Efficient Technology Mapping Methods for Robust Genetic Logic Circuits

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Genehmigte Dissertation von M.Sc. Tobias Schwarz aus Bad Ems
Tag der Einreichung: 16. Januar 2024, Tag der Prüfung: 12. April 2024

1. Gutachten: Prof. Dr.-Ing. Christian Hochberger
2. Gutachten: Prof. Marc Riedel
Darmstadt, Technische Universität Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Electrical Engineering and Information Technology Department
Institut für Datentechnik
FG Rechnersysteme

Für Mina

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, 16. Januar 2024

T. Schwarz

Zusammenfassung

Ein Forschungsziel der Synthetischen Biologie ist die Konstruktion genetischer Logikschaltungen, die die Steuerung des Informationsflusses in biologischen Zellen ermöglichen. Analog zu elektronischen digitalen Schaltungen werden Schwellwerte auf Signale angewendet, um diese in boolesch 0 und 1 zu unterscheiden. Aus genetischen Bausteinen werden Logikgatter gebildet, die primitive boolesche Funktionen implementieren und in Gatterbibliotheken zusammengefasst werden. Diese Modularisierung ermöglicht die Anwendung von Methoden der Entwurfsautomatisierung, um genetische Logikschaltungen zu konstruieren und stellt die Basis für das Forschungsfeld der Genetic Design Automation dar. In Anlehnung an die Electronic Design Automation wird die Abbildung einer funktionalen Spezifikation auf die Zieltechnologie *Technology Mapping* genannt.

Im Vergleich zur Elektronik besitzt die genetische Technologie Eigenschaften, die neue Herausforderungen für Technology Mapping-Verfahren darstellen. So werden Stoffe als Signalträger verwendet, die nicht räumlich getrennt sind. Gatter in einer Schaltung müssen daher einzigartige Träger, und somit individuelle genetische Bausteine, verwenden. Daraus resultieren heterogene Transfer-Charakteristiken, die die Optimierung der Zuweisung von Gattern auf die Schaltungstopologie erforderlich machen. Weiterhin unterliegen Zellen einer starken Variabilität und unerwünschten Wechselwirkungen zwischen synthetischen und natürlichen Komponenten, die in der Konstruktion robuster Schaltungen berücksichtigt werden müssen. Oft übernehmen bestehende Ansätze in diesem dynamischen Forschungsfeld Methoden der Electronic Design Automation, die jedoch Potential für eine verbesserte Anpassung an die Problemstruktur genetischer Schaltungen bieten.

In dieser Arbeit werden Technology Mapping-Methoden für den Entwurf kombinatorischer genetischer Logikschaltungen präsentiert. Im Unterschied zu existierenden Methoden, die die Schaltungstopologie als Nutzereingabe erwarten oder diese mit Zielfunktionen aus dem Bereich der Elektronik synthetisieren, wird die Topologie als Freiheitsgrad im Technology Mapping aufgefasst. Die Methode wird Anhand eines Satzes boolescher Funktionen, für die bereits Schaltungen *in vivo* implementiert wurden, evaluiert. Durch die Enumeration struktureller Varianten kann die Entwurfsqualität gemessen in einer traditionellen Zielfunktion um bis zu 11.3-fach und im Durchschnitt um $38\,\%$ verbessert werden. Dabei bleibt die Schaltungsgröße minimal. Weiterhin werden Methoden für die

Optimierung der Gatterzuweisung vorgestellt, die durch ihre Effizienz die Erkundung breiter Entwurfsräume ermöglichen und den Aufwand für die Evaluation komplexer Modelle kompensieren. Basierend auf der funktionalen Hierarchie genetischer Logikschaltungen und fundamentalen Eigenschaften genetischer Gatter, wurden eine Simulated Annealing-Heuristik sowie eine Branch-and-Bound-Methode mit exaktem und heuristischem Modus entworfen. Letztere weist einen 20-fachen Speedup gegenüber erschöpfender Suche für das Finden der exakten Lösung auf. Die Heuristiken erreichen einen 722-fachen Speedup für das Erzeugen quasi-optimaler Lösungen mit einer maximalen Abweichung vom optimalen Zielfunktionswert von $-0.11\,\%$ und übertreffen damit eine existierende Heuristik.

Zusätzlich wurde Robustheit gegenüber der Variabilität von Zellen als Entwurfsziel in den Technology Mapping-Prozess integriert. Dafür wurde die Simulated Annealing-Heuristik an eine alternative Zielfunktion, den E-Score, angepasst und ein heuristischer Ansatz für Robustheit entworfen und in die Brach-and-Bound-Methode integriert. Beide Methoden weisen eine gute Mapping-Effizienz auf. Ihr Vergleich offenbart die Überlegenheit der globalen Optimierung für Robustheit durch den E-Score. Weiterhin wurde ein kontextsensitives Schaltungsmodell integriert, wodurch die Generierung optimaler Designs ermöglicht wird, selbst wenn die Schaltung unerwünschten Wechselwirkungen unterliegt. Die Evaluation unterschiedlicher Annahmen über das Übersprechen zwischen Gattern einer Bibliothek zeigt, dass gewisse Nichtorthogonalitäten in Gatterbibliotheken akzeptabel sind, wenn diese im Entwurfsprozess berücksichtigt werden.

Abstract

In the field of synthetic biology, one of the aims of researchers is to control the flow of information in cells using genetic logic circuits. Similarly to digital electronic circuits, signals are divided into Boolean 0 and 1 values through the application of thresholds. Genetic building blocks are combined together to form logic gates that implement Boolean functions, which are in turn combined into genetic gate libraries. This modularization allows the application of design automation methods to construct genetic circuits and represents the basis for the field of genetic design automation. In analogy to electronic design automation, the process of mapping a functional specification to the target technology is called *technology mapping*.

Compared to electronics, genetic technology possesses characteristics that pose new challenges to technology mapping methods. Due to the use of substances as signal transmitters and the shared medium of the cell, each gate in the circuit is required to use individual genetic parts. This leads to heterogeneous transfer characteristics among the gates and requires optimizing the gate assignment to the circuit topology. Further, cells exhibit strong variability and undesired interactions among synthetic and natural parts, which have to be considered for constructing robust genetic circuits. Existing approaches in this developing field often adopt methods from electronic design automation and have the potential for further adaptation to the genetic domain.

In this work, technology mapping methods for the design of combinational genetic logic circuits are presented. While existing methods rely on circuit topologies defined by the user or synthesized using objective functions known from electronics, in this work the circuit topology is incorporated as an additional degree of freedom. The method is evaluated using a set of Boolean functions, which have previously been implemented *in vivo*. By enumerating structural variants, a maximum $11.3\times$and a mean $38\,\%$ improvement in circuit performance can be achieved, as measured by a traditional circuit score, while keeping the number of used gates minimal. Efficient methods for optimizing the assignment of gates to the circuit are presented, which allow for the exploration of broad design spaces and compensate for the computational burden of elaborate models. Based on the functional hierarchy of genetic circuits and fundamental features of genetic gates, a Simulated Annealing heuristic and a Branch-and-Bound scheme featuring both a heuristic and an exact mode are devised. For finding exact solutions, the methods provide a $20\times$speedup compared to an exhaustive search. The heuristics achieve a $722\times$speedup while delivering near-optimal solutions with a worst-case deviation of $-0.11\,\%$, outperforming an existing heuristic.

Additionally, robustness to the variability of cells is integrated into the technology mapping process as a design objective. To this end, the Simulated Annealing method is adapted to use a robustness-centric objective function, the E-Score, and a heuristic approach to robustness is devised and integrated into the Branch-and-Bound scheme. Both approaches exhibit a good mapping efficiency, and their comparison highlights the increased robustness of designs optimized globally for the E-Score. Finally, the integration of a context-aware circuit model enables the finding of optimal designs, even if the circuit is subject to undesired interactions, i.e., crosstalk. By evaluating different crosstalk scenarios, it could be shown that specific non-orthogonalities can be tolerated in genetic gate libraries, provided that they are considered in the design process.

Contents

1. Introduction

In the field of *synthetic biology*, researchers aim to build biological systems from scratch, which implement a desired functionality [27]. It represents an interdisciplinary area of research in which knowledge about the mechanisms and building parts of organisms gained in biology is captured in mathematical models and libraries of modularized building blocks. This modularization allows for the usage of engineering approaches to construct systems based on a functional specification. In this context, *synthetic regulatory genetic circuits* represent an interesting subject and an active area of research, which focuses on controlling the flow of information and substances in biological systems. Parallel to electronic circuits, early instances of synthetic genetic circuits were hand-crafted and demonstrated simple mechanisms of positive or negative feedback [23, 26]. Meanwhile, the application of digital abstraction [66] and the construction of libraries of genetic logic gates [53] has enabled the implementation of *genetic logic circuits* and sparked the field of genetic design automation (GDA). Analog to electronic design automation (EDA), GDA refers to an automated workflow transforming a functional specification of a system into an implementable design [15]. By performing comprehensive evaluation and optimization of candidate designs *in silico*, GDA approaches aim to reduce the number of costly experiments *in vivo*, i.e., in the lab. Parallel to electronic circuits, *technology mapping* represents a main task of GDA, in which the function is mapped to the target technology. To find good implementations with respect to the given design objectives, technology mapping processes rely on a thorough understanding of the technology's specifics and consequent domain adaptation. This work presents technology mapping methods for the design of combinational genetic logic circuits.

1.1. Motivation

The motivation for this work is manifold and detailed in the following.

Improved Domain Adaptation Due to GDA being in an early stage of development, which proceeds with an improved understanding of mechanisms, refined models, and lab methods, the technology mapping processes of existing tools are very heterogeneous. While some leave important steps in the design process to the user, others rely on methods adopted from EDA that lack suitable adaptation to the genetic domain. For example, this holds for the step of structuralization, in which the functional specification is transformed into a graph representation that provides a basis for the circuit implementation. If the graph structure has to be given by the user [31, 55, 71], or it is synthesized using methods and design objectives from EDA [6, 43, 53], suboptimal circuit designs may be the result. Further, the usage of general-purpose optimization methods for subsequent design steps [53] without sufficient adaptation to the problem structure potentially leads to inefficiencies and reduced quality of solutions. From this observation, the question arises, *how can the domain adaptation of the technology mapping methods currently used in GDA be improved?*

Critical Knowledge Transfer from EDA While GDA is rooted in the knowledge gained in the design of electronic circuits, major differences exist between these technologies, which have to be taken into account when designing technology mapping methods. In EDA, tools and designers can access libraries of homogeneous building blocks that adhere to well-established standards, e.g., for the voltage levels representing the two states in a digital system. Libraries of genetic parts, however, are potentially heterogeneous and hence feature limited modularity [16, 66]. The same holds for multiple instances of the same system. While microchips, for example, underlie limited and well-understood process, voltage, temperature (PVT) variations [57], variability among biological cells of a population, i.e., cell-to-cell variability, can induce deviations of output signals spanning multiple orders of magnitude [63]. Therefore, design objectives established in EDA, e.g., signal delay and area usage, do not represent suitable goals of a technology mapping process in GDA. Instead, fundamental functionality of the synthetic circuits and resilience against variability are the current main objectives [53, 58]. This raises the question, *which characteristics of the genetic technology have to be taken into account during technology mapping to improve its domain adaptation?* But also, utilizing the knowledge gained in the design of electronic circuits, *which further viable approaches exist in EDA that offer itself for an adaptation to GDA?*

Support of Refined Models In contrast to electronics, the building parts of genetic systems are still not fully understood. It is unclear which parameters fully characterize genetic parts and the interactions of these parts in genetic devices [58]. This is pictured by libraries containing models of insufficient accuracy and scope. Prominent examples of this are the limited consideration of aforementioned cell-to-cell variability [63] and context effects, such as undesired interactions among components of the circuit or between the circuit and the host organism [16]. Thus, advanced models and design objectives are required that encompass these effects and thus provide an improved predictive power. In the context of this work, this raises the question of *how these models can be integrated into a technology mapping process to leverage their improved predictive accuracy.*

Efficient Design Space Exploration In the current state of synthetic biology, the technology mapping process is part of the "design, build, test" cycle [5] for genetic circuits. That is, the circuit is designed and optimized *in silico*, built and tested *in vivo* in the lab. Then, conclusions about the design process are drawn from the measurements, representing the basis for another potential iteration. Due to the high cost of lab experiments, a high accuracy of the *in silico* prediction is desired. Further, a technology mapping process is expected to find the (near) optimal circuit design with respect to that prediction. Thus, computation time should be invested where it benefits most: to enable the exploration of broad design spaces and to allow for the evaluation of candidate designs with complex models that feature good predictive power. Given the combinatorial nature of the considered technology mapping problem, these requirements necessitate methods that explore the design space efficiently, i.e., minimize the number of required evaluations while providing high-quality solutions. By this, the usage of these methods in the iterative design process becomes practical, and the desired growth in circuit and library sizes can be supported.

1.2. Contribution

The contribution of this work comprises multiple parts:

- The method of structural variants for the design of circuit topologies is devised, which broadens the search space of technology mapping under consideration of heterogeneous genetic gate libraries. It eliminates the structural bias introduced by structuralizing the functional specification in traditional mapping approaches.
- Search methods are presented for efficiently solving the combinatorial problem of assigning specific genetic gate realizations to the abstract logic network. The stochastic method based on the Simulated Annealing (SA) heuristic leverages the functional similarity of genetic gates to control the impact of perturbations performed during the optimization. The deterministic method is based on the Branch-and-Bound (B&B) paradigm and constructs solutions following the functional hierarchy

of genetic circuits. It leverages fundamental features of the transfer characteristics of genetic gates to obtain exact or heuristic solutions efficiently.

- Robustness with respect to the variability of biological systems as a design objective is integrated into the technology mapping flow. To this end, the SA search method is adapted to an alternative, robustness-centric objective function, and a heuristic approach to robustness is devised and integrated into the B&B scheme.
- A context-aware circuit model is integrated into the technology mapping flow. It allows the consideration of undesired interactions of components and thus finding optimal designs even if such crosstalk is present. The evaluation of the impact of crosstalk on circuit performance provides insights for the design of genetic gate libraries.

1.3. Thesis Outline

Chapters 2 to 6 provide relevant background information for understanding the remainder of the work. Chapter 2 briefly introduces the technology mapping for digital circuits. It focuses on structural mapping approaches and their problem of structural bias. Chapter 3 explains the fundamental mechanisms of genetic logic circuits. It covers the basic process of protein synthesis, the mechanism of transcriptional regulation, and its application in genetic logic gates and circuits. Chapter 4 introduces essential aspects of GDA. After reflecting on challenges in mapping genetic circuits, approaches to quantifying circuit functionality and robustness are presented, which are used throughout this work. Further, a thermodynamic circuit model is introduced that considers context effects when evaluating genetic circuits. In Chapter 5, the technology mapping problem is contextualized in the GDA design flow and defined formally. Finally, Chapter 6 surveys related work.

Chapters 7 to 11 detail the contributions of this work. Chapter 7 presents the concept of structural variants to broaden the design space of genetic circuits. Efficient methods for optimizing the assignment of specific genetic gates to the circuit structure are detailed in Chapter 8. In Chapter 9, two ways to integrate robustness as a design objective in the technology mapping process are presented. Chapter 10 reports on integrating context-awareness into the process, allowing for the consideration of crosstalk. Lastly, the methods are contextualized in meaningful technology mapping scenarios and compared in a realistic work environment in Chapter 11.

Chapter 12 contains the conclusion of the work and the outlook.

Background

2. Technology Mapping

Technology mapping is a crucial step in the hardware synthesis of digital systems. During the process, a functional specification of a system is mapped to elements of the target technology. The specification is typically given as Boolean formulae describing the combinational components of the system. Generally, these formulae are derived from a more elaborate description, for example, a hardware description language. This elaboration step can be carried out by open-source tools, such as YOSYS [69]. Digital systems also contain sequential elements such as registers and circuitry providing an input/output interface. However, these elements can be mapped to a technology by simple substitutions and are thus not considered in elaborate technology mapping methods [21]. Figure 2.1 depicts the mapping process in its abstract form. Target technologies such as standard-cell libraries and gate arrays are defined by the library of available elements. Generally, this also holds for the genetic gate technology considered in this work, while some important differences apply (Section 4.1). In the following, the library of technology elements is called gate library. Besides describing the function of each gate, the library contains parameters required for the technology mapping process to optimize an objective function [18]. In the synthesis of electronic digital systems, these objectives classically are the delay and area of the resulting design. Recently, also the energy consumption of designs is considered by a growing number of tools [74]. Thus, the gates in the library are annotated with their individual propagation delay, area usage and, possibly, energy consumption. During

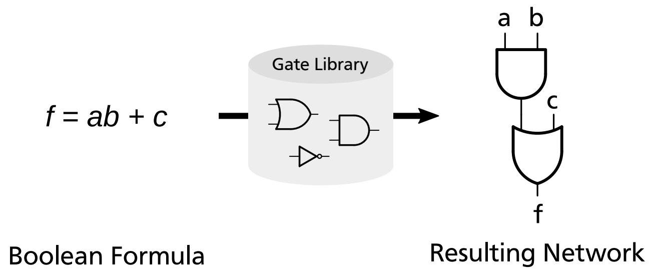


Figure 2.1.: Abstract technology mapping process

mapping, these objectives are often combined into a priority relation, e.g., minimizing the delay first, then minimizing the area while preserving the achieved delay, or a constraint relation, e.g., minimizing the delay given a maximum energy consumption. Technology mapping results in a network consisting only of gates available in the target technology. The specific choice of gates and the network structure are degrees of freedom that can be leveraged to optimize the design with respect to the objective.

2.1. Structural Mapping

The technology mapping problem in its abstract form is generally intractable [18]. To make it accessible to algorithmic solution, the Boolean formulae are usually first translated into a directed acyclic graph (DAG) representing the combinational Boolean network, the *subject graph* [35]. A DAG is a graph with directed edges that do not form a cycle. At least one *topological order* exists for a DAG, in which every node appears before its successors as defined by its outgoing edges. The subject graph is a technology-independent DAG-representation of the function with nodes representing simple logic gates, typically from a limited set of gates. Technology mapping approaches using a subject graph are called *structural mapping* approaches. They are the most common class of mapping algorithms used for standard-cell, gate array, and field programmable gate array (FPGA) designs. The structural mapping process consists of the three phases *subject graph generation*, *cell matching*, and *network covering*.

For example, the academic logic synthesis tool ABC uses subject graphs consisting solely of two input AND gates and inverters [39, 47]. Figure 2.2 depicts the mapping process based on such an AND-Inverter Graph (AIG). First, the Boolean formula $f = \bar{b}a + bc$ representing a two-input multiplexer is translated into the subject graph. Different graph structures can represent the same function as AIGs are non-canonical. In ABC, this effect is reduced by performing heuristic minimization of the AIG. Then, single nodes or larger sub-graphs of the AIG are matched with elements given in the target technology's gate library. Two approaches for cell matching exist:

Structural matching Graph representations of the library gates exist that use the same elements as the subject graph. This allows a graph-based structural comparison of library gates and sub-graphs. It requires either a canonical subject graph structure or the presence of multiple graph variants per library element to enable matching with a non-canonical subject graph. Generally, the comparison is carried out by checking the isomorphism of the two involved DAGs. However, the check is typically simplified by transforming the DAGs into trees and performing tree matching.

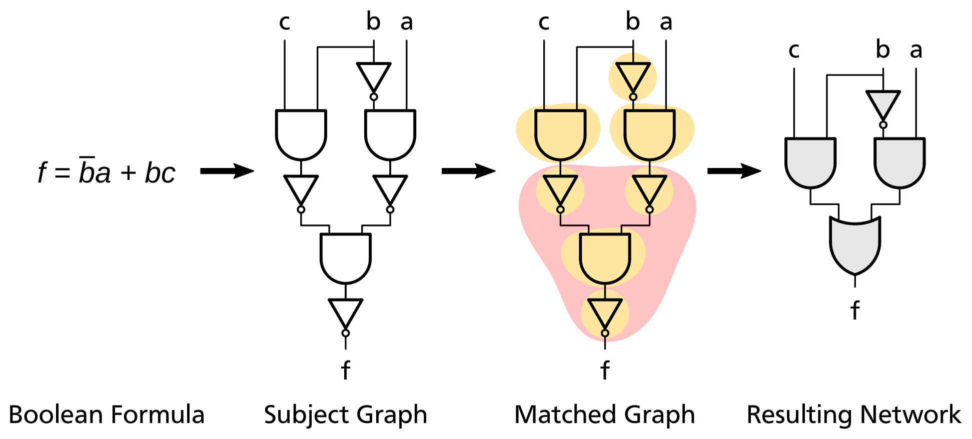


Figure 2.2.: Structural mapping using an AND-Inverter Graph (AIG) as subject graph. Blank gates represent AIG nodes, gray gates specific library gates.

Boolean matching In Boolean matching, the structural characteristics of the sub-graphs are neglected. Nodes and sub-graphs of the subject graph and the library elements are transferred into a Boolean representation, e.g., ordered binary descision diagrams (OBDDs) or truth tables. These data structures can then be compared during matching while considering possible input variable permutations. Boolean matching generally can find more matches than structural matching since it is independent of structural characteristics, but it also tends to be computationally more expensive.

The result of the matching phase is the set of all possible matches of the library elements with the subject graph. If at least one node in the graph is not part of a match, the subject graph is not implementable with the given library. In the depicted example (Figure 2.2), the subject graph is matched with the simple library of two input AND and OR gates and inverters shown in Figure 2.1. Besides the trivial matches shown in orange, a complex match of the OR gate with a sub-graph is shown in red.

The final phase of the structural mapping is the network covering. In this phase, a subset of matches is selected that covers every node in the subject graph and optimizes the objective of the technology mapping. Based on the selected matches, the mapped network is constructed that consists solely of gates present in the library. In the example, the complex match with the OR gate is chosen over the trivial matches. If equal area and delay values for all library elements are assumed, this cover leads to an area- and delay-optimal network.

2.2. Structural Bias

Irrespective of the used matching method, structural mapping approaches exhibit the phenomenon of *structural bias* [18]. It expresses the correlation of the mapped network topology with the subject graph's structure. Due to the locality of the substitutions carried out in the network covering phase, the structural characteristics of the subject graph are transferred to the final network structure. This is depicted exemplary in Figure 2.3 for the function $f = ab + c$. Figures 2.3a and 2.3b both show AIGs generated for f with ABC and the respective mapped networks. While the AIG in Figure 2.3a results directly from ABC's AIG generation routine, the graph in Figure 2.3b has been further processed with ABC's heuristic minimization. Both graphs have then been matched and covered with the objective of minimizing the delay. Yet, the locality of the matching and covering process prevented the mapper from generating the better network structure of Figure 2.3b from the subject graph of Figure 2.3a. The resulting networks are delay-optimal with respect to their particular subject graph but not concerning the functional specification that served as an input to the technology mapping process.

The implications of the structural bias depend strongly on the target technology and objective function. When the objective is a minimal network delay, minimizing the depth of the subject graph can be a good heuristic to get a subject graph that is biased towards

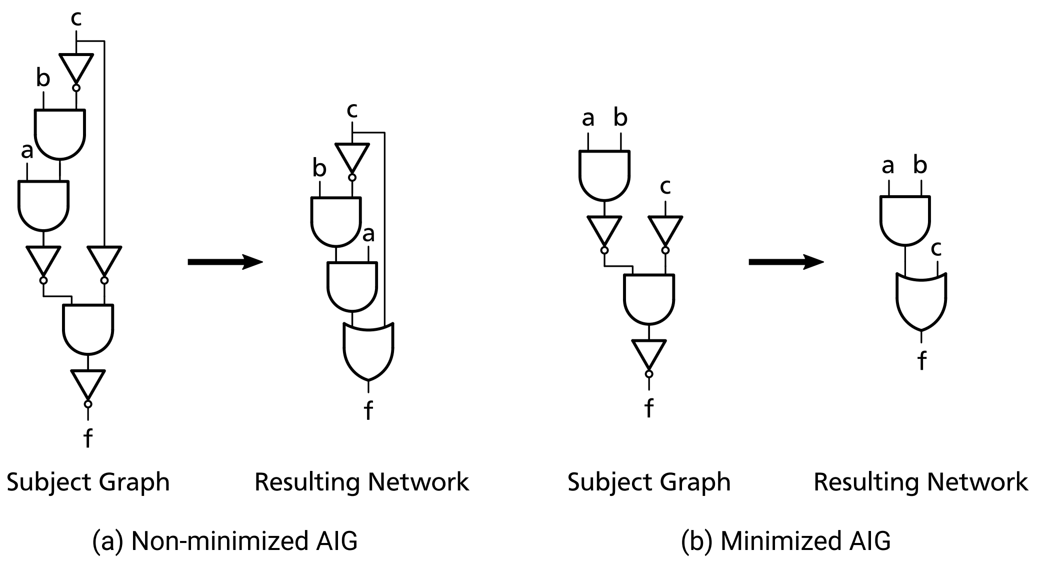


Figure 2.3.: Networks resulting from different subject graphs for function $f = ab + c$

a minimal resulting network delay. The same holds for minimizing the area usage of a design, where the number of elements of the subject graph can serve as an estimation for the resulting network size. However, if the objective function is not based on the network's inherent structural characteristics, it is impossible to correlate it with features of the subject graph. For example, this is the case for the genetic circuits considered within this work, as the corresponding objective functions are based on the simulation of the network as a whole (Sections 4.2 and 4.3).

3. Genetic Logic

The field of synthetic biology "[...] seeks to design and build new biology that does useful things" [27]. It is based on the knowledge about organisms gained from classical biology and applies it to methods from engineering disciplines. Thus, synthetic biology represents an interdisciplinary field driven by biology, engineering, computer science, and other disciplines. The applied methods span understanding the building parts of biological systems, abstracting their behavior in mathematical models, combining parts in libraries, and applying design automation methods to build new systems from these libraries. Further, standardization of data formats for parts and systems is pursued to enable simple exchange, but is still hampered by the complexity of the systems to be described.

From an abstract point of view, a biological cell resembles a data processing system [27]. It gathers information via sensors sensitive to environmental conditions like light, temperature, and substances from its surroundings or other cells. It processes this information with signaling pathways that are regulated by negative or positive feedback. The resulting signals control actuators, for example, for movement, growth, or the synthesis of substances. Regulatory genetic circuits play a vital role in this scheme by providing the cell with means to control the flow of information and substances. A main focus of synthetic biology is controlling this flow by constructing *synthetic* regulatory genetic circuits. The aim is to mimic the behavior of electronic circuits. First demonstrations of this approach are the genetic toggle switch [26] that exhibits a bi-stable behavior through positive feedback and the "repressilator" [23], the output of which oscillates, driven by negative feedback. Both circuits have been published in the year 2000 and demonstrate that synthetic biology is a relatively new field of research.

Parts used in synthetic regulatory genetic circuits are defined by the deoxyribonucleic acid (DNA) sequence that is used to build them. They are combined into devices, and devices are combined into synthetic biological systems [27]. The remainder of this chapter follows this structure and focuses on the biological systems of synthetic combinational genetic logic circuits considered in this work. First, protein synthesis and its regulation are introduced (Section 3.1). It is not only central to life but also an essential mechanism utilized in synthetic circuits. Quantitative aspects of transcriptional regulation are explained in Section 3.2. Based on this, Sections 3.3 and 3.4 describe the mechanics and building principles of genetic logic gate devices and combinational circuits.

3.1. Protein Synthesis

DNA is the storage of information of all living organisms. It is a nucleic acid molecule in the form of a double helix of nucleotides, which are based on the four nucleobases: adenine, cytosine, guanine, and thymine [38]. The information contained in the DNA is coded in the sequence of these bases. Two essential functions of the DNA are its replication which allows for the replication of the cell itself, and the expression of genes as proteins. These functions are so crucial for living organisms that they are called molecular biology's *Central Dogma*.

Central Dogma Figure 3.1 depicts the processes associated with the Central Dogma. Before the cell can divide in a replication process, the DNA has to be duplicated. In this process, a replica of the DNA is produced by multiple components summarized as the replisome [54]. The expression of proteins based on the information stored in the DNA is highly relevant for the mechanisms of genetic circuits. A section of the DNA that codes one protein is called a *gene*. In the process of *transcription*, the gene is read by a molecular machine named RNA polymerase (RNAP). While processing the DNA, it produces a messenger-RNA (mRNA) that is a complementary copy of the DNA. The mRNA serves as the template for the final step of *translation*. The ribosome, another molecular machine, clamps the mRNA and translates triplets of nucleotides to amino acids using transfer-RNA (tRNA). The amino acids are then appended to the growing polypeptide chain constituting the protein.

Transcriptional Regulation Cells need to regulate which proteins are expressed at which point in time. The bacterium *Escherichia coli* (*E. coli*), for example, which is commonly used as a host organism in synthetic biology, features a genome of 4500 genes [3]. Expressing all genes at once would quickly drain the cell's energy budget as each protein synthesis burdens the cell's metabolism. Thus, cells control the rate at which individual proteins are synthesized by controlling the rate of transcription of the corresponding genes. This mechanism called *transcriptional regulation* is organized in gene regulatory networks (GRNs) and also represents the basis of synthetic regulatory genetic circuits. Each gene contains a *coding sequence* that describes the protein, preceded by a *promoter* sequence. Here, the RNAP binds to initiate the transcription. Special proteins called *transcription factors* (TFs) can bind to the promoter, affect the binding of the RNAP, and

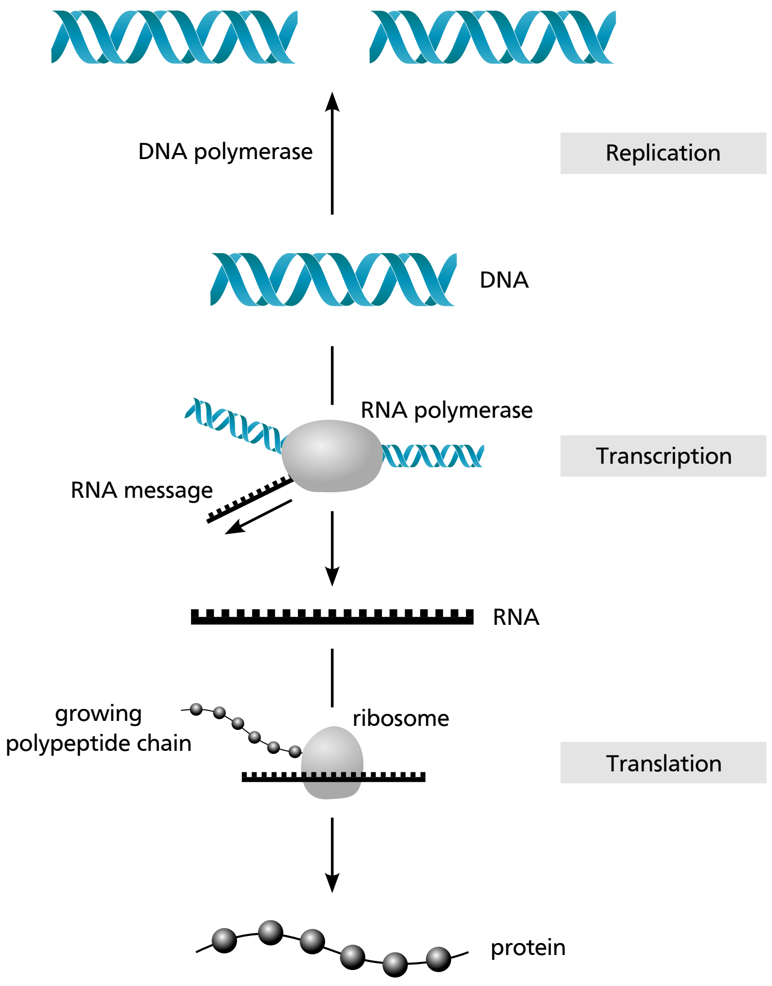


Figure 3.1.: The Central Dogma of molecular biology: DNA replication, transcription and translation (based on [54])

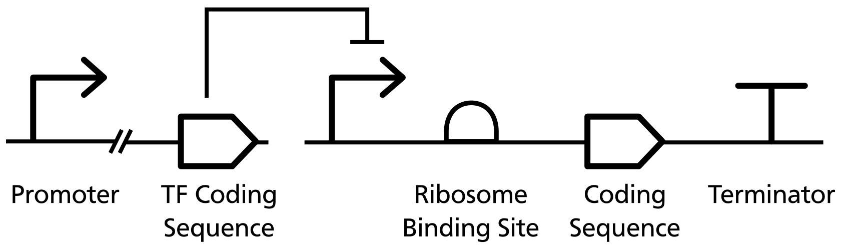


Figure 3.2.: Inhibitory relation on a DNA sequence in SBOL Visual

thus control the transcription of the succeeding coding sequence. TFs can either promote the binding of the RNAP to the promoter and thus have an activating effect on transcription, or they can block the binding of the RNAP and thus have an inhibiting effect. In this context, activating TFs are called *activators*, and inhibiting TFs are called *repressors*.

Figure 3.2 exemplifies inhibitory transcriptional regulation based on the synthetic biology open language (SBOL) Visual [7] notation. The horizontal base line depicts the DNA strand, and each symbol represents the sequence of an element of the regulatory network. Additionally to the promoters and coding sequences, a ribosome binding site (RBS) and a terminator sequence are shown. The RBS guides the ribosome for translation, and the terminator marks the end of the gene for transcription. The connection ending with the reversed "T" at the right promoter implies an inhibiting effect of the coded TF on the second promoter. An activating effect would be depicted with an arrow-headed connection. The promoter that a TF binds to with a high affinity is called its *cognate* promoter.

3.2. Transfer Functions of Regulatory Systems

The mechanism of transcriptional regulation enables natural biological cells to adapt their production of proteins to the conditions of the environment and their life cycle. In the context of synthetic biology, it is the basis for constructing synthetic regulatory genetic circuits. Understanding the quantitative aspects of regulatory relations is thus important to model building parts of synthetic circuits and design circuits based on these models.

From the perspective of Boolean logic, the inhibitory relation of the repressing TF to its cognate promoter in Figure 3.2 resembles a negation. The more TF is present, the higher the probability that instances of it bind to the promoter and thus inhibit it. The graphical representation of this relation is called the genetic system's *dose-response curve*. Quantitatively, the activity of the cognate promoter can be expressed as RNAP flux, the

"[...] number of polymerases per second that start transcription at that promoter" [66]. In the following, the RNAP flux will be called *promoter activity*. It is a function $\mathrm{g} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ of the concentration of repressor $X \in \mathbb{R}_{\geq 0}$. A function that realistically describes these relations of inhibiting or activating transcriptional regulation is the *Hill function*. It "[...] can be derived from considering the equilibrium binding of the transcription factor to its site on the promoter" [3] and arises from the concept of mass-action kinetics. The inhibitory Hill function

$$\mathrm{g}(X) = \beta \frac{K^n}{K^n + X^n} \tag{3.1}$$

relates the output promoter activity to the input repressor concentration. Its parameter β is the maximal output promoter activity. The repression coefficient K equals the repressor concentration that produces an output of $\beta/2$. The Hill coefficient n describes the number of available binding sites at the promoter and affects the steepness of the resulting function. Analog,

$$\mathrm{h}(X) = \beta \frac{X^n}{K^n + X^n} \tag{3.2}$$

represents the Hill function for an activating regulatory relation. Equations (3.1) and (3.2) describe regulatory relations without a *basal activity*, i.e., a minimal output promoter activity of 0. To account for higher basal activities, β is replaced with an interval of minimum and maximum output promoter activity $[y_{\mathrm{min}}, y_{\mathrm{max}}]$ [53], resulting in

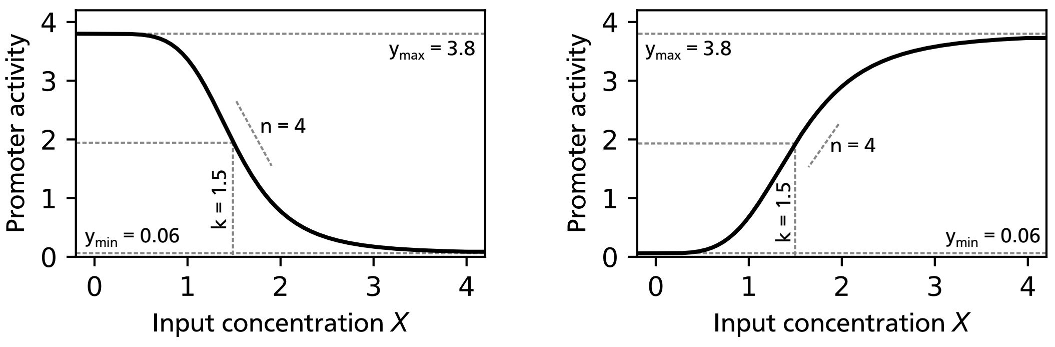
$$\mathrm{g}(X) = y_{\mathrm{min}} + \frac{(y_{\mathrm{max}} - y_{\mathrm{min}})K^n}{K^n + X^n}, \tag{3.3}$$

$$\mathrm{h}(X) = y_{\mathrm{min}} + \frac{(y_{\mathrm{max}} - y_{\mathrm{min}})X^n}{K^n + X^n}. \tag{3.4}$$

Independent of the specific parameterization, Hill functions have characteristics relevant to the mapping methods presented in this work. First, they are monotonic, as a higher concentration of TF leads to an amplified effect at the cognate promoter. This holds for inhibiting as well as activating effects. Additionally, their codomain is limited by the interval $[y_{\mathrm{min}}, y_{\mathrm{max}}]$, i.e.,

$$\lim_{X \to 0} \mathrm{g}(X) = y_{\mathrm{max}}, \quad \lim_{X \to \infty} \mathrm{g}(X) = y_{\mathrm{min}}, \quad \lim_{X \to 0} \mathrm{h}(X) = y_{\mathrm{min}}, \quad \lim_{X \to \infty} \mathrm{h}(X) = y_{\mathrm{max}}.$$

The "[...] saturation of the Hill function at high $[X]$ concentration is fundamentally due to the fact that the probability that the activator binds the promoter cannot exceed 1, no matter how high the concentration of $[X]$" [3]. The same holds for repressors binding to their cognate promoter. Figure 3.3 visualizes these characteristics using exemplary parameterizations of $\mathrm{g}(X)$ and $\mathrm{h}(X)$.

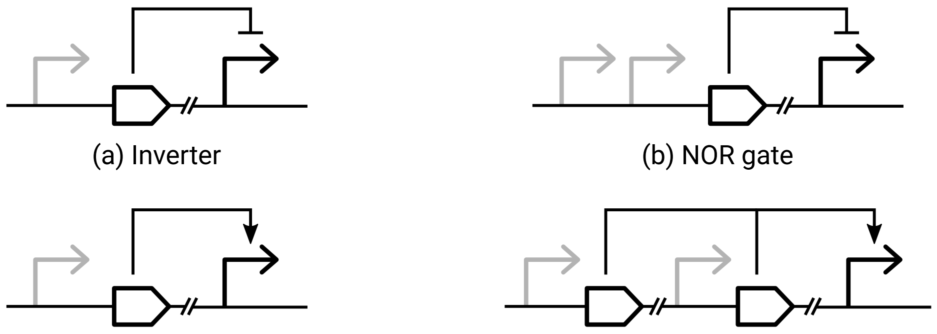


(a) Repression, $\mathrm{g}(X)$ (b) Activation, $\mathrm{h}(X)$

Figure 3.3.: Exemplary parameterizations of Hill functions

3.3. Genetic Logic Gates

Based on the fundamental transfer characteristic of transcriptional regulation, synthetic regulatory devices can be built. By applying the paradigm of digital logic abstraction known from electronic logic circuits, these devices can be interpreted as digital logic gates. To this end, a threshold is applied to interpret the continuous range of the transcriptional signals in the Boolean space $\mathbb{B} = \{0, 1\}$ [66]. Figure 3.4 shows schematics of different genetic logic gates using the presented building parts of regulatory systems. A combination of a gene coding a repressor and the repressor's cognate promoter is the core of the genetic inverter (Figure 3.4a). Its input promoter controlling the coding sequence's expression is shown in gray, as it is not a part of the gates TF-promoter system, but the cognate promoter of another TF. By adding more input promoters in front of the coding sequence, a NOR gate can be built (Figure 3.4b). Only if all input promoters are inhibited (i.e., Boolean 0), the repressor is not expressed, and its cognate output promoter is thus active (Boolean 1). Quantitatively, the input concentration of the gate results from the superimposed RNAP flux of all input promoters. While this type of NOR gate uses tandem promoters [72], i.e., a chain of promoters controlling a subsequent gene, it is also possible to construct a NOR gate using separate genes, each controlled by one promoter [19]. A simple buffer gate can be built based on activation (Figure 3.4c). The gate transfers the input promoter's activity to the output promoter's activity. Further, an activation-based AND gate can be built (Figure 3.4d) using, for example, the principle of coregulation [48], where only the presence of two related TFs leads to an activation of the promoter.



(a) Inverter (b) NOR gate

(c) Buffer (d) AND gate

Figure 3.4.: Genetic logic gates in SBOL Visual

The signaling pathway of a genetic gate is defined by the binding affinity of the TF to its cognate promoter. All gates in a genetic regulatory circuit share the same medium of the host cell, and the signal carriers, i.e., TFs, are not separated physically. This represents a contrast to electronic circuits, where signals are separated in conducting physical channels, and has substantial implications for libraries of genetic gates. First, they ideally contain multiple TF-promoter implementations of gates with the same logic function as each TF can only be used once in a synthetic regulatory circuit. These individual gate instances should ideally be *orthogonal*, i.e., do not interact with each other. In other words, the TF of each individual gate should not bind to promoters other than its designated cognate promoter. Furthermore, the interaction of the gates with components of the host cell should be minimal, e.g., the binding of synthetic TFs to promoters of the host genome. These *context effects* are detailed in Section 4.4. The required uniqueness of TFs further leads to heterogeneous transfer characteristics among gates of the same function, posing a challenge to the construction of circuits (Section 4.1).

3.4. Combinational Genetic Circuits

Given a library of modular and orthogonal genetic logic gates, genetic logic circuits can be built by cascading gates. They represent a special form of synthetic GRNs in which the paradigm of digital abstraction is applied to all network components. Genetic logic circuits that do not contain means to store information are called combinational genetic logic circuits. Analog to electronic combinational circuits, their topology is that of a DAG.

The underlying basic design principles are explained using the small exemplary circuit of two gates shown in Figure 3.5. The circuit diagram is depicted on the left, and on the

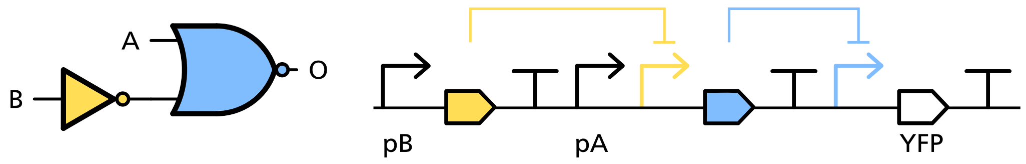


Figure 3.5.: Exemplary circuit topology and its SBOL representation

right, a DNA implementation of the circuit is shown in SBOL Visual notation. Ribosome binding sites in front of the coding sequences are omitted to simplify the visualization. The gates are color-coded to imply that individual pairs of TFs and promoters are used to implement each gate. Synthetic promoters serve as input to the circuit. Here, they are called pA and pB, referring to the inputs A and B. In implemented circuits, promoters such as the Tac promoter (pTac), the Arabinose-induced promoter (pBAD) and the Tetracycline-controlled promoter (pTet), which can be induced by TFs introduced from outside of the system, can serve as inputs [53]. Promoter pB serves as input to the inverter, which consists of the yellow coding sequence for a TF and its cognate output promoter, also colored in yellow. The output promoter is placed in front of the NOR gate's coding sequence (shown in blue) to implement the wiring of the inverter's output to the NOR gate's input.

Important characteristics of digital circuits are given by the *fan-in* and *fan-out* of gates. The fan-in defines the number of inputs of a multi-input gate and the fan-out defines the number of subsequent gates driven by the gate. In the example, tandem promoters are used, and thus, fan-in is implemented by placing multiple promoters in a row in front of the coding sequence. Fan-out, which does not occur in the example circuit, can be implemented by placing instances of the same output promoter at the input of multiple other gates.

The circuit output is implemented by placing the last gate's output promoter in front of a coding sequence for a reporter protein. Here, yellow fluorescent protein (YFP) is chosen, which allows for determining the circuit's output by measuring the cell's fluorescence. In pharmaceutical applications, for example, the output coding sequence could be that of an active agent.

