeds the age of the universe. The more fine stepped Table 2.2 helps to grasp the exponential growth. It lists traveling salesman problems for a size of 15 cities up to 25 cities. For sizes of 15, all combinations are tested in milliseconds. When adding 3 cities the computing

Name	Large cities of Hesse	Districts of Darmstadt	Federal capitals	Fabs	Territorial entities of Hesse	University hospitals	World heritage	Large cities	Constituency 2021	Territorial entities
Size	6	9	16	21	26	36	52	82	299	400
Combinations	60	20160	$7 \cdot 10^{11}$	$1 \cdot 10^{18}$	$8 \cdot 10^{24}$	$5 \cdot 10^{39}$	$8 \cdot 10^{65}$	$3 \cdot 10^{120}$	$2 \cdot 10^{609}$	$8 \cdot 10^{865}$
Computation Time	1 ns	20 ns	0.65 s	14 d	245 760 yr	<b><math>1 \cdot 10^{20}</math> yr</b>	<b><math>2 \cdot 10^{46}</math> yr</b>	<b><math>9 \cdot 10^{100}</math> yr</b>	<b><math>5 \cdot 10^{589}</math> yr</b>	<b><math>3 \cdot 10^{846}</math> yr</b>

Table 2.1.: Illustrative examples for the traveling salesman problem. If not otherwise mentioned, the entries apply to Germany. The numbers are only for exemplary purposes to demonstrate the exponential growth and were collected from freely available resources. The computation time is calculated for a hypothetical supercomputer with 1000 parallel cores, where each core needs one nanosecond per combination. The red marked time exceeds the current age of the universe.

time already increases to 3 min. 20 cities would be solved in less than a day, while two more cities would extend the time to almost a year. Already 24 cities would exceed the lifetime of a human being. While a small problem can be quickly solved with brute force, adding a few cities more suddenly makes it unfeasible.

Of course, these numbers and computing time assume brute-forcing all possible combinations. As it will be briefly touched in Section 2.1.3, there exist other more efficient methods to solve such problems. Depending on the approach, the obtained solution might not be the shortest path or there might not be any guarantee that the obtained solution is the shortest possible path. As example, the World TSP<sup>†3</sup> should be mentioned, which has 1,904,711 cities. Based on a lower bound, the current record by Keld Helsgaun from the 15.02.2021 has a gap of 0.0471% or less to the optimal solution [16].

Size	15	16	17	18	19	20	21	22	23	24	25
Combinations	$4 \cdot 10^{10}$	$7 \cdot 10^{11}$	$1 \cdot 10^{13}$	$2 \cdot 10^{14}$	$3 \cdot 10^{15}$	$6 \cdot 10^{16}$	$1 \cdot 10^{18}$	$3 \cdot 10^{19}$	$6 \cdot 10^{20}$	$1 \cdot 10^{22}$	$3 \cdot 10^{23}$
Computation Time	44 ms	0.6 s	10.5 s	3 min	53 min	16.9 h	14 d	296 d	18 yr	409 yr	9830 yr

Table 2.2.: Number of possible combinations and hypothetical computing time. A supercomputer with 1000 parallel cores is assumed, where each core calculates a possible combination in one nanosecond.

### 2.1.3. Solvers

Optimization problems are frequently encountered across various disciplines, so solving them is an important task. This can be in scientific research, natural sciences, engineering as well as countless commercial applications. For example one needs to find the best route for navigation, plan the logistics of goods, or schedule the flight plan of an airline. Such tasks can have considerable economic relevance because limited resources can be used as effectively as possible. Hence, solving such optimization problems is highly relevant and has been a research topic for decades. This section should give a short introduction to mathematical optimization, providing context for the later discussed advantages and disadvantages of the developed OIMs. The mathematical implementations are in the form of algorithms.

As it has been discussed in Section 2.1, there are multiple types of optimization problems. The objective function, which should be minimized or maximized can be convex or non-convex. The non-convex problems usually have multiple local extrema, making it more challenging to find the global optimum. The variables of the optimization problems can be continuous or discrete. Furthermore, constraints might or might not be applied. Overall, the difficulty of optimization problems can vary. Usually, solvers are specialized in certain categories or even just specific problem types.

In general, solvers can be divided into two classes: Exact solvers and approximate solvers. As a key difference, exact solvers find the optimal solution and provide a proof for that. However, the runtime is long and might exceed any feasible time span. So, in practice, they might never find and report the optimal solution within an acceptable time. Approximate solvers are often a form of heuristic or metaheuristic. Heuristic solvers, however, trade off the guaranteed optimal solution with computation speed. Hence, those usually get close to the optimal solution, but might not reach the optimal solution. However, heuristics might be able to provide solutions for problems, which can not be solved with exact approaches within the desired time [17].

<sup>†3</sup><https://www.math.uwaterloo.ca/tsp/world/index.html>, last accessed 08.08.2024

---

## Exact Methods

Exact algorithms find an optimal solution or, respectively, prove that a problem is unbounded or has no solution. Depending on the type of optimization problems, efficient methods are available. For example, the method of the least squares is a parameter minimization of the error for regression. In the case of a linear regression, the solution can be efficiently calculated in polynomial time. Another example is the simplex algorithm, which calculates the solution for linear programs [18], [19]. In practice, it solves linear programs fast, but examples causing non-polynomial runtime have also been demonstrated [20].

To solve a combinatorial optimization problem, one could simply brute-force all possible combinations. As exemplary discussed in Section 2.1.2, this is only possible for very small problems. However, it becomes unfeasible and easily reaches runtimes exceeding the current age of the universe. Other methods have been proposed, which can skip combinations by proving their non-optimality to speed up the computation. While such methods can be significantly faster in practice, they might still be non-polynomial in time [21]. The branch-and-bound method, initially proposed by Land and Doig, is such an approach to speed up the search for an optimal solution [22]. Essentially, it checks all possible combinations systematically in a clever way to guarantee the optimality of the found solution. A tree is recursively built, where upper and lower bounds for its branches are calculated in each step. This allows to skip (or cut) branches, that are worse than the current best solution based to their calculated bounds. This approach has been refined and various improvements such as the branch-and-cut have been proposed [7]. For the previously discussed traveling salesman problem (TSP) problem, a branch-and-cut algorithm exists, that finds the shortest route [23]. This made it possible to solve a TSP problem with 2,392 nodes with a guaranteed optimal solution in the year 1991.

## Heuristics & Approximate Methods

Approximate methods use metaheuristics or heuristics to find an (optimal) solution to a given optimization problem. The handbook of metaheuristics by Glover and Kochenberger describes metaheuristics as

*« solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space. Over time, these methods have also come to include any procedures that employ strategies for overcoming the trap of local optimality in complex solution spaces, especially those procedures that utilize one or more neighborhood structures as a means of defining admissible moves to transition from one solution to another, or to build or destroy solutions in constructive and destructive processes.»* Glover and Kochenberger, [24]

Such methods usually can not guarantee finding the optimal solution and typically do not find the optimum for sufficiently large problems. Examples of such algorithms are scatter search, tabu search, genetic algorithms, and simulated annealing [24], [25].

A good example are (meta)heuristics for the before discussed TSP. The study of McMenemy et al. compares three state-of-the-art solvers using 6 different sets of TSP benchmark problems [26]. Although all three solvers are developed for TSP problems, one heuristic might excel at a certain problem but perform poorly for another problem. For example, the EAX heuristic [27] outperformed the LKH based heuristics in 72.9% of cases in the study [28].

As the ‘no free lunch’ theorem by Wolpert and Macready suggests, a heuristic algorithm will only outperform others for a specific set of optimization problems [29].

*« This theorem explicitly demonstrates that what an algorithm gains in performance on one class of*

---

*problems is necessarily offset by its performance on the remaining problems»*  
Wolpert and Macready, [29]

Consequently, the TSP heuristics perform well for TSP problems, but will result in poor performance in other problem classes. The knowledge about effective strategies to solve a specific problem class benefits those heuristics in such circumstances.

Approximate polynomial time algorithms also exist, which can solve specific optimization problems within a guaranteed bound. An example is the Goemans and Williamson algorithm for the maximum cut problem, which guarantees results that reach at least 0.87856 of the optimal value [30]. They modified their approach for other optimization problems like the maximum directed cut problem and the maximum satisfiability problem.

## Hybrid Methods

While the before discussed methods are either exact or heuristic, hybrid methods trying to combine the advantages of both methods exist as well. The (meta)heuristic algorithms excel with their fast computation but might not find the optimal solution. Finding the optimal solution is not guaranteed, but an obtained solution can act as a bound. The exact algorithms guarantee to find the optimal solution at the price of a much longer runtime. The survey by Tahami and Fakhravar names the advantages that exact methods calculate bounds for the solution and can remove combinations from the search area [17]. While the combination of exact and approximate methods can be either exact or approximate, they report that most implementations are approximate methods.

## 2.2. Ising Model

The Ising model has its roots in the research on magnetism in the 1920s. The physicist Wilhelm Lenz published 1920 his thoughts to explain Curie's law, which tries to give a new view on magnetism in solid bodies and its temperature dependence [31]. His idea is based on elementary magnets inside every atom having a magnetic spin. Each spin has two preferred angles separated by  $\pi$  and stays in other angles very rarely. These spins frequently swap but can align themselves in the presence of an external magnetic field. His student Ernst Ising studied the magnetic spins and came up with a model, which was later named after himself - the Ising model [32]. He models the spins arranged in a crystal lattice with binary states, which only interact locally with their neighbors. Ising provided a solution for a linear lattice, as shown in Figure 2.5, where the magnetic spins are arranged in a line. Unfortunately, the linear case does not show a phase transition as observed in ferromagnetism. This is, however, the case in two dimensions [33], [34]. Although his original description differs from today's well-known format, similar forms have already been used in the 1930s [35], [36].

The success of the Ising model is based on its simple mathematical description, which still maintains the major physical behavior. The actual background in statistical mechanics is less important, as it is applied in a multitude of other fields today. It models how short-range effects between local, neighboring elements affect a transition of the macroscopic system [34]. Mathematically, it can be interpreted as a combinatorial optimization problem.

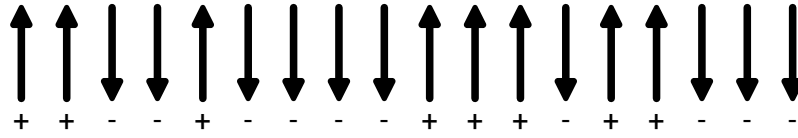


Figure 2.5.: Example of the spins for the Ising model. Redrawn example from Ising’s original publication [32]. The binary spins are drawn as  $\uparrow$  and  $\downarrow$  with the addition of ‘+’ and ‘-’ as used by Ising himself.

### 2.2.1. Definition

The Ising model is formulated as shown in Equation 2.1. The Hamiltonian  $H$ , which represents the energy of a system with  $n \in \mathbb{N}$  spins, shall be minimized.

$$H(\sigma) = - \sum_{\langle i,j \rangle} J_{ij} \cdot \sigma_i \sigma_j - \sum_{\langle i \rangle} h_i \cdot \sigma_i \quad (2.1)$$

$\sigma \in \{+1, -1\}^n$  denotes the discrete spin states. This vector has  $2^n$  possible combinations, which grow exponentially with the number of spins.  $J \in \mathbb{R}^{n \times n}$  with  $J_{ii} = 0$  denotes the interaction between the binary spins. Without an interaction between spins, the coefficient  $J_{ij}$  is set to zero. In the case of the original Ising model, only neighboring elements interact with each other. But long-range connections between distant spins are generally possible as well. Since there is no interaction within the same spin, all elements of the diagonal  $J_{ii}$  are zero. The  $h \in \mathbb{R}^n$  creates a bias for the discrete states, which originally models an external magnetic field. However, some problems like the maximum cut do not need this bias term ( $h_i = 0$ ) and it is not supported by some Ising machine implementations.

The Ising model is NP-hard as proven by Barahona [37]. However, special cases of spin interaction exist which are not NP-hard. For example, a planar graph without the bias term and a 2-dimensional graph with periodic boundary conditions (e.g. a torus) are not NP-hard [37].

### 2.2.2. Applications

Despite the previously mentioned physical origin of the Ising model, it has been used in fields far away from its origin. Essentially, it represents any arbitrary system of coupled binary states. To emphasize the importance of the Ising model and thus Ising machines to efficiently solve such problems, a few examples among various domains are presented. This serves as motivation for the whole thesis. However, it needs to be underlined that these are examples, which can be directly modeled in the Ising form. Other optimization problems can be mapped to the Ising model, which make these suitable for Ising machines as well. For example, Karp’s 21 NP-complete problems can be mapped to the Ising model [38]. In social science, the Ising model has been successfully used for economic opinions, urban segregation, and language change [39]. Additionally, it has been studied as a model for opinion formation in combination with a voter model [40].

### 2.2.3. Generalization: Potts Model

In the Ising model, the spins  $\sigma_i$  have two possible states  $\sigma_i \in \{+1, -1\}$ . The generalized Pott’s model has  $r$  discrete states [41]. In the special case of  $r = 2$ , the Pott’s model is identical to the Ising model. The discrete spin states are then equally placed along a circle so that the spin phases are  $\frac{2\pi n}{r}$  with  $n = \{0, 1, \dots, r - 1\}$ . In the case of the Ising model, the spins can be either parallel or anti-parallel and thus apart by  $\pi$ . However,

in the Pott's form, multiple possible angles exist between the spins. There are two common ways to define the energy between two spins [42]. According to Domb's naming scheme, the *standard* Potts model has two energy levels, which are calculated as  $k \cdot \delta(\sigma_i, \sigma_j)$ .  $\delta$  is the Kronecker delta defined as  $\delta_{i,j} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j \end{cases}$ .

The *planar* Potts model uses the cosine  $k \cdot \cos(\sigma_i - \sigma_j)$  to determine the energy between two spin angles. In the following work the *standard* form is used, when referring to the Potts model.

#### 2.2.4. Relevance for Mathematical Optimization

Some analytical solutions as well as physical interpretations exist, specifically about the phase change behavior. However, this is far outside the scope of this thesis. The Ising model is commonly used as an optimization problem formulation. They are treated as a mathematical definitions of the problems to solve. Hence, its history and physical use are of minor relevance for the following work. Nevertheless, the ferromagnetic origin and the phase change are an intuitive analogy for the here presented OIMs. In their hardware implementation, the oscillators can be understood as spins, which are represented by their own phase. Similar to the physical models, there is often a dominating interaction between neighboring elements, although an all-to-all connectivity is desirable.

### 2.3. Quadratic Unconstraint Binary Optimization

Another commonly used problem definition are the quadratic unconstrained binary optimization (QUBO) problems. They directly relate to the Ising model and can be transformed into each other, as shown in Section 2.3.2. Because it is NP-hard, it is computationally difficult to solve [43]. Several applications can be either directly formulated or mapped to the QUBO form. When transformed into the QUBO form, specific solvers for this type can be used [44]. So, highly optimized algorithms to solve QUBO problems are then essential applied to other problems as well. Furthermore, Ising machines, which are discussed in Section 2.4, can be also used for such QUBO problems.

#### 2.3.1. Definition

The definition of QUBO with  $n \in \mathbb{N}$  discrete variables is as follows:

$$f_Q(x) = x^T \mathbf{Q}x = \sum_{\langle i,j \rangle} \mathbf{Q}_{ij} \cdot x_i x_j \quad (2.2)$$

$x \in \mathbb{B}^n$  denotes the vector with binary states  $x_i \in \{0, 1\}$ .  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  defines the weights of the optimization problem. Variables  $x_i$  and  $x_j$  are joined by  $\mathbf{Q}_{ij}$ . The product  $\mathbf{Q}_{ij} \cdot x_i x_j$  contributes  $\mathbf{Q}_{ij}$  to the sum when both  $x_i = 1$  and  $x_j = 1$  and otherwise 0. In case of elements on the diagonal  $\mathbf{Q}_{ii}$ , this results in  $\mathbf{Q}_{ii}$  for  $x_i = 1$  and otherwise 0. Usually the matrix  $Q$  is symmetric, because  $x_i x_j = x_j x_i$ . The goal of the QUBO is to minimize  $f_Q(x)$ . This means that a vector  $x_{opt}$  need to be found, so that  $f_Q(x_{opt})$  is at least as small as any other possible  $f_Q(x)$ :  $\forall x \in \mathbb{B}^n : f_Q(x_{opt}) \leq f_Q(x)$ . This is equivalent to the maximizing  $f_{-Q}(x) = x^T - \mathbf{Q}x$ .



### 2.3.2. Transformation into the Ising Model

The variables  $x_i \in \{0, 1\}$  of the QUBO and  $\sigma_i \in \{-1, 1\}$  of the Ising model can be transformed into each other. The relation between the variables  $\sigma$  and  $x$  of both problem types can be expressed as:

$$\sigma_i = 2x_i - 1 \quad x_i = \frac{\sigma_i + 1}{2} \quad (2.3)$$

To show the transformation, Equation 2.3 is inserted in the QUBO formulation. A notation of vector  $\vec{1}_n \in (1, 1, \dots, 1)$  is used to express a vector with  $n$  one elements.  $\sigma \in \{-1, +1\}^n$  denotes the spins vector of the Ising model, as introduced in Section 2.2. The matrix  $\mathbf{Q}$  is symmetric so that  $\mathbf{Q}_{ij} = \mathbf{Q}_{ji}$  can be used for simplification.

$$f_Q(x) = x^\top \mathbf{Q} x = \left(\frac{\sigma + \vec{1}_n}{2}\right)^\top \mathbf{Q} \left(\frac{\sigma + \vec{1}_n}{2}\right) = \frac{1}{4} \cdot (\sigma^\top \mathbf{Q} \sigma + \sigma^\top \mathbf{Q} \vec{1}_n + \vec{1}_n^\top \mathbf{Q} \sigma + \vec{1}_n^\top \mathbf{Q} \vec{1}_n) \quad (2.4)$$

$$= \sum_{\langle i, j \rangle} \frac{\mathbf{Q}_{ij}}{4} \cdot \sigma_i \sigma_j + \sum_{\langle i \rangle} \sigma_i \left(\sum_{\langle j \rangle} \mathbf{Q}_{ij}\right) + \sum_{\langle i \rangle} \sigma_i \left(\sum_{\langle j \rangle} \mathbf{Q}_{ji}\right) + \sum_{\langle i, j \rangle} \mathbf{Q}_{ij} \quad (2.5)$$

$$= \sum_{\langle i, j \rangle} \frac{\mathbf{Q}_{ij}}{4} \cdot \sigma_i \sigma_j + 2 \cdot \sum_{\langle i \rangle} \sigma_i \left(\sum_{\langle j \rangle} \mathbf{Q}_{ij}\right) + \sum_{\langle i, j \rangle} \mathbf{Q}_{ij} \quad (2.6)$$

$$= - \sum_{\langle i, j \rangle} \frac{-\mathbf{Q}_{ij}}{4} \cdot \sigma_i \sigma_j - \sum_{\langle i \rangle} \frac{-\sum_{\langle j \rangle} \mathbf{Q}_{ij}}{2} \cdot \sigma_i + \sum_{\langle i, j \rangle} \mathbf{Q}_{ij} \quad (2.7)$$

After multiplying out the equation, the individual terms can be compared with the Ising model form in Equation 2.1. This results in the following transformation of the weights as  $J_{ij} = \frac{-\mathbf{Q}_{ij}}{4}, i \neq j$  and  $h_i = \frac{-\sum_{\langle j \rangle} \mathbf{Q}_{ij}}{2}$ . The constant factor  $\sum_{\langle i, j \rangle} \mathbf{Q}_{ij}$ , which is the sum of all weights, is introduced by that transformation as well. Because  $f_Q(x)$  respectively  $H(\sigma)$  should be minimized for both problem types, this constant does not affect the optimal solution. The objective of the problems can be transformed as  $f_Q(x) = \tilde{H}(\sigma) + \sum_{\langle i, j \rangle} \mathbf{Q}_{ij}$ , where  $\tilde{H}(\sigma)$  is the transformed problem into the Ising model.

Consequently, it is clearly shown that the QUBO and Ising models can be transformed into each other with a complexity of  $\mathcal{O}(n^2)$ . Hence, the Ising model and QUBO formulation of optimization problems can be treated as equivalent in the scope of this thesis. Since such hardware solver implementations are commonly called Ising machines, this thesis focuses on the Ising model. Consequently, the following will not explicitly mention the QUBO formulation in addition to the Ising model. The reader should implicitly include the equivalent QUBO formulation. However, it should be noted that the bias term  $h$  of the Ising model is not always considered by Ising machine implementations. As later discussed, the first chip generation of this work does not support that bias term  $h$  and thus can not solve problems in the QUBO formulation.

### 2.3.3. Application Examples

The survey by Kochenberger lists applications of the QUBO starting from 1968 [44]. As a lot of applications exists, which are outside the scope of this thesis, only a view especially interesting examples are listed. Common applications of the QUBO are in finance, traffic management, economic management, and machine scheduling. A proposed application is the computer-aided layout design [45]. While Karaup et al. mentioned

---

that the technique can be applied to electrical circuits, their focus is on the layout for commercial buildings. For example, the placement of departments inside a factory needs to be carefully planned for high efficiency. Requirements for the distance between all the departments exist because materials should flow in a short distance, and odors and noise affects the quality of the work environment. The QUBO was even used for seizure prediction in [46]. The activity of the brain was recorded in an electroencephalography (EEG) using multiple electrodes to predict seizures. Special characteristics of the recorded brain waveform indicate a possible future seizure. As the authors reported, the electrodes exhibiting these signals for potential seizures vary by patient and between seizures. Choosing the best-suited electrodes is the optimization problem.

Besides applications, which are naturally formulated in the QUBO model, other problem types can be reformulated as QUBO as well. Lewis et al. formulated an optimization problem to allocate tasks in a multi-processor system. Based on the cost of a task-processor assignment and necessary inter-task communication, the goal is to find the best possible allocation of tasks to the processors. As the authors reported, such assignments get difficult when the cost of the inter-task communication is similar to the cost of task-processor assignments.

## 2.4. Ising Machines

Ising machines are specialized hardware implementations to solve optimization problems in an Ising formulation. When treated as a black box, they act similarly to algorithmic solvers discussed in Section 2.1.3. An optimization problem, which is formulated in the Ising model, is given as input and after some time for calculation a solution is provided as output. However, the internal operating principle has little parallels with those discussed solvers. Traditional solvers are based on algorithms, executing before programmed instructions, which applies to exact and approximate approaches. A processor then executes the instructions and eventually terminates, providing the desired solution. Almost every high-performance processor nowadays is implemented in silicon CMOS technologies. Internally, the processor operates with digital signals, meaning low and high voltage levels. On a more abstract layer, boolean equation are implemented using logic gates. However, such a processor is not limited to just a specific algorithm, it can execute any algorithm using the supported instructions for any possible purpose. This is in contrast to the Ising machines, which are purposely build for solving problems in the Ising model. Consequently, Ising machines are defined, in the scope of this work, as purposely built physical computing systems that are specialized in solving optimization problems in an Ising model formulation. An Ising machine might serve other purposes, such as solving other types of optimization problems that can not be transformed into the Ising model. However, solving problems in Ising formulation must be the main purpose.

Usually, Ising machines exploit different physical principles for calculation. These more or less directly perform some sort of a minimization (or maximization) by their natural behavior. It is hoped, that such mechanism will perform faster and more energy efficiently than a traditional algorithm executed on a processor. However, they are limited in terms of the supported optimization problem types and might have hard limits regarding the maximum size and connectivity of a problem. Furthermore, its physical realization might be complicated and not yet commercially established. Implementations of Ising machines in digital CMOS technologies exist as well, which can be in the form of ASICs or field programmable gate arrays (FPGAs). Because of the highly varying capabilities, a meaningful comparison between Ising machines is difficult. Within the scope of this thesis, the following criteria are considered:

- Accuracy of the provided solution
- Runtime to compute a solution

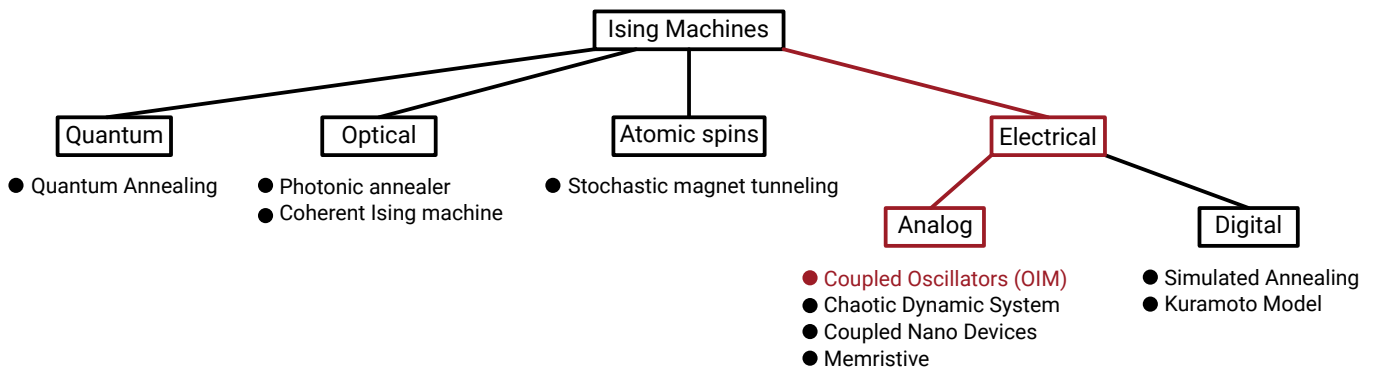


Figure 2.6.: Classification of Ising machines according to their physical implementation.

- Energy consumption of the system
- Physical size and manufacturing cost

The computed solution of an Ising machine should be as close as possible to the optimal solution of the optimization problem, which will be referred to as ‘accuracy’ in the remainder of this thesis. The solution should be obtained quickly while consuming little energy for high throughput and efficiency. Additionally, the physical implementation should be small in size and cheap to manufacture.

In the following, common physical implementations are presented. This helps with the conceptualization of the discussed OIMs in this work, which are mentioned for the sake of completeness. While not all existing implementation concepts are mentioned, an overview is provided. The maturity of the investigated approaches varies. Some approaches have been discussed based on simulation models, some exist as hardware prototypes, and other approaches are already commercially available. A categorization of Ising machines with a focus on the hardware implementation is provided in Figure 2.6. The main physical principles are classified as quantum-based, optical, electrical, or spin-based. A classification according to their general scheme into classical annealing, quantum annealing and dynamic systems as suggested by Mohensi et al. is possible as well [47]. However, a hardware focused classification is more suitable for this work due to its focus on the silicon implementations of OIM systems.

### 2.4.1. Electrical Approaches

Electrical implementations are one of the most common forms of Ising machines. The advantages of CMOS technology should be clear, as it is used in virtually any electrical device today. Besides that, implementations using emerging electrical devices have been proposed, which might be better suited than traditional CMOS implementations. Examples of computing principles are given for digital as well as analog principles.

#### Digital CMOS Implementations

Digital Ising machine implementations are usually specific implementations of an algorithm on an FPGA or ASIC. The goal is to improve the computing speed and energy efficiency compared to a software implementation on a processor. A traditional algorithm executed on a processor would technically fall in this category as well, but it is not considered as Ising machine due to the lack of specialization. Consequently, there is a blurry line between algorithms and digital Ising machines. However, digital Ising machines can be considered as specific

---

hardware accelerators for combinatorial optimization.

**Simulated Annealer** Commercial implementations are available by the companies Fujitsu and Hitachi. The first generation of Fujitsu has up to 1024 all-to-all connected spins [48], the second up to 8,192 spins [49], and the third up to 100,000 spins [49]. They are based on a Markov-chain Monte Carlo stochastic search method [48], [49]. Yamaoka et al., working at Hitachi, showed a 20,000 spin system implemented on a 12 mm<sup>2</sup> chip in a 65 nm node [50]. The group at Hitachi proposed a two chip Ising machine with 30,976 spins [51]. The later work demonstrates Ising machines as multi-chip implementations, which were spread over multiple PCBs [52], [53]. However, each chip implements 16,384 spins in a king's graph<sup>†4</sup> topology and uses a metropolis algorithm for the annealing procedure. Their system consisting of 9 PCBs with 9 chips each has a total of 1.3 million spins, however, the connectivity is sparse. A 512 spin all-to-all connected annealing processor, which can update all spins at once was presented as well [54], [55]. It is based on their new proposed stochastic cellular automata annealing algorithm, which is claimed to perform better than standard simulated annealing. Su et al. proposed two simulated annealing implementations, allowing flexible interconnects [56]–[59]. Kashimata, working at Toshiba, presented a highly parallelized implementation of an digital annealer using 79 FPGAs [60].

**Kuramoto Based** A notable digital implementation is proposed by Sreedhara et al. [61]. They implemented a digital counterpart of OIMs. It is based on the Kuramoto model, which is used as an abstract model of the behavior of coupled oscillators. A detailed explanation of the Kuramoto model is provided in Section 4.1. Essentially, the Kuramoto model of an OIM leads to a system of ordinary coupled equations. The chip solves these equations by using Euler's integration steps and fixed point arithmetic. Their chip implements 33 all-to-all connected spins on a 3 mm<sup>2</sup> chip in a 65 nm technology node. However, this needs more area than a comparable analog OIM implementation.

## Analog

While the digital approaches rely on algorithms, which are more or less efficiently executed on digital logic circuits, the analog implementations exploit the inherent behavior of a device or circuit itself as computing operations. The general motivation is that such principles might be more effective in computing an optimization problem by either being faster and/or more energy efficient than their digital counterparts. Furthermore, they might enable solving larger problems, which are unfeasible for comparable digital implementations. Especially for NP-complete optimization problems, which can not be solved in polynomial time by an algorithm<sup>†5</sup> alternatives would be desired. However, to the best knowledge of the author, there is no analog Ising machine that can guarantee to find the optimal solution. Potentially inaccurate models of the analog behavior, combined with unavoidable random manufacturing variations and mismatch of the analog properties, make providing any guarantee difficult. Furthermore, devices have inherent noise and might be undesirably influenced by crosstalk or power supply noise. However, noise might be an advantage for certain systems because it could prevent getting trapped in local extrema, which has a comparable effect to random number generators in digital Ising machines.

---

<sup>†4</sup>The connectivity is similar to the movement of a king in chess. Every node, which is placed on a two-dimensional grid, is connected to its two horizontal neighbors, two vertical neighbors and four diagonal neighbors.

<sup>†5</sup>see Section 2.1.1. The P versus NP problem is not considered here. In the case of  $P = NP$ , a polynomial time algorithm would exist.

---

Multiple physical implementations have been proposed for analog systems as well. They differ not only on a circuit level but also on the underlying device physics. Some approaches are based on CMOS technology, similar to the digital Ising machines. Others propose the use of novel devices instead.

**Oscillator-based** Ising machines using oscillators operate in the phase domain. The states of the Ising model are represented in the phase angle between oscillators. To implement the weights, the oscillators are coupled and interact with each other, manipulating their phases. Because this thesis discusses CMOS implementations of oscillator-based Ising machines, the operating principle is described in detail in the next chapter. It should be noted, there is considerable work using emerging devices such as VO<sub>2</sub> oscillators [62]–[64].

**Bifurcation** A bifurcation, based on a resistive coupled system, is presented by Afoawka et al. [65]. It is based on a capacitor to represent a spin and uses an active circuit to force its voltage in one of two different levels. Those spins are resistively coupled for their interaction. While their numerical results using highly idealized equations show promising solutions, experimental results are only provided using a breadboard with 5 spins. Although the authors suggest an integrated implementation, the real-world performance of their system remains questionable. The QuBRIM by Zhang et al. demonstrate an integrated bifurcation implementation [66].

**Other** Yu et al. propose a time-synchronous analog Ising machine implementation [67]. It is based on inverter chains, which are coupled to the neighboring horizontal and vertical spins. While this concept might appear similar to the oscillator-based variants, the chain of inverters does not act as an oscillator and does not exhibit sustained oscillations. At the start of the computation, the authors equalize the chain of inverters. When released without coupling, the inverter chain will either reach a voltage of 0 V or the supply voltage with equal probability, which represents the state of the spins. The authors reported an incredible computation time of just 13.2 ns and realized 1920 spins on a 0.49 mm<sup>2</sup> die. However, the implemented network has just connections to the two vertical and horizontal neighbors, forming a planar graph, which is not NP-hard to solve.

## 2.4.2. Optical Approaches

Another popular physical domain for the implementation of Ising machines are optical principles. The spins  $\sigma$  are often encoded by the phase of light. These spins can be either spatially distributed or time-domain multiplexed. Usually, an electrical feedback system is used, which enables seeking the ground states in multiple recurrent passes through the optical domain.

### Photonic Ising Machine

An implementation of a photonic Ising machine is proposed by Pierangeli et al. [68]. The principle is shown in Figure 2.7, as they have published in their paper. The computation is based on the spatial encoding of spins, which is obtained using an expanded laser beam. The spatial intensity of the beam is modulated using a programmable amplitude mask. The beam is modified by a liquid crystal reflective modulator with a high spatial resolution to shift the phases of these spins. An image sensor, placed in the far field, records the amplitude of the spins. A measurement and feedback method uses the detected image intensity to do multiple recurrent passes. The obtained image is compared with a target image and the liquid crystal

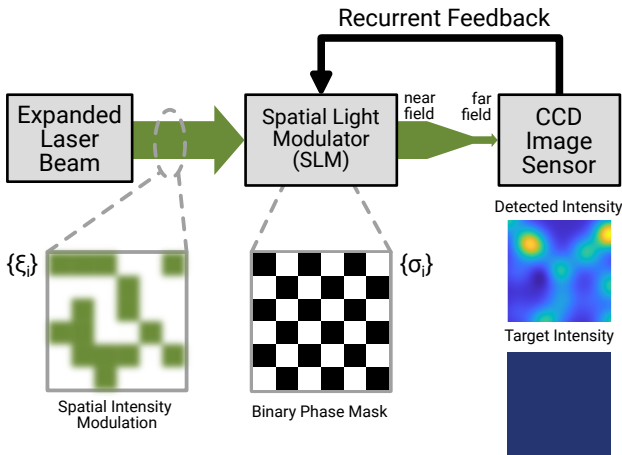


Figure 2.7.: Structure of a photonic Ising machine as proposed by Pierangeli et al. The spins are spatially decoded. Image based on [68, Fig. 1].

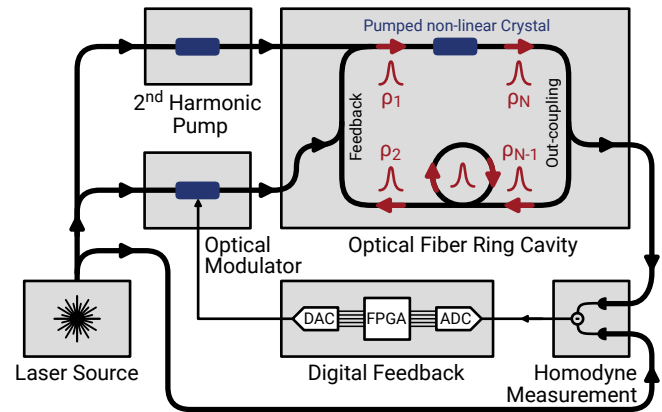


Figure 2.8.: Structure of a coherent Ising machine (CIM) as proposed by Ng et al. The optical pulses circulate inside the fiber. The feedback implementing the weights of the Ising machine is realized in the electrical domain. Image based on [69, Fig. 1b].

reflective modulator gradually updates to match both images as closely as possible. After several iterations, the configuration of the liquid crystal reflective modulator is the obtained solution for the optimization problem. This work demonstrates coupling with up to 40,000 spins.

Although it is not an Ising machine, the work of Wu et al. should also be mentioned [70]. They solve the NP-complete problem of deciding if a graph has a Hamiltonian cycle. The input graph is build in hardware using optical fibers and couplers. A light pulse travels through that optical hardware graph and explores all possible paths by splitting itself. The nodes in the optical network are constructed with a distinct delay, which allows reconstructing the taken path by measuring the propagation time through the network. The delays are chosen in a way, that the sum of the delays of all nodes is unique and cannot be reached by any other combination of delays. This ensures that if a pulse appears at the output of the network at this unique sum of all delays, the pulse must have visited all nodes exactly once. Hence, the existence of such a pulse at the output proves that the graph has a Hamilton cycle. If no pulse is observed at this unique delay, the graph does not have a Hamilton cycle. However, this work has been experimentally demonstrated with just 5 nodes and might face serious challenges when scaling up.

## Coherent Ising Machine

Multiple implementations of coherent Ising machines (CIMs) have been proposed [71]–[75]. The physical computing relies on the interaction of coherent light, where the spins are formed by degenerated optical parametric oscillators [69]. Figure 2.8 shows an exemplary principle of such a system, which is proposed by Ng et al. [69]. The spins in the form of pulses travel along the long (multiple kilometers) optical fiber cavity in a time-multiplexed scheme. Due to the operation far above the oscillation threshold, saturation effects cause bistable phase states. These phase states are used to represent the spins of the Ising model. The feedback

---

is provided by an electrical system. Each spin is measured and then the required feedback according to the weights of the Ising model is calculated. This is then optically modulated and provided as feedback into the cavity.

For example the work of Cen et al. [75] implements a system with 25,600 spins, which travel along a 20 km fiber. It is based on an optoelectronic parametric oscillators (OEPOs) topology proposed by Hao et al. [76]. The work by Honjo et al. demonstrates a 100,000 spin system which is capable of solving optimization problems in less than a millisecond [77].

### **2.4.3. Quantum Approaches**

Examples of quantum annealers that can solve problems in the Ising formulation are the systems developed by the company D-Wave. They have a quantum annealer called 'Advantage', which features 5,000 qubits and 35,000 couplers [78]. The next generation 'Advantage 2' is expected to have more than 7,000 qubits at over 60,000 couplers [79]. Different optimization problems have been solved using their quantum hardware [80], [81]. They report that not all qubits and couplers are functional even after calibration. Additionally, susceptibility to environmental noise is another challenge. In general, the cryogenic cooling needs immense energy and make those systems bulky. The whole system has the size of multiple racks, although the actual chip is small in comparison. As of now, the size and power make it suitable for data centers, but not for personal or edge devices. They use sophisticated embedding algorithms to enable solving various combinatorial optimization problems with the limited qubit connectivity.

---

## 3. Oscillators and their Ising Machines

---

This chapter discusses the fundamentals of oscillators, which are the central element for an OIM implementation. It starts from a description of oscillators and an overview of their electrical implementation. The concept of injection-locking, which influences the phase of an oscillator, is introduced, as it is the fundamental mechanism driving OIMs. Lastly, the concept of OIMs is explained, which use injection-locking in coupled oscillator networks to solve combinatorial optimization problems.

### 3.1. Oscillators

Oscillations are a naturally occurring phenomenon. A process repeats rhythmically, which can either sustain its oscillation infinitely or gradually decline over time. A simple example is a mechanical pendulum, which starts swinging after an initial deflection. When no energy is supplied to the pendulum, its amplitude will gradually decay and the whole oscillation will eventually stop. Other examples of natural oscillations are planetary motion in astronomy, tides along coasts, spiking of neurons in the human brain, and countless others.

Electrical oscillators are a fundamental component in almost all modern electronic systems. They can generate clocks for digital processors and provide the necessary frequencies for wired, wireless, and optical data transmission. There are other applications as well, such as the usage as a sensing element, whereby the quantity to be measured affects their frequency. As versatile as their applications are, their requirements for the design vary substantially.

#### 3.1.1. Barkhausen Criterion

In order to produce electrical oscillators suitable circuits are needed. An oscillator should provide a sustained oscillation at the desired frequency with a stable amplitude. This means, that the amplitude stays constant over time and neither decays nor increases. While some circuits, like amplifiers with feedback, might exhibit unintended oscillations, electric circuits do not necessarily oscillate. Only some circuits can oscillate under specific circumstances. Heinrich Barkhausen investigated the generation of electrical oscillators in his dissertation in 1907 and later work. This led to the Barkhausen criterion, which is a necessary but not sufficient criterion for oscillation in electrical circuits.

The setup for the Barkhausen criterion is shown in Figure 3.1. An oscillator is modeled by an amplifier block with the (frequency dependent) gain  $A(j\omega)$  and a feedback network  $\beta(j\omega)$ . The output of the amplifier  $A$  is connected with the input of the feedback network  $\beta$  and its output is connected with the input of the amplifier  $A$ , which forms a feedback loop with transfer function  $A(j\omega) \cdot \beta(j\omega)$ . The Barkhausen criterion is as shown in Equation 3.1a and 3.1b:



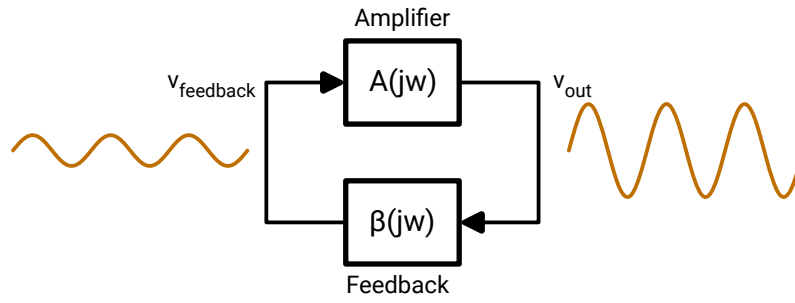


Figure 3.1.: A closed-loop feedback system as a general structure for an oscillator. It consists of an amplifier  $A(j\omega)$  and a feedback network  $\beta(j\omega)$ . Depending on the frequency response such systems can exhibit oscillation as indicated by the waveform.

$$|A(j\omega_0) \cdot \beta(j\omega_0)| = 1 \quad (3.1a) \quad \angle(A(j\omega_0) \cdot \beta(j\omega_0)) = 2n\pi \quad (3.1b)$$

$\omega_0$  denotes the oscillation frequency and  $n \in \mathbb{N}$  denotes a natural number. The loop gain, as shown in Equation 3.1a, must be exactly one. The Barkhausen criterion applies only to linear systems and is only a necessary, but not sufficient criterion. It can not be used for non-linear circuits, which is often caused by the oscillator's components, such as transistors and diodes. To the knowledge of the author, there is no equivalent theory and criterion for non-linear circuits. However, the non-linearity can be advantageous as it can limit the gain of a potential oscillator to 1. For example, the supply voltage of a ring oscillator automatically limits and stabilizes the oscillators amplitude.

### 3.1.2. Electrical Oscillators

Various implementations of oscillators exist, which have their specific advantages and disadvantages. There are too many circuits available so the following description focuses on the actual resonating element. The most relevant types, namely the LC oscillator, ring oscillator, relaxation oscillator, and crystal oscillator are discussed. In the case of OIMs, implementations using LC oscillators[4], relaxation oscillators[82], [83], and ring oscillators[84], [85] have been proposed. Although crystal oscillators are commonly used in electronic systems as well, no OIMs with crystal oscillators have been presented to the best of the authors knowledge. Besides the classical oscillator types, new emerging devices such as vanadium oxide can be used as oscillating element and have been demonstrated for Ising machines [62], [86]. Because these new technologies are not (yet) available in the standard CMOS foundry processes, they are not further considered in this work. Oscillators based on micro-electromechanical systems (MEMSs), which use kinetic and potential energy of micro mechanical structures, as well as other technologies are not considered here for the same reason. An extensive review of oscillators, which focuses on their properties for computing purpose is available in [87].

Nearly every electrical device has a digital processing core, which needs a clock for the synchronous circuits. Even DC-DC power converter applications are often switch-mode architectures due to their efficiency advantages. Wireless and wire-bound communication usually require a precise frequency as well. There are countless more applications of oscillators, which are too many to mention here. So, oscillators specialized for all of these applications is a well-researched topic.

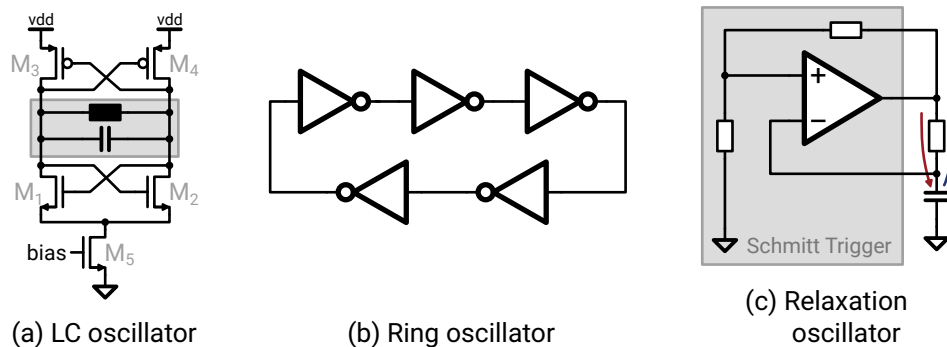


Figure 3.2.: Exemplary principle of common CMOS oscillator implementations

### LC Oscillators

LC oscillators are based on the resonance of an inductivity and capacitance. So, the circuits have at least one inductor and capacitor. The stored energy oscillates between the electrical field of the capacitor and the magnetic field of the inductor. Thus, the capacitance and inductance determine the frequency of the oscillator. Real circuits have losses in their conductors, magnetic material, and dielectrical material as well as wires connecting the components. Electro-magnetic waves might also be radiated by the currents along the physical wires, which drain energy from the oscillation. Since the losses would lead to a decaying amplitude, energy must be provided to compensate for that. To achieve a stable, constant amplitude some sort of gain control circuitry must supply exactly as much energy in each cycles as it is lost. Especially for tank circuits with a high Q factor, such oscillators can have a low power consumption as most of their stored energy is reused in the next cycle. For that, implementations such as differential LC-tank circuits and single-ended versions such as the Colpitts oscillator exist. An example of a typical implementation is provided in Figure 3.2a. The cross-coupled pairs M1/M2 and M3/M4 in combination with the amplitude regulating tail current bias of M5 compensate the losses of the LC tank. An analysis of this circuit can be found in [88].

### Ring Oscillators

Ring oscillators consist of delaying elements such as an inverter or, more generally speaking, delay cell. They are fed back in a ring structure, as illustrated in Figure 3.2b. The signal is inverted by each stage, which additionally adds a propagation delay. To achieve stable oscillation, a round trip phase shift of  $\pi + 2n\pi$  for any natural number  $n$  is required. Thus, single-ended ring oscillators need an odd number of inverting delay cells. Differential implementations can achieve a phase shift of  $\pi$  by swapping the differential signal pair, so that odd as well as even number of stages are possible. The delays of the delay cells determine the frequency, where loading of the (output) signals can further reduce the frequency. Thus, the electrical properties of the transistors and eventual capacitances determine the frequency. Consequently, temperature and the supply voltage significantly affect the frequency. The noise of the devices can cause considerable jitter of the oscillation cycles.

A 3-stage ring oscillator can be implemented with as low as 6 transistors, leading to a very compact area. More complex implementations, for example, current starved inverters or fully differential delay cells, are possible. By controlling the bias current of the delay cell, the frequencies can be varied. Although the internal stored energy is dissipated every cycle, they can be still low power due to the small (parasitic) capacitances. The Q-factor also lags behind other oscillator types.

---

## Relaxation Oscillators

Relaxation oscillators base their frequency on a well-defined time constant and a non-linear element. An exemplary implementation is shown in Figure 3.2c, which consists of a resistor, capacitor, and schmitt trigger. The capacitor is charged until an upper threshold is reached, then discharged until a lower threshold is reached. This constant charging and discharging (indicated by the red and blue arrows) leads to a sustained oscillation. The time constant of these charging and discharging determines then the oscillation frequency. Multiple similar implementations exist. For example, the charge of the capacitor could be reset upon reaching a threshold so that just the charging cycle determines the frequency. Instead of using a resistor for charging, a current source could be used as well. Frequencies of such oscillators are typically slow compared to LC and ring oscillators. The time constant of the RC network should be much larger than the delay of the comparator, which changes between charging and discharging. Otherwise the comparator's delay will have a considerable impact on the frequency.

## Crystal Oscillators & Other Types

Although the crystal oscillators can not be completely integrated in standard CMOS processes, they should be mentioned for the sake of completeness. Such oscillators rely on a piezoelectric element as resonator. The physical crystal deforms mechanically when an electric voltage is applied. An electrical implementation excites the mechanical resonance of the crystal, which yields a high Q-factor. Other electro-mechanical interactions, such as MEMS systems, can also be used as resonating element. Those are usually implemented on silicon chips, which however needs additional manufacturing steps compared to a standard CMOS process. Hence, they are not further considered.

## 3.2. Injection-Locking

Injection locking is a phenomenon where an oscillator is periodically perturbed by another periodic signal. When the frequencies are sufficiently close and the perturbation strong enough, the frequency of the disturbed oscillator will adapt to the frequency of the injected signal. The oscillator will have exactly the same frequency and exhibit a constant phase difference in the periodic steady state. In the case of OIMs, another oscillator acts as the periodic injection signal. Furthermore, the injection process is bidirectionally, meaning both involved oscillators mutually change the others phase and frequency. This kind of interaction is the basic computing operation.

### 3.2.1. An Everyday Example: Human Sleep-Wake Cycle

The concept of injection-locking is introduced with an illustrative example. The circadian rhythm, which is an internal oscillation of a biological organism, controls the sleep-wake cycle of the human body. This rhythm has a period of approximately 24 h. However, it is not exactly 24 h. In complete isolation without any information about the daytime, so neither daylight nor access to a clock, the sleep-wake cycle will desynchronize. Studies have shown that the average sleep-wake cycle extends to 25 h [89]. However, the actual sleep-wake cycle is individually different. One participant of the study even had a cycle of 33.4 h[89]. Since our circadian rhythm synchronizes with the 24 h day defined by sunlight, it is usually not noticed. However, when the day-rhythm suddenly changes, the synchronization between the circadian rhythm and the day-night cycle gets disturbed. This happens when traveling across multiple time zones by plane and can cause clearly noticeable

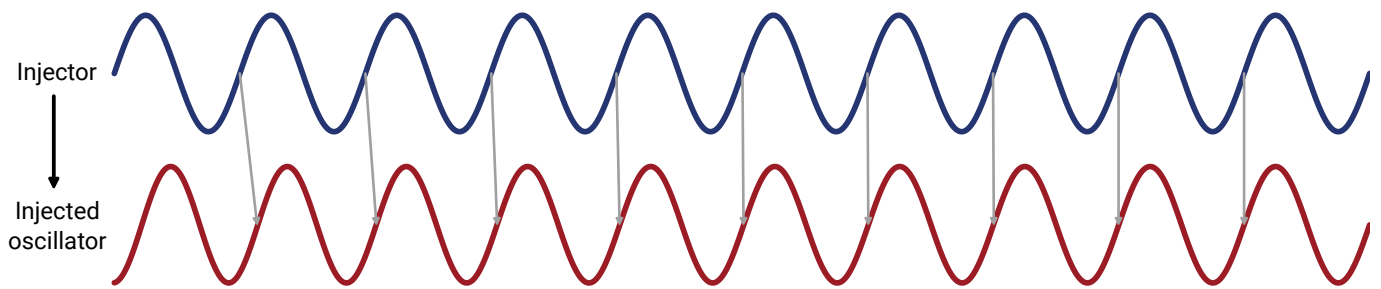


Figure 3.3.: Simplified example of injection locking of an oscillator. The injecting signal (blue) forces an alignment of the injected signal (red), which gradually adjusts its frequency and eventually achieves locking with the same frequency and a constant phase difference.

consequences on the well-being. After some time, the circadian rhythm resynchronizes with the local time again. When the circadian rhythm is interpreted as an oscillator<sup>†1</sup>, it synchronizes to another oscillator, the day-night cycle by the sun. This is a form of injection locking. The day-night time causes the circadian rhythm to synchronize, by adjusting its period to exactly match 24 h.

### 3.2.2. Injection-locking in Electrical Oscillators

Injection locking is frequently used for electrical oscillators. The fundamental principle is similar to other oscillator implementations. When the oscillator is lagging behind the injection signal, it is sped up. When leading, the oscillator is slowed-down by the injection. Consequently, when the frequencies are suitable to achieve locking, a periodic steady state is possible, where the injected oscillator adapts its frequency to match the injecting frequency precisely. The phase difference between both signals also converges to a constant offset.

A conceptual example is shown in Figure 3.3. The blue oscillator injects the red oscillator with a slightly lower frequency. Due to the injection, it catches up so that its frequency and phase match the injection after a few periods. Depending on the strength of the injection and the frequency difference, this process can be slower or faster. Injection locking in oscillators has been analytically described in literature e.g. [90]–[94]. Therefore, only relevant models for the application of the OIM design are discussed in Chapter 4.

### 3.2.3. Sub-harmonic Injection Locking

A special case of injection locking is sub-harmonic injection locking (SHIL). Instead of injection locking of oscillators with close frequencies  $f_{inj} \approx f_{osc}$ , locking is also possible when the frequencies are an approximate integer multiple  $f_{inj} \approx n \cdot f_{osc}$ , where  $n$  denotes a natural number greater than 1. The oscillator's frequency  $f_{osc}$  is a sub-harmonic of the perturbing signal  $f_{inj}$  and consequently aligns in one of  $n$  possible phase angles. This is illustrated in Figure 3.4, where 3.4a demonstrates twice the oscillation frequency for injection. The two possible phase angles are clearly visible, where the positive zero-crossing of the injected waveform aligns with one of two possible zero-crossings of the injection waveform. A similar example with three times the

<sup>†1</sup>There are strong indications, that multiple circadian oscillators exist within the body. Each controls different mechanisms like the sleep-wake cycle or the body temperature. Since they are coupled together and synchronized to the 24 h day, this is usually not noticed. The study pointed out that the sleep-wake cycle and the rectal body temperature can desynchronize and exhibit different periods. The interested reader should refer to the field of chronobiology for more information.

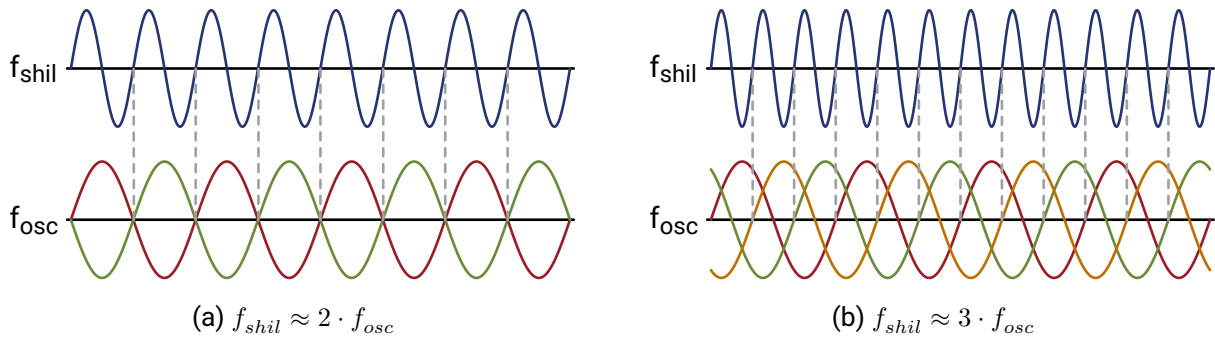


Figure 3.4.: Simplified principle of sub-harmonic injection locking (SHIL). As illustrates, the oscillators align when injected with a integer frequency ratio. However, this leads to multiple possible phase angles.

frequency is shown in Figure 3.4b, where the injected waveform has three possible angles for injection. In that example, the positive rising edge of the oscillator signal aligns with one of the rising edges of the injecting waveforms.

While SHIL works for any frequency ratio  $n$ , achieving large ratios might get difficult in practice. Depending on the oscillator, the frequency ratio must closely match an integer number so that a locking is possible. Also, higher ratios might require stronger injection to achieve locking at all. The SHIL plays a major role in OIM computing. The modeling, circuit design, and experimental evaluation of the circuits are discussed throughout this thesis. Theoretical models and descriptions are available in literature where Wang provides an illustrative explanation using metronomes [95].

### 3.2.4. Super-harmonic Injection Locking

For the sake of completeness, the super-harmonic injection locking is mentioned as well. It is the opposite of the sub-harmonic injection locking, where the injection clock is an integer fraction of the oscillator clock. So, a frequency ratio of  $f_{inj} \approx \frac{1}{n} \cdot f_{osc}$  respectively  $n \cdot f_{inj} \approx f_{osc}$  is required.

### 3.2.5. Applications of Injection Locking

Injection locking can be used in frequency generation applications. Common examples are injection-locked phase-locked loops (PLLs) [96], [97]. Also injection locked dividers [98] and quadrature oscillators have been proposed [99]–[102]. The interesting application of time synchronization based on the injection locking of two spatially distributed oscillators is explained in more detail in the following. Since it uses a form of oscillator-to-oscillator locking, it has considerable similarities to OIMs. The phase dynamics, which are affected by the propagation delay of the spatially separated oscillators, share strong similarities with the later discussed routable connections of oscillators spatially distributed across the developed OIM chip.

#### Example: Time Synchronization

Synchronizing time over a spatial distance is an interesting application of injection locking. The goal is to have a spatially distributed time stamp at different positions. This can be necessary to share a common time among

---

multiple computing or sensor nodes precisely. When transmitting a signal over any distance, it will experience an inevitable delay. This applies, for example, to electrical signals on a wire, electromagnetic waves, and fiber optic signals.

The work conducted by Hoyer et al. [103]–[105] uses injection locking between oscillators instead. Each local node consists of an oscillator, which transmits its clock and receives the clocks from other nodes. The oscillator node itself is implemented as PLL with an output frequency of 24 GHz. A phase detector measures the phase differences between its locally generated clock and the receiving clock to adjust the local oscillator. While such systems can exhibit instability and only operate as intended within certain loop configurations, the authors demonstrated successful synchronization of the oscillators for a distance of 500 m (2.5  $\mu$ s delay).

### 3.2.6. Coupling

In the scope of OIMs and more generally oscillator-based computing, usually the term ‘coupling’ is used. It could be considered a special case of injection locking. The term ‘coupling’ usually refers to a phase interaction and phase change between two or more similar oscillators. Often this is in the context of computing with oscillators. The phase encodes hereby the state information similar to the voltage level of logic circuits. Locking of the oscillators at the same frequency is usually assumed without explicitly mentioning it. Without that assumption, the definition of the phase as information carrier gets problematic. During the dynamic settling process, the frequency can temporarily change. Also, the resulting frequency of the coupled oscillators might considerably shift from the free-running frequency of the oscillators and might even vary with the coupling strength.

Traditional injection locking usually uses a steady, periodic signal for injection into an autonomous oscillator. This is different to the coupling application, where the injecting signal is an oscillator encoding the phase information and thus is not strictly periodic. It will exhibit phase changes and might even (temporarily) change its frequency. As special case for OIM, coupling usually refers to the bidirectional interaction of oscillators. So, one oscillator influences the other oscillator phase, which in turn influences the phase of the other oscillator. Consequently, both involved oscillators usually experience a phase change.

As the coupling between the oscillators is the central element of computing in OIMs, it is extensively discussed in this thesis. Multiple mathematical descriptions, simulations and experiments showcasing the behavior are provided.

## 3.3. Oscillator-based Ising Machines

The idea of using oscillators for computation was already patented by von Neumann in 1957 [106]. It was intended to overcome the speed limitation of the available devices at that time by using the phase information of oscillators as data representation [107]. Another oscillatory computing element, the Parametron, was invented by Goto in 1954 and used to build early digital computers [108]. It is a resonant circuit with a non-linear reactive element representing binary numbers in its phase [108]. Although transistor-based digital computers prevailed over time, recent research looks at oscillatory computing again. New methods are required to enhance the efficiency of traditional computing with digital circuits. Hence, oscillator-based computing gets attention again [109]. It was theoretically analyzed and experimentally proven that oscillator with suitable interaction (coupling) are able to solve combinatorial optimization problems, which sparked the OIMs [4].

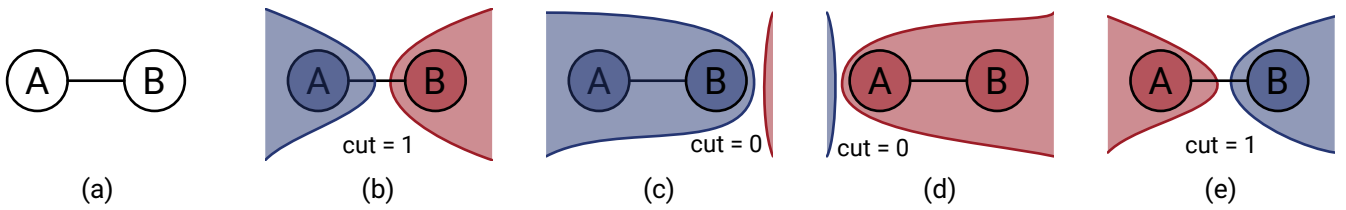


Figure 3.5.: The two oscillators maximum cut problem (a) has four different combinations (b)-(d). Two of those, where the nodes are in separate sets, lead to the maximum cut of 1.

### 3.3.1. Phenomenological Introduction

A simple maximum cut problem with two nodes  $A$  and  $B$  and an edge with weight 1 is shown in Figure 3.5. All four possible combinations are enumerated and the corresponding cut is provided. Exactly two of them lead to the maximum cut of 1, where both nodes are in different sets. This example is equivalent to a circuit realization with two coupled oscillators trying to achieve a phase difference of  $180^\circ$ . So, the discrete sets of the maximum cut problem can be represented in the oscillators phase when their continuous phase is divided into two discrete groups.

The same example can be applied to the Ising model because the maximum cut problem can be transformed into the Ising model by inverting the sign of the edge weights. In this example the Hamiltonian reduces to  $H(\sigma) = -(-1 \cdot \sigma_A \sigma_B) = \sigma_A \sigma_B$ , which should be minimized. When  $\sigma_A$  and  $\sigma_B$  are in different discrete states  $\sigma_i \in \{+1, -1\}$ , the Hamiltonian is minimized and the optimal solution is found. So, a weight  $J_{ij}$  is implemented by coupling the oscillators  $i$  and  $j$ . Positive weights  $J_{ij}$  are implemented by coupling at  $180^\circ$  phase shift ('anti-phase') and negative weights  $J_{ij}$  by coupling at  $0^\circ$  ('in-phase'). To account for different weight values, the strength of the coupling can be adjusted accordingly relative to all other weights.

### 3.3.2. Continuous and Discrete Phases

In the before-mentioned example, the coupler can establish the desired phase difference between both oscillators so that one oscillator will align at a phase angle of  $0^\circ$  and the other at  $180^\circ$ . However, the oscillator phases are generally continuous and will likely not show two discrete phases at  $0^\circ$  and  $180^\circ$ . This is illustrated in the example in Figure 3.6, where a ring of three nodes is used. Here, it is impossible to establish a phase difference of  $180^\circ$  ( $\pi$ ) for all three edges as the maximum cut for this topology is just 3. As shown by the simulation data, a phase difference of  $120^\circ$  ( $\frac{2\pi}{3}$ ) will appear between the identical oscillators. Consequently, the oscillators cannot be clearly assigned into the two discrete phase groups apart by  $180^\circ$  to determine their discrete state  $\sigma$ . To establish a discretization of the continuous phases, a SHIL with twice the frequency is applied. This SHIL will force phase alignment in two discrete states and allow to clearly determine their discrete state based on the phase.

### 3.3.3. Theoretical Background on Oscillator-based Ising Machines

An individual oscillator just receives the phase information of the oscillators linked with non-zero weights. It reacts by changing its own phase and influences the phase of the other coupled oscillators. Such an oscillator itself is not aware of the optimization problem or any other oscillators in the system that are not directly connected. As other works and this work experimentally show, these networks of coupled oscillator strive towards the ground state of the Ising model. Thus, the natural behavior of such coupled oscillator networks

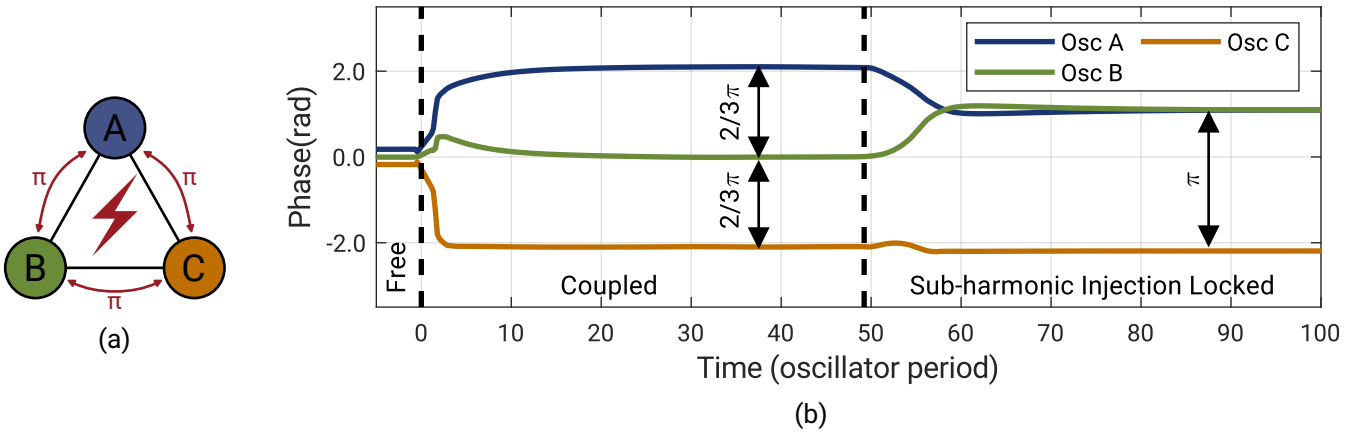


Figure 3.6.: Computing of an exemplary maximum cut problem consisting of a ring of 3 nodes. (a) The desired phase difference of  $\pi$  between the nodes representing the oscillators is not possible. (b) Simulation of the waveform, where a SHIL is applied to discretize the oscillators state.

tends to minimize the Ising Hamiltonian. However, there is no guarantee, that the OIM actually settles towards the ground state. The ground state is usually not reached for sufficient complex problems.

A theoretical analysis has been proposed by Wang and Roychowdhury. They use the generalized Adler's equation to describe the coupled oscillators analytically [94]. As the name suggests, this is a generalized version of Adler's equation, which describes the locking process specifically for LC-based oscillators [90]. By using the perturbation projection vector (PPV) to model the oscillators response, it applies to virtually any oscillator. The model of the injection locking is as follows [94].

$$\frac{d\Delta\phi}{dt} = -(f_1 - f_0) + f_o \cdot g(\Delta\phi(t)) \quad (3.2)$$

Thereby  $f_0$  is the free-running frequency of the injected oscillator and  $f_1$  the frequency of the injection signal.  $\Delta\phi$  is the phase difference between both signals and  $g$  is the periodic function of the oscillator's PPV. Based on this equation Wang and Roychowdhury have calculated the global Lyapunov function for the coupled oscillator network, which resembles the system's ground state. Under applied SHIL locking, which achieves two discrete phase states, it has the same form as the Ising model. Hence, the coupled oscillators theoretically solve problems in the Ising formulation [5], [110], [111]. They pointed out that this is only the case for the coupled oscillators under SHIL lock. It does not apply without SHIL due to the continuous phases. As they demonstrated, just mathematically rounding the continuous phases to the nearest discrete state achieves a poor solution.

While their analysis initially assumes that all oscillators operate at the same frequency, they considered frequency variations as well. It adds another term to the Lyapunov function, which can be held small by an oscillator frequency calibration so that it does not affect the results much. Therefore, the OIMs work in theory as Ising machine. However, just having the ground state as the Ising model is not equivalent to actually finding the ground state, as pointed out by Wang and Roychowdhury. The coupled oscillator could get stuck in a local minimum instead.

For a practical implementation, the non-ideal behavior of actual circuits and unavoidable random variations of manufactured elements is a key challenge. Not only does the frequency of individual oscillators vary, but their PPV will be affected as well. While a frequency calibration is straightforward, the PPV cannot be directly measured for calibration. The coupling elements, which enable the locking of the oscillators will



---

have additional variations, introducing errors from the ideal formulation. As it can be expected, experimental results show that physical OIMs usually do not find the ground state and have varying gaps depending on their design. Hence, a major challenge is to build physical OIMs that strive towards the ground state and do not get stuck in a local minimum. Since it will likely not be possible in practice to avoid any local minimum, it should be at least near the global ground state. At the same time, area and random mismatch is a key challenge in making a large, scalable coupled oscillator network integrated on a silicon chip.

### 3.3.4. Overall Procedure

The overall procedure for the computing optimization problems using OIM can be divided in five steps:

#### 1. Ising Problem Formulation

The application problem must be provided in the Ising model. It can be either directly formulated in the Ising format or an already existing problem can be transformed into the Ising form.

#### 2. Problem Embedding

The application problem must be embedded into the hardware network of the OIM system. Every discrete variable of the Ising model is assigned to a physical oscillator. For each weight, a corresponding physical coupler between these oscillators must be available, which gets configured accordingly.

#### 3. Oscillatory Computing

The oscillators are enabled after configuration and mutually interact to change their phase. After some time to naturally seek towards their ground state, the SHIL is applied to force the oscillators from continuous into discrete phase angles.

#### 4. Phase Readout

The phase angles, which encode the states of the discrete variables, need to be measured. The phase angle of an oscillator, belonging in one of the two phase groups separated by  $\pi$ , determines the state of the corresponding variable. The discrete state of each is then determined by the phase group of the oscillator.

#### 5. Solution

By using the assignment between physical oscillators and discrete variables of the embedding process, the discrete states for all variables are obtained as a solution to the optimization problem.

Because the outcome of a single computation is non-deterministic as analyzed in Chapter 7, the obtained solution will vary when repeating the computation. Therefore, steps 3 and 4 can be repeated multiple times to select the best solution out of the computed solutions. The focus of this thesis is on the CMOS implementation for the oscillatory computing. Hence a suitable problem formulation following the Ising model is assumed. The formulation of optimization problems in the Ising model or transforming existing problems into the Ising model is outside the scope of this thesis. Such tasks are highly specific to the application, which needs to be first modeled as an optimization problem. Additionally, the problem embedding in step 2 is only briefly touched by two supervised theses [167], [170] and a publication [156]. Although assigning nodes to physical oscillators might initially appear to be straightforward, it can be very difficult. Essentially, this minor embedding of the optimization problem in the (fixed) hardware graph of the OIM can create a new optimization problem. The difficulty depends on the topology of the hardware graph and is an optimization problem in itself. Depending on the topology of the hardware graph, this minor embedding can actually be NP-hard as proven for the case of broken Chimera and Pegasus graphs used by the D-Wave quantum annealers [112]. At this point, it should be noted, that such an embedding is not only specifically needed for OIMs. Other hardware implementations

---

of Ising machines, which use dedicated physical elements to represent the discrete spins of the Ising model, need a similar embedding. The embedding is trivial in the case of all-to-all connectivity.

### 3.4. Conclusion of Oscillators and OIMs

A short introduction into LC, ring, and relaxation oscillators commonly used in CMOS technology has been provided. The main operating principle of OIMs is the injection locking, where the phase of an oscillator is influenced by the other coupled oscillators to achieve locking. This means that small frequency differences are overcome and a constant phase difference between both clocks is established. OIMs are a network of mutually coupled oscillators that naturally strive towards a ground state by changing their phases. Each spin  $\sigma_i$  is represented by a physical oscillator and the interacting coefficients  $J_{ij}$  as a coupling circuit to allow interaction between the oscillators. When a SHIL is applied, which forces the oscillators to align in two de-facto discrete phases that are apart by  $\pi$ , they are essentially solving the Ising model. Since the coupling and locking process just takes several oscillation cycles, the computation can be very fast. Although it has been theoretically proven that the ground state of the coupled oscillator system matches the Ising formulation, this does not provide any guarantee that it is actually reached. Besides non-idealities and mismatch of real circuits, the system will likely settle in a local minimum. Another pre-requisite step for this computation setup is the minor embedding, where the spins are assigned to the physically available oscillators in the network, which can be a very difficult problem.

#### Oscillators and OIMs

- Coupled oscillator network with SHIL are suitable to solve optimization problems in the Ising model by a **phase-based computing** principle.
- However, there is **no guarantee** that they will actually find the **global optimum**. They will likely get stuck in a local minimum.
- Each spin  $\sigma_i$  of the Ising model is represented by a physical oscillator. Each coefficient  $J_{ij}$  is represented by a physical interaction between the corresponding oscillators with adjustable strength.
- The required **embedding** creates, in principle, **another optimization problem**. The OIM network should make the embedding as simple as possible to avoid creating a severe bottleneck.

## 4. Modeling and Simulation

A fundamental task in the design of OIMs is their modeling and simulation. It not only verifies the correct operation of the designed system, but it also provides a preliminary estimation of the expected performance to compare different designs. Solving NP-hard problems time- and energy efficiently is already a challenging task in itself. Designing an analog system to solve such problems in its underlying dynamics might be even more difficult. The device and circuit behavior is often complex and not always available as analytical equations. Modern semiconductor models use numerous parameters to fit their behavior to the measured device characteristics. Simplifications of models or circuits can be introduced, however they might significantly change the predicted behavior from reality. Hence, numerical simulations can help in designing and improving the system. Prototyping and lab testing should be kept to a minimum due to the long cycle time and high cost. Within the scope of this thesis, two ASICs have been manufactured as discussed in Chapter 5, but many more circuits have been simulated.

