TECHNISCHE
UNIVERSITÄT
DARMSTADT

Data Driven Compact Modeling of a Reconfigurable FET

DISSERTATION

Vom Fachbereich Elektro- und Informationstechnik
der Technischen Universität Darmstadt

zur Erlangung des akademischen Grades
eines Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

genehmigte Dissertation

von

Maximilian Jürgen Reuter

Erstgutachter:	Prof. Dr.-Ing. Klaus Hofmann
Zweitgutachter:	Prof. Dr.-Ing. Dr. h. c. Jürgen Becker
Tag der Einreichung:	25.06.2024
Tag der mündlichen Prüfung:	28.10.2024

Darmstadt 2024

Maximilian Jürgen Reuter: *Data Driven Compact Modeling of a Reconfigurable FET*

Darmstadt, Technische Universität Darmstadt
Jahr der Veröffentlichung auf TUprints: 2024
URN: urn:nbn:de:tuda-tuprints-286299
Tag der mündlichen Prüfung: 28.10.2024

"Essentially, all models are wrong, but some are useful."

— George Box & Norman Draper [1]

Abstract

This work contributes to predictive circuit simulation of emerging semiconductor devices by proposing two approaches to obtain characteristic device data from technology simulation, resulting in table models. From these intermediate table models, two different machine learning approaches provide candidates of machine learning based compact models. The methods are demonstrated for an ambipolar transistor, called the *planar RFET*, which allows dynamic reconfiguration of channel polarity at runtime through an additional gate electrode. As a first approach, a cluster simulation tool *PyTaurus* is proposed and allows efficient setup, simulation and refinement of table models from a factorial setup of DC parameter sweeps. To reduce model building time, PyTaurus provides a cluster simulation functionality and distributes the simulation deck to available computation nodes. A second approach, *pseudo transient simulation*, covers the bias space of the device under test within a single slowly proceeding transient simulation. Nesting of input voltages is achieved by a systematic setup of frequencies, that drive the respective harmonic electrode voltage signals. The resulting data sets satisfy structural constraints for table models in Verilog-A.

Beyond the use as plain table models, however, the obtained data sets are transformed to predictive compact models using machine learning. The high dynamic range, that the drive current of the planar reconfigurable FET (RFET) features throughout the operating regions, leads to a solution with an ensemble model. A linear model focuses on high drive currents, while a second model is trained on logarithmically transformed current samples to provide accuracy into the regime of leakage current. For the respective models, two approaches are evaluated: The deep learning approach leads to multilayer perceptrons, which are then sequentially implemented in Verilog-A. The predictions are obtained through inference of the compact model voltages in each simulator step. As a second approach, symbolic regression is employed to optimize an analytical model without structural constraints. The obtained closed form expressions can directly be implemented in Verilog-A. In addition to the DC drive current model, a transient model is formed by symbolic regression, exclusively, as the dynamic range and the expected complexity of the charge model are lower than with drive current.

The resulting neural network-based models show improvement over table models when it comes to DC simulation of digital cells. Transient simulation for timing characterization leads to similar accuracy than the table models, with a maximum deviation of 5.1% from the technology CAD (TCAD) reference. Neural network-based models accelerate the simulation with a factor of up to $17\times$. Symbolic regression-based drive current models further improve computational efficiency, but show insufficient accuracy for predictive circuit simulation.

The concluding result of this work is that it is recommended to transform a table model into a neural network-based compact model using the proposed data driven approach, which requires minimal domain knowledge. Symbolic regression is successfully employed for modeling of electrode charges, but accurate drive current modeling requires further work.

Zusammenfassung

Diese Arbeit trägt zur prognostischen Schaltungssimulation von neuartigen Halbleiterbauelementen bei, indem zwei Ansätze für das Erstellen von charakteristischen Datensätzen aus Technologiesimulation gezeigt werden, die in Tabellenmodellen resultieren. Basierend auf diesen vorläufigen Tabellenmodellen erzeugen zwei unterschiedliche Ansätze aus dem Bereich des maschinellen Lernens Kompaktmodellkandidaten. Die Methoden werden anhand eines ambipolaren Transistors, genannt *planarer RFET*, gezeigt, welcher die dynamische Rekonfiguration der Kanalpolarität zur Laufzeit mit Hilfe eines einer zusätzlichen Gate-Elektrode erlaubt. Als erster Ansatz wird ein Programm zur Verteilung Simulationsanordnungen, welches das effiziente Erstellen, Simulieren und Verbessern von Tabellenmodellsimulationen aus faktoriellen Anordnungen von DC Parameter-Sweeps. Um die Modellierungszeit zu verkürzen erlaubt es PyTaurus die Simulationen an verfügbare Netzwerkknoten zu verteilen. Ein zweiter Ansatz, die *pseudotransiente Simulation*, deckt den Raum der Eingangsspannungen des Bauelements innerhalb einer einzelnen langsam voranschreitenden Transientensimulation ab. Die Verschachtelung von Eingangsspannungen wird hierbei durch die systematischen Anordnungen von Frequenzen der harmonischen Eingangsspanungssignalen erreicht. Der resultierende Datensatz erfüllt die strukturellen Bedingungen für Tabellenmodelle in Verilog-A.

Über die direkte Nutzung als Tabellenmodell hinausgehend, werden die generierten Datensätze durch Methoden des maschinellen Lernens in prognostische Kompaktmodelle überführt. Der große Dynamikumfang, den der Kanalstrom des planaren RFET in verschiedenen Arbeitsbereichen zur Verfügung stellt, führt zu einer Lösung mit einem zusammengesetzten Modell. Ein lineares Modell fokussiert sich auf hohe Kanalströme, während ein weiteres Modell auf den Daten von logarithmisch transformierten Kanalströmen trainiert wird um Genauigkeit bis in den Bereich von Leckströmen zu erhalten. Für die jeweiligen Modelle werden zwei Ansätze evaluiert: Der Deep-Learning Ansatz führt zur Struktur eines Multilayer Perceptrons, welches dann sequenziell in Verilog-A implementiert wird. Die Vorhersagen werden durch Inferenz mit den Eingangsspannungen des Kompaktmodells in jedem Simulatorschritt erzeugt. Als zweiten Ansatz wird symbolische Regression eingesetzt, welche ein analytisches Modell ohne strukturelle Eingrenzungen optimiert. Die dadurch erhaltenen Ausdrücke in geschlossener Form können direkt in Verilog-A implementiert werden. Zusätzlich zum DC Kanalstrommodell wird ein Transientenmodell ausschließlich durch symbolische Regression erzeugt, da der Dynamikumfang und die erwartete Komplexität des Ladungsmodells geringer sind als des Kanalstrommodells.

Die resultierenden auf neuronalen Netzen basierenden Modelle zeigen Verbesserungen gegenüber den Tabellenmodellen bezüglich DC Simulation von digitalen Zellen. Transientensimulation für die Charakterisierung von Zeitverhalten führt, mit einer Abweichung von der TCAD-Referenz von maximal 5.1%, zu einer ähnlichen Genauigkeit, wie sie die Tabellenmodelle zeigen. Die auf neuronalen Netzen basierenden Modelle beschleunigen außerdem die Schaltungssimulation um bis zu $17\times$. Auf symbolischer Regression basierende Kanalstrommodelle verbessern die Effizienz der Berechnung noch weiter, bieten jedoch nicht die notwendige Genauigkeit für prognostische Schaltungssimulation.

Das zusammenfassende Ergebnis dieser Arbeit ist die Empfehlung ein Tabellenmodell in ein auf neuronalen Netzen basierendes Kompaktmodell zu überführen, indem der vorgeschlagene, auf wenig Spezialwissen basierende Ansatz verwendet wird. Symbolische Regression wird erfolgreich für Ladungsmodelle der Elektroden eingesetzt, für die präzise Modellierung von Kanalströmen ist jedoch weitere Arbeit nötig.

Acknowledgements

My special thanks go to Klaus Hofmann, who provided me with the mixture of guidance and freedom a junior scientist can only dream of.

Jens Trommer and Johannes Pfau have my gratitude for all the positive spirit, encouragement and dedicated collaboration during our joint research projects PARFAIT I, II and SENSOTERIC.

For exceptional assistance with technology models – day and night – I thank Tillmann Krauss.

Further, I would like to thank Andreas Kramer for his thorough and valuable feedback.

My parents enabled my career in the first place, for which I am very thankful.

Thank you, Dorsa, for your loving support and all the shared excitement.

Contents

List of Figures

List of Tables

List of Acronyms

ANN	artificial neural network
AIC	Akaike information criterion
API	application programming interface
BIC	Bayesian information criterion
BG	back gate
BSIM	Berkeley Short-Channel IGFET Model
BSIM-IMG	Berkeley Short-Channel IGFET Model - Independent Multi-Gate
BSON	Binary Java Script Object Notation
CG	control gate
CMC	Compact Model Council
CMD	command
CMOS	complementary MOS
CPU	central processing unit
DoE	Design of Experiments
DUT	device under test
DTCO	design technology co-optimization
EI	expected improvement
FVM	finite volume method
FET	field-effect transistor
FG	front gate
FPGA	field-programmable gate array
FD-SOI	fully depleted silicon on insulator
HiSIM	Hiroshima-University STARC IGFET Model
JSON	Java Script Object Notation
Lat1	lateral terminal 1
Lat2	lateral terminal 2
MAE	mean absolute error
MAPE	mean absolute percentage error
MLP	multilayer perceptron
MOS	metal-oxide-semiconductor

MOSCAP	metal-oxide-semiconductor capacitor
MOSFET	metal-oxide-semiconductor field effect transistor
MSE	mean squared error
NoSQL	Not only SQL
NMOS	n-type MOSFET
OFAT	one-factor-at-a-time
PARDISO	PARallel Direct SOlver
PG	program gate
PMOS	p-type MOSFET
ReLU	rectified linear unit
RF	radio frequency
RFET	reconfigurable FET
RPS	run parameter set
SOI	Silicon-On-Insulator
SIMD	single instruction/multiple data
sMAPE	symmetric mean absolute percentage error
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	static random access memory
TCAD	technology CAD
TFET	band-to-band tunneling transistor
TG	top gate
TFE	thermionic field emission
TMD	transition-metal dichalcogenide
TRL	technology readiness level
UI	user interface
VLSI	very-large-scale integration
VTC	voltage transfer characteristic

List of Symbols

Symbol	**Unit**	**Denotation**
γ	$\sqrt{V}$	Body effect factor
ΔI	A	Transient mismatch in pseudo transient simulation
$\Delta\Phi_e$	V	Work function difference between electrode e and a reference electrode
$\Delta V_{sweep,prune}$	V	Minimum granularity of a parameter sweep with respect to a sweep variable V_{sweep}
η_r	-	Performance score of runner r
θ	-	Space of parameters for symbolic regression
λ	-	Set of hyperparameters
σ_a	A	Variance of Gaussian noise for augmentation
φ	-	Probability density function
ϕ	-	Space of analytical expressions for symbolic regression
Φ	V	Electrostatic potential
Ψ_s	V	Surface potential at a surface s
$b_{l,n}$	-	Bias of a perceptron n in layer l
C_{ox}	F	Oxide capacitance
$\mathcal{D}$	-	Set of samples, corresponds to $\{(X_i, y_i)\}_{i=1}^{N_s}$
$\mathcal{D}_{train}^{i,\sigma_a}$	-	Training set augmented with Gaussian noise ($\sigma = \sigma_a$), replica i
E	-	Tuple of device electrodes $E = (E_1, E_2, ...)$
E_s	$\frac{V}{m}$	Electric field at a surface s
$f_{e,i}$	Hz	Frequency of electrode i
$g(u)$	-	Activation function
H	-	Hyperparameter
I_{DC}	A	Transistor drive current (DC)
I_{lb}	A	Lower bound for current values in a data set
I_c	A	Capacitive current
I_r	A	Resistive current
I_{on}/I_{off}	A / A	On-current / Off-current of a transistor
$L(\hat{y}, y)$	-	Loss function
m (, m_i)	-	Nesting factor of a pseudo transient simulation (per dimension i)
N_c	cm^{-3}	Density of states in the conduction band of silicon
N_d	-	Dimensionality of inputs (e. g. for a neural network model)

N_e	-	Number of electrodes; does not include the reference electrode
$N_{ops,nn}$	-	Number of scalar arithmetic operations of a neural network model
N_{os}	-	Oversampling Factor
N_s	-	Number of samples in a data set
q	C	Elementary charge
Q_e	-	Tuple of electrode charges $(Q_{e,1}, Q_{e,2}, ...)$
$S_{e,i}$	-	Set of samples per electrode E_i, $S_{e,i} = \{V_x, V_y, ...\}$
S_I, S_Q	-	Linear scaling factor of the drive current models and charge models
t_{CPU}	s	CPU time of a simulation
$t_{tr,\{rise,fall\}}$	s	Rise-time and fall-time of a digital signal
$T_{\{si,ox\}}$	m	Thickness of silicon body or gate oxide layer
V_{ch}	V	Channel voltage, electron quasi-Fermi potential with respect to the source
V_{DD}, V_{SS}	V	Supply voltages
V_e	-	Tuple of electrode voltages $(V_{e,1}, V_{e,2}, ...)$
V_{fb}	V	Flat band voltage
V_t	V	Thermal voltage
V_{th}	V	Threshold voltage
$w_{l,n,i}$	-	Input weight of a perceptron n in layer l for input i
W_e	eV	Work function of electrode e
$\boldsymbol{W}_l$	-	Weight matrix of a layer l within a multilayer perceptron
$\boldsymbol{X}_{in}$	-	Input vector of a neural network
$\boldsymbol{X}_l$	-	Output vector of a layer l within a neural network
$\hat{y}$	-	Prediction of y
$\bar{y}$	-	Average of ground truth values of y

1 Introduction

The world we live in today embraces emerging technologies more than ever, and this is only partly owed to a spirit of pioneering and curiosity. We foresee that continuous improvement of existing technologies is unlikely to be the solution for the challenges of the 21st century. A fair amount of work should be dedicated to enable and facilitate progress during the early stages of inventions, that shape our environment.

Particularly, semiconductors enhance our life in various aspects, as they provide processing of information in an increasing number of every-day appliances. Processing capabilities largely depend on the amount of transistors integrated into a unit area, so that integration density has become one of the most important figures to describe the performance of a semiconductor technology. A key to success of very-large-scale integration (VLSI), however, lies not only in the actual manufacturing capabilities. Sophisticated device models, based mostly on analytical formulations of the physical mechanisms in a transistor, enable reliable design processes for well established technologies. On the one hand, the semiconductor industry owes at least part of its success to the traditionally rather incremental improvements [2]. On the other hand, incremental improvements of a device or a process sooner or later run into limitations, which are typically physical in nature. In order to extend Moore's law [3] into the next decades, the search for disruption is, to the present day, in full swing and allows novel device concepts to emerge.