4. Genetic Design Automation

Analog to electronic design automation (EDA), genetic design automation (GDA) refers to an automated workflow that transforms a functional specification of a system into a finished design implemented in the respective target technology. It represents an alternative approach to the method of rational design that centers on the knowledge of designers when constructing new biochemical systems. In the context of genetic logic circuits, GDA aims to drastically reduce the number of lab experiments *in vivo* needed to design new circuits. To this end, elaborate simulation and optimization are carried out *in silico*, synthesizing the most promising designs to build and test in the lab. GDA can be subdivided into three fields that all spark scientific interest and feature recent advances [15]. First, libraries of genetic parts play an important role in the abstraction and modularization of the design process, both crucial concepts known from EDA. Open repositories like the iGEM library [14] and well-specified orthogonal gate libraries like those used in the Cello GDA tool [53] allow reusing parts that have been specified in extensive lab work. Second, standardization of data exchange formats can enable the interoperability between research groups and tools for different abstraction levels. Prominent examples are the synthetic biology open language (SBOL) data exchange standard [44] and the related SBOL Visual standard [7]. They feature a unified representation for genetic systems from the DNA sequence level to functional interactions between parts and offer a framework for consistent visualization. Lastly, software tools for GDA provide user-friendly means to transform a system specification into an implementable design (Figure 4.1). They access libraries of genetic parts and perform technology mapping, ideally by optimizing

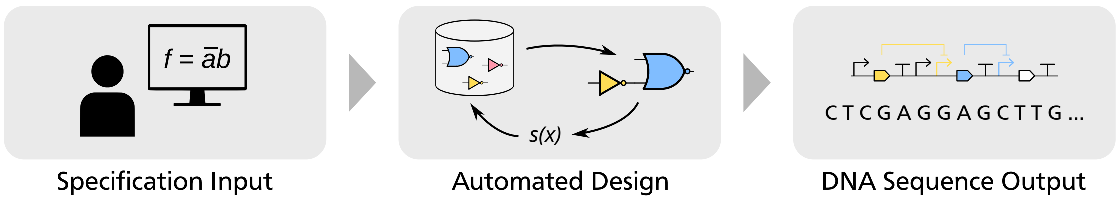


Figure 4.1.: Overview of a GDA software tool workflow

the circuit design with respect to a cost function or performance metric. Some tools also perform the physical design, transforming the mapped network into a DNA strand. Existing tools span different abstraction levels and different parts of the synthesis process. The prominent GDA tools iBioSim [51] and Cello [53], for example, cover the complete synthesis process from a high-level specification input to a DNA sequence design output.

The remainder of this chapter focuses on important aspects of the technology mapping step in the GDA workflow for combinational genetic circuits. First, the challenging properties of genetic circuits in contrast to electronic circuits and their implications for technology mapping are presented in Section 4.1. Then, a circuit model is introduced, together with a fundamental metric for determining the functionality of a circuit design (Section 4.2). The metric can be used as an objective function for technology mapping. In Section 4.3, an alternative objective function is presented that allows to make robustness to variability among cells of a population an inherent goal of the mapping process. Finally, an alternative circuit model based on thermodynamic interactions is presented that can be used to model context effects of genetic circuits (Section 4.4).

4.1. Challenges in Mapping Genetic Circuits

Genetic circuits pose additional challenges to the technology mapping process compared to electronic circuits. This is caused by the underlying properties of gene regulatory systems and the resulting building principles of genetic circuits. As shown in Chapter 3, all gates in a circuit, even those implementing the same function, must be implemented with a unique pair of TF and cognate promoter. This constraint is due to the fact that the reuse of parts would result in undesired signal paths, as the connections of the circuit are defined by the binding affinity of the TFs to the promoters. It results directly in a restriction of the maximal circuit size by the number of unique genetic gates available in the library and represents a contrast to electronic circuits. Standard-cell designs, for example, can comprise an arbitrary number of copies of each cell.

Heterogeneous Gate Characteristics Each pair of TF and promoter has a unique regulatory behavior and transfer characteristic. In the Hill model, this results in a different set of parameters $\theta = (y_{\min}, y_{\max}, K, n)$ with K and n defining the shape of the curve and $[y_{\min}, y_{\max}]$ defining the output interval spanned by basal and maximum promoter activity. This represents a difference to electronic digital circuits, where all gates operate within equal signal ranges, e.g., V_{GND} and V_{DD}, and global noise margins can be defined. These non-uniform transfer characteristics and signal levels have two consequences for the performance of genetic circuits and the technology mapping process.

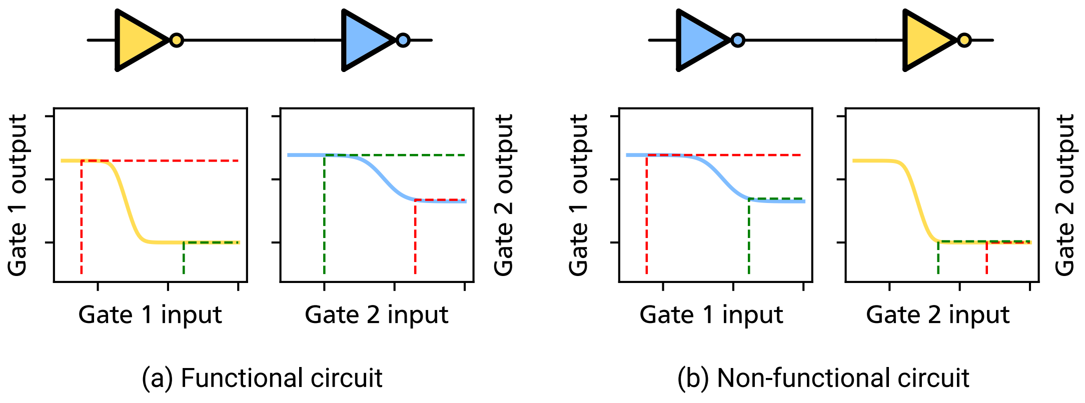


Figure 4.2.: Signal mismatch in genetic circuits (adapted from [53])

First, the output signal range of a gate may be incompatible with the input range of a subsequent gate [66]. This is demonstrated in Figure 4.2, which shows two simple circuits comprised of two inverters based on different TFs. In Figure 4.2a, the output of gate 1 representing a Boolean 1 (red dotted line) falls into the saturated low-output region of gate 2. Analog, the output of gate 1 representing a Boolean 0 (green dotted line) falls into the saturated high-output region of gate 2. As expected, the input signal of the circuit is inverted twice, and the complementary Boolean states are differentiable at the circuit's output. In Figure 4.2b, the order of the two gates is swapped. Now, the output signal of gate 1 falls into the low-output region of gate 2, irrespective of the Boolean value it represents. Thus, the Boolean output states are not differentiable at the circuit output as the information is lost due to the signal mismatch between the two gates. This full loss of functionality is caused by an extreme signal mismatch, while in more nuanced cases, the usable output interval of the succeeding gate would be reduced. Therefore, in addition to the classical process of technology mapping that transforms the functional specification into a subject graph and the subject graph into a network of technology elements, genetic circuits require the step of *gate assignment* [53]. In this step, specific TF-promoter implementations of gates are assigned to abstract logic gates in the network. This problem is non-trivial since the gate assignment affects the degree of functionality of a circuit, which has to be determined by simulating the circuit as a whole (Section 4.2).

Second, due to the non-uniform transfer characteristics of genetic gates, the circuit *topology* plays an important role in the circuit's functionality. Among multiple network structures implementing the specified Boolean function, specific ones might allow combinations of gates that work well together, while others provoke non-functional combinations. Furthermore, genetic gate libraries typically contain a limited set of gates per logic function, making exploring variants of implementations made of different types of gates worthwhile. This is examined in more detail in Chapter 7. In contrast, electronic circuits generally function irrespective of the topology, but it is used to optimize the delay and area of the circuit.

Variability and Context Effects Furthermore, bio-molecular effects influence the performance of genetic circuits. Stochasticity of processes in the cell and variability among cells of a population can introduce variance in the output characteristic of a circuit [63]. Further, undesired interactions called *context effects*, e.g., off-target binding of TFs at other promoters of the circuit or binding sites of the host cell, affect the predictability of circuit performance [16]. Thus, a technology mapping process that is aware of variability (Section 4.3 and Chapter 9) and context effects (Section 4.4 and Chapter 10) improves the meaningfulness of the mapped designs.

Metabolic Burden The expression of synthetic components has been shown to burden the cell's metabolism. This "[...] typically reduces growth, triggers stresses and leads to decrease in performance or failure of engineered cells" [10]. Generally, a positive relation of the amount of expressed synthetic components to the burden can be assumed.

In summary, genetic circuits pose the following challenges for a technology mapping method:

1. The design size is limited by the number of gates in the library and the burden imposed on the cell.
2. Genetic gates do not feature uniform transfer characteristics and signal levels, resulting in
    a) the additional, non-trivial step of gate assignment.
    b) the importance of the network topology for the functionality of a design.
3. Stochasticity and variability among cells introduce variance in the circuit output.
4. Context effects affect the circuit performance.

4.2. Quantifying Circuit Functionality

Quantifying the degree of functionality of genetic circuits is essential for technology mapping to optimize designs for adherence to the specification. The metric applied by the GDA tool CELLO [53] represents the state-of-the art and is detailed in the following. Other aspects of the tool are detailed in Section 6.1. In CELLO's circuit model, each gate is represented by the transfer function of its individual TF-promoter pair. The function is a Hill curve (Section 3.3) with parameters empirically determined in the lab. In this process called *flow cytometry*, the fluorescence of individual cells of a population is successively measured. If the cells contain single genetic gates and a fluorescent reporter output gene, the result is a population-wide distribution of gate output values for a given input value. The parameters characterizing the gate are then inferred based on the medians of the distributions for both Boolean states. In Figure 4.3, resulting Hill functions are shown for a circuit of four gates, with each gate being related to its transfer function by the color coding. The OR gate does not have an individual transfer characteristic but represents a simple superposition of the activities of its two input promoters (Section 6.1). All signal levels in the circuit are given in relative promoter units (RPU), which relate the activities of all promoters in the library to a common standard of comparison. In contrast to the basic gate model presented in Section 3.3, the input signals in CELLO's model are given in the RPU of the gate's input promoters instead of repressor concentrations. The result is a uniform quantitative representation of signals in the circuit.

The circuit output RPU value is calculated by propagating the RPU values from the circuit input promoters through the gates' transfer functions on each path to the output. Each combination of circuit input promoter activities, i.e., each Boolean input assignment, leads to a unique circuit output RPU value. This is due to the different activity levels of the input promoters and the different signal paths in the circuit consisting of heterogeneous gates. In Figure 4.3, the calculation of the circuit output value is visualized for the first input assignment. Here, both circuit inputs are assumed to represent a Boolean 0, and thus, both input promoters deliver their individual basal activity. Each further input combination leads to a different output level, even if output values represent the same Boolean value. For a circuit to fulfill its functional specification given by a Boolean formula, it is required to produce output RPU levels, which are clearly assignable to a Boolean value. To this effect, CELLO rates the minimum separation of the output values representing complementary Boolean states. Let γ and a denote the abstract circuit topology and the assignment of genetic gates from the library to this topology. Let furthermore $\overline{Y}$ be the set of circuit output RPU values representing a Boolean 1 and $\underline{Y}$ the RPU values representing a Boolean 0. Then,

$$s_{\text{Cello}}(\gamma, a) = \frac{\min_{\overline{y} \in \overline{Y}} (\overline{y})}{\max_{\underline{y} \in \underline{Y}} (\underline{y})} \tag{4.1}$$

defines the metric applied by Cello, subsequently called the *Cello score*. It is given by the ratio of the smallest output RPU value for a Boolean 1 and the largest value for a Boolean 0. In Figure 4.3, these values are labeled "lowest ON" and "highest OFF". The score rates the degree to which a circuit produces identifiable output states and thus implements the specification. To compute the Cello score, it is necessary to exhaustively determine a circuit's output RPU for all Boolean input assignments. In the following, this process is called *simulation* or *evaluation* of a circuit.

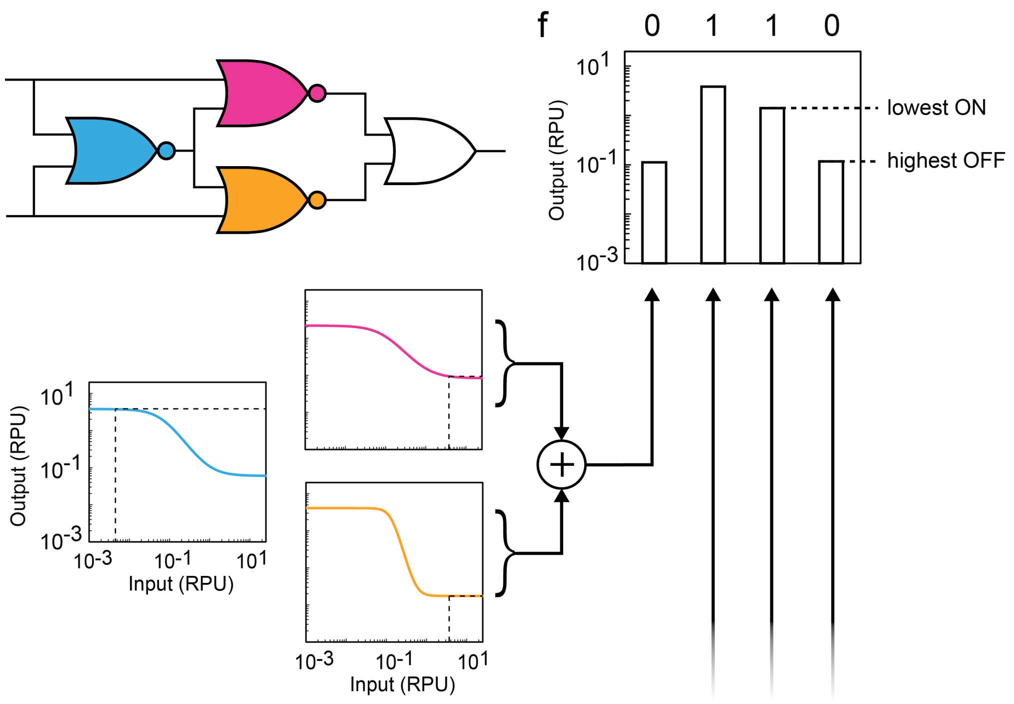


Figure 4.3.: Determination of the Cello circuit score (excerpt from [53])

4.3. Quantifying Circuit Robustness

CELLO's circuit model and the corresponding circuit score rely on median representations of gates. As explained in Section 4.2, these are inferred from distributions measured by flow cytometry of a population. Thus, they do not accurately capture the differences between individual cells, the *cell-to-cell variability* [63]. The term refers to the behavioral variance among cells of an isogenic population, i.e., cells with a largely identical DNA. It is caused by, e.g., the cells being in different stages of the cell cycle and by the stochastic nature of fundamental bio-molecular events [11]. Measuring the output signal of a genetic circuit implemented in a population of isogenic cells thus yields distributions of output signals. By sorting the output values with respect to the represented Boolean value, a distribution of output values is obtained for each Boolean state. These distributions generally feature different variances and shapes for different circuit designs. This may even lead to an overlap of the two distributions representing complementary Boolean states. Any overlap should be avoided by design since it results in a non-functional circuit lacking differentiable output states [59]. This is especially important for biomedical applications [37]. For example, malfunctioning circuits for identifying cancer cells may damage healthy cells [70]. In the following, the term *robust* is used, referring to circuits resistant to output overlap. To make robustness an objective in genetic circuit design, it has to be quantified and incorporated into the design process. This was the goal of a joint work [58] with the Self-Organizing Systems Lab, which contributed the alternative metric called expectation-based score (E-Score) presented in the following.

Principle of the E-Score In contrast to the CELLO score, the E-Score is not only sensitive to the distance of the medians of the output distributions of a circuit but also to their variance. In a design process, this enables differentiating designs with the same distance of medians but different variances. Thus, designs less prone to overlap, i.e., robust designs, can be selected based on their E-Score. The calculation of the score can be divided into three phases. First, the gate models, typically based on a single set of parameters for the Hill transfer functions, are enhanced to reflect the gates' stochastic behavior. For each gate, a set of random parameters is inferred from the flow cytometry data that captures its output distribution. Second, multiple circuit simulations are carried out instead of a single simulation. For each one, a different set of the random parameters is drawn for each gate. By this, the stochasticity of the circuit is introduced into the simulation, resulting in a set of simulated circuit output values. The values are then grouped by the Boolean state they represent, yielding a distribution of output values for each Boolean state. Finally, the circuit is scored based on a modified Wasserstein metric [28, 67]

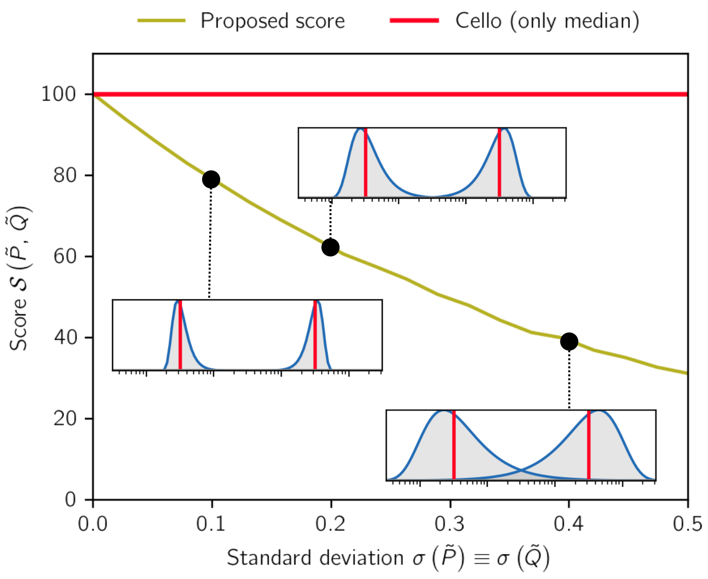


Figure 4.4.: Sensitivity of the E-Score ("proposed") to the standard deviation of output distributions in contrast to the CELLO score (excerpt from [58])

calculated for the output distributions. "The Wasserstein distance of the logarithmic output distributions emerges as a natural measure of separation corresponding to the population-wide expected on-off difference" [58]. It has been modified only to include the parts of the distributions that are visually located between the medians. As this is the distributions' section prone to overlap, the metric represents a measure for circuit robustness. In contrast to the original Wasserstein distance, this modification also allows scoring the variance of symmetric distributions. Figure 4.4 visualizes the principle of the E-Score and highlights its sensitivity to variance compared to the CELLO score. On the x-axis, the standard deviation of the output distributions is plotted, and the y-axis shows the normed score value. The medians' distance of the distributions is fixed. For this fixed distance, the CELLO score (red line) is constant irrespective of the distributions' standard deviation. In contrast, the E-Score's value, depicted in yellow, decreases with increasing standard deviation. Intuitively, the left output characteristic would be preferred over the other two, which feature nearly overlapping and overlapping distributions. This notion is quantified by the E-Score.

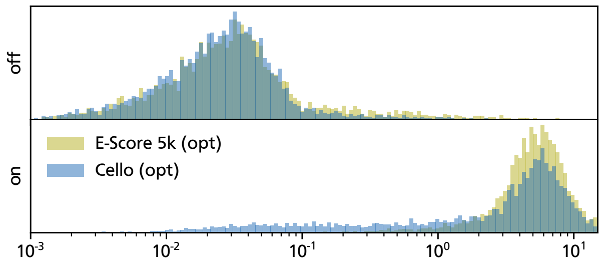
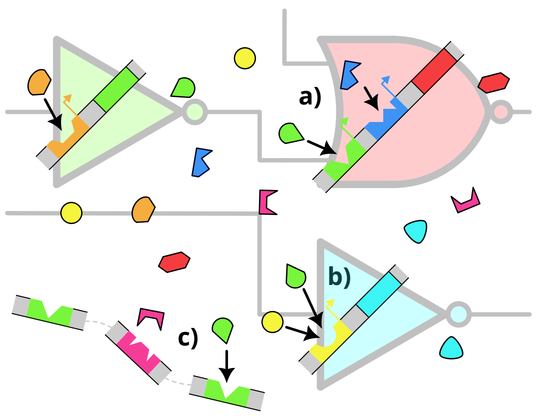


Figure 4.5.: Output distributions of an exemplary circuit optimized with respect to the E-Score and the Cello score (excerpt from [58])

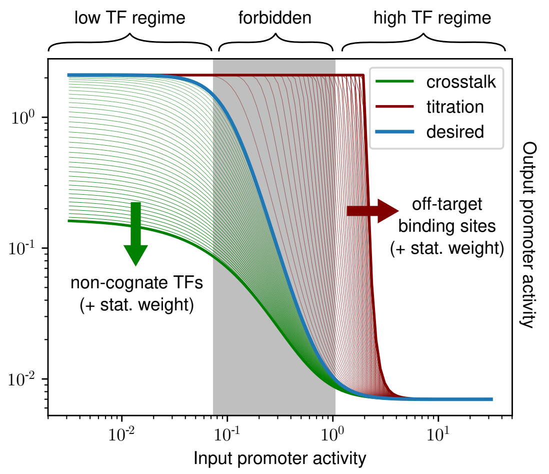
Impact of the E-Score Integrated into a circuit design process, the E-Score can lead to more robust circuit designs than a score agnostic to output variance. That is, resulting circuits are less prone to non-functionality caused by cell-to-cell variability. In [58], the robustness of 20 out of 31 circuits could be improved by using the E-Score instead of the Cello score. The median E-Score of the improved circuits increased by $71.08\,\%$. The resulting output distributions of two circuit designs with the same function are shown in Figure 4.5. In this depiction, which will recur throughout this work, the output RPU is given on the x-axis and the y-axis indicates the normalized count. In blue, the output characteristic of a design optimized with the Cello score is shown, and in yellow, a design optimized with the E-Score is shown. The top plot shows an overlay of the distributions representing a Boolean 0 output, and the bottom plot shows the complementary Boolean 1 output. While both designs feature comparable medians of their output distributions, the shapes are different. This is especially visible for the Boolean 1 case, where the design optimized with the E-Score features a narrower distribution with fewer portions overlapping with the complementary distribution. All of the presented results have been obtained by calculating the E-Score based on 5000 random parameterizations of the gates in the circuit. As each of these instances is evaluated using a circuit simulation (Section 4.2), determining the E-Score is $5000\times$ computationally more expensive compared to determining the Cello score. However, applying parallel computation techniques can reduce the resulting overhead in computation time.

4.4. Context-Aware Circuit Model

Besides cell-to-cell variability, interactions among parts of the genetic circuit and between these parts and elements of the host organism affect the performance of synthetic circuits [16]. The term *context effects* is used to subsume these undesired interactions, which are



(a) Schematic view



(b) Effect on transfer functions

Figure 4.6.: Crosstalk and titration in genetic circuits (excerpt from [24])

caused by multiple mechanisms. Synthetic parts and the host, for example, rely on and compete for energy resources of the cell, especially for translation [17]. Furthermore, the genes located on the DNA near a synthetic part affect its function [42], and retroactivity modulates the behavior of circuit elements based on the temporal context [33]. Two further important interactions are directly related to the gene regulation, caused by the fact that even though TFs exhibit the strongest binding affinity to their cognate promoter, they generally also bind to other sites. First, TFs can bind to other promoters of the synthetic circuit, leading to interdependencies of the circuit's signals. Analog to electronic circuits, this phenomenon is called *crosstalk* [25]. Second, TFs can bind to other non-cognate sites at the host genome in a process called *titration* that limits their availability as a functional component in the circuit [13]. Despite their effect on circuit function, crosstalk and titration are not considered in state-of-the-art GDA workflows. However, these phenomena can be incorporated by refining the model of gene regulation used for simulating the circuit and adapting the synthesis process accordingly. This was the goal of another joint work [24] with the Self-Organizing Systems Lab, which contributed the new circuit model that is presented in the following, starting with a more detailed explanation of the two considered effects.

Crosstalk and Titration An ideal library of regulatory parts consists of a set of TF-promoter pairs that do not interact with each other, i.e., that are orthogonal. Each

TF binds only to its cognate promoter, and no interactions between other elements of the circuit or the host occur. However, it has been shown that generally TFs bind to other sites than their cognate promoters [25]. Both crosstalk and titration are caused by this limited binding specificity of TFs. Despite having the same mechanism, both phenomena are differentiated due to their individual effect on the circuit function. Figure 4.6a visualizes the biology of the desired regulation and both undesired effects in genetic circuits. The colored lines inside the logic gates depict functional parts of the DNA strand. Stamped sections on the strands portray promoters where fitting TFs can bind, while solid sections depict coding sequences for TFs. The TFs itself are visualized by the small floating elements. Their geometric similarity to the stamped parts of the strands indicates their binding affinity to the respective promoter. Their color links them to their coding sequences and cognate promoters. Of the three phenomena shown, (a) represents the desired binding of TFs to their cognate promoters, (b) represents the crosstalk of a TF (green) with a non-cognate promoter and (c) shows the titration of TFs by sites outside of the circuit at the host genome. As shown in Section 3.3, the desired transcriptional regulation results in a gate transfer characteristic of a Hill curve. Both crosstalk and titration modulate the curve differently, shown in Figure 4.6b for the inhibitory case. In the low TF regime, where none or a limited number of TF molecules bind to the cognate promoter, its activity is maximal. The presence of a non-cognate TFs exerting crosstalk on the promoter leads to reduced activity, depicted in green. Thus, crosstalk reduces the output of a gate that represents a Boolean 1. The minimal output of a gate is not reduced, as the basal promoter activity defines its lower bound. Titration reduces the availability of TFs that can bind to the promoter, as they are drawn to binding sites outside the circuit. Thus, a higher concentration of TF is needed to regulate the promoter when titration is prevalent compared to the case free of titration. This results in a shift of the high output region of the gate, shown in red. Note that the modulated transfer function is still bound by the output interval $[y_{\min}, y_{\max}]$, and its sigmoidal characteristic persists.

Thermodynamic Circuit Model A convenient way to model off-target binding effects is that of a thermodynamic model presented in [24]. It is based on two main assumptions. First, the input promoter activity is proportional to the concentration of TF available for binding, which is in turn proportional to the expected occupancy of the possible binding sites by a RNAP. Second, single binding states, i.e., "[. . .] the permutations of assignments of available molecules to available binding sites, follow a Boltzmann distribution" [24]. Therefore, all global site occupancy states with the same energy level are equally probable. "Thus, the model reduces the underlying complicated mechanics of gene expression to combinatorics on the ensemble of RNAP binding states" [24]. The result is a parameter-reduced model of genetic circuits that only relies on the binding energies of TFs and RNAP to binding sites and proportionality constants. These parameters can be learned from existing gate libraries. Crosstalk is modeled by setting the binding energy of TFs not only for cognate promoters but also for other promoters of the library. Titration is reflected by static binding sites and respective energies that are not part of regulatory circuit parts but of the host organism. Through crosstalk, interactions in the circuit can arise that represent signal feedback and, thus, cyclic dependencies of signal levels. This increases the computational complexity significantly, since root-finding algorithms have to be used to find the solution.

4.5. ARCTIC

The examination of technology mapping methods for genetic circuits lead to the joint development of the GDA tool ARCTIC ("Automatic design of Robust CircuiTs in Cells") with the Self-Organizing Systems Lab. It comprises two main parts complementing each other (Figure 4.7). ARCTICsim represents a collection of two circuit simulators written in Python and multiple libraries of genetic gates. The classic simulator applies CELLO's circuit model (Section 4.2) to determine the output behavior of circuits and to calculate circuit scores such as the CELLO score and the E-Score (Sections 4.2 and 4.3). It uses gate libraries containing classical Hill parameterizations of genetic gates and libraries containing random perturbations of these parameters to enable the calculation of statistics-based scores. The thermodynamic simulator is based on the context-aware circuit model (Section 4.4) and can calculate the CELLO score of circuits under consideration of context effects. These effects are modeled in a corresponding set of gate libraries containing binding strength configurations for different crosstalk characteristics. The second building block of ARCTIC is ARCTICmap, a technology mapping tool written in Java. All methods presented in this work are implemented in ARCTICmap and are detailed in the remainder of this work. In short, ARCTICmap provides methods for synthesizing abstract circuit topologies for a specific Boolean function based on the given genetic gate library and problem-specific exact and heuristic optimization methods for assigning specific genetic gates to these circuit topologies.

In a typical technology mapping flow with ARCTIC, the user enters a Boolean specification of the desired circuit and defines settings such as the gate library, topology synthesis constraints, gate assignment method, circuit model, and circuit score. ARCTICmap then queries information on the genetic gate library that matches the settings. Based on this, it generates circuit topologies and corresponding gate assignments and sends them to the simulator. The simulator evaluates each design, calculates its score, and sends it back to ARCTICmap. This feedback enables the exploration of the design space for circuit implementations in an optimization loop. When the chosen technology mapping method converges, ARCTICmap returns the resulting circuit implementation. Further, ARCTICsim can be used to analyze and visualize the output behavior of the circuit.

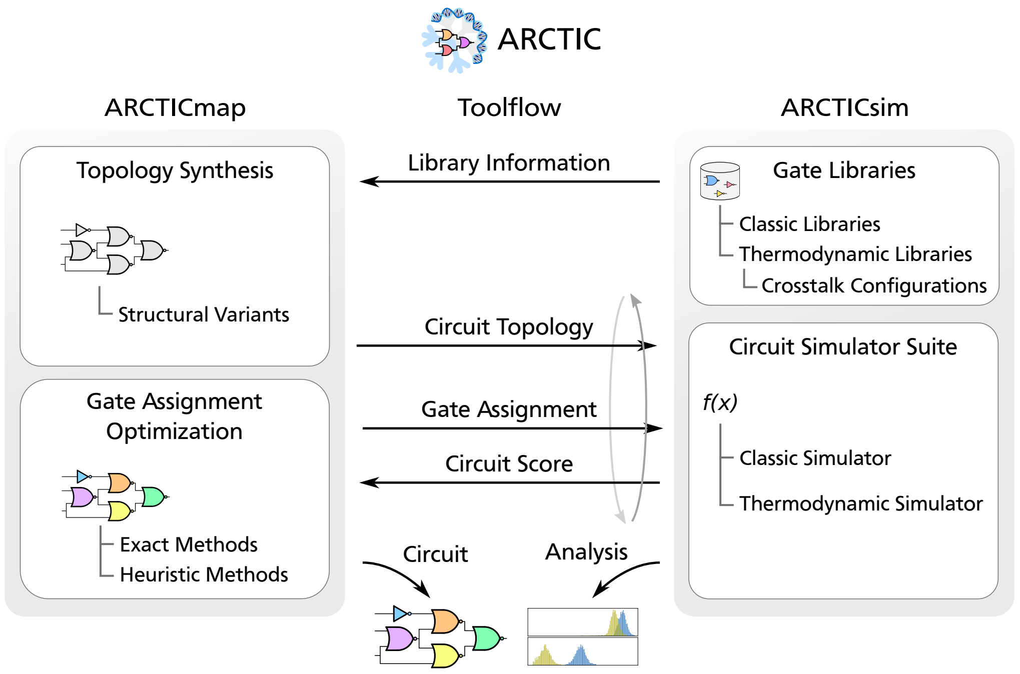


Figure 4.7.: Structure and flow of ARCTIC

5. Problem Definition

This chapter defines the problem of technology mapping for combinational genetic logic circuits in detail. First, the examined design steps are contextualized by locating them in the GDA tool flow and in comparison to a typical EDA flow in Section 5.1. Then, the problem is defined formally in Section 5.2.

5.1. Contextualization

A GDA tool flow for genetic logic circuits and a typical EDA flow are shown side to side in Figure 5.1. The main phases of the design process are printed on the right in black and characterize a variety of EDA flows for different target technologies. The shown technology mapping phases are that of a structural mapping approach (Chapter 2), which is used, for example, for standard-cell designs. The EDA flow begins with the input of a functional specification of the system. Typically, the input is given on the register transfer level (RTL) that defines the operations performed on the data and, in the case of a sequential system, the registers used to hold the data. Domain-specific languages called hardware description languages (HDLs), exist to conveniently describe systems on the RTL. The HDL input can be entered directly by a user or result from a high-level synthesis process that transforms a behavioral description of the system into the RTL representation. Next, the logic synthesis generates a Boolean representation of the system from the RTL description. Particularly, combinational paths are transformed into Boolean formulae, which are typically minimized. In technology mapping, each formula is transformed into a subject graph that is matched with a library of technology elements. A cover of these elements is chosen that minimizes an objective such as delay or area, or a combination of both. In the phase of physical design, the final layout is constructed by placing technology elements geometrically and routing the signals between them. Again, the physical design represents an optimization for design objectives, such as the critical path length. The output of the EDA flow is a set of design files fully describing the implemented system that allows, e.g., building the device in a fab or programming field programmable devices.

While the GDA flow can be subdivided into the same main phases as the EDA flow, the

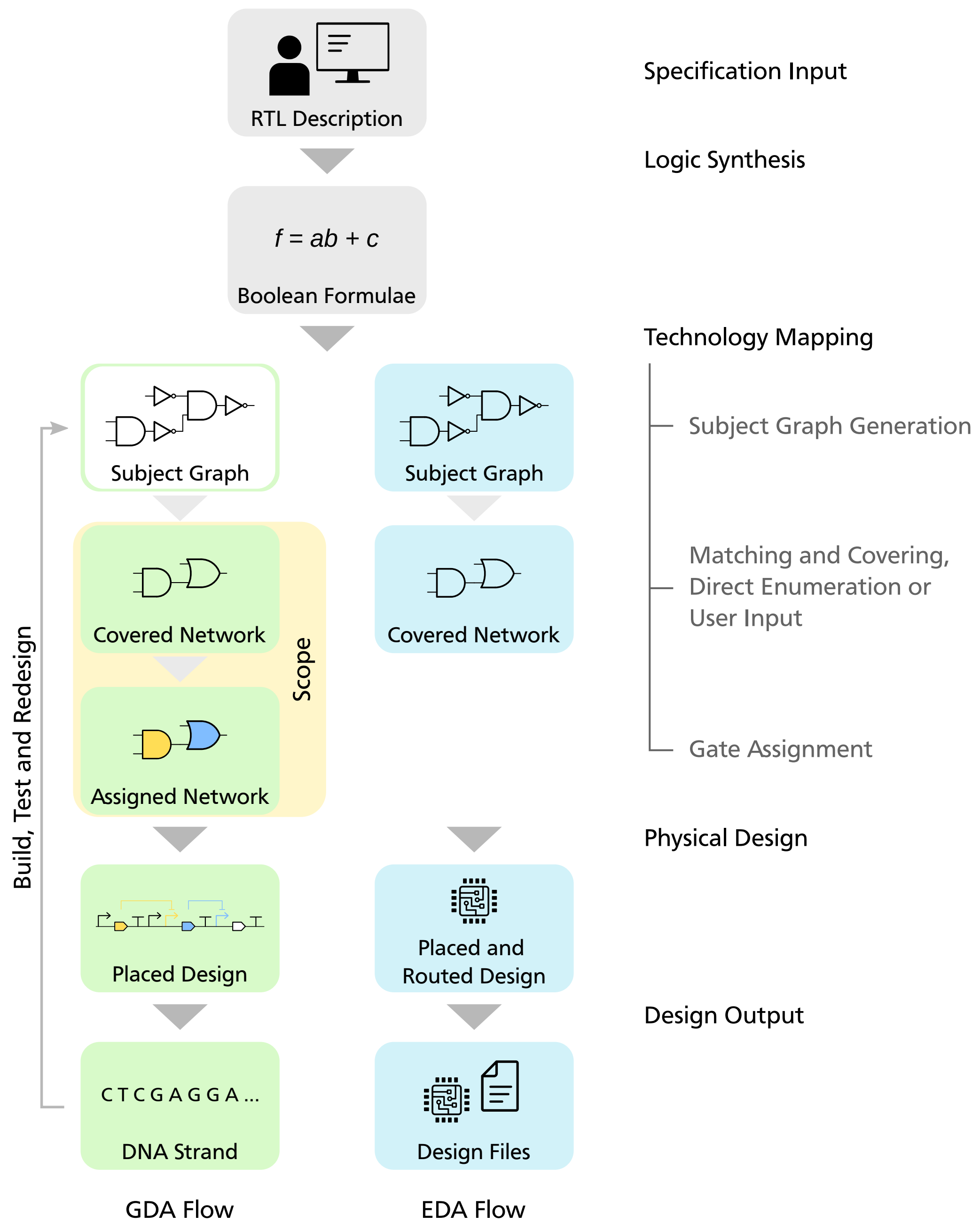


Figure 5.1.: GDA tool flow and typical EDA tool flow based on structural mapping

processes performed in each step differ. This is caused by the fundamental differences between electronic and genetic circuit technology and especially applies to the phase of technology mapping considered in this work. Furthermore, the landscape of existing GDA tools and methods is still heterogeneous. While in the following, a generalized GDA flow is used to contextualize the examined problem, a detailed survey is given in Chapter 6.

Different approaches for structuralizing the Boolean representation of genetic circuits exist. While some rely on traditional subject graphs, others skip it entirely and demand a structural user input. In this work, it is shown that given the small size of genetic logic circuits currently implementable, it is possible to directly generate a network of library elements from the Boolean representation by enumeration. In contrast to EDA, the covered network does not represent the final product of the technology mapping phase. It is merely a network of abstract logic gates that can, in principle, be implemented by the given library. That is, the network contains at most the number of gates of each logic type present in the library as orthogonal genetic gates using individual signaling molecules. In the additional phase of gate assignment, these genetic gates are mapped to the abstract network, and the circuit characteristic crystallizes. Gate assignment represents a non-trivial step in the design process, in which the specific realization of the circuit is generated and optimized for an objective function. In the physical design phase, the selected parts are placed on the DNA strand. Methods of combinatorial design [62] ensure that the order of genetic elements adheres to predefined rules reflecting experiences with the compatibility of parts. For systems implemented in a single cell or a cell-free environment, routing of signals is not applicable since all signals, i.e., TF concentrations, share the same medium. Thus, the routing step known from EDA has no counterpart in genetic circuit design. The output of the GDA flow is a sequence of base pairs that can be used to synthesize a DNA strand describing the genetic circuit. Due to the complex nature of the examined systems, the paradigm of "Design, Build, Test" [5] is prevalent in synthetic biology. To reduce the number of re-design cycles and thus expensive lab work, it is desirable to predict the system behavior with high precision and conclude the synthesis flow with a (near) optimal design with respect to that prediction.

The problem examined in this work is the technology mapping for combinational genetic logic circuits. The presented methods transform a functional Boolean specification of a circuit into a network of specific genetic logic gates optimized for a performance metric.

5.2. Formal Definition

Note: Parts of this section have already been published in [58]. Explicit self-citation is omitted to provide an improved reading flow.

In the following, the problem of mapping a Boolean function to a combinational genetic logic circuit is defined formally.

Prerequisites The circuit's functional specification is input by the user or results from logic synthesis. It is given as a Boolean formula ϕ from the space of possible specifications $\mathcal{F}$ mapping an n-dimensional Boolean input vector to a Boolean output value, i.e., $\phi \in \mathcal{F} : \mathbb{B}^n \rightarrow \mathbb{B}$ with $\mathbb{B} = \{0, 1\}$. That is, the circuits considered in this work feature a single output. Further, a genetic gate library $\mathcal{L}$ is given representing a finite set of m individual genetic logic gates, i.e., $\mathcal{L} = \{\theta_0, \theta_1, \dots \theta_{m-1}\}$ which are each represented by an individual set of Hill parameters $\theta = (y_{\min}, y_{\max}, K, n)$. Additionally, the set $\mathcal{S}$ contains the available logical gate types, i.e., the Boolean functions implemented by gates in the library.

Search Space Let $(\mathcal{G}, \Sigma)$ be the set of all labeled DAGs where $G \in \mathcal{G}$ is a DAG with $G = (V, E)$, $E \subseteq V \times V$ and labeling $\Sigma : V \rightarrow \mathcal{S}$. During the synthesis of the network structure in technology mapping, the finite set of circuit topologies $\Gamma \subset (\mathcal{G}, \Sigma)$ based on the synthesis map T from the space of specifications in terms of Boolean formulae $\mathcal{F}$ and the available gate types $\mathcal{S}$, i.e., $T : \mathcal{F} \times \mathcal{S} \rightarrow (\mathcal{G}, \Sigma)$ is searched. To make the transformation into a structural representation tractable, T is typically split into two phases. The first step is synthesizing a subject graph based on a technology-independent library, and the second step is generating a technology-dependent, covered network. Note that both phases of T are traditionally guided by an objective function related to network properties. In the given problem, T is assumed to be unguided since important metrics of the genetic circuit do not crystallize before the gate assignment optimization. Thus, T does not necessarily yield a single network implementing ϕ, but a set of networks Γ.

In the process of gate assignment, an injective function M that takes each vertex of a topology γ in Γ and assigns it one individual element of library $\mathcal{L}$, i.e., $M : \Gamma \times \mathcal{L} \rightarrow \mathcal{A} \subset V \times \mathcal{L}$ is searched. The set $\mathcal{A}$ contains all valid assignments of genetic gates to the network, that is, all assignments in which the type of each gate matches the type of the node it is assigned to and each individual TF is only used once in the circuit. The set $\Gamma \times \mathcal{A}$ represents the design space of technology mapping.

Optimization Problem Topology synthesis and gate assignment jointly result in circuits (γ, a) with $\gamma \in \Gamma$ and $a \in \mathcal{A}$. Rating a circuit is then done using a circuit score function

$S : \Gamma \times \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$. To serve as a circuit score, a function S must quantify the compliance of the circuit's characteristic with the Boolean functional requirement ϕ, but may also cover additional aspects. Proceeding from here, the process of technology mapping can be formulated as an optimization problem of the form

$$(\gamma^*, a^*) = \underset{(\gamma,a) \in \Gamma \times \mathcal{A}}{\arg\max} \; S(\gamma, a) \tag{5.1}$$

with (γ^*, a^*) being the optimal topology and assignment combination with respect to S.

Equation (5.1) illustrates the hierarchical nature of the problem at hand. Generally, a circuit can only be scored if a network topology and a complete assignment of gates to that network exist. To make the problem tractable, it is a practical approach to split it into two phases based on the inherent hierarchy. First, the set of topologies Γ is built. In contrast to EDA, metrics of the network structure do not necessarily correlate to the main objective, and thus, a new approach for topology synthesis is necessary. This problem is examined in more detail in Chapter 7. Then, the space of gate assignments $\mathcal{A}$ is explored for each topology $\gamma \in \Gamma$. Problem-specific search methods can leverage characteristics of the chosen scoring function and circuit model to traverse this space efficiently in the search for a (near) optimal solution. Different aspects of the gate assignment problem are examined in Chapters 8 to 10.

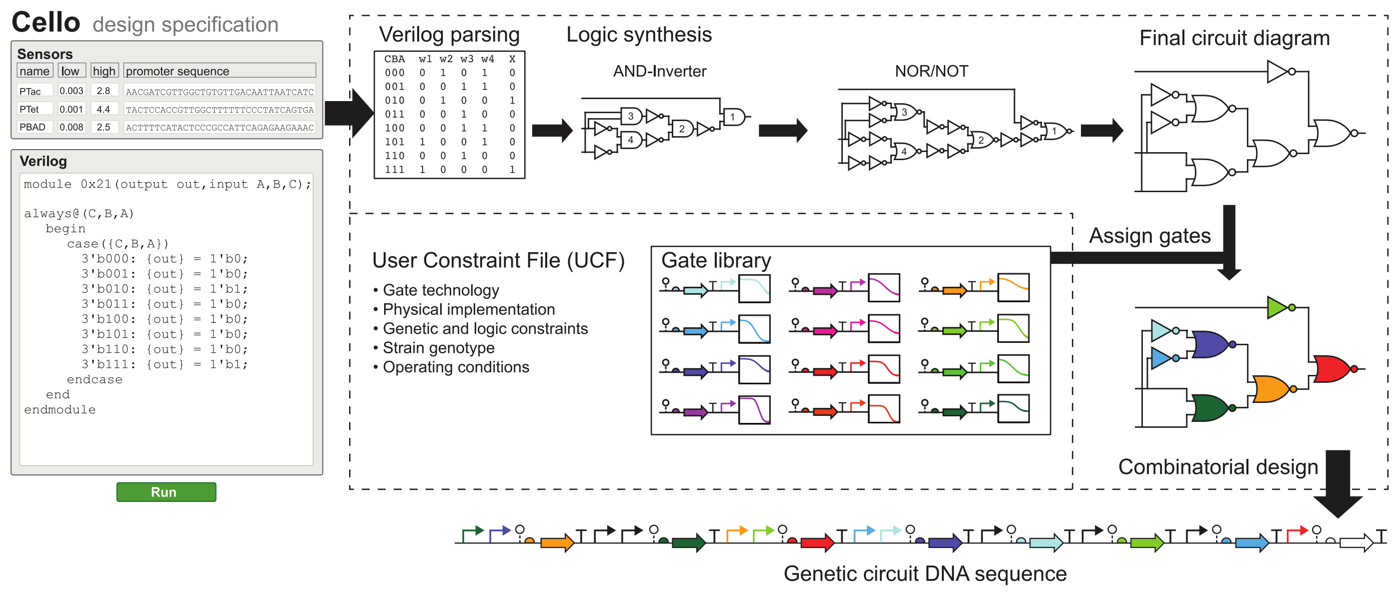
6. Related Work

In the following, existing technology mapping approaches for genetic logic circuits are introduced and contextualized. First, the GDA tools Cello and iBioSim are presented in detail in Sections 6.1 and 6.2 due to their high relevance to the work. Further relevant approaches are subsumed in Section 6.3.