Although the computing power of CPUs is steadily increasing and has shown remarkable improvements over the last decades, computing power is still a valuable and limited resource. At some point, extensive simulations with very high accuracy are not practical due to long runtimes or might not even be possible due to limited memory. As the available computing power improves over time, the complexity of simulation models tends to increase as well. Consequently, there is a trade-off between simulation runtime and accuracy. Hence, different approaches for modeling and simulation are discussed in this chapter. An overview of the different abstraction levels, which trade simulation precision for runtime, is provided in Figure 4.1. The Kuramoto model, which is a general description of the synchronization of oscillators, is discussed in Section 4.2. Since it is not a specific model for electrical oscillators or even OIMs, more circuit-specific methods are discussed in Section 4.2. It improves the accuracy at the cost of increased runtime by including a more precise model of the circuits. Analytic models are also discussed, but a numerical extraction-based model is proposed, which treats the oscillator and coupler circuits of the OIM as gray-box. The more precise but by far most time-consuming approach is discussed in Section 4.3. It uses the transistor-level circuits with the simulation models provided by the foundry for the devices.

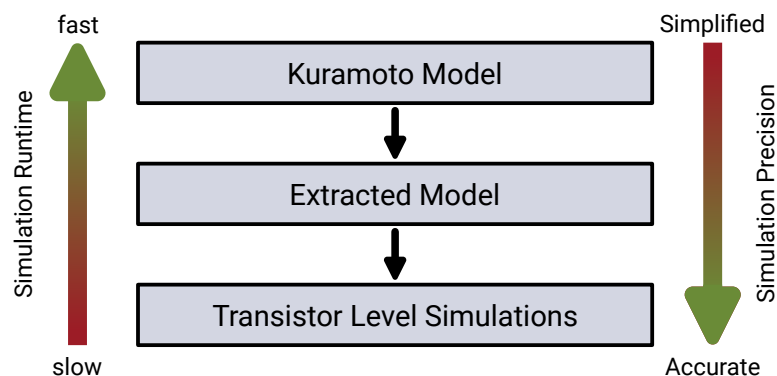


Figure 4.1.: Abstraction levels of OIM simulations. It is a trade-off between simulation accuracy and simulation runtime.

## 4.1. Kuramoto Model

Several natural phenomena are based on an interaction between oscillating elements, which sparked the interest of researchers. A mathematical description was needed to study the characteristics of such systems further. Winfree worked on the behavior of biological coupled oscillators [113]. Later, inspired by work on chemical instabilities, the Japanese physicist Yoshiki Kuramoto proposed a mathematical model, which is named after him as ‘Kuramoto model’ [114], [115]. It is a simple model using macroscopic self-sustained oscillators. Those are weakly coupled with each other so that they react on their mutual phase differences. An individual oscillator is described by Kuramoto as follows:

$$\frac{d\phi_i}{dt} = \omega_{0,i} + \frac{v}{n} \sum_{j,j \neq i}^n \sin(\phi_j - \phi_i) \quad (4.1)$$

The phase  $\phi_i$  of the  $i^{\text{th}}$  oscillator changes with its natural frequency  $\omega_{0,i}$  and the sum of the coupling interaction.  $v \in \mathbb{R}$  denotes a constant for the strength of the interaction and  $n \in \mathbb{N}$  is the number of oscillators of that system. This scales the phase change caused by the coupling, which is determined by the sinus of the phase difference. In the original Kuramoto model, every oscillator is coupled to any other oscillator with an identical strength. For OIMs this is not the case because the couplings between oscillators are established according to the optimization problem to solve. Additionally, all oscillators are sub-harmonic injection-locked for phase discretization. To adapt the Kuramoto model for the purpose of the OIM simulation, it is written as follows:

$$\frac{d\phi_i}{dt} = \omega_{0,i} + K_{cpl} \sum_{j,j \neq i}^n J_{ij} \cdot f_{cpl}(\phi_j - \phi_i) + K_{shil} \cdot A_{shil}(t) \cdot f_{shil}(2\pi\omega_{shil}t + \phi_{0,shil} - \phi_i) \quad (4.2)$$

In comparison to the original Kuramoto model, the coupling between oscillators is changed from a global factor  $\frac{v}{n}$  to the individual weights of the Ising model  $J_{ij}$ . These are then globally scaled with a factor  $K_{cpl} \in \mathbb{R}$ . The sinusoidal function on the phase difference is replaced with a more general function  $f_{cpl}$ . The additional term  $K_{shil} \cdot A_{shil}(t) \cdot f_{shil}(2\pi\omega_{shil}t + \phi_{0,shil} - \phi_i)$  models the sub-harmonic injection locking process.  $A_{shil}$  describes the strength of the sub-harmonic injection process and enables/disables the locking, which is scaled with  $K_{shil} \in \mathbb{R}$ . Initially, the oscillators couple without SHIL so that  $A_{shil}(0) = 0$  and afterwards, when the SHIL should be applied,  $A_{shil}(t_{shil}) > 0$ .  $\omega_{shil}$  denotes the SHIL frequency, which is approximately twice the frequencies of the oscillators  $\omega_{shil} \approx 2 \cdot \omega_0$  and  $\phi_{0,shil}$  the constant starting phase of the SHIL signal. It should be noted that the SHIL is an independent clock signal, which is not affected by any coupling. The function  $f_{shil}$  denotes the impact of phase differences analogous to  $f_{cpl}$ .

Since Equation 4.2 describes the behavior of an individual oscillator, the modified Kuramoto model forms a system of coupled ordinary differential equations. The main contribution of the  $\frac{d\phi_i}{dt}$  of the oscillators comes from their natural frequency. In a real physical system, the individual frequencies of the oscillators will vary due to manufacturing mismatch as discussed in Section 5.2.6. Hence, the model supports setting the natural frequency individually for each oscillator. An additional boundary condition needs to be set for the initial phase of each oscillator. An example simulation is shown in Figure 4.2, which is obtained using Matlab’s ode45 solver for differential equations. The initial phases of the oscillator were set randomly. Four SHIL intervals are applied and indicated by the gray background. Within every interval, the SHIL is ramped-up, held at the target strength, and then ramped down. The discrete states can be clearly identified during the SHIL intervals. When the oscillators are just coupled, the phase angles are continuous. So, this example emphasizes the importance of the SHIL.

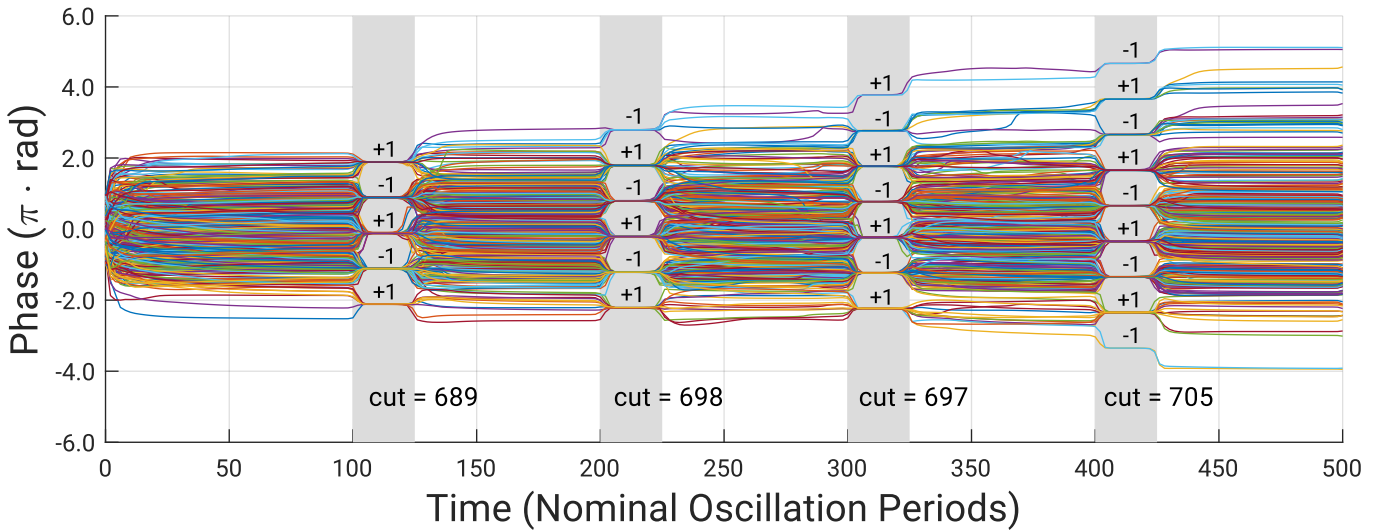


Figure 4.2.: Example of using the Kuramoto model for an OIM simulation solving a 400 node maximum cut problem. Four SHIL intervals (marked gray) are applied, which clearly show the transition from continuous phases to quasi-discrete phases. The small numbers show the corresponding spins of the phase. The cut changes between cycles and leads to an improvement.  $K_{cpl} = K_{sync} = 2.5$  and a 1% standard deviation of the oscillator frequency is used.

Simulations with the Kuramoto model show that just somehow coupling any oscillator is not enough. The choice of the global coupling strength  $K_{cpl}$  and the response to phase differences  $f_{cpl}$  shape the coupling behavior and outcome of OIM computations. Furthermore, the Kuramoto model allows investigating the impact of noise, frequency mismatch, coupling strength, coupling behavior, and SHIL. For example, Wang investigated several impacting factors on the OIM computing in his PhD thesis [5]. Wang mentioned that the coupling function  $f_{cpl}$  should be explored further and proposed a  $\tanh(\sin())$  function. The impact of  $f_{cpl}$  is emphasized in Figure 4.3, which compares five different  $f_{cpl}$  with all other parameters identical. The  $\sin^3(x)$  performs poorly compared to the other alternatives, which exhibit better performance. Especially the proposed  $\tanh(\sin())$  by Wang shows better performance than the original sinus of the Kuramoto model.

## 4.2. Abstract Model

The Kuramoto model is phenomenological and does not consider the actual oscillator behavior with great accuracy. As it was mentioned, it is used to describe the synchronization across various (scientific) domains. Consequently, it does not provide much insight for the circuit design of suitable oscillators and couplers. While the impact of  $f_{cpl}$  was already exemplarily shown, this function only considers the phase difference between the oscillators. However, the sensitivity of an oscillator to an injected current as input from the coupling is not independent of its actual phase. Within one oscillation cycle, the sensitivity to such coupling inputs varies significantly. The oscillator's sensitivity and the characteristic of the coupling element together form the complex behavior. This determines the computing and the obtained solution. Hence, the circuits need to be designed for the OIM purpose.

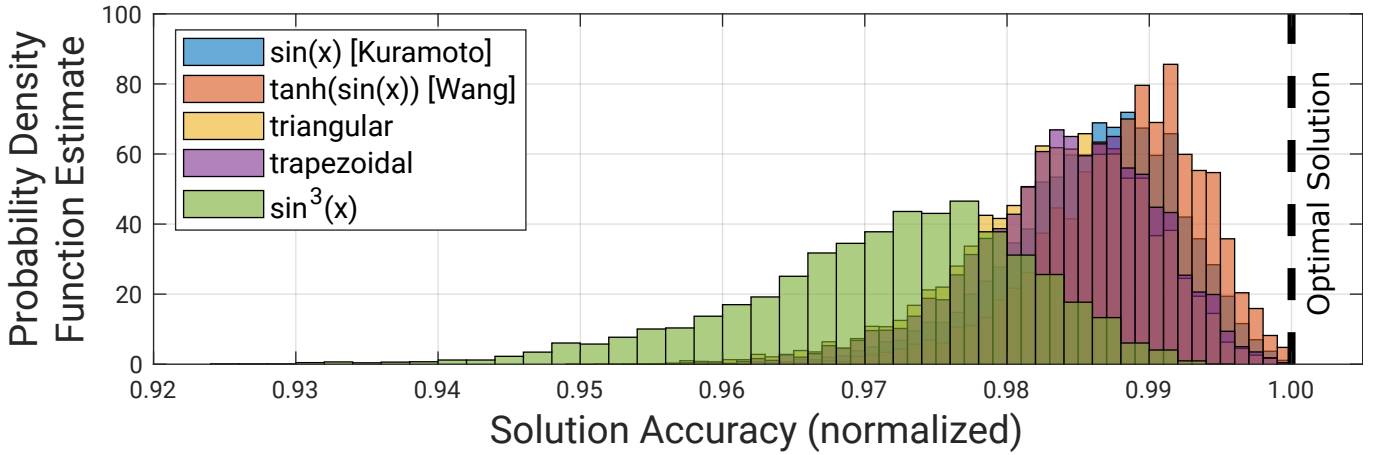


Figure 4.3.: Exemplary illustrated impact of the function  $f_{cpl}$ . A set of 100 problems with 400 nodes each were solved using the Kuramoto model. Each problem was repeatedly solved 100 times and the obtained solution was compared with the optimal solution. To allow a fair comparison,  $f_{cpl}$  was scaled so that  $\int_0^{2\pi} |f_{cpl}(x)| dx = \int_0^{2\pi} |\sin(x)| dx = 4$  applies.

#### 4.2.1. Oscillator Models

From a circuit point of view, an oscillator exhibits a periodic change of its output voltage. The waveform is usually not strictly sinusoidal or rectangular and could theoretically have any periodic shape. An analog circuit simulator calculates this waveform based on currents flowing through the transistors and other components of the oscillator. When coupled, a current is usually injected into the oscillator, which temporarily increases or decreases the frequency, leading to a phase change. Modeling the oscillator's response on such current injections is of interest for the OIM simulation. Usually, existing oscillator models are used for analysis of phase noise and jitter, which is crucial in frequency generation tasks such as PLLs. Analytic and numerical approaches have been developed to aid the design of such circuits [92], [116]. The models often use the valid assumption that the considered noise currents or voltages are very small compared to the amplitude of the oscillator signals. However, this does not hold true in the case of OIMs. They can be strongly coupled, which can easily change the phase by  $\pi$  within a few cycles. Hence, such injected currents cannot be considered small anymore.

#### Adler's equation

Analytical studies of oscillators and investigation of their behavior have been conducted in the past. Adler studied an LCR oscillator and derived conditions for the injection locking [90]. His differential equation describes the injection locking process.

$$\frac{d\alpha}{dt} = -\frac{V_{inj}}{V_0} \frac{\omega_0}{2Q} + \Delta\omega_0 \quad (4.3)$$

$\alpha$  denotes the phase difference to an undisturbed, free-running oscillator.  $V_{inj}$  is the amplitude of the injected voltage and  $V_0$  the amplitude of the oscillator.  $Q$  is the quality factor and  $\Delta\omega_0$  the angular frequency difference between the free-running frequency  $\omega_0$  and the injecting frequency. This formula, however, assumes a sinusoidal injection with constant amplitude. Generalizations have been proposed, but they still require a periodic injection [94] making them not suitable for bidirectional coupling in OIMs.

## Impulse Sensitivity Function (ISF)

While some oscillators such as LC or simple ring oscillators can be analyzed analytically, this is not feasible for circuits relying on advanced metal–oxide–semiconductor field effect transistors (MOSFETs) with their complex simulation models. The so-called impulse sensitivity function (ISF) was proposed by Hajimiri and Lee in 1998 to predict the phase noise in electrical oscillators [117]. When a perturbing pulse is injected into an oscillator, it changes its phase and amplitude. The amplitude will decay over time due to the oscillator's gain-controlling principles and is not of interest. The phase change will persist and depend on the oscillator's phase angle and the strength of the pulse. Hajimiri and Lee describe the unit impulse response for the excess phase  $h_{\Phi}(t, \tau)$  as follows.

$$h_{\Phi}(t, \tau) = \frac{\Gamma(\omega_0 \tau)}{q_{max}} \cdot u(t - \tau) \quad (4.4)$$

$q_{max}$  is the maximum charge of the oscillators capacitance<sup>†1</sup> and  $u(t)$  the unit step.  $\Gamma()$  is the  $2\pi$ -periodic impulse sensitivity function (ISF), describing the excess phase from a unit impulse at time point  $\tau$ . They suggest three calculation methods for the ISF.

**Direct Measurement** During a transient simulation, a short pulse is injected into the oscillator. After a few cycles, the excess phase change compared to an undisturbed oscillator is measured. This process is swept across a range of injection time points distributed across an oscillation period. However, Pepe et al. pointed out that the transient simulation method requires careful accuracy settings and might lead to false results for too large or small pulses [118].

**Closed Form** Hajimiri and Lee proposed a method to calculate the ISF based on the voltage waveforms of all oscillator nodes as  $\Sigma_i(x) = \frac{f_i'}{\sum_{j=1}^n f_j'^2}$ . If the waveform can not be calculated analytically, it can be obtained using a circuit simulator. However, numerical inaccuracies of the simulated waveform might be amplified by the derivation for the ISF calculation.

**Approximate** An approximate method of the closed form below was introduced as well. It is based on just the first derivative of the oscillator as  $\Sigma_i(x) = \frac{f_i'}{f_{max}'^2}$ . However, their work shows a substantial error compared to their other two methods.

**PSS-based** A periodic steady state (PSS)-based simulation was proposed by Pepe et al. [118]. They calculate the ISF from the periodic transfer function (PXF) simulation of an advanced circuit simulator such as Cadence Spectre. The PXF-based approach is less susceptible to numerical inaccuracies of the simulation and can be simulated quickly.

**Example** In this work, the *Direct Measurement* method is applied. The computing time of a couple of minutes is fast enough, while injections with different pulse widths and heights allow to roughly estimate the limits, where the oscillator sensitivity still behaves linearly. Because modern transistor models are complex and may include internal nodes hidden to the user, the *Closed Form* is not an option. Furthermore, the ISF is normalized

<sup>†1</sup>This can be the charge displacement of the actual capacitor in an LC oscillator or the node capacitance in a ring oscillator. The  $q_{max}$  term makes the  $\Gamma$  a dimensionless function.

to the injected charge. Instead of being dimensionless as originally proposed, it can be seen as phase change per injected charge. This is useful for ring oscillators, which do not have a distinct capacitance, respectively maximum charge, for normalization.

An example using a simple LC-oscillator with lossy inductors is provided in Figure 4.4a, where the current injection source is indicated in gray. The ISF in Figure 4.4b approximately follows a cosine behavior, where a phase angle of  $0^\circ$  marks the positive zero-crossing of the differential output. The transient simulation in Figure 4.4c compares the free-running oscillator with an injected oscillator at  $0^\circ$  and  $90^\circ$ . The case of a  $0^\circ$  pulse shows a strong phase change combined with virtually no amplitude change. However, at  $90^\circ$ , a strong amplitude perturbation is visible, which decays over the next cycles, but only a very small phase change remains.

An equivalent example for a 7-stage ring oscillator is shown in Figure 4.5. Compared to the LC-oscillator, the ISF exhibits a high sensitivity at the low-high and high-low transitions, but it is insensitive in between. This is illustrated by the transient waveform in part 4.5c. Similar to the LC-oscillator, the  $0^\circ$  injection causes a phase shift, while the  $90^\circ$  injection causes a temporary amplitude perturbation. Due to the lower quality factor of the ring oscillator, this amplitude modulation decays within a fraction of an oscillation period.

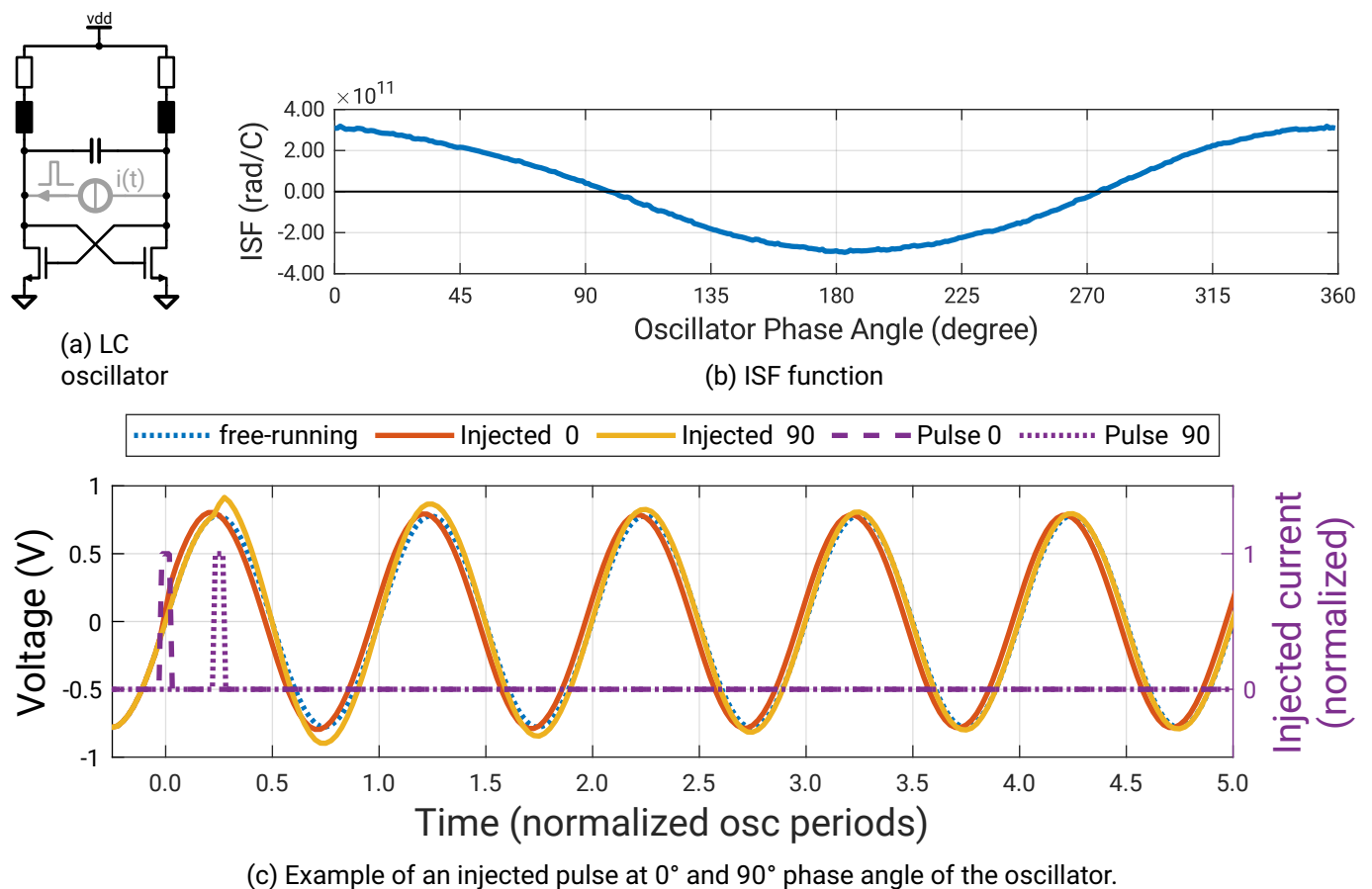


Figure 4.4.: Example of the ISF for a simple LC-oscillator with lossy inductors. (a) Schematic of the differential implementation with a charge respectively current injection source in gray. (b) Numerical extracted ISF function. (c) Transient pulse injection to show the amplitude and phase perturbation by a single pulse at  $0^\circ$  and  $90^\circ$ .



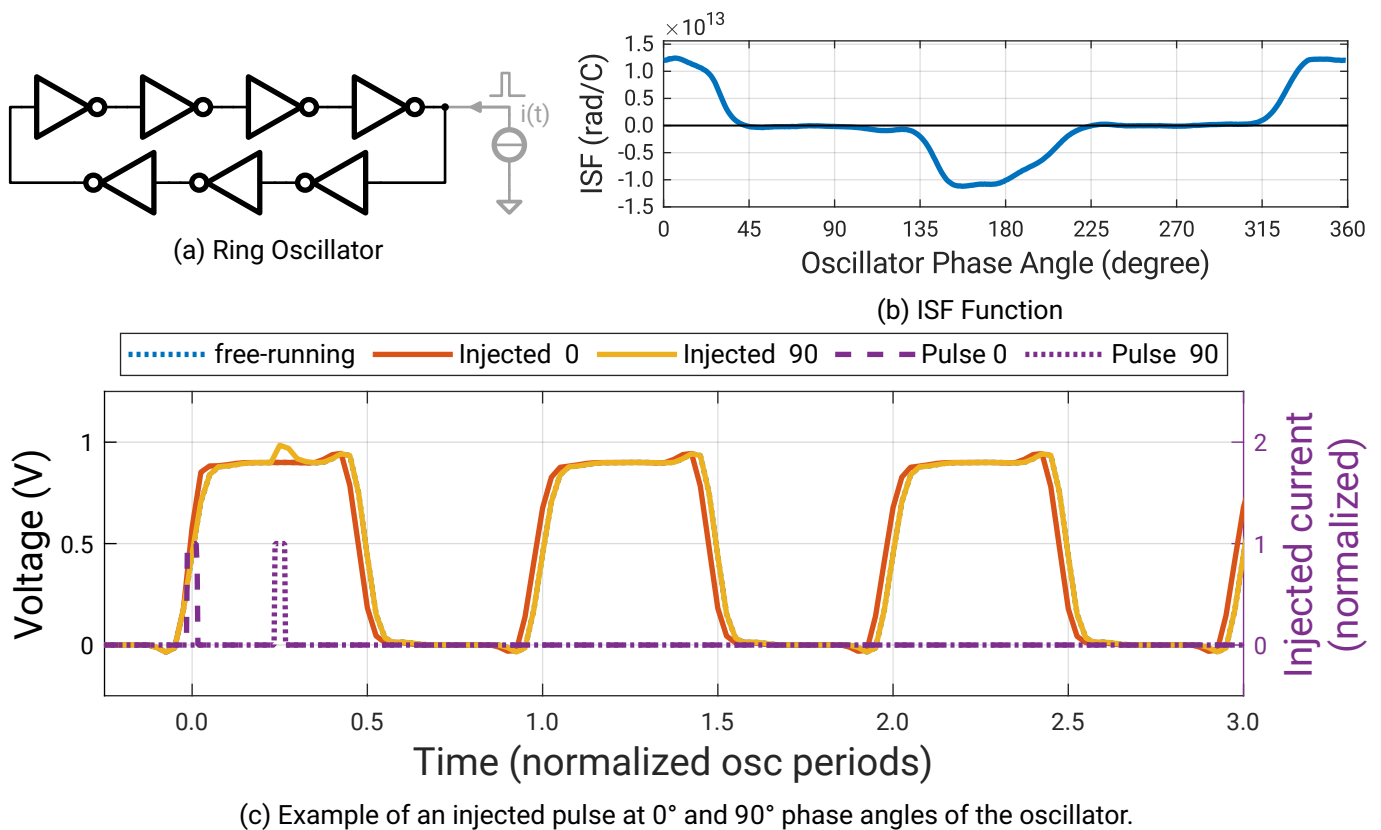


Figure 4.5.: Example of the ISF for a simple ring oscillator. (a) Schematic of the 7-stage ring oscillator consisting of standard CMOS inverters. The charge injection source is highlighted in gray. (b) Numerical extracted ISF function. (c) Transient pulse injection at a phase angle of  $0^\circ$  and  $90^\circ$  showing a phase shift for the  $0^\circ$  case and almost no influence at an angle of  $90^\circ$ .

### Sensitivity Non-linearity

The ISF principle assumes linearity in the sensitivity, which can be reasonable for the typical small noise currents. So, injecting twice the current leads to a doubled phase change. However, the currents for injection used in OIMs exceed noise currents by orders of magnitude and might even temporarily exceed the oscillator's normal operating currents. The oscillators can change the phase (e.g. by  $\pi$  to represent a state change of the discrete variable) within a few oscillation periods. To evaluate the non-linearity of the ISF approach regarding the injected current, the numerical extraction method is conducted for varying magnitudes of injected currents. The oscillator of the Glowworm tapeout is used to estimate the non-linearity in Figure 4.6. A pulse with a rising/falling time of 10 ps and a width of 100 ps is used (1% of the period). The height of the pulse is normalized to the bias tail current of the oscillator (see Section 5.2 for more details). To enhance the accuracy of Cadence Spectre, the accuracy preset 'conservative' was modified as follows:  $reltol = 1e-6$ ,  $vabstol = 1e-9$ ,  $iabstol = 1e-15$ . The obtained ISF seems to be robust despite the injection reaching or exceeding the actual oscillator's stage bias current. There is a considerable distortion at 500%. However, there are less distortions at 100% compared to the lower strengths. Numerical inaccuracies lead to slight offsets at small injected currents during the insensitive region. Even when considering the non-linearities caused by strong injections, the ISF approach will still more accurately model the oscillator behavior than the Kuramoto model.

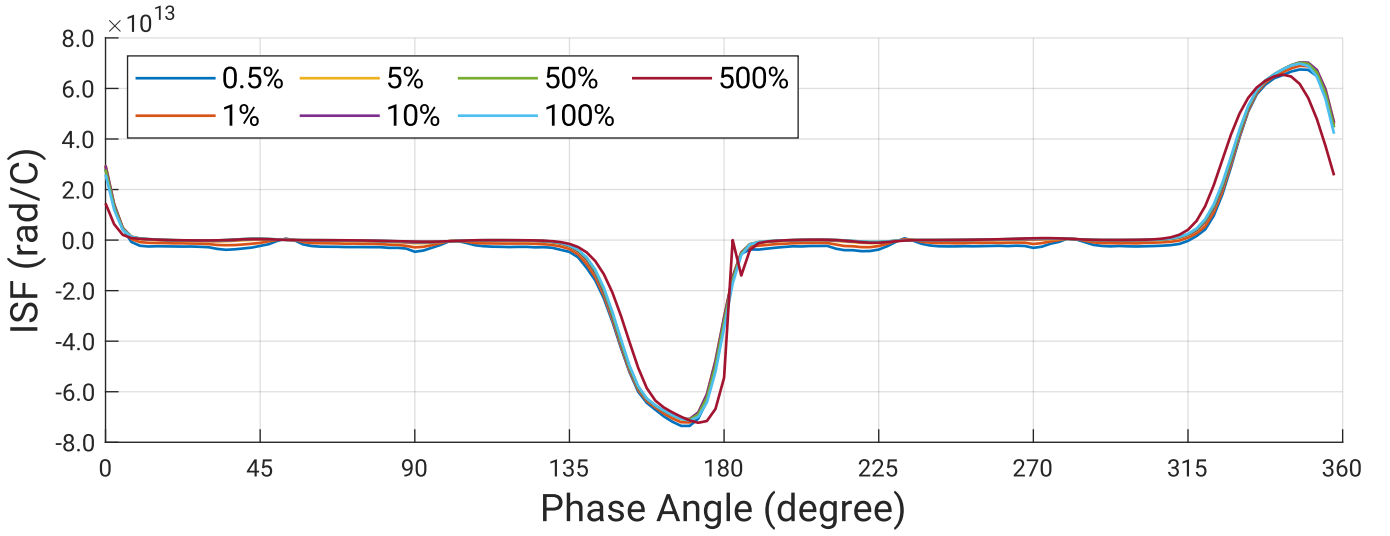


Figure 4.6.: Extraction of the ISF function of the oscillator used in the Glowworm design with varying injected pulse heights. They are normalized to the tail bias current of the oscillator, meaning that 100% corresponds to an injecting current with magnitude of the bias tail current.

### Perturbation Projection Vector (PPV)

The ISF uses a linearization of the oscillators trajectory. The assumption of orthogonality, so that tangent perturbation lead to phase changes and orthogonal changes to decaying amplitude changes has been shown imprecise [119]. So, a more precise phase macro model using the perturbation projection vector (PPV) has been proposed for describing phase noise and injection locking [116], [120], [121]. The excess phase  $\alpha$  of the oscillator is described as [116], [121]:

$$\frac{d\alpha(t)}{dt} = v_1(t + \alpha(t))b(t) \quad (4.5)$$

$v_1(t)$  is the PPV and  $b(t)$  the perturbation. So, the PPV characterizes the oscillators sensitivity, where the change depends on previous perturbances. The authors proposed various methods to obtain this PPV, which can be for example done by the Jacobian matrix of the oscillator [120]. The theory behind the PPV is discussed in literature and is less intuitive. The ISF and PPV are interchangeable for the OIM model, as discussed later.

#### 4.2.2. ISF-based OIM Model

The modeling for OIMs differs from traditional (analytical) models in two fundamental aspects:

- Periodicity: The injecting signals are not periodic and change based on the reaction of the oscillator
- Amplitude: The injection is strong and leads to changes of multiple radians per oscillation period

While the injection locking analysis usually assumes a constant, periodic injecting signal, this criterion is not satisfied by the bidirectional coupling within OIMs. When a pair of oscillators couple bidirectionally, one oscillator impacts the other and vice versa. Consequently, the injection signal does not have a constant amplitude or is strictly periodic. It even depends on the reaction of the injected oscillator itself. While noise has a small amplitude, which randomly alters the oscillation period in the form of jitter, the phase change from coupling is much stronger. A oscillator might change its phase by  $180^\circ$  within just a few cycles.

Since this thesis is focused on the circuit design of OIMs, the simulation approach should be easy to use and work with any possible topology of oscillators. An analytical method for the ring oscillator in a deep submicron technology with complicated device models is likely too complex. However, applying simplifications might not maintain the behavior accurately to compare different designs or circuit topologies<sup>†2</sup>. Hence, an ISF-based extension of the (modified) Kuramoto model is used, which is presented in the author's publication [159]. The images in this section are adapted from this publication. The purpose of the ISF is to describe the oscillator's sensitivity to perturbations<sup>†3</sup>. Adapted from Equation 4.2, a more general form can be described as:

$$\frac{d\phi_i}{dt} = \omega_i^* + \sum_{j=1, j \neq i}^n f_{cpl, nl}(\phi_i, \phi_j, J_{ij}) + f_{shil, nl}(\phi_i, \phi_{sync}) \quad (4.6)$$

Here,  $f_{cpl, nl}$  describes the reaction on the phase of a coupling input, which depends on the phases of both involved oscillators and the coupling strength. It can be composed as  $f_{cpl, nl} = ISF_{osc}(\phi_i) \cdot I_{cpl}(\phi_i, \phi_j, J_{ij})$  by the sensitivity  $ISF_{osc}$  and the injected coupling current  $I_{cpl}$ . Similarly,  $f_{shil, nl}$  describes the reaction on a SHIL input as a function of the phase of the injecting signal and the current phase. It is composed analogous to the  $f_{cpl, nl}$ . These functions are numerically extracted from the transistor circuits using an analog circuit simulator and saved as a lookup table. This allows an analysis using the differential Equation 4.6 directly or it can be included in a circuit simulation approach. The lookup table can be implemented in a verilog-A module to allow a convenient verification of the models within schematic simulations. Another lookup table containing the phase-dependent oscillator output voltage can then generate an approximation of the oscillator's waveform.

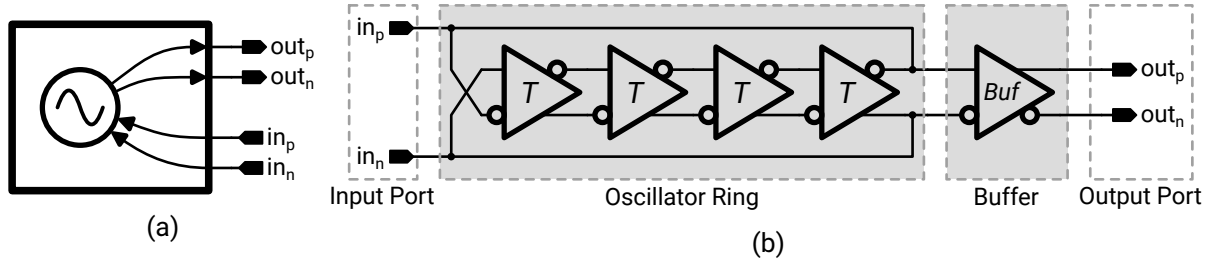


Figure 4.7.: Structure of the oscillator for extraction. (a) The oscillator can be treated as a gray box with an input and output port. (b) The general oscillator topology. For illustration purposes, a 4-stage differential ring is shown as used in the OIM design.

<sup>†2</sup>Especially differential delay cells have various possible topologies for the active load of the differential pair.

<sup>†3</sup> **A note on ISF vs PPV:** The here proposed method needs a function to describe the oscillators phase change  $d\phi$  for an injected current  $di$ , which is the response of a coupling circuit. Other techniques, such as modeling the phase noise, of the ISF and PPV approach are not applied. The Equation 4.6, has the same structure as Equation 4.5 of the PPV approach. Hence, one major critique regarding the ISF does not apply here [119]. Consequently, the differences between the ISF and PPV are only regarding their oscillator sensitivity function and its calculation. Assuming correct normalization of the ISF and PPV, both can be interchangeably used in this model. The calculation of the ISF with the transient pulse injection method approximates the PPV for sufficiently small pulses [122]. This method for the ISF allows to consider the sensitivity non-linearity as discussed in Section 4.2.1. Because the ISF is the more intuitive approach, this OIM model is named after the ISF.

## Extraction

For the numerical extraction, the scenario shown in the figure on the right is considered. Two oscillators with phase  $\phi_i$  and  $\phi_j$  interact by a coupler implementing the weight  $J_{ij}$ . This is sufficient to model a network of any size, which consists of multiple such pairs of two coupled oscillators, because a predominantly linear behavior of the interaction is assumed. The structure for the oscillator is shown in Figure 4.7a and an exemplary physical topology is shown in Figure 4.7b. Because the oscillator output can be buffered, as discussed later the design in Section 5.2, each oscillator is assumed to have an in and out port. The in port receives the coupling current and allows changing of the oscillator phase. The out port provides a buffered oscillator waveform, which is distributed to all couplers and used to determine its phase angle. Without loss of generality, the in and out ports could be set identically for the discussed model. Since the oscillator implementation uses differential signaling, the here shown signals are differential as well. The model can be also applied to single-ended systems.

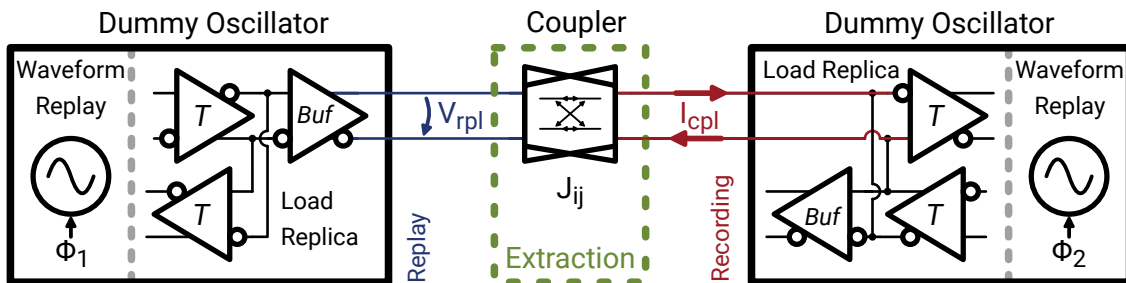
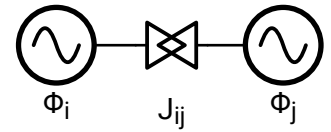


Figure 4.8.: Extraction principle for the phase angle dependent coupler current  $I_{cpl}$ . The dummy oscillators use the loop-unrolling technique shown in Figure 4.9.

The extraction principle for the coupler current  $I_{cpl}(\phi_i, \phi_j, J_{ij})$ , which is a function of the phases of both involved oscillators  $\phi_i$  and  $\phi_j$  and the coupling strength  $J_{ij}$  is outlined in Figure 4.8. The coupling is bidirectional but consists of two back-to-back connected unidirectional elements. Due to the buffered oscillator output topology, such a back-to-back topology is automatically required. Because the unidirectional elements are identical, extracting the behavior in one direction is completely sufficient. The coupler circuit is connected

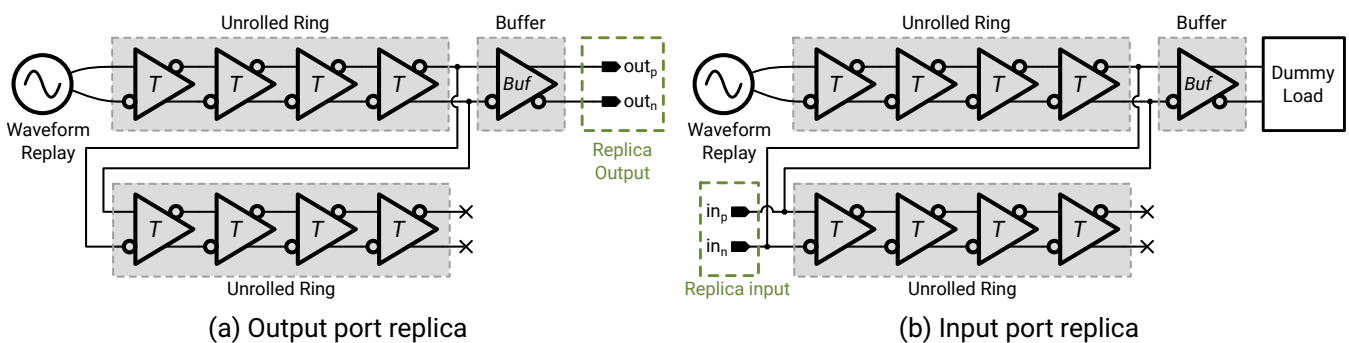
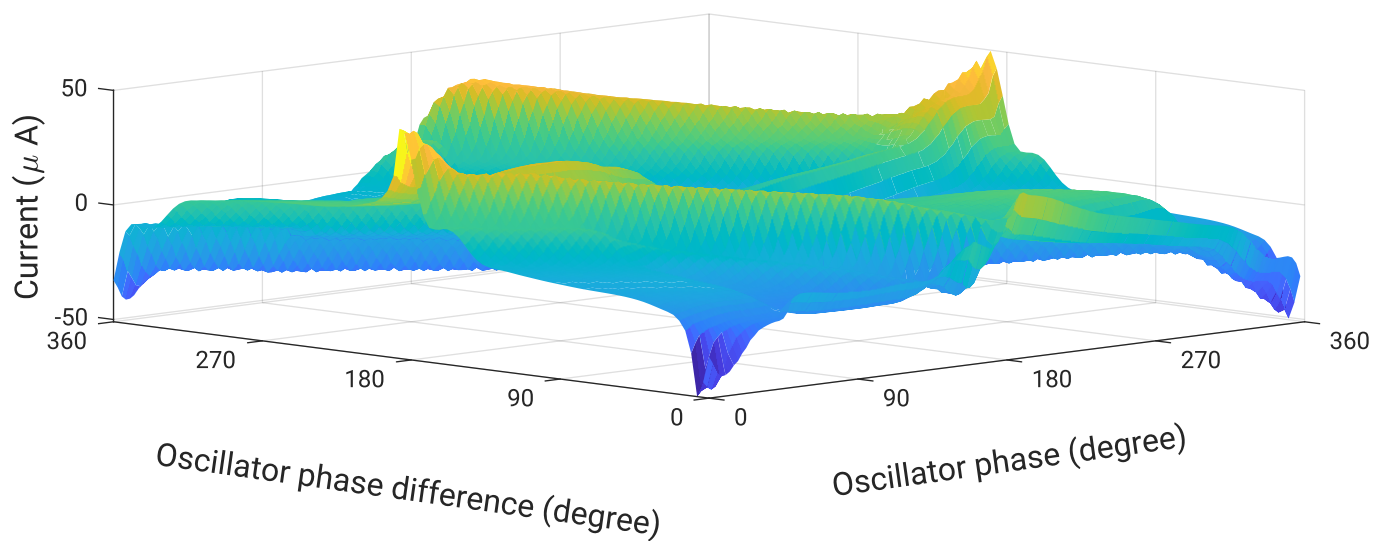
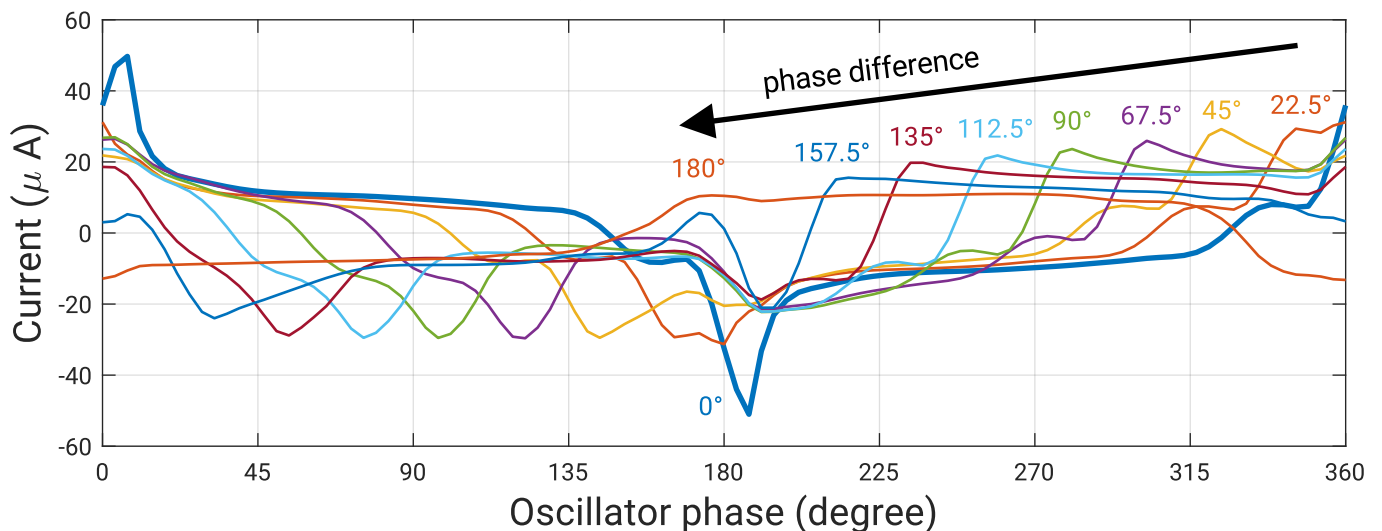


Figure 4.9.: Oscillator unrolling technique for accurate replica loads. The oscillator ring is unrolled by concatenating two (or more) rings. Two slightly different replica setups for the input and output ports are necessary. The length of the unrolled ring can be adapted to the circuit's sensitivity to loads.

to two dummy oscillators, which can be set to any arbitrary phase and have an accurate replica of the electrical load. A sweep for multiple phase difference  $\phi_2 - \phi_1$  is conducted and  $I_{cpl}$  is recorded over one period after a few initial oscillation cycles to reach a periodic steady state. Weights can be either implemented by actually extracting  $I_{cpl}$  at different weight settings or can be approximated by scaling the recorded  $I_{cpl}$  accordingly. To obtain an accurate load and drive strength replica for the dummy oscillator, a ‘loop-unrolling’ technique is used as shown in Figure 4.9. The recorded waveform travels through an ‘unrolled ring’ of the oscillator using a copy of the delay cells. So, the phase is precisely controlled by the waveform replay, while the driving strength and load (the impedances) are accurately replicated by the delay cells. The connected output load used for recording the oscillator waveform and the applied load (respectively dummy load) during the current



(a) 3-dimensional plot



(b) Current for selected phase angles

Figure 4.10.: The current of the coupler circuit is extracted from the circuits of the Glowworm design. (a) The 3-dimensional plot, where the injected current depends on its phase and the phase difference of both connected oscillators. (b) Plot for selected phase differences from 0° to 180°. Because the currents are symmetric, only phase differences up to 180° are shown.

recording must match. Any load differences in the setup will change the frequency and thus introduce phase offsets. A drawback of this proposed technique is, that a ring topology is required. Since a load replica is required, the oscillator can not be treated as a black box. However, similar replica loads could be developed for other oscillator models.

Figure 4.10a shows an exemplary coupling current as a function of the phase of one oscillator and the phase difference to the other oscillator. The injected current has two spikes because the circuit's internal capacitances are periodically charged and recharged. Especially the steep slopes of the oscillator's transitions induces a current peak. Such a charging/discharging current also remains when the coupler is disabled. Figure 4.10b shows the injected current over one oscillator period for selected phase differences of  $0^\circ$  to  $180^\circ$  in  $22.5^\circ$  steps.

## Simulation

The behavior of the model is validated exemplary using the previously used simple example of two oscillators. This setup is simulated with the proposed model and with the analog circuit simulator Cadence Spectre. The model operates in the phase domain, where the phase is converted to an output voltage for this example. The model is not intended to provide an accurate replica of the waveform. It should just match the coupling behavior in the phase domain. The comparison in Figure 4.11 shows, that the coupling behavior is very similarly replicated. The proposed model tends to couple slightly faster than the actual transistor circuit. So, it reaches the goal of qualitatively describing the coupling behavior. A comparison with the Kuramoto model is omitted because it is not modeling the circuit behavior itself. Thus, there is no direct transformation between circuit properties such as currents and the coupling strength to the Kuramoto model. Hence, this would just result in a parameter fitting. A comparison based on solving an actual optimization problem is also difficult, since any differences in their phases accumulate and will lead to significantly altered solutions, preventing a meaningful comparison.

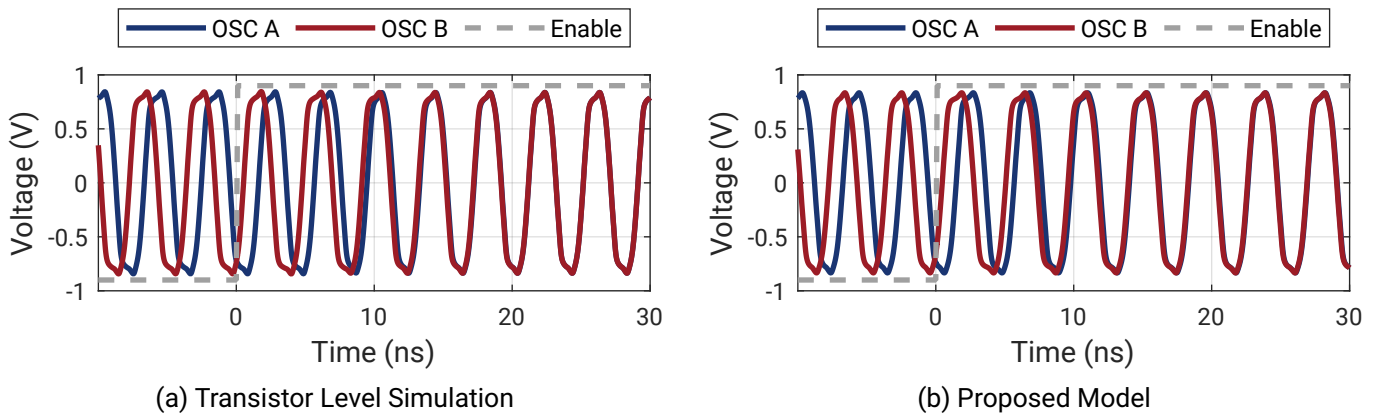


Figure 4.11.: Comparison of the proposed ISF and look-up table based model with the actual transistor circuit. Two oscillators are coupled in phase, where the coupler activation is indicated by the enable signal.

### 4.3. Schematic Simulation Setup

A higher simulation accuracy can be achieved using a schematic-level simulation. This is based on the foundry-provided transistor models and uses an analog circuit simulator. These compact models are computationally simple enough for analog circuit simulators like SPICE or Cadence Spectre [123]. Additionally, they are precise enough to predict the device behavior with sufficient accuracy for designing circuits [124]. It should be considered that random manufacturing variations will slightly alter the properties of the manufactured transistors. So the circuit needs to be robust against variations anyhow. Additionally, parasitic extraction of the layout identifies resistances and (coupling) capacitance and can be included in this approach. So, this is presumably the most accurate simulation approach for OIMs, but it also needs the longest simulation runtime. In practice, the simulation runtime is the limiting factor of this approach and restricts the maximum network size that can be simulated.

The initial simulation framework was developed in the supervised master thesis by Santhosh Nagaraj [164]. The framework got adapted and enhanced over time to meet the simulation demand. Essentially, it generates a reduced schematic of the oscillator network, which just uses the minimum of required components for simulation. This is combined with an automatic analysis of the waveforms to derive the success of the computation. The simplified flow for simulation is outlined in Figure 4.12. The flow starts with providing an input problem in the form of a maximum cut problem to the master Cadence Skill script. The nodes and edges are parsed from the optimization problem to generate a custom schematic. To save computation time, only necessary elements, which are provided using a building block library, are used. This most minimalistic setup instantiates an oscillator including SHIL injector for each node and a coupler circuit for each edge of the optimization problem. The remaining periphery and control circuitry for the operation of the OIM is replaced by ideal components and verilog-A models to reduce the number of elements in the simulation. A verilog-A module reads out the phase information and ideal current sources replace the digital-to-analog converters (DACs) for weight settings. Based on the configuration setting of the script, additional elements like disabled couplers can be included to achieve a more realistic simulation. Furthermore, frequency variations, clock skew and other non-idealities can be optionally included. A Python script generates the customized Cadence Ocean

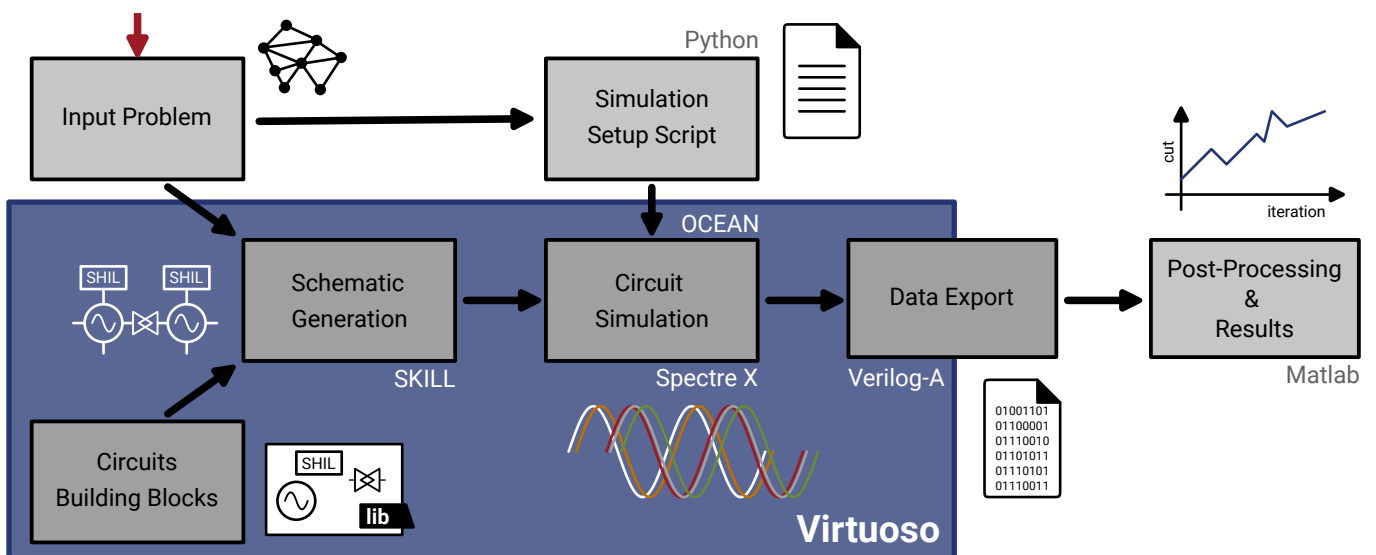


Figure 4.12.: Overview of the schematic level simulation flow. A Cadence Skill script controls the flow and uses Python, Matlab, and Cadence Ocean scripts for the sub-tasks.

---

script for each optimization problem to coordinate the simulation. The phase information, which is extracted based on the zero crossings of the differential oscillator signal, is automatically written into a temporary file. After the simulation, a Matlab post-processing script is executed, which obtains the phases and states based on the recorded zero crossings. The states allow the calculation of the problem's cut to judge the effectiveness of the simulated circuits. Other information, such as the phase distribution and waveforms, are also provided.

**Full System Simulation** Especially for verification purpose, it would be beneficial to simulate the whole schematic. However, a full system simulation, even on schematic level, was not possible for any of the developed 400 and 1440 node OIMs. The DC-operating point calculation alone, which is just the initial step for the transient simulation, did not finish within 24 h using a Ryzen Threadripper Pro 5965WX 24-core system with 256 GB RAM for the 1440 node system. Consequently, the simulation was aborted because no meaningful results could be expected within a reasonable time. A post-layout simulation for verification purposes was conducted for up to 16 oscillators, which needs over a day of simulation runtime.

**Phantom Coupling via the Simulator** Coupling needs some sort of mutual interaction between the oscillators. In the scope of OIMs, this is usually done with purposely designed coupler circuits. Other unintended coupling paths, such as cross-talk between nearby traces or disturbances via the power-supply, are possible as well. However, coupling of two independent oscillators without any mutual connection was also noticed in some simulations during this work. This is caused by the circuit simulator and is a pure simulation phenomenon. Hence, it could lead to confusion by suggesting behavior that is not physical and is just a numerical artifact.

The underlying cause for the (unwanted) coupling is the time-step control of the circuit simulator. The step size is dynamically adjusted based on the accuracy settings, as illustrated in Figure 4.13. A simple 3-stage ring oscillator with just 6 transistors is simulated. The step size decreases with higher accuracy, but it also varies throughout an oscillator period to account for the dynamic transitions and high-frequency components of the over/undershoots of the waveform. So, the step size is non uniformly distributed over an oscillation period. The frequency with 'moderate' accuracy preset is 0.02% higher and the 'liberal' is 1.6% slower than for the highest accuracy setting 'conservative'.

Figure 4.14 demonstrates the coupling via the oscillator time steps. Two identical and independent oscillators within a schematic are simulated without any intended mutual interaction. Multiple simulations with initial phase differences between  $0^\circ$  and  $360^\circ$  with steps of  $1^\circ$  are conducted. Although the phase difference should just stay constant, it tends to group in one of six phases. The alignment takes approximately 400 oscillator periods and exhibits meta-stable behavior between the groups. The coupling mechanism is via the time step control of the simulator. During the simulation with the 'conservative' accuracy preset, each oscillator requires a periodic increase (and relaxation) of the step size. Hence, depending on the phase of the other independent oscillator, the waveform might be simulated with a smaller step size. Consequently, the simulated voltage might vary by a tiny amount compared to a simulation of just one oscillator. In simplified terms, this step size change can be imagined as a modulation of the simulation accuracy. Since this will affect the simulated oscillation period, a coupling phenomenon can be observed. When the simulator is constrained to a fixed, small step size, the oscillator phase difference stays constant at its initial value as expected. This proves that the coupling mechanism is via the step size variation of the simulator.

Although the coupling via the time step control of the simulator is of limited practical relevance, it highlights the great sensitivity of coupling. Even tiny disturbances can cause locking behavior between oscillators, which might cause non-physical simulation results. However, significant frequency differences can not be overcome, because the actual phase changes of this mechanism are very small. In practice it might lead to unexpected



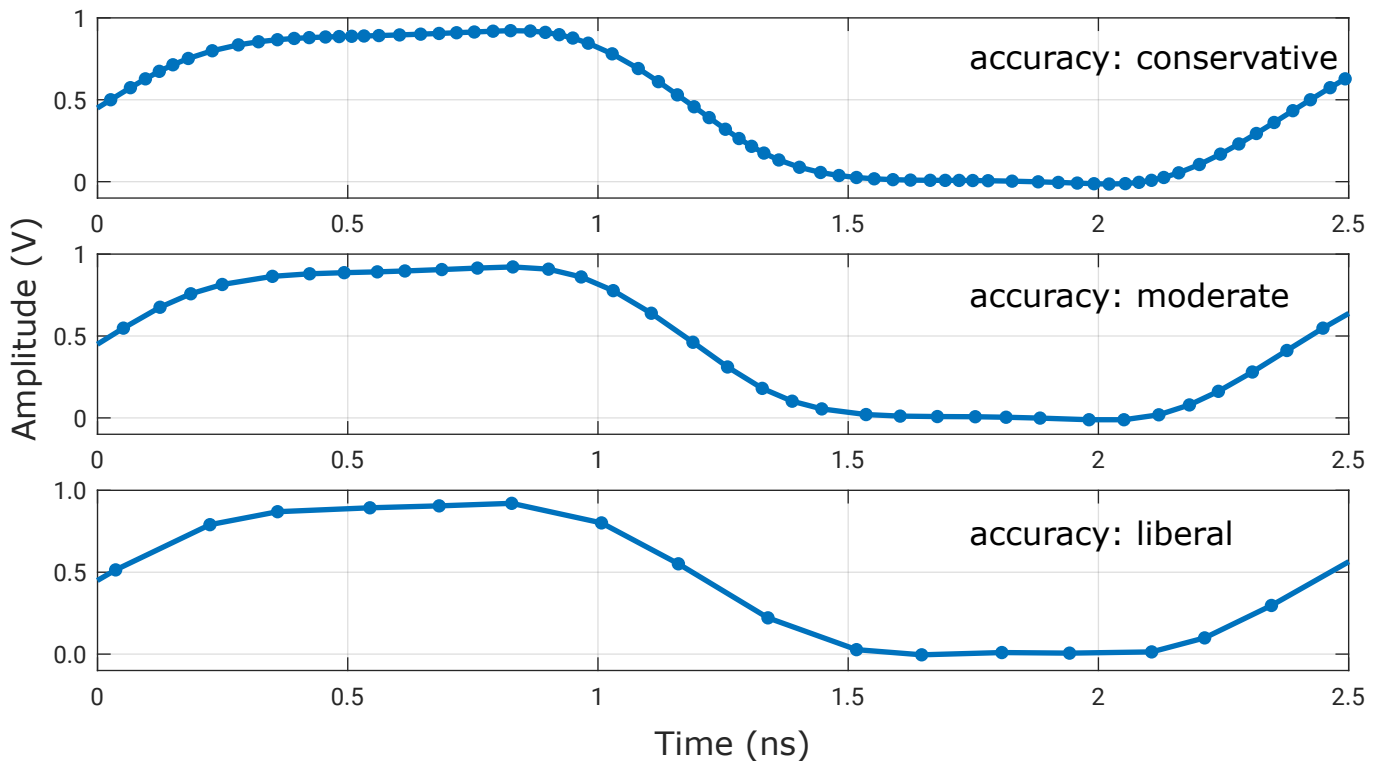


Figure 4.13.: The accuracy preset of the simulator Cadence Spectre significantly affects the step size of the transient simulation. The accuracy increases from the preset ‘liberal’ to ‘conservative’. The step size of the simulated 3-stage ring oscillator varies non-uniformly across a whole period to meet the accuracy requirements.

simulation results in some situations, but it is neglectable when any form of intended coupling is present in the simulation.

#### 4.4. Comparison

A comparison between the simulation models is provided in Table 4.1. While the Kuramoto model and the ISF-based OIM model are based on a system of differential equations, the schematic simulation approach uses a circuit simulator. Consequently, the equation based models have a higher abstraction level and are less related to the circuits. The ISF-based OIM model is a good trade-off between simulation time and accuracy, because it at least includes the approximate circuit behavior. However, the schematic simulation flow is important to verify the developed circuits. Otherwise design issues like loading or frequency shifts might not be noticed at higher abstraction levels.

The design process in this work started with the Kuramoto model. It gives first insights about the design goal and e.g. the coupling behavior such as the  $\tanh(\sin())$  can be easily explored. Then the ISF-based OIM model was developed to bridge the gap to the circuit design. Not only does it provide fast simulation, but the extracted behavior helps to understand the circuit and find suitable topologies. The more advanced the design becomes, the more the focus shifts to the schematic simulation flow. It includes other important effects such as loading, which can potentially ruin the performance. A ‘Full System Simulation’ of the whole OIM design was

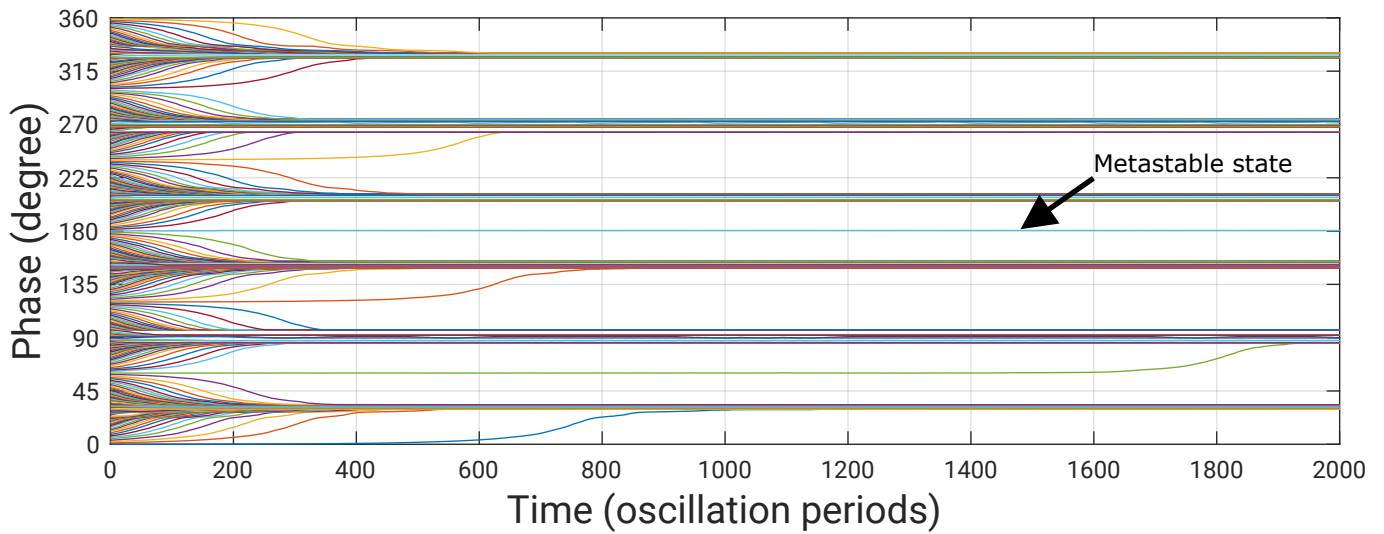


Figure 4.14.: Two identical but fully independent oscillators can couple due to the time step adjustment of the simulator. Depending on the initial phase difference between the oscillators, they align in 6 levels for the 3-stage design used as illustration. A metastable state can be reached as well, which might eventually tip into the stable levels. The shown behavior is not physical and purely caused by the simulator.

never conducted, but a reduced system with up to 16 nodes, including all circuitry, was extensively conducted for verification purposes. Later in the design, the coupling behavior was usually analyzed on a schematic level using simple setups with a handful of nodes. This helps to fine-tune the circuit topology and transistor sizes. In particular, random mismatch needs to be considered through extensive Monte Carlo simulations to analyze the contribution of each individual transistor.

	<b>Kuramoto Model</b>	<b>ISF-based OIM Model</b>	<b>Schematic Simulation Flow</b>	<b>Full System Simulation</b>
<b>Simulation Principle</b>	System of differential equations	System of differential equations	Transistor Models	Transistor Models
<b>Abstraction Level</b>	High	Medium	Low	Low
<b>Coupling Model</b>	$\sin(\phi_j - \phi_i)$	Lookup Table	Circuit	Circuit
<b>Relation to Circuit</b>	none	Extracted Behavior	Core Circuit (Oscillator, Coupler, SHIL)	Full Circuit
<b>Simulation Accuracy</b>	Low	Medium	High	High
<b>Simulation Time for systems with 100s of nodes</b>	Fast (seconds - minutes)	Medium Fast (minutes)	Slow (hours-days)	Very Slow (unknown: >days)
<b>Setup Complexity</b>	Simple	Difficult	Difficult	Medium

Table 4.1.: Comparison of the discussed OIM simulation approaches. A simulation of the full circuit is included for reference.

---

## 4.5. Conclusion of the Modeling and Simulation

The modeling and simulation of OIMs were discussed. There are three levels of abstraction, which trade-off simulation runtime and accuracy. The Kuramoto model is based on a system of differential equations. Since it is a general model, it does not directly relate to the actual circuit. It can be used on a system level and provides the behavioral guideline for further system design. Since schematic-level simulations with the transistor models are very time-consuming, an ISF-based approach is proposed. Essentially, it is a modified version of the Kuramoto model, which uses the ISF function to describe the oscillator's phase response to coupling inputs. Lookup tables are generated by a sophisticated periodic steady state setup including load replicas to approximate the circuit's behavior. However, actual schematic-level simulation is essential to verify and fine-tune the circuits. Additional Monte Carlo simulation are crucial to improve the robustness of the circuit. Because a simulation of the full system with a meaningful number of nodes is too time-consuming, a custom schematic is auto-generated, which only includes the core circuits such as oscillators, SHIL injectors, and couplers.

### Modeling and Simulation

- The **Kuramoto model** does not consider the actual OIM circuitry but is well suited for fast system-level simulations.
- The **ISF-based OIM model** extracts the behavior of the oscillator and coupler using a circuit simulator, which is included as look-up tables in the underlying coupled differential equations. Since it is based on the circuit's behavior in a periodic operating point, it is only valid within reasonable coupling strengths.
- The **Schematic Simulation Flow** provides the highest precision and simulates the transistor behavior. After the initial setup of the flow, it can quickly compare different circuits, but does not provide detailed insight about the circuit operation. Compared to both other approaches, it has by far the longest simulation runtime.
- A simulation of the **full OIM system** is **impractical** due to the excessive runtime. Therefore, only a small section with up to 16 oscillators including their peripheral circuits was post-layout simulated for verification purposes.

---

## 5. System and Circuit Design

---

This chapter discusses the circuit and chip design of OIMs. While simulations are an important and capable tool, they are insufficient to evaluate the true performance of OIMs. Hence, two designs were fabricated in a 28 nm technology to experimentally verify its performance, which is later discussed in Chapter 7. The first chip generation is called ‘Firefly’ and the second generation ‘Glowworm’. The Firefly design consists of 400 oscillators and 1482 edges on 2.2 mm<sup>2</sup>. The Glowworm provides 1440 oscillators and 11724 edges on 4.6 mm<sup>2</sup>. This chapter starts with an overview of the system design and then discusses the circuit building blocks in detail.

A trade-off between the computation capabilities, the area, and power consumption is essential for the OIM design. However, judging the computation capabilities of a circuit is not as straightforward as checking the performance of a traditional analog circuit based on specifications like the gain or phase margin. As of this writing, there is no established set of specifications directly related to the computing performance to guide the design of the individual building blocks. Additionally, the computing capabilities can not be easily measured or simulated compared to, for example, the gain of an amplifier. Also, there is no established method to guide the circuit design based on circuit level design parameters. Thus, optimizing the design for its computation capabilities creates new challenges.

This chapter starts with a system-level description in Section 5.1 to explain the structure and the design goal for the OIM implementation. Then the oscillator design is discussed in Section 5.2, the coupler in Section 5.3, and the SHIL injector in Section 5.4. The routable FPGA-like connections are discussed separately in Section 5.5 and the spin-bias connections, implementing the  $h_i$  term of the Ising model, in Section 5.6. Afterwards, the digital configuration is presented in Section 5.7 and the overall physical implementation in 5.8. An overview of both fabricated dies is provided in Section 5.9 and Section 5.10 discusses future improvements before concluding this chapter.

### 5.1. System Design

The system design starts with a simple overview of the OIM hardware structure for convenience. Then, the design goals are discussed, the system level coupled oscillator network presented, and the building blocks of an ‘oscillator node’ discussed. Such an ‘oscillator node’ contains all circuits to implement a discrete variable  $\sigma_i$  of the Ising model. Lastly, the choice of technology for fabrication is discussed.