In the domain of field-effect transistors (FETs), the journey to a robust and powerful semiconductor model, a compact model, is challenging. It is often easier to refine models along with modest changes in the target technology, than to describe fundamentally new mechanisms and structures. Accuracy of a productively used compact model is essential to design a functioning integrated circuit. In emerging technologies, however, the focus is typically not on conducting a complete integrated circuit design flow. Required is rather the prediction of the performance of a device, characterized either by measurement or technology simulation, in the simulation of circuits. Although accuracy remains one of the central aspects, this imposes different requirements on predictive models: Predictive device models can spare many of the parameters, which do not contribute to technology assessment at an early stage. Parameters such as temperature or channel length, featured by industrial compact model, are typically not of interest for predictive circuit simulation. Instead, the creation of the model has to be quick and straightforward, in order to gain performance estimates in an early phase of the endeavor, and most importantly to predict behavior in a system-environment. These estimations can then be incorporated into the decision about adoption of a technology.

1.1 Research Scope and Objective

This work contributes to predictive circuit modeling of emerging devices, by streamlining the generation of table models and the subsequent transition to analytic compact models. The focus lies on the methods to be data driven with as little domain knowledge as possible, in order to apply them to semiconductor devices, where detailed analytic models of conduction are not yet available. In particular, the predictive, and yet accurate, simulation of digital circuits is the goal, which requires the model to provide accuracy over a wide range of bias regions. Efficient implementation is a key factor to allow the simulation of circuits with a high transistor count. The specific device under test (DUT) and subject of demonstration is a planar reconfigurable FET (RFET), which, due to its reconfigurable polarity, cannot be represented by conventional and industry-standardized compact models.

The scope of this work starts where the technology model of the DUT is available. Improvements on technology level are not subject of this work and neither is physical characterization. In consequence, this ensures that the proposed methods of generation and transformation of the data remain general and applicable to a wide range of emerging technologies. Table models are an established method for circuit simulation of emerging devices, but the methods to generate them are often cumbersome. The intermediate objective is therefore the generation of characteristic device data, which allow direct use as a table model. Particular attention is paid to efficient simulation design of technology simulation, as generation of characteristic datasets from technology simulation is the most time-consuming step in this work. Table models are typically inferior to compact models, with respect to performance and accuracy. The main objective is therefore to exploit the generated data set to form predictive compact models for the planar RFET by using machine-learning based modeling approaches.

The fundamental question that resonates with these objectives is: Is the machine learning-based compact model an improvement over the direct use of the original table model?

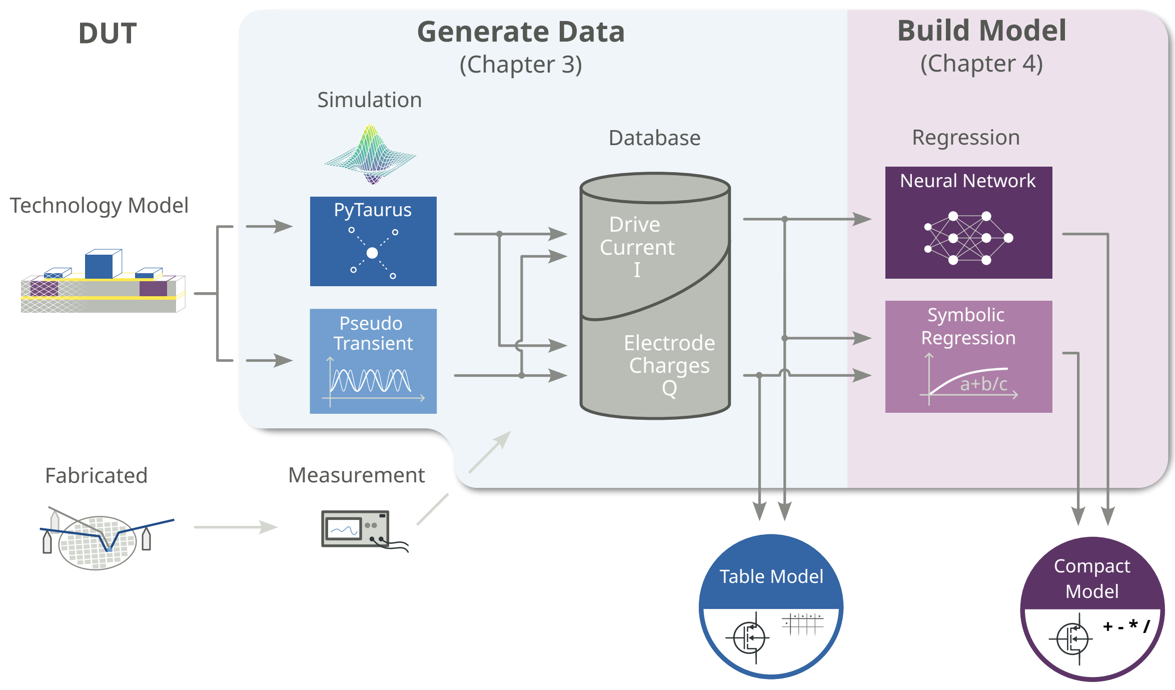


Figure 1.1: The implementation part of this work includes two methods of generating characteristic device data, which are then used to build data driven compact models using machine learning.

1.2 Thesis Outline

The remainder of this work is structured as follows: The fundamentals of RFET technologies, the DUT and possible application in circuits are explained in Section 2.1. The various semiconductor modeling approaches on different abstraction levels, from technology modeling and data generation to compact models, are subject of Section 2.2. Lastly, the fundamentals of this work comprise the topic of regression-based machine learning and introduce deep learning and symbolic regression in Section 2.3.

Figure 1.1 provides an overview over the implementation part of this thesis. The proposed methods to obtain characteristic device data for table models, the factorial simulation cluster PyTaurus and the pseudo transient approach, are described in Chapter 3 and lead to 2 table models. Having generated a data set of bias points, in Chapter 4 symbolic regression is applied to the electrode charges, in order to obtain analytic equations for a transient model. One modeling approach for a DC model is based on deep learning, while a competing candidate is created by using symbolic regression. In total, two approaches to data sets (PyTaurus and pseudo transient approach) and two approaches to DC models (deep learning and symbolic regression) result in 4 compact model candidates.

The 4 compact model candidates, along with the 2 table model candidates are evaluated in Chapter 5.

2 Fundamentals and State of the Art

The technology modeling approaches proposed in this work, although based on data-driven methods, require understanding of the target technology to some extent. The modeled technology, RFETs, are described in Section 2.1. Within the scope of data driven modeling, the fundamentals consist of the methodology to produce device characteristic data (Section 2.2.1), state-of-the-art modeling techniques, such as compact models (Section 2.2.2), and how to construct data driven device models (Section 2.2.3). The proposed compact modeling approaches rely on machine learning methods, namely symbolic regression and deep learning, which are introduced in Section 2.3.

2.1 Reconfigurable FETs

Conventional metal-oxide-semiconductor field effect transistors (MOSFETs) technologies rely on chemically doped semiconductors, providing two complementary device types: n-type (NMOS) and p-type (PMOS). The distinction comes from the type of dopant and is therefore decided at design time. Static complementary MOS (CMOS) logic design incorporates both transistor types, achieving rail-to-rail output with strong drive capability. In this section, an emerging device type, the RFET, is introduced, which features tuneable polarity at runtime. The mechanisms behind this feature are analyzed in Section 2.1.1. In order to provide general insights into the device which serves as DUT in this thesis, the planar RFET, Section 2.1.2 presents structure and characteristics. The proposed modeling approach in this work targets circuit simulation of logic cells, hence Section 2.1.3 provides an overview of established circuit topologies, which feature RFETs. The predicted implications and opportunities for circuit design are summarized.

2.1.1 RFET Fundamentals and Prototypes

Coming from the background of conventional MOSFETs, where digital circuits typically consist of two types of devices with similar structure and complementary chemical dopants, the general principle of RFETs is not far off: Doping is still required, but instead of chemically induced impurity doping, the effect is caused by the electric field in semiconductor material and hence called *electrostatic doping* [4], [5]. Conventional CMOS FETs use the electric field caused by applying a voltage between gate and bulk electrode to create an inversion layer near the gate dielectric interface, which leads to a conductive channel. This channel then enables charge transport by providing free minority carriers. Similarly, electrostatic doping describes the influence on the concentration of free charge carriers in semiconductor material by biasing an adjacent gate [6]. However, one of the main differences is, that electrostatic doping leads to volume inversion or volume accumulation – a feature of ultrathin-body technologies, with a maximum body thickness of approximately $10\,nm$ [7]. This distinction can be seen in

Figure 2.1, where a metal-oxide-semiconductor capacitor (MOSCAP) in thick-body Silicon-On-Insulator (SOI) technology is shown, along with an ultrathin-body fully depleted silicon on insulator (FD-SOI) MOSCAP. While the 2D inversion layer of the thick-body structure is described by classical physics, quantum mechanical effects in ultrathin bodies cause the charge carriers to escape 2D confinement at the interface, when a sufficiently high electrostatic field is applied. The substrate then enters volume inversion/-accumulation [8], [9]. As a result, the concentration of free carriers is distributed throughout the ultrathin body, resembling a chemically doped region in the semiconductor [5].

Unlike chemical doping, however, electrostatic doping can be applied dynamically at any time by biasing the according gate electrode. A typical application for electrostatic doping is when it comes to low dimensional materials such as transition-metal dichalcogenides (TMDs) and graphene, where the introduction of impurities into the atomic lattice without damaging the device is challenging. For these materials, electrostatic doping is already the standard doping technique [10]. Further, it is likely, that electrostatic doping can compensate increasing issues of chemical doping in highly scaled semiconductors, such as the formation of high doping gradients and the general spatial distribution due to complex geometry and growth dynamics [10]–[13].

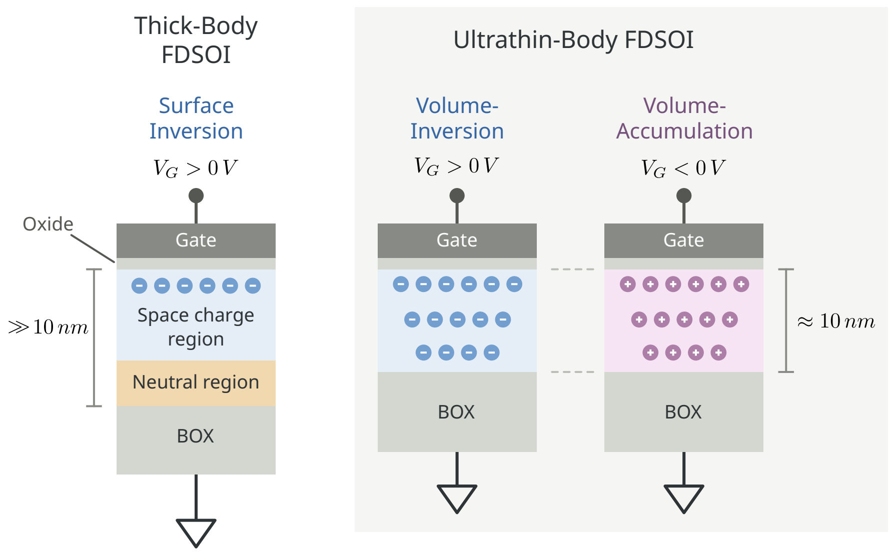


Figure 2.1: Unlike thick body technologies, ultrathin bodies down from $\approx 10\,nm$ thickness offer volume inversion/accumulation, providing free charge carriers. The effect is similar to impurity doping [5].

Electrostatic doping is one of the enabling mechanisms for RFETs, as it allows the reconfiguration of device polarity between n-type and p-type by providing the respective spatial carrier concentration. While there are other working principles, such as band-to-band tunneling [14], single-electron and spin transport [13], [15], this work focuses on the large group of Schottky barrier implementations. The principle of Schottky barrier RFETs is to program the doping of the transistor body, causing band bending with respect to Schottky barrier junctions at source and drain and selecting the conduction type [13]. Source and drain are typically silicide contacts. Two types of Schottky barrier implementations are distinguished by Mikolajick et al.

[13] as shown in Figure 2.2. In type a) the polarity of the body region is selected by a buried program gate (PG) over the entire channel length. The electrostatic doping caused by V_{PG} in type a) results in band bending along the channel with respect to the Schottky barriers and enables either n- or p-type conduction between the silicide regions, which form source and drain. Charge transport is then controlled via a centered gate electrode, the control gate (CG), which modulates a potential barrier in the respective channel segment below. Depending on V_{CG}, a depletion region forms in the middle of the channel and impedes the drive current.

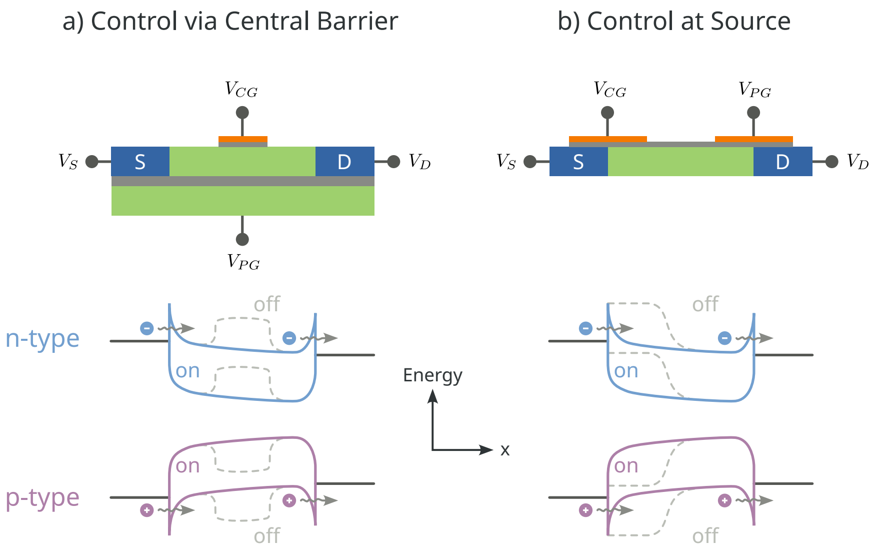
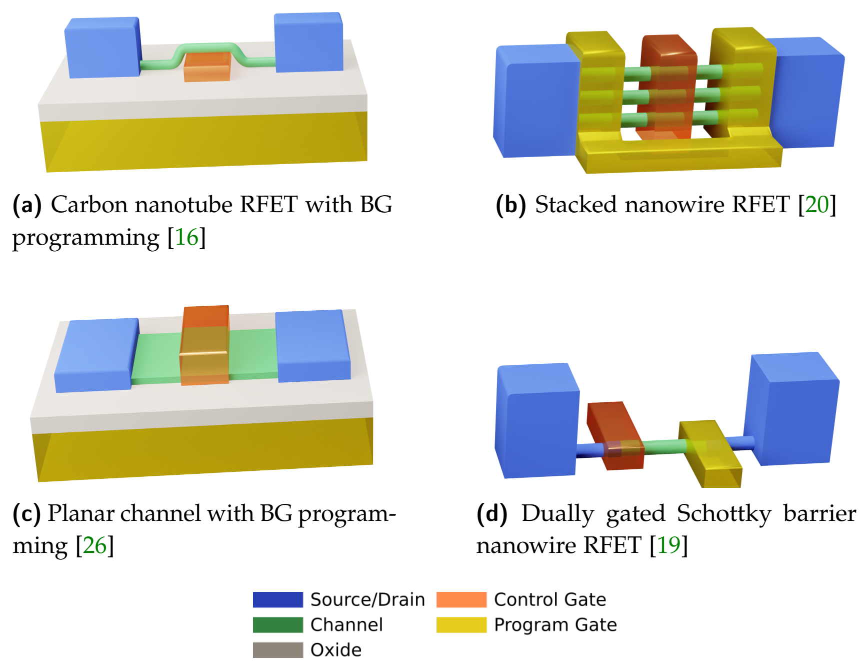


Figure 2.2: Two types of Schottky barrier RFETs are distinguished by Mikolajick et al. [13]: Type a) tunes polarity through the back gate (BG) and locally depletes the channel through a central top gate (TG). Type b.) directly influences the Schottky barriers at source and drain.