6.1. Cello

The GDA tool Cello [53] and its accompanying library of genetic logic gates contributed significantly to the field of genetic logic circuit design, and many aspects of it represent the state-of-the-art. The tool covers the complete synthesis flow from a textual circuit specification to the output of a physical design in the form of a DNA strand (Figure 6.1). As input, Cello takes a subset of the Verilog HDL describing a single combinational function. The Verilog code is parsed, and the function is transformed into a truth table representation. It serves as an input for the subsequent process called logic synthesis in the original work, which, in fact, represents the first step of the technology mapping, namely the subject graph generation. The technology mapping approach of Cello, spanning the mentioned generation of the subject graph, its transformation into a network of abstract gates, and the gate assignment optimization will be detailed below. The physical design is implemented by a process called combinatorial design in which the genetic parts are placed on the DNA strand following the library's specifications.

Gate Library The genetic gate library provided with the original Cello framework [53] is the result of extensive specification of gates in the lab, identifying modes of failure and reducing them by re-design. It contains a set of 12 repressors based on the tetracycline repressor (TetR) with corresponding cognate promoters implementable in *E. coli*. The repressive regulatory characteristic of these TF-promoter pairs enables their usage as NOT/NOR gates (Section 3.3). For some of the gates, the library contains variants using different RBS, leading to different transfer characteristics. In total, 20 individual genetic gates are present in the library. An individual strong terminator follows each coding



Sensors

name	low	high	promoter sequence
PTac	0.003	2.8	AACGATCGTTGGCTGTGTTGACAATTAATCATC
PTet	0.001	4.4	TACTCCACCGTTGGCTTTTTTCCCTATCAGTGA
PBAD	0.008	2.5	ACTTTTCATACTCCCGCCATTCAGAGAAGAAAC

Verilog

```
module 0x21(output out,input A,B,C);

always@(C,B,A)
   begin
      case({C,B,A})
         3'b000: {out} = 1'b0;
         3'b001: {out} = 1'b0;
         3'b010: {out} = 1'b1;
         3'b011: {out} = 1'b0;
         3'b100: {out} = 1'b0;
         3'b101: {out} = 1'b0;
         3'b110: {out} = 1'b0;
         3'b111: {out} = 1'b1;
      endcase
   end
endmodule
```

Verilog parsing

CBA	w1	w2	w3	w4	X
000	0	1	0	1	0
001	0	0	1	1	0
010	0	1	0	0	1
011	0	0	1	0	0
100	0	0	1	1	0
101	1	0	0	1	0
110	0	0	1	0	0
111	1	0	0	0	1

User Constraint File (UCF)

- Gate technology
- Physical implementation
- Genetic and logic constraints
- Strain genotype
- Operating conditions

Figure 6.1.: Overview of CELLO's tool flow (excerpt from [53])

sequence to prevent RNAP read-through, and individual insulator sequences insulate promoters from the surrounding genetic context. Additionally to the gates, CELLO's library contains four promoters that serve as circuit inputs since they are sensitive to TFs that can be controlled by inducer molecules from outside the cell. As circuit output, the library provides a YFP reporter gene, which allows measuring the output signal. Two promoters can be placed in front of the YFP's coding sequence, which results in the behavior of an OR gate of the output gene.

Technology Mapping CELLO's technology mapping process consists of three phases: a subject graph is generated, transformed into a network of NOT and NOR gates, and finally assigned with library gates. To generate a structural representation of the input truth table, an AIG is built using the tool ABC [12]. Being an EDA tool, ABC optimizes the depth of the graph by applying a heuristic minimization to the AIG since this would reduce the delay in resulting electronic circuits. As a secondary objective, the number of AND nodes is minimized. CELLO then transforms the AIG into a NOR-Inverter Graph (NIG) by local substitutions of AND nodes according to DeMorgan's rule. By this, the technology-independent subject graph is transformed into a network of NOT and NOR gates in the library. In a final modification of the topology, CELLO tries to substitute sub-graphs of the NIG by pre-computed circuit motifs. This step aims to further minimize the heuristic output of ABC and serves as an interface for entering potential gate types other than NOT and NOR. If the resulting circuit topology is implementable with the gate library, i.e., the number of gates of each type does not exceed the number of gates in the library, CELLO continues with the gate assignment.

To solve the assignment problem, a Simulated Annealing (SA) [56] heuristic is applied. It represents a general-purpose optimization algorithm for combinatorial problems and is explained in more detail in Chapter 8. In short, a random initial assignment of gates to the circuit structure is optimized iteratively. In each iteration, an assigned gate is replaced by another gate from the circuit or the library. While changes improving the solution quality are always accepted, worsening changes are accepted with a certain probability that decreases based on an annealing scheme. SA features a set of parameters that influence the optimization efficiency and solution quality. These parameters' values must be found in a meta-optimization since they are problem-specific. Furthermore, parameters may be tied to certain features of the specific problem to allow the algorithm to work efficiently for various problems, e.g., with varying sizes. CELLO, however, uses fixed values for critical parameters such as the starting temperature of the annealing scheme and the number of iterations (Section 8.4.3). The circuit model and objective function used in the optimization scheme are described in Section 4.2. That is, CELLO optimizes the separation of the complementary Boolean output states of the circuit based on the functional specification. Additionally, the burden ("toxicity") of each assigned circuit is calculated based on empirical values contained in the library. It rates the degree to which a synthetic circuit hampers cell growth. Circuit candidates exceeding a certain toxicity threshold are discarded during the optimization. Further, CELLO offers a heuristic check for the robustness of circuits to signal perturbations by "[...] ensuring that the minimum and maximum output values of a gate lead to an output of the subsequent gate that is higher than half the maximum or lower than twice the minimum output value" [24]. The problem of signal value compatibility is examined in greater detail in Section 9.2.

Of 60 Boolean three-input functions synthesized with CELLO, 45 were fully functional when verified *in vivo* [53]. Of all output states produced, 92 % featured the predicted Boolean signal value. Since the publication of CELLO, the successor CELLO 2.0 [34] has been published. It is based on a new, more modular software architecture aimed towards design automation for other organisms than *E. coli* such as yeast [19]. However, it features the same technology mapping methods present in CELLO.

6.2. iBioSim

The GDA tool iBIOSIM [51], currently available in its third version [68], represents an actively developed collection of models, simulation methods, and interfaces for standards such as SBOL. In contrast to CELLO it operates on the lower abstraction level of biochemical reaction network models and is thus not limited to genetic logic circuits. The tool also allows the modeling of "[...] metabolic networks, cell signaling pathways, and other biological and chemical systems [...]" [51] with a range of different analysis methods, including the simulation of temporal behavior. However, iBIOSIM contains a technology mapping scheme for genetic logic circuits [55], detailed in the following. It is in parts based on the general idea of the branch-and-bound paradigm, similar to the method presented in Section 8.3.

As input, iBIOSIM's technology mapping method takes a structural model of the GRN instead of a Boolean specification. iBIOSIM possesses no notion of the expected functional behavior of the resulting circuit as a whole. A functional specification is only present on the level of interactions between components of the input network. These are annotated either as activating or inhibiting relations. In the first phase of the scheme called graph construction, a subject DAG of the network is constructed with nodes representing promoters and coding sequences for TFs. Figure 6.2 shows an exemplary resulting DAG on the left. Diamond-shaped nodes represent promoters, and ellipse-shaped nodes represent

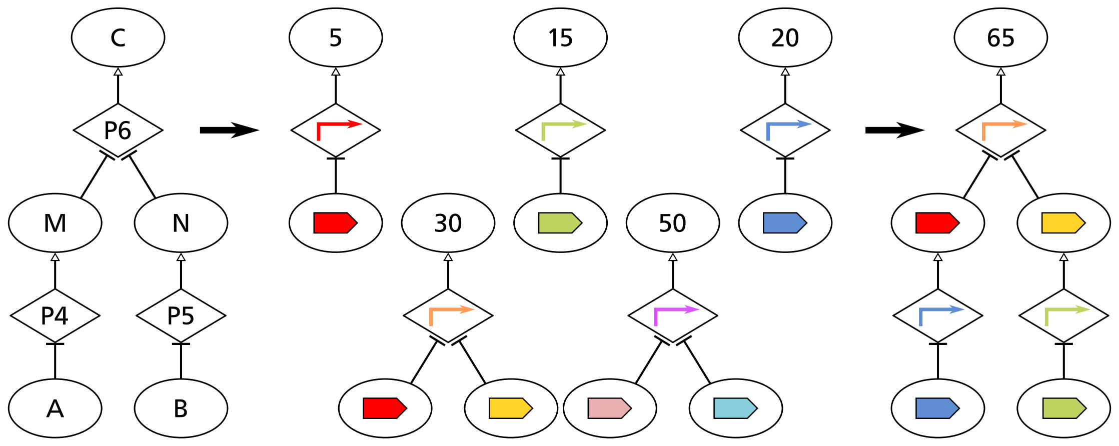


Figure 6.2.: Technology mapping scheme of iBioSim (adapted from [55])

coding sequences. Edges drawn from coding sequences to promoters depict the binding affinity of a coded TF to a promoter, and edges drawn from promoters to coding sequences depict control of transcription by the promoter. The graph construction is carried out for the input model and all library components. Thus, the system specification and the technology elements are present in structural form, making the examined technology mapping scheme a structural mapping method using structural matching (Section 2.1). As a final step in the first phase, all DAGs representing library elements are annotated with their size in the number of base pairs (Figure 6.2, center).

If the created problem DAG features nodes with more than one outgoing edge, it is split into multiple rooted DAGs representing a network with a maximum fan-out of one in a step called partitioning. These sub-problems are then solved separately. Consequently, the global solution that combines the sub-solutions is not guaranteed optimal with respect to iBioSim's objective, the number of base pairs. All rooted graphs of the problem and the library are then decomposed into a canonical form consisting only of NOT and NOR interactions. This allows structural matching, which is carried out in topological order of the problem DAG. For each node, the rooted sub-graph starting at that node is matched with all graphs in the library. Each match is recorded with the minimal cost in base pairs of the partial solution, serving as a lower bound. After matching, a minimal cover of matches is searched by depth-first search beginning at the root of the problem DAG. As potential crosstalk caused by the repeated usage of the same TFs is not considered during covering, the algorithm has to perform backtracking as soon as such an illegal combination occurs. This results in a branch-and-bound scheme that generates a valid solution in the sense

Table 6.1.: Tool flow of the examined GDA tools

	Input				
Tool	Type	Structural	Functional	Abstraction	Output
Cello	HDL	-	✓	gates	DNA
iBioSim	GRN	✓	-	parts	DNA
MatchMaker	GRN	✓	-	parts	DNA
SBROME	abstr. GRN	✓	✓	parts	GRN
Parts&Pools	truth table	-	✓	parts	GRN
GeneTech	algebraic	-	✓	gates	GRN

that each TF is only used once in the circuit (Figure 6.2, right). Furthermore, the solution for each rooted DAG is minimal with respect to the number of base pairs. iBioSim outputs structural information for the resulting network, including a DNA sequence that can be used to build the circuit *in vivo*.

iBioSim's scheme represents one of the first technology mapping methods for genetic circuits guided by a cost function. It optimizes the design regarding the number of base pairs necessary to implement it. However, there is no mechanism to evaluate the circuit's functionality during technology mapping. This lack of notion of functionality represents a strong limitation of the approach.

6.3. Further Mapping Approaches

In this section, further technology mapping approaches used in GDA tools are subsumed. The early stage of development of GDA is reflected by the heterogeneous tool flow of the examined tools. While some tools transform a functional specification into a circuit topology before mapping components to it, others expect a GRN to be input by the user. Early GDA tools operate on the abstraction level of genetic parts, e.g., promoters, RBS, and coding sequences. In contrast, more recent tools adapt the gate level abstraction from EDA. The flow of some tools ends with the output of a GRN, while others perform physical design and generate a linearized design in the form of a DNA strand. Table 6.1 provides an overview of the different tool flows. Further, the optimization performed during technology mapping differs significantly. In most of the tools, the assignment of the considered building blocks to the network is subject to optimization, and none of the tools optimizes the network topology with respect to the main objective. The objectives range

Table 6.2.: Design optimization performed by the examined GDA tools

	Subject			Method	
Tool	Topology	Assignment	Objective	Scope	Type
CELLO	-[a]	✓	CELLO score	global	heuristic
iBIOSIM	-	✓	# of base pairs	local	heuristic
MATCHMAKER	-	✓	signal compat.	global	exhaustive
SBROME	-	✓	functional error	local	heuristic
PARTS&POOLS	-[a]	✓	-	-	one-shot
GENETECH	-[a]	✓	-	-	enumeration

[a] One topology is generated in the mapping process, optimized with respect to the size instead of the main objective.

from the number of base pairs needed to implement the design to global scores for circuit functionality like the CELLO score. Lastly, the optimization methods applied either have a local scope limited to modules of the circuit or a circuit-wide global scope and are either heuristic or exhaustive. Table 6.2 contrasts the optimization approaches with each other.

MatchMaker The tool MATCHMAKER [71] transforms a network of abstract biological parts, i.e., an abstract GRN, into a specific implementation on a DNA strand. Thus, it does not cover the first step of technology mapping, the structuralization of a functional specification. Mapping to the abstract network is broken into three phases. In the feature matching phase, regulatory parts are mapped to the network in a structural mapping approach similar to iBIOSIM. Signal matching ensures that the quantitative input and output behavior of subsequent parts is compatible, similar to CELLO's signal compatibility criterion. MATCHMAKER offers a basic exhaustive optimization step to minimize the worst-case pair-wise noise margin present in the network. However, the tool does not have a notion of the desired global circuit functionality. In the final phase of part matching, the design is linearized to map it on a DNA strand.

SBROME Similarly, SBROME [31] expects the user to enter the topology of the circuit in the form of a GRN. However, the topology may consist of components of a higher abstraction level, such as elementary logic functions. In a structural mapping approach, SBROME matches modules of genetic parts from a library with the components of the input network. The used library consists of modules gathered from prior publications. After matching, the tool optimizes the modules locally by replacing genetic parts with similar mutants. It can be seen as a predecessor to Cello, operating on the lower abstraction level of parts instead of gates, lacking a uniform library of gates specified in the lab, and performing no global optimization of the circuit.

Parts&Pools Parts&Pools [43] features a technology mapping process transforming a truth table specification into a network of genetic parts. It generates a set of two-level logic implementations from the truth table and rates them with a score for biological complexity. The simplest network is then mapped and rated with a metric similar to the Cello score to predict circuit performance. However, no optimization is performed to improve the circuit characteristic, making Parts&Pools' approach a one-shot mapping.

GeneTech The tool GeneTech [6] is based on Cello's library of genetic gates. It takes a two-level Boolean expression in sum-of-products form, performs heuristic logic minimization, and transforms the expression to match the NOT and NOR logic functions present in the gate library. In contrast to Cello, GeneTech enumerates and outputs all possible gate assignments to the structure of the function. The assignments are not rated, and thus, no optimization of the circuit is performed.

Domain Adaptation

7. Structural Variants of Genetic Logic Circuits

In technology mapping schemes, the resulting network topology typically crystallizes in two steps of the mapping process. Firstly, the minimized Boolean formulae are transformed into a subject graph, defining the basic structure of the network. Secondly, the topology is altered locally by covering sub-graphs of the subject graph with library elements. In theory, the topology represents a degree of freedom useful to optimize a design for a given metric. In practice, however, the design space is limited by the structural bias introduced by the subject graph. If the main objectives of the mapping process are correlated to the structure of the subject graph, the bias can partially be mitigated by optimizing the subject graph accordingly. This is the case in EDA, where the depth and size of the graph provide reasonable estimations of the depth and size of the resulting network. In GDA, however, complex metrics for functionality are the main objectives of technology mapping, which do not correlate directly to features of the subject graph. Irrespective of this difference, existing approaches in the field either copy approaches of EDA and synthesize a subject graph minimal in depth or size or leave the specification of the network structure to the user. In this chapter, structural variants are proposed to leverage network topology as a degree of freedom in the technology mapping for genetic logic circuits. The approach is presented in Section 7.1, followed by details about the implementation (Section 7.2). Finally, the method is evaluated using an existing gate library to enumerate and score structural variants (Section 7.3).

7.1. Broadening the Search Space

Compared to electronic circuits, genetic circuits are strongly limited in size. Firstly, this is mainly caused by the fact that libraries of genetic gates contain a relatively small number of individual gates. The characteristic of every gate has to be specified in the lab, as well as its interactions with other gates in the library and the host organism. Thus, numerous time- and cost-intensive experiments are needed for library specification. Further, finding large

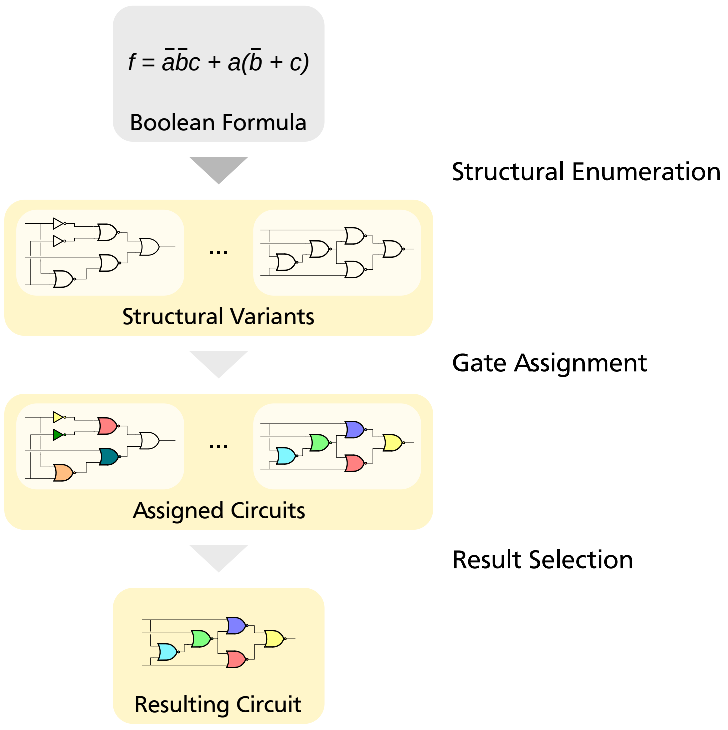


Figure 7.1.: Modified technology mapping process with structural variants

sets of gates with sufficiently orthogonal functionality is hard. During the specification of CELLO's gate library, for example, the number of TFs had to be reduced from 16 to 12 to "[. . .] eliminate toxic or cross-reacting repressors [. . .]" [53]. Secondly, circuit size is limited by the burden each synthetic part represents on the cell's metabolism (Section 4.1).

This limited size enables the enumeration of circuit variants implementing the functional specification. A related approach has been proposed in the context of logic synthesis for electronic circuits [41]. However, the presented enumeration is limited to circuit structures minimal in size and based on primitive gates to serve as technology-independent subject graphs. In the approach presented in this work, all network structures that adhere to the given synthesis constraints and that are implementable with the given gate library are enumerated and matched globally with the input function. Thus, subject graph generation, local matching, and covering with library elements are replaced. The result is the set of structural circuit variants implementing the function. The set contains minimal circuit structures but is not limited to them. It thus allows exploring the search space with respect to different objective functions. In theory, the total number of variants is only limited by the size of the library. In practice, however, the enumeration may be guided by sensible constraints like the maximum number of excess gates compared to the circuit with minimal size. The approach eliminates the structural bias of typical mapping methods by not relying on a technology-independent subject graph. Figure 7.1 depicts the modified technology mapping process. A consequence of taking structural variants into account is that the subsequent gate assignment step has to be carried out for each candidate structure.

7.2. Enumeration of Structural Variants

Note: Parts of this section have already been published in [58]. Explicit self-citation is omitted to provide an improved reading flow.

Generally, enumerating all combinational circuit structures implementable with a given gate library is a DAG-enumeration problem. A main task when enumerating circuits is checking for *isomorphism* among candidates to prune the search space during enumeration and guarantee the resulting circuits' uniqueness. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if there exists a bijection $\rho : V_1 \rightarrow V_2$ such that $(u, v) \in E_1$ if and only if $(\rho(u), \rho(v)) \in E_2$ [32]. Isomorphism of these graphs is noted $G_1 \simeq G_2$. To simplify isomorphism checking and thus enumeration, the intermediate representation of *in-trees* is chosen. An in-tree is a graph with a root node and exactly one directed path from any other node to the root node [45]. Here, the root node represents the output gate of the circuit. For the tree to represent a logic circuit, the nodes of the in-tree are labeled with a gate type $s \in \mathcal{S}$ from the set of types available in the library. This representation corresponds to a circuit structure without *fan-out*. In such a fan-out-free circuit, every gate is connected to exactly one subsequent gate. Generally, this structure is biased towards the functional redundancy of gates. It can be eliminated in a post-processing step that identifies the redundancies and merges corresponding gates into gates with fan-out. Thus, the proposed enumeration method comprises two phases. In the first phase, a set of unique labeled in-trees is enumerated based on the gate library. In the second phase, each tree is matched with the Boolean functional specification, redundant gates are removed, and final constraint and isomorphism checks are performed.

Tree Enumeration Algorithm 1 details the recursive enumeration routine for in-trees. It is called with an empty tree $\gamma \leftarrow \varnothing$ as the start element and an empty set $C \leftarrow \varnothing$ of

```
input : A circuit $\gamma$, gate types $\mathcal{S}$, maximum circuit weight $\omega$ and height $\eta$, the
        required support size $n$
inout : A a set of fan-out free circuits $C$
1  Function enumerate($\gamma, \mathcal{S}, \omega, \eta, n, C$)
2      new $h \leftarrow$ height($\gamma$);
3      new $I_\gamma \leftarrow$ get_unconnected_inputs($\gamma$)
       /* Abort criterion                                                      */
4      if $h > \eta$ then
5          return;
6      end if
       /* Iterate gate permutations that match the number of open inputs       */
7      foreach $r \in R \subset \{\mathcal{S}, \varnothing\}! : |R| = $ max($|I_\gamma|, 1$) do
           /* Build $\gamma'$ from $\gamma$ and $r$                                */
8          new $\gamma' \leftarrow \gamma \cup r$;
9          new $I_{\gamma'} \leftarrow$ get_unconnected_inputs($\gamma'$);
           /* Prune circuits that are too big                                  */
10         if weight($\gamma'$) $> \omega$ then
11             return;
12         end if
           /* Check, if $\gamma'$ supports $\phi$ and prune isomorph circuits      */
13         if $|I_{\gamma'}| \geq n \wedge \neg\exists\gamma'' \in C : \gamma' \simeq \gamma''$ then
14             $C \leftarrow C \cup \gamma'$;
15         end if
           /* Recurse                                                          */
16         enumerate($\gamma', \mathcal{S}, \omega, \eta, n, C$)
17     end foreach
```

Algorithm 1 : Recursive enumeration routine

circuit structures. Further, the gate types $\mathcal{S}$ available in the library, the support size n of the function $\phi : \mathbb{B}^n \rightarrow \mathbb{B}$ and the user constraints maximum circuit weight ω and height η are given. In this work, the metric for circuit weight used in [53] is applied, assigning each gate a weight of 1. The maximum height serves as the abort criterion (Line 4). In each iteration of the main loop (Line 7), a new circuit γ' is built by adding a new level to

γ and assigning it with another permutation of gate types. The permutations R are drawn from the set of gate types and a terminal element indicating the end of a branch, i.e., $R \subset \{\mathcal{S}, \varnothing\}!$, such that the number of elements equals the number of unconnected circuit inputs. Three checks are performed on each generated circuit in the domain of in-trees. First, the weight constraint is applied (Line 10). Second, it is checked whether the number of circuit inputs is equal or greater than the support size of ϕ, a necessary condition for γ' to implement ϕ (Line 13). Third, the isomorphism of γ' with other candidate circuits is checked (Line 13, detailed below). If all checks pass, the circuit is added to the set of candidate circuits and serves as the basis for another recursion. The result of the routine is the set of unique fan-out free circuits that are built of gate types $\mathcal{S}$, adhere to the constraints ω and η, and possess enough inputs to potentially implement ϕ. Note that the circuits do not yet feature primary inputs, and thus, their function is not determined.

Isomorphism Checking Using rooted trees as an intermediate representation for circuit structures allows for efficient checking of isomorphism among candidates, a core operation during enumeration. The Aho, Hopcroft and Ullman (AHU) algorithm can be used to determine isomorphism among unlabeled trees [1]. It features a linear time complexity with respect to the number of nodes m, i.e., $\mathcal{O}(m)$. To check isomorphism among unordered, labeled trees such as the circuit candidates, a modification of the algorithm has been proposed [32]. The modified version features a complexity of $\mathcal{O}(md\log(d))$ with d being the maximum out-degree of the tree. The out-degree equals to the maximum fan-in of the gate library. For the library considered throughout this work, $d = 2$ and thus $\mathcal{O}(2\log(2)m)$.

The algorithm is exemplified in the following based on two circuit candidate trees (Figure 7.2). It processes one level at a time, starting from the lowest level. For each node, a list is built (shown in blue) based on which the equivalence class of the node will be determined. The list contains the node label, followed by a sorted list of the child nodes' equivalence classes. In the case of leaf nodes, no children are present, and instead, the integer value of 0 is added to the list. Each unique equivalence class is assigned an integer value in ascending order (shown in red). In the example, all nodes on level 0 are leaf nodes with the label NOR. Thus, all nodes on this level feature the same list $[\mathrm{NOR}, 0]$ that is assigned the equivalence class 0. On level 1, both trees feature one parent node, each with equal equivalence classes of their children. Having different labels, however, their lists differ in the first position, leading to different equivalence classes. Since the other nodes on this level are equal among both trees, the combined equivalent classes at this level are different ($(1,2)$ vs. $(2,3)$). An implementation of the algorithm would abort after examining level 1 since isomorphism between the trees has been falsified. An

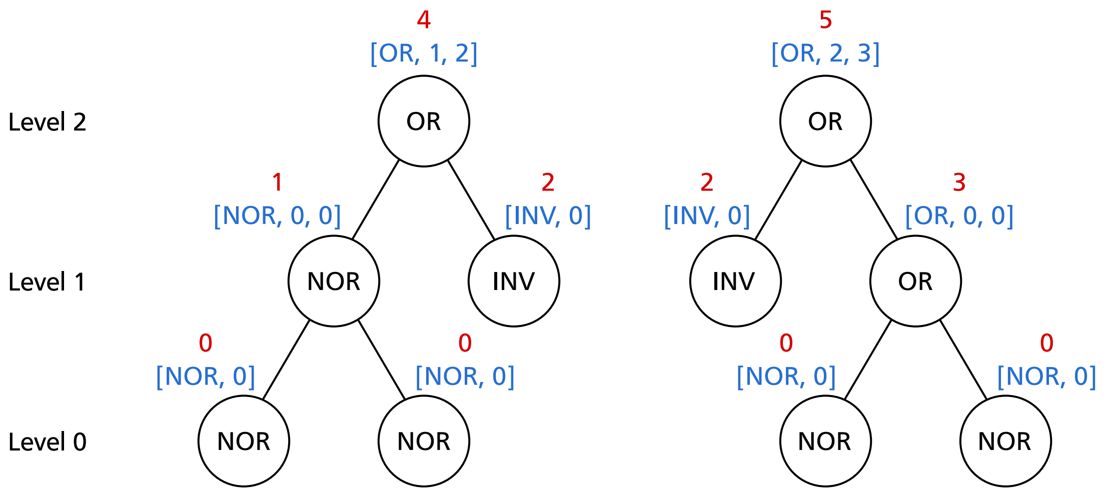


Figure 7.2.: Determining isomorphism of labeled trees

exemplary evaluation of level 2 shows that the different equivalence classes propagate and lead to different classes of the root nodes. Two rooted trees are proven to be isomorphic if their root nodes feature the same equivalence class.

Finalization The final steps of the enumeration are the functional evaluation of each circuit candidate, the removal of redundancies introduced by the tree structure, and the final checking of constraints. By this, the set of incomplete fan-out free circuits is transformed into a set of unique, irredundant implementations of the input function ϕ. This process is detailed in Algorithm 2. First, the set of primary circuit inputs $\mathcal{P} = \{p_0, \ldots, p_{n-1}\}$ according to support size n is instantiated and all possible mappings from primary inputs to unconnected circuit inputs I, i.e., $\mathcal{P} \times I$, are iterated for every circuit $\gamma \in C$ (Line 3). The circuit is wired according to the input mapping, its function is evaluated and matched with ϕ (Line 5). If the circuit is not discarded, redundancy among gates is determined by evaluating their function $f \in \mathbb{B}$ with respect to the primary inputs and comparing them pairwise (Line 8). If two gates are unique and implement the same function, one of the gates is substituted by a fan-out of the other. This process is repeated until all redundant gates have been substituted. The circuit graph is then in its final form, allowing for concluding checks. First, it is ensured that the number of gates of each type $s \in \mathcal{S}$ is not greater than the number of individual genetic gates of type s available in the library (Line 16). Further, the isomorphism of the final circuit with circuits from the set of implementations C_ϕ is checked, and the final dynamic weight constraint e is applied. It specifies the number of allowed excess gates compared to the minimum circuit found. Not shown here, the set C_ϕ has to be updated when a new minimum solution with weight ω_{min} has been found by removing circuits with a weight greater than $\omega_{\text{min}} + e$. If the circuit passes all checks, it is added to the set C_ϕ and ω_{min} is updated.

input : A set of fan-out free circuits C, a gate library $\mathcal{L}$ containing gate types $\mathcal{S}$, a Boolean function specification ϕ, the maximum number of excess gates e
output : A set C_ϕ of circuits implementing ϕ covered by $\mathcal{L}$

```
Function evaluate($C, \mathcal{L}, \phi, e$)
    new $C_\phi \leftarrow \varnothing$, $\gamma_m \leftarrow \varnothing$, $b \in \mathbb{B}$, $\omega_m$, $\omega_{\min} \leftarrow \infty$;
    /* Wire combinations of primary inputs $\mathcal{P}$ and circuit inputs $I$ */
    foreach $\gamma \in C, m \in \mathcal{P} \times I$ do
        $\gamma_m \leftarrow$ wire_inputs($\gamma, m$);
        /* Evaluate circuit and match with target function */
        if $\neg(\gamma_m \dashv \phi)$ then
            continue;
        end if
        /* Remove redundancies by functional comparison of gates */
        foreach $v \in V(\gamma_m)$ do
            foreach $u \in V(\gamma_m)$ do
                if $v \neq u \wedge f(v) = f(u)$ then
                    substitute_gate($v, u$)
                end if
            end foreach
        end foreach
        /* Check library feasibility, isomorphism and circuit weight */
        $b \leftarrow$ true;
        foreach $s \in \mathcal{S}$ do
            if $|\{v \in V(\gamma_m) : s_v = s\}| > |\{\theta \in \mathcal{L} : s_\theta = s\}|$ then
                $b \leftarrow$ false;
                break;
            end if
        end foreach
        $\omega_m \leftarrow$ weight($\gamma_m$);
        if $b \wedge \neg\exists\gamma' \in C_\phi : \gamma_m \simeq \gamma' \wedge \omega_m \leq \omega_{\min} + e$ then
            $C_\phi \leftarrow C_\phi \cup \gamma_m$;
            $\omega_{\min} = \min(\omega_{\min}, \omega_m)$;
        end if
    end foreach
    return $C_\phi$;
```

Algorithm 2 : Circuit building and evaluation routine

7.3. Evaluation

Note: Parts of this section have already been published in [58]. Explicit self-citation is omitted to provide an improved reading flow.

The enumeration of structural variants has been integrated into the GDA tool ARCTIC developed in collaboration with the Self-Organizing Systems Lab (Section 4.5). This allows to evaluate the impact of structural variants in genetic circuit synthesis. In the following, the conducted evaluation method and chosen parameters are detailed. Then, the results of circuit enumeration (Section 7.3.2) and its impact on the quality of mapped circuits (Section 7.3.3) are presented.

7.3.1. Setup

The three main parameters that need to be defined for a meaningful evaluation of the presented method are the library of genetic gates to use, the Boolean functions to map, and the set of synthesis parameters to constrain the enumeration. Further, a performance metric has to be chosen to examine the impact on circuit performance. The library used throughout this work is that published with the GDA tool CELLO for *E. coli* (Section 6.1) [53]. It represents one of the most extensive libraries of orthogonal logic gates currently available that have been specified and tested in the lab. Thus, it provides a good basis for evaluating technology mapping methods through simulation. Further, it allows comparing results to those of the state-of-the-art tool CELLO, provided that surrounding conditions are chosen fairly. As benchmark functions, the set of 33 3-input functions detailed in [53] is selected. This set is called Φ in the following. Again, circuits synthesized for these functions have previously been built and measured *in vivo*, indicating their practical feasibility and relevance. The functions $\phi_n \in \Phi$ with $n \in [1..33]$ are defined in Table A.1 using their truth tables. To constrain the enumeration sensibly, the maximum height η, maximum weight ω, and the allowed number of excess gates e must be chosen. Enumeration runs with different values for η showed that the set of functions is implementable with the given library with $\eta \geq 4$. Thus, $\eta = 5$ is chosen for the evaluation to broaden the search space while maintaining a manageable enumeration performance. The number of maximum excess gates $e = 2$ is chosen to enable the examination of the trade-off between circuit performance and circuit size. The maximum weight of any enumerated circuit $\omega_{\max}$ results from the minimum weights $\omega_{\min}$ among the set of functions and e, i.e.,

$$\omega_{\max} = \max_{\phi \in \Phi}(\omega_{\min,\phi} + e).$$

For the given library and functions, $\omega_{\max} = 9$. Based on this, a maximum circuit weight of $\omega = 12$ is chosen as a constraint, allowing fan-out to be introduced after the tree enumeration phase. Note that the output OR gate contained in the considered library is implemented by combining two output promoters preceding the coding sequence of the output reporter. Thus, no additional genetic part is needed for implementing this gate, and it is thus not counted when calculating the weight of the circuits in the following results.

To evaluate the impact of structural variants on circuit performance, the step of gate assignment has to be carried out, optimizing for a performance metric. Being state-of-the-art and allowing for a direct comparison of the approach with CELLO, the CELLO score (Section 4.2) is used as metric. To eliminate the influence of CELLO's stochastic gate assignment method, all circuit structures, including the ones generated by CELLO, are assigned using exact search methods. First, exhaustive search has been used, which has then been replaced by the exact branch-and-bound approach presented in Section 8.3 once it was available. The exactness of the gate assignments guarantees that the differences in the observed circuit scores are exclusively caused by the different circuit topologies and thus allows assessing the proposed approach independently of gate assignment quality. Additionally, CELLO's fixed assignment of input promoters to circuit inputs is adapted in this work (A: `pTac`, B: `pTet`, C: `pBAD`). Further, a fair comparison to CELLO's approach of circuit topology synthesis is ensured by calculating the toxicity of all generated circuit designs and rejecting designs that exceed the threshold applied in CELLO. Additionally, gate assignments are rejected containing specific pairs of promoters as inputs to a gate that are considered incompatible by CELLO.

7.3.2. Results of the Enumeration

Table 7.1 shows the enumeration results for the 33 Boolean functions. For each function ϕ, the minimum encountered circuit weight $\omega_{\min}$, and the number of circuit variants with the minimum weight (min.), one excess gate (+ 1) and two excess gates (+ 2) is listed. The minimum weight of the implementations $\omega_{\min}$ ranges from 2 to 7 gates. The number of structural variants differs greatly between the functions. The minimum number of 7 variants has been found for function ϕ_{13}. The maximum number of 640 variants has been enumerated for function ϕ_{16}. Collectively, a mean number of $\sim$217 variants has been synthesized per function. The minimum-weight variants are of special interest since they render possible choosing between topologies in technology mapping, even if circuit weight represents a primary objective. For 26 of the 33 functions, multiple circuits with minimum weight have been found. The maximum number of minimal variants has been generated for functions ϕ_7 and ϕ_{27}, amounting to 28 different circuit topologies. Per function, a mean number of 6.8 minimal variants, 31.2 variants with one excess gate, and 179 variants

Table 7.1.: Number of structural variants for the 33 functions with minimum circuit size, one and two excess gates.

		Variants		
Function	ω_{min}	min.	$+1$	$+2$
ϕ_1	3	2	0	6
ϕ_2	4	11	14	201
ϕ_3	3	1	6	25
ϕ_4	5	2	12	118
ϕ_5	6	13	33	226
ϕ_6	3	2	2	25
ϕ_7	5	28	80	475
ϕ_8	4	1	12	59
ϕ_9	5	2	20	96
ϕ_{10}	4	5	18	1289
ϕ_{11}	3	3	3	58
ϕ_{12}	5	4	16	132
ϕ_{13}	2	1	0	6
ϕ_{14}	5	1	15	58
ϕ_{15}	6	12	85	535
ϕ_{16}	7	22	108	510
ϕ_{17}	6	15	104	365
ϕ_{18}	7	3	77	311
ϕ_{19}	4	5	13	110
ϕ_{20}	7	22	106	501
ϕ_{21}	3	2	0	14
ϕ_{22}	5	1	16	56
ϕ_{23}	5	2	18	95
ϕ_{24}	4	1	11	57
ϕ_{25}	5	3	24	208
ϕ_{26}	3	3	6	75
ϕ_{27}	5	28	98	501
ϕ_{28}	4	3	9	86
ϕ_{29}	6	7	71	298
ϕ_{30}	6	4	20	228
ϕ_{31}	3	1	5	25
ϕ_{32}	4	11	25	251
ϕ_{33}	3	2	2	36

with two excess gates have been found.

On a standard desktop PC, a run time of $188\,\mathrm{min}$ was measured for the enumeration with $\eta = 5$, carried out in parallel with three threads. In comparison, runs with $\eta = 4$ and $\eta = 3$ featured run times of $292\,\mathrm{s}$ and $0.7\,\mathrm{s}$, respectively, emphasizing the combinatorial nature of the enumeration method.

7.3.3. Impact on Circuit Performance

After enumerating the structural variants, the optimal gate assignment with respect to the Cello score has been determined for every synthesized circuit structure and the structures synthesized by the tool Cello. Then, for each function, the highest scoring circuit from the set of minimal variants, from the set of all variants with up to one excess gate, and from the set of all variants with up to two excess gates has been selected. The results are visualized in Figure 7.3 and detailed in Table B.1. Considering only the set of minimal variants, the score of 15 of the 33 functions is improved compared to Cello's topologies, while none of the sets of variants performs worse than the structures synthesized by Cello. The circuits in this set of minimal variants feature the same weight as the topologies synthesized by Cello. A maximum $11.31\times$improvement in the score is achieved for function ϕ_4, while the mean score improves by $37.86\,\%$. Relaxing the considered circuit weight to include up to one excess gate, the circuit score for 17 of the 33 functions improves up to $11.37\times$, leading to an overall improvement of $38.82\,\%$ on average compared to Cello's topologies. Relaxing the weight by two excess gates, this trend continues with an improvement for 29 of 33 functions up to $18.91\times$and $181.09\,\%$ on average.

Figure 7.4 depicts exemplary results of the modified technology mapping process for the functions ϕ_{23} (Figure 7.4a) and ϕ_2 (Figure 7.4b). On the left, the topology synthesized by Cello for each function is given. The respective structural variant with the highest score from the set of minimal variants is shown on the right. The genetic gates assigned to the structures are coded in the colors of the gates, which are defined in Figure 7.4c. Below each circuit diagram, its simulated distribution of output values is depicted. The left histogram encompasses all output values representing a Boolean 0 output, and the right histogram all values for a Boolean 1. Potential overlap of the distributions is visualized by a darker color. Each of the variants features the same weight as the topology generated by Cello, but improves the score $8.36\times(\phi_{23})$ and $2.05\times(\phi_2)$, respectively. In both cases, the network height of the variants is increased by one compared to Cello's topologies. This hints at a limitation of Cello's approach of circuit synthesis, which is based on the EDA tool ABC, where network height is the primary optimization objective. In the case of the function ϕ_{23}, the increased height allows replacing the output OR gate with a NOR gate. The NOR gate provides a better separation of output states and, thus, a higher score.

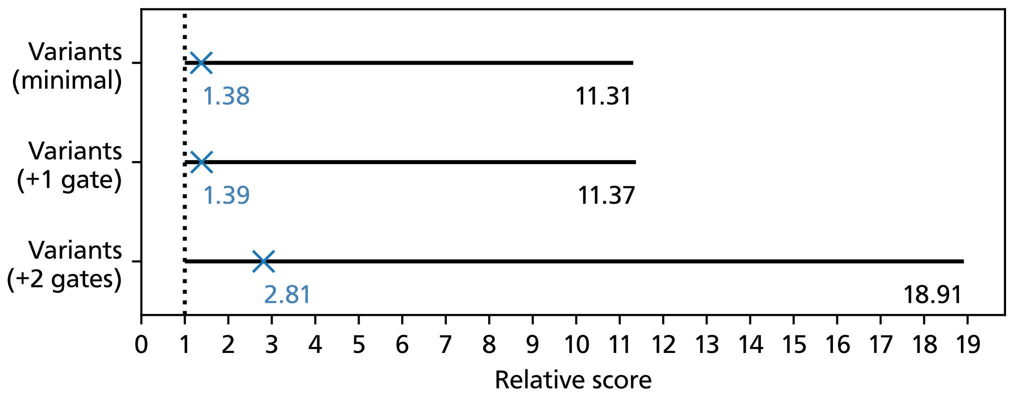
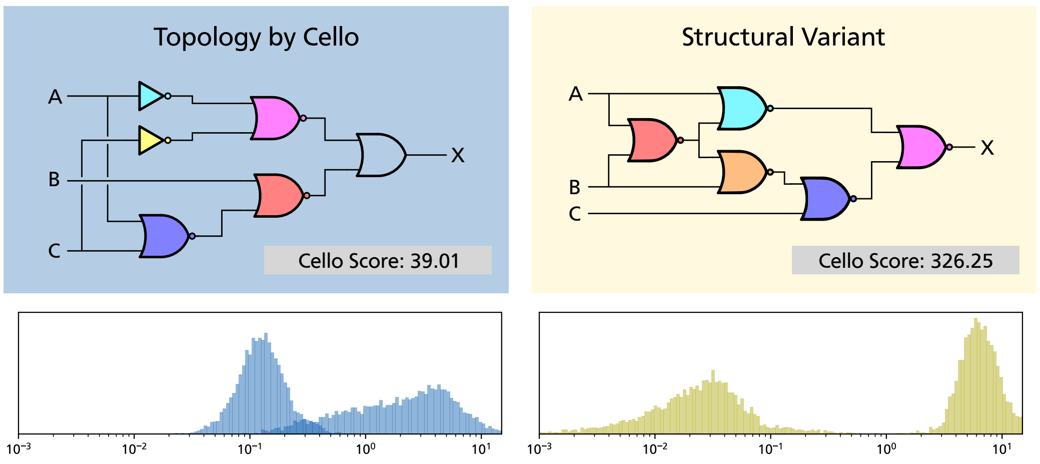
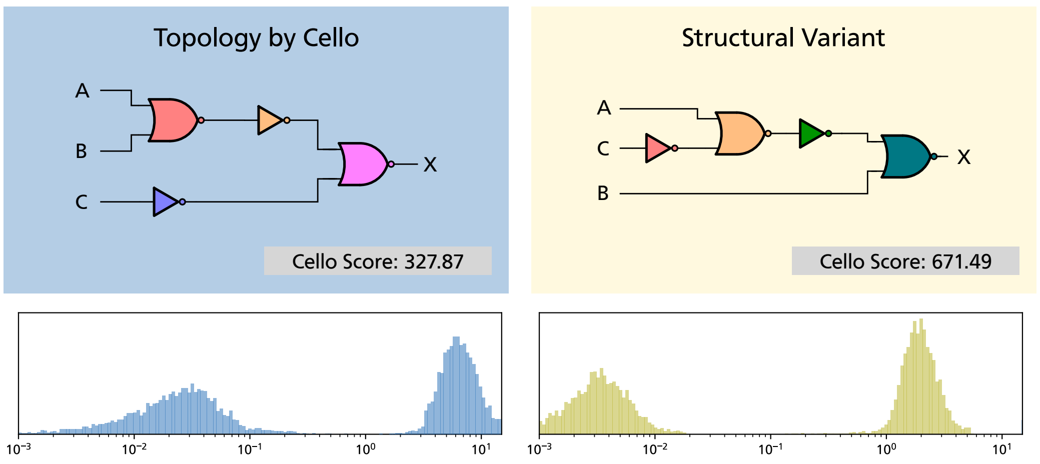


Figure 7.3.: CELLO scores of the sets of circuit variants relative to the topologies synthesized by CELLO. The blue crosses mark the relative mean score, the black bars the range of minimum and maximum relative scores.