#### 5.1.1. Overview

An optimization problem in the Ising formulation can be visualized as a graph as shown by Figure 5.1a. It consists of  $n$  nodes for the discrete variables  $\sigma_i$ , which are linked by the weights  $J_{ij}$ . This graph is implemented as a physical coupled oscillator network, as conceptually shown in Figure 5.1b, which illustrates just a fraction

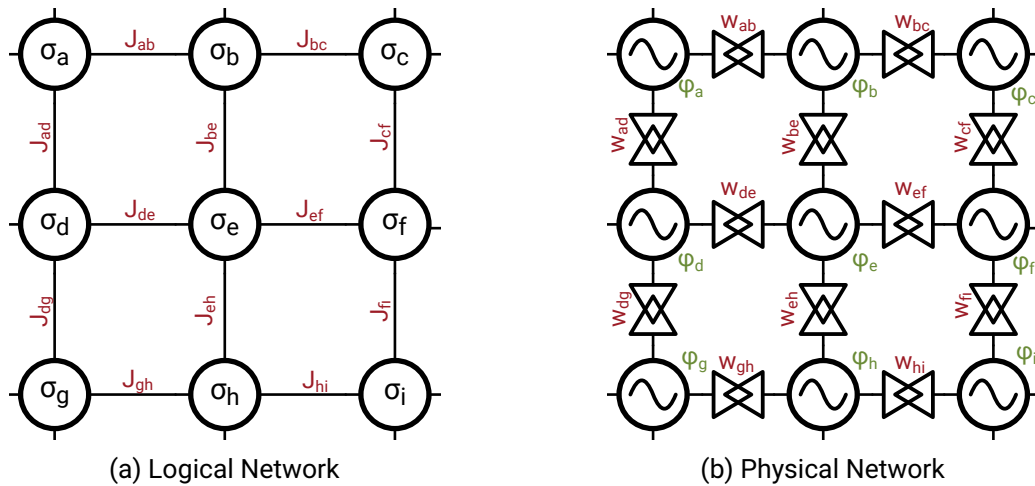


Figure 5.1.: An exemplary illustration of the Ising model as network. (a) The Ising model in a graph form. (b) Representation of the equivalent coupled oscillator network. For illustration purposes, a sparse network with only 4 connections per oscillator is shown. In general any node could be connected to any other nodes. The weights  $J_{ij}$  (red) between the nodes are implemented by a coupling weight  $w$  (red) and the discrete states of the Ising model  $\sigma$  are represented by the oscillator phases  $\phi$  (green).

of all possible connections  $J_{ij}$ . Each discrete variable  $\sigma_i$  is represented by the phase of an oscillator  $\phi_i$ . The weights  $J_{ij}$  are implemented by a coupler circuit with strength  $w_{ij}$ .

The hardware implementation of such a coupled oscillator network is formed by duplicating and connecting multiple oscillator nodes. A simplified oscillator node is presented in Figure 5.2. Such a conceptual oscillator node consists of one oscillator, its couplers and additional circuitry to support the operation. The output of the oscillator is additionally buffered for transmission to other nodes. The discretization of the oscillator phases is implemented by a SHIL injector circuit. The configuration memory provides the weights  $w_{ij}$  of the optimization problem. The DACs transform these weights into the analog domain, which controls the strength of the couplers. Therefore, the resulting interaction between the oscillators is set accordingly. A phase-to-digital converter (PDC) measures the oscillator phase after computation to derive the discrete state as a solution.

### 5.1.2. Design Goals

In order to evaluate the capabilities of OIMs and provide a feasible design, several opposing goals exist. From a system point of view, the OIM design should strive for the following criteria:

- Large network
- Dense connectivity
- High solution accuracy (finding near-optimal solutions)
- Fast computation
- Low area consumption
- Low power consumption

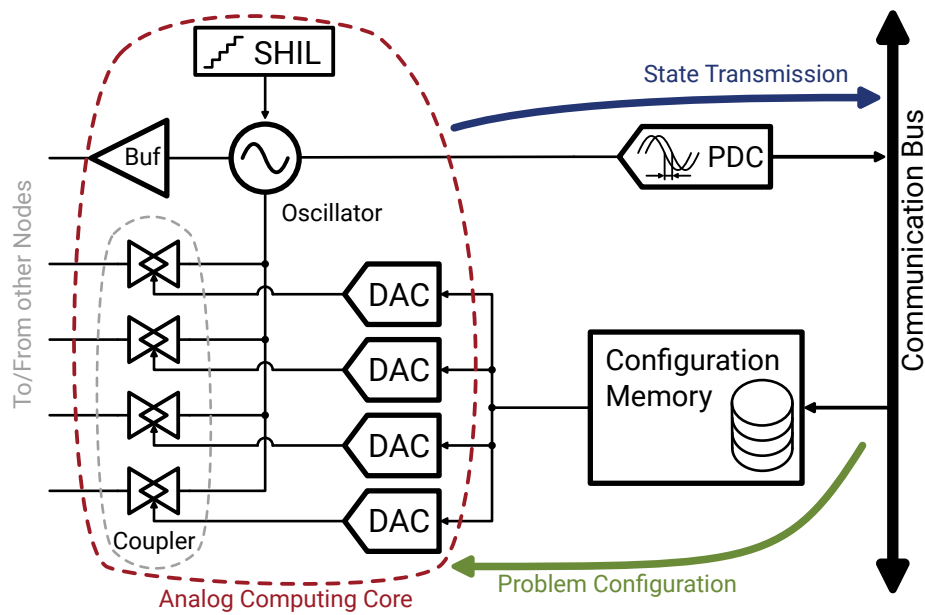


Figure 5.2.: Simplified overview of an oscillator node. The actual computing is done in the analog domain by the oscillator and couplers. A digital communication bus serves as a back-bone to configure the oscillator network and transmits the solution states of the computation.

A large network with dense connectivity enables solving large optimization problems. The obtained solutions should be as close as possible to the global optimum with a computation as fast as possible. The physical design should be small and have a low energy consumption per computation. Obviously, those targets are partially contradictory. For example, a dense network will use more area than a network with reduced connectivity.

As this thesis demonstrates the capabilities of OIMs and proposes a feasible design, a trade-off between solution accuracy and area consumption is central. A low solution accuracy makes such an OIM system of limited use, as a poor solution might not satisfy the demand of the application behind the optimization problems. The same applies to the network size and connectivity, where a small network is not usable as well. It is yet unknown how the computing time and accuracy are affected when the problem size increases. Hence, a network of meaningful size, which has more than a few nodes, is desirable from a scientific point of view. The number of possible combinations and thus the search space of such small networks is not sufficient to meaningfully determine the accuracy of a network.

The proposed designs should be scalable, meaning that they can be adapted to much larger networks if more silicon area is available. The available silicon area for prototyping in this work is limited to a few square millimeters. However, much larger die sizes are technically possible, as demonstrated by countless commercial system on chips (SoCs), CPUs, GPUs, and other designs. Even dies with an area of more than 500 mm<sup>2</sup> are possible for the 28 nm technology node used in this work<sup>†1</sup>. An area density of hundreds of nodes per square millimeter is targeted as this will lead to hundreds and more nodes at the available prototyping size. So, a meaningful network size for the performance evaluation is achieved. The power consumption is less important and of lower priority for the design in this work. Consequently, the focus is on a compact, scalable design with high computing accuracy.

<sup>†1</sup>The ten-core Intel Westemere-Ex processor has a die size of 513 mm<sup>2</sup> and was released in 2011. It uses a planar 32 nm node [125]. Nvidia's Kepler GK110 GPU has a die size of 561 mm<sup>2</sup> and was released in 2012 produced in a 28 nm node [126].

---

### 5.1.3. Network Design & Embedding

In order to solve an optimization problem on the Ising machine, it must be first embedded in the hardware network. This means that all nodes  $\sigma_i$  and edges  $J_{ij}$  need to be assigned to physical counterparts by a minor embedding algorithm. Unfortunately, this graph minor embedding can in itself be a NP-hard problem [112]. Essentially, a new optimization problem is created to solve the original problem. Consequently, Ising machines can only be beneficial if the embedding on their hardware is ‘easier’ than the actual optimization problem.

An all-to-all connected network of sufficient size would make the embedding straightforward and time-wise negligible. Because every physical oscillator is connected with any other physical oscillator, all edges  $J_{ij}$  exist in hardware. A network with  $n$  nodes needs  $\sum_{i=1}^n (n-i) = \frac{n(n-1)}{2}$  edges, which scales quadratically with the size. At the time of this writing, implementations with all-to-all connectivity have only been shown until a size of 59 nodes, which just needs 1,711 edges [127]. A system with 100 nodes would need 4,950 edges, and with 1000 nodes, already 499,500 edges.

The alternative approach is a sparse network, where each node is connected to only a (small) subset of all other nodes. For example, each node could be connected to a fixed number of nodes in their proximity or implement only a fraction of all possible edges, which drastically reduces the needed number of edges. With a fixed number of edges per node, the needed edges would scale linearly with the number of nodes. However, the network topology must be suitable for the targeted optimization problems. It is possible to connect multiple physical oscillators together to form a single logical node so that more physical connections are available for this logical node. Hence, the available oscillators can be exchanged for more connectivity.

The NP-hard embedding can be very time-consuming and might become the bottleneck of the computing [156]. To overcome this embedding bottleneck, routable connections are introduced. These allow the connection between any arbitrary oscillators (nodes/variables of the optimization problem) as long as a suitable path is available in the routing network. Such routing is inspired by FPGAs. The embedding gets greatly simplified, since the routing network can quickly establish connections on demand. Time-consuming iterations during the embedding can be reduced as discussed in the related work [167], [170], [156]. Jagielski et al. also proposed the usage of routing in OIM networks. Their topology is intended for SAT problems and they used the SATLIB benchmark set to evaluate the needed routing network size. A 1000 oscillator system with 9 tracks would be sufficient for the uf50 problems from SATLIB [128]. So, this is another confirmation of the benefits of such a routable structure. The disadvantage is the increased area and power requirement compared to fixed local connections. The routing network including configurable switches needs area and additional power is required to transmit a signal through it. Therefore, routable connections can be seen as a trade-off between all-to-all networks and sparse networks. For sparse optimization problems, they can simplify the embedding process. The flexibility allows a limited number of ‘all-to-all like’ connections at a slightly reduced area efficiency of sparse OIM systems. However, this approach is not well-suitable for dense optimization problems due to the area overhead by the channels.

### 5.1.4. Oscillator Node Overview

The actual physical implementation contains more than the components of the simplified structure already shown in Figure 5.2. Additional circuits such as biasing, calibration, buffering, and the routing network are needed for the operation as well. All building blocks of the circuit implementation are shown in Figure 5.3. The central element is the fully-differential oscillator. Its output buffer drives the signal for the distribution to other nodes and the differential-to-single-ended converter (D2S) provides a single-ended oscillator signal to the digital control block. The coupler circuits enable the interaction between oscillators, where the phase





---

there are less advantages for the analog circuitry. Consequently, the technology choice mostly affects the area of the digital design and barely influences the performance as long as the technology is suitable for analog implementations. As discussed later in Section 5.8.1, the digital circuitry accounts for approximately 25% of the OIM network area. Assuming that the area of the digital circuit scales quadratically with the technology node, while the area of the analog core stays identical, a 65 nm implementation would be 109.7% larger and a 16 nm implementation just 16.8% smaller. With more advanced nodes, the contribution of the digital part reduces, which diminishes the overall benefit.

From a practical aspect, the chosen 28 nm technology offers frequent and cost-effective multi-project wafer (MPW) runs for prototyping. From an implementation point of view, the more complex design rules of modern FinFET technologies increase the design and layout effort. Consequently, more advanced nodes offer limited benefits, while the area-compactness favors a modern technology like the 28 nm compared to older 65 nm or 130 nm technologies. Overall, the scientific knowledge gain is hardly affected by the exact choice of a node. Even when scaling up the system size for commercial production, a 28 nm technology might be a good economic trade-off.

## 5.2. Oscillator

The oscillator is the most prominent element as it is already giving its name to the OIMs. It is the central element representing the discrete states  $\sigma_i$  of the Ising model and its behavior is essential for the computation.

### 5.2.1. Oscillator Type

There are a variety of electrical oscillator architectures available, such as LC oscillators, ring oscillators, relaxation oscillators, or crystal oscillators. Because thousands of oscillators should be implemented, the oscillator itself should be as compact as possible. The first OIM implementation on a PCB level was conducted by Wang et al. using discrete LC oscillators [4]. For such a massive parallel integrated implementation, the size of the inductor makes the LC oscillator unsuitable. Even with special magnetic materials, on-chip inductance densities of up to  $1700 \text{ nH mm}^{-2}$  have been reported. However, this would yield just a small number of oscillators per square millimeter. Crystal oscillators are even more problematic due to the required external piezoelectric crystal. Hence, only ring and relaxation oscillators need to be considered, which have already been used for integrated implementations of OIMs<sup>†3</sup>.

Relaxation oscillators typically operate at lower frequencies compared to ring oscillators. Additionally, their circuits often use passive components such as resistors<sup>†4</sup> or capacitors. Hence, this work focuses on ring oscillators as they provide a much higher frequency and thus faster computation, while being very compact. The main drawback of an OIM implementation using ring oscillators is their sensitivity to coupling inputs, which varies considerably throughout the oscillation cycle. They tend to be more sensitive at their transitions compared to the remaining cycle, as already discussed in Section 4.2.1. In comparison, relaxation oscillators typically have a much more uniform sensitivity across a whole oscillation period due to their integrating element.

---

<sup>†3</sup>An overview of CMOS integrated OIMs is provided in Chapter 7, Table 7.2.

<sup>†4</sup>Implementations using switched capacitors have been proposed to replace resistors [82]. Since the clock of such a switched capacitor implementation needs to be considerably faster than the oscillator's operating frequency this approach is limited to slow oscillators.

---

## 5.2.2. Single-ended and Differential Signaling

Ring oscillators are commonly implemented with single-ended, pseudo-differential, and differential signaling. Although a single-ended implementation is advantageous regarding area, power, and design complexity, a differential oscillator design is chosen. The differential oscillator has a higher rejection of undesired common mode signals, which would otherwise superimpose the actual signal in single-ended implementations. For example, such common-mode components can come from the power supply or capacitive coupling from nearby physical traces. Although the power supply grid is designed to keep the voltage drop across the chip low, local supply voltage variations are inevitable. Since the frequency of differential designs is determined by the bias current of the oscillator, the frequency is well controlled despite an imperfect power supply. Theoretically, the oscillators could even undesirably couple through disturbances of the power supply, which would alter the optimization problem that is solved by the network. Additionally, a symmetric, differential design will typically have less variations of the duty cycle compared to a single-ended design.

Unfortunately, it is difficult to quantify or even compare the benefits of differential and single-ended systems. While oscillators can be coupled in simulation via the power supply, accurate models for the parasitic coupling and power supply distribution in OIMs are needed. A straightforward post-layout simulation is computationally too expensive. Hence, future research can consider whether differential designs are really necessary. Competing designs usually use single-ended implementations instead. However, the mentioned parasitic coupling or power supply disturbances could potentially be one factor for lower solution accuracy.

## 5.2.3. Frequency

A crucial system-level design criterion is the oscillator frequency. It determines the computing speed but also affects various other aspects of the design. The SHIL injection, phase measurement, and on-chip clock distribution must be designed accordingly. For both designs, a frequency of roughly 100 MHz was chosen. It is a trade-off between computing speed and design complexity, where this frequency can be comfortably reached in the 28 nm technology. Inevitable delays when transmitting the oscillator signals within their proximity are negligible compared to the oscillation period<sup>†5</sup>. Additionally, the skew requirements for the SHIL and system clock are also relaxed. The main benefit of a higher frequency is the reduced time per computation. However, the power consumption will typically increase as well. Assuming that the computation takes approximately the same amount of oscillation cycle to settle independent of the frequency, the needed energy per computation will be less affected by the choice of the frequency. When considering the full-stack computing cycle, starting from the problem in the Ising form to the obtained solution, other tasks, such as the embedding and data transmission to the OIM chip will have a non-negligible share of the overall computation time. Although OIM designs with up to 1 GHz have been presented [85], the targeted 100 MHz could be considered sufficiently quick.

## 5.2.4. Number of Stages

When implementing a ring oscillator, the first degree of freedom is its number of stages. While single-ended implementations require an odd number of stages, differential designs can be implemented with odd and even number of stages. By swapping the positive and negative signals in differential designs, an additional phase shift of  $\pi$  can be easily introduced. The frequency of the oscillator is determined by the delay of all elements in the ring. So, oscillators with different number of stages could have the same frequency.

---

<sup>†5</sup>More details about this delay are provided in Section 5.5.

---

Therefore, the targeted frequency and the number of stages are considered separately. The stage delay can be kept very small in the used 28 nm technology node, as ring oscillators with frequencies in the multiple Gigahertz are easily possible. Consequently, the targeted frequency of 100 MHz could even be reached with a high number of stages, which allows selecting the number of stages almost independently from the frequency. At the same frequency, a higher number of stages needs a lower delay per stage.

Different numbers of stages and various delay cell topologies were empirically studied for their sensitivity. When a delay cell drives a constant high or low output, the oscillator is much less sensitive to coupling inputs. With fewer stages, the individual cells need a larger delay. This affects the rise and fall times, so the transition takes more time relative to an oscillation period. Hence, the number of stages should be kept as low as possible to achieve a more uniform sensitivity. However, some simulations of a tested 3-stage design showed that the oscillations were not sustained reliably. A few cases were noticed where the oscillation was stopped by the coupling<sup>†6</sup>. Hence, a 4-stage design was chosen, which did not exhibit such a behavior. However, this analysis should be done with the specific circuits and might not generally apply to any ring oscillator design.

### 5.2.5. Oscillator Design

Figure 5.4 shows the 4-stage differential oscillator design. The output is buffered so that the drive+ and drive- signals can be distributed to neighboring nodes without affecting the capacitive load seen by the oscillator. Because any capacitive load affects the frequency, this load should be kept as constant as possible. Since the coupling circuits will be individually enabled based on the optimization problem, the buffer is a part of the load mitigation strategy of the coupler, as later discussed in Section 5.3. The buffering generally enables to easily change and scale up the connectivity, since the oscillator does not need a redesign to consider any altered capacitive loads. The coupler circuits inject a current into the couple+ and couple- terminals to modify the oscillator phases. The SHIL is applied to the same stage as the coupler circuits. Because this 4<sup>th</sup> delay cell experiences a higher load than the remaining cells, its bias current  $I_{stage}$  is increased compared to the other 3 cells. When the oscillator is disabled, the bias current is switched off and a pull-up transistor is activated (gray transistor).

The schematic of a delay cell stage is shown in Figure 5.4b for the 1<sup>st</sup> generation Firefly and in Figure 5.4c for the 2<sup>nd</sup> generation Glowworm. The core topology is identical between both iterations. It consists of an N-channel metal-oxide-semiconductor field effect transistor (NMOS) differential pair M1/M2, a cross-coupled P-channel metal-oxide-semiconductor field effect transistor (PMOS) load M3/M4, and a diode-connected PMOS load M5/M6. The cross-coupled pair M3/M4 helps to achieve a more uniform sensitivity for the oscillator, while the diode-connected load M5/M6 stabilizes the amplitude. The bias current  $I_{stage}$ , which determines the oscillation frequency, is provided via the current mirror M7/M8. The main difference between both generations is the calibration scheme. For the Firefly implementation, each stage bias current  $I_{stage}$  can be adapted with a 3-bit resolution by the transistors M8T (“T” indicates transistors for frequency tuning). Since a finer step size is desired for the Glowworm design, a fine and coarse tuning concept is used instead. The coarse tuning is done by increasing the bias current  $I_{stage}$  of all stages with a 3-bit word. The fine-tuning changes the capacitive load of each delay cell individually by transistors M9T/M10T. More details about the calibration are discussed in the following section. Another improvement are the transistor sizes, which have been optimized for lower power consumption, less variation and compact layout. The second design achieves a 58% area reduction while the uncalibrated frequency variation is lowered by 18%.

---

<sup>†6</sup>This evaluation was done with ideal resistors as coupling elements. With AC coupling of the injected current, it should be impossible to stop the oscillator completely. However, the oscillator could still be severely distorted.

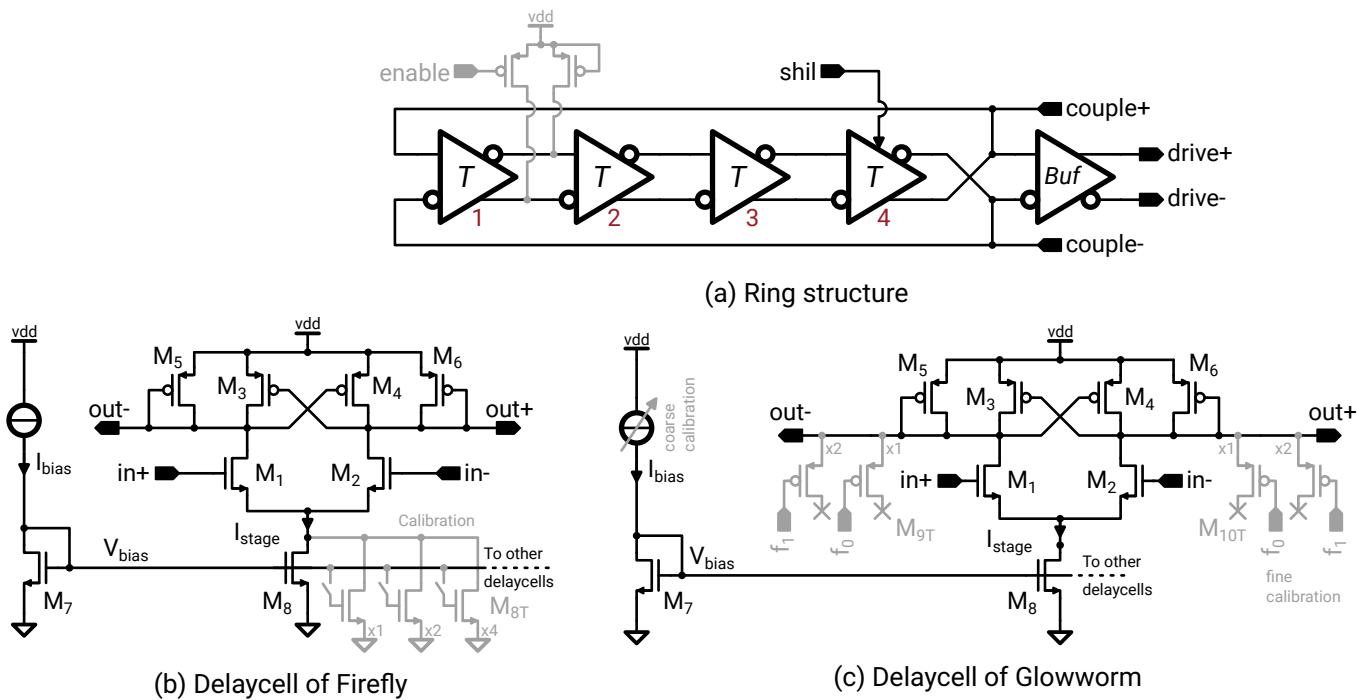


Figure 5.4.: Schematic of the 4-stage differential ring oscillator design. (a) The differential ring with an additional output buffer, where the stage number is annotated in red. The enable signal controls the bias of all delay cells and brings the ring in a predefined state. (b) Schematic of the delay cell of the Firefly design. (c) Schematic of the delay cell of the Glowworm design. The frequency calibration scheme of the delay cells is indicated in gray.

### 5.2.6. Frequency Calibration

The oscillator will exhibit individual frequency variations caused by random manufacturing mismatch. Figure 5.5 illustrates the impact on the coupling of a pair of oscillators. The frequency difference must be overcome

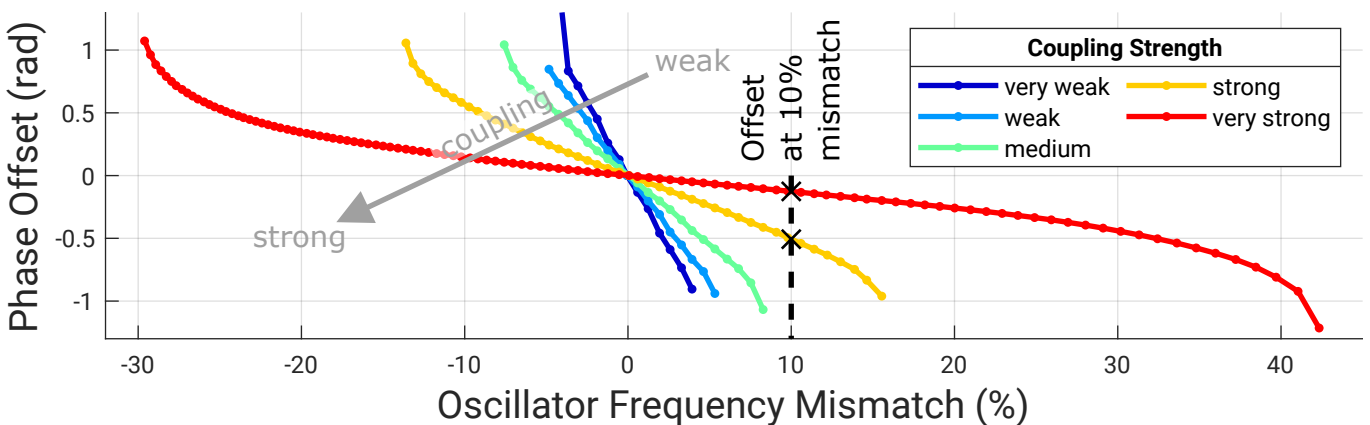


Figure 5.5.: Impact of the mismatch introduced phase offset when coupled, where two oscillators can only couple at the shown points. An exemplary mismatch of 10% is annotated (dashed line), which can only couple at the two highest strengths.

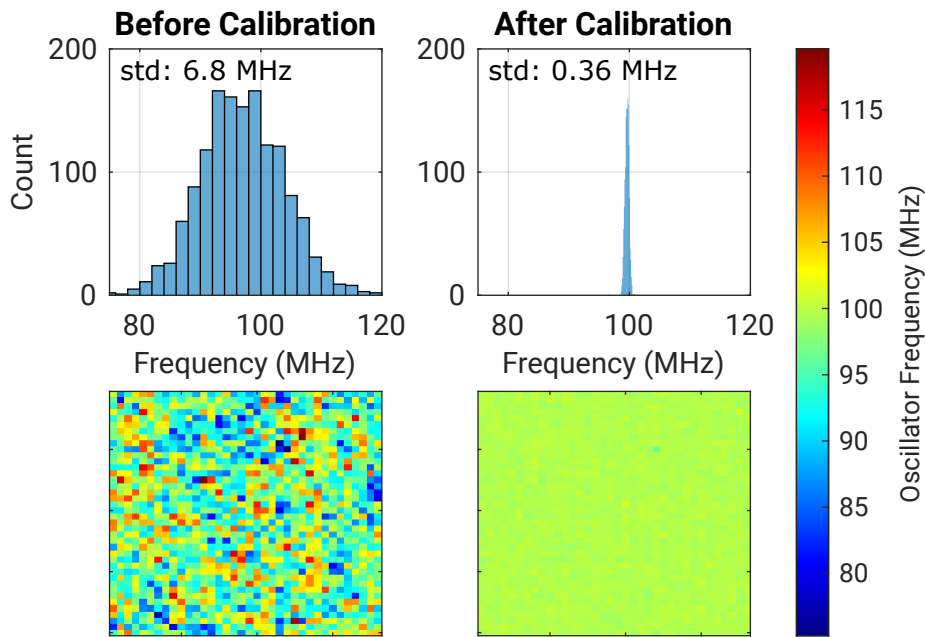


Figure 5.6.: Comparison of the frequencies before and after calibration. Without calibrations (left side), the frequencies vary by approximately  $\pm 23\%$  around the mean frequency of all oscillators. After calibration, the standard deviation of the oscillators is reduced from 6.8 MHz to just 0.36 MHz.

when coupled, which requires an energy transfer from the faster to the slower oscillator and thus introduces a phase offset. Coupling is only possible within a certain range, which depends on the coupling strength. At the weakest shown coupling, the frequency difference must be less than  $\pm 5\%$ , while the highest strength can tolerate differences of up to 30%. An experimental evaluation of the impact of frequency variations on the OIM computing itself is available in Section 7.3.6.

Therefore, a calibration is used to compensate for random manufacturing mismatch and achieve close matching of the individual oscillators. Figure 5.6 shows the measured frequencies of all 1440 oscillators on the same die. Without calibration, the individual frequencies vary by  $\pm 23\%$ . The oscillator location does not have a noteworthy impact on its actual frequency. The design goal is to achieve a frequency mismatch of less than 1% of the oscillators operating frequency as this introduces just a negligible accuracy loss for the computation as later experimentally confirmed. Consequently, the discussed fine adjustment is needed, which should be monotonic to simplify finding the optimal calibration code. The tuning range should exceed the random variations, obtained by Monte Carlo simulation. The average frequency, which is affected by process, voltage, temperature (PVT) variations, does not need to be compensated for by the calibration. Instead, the global bias voltage for the oscillators, which scales all  $I_{stage}$  equally, is used to compensate for any influence affecting all oscillators. After successful calibration, the oscillators operate within a narrow range of frequencies.

## Frequency Tuning

For the sake of completeness, the calibration principle should be mentioned. It ensures a close matching of frequencies, but the implementation differs between both fabricated designs.

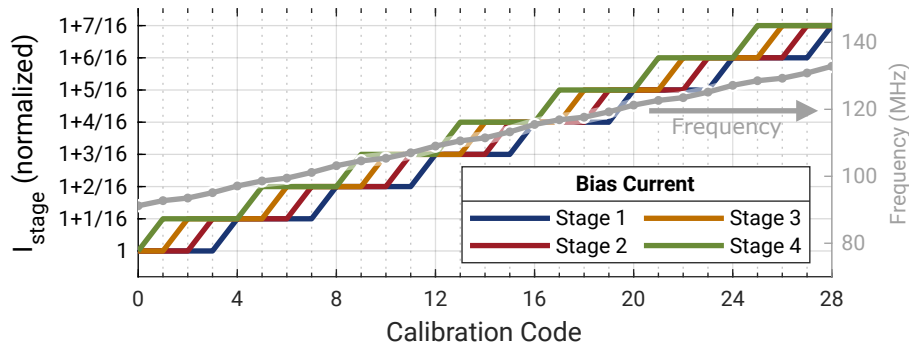


Figure 5.7.: Circular thermometer coded tuning scheme. The resulting measured average frequency is shown in gray. This scheme achieves a high monotonicity, where 99.94% of all steps are increasing.

**1<sup>st</sup> Gen Firefly** The Firefly design increases the bias current of the four delay cells individually. This approach allows for a finer control of the frequency compared to changing the bias for all stages simultaneously with the same step size. The current of each cell can be controlled via a 3-bit code and increases the current in step size of  $\frac{1}{16}$  (M8T in Figure 5.4b). A thermometer code style in a circular pattern is used, as shown in Figure 5.7. Each code increases the bias current of one stage while keeping the differences between the stages as small as possible. The resulting increase in frequency is non-linear because of the higher loading of stage 4.

**2<sup>nd</sup> Gen Glowworm** The calibration scheme was changed for the Glowworm design to achieve a wider tuning range with finer step sizes. This should account for rare outliers, which showed more variation than predicted by Monte Carlo simulations. To maintain monotonic behavior over the increased tuning range, a coarse and fine tuning scheme is employed. A coarse tuning adjusts the bias current of all oscillator stages with a resolution of 3-bit, providing 8 coarse steps. A secondary fine tuning changes the capacitive load of the delay cells (see M9T and M10T in Figure 5.4c). The small change in capacitance from turning the almost minimum-size transistor on or off is enough to change the frequency. A similar pattern as the calibration of the Firefly design adjusts the 2-bit fine codes.

The changes compared to the predecessor generation was necessitated for matching reasons. The wider tuning range and finer step size quadrupled the number of steps, so that monotonicity with the previous approach could not be achieved. Changing the bias current requires a current mirror with multiple parallel transistors, which can be enabled/disabled (see M8T in Figure 5.4b). For a very fine current adjustment, the W/L ratio of the transistor implementing the smallest step needs to be small in comparison to the main current mirror transistor. Because the minimum width of a transistor is restricted, the tuning step size is limited as well. The capacitive load approach provides a finer step size. Only the change in capacitance between the enabled/disabled transistor matters, while the unavoidable static load (e.g. parasitics of additional wires) introduces a steady capacitance. Since the smallest possible transistor (with minimum length and width) can be used with its drain left unconnected, very small capacitance changes can be implemented. Essentially, the capacitance change is caused by the saturated channel and the drain contact.

**Calibration Discussion** The digital calibration algorithm starts at the slowest calibration code and increments it step-by-step until the target frequency is reached or exceeded. The detailed implementation including a frequency map showing the individual oscillators is available in the Appendix A.1. The design has sufficient tuning range, so that there are little outliers present after calibration. However, the on-chip calibration algorithm yields a slightly worse calibration than ideally possible with the oscillators. Figure 5.8 compares

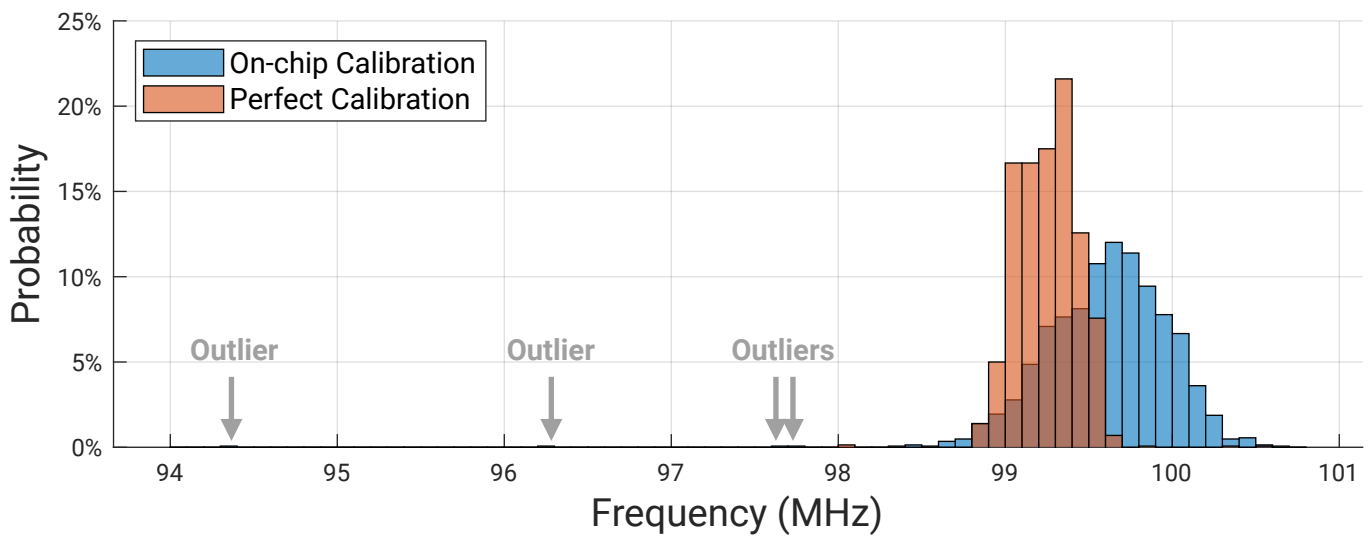


Figure 5.8.: Comparison of the discussed on-chip calibration with the perfect calibration codes externally computed. These were obtained by recording the frequency for each oscillator and code and then calculating the codes, which minimize the standard deviation.

the on-chip calibration with the optimum calibration codes, which shows that the standard deviation of the on-chip calibration is twice as large. Additionally, there are 4 outliers out of 1440 oscillators, which are caused by suboptimal calibration codes. However, the remaining frequency mismatch does not have a noteworthy negative impact on the OIM computing accuracy as analyzed in Section 7.3.6.

### 5.2.7. Output Buffer

An output buffer for the oscillator is added for loading reasons. It should provide a buffered output voltage to drive other coupling circuits without affecting the oscillator. A low delay respectively phase shift is desirable for the phase based computing. The delay of the buffer is a part of the feedback loop of two coupled oscillators. Therefore, the received phase by other oscillators is perceived phase shifted to the actual oscillator phase. This issue is very similar to the later discussed delay of the routing system in Section 5.5.

The design of the output buffer is kept simple and discussed for the sake of completeness. The buffer is implemented as two source-followers as shown in Figure 5.9a. A current mirror provides the biasing of both branches as shown in Figure 5.9b. The buffering causes a voltage drop of the common mode by one gate-to-source voltage, which is kept low by using ultra-low threshold transistors. Due to the AC coupling of the coupler circuits (discussed in Section 5.3.2), the common mode voltage drop is uncritical. The main drawback of the approach is the high current consumption of the source-follower design, where most of the power is wasted for biasing. As long as the biasing current is sufficient for buffering, the computed solution is insensitive to variations of the biasing current.

In the future, enhanced buffer topologies could be investigated to reduce the power consumption. However, the impact of a suitable output buffer on the actual computation is expected to be minimal. Since the waveform should be sufficiently buffered without introducing significant delay, there should not be any substantial differences for the rest of the system. Since the buffer is already compact and contributes only a small fraction to the overall area, not much savings can be expected. Hence, more sophisticated buffer designs do not significantly contribute to the scientific findings of this work.

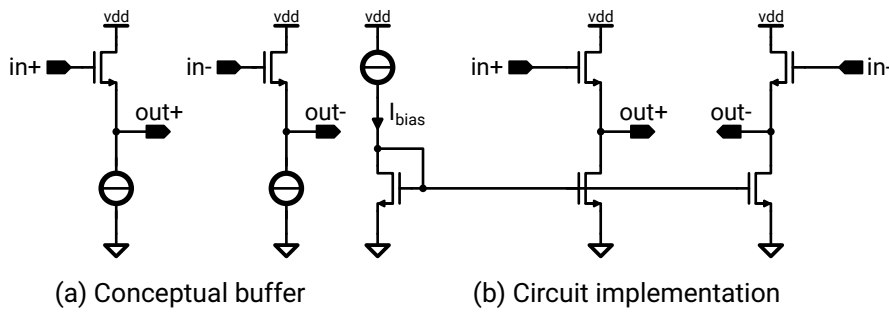


Figure 5.9.: Output buffer of the oscillator based on a simple source follower architecture. (a) The two source followers for the positive and negative differential signal. (b) The actual circuit implementation with a current mirror.

### 5.2.8. Differential-to-Single Ended Conversion

The output of the oscillator is a differential signal, which carries the phase information of the oscillator. As part of the computation, the phases of all oscillators need to be measured and transmitted off-chip as a solution to the optimization problem. For the phase measurement, a synthesizable all-digital PDC is used, as described in Section 5.7.2. Thus, a differential-to-single-ended converter (D2S) is necessary. The phase information needs to be accurately maintained. Additionally, the converted signal is also used for the routing system. Circuit-wise, a time-continuous comparator is used. A positive zero crossing of the differential waveform should cause a transition from a '0' to a '1' and a negative zero crossing should cause the opposite transition. The delay for the conversion is irrelevant if it is identical for all oscillators. However, unavoidable manufacturing mismatch will cause random deviations, which puts a constraint on the design. In the case of the converter's usage in the routing network, the delay of the conversion is part of the feedback loop and delay compensation. While it is not critical, the delay should be small within the compensation budget discussed later in Section 5.5.3.

The schematic is shown in Figure 5.10, which is a common architecture for comparators. The input is converted into currents by the differential pair M1/M2. M3/M4 and M7/M8 mirror the current to the output for a logic high. M5/M6 as counterpart provide a logic low. Standard CMOS inverters buffer the output. To save power, the D2S can only be activated when its output is needed. Since the biasing transistor M9 is shared with the output buffer, only the gate voltage of M10 is disabled.

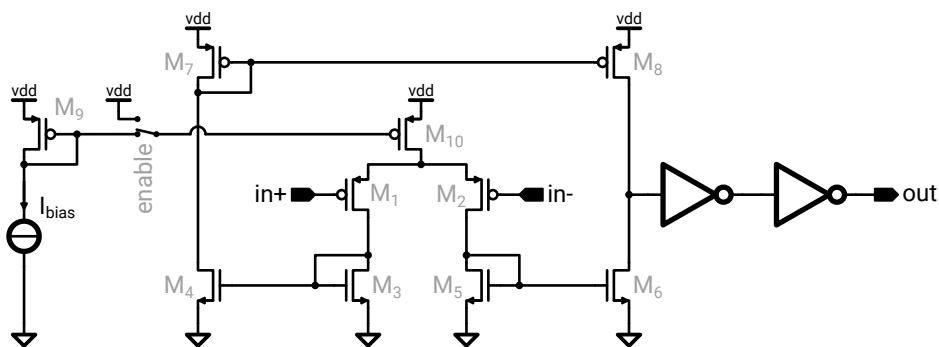


Figure 5.10.: Schematic of the differential-to-single-ended converter (D2S) circuit responsible for converting the differential oscillator signal to a single-ended, phase-maintaining signal.



## 5.3. Coupler

The coupling behavior is not only shaped by the oscillator but is largely determined by the coupler. It creates the interaction by determining the current, which changes the phase of the oscillators. Passive implementations are discussed first, as they motivate the proposed active implementation. Furthermore, the introduced phase offsets and the disabled state of the couplers are considered.

### 5.3.1. Passive Couplers

Simple passive elements can be used as couplers. Resistors and capacitors have already been used for OIMs [4], [82]. Figure 5.11 shows the conceptual resistive coupling of two oscillators A and B. Their voltages,  $v_A(t)$  and  $v_B(t)$ , cause a current exchange of  $I_{AB} = \frac{v_A(t) - v_B(t)}{R}$  between the oscillators. The current slows down the leading oscillator and speeds up the lagging oscillator so that their phases align in-phase. When both phases are identical, no current flows.

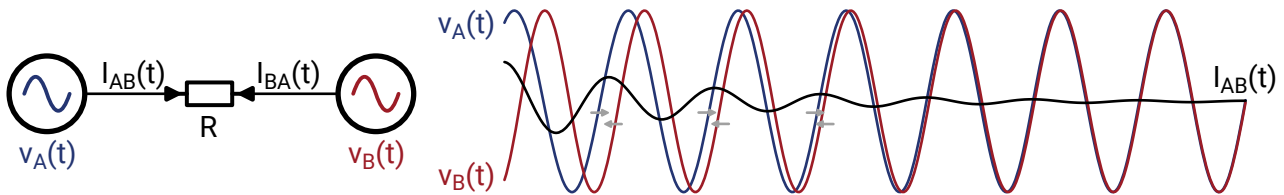


Figure 5.11.: Principle of resistively coupled oscillators.

Despite being excellent for coupling, this work wants to avoid passive coupler circuits. They exhibit two major disadvantages. To allow an adequate current, resistances of 100 k $\Omega$  and more would be needed. When implementing weak coupling, which would be necessary for a high resolution of the  $J_{ij}$  coefficients, resistances over 1 M $\Omega$  would be likely required. From an area point of view, this is highly undesirable. Additionally, implementing multiple weights of a coupling connection would require tuneable passive values. This could be realized by switching multiple components in parallel or series. Hence, implementing high-resolution Ising machines with resistors is not favorable due to the high area usage. Capacitors would be favorable area-wise since weak coupling needs small capacitances, which are compact to implement. However, small capacitors are susceptible to parasitic capacitances and might suffer from high relative manufacturing variation in their value, potentially changing the effective capacitance. Common for both approaches is that its resistance/capacitance is defined by design and can not be feasibly changed after manufacturing. Especially for research purposes, where the impact of the coupling strength on OIM computing is of interest, such passive approaches are not favorable.

### 5.3.2. Active Couplers

As discussed before, a resistive behavior is well-suitable for coupling, but an implementation using resistors is unfavorable. Hence, an active design should avoid these disadvantages of passive components and deliver equal or better OIM computing performance. The focus of the active coupler design can be summarized as follows. It should be as area-efficient and compact as possible. The weights should be set via a provided current or voltage. This allows the usage of a DAC to configure the coupler with the desired weight. To increase the robustness against manufacturing mismatch, the design should rely on matching between transistors. So, it should not be based on absolute properties such as the on-resistance of a transistor.

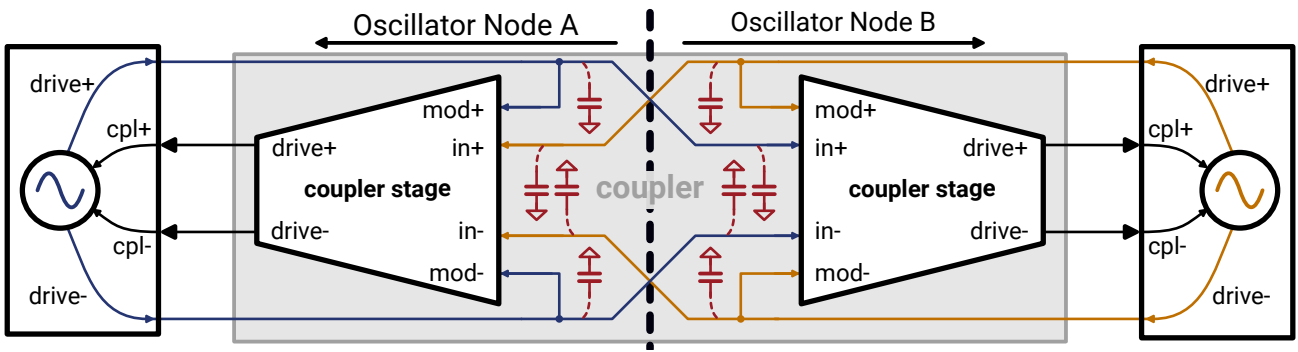


Figure 5.12.: Principle of the back-to-back connected coupler stages. Each stage is physically located in front of the connection to the sensitive oscillator cpl port to keep it as short as possible. The buffered drive signal is distributed across the oscillator nodes with parasitic capacitances indicated in red. Each coupler stage determines the injected current based on the injected and injecting oscillator.

Active circuits are typically unidirectional by having a dedicated input and output port. However, bidirectional coupling between oscillators is required for the computation. Hence, two identical circuits, referred to as ‘stages’ in the following, are connected back-to-back, as illustrated in Figure 5.12. Together, these two coupler stages form the actual coupler and follow the buffered oscillator design approach. The injecting stage is physically located close to the oscillator to minimize any parasitic capacitive load on the sensitive cpl port. Only the buffered oscillator signals are transmitted between oscillator nodes, which determine the injected current. Both stages are supplied by a dual-output DAC, which provides the same current to both stages. In principle, this topology supports unidirectional coupling when just one stage is enabled.

The coupler stage design of both fabricated chips follows the same principles. The goal is to roughly approximate resistive coupling. However, the current injection can be more ‘aggressive’, so that it causes a strong phase change already at low phase differences. The maximum injected current should be limited to avoid too strong distortions of the oscillator. Therefore, the injected current roughly approximates a trapezoidal shape, which worked well in simulations based on the Kuramoto model<sup>†7</sup>. When the oscillators are in the desired phase no current should be injected. For small phase differences, the injected current should rapidly increase and then saturate at a high level for medium and strong phase differences.

The core of the active coupler stage is based on two parallel differential stages, as shown in Figure 5.13. The topology will be referred to as a double differential stage in the following. The coupling strength is defined by  $I_{bias}$ , which gets mirrored by M5/M6/M7. The injecting current  $I_{inj} = I_{main} - I_{mod}$  is the subtraction of  $I_{main}$  by the input stage M1/M2 driving, which is driven by the interacting oscillator, and  $I_{mod}$  by the modulation stage M3/M4, which is driven by the injected oscillator. A differential stage has an approximately linear output current at very small differential input voltages. At larger differential voltages, this approximation worsens and the current will start to saturate. Eventually, one branch carries the full bias

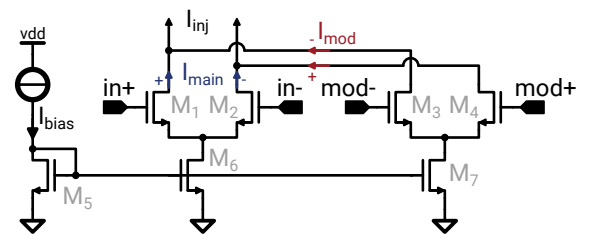


Figure 5.13.: General principle of the double differential stage coupler approach.

<sup>†7</sup>The coupling behavior is determined by the injected current and the oscillator sensitivity. The Kuramoto model describes this originally as  $\sin(\phi_j - \phi_i)$ . See Section 4.1 for more details.

current  $I_{bias}$ . The maximum injected current is limited at  $2 \cdot I_{bias}$ , which will be applied at strong differential voltages. This plateau of the current prevents excessive distortions while allowing the injection of strong currents already at small phase differences. Although this does not resemble the behavior of resistive elements, it matches the desired ‘aggressive’ behavior for the OIM operation mentioned above. By providing  $I_{bias}$  with a current-steering DAC the output current of the coupler scales linearly and thus the strength of the coupling is approximately linear as well.

This idea and principle for the coupler were used for both chip generations Firefly and Glowworm. The second generation Glowworm has some improvement regarding area, mismatch, and the caused frequency shift. The schematic for the first generation is shown in Figure 5.14. The core consists of two double differential stages, where the first stage (M1-M4) implements in-phase coupling and the second stage (M5-M8) implements anti-phase coupling. These are almost identical, except that the polarity of the in terminal is swapped. The outputs of the coupler are AC coupled to prevent a static current flow into the oscillator. Consequently, M15/M16 and the differential pair M1-M8 need to be properly biased. Theoretically, the currents  $I_{cm,in-phase}$  and  $I_{cm,anti-phase}$ , mirrored by M17/M18 and flowing into M19, should ensure a proper operating point of M15/M16. However, Monte Carlo simulations showed poor robustness against manufacturing variations in

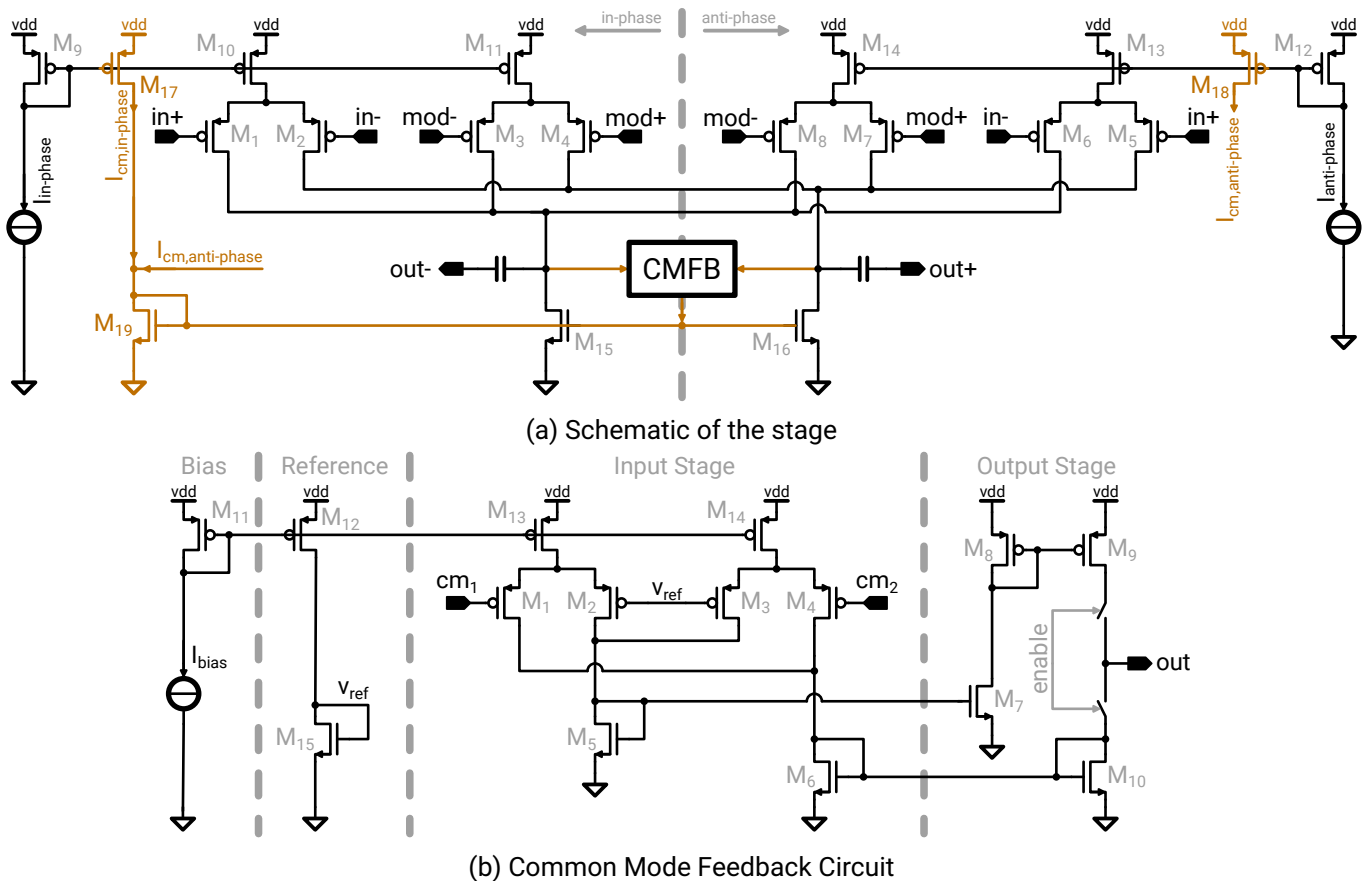


Figure 5.14.: Schematic of the coupler used in the Firefly design. (a) Circuit of a coupler stage, which consists of two branches for in-phase and anti-phase coupling via two independent biasing currents. The DC level of the active load is regulated by a common-mode feedback (CMFB). (b) The CMFB compensates for mismatch in the biasing of the active load and keeps a common mode level of roughly  $v_{ref}$ .

the coupler circuit due to highly varying DC levels. Having a very precise DC level is not critical and variations of  $\pm 100$  mV at 900 mV supply are sufficient. Thus, a dedicated CMFB, as shown in Figure 5.14b, is needed to compensate for mismatch related variations of the operating point. It sources or sinks just a small current to overcome the random variations. The CMFB is kept as simple as possible, trying to stabilize the dc-voltage to roughly  $v_{ref}$ . Compared to an ideal CMFB, the implementation loses a bit of computing accuracy according to simulations. This is presumably caused by a small remaining ripple on the common mode. However, the CMFB is sufficient to stabilize the common mode and prevent excessively high or low common modes.

Since a current steering DAC architecture is used, the overhead is low for providing two complementary currents for in-phase  $I_{in-phase}$  and anti-phase  $I_{anti-phase}$ , where  $I_{anti-phase} = I_{dac} - I_{in-phase}$ .  $I_{dac}$  denotes the constant bias current of the DAC. Having a branch for in-phase and one for anti-phase coupling leads to two options to configure coupling at reduced strengths. Either one branch is completely disabled while the other operates at very small currents, or both branches are simultaneously enabled, where one branch operates at a slightly higher current than the other. This difference in strength of both branches establishes the small preference in the desired direction. Because the total current of the coupler  $I_{in-phase} + I_{anti-phase} = I_{dac}$  is kept constant, it was hoped that this approach would lead to better performance at small strengths. However, worse accuracy when solving problems was observed experimentally compared to using just one stage at a low bias current. Consequently, the latter approach was dropped in the second generation to save area. Presumably, mismatch of the mirrored currents will superimpose small current differences between both branches, making them ineffective. Instead, the second generation uses just one branch and swaps the inputs to implement in-phase and anti-phase coupling.

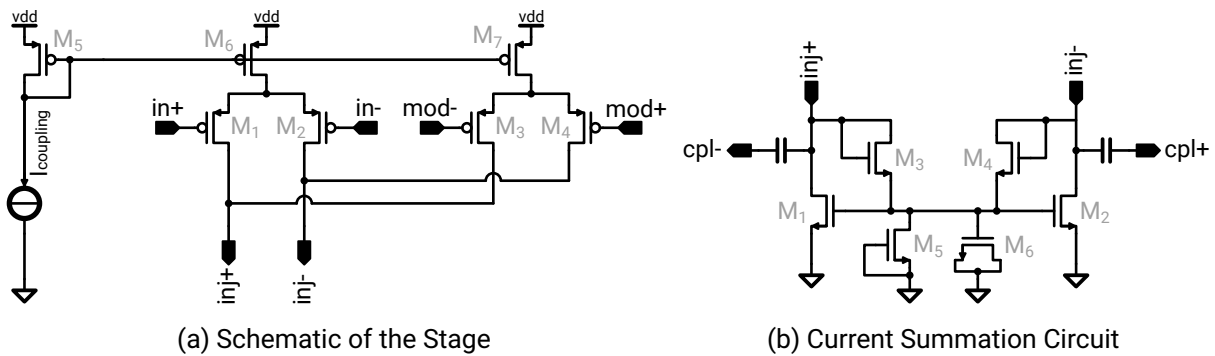


Figure 5.15.: Schematic of the coupler used in the Glowworm design. (a) Circuit of a coupler stage. To achieve in-phase and anti-phase coupling, the input of the differential stages is swapped. The current of the coupler stage is forwarded to the current summation circuit. (b) The current summation circuit is an active load implemented by transistors  $M_1$  and  $M_2$ . The remaining circuitry stabilizes the common mode level to approximately one threshold voltage.

The schematic of the improved stage of the 2<sup>nd</sup> generation is shown in Figure 5.15a. The core double differential stage with  $M_1/M_2$  and  $M_3/M_4$  is kept and similarly supplied by the current  $I_{coupling}$ , which gets mirrored by  $M_5-M_7$ . A simple transmission-gate input switch matrix can swap the connections to  $in+$  and  $in-$  to implement in and anti-phase, which saves considerable area compared to the previous design. Additionally, the active load with the CMFB is replaced by a common current summation circuit. It is a simple and robust self-biasing circuit as shown in Figure 5.15b, which combines the current of all coupler circuits of an oscillator.  $M_1/M_2$  act as an active load similar to  $M_{15}/M_{16}$  of the Firefly design. The very weak transistors  $M_3/M_4$  act as a peak sampler and limit the positive voltage. Since they charge the gate-source voltage of  $M_3/M_4$ , the amplitude is self-regulated and therefore the DC-operating stabilized. When charged higher, the transistors  $M_1/M_2$  reduce the DC level, so that they limit the peak voltage to a bit more than approximately two threshold

voltages. M6 acts as a capacitor stabilizing the gate-source voltage and M5 is used as a leaky element to slowly discharge the node. Since the oscillation frequency is sufficiently higher than the discharge by M5, the DC drain-source voltage is sufficiently stabilized. Additionally, the current summation is conducted before the AC coupling. This avoids that the current summing is distorted by insufficient coupling capacitances. Since it can be expected, that at least some coupling inputs will cancel each other out, the capacitance can be slightly reduced. Even in the unlikely case that no mutual cancellation happens, it would just saturate the maximum injected current. The drawback of this self-biasing approach is that it consumes a small amount of coupling current.

### 5.3.3. Disabled Mode

While the previous discussion considered the coupling operation, attention is also needed for the disabled state of the couplers. Since each edge weight  $J_{ij}$  of the Ising model is represented by a coupler, the physical network typically has more couplers than those actively used by an optimization problem. The unused couplers, when solving a given problem, should not influence the computation of the oscillators. In that disabled state, they should prevent any coupling through the remaining (parasitic) capacitances of the circuit. Hence, an input switch matrix as shown in Figure 5.16 is used. In the disabled state (Figure 5.16a), the outputs  $\tilde{in}$  are shorted to the supply and the input selection switches are left open. Hence, any current potentially coupling through the turned-off switches/transistor is shunted into the supply and thus effectively rejected. Additionally, the double differential pair (M1-M4, Figure 5.15a) of the coupler stage gets turned off. The capacitive load of the coupler seen by the oscillator barely changes. Thus, the resulting frequency shift of the oscillators is minimized. This is crucial because some oscillators might be connected to more active couplers than others but should still operate at approximately the same frequency for successful coupling. When enabled (Figure 5.16b), the switch matrix allows swapping of the differential signals to select between in- and anti-phase coupling.

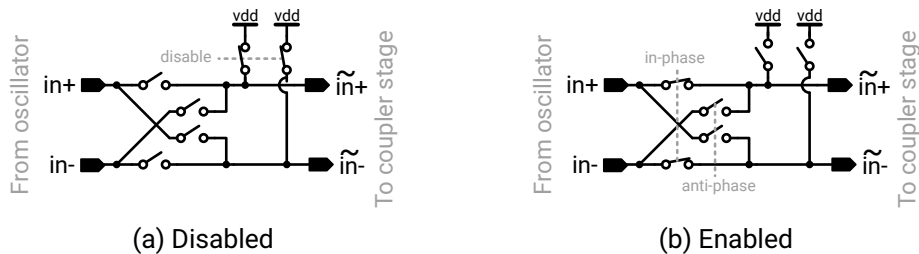


Figure 5.16.: Input switch matrix for the coupler stages used in the Glowworm design. By swapping the polarity, the coupler is changed between in- and anti-phase coupling. In the disabled state, the signals for the coupler are shunted to the supply.

### 5.3.4. Coupling Strength

So far, the coupling strength has been qualitatively discussed but not directly explained. Figure 5.17 illustrates the coupling strength based on a pair of oscillators. The settling becomes faster with increasing coupling strength. While it can take over 100 oscillator cycles for weak coupling, this reduces to just a few cycles at stronger interaction. However, these simulations are based on the nominal process corner with ideal resistors as coupling elements to emphasize the principle. Therefore, the experimentally obtained behavior of the coupling strength is provided in Figure 5.18 for comparison. The phases are extracted from measured oscillator

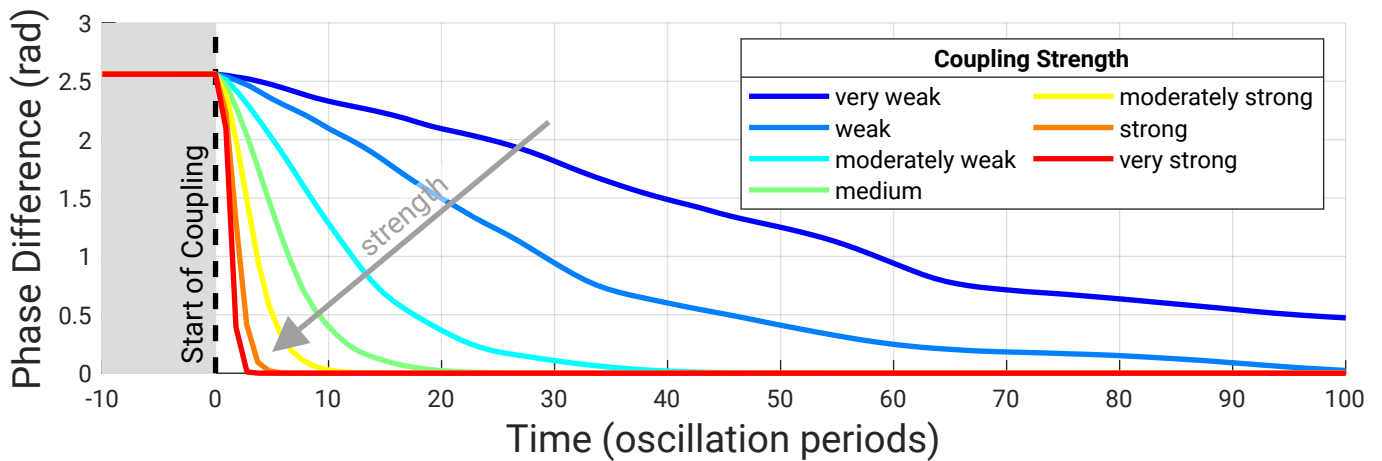


Figure 5.17.: Transient simulation to illustrate the impact of the coupling strength on the settling time. Two oscillators with an initial phase difference of  $145^\circ$  are coupled in-phase. Multiple strengths, from very weak to very strong, are exemplary shown.

waveforms. Due to the lack of precise control of the oscillator’s phase, there is a noteworthy difference in the initial phases. The settling time clearly reduces for higher coupling strengths. However, a phase offset visibly remains. This is caused by unavoidable frequency differences as already discussed in Section 5.2.6. At the

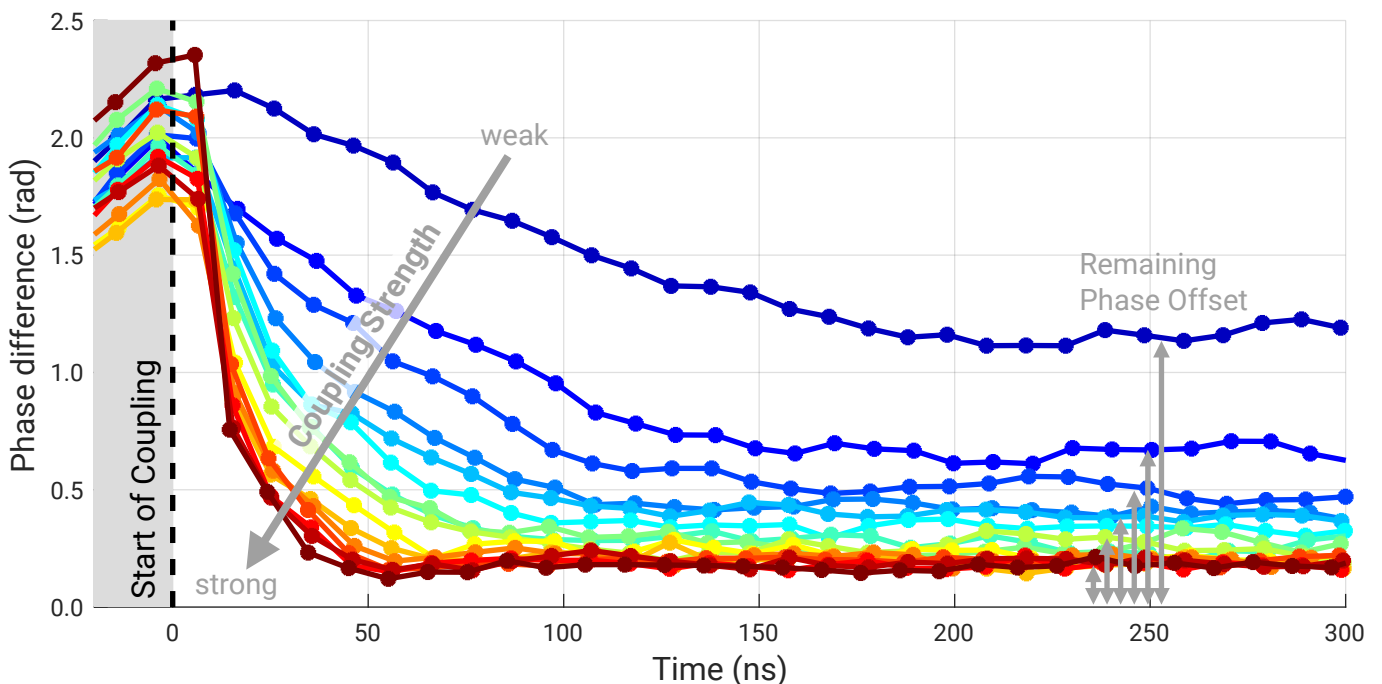


Figure 5.18.: Experimentally obtained impact of the coupling strength. The settling from a similar initial phase is provided for all 4-bit DAC settings of the coupler bias current, which are color-coded from weak (blue) to strong (red). These measured off-chip signals can have a phase offset compared to the actual internal oscillator due to the single-ended conversion and the signal transmission. The oscillators have a nominal period of 10 ns.

---

weakest DAC setting this is more than  $\frac{1}{3}\pi$  and reduces to less than  $\frac{1}{10}\pi$ . Hence, higher strengths reduce the sensitivity to frequency mismatch and could potentially enable faster computing. A detailed analysis of the coupling strength on the OIM computing is provided as part of the evaluation in Chapter 7.

### 5.3.5. Manufacturing Mismatch

The circuit and the transistors were optimized for robustness against manufacturing mismatch. It has two relevant and easy to measure consequences on the coupling behavior. Phase offsets are introduced, so the oscillators do not perfectly settle as desired in-phase (or anti-phase). Additionally, frequency shifts can occur. Then, the frequency of a coupled oscillator pair varies from its free-running frequency. This has a circuit topology-based component (e.g. a remaining non-zero current injection when settled) and another caused by random variations of the transistors. As a design goal, the coupler-induced phase offset should be below 0.15 rad. The largest contribution comes from transistors M6/M7 as well as M1/M2 and M3/M4 of the coupler stage (Figure 5.15a), which were therefore sized accordingly.

## 5.4. Sub-Harmonic Injection Locking

The sub-harmonic injection locking (SHIL) is used to discretize the naturally continuous oscillator phase angles, because COPs have discrete variables. In case of the Ising models, the variables  $\sigma_i$  have just two discrete states  $\sigma_i \in \{-1, +1\}$ . Hence, the natural continuous phases need to be forced into quasi-discrete states by injecting a clock signal with an approximate integer multiple of the oscillators' frequency. So, twice the frequency is injected to achieve two discrete states for the variables  $\sigma_i$ . A higher frequency ratio can be chosen for optimization problems with more discrete states, such as the Potts model. The theoretic fundamentals of injection locking and sub-harmonic injection locking have been discussed in Section 3.2. This section discusses the circuit implementation.

It is important to note that this section uses the term 'discrete phase' for the sake of simplicity. Technically, the phases are still continuous even when sub-harmonic injection locked. When coupled, every oscillator has its own preferred phase, depending on factors such as the natural frequency and coupling input from other nodes. When locked, the oscillator is forced towards the defined frequency and phase by the injecting clock. So, there is a fight between the phases caused by coupling and the injection itself. Although the SHIL is stronger, a small phase offset is introduced. Additionally, the oscillator is usually slowed down or sped up to match the sub-harmonic frequency. This can be thought of as an energy transfer from the injecting clock to the injected oscillator, which is only possible for non-identical phase angles and causes additional small phase offsets. In the scope of COPs with just two discrete states, such as the Ising model, the phase offsets are small compared to the phase difference for distinguishing the discrete state. Consequently, the quasi-discreteness of the phase under SHIL is only explicitly mentioned when needed and otherwise referred to as discrete phase. The quasi-discreteness of the phase angles is typically not relevant in practice when the SHIL has sufficient strength. The phase measurement has a resolution of a few bits. So, the quantization noise of the measurement tends to be much greater than the quasi discrete phase distribution.

As mentioned in Section 3.3, the SHIL is not just a straightforward rounding operation of the computation. It contributes to the actual computation. Although it is generally clear why SHIL is needed, the exact dynamic mechanisms by which the computed solutions are influenced are not. Hence, the circuit implementation should add flexibility for experimental analysis.

---

### 5.4.1. Parameters of the SHIL

To ensure successful locking, the injection strength must be sufficient to overcome any frequency difference between the oscillator and the corresponding sub-harmonic of the injection frequency. Hence, the oscillators can only lock within a certain range, which depends on the strength of the injection. Therefore, the strength of the SHIL injection can be chosen during the runtime via an external bias voltage. If the frequency is outside the locking range, the oscillator's frequency will not adapt to the SHIL and the intended phase discretization will not be possible. Although it might seem that a strength as high as possible could be chosen, it was experimentally observed that too high strengths negatively impact the solution accuracy. Presumably, the induced distortions of the oscillation cycle are the underlying cause. Therefore, the SHIL strength is adjustable and was experimentally determined, as later discussed in Chapter 7.

The SHIL frequency has been usually mentioned as roughly twice the oscillator frequency. In principle, any frequency within the locking range of all oscillators would be possible. The chosen frequency has an impact as experimentally experienced and discussed later in Chapter 7. Since the SHIL clock is generated off-chip, this has no further implications on the design. The clock tree should distribute the SHIL within an acceptable skew. Any skew of the SHIL clock will directly shift phases in the locked state. However, the reference clock for the phase measurement will also experience a shift. It is enough to keep the skew low enough, that the caused phase shift is below the measurement resolution.

The dynamic locking process can influence the outcome of the computation. When a sufficiently strong SHIL is applied, the oscillator phase differences are practically frozen and do not change anymore. However, using the SHIL leads to better results than rounding the continuous phase in the nearest discrete state, as previously mentioned in Section 3.3. Hence, the influence of the SHIL must be during the transition between the coupled state and the locked state. Thus, the SHIL can be gradually applied to enable a smooth transition between the purely coupled oscillator and the additional locking.

### 5.4.2. Injection Position

There are multiple possibilities to sub-harmonic injection lock an oscillator. In general, the injection can be applied to any of the four oscillator stages<sup>†8</sup>. Stage 4 is chosen because it is connected to the couplers. Hence, the stage's phase changes potentially non-periodically due to the interaction with other oscillators. Additionally, the buffer directly distributes its signal to the neighbors. So, the SHIL is directly combined with the coupling to provide the oscillator's response for the coupling connections. Furthermore, there is the option to either inject a differential or a single-ended signal. Since the SHIL clock is distributed in a single-ended form, a single-ended injection would be favorable to avoid a conversion into a differential signal. The principle is outlined in Figure 5.19. The tail current  $I_{stage}$  of the 4<sup>th</sup> delay cell is used, which modulates the delay of the stage. This causes an alignment of the transitions of the delay cell and leads to phase alignment with the injecting clock<sup>†9</sup>. While the single-ended injection into the tail current provides a very good locking range, the positioning at one of the 4 delay cells might need further investigations.

---

<sup>†8</sup>see Figure 5.4a

<sup>†9</sup>The injection is into the common element of a differential design. Even when injecting the same frequency as the oscillator, two discrete phase angles are caused due to the symmetry. At the injection point of  $I_{stage}$ , a positive and a negative transition appear identical. See Section 7.3.4 for experimental data on the locking.