Type a) is attributed to Lin et al. [16], [17] in 2004, who achieve the elimination of ambipolar charge transport in favor of clear n-/p-type conduction in a carbon nanotube channel, placed over an aluminum gate electrode, as shown in Figure 2.3a. The BG electrode is used for polarity configuration, while the centered aluminum gate between BG oxide and the carbon nanotube switches the transistor on and off. In pursuit of improving the programming capabilities, which have up to then typically been offered by the BG [18], [19], De Marchi et al. [20] present a top-down fabricated stacked silicon nanowire RFET, shown in Figure 2.3b. Placing the PG over and especially around the Schottky junctions at source and drain increases the on-/off current ratio and the use of weakly p-doped nanowires benefit symmetry of n- and p-type conduction. Vertically stacking the nanowire channels further increases the drive current without demanding more area. The resulting device achieves drive currents up to $75\,\mu A$ for the n-type configuration at a programming voltage of $V_{PG} = -4\,V$; a logic gate demonstration shows a symmetric inverter with a steep transition for $V_{DD} = 1\,V$. Another device with a central CG emerges by Wessely et al. [21], [22] in 2012, who use a silicon nanowire channel above which the CG electrode is placed. The trend at Technical University of Darmstadt, however, moves towards planar devices as Krauss et al. [23]–[27] show reconfigurability of



(a) Carbon nanotube RFET with BG programming [16]

(b) Stacked nanowire RFET [20]

(c) Planar channel with BG programming [26]

(d) Dually gated Schottky barrier nanowire RFET [19]

Figure 2.3: Distinct RFET prototypes, based on different structures and materials, have been proposed in literature.

FD-SOI based planar channel geometries down to a length of $250\,nm$, as shown in Figure 2.3c. Drive currents in μA region are reached by a BG bias of at least $|V_{PG}| = 2\,V$ and $|V_{DS}| = 1\,V$. These planar devices are the predecessors of the planar RFET, which is characterized in this work and described in Section 2.1.2.

Devices of type b) in Figure 2.2 allow the control of carrier injection directly at source, while the PG bias at drain blocks the respectively other carrier type [13]. This RFET type goes back to Weber et al. [19] who propose a dually gated silicon nanowire FET. During annealing, the nanowire forms nickel-silicide (NiSi) segments, extending the nickel source and drain regions from both sides towards the channel [18]. As shown in Figure 2.3d, the CG is placed over the Schottky junction at source to control the on/off state of the device. Located at the drain side, the PG allows selection of device polarity. Heinzig et al. [28] propose a similar top-gated silicon nanowire RFET but achieve an enhancement regarding I_{on}/I_{off} ratio from $\approx 10^3$, reported by Weber et al. [19], to $10^8/10^9$ in highly symmetrical p-/n-type mode. Follow-up devices feature further improvements by inducing mechanical stress as a measure to precisely tune symmetry, which then allows the fabrication of a CMOS style inverter [29], [30]. Another significant step ahead is the co-integration of back gated RFETs within the GlobalFoundries 22FDX platform by introducing NiSi drain and source regions without changes to design rules and without the need for additional masks [31], [32]. Logic operation at $V_{DD} = 0.8\,V$ is enabled by a program voltage of $|V_{PG}| = 5.2\,V$ at the BG, which allows currents of up to $10\,\mu A/\mu m$ [31].

Symmetry between p- and n-type conduction is particularly important for a device, which aims at providing the flexibility of runtime polarity configuration. Conventional CMOS technologies can adapt to material properties and allow scaling of n-type MOSFET (NMOS) and p-type MOSFET (PMOS) channel width according to their targeted drive capabilities, but this is only possible because their respective role within the circuit is predetermined at design time. For devices where the decision between n- or p-type conduction is not made ahead of fabrication, structural adaptations cannot be exploited to balance drive strength and switching symmetry. As a consequence and similar to conventional CMOS scaling attempts, other materials are explored to eventually replace silicon and provide better symmetry, higher I_{on}/I_{off} ratios and lower supply voltages [33]. An important path is taken by Trommer et al. [34], who compensate the lower band gap energy $(0.66\,eV)$ of a germanium nanowire transistor with electrostatic control over the respective Schottky barriers. With this approach, off-state leakage is suppressed and the advantages of germanium, i. e. low threshold voltages $V_{th,n} = 0.4\,V$,$V_{th,p} = -0.2\,V$ and high on currents, can be exploited. Germanium nanowire FETs offer an additional feature: Böckle et al. [35] find distinctive electrostatically tuneable negative differential resistance at room temperature in a germanium nanowire transistors, which is subsequently incorporated into a germanium nanowire RFET by Sistani et al. [36].

From a material point of view, the combination of Schottky junctions with fully depleted body appears to be suitable for operation in an extended temperature range, as two major sources of temperature dependent leakage are avoided: Firstly, the PN junctions between source/drain and the channel, and secondly the body diode between drain and bulk [37]–[39]. Especially for RFETs with virtually intrinsic body, the lack of chemical doping induced impurities is expected to further reduce phonon scattering effects and carrier freezeout, and therefore temperature dependence [37], [40]. Galderisi et al. [41] analyze the characteristic of a three-gated RFET over the temperature range of $80\,K$ to $475\,K$ and observe that the leading conduction mechanism at low temperature, thermionic field emission (TFE), leads to increasing threshold voltage at decreasing drive current towards the lower end of the observed temperature range. It is further stated, that this effect can be mitigated with thinner Schottky barriers, e. g. through higher PG bias, which increases temperature independent tunneling [41]. In high temperature regions up to $425\,K$, the threshold voltage decreases only marginally for ambipolar conduction mode, indicating that the device proposed in [41] can be a suitable candidate for harsh temperature environments. In general, the I_{on}/I_{off}-ratio in Schottky barrier FETs is limited by the band gap of the channel material, and the respective range of current, in case of the three-gated RFET, shifts significantly within the analyzed temperature range [41]. However, apart from material engineering, there is a second way to influence I_{on}/I_{off}, which is the suppression of leakage by exploiting additional PGs [34], [42]. Inherently provided by RFET devices, PGs allow the adjustment of I_{on} and I_{off} independently, as the Schottky barrier can be actively adjusted by electrostatic doping depending on device state and operating temperature.

2.1.2 The Planar RFET

The device subject to characterization in this work follows the line of top-down planar devices by Krauss et al. [23]–[27]. The name of the device concept originates from a specific switching principle, which can be understood as a combination of enhancement- and depletion mode transistor, as described by type a) in Figure 2.2. Similarity to enhancement mode devices arises as the virtually intrinsic body, with a p-type boron background doping of $1 \cdot 10^{15}\,cm^{-3}$, is electrostatically doped via the BG. The BG, in form of an n-type substrate with chemical doping

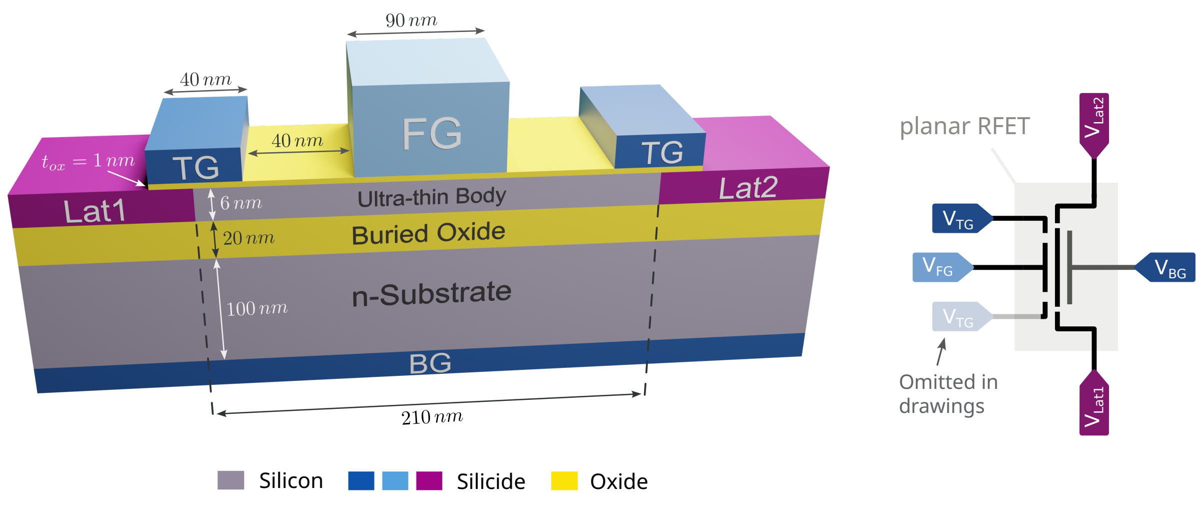


Figure 2.4: The planar RFET is based on a ultrathin-body FD-SOI technology. The BG and the top gates (TGs) allow programming of device polarity, while the front gate (FG) modulates the drive current. Only one TG is drawn in the schematic symbol, as the TGs are electrically connected. Figure is not to scale. [43]

of $5 \cdot 10^{19}\,cm^{-3}$, generates the respective n- or p-type channel in the body. From the top side of the device, the centered front gate (FG) locally depletes the channel by formation of a potential barrier, resembling depletion mode operation. The name *Dehancement Mode FET*, or short *DeFET*, emerges and Krauss [27] describes the respective technology simulation, fabrication and design space evaluation of the dual gate RFET.

In an attempt to increase the control over the mid-gap Schottky barriers at source and drain, and similar to the stacked nanowire approach [20], Krauss et al. [27], [44] propose an extension to the dual gated DeFET concept. To enhance electrostatic doping of the Schottky barriers at source and drain, two gate electrodes, called *top gates*, are placed over the junctions, as shown in Figure 2.4. For the TG variant on a $6\,nm$ thin and $210\,nm$ long body the name *planar RFET* is adopted from [45] to make a clear distinction to the predominantly low dimensional material based RFET prototypes referred to in Section 2.1.1. The FG control mechanism is similar to the devices of type a) in Figure 2.2, while the arrangement of the TGs resembles the structure of type b). The two TGs are, to this date, exclusively used in combination and therefore regarded as a single electrode, modulating channel polarity and therefore Schottky barrier height at the source and drain junctions equally. What started out as an enhancement to BG programming was found to be a good replacement instead, as TG-only programming allows a threshold voltage of $V_{th} \approx 500\,mV$ in logic cells of $V_{DD} = 1.4\,V$. The symmetric structure of the device allows both channel adjacent terminals to act as source and drain of the charge transport, depending on the respective operating point. For the purpose of modeling, this work refers to both channel adjacent terminals as *Lat1* and *Lat2*.

The structure of the planar RFET along with the circuit symbol, which represents the planar RFET in this work, proposed in [46], are shown in Figure 2.4. Although both TGs are used in a configuration where they are electrically connected, the symbol is designed to show only one TG connection to improve schematic readability. The one input at V_{TG} represents both electrically connected TGs.

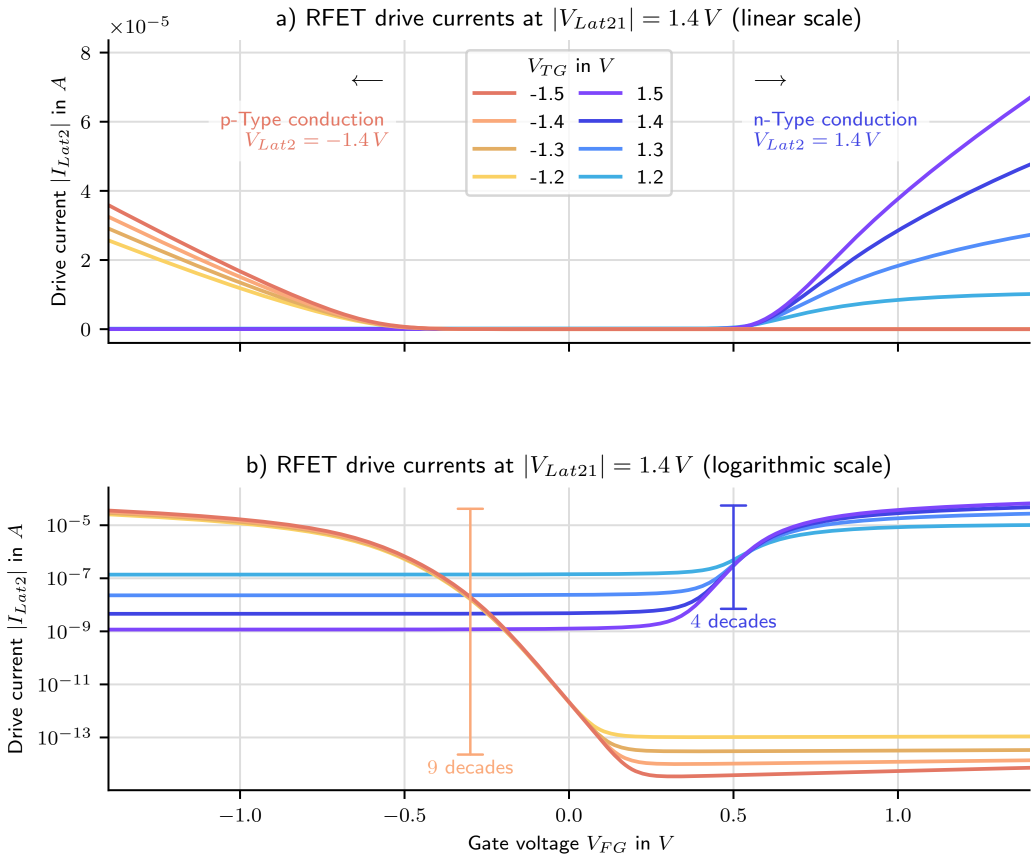


Figure 2.5: The planar RFET, shows a balanced maximum drive current for $|V_{Lat21}| = 1.4\,V$. The I_{on}/I_{off} ratio for n- and p-type conduction modes, however, lacks symmetry [43].