This can also be seen in the histograms, where the median of the 0-distribution is lower, and the median of the 1-distribution is higher, comparing the variant to CELLO's topology. Further, the 1-distribution of the variant features a reduced variance, leading to a clear separation of both distributions. On the other hand, the variant's 0-distribution features a higher variance but is skewed towards low output values and thus does not provoke an overlap. In the case of function ϕ_2, both topologies are composed of the same gate types. Here, the higher score is caused by allowing the gate assignment optimization to draw a different set of specific gates from the limited library. The shapes and variances of the output histograms of both circuits are comparable, but the variant's distributions are moved towards lower output values. This displacement is higher for the 0-distribution, leading to an increased separation and score.



(a) Function ϕ_{23}



(b) Function ϕ_2

🟥 A1_AmtR 🟧 B3_BM3R1 🟨 E1_BetI 🟦 H1_HlyIIR 🟩 P1_PhlF
🟪 P3_PhlF 🟦 S2_SrpR 🟦 S3_SrpR ⬜ Output OR

(c) Gate legend

Figure 7.4.: Circuit topologies generated by CELLO and structural variants with optimal gate assignment and respective output characteristics

7.4. Discussion

The presented results show that the circuit topology represents an essential degree of freedom in the technology mapping for genetic logic circuits. Synthesizing the topology without regard to the objective function introduces a strong structural bias since it narrows the search for well-functioning combinations of specific gate implementations. Compared to EDA, the introduced bias has a greater impact, as standard objective functions used in GDA do not correlate directly to features of the subject graph. In the current state of GDA, the effect is aggravated by the gate libraries containing small numbers of gates with heterogeneous transfer functions. Existing technology mapping approaches for genetic circuits do not leverage the circuit topology by leaving its specification to the user or borrowing methods and tools from EDA. These tools optimize designs for objectives that are not necessarily suitable or primary for GDA, for example, circuit height. The presented enumeration based on CELLO's gate library showed that the design quality benefits from considering structural variants of circuits, even when minimality with respect to circuit weight is the main objective. In that case, evaluating variants of minimal topologies allows improving on secondary objectives, e.g., scores rating the circuit's functionality. If minimality is no strict requirement, the approach allows trading off circuit weight and other objectives by considering non-minimal designs.

The approach of exhaustive enumeration of circuit variants allows for finding the optimal topology for a given problem. It is feasible for the currently available small libraries of genetic gates and small circuits that are implementable *in vivo*. However, its strong combinatorial nature could be shown in the evaluation. Thus, the exact approach will become infeasible once the implementation of larger genetic circuits is possible or libraries gain functional variety. However, the enumeration core can serve as a basis for hybrid approaches, such as the *supergate* method proposed to reduce structural bias in EDA [18]. In this approach, small sub-circuits are built based on the gate library and are stored together with their function in a library of supergates. These building blocks can be mapped on the subject graph in a classical structural mapping scheme, reducing the locality of the substitutions and thus reducing the structural bias introduced by the subject graph. As this approach does not represent an exact method like the one presented, structural variants are not explored exhaustively, and the structural bias is not completely eliminated. However, it allows trading off exactness for computational feasibility if the size of the genetic circuit to map necessitates it.

8. Efficient Methods for Gate Assignment

After topology synthesis, gate assignment is the second phase in the technology mapping for genetic logic circuits. It has no analogy in EDA flows since it is necessitated by the libraries of heterogeneous, unique genetic logic gates used in GDA. During gate assignment, a mapping from the library's gates to the circuit topology's abstract gates is generated. Only in this crucial step does the circuit function crystallize, as the output characteristic of the circuit is defined by the interactions of the unique transfer characteristics of the selected gates. Objective functions quantitatively match the circuit characteristic to the desired function (Section 4.2) or cover additional features, e.g., robustness (Section 4.3). Thus, the gate assignment represents a combinatorial optimization process that is, in contrast to the topology synthesis, guided by an objective function.

The desirable characteristics of a gate assignment method are efficiency and (near) optimality. Here, efficiency refers to the number of gate assignments that must be evaluated until the process finishes in relation to the total number of possible different assignments. The efficiency is relevant to the run time of the design process since the evaluation of each assignment requires a circuit simulation, which represents the computationally most expensive part of the optimization. An efficient optimization enables the exploration of broad search spaces, such as the structural variants. It furthermore reduces the penalty introduced by complex simulation methods and thus contributes to a refined prediction of the circuit behavior. The designs generated during gate assignment should be (near) optimal since the implementation and testing of circuits in the lab are expensive. It is thus desired to enter the lab phase of a design process with a (near) optimal circuit design for the chosen objective.

In this chapter, gate assignment methods are proposed to optimize designs efficiently for the Cello score that relates the circuit function to the functional specification. First, the exhaustive search for the optimal gate assignment and the underlying combinatorics are detailed in Section 8.1. Due to its exploration of the whole space of assignments, it is of limited usefulness in a practical GDA process. However, it allows to produce exact reference values for the number of possible assignments and solution quality. In Section 8.2, a gate assignment method based on the general purpose optimization scheme Simulated Annealing (SA) is presented. It performs a heuristic search that is guided by a neighborhood structure based on the functional similarity of genetic logic gates. Finally, a method based on the Branch-and-Bound (B&B) approach is presented in Section 8.3 that features an exact and a heuristic mode. It is based on fundamental features of the transfer functions of genetic gates and performs an implicit enumeration of possible solutions, pruning parts of the search space containing solutions proven to be sub-optimal. In Section 8.4, specific aspects of the presented methods are evaluated in detail, followed by a comprehensive comparison of the methods in terms of efficiency, runtime, and solution quality.

8.1. Exhaustive Search

In the exhaustive approach for gate assignment optimization, all possible mappings from library gates to abstract gates of the circuit topology are enumerated. For each assignment, the objective function is evaluated by circuit simulation, and the assignment leading to the best objective function value is stored. By this, the exact solution to the problem is determined. The exhaustive search illustrates the combinatorial complexity of the problem. It does not represent a practical approach to the gate assignment problem, apart from mapping very small circuits. However, it is used in this work to generate a baseline for the comparison of more elaborate approaches for two reasons. First, it is guaranteed to deliver exact results for every choice of objective function. Thus, it serves as a basis to rate the solution quality of heuristic approaches and enables the empirical validation of exact approaches. Second, it explicitly traverses the whole space of possible gate assignments and with the number of visited assignments provides a baseline for comparing the efficiency of more elaborate methods.

Generally, the size of the set of all gate assignments traversed in an exhaustive search is defined as follows. Let $\mathcal{L}$ be the library of genetic logic gates containing gate types, i.e., available Boolean functions, $\mathcal{S}$. Let further n_s be the number of individual genetic gates of type $s \in \mathcal{S}$ in the library and k_s the number of abstract logic gates of type $s \in \mathcal{S}$ in the circuit topology γ. The number of possible mappings from the library to the circuit for each gate type is defined by the number of k-permutations. This is due to the fact that the circuit function is not only determined by the assigned combination of library gates but also by their permutation that defines which gate from the combination is mapped to which abstract gate of the circuit. Thus, the size of the set $\mathcal{A}$ of possible gate assignments for the whole circuit is defined as the product of the number of k-permutations for each gate type, i.e.,

$$|\mathcal{A}| = \prod_{s \in S} \binom{n_s}{k_s} k_s!. \tag{8.1}$$

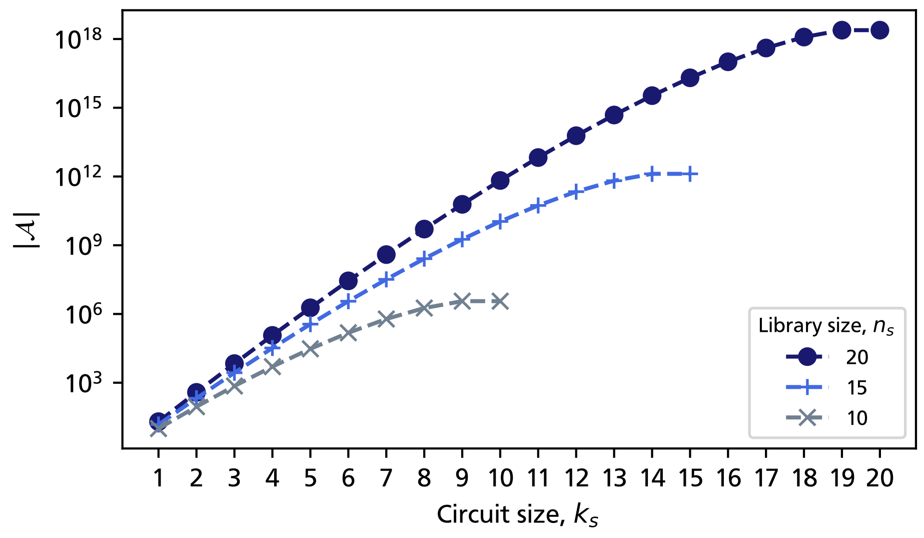


Figure 8.1.: Number of assignments for different library and circuit sizes

Figure 8.1 depicts $|\mathcal{A}|$ for circuit sizes ranging from 1 to 20 gates and libraries containing 10, 15 and 20 gates, respectively. Here, each gate from the libraries is assumed to be mappable to each gate in the circuit topology, i.e., all gates are of the same type s. Given a library of 20 gates, a circuit containing 8 gates can be assigned 5×10^9 k-permutations of individual genetic gates. Assuming the exemplary run time of one simulation to be $1\,\mathrm{ms}$, performing exhaustive gate assignment for that circuit sequentially would require $\sim 1400\,\mathrm{h}$. Note that specific libraries of genetic gates may have additional characteristics that influence the number of possible assignments. CELLO's library, for example, is based on 12 different TFs. However, some of these gates are present as variants with different RBS, amounting to a total of 20 gates [53]. Thus, the total number of assignments possible with this library exceeds the one corresponding to $n_s = 12$.

8.2. Proximity-Based Simulated Annealing

Note: Parts of this section have already been published in [58, 61]. Explicit self-citation is omitted to provide an improved reading flow.

Simulated Annealing (SA) is a general-purpose approach for combinatorial optimization. It is based on the analogy between the search for an optimum solution and the simulation of physical systems reaching a low energy state [56]. In theory, the approach itself is proven to determine the global optimum of a problem. However, a trade-off between efficiency and solution quality has to be found in practical applications of the method. Thus, typical instances of SA represent heuristic optimization methods. The approach has widely been used for solving circuit design problems in EDA. Even the first publication of the method by Kirkpatrick *et al.* [36] is focused on the partitioning, placement, and routing of digital circuits with SA. Since then, it has been used and adapted constantly, for example, for the placement of logic cells on FPGAs [8, 73] and the logic synthesis for memristive crossbars [65]. Further, a very basic instance of SA has been applied to the gate assignment problem in GDA [53].

In the remainder of this section, an instance of SA is presented to solve the gate assignment problem efficiently based on the functional proximity of the genetic logic gates in the library. In Section 8.2.1, the general approach and basic algorithmic framework of SA are explained in detail. Based on this, the general parameterization of the scheme for solving the gate assignment problem is detailed in Section 8.2.2. Finally, the proximity measure for genetic gates and its integration into the SA scheme, which results in a proximity-based neighbor generation, are presented in Section 8.2.3.

8.2.1. General Approach

The general idea of SA is based on an approach by Metropolis *et al.* [46] to simulate the behavior of a collection of atoms in condensed matter. In each step of the algorithm, a small change to the grid structure of the system is applied, e.g., the displacement of a single atom. Then, the change in energy ΔE compared to the previous state is computed. If the perturbation reduces the energy, i.e., $\Delta E \leq 0$, the changed state is accepted as a new system state. If the displacement leads to increased energy, i.e., $\Delta E > 0$, a random decision on the acceptance is made. Let T denote the current temperature of the system and k_B the Boltzmann constant, then the change leading to the increased energy is accepted with probability

$$P(\Delta E) = \exp(-\frac{\Delta E}{k_B T}). \tag{8.2}$$

The process is repeated iteratively with decreasing temperature to simulate the annealing process. If the number of perturbations per temperature level is sufficient to allow the system to reach equilibrium at each temperature, the system will eventually evolve into a Boltzmann distribution [36].

Generally, systems subject to combinatorial optimization do not feature a temperature, and objectives other than the energy level are prevalent. Thus, these physical properties are abstracted in applications of SA for optimization. While the term temperature is kept in typical SA implementations, it represents a general parameter for controlling the optimization process not necessarily rooted in a physical property of the system. The energy is typically replaced by the respective objective function. However, essential properties of the Metropolis method can be found in all instances of SA. First, the iterative nature of the method with gradual changes to the system in every iteration makes them methods of *iterative improvement*. Second, the probabilistic acceptance of higher energy states, i.e., solutions with worse objective function values, enables SA approaches to leave local minima and represents a significant advantage compared to simple hill-climbing methods. An instance of SA generally consists of two building blocks. One is the basic algorithmic structure that characterizes most instances, and the other is the set of functions and parameters setting the algorithm up for a specific problem, called *annealing schedule* [56].

```
Function simulated_annealing()
    $n \leftarrow 0, k \leftarrow 0$;
    $T_0 \leftarrow$ get_initial_temperature();
    $a_0 \leftarrow$ get_initial_solution();
    while outer-loop criterion is not satisfied do
        while inner-loop criterion is not satisfied do
            $a_j \leftarrow$ generate($a_k$);
            if accept($a_j$, $a_k$, $T_n$) then
                $a_{k+1} \leftarrow a_j$;
            else
                $a_{k+1} \leftarrow a_k$;
            end if
            $k = k + 1$;
        end while
        $T_{n+1} \leftarrow$ update($T_n$);
        $n \leftarrow n + 1$;
    end while
```

Algorithm 3 : Basic structure of the SA algorithm (adapted from [56])

The basic framework of SA, which is also used as a basis for the presented gate assignment method, is detailed in Algorithm 3. Here, $n \in \mathbb{N}$ and $k \in \mathbb{N}$ are counters for the outer and inner loops, respectively. T_n and a_k represent the temperature of outer loop iteration n and the current solution of inner loop iteration k, respectively. The current solution is a potentially sub-optimal but valid and complete assignment of library gates to the circuit from the set of possible assignments, i.e., $a_k \in \mathcal{A}$. At the beginning of the annealing, both T_n and a_k are initialized with an initial value corresponding to the annealing schedule. Every iteration of the outer loop (Line 5) represents one annealing phase with constant temperature, in which the system subject to optimization is sought to reach equilibrium. At the end of every outer loop iteration, the temperature value is updated according to the annealing schedule. In the inner loop (Line 6), a perturbed solution a_j is generated based on the current solution a_k. Solution a_j is called a *neighbor* of a_k, referring to the desired similarity between these two solutions. Then, the objective function value of a_j is evaluated and, based on an acceptance function, a_j is accepted as a_{k+1} or rejected (Line 8).

8.2.2. Annealing Schedule

The SA method is adapted to the gate assignment problem by choosing appropriate functions and parameters for the annealing schedule. In detail, the schedule consists of functions determining the initial solution and temperature, criteria for leaving each of the loops, the function for generating a neighboring solution, the acceptance function, and a function determining the update strategy for the temperature. The schedule elements are based on existing approaches that have been proven empirically to deliver good results and have been adapted to the gate assignment problem. The scheme represents a *dynamic* annealing schedule, as its parameters are derived from the specific optimization problem [56]. Compared to a *static* schedule, this allows optimizing a broader range of problems of, for example, different sizes. Note that the presented annealing schedule refers to a *maximization* problem, based on typical objective functions used in GDA like the CELLO score. However, the schedule can be easily adapted to a minimization problem.

Initial Solution The initial solution, i.e., the initial assignment of library gates to the circuit, is generated randomly [8]. This is implemented by starting with an empty assignment $a_0 \leftarrow \varnothing$ and subsequently drawing random gates from the library and assigning them to the circuit until it is fully assigned. An SA scheme that is well adapted to the problem enables leaving any local optimum, and thus the random starting point of the optimization is adequate.

Initial Temperature The initial temperature T_0 is determined based on the spectrum of objective function values of randomly generated solutions to allow for a high mobility of the search in the early phase. Denote by $s(\gamma, a)$ the objective function value for circuit topology γ and gate assignment $a \in \mathcal{A}$. Let further

$$Y = \{s(\gamma, a_0), s(\gamma, a_1), \dots, s(\gamma, a_{n-1})\}$$

be a sample of n objective function values based on randomly drawn assignments $a_0, \dots, a_{n-1}$. Then, similar to [8], the initial temperature is given by the scaled standard deviation of Y, i.e.,

$$T_0 = 20 * \mathrm{sd}[\mathrm{Y}].$$

By this, most of the perturbations of the solution are accepted at the beginning of the annealing process, even those leading to a clear worsening of the objective function value. Sample sizes of $n = 10^i, i \in 1..4$ have been tested and $n = 1000$ proved to provide stable temperature values. Thus, this sample size is chosen in the given implementation.

Inner-Loop Criterion The inner-loop criterion defines the number of iterations per temperature level and is chosen to enable the process to reach equilibrium at each level. It has been shown that this behavior can be reached by selecting a dynamic criterion that considers the problem size [64]. For the given problem, a good estimation for the complexity is given by (8.1), as it defines the design space size for a fixed circuit topology and gate library. Similar to [8], the problem size $|\mathcal{A}|$ is transformed into the number of iterations of the inner loop k_{max}. However, an adapted multiplier and exponent have been determined in a sweep to provide a good solution quality for all occurring problem sizes. The number of iterations

$$k_{\max} = 50 * \sqrt[4]{|\mathcal{A}|}$$

results in an inner-loop abort criterion of $k \geq k_{\max}$.

Outer-Loop Criterion The outer-loop criterion determines when the annealing process is terminated and the current best solution is returned as the overall solution. During the tuning of the parameters for the gate assignment problem, a combined criterion has proven to deliver good results. First, the *acceptance rate* is determined for each temperature level [8]. Let N_{acc} be the number of accepted perturbations at a temperature level, then

$$\alpha = \frac{N_{\text{acc}}}{k_{\max}} \tag{8.3}$$

denotes the acceptance rate. It provides a metric for the current state of the annealing process. High values of α signify a random walk characteristic, describing a process in which most perturbations are accepted. Low values, on the other hand, signify a hill-climbing characteristic coined by the rejection of perturbations leading to a worse objective function value. Generally, high-temperature values lead to high values of α, and thus α tends to decrease during the annealing process. The first part of the outer-loop criterion is defined as $\alpha < 0.2$, preventing the process from terminating in an early phase.

Additionally, a new criterion is devised to keep track of changes in the yielded objective function values throughout the annealing. Let $Z = \{s(\gamma, a_0), s(\gamma, a_1), \ldots, s(\gamma, a_{m-1})\}$ be the sample containing the last m objective function values. Let further $U \subset Z$ be the subset containing the $\frac{m}{10}$ highest values of Z and $s_{\max}$ the best currently known objective function value. Then, the second outer-loop criterion is defined as

$$\mathrm{sd}[U] < 0.001 * s_{\max},$$

providing a measure for the plasticity of the annealing with respect to the best solution encountered. The factor of 0.001 and the sample size of $m = 2000$ have been determined by a sweep similar to the initial temperature. They have been chosen to ensure that the solution quality of the SA is stable across multiple runs.

Both criteria are used in conjunction to form the outer-loop criterion.

Acceptance Function As an acceptance function, the commonly applied modified version of the Metropolis criterion (8.2) is used [56]. Let $\Delta s = s(\gamma, a_k) - s(\gamma, a_j)$ be the difference of objective function values of the current solution a_k and generated neighbor a_j. Then, at current temperature T_n, the neighbor is accepted as the new current solution with a probability

$$P(\Delta s) = \begin{cases} \exp(-\frac{\Delta s}{T_n}), & \text{if } \Delta s \geq 0 \\ 1, & \text{otherwise.} \end{cases}$$

The two cases are also distinguished in the code to avoid the unnecessary evaluation of the exponential function and to provide legal probability values.

Cooling Scheme In each iteration of the outer loop, the new temperature T_{n+1} is determined based on the current temperature T_n and the acceptance rate α (8.3). An adaptive scheme adapted from [8] is applied that allows the process to quickly cross temperature ranges characterized by very high and low acceptance rates.

$$T_{n+1} = \begin{cases} 0.8 \; * T_n & \text{if } \alpha > 0.9 \\ 0.99 * T_n & \text{if } \alpha > 0.2 \\ 0.9 \; * T_n, & \text{otherwise.} \end{cases}$$

The last component of the annealing schedule, the neighborhood generation, is explained in detail in the following section.

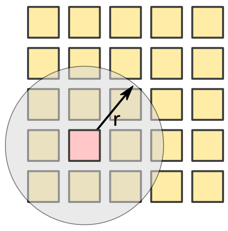


Figure 8.2.: Scheme of VPR's radius based placement of logic cells

8.2.3. Neighbor Generation

The neighbor generation is a crucial part of the annealing schedule. It is a function that takes the current solution a_k, performs a perturbation to it, and returns this perturbed solution as a new current solution a_j. By this, the neighbor generation determines how the heuristic search traverses the search space. If only marginal changes are performed, leading to a small number of solutions reachable, i.e., a small neighborhood size, the search tends to be stuck in local optima. Suppose the changes are too extensive, a_k and a_j become increasingly unrelated, leading to the characteristic of a random walk. It has been shown before that the size of the neighborhood affects the solution quality and efficiency of SA [29]. Generally, "[in] early applications [of metaheuristics], the impact of the neighborhood selection has been underestimated" [2]. Therefore, an adaptive neighbor generation strategy from the design automation for FPGAs is introduced in the following. Inspired by this, a strategy for the gate assignment problem based on the functional proximity of genetic gates is devised.

Proximity-based Neighbor Generation in FPGA Placement

The academic tool for the physical design for FPGAs Versatile Place-and-Route (VPR) uses an optimization scheme for the placement of logic cells on the chip based on SA [8]. In each iteration of the method, the position of one or two two cells is changed by, e.g., swapping their position. The neighborhood of a given placement thus consists of all the placements that are reachable by performing one swap of any of the cells. As an objective function, the routing cost of each placement is estimated. Generally, minor changes to the placement, i.e., swaps with a short distance between the involved cells, lead to small changes to the estimated routing cost. On the other hand, swaps with longer distances tend to have a bigger impact on the objective function value. It has been shown that for the placement problem, it is desirable to keep the acceptance rate α (8.3) at a value of 0.44 for large parts of the annealing process. In VPR, this is achieved by adaptively controlling the impact of perturbations allowed in the neighbor generation. As an additional control parameter, a radius of operation for performing swaps is introduced (Figure 8.2). The radius limits the distance at which cells are considered for a swap. At the beginning of the annealing, it covers the whole FPGA, allowing the search to cross large portions of the search space quickly. Towards the end of the optimization, it limits the swaps to directly neighboring cells. This reduces the evaluation of significantly worse solutions, which is beneficial for the scheme's efficiency since these solutions would be rejected in the late stage of the annealing with a high probability due to the low temperature. In each iteration of the outer loop, the new radius r_{n+1} is determined based on the previous radius and the acceptance rate of the previous iteration, i.e.,

$$r_{n+1} = r_n(1 - 0.44 + \alpha_n).$$

Functional Proximity of Genetic Gates

In contrast to the placement of logic cells, the gate assignment problem does not possess a geometrical dimension that may be leveraged to control the neighbor generation. However, an analogy can be drawn between the two problems, allowing for a transfer of the radius-based neighbor generation approach. As shown in Section 4.1, genetic gates in a circuit are built of individual genetic parts in favor of functional orthogonality. This results in an individual transfer characteristic for each gate, expressed by individual parameters for the Hill transfer function. Objective functions applied in genetic circuit design, e.g., the Cello score or the E-Score, relate the circuit function to the functional specification and thus are based on the circuit output characteristic. When optimizing a gate assignment in an iterative scheme swapping single gates, the impact of a swap on the circuit characteristic and the objective function value depends on the functional similarity between the involved gates. This is analog to the placement problem for logic cells, in which the *local* similarity between two swapped cells impacts the estimated routing cost. Thus, the possibility of controlling the neighbor generation can be transferred to the gate assignment problem by quantitatively determining the functional similarity between genetic gates. To this end, characteristic features of the transfer functions are presented in the following. These are then combined into a proximity-measure, allowing the control by a radius of operation.

For readability, the inhibitory Hill function introduced in Section 3.2 is repeated (8.4), with parameters $\theta = (y_{\min}, y_{\max}, K, n)$.

$$\mathrm{g}(x) = y_{\min} + \frac{(y_{\max} - y_{\min})K^n}{K^n + x^n} \tag{8.4}$$

The sigmoidal Hill transfer functions of genetic gates resemble the transfer curves of electronic buffers and inverters. A feature with impact on the characteristic of electronic

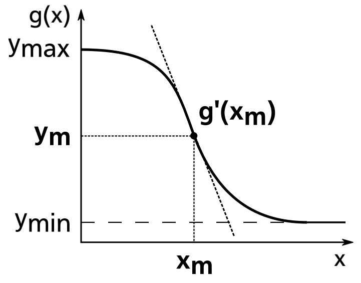


Figure 8.3.: Inhibitory Hill curve with characteristic features

inverters is the switching threshold voltage V_{m} where $V_{\mathrm{in}} = V_{\mathrm{out}}$ that can be found near the transfer curve's inflection point for well-built devices. Since there are no global signal levels in genetic circuits, the switching threshold is redefined for genetic gates to be the point on the Hill curve where the output signal value is halfway between the minimum and maximum output values, y_{min} and y_{max}. Based on (8.4) and exemplified for the inhibitory case, the output value at the switching threshold

$$d_1 := y_{\mathrm{m}} = \frac{y_{\mathrm{max}} + y_{\mathrm{min}}}{2}$$

and the corresponding input value

$$d_2 := x_{\mathrm{m}} = K\left(\frac{y_{\mathrm{max}} - y_{\mathrm{min}}}{y_{\mathrm{m}} - y_{\mathrm{min}}} - 1\right)^{\frac{1}{n}}$$

are defined as the first two characteristic features of the proximity-measure, d_1 and d_2. Further examination of parameterizations of genetic gates shows that they differ greatly in their gradient at the switching threshold, which is defined as the third feature

$$d_3 := \mathrm{g}'(x_{\mathrm{m}}).$$

Figure 8.3 depicts the three selected characteristic features for an exemplary parameterization of an inhibitory Hill curve. Similarly, the features can be derived from the Hill function describing activating interaction.

Proximity-Measure for Neighbor Generation

The characteristic features are combined into a proximity-measure to calculate the functional proximity of two gates during a swap operation in the SA scheme. Consequentially,

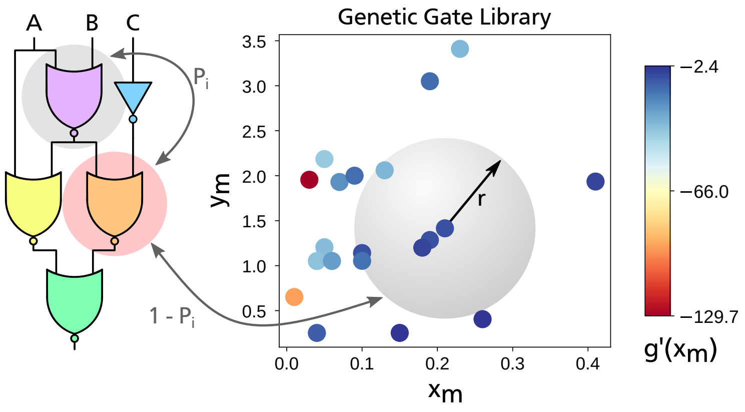


Figure 8.4.: Neighbor selection scheme based on proximity of genetic gates

for each gate q, the features are combined into the three-dimensional feature vector $\mathbf{d}_q := (d_{1,q}, d_{2,q}, d_{3,q})$. By this, the Euclidean distance between two gates in the feature space emerges naturally as a proximity-measure. Additionally, each dimension in the feature space is assigned an individual weight, which enables tuning the measure for a specific gate library. To this end, the diagonal weighting matrix $\mathbf{W} \in \mathbb{R}^{3\times3}$ with entries $W_{nm} = w_n/\delta_n$ for $n = m$ is defined. Here, $w_n \in [0,1]$ is the adjustable weight for feature n and δ_n the maximal absolute difference in the n-th feature between two gates across the whole library. Then the similarity between any two gates q and p in library can be quantified by the $\mathbf{W}$-norm

$$D_{qp} = ||\mathbf{d}_q - \mathbf{d}_p||^2_{\mathbf{W}}.$$

Section 8.4.1 details the tuning of the weight factors for a specific gate library.

The proximity-measure is integrated into the neighbor generation analog to VPR. The process is depicted in Figure 8.4. First, the abstract gate of the circuit subject to the perturbation is selected randomly, visualized by the red highlight. Then, it is randomly decided whether an in-circuit swap is performed, i.e., two gates currently assigned to the circuit are swapped, or an external swap is performed, in which the selected gate is swapped with a currently unused gate from the library. The probability of an in-circuit swap is given by $P_i = \frac{|a|}{|\mathcal{L}|}$, with $|a|$ the number of gates in the assignment and $|\mathcal{L}|$ the number of gates in the library. If such an in-circuit swap is performed, another gate is randomly selected from the circuit (indicated by the gray highlight), and the assignment of the two selected gates is swapped. For external swaps, the proximity-metric is applied by first calculating the proximity of the genetic gate currently assigned to the selected gate to the other gates in the library. In the visualization, the distribution of the gates of

CELLO's gate library in the feature space is depicted. All gates within the current radius r_n are candidates for the swap, from which one gate is drawn. At the beginning of the annealing, the radius is initialized with the maximum distance of any two gates in the library, allowing for maximum mobility of the search. In a scheme similar to VPR, but with modified dynamic and desired acceptance rate, the radius is updated according to the current acceptance rate α, i.e.,

$$r_{n+1} = \begin{cases} 0.95 * r_n & \text{if } \alpha > 0.4 \\ 1.05 * r_n, & \text{otherwise.} \end{cases}$$

To enable movement of the search in the late stage of the annealing with r_n possibly being very small, a minimum number of gates taken into account for a swap has to be set. In the given implementation, the 5 nearest gates are considered during neighbor generation irrespective of the current radius.

The SA scheme is evaluated in Section 8.4 together with the B&B method presented next.

8.3. Branch-and-Bound

Note: Parts of this section have already been published in [61]. Explicit self-citation is omitted to provide an improved reading flow.

When examining the gate assignment optimization for circuit functionality in detail, it becomes clear that the single gates exhibit a functional hierarchy. In the cascade of genetic gates representing a circuit, the last gate determines the maximum range of the circuit's output. This is caused by the interval of each gate being limited by its inherent basal and maximum promoter activities. From this, it follows that the choice of preceding gates in the circuit only determines which range of the last gate's output interval is used. Thus, a *constructive* optimization approach suggests itself, in which the gate assignment is successively built according to this inherent hierarchy. Algorithms based on the Branch-and-Bound (B&B) paradigm are well known for solving (mixed) integer programming as well as combinatorial optimization problems efficiently [20, 49]. While various specific algorithms adhere to the paradigm, their shared core is the *implicit* enumeration of the search space. In contrast to exhaustive search, parts of the search space are not explicitly evaluated. Instead, a *bounding function* or *estimator* is used to identify solutions inferior to the current best solution, which can, in turn, be ignored. If the estimator is strictly optimistic, B&B is guaranteed to be exact. However, finding an optimistic *and* realistic estimator, which minimizes the number of solutions evaluated, requires the incorporation of problem-specific knowledge. Thus, algorithms based on the B&B paradigm are typically highly problem-specific. In the following, a gate assignment method based on B&B is presented. Its estimator relies on fundamental features of genetic gates, selectively allowing for the generation of optimal or near-optimal gate assignments.

In Section 8.3.1, the general approach for mapping the gate assignment problem to a search tree traversable by the B&B scheme is presented. The successive construction of partial solutions is detailed, in which empty solutions are progressively extended to complete solutions. In Section 8.3.3, strategies for the traversal of the search tree that have been examined are presented. The problem-specific optimistic estimator for partial solutions is the core of the method, which is detailed in Section 8.3.4. First, desired properties of estimators are presented, followed by the general approach and a detailed examination showing that the estimator allows for an exact search in the B&B scheme.

8.3.1. Search Tree

To enable the application of the B&B method to the gate assignment problem, its search space is mapped to a tree. Generally, a search tree is a rooted tree in which every path from the root to a leaf node defines a complete solution. The nodes on one level of the tree represent the choices that can be made regarding a specific part of the solution. Together with the search strategy, the problem's mapping to the search tree determines how a B&B algorithm navigates the search space. For the gate assignment problem, the mapping emerges naturally from the inherent hierarchy of the problem. When circuit functionality is the main objective of the assignment optimization, it is beneficial to construct the gate assignment starting from the circuit's output gate. In this context, the root node of the search tree represents the empty gate assignment $a_{\varnothing}$. The circuit topology, given by a DAG, is then traversed in reverse topological order, i.e., an abstract gate is visited not before all subsequent gates have been visited. When a gate is visited, the search tree is expanded by one level, with nodes corresponding to the possible choices of genetic gates available in the library. That is, a *polytomic branching strategy* [20] is applied, corresponding naturally to the choices offered by the library. While a path from the root node to a leaf node determines a complete assignment $a \in \mathcal{A}$, a path from the root to an intermediate node determines a partial gate assignment with some unassigned gates in the circuit topology. In the following, such a partial assignment $a_p \subset a \in \mathcal{A}$ is called a *partial solution*. An essential feature of partial solutions generated in the presented, ordered way is that each partial solution represents a valid *sub-circuit*. Every partial solution is a cone of assigned gates starting from the output gate with no unassigned gates constituting gaps in the assignment. This is important for the estimation of the quality of partial solutions presented in Section 8.3.4.

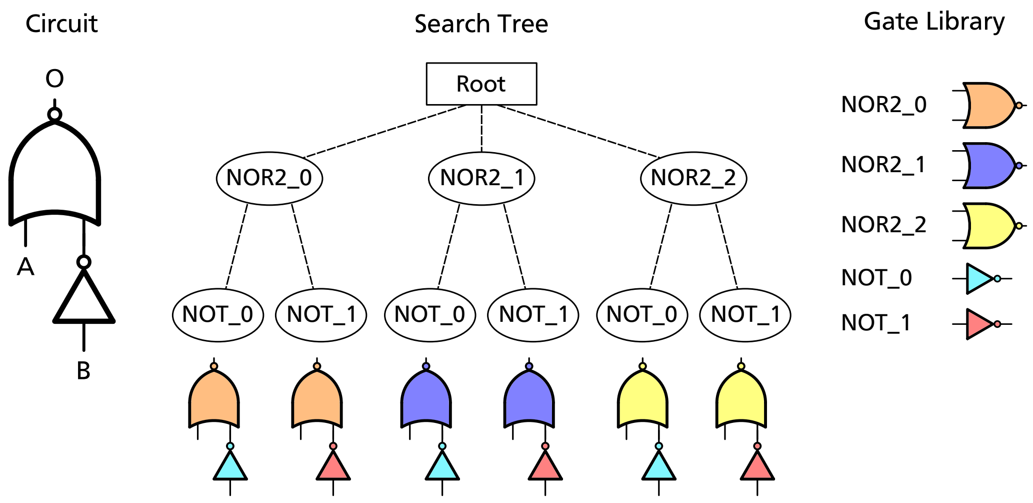


Figure 8.5.: B&B search tree for a given circuit and gate library

Figure 8.5 exemplifies the search tree for a simple circuit topology consisting of two gates and a library containing five individual genetic gates. First, the output NOR gate is visited, and the search tree is expanded by one level, starting from the root node. On this level, three nodes exist, each representing an assignment choice from one of the three NOR gates in the library to the output gate. Then, the remaining NOT gate in the circuit is visited since it is the next gate in the reverse topological order. Again, the tree is expanded by one level, containing nodes for the choices of NOT gates from the library. Below the leaf nodes, the complete assignments that result from following the corresponding path are depicted by color coding. It becomes clear that the leaf nodes exhaustively enumerate the search space. Thus, following each path from the root to the leaf nodes and evaluating the corresponding complete solutions results in an exhaustive search. The B&B paradigm allows leveraging the search tree's structure by building and traversing it only partly, resulting in an *implicit* enumeration (Section 8.3.2). The order in which the tree nodes are visited is determined by the *search strategy* (Section 8.3.3).

8.3.2. Implicit Enumeration

The algorithmic framework used for the guided search tree traversal and implicit enumeration is detailed in Algorithm 4. It represents a typical B&B scheme in which nodes to be potentially visited are either stored in a buffer or discarded, based on the comparison of their individual *upper bound* to the current global *lower bound* [20]. Note that the terms upper and lower bound refer to the maximization problem at hand. In this context, the lower bound $b_l \in \mathbb{R}$ is the best, i.e., highest, score value of any complete solution evaluated so far. The upper bound $b_u \in \mathbb{R}$, on the other hand, is an optimistic estimation of the score value that is reachable for a complete solution if the corresponding partial solution is part of it. Its determination is detailed in Section 8.3.4. Accounting for the given gate assignment problem, the buffer q holds elements (a, b_u) with partial or complete assignment a and its respective upper bound value b_u. The specific implementation of the buffer can be adapted to implement different search strategies (Section 8.3.3). For example, a first in - first out (FIFO) implementation results in a breadth-first search (BFS), while a stack implementation results in a depth-first search (DFS). Here, the functions `pop` and `add` are used to denote the retrieval and addition of elements, irrespective of the specific buffer implementation.

Algorithm At the beginning of the search process, the lower bound is initialized with an arbitrary negative number (Line 2), and the element $(a_{\varnothing}, 0)$ is added to the buffer (Line 4), representing the root node of the search tree. While there are elements in the buffer, one element at a time is retrieved (Line 6), its associated upper bound is compared to the current lower bound (Line 7) and the element is discarded if its upper bound does not exceed the lower bound. By this, the branching on nodes is avoided if an updated lower bound has surpassed their upper bound. For all other elements, the `branch` operation is performed on the retrieved assignment a_{cur} (Line 10). As explained in Section 8.3.1, the `branch` operation takes the next unassigned gate from the circuit topology in reversed topological order and generates a new set of assignments $a_1, \dots, a_n$. In each new assignment, one of n matching genetic gates left unassigned in the library is assigned to the selected circuit gate. For each assignment a_i with $i \in 1, \dots, n$, its individual upper bound b_u is calculated by applying the bounding function (Line 12). Only if the upper bound exceeds the current lower bound, a_i is processed further (Line 13). That is, partial solutions proven to be inferior to the current best solution are discarded by not adding them to the buffer. In other words, the sub-tree of the search tree springing from a_i is *pruned*. The further processing of undiscarded solutions depends on whether they represent complete or partial solutions. Complete solutions $a_i \in \mathcal{A}$ are saved as new best known solution a_{res}, and their upper bound is adopted as new lower bound (Line 14). This is possible because the bounding function value of complete solutions does not represent an estimation but an exact objective function value, i.e., score, of the complete assignment. Partial solutions $a_i \subset a \in \mathcal{A}$ are added to the buffer together with their upper bound. They could not be proven inferior, and thus, the search has to further traverse the search tree springing from a_i (Line 18). After the implicit enumeration is complete, a_{res} is

```
Function branch_and_bound()
    $b_u, b_l \leftarrow -\infty$;
    $a_{\text{cur}}, a_{\text{res}}$;
    $q \leftarrow \{(a_\varnothing, 0)\}$;
    while $q \neq \varnothing$ do
        $(a_{\text{cur}}, b_u) \leftarrow$ pop($q$);
        if $b_u \leq b_l$ then
            continue;
        end if
        $a_1, \ldots, a_n \leftarrow$ branch($a_{\text{cur}}$);
        for $i = 1, \ldots, n$ do
            $b_u \leftarrow$ bound($a_i$);
            if $b_u > b_l$ then
                /* Superior complete solutions are saved */
                if $a_i \in \mathcal{A}$ then
                    $b_l \leftarrow b_u$;
                    $a_{\text{res}} \leftarrow a_i$;
                /* Potentially superior partial solutions are added to $q$ */
                else
                    add($q$, $(a_i, b_u)$);
                end if
            end if
        end for
    end while
    return $a_{\text{res}}$;
```

Algorithm 4 : Implicit enumeration scheme of B&B (adapted from [20])

returned as the result of the search.

Eager and Lazy Branch-and-Bound The presented algorithm implements an *eager* B&B scheme, characterized by the fact that each (partial) solution resulting from a branching operation is immediately evaluated using the bounding function. The decision about adding the solution to the buffer is made based on the individual upper bound. In contrast, in a *lazy* B&B approach, the upper bound is only determined as soon as an assignment is subject to a branching operation [20]. Until then, it is annotated in the buffer with the upper bound of its parent node. While lazy B&B potentially avoids the evaluation of partial solutions that are discarded when the lower bound is updated, eager B&B provides more information in the buffer to implement search strategies based on the upper bound value. Both variants of B&B have been implemented in the presented method.

8.3.3. Search Strategies

The search strategy determines the order in which the nodes of the search three are visited in a B&B scheme. Thus, it impacts which node a branching operation is performed on next and which (partial) solutions are evaluated. Ideally, it quickly guides the search to a complete solution with a high objective function value, which can then be set as a new lower bound. Generally, the higher the lower bound and the earlier it is raised in the search process, the more parts of the search tree can be pruned. In the presented B&B scheme, the search strategies breadth-first search (BFS), depth-first search (DFS), best-first search (BeFS), and cyclic BeFS have been implemented and adapted to the gate assignment problem.

Breadth-First Search and Depth-First Search

BFS and DFS are common strategies for graph traversal. In BFS, a node is only visited after all nodes on the prior level have been visited. By this, BFS traverses the tree level by level from the root node until it reaches the leaf nodes. In the given buffer-based B&B scheme, a BFS behavior is accomplished by implementing the buffer as a FIFO. That is, the `add` operation adds new elements at the end of the buffer, while the `pop` operation retrieves the first element. In contrast, a DFS traverses each branch to the leaf node and then performs backtracking to traverse the next branch. In the B&B scheme, this is implemented by realizing the buffer as a stack. That is, the `add` operation adds new elements at the end of the buffer, and the `pop` operation retrieves the last element.

In these general instances of BFS and DFS, no other information than the hierarchy of the tree is taken into account. Thus, the exact order in which nodes are visited arbitrarily depends on the order in which the nodes resulting from a branching operation are added to the buffer. However, visiting nodes with a high associated upper bound is desirable since these will more likely lead to good complete solutions. Thus, an order is introduced in which nodes are added to the buffer based on their upper bound. This is accomplished by first determining the upper bound of all nodes resulting from a branching operation, sorting the set of nodes based on their upper bound, and adding the sorted set of nodes to the buffer. A descending order is applied in BFS, as the FIFO preserves the order of

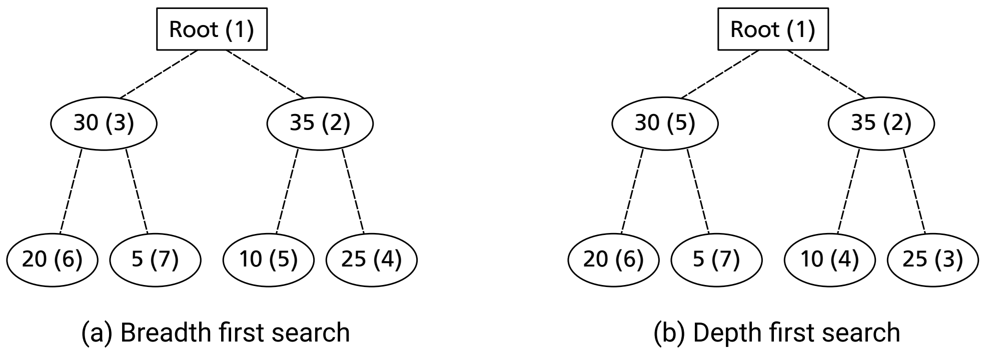


Figure 8.6.: Traversal order of BFS and DFS for an exemplary search tree

nodes. The stack used in DFS reverses the order of added nodes, and thus, added nodes are sorted in ascending order. Figure 8.6 depicts the traversal order for both strategies on an exemplary search tree. The nodes are annotated with their respective upper bound value. The order is denoted by the values given in parentheses. It becomes clear that BFS (Figure 8.6a) visits all nodes on one level before proceeding to the next. Further, in each set of nodes resulting from a branch, the nodes are visited in descending order of their upper bounds. DFS, on the other hand, traverses the rightmost branch to the leaf node before tracking back (Figure 8.6b). Again, the next branch to traverse is chosen based on the order of upper bound values.

In eager B&B, partial solutions are stored with their corresponding upper bound in the search tree. In lazy B&B, however, the upper bound of the parent node is transferred to the children during the branching operation. Thus, lazy B&B lacks individual upper bound values at the time nodes are added to the buffer. The following approximation is performed to allow the sorting of nodes before adding them to the buffer. Instead of the upper bound, the ratio $y_{\text{max}}/y_{\text{min}}$ of the genetic gate added to the assignment during the branching is used to sort the nodes. By this, solutions containing gates with a wide output interval are preferred during traversal, providing an estimation for the reachable CELLO score.