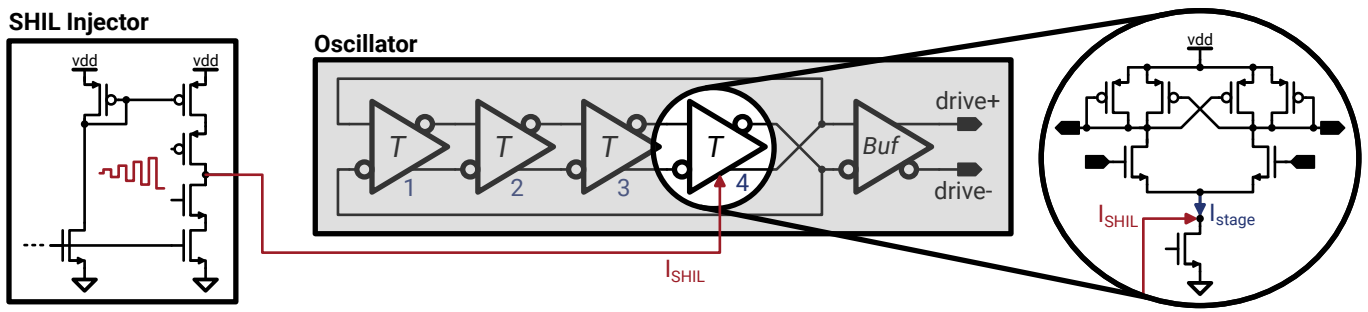


Figure 5.19.: Scheme of the single-ended injection into the tail current mirror of the 4<sup>th</sup> delay stage. The injected current  $I_{shil}$  modulates the stage bias current  $I_{stage}$ .

### 5.4.3. Injector Circuit

The simplified circuit implementation of the injector circuit is shown in Figure 5.20. The strength of the injection is controlled via a digital ramp generator. When enabled, the ramp generator will count upwards and increases the set value for the DAC step by step until its maximum is reached with 4 bit resolution. The strength of each step is incremented every two SHIL periods, which corresponds to roughly one oscillation period. The 1<sup>st</sup> generation Firefly has adjustable slopes for ramping up and down the strength, requiring additional configuration memory and logic. Based on experimental findings discussed in Section 7.3.4, just a fixed ramp-up in 16 steps and no gradual ramp-down was implemented in the 2<sup>nd</sup> generation Glowworm to save area. The current  $I_{bias,shil}$  is mirrored via M1 into M2 and M3, where a sourcing current is then provided via M4/M5. The SHIL clock itself, then switches M6/M7 on and off, which alternately injects the currents  $I_{up}$  and  $I_{down}$  into the oscillator based on the clock phase.

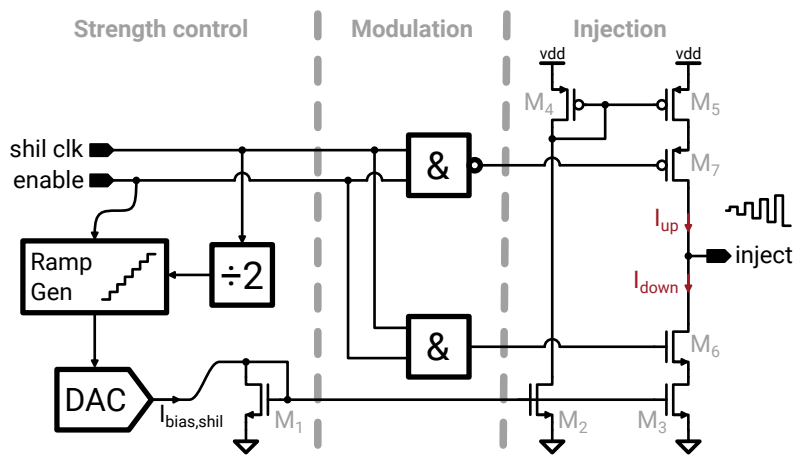


Figure 5.20.: Simplified principle of the SHIL injection. A digital ramp generator provides a 4-bit code to gradually increase and decrease the provided current by the DAC. This DAC provides a biasing current for the output stage, which is periodically sourced and sunk at the output stage.

## 5.5. Routing

The task of the routing network is to connect (almost) any oscillator with any other oscillator in the OIM chip. So, the embedding of the optimization problem into the coupled oscillator network can potentially be simplified by using these flexible connections. The routing forms a configurable interconnection network, which transmits and distributes the oscillator phase information. Since the coupling is a time-continuous process, the phase information needs to be transmitted time-continuously as well. These globally routed connections should behave similarly to local oscillator-to-oscillator connections since both implement the  $J_{ij}$  term of the Ising model. Thereby, a significant reduction in accuracy can be avoided. The concept of the routing was published in the associated paper of this work [154].

### 5.5.1. Signaling

The oscillators themselves are differential for signal integrity reasons, as discussed in Section 5.2. Distributing such a differential signal across the network could be more robust than a regular single-ended signal. Disturbances of the power supply and common mode noise can be effectively suppressed, which avoids faulty phase transmissions. However, the direct transmission of the differential oscillator signal has two major drawbacks. The signal will be unavoidably dampened when propagating through the routing network over long distances. Hence, multiple buffers need to be inserted in the path, which cause additional delay, respectively phase lag. Also, the analog waveform would not be preserved if the buffer gain does not exactly compensate for the losses. Additionally, the power consumption and area of such amplifiers is considerable. On top, twice the number of routing tracks and switches are needed for the differential transmission compared to a single-ended signal.

While a differential routing network is certainly possible, the immense area overhead is very problematic for OIMs. It needs considerably more area and power than a single-ended implementation. Additionally, it is much more efficient to transmit a digital signal than an analog signal across the routing network. A digital signal can be refreshed by simple inverters. A minimum-size inverter could be sufficient to transmit the signal from one switch block (SB) to the next. Maybe even the multiplexers of an SB could sufficiently refresh the digital signal. In contrast, an analog signal needs low on-resistance transmission gates and analog buffers for routing the signal.

### Signal Conversion

The oscillators and couplers, as discussed in Section 5.3, operate with analog differential signals. Hence, a conversion to and from the single-ended digital signal transmission is needed. The D2S converter of the OIM implementation already provides such a single-ended oscillator signal. Reusing this converter to obtain a single-ended oscillator signal saves additional area and power. When receiving the signal through the routing network, it needs to be converted back to a differential signal for coupling. The conversion circuit is shown in Figure 5.21, which provides just a pseudo-differential signal. It is based on a weak buffer and inverter in the branches with symmetric rise and fall times. Using minimum length transistors makes the skew between the buffered and inverted signal negligible. Additionally, the slope is reduced by the capacitive load of the actual transistors. However, selecting the slope is a trade-off. It should imitate the slope of the oscillator, so that the

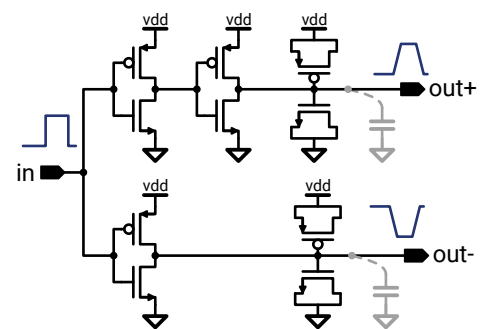


Figure 5.21.: Single-ended to differential converter.

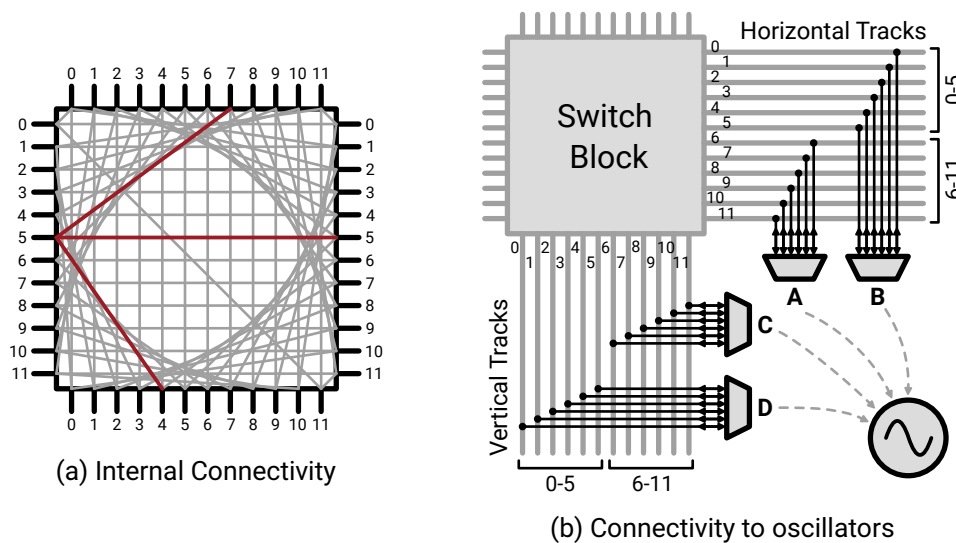


Figure 5.22.: The implemented connectivity within a switch block and access of the oscillators. (a) The internal connectivity is implemented as a 12-track Wilton style switch block. The connections of the left track with the number 5 are highlighted in red. (b) Simplified access scheme of the oscillator connections to the horizontal and vertical tracks. Just one of 4 oscillators, which are connected to a switch block, is shown for better visibility.

coupler behaves as similar as possible to a local connection directly driven by the oscillator. However, such a slow slope generates an unavoidable delay, which affects the coupling dynamics as discussed in the following.

### 5.5.2. Routing Network

The routing network is inspired by FPGAs and adapts established concepts from those. However, the network implementation of OIM differs. Instead of having unidirectional connections from an output of a logic cell to an input of another, bidirectional connections between the oscillators are required.

#### Switch Block

The SB structure is borrowed from FPGA designs, where this work implements a ‘Wilton’ style SB [129]. This ‘Wilton’ SB changes the track when turning from a horizontal to a vertical track or vice versa. The track number is then usually incremented or decremented by 1. So, a routing algorithm can change the tracks by inserting turns when needed to avoid blockages. The connectivity of the SB is shown in Figure 5.22a.

#### Oscillator connection

At each SB there are 4 connections each to all 4 adjacent oscillators. So, every switch block provides 16 routable connections. Each oscillator has two connections to the horizontal tracks and two to the vertical tracks as illustrated in Figure 5.22b. The tracks are split into a lower and an upper section so that each track is served. These tracks are accessed via a multiplexer and a demultiplexer.

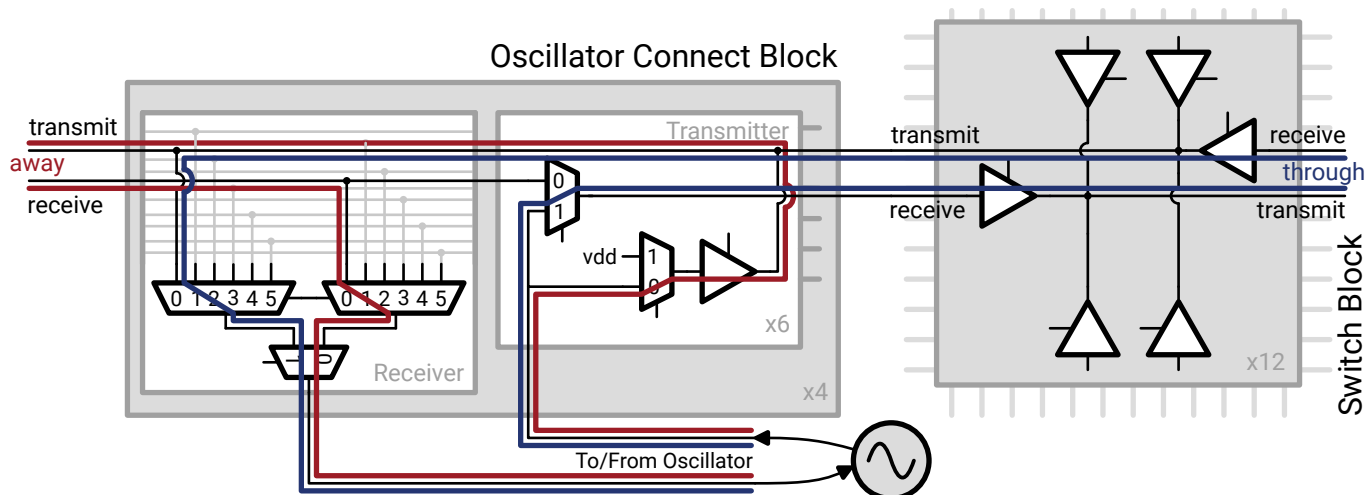


Figure 5.23.: Scheme of the routing block implementation. The switch block is implemented using tri-state buffers. The oscillator connections can either send signals through the switch block (blue) or in the opposite direction away from the switch block (red).

## Implementation

The switch block and oscillator connections are implemented together in a routing block, which additionally contains the required configuration memory. The implementation scheme is shown in Figure 5.23. The switch block implementation is based on tri-state buffers because they have a lower delay than equivalent multiplexers in this specific circuit topology and CMOS technology. However, this has the disadvantage that unused connections might be left floating. As there could be considerable capacitive coupling between neighboring tracks, floating nets should be avoided. Hence, the transmitting tri-state buffers of the oscillator connect block will drive the unused nets to a static voltage. Every oscillator can be either transmitted through or away from the SB. Driving away from the SB is simply done with a tri-state buffer. Driving through the SB uses an additional multiplexer in the signal path. Although the multiplexer adds a delay to the signal propagation path, it effectively prevents that the same net is concurrently driven by the transmitting oscillator and neighbor routing block. The receiver is just a multiplexer to select a track. The delays in the nominal corner and nominal operating conditions using post-layout simulations are as follows: Transmitter: 50 - 80 ps; Each SB: 57 - 66 ps; Receiver 180 - 260 ps. The span of delays accounts for differences caused by the parasitics of the physical design, having slight variations depending on the track, location and direction. Overall, a transmission from the left to the right side of the chip would take less than 1.6 ns.

The whole design was implemented in Verilog and synthesized with strong constraints. Cells such as the tri-state drivers and the 2:1 multiplexer were explicitly specified for the synthesis tools. This ensures that the bidirectional routing tracks in both directions and all physical channels are identical. Just the receiving multiplexer and the configuration memory were optimized by the tools. The place & route was done by Cadence Innovus.

## Routing Delay

The signal transmission through the routing network gets inevitably delayed. Since the coupling is based on the phase of oscillators, this delay is perceived as a phase lag. Figure 5.24 outlines this issue. The waveform of

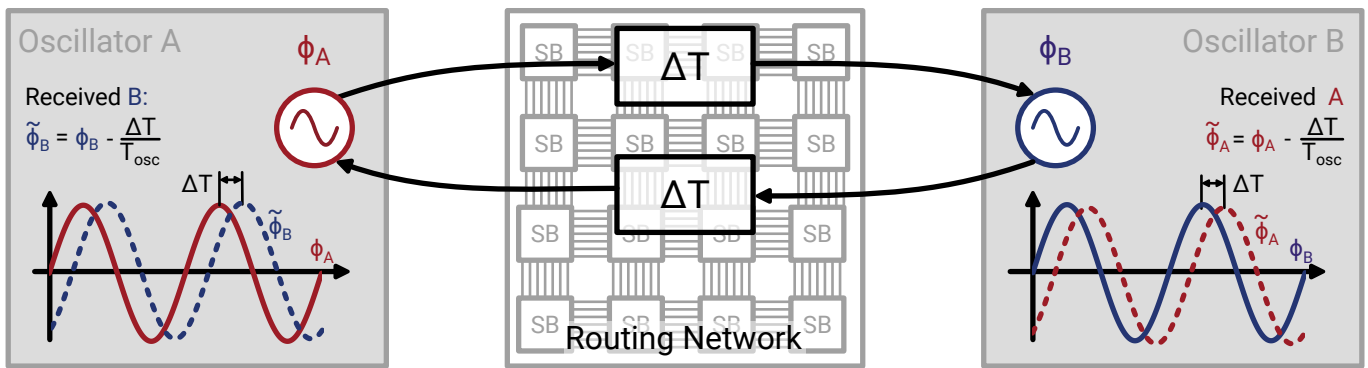


Figure 5.24.: Illustration of the issue caused by the unavoidable delay in a routing network. In this example, the oscillators have identical phases  $\phi_A$  and  $\phi_B$ . Due to the transmission, the received phases  $\tilde{\phi}_A$  and  $\tilde{\phi}_B$  appear phase shifted.

oscillator A is transmitted through the network to oscillator B and vice versa. The signal propagation through the network introduces the delay  $\Delta T$ , which causes a phase lag of the received signal. In the shown example, both oscillators are in the same phase  $\phi_A = \phi_B$ . However, both oscillators are locally leading the phase by  $\frac{\Delta T}{T_{osc}}$ , where  $T_{osc}$  denotes the oscillation period. Consequently, both will react based on that perceived phase so that the delay leads to undesired coupling behavior.

A straightforward goal for the design would be to keep the routing delay  $\Delta T$  negligible compared to the oscillation period. However, there is an unavoidable physical limitation of the delay by the transmission over a given physical distance. In practice, the chosen silicon technology will put a limit on the minimum achievable delay. Additionally, the power consumption puts another practical constraint on the minimum delay. Assuming that the routing delay cannot be reduced further, there are two possibilities. Either the operating frequency of the oscillator could be reduced so that the delay becomes insignificant, or the delay could be compensated.

The simulation data provided in Figure 5.25 illustrates the effect of coupling in the presence of a delay. The color-coded map shows the settled phases for different routing delays and initial phases. The configured

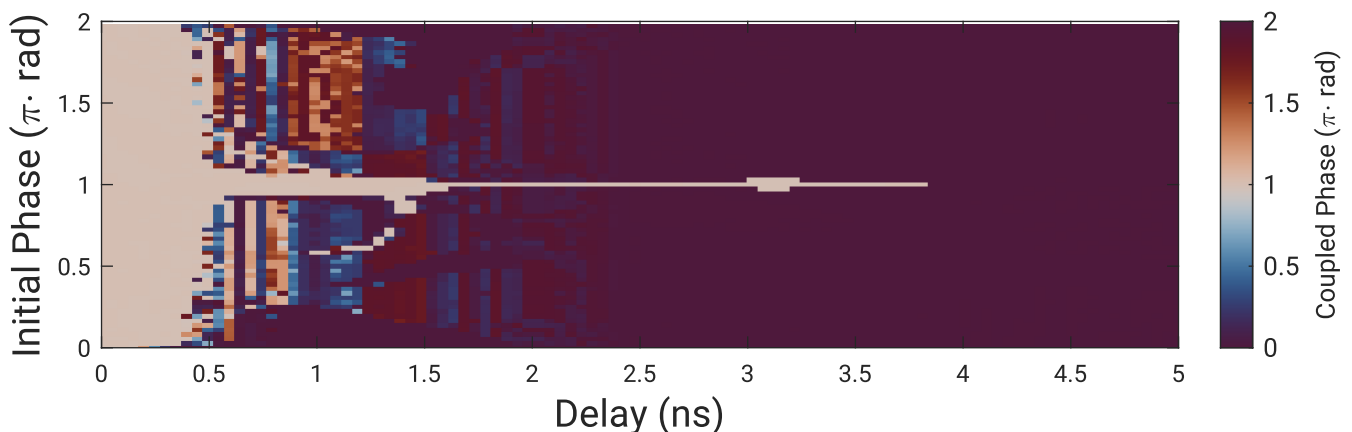


Figure 5.25.: Simulated coupling of two oscillators with a delay. The map shows the phases after letting the oscillators couple for a fixed time of 250 ns. The initial starting phase is shown by the y-axis and the delay on the x-axis. The oscillators are configured to couple with a phase of  $\pi$ .

anti-phase coupling is only achieved for small delays up to roughly 250 ps. For higher delays, the oscillators only settle as intended when starting from certain initial phases. Consequently, such delays must be avoided since coupling is not reliable for any initial phases. This data might suggest a rather premature conclusion. The oscillators couple consistently exactly opposite as intended for large delays. So, the configured coupling could be inverted while ensuring a large delay. However, such an approach will not lead to the desired behavior, because coupling in the unintended opposite phase causes a strong frequency shift.

The frequency shifts when coupled are illustrated in Figure 5.26. Since the coupling can settle as intended or not at a given delay, the behavior of the majority is displayed. So, if there are more initial phases that lead to settling in the unintended opposite phase, this delay is marked as opposite and only the frequency of this group is considered. Hence, the real behavior is somewhat blurry at the interface between intended and unintended behavior. The frequency shift away from the free-running frequency is highly undesirable for the coupling operation. Essentially, only the points with a delay of an integer multiple of the period are actually desired. So the delay should be as close as possible to these points.

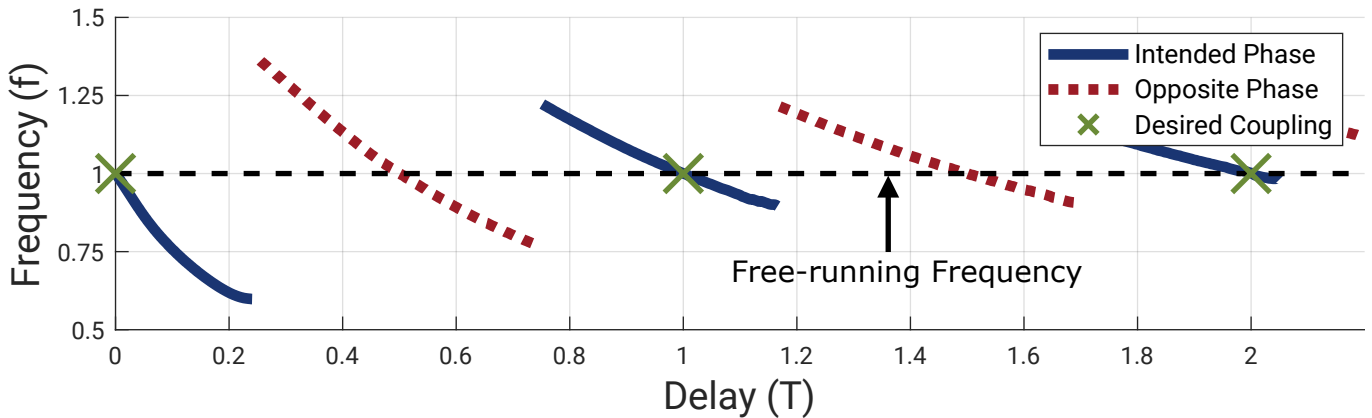


Figure 5.26.: Plot of the frequency shift induced by coupling with delays. The blue range shows where the oscillators settle as intended. The red shows, where the oscillators settle in the undesired opposite phase. Since this can depend on the initial phase, each point corresponds to the behavior of the majority.

### 5.5.3. Compensation Scheme

Reducing the oscillator frequency is a straightforward but unfavorable approach. The delay for large OIM systems with thousands of nodes could quickly accumulate, resulting in operating frequencies in the lower megahertz range or slower. Hence, this work focuses on a compensation approach, which prevents frequency reductions and is more interesting from a research perspective. The fundamental limitation is the unavoidable propagation delay of the routing network. When the phase information from one physical location is transmitted to another physical location, it will be perceived delayed at the receiver. In integrated electronics, the delay of electrical signal transmissions is mostly determined by the physical trace and delay of eventual buffering circuits.

The compensation approach is based on the circular traveling signal in a ring oscillator. As shown in Figure 5.27a, each delay cell of the oscillator produces a time, respectively phase, shifted waveform of the previous stage. The output of stage 3 is leading in time compared to stage 4, which is, by definition, the

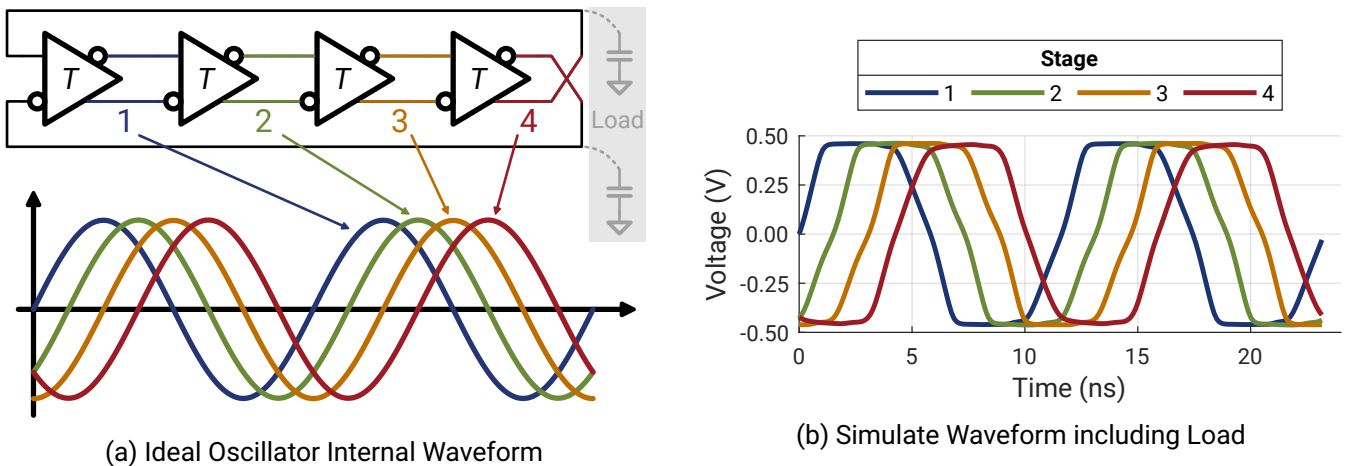


Figure 5.27.: The internal waveforms of the 4-stage oscillator<sup>†10</sup>. (a) The ideal conceptual waveforms using a sinusoidal signal. (b) Actual simulated waveforms.

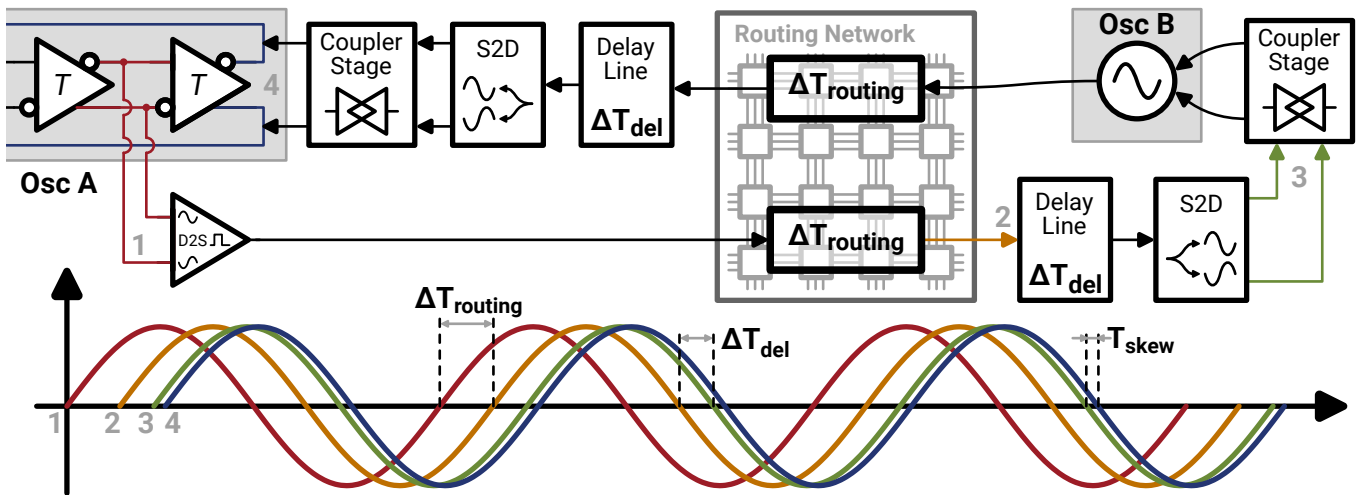


Figure 5.28.: Delay compensation scheme for the routing. The signal of the previous oscillator stage (1) is transmitted through the routing network (2). The delay line supplements the delay so that the received signal (3) matches the actual oscillator waveform (4). S2D: single-ended to differential converter

oscillator output. This could roughly be interpreted as ‘negative delay’<sup>†11</sup>. So, the output of a previous stage can be transmitted and gets delayed by the propagation through the routing network. When this routing delay matches the delay of the 4<sup>th</sup> stage, then the received phase is synchronous with the actual phase of the oscillator. Although theoretically each stage should contribute a delay of  $\frac{T_{osc}}{8}$ , the individual delays slightly vary due to the load. As shown by the simulated oscillator waveform in Figure 5.27b, the delays are close but not perfectly identical. Figure 5.28 shows the compensation scheme, which aligns the phase of the

<sup>†10</sup>The waveform is actually inverted after each stage. So, the shown waveforms of the 2<sup>nd</sup> and 4<sup>th</sup> stage are drawn inverted for better visibility. This visually emphasizes the phases of the individual stages.

<sup>†11</sup>Of course, this is not a true negative delay. Due to the periodicity of the signal, this appears to be like a negative delay and provides an intuitive understanding. When a coupling input at stage 4 changes its phase, this disturbance must travel through stages 1 and 2 before it reaches stage 3. So, the causality principle is not violated and this is not a true negative delay.

arriving signal with the oscillator output. The delay of the routing network  $\Delta T_{routing}$ , which varies with the distance of the connection, gets supplemented by the delay line with  $\Delta T_{del}$ . The resulting signal at the receiving oscillator B is approximately phase synchronous with the oscillator itself, so that only a small  $T_{skew}$  remains. The delay line has a resolution of approximately 100 ps under nominal conditions so that a skew  $|T_{skew}| \leq 50$  ps is achieved. A delay compensation is only reached in a periodic steady state. For dynamic frequency changes, which happen during the dynamic coupling process, the phases are not perfectly aligned anymore. Furthermore, the phase dynamics are affected by the delay compensation scheme. Every change of an oscillator phase is received by the other oscillator approximately 1 full period later. Sampled coupling schemes, which consider the oscillator phases only at discrete time points, might be immune to these changed feedback dynamics. Thus, the delay compensation scheme might maintain an identical coupling behavior in this special case.

## 5.6. Spin-Bias Connection

The figure on the right shows the difference between the regular oscillator-to-oscillator and spin-bias connections using the Ising model. The couplers, as discussed in Section 5.3 implement the  $J_{ij} \cdot \sigma_i \sigma_j$  term, which are direct oscillator-to-oscillator connections. The spin-bias connections are responsible for implementing the  $h_i \cdot \sigma_i$  term, which just depends on the state of a single oscillator. For problems with just  $J_{ij}$  coefficients, the states  $\sigma_i = +1$  and  $\sigma_i = -1$  are ambiguous since  $J_{ij} \cdot \sigma_i \sigma_j = J_{ij} \cdot (-\sigma_i) \cdot (-\sigma_j)$ . Hence, all states  $\sigma$  can be inverted without affecting the Ising Hamiltonian. As soon as the bias term  $h_i$  is introduced, this is not valid anymore, because it depends on just one variable:  $h_i \cdot \sigma_i \neq h_i \cdot (-\sigma_i)$ . So, the implementation of such connections needs a phase reference to define the state for  $+1$  and  $-1$ . The spin-bias connections couple the oscillator to this fixed reference.

$$H(\sigma) = - \sum_{\langle i,j \rangle} J_{ij} \cdot \sigma_i \sigma_j - \sum_{\langle i \rangle} h_i \cdot \sigma_i$$

**Osc-to-Osc Connection**

**Section 5.3**

**Spin-Bias Connection**

**Section 5.6**

### 5.6.1. All-to-all Network Implementations

A method for the bias term has been proposed for all-to-all connected networks [130]. This essentially defines one of the available oscillators as a reference, whose state is always  $+1$  by definition<sup>†12</sup>. The spin-bias connections are then simply coupling connections to this reference oscillator. Hence, it essentially is a normal oscillator-to-oscillator coupling, where the reference oscillator just acts as a definition for the state  $+1$ . While this method is an elegant implementation, it requires an oscillator, which is connected to all other oscillators. For all-to-all connected networks, such an oscillator is naturally available and would just reduce the number of available oscillators by one. However, at sparse networks such an oscillator is not available and its implementation much more challenging. This one reference oscillator must be coupled to every other oscillator. In the Glowworm implementation, such a special oscillator would need to be coupled with the 1440 other oscillators, which is clearly undesirable.

<sup>†12</sup>or  $-1$ , since it is per definition, one can arbitrarily choose the state.



## 5.6.2. Implementation

The spin-bias couplings aim to be as similar as possible to the regular couplings while requiring a minimum area. The spin-bias implementation of all-to-all networks is not directly applicable to sparse networks. The here proposed implementation avoids such a special, highly connected oscillator by leveraging the already distributed clocks. The main system reference clock is used to define the phase for the +1 state. Then, every oscillator can be unidirectionally coupled to that clock. When coupled in-phase, the coupling strives to establish a +1 state and anti-phase coupling a -1 state.

Compared to a local oscillator-to-oscillator coupling, it just needs a small additional conversion from the single-ended clock to a differential signal for the input of the coupler stage. The same conversion circuit as for the routing system in Section 5.5.1 is used. The coupler stage circuit is identical with the oscillator-to-oscillator connections. Additionally, the maximum coupling strength of this spin-bias connection was increased by a factor of 4 and the resolution increased by one bit. When QUBO problems are transformed to the Ising model, multiple  $Q_{ij}$  coefficients map to a single  $h_i$  coefficient, potentially necessitating higher strengths for  $h_i$  than ordinary  $J_{ij}$  coupling connections (see the transformation in Section 2.3.2).

However, this implementation has minor drawbacks. The coupling connection is not completely identical because the spin-bias coupler is driven by a reference clock. The converted differential signal from the single-ended clock might not imitate an oscillator signal accurately. Furthermore, the fixed phase and frequency of the reference clock forces the whole coupled oscillator network to adapt to its frequency. In contrast, the oscillators in the bidirectional connections can (temporarily) change their frequency.

## 5.6.3. Phase Calibration

A phase calibration is necessary to ensure, that the two discrete phase groups created by the SHIL align with the phase groups caused by the spin-bias connections. As previously discussed, different mechanisms are used for the injections into the oscillator. The SHIL current modulates the stage bias current, while the

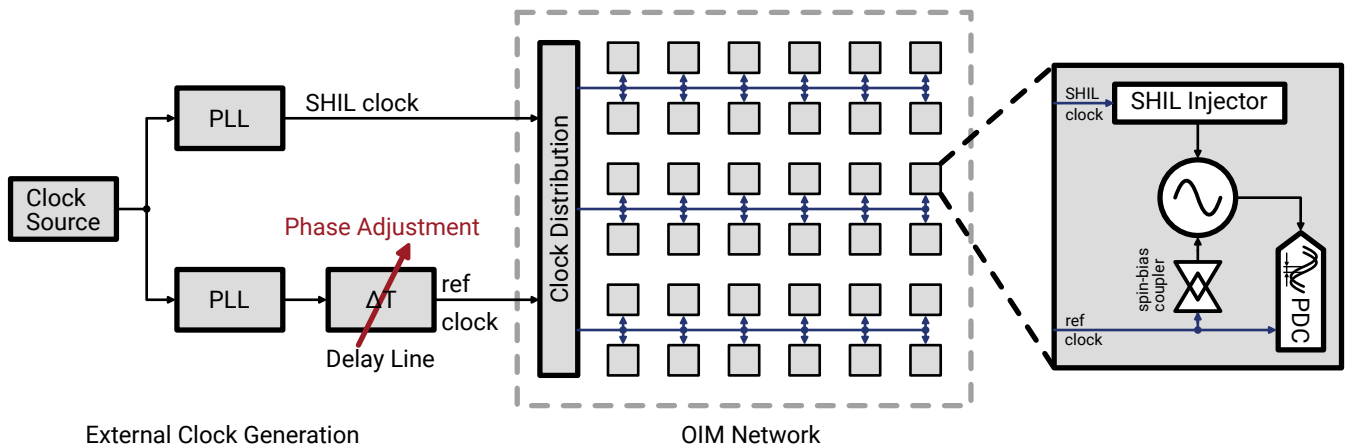
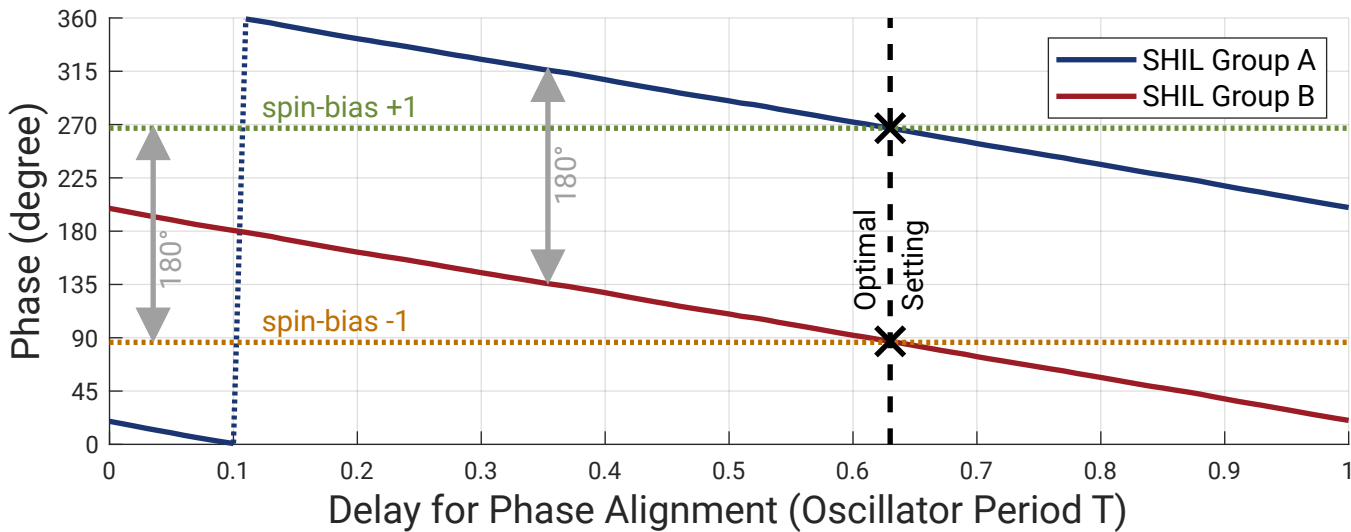


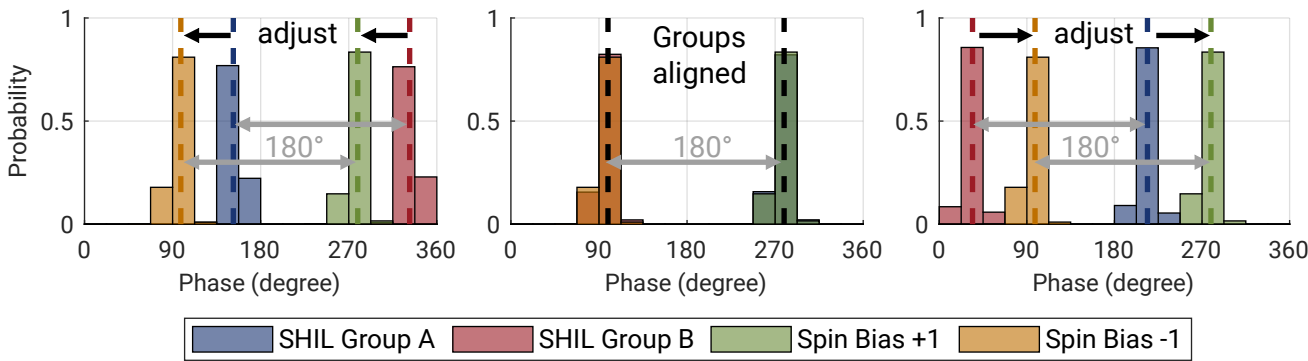
Figure 5.29.: Overview of the clocking system, which requires a phase calibration between the SHIL and reference clock. The SHIL and reference clocks are derived from the same source to guarantee an integer ratio between both clocks. An adjustable delay line in the reference clocking path can adjust the phase to align the SHIL and reference clock at the oscillator's locations. The clock tree is shown very simplified.

oscillator-to-oscillator couplings are directly injected into the differential signal. Thus, the phase angles caused by these injections need to be aligned. Additionally, there could also be skews in the clock generation and distribution. The alignment of both mechanisms is crucial, so that the discrete phase groups match the phases intended by the spin-bias couplers.

The system's clock distribution and usage are shown in Figure 5.29. The reference and the SHIL clock are generated from the same source to ensure they form an exact harmonic. If the ratio between both clocks is not an integer ratio, the spin-bias and SHIL would force the oscillators into different frequencies, making an alignment impossible. An external delay line is used to precisely adjust the phase between both clocks so that the injections at the oscillators align. Because those clocks operate at an integer frequency ratio, the following description of the alignment uses the term delay instead of phase.



(a) Delay adjustment to match the SHIL groups with the spin-bias groups



(b) Illustration of the adjustment procedure with the actual phase distributions

Figure 5.30.: Illustration of the spin-bias calibration using measurement data. As indicated by the gray arrows, the phase groups caused by the SHIL and spin-bias are always  $180^\circ$  apart. (a) The two phase groups caused by the SHIL, A and B, need to be aligned to the phase groups caused by the spin-bias groups +1 and -1. The delay setting, which matches the phase centers (black dashed line) is then the calibrated setting. (b) Example of the actual distribution of the phases and the derived center of these phase groups (dashed) line. When successfully calibrated, the histograms and their centers are superimposed.

The actual calibration is straightforward and essentially a search problem. The optimal delay causes the phase groups of the SHIL and spin-bias connections to match as illustrated in Figure 5.30a. It is sufficient to get a close alignment of the phase groups so that it does not noticeably affect the computing itself. Any of the two discrete phase groups caused by the SHIL, A and B, need to be matched to the spin-bias  $+1$  and  $-1$  groups. The assignment of groups A and B to a specific spin-bias  $\pm 1$  does not matter. The actual matching of the phase groups is shown in Figure 5.30b. On the left side, the delay is too small, while it is too large on the right side. The middle image shows the desired alignment of both phase groups at the correct delay setting.

## 5.7. Digital Control & Configuration

The digital configuration is mentioned for the sake of completeness. All digital circuitry are combined into one block, which contains the following main functional areas:

- Configuration Memory
- Phase Measurement
- SHIL-Ramping
- Frequency Measurement & Calibration

They support the OIM operation. A block diagram showing the interaction and organization of the blocks is provided in Figure 5.31. Since the digital configuration is not directly involved in the analog computation, it does not influence the outcome. There is an indirect influence by the oscillator frequency calibration. Of course, this assumes that the phase measurement and weight settings are conducted as intended. The SHIL ramping and the frequency measurement of the oscillators were previously discussed together with their analog counterparts. The configuration memory and the phase measurement are described in the following sections.

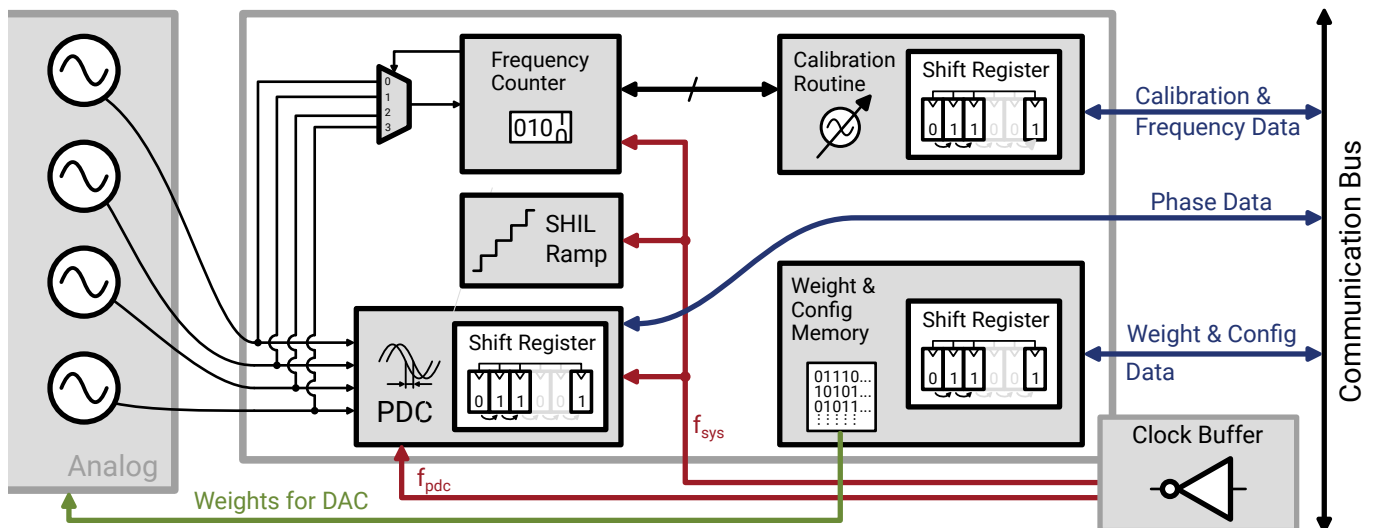


Figure 5.31.: Simplified diagram of the digital configuration system, which serves as digital interface for the analog computing. It combines multiple independent digital subsystems into one major block.

### 5.7.1. Weight & Configuration Memory

The main purpose of the configuration memory is to store the weights of the optimization problems. Additional configuration bits for the experimental analysis are included as well. Flip-flops store all needed configuration bits. They are automatically inferred and laid out by the synthesis and place & route tools. The flip-flops are arranged as large shift registers to simplify writing and reading data. This makes the blocks self contained and scalable. As many blocks as desired can be chained together, without needing any additional circuitry or wiring. This approach is well suited for a research oriented design. However, achieving the highest throughput for the computation would need a faster data transmission approach instead.

Compared to static random-access memory (SRAM), a single flip-flop occupies significantly more area per bit. However, SRAM needs additional circuitry to access the bit and word lines and might require sense amplifiers. Also, a custom SRAM implementation with a parallel output would be needed to provide all bits for the DACs concurrently. Additionally, some sort of interface would be needed to sequentially write the data into such an SRAM memory. A potential implementation using distributed SRAM in each control block is provided in Figure 5.32. Nevertheless, SRAM will likely be a better choice for area-optimized design. Currently, the whole digital block including the configuration memory accounts for approximately 25% of the overall network area, as later discussed in Section 5.8.1. Thereof, the memory occupies the most area. However, such an SRAM implementation is an engineering task, which provides hardly any scientific benefit for the scope of this thesis.

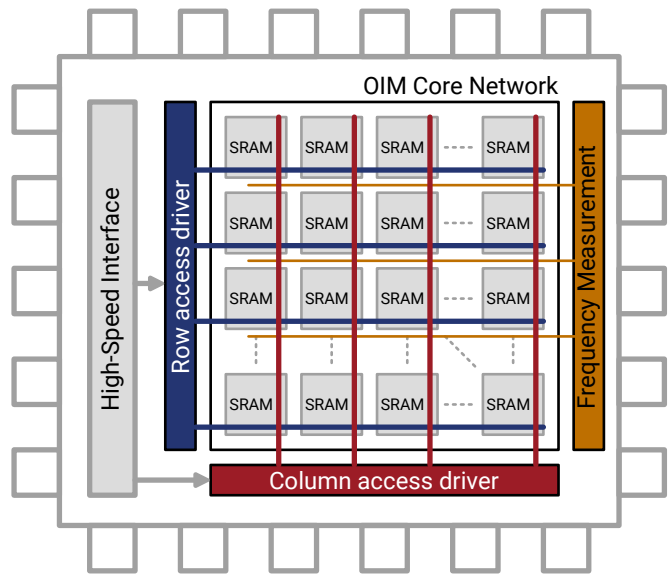


Figure 5.32.: Possible implementation using SRAM instead of flip-flop memory. A row and column driver accesses the SRAM cells of every block. The frequency measurement can be moved from the digital block to a central place, which is connected to the oscillators via the routing network.

### 5.7.2. Phase-to-Digital Converter

The phases of the oscillators should be determined quickly with sufficient resolution. Because the spins of the Ising model have two discrete states, a binary phase detection would theoretically be sufficient. However, the phase threshold of such a 1-bit measurement would have to match the discrete phase group angles caused by the SHIL. For higher order models, such as the Potts model, a higher phase measurement resolution is

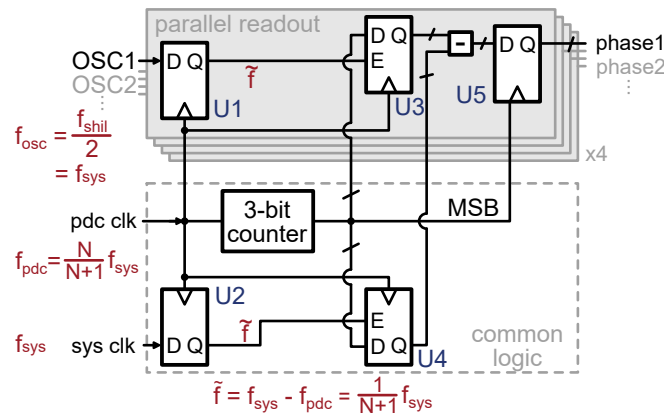


Figure 5.33.: Simplified principle of the parallel PDC operation with the frequencies annotated in red. The double flip-flop synchronizers and glitch filters needed in this asynchronous design are not shown for better visibility.

necessary. Having not only 2 but  $n$  discrete states that are phase encoded in the oscillators, increases the demand for the resolution of the phase measurement accordingly.

This work uses the all-digital phase-to-digital converter (PDC) implementation proposed by Nico Angeli and Klaus Hofmann for the phase measurement [131]. The fully synthesizable design can be customized regarding the needed phase resolution. However, a higher measurement resolution increases the duration for a single measurement and slightly increases the needed area. Since the PDC principle is based on an asynchronous counter, the concept can be easily parallelized to measure multiple phases concurrently. Hence, this flexibility and ease of integration make it well-suited for the OIM prototype in this work.

For convenience, the principle is summarized here, but more detailed information about the general principle are available in [131]. The PDC approach was customized for the OIMs with the support of Nico Angeli. A simplified overview of the principle is depicted in Figure 5.33. The system clock  $f_{\text{sys}}$  is used as global phase reference for the chip, which is identical to the frequency of the oscillators under SHIL lock. The required asynchronous clock  $f_{\text{pdc}} = \frac{N}{N+1} \cdot f_{\text{sys}}$  is slightly slower, which is determined by the phase resolution  $N$ . The Firefly design has a 3-bit resolution with  $N = 8$  and the Glowworm design has a 4-bit resolution with  $N = 16$ . The first stage of flip-flops U1/U2 downsamples the input similar to an analog mixer. The phase of the input signal is preserved, so that U3/U4 sample the time point of the rising edge. By calculating the difference between these two timepoints, the phase angle is obtained and stored in U5. The phase measurement is then easily conducted in parallel for multiple oscillators, as indicated by the gray area.

The actual implementation uses additional double flip-flop synchronizers and glitch filters, which are needed due to the asynchrony of the signals. Additional clock gating is applied to save power. Although the power consumption of this block is not directly experimentally measurable, the analog circuitry dominates the overall energy. As a drawback, the PDC requires the  $f_{\text{pdc}}$  clock, which has to be in a specific ratio to the oscillator frequency  $f_{\text{osc}}$ . Since the oscillators are SHIL locked during read-out, their frequency is fixed at the sub-harmonic of the SHIL and therefore controlled by the clock. However, not all oscillators always remain in their discrete state, as experimentally shown in Chapter 7. Despite strong SHIL, some might change their phase during a measurement, which potentially leads to a false result. However, the concept of phase breaks down without an identical frequency of all signals of interest, which would make a phase measurement pointless anyhow.

For future implementations, it is desirable to replace the PDC with a faster measurement approach. The

biggest improvement is a reduced measurement time, where the phase measurement currently accounts for 44.5% (Firefly) and 59.8% (Glowworm) of the time per computation. When considering the energy needed for a computation, while neglecting the writing of the configuration memory, it affects the total energy consumption by approximately the same share. Theoretically, a single oscillation cycle would be sufficient to measure the phase. When using the Ising model, which only has two discrete states, a single sampling of the oscillator using an edge-triggered flip-flop could be already enough. The oscillator waveform could be sampled in the middle between the low-high/high-low transitions of the locked state. However, the sampling time-point might need additional alignment to compensate for potential variations in the frequency and phases.

## 5.8. Physical Design

The layout is organized in one (Firefly) or two (Glowworm) tiles. These tiles contain the necessary circuitry, including periphery, so that the OIM network can be established by just placing the tiles in a grid. Each tile contains the analog and digital circuitry, where 4 oscillators were combined into one cell to share parts of the digital circuitry. This saves area, although any other number of oscillators could be implemented in a single tile. The layout view of the tiles is shown in Figure 5.34. The larger first generation tile is in 5.34a and the second generation is in 5.34b. Both tiles have a very similar topology, where the shared digital control block is placed centrally and the analog circuitry arranged in the four quadrants around. By vertical and horizontal flipping of the analog oscillator node layout, each quadrant is filled with exactly the same circuit. Only a few common elements are not symmetric, like the clock buffering and shared biasing circuit, which are placed at the top and bottom middle. It should be noted, that the tile of the second generation does not occupy the full rectangular area. The corners are intentionally left empty to provide space for the routing network.

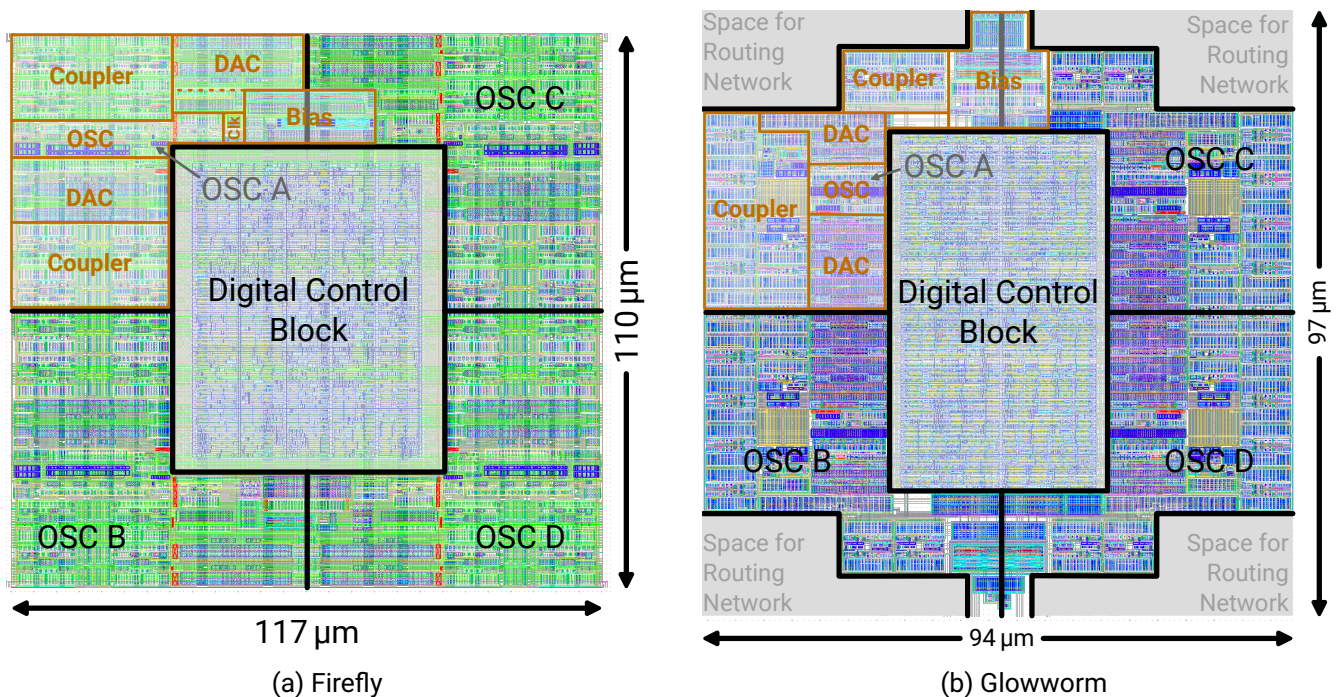


Figure 5.34.: Layout of the oscillator tiles. The second generation is more compact despite offering 16 instead of just 8 connections per oscillator and a routable network.

The separate tile for the routing network is shown in Figure 5.35. It fits exactly in the empty area of the regular tile and provides the connections to the routing network for the adjacent oscillators. The central block was implemented using a place & routed tool and contains the switch-block, configuration memory, and network access circuitry. The 16 delay lines for the compensation scheme for the routing delay are implemented manually at the sides.

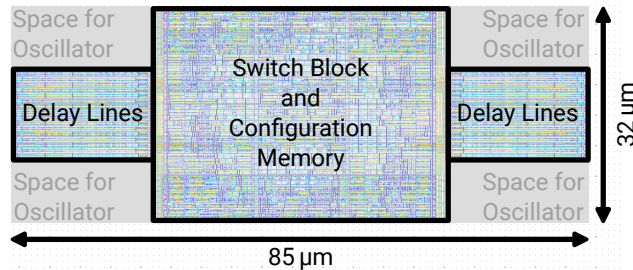


Figure 5.35.: Layout view of the routing tile. The central SB is synthesized and includes the needed configuration memory. 8 delay lines are placed on each side for the compensation scheme.

An overview of the layout of the full system can be found in Figure 5.36. The first generation Firefly is shown in part 5.36a. The OIM core network consists of a 10x10 grid of the before-mentioned tiles, resulting in a total of 400 oscillators. The additional clock receiver and I/O output driver for debugging and external monitors are placed in the small IO Driver section. The equivalent full layout view for the Glowworm tapeout is shown in Figure 5.36b. It consists of a 20x18 grid of oscillator tiles and 19x17 routing cells implementing 1440 oscillators. The outermost oscillators at the edges of the grid are not connected to the routing network because there is not enough space for the routing tile.

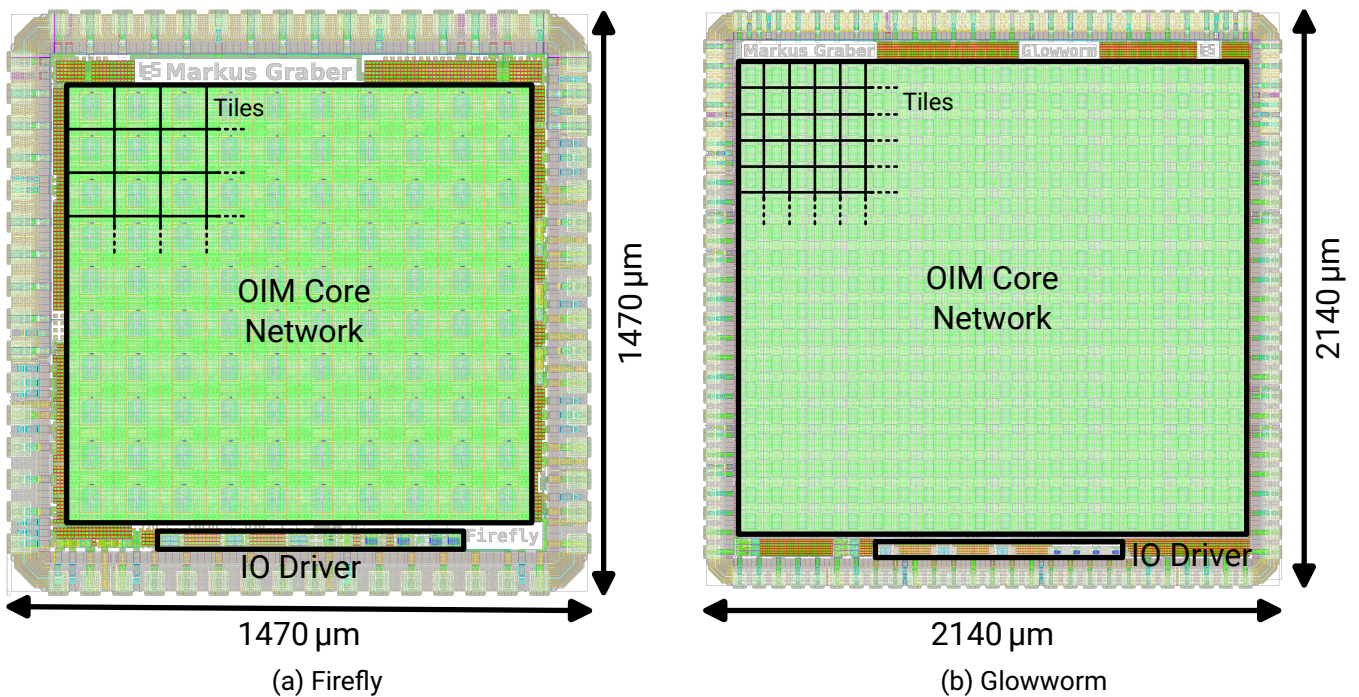


Figure 5.36.: Layout view of both implemented systems.

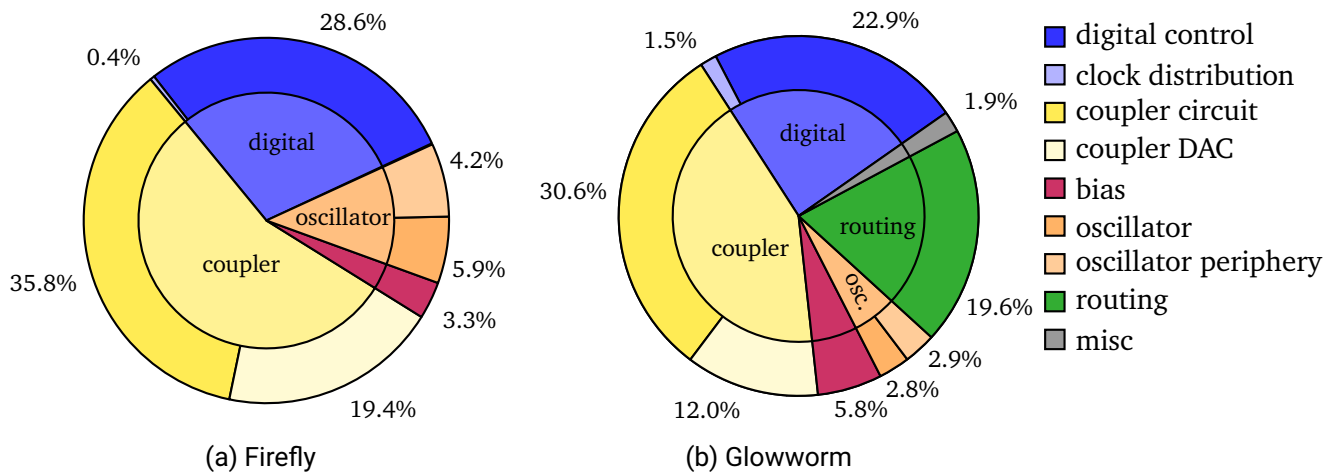


Figure 5.37.: Breakdown of the occupied area by the coupled oscillator network. Any area not belonging to the oscillator network, such as the IO cells, electrostatic discharge (ESD) protection, or logos, are not considered.

### 5.8.1. Area Breakdown

A breakdown of the area consumption of the building blocks is provided in Figure 5.37. Despite area optimization from the 1<sup>st</sup> to the 2<sup>nd</sup> generation, the relative share of the building blocks remains similar. The digital configuration memory and readout circuit occupies approximately a quarter of the overall area. By using SRAM instead of flip-flops and/or changing to even smaller technology nodes, its contribution can be reduced in future generations. The actual oscillator, including its periphery, has a low share of 6% to 10%, while the area needed for the coupler circuits accounts for approximately half of the area. Given the high number of couplers, this is expected and creates one of the major challenges when scaling up the system to larger, more densely connected networks. Also, it underlines that the coupler is a crucial component: It has the highest share of area, a significant influence on the computing, and needs to implement precise weight coefficients. Especially for all-to-all connected networks, the quadratically increasing number of connections creates a serious challenge. A detailed discussion about the scaling is available in a related publication of this work [154].

## 5.9. Overview of Firefly & Glowworm

The die-shots of both generations are available in Figure 5.38. An overview of the differences is provided in Table 5.1. Glowworm is an area and power optimized, scaled-up version with new features, which was developed after the successful testing of Firefly. From a network point of view, it introduces the routing network to connect any nodes within the network flexibly and adds capabilities for the biasing term  $h_i$  of the Ising model. Compared to the King's graph of Firefly, it has an extended connectivity to 11 neighboring nodes and 4 additional connections to the routing network. The number of nodes is increased by a factor of 3.6 and the number of edges by a factor of 7.9, while the core area only increases by a factor of 2.6, which emphasizes the effective area optimizations. The total power consumption increases by only 50% despite the much larger network, highlighting the improved efficiency. The higher computation time is only caused by the increased resolution of the PDC, while the oscillator interaction and SHIL ramping stays identical between both generations. A single computation with the Glowworm design takes 950 ns. 200 ns are used to let the



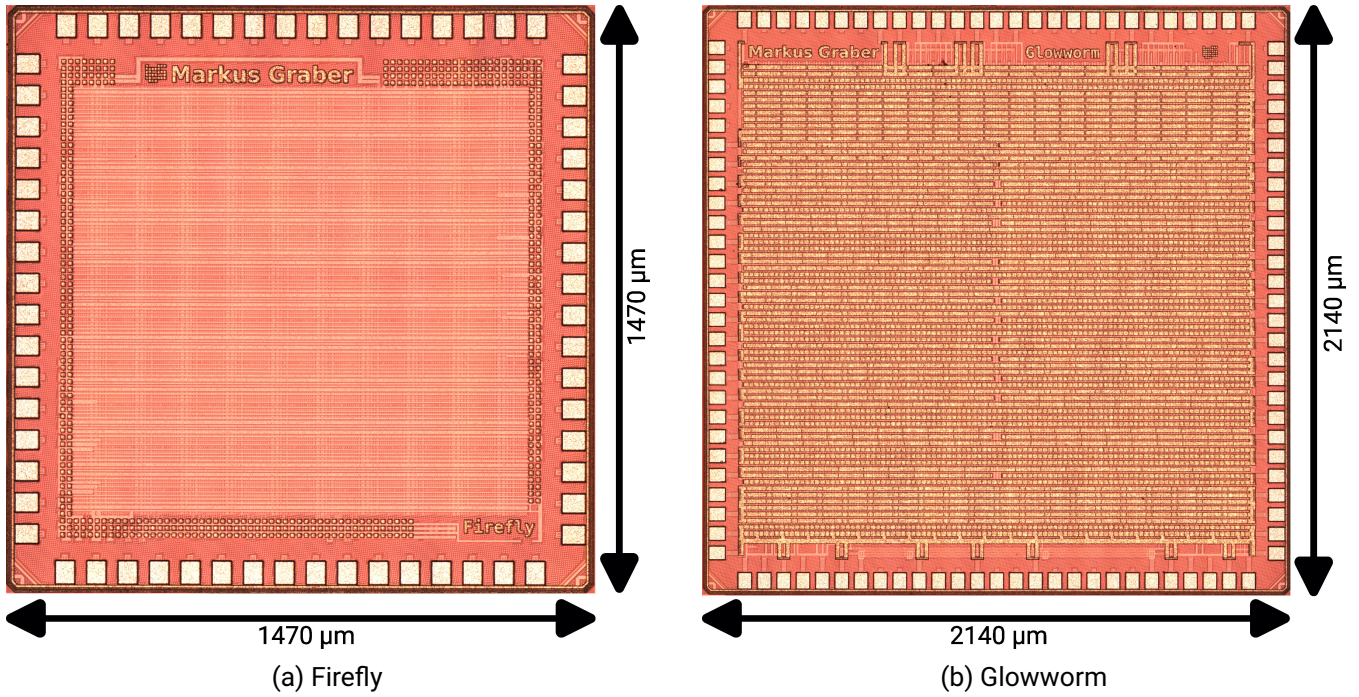


Figure 5.38.: Die-shot of Firefly and Glowworm.

	Firefly	Glowworm
Technology	28 nm	28 nm
Number of Nodes	400	1,440
Number of Edges	1,482	11,724
Network Topology	King's Graph	Local + Routing
Edges per Node	8	15 + spin-bias
Oscillator Frequency	typical 110 MHz	100 MHz
Edge weight resolution	6 bit + sign	4 bit + sign
Routable Edges	no	yes
Bias term edges ( $h_i$ )	no	yes
Calibration	Frequency	Frequency, Reference Phase, Routing Delay
Total Area	2.2 mm <sup>2</sup>	4.6 mm <sup>2</sup>
Core Area	1.3 mm <sup>2</sup>	3.33 mm <sup>2</sup>
Area per node	3150 μm <sup>2</sup>	2313 μm <sup>2</sup>
Area per edge	877 μm <sup>2</sup>	284 μm <sup>2</sup>
Phase Readout Resolution	3 bit	4 bit
Typical Power Consumption	302.9 mW	460.3 mW
Typical Consumption Time	713 ns	950 ns
Publications	[151], [155], [158]	[154], [150], [153]

Table 5.1.: Comparison between the Firefly and Glowworm designs.

---

oscillators couple. The SHIL ramp-up takes 182 ns and the phase measurement takes 568 ns.

Regarding the core analog circuit architecture, both designs share the same design idea. Besides multiple optimizations for area, matching, and power consumption, the most significant changes are the coupler circuits. The general idea was kept, but the AC coupling capacitor and the CMFB of the first generation were replaced by the current summation circuit, which is more area and power-efficient. Additionally, the currents of the couplers are summed first and then injected through the AC coupling.

## 5.10. Future Features

During the design cycles, there have been more ideas to extend and improve OIMs as discussed. Due to time and area restrictions, some were not physical implemented on chip, but extensively studied. Possible future features are mentioned in the following sections.

### 5.10.1. Weight Setting DACs

The coupler was designed for its compactness. However, the setting of the weights is implemented using simple current DACs. The proposed implementation uses an individual DAC for each coupling connection. Hence, there are currently as many DACs implemented as coupling connections. The DACs alone occupy approximately 12.0% of the OIM core area<sup>†13</sup>. A time-multiplexed DAC scheme could be developed to save area. A single DAC could sequentially drive the bias currents for the coupler circuits. The resulting gate-source voltage of the current mirror could be stored on a small capacitance and just refreshed or reconfigured. Since the computation is fast, within approximately a microsecond, leakage might not become an issue. Memristors could be even better suited to store such weights, since they can retain analog values in a single device for long periods.

A possible time-multiplexed implementation could, for example, use just one DAC per node. Alternatively, a crossbar-like structure for the whole network could be employed. Each column could be driven by a DAC and the access is controlled via the rows. Such a time multiplexing approach could not only save area but might enable the use of a few high-quality DACs. Thus, the resolution could be increased and the inaccuracy of the DACs, such as the differential non-linearity (DNL), could be lowered. This would also enable the implementation of a calibration scheme for the weights to compensate for random variations of the couplers.

### 5.10.2. Bias Calibration & Distribution

As will be discussed in the experimental section in 7.3.3, the matching of the coupling strength is important. Especially for weighted problems, the strength of the coupler should have a minimal variation from their nominal value despite mismatch. The random spread of manufactured device parameters, will make some coupling connections stronger than others. While the DACs set the individual strength levels, their bias currents determine the absolute strength. Hence, a precise bias current with low variations is needed. While both presented designs in this thesis use a simple resistor-based biasing circuit, more advanced schemes are discussed in a supervised thesis and associated paper [168], [160]. Since the area of a biasing circuit is critical, the precision vs. area trade-off must be carefully chosen. The supervised thesis investigated and optimized multiple possible bias implementations for their precision-to-area ratio. A bias circuit calibration

---

<sup>†13</sup>Only the area for all OIM components are considered. I/O ring and other periphery occupying the remaining silicon area are not considered in this number.

---

scheme, which uses a central comparison block, is the most promising approach. A separate network routes the current of each bias into the central comparison block. Then, the bias currents can be adjusted to achieve the same currents in each bias source. A further step would be an actual calibration scheme for the coupling strength. The bias current is just an intermediate step that compensates some but not all of the error sources of the coupling weights. However, measuring the bias current is much simpler than determining the coupling strength on an absolute scale.

### 5.10.3. Objective Computing Core

The experimental evaluation in Chapter 7 demonstrates that computation using OIMs can produce non-deterministic outcomes. So, when computing the same problem repeatedly, some solutions will be better than others. Hence, it makes sense to compute a problem multiple times and select the best out of the computed solutions by comparing the Ising Hamiltonian. Because transmitting the computed solutions to a host system is energy-consuming and might take considerable time, the objective of the computed problem could be computed directly on-chip. So, the best solution could be selected and transmitted, while worse results would be dropped immediately.

A concept like this requires an on-chip objective computing core to select the best solution. Possible concepts were evaluated in the supervised thesis by Xu [172]. The special property of such an approach is that the individual states of the discrete variables, represented by the physical oscillators, are initially spatially distributed across the whole coupled oscillator network. Hence, this results in two different approaches to compute the objective. The states can be collected and transmitted across the chip to a central location, where the objective is computed. Alternatively, the computing of the Hamiltonian, which is essential just a large summation can be conducted spatially distributed in the proximity of each node.