With a background doping of $5 \cdot 10^{17}\,cm^{-3}$ and work functions $W_{TG} = 5\,eV$ and $W_{FG} = 4.75\,eV$, the RFET is optimized for symmetric I_{on} in digital cells of $V_{DD} = 1.6\,V$. Following the approach in [43], in this work, the device is employed to digital cells with $V_{DD} = 1.4\,V$, which leads to the characteristic shown in Figure 2.5. At $V_{Lat21} = 1.4\,V$, the I_{on}/I_{off} ratio of p-type conduction with $\approx 10^9$ exceeds the n-type conduction I_{on}/I_{off} ratio of $\approx 10^4$, while the maximum drive current remains in a similar regime of $40\,\mu A$.

2.1.3 Circuits with Reconfigurable Devices

The goal to reduce chip area, aligned with the typical VLSI trends, extends to RFETs on device level, where a FG length of $20\,nm$ is already demonstrated [31]. In general, conventional static CMOS circuits can be implemented in RFET technologies, by programming the electrostatic doping of the respective devices in a static way, e. g. by hard-wiring PGs to a supply voltage. With only V_{DD} and V_{SS} available, the involved voltages between PGs and the respective source terminal have to be brought to the range of $|V_{PGS}| \leq 2\,V$ and still provide clear polarity control. Enabled by the high symmetry between p- and n-type conduction in [28], Heinzig et al. [29]

present the first inverter circuit based on two statically configured silicon nanowire transistors for $V_{DD} = 2\,V$, as shown in Figure 2.6. The proposed RFET inverter resembles the conventional static CMOS inverter, with the PG of the pull-down device connected to V_{DD} and the PG of the pull-up devices connected to V_{SS}. In the schematic drawing of Figure 2.6 the inverter is depicted as a dual gate device and uses a circuit symbol representation from [13]. The principle of replicating static CMOS logic topology with RFETs can be applied to other common logic gates. INV, NAND, NOR gates show correct switching behavior in technology CAD (TCAD) simulation and their performance is comparable to an SOI technology of similar feature size, as shown in [45]. Using RFETs as statically configured devices in CMOS-style topologies further allows threshold voltage scaling over a wide range, similar to the body-biasing in FD-SOI [47]. A possible use-case for the wide range tuning ability of the planar RFET is region-based dynamic scaling of threshold voltage, e. g. in field-programmable gate arrays (FPGAs), as proposed by Pfau et al. [48].

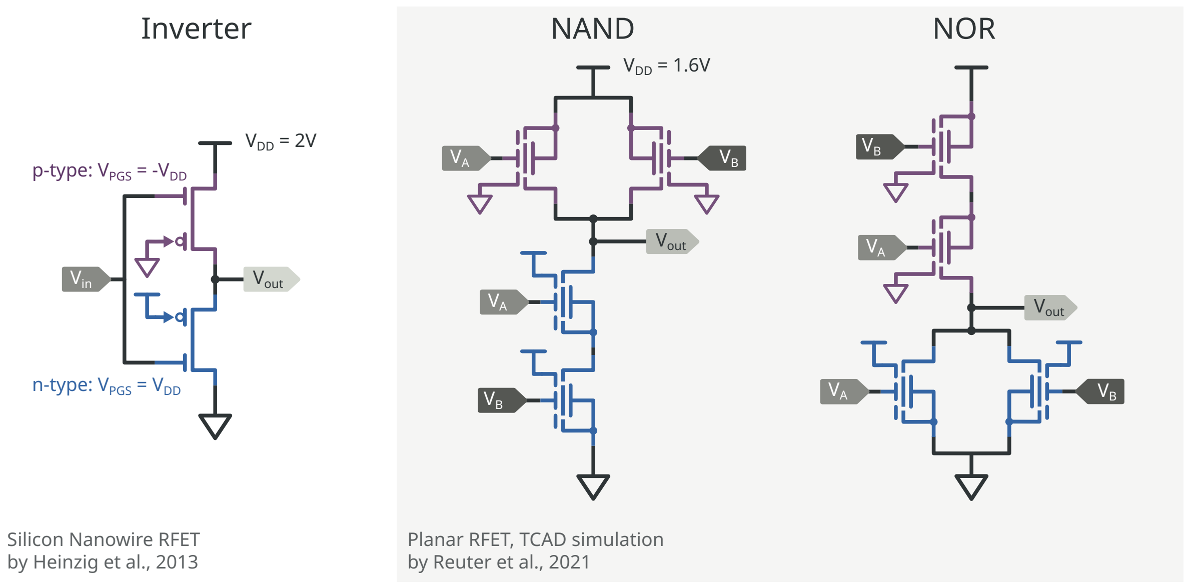


Figure 2.6: RFETs can be programmed statically, by hard-wiring the PG to a supply rail. This allows static CMOS topologies to be adopted for RFET technologies. Heinzig et al. [29] propose a silicon nanowire inverter, *NAND* and *NOR* implementations with the planar RFET are demonstrated in [45].

The mere replacement of NMOS/PMOS devices with statically configured RFETs, however, subjects the success of the RFET technology to a similar down scaling pace as conventional CMOS technologies. Instead, a way to outperform chemically doped CMOS technology can be to exploit the inherent features, such as dynamic reconfiguration. RFETs allow the design of circuits which either change behavior in order to provide multiple functions or to use the PG as additional signal input for reduced transistor count per logic function [49]. Examples of both cases have already been demonstrated in the early phases of RFETs, after Lin et al. [16] proposed the polarity tuneable carbon nanotube RFETs. With similar carbon nanotube structures as [16], O'Connor et al. [50] propose two stage dynamic logic cells, where the PG of the logic devices is used to configure the cell function. Pre-charge and evaluation transistors are configured with static polarity. 8 different forms of binary logic functions using *AND* and *OR* conjunctions are possible with 3 logic transistors in [50]. *XOR* operation requires 5 logic transistors implemented with RFETs.

Fully complementary digital circuits with full output swing are subject to structural constraints: The pull-up branch consists of p-type devices and the pull-down branch consists of n-type devices. Deviation from this topology results in reduced output swing, and devices with wrong polarity in their respective branch are called *misconfigured* [51], [52]. A basic example for this constraint is the transmission gate, where two complementary devices are required for strong drive, to avoid the loss of threshold voltage $V_{th,n/p}$ over the driving transistor. For conventional doped CMOS, this requirement can easily be regarded during logic cell design. But reconfiguring an RFET from n- to p-type conduction in a pull-down branch, and vice versa for a pull-up branch, results in a significant deterioration of output swing. Ben Jamaa et al. [53] conclude that polarity reconfiguration with full output swing in CMOS-style topologies can be established by providing the branches in each pull network as parallel RFETs. The resulting topology is similar to a transmission gate and pairs devices of the respectively wrong polarity configuration with their complementary counterpart. The demonstration, based on a carbon nanotube RFET, includes statically configured circuits, such as the logic cells shown in Figure 2.6, but extends up to more complex static CMOS gates. $Y = \overline{(A \oplus B) \cdot (C \oplus D)}$ is realized with 8 devices, where signal inputs are used for polarity configuration of transmission gate pull branches [53].

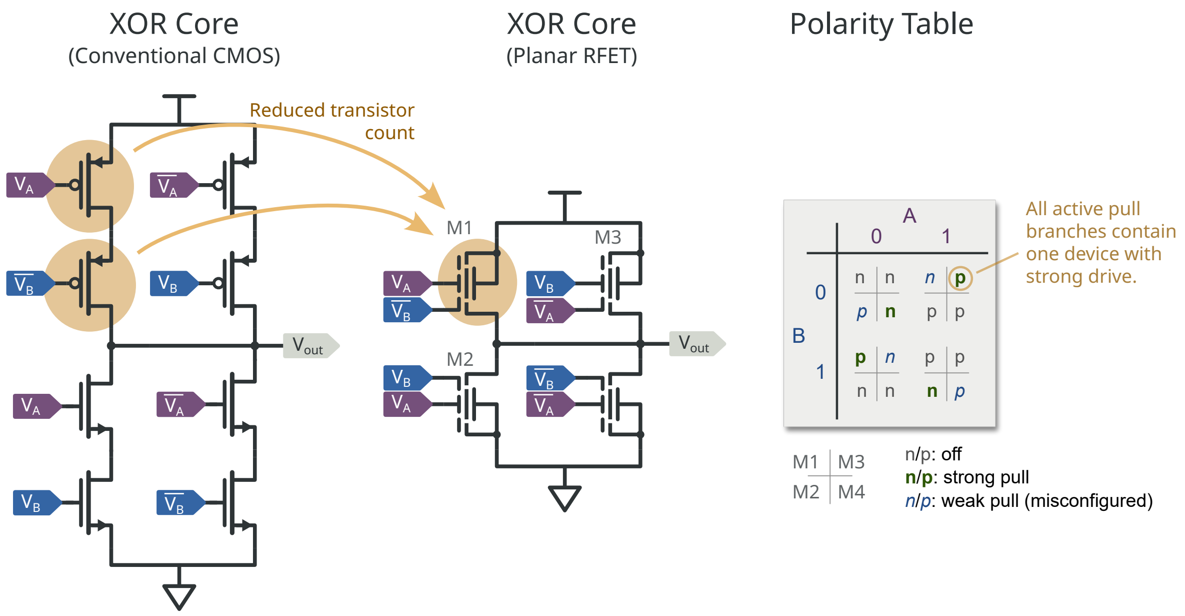


Figure 2.7: The *XOR* gate shows how RFETs can reduce the transistor count by providing higher expressiveness than a conventional MOSFET [45], [54].

A basic example for transistor count reduction, by using PGs as inputs, is the *XOR* gate proposed by Gaillardon et al. [54]. The core of the *XOR* cell consists of 4 RFETs and relies on the availability of inverted inputs. This *XOR* core is analyzed with 4 planar RFETs in [52], as shown in Figure 2.7. The key to the transistor count reduction is the integration of the logic functionality of two serial MOSFETs into one, by using the PG as signal input [53]. The resulting polarity states of each device for each input combination of A and B of the planar RFET implementation is shown in Figure 2.7 and reveals that the active pulling branches in every state consist of a strong pull device, paired with a weak pull device [52]. The left side of the *XOR* core, M1 and M2, can be interpreted as an inverter of A, if $B = 1$, as both devices are configured with correct polarity and therefore provide strong pull, respectively. Therefore, the transitions of A with $V_B = V_{DD}$ are dominated by the FG switching behavior of the RFET. Further, there are transitions where the switching comes from flipping the polarity of strong pulling devices, such as the transition of A, if $B = 0$. Here, the full swing output is provided by the planar RFET in particular, and for a sufficiently high I_{on}/I_{off} ratio in general. The principle of integrating the functionality of multiple MOSFETs into one extends to more than two inputs: Trommer et al. [55] show an RFET with 4 independent gates, which provides wired-*AND* functionality.

While the higher expressiveness of the single device provides further benefits on circuit level, e. g. for compact multiplexing [56], implementation of area-efficient neuromorphic circuits [57], hardware security through key based authentication [58] and tuneable differential stages [59], at the point of physical implementation, this can come at a cost. The availability of additional PGs aggravates routing congestion and calls for novel place and route strategies [49], [60]–[62].

The scope of this work is restricted to basic digital circuits, such as the shown *INV*, *NAND*, *NOR* and *XOR* cell, which have been demonstrated with the planar RFET in [45], in order to evaluate the proposed device modeling approaches on fundamental static and reconfigurable logic gates.

2.2 Modeling Semiconductor Devices

On a low technology readiness level (TRL), the evaluation of fabricated emerging devices typically focuses on device level characteristics, such as leakage, drive current and sub-threshold slope. Early prototype devices are typically fabricated in arrays of solitary devices, which can be subject to substantial process variation. On the one hand, evaluation in form of comparison against devices from established technology nodes is fair, as the plain characteristic values allow to grasp the achieved performance of the emerging device. On the other hand, semiconductor devices are typically parts of integrated circuits, rendering circuit level evaluation a convincing argument for acceptance of emerging device concepts. And then there are devices with novel features, such as RFETs, which are unlikely to compete with established and highly scaled technology nodes on device level, while their benefits play out on circuit level [13]. One way to demonstrate these benefits is the actual fabrication and measurement of circuits. But fabrication of emerging technologies typically means going off the beaten track and the resulting DUTs often suffer from issues such as low yield, high process variation and degradation during measurement. For compact circuits with low transistor count, such as logic cells, this can be a feasible approach [29], [54]. Simulation is, in turn, often more cost-efficient and leads to reproducible results with fully controllable device variation without degradation during characterization. As this work contributes to predictive modeling and simulation of emerging semiconductor devices, it is important to inspect existing tools and work out capabilities and limitations.