Best-First Search and Cyclic Best-First Search

BeFS [22] and its extension cyclic BeFS [50] deviate from the traditional graph traversal strategies by ignoring any hierarchical information given by the search tree. Instead, BeFS operates on a list with a global node order. Analog to BFS, this order is given by the descending upper bound values of the (partial) solutions for eager B&B and the ratio

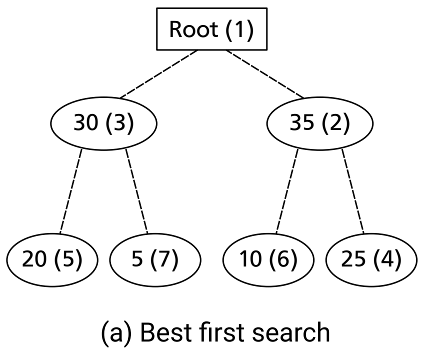


Figure 8.7.: Traversal order of BeFS and cyclic BeFS for an exemplary search tree

$y_{\max}/y_{\min}$ of added gates for lazy B&B. The behavior of BeFS is demonstrated in Figure 8.7a using the exemplary search tree introduced before. In contrast to BFS, BeFS can jump between levels of the tree and between sets of child nodes on the same level, exemplified by the order of visited leaf nodes.

Cyclic BeFS represents an extension of BeFS by maintaining an ordered list of nodes for each level in the search tree. Nodes are retrieved from the set of lists in a round-robin scheme, querying one list after the other in a cyclic manner. Figure 8.7b depicts an exemplary resulting traversal order. On each level, the nodes are visited in descending order of their upper bounds, corresponding to the ordered list implementation. The strategy jumps between the two levels in the tree according to the round-robin scheme until both nodes on the upper level have been visited. Then, the remaining nodes on the lower level are processed.

8.3.4. Optimistic Estimation

The bounding function, or estimator, represents the core of a B&B scheme. It provides upper bound values on the objective function for partial solutions and thus impacts the efficiency of the search and determines whether it delivers the optimum, i.e., exact, result. For exactness, the estimation has to be strictly optimistic, as this guarantees that only inferior branches of the search tree are pruned. On the other hand, the estimation of a partial solution has to be as close to the objective function value of the corresponding complete solution as possible, allowing for effective pruning. Thus, the estimator is highly problem-specific and has to consider the problem structure and objective function. In the following, the estimator used in the implemented B&B scheme is presented. It estimates the quality of partial assignments representing sub-circuits of the original problem with respect to the CELLO score. First, general requirements for the estimator are presented, followed by the approach taken to compose optimistic input intervals for sub-circuits. Then, it is shown that the CELLO score can be generally bound from above by considering the parameters of the circuit's output gate. This is generalized to cascades of single-input gates by showing that optimistic values assumed for the inputs propagate through the cascade. Finally, it is shown that this propagation does not hold for multi-input gates, and a solution for this limitation is presented, making the estimator strictly optimistic for general logic circuits.

Desired Properties

Let $\mathrm{s}_{\mathrm{Cello}}(\gamma, a) \in \mathbb{R}$ denote the CELLO score for circuit topology γ and gate assignment $a \in \mathcal{A}$. Then the heuristic $\mathrm{e}(\gamma, a_p) \in \mathbb{R}$ with partial or complete assignment $a_p \subseteq a$ is called an estimator, if

$$\mathrm{e}(\gamma, a_p) \geq \mathrm{s}_{\mathrm{Cello}}(\gamma, a) \ \forall \ a_p \subseteq a, a \in \mathcal{A}. \tag{8.5}$$

That is, the heuristic overestimates or equals the score of the complete assignment a when a partial assignment a_p is given that represents a subset of a. This optimistic estimation guarantees that only branches of the search tree are pruned, which lead to complete solutions proven to be inferior to the current best known solution determining the lower bound. Thus, it guarantees the exactness of the B&B scheme. Further, the estimator should deliver exact objective function values if it is given a complete solution a, i.e.,

$$\mathrm{e}(\gamma, a) = \mathrm{s}_{\mathrm{Cello}}(\gamma, a) \ \forall \ a \in \mathcal{A} \tag{8.6}$$

This enables the estimator's unified usage to rate partial and complete solutions, respectively. Finally, the criterion for optimistic estimation should not only be fulfilled for partial and complete solutions but also for partial solutions built on each other. Let $a_{p,2}$ denote a partial solution resulting from refining $a_{p,1}$. Then

$$\mathrm{e}(\gamma, a_{p,1}) \geq \mathrm{e}(\gamma, a_{p,2}) \ \forall \ a_{p,1} \subseteq a_{p,2} \subseteq a \in \mathcal{A}. \tag{8.7}$$

In the context of the search tree, this means that the upper bound values of partial solutions on a path from the root node to a leaf node decrease monotonically.

Approach

As shown in Section 8.3.1, the partial solutions subject of the estimation represent sub-circuits of the original problem. Due to the successive addition of genetic gates to the assignment in the reversed topological order of the circuit, these sub-circuits are cones

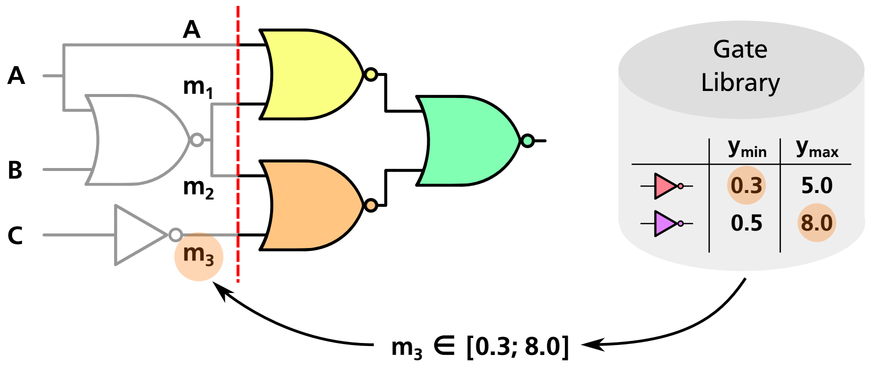


Figure 8.8.: Optimistic input interval composition approach for estimation

ending at the output gate that do not feature gaps. That is, every path from the inputs of the sub-circuit to the output traverses only logic gates that have been assigned specific genetic gates. That allows simulating the output values of the sub-circuits similar to completely specified circuits as introduced in Section 4.2. Figure 8.8 exemplifies the estimation of a partial solution and is referenced in the following to introduce the approach. The complete circuit topology consists of one NOT gate, four NOR gates, and three primary inputs: A, B, and C. The shown partial solution is an assignment to the three last gates in the circuit, depicted by their fill color. In the search tree, this partial solution could be found by traversing a path from the root node to the third level of the tree. First, the sub-circuit is constructed by discarding the unassigned gates of the original circuit, as depicted by the dashed red line. By this, a new set of inputs arises, which represented intermediate signals in the original circuit but constitutes primary inputs in the sub-circuit. In the given example, these new primary inputs are named A, m_1, m_2 and m_3. For the simulation of complete circuits, promoter activity values representing Boolean 0 and 1 are given for each primary input. For the newly introduced inputs, however, it is a non-trivial question which signal values to assume for the estimation.

The approach taken to determine the promoter activity values representing Boolean 0 and 1 for added inputs is called *optimistic input interval composition*. It is based on the fact that newly introduced inputs into the sub-circuit are sourced by other genetic gates in the complete circuit. As shown in Section 3.2, transcriptional genetic gates are characterized by a fixed output interval given by the basal and maximum promoter activity and specified in the parameters $y_{\min}$ and $y_{\max}$ of their Hill transfer curve. By considering the genetic gates left in the library, i.e., not assigned to the circuit, assumptions can be made about the minimum and maximum values to be expected at the sub-circuit's inputs. Figure 8.8

shows this exemplarily for input m_3. In the original circuit, this input is sourced by a NOT gate. To determine the interval of expected values, the widest possible output interval of matching gates left in the library is composed. In the example, two genetic NOT gates are applicable, and the values to be expected at m_3 are in the interval $[0.3, 8.0]$. Thus, 0.3 is a lower bound for values representing a Boolean 0 at m_3 and 8.0 is an upper bound for values representing a Boolean 1. When simulating the circuit output with respect to all Boolean input combinations as required to determine the CELLO score, these bounds can be used as input values. Caused by the monotonous Hill transfer characteristic of the genetic gates, they are propagated through the circuit. As a result, the simulated circuit output values also represent lower bounds for Boolean 0 and upper bounds for Boolean 1. If the CELLO score is calculated based on these values, it represents an optimistic estimation of the score reachable with this partial solution. This is shown in detail in the next section. Further, the propagation of optimistic input values through cascades of single-input and multi-input gates is detailed below.

When evaluating a sub-circuit, the feasibility of Boolean input combinations must be considered. In the given example, the original circuit features three primary inputs, while the sub-circuit features four inputs. Naturally, not all 16 combinations of Boolean values for the four inputs of the sub-circuit can be evoked by the 8 combinations of the original circuit. More specifically, expressions $A = m_1 = 1$, $A = m_2 = 1$ and $m_1 \neq m_2$ all yield false and thus represent Boolean combinations which cannot occur in the circuit. These illegal combinations may lead to output values not achievable with the original circuit. Since these values can reduce the worst-case output interval rated by the CELLO score, a non-optimistic estimation could be the result. Thus, they are explicitly excluded when evaluating sub-circuits.

Bounding the Cello Score from Above

The following shows that an upper bound can be found for a circuit's CELLO score by only considering the gate producing the output signal. Let this gate be represented by its Hill parameters θ_j and either be based on transcriptional repression, with the inhibitory Hill function $\mathrm{g}_j(x)$ being its transfer function, or based on activation with Hill function $\mathrm{h}_j(x)$. Further, let x and y denote the gate's input and output promoter activities, respectively. The CELLO score presented in Section 4.2 is reintroduced in (8.8a) and (8.9a). Here, $\overline{Y}$ denotes the set of circuit output values representing a Boolean 1 and $\underline{Y}$ the values representing a Boolean 0. The score is determined by simulating the circuit with respect to all Boolean input combinations, sorting the output values into $\overline{Y}$ and $\underline{Y}$ and calculating the ratio of the minimum value $\overline{y} \in \overline{Y}$ and the maximum value $\underline{y} \in \underline{Y}$.

$$s_{\text{CELLO}}(\gamma, a) = \frac{\min_{\overline{y} \in \overline{Y}}(\overline{y})}{\max_{\underline{y} \in \underline{Y}}(\underline{y})} \tag{8.8a}$$
$$= \frac{\min_{\underline{x} \in \underline{X}}(\mathrm{g}_j(\underline{x}))}{\max_{\overline{x} \in \overline{X}}(\mathrm{g}_j(\overline{x}))} \tag{8.8b}$$
$$\leq \frac{y_{\text{max,j}}}{y_{\text{min,j}}} \tag{8.8c}$$

Equation (8.8a) can be transformed into (8.8b) by inserting the gate's transfer function. Due to its monotonically decreasing characteristic, the numerator is given by the minimization of the transfer function value $\mathrm{g}_j(\underline{x})$ over the set of input values $\underline{x} \in \underline{X}$ representing a Boolean 0. Analog, the denominator can be transformed into a maximization of $\mathrm{g}_j(\overline{x})$ over the input values $\overline{x} \in \overline{X}$ for a Boolean 1. From Section 3.2, it is known that the codomain of the Hill function is defined by $[y_{\text{min}}, y_{\text{max}}]$. It follows that quotient (8.8c) is an upper bound for (8.8b), because

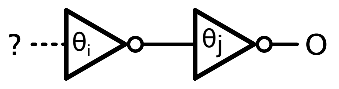
$$y_{\text{max,j}} \geq \mathrm{g}_j(\underline{x}) \wedge y_{\text{min,j}} \leq \mathrm{g}_j(\overline{x}) \;\forall\; \underline{x}, \overline{x} \in \mathbb{R}.$$

The same upper bound for the CELLO score can be derived by assuming an output gate based on activation, as shown in (8.9b). Since $\mathrm{h}_j(x)$ is monotonically increasing, the sets of input values used for minimization and maximization are swapped.

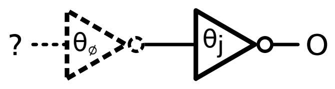
$$s_{\text{CELLO}}(\gamma, a) = \frac{\min_{\overline{y} \in \overline{Y}}(\overline{y})}{\max_{\underline{y} \in \underline{Y}}(\underline{y})} \tag{8.9a}$$
$$= \frac{\min_{\overline{x} \in \overline{X}}(\mathrm{h}_j(\overline{x}))}{\max_{\underline{x} \in \underline{X}}(\mathrm{h}_j(\underline{x}))} \tag{8.9b}$$
$$\leq \frac{y_{\text{max,j}}}{y_{\text{min,j}}} \tag{8.9c}$$

Propagation of Optimistic Input Values

In the previous section, it has been shown that the score of a circuit can be bound from above based on the output interval of its final gate. In the following, the propagation of optimistic input values through a cascade of gates will be detailed, which enables the optimistic estimation of the cascade's score. It represents the basis for the optimistic input interval composition approach presented above.



(a) Indentities known



(b) Indentity of first gate unkown

Figure 8.9.: Cascades of single-input gates with unknown input values

Assume a cascade of two assigned single-input gates based on repression, represented by their set of parameters θ. Let gate θ_i deliver the input signal of gate θ_j, which represents the output gate analog to the previous section, and let the the input values of gate θ_i be unknown (Figure 8.9a). Let further $[y_{\mathrm{min,i}}, y_{\mathrm{max,i}}]$ be the output interval of gate θ_i. Denote by (8.10a) the repetition of the CELLO score with inserted transfer function $\mathrm{g}_j(x)$.

$$s_{\textsc{Cello}}(\gamma, a) = \frac{\min_{\underline{x} \in \underline{X}}(\mathrm{g}_j(\underline{x}))}{\max_{\overline{x} \in \overline{X}}(\mathrm{g}_j(\overline{x}))} \tag{8.10a}$$

$$\leq \frac{\mathrm{g}_j(y_{\mathrm{min,i}})}{\mathrm{g}_j(y_{\mathrm{max,i}})} \tag{8.10b}$$

Then (8.10b) is an upper bound of the score achievable with these two gates, because

$$\mathrm{g}_j(y_{\mathrm{min,i}}) \geq \mathrm{g}_j(\underline{x}) \wedge \mathrm{g}_j(y_{\mathrm{max,i}}) \leq \mathrm{g}_j(\overline{x}) \;\forall\; \underline{x}, \overline{x} \in [y_{\mathrm{min,i}}, y_{\mathrm{max,i}}].$$

That is, without any knowledge of the specific sets of output values $\underline{X}, \overline{X}$ of gate θ_i, the score achievable with the cascade of gates can be bound from above, i.e., estimated optimistically. The estimation is based on the output interval of the first gate in the cascade and the known transfer function of the subsequent gate. It holds transitively for an arbitrarily long cascade of single-input gates. For completeness, the same holds for gates based on activation, i.e.,

$$s_{\textsc{Cello}}(\gamma, a) = \frac{\min_{\overline{x} \in \overline{X}}(\mathrm{h}_j(\overline{x}))}{\max_{\underline{x} \in \underline{X}}(\mathrm{h}_j(\underline{x}))} \leq \frac{\mathrm{h}_j(y_{\mathrm{max,i}})}{\mathrm{h}_j(y_{\mathrm{min,i}})}$$

To further approach the situation that can be found during the estimation of partial solutions in B&B, assume that not only the specific sets of output values of gate θ_i are unknown, but also the identity of the gate, i.e., $\theta_i = \theta_\varnothing = \varnothing$ (Figure 8.9b). In the context of the B&B scheme, gate θ_i then represents the first gate in reversed topological order of the circuit structure that is not assigned. Therefore, particularly its output interval is unknown and not available for the estimation performed in (8.10b). However, in every phase of the gate assignment process, the set $\mathcal{C}$ of gates representing candidates for an assignment available in library $\mathcal{L}$ is known, i.e., $\mathcal{C} = \{\theta \in \mathcal{L} : \theta \notin a\}$. Thus, also the output intervals of these gates are known. Let $y_{\text{min},\mathcal{C}}$ denote the smallest y_{min} of all candidate gates left in the library and $y_{\text{max},\mathcal{C}}$ the highest y_{max} of these gates. Then, irrespective of the assignment choice for gate θ_i, it holds that

$$y_{\text{min},\mathcal{C}} \leq y_{\text{min,i}} \wedge y_{\text{max},\mathcal{C}} \geq y_{\text{max,i}}. \tag{8.11}$$

Taking into account (8.11), the optimistic estimation (8.10b) can be adapted to the presented optimistic input interval composition, i.e.,

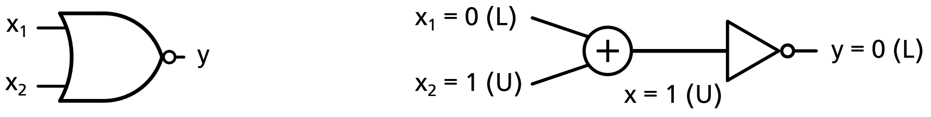
$$s_{\text{CELLO}}(\gamma, a) \leq \frac{g_j(y_{\text{min,i}})}{g_j(y_{\text{max,i}})} \leq \frac{g_j(y_{\text{min},\mathcal{C}})}{g_j(y_{\text{max},\mathcal{C}})}$$

Again, the same holds transitively for an arbitrarily long cascade of single-input gates as well as for gates based on activation. Consequentially, the values at each site of the cascade evaluated with these optimistic input values represent optimistic estimations. Specifically, values for a Boolean 0 represent lower bounds, and values for a Boolean 1 represent upper bounds with respect to the values at these sites when all gates in the cascade are assigned specific genetic gates and specific input values into the cascade are known.

Propagation Through Multi-Input Gates

In the previous section, it has been shown that the optimistic input interval composition approach is guaranteed to bound the CELLO score from above that can be reached with a cascade of single-input gates preceded by a gate of unknown parameters. However, general combinational logic circuits feature multi-input gates that combine n signal paths into one by applying a n-ary Boolean operation. Thus, it has to be shown that the presented approach for estimation also holds for multi-input gates. In the following, a counter-example is given that shows how the approach is violated by a simple two-input NOR gate. Consequentially, a general solution is presented that restores the guarantee of optimistic estimation, called *substitution* of invalid bounds.

Counter-Example Assume a two-input NOR gate with inputs x_1 and x_2 and output y (Figure 8.10a). On the level of digital abstraction, a Boolean 1 output is expected only in case both inputs represent a Boolean 0. On the genetic technology level, this behavior is accomplished by superposing both input promoter activities and the inhibitory relation of the superimposed signal to the output promoter, modeled by the inhibitory Hill function. Figure 8.10b depicts this decomposition of the NOR gate with intermediate



(a) External view (b) Decomposition with exemplary Boolean signals

Figure 8.10.: External and internal signals of two-input NOR gate

signal x and exemplary Boolean values for all signals. Assume that the gate is part of a partial assignment in the B&B scheme, such that the input values represent estimated upper or lower bounds, respectively, gained from optimistic input interval composition. In parentheses, it is given for each signal whether it is expected to be an upper bound (U) or a lower bound (L) to the values possible at this site for the signals to adhere to the paradigm of optimistic estimation. Exemplarily, the value of input x_1 represents a valid lower bound, and the value of x_2 is a valid upper bound. According to the given input combination of mixed Boolean valency, a Boolean 0 output of the NOR gate is expected to represent a lower bound. Thus, given the monotonicity of the Hill function, the intermediate signal x is expected to represent an upper bound. However, the superposition of x_1 and x_2 does not yield a valid upper bound because x_1 represents a lower bound to the values possible at this site. That is, according to the presented estimation scheme, x_1 will not feature *lower* values irrespective of how the specific assignment of gates leading to x_1 materializes. Thus, x_1 may feature higher values for refined partial or complete assignments. Therefore, $x = x_1 + x_2$ may also feature higher values, and $y = \mathrm{g}(x)$ may feature lower values, violating the expected lower bound at y. Note that this also holds for implicit genetic OR gates, which feature only the superposition operation, and multi-input gates with an arbitrary number of inputs.

Substitution of Invalid Bounds To restore the guarantee of optimistic estimation and thus exactness of the B&B scheme, it has to be ensured that the output values of multi-input gates represent valid bounds according to their logically implied Boolean values. In the given example, the value of x_1 has to be substituted with a guaranteed upper bound so that x represents a valid upper bound and y a valid lower bound, respectively. Generally, if the result of a superposition of promoter activities is a Boolean 1, all input signals of this superposition have to represent valid upper bounds. The value of each signal, which is no upper bound, has to be substituted accordingly. Table 8.1 shows the input signals subject to substitution for all combinations of a Boolean superposition of two input signals. As a a substitute, a value that is guaranteed to be an upper bound to the concerning signal must be determined. Additionally, the substitute has to be as realistic as possible to reduce the

Table 8.1.: Boolean superposition of input signals and required substitutions

x_1	x_2	$x = x_1 + x_2$	substitution
0	0	0	-
1	0	1	$x_2 \rightarrow y_{\max(,\mathcal{C})}$
0	1	1	$x_1 \rightarrow y_{\max(,\mathcal{C})}$
1	1	1	-

impact of the substitution on the estimation's accuracy. Values that fulfill both criteria are given by the fundamental parameters of the genetic gates. Therefore, analog to the optimistic input interval composition, the upper bound is taken from the preceding gate's output range $[y_{\min}, y_{\max}]$. If the gate sourcing the input subject to substitution is known, i.e., assigned, the appropriate upper bound is given by the gate's maximal output $y_{\max}$. If the input is sourced by a gate that has not yet been assigned a specific genetic gate, the maximum output of the gates left in the library $y_{\max,\mathcal{C}}$ is determined and used as a substitute.

Exact and Heuristic Mode Substituting invalid bounds restores the guaranteed optimistic estimation of partial solutions and thus the exactness of the B&B scheme. As shown above, however, values representing a Boolean 0 have to be substituted with the maximum value possible for the given signal. Naturally, this reduces the tightness of the estimation by introducing substitute values that differ strongly from the original values. Thus, a reduction of the search efficiency has to be assumed. Consequentially, the substitution is implemented in the presented B&B method as an optional feature. This gives rise to an exact mode, in which the substitution is carried out, and a heuristic mode, in which the substitution is omitted. The heuristic mode is characterized by a more accurate estimation of partial solutions' quality, which is not strictly optimistic. Therefore, it is expected to traverse the search tree more efficiently but runs the risk of pruning the tree branch containing the exact solution.

In the exact mode, the estimator complies with the desired properties stated at the beginning of the section. First, the estimation for every partial solution $a_p \subseteq a$, i.e., subcircuit of assigned gates with preceding unassigned gates, provides an upper bound to the CELLO score of a (8.5). This has first been shown for cascades of single-input gates and then generalized for multi-input gates, giving rise to the substitution of invalid bounds. Second, the estimator yields exact values for complete solutions (8.6). This criterion is fulfilled naturally since the estimator is based on the same calculation as the CELLO score. The more estimated input values are replaced by outputs of assigned gates in the course of the B&B scheme, the more the estimation converges to the exact score. Finally, when rating partial solutions that are built on each other, the estimator never rates the refined solution higher than the original solution (8.7). When an assignment is refined, i.e., extended by one gate, the choices in the library for composing the optimistic input intervals are reduced, and the sub-circuit is extended by one gate with an individual transfer function. Both refinements lead to an equal or reduced output interval of the refined sub-circuit compared to the original sub-circuit.

8.4. Evaluation

Note: Parts of this section have already been published in [58, 61]. Explicit self-citation is omitted to provide an improved reading flow.

The presented methods for gate assignment have been implemented in the GDA tool ARCTIC (Section 4.5). For evaluating the methods, a set of 66 circuit structures has been compiled, which is used as a benchmark set. One half of the set encompasses the circuit topologies synthesized by CELLO for 33 Boolean functions that have already been implemented *in-vivo* [53]. The other half represents structural variants of these circuits synthesized using the enumeration method presented in Chapter 7. More specifically, for each function, the best-performing variant with respect to the CELLO score has been selected from the set of minimal variants. The result is a set of 66 circuit topologies with sizes ranging from 2 to 8 logic gates.

The remainder of this section begins with an examination of specific aspects of the two presented gate assignment methods using subsets of the set of circuits. First, the proximity-measure of the SA heuristic is tuned for the specific gate library used (Section 8.4.1). Then, the different modes and search strategies of the B&B scheme are evaluated (Section 8.4.2). Considering the results of these analyses, the methods are compared in terms of efficiency and solution quality using the whole set of benchmark circuits (Section 8.4.3).

8.4.1. Tuning the Proximity-Measure

The proximity-measure presented in Section 8.2.3 is used to generate neighboring solutions in the SA scheme based on a dynamically controlled radius. Characteristic features of the genetic gates' transfer functions are mapped to a feature space in which their Euclidean distance defines the proximity of two gates. Compared to analog applications of this technique, e.g., the placement of logic cells on a chip, the dimensions in the feature space do not stem from geometric features of the problem. However, they are based on characteristics that are assumed to impact the similarity of solutions in terms of their objective function value. Thus, a diagonal weighting matrix has been introduced in the referenced section. Each weight factor allows scaling one dimension in the feature space. By this, the proximity-measure can be further adapted to the gate assignment problem and to the specific gate library to be used. Let $\mathbf{w} := (\omega_{y_\mathrm{m}}, \omega_{x_\mathrm{m}}, \omega_{\mathrm{g}'})$ denote the vector of weight factors from the diagonal of the weighting matrix, referring to characteristic features $d_1 := y_\mathrm{m}$, $d_2 := x_\mathrm{m}$, and $d_3 := \mathrm{g}'(x_\mathrm{m})$ (Section 8.2.3). In the following, the process of finding values for the weight factors that lead to a maximum increase in the SA scheme's efficiency is detailed.

Table 8.2.: Mean number of simulations needed per circuit for gate assignment with SA using different weight configurations

Configuration	ω_{y_m}	ω_{x_m}	$\omega_{g'}$	Simulations	Speedup
none	0.0	0.0	0.0	639 855	1.00
equal	1.0	1.0	1.0	396 763	1.61
best	0.1	0.9	0.0	280 844	2.23

First, topologies with 5 gates or more have been selected from the set of 66 circuits. These form a subset of 32 medium and large circuits, for which the optimal gate assignment cannot easily be found by exhaustive search. Then, the SA heuristic has been used to assign the gates of CELLO's gate library (Section 6.1) to these circuits. Since SA represents a stochastic method, its convergence and yielded solution can vary between multiple runs with the same parameters. Thus, all gate assignment optimization runs have been carried out ten times, allowing for aggregating resulting data and resulting in a total of 320 SA runs for assigning the selected circuits. These runs have been carried out with various values for the weight factors to examine the impact of the proximity-measure on the method's convergence.

As a baseline, the configurations $\mathbf{w} = (0.0, 0.0, 0.0)$ and $\mathbf{w} = (1.0, 1.0, 1.0)$ have been chosen, leading to a deactivation of the proximity-measure and an equal weighting of all dimensions, respectively. The results for these configurations are depicted in the first two rows of Table 8.2, named "none" and "equal". For each configuration, the mean number of simulations per circuit topology required for the SA scheme to converge is shown. Furthermore, the speedup compared to the basic SA scheme without proximity-measure is noted. It can be seen that the usage of the proximity-measure without further tuning provides a speedup of 1.61 over the basic SA. A parameter sweep over a total of 66 different combinations of weight factors has been performed to tune the proximity-measure to the specific problem and gate library. The combinations are obtained by sweeping each parameter in the interval $[0.0, 1.0]$ in steps of 0.1 and requiring the sum of all factors to equal 1.0. The resulting convergence behavior of the SA scheme is depicted in Figure 8.11. In the shown plane, each factor is maximal in the corner its label is placed in and minimal at the opposite of this corner. Each colored hexagon represents the result of one gate assignment run with its color coding the mean number of simulations needed. The lighter the color, the lower the number of simulations needed. It can be seen that in the worst case, for configuration $\mathbf{w} = (0.2, 0.0, 0.8)$, the efficiency is slightly worse compared to the "none" configuration with $\sim 6.44 \times 10^5$ simulations needed, compared to $\sim 6.4 \times 10^5$ simulations.

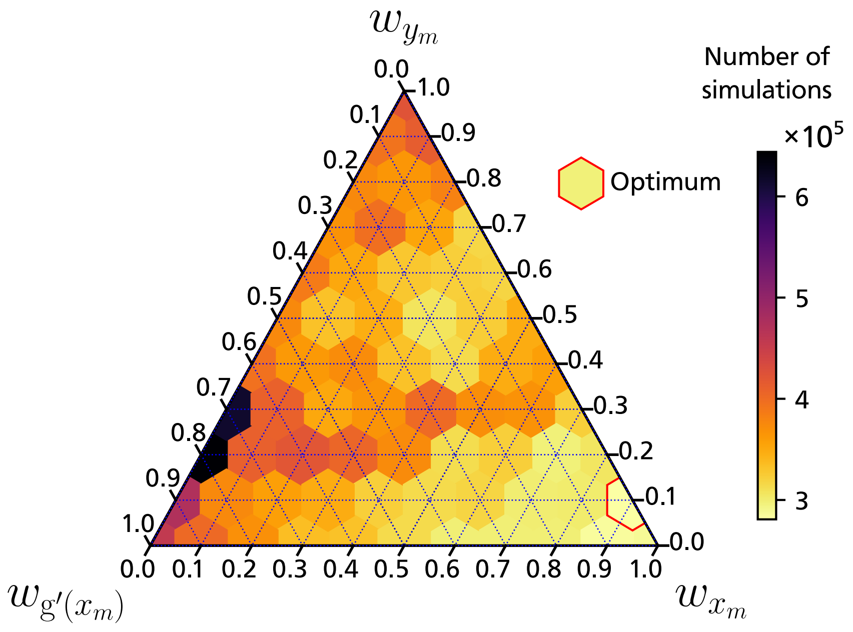


Figure 8.11.: Convergence of SA for 66 weight configurations

For all other configurations, the efficiency exceeds the method without proximity-measure. The optimum, which is marked in red, is given by $\mathbf{w} = (0.1, 0.9, 0.0)$ with a mean number of $\sim 2.8 \times 10^5$ simulations needed. Compared to the "none" configuration, a speedup of 2.23 is achieved with the "best" configuration.

The parameter sweep results allow for conclusions about the choice of characteristic features. The optimal configuration of weight factors heavily emphasizes feature x_m, while omitting $\mathrm{g}'(x_\mathrm{m})$. Thus, for the given gate library, the similarity in the steepness of the transfer functions of two gates does not represent a good indicator of their swap's impact on the circuit output characteristic. The opposite holds for their input value x_m evoking the transfer curves inflection point. This can be explained by the fact that for an individual swap during the annealing scheme, the gates providing the input for the gate subject to the swap remain the same. Thus, the input values of the swapped gate remain the same, and a strong shift in its x_m may lead to portions of the input values falling into a different region of the transfer curve, provoking a changed output behavior of the gate.

For the remainder of this work, the SA method is examined using the optimum weight configuration.

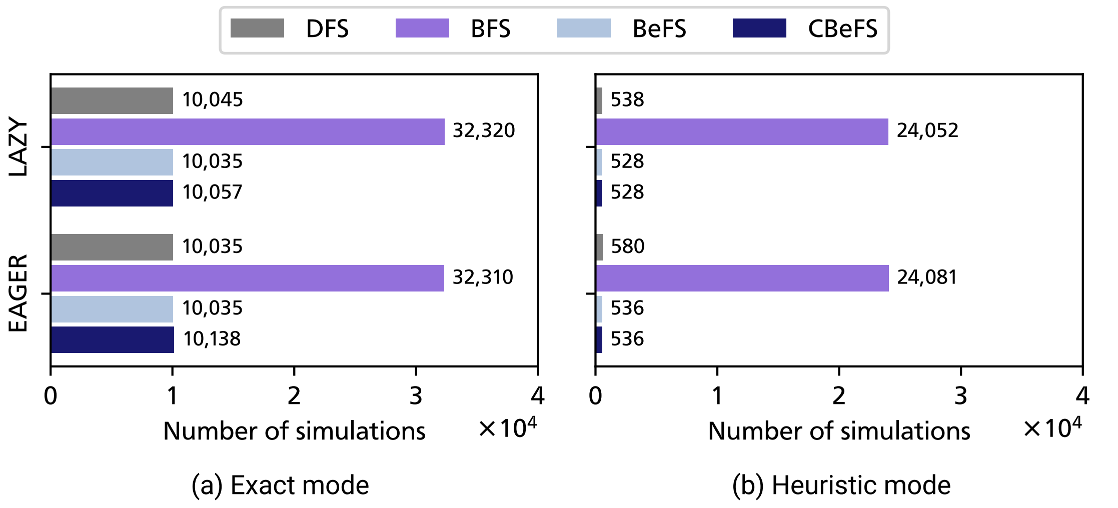


Figure 8.12.: Efficiency of B&B modes and search strategies

8.4.2. Modes and Strategies of Branch-and-Bound

For the B&B method presented in Section 8.3, a range of modes of operation and search strategies have been implemented. These are examined exhaustively in the following to determine the best setup of the method for the gate assignment problem. A subset of 6 circuits has been selected from the complete set, consisting of small and medium-sized circuits ranging from 2 to 6 gates. Then, the gate assignment optimization has been carried out for these circuits using Cello's gate library and 16 different B&B setups. They result from combining the two B&B modes lazy and eager with the four search strategies depth-first search (DFS), breadth-first search (BFS), best-first search (BeFS) and cyclic BeFS. Each of these combinations has been evaluated for the exact mode of B&B, in which the substitution of invalid bounds is performed, and the heuristic mode, respectively. The metrics examined in the following are the efficiency of the search tree traversal, measured in circuit simulations needed, and the efficiency in terms of memory usage, measured in the maximum number of elements in the search buffer.

Figure 8.12 shows the evaluation results regarding the search efficiency. In the exact mode (Figure 8.12a), BFS performs considerably worse than all other search strategies for both eager and lazy B&B. The other search strategies deliver comparable results, while the best-performing setups are DFS with eager B&B and BeFS with eager or lazy B&B ($10\,035$ simulations). In heuristic mode (Figure 8.12b), the efficiency is generally increased, but the comparison of setups yields results similar to the exact mode. While

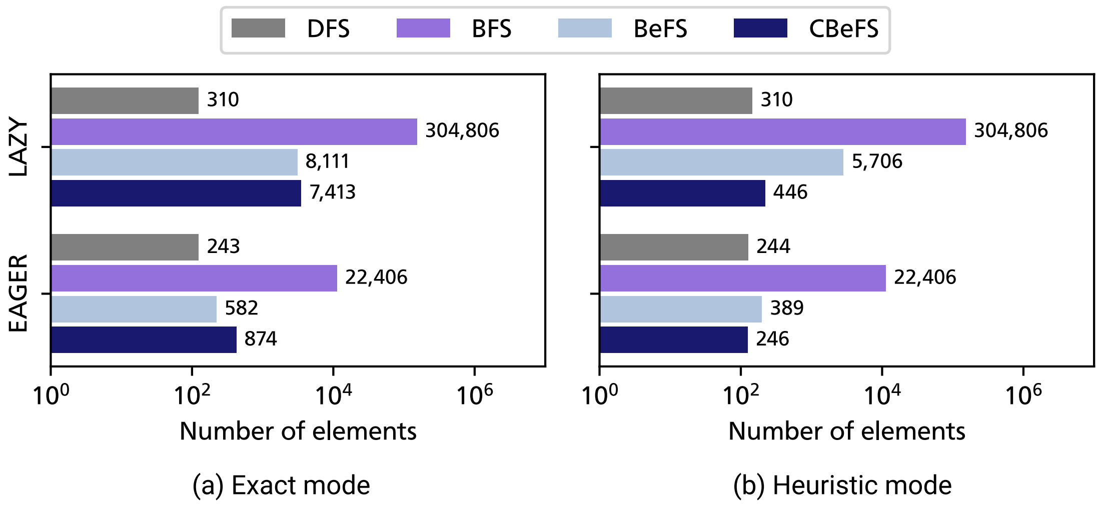


Figure 8.13.: Maximal element buffer sizes for B&B modes and search strategies

BFS again performs worst, the other strategies are on the same level. The best efficiency in heuristic mode is achieved by BeFS and cyclic BeFS with lazy B&B (528 simulations).

Figure 8.13 depicts the maximum number of elements in the search buffer encountered during the gate assignment. The results are plotted on the logarithmic scale due to the wide range of resulting buffer sizes. For this metric, a considerable difference can be observed for lazy and eager B&B, with eager B&B being much more memory efficient. Again, BFS performs worst for both lazy and eager B&B in exact (Figure 8.13a) and heuristic (Figure 8.13b) mode (up to $\sim 3 \times 10^5$ elements). DFS features the best efficiency in terms of buffer elements for all combinations of modes. BeFS and cyclic BeFS yield comparably good results, if eager is chosen as B&B mode. The best-performing setup is DFS with eager B&B for both exact and heuristic modes (243 and 244 elements, respectively).

It can be concluded from the examination of both aspects that BFS is not suited for the gate assignment problem. This is caused by the fact that the search tree is perfectly balanced, i.e., the paths from the root to every leaf node are equally long. Thus, BFS evaluates all partial solutions before visiting the first complete solution that sets the lower bound and enables pruning. All other search strategies provide means to visit the leaf nodes, i.e., complete solutions, earlier in the search process. Further, it can be observed that the choice of eager or lazy B&B has no considerable effect on the search efficiency. However, it impacts the memory efficiency, especially when BeFS or cyclic BeFS are used. The maximum number of elements in the buffer is lower for eager B&B because the upper bound is determined before adding elements to the buffer. By this, the addition of inferior

Table 8.3.: Search efficiency and runtime of the gate assignment methods

	Search Efficiency			Runtime	
Method	Simulations	Gain	Trav. (%)	Runtime (h)	Speedup
B&B (heuristic)	858 874	611.2×	0.2	13.8	721.9×
Simulated Annealing	6 642 130	79.0×	1.3	92.7	107.5×
B&B (exact)	34 039 365	15.4×	6.5	505.2	19.7×
Exhaustive Search	524 937 849	1.0×	100.0	9962.3	1.0×

partial solutions to the buffer is avoided. For the remainder of this work, eager B&B with BeFS is chosen as B&B configuration since it provides a good combination of search efficiency and memory efficiency for both the exact and the heuristic search.

8.4.3. Comparison of the Methods

A comprehensive comparison of the gate assignment methods has been carried out by assigning Cello's library of genetic logic gates to the complete set of 66 circuit topologies with each method. For each single gate assignment process, the number of needed simulations, the runtime, and the quality of the resulting assignment in terms of the objective function value, i.e., the Cello score, has been determined. As a baseline, an exhaustive search (Section 8.1) has been carried out. It provides information on the size of the space of possible assignments and the exact solution for each circuit. The SA scheme (Section 8.2) has been set up with the optimal proximity-measure configuration (Section 8.4.1). All gate assignment runs with SA have been repeated ten times, taking into account its stochastic nature and allowing for aggregating the resulting data. Further, the B&B method (Section 8.3) is evaluated using the combination of eager B&B and BeFS, that is efficient in terms of runtime and memory usage (Section 8.4.2). Both the exact and heuristic modes of B&B are considered for the evaluation. At the end of the section, the methods are compared to the gate assignment scheme of the GDA tool Cello. All runs have been carried out on an AMD EPYC server supporting 128 threads.

Search Efficiency

Figure 8.14a depicts the search efficiency of the methods in terms of needed circuit simulations for assigning all 66 circuits. Note that the results are plotted on the logarithmic scale starting at 1×10^5 to improve the clarity of the figure. Columns 2 to 4 of Table 8.3

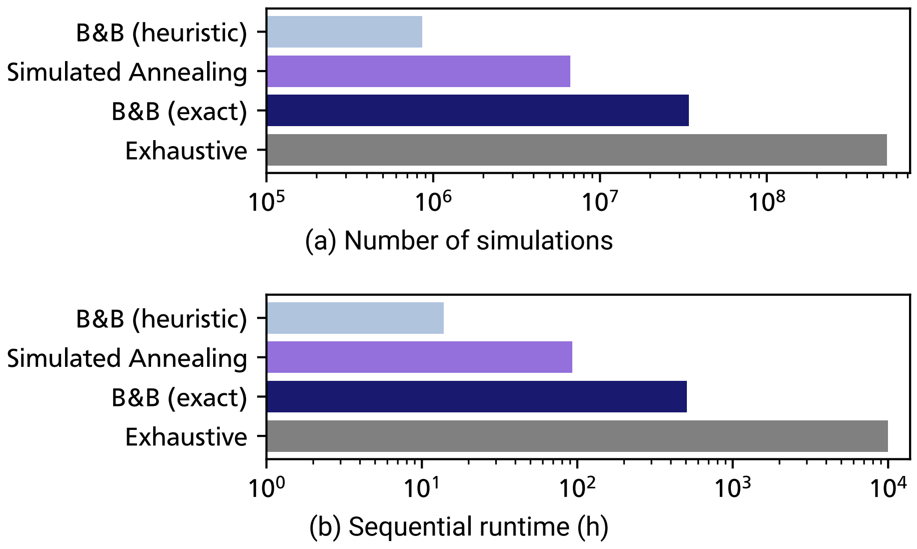


Figure 8.14.: Efficiency and runtime measurements of the gate assignment methods

additionally show the underlying number of simulations, the efficiency gain compared to exhaustive search, and the percentage of the search space traversed. When searching for the exact solution, B&B requires $\sim 3.4 \times 10^7$ simulations, providing a $15.4\times$efficiency gain compared to exhaustive search, which finishes after $\sim 5.2 \times 10^8$ simulations. Thus, to find the exact solution, B&B traverses only $6.5\,\%$ of the search space. The mean number of simulations required for the SA heuristic to converge is $\sim 6.6 \times 10^6$, providing a $79\times$efficiency gain compared to exhaustive search and traversing $1.3\,\%$ of the search space. Heuristic B&B again reduces the number to $\sim 8.6 \times 10^5$ simulations, evaluating only $0.2\,\%$ of the possible assignments. Compared to the other heuristic SA, B&B provides an efficiency gain of $7.7\times$. Compared to exhaustive search, the gain comprises $611.2\times$.

Runtime

The measured runtime in hours is plotted in Figure 8.14b on the logarithmic scale. Note the different scaling of the axis compared to Figure 8.14a. Further, columns 5 and 6 of Table 8.3 show the underlying measurements and the speedup compared to the exhaustive search, respectively. To use the server to capacity, the runs have partly been carried out in parallel, and the measured runtimes have been added up, corresponding to a sequential, single-threaded execution. The results parallel those for the number of required simulations. However, the obtained speedup surpasses the observed efficiency gain. When searching

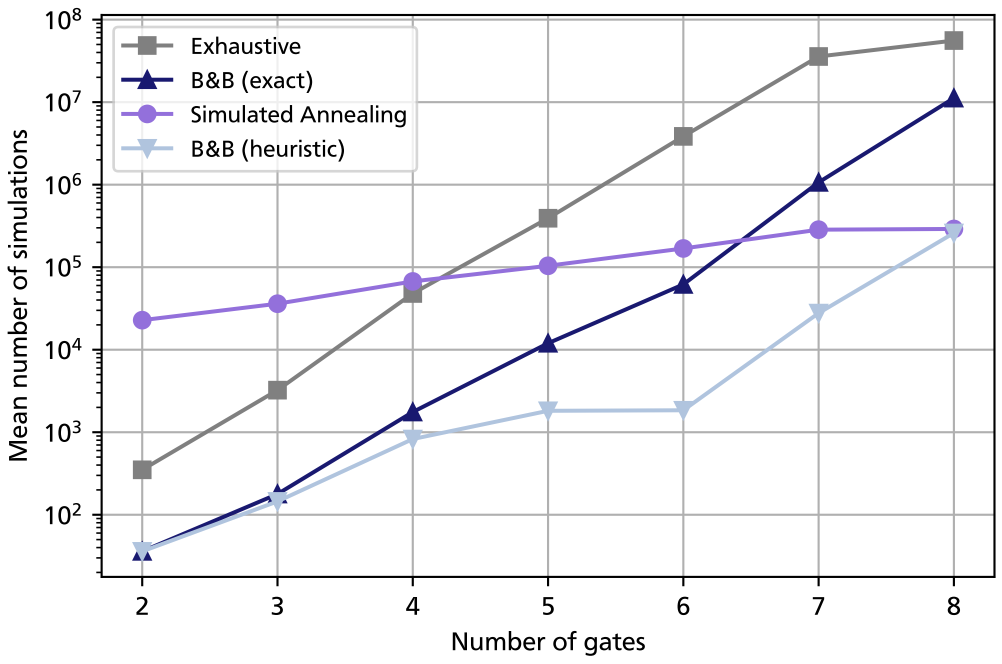


Figure 8.15.: Mean number of simulations by circuit size

for the exact solution, B&B requires $\sim 505\,\mathrm{h}$, providing a $19.7\times$speedup compared to exhaustive search, which requires $\sim 1 \times 10^4\,\mathrm{h}$. The heuristic methods SA and heuristic B&B require $92.7\,\mathrm{h}$ and $13.8\,\mathrm{h}$, respectively. Thus, for finding a heuristic solution, B&B provides a $6.7\times$reduction in runtime compared to SA and a $721.9\times$speedup compared to exhaustive search. The speedup surpasses the efficiency gain because the absolute number of simulations that can be spared grows with the circuit size, and bigger circuits generally require more runtime per simulation.