## 5.11. Conclusion of the System & Circuit Design

The first and central design decision for OIMs is the network topology. This depends on the optimization problem, as a sufficient number of oscillators and couplers in the network need to be available. All-to-all networks have the best possible connectivity but are limited to a small number of nodes since the needed area scales quadratically. Especially, when solving sparse optimization problems with many nodes, the number of oscillators is likely insufficient and most of the couplers will be unused. Hence, a sparse network with up to 1440 nodes is implemented in this work. In theory, multiple physical oscillators could be combined to form a logical node, trading in the number of nodes for more connectivity, as this is commonly done for the D-Wave quantum annealer. The main disadvantage of sparse topologies is the time-consuming embedding, which can easily exceed the computation time by the OIM. Hence, a flexible routing network, similar to the principle of FPGAs, is proposed to reduce that bottleneck. Unfortunately, the unavoidable propagation delay of transmitting an oscillator signal through the network is highly problematic for the coupling. So, a delay compensation scheme is proposed to enable coupling even in the presence of such delays, although there is a remaining impact on the phase dynamics. Additional spin-bias connections implement the  $h_i$  term of the Ising model so that QUBO problems can also be solved.

The circuit is based on a 4-stage differential ring oscillator topology. The low number of stages make the oscillator more sensitive to coupling inputs and the differential architecture makes it more robust against disturbances on the power supply and common mode noise. A frequency calibration is introduced to compensate for mismatch, reducing the frequency standard deviation to approximately 0.35 MHz. The coupler is based on

---

an active double differential stage so that the strength is defined by the bias current. Thus, an external bias voltage adjusts the global coupling strength while the weight coefficients are set using a DAC. Since such a stage is just unidirectional, they are connected back-to-back. The buffered oscillator output is distributed to the coupler stages of the neighboring nodes to prevent loading of the sensitive oscillator node. The self-biased, AC coupled summing circuit implemented in the second generation proved to be a significant area improvement. The remaining circuitry for operation such as the SHIL, frequency calibration, phase measurement, and digital memory is presented as well. The second generation chip Glowworm contains 1,440 oscillators and 11,724 coupling circuits on a core area of just  $3.3 \text{ mm}^2$  (die size:  $4.6 \text{ mm}^2$ ).

### System & Circuit Design

- A **scalable architecture** is presented, which uses a differential ring-oscillator implementation. A **buffered topology** is proposed to mitigate loading effects on the sensitive oscillator nodes.
- **Routing channels** allow to connect (almost) every possible pair of oscillators, by establishing a connection on demand. Due to the inevitable delay of the signal transmission over long distances, a **compensation scheme** is introduced.
- The coupler behavior is one of the factors shaping the oscillator-to-oscillator interaction. A compact, **double differential pair topology** is proposed, whose interaction strength is determined by the provided bias current. So, simple current DACs set the weights according to the optimization problem.
- The **couplers** are the area dominating element and already occupy almost **half of the area** in the proposed designs. When scaling this up to larger, more densely connected systems or even all-to-all connectivity, they will occupy most of the chip area.
- A **compact** implementation for the **spin-bias** connections is proposed, enabling support for the  $h_i$  term of the Ising model. The concept reuses the existing clock of the system and is well-suited for **sparse networks**.
- The second generation fabricated chip in a 28 nm technology implements **1,440 oscillators** and **11,724 couplers** on a  $4.6 \text{ mm}^2$  die.

## 6. Experimental Setup

This chapter describes the experimental setup used for the performance evaluation in the following Chapter 7. It supports the OIM research by interfacing with the fabricated chips and controlling the operating conditions. The focus of the setup is to keep it as simple as possible while remaining versatile enough to not limit the experimental purposes.

### 6.1. Concept

The test concept is depicted in Figure 6.1. The host system coordinates all tests by configuring the OIM chip, setting all operating parameters and analyzing the results. An FPGA acts as an interface to read and write data to the OIM chip and provides timing signals to control its internal operating states. The communication interface between the host system and FPGA limits the maximum throughput of the setup. Transferring data from and to the chip takes longer than solving a problem on the chip. Other peripherals such as PLLs, bias voltages, and power supply are controlled via a microcontroller. Three PLLs, which use the same clock source, provide the clocks for the OIM system. These are the SHIL, the reference, and the dedicated clock for the PDC. Additionally, the reference clock can be delayed for the phase calibration of the spin-bias connections. The optional oscilloscope is not required for the computing operations. However, it can be used to observe the

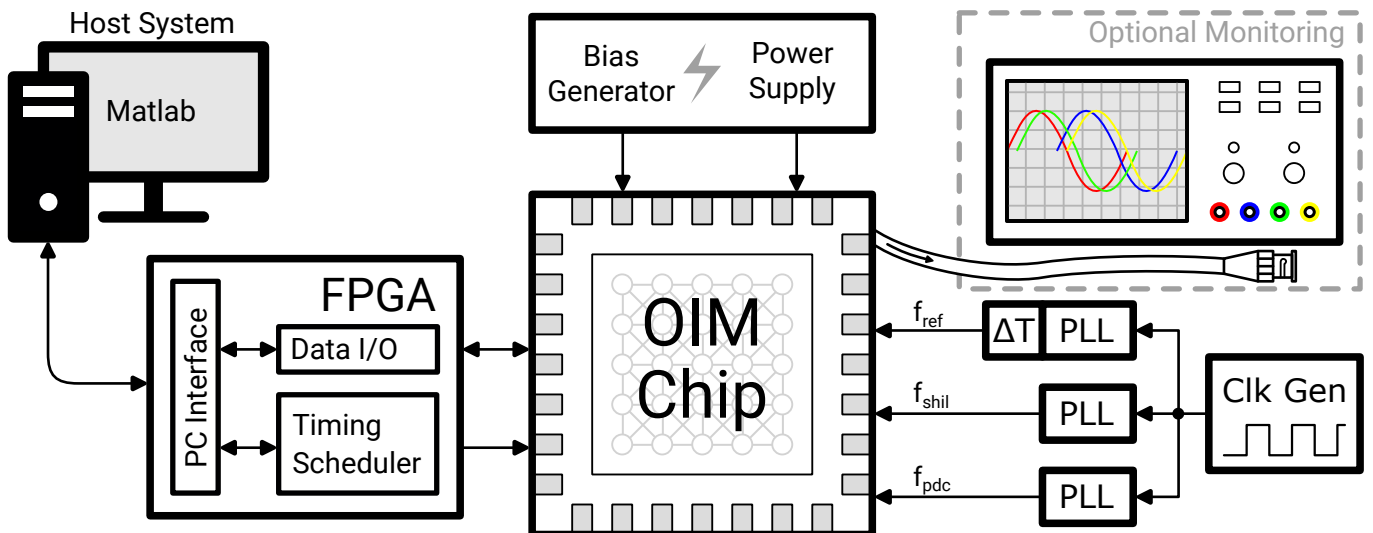


Figure 6.1.: Block diagram of the experimental setup. A host PC running Matlab controls the operation of the OIM chip. An FPGA assists with the data transfer and provides a flexible timing schedule. External bias voltages and clocks can be configured from the host system. An optional oscilloscope allows monitoring of 4 oscillators, which is used in Chapter 7.

dynamic behavior of selected oscillators using its off-chip drivers. These do not preserve the internal analog waveform, so only phase information can be obtained. Examples of the dynamic behavior, such as the settling of two coupled oscillators, are presented in Chapter 7. Otherwise, the OIM chip can sample the phase and frequency information using the internal PDCs and counters, but does not provide insights about the dynamic behavior.

The physical setup for both generations with the dedicated test boards is shown in Figure 6.2. It provides the power supply, bias voltages, clocks, and timing signals for the OIM chip under test. Due to changes of the OIM chip pins and power supply between generations, a similar but revised board is used. The OIM chip is mounted on a dedicated carrier PCB to allow quick swapping and testing of multiple dies.

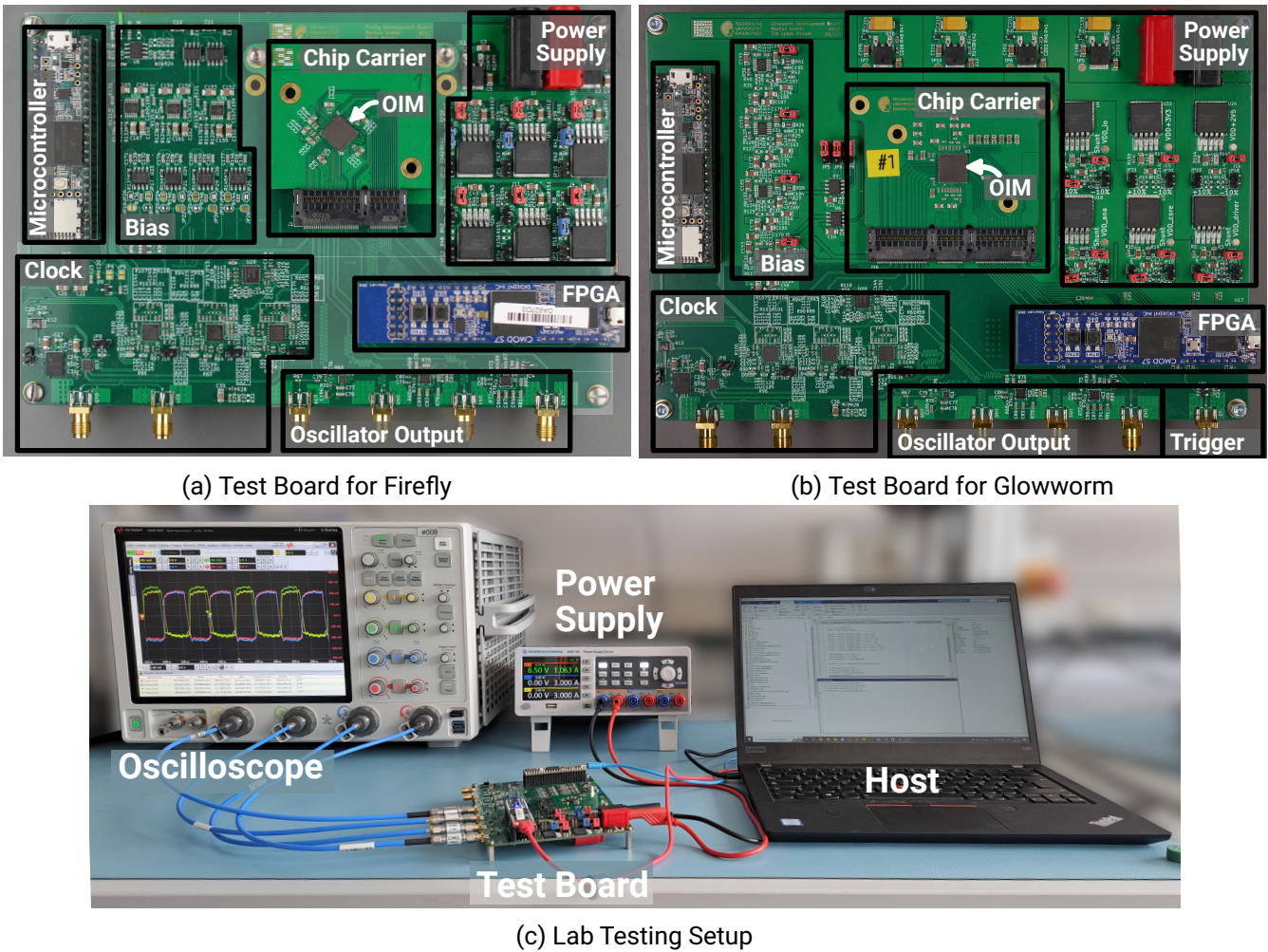


Figure 6.2.: Experimental lab setup for testing the OIM chips. The chips are soldered on carrier boards, which can be easily plugged into the test board.

## 6.2. Software Flow

The software flow for testing and evaluating the OIM chips is shown in Figure 6.3. It can be divided into three steps: the preparation, the configuration and the analysis. For the preparation, a benchmark problem

database was generated and is repeatedly used. This benchmark database consists of randomly generated problems grouped in different sets. Their generation is discussed in Section 6.3. The optimal solution to these problems serves as a baseline for determining the accuracy of the OIM system. Solving a problem can be divided into the configuration before the computation and the analysis of the obtained results. The chip configuration is based on a ‘Run Configuration’, which defines the operating parameters of the OIM such as coupling strength, SHIL frequency, and timing schedule. Together with the optimization problem, it forms the test setup, from which the bitstream, timing, and setup information are derived. The bitstream configures the couplers, oscillators, and routing network on the chip. Additionally, the supporting components on the test board are set up. After the computation, the raw phase data is sent back from the OIM chip to the host computer. Based on the individual phases of the oscillators, the states of the discrete variables are determined. Then, the Ising Hamiltonian is calculated, which can be compared with the optimal solution to determine the accuracy.

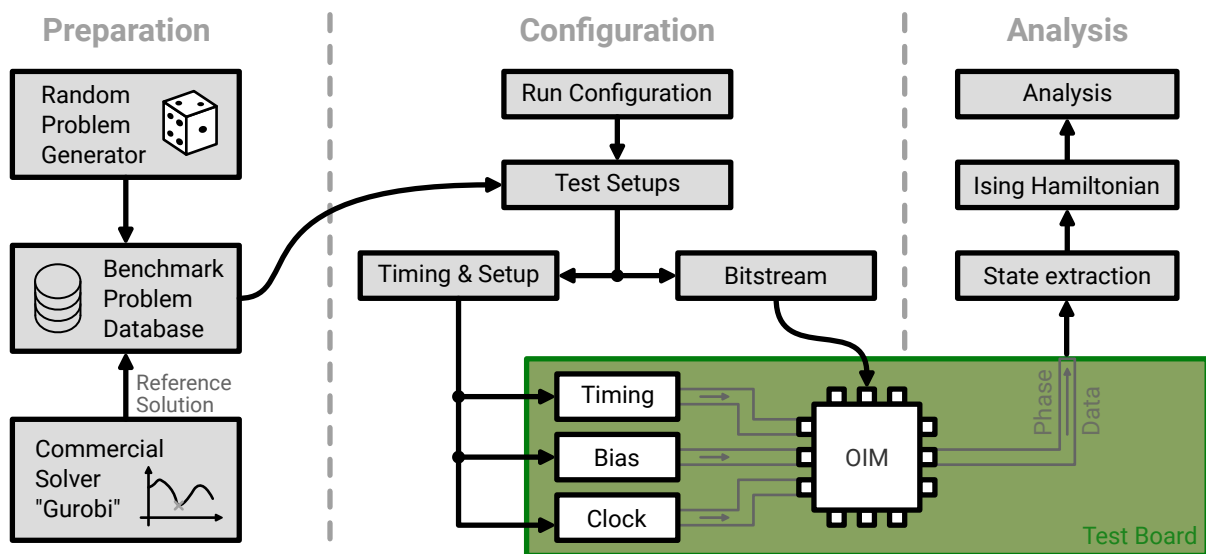


Figure 6.3.: Conceptual overview of the benchmarking concept of the chip, which can be divided into three steps. A preparation step provides a benchmark problem database for evaluation. The configuration step sets up the OIM and applies the operating condition. The analysis step then investigates the computed outcome, which is available as phase angles for each oscillator.

### 6.2.1. Routing Paths

To connect two arbitrary oscillators on the chip using the routable connections, a link through the network must be established by setting all needed physical switches. Usually, it would be the task of the embedding algorithm to assign the tracks for such routable connections. The physical switches are then be configured accordingly. Since no capable embedding algorithm is available for this work, a dedicated routing algorithm is used to connect two given oscillators. The shortestpath function of Matlab is used to find a path in a graph representation of the routing network. All desired oscillator connections are sequentially applied to the network. Because this simple approach does not consider all desired routing connections simultaneously, congestions of the routing network can become an issue. However, it provides fast results and is sufficient for the evaluation.

---

### 6.2.2. Bitstream Generation

A bitstream for the configuration of the OIM is generated based on the local connections and routing connections. For every local connection, the bits for the corresponding DACs are set. For routing connections, the bits for the needed physical switches of the network are set additionally. The timing schedule, which controls the enabling, coupling, SHIL injection, phase measurement and data transmission is transferred to the FPGA. The Glowworm project uses 147,528 bits for configuration, where 69,768 bits are needed for the routing channels and the remaining 77,760 bits for the coupling configuration. The Firefly design has 19,400 bits for the coupling configuration.

### 6.2.3. State Extraction

Since the internal PDC of the OIM-chip provides the raw phase data, the discrete states  $\sigma \in \{+1, -1\}$  must be extracted. The PDC in Firefly has a 3-bit resolution and in Glowworm a 4-bit resolution, providing a resolution of  $45^\circ$  respectively  $22.5^\circ$ . Hence, a range of phase angles needs to be defined for each of the two phase groups. An exemplary distribution of the phases is provided in Figure 6.4. Usually, two distinct phase groups naturally mark the phase center of the discrete states. A threshold for distinguishing should be in the middle between both centers. For the extraction, three possible approaches are considered:

- Fixed threshold, which is once determined
- Identification based on the two distinct groups
- Brute force testing of all possible thresholds

The fixed threshold operation is suitable for high-performance computing tasks, but requires precisely repeatable phase angles. It does not introduce any sophisticated computation, so that a simple threshold comparison can be implemented directly in the hardware to reduce the communication overhead between the OIM-chip and host system. However, in scope of this evaluation, the phase groups are not repeatable enough for this approach as multiple parameters, which should be evaluated, affect the phase groups. Having to repeatedly calibrate this phase threshold is too time consuming and potentially error-prone.

The identification method finds the two distinct phase groups, which are typically spread among two to three consecutive bins, and calculates their mean. The threshold is then defined by the middle between both identified phase groups. However, this method breaks down if the phase groups can not be identified, which is usually the case at suboptimal operating parameters. If the locking does not overcome the coupling, for example, because the locking is too weak or the frequency difference is too high, the phases are spread over many phase bins. Since the parameter sweeps for evaluation likely include such suboptimal operating parameters, this method is not well suited for a reliable operation.

The brute-force approach is used for evaluation in this thesis. The objective at all possible phase thresholds is calculated and then the best objective is selected. Although this is computationally more expensive, it provides robust results. Even if the measured phases do not have clearly distinguishable phase groups, it leads to reliable results. However, such a brute-force approach is undesirable for high-performance computing, but its robustness makes it well suited for the experimental evaluation. It should be noted, that this approach could lead to slightly better results than the other two. When the OIM computing has two clearly distinguishable phase groups, the brute force yields very similar results to the other methods. Often, the objective is identical, although occasionally the brute-force approach provides a slightly better value. In such cases, one or more oscillator phases are in the middle between both groups close to the threshold. As noticeable in Figure 6.4, there are few oscillators in the phase bin at the threshold. The phase group identification method does not



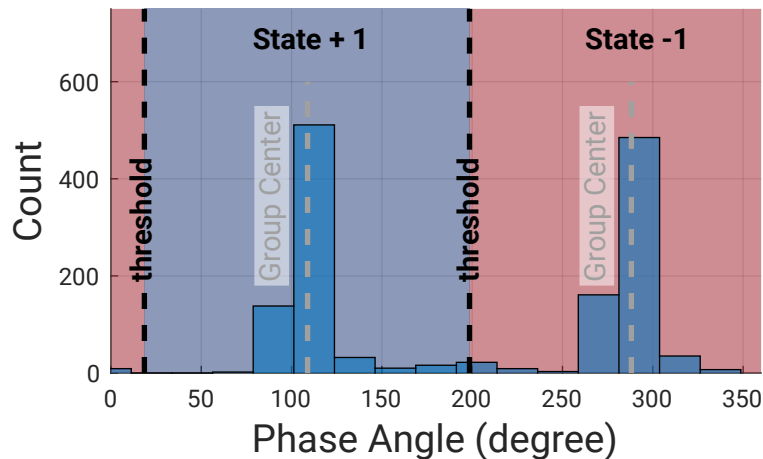


Figure 6.4.: Exemplary phases as an outcome of a computing run illustrating the state extraction. The centers of the +1 and -1 state groups are marked in gray and the threshold, which is set between the centers, is shown as a black dashed line.

consider these outlier bins. Hence, their assignment is more or less random. The brute-force approach always selects the better of these two possibilities. Empirically, the differences are very small, since these nodes at the threshold hardly change the objective of the optimization problem. Usually, they lead to the same objective independent of their state being +1 or -1. If the objective is affected, than its changes are very small. In the case of poor operating parameters leading to unclear phase groups, there can be a considerable difference between these approaches. In the extreme case with hardly any distinct phase groups, the brute-force approach will provide much better results. Hence, it should be considered that a breakdown of the computing principle is less pronounced for the brute-force approach than for the phase-identification. However, for the evaluation it is sufficient to detect a breakdown of the computing accuracy. A precise quantification of the reduction in accuracy is much less important then.

### 6.3. Benchmark Problems

The benchmark problems for the evaluation of the fabricated dies were randomly generated. To avoid the time-consuming embedding, the problems are generated to fit the actual network topology of the chip by design. This leads to a representative benchmarking of the chip, because all available edges of the full topology are potentially used. The procedure of the random problem generation is shown in Figure 6.5. It starts from a graph that includes all physically available nodes and edges, representing all oscillators and couplers in the hardware implementation. Figure 6.5a illustrates this using a small exemplary King's graph, which is the network topology used in Firefly. Then, a targeted edge density is randomly chosen, which determines the number of used edges out of the available edges. The random problems were restricted to contain 30% to 70% of all physical available edges. Edges are randomly chosen and removed from the network until this targeted number of edges is reached, as shown in Figure 6.5b. An optional step could check if the graph becomes disjunct by the removal of that edge and prevent such operations. After removing the target number of edges, the randomly generated graph is obtained as shown in Figure 6.5c.

The basic connectivity for the four different benchmark sets of the Glowworm tapeout are shown in Figure 6.6. They include only certain features of the network to allow a specific evaluation of these. The actual problems are then randomly generated from that connectivity as discussed before. The simple problem

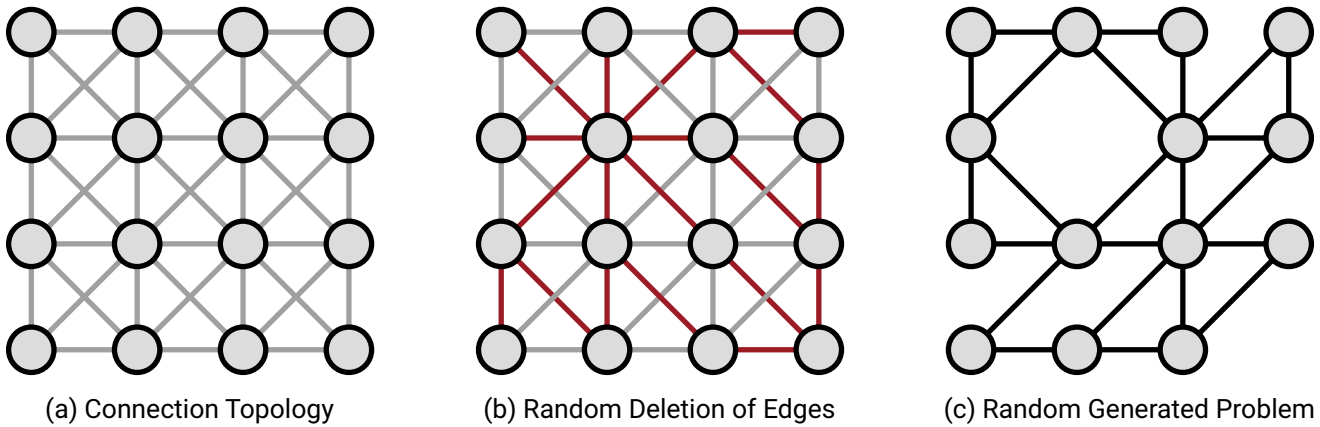


Figure 6.5.: Principle of the topology aware random benchmark problem generation. (a) The process starts from a graph containing all physically available edges marked in gray. (b) Edges are randomly selected for removal, which are marked in red. (c) The finished randomly generated graph.

just consists of the connections of the direct horizontal and vertical neighbors as shown in Figure 6.6a. By construction, a checker board pattern is an optimal solution when restricted to positive weights<sup>†1</sup>, where every edge can be cut. This is therefore a suitable benchmark to verify the general coupling and the ability to synchronize across the whole chip. The `difficult` problem contains all local connections to 11 neighbors as shown in Figure 6.6b. In case of the first generation Firefly this local connectivity just resembles a King's graph, providing 8 connections to all horizontal, vertical and diagonal neighbors. The `bias` set, which is only applicable for Glowworm, adds the spin-bias connections as shown in Figure 6.6c. Finally, the `weighted` set uses the full weight resolution of 4-bit (Glowworm) and 6-bit (Firefly) instead of the default edge weight of 1.0. The 1440 node benchmark sets used for benchmarking Glowworm and the 400 node benchmark sets used for Firefly are available at Figshare [132]–[136].

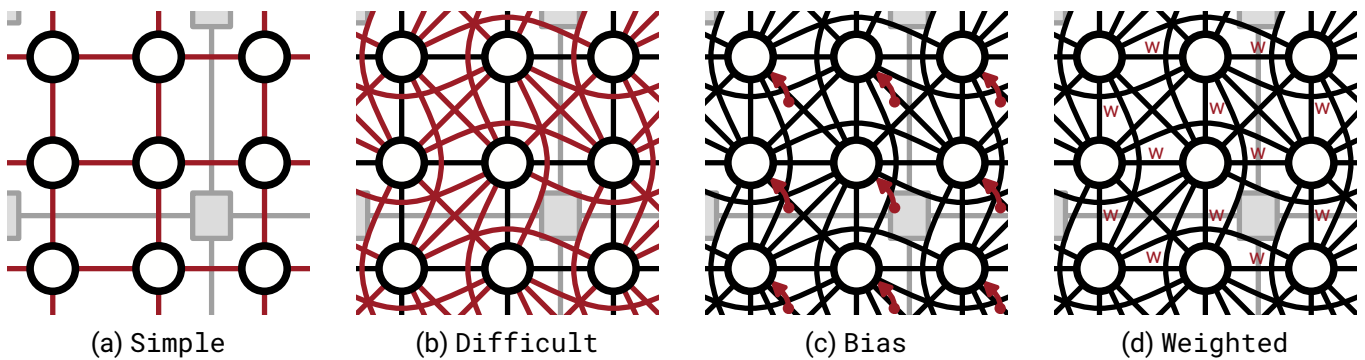


Figure 6.6.: Visualization of the connectivity of the different problem sets. The changes of the connectivity between the problems from left to right are highlighted in red. For the weighted problems just some some weights  $w$  are highlighted for better visibility. However, all edges have random weights.

<sup>†1</sup>The weights are positive when interpreted as a maximum cut problem. Transformed into the Ising model this means negative  $J_{ij}$  weights, respectively anti-phase coupling.

---

The commercial software Gurobi [137] was used to obtain the optimal solution for each problem. Thereby, the software proves that the found solution is optimal. Solving the 400-node difficult benchmark problems for Firefly took an average of 1.02 s per problem and a maximum of 12.87 s on an Intel Core i7-7700K desktop computer. The 1440-node problems already took 1415 s on average with a maximum of 4.67 h per problem on a more powerful 24-core AMD Ryzen 5965WX system. There are considerable differences in runtime between the individual problems as the minimum time for one problem of this set was just 3.37 s. Because the simple problems have the checkerboard pattern as optimal solution, those were solved quicker by a factor of approximately 10,000 than the difficult problems. The runtime should just provide some sort of estimation of how hard these problems are for state-of-the-art approaches. It needs to be noted, that the solution was guaranteed to be optimal as required for the benchmarking. Hence, it would not be a fair comparison, since guaranteeing an optimal solution and reaching a non-optimal solution massively vary in their difficulty.

## 6.4. Embedding

Although previous work on the embedding was done [167], [170], [156], the developed algorithms are not capable enough for the size of the fabricated coupled oscillator network. The algorithms do not provide a valid embedding within a reasonable time. Hence, the embedding was avoided for the experimental setup by using the chip-specific benchmark problems. Although using purposely generated benchmark problems restricts the comparability between different systems, a meaningful comparison is already limited by the network topology of the OIM system, which greatly varies between publications. A capable minor embedding algorithm would be desirable, but its lack does not compromise the scientific findings of this work. Since the goal is to evaluate the computing principle and find hardware implementations, benchmarking with the above-mentioned approach is well suited to demonstrate and evaluate the performance.

When using an embedding algorithm, the computing performance could not be only affected by the hardware but also by the algorithm. As long as an algorithm just assigns each discrete variable to exactly one physical oscillator, it should not affect the computation outcome. However, if the degree of a node is greater than the degree of the network <sup>†2</sup>, then a discrete variable would need to be mapped to multiple physical nodes and edges. Since the computed problem in the hardware network differs from the actual problem, the computing outcome could be negatively influenced. Thus, the embedding algorithm and hardware would be benchmarked together. While a much more flexible benchmarking approach regarding the supported problems is desirable from a perspective of real-world applications, it does blur the actual performance of the hardware. The development of a high-performance embedding algorithm is outside the scope of this thesis.

## 6.5. Conclusion of the Experimental Setup

The setup is tailored for a flexible evaluation of both designed chips. The adjustable bias rails, timing schedule, and frequencies allow finding the optimal operating parameters for the computation experimentally. Similar to other works, custom randomly generated problems tailored for the network topology are used. This allows to investigate the actual performance of the OIM hardware. A suitable problem transformation and embedding would be needed to solve open-source benchmark sets. This process is outside the scope of this thesis and can have additional influence on the obtained accuracy. Computing results and transient waveforms of the oscillator's behavior are provided as part of the evaluation in the next chapter.

---

<sup>†2</sup>In other words: The number of connections of a discrete variable exceeds the number of couplers per oscillator.

---

### Experimental Setup

- The versatile setup is needed to find the **optimal operating conditions** during runtime experimentally.
- **Purposely generated maximum cut** problems of different topologies can evaluate the whole available oscillator network effectively and avoid time-consuming embedding. The features, such as weight resolution, spin-bias, and routing connections, are systematically targeted.
- This setup lacks a **powerful embedding** to test other open-source benchmark sets and allow a wider comparison with different Ising machines based on other physical implementations. However, the complexity of the embedding is outside the scope of this thesis.

---

## 7. Performance Evaluation and Discussion

---

This chapter presents the experimental evaluation of the two fabricated prototype designs. The experiments give a well-founded insight into the actual performance, including real-world factors such as noise and manufacturing mismatch. Time-consuming (if possible at all) simulations, as discussed in Chapter 4, are avoided and there is no risk that simplifications might not or just inaccurately model the actual behavior. Additionally, the results are used to analyze the impact of random manufacturing mismatch, which is unavoidably present in the fabricated chips. It is essential that parameters such as the strength of the coupling connections or the strength of the SHIL are carefully chosen for the best computing performance. Operating an OIM system with suboptimal parameters can significantly reduce the accuracy or even break down the phase-based computing. Additionally, the statistical properties, such as the correlation between oscillators as well as the obtained solutions to the optimization problems, are evaluated. Using an experimentally driven approach, a sufficient number of data points for a statistical evaluation can be obtained. Unless otherwise noted, all presented results are obtained with the 2<sup>nd</sup> generation design Glowworm. Although only a limited number of 10 dies per design are available, these samples give an impression of the expected behavior of the proposed systems.

First, the general operation of the oscillators is presented using experimental data and waveforms in Section 7.1. Then, the evaluation methodology is explained in Section 7.2. The inherent randomness of the computing, the impact of operating parameters such as the coupling strength and SHIL, and the influence of mismatch is discussed in Section 7.3. Special attention is given to the spin-bias connections implementing the  $h_i$  terms of the Ising model in Section 7.4. Coupling via the routing channels is discussed separately in Section 7.5. Benchmarks are provided in Section 7.6 before the comparison with other works in Section 7.7. Finally, the future opportunities for OIM are discussed in Section 7.8 and the evaluation is concluded.

### 7.1. Fundamental Operation

Before the start of the actual evaluation of the computing, experimental examples are provided to illustrate the fundamental operation. For a single computation, the oscillators cycle through three possible modes of operation: free-running, coupled, and SHIL-locked. In the free-running mode, all coupling circuits are disabled to prevent any interaction between the oscillators. When coupled, the configured coupler circuits are enabled to create the intended interaction between the oscillators based on the optimization problem. The SHIL state locks and discretizes the phases by gradually ramping up and holding the strength of the injection, while the couplings remain enabled. The phases are read-out within this hold period, because it ensures proper discretized phase groups. The following Section 7.1.1 shows the waveforms of up to 4 oscillators, which are captured by a Keysight DSAV164A oscilloscope with a sampling rate of up to  $80 \text{ GSa s}^{-1}$ <sup>†1</sup>. Afterwards, an example of an illustrative optimization problem is shown in the Section 7.1.2.

Due to the limited number of available pins, a single-ended transmission is required. The used off-chip single-ended drivers are ‘digitally’ buffering the oscillator output provided by the D2S. The internal buffering

---

<sup>†1</sup>If only two waveforms are shown, the full  $80 \text{ GSa s}^{-1}$  are used. For three and more, the maximum sample rate reduces to  $40 \text{ GSa s}^{-1}$

for signal conditioning introduces an additional delay at the observed output. Potential skew between the channels is minimized by keeping the delay of the chip's internal path similar, employing delay matching techniques on the PCB, and using cables of identical length. Although the waveforms might look 'analog', only the zero crossings of the internal differential oscillators are maintained as a transition of the output signal. Duty-cycle imperfections and delay variations of the internal D2S might cause a slight skew of the observed waveform. Any glitches in the waveform are likely caused by cross-talk, presumably from the bond wires and the power supply of these off-chip drivers, which do not provide any phase information of the actual oscillator.

### 7.1.1. Coupling & Locking

The coupling behavior is first demonstrated by the simplest possible example of just a pair of two coupled oscillators. Then the example of a three-node ring with anti-phase coupling is used, as all three couplings in this configuration cannot be satisfied. The resulting fight emphasizes characteristics that are relevant for optimization problems as well.

#### Pair of Oscillators

The in-phase coupling of two oscillators is shown in Figure 7.1. The upper plot shows the transition from the free-running into the coupled mode. The initial phase (and frequency) difference is overcome within approximately 3 oscillator cycles so that the oscillators align their phase (and frequency) as expected in the coupled case. The transition between the coupled and SHIL mode is not that pronounced and requires a more careful inspection of the lower plot. As indicated by the gray staircase, the strength (amplitude of the SHIL

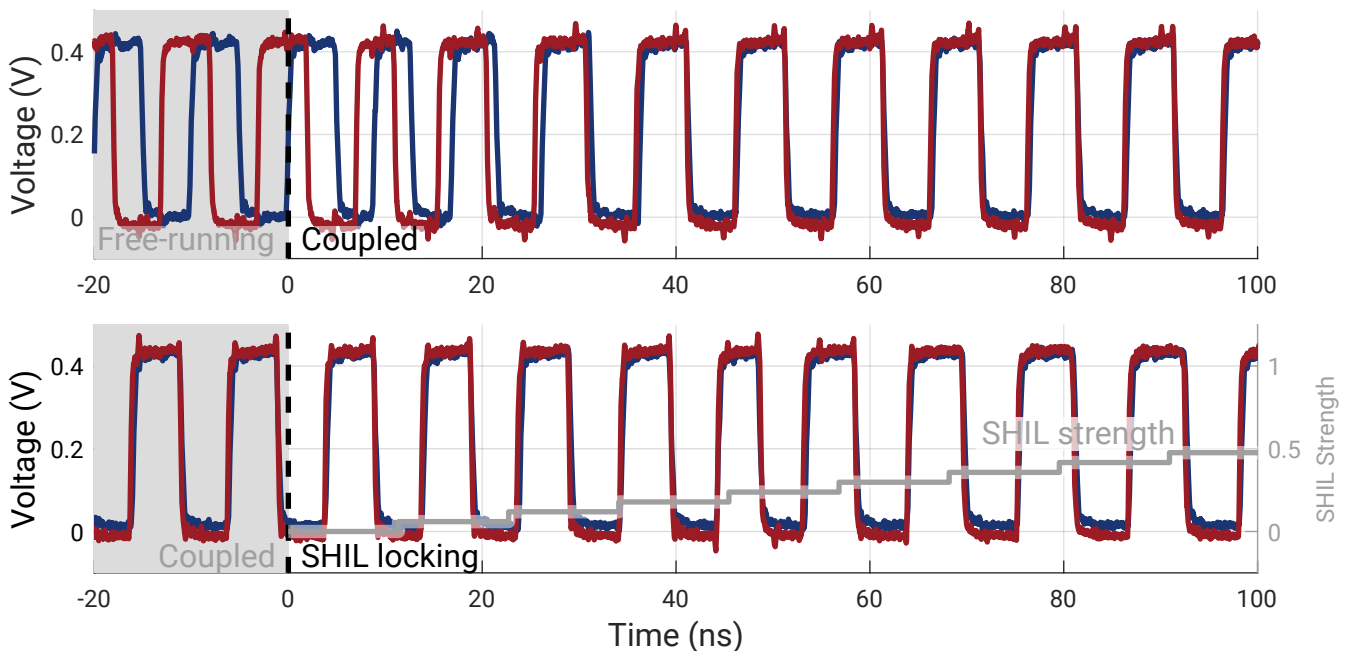
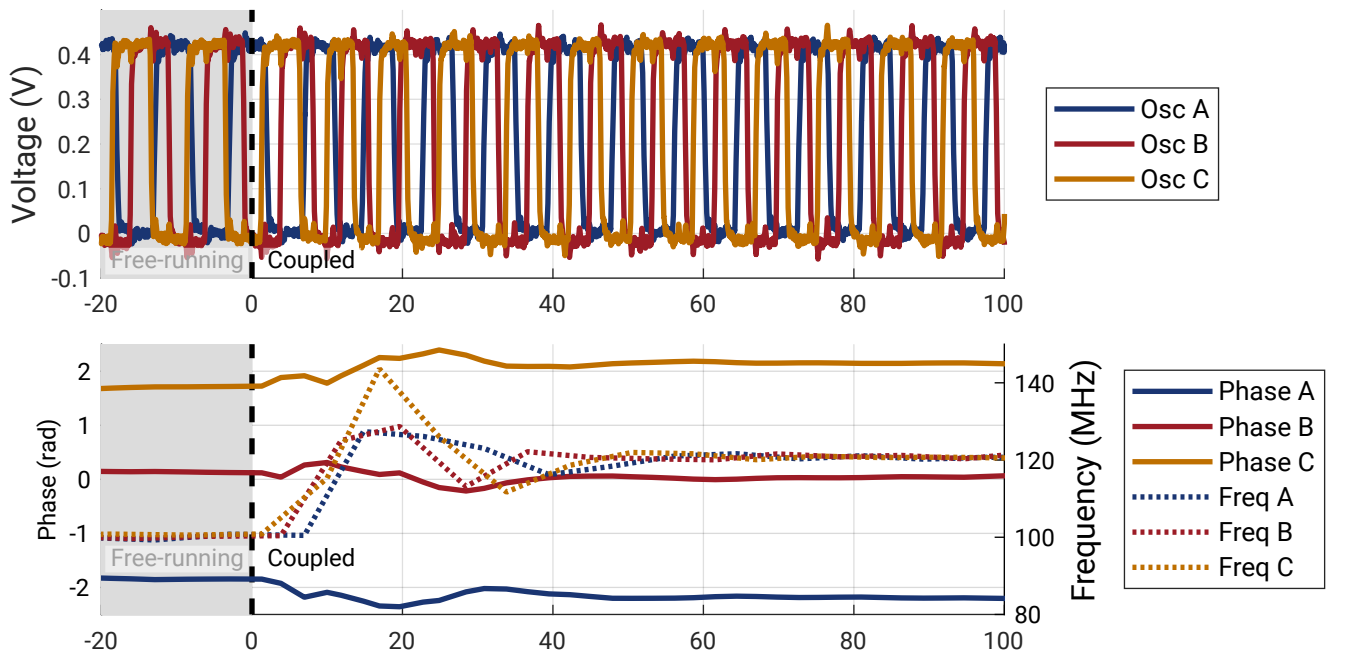
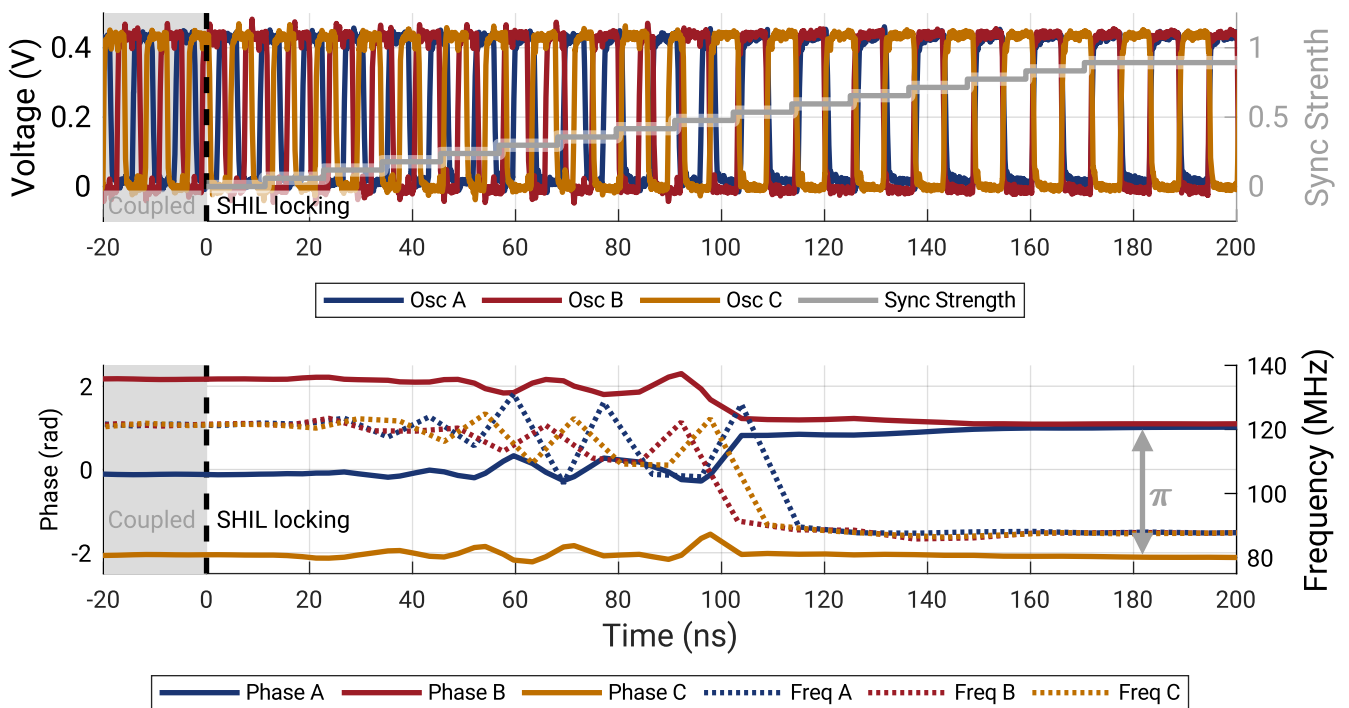


Figure 7.1.: The transient coupling is shown in the upper plot and the SHIL locking process in the lower plot. Since there are no visible changes after approximately 60 ns, the full ramp-up of the SHIL strength is omitted.



(a) Transition Free-running to Coupling



(b) Transition Coupling to SHIL

Figure 7.2.: Transient coupling and SHIL locking behavior for an anti-phase ring topology with three nodes. The phase and frequency derived from the waveform are additionally shown for convenience. (a) Because the initial phases are similar, the absolute change in phase is not very pronounced. (b) The locking forces the oscillators into two discrete phase angles separated by  $\pi$  and adjusts their frequency into the sub-harmonic.

---

with roughly twice the oscillator’s frequency) is gradually ramped-up. Since the oscillators are already in identical phases and oscillating with approximately half of the SHIL injection frequency, the locking process is hardly noticeable in this example.

### Anti-Phase Ring of Oscillators

The anti-phase coupling in a ring of three oscillators is much more distinctive for the oscillator behavior since it is impossible to satisfy all couplings in this configuration. Therefore, the couplers ‘fight’ against each-other, which emphasizes the difference between the operating modes. The transition from the free-running into the coupled phase is shown in Figure 7.2a. The waveform is in the upper plot and the derived phase and frequency in the lower plot. It takes roughly 6 oscillator periods to establish a constant phase with  $\frac{2}{3}\pi$  difference<sup>†2</sup>. The transition from the coupled into the SHIL locked state is shown in Figure 7.2b, with the waveform in the upper plot and the phase and frequency in the lower plot. In order to lock, the SHIL must fight and overcomes the couplers, since only two of the three coupling connections can be satisfied. The frequency adapts to exactly half of the SHIL frequency to match the sub-harmonic.

### 7.1.2. System Level

The three operating modes are illustrated using a simple example problem showing an image of the character ‘G’ (for Glowworm) in Figure 7.3. To ensure a good visualization, even when coupled, all connections of that example problem are satisfiable to ensure that all oscillator can be in one of the two phase groups when just coupled<sup>†3</sup>. During the free-running mode, all phases are in random phase angles, which constantly change due to differences in their free-running frequency. When coupled, the oscillators mutually interact and adapt their phases. While the ‘G’ form is clearly visible in the figure, there is a considerable phase gradient between the bottom left and top right. Although the oscillators should be in exactly the same phase, a considerable gradient of 180° across the whole chip is formed. Since this phase shift exceeds the skew of the internal clock-tree, it must come from the oscillators themselves. Similar phase gradient behavior was also observed in the Firefly design, where it was intensively verified that the gradient is caused by the oscillator coupling. When SHIL locked, the phases are forced into two discrete states, which are clearly visible. While the ‘G’ form is perfectly solved in the shown example, this is not always the case as later discussed in Section 7.3.3. Although initially appealing, just freezing the oscillators by applying SHIL might become an issue due to the continuous phase gradient of more than  $\pi$  before freezing. Unfortunately, the dynamic transition from the coupled to the locked state cannot be measured for more than 4 oscillators by the external oscilloscope. More complex optimization problems usually cause continuous phase distributions, in which the oscillators cannot be as clearly assigned into the discrete states. This underlines the necessity of the SHIL, which has already been shown in theoretical work [5], [111].

---

<sup>†2</sup>The desired phase difference of  $\pi$  is not possible. Hence, it will result in a lower phase difference. All three oscillators are identical and the structure is therefore completely symmetric, which leads to the difference of  $\frac{2\pi}{3}$ . The frequency increases by roughly 20% in the coupled state, which, however, depends on the coupling strength.

<sup>†3</sup>The internal PDC requires a specific frequency ratio between its clocks. A measurement takes several clock cycles for a conversion. If an oscillator would change its frequency or phase angle during the measurement period, this would cause a faulty reading with the PDC. Since the oscillators are SHIL locked during normal operation, this is not an issue. However, it restricts the possibility to measure phases in the coupled state, which can fluctuate. Since defining and measuring a phase in a fluctuating system might not be very meaningful, this is only an issue for this specific demonstration.



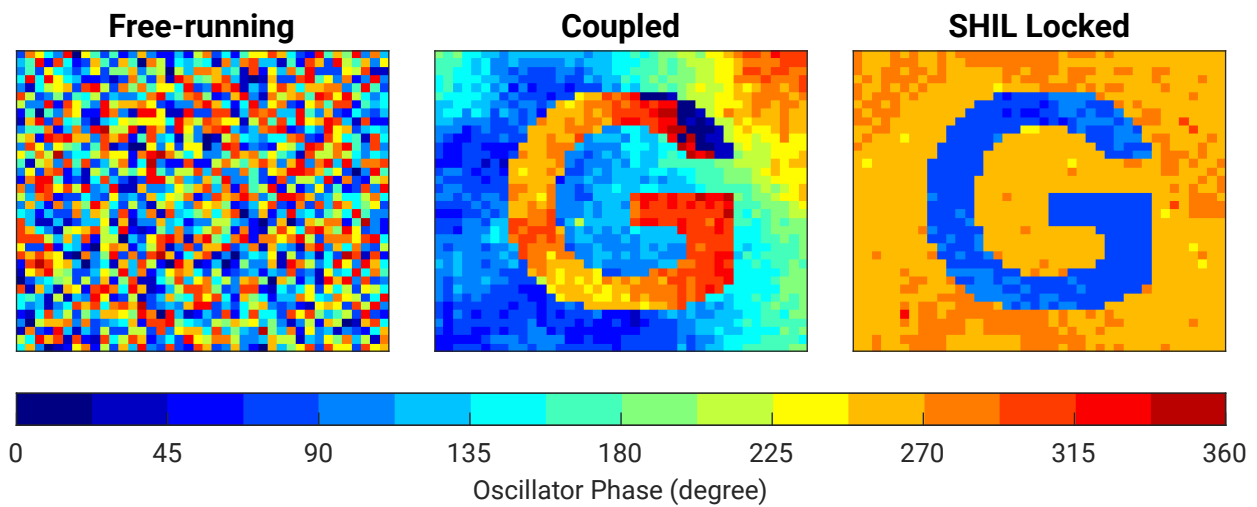


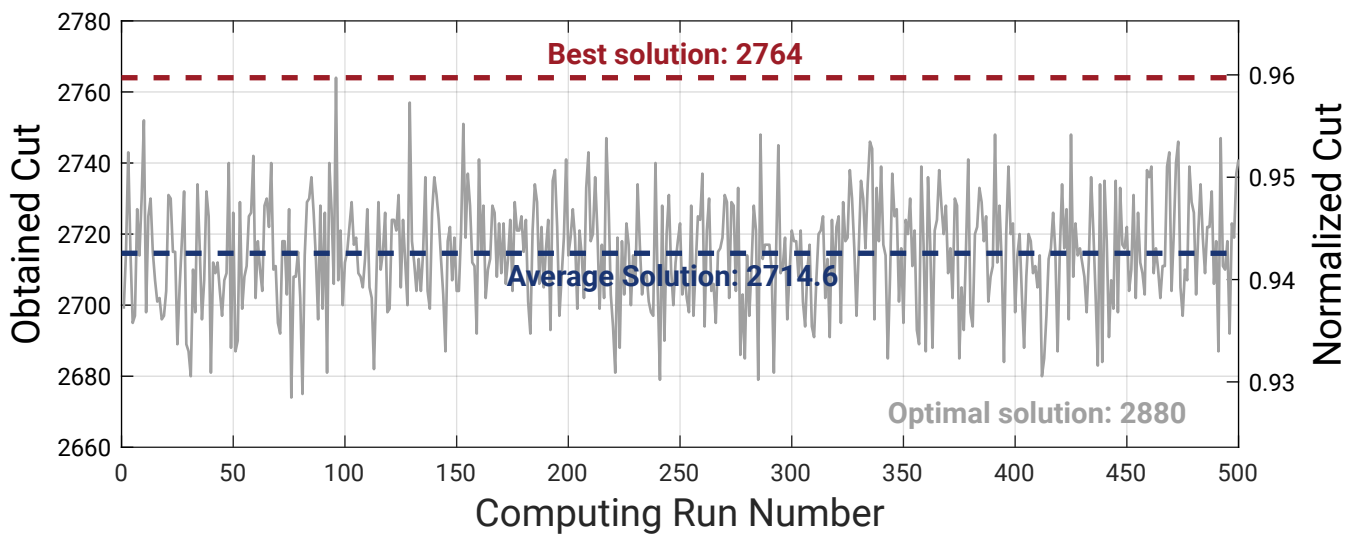
Figure 7.3.: Illustration of solving an exemplary problem. The example problem of a 'G' is generated from a black-and-white image. The coupling connections between pixels are configured to be in-phase if the corresponding pixels have equal colors or anti-phase if the corresponding pixels have different colors. This is applied for all available horizontal, vertical and diagonal neighbors.

## 7.2. Evaluation Methodology

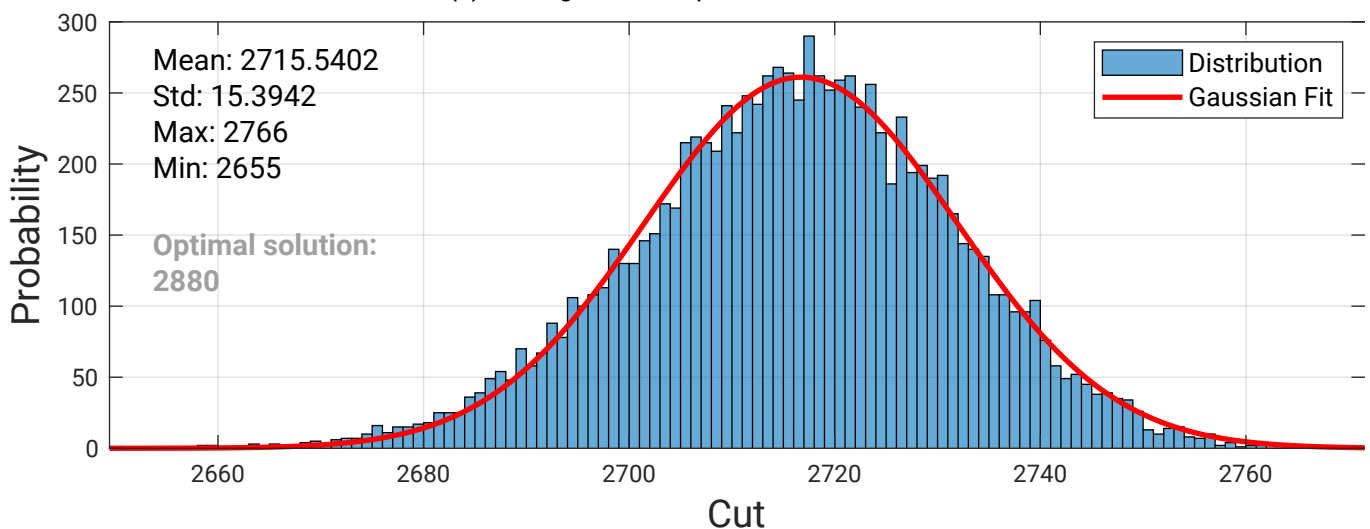
Current OIM systems often do not find the optimal solution to a problem. Even if the optimal solution is found, they do not offer any guarantee that the found solution is optimal. Hence, a methodology is needed to judge how close the obtained solution is to the problem's global optimum. Additionally, the outcome of an OIM computation is non-deterministic. Mainly, noise and possibly chaotic computing<sup>†4</sup> cause different solutions when repeatedly computing the same optimization problem. Therefore, a meaningful and reproducible metric to summarize the performance is necessary. The obtained results could be treated as a random distribution. Hence, just taking a single sample is not sufficient to judge the actual expected performance.

An example of the computing outcome is provided in Figure 7.4a. The same problem, which is treated as a maximum cut problem, is repeatedly solved a total of 500 times, where one computation is referred to as a 'run' in the following. The left y-axis shows the obtained cut. For the demonstrated problem, the maximum cut is 2880. To allow a better comparison of the problem specific cut, the normalized cut (right y-axis) is used, which divides the obtained cut by the maximum cut. For maximum cut problems with just positive weights, possible cuts range between 0 and that maximum cut. The average cut and the best cut out of those 500 runs are reported to characterize the performance. Although the number of taken samples influences the best cut, it provides an estimation of the upper limit. Solving the same problem repeatedly and picking the best solution could also be a realistic scenario, where a better solution is more important than time and energy savings. Hence, the best solution was favored instead of the standard deviation as additional metric. The distribution of the obtained cut is shown in Figure 7.4b, which uses a total of 10,000 runs. Their cuts are approximately Gaussian distributed. The similarity between computed solutions is discussed in Section 7.3.2. In the following, each problem will be repeatedly solved 500 times if not explicitly mentioned otherwise. Usually, the mean and best cut of these 500 runs will be reported.

<sup>†4</sup>The phases of the oscillators evolve from their initial phases in the free-running state. This process tends to be very sensitive to the starting phases. So even small changes will likely cause dissimilar solutions.



(a) Solving the same problem 500 times



(b) Distribution of 10,000 runs

Figure 7.4.: Exemplary computing and the variation between individual runs. (a) Solving the same optimization problem repeatedly 500 times. The obtained cut varies, where the best and the average are used as characteristics. (b) Distribution of 10,000 runs of solving the same problem. The individual cuts are approximately gaussian distributed.

### Other Methodologies

Another commonly reported way to judge the quality of the solution is to quantify the probability that the solution exceeds a certain threshold. For example, the work [130] reports the probability that a solution better than 95% of the optimum is reached. Other thresholds, such as 99% or similar, are also used [127]. This work decides against such a method. While it provides an excellent comparison between approaches with very similar precision, it is not suitable for substantial differences. To provide any meaningful comparison criterion, the system must reach the threshold. If the threshold is either never or always reached, there is just limited insight provided by such a characteristic. Taking the example from Figure 7.4b, the range of the

obtained individual solutions varies between 92.18% and 96.04%. Especially for the later analyzed behavior, which also considers where the computing principle breaks down, such a strict border is not well comparable. As soon as the computed solutions do not reach the threshold anymore, no information is gained about the actual precision. Nevertheless, the reported mean and best run can be used to roughly estimate the probability of reaching a certain threshold.

### 7.3. Operating Parameters

This section investigates the operating parameters of the OIM. It starts with the randomness of the initial phases. Then the coupling, including strength and mismatch is analyzed. Afterwards, the SHIL, coupling time, and oscillator frequency mismatch are considered.

#### 7.3.1. Initial States

The operation of OIMs is, in general, affected by noise. Additional, oscillators might have a potential undesired mutual influence. The phases might be affected by, for example, disturbances of the power supply or by (capacitive) coupling of nearby physical traces. This section statistically evaluates such phenomena by analyzing the correlation between neighboring oscillators. Then, the time needed for an oscillator decorrelate its state is investigated.

Both fabricated chips do not have any method to control the initial phases of the oscillator start-up. Exemplary waveforms of the start-up are shown in Figure 7.5. The oscilloscope was triggered by the enable signal of the oscillator at time  $t = 0$  and the waveform of a single oscillator was captured. This procedure is repeated ten times for the same oscillator. As it is clearly visible, the start-up is not reproducible. Thus, the initial phase is not predictable. However, the initial phases seem to be not truly uniformly distributed, but have a slight preference.

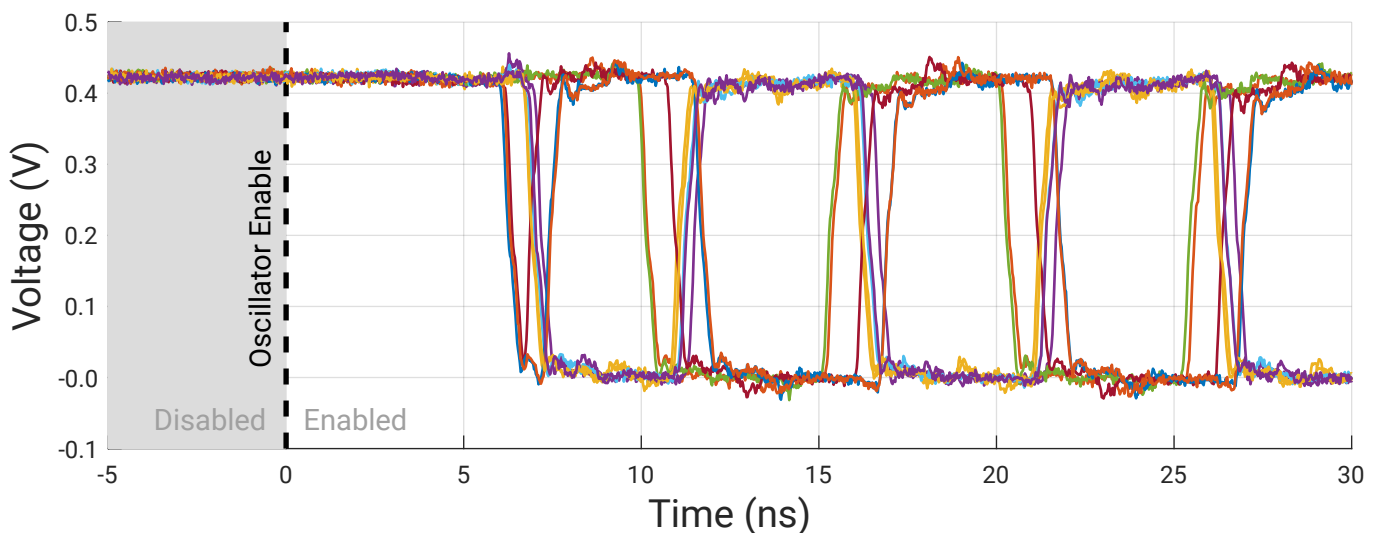


Figure 7.5.: Waveform of the oscillator start-up recorded by the Keysight oscilloscope. 10 start-up waveforms of the same oscillator are recorded. The coupling is enabled at  $t = 0$  and the oscillator starts shortly afterwards. As should be clear, the initial phases are not repeatable.

## Correlation between Neighboring Oscillators

Another relevant property of the designed system is the correlation between (neighboring) oscillators. Although the initial states might be random, there could be a tendency for oscillators in the proximity to have the same (or opposite) phase. Intuitively, it should be more likely that oscillators in close proximity exhibit a correlation of their phase. This can be confirmed experimentally as shown in Figure 7.6, where a distinct reduction of correlation with increasing distance is visible. Especially for direct neighbors (distance = 1), there is a correlation. However, it might be considered small. This is also emphasized by the histograms shown in Figure 7.7. For distances of 5 or greater, the calculated correlation matches the expected coefficients of two truly independent binary variables. Therefore, the data suggests that there might be a very small undesired physical interaction such as crosstalk between close neighbors. The practical relevance, however, should be small as already 95% of next-neighbors have an absolute correlation coefficient of 0.12 or less. The first chip generation did not show any noteworthy correlations at all.

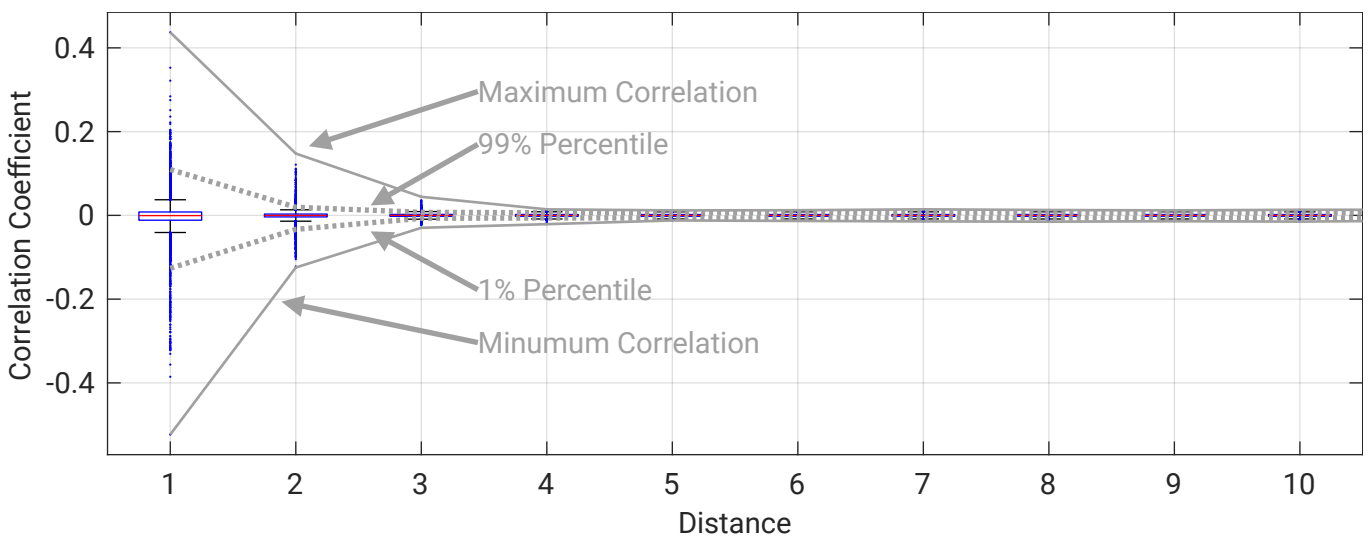


Figure 7.6.: Boxplot of the correlation coefficients of the initial oscillator phases. The box contains 50% of all samples and the medians are noted by the red lines. The correlation coefficients are shown for each physical distance between oscillators, defined as Manhattan distance.

## Decorrelation when Free-Running

When the initial phases of the oscillators are desired to be random after a computation, they can be simply let free-running. Due to inevitable differences in their free-running frequency and noise, their phases will drift. Especially lower frequency noise components in the bias current mirrors can lead to random variations of the frequency and thus phase. By simply letting the oscillators run, new random phases can be obtained. The decrease of the auto-correlation versus time is shown in Figure 7.8. The autocorrelation significantly reduces after a few hundred nanoseconds and can be considered decorrelated after 1200 ns. So, after letting the oscillator run freely for 1200 ns, their states are random and independent of their previous state. It should be noted, that no auto-correlation of 1 could be experimentally observed, which would be expected for very small time intervals. One reason is that the free-running duration can only be set in discrete steps of 8.33 ns, which does not align to the oscillator period of 10 ns.

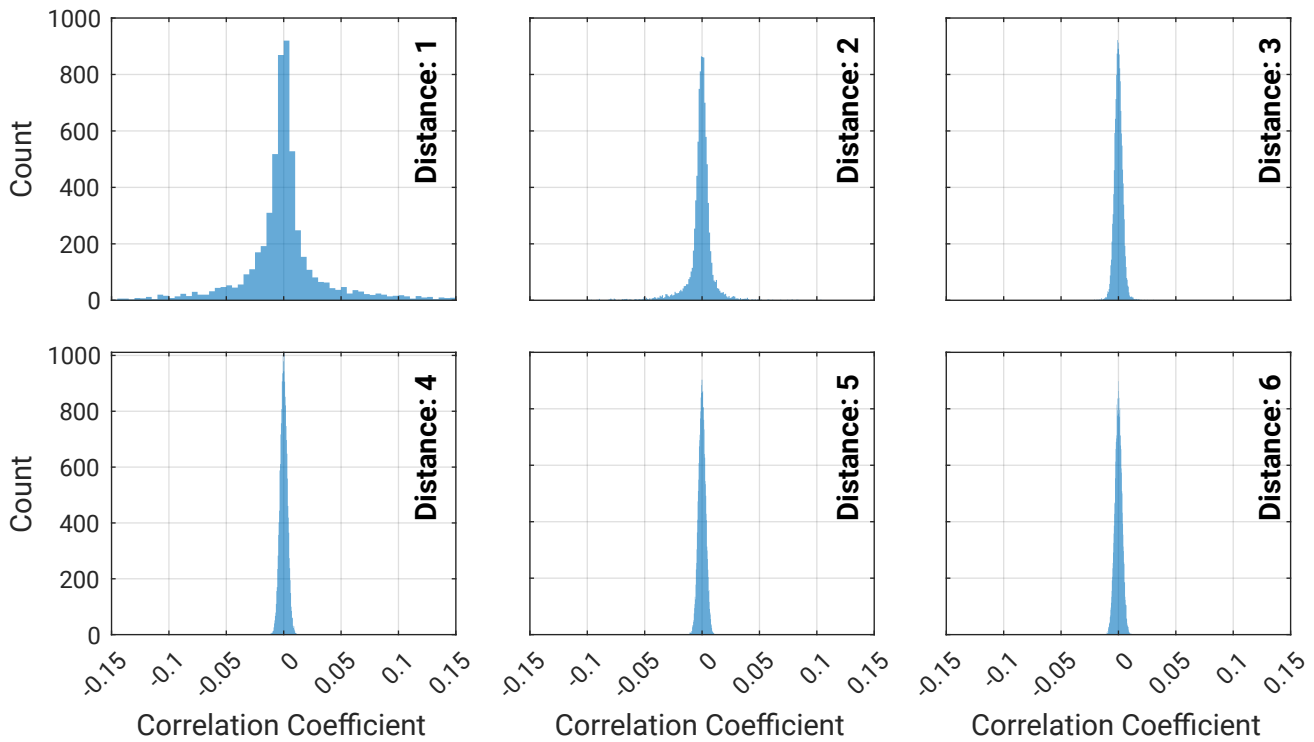


Figure 7.7.: Histogram data of the correlation coefficients sorted by the Manhattan distance of the oscillators on the OIM grid. All histograms have the same scale for better comparability. However, the tail of the distribution at a distance of 1 is clipped. Data of these outliers are available in Figure 7.6.

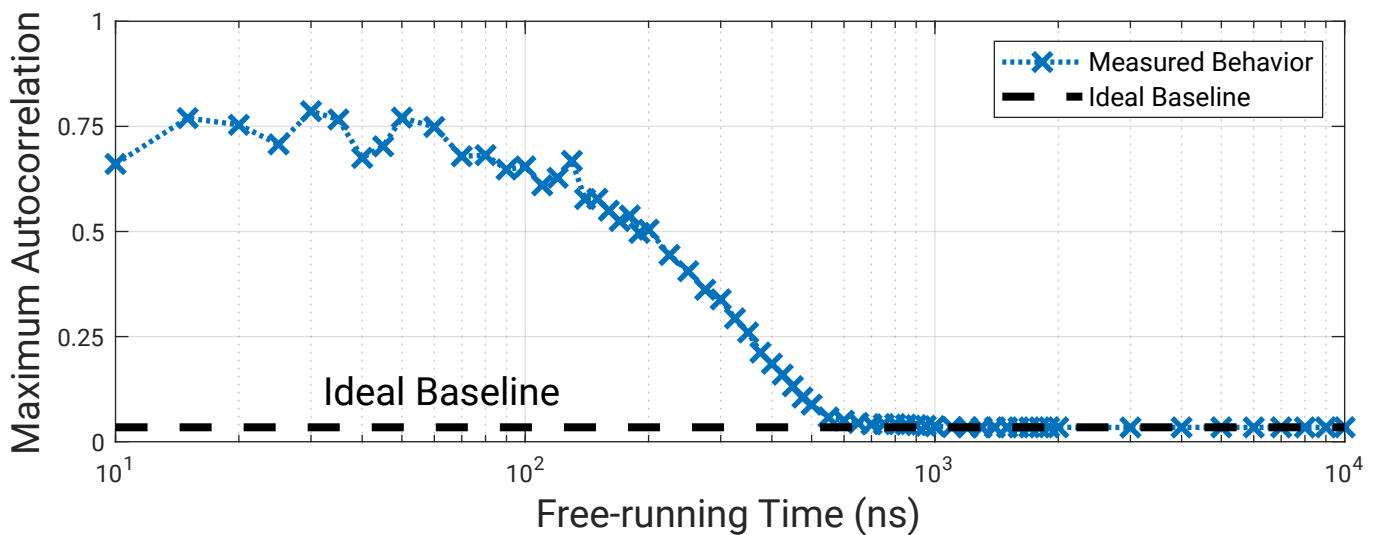


Figure 7.8.: Evaluation of the needed free-running time so that the oscillator phase groups (spins) randomize. The maximum absolute autocorrelation is chosen as a measure of the similarity between consecutive spins of the same oscillator. The ideal baseline shows the expected outcome when applying the same methodology on an ideal random variable, so that any results matching this baseline can be considered uncorrelated.

### 7.3.2. Randomness of the Computing

As already shown in Figure 7.4, doing a repeated computation of the same problem with identical operating conditions leads to different results. Therefore, the obtained solution is non-deterministic and randomly varies between runs. One factor of the variations is the randomness of the initial states as previously discussed. Since the system will evolve from these states, the found (local) minimum in which the oscillators will eventually settle can change. Additionally, noise in the devices can affect the computation itself. During the oscillator coupling prior to the SHIL, any noise can influence their continuous phases. Therefore, the obtained result after locking can be influenced. Once locked, the noise will likely not be able to change the phases anymore. Therefore, the obtained result after SHIL locking can be influenced, although noise will likely not be able to change any phases when locked. Since the Kuramoto model can show chaotic behavior [138]–[140], OIMs might exhibit chaotic behavior as well. However, the imperfections of the actual circuit will lead to slight differences to the ideal Kuramoto model. While it is likely that OIMs could show chaotic behavior in principle, any mathematical proof is outside the scope of this thesis.

In order to quantify the difference between two solutions, a modified version of the Hamming distance is used. The Hamming distance itself is defined as  $\Delta(x, y) = |\{x_i \neq y_i, i \in \{1, \dots, n\}\}|$ , where  $x_i$  and  $y_i$  are the elements of two vectors with identical size. Applied here,  $x$  and  $y$  are the solution vectors  $\sigma \in \{+1, -1\}^n$  assuming  $n$  discrete variables, respectively, the number of oscillators. When considering only the oscillator-to-oscillator connections  $J_{ij}$  and assuming  $h_i = 0$ , there are always two equivalent solutions. The states  $+1$  and  $-1$  can be swapped without changing the Ising Hamiltonian respectively the cut of a maximum cut problem. Hence, a modified version  $\tilde{\Delta}(x, y) = \frac{2}{n} \cdot \min(\Delta(x, y), \Delta(x, \bar{y}))$  is used, where  $\bar{y}$  denotes the inverted state of  $y$ . Because it considers the minimum distance to any of the equivalent solutions, which have the maximum possible Hamming distance between each other, the modified version uses a scaling factor of  $\frac{2}{n}$ . A value of 0 indicates an identical solution and a value of 1 indicates the maximum possible difference. As shown in Figure 7.9, the computed solutions are vastly different. The most similar pair out of 10,000 computed solutions still has a modified Hamming distance of over 0.6, indicating strong dissimilarity between the solutions. When compared with the Monte Carlo sampled vectors, it is clear that the computation results tend to be more similar than the pure random results.