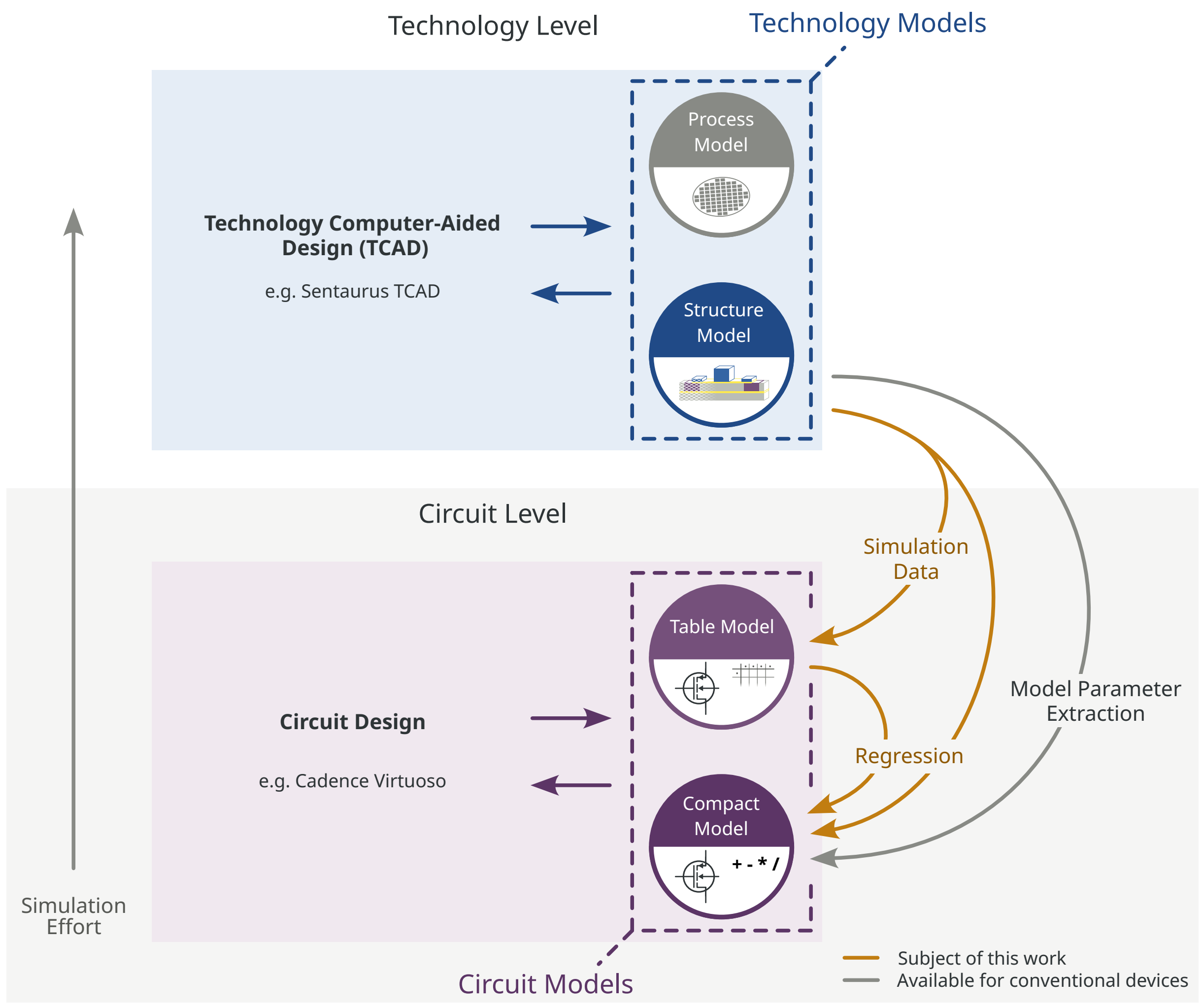


Figure 2.8: Conventional semiconductor devices can be modeled using industry standard compact models. A typical approach for emerging devices is to build a table model. Further, this work explores machine learning-based regression methods.

Two abstraction levels of simulation models are relevant for this work and therefore shown in Figure 2.8, namely technology models and circuit models. In circuit simulation physical details of conduction mechanisms become less relevant and efficiency gains importance. In consequence, models with less computational effort, e. g. through abstraction, have to be constructed. The step from technology simulation to circuit simulation can be overcome in various ways. Figure 2.8 shows the three commonly taken paths. On the technology level, TCAD suites provide tools for model parameter extraction, such as TCAD2SPICE in Synopsys TCAD. If structure and working principles of the DUT are represented by an existing and supported compact model, the path from technology model to a parameter set for an already established compact model is quick and typically automatic. In this work, conventional parameter extraction methods are not possible, as there are no available compact models for the planar RFET device concept. Because of their importance in the state of the art, compact models are introduced in Section 2.2.2. The second approach, the table model, is a data driven approach, where the characteristic data from TCAD simulation are provided to the circuit simulator and interpolated at runtime. Performance and accuracy are to be balanced, as higher sample count leads to higher sample density, which in turn negatively affects memory requirement and execution time of the interpolation algorithm. Table models are covered in Section 2.2.3. A third way is using the characteristic data, e. g. the table model, to perform regression and obtain analytic expressions which then form a compact model. Previous work in the area of regression-based modeling, to which this work contributes, is summarized in Section 2.2.2.

The particular TCAD suite used for this work is Sentaurus TCAD by Synopsys and the program structure on which the methods in Chapter 3 are build upon are introduced in Section 2.2.1. Circuit level simulation is conducted in Cadence Custom IC Design Environment (2022) [63], using the Simulation Program with Integrated Circuit Emphasis (SPICE) simulator SPECTRE.

2.2.1 Technology Models and TCAD Simulation

TCAD refers to a suite of tools for design technology co-optimization (DTCO), which target predictive technology- and process modeling. Technology modeling resorts to physical models implemented by the TCAD tool, to be solved numerically in discretized parts of a structural semiconductor model. The computational effort of technology simulation depends on the granularity of the computational mesh and is typically substantial, compared to circuit level device models. In return, technology simulation provides insights into the internal physical parameters of the model beyond measurement capabilities. Another strong argument for the use of technology simulation is the arbitrary variation of boundary conditions and material parameters for prediction and optimization. Today, the employed physical models can be considered reliable for extensively studied semiconductor materials such as silicon, decreasing the number of exploratory fabrication runs required [64].

In this work, Sentaurus TCAD is used for generating device characteristic data, which are then either used as a table model or transformed into a compact model. The main use of TCAD for this work lies in generating characteristic device data for the data driven modeling approaches. This section provides an overview of the features relevant to efficiently generate these data samples. Further, details about the Sentaurus TCAD device simulator implementation with respect to the efficient generation of data are provided.

This work builds upon a structural technology model developed in [27] using Sentaurus Structure Editor and focuses on electrical device simulation using Sentaurus Device. The topic of process simulation is out of scope of this work. A typical simulation flow of a structural technology model in Sentaurus TCAD is shown in Figure 2.9. The first step is to create a structural technology model, which defines device geometry along with the respective materials, doping profiles and other spatial properties in either 2 or 3 dimensions. After definition of structural parameters, a mesh is placed onto the structure and defines the discretized volume in which the semiconductor equations are to be solved during electrical simulation using finite volume method (FVM). The interface to electrical simulation is provided in form of equipotential electrode definitions, which form part of the boundary conditions of the simulation. For electrical device simulation it is important to recognize that convergence and computational effort depends, aside from the numeric setup of the device simulator, heavily on the meshing of the structure model.

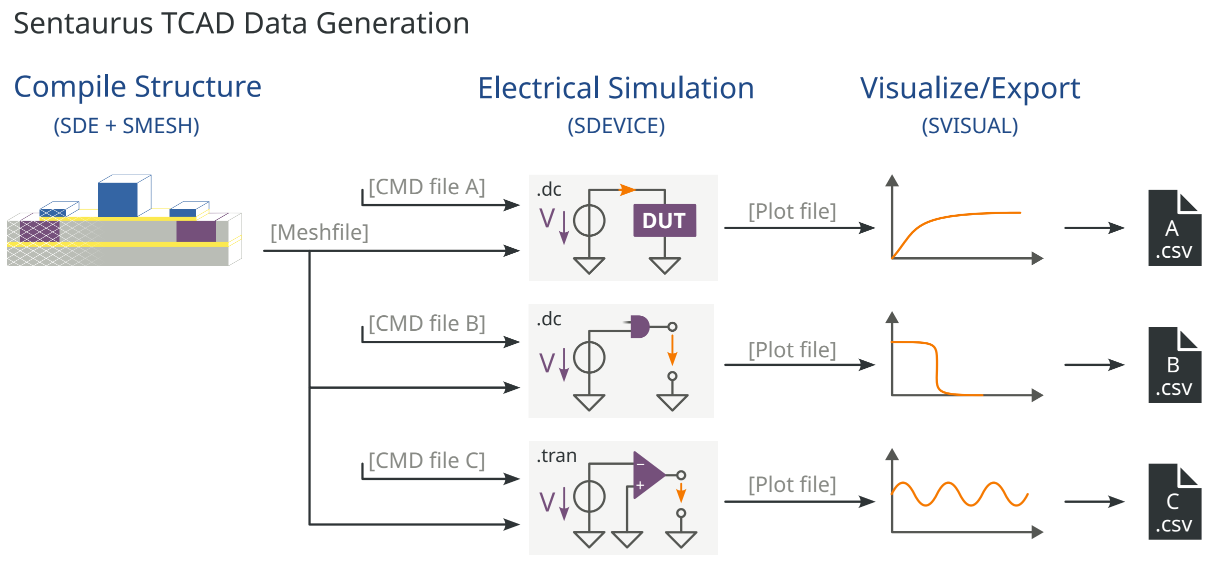


Figure 2.9: The tool-flow to generate device characteristic data (without process simulation) typically involves a structure editor along with a meshing tool, the actual electrical simulation, e. g. DC or transient simulation, and the export or visualization. For this purpose, Sentaurus TCAD provides Sentaurus Structure Editor, Sentaurus Mesh, Sentaurus Device and Sentaurus Visual.

Electrical device simulation solves relevant physical equations, such as the drift-diffusion equation, to obtain characteristic device data in the constrained bias points. As shown in Figure 2.9, device simulation further enables predictive circuit simulation, which allows evaluation of DUTs in digital cells or small analog circuits. SPICE-like simulation types, such as quasistationary and transient simulations are possible; SPICE models can be included. Due to the high simulation effort of the physical models in technology simulation, however, the available computational resources typically restrict the netlist to a transistor count short of 10 technology model instances. Apart from the mesh files of all technology models to be instantiated, a user defined command file describes the particular device simulation setup including netlist, sources, DUTs and solver parameters. The simulation result, which in this work consists only of the electrical traces with respect to time or sweep variable, is stored in the plot file, a structured text file. Synopsys further, provides the tool Sentaurus Visual for visualization and script based export of simulation results, which is not used in this work.

Simulation on Technology Level

When probing and measuring a DUT in hardware, the structure of the resulting data set, i. e. respective independent variables, their range and granularity, can be selected freely within the electrical boundaries of the DUT and measurement equipment. In technology simulation, the availability of a data sample $S(V_e)$, where S is an observed characteristic such as drive current I_{DC} or electrode charge $Q_{e,i}$ of electrode $i \in \mathbb{N}$, at a boundary condition defined by a tuple of electrode voltages $V_e = (V_{e,1}, V_{e,2}, ..., V_{e,N})$ depends mainly on convergence of the semiconductor equations at the respective simulator step. This section describes briefly the working principle of quasistationary and transient simulation in Sentaurus TCAD, with an emphasis on how solutions for the constrained simulator steps are obtained.

The first simulation type relevant for this work is quasistationary simulation [65]. The physical perspective on quasistationary electromagnetic fields is set by the assumption that field propagation is much faster than their respective change in time. All structures are small compared to the wavelength of the field change in time [66]. In approximation, all electromagnetic disturbances therefore propagate through the system with infinite velocity [67]. As a consequence, the solution of a quasistationary simulation step is calculated based on the momentary boundary conditions alone, and without dependence on time or time-derivatives. During a quasistationary simulation, boundary conditions of the device or circuit are ramped to a specified goal, e. g. until a specified sweep electrode voltage $V_{e,sweep}$ reaches $V_{e,sweep} = V_{goal}$. The partial differential equation systems of the respective electrical properties, such as $I, Q, ...$, are solved for every step.

The solution of each simulation step is found through linearization of coupled semiconductor equations using a Newton-like solver [65]. Solutions of the linearized partial differential equations are found using a direct linear solver. For electrical simulations of the planar RFET PARallel Direct SOlver (PARDISO) ([68], [69]) has shown the best convergence of all available solvers when used together with the provided planar RFET technology model and is therefore used exclusively throughout this work. The respective semiconductor equations are not subject of the proposed data driven modeling approach and the reader is referred to Bank et al. [70] for detailed descriptions on the matter. The Newton iterations are configured mainly by specifying the maximum allowed iterations, damping of iterations and error figures as criteria for accepting a solution. For a robust configuration of the linearization method, the user has to decide between the competing factors convergence, precision and simulation time.

Adaptive stepping leads to a higher sample density in operating regions with low convergence. Unlike in circuit simulation, technology simulation often faces convergence difficulties, preventing the solution to converge at arbitrary boundary conditions. It is crucial to start simulation at an initial condition with good convergence, usually with all electrode voltages $V_{e,i} = 0\,V$, and adapt step size towards the designated goal. This means, that device simulations typically consist of an initial sweep which ramps up all initial electrode voltages, such as supply voltages and set input voltages required at $t = 0\,s$, after which the actual simulation of interest can be performed. The allowed step sizes throughout the quasistationary ramp are constrained by specifying a minimum and maximum. Solutions that are accepted result in an increase in step size by a specified increment factor until the maximum allowed time step is reached. If a step fails to converge with the maximum allowed number of Newton-iterations, the current step size is decreased by a decrement factor. At the point where the minimum allowed step constraint is met without an accepted solution, the quasistationary simulation fails.

Figure 2.10 shows a typical quasistationary simulation, with an initial step size of $3.2\,mV$. The step size is observed to increase until $V_{sweep} \approx -0.5\,V$, where steps fail to converge and therefore step size decreases to a low level. The operating region of $-0.5\,V < V_{sweep} < 0.5\,V$ shows convergence difficulties and therefore step size remains low. The increment factor leads

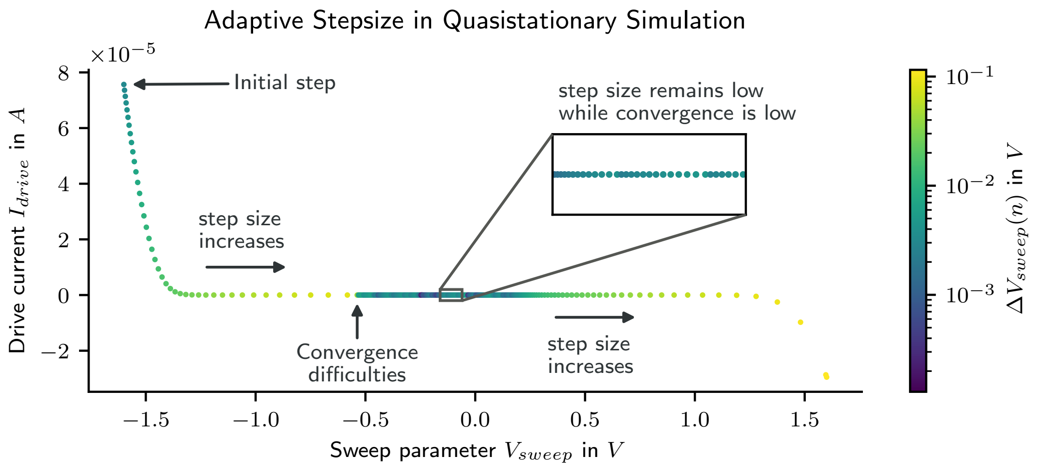


Figure 2.10: The size of simulation steps depends on convergence. Non-uniform sampling is the result.

then to an increase in step size at $V_{sweep} \approx 0.5\,V$ until simulation completes at the designated goal of $V_{sweep} = 1.6\,V$.

The second important simulation type in this work is transient simulation. Similar to the quasistationary parameter sweep with adaptive stepping, the transient solver operates on discrete points in time. Transient simulation in Sentaurus Device solves the set of ordinary differential equations

$$\frac{d}{dt}q(z(t)) + f(t, z(t)) = 0, \tag{2.1}$$

with the transient solution $q(z(t))$, the respective change over time $z(t)$, and the external excitation vector $f(t, z(t))$ [70]. Discretization of time steps is performed using trapezoid rule and the backward differentiation formula, which adapts the step size to the estimated transient error.