Impact of Circuit Size

The mean number of simulations needed with respect to the circuit size is depicted in Figure 8.15. It can be observed that all methods exhibit an exponential complexity. Exact B&B parallels the characteristic of exhaustive search but with the number of simulations needed reduced by one to two orders of magnitude. The same holds for heuristic B&B

Table 8.4.: Solution quality of the gate assignment methods

Method	Exact solutions	Mean deviation (%)	Max. deviation (%)
B&B (heuristic)	62/66	-1.5×10^{-3}	−0.11
Simulated Annealing	59/66[a]	−0.02	−3.15
B&B (exact)	66/66	0.00	0.00

[a] Circuits for which SA returned the exact solution in all 10 runs.

compared to exact B&B for circuits with 6 gates or more, while it approaches the exact variant for smaller circuits. SA exhibits excess simulations for small circuits but features a reduced dependency on the circuit size. For the biggest circuits assigned comprising 8 gates, SA approaches the efficiency of heuristic B&B.

Solution Quality

Besides the runtime, the solution quality plays a major role when evaluating heuristic methods. Further, comparing the quality of the solutions provided by the exact B&B method to those obtained from exhaustive search allows for a verification of B&B's exactness. Table 8.4 shows the results regarding the solution quality based on the results from an exhaustive search. For each method, the number of exact solutions found for the set of 66 circuits, the mean deviation from the score of the exact solution, and the maximum deviation from this score are depicted. For SA, two of these metrics are based on a worst-case evaluation of the 10 runs. First, a solution is only counted as exact if SA found it in all 10 runs. Second, the maximum deviation is determined based on the complete set of 10×66 SA gate assignment runs. A first important observation is that the results verify the exactness of the exact B&B method. It obtains the optimum solutions for all 66 circuits and thus does not exhibit a deviation from the score obtained by exhaustive search. The SA heuristic does not find the exact solution for 7 circuits, features a mean deviation of −0.02 % and a worst-case deviation of −3.15 %. B&B in its heuristic mode misses the exact solution for 4 circuits. It also features a smaller deviation from the optimum than SA with a minuscule mean deviation of -1.5×10^{-3} % and a worst-case deviation of −0.11 %.

Comparison to Cello

Finally, the methods are compared to the gate assignment scheme of the GDA tool Cello. Since the software does not support alternative circuit topologies, this comparison is based

Table 8.5.: Comparison of the gate assignment methods to CELLO based on a reduced set of 33 circuits

Method	Simulations	Exact solutions	Mean dev. (%)	Max. dev. (%)
B&B (heuristic)	156 075	31/33	-4.8×10^{-5}	-2.7×10^{-3}
SA	2 994 648	30/33[a]	−0.02	−2.03
B&B (exact)	6 905 142	33/33	0.00	0.00
CELLO's SA	990 000	24/33[a]	−1.6	−64.94

[a] Circuits for which SA returned the exact solution in all 10 runs.

on the subset of 33 circuit structures stemming from CELLO itself and does not include the 33 structural variants. CELLO applies a simple SA method with a static annealing schedule [53] (Section 8.2.2). In this scheme, 50 separate optimization runs are carried out subsequently, each starting at an individual random initial solution. The outer-loop criterion is given by the fixed limit of 600 outer-loop iterations. The temperature is initialized with a fixed value and decreased in every iteration, irrespective of the acceptance rate. The schedule does not feature an inner-loop, i.e., the inner-loop criterion s given by $k_{\max} = 1$. Consequentially, on each temperature level, only one perturbation is performed to the solution. In total, $30\,000$ circuit simulations are performed during a gate assignment optimization, irrespective of the problem size. Due to the quick annealing schedule, the scheme resembles 50 hill-climbing runs, starting at random points in the search space. The neighbor generation equals the scheme of single swaps used in the presented SA method, without the proximity-metric (Section 8.2.3).

Table 8.5 lists important metrics of the presented gate assignment methods and CELLO's SA scheme. Based on the number of circuit simulations needed for assigning the 33 circuits, CELLO features a $\sim 3\times$improved efficiency compared to the presented SA scheme, while the heuristic B&B method again features a $\sim 6.3\times$improvement over CELLO. Since CELLO's SA is a stochastic method, too, it has been run 10 times to aggregate data regarding its solution quality. In these runs, it converged on the exact solution in every run for ~ 73 % of the 33 circuits, compared to ~ 90.9 % achieved by the proximity-based SA and ~ 94 % by the heuristic B&B. Its mean deviation from the optimum score is $80\times$and $\sim 3.3 \times 10^4\times$higher compared to the proximity-based SA and the heuristic B&B, respectively. Most substantially, the worst-case deviation measured with CELLO's SA of ~ -65 % is by an order of magnitude higher compared to the presented SA and multiple orders of magnitude higher than the one of heuristic B&B.

8.5. Discussion

In this chapter, gate assignment methods for optimizing the adherence of genetic circuits to the functional specification, expressed by the Cello score, have been presented. All methods consider characteristics of the considered technology of genetic logic gates. In the SA scheme, similarity between the transfer functions of genetic gates is used to control the impact of perturbations performed during the generation of neighboring solutions. By gradually enforcing increased similarity of the neighbors during the annealing, the convergence of the method is improved. The presented B&B method is based on fundamental features of transcriptional regulation. It leverages the natural hierarchy of the problem that originates from the bounded output intervals of genetic gates and that results in the last gate of the circuit having the most significant impact on the output characteristic. Based on this observation, partial solutions are built successively in the reversed topological order of the circuit. The monotonicity of the gates' transfer functions is used to estimate the quality of these sub-circuits. This enables pruning branches of the search tree and traversing the search space efficiently. If the strictly optimistic estimation mode is chosen, the method is guaranteed to be exact. If the criterion of optimistic estimation is relaxed, the B&B scheme represents an alternative heuristic method.

Suppose optimality of the gate assignment with respect to the Cello score is a strict requirement of the GDA process. In that case, the exact B&B is guaranteed to obtain this exact solution with a $\sim 20\times$speedup compared to exhaustive search. If exactness is not strictly required, the SA method and the heuristic variant of B&B provide gate assignments with good quality while providing a further $\sim 5.4\times$and a $\sim 36.6\times$speedup compared to the exact B&B, respectively. The heuristic B&B method delivers exact solutions for $94\,\%$ of the tested circuits and near-optimal results for the remaining circuits with a maximum deviation from the optimum score of $-0.11\,\%$. Given the natural variability of biological systems, it is questionable whether this observed minuscule worst-case deviation has any practical implications for the synthesized circuit. Thus, heuristic B&B is the most practical choice for mapping genetic circuits of sizes currently implementable. Its efficiency enables exploring extended design spaces, such as structural variants of circuits (Chapter 7) and allows for using more complex circuit models that increase the computational complexity of circuit simulations. Further, it outperforms the heuristic gate assignment method of the GDA tool Cello in both efficiency and quality of generated assignments. The advantage of the proximity-based SA scheme is its flexibility concerning the circuit score used. Due to its observed efficiency being less dependent on the circuit size, it may become a relevant alternative if circuits or libraries grow.

Increased Meaningfulness of Designs

9. Technology Mapping for Robustness

The gate assignment methods presented in Chapter 8 provide means to optimize genetic circuits efficiently for the Cello score. Its calculation is based on median representations of gates obtained by flow cytometry (Section 4.2). Thus, a population of cells implementing a circuit is represented by one instance of the circuit built from these median-based gates. In some applications, such as biotechnology, quantifying and optimizing the separation of output states of this representative circuit instance may provide a suitable means to control processes adequately. In biomedical applications, however, the behavior of individual circuit instances, i.e., individual cells, plays a vital role in avoiding undesired damage to the patient. Thus, a design process for genetic circuits destined for such sensitive applications has to consider cell-to-cell variability and ensure that outliers in terms of functionality are reduced or avoided. In this work, the term robust is used to describe circuits that retain their functionality even under assumed cell-to-cell variability (Section 4.3). A technology mapping process that delivers robust circuits makes *robustness* its primary or secondary objective.

In this chapter, two methods to integrate robustness as an objective in the technology mapping for genetic circuits are presented. First, an alternative circuit score is used that integrates robustness and circuit functionality, the E-Score (Section 4.3). To optimize circuits for it, the flexibility of the SA gate assignment method presented in Section 8.2 with respect to the used objective function is leveraged. The results in terms of efficiency, runtime, and solution quality are presented in Section 9.1. Second, a way to integrate robustness as an additional constraint into the technology mapping process is presented in Section 9.2. It is based on a new criterion for the compatibility of gates, taking into account their transfer characteristics, and allows considering robustness when traditional, non-statistical scores such as the Cello score are used. It has been integrated into and evaluated with the B&B method from Section 8.3. Finally, both methods are compared in Section 9.3 regarding runtime and solution quality based on the assumption that the E-Score represents the ground truth quantifying the robustness of circuits.

9.1. Optimization for the E-Score

The E-Score introduced in Section 4.3 represents an alternative circuit score for genetic logic circuits. Its calculation is based on a statistical description of gates from which random samples are drawn. By this, n random instances of the circuit are simulated to obtain output distributions rooted in the statistical description of the gates. This allows a comprehensive statistical evaluation of the output characteristic instead of rating the distance of medians as done by the CELLO score. The statistical measure used by the E-Score is based on the Wasserstein metric and penalizes overlapping output distributions. If the statistical description of the gates takes cell-to-cell variability into account, a circuit optimized for the E-Score is less prone to evoke instances with non-differentiable output states in a population of cells exhibiting variability.

Generally, the SA heuristic for gate assignment presented in Section 8.2 is flexible regarding the used objective function. Thus, it is possible to exchange the CELLO score applied in Chapter 8 for the E-Score and perform technology mapping for robust circuits. In this section, the performance of the SA method is evaluated in terms of search efficiency, runtime, and solution quality when the E-Score is chosen as the objective function. To this end, the method is used to assign the set of 66 circuit structures (33 functions, two circuit variants per function) the genetic logic gates of CELLO's gate library, analog to Section 8.4. The number of drawn samples and circuit instances per scoring of $n = 5000$ is chosen. To account for SA's stochastic behavior, the process has been repeated 10 times, allowing for aggregating the resulting data. Note that the E-Score is not quantitatively comparable to the CELLO score, as it is based on a different calculation.

Adaptation of the Inner-Loop Criterion

First, the SA scheme has been used to optimize circuits for the E-Score with the parameters presented in Section 8.2.2. These original parameters have been obtained by tuning the method for the CELLO score, and it showed that the E-Score is of a more complex nature, leading to a degradation in solution quality for large circuits. Across the set of circuits, a mean deviation compared to the optimal solution of $-0.2\,\%$ has been measured with the E-Score. This represents a $10\times$degradation compared to mapping with the CELLO score ($-0.02\,\%$ mean deviation). Further, a worst-case deviation of $-24.9\,\%$ could be found among the 10 repeated SA runs. The number of 60 circuits for which the exact solution could be found in all runs is similar to that of using the CELLO score (59 circuits). The worsened deviation only occurred for circuits featuring 7 or more gates, hinting at an insufficient impact of the problem size on the annealing schedule for mapping with the E-Score. The parameter that introduces the problem size to the annealing schedule

Table 9.1.: Search efficiency and runtime of Simulated Annealing (SA) compared to an exhaustive search (Exh.) with the E-Score and the Cello score

		Search Efficiency			Runtime	
Method	Score	Simulations	Gain	Trav. (%)	Runtime (h)	Speedup
SA	E-Score	11 580 687	45.3×	2.2	128.7	127.4×
Exh.		524 937 849	1.0×	100.0	16 398.9	1.0×
SA	Cello	6 642 130	79.0×	1.3	92.7	107.5×
Exh.		524 937 849	1.0×	100.0	9962.3	1.0×

is the inner-loop criterion $k_{\max}$, which defines the number of iterations spent on each temperature level (Section 8.2.2). Since the degradation only occurs for large circuits, increasing the weight of the problem size suggests itself, instead of increasing $k_{\max}$ for all problem sizes. Thus, the inner-loop criterion is adapted to the E-Score by increasing the exponent of $|\mathcal{A}|$ the problem size from 0.25 to 0.5, i.e.,

$$k_{\max} = \begin{cases} 50 * \sqrt[4]{|\mathcal{A}|}, & \text{if the Cello score is used,} \\ 1 * \sqrt{|\mathcal{A}|}, & \text{if the E-Score is used.} \end{cases}$$

Further, the leading factor is decreased to reduce the overhead for small circuits. The impact of the modified criterion on the number of simulations with respect to the circuit size is visualized in Figure 9.1. For circuits with 7 or more gates, the adapted criterion increases the number of simulations compared to the original criterion. The evaluation presented in the following regarding efficiency and solution quality considers the adapted criterion.

Search Efficiency and Runtime

Table 9.1 shows the results for the search efficiency of the SA method compared to exhaustive search. For reference, the lower two rows repeat the results of Section 8.4.3 obtained with the Cello score. All results for the SA scheme are mean values of the 10 performed runs. The SA method needed $\sim 11.6 \times 10^6$ circuit simulations to optimize the set of circuits with the E-Score. Compared to exhaustive search, this equals a $\sim 45\times$gain in efficiency, which translates to $2.2\,\%$ of the search space explored. Analog, the runtime of SA amounts to a mean $\sim 129\,\text{h}$, representing a $\sim 127.4\times$speedup over exhaustive search. Two

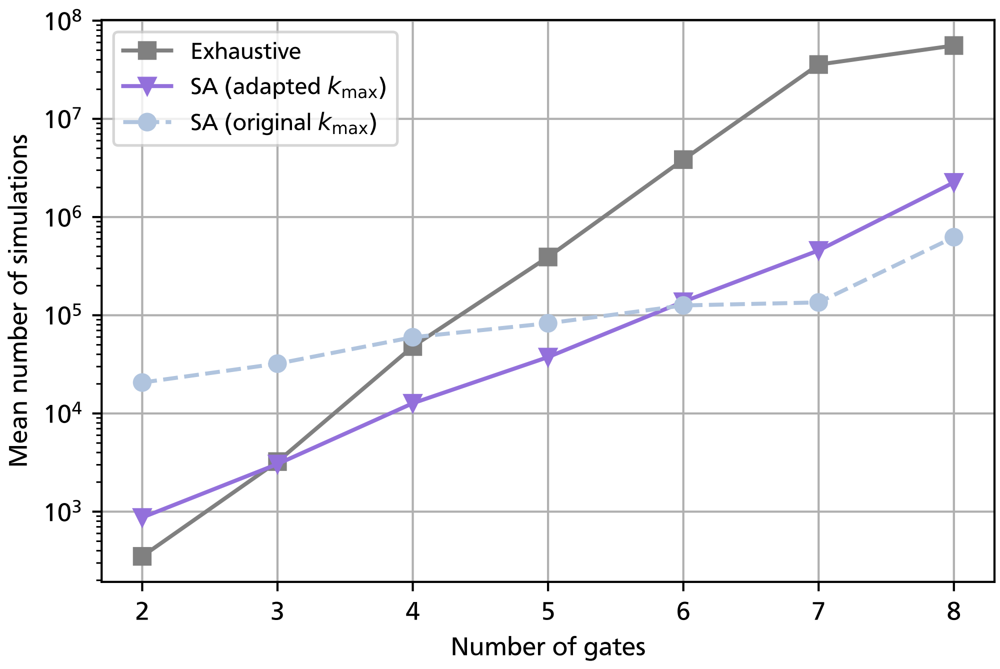


Figure 9.1.: Mean number of simulations by circuit size for mapping with the E-Score with SA (adapted and original $k_{\max}$), compared to an exhaustive search

important observations can be made by comparing these results to the results obtained for mapping with the CELLO score.

First, SA provides a gain in search efficiency and runtime in roughly the same order of magnitude for both scores. In detail, the efficiency gain with the E-Score is reduced by $\sim 42\,\%$ compared to the CELLO score, caused by the adapted inner-loop criterion. However, the speedup is improved by $\sim 19\,\%$. This hints at an increased impact of the circuit size on the simulation runtime for the E-Score since the share of spared simulations is larger for circuits with more gates. This hypothesis is verified by the fact that the mean runtime for both scores is comparable for up to 6 gates and diverges for larger circuits. For example, the E-Score requires up to $3.4\times$the runtime of the CELLO score for circuits with 8 gates.

Second, despite the E-Score with $n = 5000$ samples being approximately $5000\times$computationally more complex than the CELLO score, the absolute measured runtime for exhaustive search increased only by $\sim 64.6\,\%$. This is due to the fact that an

Table 9.2.: Solution quality of SA with the E-Score and the CELLO score

Score	Exact solutions[a]	Mean deviation (%)	Max. deviation (%)
E-Score	60/66	−0.05	−7.9
CELLO	59/66	−0.02	−3.15

[a] Circuits for which SA returned the exact solution in all 10 runs.

optimized version of the circuit simulator has been used, in which parallel computation techniques are consistently used. In Figure 9.1, the mean number of simulations needed with respect to the circuit size in the number of gates is depicted. Besides the results for SA with both the original and adapted $k_{\max}$, the results of an exhaustive search are shown.

Solution Quality

Another important aspect when evaluating heuristic methods is the quality of delivered solutions. To this end, the scores of the SA gate assignment runs have been compared to those obtained by exhaustive search. Table 9.2 shows three metrics for the quality of solutions obtained with SA for the E-Score and the CELLO score, repeated from Section 8.4.3. On the one hand, the heuristic finds the exact solution in all 10 repetitions for one more circuit when the E-Score is used. More specifically, this is the case for $\sim 91\,\%$ of the circuits, compared to $\sim 89\,\%$ for the CELLO score. The circuits for which the optimum could not be found in every run comprise at least 7 gates and thus are a subset of the largest circuits tested. On the other hand, the solutions feature roughly doubled mean and maximum deviations from the optimum score when the E-Score is used.

Conclusion

In conclusion, the results confirm the flexibility of the SA method concerning the used objective function. However, they also highlight SA's highly parameterized nature. On the one hand, changes to the problem structure, e.g., the score function, require adapted parameters to retain the qualities of the method regarding efficiency and solution quality. On the other hand, changes to the parameters can easily be conducted once the problem has been identified. In the given case, the more complex nature of the E-Score compared to the CELLO score leads to a reduced solution quality for large problems. By adapting the inner-loop criterion, the worsened quality could again be improved without changing the scheme fundamentally. The result is a suitable method for the technology mapping for robust genetic circuits.

9.2. Signal-Compatibility of Genetic Gates

Note: Parts of this section have already been published in [24]. Explicit self-citation is omitted to provide an improved reading flow.

The E-Score (Sections 4.3 and 9.1) provides a means to rate the robustness of a complete circuit design. The same holds for the heuristic noise margin check of the GDA tool Cello (Section 6.1), which is detailed in Section 9.2.1. Inspired by this, a way to integrate robustness into constructive gate assignment methods such as B&B (Section 8.3) based on information available in libraries of genetic gates is presented in Sections 9.2.2 to 9.2.4. Instead of simulating completely assigned circuits, local compatibility checks between groups of gates are performed during the assignment optimization. By this, combinations of gates sensitive to signal perturbations caused by cell-to-cell variability can instantly be rejected, making constructing robust circuits more efficient. The method has been devised in cooperation with the Self-Organizing Systems Lab.

9.2.1. Pair-Wise Compatibility

Cello offers a routine for checking the noise margin for pairs of gates in a circuit [53]. It requires the simulation of the circuit characteristic to be completed, as it relies on the values of all signals in the circuit for all Boolean input assignments. Generally, the check ensures that the minimum and maximum output values of a gate lead to an output of the subsequent gate that is higher than half the maximum or lower than twice the minimum output value of that subsequent gate. Formally, denote by $\underline{X}$ the set of input values of the subsequent gate representing a Boolean 0, and by $\overline{X}$ the set of values representing a Boolean 1. Further, let g be the inhibitory Hill function describing the transfer characteristic of said gate and g^{-1} its inverse. Be the output interval of the gate limited by $[y_{\mathrm{min}}, y_{\mathrm{max}}]$. Then, the noise margin criterion introduced used in [53] can be stated formally by the criteria

$$\forall \underline{x} \in \underline{X} : \; \underline{x} < \mathrm{g}^{-1}(\frac{1}{2}\, y_{\mathrm{max}}) \tag{9.1}$$

and

$$\forall \overline{x} \in \overline{X} : \; \overline{x} > \mathrm{g}^{-1}(2\, y_{\mathrm{min}}). \tag{9.2}$$

Both criteria have to be fulfilled for every pair of gates in the circuit, with the preceding gate contributing the input values $\underline{X}, \overline{X}$ and the subsequent gate contributing the set of parameters θ that defines the transfer function g. Note that the criteria can also be formulated for gates based on activation with Hill function h. The method is a post-simulation check and thus is not applicable for avoiding the construction and simulation of circuits that fail the criteria. Further, it considers only pairs of gates, even if the subsequent gate has multiple inputs. By this, it potentially excludes circuit designs that actually fulfill both criteria. Assume a genetic NOR gate with two inputs, x_1 and x_2, defined by its Hill transfer function and input superposition $\mathrm{g}(x_1 + x_2)$. Assume further that input x_1 holds the highest signal value from the set of values for a Boolean 1, i.e.

$$x_1 := \max_{\overline{x} \in \overline{X}} \overline{x}.$$

While criterion (9.2) may be violated when taking into account only x_1 in a pair-wise check, i.e.,

$$x_1 \ngtr \mathrm{g}^{-1}(2\, y_{\mathrm{min}}),$$

it may be fulfilled when taking into account the other input of the gate, x_2, and assuming it to be non-zero, i.e.,

$$x_1 + x_2 > \mathrm{g}^{-1}(2\, y_{\mathrm{min}}), x_2 > 0.$$

The assumption $x_2 > 0$ is valid for genetic logic gates since their basal output promoter activity is generally non-zero. Thus, in a circuit containing multi-input gates, the pair-wise checking of noise margins can lead to the rejection of designs that actually fulfill the criteria.

An alternative approach is devised in the following that considers the compatibility of $(n + 1)$-tuples of gates, depending on the maximum number of inputs n of gates present in the library. By this, false negatives are avoided when rejecting combinations of gates. Further, the method is based solely on information available in genetic gate libraries. Therefore, no simulation of the circuit is needed to apply the criteria. This allows integrating the check into the constructive B&B gate assignment method so that incompatible combinations of gates can be pruned directly in the search tree (Section 9.2.3).

9.2.2. Compatibility of (n+1)-Tuples

The advanced compatibility measure is based on individual threshold points for low and high output values of genetic gates. These thresholds are inspired by the $3\,\mathrm{dB}$ noise margins commonly applied in communication systems and are calculated using the individual output interval of each gate. In a circuit, the thresholds are then used to determine the compatibility of connected gates.

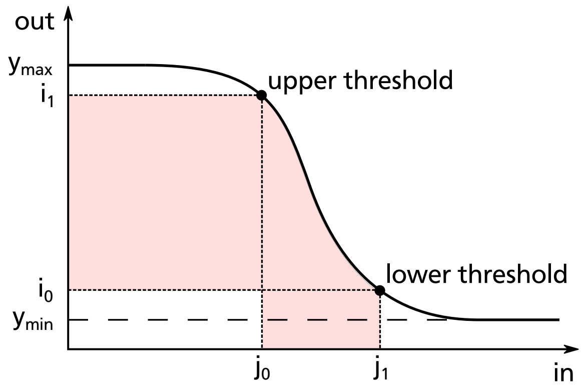


Figure 9.2.: Thresholds applied by the compatibility measure and forbidden regions (red)

First, the calculation of the individual input and output values representing the two threshold points on the transfer function is detailed. All involved variables are depicted in Figure 9.2 with an exemplary parameterization of an inhibitory Hill curve. Let $[y_{\min}, y_{\max}]$ be the output interval of the genetic gate under consideration, then

$$i_1 := y_{\min} + \sqrt{\frac{1}{2}}(y_{\max} - y_{\min})$$

defines the output value of the gate at the upper threshold. This output value corresponds to the input value $j_0 = \mathrm{g}^{-1}(i_1)$, with g^{-1} being the inverse of the inhibitory Hill function. Analog, the output value at the lower threshold is defined as

$$i_0 := y_{\min} + (1 - \sqrt{\frac{1}{2}})(y_{\max} - y_{\min})$$

and it corresponds to the input value $j_1 = \mathrm{g}^{-1}(i_0)$.

The thresholds can be used to define criteria ensuring that a genetic gate never operates in the transition region marked in red in Figure 9.2. These criteria solely use the thresholds of the considered gate and all its input gates. For a given library of genetic gates, the number of input gates is defined by the fan-in of the subsequent gate. Thus, the criteria are generalized for an arbitrary number of gate inputs n, considering groups of n gates contributing input signals and one subsequent gate. These gate groups are subsequently called $(n+1)$-tuples. Denote by $i_{0,k}$ with $k \in [1..n]$ the output values of the gates providing their lower threshold to the subsequent gate and by $j_{0,t}$ the lower input threshold of this subsequent gate. Then, the first criterion

$$\sum_{k=1}^{n} i_{0,k} \leq j_{0,t}$$

ensures that the superposition of the highest possible input values representing a Boolean 0 does not push the target gate over its input threshold j_0 into the transition region. Further, let $y_{\min,k}$ denote the minimum output value of input gates $k \in [1..n]$, then the second criterion

$$\forall l \in [1..n] : i_{1,l} + \sum_{\substack{k=1 \\ k \neq l}}^{n} (y_{\min,k}) \geq j_{1,t}$$

ensures that all minimal input value combinations representing a Boolean 1 do not push the target gate below its input threshold j_1. In each combination of input values, one input gate is assumed to provide the lowest possible value representing a Boolean 1, i.e., i_1, and all other input gates provide their minimal output value, $y_{\min}$. By enforcing these criteria for every gate in the circuit during the construction of a gate assignment, it can be asserted that signal levels in a circuit of cascaded compatible gates do not cross the defined thresholds and thus exhibit a defined perturbation margin. Note that the criteria are formulated for genetic gates based on repression, which represent NOR gates if multiple inputs are present. However, a similar formulation is possible for gates based on activation, e.g., AND gates based on coregulation.

The genetic gate library of CELLO has exemplarily been examined with the presented compatibility criteria. As it features NOR gates with a maximum fan-in of $n = 2$, maximally 3-tuples of gates, consisting of two source gates and one target gate, have to be considered. In total, 21 % of the triples that can be formed by the library feature transfer characteristics which are compatible according to the criteria. Figure 9.3 depicts the results in detail. For a fixed source gate, the number of compatible successor gates varies depending on the second source gate. This highlights the more exact compatibility check compared to a pair-wise approach.

9.2.3. Compatibility Lookahead in Branch-and-Bound

Since the criteria for compatibility are solely based on information available in genetic gate libraries, they can be used to pre-compute a $(n + 1)$-dimensional matrix of Boolean elements that signify if a combination of gates adheres to the criteria. This allows for a straightforward compatibility check of gate assignments during the optimization (Chapter 8). The approach integrates naturally into constructive methods such as the B&B method (Section 8.3). With every branching operation (Section 8.3.1), the gate assignment is

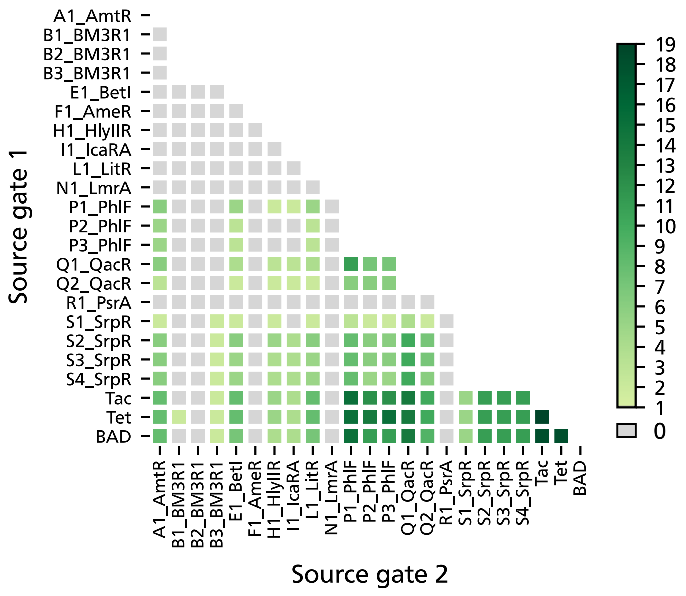


Figure 9.3.: Number of compatible successor gates for each pair of gates from CELLO's gate library

extended by one gate in the reversed topological order of the circuit. From this, a new combination of gates with a predecessor-successor relation arises, which is subject to the compatibility check. Suppose this new combination is deemed to be incompatible. In that case, the new branch of the search tree can instantly be pruned since all refined gate assignments based on this partial assignment will contain the incompatible gates. Further, a more elaborate pruning approach is possible, which performs a compatibility lookahead. It is based on the observation that a choice for assigning a specific gate may be valid in terms of compatibility with its successors but may impede the construction of a complete compatible gate assignment later in the B&B process. Consequentially, this branch may also be pruned to avoid exploring dead-ends in the search tree.

Formulation of the SAT Model

The lookahead approach is formulated as a Boolean satisfiability problem (SAT) [9]. This allows the application of specialized SAT solvers online during the branching operation to decide whether the branch should be kept or pruned. The problem consists of four clauses, one of which is optional. Let $g \in G$ be a gate from the set G of all abstract gates in the circuit topology. Further, let $\theta \in R \subset \mathcal{L}$ be a specific genetic gate represented by its set of parameters θ, grouped in R with gates using the same TF and being part of the gate library $\mathcal{L}$. Then $a_{\theta \to g} \in \mathbb{B} = \{0, 1\}$ is the Boolean variable indicating the mapping of implementation θ to gate g. Let further $s \in \mathcal{S}$ be a logic type from the types $\mathcal{S}$ present in the library and $\mathrm{t}(g) : G \to \mathcal{S}$, $\mathrm{t}(\theta) : \mathcal{L} \to \mathcal{S}$ the functions mapping gates and implementations to their logic type, respectively. Then the first clause of the SAT problem

$$\forall g \in G, \forall \theta \in \mathcal{L} : \exists a_{\theta \to g} : a_{\theta \to g} = 1 \wedge t(g) = t(\theta)$$

ensures that every gate in the circuit is assigned at least one specific gate implementation of the same type. Further, the second clause

$$\forall R \subset \mathcal{L} : |\{\forall \theta \in R, \forall g \in G : a_{\theta \to g}\} = 1| \leq 1$$

states that every TF must be part of maximally one assignment, avoiding direct crosstalk in the circuit. Let furthermore n be the maximum number of inputs of any gate in the library. Then $(\theta_1, \theta_2, \ldots, \theta_n, \theta_t) \in T$ represents an $(n+1)$-tuple of n specific gates providing the inputs of gate θ_t from the set T of such tuples in the circuit. Denote by $\mathrm{c}(\theta_1, \theta_2, \ldots, \theta_n, \theta_t) : \mathcal{L}^n, \mathcal{L} \to \mathbb{B}$ the function that queries the compatibility matrix for a given $(n+1)$-tuple of gates. Then, the third clause

$$\forall (\theta_1, \theta_2, \ldots, \theta_n, \theta_t) \in T : c(\theta_1, \theta_2, \ldots, \theta_n, \theta_t) = 1$$

ensures that every tuple present in the circuit adheres to the compatibility criteria. The final clause

$$\forall g \in G, \forall \theta \in \mathcal{L} : \exists a_{\theta \to g} = 1 \wedge \forall \tau \neq \theta \in \mathcal{L} : \nexists a_{\tau \to g} = 1$$

states that every gate in the topology is assigned maximally one specific genetic gate. It is optional for evaluating the compatibility but can be useful as it enables retrieving valid assignments from the SAT model. To speed up the evaluation of the SAT problem, it is set up before the technology mapping process based on the circuit structure and the library of gate implementations. When it is used to evaluate a (partial) assignment, only the values of the variables a have to be set according to the existing assignment.

Refinement of the Exact Mode

As shown in Section 8.3.4, the B&B method requires the substitution of invalid bounds for a guaranteed exactness. In the B&B scheme, signals representing a Boolean 0 are assumed to be lower bounds to the values possible at the respective site. Thus, for a superposition at a multi-input gate involving such a signal to represent a valid upper bound, the value

Table 9.3.: Refined substitution of invalid bounds based on compatibility criteria

x_1	x_2	$x = x_1 + x_2$	substitution
0	0	0	-
1	0	1	$x_2 \rightarrow i_{0(,\mathcal{C})}$
0	1	1	$x_1 \rightarrow i_{0(,\mathcal{C})}$
1	1	1	-

has to be substituted with the highest possible signal at this site. For the lack of any other valid upper bound, this is $y_{\max}$ for an assigned gate. If it is not assigned, this is the highest output possible, i.e., $y_{\max,\mathcal{C}}$, of all candidate gates $\mathcal{C} = \{\theta \in \mathcal{L} : \theta \notin a\}$ left in the library, with a being the gate assignment.

However, if an assignment of genetic gates adheres to the presented compatibility criteria, all signals are guaranteed to operate within the defined ranges. As this includes a maximum output value for signals representing a Boolean 0, i.e., i_0, the substitution can be refined as shown in Table 9.3 for the superposition of two inputs. Analog to the approach for substitution without compatibility criteria, the maximum guaranteed output of the specific gate, i.e., i_0, or the maximum guaranteed output of all gates being candidates for the assignment, i.e., $i_{0,\mathcal{C}}$, is chosen as a substitute.

9.2.4. Implications for the Gate Assignment Search

The implications of applying the presented compatibility criteria for the gate assignment process are evaluated in the following. Analog to Section 8.4, the set of 66 circuit topologies is assigned the genetic gates of Cello's library. The process has been carried out with the B&B method (Section 8.3) using the SAT-based compatibility lookahead and refined estimator for the exact mode (Section 9.2.3). For comparison, an exhaustive search has been carried out, also with enforced compatibility criteria.

For one of the 66 circuit topologies, which contains 5 NOR gates and 2 NOT gates, no gate assignment that adheres to the compatibility measure exists. This visualizes the strictness of the criteria and represents a difference to the E-Score-based optimization. Both methods are compared in detail in Section 9.3. Table 9.4 shows the gate assignment results regarding search efficiency and runtime. While the upper two rows correspond to the usage of the compatibility criteria, the lower two rows are repeated from Section 8.4.3 to allow for a comparison to the gate assignment without applying the criteria. Considering the exhaustive search results, the enforced compatibility dramatically reduces the space

Table 9.4.: Search efficiency and runtime of exact Branch-and-Bound (B&B) and exhaustive search (Exh.) with and without enforced compatibility

		Search Efficiency			Runtime	
Method	Compatibility	Simulations	Gain	Trav. (%)	Runtime (h)	Speedup
B&B	✓	142 584	1.4×	73.3	5.0	83.3×
Exh.		194 591	1.0×	100.0	416.6	1.0×
B&B	-	34 039 365	15.4×	6.5	505.2	19.7×
Exh.		524 937 849	1.0×	100.0	9962.3	1.0×

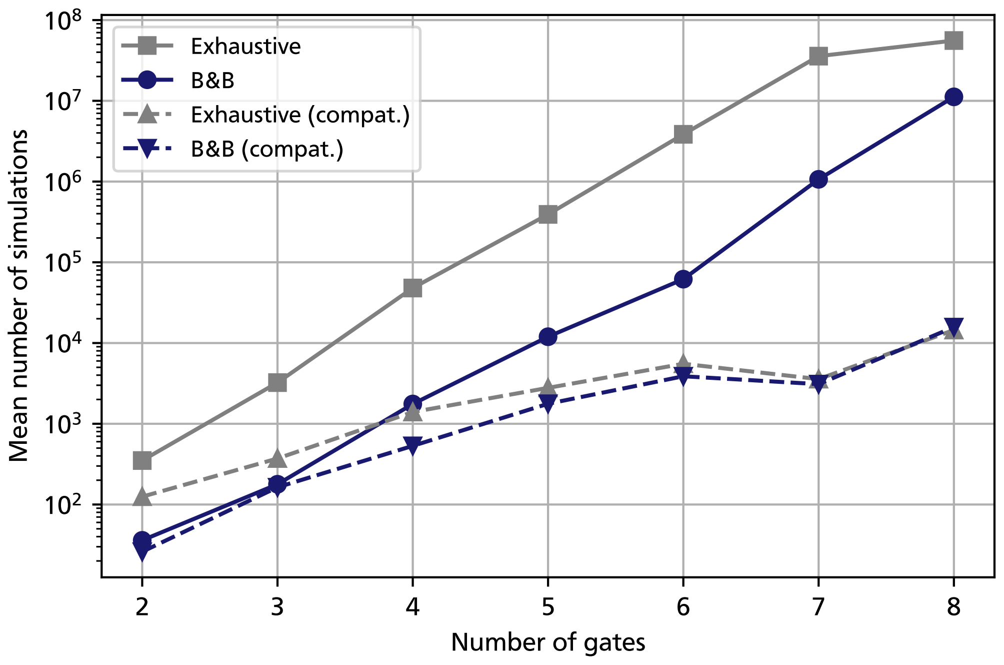


Figure 9.4.: Mean number of simulations by circuit size for exhaustive search and exact B&B, with and without enforcing the compatibility criteria

of valid gate assignments $\sim 2.7 \times 10^3\times$. Given this reduced search space, B&B exhibits a reduced $1.4\times$gain in efficiency when the compatibility criteria are used, corresponding to a traversal of $73.3\,\%$ of the search space.

Note that the absolute numbers in runtime are not comparable between the rows with compatibility and those without since another version of the circuit simulator has been used for the former, based on a thermodynamic circuit model (Chapter 10). However, the depicted speedups are valid, as they refer to the corresponding exhaustive search, which has been carried out with the same simulator. While the search efficiency of B&B decreases, the speedup increases considerably to $83.3\times$when the compatibility criteria are used. This diverging behavior is caused by the lookahead pruning implemented in B&B. While the exhaustive search traverses the whole search space of $\sim 5.2 \times 10^8$ assignments and determines the compatibility of the genetic gates for each of the assignments, the lookahead pruning of B&B avoids the compatibility evaluation of large parts of the search space. Figure 9.4 shows the mean number of simulations for the four gate assignments methods with respect to the number of gates in the circuit. With enforced compatibility criteria, shown by the dashed lines, the reduced efficiency advantage of B&B in circuit simulations becomes visible.

9.3. Comparison of the Methods

After the isolated evaluation of both methods for integrating robustness into technology mapping in Sections 9.1 and 9.2.4 regarding efficiency, runtime, and solution quality, they are compared in this section. First, efficiency and runtime are compared, followed by a comparison of the mapping results regarding the central objective in this chapter, robustness.

Search Efficiency and Runtime

Table 9.5 repeats the main results of assigning Cello's gates to the set of 66 circuit topologies with SA and the E-Score as an objective function (Section 9.1), as well as with B&B using the compatibility criteria and the Cello score as objective function (Section 9.2.4). While SA requires $\sim 11.6 \times 10^6$ circuit simulations to assign gates to all topologies, B&B needs $\sim 14.3 \times 10^4$. This equals a gain in search efficiency of B&B compared to SA of $\sim 81\times$. The improved traversal of the search space translates to a runtime of $5\,\mathrm{h}$ of B&B, compared to $\sim 129\,\mathrm{h}$ of SA. Note, however, that the runtimes have been measured with different versions of the circuit simulator. While for SA and the E-Score, the classical simulator has been used, B&B with compatibility criteria has been evaluated using the

Table 9.5.: Number of simulations and runtime of SA using the E-Score and B&B using the compatibility criteria

Method	Number of simulations	Runtime (h)[a]
SA with E-Score	11 580 687	128.7
B&B with compatibility	142 584	5.0

[a] Measured with different versions of the circuit simulator and thus allowing a qualitative comparison.

thermodynamic simulator. Since the thermodynamic simulator performs more complex computations than the traditional one, the shown runtime of B&B represents a pessimistic upper bound. Nonetheless, the runtime advantage of B&B, even in this pessimistic case, highlights its improved efficiency compared to SA.

Robustness

The quantitative comparison of the robustness of circuit designs generated by both methods requires a common metric. To this end, the E-Score is chosen, as it represents a measure for both adherence to the functional specification and robustness, grounded on a statistical evaluation of the circuit (Section 4.3). Since the presented SA method optimizes designs directly for the E-Score, the resulting score values can immediately be used for the comparison. However, the B&B method optimizes circuits with respect to the Cello score and introduces robustness heuristically by constraining the search space with the compatibility criteria. Thus, the resulting designs are re-scored with the E-Score, allowing for a comparison to SA's results. Figure 9.5 summarizes this comparison for the 65 circuits that could be mapped using the constrained B&B method. It categorizes each circuit topology assigned with B&B regarding its relative deviation of E-Score compared to the same circuit mapped with SA. The worst E-Score encountered in any of the 10 runs performed with SA has been chosen for every circuit. For 8 circuits, assigning gates using the compatibility criteria and B&B leads to the same E-Score as using SA. Further, 26 circuits exhibit a minor relative deviation of up to −10 %. A deviation in the range of −30 % to −10 % can be observed for another 22 circuits. Finally, 9 circuits represent outliers with a worse deviation than −30 %, two of which exhibit the measured worst-case of ∼−70 %.

Figure 9.6 exemplifies the output characteristic of two resulting circuit designs for one topology. It consists of 7 NOR gates and implements function ϕ_{15} defined in Table A.1. In Figure 9.6a, the characteristic of the circuit mapped with SA and the E-Score is

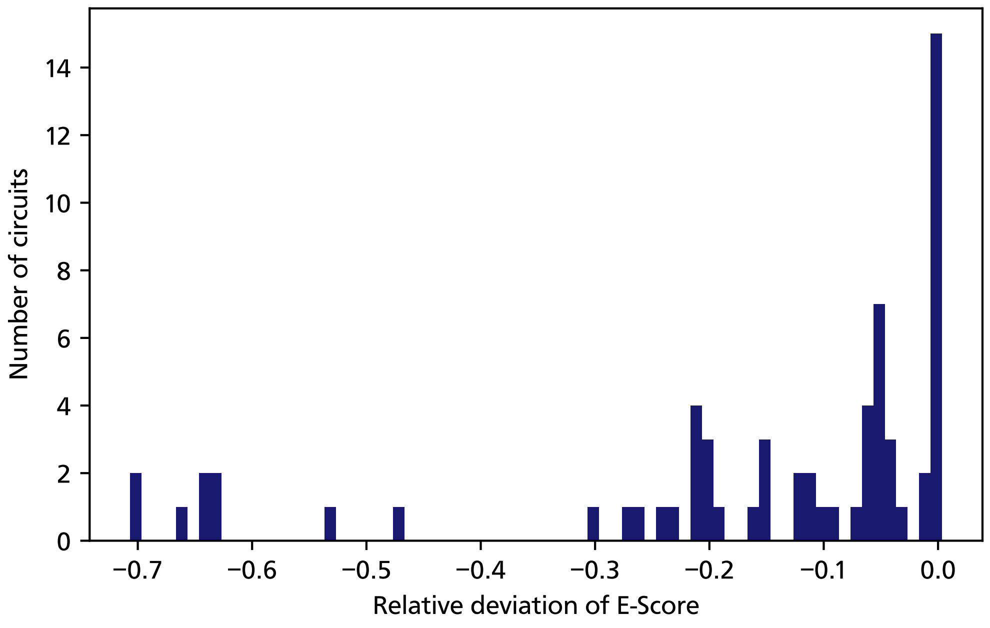


Figure 9.5.: Relative deviation of E-Score of circuits assigned with B&B and compatibility criteria compared to SA with E-Score

shown. Figure 9.6b depicts the characteristic of the circuit mapped with B&B, the Cello score and the compatibility criteria. The SA-circuit achieves an E-Score of 141.1 and the B&B-circuit an E-Score of 41.4 (rescored, originally a Cello score of 75.5). Thus, the B&B-circuit features a relative deviation of $\sim -70\,\%$ in E-Score, representing one of the outliers in Figure 9.5. Both circuit designs have in common that their respective output characteristics do not feature an overlap of the Boolean 0 and 1 states. This represents a key feature of robust circuit designs. However, the separation of output distributions is higher for the design generated with SA and the E-Score. This is especially caused by the distribution representing the Boolean 1 state, which is shifted by approximately one order of magnitude. Thus, both designs are robust regarding non-overlapping output distributions. However, the circuit optimized directly for the E-Score features a higher margin for unforeseen perturbations to the circuit characteristic, e.g., caused by cell-to-cell variability. This highlights the heuristic nature of the compatibility criteria. On the one hand, they avoid overlapping output distributions by introducing perturbation margins

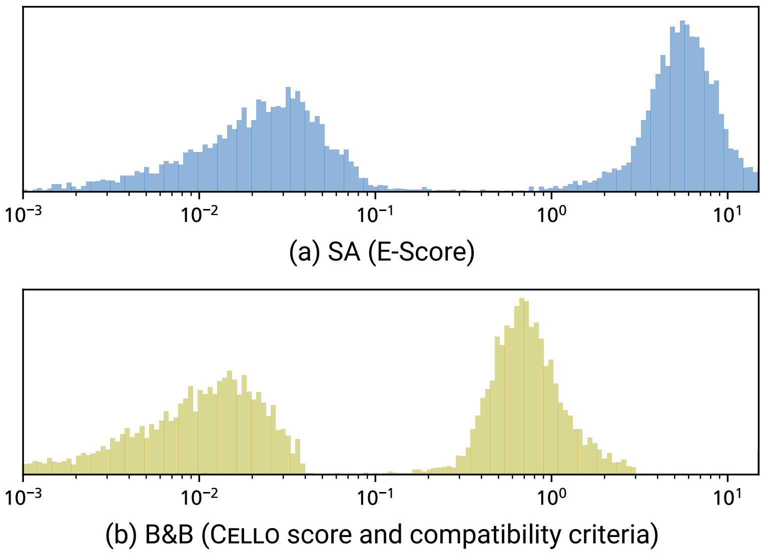


Figure 9.6.: Output characteristics of circuits for function ϕ_{15} assigned with different methods for robustness

locally among gates. On the other hand, they are based on strict thresholds, limiting the optimizer's opportunities to cross these thresholds and explore solutions that represent good trade-offs.