Since there is no direct control over the initial phases for either design, their contribution to the randomness of the computing can not be isolated. However, different approaches were conducted to get evidence if the initial phases have a considerable impact on the computing. By activating the oscillator coupling already at their start-up, the initial phases tend to be more repeatable<sup>†5</sup>. Additionally, the computing can be either continued from the previously computed state by just releasing the SHIL while keeping the oscillators coupled or from a new state by setting the oscillators free-running between the computations. Under all of these conditions, we did not find any noteworthy difference in the similarity of the computed solutions. These can be seen as an experimental hint, that the computing itself has inherent elements of randomness and does not only rely on the initial phases. However, this behavior could be design-specific and might not apply to OIMs in general. It is possible that other OIMs, which use weaker coupling, might depend on the initial states. At least in simulations excluding noise, the computing continuing from the same initial state tends to improve the obtained solutions.

---

<sup>†5</sup>The phases are not strictly repeatable but show a tendency, as detailed in Section 7.3.1 and Figure 7.5.

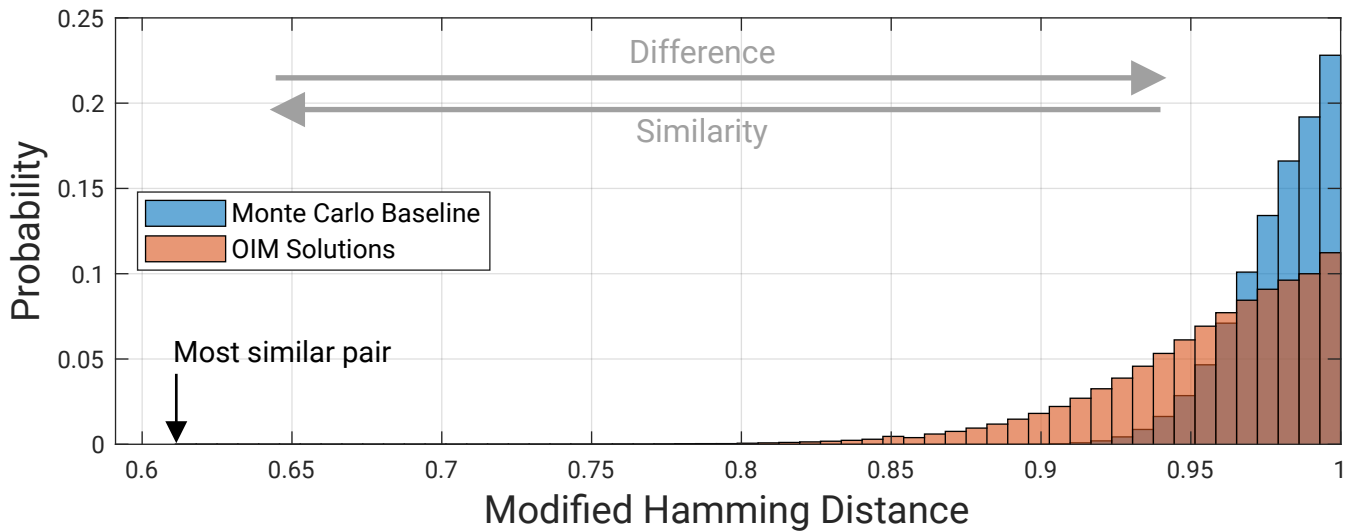


Figure 7.9.: Distribution of the modified Hamming distance when the same problem is solved 10,000 times. A value of 0 indicates identical results, 1 completely different results. As a baseline, the modified Hamming distance is computed using 10,000 Monte Carlo samples of random state vectors.

### 7.3.3. Coupling

The investigation of the coupling focuses on two aspects. The coupling strength can be scaled so that all connections cause weaker or stronger interactions between the oscillators. Additionally, the unavoidable random manufacturing mismatch between individual coupling connections is analyzed.

#### Coupling Strength

While the coupling strength was discussed during the design, choosing the exact strength was postponed to an experimental evaluation. To avoid time-consuming and potential erroneous simulations, the coupling strength can be adjusted during runtime. So, it can additionally compensate for PVT variations and even be individually adapted to the actual optimization problem or its connectivity. The impact of the coupling strength is illustrated in Figure 7.10. The coupler strength, which is defined as the measured average bias current flowing into the DAC, is varied between approximately  $1\ \mu\text{A}$  and  $10\ \mu\text{A}$ . The obtained solution is divided into the simple, difficult, and weighted sets, each containing 100 problems, as discussed in Section 6.3. The boxplots show the distribution of the average solution of those problems. The simple set works best in a wide range of coupling strengths between  $4\ \mu\text{A}$  and  $6\ \mu\text{A}$ , with little influence on the accuracy. The behavior of the difficult set is not as clear because the spread between problems changes significantly with the strength setting. Around  $3.5\ \mu\text{A}$ , the smallest spread between problems is achieved at an already high solution accuracy. However, the highest median is reached at a higher strength of  $4.2\ \mu\text{A}$ , but the spread starts to increase at higher strengths. It should be pointed out that some problems seem to benefit from even higher strength, as some problems perform best at  $6.5\ \mu\text{A}$ . However, other problems seem to get much worse. A similar behavior can be observed for the weighted problems, which has half the coupling strength on average due to the 4-bit weights. Hence, a coupling strength of  $8\ \mu\text{A}$  is equivalent to the  $4\ \mu\text{A}$  point of the difficult set. Here, the lowest spread is reached at around  $7.5\ \mu\text{A}$ , but the best median performance is reached by higher strengths of  $10\ \mu\text{A}$ . However, the spread increases and already 5 problems start to perform much worse, although most problems remain similar.

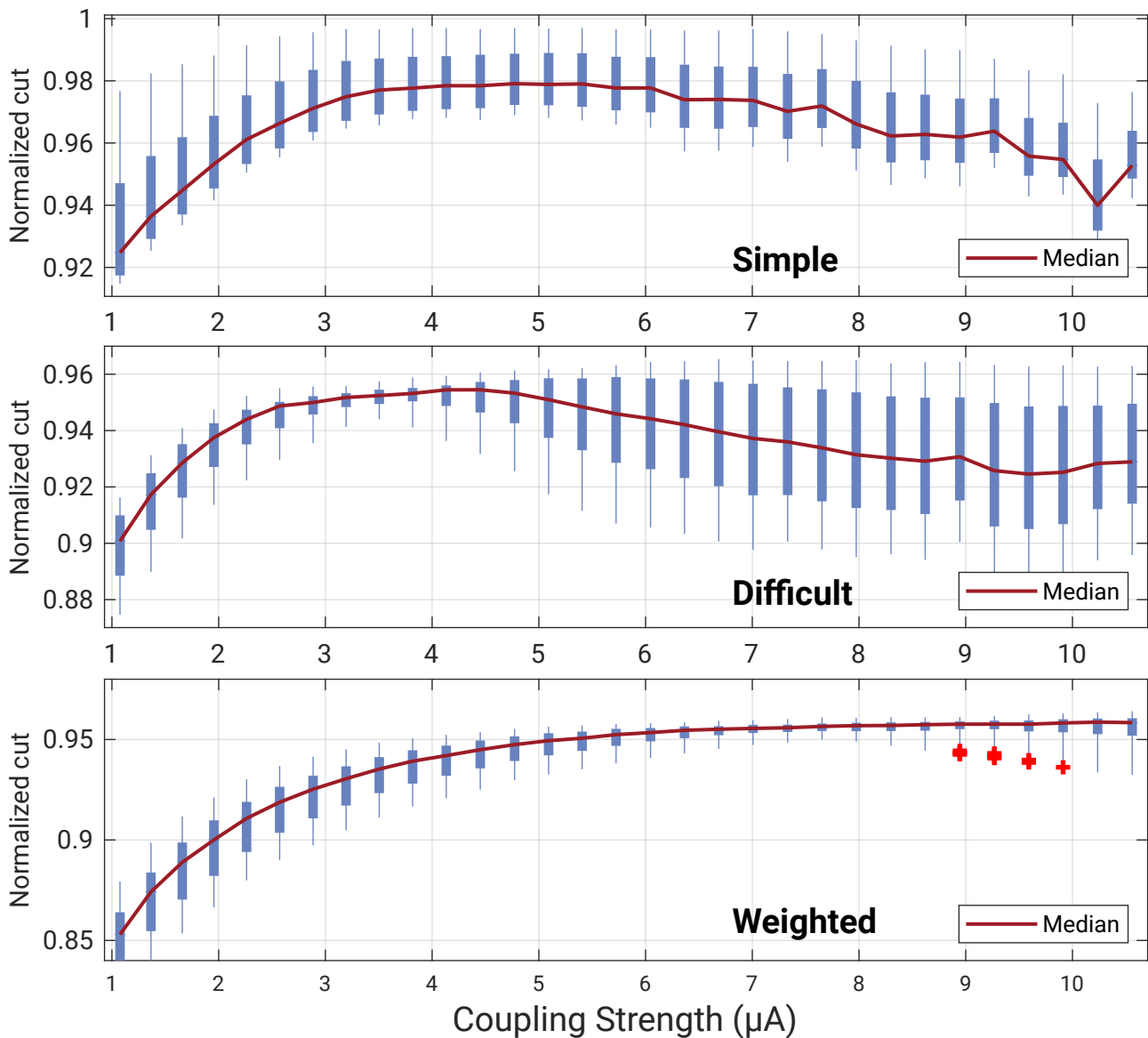


Figure 7.10.: Impact of the coupling strength on the solution accuracy using the simple, difficult, and weighted test set. The boxplots show the distribution of the average solution for the 100 problems contained in each set. The median is highlighted as a red line.

Due to these distinct differences between the problem classes, it makes sense to adjust the strength depending on the structure of the optimization problem. Criteria such as the connectivity per node and resolution of the weights are straightforward to extract from a given problem so that the OIM can be fine-tuned on demand. Depending on the required accuracy of the problem and the available time and energy, the same problem could be tested at different strengths to select the best outcome. Nevertheless, such sweeps must be considered cautiously as the operating parameters are multidimensional and could be interdependent. Other parameters, such as the SHIL strength or frequency, are kept constant during the sweep. So, some coupling strengths might perform better when such parameters would be adapted accordingly.



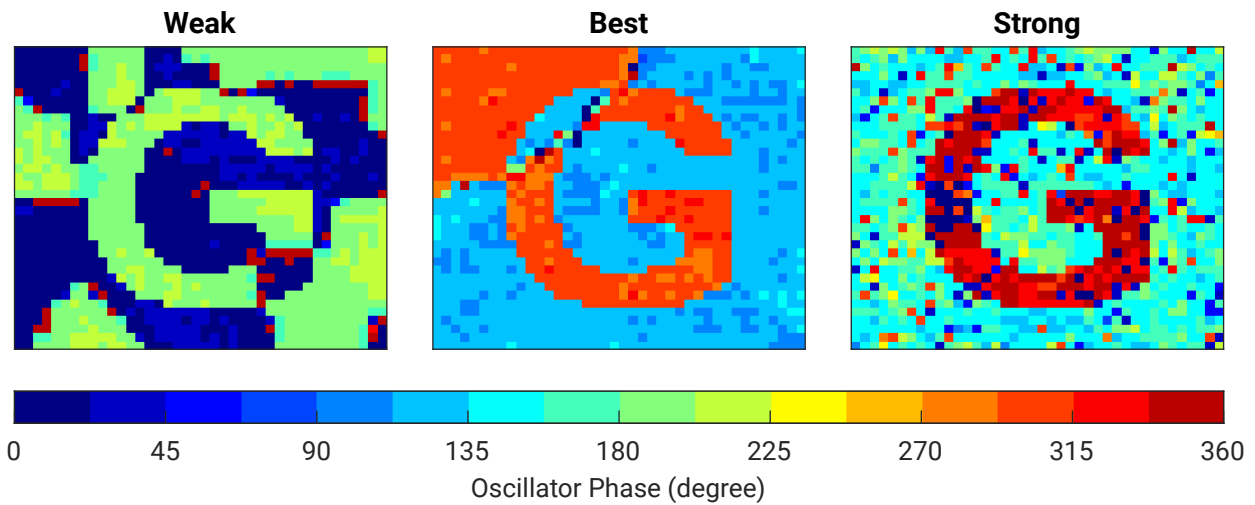


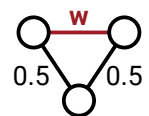
Figure 7.11.: Visual example showing the ‘G’ letter problem with different coupling strengths. Even at the best strength, the problem is often not solved with an optimal solution and a cluster remains.

While the impact of the coupling strength on the settling time and potential phase offsets was previously discussed in Section 5.3.4 during the design, it is unclear how this translates to a full system with large optimization problems. An illustrative example showing how the strength affects the computing outcome is provided in Figure 7.11. It shows the familiar example image, obtained with a weak, a strong, and the best coupling strength. At very small coupling strengths, clusters appear that find the optimal solution locally. One reason is likely the phase gradient in the coupled state, as already shown in Figure 7.3, which is more pronounced at weaker coupling. Groups of oscillators couple around a seed spot, but boundaries to similar neighboring clusters appear. At very high strengths, the coupling is much more dynamic and can quickly change the phase state of an oscillator. The clustering is less pronounced since the phase gradient is smaller, but more faulty spots appear instead. At such spots, individual oscillators are not coupled as expected by the states and connections to the neighbors. However, this could be fixed to some degree by a simple post-processing. At the best coupling strength, the cluster formation is much less pronounced (but can still occur as shown) and these faulty spots are rare.

This analysis emphasizes the advantage of the bias current topology proposed in this work. Competing implementations use fixed elements such as capacitors, transmission gates, tri-state buffers, or inverters, which are connected in parallel to implement different weights. Since their coupling strength is determined by the physical properties (e.g. area of a capacitor, conductance of a material, or width and length of a transistor), the strength is hardly adaptable after manufacturing. In contrast, the coupling strength in this work can be adapted by simply changing the external bias voltage, which is internally converted into the currents for the DACs. This adaption can be used to compensate for PVT variations and can even be fine tuned for the optimization problem topology.

### Coupling Mismatch

Due to random manufacturing variations, each oscillator has varying frequencies and each coupler has varying strengths of the caused interaction. Quantifying the mismatch between the individual couplers is difficult as it cannot be directly measured. Especially the DACs,



which set the bias current for the coupler and thus their strength, are important. The setup to demonstrate the mismatch between couplers is shown in the figure on the right. A ring of three oscillators is used, where two edges have a fixed weight of  $0.5^{†6}$ . The third edge has an adjustable weight, which is varied from 0 to 1. Three cases exist when treating this as a maximum cut problem, as shown in Table 7.1. For  $w < 0.5$  the edge  $w$  will never be cut to achieve the optimal solution  $p_{cut-w} = 0$ . At  $w = 0.5$ , all edges have equal weights, leading to a probability of  $p_{cut-w} = \frac{2}{3}$ . If  $w > 0.5$ , then the edge  $w$  needs to be always cut to achieve the best results, leading to  $p_{cut-w} = 1$ .

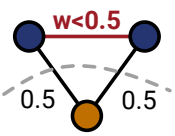
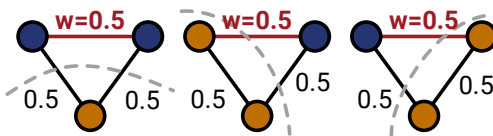
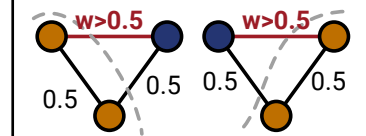
	Weight		
	$w < 0.5$	$w = 0.5$	$w > 0.5$
Maximum cut	1	1	$0.5+w$
Number of optimal solutions <sup>*1</sup>	1	3	2
Solutions			
Probability of cut at edge $w$	0	$\frac{2}{3}$	1

Table 7.1.: Optimal solutions of the anti-phase coupled ring with variable weight  $w$ .

<sup>\*1</sup> Only different cuts between the nodes are considered. Any solution obtained by swapping the states is not considered.

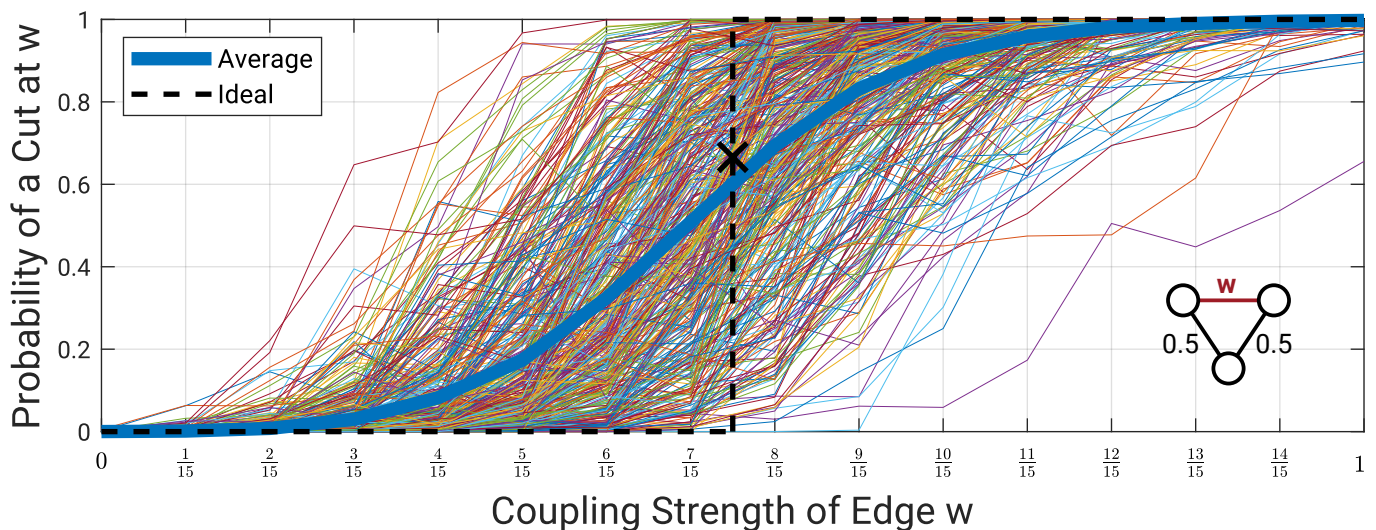


Figure 7.12.: Coupling mismatch illustrated by a ring of three nodes with anti-phase coupling. The probability, that the adjustable edge with weight  $w$  is cut depends on its value and gives a good indication about the sharpness of the transition.

<sup>†6</sup>The internal DACs have a resolution of 4-bit with the maximum strength normalized to 1.0. So, the weight 0.5 is rounded to  $\frac{8}{15}$ .

This setup was conducted for a total of 468 rings spatially distributed on one die. The results are shown in Figure 7.12, where the average over all pairs is displayed in **bold**. When comparing this to the ideal result, the analog behavior does not exhibit such a strong step. Instead, it is a more gradual increase over multiple steps in strength. The transition point, where the edge  $w$  is cut or not cut, varies with the individual rings. Hence, the actual physical weights of the edges vary individually by the random mismatch.

Another test method to judge the mismatch of the coupler with 3 oscillators is also investigated. Instead of forming a ring, the oscillators are connected in a single chain. Then, the two endpoints of the chain are fixed using the much stronger spin-bias coupling connections. One is in the  $+1$  state, while the other is in the  $-1$  state<sup>†7</sup>. The oscillator in the middle couples to both endpoints in-phase. As a consequence, the phase of the middle point is determined by the strengths of those couplers. Since their intended phase is contradictory, they fight against each other. A continuous phase will be established, which is in between the two discrete states and provides insight into the actual strength of both couplers. A stronger coupler will cause the continuous phase to shift towards its intended phase. The SHIL then discretizes the phase, which allows clear identification of the dominating coupler. The data is provided in Figure 7.13, which shows a distinct transition based on the coupler's strength. As can be seen, the coupling strength is in some cases non-monotonic. Also, the transition varies between the individual test pairs, further illustrating the effect of random mismatch.

Although both setups do not provide conclusive data on the effect of random mismatch of the couplers on the computation accuracy, both highlight the consequences of mismatch on the coupling. The considerable spread between the individual coupling connections additionally pose a limit to the actual usable resolution for the coupling weights. Despite having on average an accurate behavior, local variations will likely superimpose small weight differences.

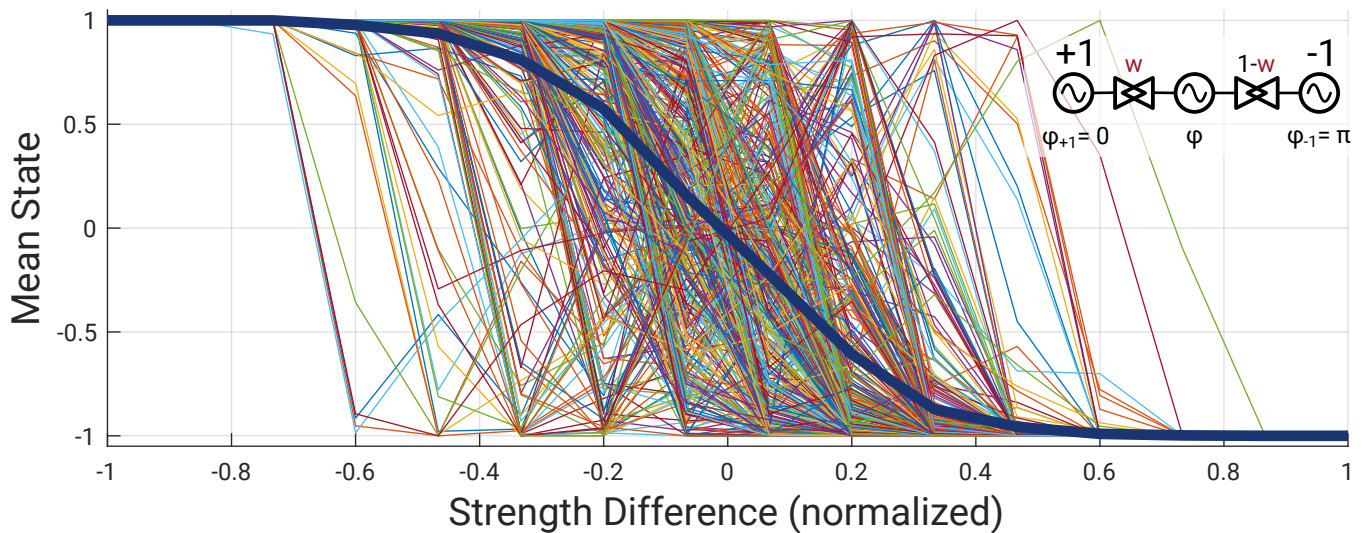
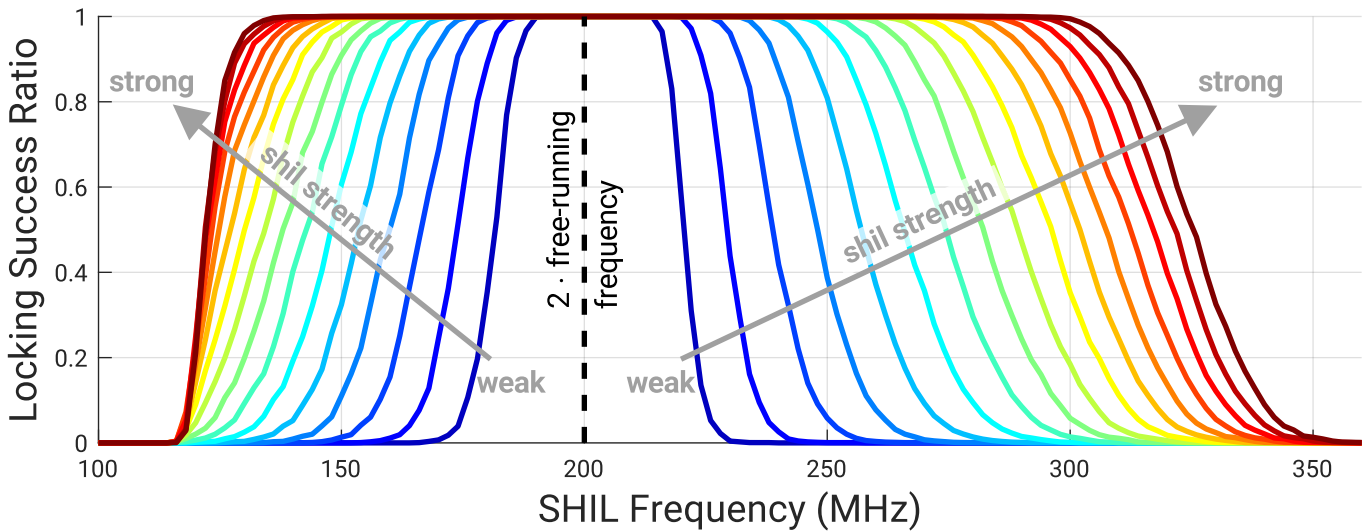


Figure 7.13.: Exemplary mismatch data based on an oscillator, which is forced by one coupler in the state  $+1$  and by another coupler to the state  $-1$ . Depending on the imbalance of the weights  $w$ , the oscillator will strive towards one of the two states.

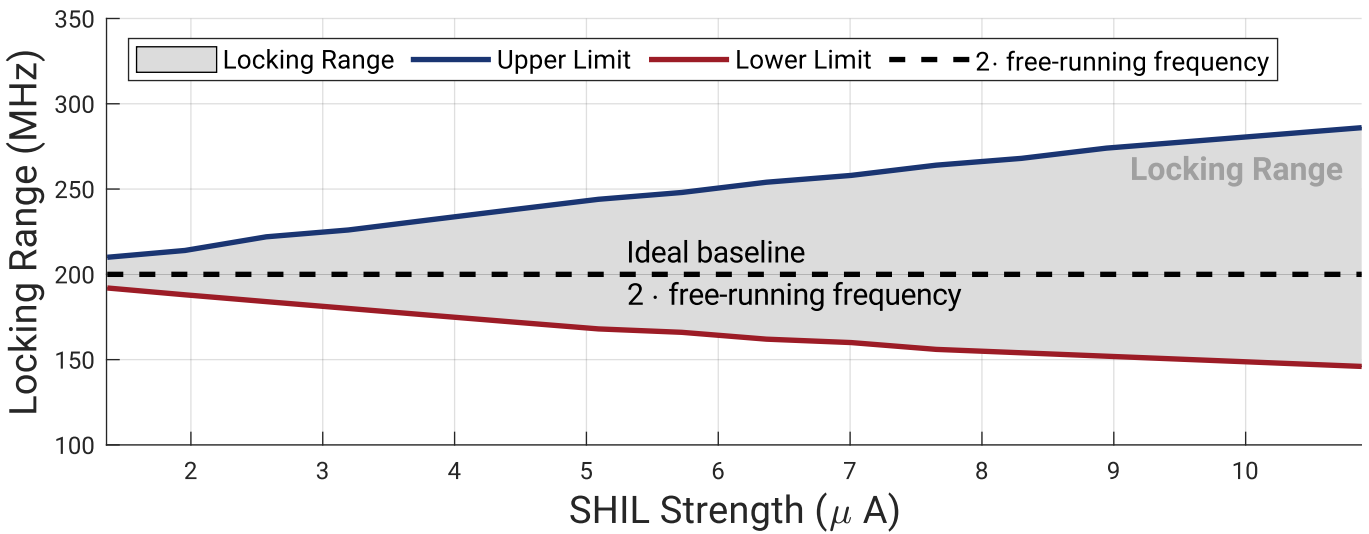
<sup>†7</sup>The spin-bias determines the state. To ensure that the endpoints stay as stable in their phase (respectively state) as possible, they are additionally coupled to other fixed endpoints to maximize their resilience.

### 7.3.4. SHIL

Although the SHIL is not directly creating interactions between the oscillators, it is crucial for the computing process as well. It is responsible for the phase discretization and, thus, the assignment of the discrete states. The frequencies of the oscillators are forced into the sub-harmonic, which ensures a meaningful definition of the phase between those. When coupled, some oscillators could (temporarily) run at different frequencies.



(a) Locking Success Rate at Different SHIL Strengths



(b) Locking Range vs SHIL Strength

Figure 7.14.: Experimentally obtained locking range for free-running oscillators. The SHIL strength is varied from the weakest to strongest settings, which are supported by the Glowworm chip. (a) The locking ratio for color-coded strengths from **weak** (blue) to **strong** (red). (b) The resulting usable locking range, where all oscillators lock correctly to the SHIL signal.

---

## SHIL-locking Range

The bare requirement for phase discretization is frequency locking. The frequency difference between the actual operating frequency of the oscillators and the targeted SHIL frequency needs to be overcome by a sufficient locking strength. Experimentally, one (or more) frequency measurements are conducted to determine if the oscillator is locked. An oscillator is determined as locked if its frequency during the applied locking signal is at the expected fundamental frequency<sup>†8</sup>. This is repeated 100 times for each oscillator out of all 1440 oscillators of the Glowworm chip and then averaged. A locking ratio of 1.0 means, that every oscillator is successfully and repeatably locked by the SHIL signal. Any locking ratio below one means that some (or all) oscillators failed to lock in at least one of the repetitions. The derived locking range is then the largest interval, where a locking ratio of 1.0 is achieved.

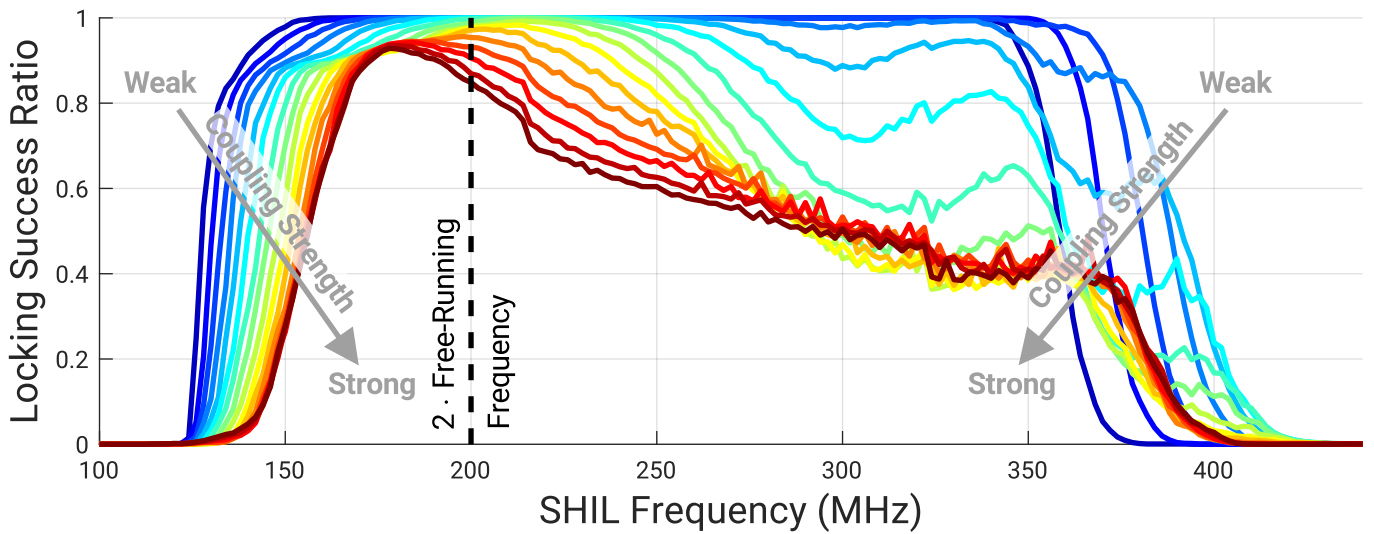
**Locking in the Ising Model (n=2)** The experimentally investigated locking ratio of free-running oscillators is shown in Figure 7.14a. A frequency calibration was conducted before so that all oscillators run approximately at a free-running frequency of 100 MHz, while the SHIL frequency is swept. Theoretically, the SHIL frequency should be twice the frequency at 200 MHz. The strength of the SHIL injection is color coded from strong (red) to weak (blue). The transition between a fully locked system (locking ratio = 1) and a non-locked system (locking ratio = 0) is gradual. While a single oscillator has a sharp transition between either being locked or not, the border blurs due to the slight variance of the manufactured oscillators. The oscillator and the applied SHIL strength have additional individual random variations. The derived locking range determined at different SHIL strengths is shown in Figure 7.14b. As expected, a stronger SHIL strength tolerates greater frequency deviations from the doubled free-running frequency. At small strengths, locking is only possible for very small frequency differences. It increases roughly linearly with the strength, but is slightly off-center from the expected double of the free-running frequency.

Assuming that the oscillators are free-running is not representative for OIMs. Since the oscillators mutually couple, their phases and frequencies are additionally influenced, which can act against the locking and reduce the usable locking range. Hence, the setup of a ring-of-three anti-phase coupled oscillators acts as a conservative estimation of such effects. The ring topology leads to a strong frequency shift and creates a fight between the coupling and the SHIL because both cannot be satisfied simultaneously. This effect tends to be stronger for the anti-phase ring-of-three structure than for usual couplings when solving optimization problems. Therefore, it is reasonable to consider this as a conservative estimation of the usable frequency range. Similar to the discussed free-running case, locking in the presence of coupling connections is shown in Figure 7.15. The maximum SHIL strength is always used, while the coupling strength is varied. As shown in Figure 7.15a, locking gets worse with higher coupling strengths. A stronger coupling reduces the usable SHIL locking range until locking is not possible anymore, as shown in Figure 7.15b. It should be noted that the locking range converges to a frequency slightly above the free-running frequency of the oscillators. This is caused by the shift of the frequency in the coupled state as indicated by the black dotted line.

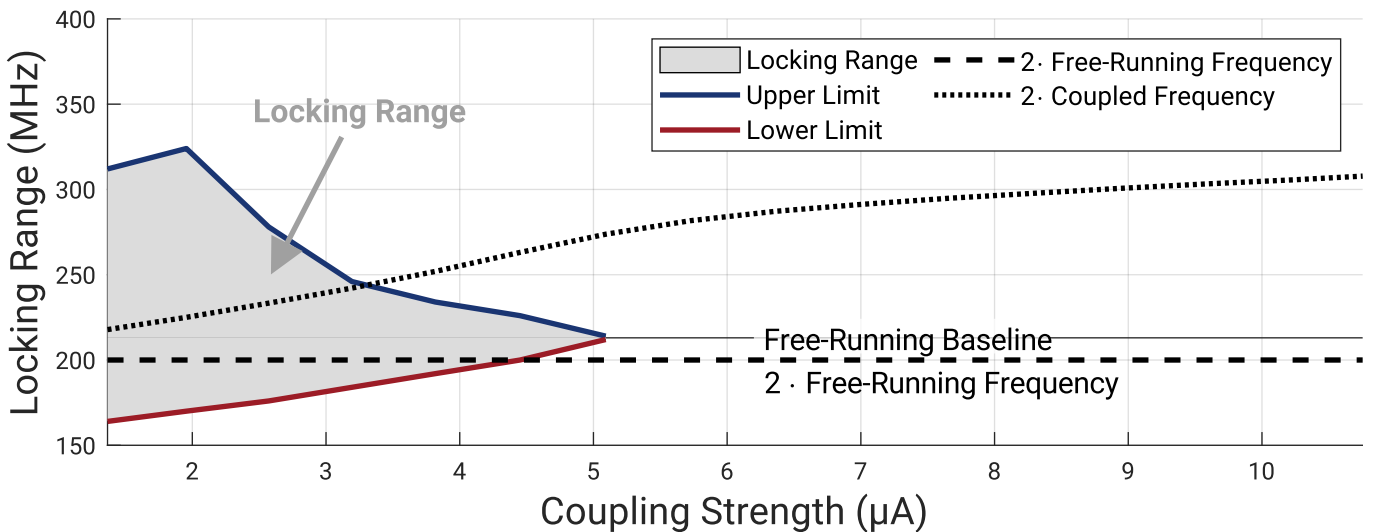
**Higher Order (n > 2)** The SHIL was optimized for the second harmonic locking (n=2) as previously shown. However, higher frequencies (n > 2) can be injected, leading to more than two discrete states. Even harmonics create the same number of discrete states. However, odd harmonics actually cause twice as many discrete states due to the common mode injection in the differential topology (see Section 5.4). So, injecting the fundamental frequency leads to two discrete states. The third harmonic creates 6 discrete states. The locking for higher-order SHIL is shown in Figure 7.16. Since the design is optimized for n = 2, it yields the largest

---

<sup>†8</sup>  $\frac{f_{shil}}{2}$  for Ising;  $\frac{f_{shil}}{n}$ , where  $n \in \mathbb{N}$  in the general form. A small error margin for the measurement is taken into account.



(a) Locking Success Rate at Different Coupling Strengths



(b) Locking Range vs Coupling Strength

Figure 7.15.: Evaluation of the locking range in the presence of a 3-node anti-phase coupled ring to provoke a fight between the SHIL locking and couplers. (a) Locking range at maximum SHIL injection where different strengths of the local coupling from **weak** (blue) to **strong** (red) are used. (b) Graphical representation of the locking range with additional plotted frequencies in the coupled state. When the coupling strength exceeds  $5 \mu\text{A}$  no locking is possible anymore.

locking range. Unfortunately, not all oscillators can lock at  $n = 4$ . However, there is a larger distinct locking range for  $n = 6$ . While the locking range should decrease with higher frequencies, this is against the trend for unknown reasons. At  $n = 8$  (not shown) there is no considerable locking. Odd harmonics have considerably reduced locking ranges as the SHIL-injector design, which modulates the stage bias current of the oscillator, favors an even number of harmonics. Since the fundamental injection  $n = 1$  also leads to two discrete states, it could be used instead of the injection of the second harmonic  $n = 2$  for the Ising model. However, it shows worse computing accuracy and a reduced locking range compared to the case of  $n = 2$ .



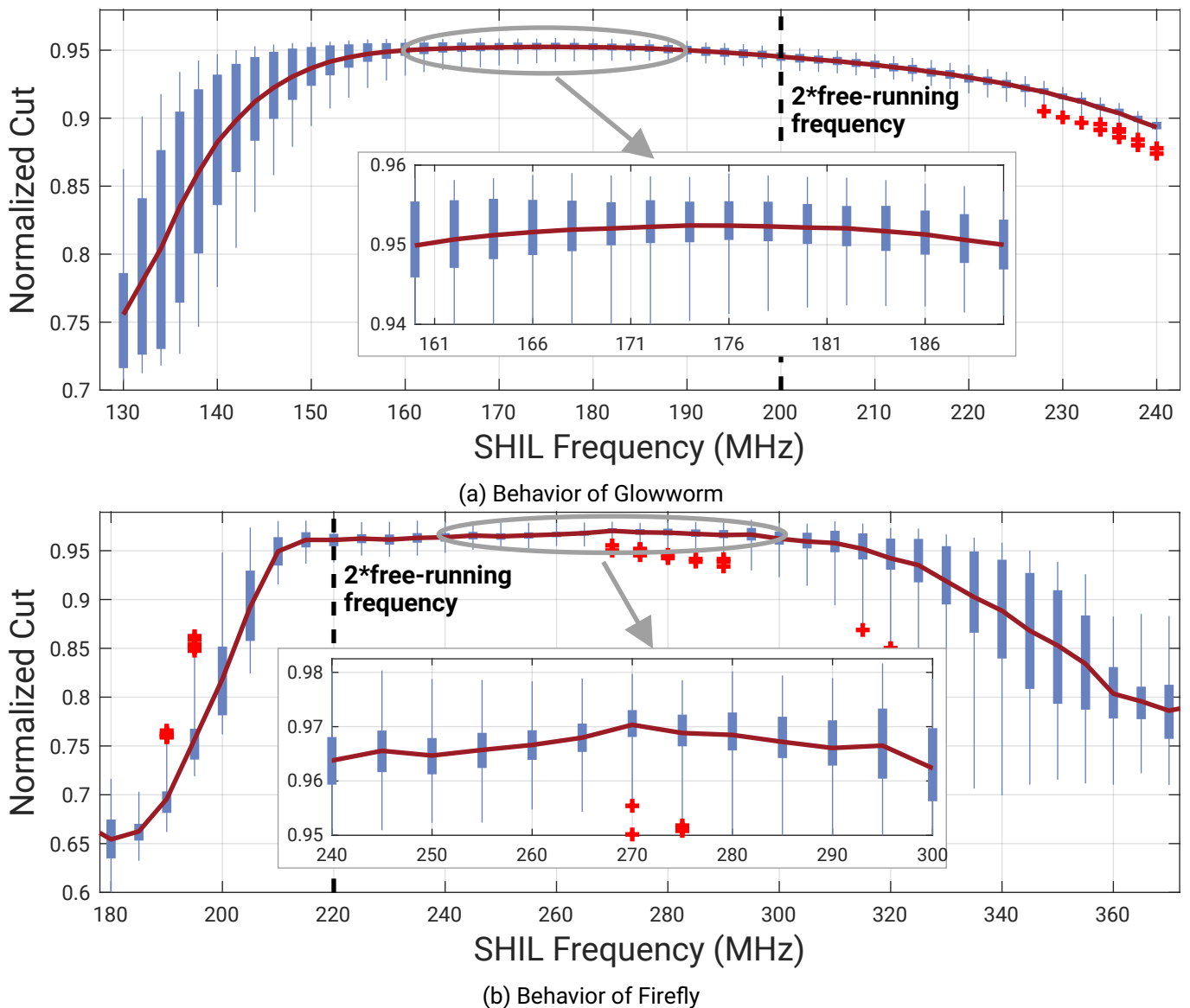


Figure 7.17.: Impact of the SHIL frequency on the computing, shown for both designed chips. The boxplots show the distribution of the average cut of the individual problems from the `difficult` set.

### SHIL-Strength

Not only the frequency, but also the strength of the SHIL is important for the locking. As shown, the locking range has a clear trend of increasing with higher strength. Figure 7.18 solves the same set of benchmark problems for different strengths. The distribution of the average outcome for each problem is additionally indicated by the boxplots. Overall, this shows a clear trend and confirms that stronger SHIL is beneficial for the computing. The best performance is reached at high strengths, where the spread between problems is reduced. The spread especially widens for medium strengths, where some problems are already solved well, while others perform better at higher strengths. The additionally shown locking fail rate exhibits an inverted behavior compared to the obtained cut. It should be noted, that the locking fail rate approaches but does not reach 0%. Even at high strengths, there are a few oscillators which do not always properly lock.



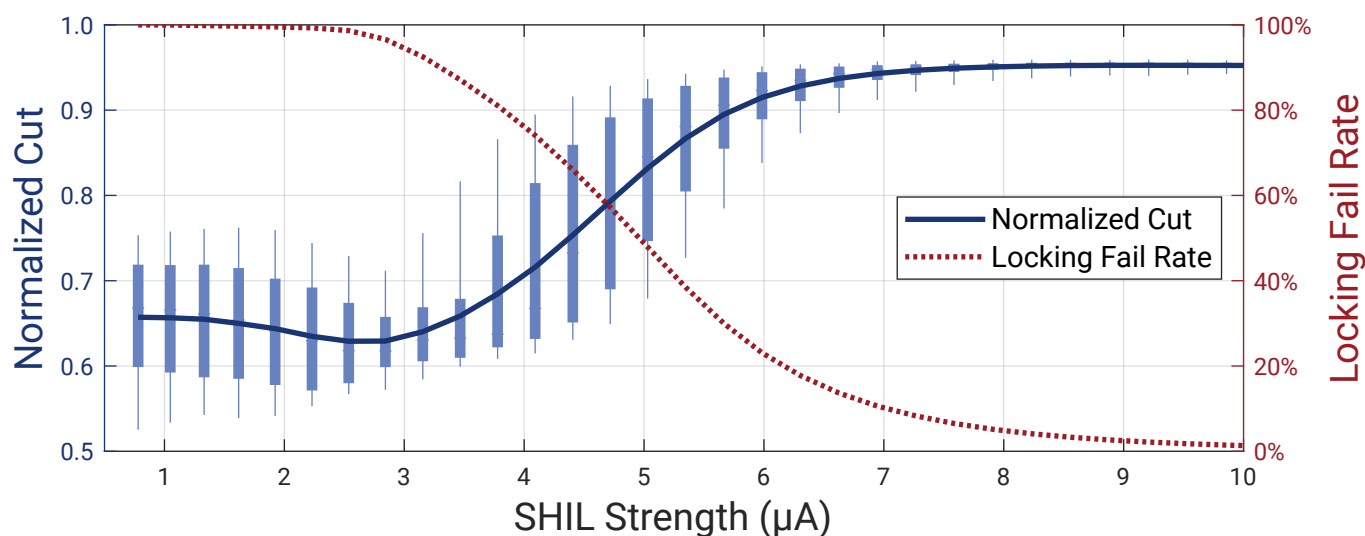


Figure 7.18.: Impact of the SHIL strength when solving optimization problems of the `difficult` problem set. The boxplots show the variations between the individual problems within the set. Additionally, the locking fail rate is shown, which describes the percentage of non-locked oscillators.

In Figure 7.18, there is a plateau reached at the highest strength settings. While this would suggest increasing the strengths further, the performance will eventually drop, as demonstrated with Firefly. It supports even higher strengths, which can be configured in 8 discrete levels. As shown in Figure 7.19, there is a distinct drop for high strengths. The `difficult` set performs best for a strength setting of 0.5 and slightly declines up to settings of 1.25. At even higher strengths, the reduction is even more pronounced. A similar characteristic of poor accuracy at low strengths, with a good performance at medium strengths and slight drop off at high strengths is also noticeable for the other benchmark sets. Since this drop-off was present for all benchmark problems, the maximum strength of the Glowworm tapeout was reduced to a value, which roughly matches 0.81 at the shown Firefly data. In return, the strength can be finely controlled by an external bias voltage in Glowworm. Finally, it should be noted that the strength and the ramp-up, discussed in the following, are affecting each other. Since the ramp-up has a fixed number of steps, the step size is scaled with the strength and cannot be analyzed independently.

## SHIL Ramp

As discussed in the design phase, it is possible to ramp-up the SHIL signal, so that it gradually increases the strength and forces the oscillators into their discrete states. One could think of this gradual increase in strength as a freezing operation. Similar to the formation of ice crystals from water, the outcome of a computation could depend on these ‘freezing’ operations as well. When the full strength is applied immediately, the oscillators are basically forced to align into the nearest discrete state, which can result in a significant phase change within a fraction of an oscillation cycle and introduce temporary distortions of the waveform. When the strength is gradually ramped up, the oscillators are gently pushed into a discrete state. At the beginning of the ramp-up, coupling interactions between the oscillators could still change their phases. With increasing SHIL, the influence of the couplers slowly reduces, which makes it harder to change the now almost discrete phase states. Hence, changes of the quasi discrete phase states will get less likely, but are still possible. The motivation is that this process will allow some oscillators to change their phase groups before completely freezing the groups. This can be useful if they are, for example, not clearly in one of the two main phase groups or might react based on the settling of neighboring oscillators.

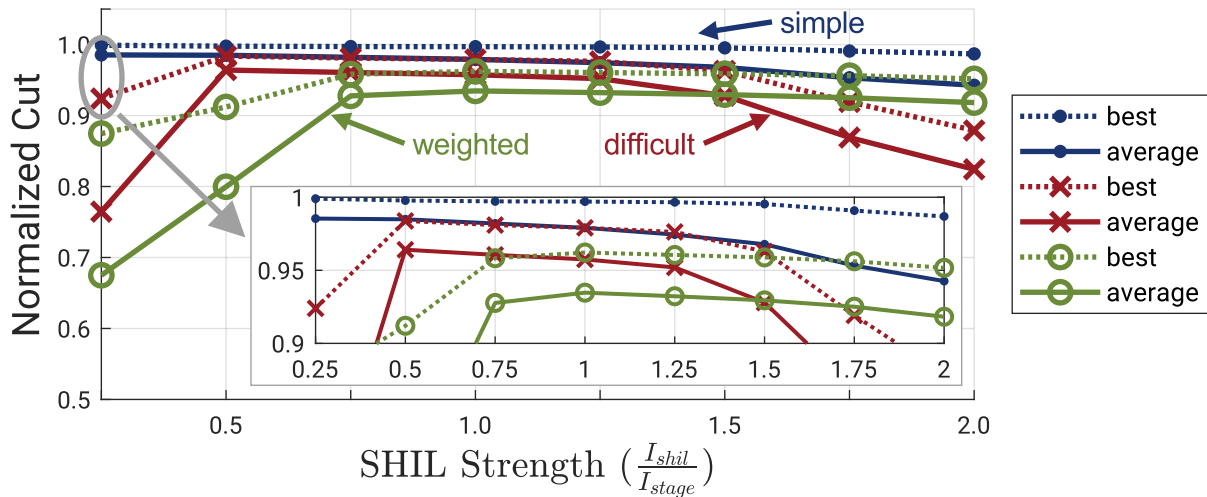


Figure 7.19.: Impact of the SHIL strength shown with the Firefly. It has 8 discrete SHIL injection strengths, which determine the ratio between injected current ( $I_{shil}$ ) and stage bias current ( $I_{stage}$ ). The best and average solution out of the problem sets simple, difficult, and weighted are noted.

The impact of the ramp-up is evaluated using the Firefly design, because it features an adjustable ramp. As shown in Figure 7.20, there is a clear trend that the computing performance drops with larger computation times, where the simple set tolerates a slower ramp than the others. This downward trend is unexpected and no possible explanation can be given. Rather, the solution quality is expected to settle on a constant, high accuracy for very long ramp durations. There is no apparent reason, why the oscillators should settle in a worse solution. Although the experiment configuration was verified, the dynamic ramp-up within the chip cannot be directly measured to rule out any potential bug. The best performance is reached at a ramp-up of 16 cycles (4-bit resolution), which provides a better accuracy of approximately 0.2%. However, the system already provides good results without any ramping. Consequently, the flexible ramp-up was fixed to 16 cycles in the second generation for Glowworm to save area. Alternatively, the ramping can also be completely disabled for the Glowworm design. This saves computation time at the cost of reduced computing accuracy. The performance with and without ramp is shown as part of the coupling time investigation in the next section. Interestingly, there is a greater difference between an activated and deactivated ramping for the Glowworm than for the Firefly design. In the case of the difficult problem, the ramping improves the performance by approximately 4% for Glowworm, which is considerably more than the results from the Firefly design.

Unfortunately, no clear explanation for the differences between both designs can be provided. The SHIL injection circuits follows the same design principle and has just minor differences between both generations. Also, the oscillators were further optimized, but the principle stayed identical. However, the data from the Glowworm design emphasizes the weak trend of Firefly, suggesting that a SHIL ramp-up is beneficial and should be included. The observed differences might be a side-effect of the AC coupling of the Firefly design, which limits the maximum injected current into the oscillator. Future research might additionally try to add a spatial component to the SHIL ramp. For example, the ramping of every second oscillator could be started delayed or varied in strength. Such small local differences might improve the locking of oscillators, leading to better annealing.

The SHIL can not only be ramped up, but also ramped down similarly. The hypothesis was, that this gently releases the oscillators, so that they can evolve from the current to a new solution. However, no noteworthy impact of the ramp down could be identified. This might be an indication that the coupling of oscillators

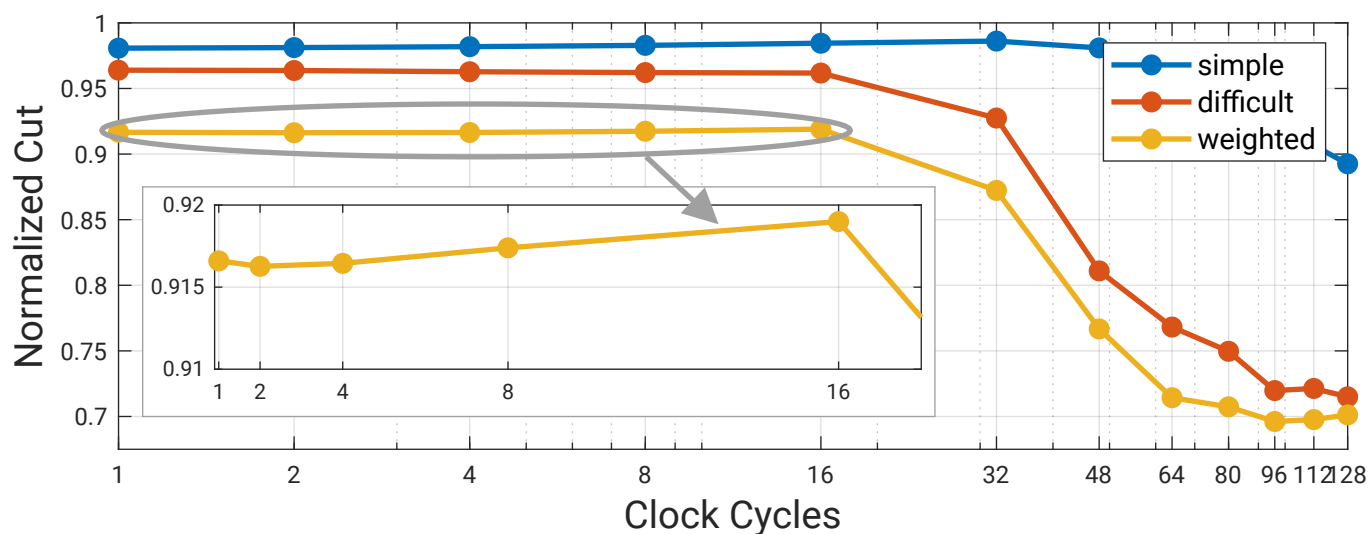


Figure 7.20.: The impact of ramping-up is shown based on the Firefly tapeout. When a SHIL cycle is triggered, the strength is ramped-up in up to 16 discrete steps. The total time for the ramp can be varied by choosing the step size and time for each step. The x-axis depicts the time in nominal oscillator cycles it takes to complete the ramp.

could be of such a strong chaotic behavior, that they find new solutions independent of the initial phase. Consequently, the ramp-down feature was removed for the Glowworm tapeout.

Overall, the impact of the ramp is not as clear as it was expected. The Firefly design shows an unexpectedly strong drop for large ramp-ups. Especially the fact that the computation starts to massively lose performance has no reasonable explanation. However, the ramping itself provides a noticeable improvement, which is confirmed by both chip generations. The Glowworm design seems to benefit much more from the ramping than Firefly.

### 7.3.5. Coupling Time

Another potential impact is the coupling time, which determines how long the oscillators mutually couple before the ramp-up of the SHIL is started. This is analyzed in Figure 7.21, where it is evaluated with (solid line) and without (dashed line) ramping. The time resolution is limited by the clock frequency of the external FPGA. Differences in the propagation delay between the external signals will likely affect the actual elapsed time. The measurements suggest that a coupling time of  $16.7 \text{ ns}^{\dagger 9}$  is already enough to reach the best possible solution accuracy. However, the oscillators still have the opportunity to react and change phases during the ramp-up. If the SHIL ramp-up is disabled, the outcome exhibits a higher variance between the tested points, but does not show any clear trend. Since the accuracy is reduced by up to 4% without the SHIL ramp, this case does not need to be considered in more detail.

Although the results suggest, that a coupling time of  $16.7 \text{ ns}$  is enough, a coupling time of  $200 \text{ ns}$  was chosen as default. Since this is approximately one order of magnitude larger, it should ensure sufficient time for coupling under any relevant operating parameters. Since the coupling time accounts for only 21% of the overall computing cycle, this does not lead to an excessive prolongation. If needed, the coupling time could be reduced to achieve higher speeds.

<sup>†9</sup>This corresponds to two FPGA clock cycles. The actual elapsed time will differ due to asymmetric signal rise and fall times.

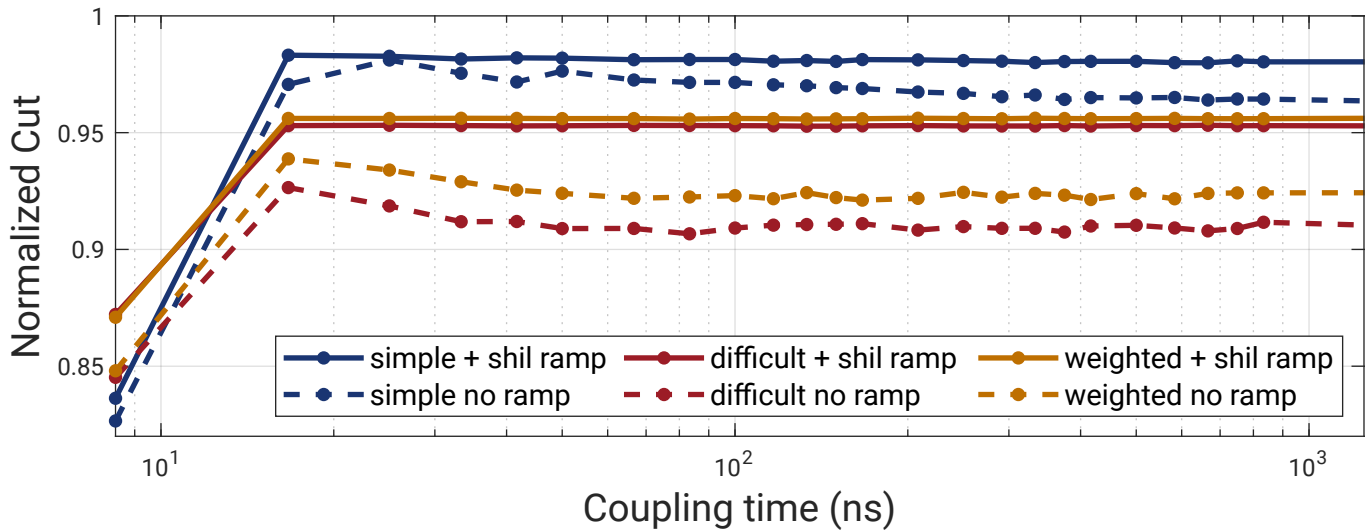


Figure 7.21.: Impact of the coupling time on the solution. The same problem set is once solved with enabled ramping (solid line) and without (dashed line).

### 7.3.6. Frequency Mismatch

The calibration of the individual oscillators were already discussed during the design in Section 5.2.6. The proposed calibration focuses on compensating the unavoidable manufacturing mismatch. This section evaluates the actual impact of frequency variations on the computing accuracy. By purposely overwriting the calibration codes, the impact of mismatch can be tested. The results are shown in Figure 7.22, which uses three different coupling strength configurations. In addition to the normal strength, which yields the best accuracy, a weaker and a stronger setting were chosen to highlight its sensitivity. Comparing these three settings confirms the prediction in the design section, that weaker coupling is more sensitive to frequency mismatch than stronger. At the weak setting, the accuracy drops as soon as the mismatch increases, while small variation just slightly affect the normal coupling strength. However, the strong coupling is hardly influenced and starts to lose accuracy only at very strong mismatch. For the chosen normal coupling strength, the frequency mismatch should be kept at a standard deviation of less than approximately 1 MHz (1% of the free-running frequency).

From a system design point of view, this introduces another trade-off. Reducing the frequency mismatch is usually area costly. For example, the circuitry could be better matched or a precise high-resolution calibration technique could be used. However, the frequency matching can not be reduced to arbitrary levels and becomes increasingly difficult. Low-frequency noise components will additionally cause a random drift between the oscillators. Due to individual variations in the temperature coefficients of each oscillator, local temperature differences will impact the frequency matching as well. So, the frequency mismatch introduces a trade-off between a well-calibrated but larger design and a design with stronger coupling, which is potentially power-hungry. The choice of the coupling strength will however affect the accuracy as well. Hence, a compromise is needed, where the frequency mismatch is effectively reduced without requiring too much area, while the coupling strength is kept at a reasonably low level.

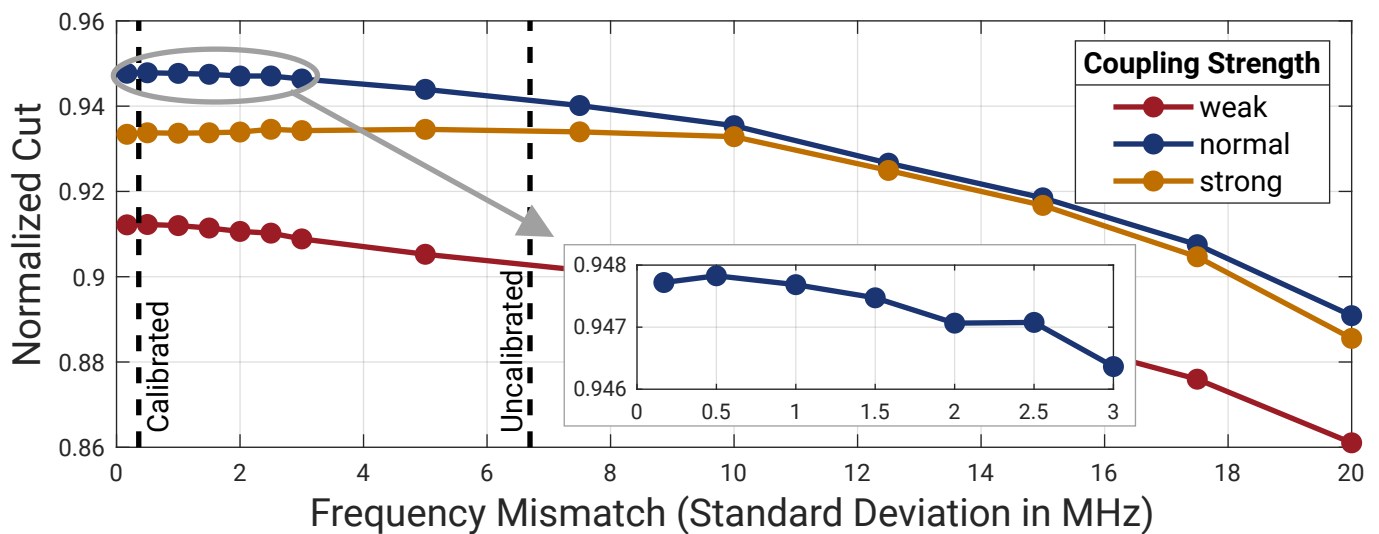


Figure 7.22.: Impact of the oscillator frequency mismatch on the computing accuracy. The same set of optimization problems were solved at different frequency variations by intentionally overwriting the calibration code. Besides the normal coupling strength, a stronger and weaker value is additional shown to emphasize the strength dependent sensitivity to mismatch.

## 7.4. Spin-Bias Coupling

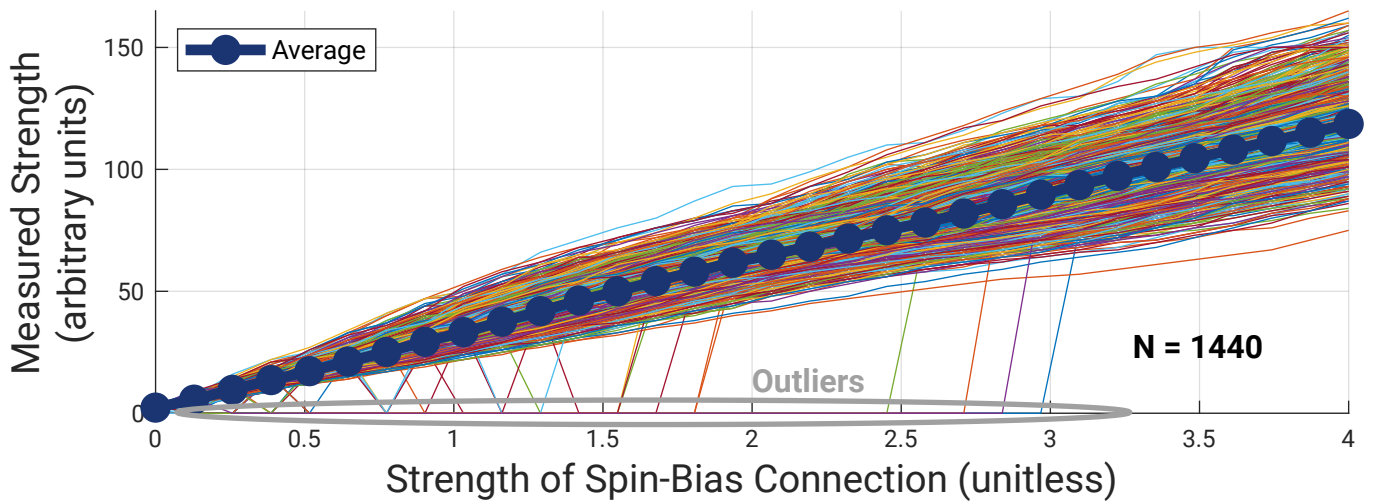
The spin couplings implementing the  $h_i$  weights just involve a single spin  $\sigma_i$ , while the ordinary coefficients  $J_{ij}$  use the interaction of two spins  $\sigma_i \cdot \sigma_j$ . As discussed in Section 5.6, the  $h_i$  term is implemented using the reference clock. Since the underlying implementation differs from the oscillator-to-oscillator coupling, their behavior could vary. So, this section evaluates the reference clock-based couplings and checks for potential differences to the regular oscillator-to-oscillator coupling. The mismatch and monotonicity of the spin-bias couplings are evaluated separately.

### 7.4.1. Strength Mismatch

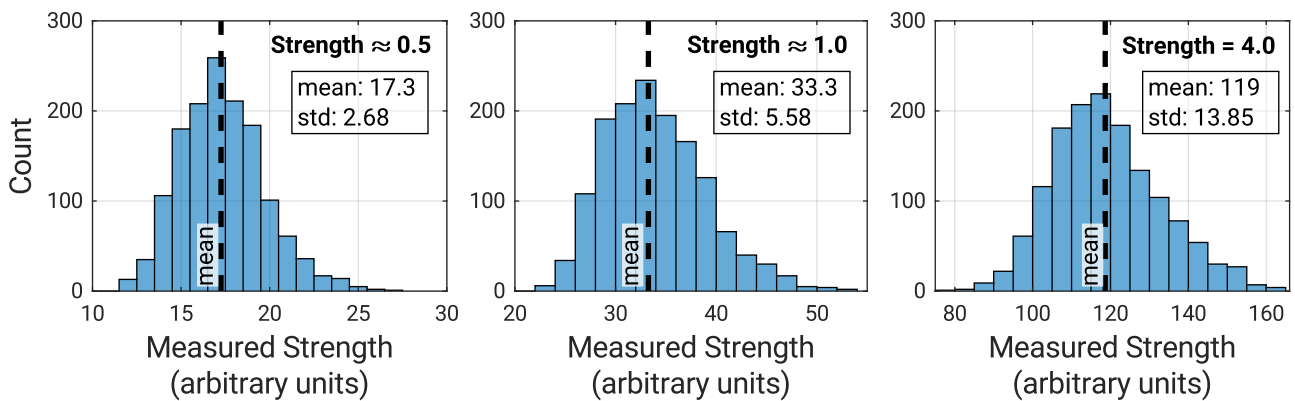
To measure the mismatch of the spin-bias connections, a measurement methodology different from that previously used in Section 7.3.3 is needed. Since they couple to a fixed reference, it is straightforward to check if they couple as intended or were forced into a different state by other connections. Two approaches are used to judge potential mismatch. One is based on a locking range and the other uses a ‘fight’ between couplers.

The first approach measures the locking range of the spin-biases, where a stronger coupling automatically leads to a larger locking range. Here, the oscillators are calibrated to 100 MHz, while the frequency of the reference clock is increased step by step. Hence, the maximum frequency at which a spin-bias still successfully couples (or locks) is a measure of the coupling strength. By conducting a frequency measurement in the coupled state, it can be determined if the oscillators are locked<sup>†10</sup>. For convenience, just the frequency difference to the free-running oscillator is used as an arbitrary unit for the coupling strength. This method does not guarantee

<sup>†10</sup>Additionally the phase could be considered as well. However, the phase angle will shift slightly at higher frequency differences and can have inherent offsets. Defining a phase range where it is still coupled is therefore more error prone. The frequency-based locking has a sharp transition, where the oscillators follow the reference frequency if locked and will operate at a lower frequency if not locked.



(a) Linearity of the strength



(b) Histogram at selected strengths

Figure 7.23.: Quantification of the strength of reference connections based on the locking strength.

- (a) The strength is on average approximately linear, but individual spin-bias connections exhibit non-linearity. A few outliers are present, which are likely caused by a failed measurement.
- (b) Variation of the quantified strength for all 1440 nodes showing a considerable spread between individual nodes.

any linearity but is monotonic and tends to be approximately linear<sup>†11</sup>. The possible spin-bias strength ranges from 0.0 to 4.0, which is normalized to the maximum strength of the oscillator-to-oscillator  $J_{ij}$  connection with a value of 1.0. The sweep in Figure 7.23a shows an approximately linear trend, which is monotonic for 1387 out of 1440 oscillators (96.3%)<sup>†12</sup>. The data exhibits some outliers, which could not clearly be identified as locked. Since they were influenced by the coupling and operated as expected at higher strength settings, the reason for the outliers is unknown. Monotonic behavior can be expected, because the DACs providing the current for the couplers were designed to have a DNL of less than 1. However, other factors such as the bias currents, coupler circuit and oscillator sensitivity add more variations that could result in non-monotonic behavior.

<sup>†11</sup>See the similar SHIL locking range in Section 7.3.4

<sup>†12</sup>Strictly monotonic for 1221 out of 1440 oscillators. The average step size is 3.75 arbitrary strength units, where the measurement was conducted with a resolution of 1 strength unit.

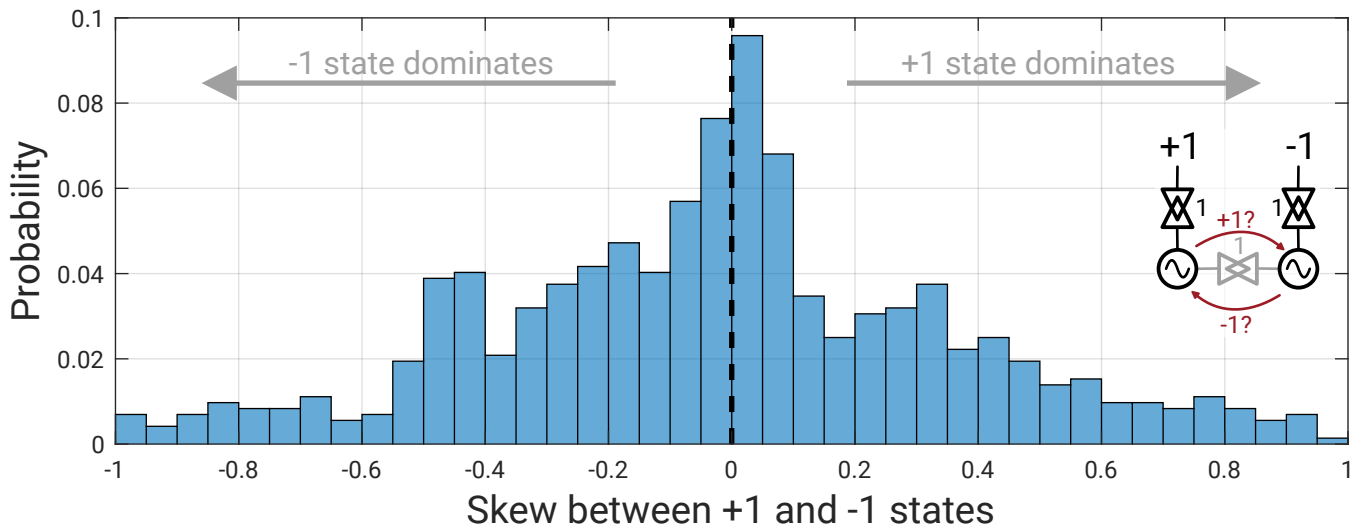


Figure 7.24.: Mismatch of the spin-bias couplings based on the pairwise skew based on the shown topology

The distribution at selected strength points of 0.5, 1.0 and  $4.0^{†13}$  is shown in Figure 7.23b. The standard deviation is approximately 10% to 15% of the mean strength. While usually, a higher strength setting is actually stronger than a lower setting, this can not be guaranteed for every oscillator. Nevertheless, very good solutions to optimization problems are found, as discussed later. Multiple mechanisms, such as the coupling sensitivity of the oscillator, mismatch of the coupler, local bias current, and DAC non-linearity accumulate into the observed distribution. All of these are heavily influenced by random manufacturing variations and are area costly to reduce, while being challenging to calibrate. The shown approach could be used to calibrate the strength of the  $h_i$  couplers. However, this would require a mechanism to fine-tune the DAC output to compensate for the variations.