Sentaurus Device further features a plot function *CurrentPlot*, which saves solutions at defined simulation steps to a plot file [65]. While the adaptive stepping aims at good convergence with low simulation time, the plot function allows the user to shape the sample density by specifying the sampling constraints. A general constraint is the range of acquisition, which results in all solutions within the specified range to be extracted. The range can be combined with the specification of uniform or minimum intervals. Further, explicit steps or conditions are supported. The additional steps to be exported do not alter the state of the adaptive stepping algorithm. Plot constraints can be omitted, causing the simulator to save every complete simulation step, which results from the adaptive stepping.

Apart from the substantial computational effort that is required to conduct technology simulation, there are two main issues to occur. When convergence cannot be achieved within the solver and stepping constraints, the simulation fails. Typically, user interaction is required, as refinement of numeric parameters, such as the number of Newton iterations or the increment factor, often enable convergence at cost of increased resource requirements. Aside from convergence issues simulations can stall and cease to progress further. In this condition neither log file nor output file(s) advance, although the simulation process does not return. For the technology model used in this work, a simulation is assumed to be in a stalled state, e. g. in a deadlock, if no progress is written to the log file for more than 40 minutes of time (wall clock). Both described fail states can be handled manually if the number of conducted technology simulations is manageable. In a large simulation deck, e. g. to build a table model, the user benefits from systematic handling of all issues that impede the completion of simulations, as soon as all convergence aids are exhausted.

Design of TCAD Experiments

Design of Experiments (DoE) describes methods to obtain the maximum information about an observable process with the minimum amount of resources [71]. Typically, DoE aims at extracting information about a process with the intention of screening, optimizing or characterizing the output (*response*), with respect to inputs (*factors*) [72]. In general, the computational cost of conducting experiments in technology simulation is high, and so is the reward for reducing the number of experiments required. However, the goal of the experiments in this work has to be kept in mind: On the one hand, technology simulation can serve the purpose of finding specific properties, such as a maximum drive current or the onset of inversion, where strategic DoE can reduce the required simulation deck. On the other hand, when creating a data driven circuit-level model, such as a table model, the focus shifts from the response variable back to the expected factor values during runtime of the model. DoE only helps with the former.

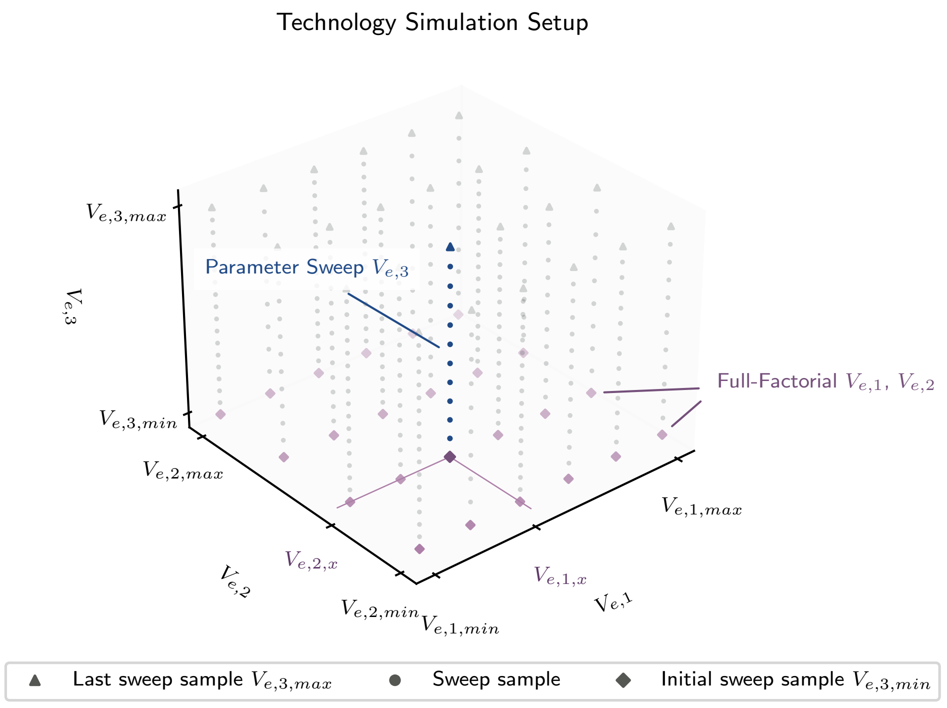

Technology Simulation Setup
$V_{e,3,max}$
$V_{e,3}$
$V_{e,3,min}$
Parameter Sweep $V_{e,3}$
Full-Factorial $V_{e,1}$, $V_{e,2}$
$V_{e,2,max}$
$V_{e,2,x}$
$V_{e,2}$
$V_{e,2,min}$
$V_{e,1,min}$
$V_{e,1,x}$
$V_{e,1}$
$V_{e,1,max}$
▲ Last sweep sample $V_{e,3,max}$
● Sweep sample
◆ Initial sweep sample $V_{e,3,min}$

Figure 2.11: The sample space $\mathbb{R}^3$ comprises $V_{e,1}$, $V_{e,2}$ and $V_{e,3}$. Characteristic samples can be generated by a full factorial setup with respect to $V_{e,1}$ and $V_{e,2}$ on which the $V_{e,3}$-sweeps are executed.

Instead, the key question for data driven modeling is about the range of the input features and the required granularity to obtain a sufficiently accurate model. The lookup table structure for table model simulation, explained in Section 2.2.3, almost dictates the method of conducting the experiments, as it is tailored to parametric sweeps. The parametric sweep is an implementation of the one-factor-at-a-time (OFAT) method of experiments, in which all but one parameter are fixed, and the remaining parameter is changed over a specified range [73]. Figure 2.11 shows the simulation setup, where the samples are determined by a tuples of three electrode voltages $V_e = (V_{e,1}, V_{e,2}, V_{e,3})$. The setup is full factorial with respect to $V_{e,1}, V_{e,2}$, excluding the sweep variable. The sets of voltage steps in each dimension have a cardinality of $|S_{e,1}| = 6$ and $|S_{e,2}| = 4$, resulting in $|S_{e,1}| \cdot |S_{e,2}| = 24$ parameter sweeps of V_C to cover the remaining dimension in $\mathbb{R}^3$.

The example of Figure 2.11 can be generalized to $N_e \in \mathbb{N}^*$ electrodes. After assessment of the model use case, i. e. the range of input voltages and their required accuracy, the granularity and range of every $S_{e,i}$ in each dimension $i = 1, ..., N_e - 1$ are selected. The TCAD parameter sweeps of V_{N_e} are configured according to the principles explained in Section 2.2.1. The factorial setup of the sets $S_{e,1}$,...,S_{N_e-1} then generates a set of tuples, which represents the simulation deck

$$S_{deck} = \prod_{i=1}^{N_e-1} S_{e,i} = S_{e,1} \times S_{e,2} \times ... \times S_{N_e-1}, \tag{2.2}$$

and accordingly, the amount of simulations to be conducted follows

$$|S_{deck}| = \prod_{i=1}^{N_e-1} |S_{e,i}| = |S_{e,1}| \cdot |S_{e,2}| \cdot ... \cdot |S_{N_e-1}| \,. \tag{2.3}$$

2.2.2 Compact Models

This section provides an overview of established and commonly used compact models implementations and their basic functionalities. At first, prominent bulk CMOS compact models are presented to provide general insight into compact modeling techniques. The fundamentals of a model for multi gate structures with high similarity to the planar RFET, the Berkeley Short-Channel IGFET Model - Independent Multi-Gate (BSIM-IMG), is then analyzed. At last, this section elaborates on how existing models can be applied to the planar RFET. Detailed physical equations are reduced to a minimum, as they do not contribute to understanding the data driven predictive modeling approaches proposed in this work.

Compact models bridge the gap between technology and circuit design [74]. While technology simulation typically serves the purpose of understanding internal physical behavior, the compact model aims for an efficient and more abstract implementation, in order to conduct fast and reliable circuit simulation. And having suitable compact models available for circuit simulation was initially one of the main success factors of SPICE simulation, after all [75]. Compact models abstract from technology level and implement the device behavior regarding current - voltage, *I-V*, and capacitance - voltage, *C-V*, in form of closed-form analytic equations [76]. Both abstraction levels require accurate models, but while technology simulation aims at implementing true physical behavior, compact models target computational performance. It is therefore common to introduce mathematical fitting parameters, along with the comprehensive physical parameters, which are extracted either from technology simulation or measurement [77], [78]. Typically, a compact model is composed of two parts: The *core* represents the ideal physical understanding of the device. In order to account for parasitic effects, short channel effects, process variability and other realistic influences, a second layer, the *real* model, adapts the compact model to provide accurate prediction [74], [78].

Modeling in Verilog-A

When implementations of algorithms have to meet strict performance goals, programmers typically resort to the C language. In the history of compact modeling this is no different [79]. However, the trend shifted towards Verilog-A when compact modeling support was greatly extended with version 2.2 in 2004 [80]. Following recommendations of Lemaitre et al. [81], the access to derivatives and simulator variables, among other improvements, facilitates development and debugging of models. The runtimes of Verilog-A models are, according to [80] and in the worst case, at a factor of 2 with respect to C. Zhou et al. [82] even demonstrate a Verilog-A model that even exceeds the C implementation in performance with a benchmark circuit.

As Verilog-A models are evaluated in circuit simulation, it is important to write models which are computationally efficient and avoid constructs which lead to convergence issues. Simulators are typically SPICE-like and solve a system of non-linear differential equations, which represents the circuit [83]. A way to reduce variables during modified nodal analysis ([84]) is to avoid voltage sources and to express device behavior as voltage controlled current elements $I(V)$. For transient simulation, time derivative of voltage controlled charge elements $\frac{dQ(V)}{dt}$ provide good convergence [80]. Especially for capacitive currents, in fundamental literature for constant capacitance often expressed as $I_c = C \cdot \frac{dV}{dt}$, it is best practice to construct a charge-based expression, such as `q = f(V(b_cap)); I(b_cap)<+ ddt(q);` for node `b_cap` [80]. In general, continuity and differentiability, play an important role for a robust model [85]. Ideally the model equations are continuous, not section wise defined, and their derivatives are C^∞ continuous [80]. In practice, continuity of the derivatives until 3rd degree (for harmonic distortion simulation at least 5th degree) is sufficient for circuit simulation [85].

Threshold Voltage-, Charge- and Surface Potential-Based Compact Models

There are mainly three conventional types of compact model implementations, namely threshold voltage-based, surface potential-based and charge-based [86]. Threshold voltage-based compact models divide the operating region into segments, depending on the strength of inversion and the conduction properties of the channel, accordingly. A prominent representative of threshold voltage-based compact models is the Berkeley Short-Channel IGFET Model (BSIM) [87]. The first version of the BSIM compact model from 1987 provides drain current equation for cut-off, linear and saturation region, similar to the commonly used textbook MOSFET model by Shichman and Hodges [88]. Mathematical simplifications, such as the numeric approximation of the effect of barrier lowering on V_{th} caused by substrate bias, allowed fast computation during circuit simulation [87]. With version 3v3 [89], the BSIM group created the first model to be standardized by the Compact Model Council (CMC), today called Compact Model Coalition, in 1996 [77]. Aside from accuracy and computational performance, a significant contributor to the success of BSIM 3v3 is the continuous modeling of the I-V characteristic in form of a single equation for all operating regions, which boosts convergence. BSIM 4 has been presented shortly after and was widely adopted by semiconductor manufacturers and designers, as it allows parameterization for a wide range of technologies from $0.35\,\mu m$ to $28\,nm$ [77].

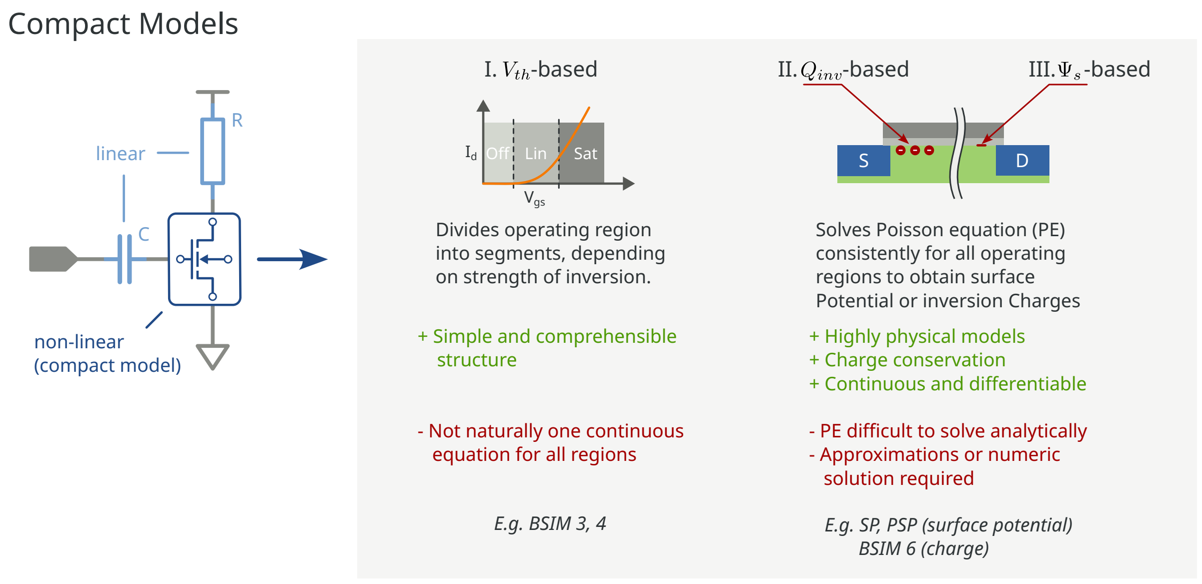

Compact Models
linear
R
C
non-linear (compact model)
I. V_{th}-based
I_d
Off
Lin
Sat
V_{gs}
Divides operating region into segments, depending on strength of inversion.
+ Simple and comprehensible structure
- Not naturally one continuous equation for all regions
E.g. BSIM 3, 4
II. Q_{inv}-based
III. Ψ_s-based
S
D
Solves Poisson equation (PE) consistently for all operating regions to obtain surface Potential or inversion Charges
+ Highly physical models
+ Charge conservation
+ Continuous and differentiable
- PE difficult to solve analytically
- Approximations or numeric solution required
E.g. SP, PSP (surface potential) BSIM 6 (charge)

Figure 2.12: The three typical compact model approaches are either threshold voltage V_{th}-, inversion charge Q_{inv}-, or surface potential Ψ_s-based.