Conclusion

In this section, two methods for the technology mapping for robust genetic circuits have been compared. One method leverages the flexibility of the SA gate assignment scheme concerning the used objective function and optimizes circuits for the E-Score. The other constrains the search space for gate assignments by enforcing perturbation margins, i.e., signal compatibility, among gates in the circuit and is integrated into the B&B gate assignment method. Its efficiency in traversing the search space and application of the computationally simple Cello score results in an at least $25.7\times$advantage in runtime compared to the method based on the E-Score. For 34 of the 66 tested circuits, a comparable quality in E-Score is obtained using the compatibility heuristic. Further, an outlier for which the B&B method yielded a $70\,\%$ worse E-Score compared to SA was analyzed in detail. It was shown that both designs do not exhibit overlapping output distributions. However, the circuit optimized directly for the E-Score features a larger separation of output distributions, i.e., an increased margin for potential perturbations. In summary, the compatibility criteria can enforce non-overlapping output distributions and constrain the search space efficiently in that regard. In approximately half of the tested cases, the designs obtained in this way are comparable to those obtained by direct optimization for the E-Score. However, the results for the other half hint at the exclusion of good solutions by the strict, local criteria.

10. Context-Aware Technology Mapping

In synthetic genetic circuits, context effects can lead to undesired interactions of parts of the circuit with each other or the host organism (Section 4.4). If the circuit model does not consider these effects, its predictive power is limited. Using such a limited model in a technology mapping process may lead to non-optimal designs, as the context the design is embedded into is ignored during its optimization. The thermodynamic circuit model presented in Section 4.4 allows taking into account two important context effects directly related to gene regulation, i.e., titration and crosstalk. Titration modulates the transfer characteristic of genetic gates but has a minor impact on the technology mapping process. Crosstalk, however, introduces additional signal paths and thus affects the problem structure. The implications of both effects are detailed in Section 10.1. The introduction of crosstalk disrupts some of the assumptions on which the B&B gate assignment method (Section 8.3) is based. The adaptation of the method to these new conditions is presented in Section 10.2, aiming to transfer the efficient and exact technology mapping method to circuits under crosstalk. First, the paradigm of optimistic input interval composition (Section 8.3.4) is extended to *crosstalk environments* in Section 10.2.1. A refinement of crosstalk estimation in B&B's heuristic mode is detailed in Section 10.2.2. Finally, the effect of assumed crosstalk on the functionality of circuit designs and the efficiency of B&B is evaluated in Section 10.2.3.

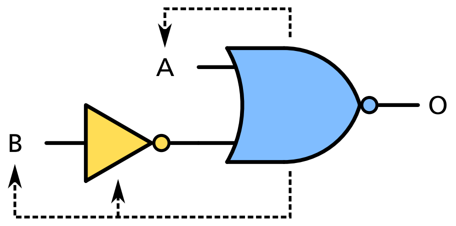
10.1. Implications of Context Effects

The effects of titration and crosstalk on the transfer function of genetic gates have been introduced in Section 4.4. The implications arising from this for technology mapping are detailed in the following.

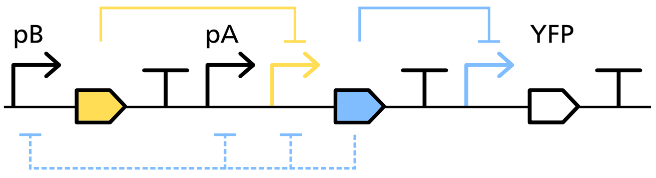
Titration As depicted in Figure 4.6b for a gate based on repression, titration modulates the low-input regime of the function. This is caused by the binding of TFs to non-cognate binding sites, e.g., at the host genome, which is undesired in the context of the circuit function. By this, the availability of the TF as a functional component of the circuit is reduced, leading to higher required TF levels to regulate the gate. This can be modeled as a static effect since it depends solely on the binding sites present in the circuit's environment. Thus, it can be computed prior to technology mapping using the thermodynamic model and information of the gate library and environment. The result is a modified gate library with modulated transfer characteristics reflecting the current titration in the circuit's environment. The problem structure of gate assignment, as well as important features of the transfer curves leveraged in the methods presented in Chapter 8, i.e., monotonicity and bounded output interval (Section 3.2), remain unchanged. Consequentially, titration can be integrated into the presented technology mapping methods by simply using an adapted gate library.

Crosstalk The binding of functional TFs at non-cognate promoters that are part of the circuit leads to the effect of crosstalk. Again, this is reflected in a modulation of the transfer curve (Figure 4.6b). The gate, which is subject to crosstalk, exhibits an undesired regulation, potentially reducing the output in the low-input regime in the case of repression. From this *gate-to-gate* crosstalk, two significant consequences follow. First, its effect is dynamic since the degree of regulation of non-cognate promoters scales with the abundance of the crosstalk-inducing TF. Thus, its influence on the circuit characteristic cannot be reflected statically in the library but must be determined during circuit simulation. Second, it introduces regulatory cycles in the circuit since a TF can potentially exert crosstalk on promoters belonging to preceding gates. This is exemplified in Figure 10.1 for a simple circuit consisting of a NOT gate and a NOR gate. Figure 10.1a depicts the additional signal paths that are introduced by potential crosstalk of the NOR gate's TF as dashed lines. If any of these three crosstalk-relations are present, the TF's concentration or flux is subject to a cyclic dependency. This is shown in more detail in Figure 10.1b, using the SBOL Visual representation of the circuit. The dashed line below the diagram depicts the repressing effect of the TF on both primary input promoters and the NOT gate's output promoter. When repressing the NOT gate's output promoter or pA, the TF attenuates its own synthesis, while it increases it when repressing pB. Note that also the TF of the NOT gate can potentially induce crosstalk at all promoters present in the circuit.

Like titration, crosstalk does not affect the fundamental properties of monotonicity and bounded output of genetic gates' transfer functions. In principle, the presented optimization methods for gate assignment are still applicable if crosstalk is present. However, the possible introduction of arbitrary interactions in the circuit necessitates an extension of the optimistic input interval composition of the B&B method (Section 8.3.4). In its basic version, it only considers inputs wired in the circuit diagram, i.e., desired interactions, and estimates the values of these input signals optimistically with respect to the circuit



(a) Additional signal paths



(b) Additional regulatory relations

Figure 10.1.: Potential crosstalk exerted on non-cognate promoters exemplified by the NOR gate's TF

score. To support crosstalk, the contribution of other TFs must also be estimated optimistically. To this end, *beneficial* and *detrimental* crosstalk must be distinguished, and only the former has to be included in favor of an optimistic estimation. Further, potential cyclic dependencies of signals must be opened up, and upper bounds for signals subject to cycles must be found. This is detailed in Section 10.2. Lastly, the presence of cycles makes the determination of the circuit characteristic, i.e., its operating point in steady state, a root finding problem. Thus, the simulation of circuits with crosstalk is potentially much more computationally complex than the crosstalk-free case.

10.2. Context-Aware Branch-and-Bound

Note: Parts of this section have already been published in [24]. Explicit self-citation is omitted to provide an improved reading flow.

As marked out in Section 10.1, genetic gate transfer characteristics are monotonic and bounded, even if titration and crosstalk are considered. Thus, the propagation of optimistic input values through the circuit applied in the B&B method (Section 8.3.4) to optimistically estimate the CELLO score of a partial assignment is generally feasible when considering

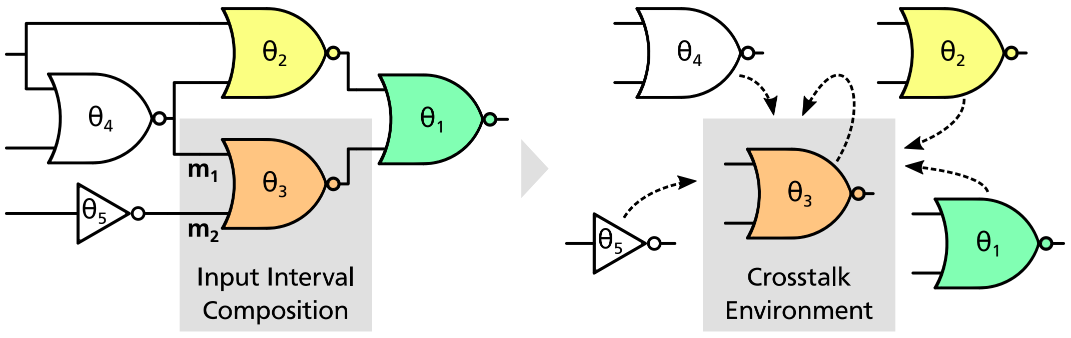


Figure 10.2.: Generalization of the optimistic input interval composition to crosstalk environments

context effects. However, the estimation has to be adapted since the inputs of a gate are not only given by the desired connections shown in the wiring diagram but potentially by all gates present in the circuit. Furthermore, the introduction of cyclic dependencies breaks the assumption of the undisturbed propagation of optimistic values through the circuit. Potentially, an assumed optimistic output value at any site of the circuit can deteriorate the circuit characteristic by inducing crosstalk at other gates. Thus, these cycles have to be handled by an estimator to allow for an optimistic estimation and eventually guarantee the exactness of the method. In the following, the adaptation of the estimator is presented, which generalizes the optimistic input interval composition approach to crosstalk environments.

10.2.1. Crosstalk Environments

In the optimistic input interval composition approach for estimation presented in Section 8.3.4, a partial assignment of genetic gates to a circuit topology is embedded into an optimistic complete solution. This principle is visualized in Figure 10.2 on the left. In the shown circuit of 5 gates, gates θ_1 to θ_3 are assigned specific genetic gates, and gates θ_4 and θ_5 are unassigned. This represents an exemplary partial solution to the gate assignment problem from B&B's search tree (Section 8.3.1). A set of optimistic input values is then assumed for each input into the sub-circuit defined by the partial assignment. In the example, this is marked out exemplarily for inputs m_1 and m_2 of gate θ_3. Based on the gates left in the library, the minimal possible input value is assumed if the signal represents a Boolean 0, and the maximal possible value is assumed if it represents a Boolean 1. In the substitution step of invalid bounds, the considered gate's desired Boolean output value is considered when composing its input values during the estimation to guarantee optimistic values and exactness of the search. Then, the effect of the gate's TF on its cognate output promoter is estimated by applying the gate's transfer function to the estimated input promoter activities.

This composition of input values for signals defined in the circuit schematic, i.e., desired interactions, has to be generalized to a global view to estimate the quality of partial assignments if crosstalk is present. In the thermodynamic circuit model, no qualitative difference exists in the relation of TFs to cognate and non-cognate promoters. Merely, every TF potentially binds to all promoters, and each relation is characterized quantitatively by an individual binding affinity. Thus, every TF present in the circuit must be considered when estimating a gate's output value since it potentially regulates the considered gate. This is implied on the right side of Figure 10.2. Note that all gates in the circuit potentially contribute to the crosstalk environment of gate θ_3, including θ_3 itself. Consequentially, the crosstalk environment of a gate consists of all possible contributions of other gates, which benefit its desired output value concerning the CELLO score. In the following, the crosstalk-environments for a desired Boolean 0 and 1 output of a gate are called *0-environment* and *1-environment*, respectively. Concisely, for each environment, an optimistic input interval composition and estimation of produced TF has to be carried out for all gates in the circuit to estimate their effect on the output promoter of the considered gate. This process is detailed in the following.

Definitions

Let θ denote an individual genetic gate with a unique TF and cognate promoter, and $\mathcal{L}$ a library of gates, i.e., $\theta \in \mathcal{L}$. In addition to the previous model, denote by $\mathrm{b} : \mathcal{L} \times \mathcal{L} \to \mathbb{R}$ the function retrieving the binding affinity of one gate's TF to another gate's cognate promoter. For example, $\mathrm{b}(\theta_2, \theta_3) > 0$ implies a non-zero binding affinity of the yellow gate's TF to the output promoter of the orange gate in Figure 10.2. Let further $m \in \mathbb{N}$ be the number of gates in the considered circuit. Let the target gate, for which the crosstalk-environment shall be composed, be called θ_j with $j \in [1..m]$ and all other gates, i.e., source gates, be called θ_i with $i \in m, i \neq j$. To calculate the effect of one gate on another gate, the thermodynamic circuit model requires two values. The first component is the total activity of the source gate's input promoters $c \in \mathbb{R}$, which defines the quantity of produced TF. The second component is the binding affinity of the produced TF to the target gate's output promoter $b \in \mathbb{R}$. In the following, this tuple $\mathbf{c} = (c, b)$ is called the *contribution* of a source gate to the target gate's crosstalk environment. The 0-environment of a gate is composed of all contributions that benefit its 0-output, and the 1-environment is composed of all contributions that benefit its 1-output. In the case of gates based on repression detailed

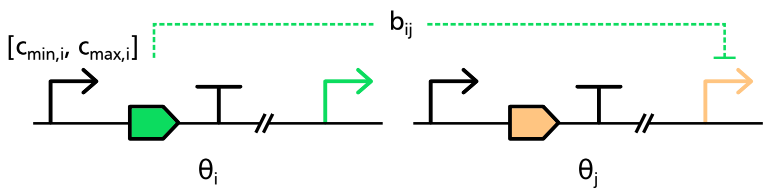


Figure 10.3.: Crosstalk exerted by assigned gates and components of their contribution to the crosstalk environments

below, the contributions that constitute the 0-environment represent an upper bound on the repressing effect of the source gates. Vice versa, the contributions in the 1-environment represent a lower bound on their exerted repression.

Contribution of Assigned Gates

In the following, the contribution of gates that are part of the partial assignment, i.e., are assigned a specific genetic gate, is detailed. This setup is visualized for a pair of source gate θ_i and target gate θ_j in Figure 10.3. First, lower and upper bounds for the total input promoter activity of θ_i must be found to estimate the minimum and maximum quantities of produced TF. For every input of θ_i, it is differentiated whether the identity of the input promoter is defined, i.e., the gate serving as an input is assigned or not. If the input promoter is known, its basal and maximum activity, i.e., $[y_{\min}, y_{\max}]$, provide valid bounds. If the input promoter is of unknown identity, these bounds have to be determined by finding the minimum and maximum promoter activity in the set $\mathcal{C} = \{\theta \in \mathcal{L} : \theta \notin a\}$ of candidate gates left in the library, with a the current partial assignment. The resulting extremal combined input promoter activities of gate θ_i, called $c_{\min,i}$ and $c_{\max,i}$, are obtained by summing up the minimum and maximum values of all its input promoters, respectively. In the 0-environment of θ_j, $c_{\max,i}$ is assumed since the highest possible input promoter activity of gate θ_i leads to the highest quantity of produced TF and thus to the most impactful repression of gate θ_j. Vice versa, $c_{\min,i}$ is assumed in the 1-environment. Since $c_{\min,i}$ and $c_{\max,i}$ represent extremal values rooted in the characteristics of transcriptional regulation, the estimation of induced crosstalk is also extremal. Further, the estimation is not based on activity values obtained during the circuit simulation and thus eliminates the cyclic dependencies introduced by crosstalk.

The second component necessary to estimate the effect of the source gate on the target gate is the binding affinity of the crosstalk-inducting TF to the target gate's output promoter. Since both gates are assumed to be assigned a specific genetic gate, the affinity can be retrieved by querying the library, i.e., $b_{ij} := \mathrm{b}(\theta_i, \theta_j)$. The second column of Table 10.1

Table 10.1.: Contributions of source gates to the target gate's crosstalk environments for exact B&B

	Source gate θ_i	
Target gate θ_j	Assigned	Unassigned
0-environment	$(c_{\max,i}, b_{ij})$	$(c_{\max,i}, b_{\max,j})$
1-environment	$(c_{\min,i}, b_{ij})$	$(c_{\min,i}, b_{\min,j})$

summarizes the contributions for both the 0-environment and the 1-environment.

Contribution of Unassigned Gates

If gates are not part of the partial assignment, i.e., not assigned a specific genetic gate, the identity of the gates' expressed TF is unknown. This setup is visualized in Figure 10.4. To estimate their contribution to the crosstalk environment, lower and upper bounds for the binding affinity to the output promoter of θ_j must be found. Based on the set of candidate gates $\mathcal{C}$ for an assignment to θ_i, the minimum affinity $b_{\min,j}$ and the maximum affinity $b_{\max,j}$ can be found by querying the library, i.e.,

$$b_{\min,j} := \min_{\theta_i \in \mathcal{C}} \mathrm{b}(\theta_i, \theta_j),$$

$$b_{\max,j} := \max_{\theta_i \in \mathcal{C}} \mathrm{b}(\theta_i, \theta_j).$$

The maximum affinity is assumed in the 0-environment of θ_j, and in its 1-environment, the minimum affinity is assumed (Table 10.1, third column). The input promoter activity of unassigned gates is determined analog to assigned gates. Since the gates serving as input to unassigned gates are also unassigned in a partial solution of B&B, the input promoter activity must always be estimated based on $\mathcal{C}$.

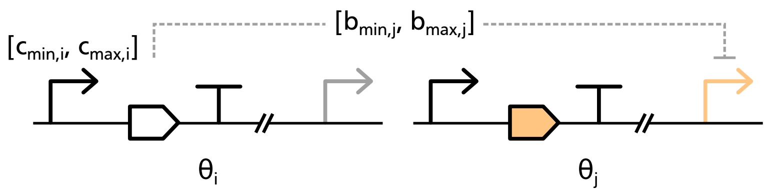


Figure 10.4.: Crosstalk exerted by unassigned gates and components of their contribution to the crosstalk environments

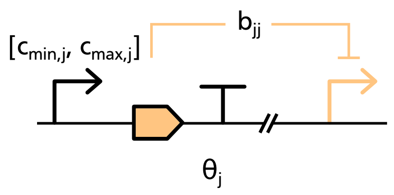


Figure 10.5.: Contribution of the target gate itself to its crosstalk environments

Contribution of the Gate Itself

The target gate θ_j contributes to the regulation of its output promoter via its own TF (Figure 10.5). Since the crosstalk environments are composed only for gates part of the partial assignment, the identity of their TF is known. Thus, their contribution can be composed analog to other assigned gates. That is, the binding affinity is given by the affinity of their TF to its cognate promoter, i.e., $b_{jj} := \mathrm{b}(\theta_j, \theta_j)$. Further, if the gate inputs are sourced by unassigned gates, the minimum and maximum activities of the gate's input promoters are given by the extremal output values of gates left in the library. However, a significant difference applies in the case of inputs sourced by assigned gates. Then, the input values stem from gates, which are also part of the partial assignment subject to the estimation. Thus, the simulated output values of these gates are forwarded and assumed as input promoter activity of the target gate. By this, the optimistically estimated values are propagated through the cascade of logic gates, similar to the estimation presented in Section 8.3.4. This includes substituting invalid bounds, in which valid bounds replace input values obtained by simulation that violate the optimistic estimation. For completeness, the contributions of the target gate itself are given in Table C.1.

Application for Estimation

For estimating the quality of partial solutions during B&B if crosstalk is present, each gate is embedded into its individual crosstalk environment. First, the desired Boolean output value of the considered gate is determined based on the functional specification of the circuit and the currently simulated Boolean input assignment. Then, either the 0-environment or the 1-environment is composed by superimposing all contributions to the regulation of the gate's output promoter. As presented above, this includes contributions of unassigned gates not contained in the partial assignment, other assigned gates of the partial assignment, and the gate itself. Based on the superimposed contributions, the output value of the gate is determined. Analog to the estimation presented in Section

8.3.4, but considering potential crosstalk, the estimation is optimistic with respect to the CELLO score. That is, values representing a Boolean 0 are lower bounds to values possible at the gate's output in refined partial assignments, and values representing a Boolean 1 are upper bounds.

10.2.2. Compatibility and Heuristic Mode

Analog to the basic B&B scheme that does not consider crosstalk, the compatibility criteria presented in Section 9.2 can be applied in the crosstalk-aware B&B. This allows for optimizing circuit designs that exhibit context effects with a robustness constraint. Additionally, the information provided by the compatibility criteria can be used to refine the crosstalk estimation. While the refinement presented in Section 9.2.3 for the basic B&B is applicable for B&B's exact mode, the refined estimation of crosstalk detailed in the following is applicable for the heuristic mode. It requires a differentiation between the output states of the source gates, which exert crosstalk. Thus, besides differentiating the target gate's 0-environment and 1-environment, the environment in which the source gate is assumed based on the circuit function must be differentiated.

Refinement for Assigned Gates

First, assume the source gate θ_i to be part of the partial assignment, i.e., assigned a specific genetic logic gate. Then, its contribution to the crosstalk environments of target gate θ_j can be refined based on the compatibility thresholds j_0 and j_1 calculated for every gate (Section 9.2.2) for two of four combinations of the target gate's and the source gate's assumed Boolean outputs. Table 10.2 shows the four combinations resulting from each gate being in one of its crosstalk environments. If the target gate θ_j is assumed in its 0-environment, maximum repression by crosstalk must be estimated. If, additionally, the source gate θ_i is in its 1-environment, its maximum input promoter activity can be estimated more realistically by $j_{0,i}$ instead of $c_{\max,i}$, reducing the estimated crosstalk. The second possibility for refinement arises if θ_j is in its 1-environment and θ_i in its 0-environment. Then, minimum crosstalk has to be assumed to estimate the output optimistically. Analog to the first case, the minimum input of θ_i in its 0-environment can be estimated by $j_{1,j}$ instead of $c_{\min,i}$, potentially increasing the estimated crosstalk. Note that these refinements are only applicable in B&B's heuristic mode, even if compatibility of gates is enforced, because j_0 and j_1 do not represent valid thresholds if crosstalk is present.

Table 10.2.: Contributions of assigned source gates to the target gate's crosstalk environments for heuristic B&B

	Source gate θ_i	
Target gate θ_j	0-environment	1-environment
0-environment	$(c_{\max,i}, b_{ij})$	$(j_{0,i}, b_{ij})$
1-environment	$(j_{1,i}, b_{ij})$	$(c_{\min,i}, b_{ij})$

Refinement for Unassigned Gates

If the source gate is unassigned, i.e., not part of the partial assignment, its contribution to the crosstalk environment can be refined similarly. However, the identity of its TF and thus the binding affinity of that TF to the target gate's promoter is unknown. Thus, it has to be estimated based on the gates left in the library analog to Section 10.2.1. The resulting contributions are given in Table C.2 for completeness.

10.2.3. Evaluation

To evaluate the impact of crosstalk on technology mapping and predicted performance of genetic logic circuits, a set of gate assignment runs has been carried out for the 66 circuit topologies introduced in Section 8.4. For Cello's [53] library of genetic gates, 35 different crosstalk configurations have been generated and supplied by the Self-Organizing Systems Lab. Then, context-aware gate assignment has been carried out using an exhaustive search, exact B&B (Section 10.2.1) and heuristic B&B (Section 10.2.2) as optimization methods, the Cello score (Section 4.2) as objective function and the thermodynamic circuit simulator (Section 4.4). To make the execution of the high number of gate assignment runs with different crosstalk configurations feasible, the compatibility criteria (Section 9.2.2) have been applied to constrain the search space. The exhaustive search results have been used to successfully verify the exactness of the B&B method under crosstalk. In the remainder of this section, the used crosstalk configurations are detailed, and the impact of crosstalk on the technology mapping process and circuit performance is analyzed.

Assumed Crosstalk Configurations

The assumed crosstalk configurations used for evaluation have been generated randomly, based on two parameters characterizing crosstalk distribution across a library of genetic gates [24]. For each promoter in the library, random values have been sampled from a

Dirichlet distribution for each non-cognate TF. From these values, normalized relative binding affinities for non-cognate TFs have been obtained relative to the binding affinity of the cognate TF. One concentration hyperparameter is used for controlling the distribution, determining how sparse the random crosstalk is distributed across the non-cognate TFs. The second parameter controls the overall intensity of the binding affinity of non-cognate TFs, while an intensity of 1 equals the affinity of the cognate TF.

By controlling these two parameters, 35 crosstalk configurations have been generated for CELLO's library of genetic gates. In total, 7 different Dirichlet configurations have been used, with Configuration 0 representing a very peaked distribution, i.e., strong crosstalk by few TFs, and Configuration 7 representing a very noisy distribution, i.e., evenly distributed crosstalk across all TFs. Further, the intensity has been scaled in 5 steps of 0.05, 0.1, 0.15, 0.25 and 0.5. Figure 10.6 visualizes the different crosstalk configurations with 6 examples. The three Dirichlet configurations 0, 3, and 6 are shown from left to right. Each configuration is depicted with an intensity of 0.05 in the top row and an intensity of 0.5 in the bottom row. Vertically in each matrix, the promoters of CELLO's gate library are listed, while its TFs are listed horizontally. Each color-coded square in the grid represents the binding affinity of one TF to one promoter relative to the affinity of the corresponding cognate TF. It can be observed that the distribution of crosstalk is indeed very peaked for Configuration 0, with single binding affinities of non-cognate TFs exceeding those of the other configurations in the high-intensity case. On the other hand, Configuration 6 leads to a noise floor of evenly distributed crosstalk of varying intensity.

Impact on Search Efficiency

The presence of crosstalk in a genetic circuit is expected to impact the efficiency of B&B since all crosstalk relations have to be estimated optimistically, and feedback loops have to be opened up in that estimation (Section 10.2.1). The more crosstalk is estimated in a partial solution, the more the estimation is expected to deviate from the circuit characteristic with intact crosstalk cycles. Figure 10.7 shows the loss of search efficiency for all tested crosstalk configurations for exact (Figure 10.7a) and heuristic (Figure 10.7b) B&B. Each square in the grid depicts the relative overhead of B&B in the respective configuration, measured in the number of simulations required for mapping the set of circuits divided by the number required without crosstalk. Purple colors signify a positive overhead, with increased color saturation illustrating increased overhead. Light yellow, on the other hand, signifies an overhead below 1, i.e., improved efficiency. Generally, the overhead increases for both B&B modes with rising crosstalk intensity and rising noisiness, respectively. In exact mode, the baseline is given by $\sim 1.6 \times 10^5$ simulations without crosstalk. For one crosstalk configuration (Dirichlet configuration 0, intensity

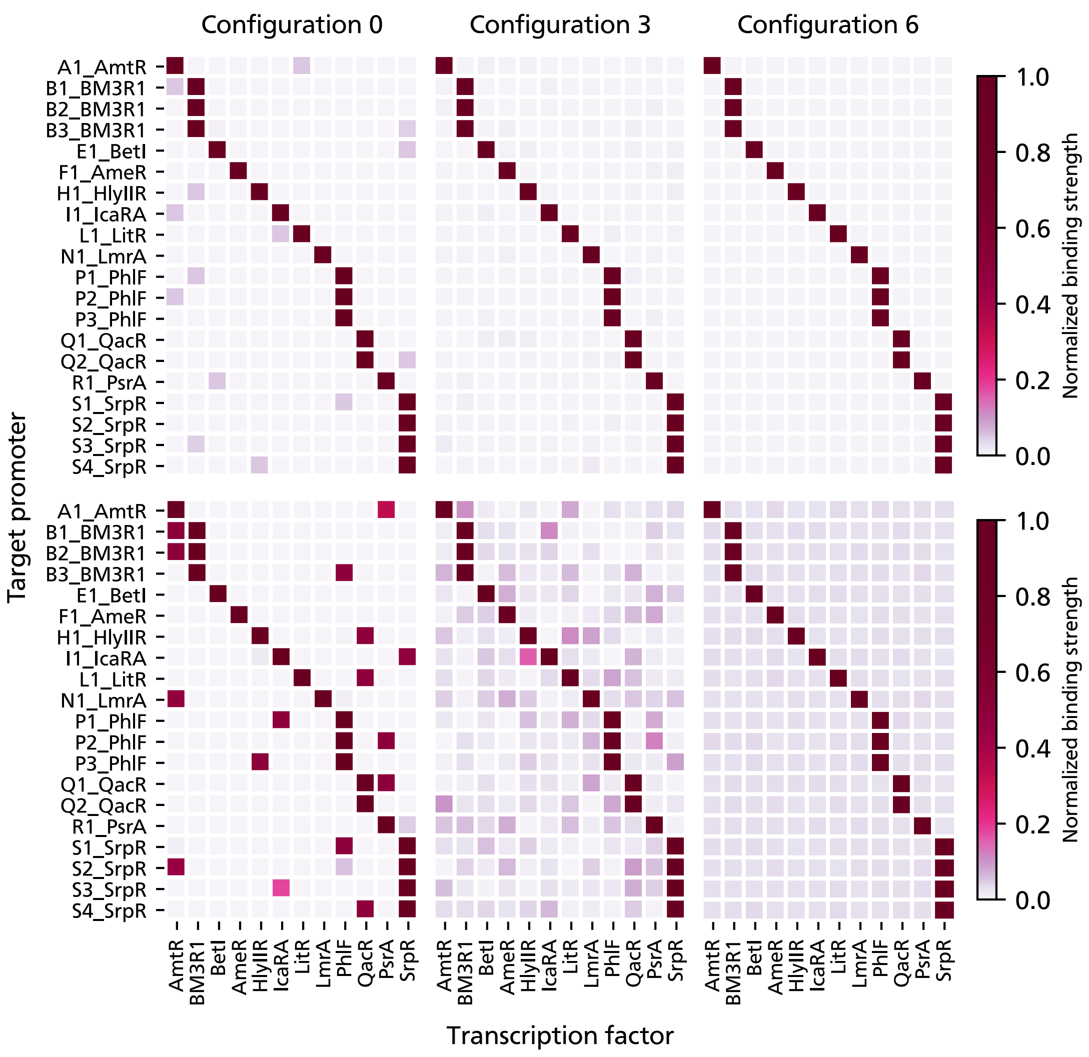
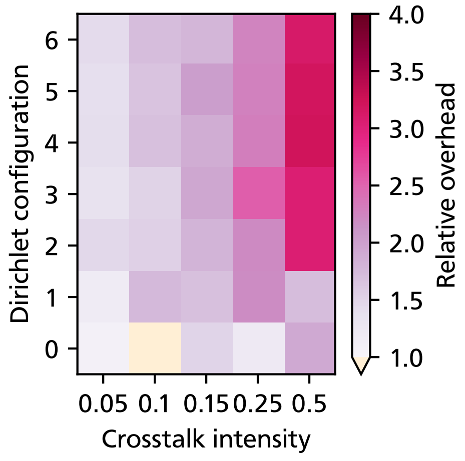
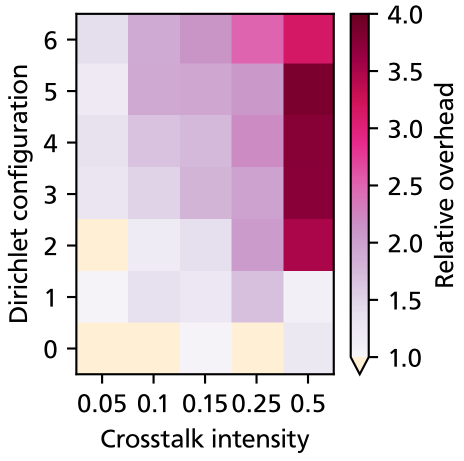


Figure 10.6.: Characteristics of assumed crosstalk: peaked to noisy distribution (left to right), low to high intensity (top: 0.05, bottom: 0.5)



(a) Exact mode



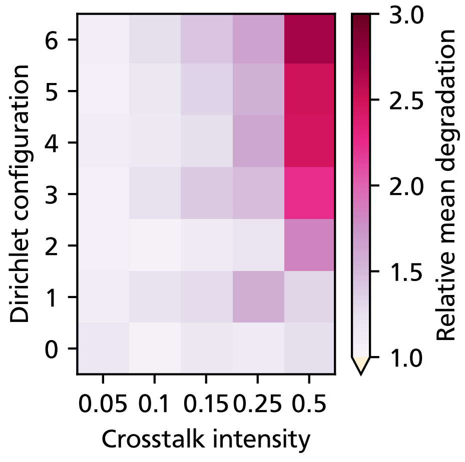
(b) Heuristic mode

Figure 10.7.: Loss in B&B efficiency in number of simulations for the different crosstalk configurations, relative to the crosstalk-free case

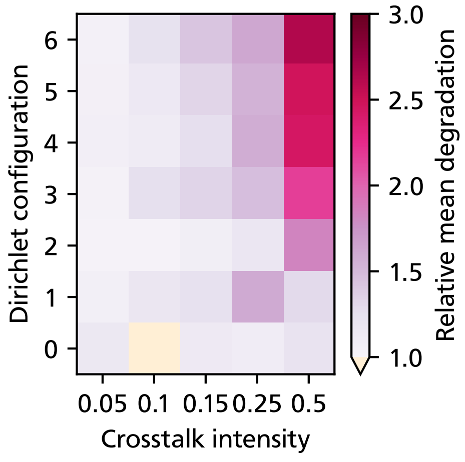
0.1), the search efficiency improved to $\sim 1.58 \times 10^5$ simulations, while it worsened for all others. The worst-case is measured for Dirichlet configuration 4 and intensity 0.5, amounting to $\sim 5.14 \times 10^5$ simulations. This equals a maximum $3.2\times$loss in efficiency compared to the crosstalk-free case. A similar characteristic with increased divergence can be observed for the heuristic mode. Here, four configurations lead to increased efficiency, with the best-case again obtained for Dirichlet configuration 0 and intensity 0.1. In that case, $\sim 0.22 \times 10^5$ simulations have been required, compared to the crosstalk-free baseline of $\sim 0.28 \times 10^5$. The worst-case in heuristic mode is given by $\sim 1.08 \times 10^5$ simulations for Dirichlet configuration 5 and intensity 0.5, equaling a $3.8\times$reduction in efficiency.

Impact on Circuit Performance

Besides impacting the efficiency of technology mapping, crosstalk can degrade the predicted circuit performance since undesired interactions potentially work against desired regulatory relations. Figure 10.8 shows the degradation of predicted circuit performance for the circuits generated with exact (Figure 10.8a) and heuristic (Figure 10.8b) B&B for all tested crosstalk configurations. Each square depicts the relative mean degradation of the circuits mapped with the corresponding configuration, measured in the mean Cello



(a) Exact mode



(b) Heuristic mode

Figure 10.8.: Degradation of the mean CELLO score for the different crosstalk configurations, relative to the crosstalk-free case

score of the circuits without crosstalk divided by the mean score of the circuits mapped with the given crosstalk configuration. In other words, the color codes the multiplier required to obtain the baseline score from the score determined with crosstalk. Again, purple colors signify a positive quotient, with increased color saturation illustrating increased degradation. Light yellow signifies a degradation below 1, i.e., improved predicted circuit performance. Paralleling the efficiency loss, the circuit performance generally degrades with rising crosstalk intensity and rising noisiness of its distribution. In exact mode, the worst-case degradation of $\sim 2.7\times$has been measured for Dirichlet configuration 6 and intensity 0.5, with a mean score of 114.2 compared to 307.2 without crosstalk. In heuristic mode, the same configuration leads to a similar worst-case degradation of $\sim 2.6\times$, with a mean score of 112.9 compared to 296.4 without crosstalk. For Dirichlet configuration 0 and intensity 0.1, even a slight improvement in predicted performance can be observed, with a $\sim 0.98\times$degradation. This improvement, however, is an artifact of the heuristic gate assignment optimization.

Conclusion

The context-aware B&B gate assignment method allows for the exact and heuristic technology mapping for genetic circuits that exhibit crosstalk. If the undesired interactions are distributed sparsely across the gate library, i.e., non-zero binding affinities of TFs to a limited number of non-cognate promoters exist, the efficiency of the optimization is maintained even for high intensities of crosstalk. The same holds for evenly distributed, noise-like crosstalk with limited intensity. With the rising intensity of noisy crosstalk, B&B's estimator becomes increasingly non-descriptive with respect to the prediction of the final score, leading to a loss in mapping efficiency.

The evaluation showed that the predicted circuit performance is similarly impacted by crosstalk. Crosstalk of low intensity does not substantially harm the circuit function, irrespective of its distribution. Even strong undesired binding affinities in the gate library are manageable if they are distributed sparsely among TFs and promoters. Their impact can be mitigated by considering them during the gate assignment optimization and constructing context-aware circuit designs. As expected, noisy crosstalk of high intensity harms the circuit performance since it degrades the performance of all gates and thus cannot be avoided by clever design. These results have an important implication for the design of genetic gate or part libraries. The paradigm of strict functional orthogonality can be relaxed to allow limited undesired interactions if these relations are sparse or of low intensity. As shown, good circuit designs can be found despite such non-orthogonalities if they are considered during technology mapping. This may open up new opportunities for constructing larger genetic gate and part libraries.

Synopsis

11. Combined Technology Mapping Flow

In the previous chapters, methods covering specific aspects of the technology mapping process for genetic circuits have been presented and evaluated in detail. In this chapter, the methods are embedded into the context of a comprehensive technology mapping flow (Section 11.1). Based on this, meaningful scenarios for technology mapping are composed and evaluated exemplary (Section 11.2).

11.1. Overview of the Methods

Figure 11.1 depicts the technology mapping flow and subsumes the methods presented and applied in this work.

Topology Synthesis First, a Boolean formula defining the desired function of the circuit is transformed into a structural representation in the step of topology synthesis. In Chapter 7, the principle of structural variants has been presented, allowing the synthesis of multiple circuit structures adhering to given synthesis constraints. Compared to classic technology mapping approaches known and adopted from EDA, the search space for circuit designs is broadened, taking into account heterogeneous genetic gate libraries. Since purely combinational circuits are considered in this work, the resulting structures are DAGs with nodes representing logic gates of the types given in the gate library. The topology synthesis is undirected and only limited by the gate library and constraints.

In the next step of gate assignment optimization, the circuits are optimized for different objectives. In the overview chart, the methods that can be applied in this phase of the technology mapping flow are separated into four categories.

Optimization of Functionality First, the adherence of the circuit to the functional specification can be measured either by the Cello score (Section 4.2), which rates the worst-case distance of the medians of the circuit's output distributions, or the E-Score (Section 4.3).

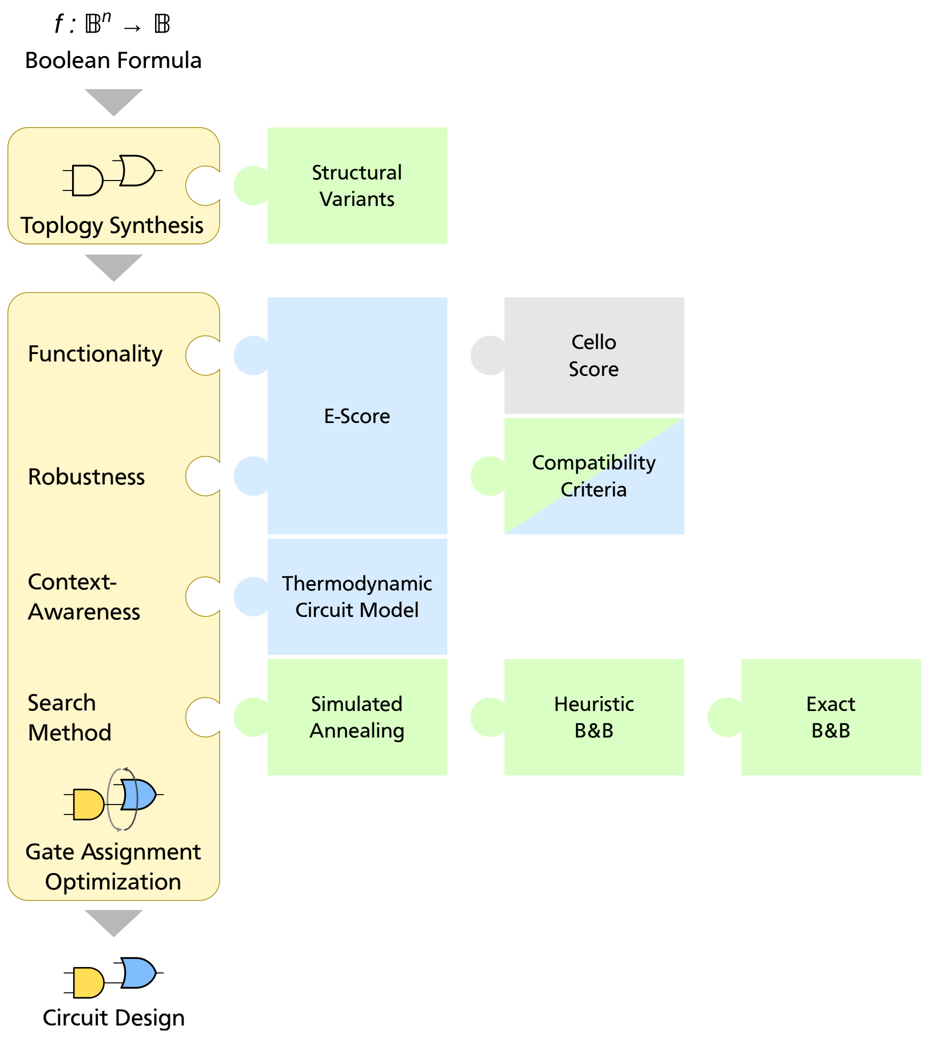


Figure 11.1.: Overview of technology mapping methods presented in this work (green), contributed by the Self-Organizing Systems Lab (blue) and adopted from the literature (gray)

Optimization for Robustness Since the E-Score performs a statistical evaluation of the circuit characteristic, it allows rating not only the distance but also the shape and overlap of the distributions. Thus, it enables the design of robust circuits and additionally covers the second category, robustness. As an alternative to this global optimization for robustness, compatibility criteria have been presented, which are based on local perturbation margins of signals and represent a heuristic for robustness that integrates into the B&B gate assignment method (Section 9.2).

Context-Awareness Third, context effects can be considered by using the thermodynamic circuit model to simulate and determine the output characteristic (Section 4.4).

Search Methods Lastly, multiple methods for efficiently traversing the search space of gate assignments have been presented, enabling the exploration of variants and the application of complex circuit models and scores. In Chapter 8, the proximity-based SA heuristic, as well as exact and heuristic versions of the B&B method, have been presented, based on the state-of-the-art circuit model and score of CELLO. In Chapters 9 and 10, the methods have been further developed to support the design of robust circuits and to consider context effects. The result of the flow is the circuit design, i.e., topology and gate assignment, found to deliver the highest score by the chosen search method.

Limitations Not all presented methods can be combined freely due to limitations detailed in the following. First, the B&B search method only supports optimization for the CELLO score due to its highly problem-specific estimator and the increased complexity of the E-Score. Thus, robust circuits can be designed either by optimizing for the E-Score using SA, or by optimizing for the CELLO score with applied compatibility criteria using either of the search methods. Second, the thermodynamic circuit simulator only supports rating circuits with the CELLO score. An alternative for designing robust circuits while considering context effects is given by combining the thermodynamic evaluation with the compatibility criteria.

11.2. Evaluation

Based on the overview given in Section 11.1, combinations of the methods are composed in Section 11.2.1 to meaningful technology mapping scenarios. Then, these scenarios are evaluated exemplary and compared to each other.

11.2.1. Technology Mapping Scenarios

With the methods presented in this work, many different technology mapping flows can be implemented. In the following, those combinations are identified which represent meaningful scenarios with respect to a given mapping objective. Table 11.1 provides an overview of the considered scenarios. All scenarios commonly make use of the enumeration of structural variants since it has been shown to broaden the search space effectively and thus is applicable irrespective of the mapping objective or search method.