The second setup is based on the ratiometric behavior of a pair of two oscillators. One oscillator is coupled using the spin-bias connections to +1 and the other to -1. An in-phase oscillator-to-oscillator coupling connects both oscillators. Clearly, at most two of these three couplings can be achieved. Either one of the spin-bias couplings or the oscillator-to-oscillator connection is violated. If all connections have equal strengths, then there would be a probability of  $33.\bar{3}\%$  for each edge to be cut. However, this analysis focuses on the mismatch between both spin-bias connections. Hence, only the probability difference, that the +1 respectively -1 states as intended by the spin-bias connections is reached, is considered. Ideally, their probability should be exactly equal even if there are systematic differences in their strength compared to the regular oscillator-to-oscillator connections. So, any variations indicate mismatch between both spin-bias connections. For convenience, the probability difference is expressed by the average of the states, which is defined as skew. A skew value of 0 is desired, where it is equally likely to end up in the +1 and -1 state. A skew value of -1 means that this connection dominates and vice versa for +1. The distribution of that skew for 720 pairs is shown in Figure 7.24. While there is an aggregation at the desired skew of 0, several pairs exist where one connection completely dominates. When considering that usually more than one least significant bit (LSB) is needed to make a coupler dominant, it means that there is a considerable mismatch in the strength of every individual connection. Considering the spread of the previous setup, such a behavior is not surprising. However it means, that some solutions to optimization problems are highly unlikely, because the random mismatch can already cause a strong preference into one spin state.

<sup>†13</sup>Due to the 5-bit resolution of the spin-bias DAC, the actual values are 0.5161 and 1.0323. The strengths are normalized with respect to the oscillator-to-oscillator coupler strengths, whose maximum strength is 1. No coupling corresponds to 0.

## 7.4.2. Comparison to Local Connections

A similar setup can be used to compare the effective coupling strength between the oscillator-to-oscillator connections and the spin-bias connections. This setup uses the same topology as the evaluation in the previous section. Now, one oscillator acts as a fixed reference, which is anchored to the  $+1$  state<sup>†14</sup>. The other oscillator is coupled with variable strength to the  $-1$  state, so that the in-phase oscillator-to-oscillator connection creates a conflict. Hence, the spin-bias and the oscillator-to-oscillator connection are fighting against each other, where the stronger coupling will determine the phase. 500 samples per strength setting and pair are taken so that the probability to end up in both possible states is quantified. If both connections have an equal effective physical strength, the probability of ending up in either of the two opposing states should be 0.5. The sweep is shown in Figure 7.25, where all 720 pairs individually and their average are provided. The equilibrium point of 0.5, which is reached at a spin-bias strength of 0.82, indicates that the spin-bias connections tend to be stronger at the same strength setting, respectively bias current. However, this could be easily compensated for by scaling the spin-bias strength accordingly. Some pairs exhibit considerable non-monotonicity, which is especially visible by the severe outliers on the right-hand side. These are likely caused by very unfortunate combinations of non-linearities of the internal DACs for the connections. Despite the monotonicity of the majority, extreme outliers are visually more prominent due to plotting all pairs into one graph.

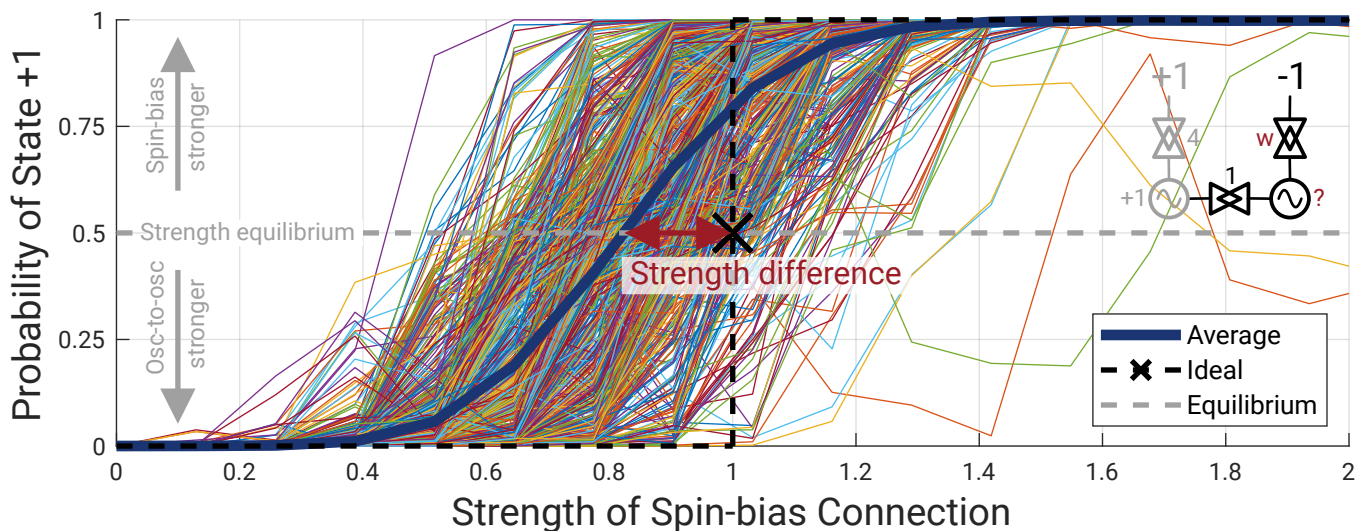


Figure 7.25.: Comparison between the spin-bias and the oscillator-to-oscillator connections. The oscillator-to-oscillator connection strength is held constant and the spin-bias connection ( $w$ ) is swept. Depending on the effective strength, either the oscillator-to-oscillator connection or the spin-bias connection will dominate.

While this setup can provide a quantitative comparison of the strength, it does not consider the coupling dynamics. Deviations in the dynamics might cause additional differences when solving actual optimization problems. Additionally, this setup does not consider that oscillator-to-oscillator connections can lead to frequency changes of the involved oscillators. However, the spin-bias connections force the oscillators into the system reference frequency.

<sup>†14</sup>The maximum strength of the spin-bias coupling is used. As this is 4x larger than the maximum oscillator-to-oscillator connection, the spin state can be treated as fixed.



## 7.5. Routing

As already discussed during the design in Section 5.5, the coupling dynamics for routable channels are inevitably changed due to the unavoidable propagation delay. While the delay for transmitting the oscillator signal between local neighbors is negligible compared to the operating frequency, that is not the case anymore for such long-routed connections. The proposed delay compensation scheme addresses this issue, however it affects the coupling dynamics. The difference in behavior to local couplings is first analyzed based on the coupling between pairs of oscillators and then in the context of optimization problems.

### 7.5.1. Pairwise Coupling

For a local connection, the coupling of a pair of oscillators is unspectacular. It couples as intended and then locks in these intended phases under SHIL injection with very high reliability. Doing the same for the routed connections is more interesting, as the propagation delay between those oscillators affects the coupling. This delay consists of the transmission through the routing network, which can roughly vary between 287 ps and 2.78 ns depending on the length<sup>†15</sup>. Additionally, the delay line adds a configurable delay between nominal 0 ns and 3.1 ns with an approximate step size of 100 ps.

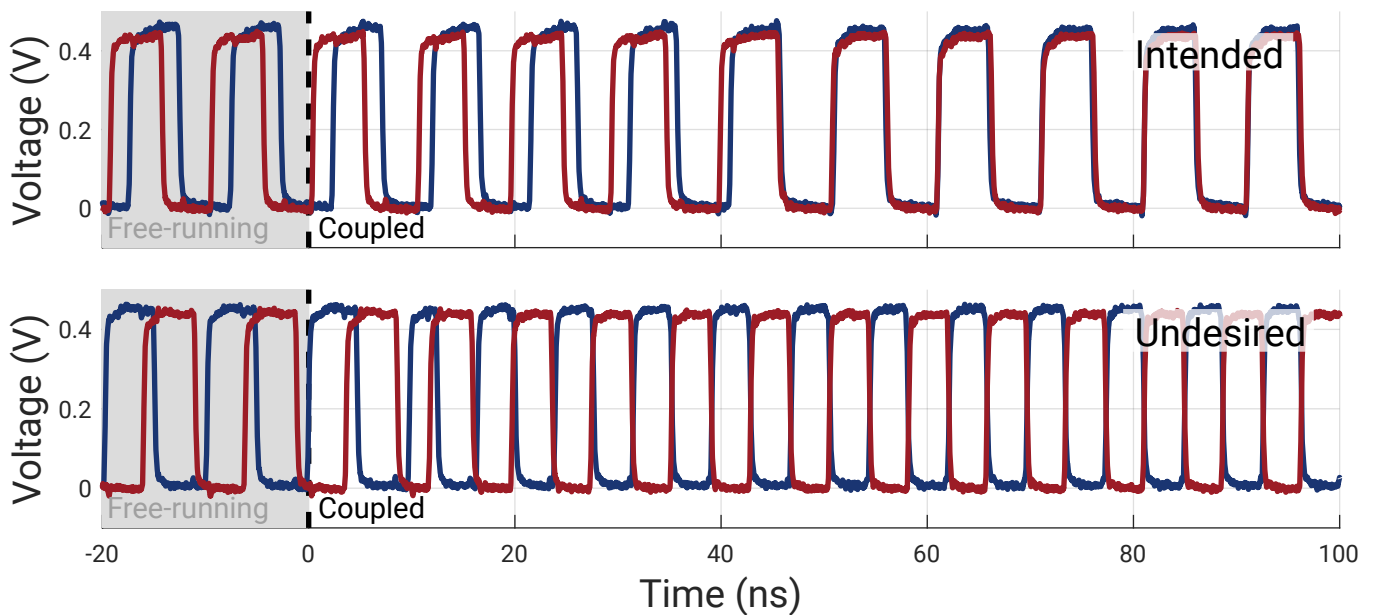


Figure 7.26.: Example waveform of two oscillators, which are in-phase coupled over the routing network. At the top, they couple as intended. However, they settle in the opposite-phase in the bottom plot. Both waveforms were recorded consecutively under the exactly identical operating conditions. There is a physical distance of 28 grid-blocks and an additional delay of 1.5 ns by the delay line.

Only the coupling of a pair of oscillators is considered, which is not as straightforward as for the local connections. Whether the pair of oscillators couples as intended is a matter of probability, which depends on the delay. Figure 7.26 provides two measured waveforms of the same pair of coupled oscillators under identical conditions. At the top, it couples exactly as intended, while at the bottom, it settles undesirably in the opposite phase. In the bottom case, the frequency additionally increases by roughly 31% to 131.4 MHz. In

<sup>†15</sup>Post-layout simulated at nominal conditions. See Section 5.5.3 for more details.

contrast, the top case stays when coupled at 99.6 MHz, which is very close to the free-running frequency. The behavior exhibiting two possible outcomes with intended or undesired coupling is highly undesired when actually solving optimization problems.

The coupling success rate is defined as the percentage of trials in which the oscillators couple as intended at a certain delay. The delay does not only depend on the path, but also has an adjustable supplementary delay line as introduced in Section 5.5. A total of 34 pairs<sup>†16</sup> are simultaneously coupled, which is repeated 500 times. Figure 7.27 shows the success rate for various routing distances and supplementary delays<sup>†17</sup>. The distance is expressed in grid-units, which are defined as the Manhattan distance based on the edge length of an oscillator node as basic unit of length. A short distance is color-coded in blue and a long distance in red. The success rate already drops at small supplementary delays for large distances. For short distances, the success rate starts dropping at higher supplementary delays. It should be noted that for very large distances, no success rate of 100% can be achieved. The shown data might suggest that a supplementary delay of 0 ps could be chosen to yield a very high success rate, regardless of the actual distance. Technically, larger delays tend to reduce the success rate. However, the data does not include the frequency in the coupled state, which should be considered as well.

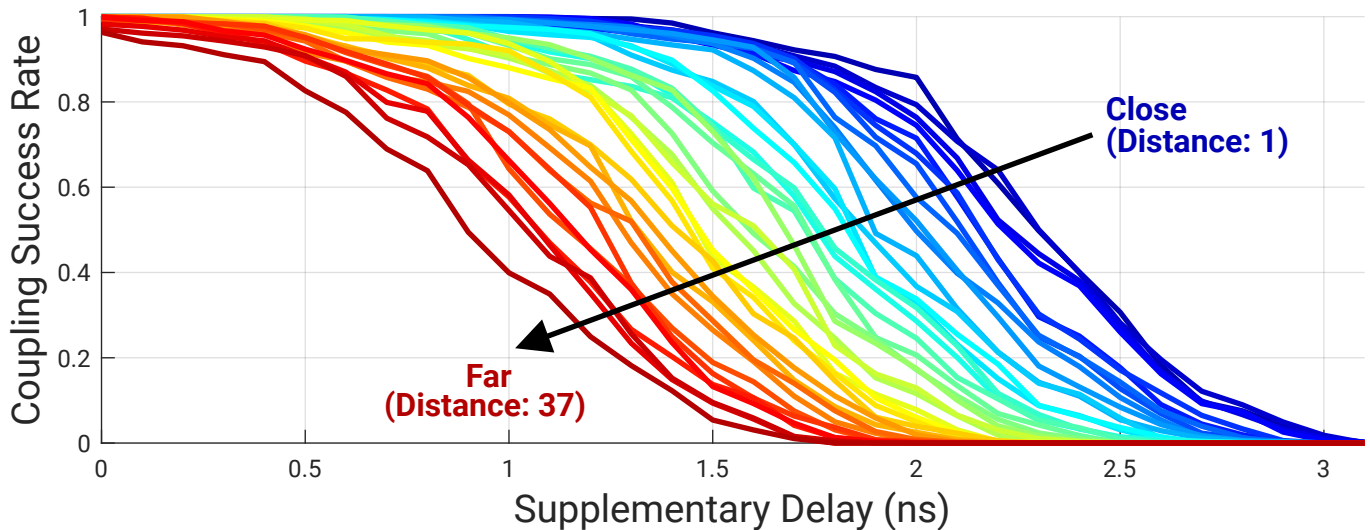


Figure 7.27.: Evaluation of the success rate for coupling with the routing system. Distances from 1 grid-unit (blue) to 37 grid-units (red), which is the largest possible horizontal distance in the routing network, are evaluated at all possible delays. As expected, there is a clear trend that the success rate starts to drop earlier at larger distances.

The success rate and frequency in the coupled state are shown in detail in Figure 7.28 for distances of 1, 12, 24, and 37 grid-units. The frequency of the coupled oscillators depend on the delay of their connections, which can be increased on purpose by the (supplementary) delay line. A small supplementary delay leads to higher frequencies, while a higher supplementary delay reduces the frequency, which qualitatively matches the simulated behavior of Section 5.5. Ideally, the coupled frequency should be close to the free-running frequency,

<sup>†16</sup>In each row, there are 38 oscillators with access to the routing network. Hence, the largest possible horizontal distance is 37 grid-units. In each column, there are 34 oscillators with access to the routing system.

<sup>†17</sup>The decision, whether they couple as intended or not, was made based on a frequency measurement. A phase measurement with the PDC is technically not feasible, since it requires a specific frequency of the oscillator, which is normally ensured by the SHIL locking. Adapting the PDC frequency is impractical since every pair at every evaluated point has another frequency and even varies when coupled as intended or not.

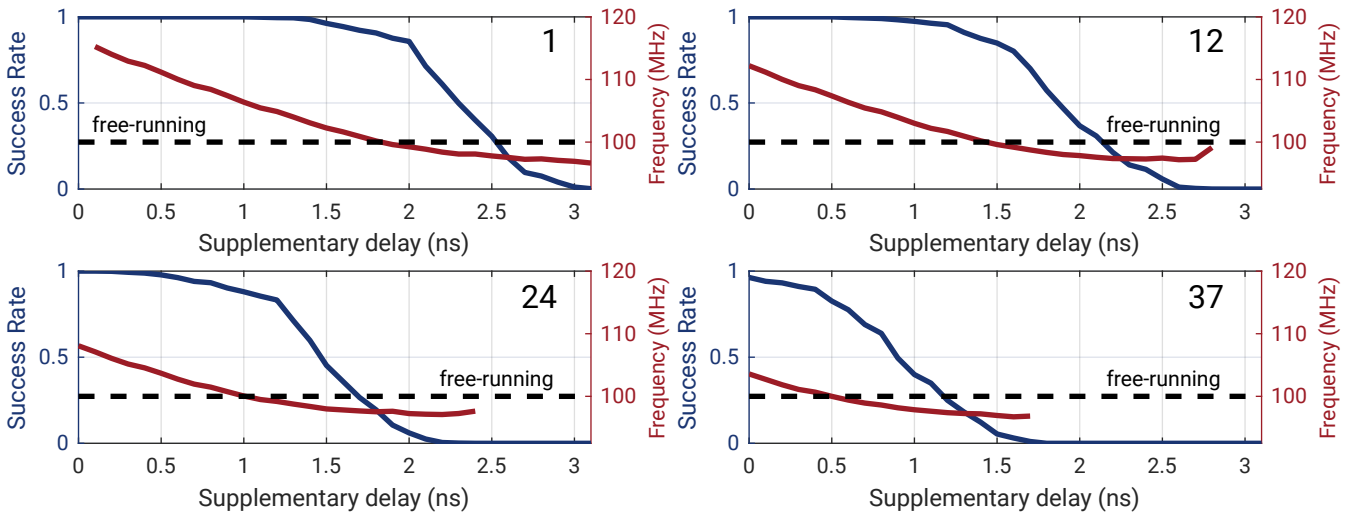


Figure 7.28.: Detailed data showing the success rate and frequency when coupled for selected distances of 1, 12, 24, and 37 grid-units. The success rate is shown on the left axis in blue and the corresponding frequency on the right axis in red. Frequency data points are excluded from the averaging when the oscillators couple in the opposite phase. The free-running frequency is annotated as a black dashed line for reference.

while the success rate should be 1. In this case, the periodic steady state of the coupled pair would match the behavior of regular local coupling links. Unfortunately, such a point does not exist. When the frequency reaches the free-running baseline, the success rate already dropped considerably to 0.883 on average. This is consistent for all distances, as the success rate varies between 0.826 and 0.94.

It should be mentioned, that a coupled pair might not be stable for infinite time. Some cases were observed with an oscilloscope, where a pair of oscillators initially couple as intended, but then flip in the undesired state after a short duration. However, this time exceeded the maximum acquisition time at the required sampling rate of the used oscilloscope (slightly above 50 ms) so that no data can be provided.

Reliable coupling with a success rate of 100% is only possible when the coupled frequency is a bit higher than the free-running frequency. This assumes that such a point even exists, which is only the case for distances up to 28 grid-units. Nevertheless, coupling is possible for longer distances but the quality might gradually deteriorate. However, a coupled frequency achieving the free-running frequency exist for all investigated distances. While this section analyzes the pairwise coupling behavior, it is not enough to determine the optimal delay setting to yield the best OIM computing results. The effect of SHIL on the pairwise coupling is analyzed next section.

### 7.5.2. Pairwise Coupling With Sub-harmonic Locking

As analyzed before, the oscillators either couple as intended or undesirably in the opposite phase, so that the phases are already de facto discrete. However, they still need to align to the phases forced by the SHIL and adapt their frequency into the sub-harmonic. For locally coupled pairs, this is a straightforward procedure, where the oscillators essential stay in their phase groups (as previously discussed in Section 7.1.1). However, when using routed couplings, the SHIL process can cause their phases to suddenly change.

Examples of such cases are provided in Figure 7.29. The top waveform shows the intended behavior. The

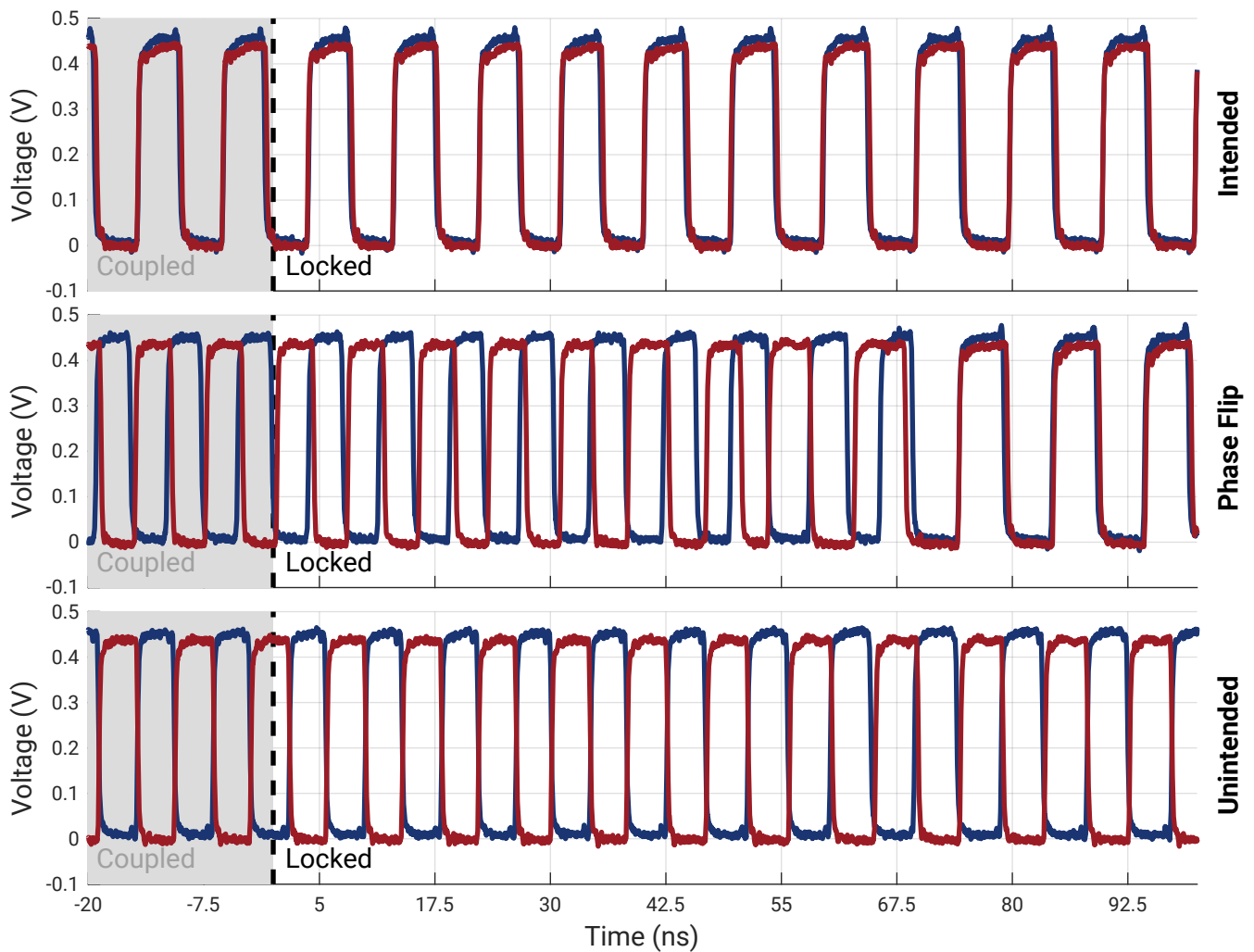


Figure 7.29.: Exemplary waveforms of the transition from the coupled state in the locked state. The behavior at the top is as intended, where the oscillators couple in-phase and stay in-phase when locked. The waveform in the middle shows a phase flip, where the oscillators are undesirably in the anti-phase, although the coupling is set to in-phase. When locked, they surprisingly change to in-phase (as intended). Opposed to that, the oscillators stay in the undesired anti-phase in the bottom waveform.

oscillators couple in-phase and then stay in this phase when locked. The actual alignment to the SHIL clock is hardly visible because almost no change in phase or frequency is necessary, very similar to the local coupling. The middle waveform exhibits a phase flip during SHIL, where the oscillators initially undesirably couple in the opposite configuration. During the SHIL process, they flip their phase and transition from the undesired anti-phase to the intended in-phase configuration. The bottom waveform shows a similar scenario, where the phases, however, stay in the undesired anti-phase when locked. The fourth possible combination, coupling as intended but having a phase flip when locking, was not observed.

The actual behavior depends on the delay, coupling strength and frequency. Depending on these settings, the outcome (settling as intended, locking as intended) is following a probability, where both possible outcomes

---

can be observed. In the case of the proposed system, only the success rate after locking can be systematically analyzed (as previously shown in Figure 7.27). Hence, it cannot be distinguished if the oscillators couple as intended and stay in their phase when locked, or couple undesirably but have a phase flip when locked, leading to the correct phase. The coupled cases were studied off-chip using an oscilloscope since the variations in frequency make a systematic approach using the PDC impractical<sup>†18</sup>.

### Limitation of the Pairwise Coupling Analysis

While this analysis helps to understand the behavior of the oscillators, it provides limited insight on its consequences on the OIM computing. From an operation point of view, only the optimal choice of the supplementary delay setting is of interest, which depends on the physical distance between the connected oscillators. Hence, a rule to select the optimal delay based on the coupling success rate and frequency shift is needed to generate a lookup table from the before provided data. Although there is a strong indication suggesting that a high success rate in combination with a coupled frequency close to the free-running frequency is desirable, a trade-off is required because this desired point does not exist. It needs to be verified that the previously identified delay setting actually leads to good computational results. Therefore, the following section investigates this based on solving a small problem only with routable connections and mixed problems using local and routing connections.

### 7.5.3. Routing Only Connections

A simple problem is solved two times by using either the routing or the local connections. The results are then compared to quantify the effectiveness of the routing connections. Since there are only 4 routing connections per oscillator available, randomly generated benchmark problems follow a grid structure with horizontal and vertical connections similar to the simple benchmark problem set. Such a structure has an optimal solution of a chessboard pattern, which satisfies all routing connections, as described in Section 6.3. Hence, it is a good baseline to ensure correct and reliable operation of the couplings. The nodes of the grid are spread apart to achieve a routing distance of three grid-units so that a 100-node problem in a 10x10 grid is formed.

The results are shown in Figure 7.30, where the same set of problems are solved for all available delay compensation settings. A sweep over the full range is provided in Figure 7.30a, which includes a random Monte Carlo sampling as dotted baseline. This sweep impressively shows the relevance for the correct compensating delay. Without the supplementary delay, the system could perform like a random number generator for delays around 0 ns and over 1.7 ns and is therefore useless<sup>†19</sup>. The best computing accuracy is obtained for delay settings in the range of 0.7 ns to 1.2 ns as highlighted in Figure 7.30b. Depending on how strictly a tolerable range of the best cut is defined, a 500 ps wide window results in usable behavior. The success rate of the pairwise coupling within this interval stays above 99%, while the frequency is slightly higher than the free-running frequency. At 0.7 ns it is 109.1 MHz and drops to 104.6 MHz at 1.2 ns. It needs to be emphasized that the shown range only applies for routings with a distance of 3 nodes operating at 100 MHz. Based on this data, frequency shifts below approximately 110 MHz (10% above free-running) and a success rate above approximately 99% should be selected. Since the oscillators only experience the delay after the compensating delay line and not the physical delay, the criteria also apply to any other distances.

---

<sup>†18</sup>The frequency is different whether the oscillators couple as intended or not. Also the delay and coupling strength affect the frequency as well. When coupled, each pair will settle in a slightly different individual frequency.

<sup>†19</sup>Technically, consistently performing a cut of 0 would be equivalent to finding the minimum solution. Hence, a very low normalized cut could be useful as well.

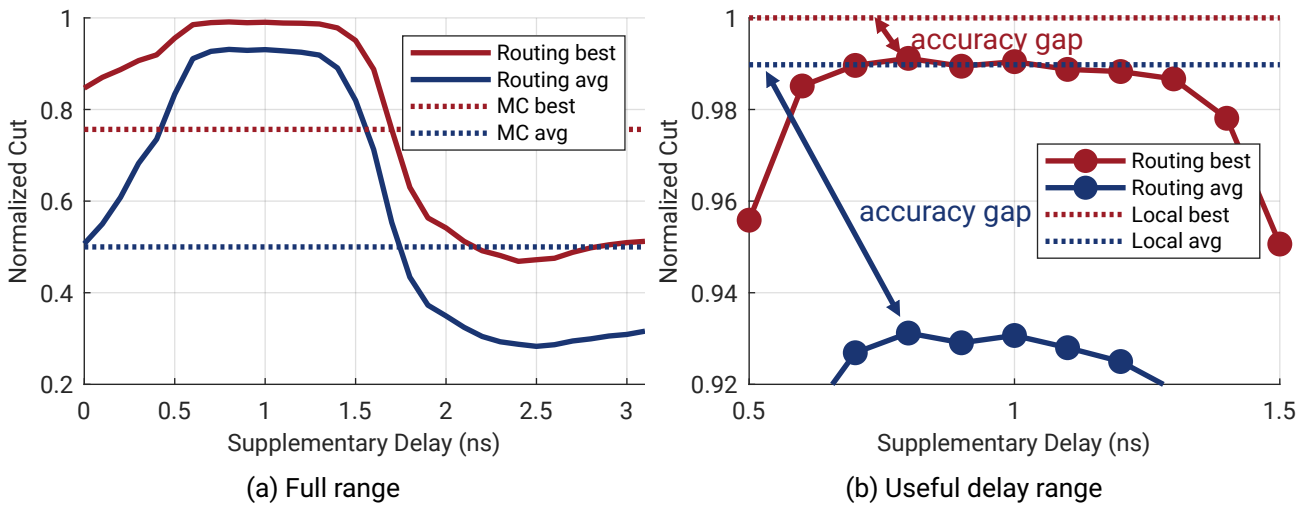


Figure 7.30.: Solving a small 10x10 problem using only routing channels. (a) The comparison with a Monte Carlo baseline shows, that the computing works only well within a small range. (b) Despite choosing the best possible operating parameters, the accuracy of the obtained solutions is worse than that of the same problem with local connections.

#### 7.5.4. Mixed Routing Problems

When using such an OIM, local and routing connections will be used side by side. Hence, such problems are evaluated here and compared to the accuracy when solving problems with only local connections. Therefore, random problems are generated, which follow the `difficult` structure for the local connections. The number of additional routing connections is selected based on a density, which defines the number of used routing couplers. Each routing connection has a randomly chosen distance between 3 and 20 grid-units, where especially long routes cause congestion and blockages of the tracks in the routing network. Hence, only a limited number of routing couplers can be used concurrently. Figure 7.31 shows the histograms of the obtained average and best cut for those routing problems. Routing densities of 5%, 10%, and 20% are shown<sup>†20</sup>, where a similar problem with just local connectivity is shown as dashed line. With increasing routing density, the accuracy slightly reduces, which can be expected due to the discussed lower performance of routing connections. This additionally confirms that the routing connections perform slightly worse than local connections. It should be noted, that these problems with long distance routing connections might be considerable more difficult to solve than comparable `difficult` problems. The commercial solver Gurobi [137] needed considerable more time to compute the optimal solution<sup>†21</sup>. A baseline test of the same problem with disabled routing network was conducted. The normalized cut then drops to 0.87 on average, with results between 0.82 and 0.89. This undoubtedly confirms, that the routing connections are beneficial for solving a problem.

<sup>†20</sup>The total number of connections is as follows: local connections: random from 2310 to 5390; routing connections: 5%: 129, 10%: 258, 20%: 517

<sup>†21</sup>The runtime exceeded feasible limits. Since problems from the set did not finish after 2 days, a time restriction of 4 hours was applied. 97% of the problems from the `difficult` set were solved within this time. The best solution obtained within the time restriction was treated as best-known solution. Typically it has a guaranteed gap of less than 1% to the actual optimum and is therefore noted 'best-known' solution in Figure 7.31.

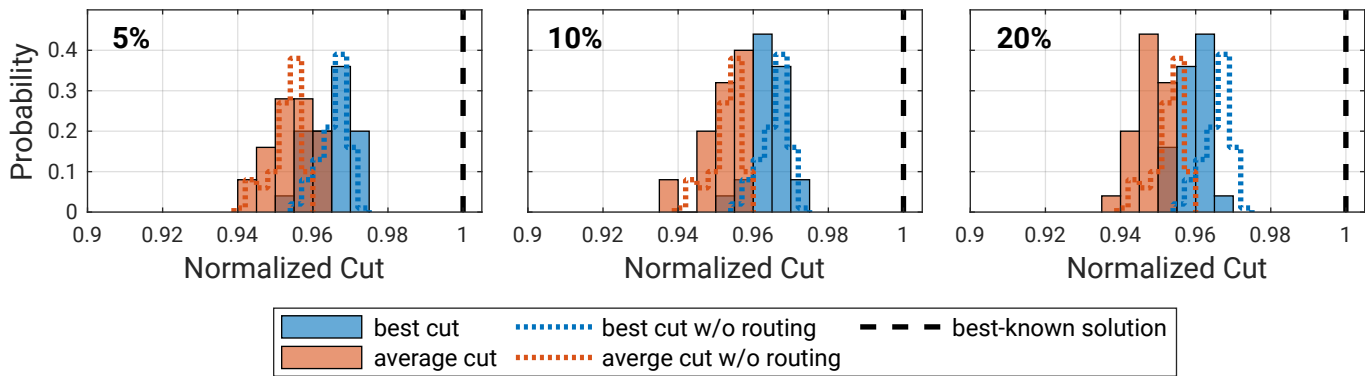


Figure 7.31.: Evaluation of mixed problems, which use local and routing connections. The shown percentage is the density of active routing couplers. The dotted lines show the accuracy of similar problems with only local connections as a baseline.

### 7.5.5. Routing Conclusion

As it is already discussed at the design, the unavoidable propagation delay of long routed connections affects the phase dynamics despite the compensation scheme. This scheme itself is successful, as the connections become ineffective without any proper delay calibration. Without calibration they might just provide an accuracy like a random number generator. Once generated, a lookup table provides the needed delay settings for successful compensation. The compensating delay has a tolerance range of approximately 500 ps ( $\approx 5\%$  of the oscillation period). After calibration, the behavior of the connections within the compensation range does not vary with the actual routing length. So, there is no difference in the behavior between a short and long connection. The direct comparison of the same problems solved utilizing either the routing connections or the local connections shows, that there is a gap of approximately 6% for the average computation outcome. In case of the mixed problems with local and routing connections, the accuracy slightly reduces with increasing share of the routing connections. Compared to an equivalent problem with only local connections, the accuracy gap is less because there are considerably fewer routing connections than local connections. However, this successfully highlights that the routing connections are well-suited to simplify the embedding. Further research is needed to improve the routing connections to be on-par with the local connections.

## 7.6. Benchmarks

To judge the performance of the OIM concept, benchmarks for the Glowworm chip are provided. A comprehensive overview containing the four benchmark sets *simple*, *difficult*, *bias* and *weighted* is provided in Figure 7.32, which is extensively discussed in the next two paragraphs. For convenience, the topologies of the benchmark sets are illustrated at the side as well. Each problem topology was solved for a small (10x10 nodes), medium (25x25 nodes) and full-size (40x36) network using the Glowworm chip. The variety of network sizes should provide an idea of how the solutions will be affected when scaling up the network size. A comprehensive analysis using the first generation Firefly is available in the related publication [154].

The obtained average cut shows that the solution accuracy stays on a similar level independent of the network size. However, the accuracy of the best cut declines with increasing size, which is caused by a reduction of the spread within a problem. This can be expected as the search space grows. Also, the standard deviation of the normalized cut, when repeatedly solving a problem, tends to show a reduction for greater problem sizes. Since the average solution is approximately constant over the size, this causes the best solution

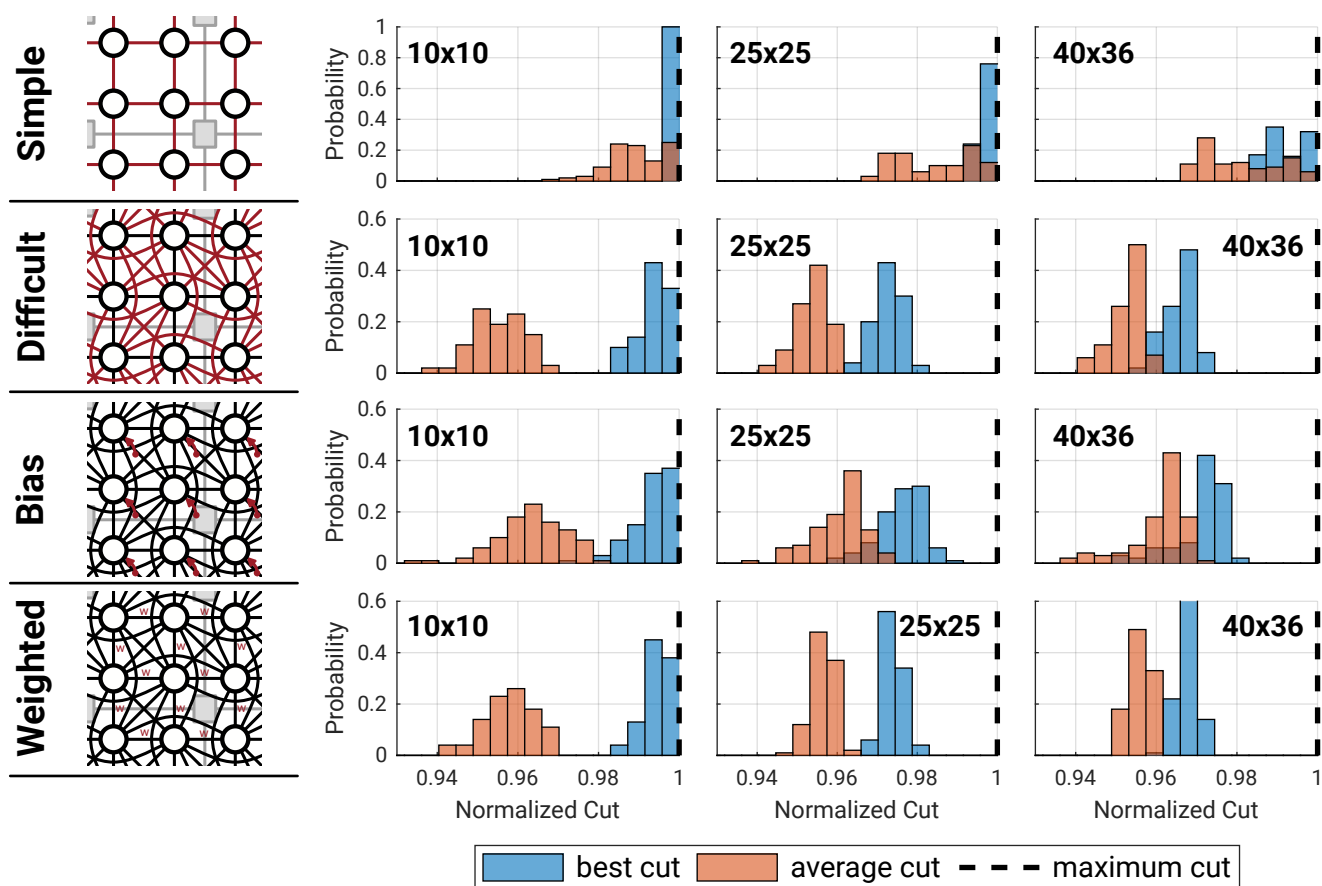


Figure 7.32.: Histograms of the benchmark sets for a small (10x10 nodes), a medium (25x25 nodes), and the full-size (40x36 nodes) network. The distribution of the average and best run from 500 computations per problem are recorded as already discussed in Section 7.2 to account for the non-deterministic behavior.

to lower. It needs to be considered that the timing schedule was identical for all sizes, so it cannot be judged how the computing time will increase with larger problems. However, even the needed coupling time for the largest problem size investigated in Section 7.3.5 is so short that the available time resolution in this work would limit such an analysis. So, the dependency of the computing time on the network size needs further analysis using much larger systems.

Next, the four different benchmark sets can be compared. It should be noted that different operating parameters for each benchmark set were used, but the parameters were held constant for each problem within a benchmark set. As expected, the simple set provides the best accuracy by far. Since it has a trivial optimal solution of a chessboard pattern with all edges being cut, it acts as a good test for the successful coupling of the oscillators. Although the oscillators are not aware of this trivial solution, they reach a solution that is close to the optimum and occasionally even the optimum. It can be suspected that internal clustering effects prevent them from reaching the optimal solution reliably as discussed in Section 7.3.3. Due to the fully local, parallel computing, it is still not trivial to find the optimal solution for the OIM principle. The performance between the difficult and weighted problems are almost identical, while the bias set performs slightly better. This indicates that the reference clock coupling principle of the spin-bias connections was successfully implemented in hardware. Additionally, this underlines, that the system is also well suited to solve QUBO



---

problems, which usually need the spin-bias connections. To compare both chip generations, problems of the difficult benchmark set achieve similar accuracy for the Glowworm and Firefly design<sup>†22</sup>. However, in the first generation Firefly the weighted problems perform 3 to 5% worse than the difficult problems. This is not the case in the second generation Glowworm, where weighted problems do not lose performance. Especially the improved coupler architecture is likely responsible for this success. All-in-all, the designed OIM can solve problems with near to optimal solutions. In the case of a simple or small system, even the optimal solution is sometimes found.

## Power Consumption

The power consumption during a computation is highly dynamic and thus difficult to measure directly<sup>†23</sup>. Instead, the power consumption is obtained by measuring the current in every operating state. The actual power consumption is then calculated based on the timing schedule. In general, the consumed power depends considerably on the optimization problem to be solved. Since every connection of the problem is assigned to a physical coupler circuit, the power consumption is directly influenced by the number of edges of the optimization problem. Furthermore, the operating parameters, such as the coupling strength, which uses different settings based on the problem set, respectively problem topology, also directly affect the power consumption. So, the difficult problem set was chosen for a realistic approximation, since it has a high coupling strength and dense connectivity. Among the 100 problems of the set, the energy per computation varies between 404.7 nJ and 466.3 nJ. On average, a single computation takes 437.3 nJ, which at a computing time of 950 ns relates to an average power consumption of 460.3 mW. The total power consumption is dominated by the analog circuit, which accounts for 383.8 mW thereof. Another considerable contribution is the idle power consumption of the chip. With disabled oscillators and couplers but applied clocks at 100 MHz, the Glowworm chip consumes 103.4 mW. This is a combination of the analog system that consumes 38.4 mW and the digital circuits that consume 64.6 mW, where leakage of the routing network is a significant contribution. When optimizing the design for power efficiency and less aggressively using the fast standard cells for the routing network, the static power could be considerably reduced.

When breaking down the power consumption to the individual components, an oscillator node uses approximately 113.3  $\mu$ W and a local coupling connection approximately 23.2  $\mu$ W. The power consumption of a routing connection depends on the length and scales approximately linear with the distance. Based on a linear regression, the basic power consumption of a routable connection is 57.9  $\mu$ W with an additional 2.9  $\mu$ W per grid-unit of distance. The measured consumption of a short connection with a distance of 3 grid-units is 62.5  $\mu$ W and for a long connection with 30 grid-units distance 145.5  $\mu$ W.

## Die-to-die variation

While the analysis was based on the same die so far, the variations between the fabricated dies also need to be discussed. These die-to-die variations give an impression how severely the performance could be affected by different random mismatch, which indicates the robustness of the design. Unfortunately, all ten available dies are from the same wafer, so that there are limited process variations. No results for die #9 could be obtained due to a defective carrier PCB. The analysis focuses on the four benchmark sets at the full network size of 40x36 nodes. For better visibility, the distributions of the average and best cut are reduced to a box

---

<sup>†22</sup>The network connectivity varies and is not fully compatible between both generations.

<sup>†23</sup>Just measuring the average power consumption when repeatedly solving a problem does include the power consumption for the data transmission. Since the data transmission is approximately three orders of magnitude slower, such an approach does not provide meaningful data for the power consumption of the computation.

plot per die as shown in Figure 7.33. All operating parameters were selected based on die #1 and are kept identical for all. Overall, the performance and variation are very similar between the dies. Die #7 performs slightly worse than the others, while die #6 performs slightly better. However, the overall changes are rather small compared to the potential impact of suboptimal operating parameters such as the coupling strength. The frequency variations between the dies, available in Appendix A.1.4, provide some information about the process of the fabricated dies. Only die #10 has a distinctive lower average frequency before calibration, indicating a slightly slower process. However, this die provides very similar performance as the others. The best (#6) and worst (#7) die for the computing accuracy show no abnormalities in their frequency. A similar variance between dies was also obtained for the first generation Firefly. Out of 10 tested dies, 9 had very similar performance, while one had a slight accuracy loss of 0.7% for unknown reasons.

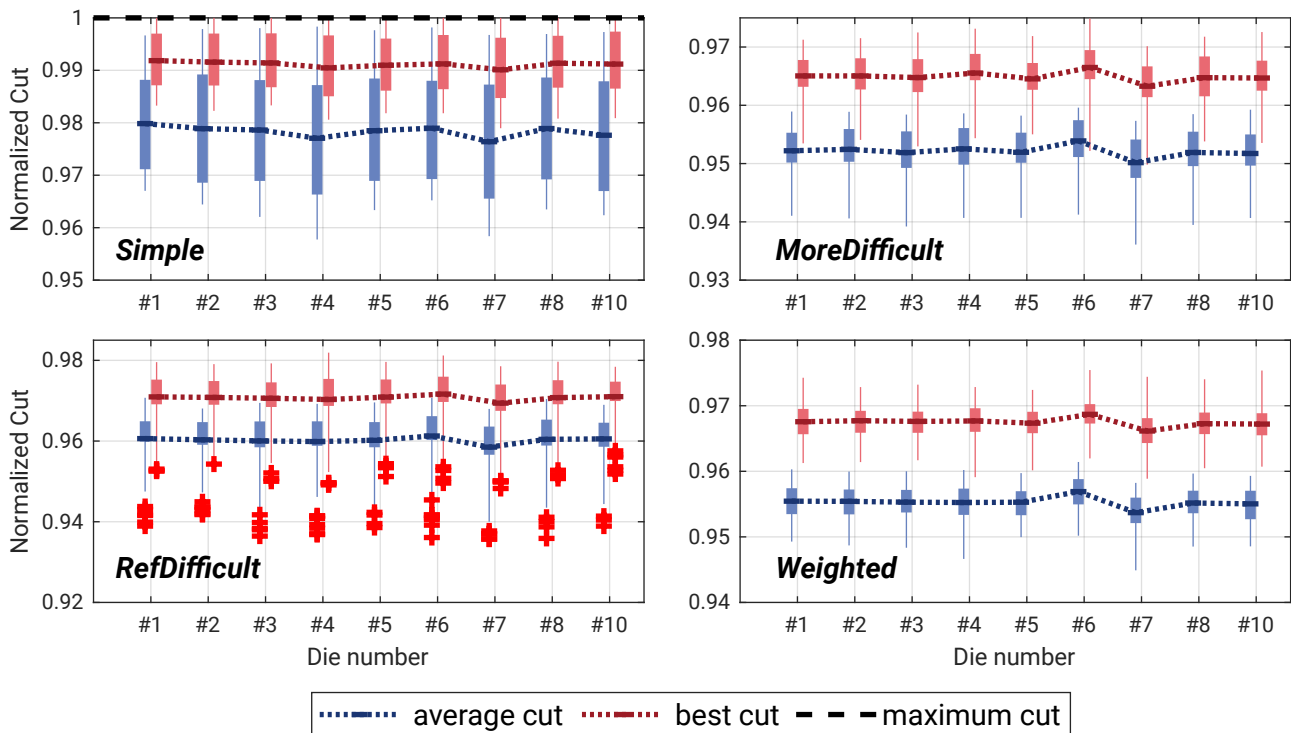


Figure 7.33.: Comparison of 9 tested Glowworm dies using the benchmark sets simple, difficult, bias, and weighted. The boxplots show the variation among each set, where the blue shows the average cut and the red the best cut out of 500 runs per problem.

It can be confirmed that there are variations between the Glowworm dies based on the previous analysis. The next step is therefore to investigate how the performance for each individual problem varies. Either a problem might work well on one die but might perform worse on another die, or one die might consistently provide better accuracy for all problems. In the first case, random mismatch could act in favor of some problems, while process variations could explain a shift across all problems in the set. In reality, there will likely be a mix between both effects. In order to evaluate these two principles, the individual performance of the problems is analyzed across the dies.

Usually, a balanced two-way analysis of variance (ANOVA) would be used to statistically separate the effect of manufacturing mismatch of the dies and optimization problem. Due to the large variations between the individual runs on the same die compared to the small difference between the different dies, and the non perfectly Gaussian distributed data, such an analysis has a high uncertainty. Therefore, a graphical analysis

approach is provided in Figure 7.34 instead. For each chip, the individual problems are sorted using their average computing accuracy, which makes the analysis independent of a general shift of the performance across all dies. Then, the average ranking across all dies is calculated for each problem as a baseline. A scatter plot shows the relative performance variation between the individual dies, by plotting the individual rank against the average rank of all dies. Any shift in the solution accuracy affecting all problems is canceled by the ranking and only the relative performance compared to the other problems is considered. A point above the diagonal means that the problem is solved better by this die than on average. Below the diagonal, it is solved worse than the average. The closer all points are to the diagonal, the more consistent they are solved across all dies. A drawback of this rank-based scatter plot approach is that it does not consider the absolute difference in solution accuracy between the individual problems. A tiny change in absolute accuracy might lead to a considerable change in rank if there are plenty of problems with similar accuracy<sup>†24</sup>.

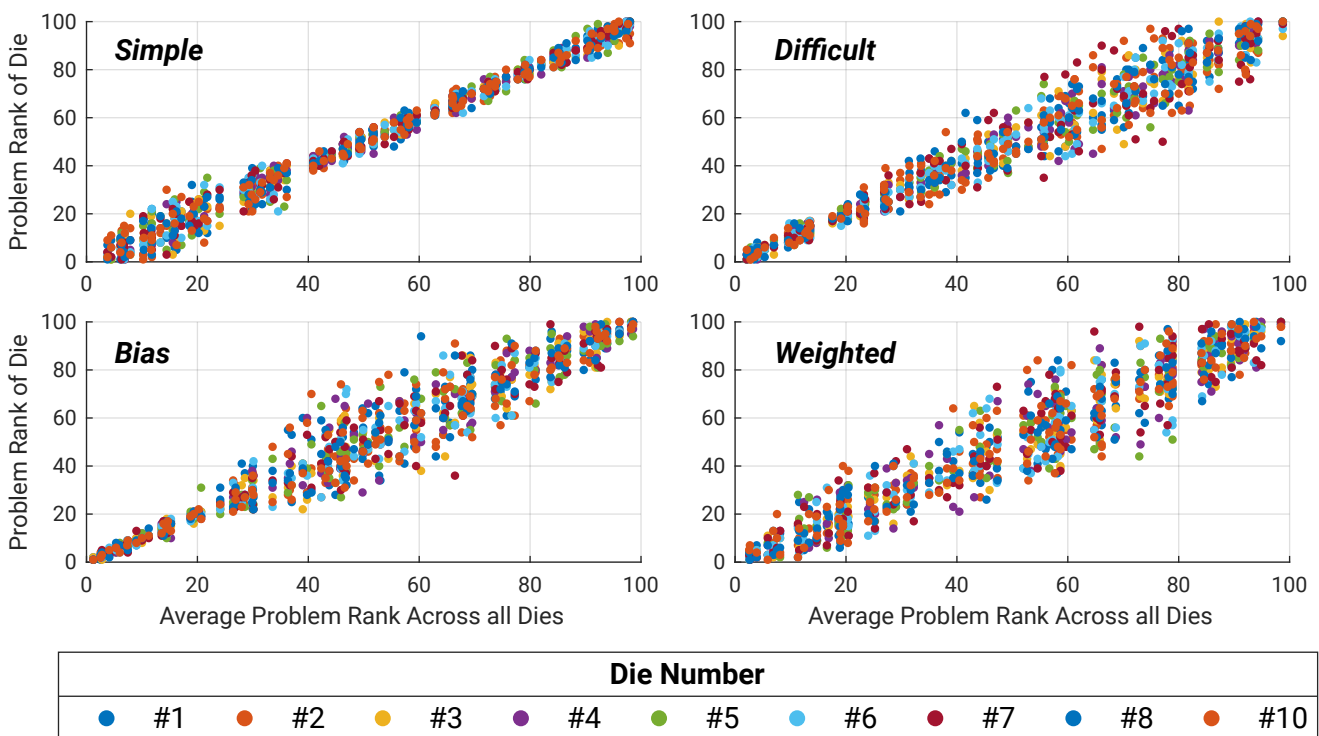


Figure 7.34.: Scatter plot of the individual problems solved on multiple dies. A point on the diagonal indicates a similar accuracy among problems. Above means it is solved better and below worse.

The individual simple problems are consistently solved similarly. So, the random mismatch between dies has less impact. For the difficult problems there is a stronger scattering among the upper half of the problems. The bias set shows a scattering among the intermediate problems, which are neither solved very well nor very bad. The weighted set has the most scattering of all problems. Overall, it can be concluded that the actual optimization problem has a higher impact on the accuracy than random variation acting in favor or against it. The effect of random mismatch on the problems seems to be the smallest for the simple type and the strongest for the weighted. This is expected as weights are severely affected by mismatch. However, the effects of random mismatch will likely average out for such a large system of 1440 nodes having even more coupling connections.

<sup>†24</sup>This is the reason that the average rank can have gaps and might not span from 1 to 100. Also, it explains the horizontal gaps in the scatter plot of Figure 7.34.

---

## 7.7. Comparison with Other Works

A comparison with other CMOS OIM implementations is provided in Table 7.2. The network topology shows two diverging trends. All-to-all connected networks with 16, 30, 48, and 59 nodes have been proposed, while 6 (including this work) larger sparse networks with up to 1968 nodes have been presented. This work has the second most oscillators and edges. However, two different designs are leading in each of these categories. Additionally, it is the only implementation featuring routing channels. Also, it is the only sparse implementation supporting the  $h_i$  term of the Ising model with the spin-bias connections. While two all-to-all implementations offer the  $h_i$  term, the implementation is considerably simpler in such networks and can not be applied to sparse networks by design. Unique is the coupling principle, which can be adjusted by the DACs and offers the highest resolution among all proposed work. Competing designs use either capacitors, inverters, tri-state buffers, or transmission-gates, which are connected in parallel to implement weights. The operating frequency has a wide range from 45 kHz to 1 GHz, which roughly determines the computing time. Only two implementations are faster than this work but exhibit considerably worse accuracy. The proposed work has one of the best accuracies, which was considered without post-processing to get the actual performance of the chip. The work of Lo et al. [130] reports the probability of reaching an accuracy above 95%. Since this work usually reaches more than 94% (or 95%) on average, a high probability will be reached in that metric. The work of Karpuzcu et al. [127] uses a 99% accuracy criterion, which makes the comparison more difficult. Similar to the other work, both of these all-to-all system have slightly lower accuracy for sparse networks (density 10%). The reported data without post-processing at low densities reaches this 99% criterion with a probability of approximately 0.1 - 1%. Similar-sized 100 node graphs in this work also reach 99% and more. Despite the same graph density, the comparison might still not be fair due to underlying differences in the topology. Hence, it cannot be clearly judged which design has a higher accuracy under comparable circumstances. Overall, this work has one of the highest reported accuracy among all other proposed implementations. The absolute power consumption is higher by a factor of 10, but there are less differences when considering the power per node. Considering the high frequency, resulting quick computation, and the number of edges, the energy per computation is more competitive. Despite using an advanced technology node, this design is more area-hungry. This is expected, as the design was successfully optimized for accuracy, with less priority given to power and area consumption. Optimization regarding area and power could be applied to make this design more competitive. Post-processing techniques were intentionally not used to focus on the true performance of the hardware. Additionally, their slow computation time could negate the benefit of quick computations of less than a microsecond.

Some works reported an increasing trend of the accuracy with the computing time<sup>†25</sup> [83], [127]. So, the accuracy is improved when letting the oscillators couple longer, creating a time vs. accuracy trade-off. Although a similar tendency was noticed in simulation, this could not be experimentally confirmed in this work (see Section 7.3.5). No apparent reason for this behavior can be given, but it can be suspected that it is caused by the strong coupling. Such strong couplings were chosen because of simulation and experimental results. The ‘chaotic’ like response from the strong coupling might prevent a gradual settling. When observing the oscillators with an oscilloscope, complex problems usually do not settle into a stable phase state when coupled. Instead, the phase difference is changing frequently. More sophisticated strategies, such as modulating of the coupling strength or SHIL strength during computation, might be an answer and could improve the accuracy further. Additionally, the accuracy might be improved by applying digital post-processing of the phases as it is common in other works. Another interesting approach is the proposed principle by Sikhakollu et al. using a sampled coupling scheme instead of time-continuous coupling, as presented in their works [141].

---

<sup>†25</sup>These tend to slowly settle towards a certain accuracy level below the optimal solution.

	Bashar et al.'20 [82], [142]	Ahmed et al.'21 [84], [143]	Mallick et al.'21 [83]	Moy et al.'22 [85]	Lo et al.'23 [130]	Karpuzcu et al.'24 [127]	Liu et al.'24 [144]	Firefly	Glowworm
Technology	65nm	65nm	65nm	65nm	65nm	65nm	180nm	28nm	28nm
Number of nodes	30	560	600	1,968	48	59	1,024	400	1,440
Number of edges	435	1,585	29,076	7,607	1,236	1,711	3,810	1,482	11,724
Weight resolution	fixed	fixed	fixed	5 levels	≈4-bit + sign	≈4-bit + sign	9 levels	6-bit + sign	4-bit + sign
Topology	all-to-all	hexagonal	intra/inter tile	king's graph	all-to-all	all-to-all	king's graph	king's graph	local + routing
Edges per node	29	6	111	8	47+1	58+1	8	8	15+1
Oscillator type	relaxation	ring	relaxation	ring	ring	ring	ring	diff. ring	diff. ring
Coupling principle	capacitor	inverter	capacitor	transmission gate	transmission gate	tri-state inverter	capacitive	DAC driven stage	DAC driven stage
Ising model support	$J_{ij}$	$J_{ij}$	$J_{ij}$	$J_{ij}$	$J_{ij}$ and $h_i$	$J_{ij}$ and $h_i$	$J_{ij}$	$J_{ij}$	$J_{ij}$ and $h_i$
Operating frequency	45 kHz	118 MHz	45 kHz	1 GHz	≈ 28.5 MHz	5.19-5.36 MHz	18 MHz	50 - 200 MHz	100 MHz
Computing time	-	200 ns <sup>*1</sup>	6.4 ms	50 ns	≈ 2-14 μs	≈ 100 μs <sup>*2</sup>	2.2-100 μs	713 ns	950 ns
Solution accuracy <sup>*3</sup>	-	≥ 82%	≈ 80% <sup>*4</sup>	up to 95% <sup>*5</sup>	NR <sup>*6</sup>	NR <sup>*7</sup>	≈83% <sup>*8</sup>	≥ 95%	≥ 94%
Power	1.76 mW	23 mW	25 mW	42 mW	16-105 mW	8.2-10.0 mW	30 mW	302.9 mW	460.3 mW
Power per node	58.7 μW	41 μW	41.7 μW	21.3 μW	0.33 - 2.19 mW	139 - 169 μW	29.3 μW	757.3 μW	319.7 μW
Chip area	1.44 mm <sup>2</sup>	1.44 mm <sup>2</sup>	4 mm <sup>2</sup>	2.1 mm <sup>2</sup>	1.8 mm <sup>2</sup>	4.8 mm <sup>2</sup>	>12.8 mm <sup>2</sup> <sup>*9</sup>	2.2 mm <sup>2</sup>	4.6 mm <sup>2</sup>

<sup>\*1</sup> Simulated; Oscillator coupling only

<sup>\*2</sup> The time-to-solution depends on the desired accuracy. The reported number is explicitly mentioned by the author.

<sup>\*3</sup> Accuracy measurement methodology varies between publications. If reported, digital post-processing was excluded to emphasize the hardware accuracy. Largest reported problem size chosen

<sup>\*4</sup> Without additional digital post-processing

<sup>\*5</sup> Oscillator network performance only, digital post-processing excluded

<sup>\*6</sup> Reported accuracy not compatible: 95% accuracy threshold used

<sup>\*7</sup> Reported accuracy not compatible: 99% accuracy threshold used

<sup>\*8</sup> At the full network size

<sup>\*9</sup> Core area based on reported unit layout

Table 7.2.: Comparison of integrated CMOS OIMs including both generations Firefly (related publications: [151], [155], [158]) and Glowworm (related publications: [154], [150], [153]) of this work. The accuracy can only be compared to a limited degree due to varying size, connectivity, and edge weights as well as post-processing. The 16-node all-to-all connected network with 5-bit weight resolution operating at 1-4 MHz by Delacour et al. is not included due to the limited space of the table [145].

---

## 7.8. Further Directions

While this work demonstrated the effectiveness of oscillator-based Ising machines and their accuracy combined with low power consumption and fast computation, more research needs to be done. Probably one of the main interests is to improve the solution accuracy. Because an average solution in this work typically reaches more than 94% of the optimal solution, there is still room for improvement. It is desirable to reach even higher levels of precision, approaching the optimal solution. So, further fine-tuning of the developed circuits and improving the robustness against manufacturing variations needs to be done in the future. The coupler circuit is probably the most crucial element as it determines the oscillator interaction and thus computing accuracy, while it dominates the area at the same time. It needs to provide a high resolution for the interaction between oscillators, which needs to be precise despite random manufacturing mismatch. However, calibrating the strength of such an interaction is difficult, because it can not be directly measured like a frequency. Since the strength between couplers can be compared, an algorithm could balance the strengths iteratively. At the same time, large real-world problems demand even larger systems with more discrete variables and coefficients. So, more oscillators and couplers are needed, which will further increase the area dominance of the couplers. A detailed discussion about scaling up the approach of this work is available in the related paper [154].

Another direction for further research is the integration into a powerful software/hardware flow. To be beneficial in real-world applications, a capable software ecosystem is needed. From a user perspective, an optimization problem will be given, where additional information about the desired accuracy and time constraint might be provided. First, a software interface would need to transform the problem into an Ising or other supported form. Then, the problem must be embedded into the hardware network before the actual computation can be started. The embedding might have to combine multiple physical oscillators to form discrete variables with greater connectivity. Multiple computations can be done and even the operating parameters tuned to improve the accuracy of the solutions. Lastly, the obtained discrete states based on the phase angles could be further post-processed. For such a flow, the problem transformation and embedding might potentially introduce additional inaccuracies. Especially the limited weight resolution and reaching non-optimal solutions need to be considered by the software. Consequently, more research is needed on software integration and its impact on precision when solving optimization problems. Furthermore, it is desirable to keep the software overhead as small as possible in order to fully benefit from the speed and energy advantages of OIMs. The software and hardware can be tailored to each other's needs to achieve the best possible performance.

### 7.8.1. Post-processing

Other works already use post-processing [83] of the obtained spins to improve the solution accuracy. However, such post-processing will likely add a considerable amount of time and energy for the computation. An OIM itself can be faster than 1  $\mu$ s, as demonstrated in this work. Clearly, it is more desirable to get high accuracy directly from the OIM computation instead of applying post-processing. Hence, the focus of this work is geared more towards the OIM hardware implementation, while the topic of post-processing was not considered.

Common approaches use gradient descent methods to flip spins and keep changes if the objective is improved [83]. If the oscillator computation yields good results, one would expect that flipping any spin would not change the objective. Every oscillator should couple in the optimal phase respectively spin as to minimize the Hamiltonian. Hence, flipping just one spin should result in the same or a worse Hamiltonian for the system. Consequently, such a simple post-processing could be interpreted as a form of error correction algorithm.

More sophisticated post-processing algorithms might even further improve the accuracy. They are not limited to a pure mathematical optimization of the found solution, but could leverage unused potential in

---

the analog computing nature. For example, the actual phase information with its higher resolution could be used instead of just focusing on the spins. Although the assignment into both discrete phase groups is done based on the measured phase angles, they could offer valuable information for the post-processing. Not all oscillators are in phase angles that clearly follow the two distinct phase groups. Oscillators exhibiting phase angles between those distinct phase groups provide an opportunity for post-processing their state and the corresponding connections. Additionally, oscillators failing to lock can be identified and subject to further post-processing.

### 7.8.2. Hybrid Approaches

One of the limiting factors are, that current OIMs operate fully parallelized in hardware. This means that every discrete variable is assigned to at least one physical oscillator and every coefficient to a physical coupler. Although this is one of the reasons for the high computing speed, it is also a limiting factor. Problems can only be solved if they can be embedded into the physical hardware network. In contrast, digital algorithms sacrifice computing time but can solve much larger problems, which would theoretically only be limited by the memory. In practice, the runtime when solving NP-hard problems gets quickly unfeasible. Consequently, a logical next step would be to combine OIMs with digital processors and algorithms. An algorithm executed on a digital machine could partition large problems into small enough sections for the OIM system. The obtained results are then stitched back together as solution for the much larger problem.

D-Wave systems, which uses their quantum annealer as underlying system, have successfully demonstrated a strategy like this [146]. However, the OIMs offer considerable advantages compared to their quantum annealer. Since they can be manufactured with CMOS technologies like virtually all processors nowadays, a seamless integration between traditional digital high-performance computing systems and OIMs is possible. For example, integration on the same die or as a chiplet in the same package is technically possible. Even a PCB-based assembly of the digital computing system with the OIMs could be advantageous compared to a quantum annealer. The OIM system operates at room temperature, avoiding costly cryogenic cooling. Additionally, the electrical interface between the circuits is much simpler, energy efficient, and could have higher data rates. No signals need to cross between room and cryogenic temperatures.

Consequently, this might be one of the most promising approaches in the near future. OIMs could even be implemented within edge devices despite their limited footprint and available power budget. Additionally, they benefit from available high-end manufacturing technology, which can quickly bring such systems to market. This sets the hybrid approach apart from potential competing quantum technologies or novel devices.

### 7.8.3. Application Specific Oscillator-based Solver

In reference to the ASIC implementation discussed in this work, an interesting research direction is something which could be called application specific oscillator-based solver (ASOS). This could be envisioned as OIM network implementation, which is tailored for specific applications needing to solve optimization problems quickly. Ideally, the optimization problems have a similar structure, where weights or some connections vary frequently and need a quick solution. A very good example of such an application is the multi-user multiple-input-multiple-output (MU-MIMO), as proposed by the group of J. Roychowdhury [147], [148]. In cellular communications, multiple transmitting and receiving antennas are used to transfer data. Since the channel can be characterized, recovering the transmitted data from the received superimposed signal is an optimization problem. The group has shown, that OIMs are well suited for this task and could outperform

---

state of the art algorithms by delivering a much better bit error rate. Clearly, finding a solution fast and energy-efficiently is a critical requirement.

However, not every application is well suited for OIMs, as the requirements should match the advantage of OIMs. Solutions for an optimization problem should be needed fast (micro/milliseconds), while a near-optimal solution is sufficient. If guaranteed optimal solutions are required, OIMs might never be suited for that. Additionally, the structure should be similar between consecutive problems to avoid complex, time-consuming embedding for each new problem. For example, small differences like changed weights or a few altered connections will likely be much easier and quicker to apply than doing a complete embedding. The total number of discrete variables and coefficients should be kept reasonable as scaling up OIMs to massive sizes introduces new challenges.

Providing applications as examples is outside the scope of this thesis, so just speculations of possible areas are mentioned. High throughput applications in the field of communication systems, which need to quickly identify or distribute data, might be well suited. Also the emerging field of artificial-intelligence might strongly benefit from ASOSs. Fast and energy-efficient computations are important for training of the models. Additionally, the network topology is already known, which simplifies the design. However, it is unlikely that OIMs will ever be able to keep up with the immense parameters of, for example, large language models. As of this writing, the GPT-3 of OpenAI has 175 billion parameters [149]. The size of GPT-4 by OpenAI is proprietary and not publicly available. However, it will have more parameters than the previous version.

In conclusion, the rough structure of optimization problems needs to be more or less known, so that the number of oscillators and their coupler connectivity can be made to meet the needs. Additionally, this knowledge should be usable for an efficient embedding.

## 7.9. Conclusion of the Evaluation

A comprehensive experimental analysis of the performance and properties of the developed OIMs has been shown. The waveforms show that the oscillators can couple within a few periods and quickly lock when ramping up the SHIL. Solutions are computed within just 950 ns, where a very conservative time of 200 ns is used to let the oscillators couple. The already impressive computation time could be roughly halved with a few modifications in future designs. Critical for a high accuracy are the coupling strength, SHIL strength and SHIL frequency. When carefully chosen, the average outcome of a computation reaches more than 94% of the optimal solution. By selecting the best out of multiple computations of the same problem, even higher accuracy, around 97% to 98%, can be obtained. The best accuracy is only reached within a small range of parameters. It gradually gets worse and eventually breaks down when exceeding this range. Fine-tuning these parameters for the problem topology of the simple, difficult, weighted, and bias benchmark sets was done to ensure the best performance.

The random mismatch is likely one of the limiting factors in this design. Hence, the individual oscillator frequencies are calibrated. A frequency standard deviation of 1% is sufficient to avoid a noteworthy reduction in accuracy. However, this limit depends on the coupling strength and desired accuracy level. Higher coupling strengths can tolerate more mismatch and weaker couplings are more susceptible to frequency variations. The coupler circuits are also affected by mismatch, which directly influences the computation. Despite configuring the same strengths, some couplers are significantly stronger than others. This was illustrated with multiple examples and is presumably a significant factor limiting the computing accuracy. Calibration schemes could be developed in the future, which is challenging because the strength can not be directly measured. Multiple mechanisms affected by mismatch, such as the injected current by the coupler, the sensitivity of the oscillator, and potential phase offset, contribute to the indirectly measurable coupling strength.



To the best of the authors knowledge, the spin-bias connections were first proposed for a sparse system in this work. They have proven to be successful and actually slightly increased the accuracy. They tend to be slightly stronger than the local connections at the same weight configuration. However, this can simply be compensated by a scaling factor. The behavior of the routing connections is more complicated. Depending on the delay, they might operate as intended or exhibit undesired behavior. A routing connection might couple exactly in the opposite phase as intended, significantly increasing the frequency by roughly 50%. Unfortunately, this coupling behavior follows a probability function depending on the delay. This results in a blurry line between intended and undesired behavior. Even when choosing the correct delay compensation, there might be a chance that the oscillators do not couple as intended. Nevertheless, the delay compensation is crucial, as these connections otherwise stop working and essentially perform similar to a random number generator. Overall, the routing connections operate less reliably than local connections, leading to slightly reduced accuracy. However, they can provide connections of almost any oscillator across the chip on demand, which simplifies the problem embedding.

Since new circuits were specifically created for the emerging concept of OIMs, they can be further improved to reach even higher computing accuracy. There is still a noteworthy gap to the optimal solution. As shown by the mismatch analysis, such an analog implementation is limited to low resolutions. Even with more sophisticated calibration algorithms, it is likely very challenging to achieve usable weight resolutions of 8 bit or even 16 bit.

Although only Ising machines were considered in this work, the circuits and findings might be transferable with some modifications to other computing schemes, such as oscillatory neural networks (ONNs), Hopfield networks, or restricted Boltzmann machines. The active coupler topology could be used for unidirectional coupling when separating the enable/disable signal of the back-to-back connected stages. The spin-bias connections could also be used to set the oscillators into predefined states. Even continuous initial phase angles are possible, when using the spin-bias connections to couple an oscillator simultaneously in-phase and anti-phase with different strengths.

### Conclusion of the Performance Evaluation & Discussion

- The **computation is non-deterministic** and the obtained solutions when repeatedly solving the same problem are considerably different.
- Optimization problems with up to 1440 nodes can be computed within 950 ns at a power consumption of 460 mW, achieving typically **more than 94%** of the optimal solution. By selecting the best solution out of multiple runs, an even higher accuracy is reached.
- Choosing the operating parameters such as **coupling strength, SHIL strength, and SHIL frequency is crucial**. Depending on the optimization problem connectivity, these parameters might need fine tuning.
- The routing network proved **effective, but is less reliable** than local connections. The proposed delay calibration is mandatory to ensure successful operation.
- The **spin-bias** connections, which implement the  $h_i \cdot \sigma_i$  term of the Ising model, perform **on par** with the regular **oscillator-to-oscillator** connections.
- **Manufacturing mismatch** is one of the challenges and **limitations** of such an analog computing principle. While the oscillator frequency can be easily compensated for, the mismatch in the coupling strength is one of the remaining challenges.

---

## 8. Conclusion

---

This work presents the design of oscillator-based Ising machines to solve combinatorial optimization problems fast and energy-efficiently. It first discusses the simulation and design, followed by a comprehensive experimental performance evaluation. A behavioral model is proposed based on the sensitivity of the oscillators and the current injection of the couplers for fast simulations. The work shows the design of a differential oscillator and an active coupler circuit, whose strength can be set via DACs, which are the crucial elements determining the computing accuracy and dominating the chip area. To verify and evaluate the performance of this design, two chips were fabricated using a 28 nm CMOS technology. The results identified that the coupling strength, SHIL strength, and SHIL frequency are crucial operating parameters to obtain the best performance, reaching near-optimal solutions. Furthermore, random manufacturing mismatch was confirmed to be another key factor for the accuracy that was already considered during the design. Multiple examples show how mismatch affects the coupling and causes significant strength variations. However, the on-chip frequency calibration successfully prevented a negative influence from the oscillator frequency variations on the computing accuracy, because the standard deviation is less than 1%. The results of this work confirm that OIMs can be integrated compactly on a single silicon die and demonstrate impressive computing speeds below one microsecond and low energy consumption.