Charge-based approaches express the inversion charge densities and the electrode charges in dependence of the input voltages, inherently providing charge conservation [90]. The inversion charge formulation allows highly physical modeling of the sub-threshold region in weak inversion, which is an inherent weakness of threshold voltage-based modeling [91]. Today widely used for bulk MOSFETs is BSIM version 6 [74], which relies in its core on a charge-based modeling approach to solve the Poisson equation, similar to surface potential-based models. Analog and radio frequency (RF) designs often require the drain current and its derivative to be symmetric to $V_{DS} = 0\,V$, which is a common point of criticism in BSIM 4 [92]. The charge based approach of BSIM 6 solves this issue and, within the allowed parameterization, derivatives of at least up to the order of 3 are continuous. The real model, on top of the core model, is mostly adopted from BSIM 4.

The third type, surface potential-based compact models, derive the I-V and C-V characteristics from the surface potential of the channel at the oxide interface, which is constrained by the given input voltages at both ends of the channel [86]. A physical model of the surface potential Ψ_s related to substrate [93]

$$(V_{GB} - V_{fb} - \Psi_s)^2 = \gamma^2 \big(\Psi_s - V_t + V_t \cdot e^{\frac{\Psi - 2\Phi_b - \Phi_n}{V_t}}\big), \tag{2.4}$$

with gate-bulk voltage V_{GB}, flat band voltage V_{fb}, body effect factor γ, thermal voltage V_t, bulk potential Φ_b and source/drain-bulk potential Φ_n, was proposed by and named after Pao and Sah [94] in 1966. A drawback of Equation 2.4 is that it is an implicit function. Solutions for

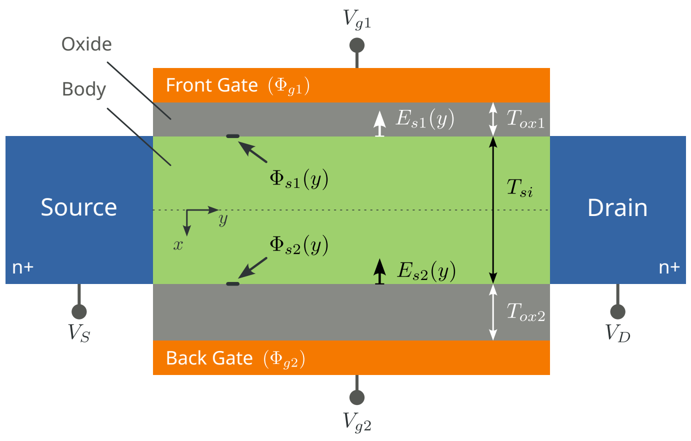


Figure 2.13: BSIM-IMG targets FD-SOI devices with two independent gates on both sides of the body [104].

the Pao-Sah model are either based on iterative methods or analytic approximation, with one of the latter proposed by Gildenblat et al. [95]. For early surface potential-based models the computational effort was therefore substantially higher than contemporary threshold voltage based compact models [90]. Fast converging and highly accurate iterative solutions, such as [96] and [97] contributed to the success of metal-oxide-semiconductor (MOS) Model 11 [97] and Hiroshima-University STARC IGFET Model (HiSIM) [98], [99]. An example of analytic approximation of Equation 2.4 is implemented in the SP compact model [95] and reduces the number of complex operations required to one logarithm, one exponent and 3 square-roots [93]. PSP [91] emerges in 2006 as a fusion between SP and MOS Model 11 with the analytic approximation of the surface potential from [95], and is the most commonly used surface potential-based compact model for bulk MOSFETs.

BSIM-IMG: A Compact Model for Multiple Gates on FD-SOI

Semiconductor devices evolve along with materials, structures and features, and so do, necessarily, compact models. While change in the semiconductor industry is often incremental, disruptive technologies, such as FinFETs and FD-SOI emerge at times and require accurate compact modeling. This section deals with existing compact models in the vicinity of structure, materials and working principles of the planar RFET. Multi-gate devices are divided into two groups, depending on whether the respective gates are electrically connected, such as FinFETs, or independent, such as FD-SOI with front- and back gate. A cross section of a dual gate FD-SOI FET is shown in Figure 2.13. The most widely supported and CMC standardized compact models for independent gate devices are BSIM-IMG [100], L-UTSOI [101] and HiSIM-SOTB [102]. All three models are based on surface potential and target ultrathin body devices. The solutions are found through an iterative method in HiSIM-SOTB, while both former compact models provide analytical expressions. To provide an example of the surface potential calculation of a multiple independent gate FET on FD-SOI, such as the planar RFET, the core of BSIM-IMG is presented in the following. The model outline is mainly based on the technical manual of BSIM-IMG [103] and the work of Khandelwal et al. [100].

In BSIM-IMG the multiple gates are allowed to have different work functions, dielectric thickness and dielectric constants. The typical target for BSIM-IMG is a planar FD-SOI device, where the two independent gates are located at the opposite side of the fully depleted channel, as shown in Figure 2.13 and described in [104]. The two gates, FG and BG are denoted 1 (FG) and 2 (BG). The core model approach is surface potential-based, and solves the one-dimensional poisson equation of the channel in x-direction in form of

$$\varepsilon_{si}\frac{d^2\Psi(x,y)}{dx^2} = qN_c \cdot e^{\frac{(\Psi(x,y)-V_{ch}(y))}{V_t}}, \tag{2.5}$$

with the dielectric constant of silicon ε_{si}, elementary charge q, thermal voltage V_t, density of states in the conduction band in silicon N_c, and channel voltage, i. e. the electron quasi-Fermi potential relative to source potential, $V_{ch}(y)$ [104]. In general and unlike in [93], surface potentials (Ψ) computed by the BSIM-IMG model are all source-relative, as there is no bulk potential available to reference to. The body is considered lightly-doped, therefore ionized dopants are not considered for the charge density calculation within the channel. The total charge density of the inversion carriers in Equation 2.5 is therefore represented using Boltzmann's distribution. After integration, the surface potentials Ψ_{s1} and Ψ_{s2} are formulated in

$$E_{s1}^2 - E_{s2}^2 = \frac{2qN_cV_t}{\varepsilon_{si}}\left(e^{\frac{\Psi_{s1}-V_{ch}}{V_t}} - e^{\frac{\Psi_{s2}-V_{ch}}{Vt}}\right), \tag{2.6}$$

where $E_{s\{1,2\}}$ is the electric field at the respective surface. Inversion charge at the interface of the back-oxide is explicitly regarded in BSIM-IMG [100]. In addition to Equation 2.6, two boundary conditions at the surfaces $s1$ and $s2$, which stem from continuity of displacement fields,

$$C_{ox\{1,2\}}(V_{g\{1,2\}} - \Delta\Phi_{\{1,2\}} - \Psi_{s\{1,2\}}) = \varepsilon_{si}E_{s\{1,2\}} \tag{2.7}$$

$$\text{with } C_{\{si,ox1,ox2\}} = \frac{T_{\{si,ox1,ox2\}}}{\varepsilon_{\{si,ox1,ox2\}}} \tag{2.8}$$

are formulated. As a second independent equation of the two unknown Ψ_{s1} and Ψ_{s2} is obtained under two assumptions [105]: Firstly, the charge distribution towards the FG is represented by charge sheet approximation. And as the BG is only in weak inversion, secondly, the inversion charge distribution is not considered to be volumetric. Subsequently, assuming a constant displacement field in x-direction between the two gates allows the formulation of the second independent equation of the surface potentials [106]

$$\Psi_{s2} = \frac{C_{si}}{C_{si}+C_{ox2}} \cdot \Psi_{s1} + \frac{C_{ox2}}{C_{si}+C_{ox2}}(V_{g2} - \Delta\Phi_2). \tag{2.9}$$

The solution to the implicit relation of Equation 2.6 with Equation 2.9 is then calculated by using the approximate solution proposed by Lu et al. [104] for E_{s2}, and serves as initial guess for Ψ_{s1}, i. e. Ψ'_{s1}. However, the assumption of constant displacement field in the vertical direction of the device is valid for weak inversion with low carrier density within the body [106]. A final step, generalizes the surface potential for moderate and strong inversion by using the initial guess Ψ'_{s1} for Ψ_{s1} in Equation 2.9, and provides solutions for both surface potentials Ψ_{s1} and Ψ_{s2}.

Compact Models for RFETs

Although not yet standardized by CMC, there are compact models which target reconfigurable FETs. This section provides a brief overview of existing and relevant compact models.

For the early carbon nanotube RFETs with heavily doped source and drain regions O'Connor et al. [50] and Goguet et al. [107] propose a purely physical compact model. The polarity (n/p) is modeled as a binary parameter and the simulation of digital for statically configured devices is demonstrated.

Martinie et al. [101] provide a simplistic and surface potential-based compact model for a device with similar structure as the planar RFET. The main difference is that the PGs are not aligned with the Schottky barriers at source and drain. The proposed compact model in [101] does, however, not allow dynamic reconfiguration via an externalized PG node.

Another and equally surface potential-based approach is proposed by Roemer et al. [108], [109], who provide the PG voltage V_{pg} of a dual gate RFET, attributed to type b) in Figure 2.2, as a direct input. [109] identifies the main on-current contribution of the type b) device to stem from field emission over the source-side Schottky barrier and models the total current as a superposition of electron current and hole current. Dynamic reconfiguration via V_{pg} is, however, not demonstrated, and the model is not publicly available. Ni et al. [110] propose a compact model for a structure similar to [109], but do not show dynamic reconfiguration, either.

While the existing compact models in this section fall short of demonstrating the integration of multiple independent gate electrodes for dynamic reconfiguration into a closed form expression, especially [101], [109] and [110] can be a promising starting point for physical compact modeling of the planar RFET.

Machine Learning-Based Compact Modeling Approaches

The increasing adoption of machine learning methods in engineering disciplines does not spare the domain of compact modeling. Especially neural networks, introduced in detail in Section 2.3.1, are chosen by various groups as model architecture. This section provides an overview of machine learning-based approaches for compact modeling.

In 1992 Litovski et al. [111] already exploit the continuous derivative provided by an artificial neural network (ANN) to replicate the characteristics of an existing compact model. The network contains a single hidden layer, which depends in size on an initial estimation of device complexity. This approach represents a baseline for machine learning-based modeling, which has been refined by various groups mentioned in this section.

Hutchins et al. [112] describe a modeling framework for multi state devices, e. g. a hafnium oxide memristor, using a neural network with 3 hidden layers. The inputs of the models are electrode voltages, a one-hot encoded device state and device specific parameters, such as size and temperature. Training data are generated from an available compact model and the neural network is implemented in Verilog-A. In order to improve robustness of the model, Gaussian noise augmentation, further explained in Section 4.2.3, is applied.

With special attention on drain-source symmetry, Xu et al. [113] propose a neural network based modeling flow for MOSFETs in high frequency applications. The symmetry constraints allow the applications in circuits where $V_{ds} = 0\,V$ is crossed, such as mixer circuits. Training data for the neural network are only required for one of the symmetric half-spaces around $V_{ds} = 0\,V$. The trained model is then duplicated and the identiacal copy of the original neural network is operated with inverted input voltages to predict the current and charges of the respective other half-space.

Wang et al. [114] and Zhao et al. [115] propose a single neural network, which generates outputs for current and charges. The output values of the neural network are transformed using dedicated functions, such as

$$I_{ds} = I_0 \cdot V_{ds}/1\,V \cdot 10^{y} \tag{2.10}$$

for the drive current, with a linear scaling factor I_0 in A and neural network output y [114]. For sake of dimensional consistency, the division by $1\,V$ is added by the author of the present work. A valuable finding in [115] is that out of three investigated activation functions, which are logistic function, rectified linear unit (ReLU) and tangens hyperbolicus (tanh), for neurons of the hidden layer, tanh performs best in terms of mean squared error (MSE). The use of tanh results in an MSE of 0.01, while ReLU and the logistic function score an MSE of 0.07 and 0.23. The modeling approach in [114] targets DTCO and is therefore embedded into an optimization flow for model accuracy, modeling turnaround time and SPICE simulation effort.

A similar approach is taken by Klemme et al. [116], who propose a neural network-based approach to model a FinFET. The drive current model consists of two neural networks: A model trained from logarithmically transformed input data represents currents below the threshold voltage V_{th}, similar to [114], while a model trained on the raw input data represents all currents above V_{th}. At $V_{gs} = V_{th}$, the model is interpolated within a fixed width of $V_{th} \pm 0.1\,V$ to mitigate the discontinuity between both segments.

Habal et al. [117] propose a bulk MOSFET model in form of a neural network. Features are constructed from expanding the polynomials of the Shichman-Hodges model [88], which results in terms such as V_{ds}^2, $V_{bs}^2 V_{ds}$, $V_{ds} V_{gs}/L$ and others. Two hidden layers with 52 and 26 neurons, respectively, provide sufficient accuracy for predictive circuit simulation. A requirement for this modeling approach is to have physical insights into charge transport mechanisms in order to construct suitable features.

A physics inspired neural network is proposed by Li et al. [118], who stray away from the concept of a homogeneous structure of the hidden neurons. Instead, the model is partitioned into subnets with different activation functions and available parameters. The relationship between drain current I_d and channel voltage V_{ds} comes from a subnet with tanh activation function, inspired by the expected physical context. Further, the $I_d(V_{ds} = 0\,V) = 0\,A$ is ensured by removing all bias parameters from the subnet. The influence of the gate voltage V_{TG} on the channel current I_d, however is expected to experience an exponential and a polynomial phase, which is represented by the logistic function and justifies a separate subnet.

2.2.3 Table Models

Table models provide a straight forward data driven modeling approach. Instead of modeling behavior or physical structure of a device, available data points, are organized in form of a table. Essentially, a table model is a data set

$$\mathcal{D}_{table} = \{(X^{(i)}, y^{(i)})\}_{i=1}^{N_s} \tag{2.11}$$

which contains N_d-dimensional tuples $X^{(i)} = (x_1^{(i)}, x_2^{(i)}, ..., x_{N_d}^{(i)}) \in \mathbb{R}^{N_d}$ of individual variables and a dependent value $y^{(i)}$. The total sample count in $\mathcal{D}_{table}$ is then N_s and the number of values to be stored is $N_s \cdot (N_d+1)$. Table 2.1 shows a structured table of $\mathcal{D}_{table}$. During evaluation of a device model, the required data tuple $X^{(i)}$ is looked up in the table and the result $y^{(i)}$ is provided to the simulator. Data tuples that are not explicitly included in the table, but are contained within the convex hull of the set $\{X^{(i)}\}_{i=1}^{N_s}$, can be obtained by interpolation. Commonly used interpolation methods are nearest neighbor, piece-wise linear and piece-wise polynomial. Extrapolation of table models is typically discouraged due to the inherently weak generalization of non-linear systems outside the available data.