Scenario A: Classic Technology Mapping

The first scenario parallels the base configuration used to construct and evaluate the basic search methods (Chapter 8). That is, the CELLO score is used to optimize gate assignments, robustness is not explicitly taken into account, and the classic circuit simulator which applies CELLO's circuit model (Section 4.2) is used. Since the heuristic B&B method has been shown to deliver quasi-optimal results and an outstanding efficiency (Section 8.4.3), it is used as a search method in this scenario.

Table 11.1.: Overview of considered technology mapping scenarios

Scenario	Structural Variants	Functionality	Robustness	Context-Effects	Search Method
A: Classic	✓	CELLO score	-	-	heur. B&B
B.1: Robustness	✓	E-Score	E-Score	-	SA
B.2: Robustness	✓	CELLO score	Compat.	-	heur. B&B
C.1: Context-Aware	✓	CELLO score	-	✓	heur. B&B
C.2: Context-Aware	✓	CELLO score	Compat.	✓	heur. B&B

Scenarios B.1 and B.2: Technology Mapping for Robustness

Scenarios B.1 and B.2 are centered around designing robust genetic circuits and represent alternative approaches to this problem. Scenario B.1 applies the E-Score as objective, which covers both functionality and robustness of circuit designs. Since this score is only supported by the SA scheme, it is used as a search method with the inner-loop criterion configured for the E-Score (Section 9.1). Scenario B.2 represents an alternative approach aimed towards a reduced mapping runtime, which uses the simpler CELLO score and the compatibility criteria to heuristically optimize the robustness of the design. This allows the use of the heuristic B&B search method, which is highly optimized for the score and the compatibility criteria (Section 9.2.3).

Scenarios C.1 and C.2: Context-Aware Technology Mapping

Finally, scenarios C.1 and C.2 allow for considering context effects during technology mapping. Scenario C.1 uses the thermodynamic circuit simulator to model context effects and calculate the CELLO score of designs. Again, the heuristic B&B search method is used in its context-aware version (Section 10.2.2) to reduce the number of costly simulations of the thermodynamic model. Based on this, scenario C.2 incorporates robustness heuristically by selecting gate assignments based on the compatibility criteria.

11.2.2. Setup

To evaluate the different technology mapping flows comparatively, they are subsequently carried out on a standard desktop PC for an exemplary Boolean function. This allows assessing their runtime in a realistic work environment. Both steps in the flow, i.e., the enumeration of structural variants and the gate assignment optimization, are carried out in parallel with four threads on a $3.2\,\mathrm{GHz}$ processor with four cores. As gate library, CELLO's [53] collection of repressor-based NOT and NOR gates is used. The Boolean function $f = \neg A \neg B \lor A(B \lor C)$ has been chosen as functional specification. It is implementable in the given technology with a minimum of 5 gates, with one of these gates being an implicit OR gate at the output. Thus, it leads to medium-sized circuits in the context of circuits considered in this work. For the enumeration of structural variants, the maximum weight $\omega = 8$ is chosen to allow for exploring non-minimal circuit structures. However, the search space is constrained by setting the number of allowed excess gates to $e = 1$. That is, all structures featuring more than one excess gate compared to the minimum structure are filtered out during the enumeration. This enables a limited trade-off of optimization objectives and circuit size. For evaluating the context-aware scenarios, one of the crosstalk configurations presented in Section 10.2.3 is assumed. Specifically, Dirichlet configuration 2 and an intensity of 0.15 is chosen.

11.2.3. Results

In this section, the runtime of the different technology mapping scenarios, as well as the resulting circuit designs, are detailed.

Runtime

Table 11.2 shows the runtime of each mapping scenario for both the enumeration of structural variants and the optimization of the gate assignment. Since the topology synthesis is not controlled by the mapping objective, it is invariant among all scenarios. In total, 11 circuit topologies have been synthesized that adhere to the given constraints in $\sim 23\,\mathrm{min}$. The number of gate assignments that are evaluated on these structures differs by approximately two orders of magnitude in the different scenarios. While the heuristic B&B search method, if constrained by the compatibility criteria, evaluates only $\sim 11.5 \times 10^3$ (B.2) and $\sim 16.7 \times 10^3$ (C.2) designs, the SA method evaluates $\sim 1.45 \times 10^6$ assignments when optimizing for the E-Score (B.1). The unconstrained usage of the heuristic B&B method (A and C.1) leads to numbers of evaluated circuits between these two orders of magnitude. The resulting runtimes additionally depend on the used circuit model and score. Compared to the classic scenario, in which CELLO's circuit model and score are applied, both the E-Score (B.1) as well as the thermodynamic circuit model (C.1 and C.2) increase the computational complexity of circuit evaluation. The total runtime for technology mapping spans from $\sim 0.5\,\mathrm{h}$ (A, B.2 and C.2) to $\sim 5\,\mathrm{h}$ (B.1).

Table 11.2.: Runtime of mapping function $f = \neg A \neg B \vee A(B \vee C)$ in different scenarios

	Structural Variants		Assignment Optimization		
Scenario	Number	Runtime	Simulations	Runtime	Total (h)
A: Classic	11	22 min 42 sec	111 049	11 min 4 sec	0:34
B.1: Robustness			1 451 252	276 min 13 sec	4:59
B.2: Robustness			11 490	6 min 43 sec	0:29
C.1: Context-Aware			299 236	117 min 56 sec	2:21
C.2: Context-Aware			16 731	9 min 15 sec	0:32

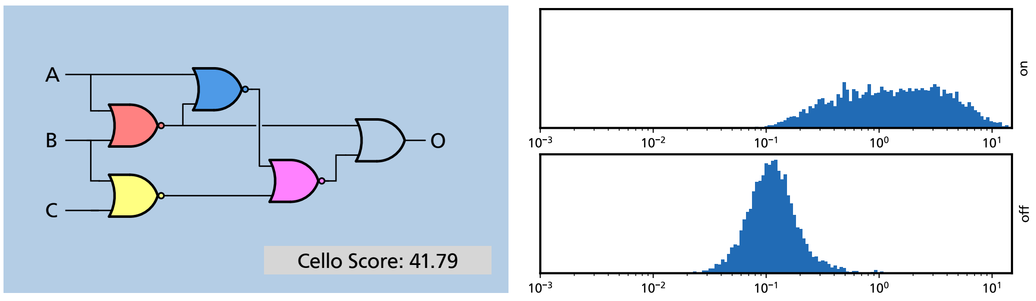
Circuit Designs

Figures 11.2 and 11.3 depict the circuit topologies, gate assignments, and achieved score values for all technology mapping scenarios. Figure 11.2 also shows the circuits' simulated population-wide output characteristic. It can be observed that in scenarios A, B.1, and B.2 (Figures 11.2a, 11.2b and 11.2c), the same circuit topology is selected from the set of 11 structures. It consists of four NOR gates and one OR gate. In the context-aware scenarios C.1 and C.2 (Figures 11.3a and 11.3b), however, the mapping process makes use of the allowed excess gate. Two different topologies featuring an additional NOT gate are selected in these cases. Furthermore, it can be observed that each mapping scenario leads to an individual selection and assignment of genetic gates from the library to the circuit structures.

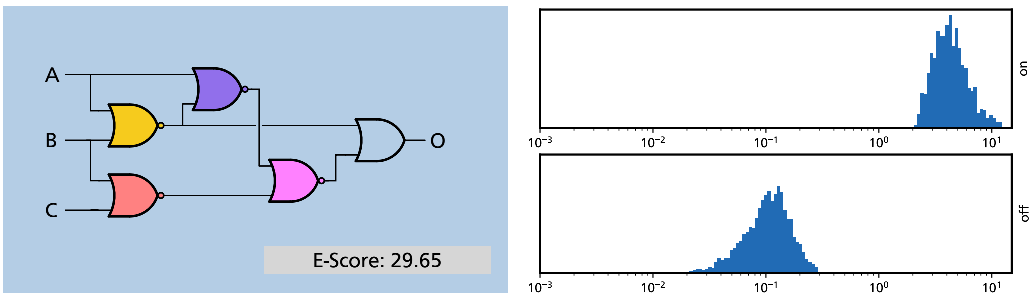
Comparing the output characteristics of the circuits mapped for robustness (B.1 and B.2) to the one obtained in the classic scenario (A), a strong separation and an eliminated overlap of the output distributions is visible. Both robustness-centric solutions B.1 and B.2 feature comparable E-Score values and nearly identical output characteristics. That is, the heuristic approach to robustness, based on the compatibility criteria and heuristic B&B, delivers a result of similar quality compared to the direct optimization for the E-Score. However, it has been previously shown that this heuristic is not guaranteed to provide such good results (Section 9.3). The results for scenarios C.1 and C.2 show that under assumed crosstalk, wholly different circuit designs prove advantageous. Scenario C.1 parallels scenario A and C.2 parallels B.2 concerning the mapping parameters, other than assumed context effects. In both cases, however, different circuit topologies and gate assignments are selected, providing the best performance under crosstalk.

Conclusion

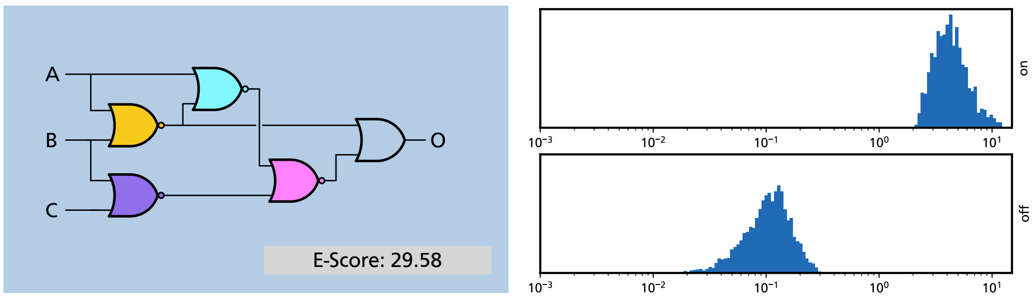
In this section, a Boolean function with three inputs has been mapped to a library of genetic NOT and NOR gates. The mapping has been performed with different parameters, i.e., scenarios, for the technology mapping flow. The runtimes measured on a desktop PC showcase the mapping methods' feasibility in a standard working environment. Further, they highlight the computational burden of statistical scoring and thermodynamic modeling and thus emphasize the necessity of efficient search methods. The resulting circuit designs differ among the scenarios and demonstrate the benefits of taking structural variants into account, making robustness a design objective, and performing a context-aware evaluation of the circuit if crosstalk is present among gates in the library.



(a) Scenario A: Classic Technology Mapping



(b) Scenario B.1: Technology Mapping for Robustness

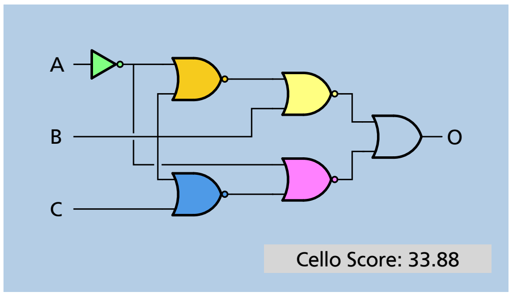


(c) Scenario B.2: Technology Mapping for Robustness

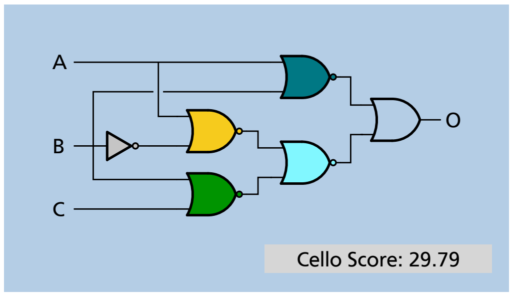
🟥 A1_AmtR	🟨 E1_BetI	🟦 H1_HlyIIR	🟦 I1_IcaRA	🟧 L1_LitR	🟩 P1_PhlF
🟪 P3_PhlF	⬜ Q1_QacR	🟪 Q2_QacR	🟦 S2_SrpR	🟩 S4_SrpR	⬜ Output OR

(d) Gate legend

Figure 11.2.: Circuits implementing function $f = \neg A \neg B \vee A(B \vee C)$ resulting from classic and robustness-aware technology mapping



(a) Scenario C.1



(b) Scenario C.2

Figure 11.3.: Circuits implementing function $f = \neg A \neg B \vee A(B \vee C)$ resulting from context-aware technology mapping (gate legend in Figure 11.2d)

12. Conclusion and Outlook

In this chapter, the work is concluded in Section 12.1, followed by a discussion of potential future work in Section 12.2.

12.1. Conclusion

Within this work, the problem of technology mapping for genetic circuits was examined from multiple perspectives. First of all, methods of existing GDA tools were analyzed to identify the potential for improved domain adaptation. Further, the problem structure was examined in detail to identify characteristics that can be leveraged for efficient mapping. Lastly, a critical knowledge transfer from EDA was performed, deducing analogies between the two fields but also considering their differences.

Revisiting the motivation for the research presented in this work, one can formulate three central objectives which follow from the current state of GDA. First, existing technology mapping methods often exhibit a limited adaptation to the problem structure and thus leave out important parts of the design space or deliver solutions with sub-optimal quality. However, achieving a high quality of designs generated in technology mapping is necessary to use the given models to their fullest potential and to enter the lab with the best possible design. Second, the exploration of broad design spaces, application of complex models, integration into the "design, build, test" cycle, and the potential growth of circuit and library sizes all demand efficient technology mapping methods. Third, GDA design flows suffer from the limited predictive power of the used models, and the support of refined models can improve the meaningfulness of mapped designs.

Domain Adaptation The methods presented in this work improve the domain adaptation of the technology mapping compared to state-of-the-art GDA approaches twofold.

First, incorporating structural variants in the design process broadens the search space for circuit designs. The circuit topology thereby also becomes a subject of optimization instead of demanding it as an input from the user or adopting the topology synthesis from EDA. This approach is motivated by the restricted and heterogeneous genetic gate libraries and the reduced correlation between features of the circuit topology and mapping objectives. It could be shown that it improves the quality of mapped designs substantially by up to $11.3\times$and a mean $38\,\%$, while keeping the circuit size minimal. The currently applied enumeration of variants delivers exact results for the given synthesis constraints. Enumeration is feasible for the circuit sizes implementable at present but is expected to become impractical when dealing with considerably larger circuits. Thus, approximate methods could be applied to leverage the concept of structural variants if genetic circuits grow in size (Section 12.2).

Second, the discussed optimization methods reliably deliver high-quality solutions based on thorough problem adaptation. If optimality is a strict requirement in the GDA flow, the exact B&B method provides exact solutions substantially faster than an exhaustive search. This is achieved by considering the functional hierarchy of genetic circuits and fundamental features of genetic gates based on transcriptional regulation. The devised heuristic methods apply well-founded approximations and deliver near-optimal results. In detail, the SA method exhibits a worst-case deviation of $-3.15\,\%$ and the B&B method of $-0.11\,\%$, outperforming an existing heuristic in both efficiency and quality. Applying heuristic methods and trading off exactness for the possibility of exploring broad design spaces may represent a practical choice in many scenarios, given the natural variability of biological systems.

Mapping Efficiency All presented optimization methods for the gate assignment substantially reduce the number of needed circuit evaluations compared to an exhaustive search. The reduction is particularly large for the heuristic methods, which exhibit a $79\times$(SA) and a $611\times$(B&B) gain in efficiency, respectively. The increased efficiency translates to a comparable speedup of the technology mapping process ($108\times$with SA, $722\times$with B&B). In the context of this work, this speedup allows for the exploration of structural variants and the evaluation of elaborate circuit models in a standard computing environment. In a general GDA flow, it further enables the practical integration of the methods into an iterative design process and the support of potentially growing circuit and library sizes.

Support of Refined Models This work improves the meaningfulness of circuit designs generated by technology mapping twofold.

First, the SA gate assignment scheme has been adapted to an alternative robustness-centric objective function that considers cell-to-cell variability, the E-Score. On the one hand, this highlights the flexibility of the SA method concerning the used objective function. On the other hand, the required adaptation of a parameter of the annealing schedule shows the highly parameterized nature of the method. Since the used E-Score is more complex than the traditional median-based score, mapping efficiency and solution quality are slightly reduced. Alternatively, a heuristic approach to robustness has been devised based on local perturbation margins of signals. Integrating these compatibility criteria into the B&B method with a lookahead scheme leads to a higher mapping efficiency compared to the E-Score. The separation of output distributions, however, is reduced for some of the tested circuits. Thus, the global optimization for robustness with the E-Score is a worthwhile investment in applications where a high tolerance to cell-to-cell variability is a central design objective, e.g., in biomedical applications.

Second, the integration of a context-aware circuit model into the technology mapping flow allows for the exact and heuristic mapping for circuits that exhibit crosstalk. To this end, the B&B gate assignment scheme is adapted to this model by embedding each gate into its individual optimistic crosstalk environment during estimation. This improves the predictive power of the mapping process and enables the designer of genetic gate libraries to conclude the implications of potential crosstalk for the quality of designs. For the library considered in this work, it could be shown that evenly distributed crosstalk of high intensity degrades design quality. Crosstalk of low intensity or sparsely distributed crosstalk, however, proved to be tolerable in the technology mapping process. Thereby, the potential for relaxing the orthogonality criterion arises, simplifying the future extension of libraries.

12.2. Outlook

The technology mapping for genetic circuits offers a multitude of opportunities for further research. Some arise from the experience gained in implementing and evaluating the methods presented in this work and are directly associated with these methods. Others represent opportunities evolving from the broader context of GDA and the progress in this field.

Heuristic for Structural Variants The enumeration of circuit topologies is exact with respect to the given synthesis constraints but exhibits a combinatorial nature. It spans a broad search space for the currently implementable small genetic logic circuits but is expected to become infeasible for larger circuits. Thus, approximate methods are necessary to benefit from structural variants if the size of circuits increases. A possible approach is using *supergates* [18], i.e., pre-computed building blocks consisting of multiple gates. These can be applied in a traditional structural mapping scheme, in which variants are generated by covering the subject graph with different supergates. To adapt the approach for genetic circuits, these building blocks can be selected based on their output characteristic in the context of the considered gate library.

E-Score in Branch-and-Bound The B&B gate assignment methods have been developed for the median-based CELLO score for circuit functionality. Their ability to prune large parts of the search tree is based on the optimistic estimation of the score for partial solutions. It can be assumed that the functional hierarchy of genetic circuits leveraged in the B&B scheme also holds for optimization with the E-Score. Thus, support for the E-Score could improve the efficiency of technology mapping for robustness compared to the SA scheme. It is questionable if an optimistic estimator for this more complex score can be found, which enables finding the exact solution. However, heuristic B&B has been proven to provide a near-optimal solution quality.

Energy-Aware Technology Mapping Cells are known to exhibit a trade-off between function and energy [30]. Thus, the mutual exclusivity of both objectives must be assumed when designing synthetic genetic circuits. In the context of technology mapping, this implies a multi-objective optimization, allowing the designer to constrain one of the objectives or explore the Pareto set. First, compact models that enable evaluating the energy consumption of genetic circuits must be found.

In Vivo Implementation of Designs The presented methods allow for the automated design and evaluation of genetic circuits *in silico*. They are based on fundamental features of genetic gates, i.e., their heterogeneous, bounded, and monotone transfer characteristics. Further, they have been evaluated empirically using an existing library of genetic gates, which has been extensively tested in the lab and has been used for *in vivo* implementations. However, since the technology mapping methods are part of the iterative "design, build, test" cycle, implementing the generated designs and measuring their characteristic can provide further insight into the technology mapping flow and spark potential future research.

Sequential Circuits This work is focused on the technology mapping for combinational genetic circuits. However, approaches for implementing sequential logic circuits exist [4, 40, 52]. These are mostly based on constructing latches from genetic gates to provide memory elements. Whether this approach is the key to sequential genetic circuits or whether more energy-efficient solutions can be found for storing information is unclear. Nonetheless, sequential logic represents a new challenge for technology mapping methods since the temporal behavior of such designs has to be considered in modeling and optimization.

Acknowledgments

I am deeply grateful to my supervisor Christian Hochberger for giving me the opportunity to follow my research interests and for providing invaluable guidance and support.

I am also thankful to Marc Riedel for reviewing this work and for participating in the defense committee.

Special thanks to Nicolai Engelmann and Erik Kubaczka of the Self-Organizing Systems Lab for the fruitful collaboration on genetic design automation that enabled the research presented in this work.

I express my profound thankfulness to my parents Christel and Thomas, who always believed in me and supported me in every possible way.

Lastly, words cannot express my gratitude to my wife Christin, who cheered me up in challenging times and celebrated the beautiful moments with me.

Appendix

A. Functions Used for Enumeration

Table A.1 details the Boolean functions used for evaluating the enumeration of structural variants.

Table A.1.: Truth tables of the 33 mapped three input functions with output values ordered from $A = 1, B = 1, C = 1$ to $A = 0, B = 0, C = 0$

Function	Truth table
ϕ_1	00000001
ϕ_2	00000010
ϕ_3	00010011
ϕ_4	00010100
ϕ_5	00010111
ϕ_6	00011111
ϕ_7	00100000
ϕ_8	00101010
ϕ_9	00101011
ϕ_{10}	00101111
ϕ_{11}	00110001
ϕ_{12}	00111110
ϕ_{13}	01010100
ϕ_{14}	01010110
ϕ_{15}	01100010
ϕ_{16}	01111101
ϕ_{17}	10000000
ϕ_{18}	10000001
ϕ_{19}	10001010
ϕ_{20}	10010000
ϕ_{21}	10101000
ϕ_{22}	10101001
ϕ_{23}	10110010
ϕ_{24}	10110011
ϕ_{25}	10111001
ϕ_{26}	11001110
ϕ_{27}	11011111
ϕ_{28}	11100010
ϕ_{29}	11100110
ϕ_{30}	11100111
ϕ_{31}	11101100
ϕ_{32}	11111101
ϕ_{33}	11111110

B. Scores of Structural Variants

Table B.1 contains the Cello scores reached with Cello's circuit structures and with structural variants encompassing a different number of excess gates allowed.

Table B.1.: Cello score reached with Cello's topologies and structural variants with different number of excess gates allowed.

		Variants		
Funct.	**Cello**	$e = 0$	$e = 1$	$e = 2$
ϕ_1	610.96	654.26	654.26	671.49
ϕ_2	327.87	671.49	671.49	671.49
ϕ_3	46.90	46.90	46.90	598.99
ϕ_4	43.19	488.51	491.24	610.95
ϕ_5	46.90	46.90	46.90	597.17
ϕ_6	47.14	47.14	47.14	599.01
ϕ_7	327.87	671.47	671.47	671.47
ϕ_8	43.53	43.53	43.47	595.80
ϕ_9	43.39	43.39	43.39	597.46
ϕ_{10}	47.14	47.14	47.14	598.78
ϕ_{11}	671.34	671.34	671.34	671.43
ϕ_{12}	43.41	43.72	43.72	597.21
ϕ_{13}	611.04	611.04	611.04	611.04
ϕ_{14}	37.74	307.17	324.38	593.85
ϕ_{15}	43.08	324.70	324.70	324.70
ϕ_{16}	40.04	44.92	44.92	593.85
ϕ_{17}	327.66	327.90	328.16	328.18

		Variants		
Funct.	**Cello**	$e = 0$	$e = 1$	$e = 2$
ϕ_{18}	308.68	308.44	327.28	327.31
ϕ_{19}	327.86	327.87	327.86	595.85
ϕ_{20}	316.00	327.30	327.45	327.45
ϕ_{21}	327.87	327.87	327.87	327.91
ϕ_{22}	31.34	31.34	36.94	592.75
ϕ_{23}	39.01	326.25	327.39	596.14
ϕ_{24}	44.76	308.48	327.28	597.10
ϕ_{25}	39.82	39.82	39.82	597.12
ϕ_{26}	458.31	458.31	458.31	678.07
ϕ_{27}	418.52	418.53	418.54	678.07
ϕ_{28}	327.63	327.66	327.65	595.96
ϕ_{29}	37.79	39.11	39.17	595.78
ϕ_{30}	39.25	41.79	41.79	596.63
ϕ_{31}	207.04	327.93	327.93	678.05
ϕ_{32}	319.21	469.35	469.35	678.08
ϕ_{33}	183.04	183.04	183.04	677.91

C. Contributions to Crosstalk Environments

In the following, tables detailing contributions to crosstalk-environments are listed, which have been left out in Section 10.2. Table C.1 shows the contribution of the target gate to its own crosstalk environments. Table C.2 shows the contribution of unassigned gates to crosstalk environments for heuristic B&B.

Table C.1.: Contributions of the target gate itself to its crosstalk-environments for optimal B&B

Target gate θ_j	Contribution
0-environment	$(c_{\max,j}, b_{jj})$
1-environment	$(c_{\min,j}, b_{jj})$

Table C.2.: Contributions of unassigned source gates to the target gate's crosstalk-environments for heuristic B&B

	Source gate θ_i	
Target gate θ_j	0-environment	1-environment
0-environment	$(c_{\max,i}, b_{\max,j})$	$(j_{0,i}, b_{\max,j})$
1-environment	$(j_{1,i}, b_{\min,j})$	$(c_{\min,i}, b_{\min,j})$

D. Bibliography

[1] Alfred V. Aho, John E. Hopcroft, and Ullman D. Jeffrey. *The Design and Analysis of Computer Algorithms*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., Jan. 1, 1974. ISBN: 0201000296.

[2] Saed Alizamir, Steffen Rebennack, and Panos Pardalos. "Improving the Neighborhood Selection Strategy in Simulated Annealing Using the Optimal Stopping Problem". In: *Global Optimization: Focus on Simulated Annealing*. Ed. by Cher Ming Tan. I-Tech Education and Publication, Sept. 2008, pp. 363–382. ISBN: 978-953-7619-07-7. DOI: `10.5772/5571`.

[3] Uri Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman & Hall/CRC Computational Biology Series. CRC Press, 2019. ISBN: 9781000001327. DOI: `10.1201/9781420011432`.

[4] Lauren B. Andrews, Alec A.K. Nielsen, and Christopher A. Voigt. "Cellular checkpoint control using programmable sequential logic". In: *Science* 361.6408 (2018).

[5] Evan Appleton et al. "Design Automation in Synthetic Biology". In: *Cold Spring Harbor Perspectives in Biology* 9 (Feb. 2017). DOI: `10.1101/cshperspect.a023978`.

[6] Hasan Baig and Jan Madsen. "A Top-down Approach to Genetic Circuit Synthesis and Optimized Technology Mapping". English. In: *Proceedings of 9th International Workshop on Bio-Design Automation*. 9th International Workshop on Bio-Design Automation, IWBDA 2017. 2017.

[7] Jacob Beal et al. "Communicating Structure and Function in Synthetic Biology Diagrams". In: *ACS Synthetic Biology* 8.8 (2019), pp. 1818–1825. DOI: `10.1021/acssynbio.9b00139`.

[8] Vaughn Betz and Jonathan Rose. "VPR: a new packing, placement and routing tool for FPGA research". In: *Field-Programmable Logic and Applications*. Ed. by Wayne Luk, Peter Y. K. Cheung, and Manfred Glesner. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 213–222. ISBN: 978-3-540-69557-8.

[9] Armin Biere et al., eds. *Handbook of Satisfiability - Second Edition*. Vol. 336. Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. ISBN: 978-1-64368-160-3. DOI: 10.3233/FAIA336.

[10] Olivier Borkowski et al. "Overloaded and stressed: whole-cell considerations for bacterial synthetic biology". In: *Current Opinion in Microbiology* 33 (2016). Antimicrobials • Microbial systems biology, pp. 123–130. ISSN: 1369-5274. DOI: 10.1016/j.mib.2016.07.009.

[11] Clive G. Bowsher and Peter S. Swain. "Identifying sources of variation and the flow of information in biochemical networks". In: *Proceedings of the National Academy of Sciences* 109.20 (2012), E1320–E1328. DOI: 10.1073/pnas.1119407109.

[12] Robert Brayton and Alan Mishchenko. "ABC: An Academic Industrial-Strength Verification Tool". In: *Computer Aided Verification*. Ed. by Tayssir Touili, Byron Cook, and Paul Jackson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 24–40. ISBN: 978-3-642-14295-6.

[13] Robert C. Brewster et al. "The transcription factor titration effect dictates level of gene expression". In: *Cell* 156.6 (2014), pp. 1312–1323.

[14] J. Brown. "The iGEM competition: building with biology". English. In: *IET Synthetic Biology* 1 (1 June 2007), 3–6(3). ISSN: 1752-1394.

[15] Lukas Buecherl and Chris J. Myers. "Engineering genetic circuits: advancements in genetic design automation tools and standards for synthetic biology". In: *Current Opinion in Microbiology* 68 (2022), p. 102155. ISSN: 1369-5274. DOI: 10.1016/j.mib.2022.102155.

[16] Stefano Cardinale and Adam Paul Arkin. "Contextualizing context for synthetic biology – identifying causes of failure of synthetic biological systems". In: *Biotechnology Journal* 7.7 (2012), pp. 856–866. DOI: 10.1002/biot.201200085.

[17] Francesca Ceroni et al. "Quantifying cellular capacity identifies gene expression designs with reduced burden". In: *Nature Methods* 12.5 (May 2015), pp. 415–418. ISSN: 1548-7105. DOI: 10.1038/nmeth.3339.

[18] Satrajit Chatterjee et al. "Reducing structural bias in technology mapping". In: *ICCAD, 2005.* Nov. 2005, pp. 519–526. DOI: 10.1109/ICCAD.2005.1560122.

[19] Ye Chen et al. "Genetic circuit design automation for yeast". In: *Nature Microbiology* 5.11 (2020), pp. 1349–1360.

[20] Jens Clausen. *Branch and Bound Algorithms – Principles And Examples*. 1999.

[21] Giovanni De Micheli. *Synthesis and Optimization of Digital Circuits*. Electrical and Computer Engineering Series. McGraw-Hill, 1994. ISBN: 9780070163331.

[22] Rina Dechter and Judea Pearl. “Generalized Best-First Search Strategies and the Optimality of A*”. In: *J. ACM* 32.3 (July 1985), pp. 505–536. ISSN: 0004-5411. DOI: `10.1145/3828.3830`.

[23] Michael B. Elowitz and Stanislas Leibler. “A synthetic oscillatory network of transcriptional regulators”. In: *Nature* 403.6767 (Jan. 2000), pp. 335–338. ISSN: 1476-4687. DOI: `10.1038/35002125`.

[25] Tamar Friedlander et al. “Intrinsic limits to gene regulation by global crosstalk”. In: *Nature Communications* 7.1 (Aug. 2016), p. 12307. ISSN: 2041-1723. DOI: `10.1038/ncomms12307`.

[26] Timothy S. Gardner, Charles R. Cantor, and James J. Collins. “Construction of a genetic toggle switch in Escherichia coli”. In: *Nature* 403.6767 (Jan. 2000), pp. 339–342. ISSN: 1476-4687. DOI: `10.1038/35002131`.

[27] Kathryn L. Garner. “Principles of synthetic biology”. In: *Essays in Biochemistry* 65.5 (Nov. 2021), pp. 791–811. ISSN: 0071-1365. DOI: `10.1042/EBC20200059`.

[28] Alison L. Gibbs and Francis Edward Su. “On Choosing and Bounding Probability Metrics”. In: *International Statistical Review* 70 (2002). DOI: `10.1111/j.1751-5823.2002.tb00178.x`.

[29] Larry Goldstein and Michael Waterman. “Neighborhood Size in the Simulated Annealing Algorithm”. In: *American Journal of Mathematical and Management Sciences* 8.3-4 (1988), pp. 409–423. DOI: `10.1080/01966324.1988.10737247`.

[30] Jean Hausser et al. “Central dogma rates and the trade-off between precision and economy in gene expression”. In: *Nature communications* 10.1 (2019), p. 68.

[31] Linh Huynh et al. “SBROME: A Scalable Optimization and Module Matching Framework for Automated Biosystems Design”. In: *ACS Synthetic Biology* 2 (Mar. 2013). DOI: `10.1021/sb300095m`.

[32] Florian Ingels. “On the similarities of trees : the interest of enumeration and compression methods”. Theses. Ecole normale supérieure de lyon - ENS LYON, Sept. 2022. URL: `https://theses.hal.science/tel-03908078`.

[33] Shridhar Jayanthi, Kayzad Soli Nilgiriwala, and Domitilla Del Vecchio. “Retroactivity Controls the Temporal Dynamics of Gene Transcription”. In: *ACS Synthetic Biology* 2.8 (2013), pp. 431–441. DOI: `10.1021/sb300098w`.

[34] Timothy S. Jones et al. “Genetic circuit design automation with Cello 2.0”. In: *Nature Protocols* 17.4 (2022), pp. 1097–1113.

[35] Kurt Keutzer and Kaushik Ravindran. “Technology Mapping”. In: *Encyclopedia of Algorithms*. Ed. by Ming-Yang Kao. Boston, MA: Springer US, 2008, pp. 944–947. ISBN: 978-0-387-30162-4. DOI: 10.1007/978-0-387-30162-4_420.

[36] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by Simulated Annealing”. In: *Science* 220.4598 (1983), pp. 671–680. DOI: 10.1126/science.220.4598.671.

[37] Tasuku Kitada et al. “Programming gene and engineered-cell therapies with synthetic biology”. In: *Science* 359.6376 (2018).

[38] William S. Klug et al. *Essentials of genetics*. 10th ed. Pearson Education, 2019. 608 pp. ISBN: 9780134898414.

[39] Andreas Kuehlmann and Florian Krohm. “Equivalence Checking Using Cuts and Heaps”. In: *Proceedings of the 34th Annual Design Automation Conference*. DAC ’97. Anaheim, California, USA: Association for Computing Machinery, 1997, pp. 263–268. ISBN: 0897919203. DOI: 10.1145/266021.266090.

[40] Matthew Lebovich, Min Zeng, and Lauren B. Andrews. “Algorithmic Programming of Sequential Logic and Genetic Circuits for Recording Biochemical Concentration in a Probiotic Bacterium”. In: *ACS Synthetic Biology* (2023). DOI: 10.1021/acssynbio.3c00232.

[41] Siang-Yun Lee et al. “Enumeration of Minimum Fanout-Free Circuit Structures”. In: *IWLS-2019. International Workshop on Logic & Synthesis, 2019*. 2019.

[42] Chunbo Lou et al. “Ribozyme-based insulator parts buffer synthetic circuits from genetic context”. In: *Nature Biotechnology* 30.11 (Nov. 2012), pp. 1137–1142. ISSN: 1546-1696. DOI: 10.1038/nbt.2401.

[43] Mario Andrea Marchisio. “Parts & pools: a framework for modular design of synthetic gene circuits”. en. In: *Front Bioeng Biotechnol* 2 (Oct. 2014), p. 42.

[44] James Alastair McLaughlin et al. “The Synthetic Biology Open Language (SBOL) Version 3: Simplified Data Exchange for Bioengineering”. In: *Frontiers in Bioengineering and Biotechnology* 8 (2020). ISSN: 2296-4185. DOI: 10.3389/fbioe.2020.01009.

[45] Kurt Mehlhorn and Peter Sanders. *Algorithms and Data Structures: The Basic Toolbox*. Springer Berlin, Heidelberg, June 23, 2008, p. 300. ISBN: 978-3-540-77977-3. DOI: 10.1007/978-3-540-77978-0.

[46] Nicholas Metropolis et al. “Equation of State Calculations by Fast Computing Machines”. In: *The Journal of Chemical Physics* 21.6 (Dec. 1953), pp. 1087–1092. ISSN: 0021-9606. DOI: 10.1063/1.1699114.

[47] Alan Mishchenko et al. *FRAIGs: A unifying representation for logic synthesis and verification*. Tech. rep. ERL Technical Report, 2005.

[48] Tae Seok Moon et al. “Genetic Programs Constructed from Layered Logic Gates in Single Cells”. In: *Nature* 491 (Oct. 2012). DOI: 10.1038/nature11516.

[49] David R. Morrison et al. “Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning”. In: *Discrete Optimization* 19 (2016), pp. 79–102. ISSN: 1572-5286. DOI: 10.1016/j.disopt.2016.01.005.

[50] David R. Morrison et al. “Cyclic best first search: Using contours to guide branch-and-bound algorithms”. In: *Naval Research Logistics (NRL)* 64.1 (2017), pp. 64–82. DOI: 10.1002/nav.21732.

[51] Chris J. Myers et al. “iBioSim: a tool for the analysis and design of genetic circuits”. In: *Bioinformatics* 25.21 (July 2009), pp. 2848–2849. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btp457.

[52] Tramy Nguyen et al. “Design of Asynchronous Genetic Circuits”. In: *Proceedings of the IEEE* 107.7 (2019), pp. 1356–1368. DOI: 10.1109/JPROC.2019.2916057.

[53] Alec A. K. Nielsen et al. “Genetic circuit design automation”. In: *Science* 352.6281 (2016). ISSN: 0036-8075. DOI: 10.1126/science.aac7341.

[54] Rob Phillips et al. *Physical Biology of the Cell*. 2nd ed. Garland Science, Oct. 2012. ISBN: 9780429168833. DOI: 10.1201/9781134111589.

[55] Nicholas Roehner and Chris J. Myers. “Directed Acyclic Graph-Based Technology Mapping of Genetic Circuit Models”. In: *ACS Synthetic Biology* 3.8 (Aug. 15, 2014), pp. 543–555. DOI: 10.1021/sb400135t.

[56] Fabio Romeo and Alberto Sangiovanni-Vincentelli. “A theoretical framework for simulated annealing”. In: *Algorithmica* 6.1 (June 1991), p. 302. ISSN: 1432-0541. DOI: 10.1007/BF01759049.

[57] Selahattin Sayil. “PVT Variations”. In: *Noise Contamination in Nanoscale VLSI Circuits*. Cham: Springer International Publishing, 2022, pp. 131–134. ISBN: 978-3-031-12751-9. DOI: 10.1007/978-3-031-12751-9_8.

[59] Christopher Schneider et al. “ROC’n’Ribo: Characterizing a Riboswitching Expression System by Modeling Single-Cell Data”. In: *ACS Synthetic Biology* 6.7 (2017), pp. 1211–1224. DOI: 10.1021/acssynbio.6b00322.

[62] Michael J. Smanski et al. “Functional optimization of gene clusters by combinatorial design and assembly”. In: *Nature Biotechnology* 32.12 (Dec. 1, 2014), pp. 1241–1249. ISSN: 1546-1696. DOI: 10.1038/nbt.3063.

[63] Berend Snijder and Lucas Pelkmans. “Origins of regulated cell-to-cell variability”. In: *Nature Reviews Molecular Cell Biology* 12.2 (Feb. 2011), pp. 119–125. ISSN: 1471-0080. DOI: 10.1038/nrm3044.

[64] William Swartz and Carl Sechen. “New algorithms for the placement and routing of macro cells”. In: *1990 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*. Los Alamitos, CA, USA: IEEE Computer Society, 1990, pp. 336–339. DOI: 10.1109/ICCAD.1990.129918.

[65] Phrangboklang L. Thangkhiew and Kamalika Datta. “Scalable in-memory mapping of Boolean functions in memristive crossbar array using simulated annealing”. In: *Journal of Systems Architecture* 89 (2018), pp. 49–59. ISSN: 1383-7621. DOI: 10.1016/j.sysarc.2018.07.002.

[66] Prashant Vaidyanathan et al. “A Framework for Genetic Logic Synthesis”. In: *Proceedings of the IEEE* 103.11 (2015), pp. 2196–2207. DOI: 10.1109/JPROC.2015.2443832.

[67] Cédric Villani. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003. ISBN: 9780821833124.

[68] Leandro Watanabe et al. “iBioSim 3: A Tool for Model-Based Genetic Circuit Design”. In: *ACS Synthetic Biology* 8.7 (2019), pp. 1560–1563. DOI: 10.1021/acssynbio.8b00078.

[69] Clifford Wolf, Johann Glaser, and Johannes Kepler. “Yosys-a free Verilog synthesis suite”. In: *Proceedings of the 21st Austrian Workshop on Microelectronics (Austrochip)*. 2013, p. 97.

[70] Zhen Xie et al. “Multi-input RNAi-based logic circuit for identification of specific cancer cells”. In: *Science* 333.6047 (2011), pp. 1307–1311.

[71] Fusun Yaman et al. “Automated Selection of Synthetic Biology Parts for Genetic Regulatory Networks”. In: *ACS Synthetic Biology* 1.8 (2012), pp. 332–344. DOI: 10.1021/sb300032y.

[72] Richard A. Young and Joan A. Steitz. “Tandem promoters direct E. coli ribosomal RNA synthesis”. In: *Cell* 17.1 (1979), pp. 225–234. ISSN: 0092-8674. DOI: 10.1016/0092-8674(79)90310-6.

[73] Junqi Yuan et al. “RBSA: Range-based simulated annealing for FPGA placement”. In: *2017 International Conference on Field Programmable Technology (ICFPT)*. 2017, pp. 1–8. DOI: `10.1109/FPT.2017.8280114`.

[74] Sunan Zou et al. “PowerSyn: A Logic Synthesis Framework With Early Power Optimization”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43.1 (2024), pp. 203–216. DOI: `10.1109/TCAD.2023.3297069`.

E. Publications

[24] Nicolai Engelmann et al. “Context-Aware Technology Mapping in Genetic Design Automation”. In: *ACS Synthetic Biology* 12.2 (2023), pp. 446–459. DOI: `10.1021/acssynbio.2c00361`.

[58] Tobias Schladt et al. “Automated Design of Robust Genetic Circuits: Structural Variants and Parameter Uncertainty”. In: *ACS Synthetic Biology* 10.12 (2021), pp. 3316–3329. DOI: `10.1021/acssynbio.1c00193`.

[60] Tobias Schwarz and Christian Hochberger. “Memristor Based FPGAs: Understanding the Effect of Configuration Memory Faults”. In: *Architecture of Computing Systems*. Ed. by Martin Schulz et al. Cham: Springer International Publishing, 2022, pp. 167–180. ISBN: 978-3-031-21867-5.

[61] Tobias Schwarz and Christian Hochberger. “Technology Mapping of Genetic Circuits: From Optimal to Fast Solutions”. In: *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. ICCAD ’22. San Diego, California: Association for Computing Machinery, 2022. ISBN: 9781450392174. DOI: `10.1145/3508352.3549344`.

F. Supervised Theses

Project Seminars

Lukas Freiberger. "Fehlertolerante Nutzung von Memristor-basierten LUTs". Technische Universität Darmstadt, Jul. 2021.

Tobias Steinbach. "Fehlertolerantes Technology Mapping". Technische Universität Darmstadt, Jul. 2021.

Erik Kubaczka. "Branch-and-Bound-basiertes Technology Mapping für Genetische Schaltungen". Technische Universität Darmstadt, Oct. 2021.

Christoph Flothow. "QRBDD-based LUT Mapping". Technische Universität Darmstadt, Apr. 2023.

Bachelor's Theses

Katja Münch. "Optimierung von Schaltungs-Bibliotheken für das Technology Mapping in der Synthetischen Biologie". Technische Universität Darmstadt, Nov. 2019.

Tobias Steinbach. "Implementierung eines fehlertoleranten Technology Mapping-Verfahrens in VPR". Technische Universität Darmstadt, Jan. 2022.

Lukas Freiberger. "Evaluation des FPGA-Routings mit fehlerbehafteten Elementen". Technische Universität Darmstadt, Feb. 2022.

Jonas Schoenen. "Entwurf fehlertoleranter Logik-Elemente für NV-FPGAs". Technische Universität Darmstadt, Sep. 2022.

Mohamed Saleh. "Optimierung einer Bibliothek zur Erzeugung von Binary Decision Diagrams". Technische Universität Darmstadt, Oct. 2023.

Stefan Reichel. "Entwurf und Evaluation einer fehlertoleranten Architektur für FPGA-Switchbox-Elemente". Technische Universität Darmstadt, Jan. 2024.

Master's Theses

Lucas Gaia de Castro. "Fehlertolerante Routing-Elemente für Non-Volatile Field-Programmable Gate Arrays". Technische Universität Darmstadt, Oct. 2022.

Christoph Flothow. "Technology Mapping für Lookup-Tables auf Basis von Binary Decision Diagrams". Technische Universität Darmstadt, Oct. 2023.

G. List of Abbreviations

AHU Aho, Hopcroft and Ullman

AIG AND-Inverter Graph

BFS breadth-first search

BeFS best-first search

B&B Branch-and-Bound

DAG directed acyclic graph

DFS depth-first search

DNA deoxyribonucleic acid

E. coli Escherichia coli

E-Score expectation-based score

EDA electronic design automation

FIFO first in - first out

FPGA field programmable gate array

GDA genetic design automation

GRN gene regulatory network

HDL hardware description language

mRNA messenger-RNA

NIG NOR-Inverter Graph

OBDD	ordered binary descision diagram
PVT	process, voltage, temperature
RBS	ribosome binding site
RNA	ribonucleic acid
RNAP	RNA polymerase
RPU	relative promoter units
RTL	register transfer level
SA	Simulated Annealing
SBOL	synthetic biology open language
SAT	Boolean satisfiability problem
TetR	tetracycline repressor
TF	transcription factor
tRNA	transfer-RNA
VPR	Versatile Place-and-Route
YFP	yellow fluorescent protein