### Scientific Findings

The main scientific contributions are based on the OIM prototypes including, their experimentally demonstrated performance. The latest generation features 1,440 oscillators with 11,724 coupling circuits on a 4.6 mm<sup>2</sup> die in a 28 nm technology node. It can solve an optimization problem in 950 ns, while consuming roughly 320  $\mu$ W per node. Depending on the problem topology, an average outcome reaches more than approximately 94% of the optimal solution, which is a considerable accuracy improvement compared to existing sparse OIM implementations. This work is the first to introduce flexible routing channels, which can connect any oscillators on the chip. Also, it provides the only known implementation to date of the Ising model's  $h_i$  term in sparse networks.

- **How can oscillators be designed to effectively solve combinatorial optimization problems?**

The OIM implementation is based on a 4-stage differential ring oscillator, which is very sensitive to coupling currents and is robust against common-mode disturbances. However, the couplers shaping the interaction of the oscillators are similarly important for the computation. An active coupler circuit, consisting of two back-to-back connected stages, which are based on a double differential transistor pair, is proposed. The strength of the interaction can be adjusted by a DAC setting the coupler bias current.

- **What are the critical design factors for such systems to find solutions that are as close to the global optimum as possible?**

The experimental evaluation demonstrated that the right balance of coupling strength, SHIL strength, and SHIL frequency is crucial. Unfortunately, the best choice of these operating parameters depends on the

---

optimization problem connectivity, but they could be adjusted during runtime. Random manufacturing mismatch is one of the key challenges. The oscillator frequencies are calibrated to achieve a standard deviation below 1% of the nominal frequency, preventing a noteworthy reduction of the computation accuracy. However, the random mismatch of the coupling connections causes considerable variations in the actual strength despite being identically configured. Because these variations essentially change the optimization problem represented in hardware, the accuracy of the solution is reduced. Furthermore, this limits the effective weight resolution to a few bits due to the analog computing principle.

- **How can this approach be scaled up to thousands of oscillators?**

The system-level network of OIMs is a central trade-off in the design. All-to-all connectivity is limited to a very small number of oscillators due to the immense area requirement of the needed couplers. Sparse networks, however, require a very time-consuming embedding. As a trade-off, the usage of FPGA-like routing channels is presented. A compensation scheme is implemented due to the unavoidable delay when transmitting an oscillator signal across long distances. However, the phase dynamics of the coupled oscillators are changed, leading to slightly less reliable coupling compared to local connections. The physical implementation is based on layout tiles, which contain all necessary circuits for an oscillator node and get replicated to form the network of the desired size. Circuit-wise, the buffered oscillator approach mitigates loading effects and allows for a simple implementation of different connectivity.

- **What are the strengths, weaknesses, and limits of oscillator-based computing?**

This approach can solve optimization problems in 950 ns or less while consuming just 320  $\mu$ W per node. The computing outcome itself is non-deterministic and solutions reach more than 94% of the optimal solution on average. By selecting the best out of multiple computing runs, a higher accuracy of typically 97% to 98% is achieved. However, the analog principle, in combination with noise and manufacturing mismatch, limits the possible weight resolution. Such OIMs are well suited for applications that need to frequently solve similar structured optimization problems with near-optimal solutions.

## Outlook

Future research can be categorized into three directions: accuracy improvements, up-scaling, and software support. Since both proposed OIM chips show a small gap to the optimal solution, future work can improve the circuits to increase the accuracy. A coupling strength calibration scheme is likely beneficial to compensate for the random mismatch of the coupling connections. A digital post-processing, which might even consider the individual phases of the oscillators, could be used to improve the accuracy.

The number of nodes and couplers needs to be further scaled up to be competitive with digital heuristics and demonstrate how the computing time will increase with the network size. Either larger chips can be produced, or the circuits can be improved for area compactness. Novel devices such as VO<sub>2</sub> oscillators and memristors could be promising to improve the network density, save considerable area, and reduce the power consumption of OIMs. Their characteristics might even lead to better accuracy.

The supporting software stack currently limits the capabilities to solve real-world optimization problems of the presented OIMs. Powerful embedding techniques and a close software/hardware co-design are desirable. Hybrid algorithms might help to overcome the inherent size and connectivity limitations of OIMs. These could partition large problems into small pieces, which can be solved by an OIM used as a hardware accelerator. Furthermore, specific applications that leverage the full potential and advantages of OIMs need to be identified. A high-performance but also area and energy-efficient system could be obtained by tailoring an OIM to a specific application.

---

## References

---

- [1] Gurobi Optimization LLC and M. North, *Tackling One of the World's Most Complex Scheduling Problems*, [https://www.gurobi.com/case\\_studies/national-football-league-scheduling/](https://www.gurobi.com/case_studies/national-football-league-scheduling/), Case Studies: NFL. (visited on 10. Aug. 2024).
- [2] G. E. Moore, “Cramming more components onto integrated circuits”, *Electronics*, vol. 38, no. 8, 1965.
- [3] G. Moore, *Gordon Moore: The Man Whose Name Means Progress - IEEE Spectrum*, <https://spectrum.ieee.org/gordon-moore-the-man-whose-name-means-progress>, May 2015. (visited on 10. Aug. 2024).
- [4] T. Wang, L. Wu, and J. Roychowdhury, “Late Breaking Results: New Computational Results and Hardware Prototypes for Oscillator-based Ising Machines”, in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, Jun. 2019, pp. 1–2.
- [5] T. Wang, “Novel Computing Paradigms using Oscillators”, Ph.D. dissertation, University of California, Berkeley, 2019.
- [6] NEOS-Guide, *Optimization Problem Types*, <https://neos-guide.org/guide/types/>. (visited on 5. Feb. 2024).
- [7] B. Korte and J. Vygen, *Combinatorial Optimization: Theory and Algorithms* (Algorithms and Combinatorics). Berlin, Heidelberg: Springer, 2018, vol. 21, ISBN: 978-3-662-56039-6. DOI: 10.1007/978-3-662-56039-6.
- [8] S. A. Cook, “The complexity of theorem-proving procedures”, in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ser. STOC '71, New York, NY, USA: Association for Computing Machinery, May 1971, pp. 151–158, ISBN: 978-1-4503-7464-4. DOI: 10.1145/800157.805047.
- [9] L. Fortnow, “The status of the P versus NP problem”, *Communications of the ACM*, vol. 52, no. 9, pp. 78–86, Sep. 2009, ISSN: 0001-0782, 1557-7317. DOI: 10.1145/1562164.1562186.
- [10] R. M. Karp, “Reducibility among Combinatorial Problems”, in *Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations, Held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and Sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, ser. The IBM Research Symposia Series, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds., Boston, MA: Springer US, 1972, pp. 85–103, ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2\_9.
- [11] C. W. Commander, “Maximum cut problem, MAX-CUTMaximum Cut Problem, MAX-CUT”, in *Encyclopedia of Optimization*, C. A. Floudas and P. M. Pardalos, Eds., Boston, MA: Springer US, 2009, pp. 1991–1999, ISBN: 978-0-387-74759-0. DOI: 10.1007/978-0-387-74759-0\_358.
- [12] F. Hadlock, “Finding a Maximum Cut of a Planar Graph in Polynomial Time”, *SIAM Journal on Computing*, vol. 4, no. 3, pp. 221–225, Sep. 1975, ISSN: 0097-5397, 1095-7111. DOI: 10.1137/0204019.

- 
- [13] C. Kuratowski, “Sur le problème des courbes gauches en Topologie”, *Fundamenta Mathematicae*, vol. 15, pp. 271–283, 1930, ISSN: 0016-2736, 1730-6329. DOI: 10.4064/fm-15-1-271-283.
- [14] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt, “An Application of Combinatorial Optimization to Statistical Physics and Circuit Layout Design”, *Operations Research*, vol. 36, no. 3, pp. 493–513, Jun. 1988, ISSN: 0030-364X. DOI: 10.1287/opre.36.3.493.
- [15] Y. Boykov and M.-P. Jolly, “Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images”, in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 1, Vancouver, BC, Canada: IEEE Comput. Soc, 2001, pp. 105–112, ISBN: 978-0-7695-1143-6. DOI: 10.1109/ICCV.2001.937505.
- [16] *World Traveling Salesman Problem*, <https://www.math.uwaterloo.ca/tsp/world/index.html>, University of Waterloo. (visited on 29. Jan. 2024).
- [17] H. Tahami and H. Fakhravar, “A Literature Review on Combining Heuristics and Exact Algorithms in Combinatorial Optimization”, *European Journal of Information Technologies and Computer Science*, vol. 2, no. 2, pp. 6–12, Apr. 2022, ISSN: 2736-5492. DOI: 10.24018/compute.2022.2.2.50.
- [18] G. B. Dantzig, “Linear Programming and Extensions”, vol. 25, no. Princeton Landmarks in Mathematics and Physics, 1963. DOI: 10.1515/9781400884179.
- [19] G. B. Dantzig, “Origins of the simplex method”, in *A History of Scientific Computing*, New York, NY, USA: Association for Computing Machinery, Jun. 1990, pp. 141–151, ISBN: 978-0-201-50814-7.
- [20] V. Klee and G. J. Minty, “How good is the simplex algorithm?”, in *Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; Dedicated to the Memory of Theodore S. Motzkin)*, Academic Press, New York-London, 1972, pp. 159–175.
- [21] G. J. Woeginger, “Exact Algorithms for NP-Hard Problems: A Survey”, in *Combinatorial Optimization — Eureka, You Shrink!: Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*, ser. Lecture Notes in Computer Science, M. Jünger, G. Reinelt, and G. Rinaldi, Eds., Berlin, Heidelberg: Springer, 2003, pp. 185–207, ISBN: 978-3-540-36478-8. DOI: 10.1007/3-540-36478-1\_17.
- [22] A. H. Land and A. G. Doig, “An Automatic Method of Solving Discrete Programming Problems”, *Econometrica*, vol. 28, no. 3, pp. 497–520, 1960, ISSN: 0012-9682. DOI: 10.2307/1910129.
- [23] M. Padberg and G. Rinaldi, “A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems”, *SIAM Review*, vol. 33, no. 1, pp. 60–100, 1991, ISSN: 0036-1445. JSTOR: 2030652.
- [24] F. Glover, G. A. Kochenberger, and F. S. Hillier, Eds., *Handbook of Metaheuristics* (International Series in Operations Research & Management Science). Boston, MA: Springer US, 2003, vol. 57, ISBN: 978-0-306-48056-0. DOI: 10.1007/b101874.
- [25] C. Blum and A. Roli, “Metaheuristics in combinatorial optimization: Overview and conceptual comparison”, *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, Sep. 2003, ISSN: 0360-0300. DOI: 10.1145/937503.937505.
- [26] P. McMenemy, N. Veerapen, J. Adair, and G. Ochoa, “Rigorous Performance Analysis of State-of-the-Art TSP Heuristic Solvers”, in *Evolutionary Computation in Combinatorial Optimization*, A. Liefooghe and L. Paquete, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2019, pp. 99–114, ISBN: 978-3-030-16711-0. DOI: 10.1007/978-3-030-16711-0\_7.

- 
- [27] Y. Nagata and S. Kobayashi, “A Powerful Genetic Algorithm Using Edge Assembly Crossover for the Traveling Salesman Problem”, *INFORMS Journal on Computing*, vol. 25, no. 2, pp. 346–363, May 2013, ISSN: 1091-9856. DOI: 10.1287/ijoc.1120.0506.
- [28] R. Tinós, K. Helsgaun, and D. Whitley, “Efficient Recombination in the Lin-Kernighan-Helsgaun Traveling Salesman Heuristic”, in *Parallel Problem Solving from Nature – PPSN XV*, A. Auger, C. M. Fonseca, N. Lourenço, P. Machado, L. Paquete, and D. Whitley, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2018, pp. 95–107, ISBN: 978-3-319-99253-2. DOI: 10.1007/978-3-319-99253-2\_8.
- [29] D. Wolpert and W. Macready, “No free lunch theorems for optimization”, *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, Apr. 1997, ISSN: 1941-0026. DOI: 10.1109/4235.585893.
- [30] M. X. Goemans and D. P. Williamson, “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming”, *Journal of the ACM*, vol. 42, no. 6, pp. 1115–1145, Nov. 1995, ISSN: 0004-5411, 1557-735X. DOI: 10.1145/227683.227684.
- [31] W. Lenz, “Beitrag zum Verständnis der magnetischen Erscheinung in festen Körpern”, vol. 21, pp. 613–615, 1920.
- [32] E. Ising, “Beitrag zur Theorie des Ferromagnetismus”, *Zeitschrift für Physik*, vol. 31, no. 1, pp. 253–258, Feb. 1925, ISSN: 0044-3328. DOI: 10.1007/BF02980577.
- [33] R. Peierls, “On Ising’s model of ferromagnetism”, *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 32, no. 3, pp. 477–481, Oct. 1936, ISSN: 1469-8064, 0305-0041. DOI: 10.1017/S0305004100019174.
- [34] B. A. Cipra, “An Introduction to the Ising Model”, *The American Mathematical Monthly*, vol. 94, no. 10, pp. 937–959, 1987, ISSN: 0002-9890. DOI: 10.2307/2322600.
- [35] W. Pauli, *Rapports et Discussions Du Sixième Conseil de Physique Tenu à Bruxelles Du 20 Au 25 Octobre 1930*. Paris, 1932.
- [36] J. Mehra, “Magnetism”, in *The Solvay Conferences on Physics: Aspects of the Development of Physics Since 1911*, J. Mehra, Ed., Dordrecht: Springer Netherlands, 1975, 182–205: 198, ISBN: 978-94-010-1867-8. DOI: 10.1007/978-94-010-1867-8\_7.
- [37] F. Barahona, “On the computational complexity of Ising spin glass models”, *Journal of Physics A: Mathematical and General*, vol. 15, no. 10, p. 3241, Oct. 1982, ISSN: 0305-4470. DOI: 10.1088/0305-4470/15/10/028.
- [38] A. Lucas, “Ising formulations of many NP problems”, *Frontiers in Physics*, vol. 2, Feb. 2014, ISSN: 2296-424X. DOI: 10.3389/fphy.2014.00005.
- [39] D. Stauffer, “Social applications of two-dimensional Ising models”, *American Journal of Physics*, vol. 76, no. 4, pp. 470–473, Apr. 2008, ISSN: 0002-9505. DOI: 10.1119/1.2779882.
- [40] A. Lipowski, D. Lipowska, and A. L. Ferreira, “Phase transition and power-law coarsening in an Ising-doped voter model”, *Physical Review E*, vol. 96, no. 3, p. 032145, Sep. 2017. DOI: 10.1103/PhysRevE.96.032145.
- [41] R. B. Potts, “Some generalized order-disorder transformations”, *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 48, no. 1, pp. 106–109, Jan. 1952, ISSN: 1469-8064, 0305-0041. DOI: 10.1017/S0305004100027419.
- [42] C. Domb, “Configurational studies of the Potts models”, *Journal of Physics A: Mathematical, Nuclear and General*, vol. 7, no. 11, p. 1335, Jul. 1974, ISSN: 0301-0015. DOI: 10.1088/0305-4470/7/11/013.

- 
- [43] P. M. Pardalos and S. Jha, “Complexity of uniqueness and local search in quadratic 0–1 programming”, *Operations Research Letters*, vol. 11, no. 2, pp. 119–123, Mar. 1992, ISSN: 0167-6377. DOI: 10.1016/0167-6377(92)90043-3.
- [44] G. Kochenberger, J.-K. Hao, F. Glover, *et al.*, “The unconstrained binary quadratic programming problem: A survey”, *Journal of Combinatorial Optimization*, vol. 28, no. 1, pp. 58–81, Jul. 2014, ISSN: 1573-2886. DOI: 10.1007/s10878-014-9734-0.
- [45] J. Krarup and P. M. Pruzan, “Computer-aided layout design”, in *Mathematical Programming in Use*, ser. Mathematical Programming Studies, M. L. Balinski and C. Lemarechal, Eds., Berlin, Heidelberg: Springer, 1978, pp. 75–94, ISBN: 978-3-642-00796-5. DOI: 10.1007/BFb0120827.
- [46] L. Iasemidis, P. Pardalos, J. Sackellares, and D.-S. Shiau, “Quadratic Binary Programming and Dynamical System Approach to Determine the Predictability of Epileptic Seizures”, *Journal of Combinatorial Optimization*, vol. 5, no. 1, pp. 9–26, Mar. 2001, ISSN: 1573-2886. DOI: 10.1023/A:1009877331765.
- [47] N. Mohseni, P. L. McMahon, and T. Byrnes, “Ising machines as hardware solvers of combinatorial optimization problems”, *Nature Reviews Physics*, vol. 4, no. 6, pp. 363–379, Jun. 2022, ISSN: 2522-5820. DOI: 10.1038/s42254-022-00440-8.
- [48] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura, “An Accelerator Architecture for Combinatorial Optimization Problems”, *FUJITSU Sci. Tech. J.*, vol. 53, no. 5, 2017. [Online]. Available: <https://www.fujitsu.com/global/documents/about/resources/publications/fstj/archives/vol53-5/paper02.pdf> (visited on 9. Feb. 2024).
- [49] H. Nakayama, J. Koyama, N. Yoneoka, and T. Miyazawa, “Third Generation Digital Annealer Technology”, *Fujitsu*, 2021. [Online]. Available: [https://www.fujitsu.com/global/documents/about/research/techintro/3rd-g-da\\_en.pdf](https://www.fujitsu.com/global/documents/about/research/techintro/3rd-g-da_en.pdf) (visited on 9. Feb. 2024).
- [50] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, “24.3 20k-spin Ising chip for combinatorial optimization problem with CMOS annealing”, in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, Feb. 2015, pp. 1–3. DOI: 10.1109/ISSCC.2015.7063111.
- [51] T. Takemoto, M. Hayashi, C. Yoshimura, and M. Yamaoka, “2.6 A  $2 \times 30$ k-Spin Multichip Scalable Annealing Processor Based on a Processing-In-Memory Approach for Solving Large-Scale Combinatorial Optimization Problems”, in *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*, Feb. 2019, pp. 52–54. DOI: 10.1109/ISSCC.2019.8662517.
- [52] T. Takemoto, K. Yamamoto, C. Yoshimura, *et al.*, “4.6 A 144Kb Annealing System Composed of  $9 \times 16$ Kb Annealing Processor Chips with Scalable Chip-to-Chip Connections for Large-Scale Combinatorial Optimization Problems”, in *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 64, Feb. 2021, pp. 64–66. DOI: 10.1109/ISSCC42613.2021.9365748.
- [53] K. Yamamoto, T. Takemoto, C. Yoshimura, M. Mashimo, and M. Yamaoka, “A 1.3-Mbit Annealing System Composed of Fully-Synchronized 9-board  $\times$  9-chip  $\times$  16-kbit Annealing Processor Chips for Large-Scale Combinatorial Optimization Problems”, in *2021 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, Nov. 2021, pp. 1–3. DOI: 10.1109/A-SSCC53895.2021.9634769.
- [54] K. Yamamoto, K. Ando, N. Mertig, *et al.*, “7.3 STATICA: A 512-Spin 0.25M-Weight Full-Digital Annealing Processor with a Near-Memory All-Spin-Updates-at-Once Architecture for Combinatorial Optimization with Complete Spin-Spin Interactions”, in *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2020, pp. 138–140. DOI: 10.1109/ISSCC19947.2020.9062965.

- 
- [55] K. Yamamoto, K. Kawamura, K. Ando, *et al.*, “STATICA: A 512-Spin 0.25M-Weight Annealing Processor With an All-Spin-Updates-at-Once Architecture for Combinatorial Optimization With Complete Spin-Spin Interactions”, *IEEE Journal of Solid-State Circuits*, vol. 56, no. 1, pp. 165–178, Jan. 2021, ISSN: 1558-173X. DOI: 10.1109/JSSC.2020.3027702.
- [56] Y. Su, J. Mu, H. Kim, and B. Kim, “A 252 Spins Scalable CMOS Ising Chip Featuring Sparse and Reconfigurable Spin Interconnects for Combinatorial Optimization Problems”, in *2021 IEEE Custom Integrated Circuits Conference (CICC)*, Apr. 2021, pp. 1–2. DOI: 10.1109/CICC51472.2021.9431401.
- [57] Y. Su, T. T.-H. Kim, and B. Kim, “FlexSpin: A Scalable CMOS Ising Machine with 256 Flexible Spin Processing Elements for Solving Complex Combinatorial Optimization Problems”, in *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, vol. 65, Feb. 2022, pp. 1–3. DOI: 10.1109/ISSCC42614.2022.9731680.
- [58] Y. Su, J. Mu, H. Kim, and B. Kim, “A Scalable CMOS Ising Computer Featuring Sparse and Reconfigurable Spin Interconnects for Solving Combinatorial Optimization Problems”, *IEEE Journal of Solid-State Circuits*, vol. 57, no. 3, pp. 858–868, Mar. 2022, ISSN: 1558-173X. DOI: 10.1109/JSSC.2022.3142896.
- [59] Y. Su, T. T.-H. Kim, and B. Kim, “FlexSpin: A CMOS Ising Machine With 256 Flexible Spin Processing Elements With 8-b Coefficients for Solving Combinatorial Optimization Problems”, *IEEE Journal of Solid-State Circuits*, pp. 1–12, 2024, ISSN: 1558-173X. DOI: 10.1109/JSSC.2024.3352907.
- [60] T. Kashimata, M. Yamasaki, R. Hidaka, and K. Tatsumura, *Efficient and Scalable Architecture for Multiple-chip Implementation of Simulated Bifurcation Machines*, Nov. 2023. DOI: 10.48550/arXiv.2311.17370.
- [61] S. Sreedhara, J. Roychowdhury, J. Wabnig, and P. Srinath, “Digital Emulation of Oscillator Ising Machines”, in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Apr. 2023, pp. 1–2. DOI: 10.23919/DATE56975.2023.10137084.
- [62] O. Maher, M. Jimenez, C. Delacour, *et al.*, “A CMOS-compatible oscillation-based VO<sub>2</sub> Ising machine solver”, *Nature Communications*, vol. 15, no. 1, p. 3334, Apr. 2024, ISSN: 2041-1723. DOI: 10.1038/s41467-024-47642-5.
- [63] S. Dutta, A. Khanna, J. Gomez, K. Ni, Z. Toroczkai, and S. Datta, “Experimental Demonstration of Phase Transition Nano-Oscillator Based Ising Machine”, in *2019 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2019, pp. 37.8.1–37.8.4. DOI: 10.1109/IEDM19573.2019.8993460.
- [64] S. Dutta, A. Khanna, and S. Datta, “Understanding the Continuous-Time Dynamics of Phase-Transition Nano-Oscillator-Based Ising Hamiltonian Solver”, *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 6, no. 2, pp. 155–163, Dec. 2020, ISSN: 2329-9231. DOI: 10.1109/JXCDC.2020.3045074.
- [65] R. Afoakwa, Y. Zhang, U. K. R. Vengalam, Z. Ignjatovic, and M. Huang, “BRIM: Bistable Resistively-Coupled Ising Machine”, in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2021, pp. 749–760. DOI: 10.1109/HPCA51647.2021.00068.
- [66] Y. Zhang, U. K. R. Vengalam, A. Sharma, M. Huang, and Z. Ignjatovic, “QuBRIM: A CMOS Compatible Resistively-Coupled Ising Machine with Quantized Nodal Interactions”, in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD ’22, New York, NY, USA: Association for Computing Machinery, Dec. 2022, pp. 1–8, ISBN: 978-1-4503-9217-4. DOI: 10.1145/3508352.3549443.



- 
- [67] C. Yu, J. Mu, K. Chai, T. Kim, and B. Kim, “A Continuous-Time Ising Machine using Coupled Inverter Chains Featuring Fully-Parallel One-Shot Spin Updates”, in *2023 IEEE Custom Integrated Circuits Conference (CICC)*, Apr. 2023, pp. 1–2. DOI: 10.1109/CICC57935.2023.10121286.
- [68] D. Pierangeli, G. Marcucci, and C. Conti, “Large-Scale Photonic Ising Machine by Spatial Light Modulation”, *Physical Review Letters*, vol. 122, no. 21, p. 213902, May 2019. DOI: 10.1103/PhysRevLett.122.213902.
- [69] E. Ng, T. Onodera, S. Kako, P. L. McMahon, H. Mabuchi, and Y. Yamamoto, “Efficient sampling of ground and low-energy Ising spin configurations with a coherent Ising machine”, *Physical Review Research*, vol. 4, no. 1, p. 013009, Jan. 2022. DOI: 10.1103/PhysRevResearch.4.013009.
- [70] K. Wu, J. García de Abajo, C. Soci, P. Ping Shum, and N. I. Zheludev, “An optical fiber network oracle for NP-complete problems”, *Light: Science & Applications*, vol. 3, no. 2, e147–e147, Feb. 2014, ISSN: 2047-7538. DOI: 10.1038/lsa.2014.28.
- [71] P. L. McMahon, A. Marandi, Y. Haribara, *et al.*, “A fully programmable 100-spin coherent Ising machine with all-to-all connections”, *Science*, vol. 354, no. 6312, pp. 614–617, Nov. 2016. DOI: 10.1126/science.aah5178.
- [72] T. Inagaki, Y. Haribara, K. Igarashi, *et al.*, “A coherent Ising machine for 2000-node optimization problems”, *Science*, vol. 354, no. 6312, pp. 603–606, Nov. 2016. DOI: 10.1126/science.aah4243.
- [73] A. Marandi, Z. Wang, K. Takata, R. L. Byer, and Y. Yamamoto, “Network of time-multiplexed optical parametric oscillators as a coherent Ising machine”, *Nature Photonics*, vol. 8, no. 12, pp. 937–942, Dec. 2014, ISSN: 1749-4893. DOI: 10.1038/nphoton.2014.249.
- [74] F. Böhm, G. Verschaffelt, and G. Van der Sande, “A poor man’s coherent Ising machine based on opto-electronic feedback systems for solving optimization problems”, *Nature Communications*, vol. 10, no. 1, p. 3538, Aug. 2019, ISSN: 2041-1723. DOI: 10.1038/s41467-019-11484-3.
- [75] Q. Cen, H. Ding, T. Hao, *et al.*, “Large-scale coherent Ising machine based on optoelectronic parametric oscillator”, *Light: Science & Applications*, vol. 11, no. 1, p. 333, Nov. 2022, ISSN: 2047-7538. DOI: 10.1038/s41377-022-01013-1.
- [76] T. Hao, Q. Cen, S. Guan, *et al.*, “Optoelectronic parametric oscillator”, *Light: Science & Applications*, vol. 9, no. 1, p. 102, Jun. 2020, ISSN: 2047-7538. DOI: 10.1038/s41377-020-0337-5.
- [77] T. Honjo, T. Sonobe, K. Inaba, *et al.*, “100,000-spin coherent Ising machine”, *Science Advances*, vol. 7, no. 40, eabh0952, Sep. 2021. DOI: 10.1126/sciadv.abh0952.
- [78] C. McGeoch and P. Farré, “Advantage Processor Overview”, *D-Wave Technical Report*, Jan. 2022. [Online]. Available: [https://www.dwavesys.com/media/3xvdipcn/14-1058a-a\\_advantage\\_processor\\_overview.pdf](https://www.dwavesys.com/media/3xvdipcn/14-1058a-a_advantage_processor_overview.pdf) (visited on 10. Aug. 2024).
- [79] C. McGeoch, P. Farre, and K. Boothby, “The D-Wave Advantage2 Prototype”, *D-Wave Technical Report*, Jun. 2022. [Online]. Available: [https://www.dwavesys.com/media/eixhdtpa/14-1063a-a\\_the\\_d-wave\\_advantage2\\_prototype-4.pdf](https://www.dwavesys.com/media/eixhdtpa/14-1063a-a_the_d-wave_advantage2_prototype-4.pdf) (visited on 10. Aug. 2024).
- [80] H. N. Djidjev, G. Chapuis, G. Hahn, and G. Rizk, *Efficient Combinatorial Optimization Using Quantum Annealing*, Jan. 2018. DOI: 10.48550/arXiv.1801.08653.
- [81] D. Willsch, M. Willsch, C. D. Gonzalez Calaza, *et al.*, “Benchmarking Advantage and D-Wave 2000Q quantum annealers with exact cover problems”, *Quantum Information Processing*, vol. 21, no. 4, p. 141, Apr. 2022, ISSN: 1573-1332. DOI: 10.1007/s11128-022-03476-y.

- [82] M. K. Bashar, A. Mallick, D. S. Truesdell, B. H. Calhoun, S. Joshi, and N. Shukla, "Experimental Demonstration of a Reconfigurable Coupled Oscillator Platform to Solve the Max-Cut Problem", *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 6, no. 2, pp. 116–121, Dec. 2020, ISSN: 2329-9231. DOI: 10.1109/JXCDC.2020.3025994.
- [83] A. Mallick, M. K. Bashar, D. S. Truesdell, B. H. Calhoun, and N. Shukla, "Overcoming the Accuracy vs. Performance Trade-off in Oscillator Ising Machines", in *2021 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2021, pp. 40.2.1–40.2.4. DOI: 10.1109/IEDM19574.2021.9720612.
- [84] I. Ahmed, P.-W. Chiu, W. Moy, and C. H. Kim, "A Probabilistic Compute Fabric Based on Coupled Ring Oscillators for Solving Combinatorial Optimization Problems", *IEEE Journal of Solid-State Circuits*, vol. 56, no. 9, pp. 2870–2880, Sep. 2021, ISSN: 1558-173X. DOI: 10.1109/JSSC.2021.3062821.
- [85] W. Moy, I. Ahmed, P.-w. Chiu, J. Moy, S. S. Sapatnekar, and C. H. Kim, "A 1,968-node coupled ring oscillator circuit for combinatorial optimization problem solving", *Nature Electronics*, vol. 5, no. 5, pp. 310–317, May 2022, ISSN: 2520-1131. DOI: 10.1038/s41928-022-00749-3.
- [86] C. Delacour, S. Carapezzi, M. Abernot, and A. Todri-Sanial, "Energy-Performance Assessment of Oscillatory Neural Networks Based on VO<sub>2</sub> Devices for Future Edge AI Computing", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 7, pp. 10 045–10 058, Jul. 2024, ISSN: 2162-2388. DOI: 10.1109/TNNLS.2023.3238473.
- [87] G. Csaba and W. Porod, "Coupled oscillators for computing: A review and perspective", *Applied Physics Reviews*, vol. 7, no. 1, p. 011 302, Jan. 2020, ISSN: 1931-9401. DOI: 10.1063/1.5120412.
- [88] A. Hajimiri and T. Lee, "Design issues in CMOS differential LC oscillators", *IEEE Journal of Solid-State Circuits*, vol. 34, no. 5, pp. 717–724, May 1999, ISSN: 1558-173X. DOI: 10.1109/4.760384.
- [89] J. Aschoff, "Biologische Uhren", *Gießener Universitätsblätter*, vol. 14, pp. 9–20, 1981, ISSN: 0533-8689. DOI: 10.22029/j1upub-5171.
- [90] R. Adler, "A Study of Locking Phenomena in Oscillators", *Proceedings of the IRE*, vol. 34, no. 6, pp. 351–357, Jun. 1946, ISSN: 2162-6634. DOI: 10.1109/JRPROC.1946.229930.
- [91] L. Paciorek, "Injection locking of oscillators", *Proceedings of the IEEE*, vol. 53, no. 11, pp. 1723–1727, Nov. 1965, ISSN: 1558-2256. DOI: 10.1109/PROC.1965.4345.
- [92] B. Razavi, "A study of injection locking and pulling in oscillators", *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1415–1424, Sep. 2004, ISSN: 1558-173X. DOI: 10.1109/JSSC.2004.831608.
- [93] X. Lai and J. Roychowdhury, "Analytical equations for predicting injection locking in LC and ring oscillators", in *Proceedings of the IEEE 2005 Custom Integrated Circuits Conference, 2005.*, Sep. 2005, pp. 461–464. DOI: 10.1109/CICC.2005.1568706.
- [94] P. Bhansali and J. Roychowdhury, "Gen-Adler: The generalized Adler's equation for injection locking analysis in oscillators", in *2009 Asia and South Pacific Design Automation Conference*, Jan. 2009, pp. 522–527. DOI: 10.1109/ASPDAC.2009.4796533.
- [95] T. Wang, *Sub-harmonic Injection Locking in Metronomes*, Sep. 2017. DOI: 10.48550/arXiv.1709.03886.
- [96] J. Lee and H. Wang, "Study of Subharmonically Injection-Locked PLLs", *IEEE Journal of Solid-State Circuits*, vol. 44, no. 5, pp. 1539–1553, May 2009, ISSN: 1558-173X. DOI: 10.1109/JSSC.2009.2016701.
- [97] J. Lee, H. Wang, W.-T. Chen, and Y.-P. Lee, "Subharmonically injection-locked PLLs for ultra-low-noise clock generation", in *2009 IEEE International Solid-State Circuits Conference - Digest of Technical Papers*, Feb. 2009, 92–93, 93a. DOI: 10.1109/ISSCC.2009.4977323.

- 
- [98] H. Rategh and T. Lee, “Superharmonic injection-locked frequency dividers”, *IEEE Journal of Solid-State Circuits*, vol. 34, no. 6, pp. 813–821, Jun. 1999, ISSN: 1558-173X. DOI: 10.1109/4.766815.
- [99] S. Gierkink, S. Levantino, R. Frye, C. Samori, and V. Boccuzzi, “A low-phase-noise 5-GHz CMOS quadrature VCO using superharmonic coupling”, *IEEE Journal of Solid-State Circuits*, vol. 38, no. 7, pp. 1148–1154, Jul. 2003, ISSN: 1558-173X. DOI: 10.1109/JSSC.2003.813297.
- [100] A. Mazzanti, P. Uggetti, and F. Svelto, “Analysis and design of injection-locked LC dividers for quadrature generation”, *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1425–1433, Sep. 2004, ISSN: 1558-173X. DOI: 10.1109/JSSC.2004.831596.
- [101] A. Mirzaei, M. E. Heidari, R. Bagheri, S. Chehrazi, and A. A. Abidi, “The Quadrature LC Oscillator: A Complete Portrait Based on Injection Locking”, *IEEE Journal of Solid-State Circuits*, vol. 42, no. 9, pp. 1916–1932, Sep. 2007, ISSN: 1558-173X. DOI: 10.1109/JSSC.2007.903047.
- [102] A. Musa, R. Murakami, T. Sato, W. Chaivipas, K. Okada, and A. Matsuzawa, “A Low Phase Noise Quadrature Injection Locked Frequency Synthesizer for MM-Wave Applications”, *IEEE Journal of Solid-State Circuits*, vol. 46, no. 11, pp. 2635–2649, Nov. 2011, ISSN: 1558-173X. DOI: 10.1109/JSSC.2011.2166336.
- [103] C. Hoyer, L. Wetzel, D. Prousalis, J. Wagner, F. Jülicher, and F. Ellinger, “Entrainment of Mutually Synchronized Spatially Distributed 24 GHz Oscillators”, *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 70, no. 7, pp. 2665–2678, Jul. 2023, ISSN: 1549-8328, 1558-0806. DOI: 10.1109/TCSI.2023.3264996.
- [104] C. Hoyer, L. Wetzel, D. Prousalis, J. Wagner, F. Jülicher, and F. Ellinger, “Mutual Synchronization of Spatially Distributed 24 GHz Oscillators up to Distances of 500 m”, *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 9, pp. 3689–3693, Sep. 2022, ISSN: 1558-3791. DOI: 10.1109/TCSII.2022.3176827.
- [105] L. Wetzel, D. Prousalis, F. Jülicher, *et al.*, “How to Implement Mutual Network Synchronization in the Presence of Large Cross-Coupling Delays”, in *2022 Joint Conference of the European Frequency and Time Forum and IEEE International Frequency Control Symposium (EFTF/IFCS)*, Apr. 2022, pp. 1–4. DOI: 10.1109/EFTF/IFCS54560.2022.9850482.
- [106] J. von Neumann, “Non-linear Capacitance or Inductance Switching, Amplifying and Memory Organs”, U.S. Patent 2,815,488, Dec. 1957.
- [107] R. Wigington, “A New Concept in Computing”, *Proceedings of the IRE*, vol. 47, no. 4, pp. 516–523, Apr. 1959, ISSN: 0096-8390. DOI: 10.1109/JRPROC.1959.287311.
- [108] E. Goto, “The Parametron, a Digital Computing Element Which Utilizes Parametric Oscillation”, *Proceedings of the IRE*, vol. 47, no. 8, pp. 1304–1316, Aug. 1959, ISSN: 2162-6634. DOI: 10.1109/JRPROC.1959.287195.
- [109] A. Raychowdhury, A. Parihar, G. H. Smith, *et al.*, “Computing With Networks of Oscillatory Dynamical Systems”, *Proceedings of the IEEE*, vol. 107, no. 1, pp. 73–89, Jan. 2019, ISSN: 1558-2256. DOI: 10.1109/JPROC.2018.2878854.
- [110] T. Wang and J. Roychowdhury, “OIM: Oscillator-Based Ising Machines for Solving Combinatorial Optimisation Problems”, in *Unconventional Computation and Natural Computation*, I. McQuillan and S. Seki, Eds., Cham: Springer International Publishing, 2019, pp. 232–256, ISBN: 978-3-030-19311-9. DOI: 10.1007/978-3-030-19311-9\_19.
- [111] T. Wang, L. Wu, P. Nobel, and J. Roychowdhury, “Solving combinatorial optimisation problems using oscillator based Ising machines”, *Natural Computing*, vol. 20, no. 2, pp. 287–306, Jun. 2021, ISSN: 1572-9796. DOI: 10.1007/s11047-021-09845-3.

- 
- [112] E. Lobe and A. Lutz, “Minor embedding in broken chimera and derived graphs is NP-complete”, *Theoretical Computer Science*, vol. 989, p. 114369, Mar. 2024, ISSN: 0304-3975. DOI: 10.1016/j.tcs.2023.114369.
- [113] A. T. Winfree, “Biological rhythms and the behavior of populations of coupled oscillators”, *Journal of Theoretical Biology*, vol. 16, no. 1, pp. 15–42, Jul. 1967, ISSN: 0022-5193. DOI: 10.1016/0022-5193(67)90051-3.
- [114] Y. Kuramoto and T. Tsuzuki, “Reductive Perturbation Approach to Chemical Instabilities”, *Progress of Theoretical Physics*, vol. 52, no. 4, pp. 1399–1401, Oct. 1974, ISSN: 0033-068X. DOI: 10.1143/PTP.52.1399.
- [115] Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulence* (Springer Series in Synergetics), H. Haken, Ed. Berlin, Heidelberg: Springer, 1984, vol. 19, ISBN: 978-3-642-69689-3. DOI: 10.1007/978-3-642-69689-3.
- [116] A. Demir, A. Mehrotra, and J. Roychowdhury, “Phase noise in oscillators: A unifying theory and numerical methods for characterization”, *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 47, no. 5, pp. 655–674, May 2000, ISSN: 1558-1268. DOI: 10.1109/81.847872.
- [117] A. Hajimiri and T. Lee, “A general theory of phase noise in electrical oscillators”, *IEEE Journal of Solid-State Circuits*, vol. 33, no. 2, pp. 179–194, Feb. 1998, ISSN: 1558-173X. DOI: 10.1109/4.658619.
- [118] F. Pepe, A. Bonfanti, S. Levantino, P. Maffezzoni, C. Samori, and A. L. Lacaita, “An efficient linear-time variant simulation technique of oscillator phase sensitivity function”, in *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, Sep. 2012, pp. 17–20. DOI: 10.1109/SMACD.2012.6339406.
- [119] A. Demir and J. Roychowdhury, “On the Validity of Orthogonally Decomposed Perturbations in Phase Noise Analysis”, *Bell Laboratories, Murray Hills*, 1997.
- [120] A. Demir and J. Roychowdhury, “A reliable and efficient procedure for oscillator PPV computation, with phase noise macromodeling applications”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 2, pp. 188–197, Feb. 2003, ISSN: 1937-4151. DOI: 10.1109/TCAD.2002.806599.
- [121] X. Lai and J. Roychowdhury, “Capturing oscillator injection locking via nonlinear phase-domain macromodels”, *IEEE Transactions on Microwave Theory and Techniques*, vol. 52, no. 9, pp. 2251–2261, Sep. 2004, ISSN: 1557-9670. DOI: 10.1109/TMTT.2004.834579.
- [122] S. Galeone and M. P. Kennedy, “A comparison of simulation strategies for estimating phase noise in oscillators”, in *2017 13th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, Jun. 2017, pp. 213–216. DOI: 10.1109/PRIME.2017.7974145.
- [123] L. Nagel and D. Pederson, “Simulation Program with Integrated Circuit Emphasis (SPICE)”, *16th Midwest Symposium on Circuit Theory*, Apr. 1973.
- [124] G. Gildenblat, Ed., *Compact Modeling*. Dordrecht: Springer Netherlands, 2010, ISBN: 978-90-481-8614-3. DOI: 10.1007/978-90-481-8614-3.
- [125] *Westmere - Microarchitectures - Intel - WikiChip*, [https://en.wikichip.org/wiki/intel/microarchitectures/westmere\\_\(client\)](https://en.wikichip.org/wiki/intel/microarchitectures/westmere_(client)). (visited on 6. Mar. 2024).
- [126] *NVIDIA GK110 GPU Specs in TechPowerUp GPU Database*, <https://www.techpowerup.com/gpu-specs/nvidia-gk110.g136>. (visited on 6. Mar. 2024).

- 
- [127] U. Karpuzcu, H. Cilasun, W. Moy, *et al.*, *COBI: A Coupled Oscillator Based Ising Chip for Combinatorial Optimization*, Apr. 2024. DOI: 10.21203/rs.3.rs-4208492/v1.
- [128] T. Jagielski, R. Manohar, and J. Roychowdhury, *FPIM: Field-Programmable Ising Machines for Solving SAT*, Sep. 2023. DOI: 10.48550/arXiv.2306.01569.
- [129] S. J. E. Wilton, “Architectures and Algorithms for Field-Programmable Gate Arrays with Embedded Memory”, Ph.D. dissertation, University of Toronto, 1997.
- [130] H. Lo, W. Moy, H. Yu, S. Sapatnekar, and C. H. Kim, “An Ising solver chip based on coupled ring oscillators with a 48-node all-to-all connected array architecture”, *Nature Electronics*, vol. 6, no. 10, pp. 771–778, Oct. 2023, ISSN: 2520-1131. DOI: 10.1038/s41928-023-01021-y.
- [131] N. Angeli and K. Hofmann, “A Scalable Fully Synthesized Phase-to-Digital Converter for Phase and Duty-Cycle Measurement of High-Speed Clocks”, in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2018, pp. 1–5. DOI: 10.1109/ISCAS.2018.8351118.
- [132] M. Graber, *Simple Benchmark Set*, Jan. 2024. DOI: 10.6084/m9.figshare.25018898.v1.
- [133] M. Graber, *Difficult Benchmark Set*, Jan. 2024. DOI: 10.6084/m9.figshare.25018892.v1.
- [134] M. Graber, *Bias Benchmark Set*, Jan. 2024. DOI: 10.6084/m9.figshare.25018862.v1.
- [135] M. Graber, *Weighted Benchmark Set*, Jan. 2024. DOI: 10.6084/m9.figshare.25018952.v1.
- [136] M. Graber, *Firefly Benchmark Set*, Jul. 2024. DOI: 10.6084/m9.figshare.26311573.v2.
- [137] Gurobi Optimization LLC, *Gurobi Optimizer Reference Manual*, <https://www.gurobi.com>, 2024.
- [138] Y. L. Maistrenko, O. V. Popovych, and P. A. Tass, “Chaotic attractor in the kuramoto model”, *International Journal of Bifurcation and Chaos*, vol. 15, no. 11, pp. 3457–3466, Nov. 2005, ISSN: 0218-1274. DOI: 10.1142/S0218127405014155.
- [139] Y. Maistrenko, O. Popovych, and P. Tass, “Desynchronization and Chaos in the Kuramoto Model”, in *Dynamics of Coupled Map Lattices and of Related Spatially Extended Systems*, J.-R. Chazottes and B. Fernandez, Eds., Berlin, Heidelberg: Springer, 2005, pp. 285–306, ISBN: 978-3-540-31520-9. DOI: 10.1007/11360810\_12.
- [140] G. Miritello, A. Pluchino, and A. Rapisarda, “Central limit behavior in the Kuramoto model at the “edge of chaos””, *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 23, pp. 4818–4826, Dec. 2009, ISSN: 0378-4371. DOI: 10.1016/j.physa.2009.08.023.
- [141] V. P. S. Sikhakollu, S. Sreedhara, R. Manohar, A. Mishchenko, and J. Roychowdhury, “High Quality Circuit-Based 3-SAT Mappings for Oscillator Ising Machines”, in *Unconventional Computation and Natural Computation*, D.-J. Cho and J. Kim, Eds., Cham: Springer Nature Switzerland, 2024, pp. 269–285, ISBN: 978-3-031-63742-1. DOI: 10.1007/978-3-031-63742-1\_19.
- [142] M. K. Bashar, A. Mallick, and N. Shukla, “Experimental Investigation of the Dynamics of Coupled Oscillators as Ising Machines”, *IEEE Access*, vol. 9, pp. 148 184–148 190, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3124808.
- [143] I. Ahmed, P.-W. Chiu, and C. H. Kim, “A Probabilistic Self-Annealing Compute Fabric Based on 560 Hexagonally Coupled Ring Oscillators for Solving Combinatorial Optimization Problems”, in *2020 IEEE Symposium on VLSI Circuits*, Jun. 2020, pp. 1–2. DOI: 10.1109/VLSICircuits18222.2020.9162869.
- [144] Y. Liu, Z. Han, Q. Wu, *et al.*, “A 1024-Spin Scalable Ising Machine With Capacitive Coupling and Progressive Annealing Method for Combination Optimization Problems”, *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2024, ISSN: 1558-3791. DOI: 10.1109/TCSII.2024.3432799.

- 
- [145] C. Delacour, S. Carapezzi, G. Boschetto, *et al.*, “A mixed-signal oscillatory neural network for scalable analog computations in phase domain”, *Neuromorphic Computing and Engineering*, vol. 3, no. 3, p. 034004, Aug. 2023, ISSN: 2634-4386. DOI: 10.1088/2634-4386/ace9f5.
- [146] G. Bass, M. Henderson, J. Heath, and J. Dulny III, *Optimizing the Optimizer: Decomposition Techniques for Quantum Annealing*, Jan. 2020. DOI: 10.48550/arXiv.2001.06079.
- [147] J. Roychowdhury, J. Wabnig, and K. Pavan Srinath, *Performance of Oscillator Ising Machines on Realistic MU-MIMO Decoding Problems*, Sep. 2021. DOI: 10.21203/rs.3.rs-840171/v1.
- [148] S. Sreedhara, J. Roychowdhury, J. Wabnig, and P. K. Srinath, “MU-MIMO Detection Using Oscillator Ising Machines”, in *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, Oct. 2023, pp. 1–9. DOI: 10.1109/ICCAD57390.2023.10323680.
- [149] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language Models are Few-Shot Learners*, Jul. 2020. DOI: 10.48550/arXiv.2005.14165.

---

## List of Own Publications

---

- [150] M. Graber and K. Hofmann, “An Integrated Coupled Oscillator Network to Solve Optimization Problems”, *Communications Engineering*, vol. 3, no. 1, pp. 1–11, Aug. 2024, ISSN: 2731-3395. DOI: 10.1038/s44172-024-00261-w.
- [151] M. Graber and K. Hofmann, “Firefly: A Versatile Experimental Platform for Oscillator-based Ising Machines”, *IEEE Transactions on Circuits and Systems–I: Regular Papers*, pp. 1–12, Sep. 2024, ISSN: 1558-0806. DOI: 10.1109/TCSI.2024.3448531.
- [152] K. Hofmann and M. Graber, *Invited Talk: Coupled oscillator networks to efficiently solve combinatorial optimization problems*, IEEE Nano Workshop, Gijon, Spain, Jul. 2024.
- [153] M. Graber and K. Hofmann, “Flexible Routing to Overcome the Embedding Bottleneck of Oscillator-based Ising Machines”, in *2023 30th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec. 2023, pp. 1–4. DOI: 10.1109/ICECS58634.2023.10382841.
- [154] M. Graber and K. Hofmann, “An Enhanced 1440 Coupled CMOS Oscillator Network to Solve Combinatorial Optimization Problems”, in *2023 IEEE 36th International System-on-Chip Conference (SOCC)*, Sep. 2023, pp. 1–6. DOI: 10.1109/SOCC58585.2023.10256945.
- [155] M. Graber and K. Hofmann, “A Coupled Oscillator Network to Solve Combinatorial Optimization Problems with Over 95% Accuracy”, in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2023, pp. 1–5. DOI: 10.1109/ISCAS46773.2023.10181365.
- [156] M. Graber, M. Wesp, and K. Hofmann, “A Fast Graph Minor Embedding Heuristic for Oscillator Based Ising Machines”, in *2022 Austrochip Workshop on Microelectronics (Austrochip)*, Oct. 2022, pp. 41–44. DOI: 10.1109/Austrochip56145.2022.9940722.
- [157] M. Graber and K. Hofmann, “Analysis and Design of Oscillator Coupling for Solving Combinatorial Optimization Problems”, in *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Oct. 2022, pp. 1–4. DOI: 10.1109/ICECS202256217.2022.9970974.
- [158] M. Graber and K. Hofmann, “A Versatile & Adjustable 400 Node CMOS Oscillator Based Ising Machine to Investigate and Optimize the Internal Computing Principle”, in *2022 IEEE 35th International System-on-Chip Conference (SOCC)*, Sep. 2022, pp. 1–6. DOI: 10.1109/SOCC56010.2022.9908118.
- [159] M. Graber, N. Angeli, and K. Hofmann, “An Efficient Modeling Approach for Large Ring Oscillator Based Ising Machines”, in *SMACD / PRIME 2021; International Conference on SMACD and 16th Conference on PRIME*, Jul. 2021, pp. 1–4.
- [160] H. Ahrens, M. Graber, and K. Hofmann, “A Precise and Scalable Biasing Network for Massive Parallel Analog Applications”, in *2024 31th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Nov. 2024, pp. 1–4.
- [161] M. Graber and K. Hofmann, “CMOS Design for Oscillator-based Ising Machines”, in *[Under Preparation for Submission] Oscillatory Neural Networks - from Devices to Computing Architectures*, A. Todri-Saniai and G. Csaba, Eds., Springer Nature.

---

## Supervised Theses

---

- [162] R. Koch, “Implementation of a Highly Integrated, Adaptable Sensor Frontend for Pulse Oximetry Applications in UMC 65 nm Technology”, Master of Science Thesis, Technische Universität Darmstadt, Darmstadt, Apr. 2020.
- [163] Y. Guo, “Analysis & Design of Efficient Phase Measurement Concepts for Large Scale CMOS Oscillator Based Annealing Computing”, Master of Science Thesis, Technische Universität Darmstadt, Darmstadt, Dec. 2021.
- [164] S. Nagaraj, “Design of a Versatile High-Performance Simulation Environment for CMOS Oscillator based Annealing Computing”, Master of Science Thesis, Technische Universität Darmstadt, Darmstadt, May 2021.
- [165] L. Roland, “Integrated Differential to Single-ended Conversion of a DC-500MHz Clock Signal”, Bachelor of Science Thesis, Technische Universität Darmstadt, Darmstadt, Sep. 2021.
- [166] R. Huang, “Design of a Versatile Clock-Tree Generator in a Full-Custom Analog & Mixed Signal Design Flow”, Master of Science Thesis, Technische Universität Darmstadt, Darmstadt, Apr. 2022.
- [167] M. Wesp, “Mapping Optimization Problems on Integrated Oscillator-based Ising Networks”, Master of Science Thesis, Technische Universität Darmstadt, Darmstadt, May 2022.
- [168] H. Ahrens, “Scalable, Precise & Area-Efficient Biasing Solutions for Massive Parallel Oscillator Networks”, Master of Science Thesis, Technische Universität Darmstadt, Darmstadt, May 2023.
- [169] T. L. Kessel, “Optimization of a Precise and Area-efficient Frequency Calibration for Oscillator-based Computing”, Bachelor of Science Thesis, Technische Universität Darmstadt, Darmstadt, Jun. 2023.
- [170] B. Li, “Advanced Mapping Strategies for Integrated Oscillator-based Ising Machine Networks”, Master of Science Thesis, Technische Universität Darmstadt, Darmstadt, Mar. 2023.
- [171] D. J. Madzellan, “Development of a Matlab-FPGA Interface for Communication and Timing Control of an Experiment-Setup”, Bachelor of Science Thesis, Technische Universität Darmstadt, Darmstadt, Jan. 2023.
- [172] H. Xu, “HDL-level Design of a High-Performance Hardware Preprocessing for Distributed Oscillator Computing Networks”, Master of Science Thesis, Technische Universität Darmstadt, Darmstadt, Feb. 2023.



---

## Supervised Seminars

---

- [173] X. An, “Test and Verification of the AICD lab chip”, Seminar, Technische Universität Darmstadt, Darmstadt, Sep. 2020.
- [174] M. Wesp, “PCB Design for Automated ASIC Testing”, Seminar, Technische Universität Darmstadt, Darmstadt, Apr. 2020.
- [175] Y. Guo, “Adapting the SDF back-annotation for Asynchronous Signals”, Seminar, Technische Universität Darmstadt, Darmstadt, Apr. 2021.
- [176] W. Lo, “Peltier Based Temperature Control System”, Seminar, Technische Universität Darmstadt, Darmstadt, Apr. 2021.
- [177] M. Wesp, “Testing of Current Starved Inverter for Oscillator-Based Annealing”, Seminar, Technische Universität Darmstadt, Darmstadt, Sep. 2021.
- [178] N. Dudeja, “Implementation of a Schmitt Trigger Oscillator for Solving Combinatorial Optimization Problems”, Seminar, Technische Universität Darmstadt, Darmstadt, Dec. 2022.
- [179] M. N. M. Muhammed Basheer, “Custom Bias Control Circuit for a Chip Testing Platform”, Seminar, Technische Universität Darmstadt, Darmstadt, Sep. 2022.
- [180] W. Xing, “Application of the Maximum Cut to Solve Optimization Problems”, Proseminar, Technische Universität Darmstadt, Darmstadt, Nov. 2022.
- [181] H. Xu, “Custom Frequency Counter for a Chip Testing Platform”, Seminar, Technische Universität Darmstadt, Darmstadt, Aug. 2022.
- [182] T. L. Kessel, “Implementation of Oscillator Calibration with Digital Logic”, Project Seminar, Technische Universität Darmstadt, Darmstadt, Jan. 2023.

# A. Appendix

## A.1. Frequency Calibration

### A.1.1. Calibration Algorithm

The calibration algorithm is based on a frequency measurement using two counters, as shown in Figure A.1. One counter measures a fixed time interval using the reference clock. Another counts the oscillator cycles within this duration. At the start of the calibration, the oscillator is set to their slowest calibration code, which is monotonic increasing by design. Then both counters are started. As soon as the reference counter reaches the target value of 128, both counters are stopped. If the number of oscillation cycles is below 128, then the calibration code is increased. If the number is greater or equal to 128, the calibration code is found and the routine is finished. Consequently, the calibration procedure uses the smallest calibration code exceeding the target frequency. When the maximum calibration code is reached although the oscillator frequency is still below the target, the calibration is stopped as well.

### A.1.2. Calibration Errors

The simple calibration routine was chosen for its simplicity and area efficient implementation. As a disadvantage, it does not guarantee to find the optimal calibration code. A non-optimal value is selected, if the nearest code below the target frequency is closer than the nearest code above the target frequency. Furthermore, the counter-based frequency measurements introduce additional uncertainty due to the inherent quantization

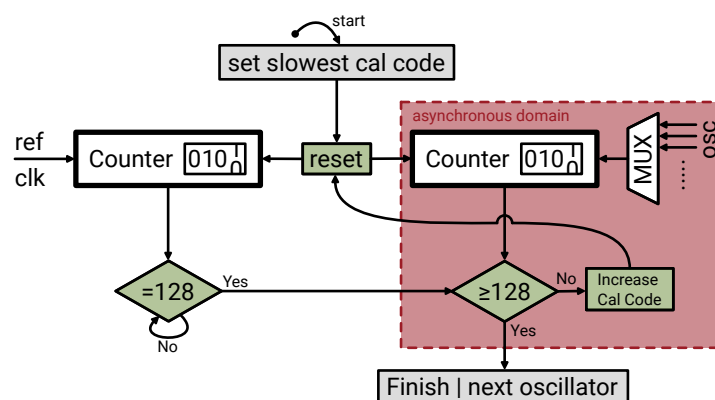


Figure A.1.: Simplified calibration procedure, which is based on two counters for frequency measurement. The first provides a reference duration using the reference clock. The second asynchronous 8-bit counter measures the frequency of the oscillators and increases the calibration code step-by-step until the frequency matches the reference.

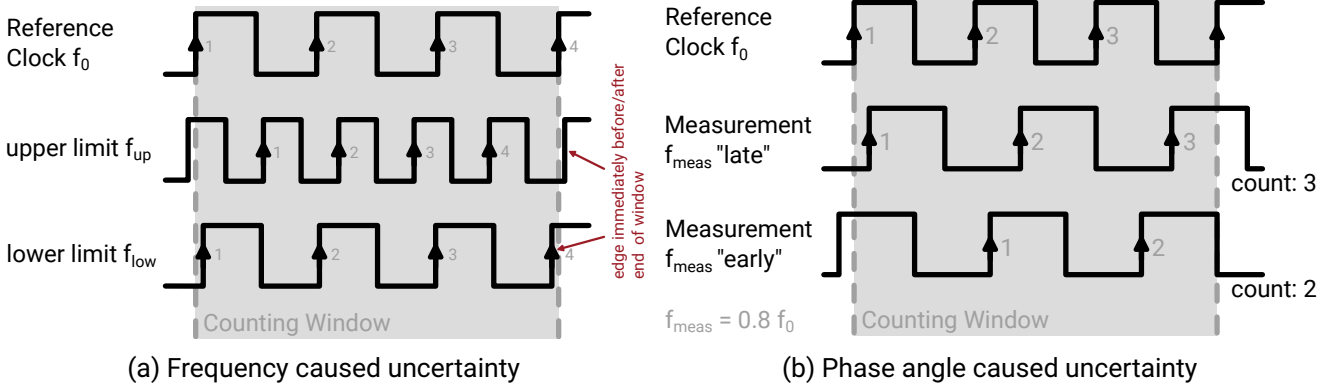


Figure A.2.: The measurement has an uncertainty, since the phase (and frequency) of an oscillator might yield different counts. (a) The number of the recorded edges within the counting window can change by 1 for a signal with the same frequency. (b) Different frequencies can still have the same count.

and the asynchronicity of the oscillator clock. A range of frequencies yields an identical count as illustrated by Figure A.2a. Let  $n_0$  be the number of counted periods of the reference clock  $T_0 = \frac{1}{f_0}$ , then the counting window is  $t_{window} = n_0 * T_0 = \frac{n_0}{f_0}$ . Neglecting possible timing violations, the highest possible frequency  $f_{up}$  having  $n_{meas}$  cycles has its first edge immediately after the starting edge of the counting window and the last edge immediately before the last edge of the counting window, which results in  $f_{up} \approx \frac{n_{meas}+2}{t_{window}} = \frac{n_{meas}+2}{n_0} \cdot f_0$ . The lowest possible frequency  $f_{low}$  has an edge immediately after the start of the counting window and an edge immediately before the end of the timing window. Hence,  $f_{low} \approx \frac{n_{meas}}{t_{window}} = \frac{n_{meas}}{n_0} \cdot f_0$  is the lower bound for a frequency measurement of  $n_{meas}$  cycles. Consequently the measurement frequency is approximately in the range  $f_{meas} \in \{\frac{n_{meas}}{n_0} \cdot f_0, \frac{n_{meas}+2}{n_0} \cdot f_0\}$ . Considering the (unknown) jitter of the reference clock and the oscillator as well as the consequences of timing violation when the timing window starts or ends directly at an edge, the possible frequency range would even increase. However, those errors are small and are neglected for this calibration.

The same frequency can cause two different measurement counts  $n_{meas}$  as illustrated by Figure A.2b. Since the count depends on the starting point of the sampling, this can be treated as de facto random due to the asynchronicity. So, a measurement just indicates, that the actual frequency lies within a range. However, multiple measurements can be averaged to get a more precise frequency measurement.

Besides the errors in the frequency measurement, it should be considered that the frequencies of the oscillators are not stable themselves. They have varying temperature coefficients and slowly drift due to 1/f noise. Therefore, the frequencies will diverge between calibration intervals, which creates a lower bound for a practical calibration precision. A detailed analysis of calibration algorithms discussing the trade-off between area, duration for calibration and the ability to find the optimal code is provided in the supervised thesis by Tim Lukas Kessel [169]. Faster calibration procedures, which guarantee the optimal calibration code, exist at the price of additional area.

### A.1.3. Clock Domain Crossing

The oscillators are asynchronous to the system clock. So, the oscillation cycle counter runs asynchronous as well. Due to the resulting clock domain crossing, accessing the asynchronous counter value might lead to false

readings and/or metastability if the counter is sampled at the same time when its value changes. Hence, a simplistic but safe synchronization mechanism is implemented, as illustrated in Figure A.3. The counter in the asynchronous oscillator domain is started/stopped via the run measurement signal, which is synchronized using a double flip-flop stage. The system clock domain accesses the oscillator counter after a certain hold-off time. This time attributes for the worst-case delay of the synchronizer and an additional cycle for stopping the counter. For that, a minimum clock frequency must be defined, which is derived based on the slowest oscillator frequency in the analog design. If the oscillator is slower than this minimum frequency, no safe clock-domain crossing is possible anymore.

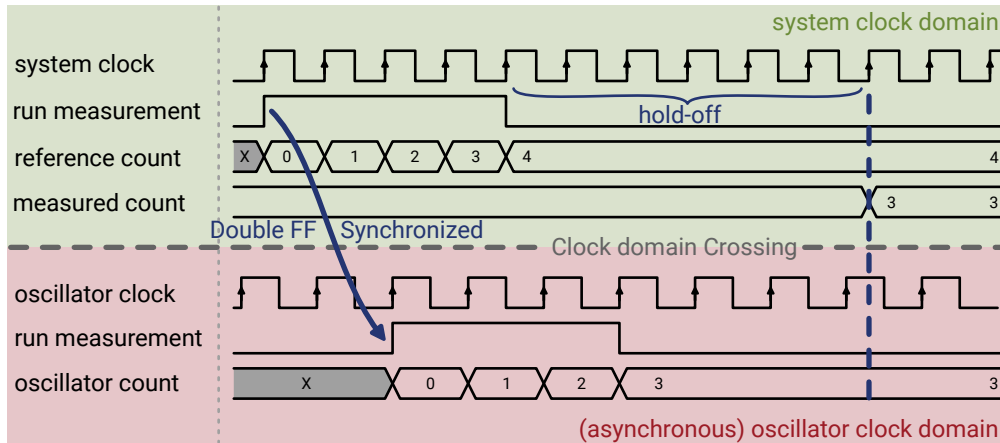


Figure A.3.: Principle of the clock domain crossing for the frequency measurement. The calibration routine operates in the system clock domain (green) while each oscillator generates its own asynchronous domain (red). The run measurement signal is transferred into the oscillator domain by a double flip-flop synchronizer circuit. The measured count of oscillation cycles is then safely read back after a hold period.

#### A.1.4. Calibration Results

The frequency calibration of the Glowworm chip for all 9 tested dies is provided here. Table A.1 lists the mean, standard deviation, minimum and maximum frequency before and after calibration. The corresponding frequency maps, showing the oscillator frequency at their positions graphically is available in Figure A.4.

Die	Before Calibration					After Calibration				
	Mean (MHz)	Std (MHz)	Min (MHz)	Max (MHz)	Spread (MHz)	Mean (MHz)	Std(MHz)	Min(MHz)	Max (MHz)	Spread (MHz)
#1	99.237	6.797	77.58	122.86	45.28	99.612	0.3554	96.69	101.33	4.64
#2	99.228	6.416	78.4	126.52	48.12	99.424	0.3362	97.99	101.24	3.25
#3	100.407	6.347	84.8	119.65	34.84	99.71	0.3533	98.68	101.16	2.47
#4	100.322	6.307	78.97	126.75	47.78	99.663	0.3367	98.55	101.83	3.28
#5	100.147	6.59	77.5	121.67	44.17	99.616	0.3787	97.92	104.1	6.18
#6	101.125	6.351	81.67	126.91	45.25	99.691	0.3516	98.6	100.78	2.18
#7	100.564	6.482	80.75	121.81	41.05	99.589	0.3533	98.57	102.15	3.58
#8	100.219	6.483	79.73	126.61	46.88	99.677	0.3336	98.64	101.88	3.25
#9										
#10	96.46	6.357	74.81	120.34	45.53	99.604	0.3885	94.36	100.61	6.26

Table A.1.: Frequency distribution and calibration of all tested dies. Unfortunately, no data of die #9 can be provided due to an intermittent contact issue with the carrier PCB. The mean frequency in the uncalibrated state is a good indication of the process variation between the dies. After on-chip calibration, the standard deviation of the frequency is between 0.3336 MHz and 0.3885 MHz.

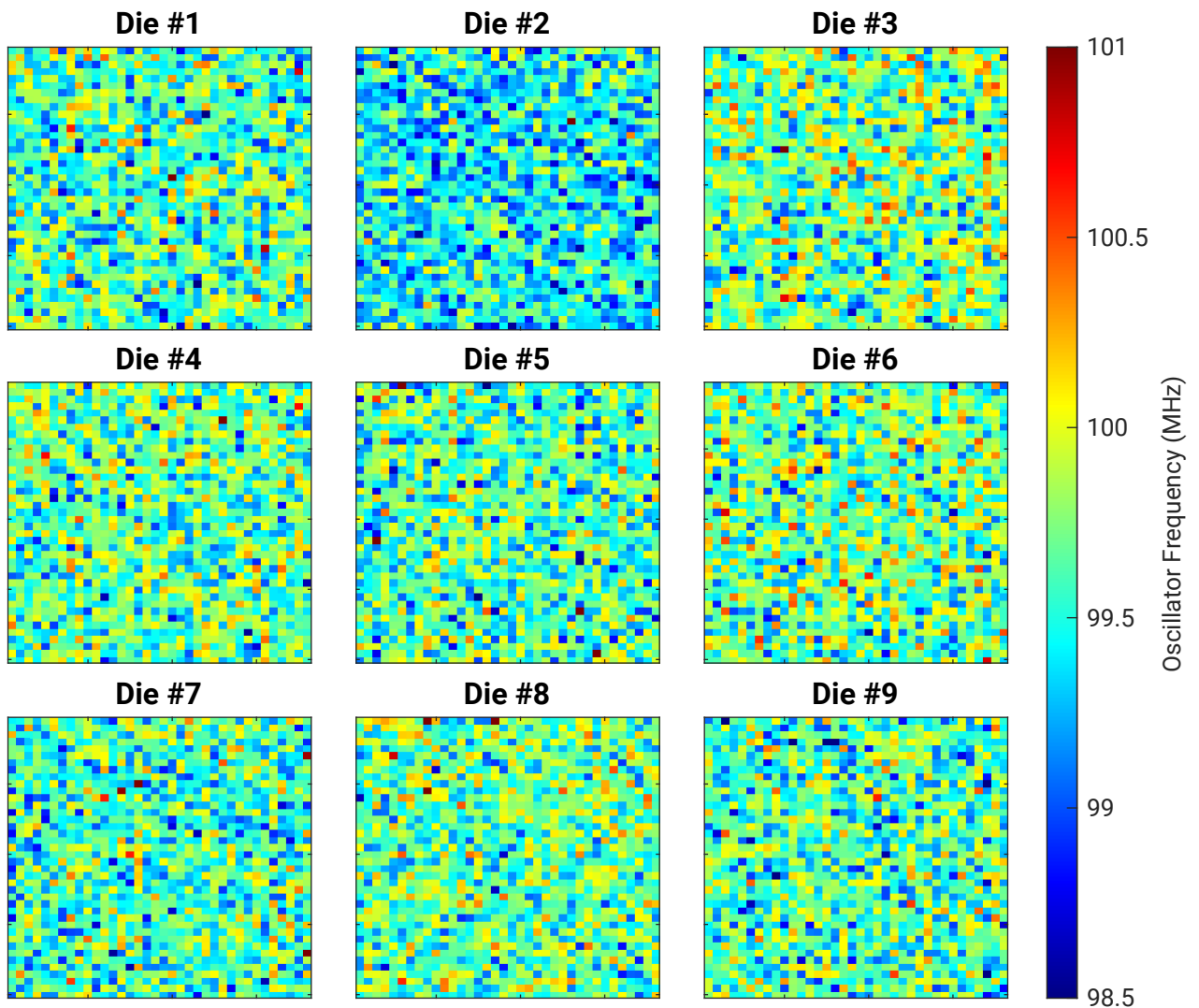


Figure A.4.: Color-coded oscillator frequency maps after calibration for all 9 tested dies.

# Chip Gallery

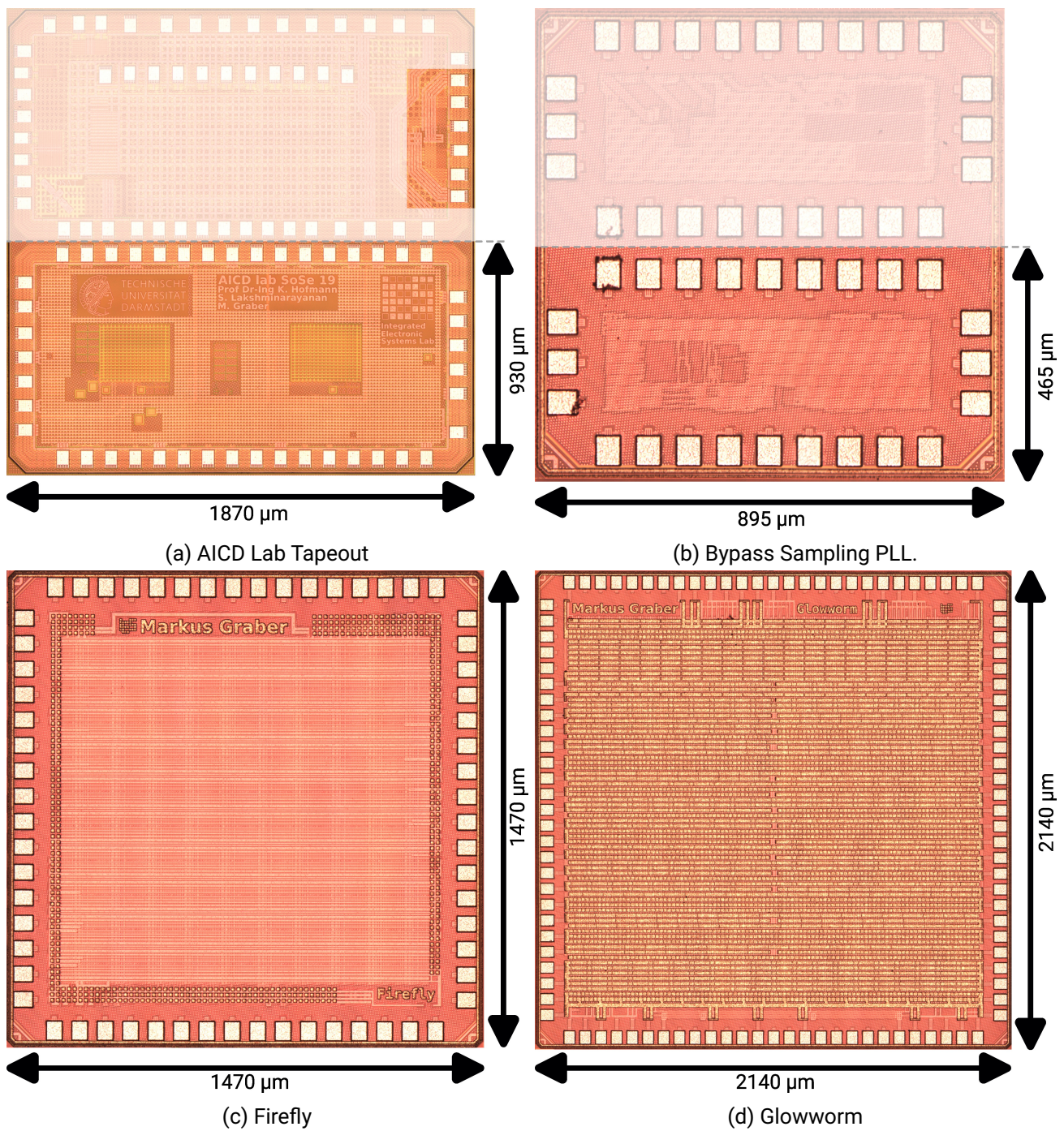


Figure A.5.: All 4 tapeouts during my time at the Integrated Electronic Systems Lab, Technical University of Darmstadt. The light-gray shaded areas are unrelated.