Table 2.1: A lookup table assigns an output value y to an N_d-dimensional input tuple.

i	x_1	x_2	x_3	...	x_{N_d}	y
1	$x_1^{(1)}$	$x_2^{(1)}$	$x_3^{(1)}$	...	$x_{N_d}^{(1)}$	$y^{(1)}$
2	$x_1^{(2)}$	$x_2^{(2)}$	$x_3^{(2)}$	...	$x_{N_d}^{(2)}$	$y^{(2)}$
3	$x_1^{(3)}$	$x_2^{(3)}$	$x_3^{(3)}$	...	$x_{N_d}^{(3)}$	$y^{(3)}$
...	...	...	...	...	...	
N_s	$x_1^{(N_s)}$	$x_2^{(N_s)}$	$x_3^{(N_s)}$	...	$x_{N_d}^{(N_s)}$	$y^{(N_s)}$

Table models are used in a multitude of circumstances, where the true system transfer function $f_{system}(X) = y$ is not available or not convenient to compute. It is common to use lookup tables for trigonometric functions in embedded systems in order to avoid costly floating point-based algorithms [119]. Further, lookup tables are a convenient method to model and compensate non-linearity of sensors [120] on microcontrollers. Saripalli et al. represent band-to-band tunneling transistors (TFETs) with table models for simulation of various static random access memory (SRAM) cells [121].

Another use case for lookup tables, and subject of this work, lies in modeling of semiconductor devices for circuit simulation, which is a common approach for emerging devices [122], [123]. Device models which invoke lookup tables, further called table models, are typically provided in form of Verilog-A modules. Verilog-A specifies the function `$table_model` to handle lookup, interpolation and extrapolation of data table regarding a single input tuple $X_{in} = (x_{1,in}, x_{2,in}, ..., x_{N_d,in})$ [124]. The function `$table_model` operates on data that are typically generated from one-dimensional parametric sweeps, a typical method of finite element simulation further elaborated on in Section 2.2.1. A one-dimensional parametric sweep maps an input line $X \in \mathbb{R}^1$ to an output line $Y \in \mathbb{R}^1$. This is a common use case for linearization of non-linear sensor values, as a scalar input from the sensor (x_{sweep}) value is transformed into a scalar output value (y). Adding dependencies, e.g. operating temperature as x_1, the parameter sweep of x_{sweep} requires the setup of a factorial experiment design, explained in Section 2.2.1.

The Verilog-A function `$table_model` provides multiple interpolation methods such as linear interpolation, closest point lookup, quadratic and cubic splines, which can be freely selected for arbitrary dimensions [124]. In continuation of [43], this work restricts itself to linear interpolation, as spline interpolation has shown to often impede convergence in circuit simulation and to introduce oscillations. The general approach to optimize model accuracy and to limit linear interpolation error in this work is to constrain sufficiently fine granularity of the generated data set.

A parametric sweep of $x_{sweep}(= x_{N_d})$ with constant $x_1, ..., x_{N_d-1}$ generates a subset of the table model called iso line. Verilog-A's `$table_model` function requires all data points to be part of iso lines, as a prerequisite for applying a simple multivariate interpolation algorithm [124]. An example of the multivariate linear interpolation with $N_d = 2$ and $X = (x_1, x_2) = (V_{gs}, V_{ds})$ is shown for an NMOS device in Figure 2.14. The iso lines are described by parameter sweep of $x_{N_d} = V_{ds}$ over a constant $x_1 = V_{gs}$. Multivariate interpolation for a bias point $(V_{gs,i}, V_{ds,i})$ starts with finding the adjacent iso lines with respect to V_{gs} (step I) and performing 1-dimensional interpolation first on the respective iso lines (step II and II) at $V_{ds} = V_{ds,i}$. The obtained points on the adjacent iso lines then allow linear interpolation (step IV) of the point $(V_{gs,i}, V_{ds,i})$, which returns a prediction of drain current $I_{d,i}$. Details of the interpolation algorithm are not relevant for this work, as the entire table model functionality is implemented by the simulator and cannot be altered by the user. However, it is important to understand that the requirement of the iso line structure means that available data points cannot be added at random. Either the data to be added are part of an existing iso line or, in case of multiple data points to be added, they form a new iso line of at least two members.

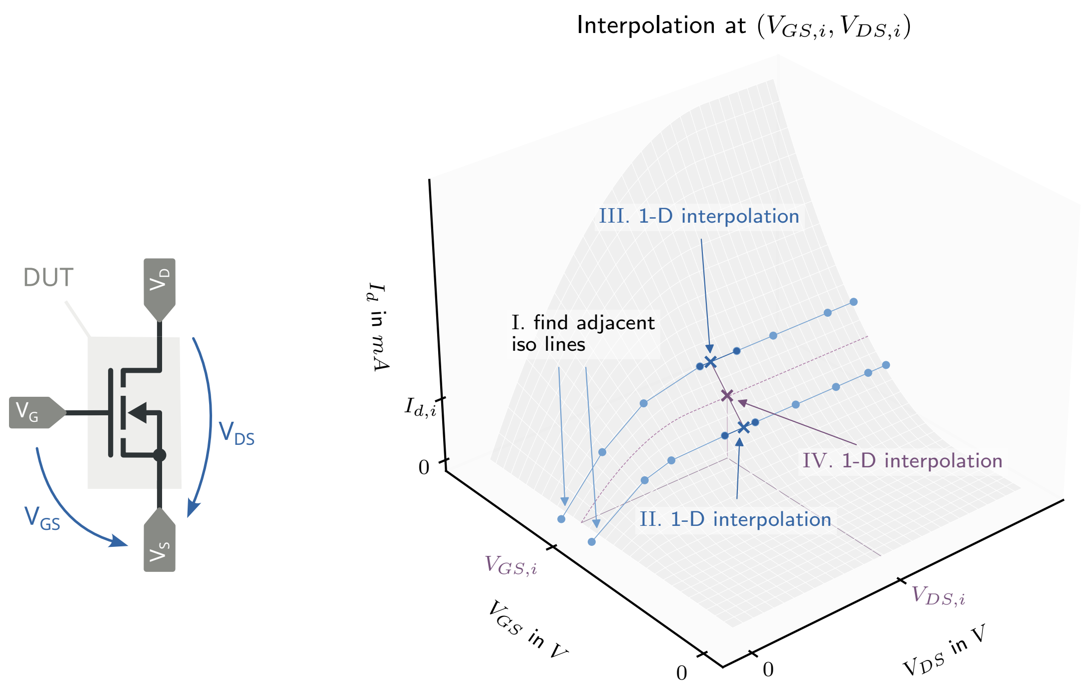


Figure 2.14: The data which are interpolated by Verilog-A's table model function have to feature iso lines for all but the sweep variable, which enables a simple multivariate interpolation mechanism.

There are inherent limitations in what a table model can represent. As tables consist of quasistationary data points and therefore hysteresis and memory effects cannot be represented. Modeling of interface traps or memristive devices cannot be achieved with I-V or Q-V tables, while tables state information and other behavioral data are conceivable. Further, there are properties, such as the inversion capacitance, which cannot be exactly represented by a table model from a DC data set. Figure 2.15 demonstrates in an excerpt of the characteristic of the planar RFET, that inversion capacitances depend heavily on frequency. The capacitance that is computed from the charge table Q-V clearly follows the DC capacitance, while the characteristic diverges for AC simulation in TCAD. The table model then assumes a higher, pessimistic, capacitance.

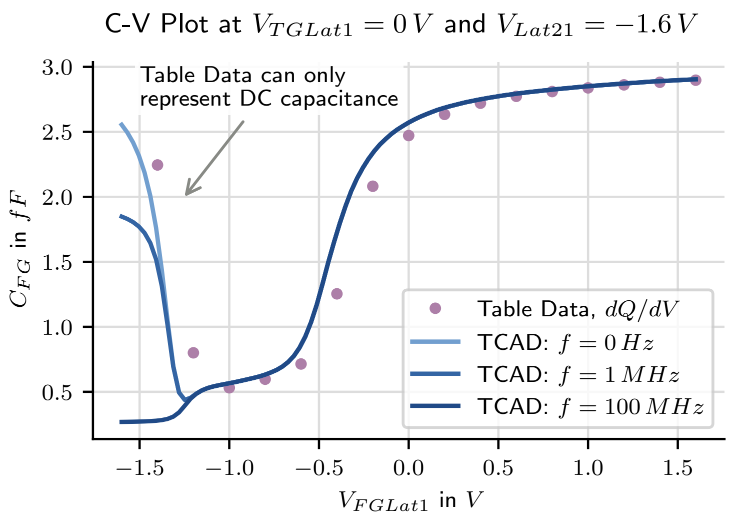


Figure 2.15: Inversion capacitance is frequency dependent. As the table model data set is typically generated from DC data, predictive circuit simulation is prone to errors.

2.3 Fundamentals of Machine Learning Models

The fascination with the working principles of the human brain is ancient. In an attempt to grasp the structure of human information-processing and decision-making, Aristotle describes a physical process to compare and distinguish sensations, already in 300 B.C. [125]. Aristotle's theory of *Associationism* is based on the assumption of discrete psychological processes and the capabilities of the human mind are enabled by associating these processes [126]. More than two millennia later and regardless of the applicability of *Associationism* on the human brain, we see that this concept of densely interconnected computational units is what became an important subset of machine learning methods: *deep learning*.

Regression analysis is a technique in the domain of statistical analysis, which in general aims at finding the influence that one or many independent variables have on others [127]. Today, regression is often associated with machine learning and the optimization of model predictions based on a set of training data, similar to curve fitting. Curve fitting is efficient when the complexity and the general structure of the process to be modeled is known ahead of time. A generic model can be constructed and then be optimized by altering the respective parameters.

However, without insights about the structure of the process available, good parameterization cannot make up for an inaccurate mathematical model structure. With the premise of a black box process, an approach called *symbolic regression* aims at finding symbolic equations for the respective training data.

In general, both methods reside in the field of machine learning, which thrives to find a hypothesis $h(x)$ to approximate the ground-truth of a data set $\mathcal{D}$ [128]. Subject of this section is to explain the fundamental mechanisms and properties of deep learning and symbolic regression that are used for modeling of semiconductor devices.

2.3.1 Deep Learning

Deep learning revolves around interconnected layers of artificial neurons, called artificial neural networks (ANNs). An early neuron model is proposed by McCulloch and Pitts in 1943 [129] and describes neural events by means of propositional logic. To account for the "all-or-none character of nervous activity" [129], the output of each McCulloch-Pitts (MP) neuron is either 1, if the sum of all binary inputs exceeds a threshold, or 0 otherwise. This basic neural model maps the binary input numbers to a binary output value, which already enables the implementation of logic functions, such as binary *AND*, *OR* and inversion. The abstraction from biologically inspired neural models to more numerically expressive mathematic models is shown by Rosenblatt [130] in 1958, who introduces custom weight and bias parameters for a cell that they name *perceptron*, shown in Figure 2.16. The perceptron attaches a weight w_i to each input i and many implementations store an additional constant b as invariant bias [131]. Similar to the MP neuron, the perceptron relies on a binary activation function $g(u)$, but here the *sign* function maps the sum of weighted inputs and bias to the output values $y \in \{-1, 1\}$. The mathematical model of the perceptron for $i \in N_{in}$ inputs is

$$y = \text{sign}\Big(\sum_{i=1}^{N_{in}} w_i \cdot x_i + b\Big). \tag{2.12}$$

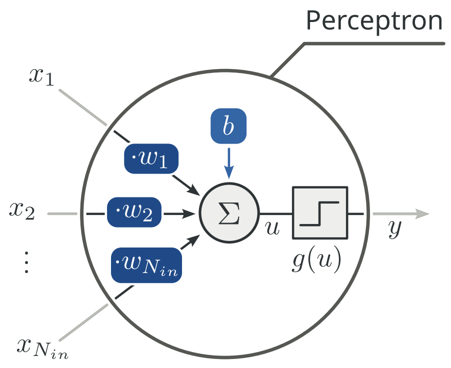


Figure 2.16: The original form of the perceptron by Rosenblatt [130] features weights, bias and *sign* as a binary activation function $g(u)$.

The concept of Rosenblatt's perceptron in 1958 [130] was not substantially different from what were already established machine learning methods at that time. The relevance rather lies in the understanding of the perceptron as a computational unit, which can be aggregated as a network to create models more powerful than those enabled by traditional machine learning [131]. The idea of – by then still linear – multilayer perceptrons (MLPs) was marginalized by Minsky and Papert [132] in 1969 for providing little benefit over an elementary perceptron. And yet, aside from increase of computational resources, two advances are key to the substantial impact that neural network have exerted in the last decades: Firstly, backpropagation, derived from Leibniz product rule, was established as an efficient gradient based optimization algorithm by Linnainmaa [133], which allows training of MLPs. Martens [134] further enables deep networks by proposing a second order method to deal with vanishing gradients during training. Secondly, applying backpropagation to MLPs with smooth and non-linear activation functions, e. g. by Werbos [135] and Rumelhart et al. [136], enables universal approximation of non-linear processes [137], [138].

The Multilayer Perceptron

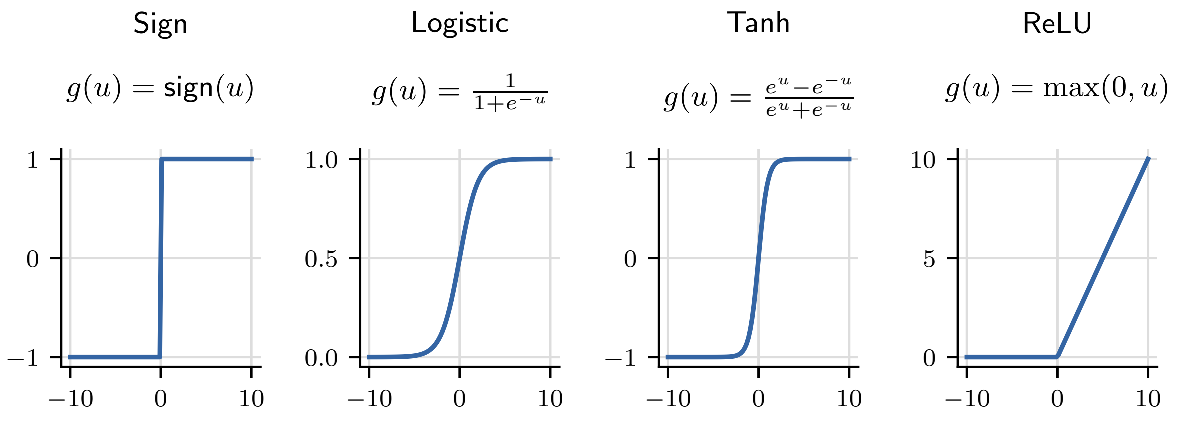


Figure 2.17: The choice of activation function has a significant impact in ANN performance [139].

Linear activation functions restrict the approximation capabilities of MLPs to linear functions, and while the biological inspiration for the perceptron suggests an abrupt and not necessarily continuous step activation, Mitchell [138] proposes differentiable non-linear activation. A special type of sigmoid function, the logistic function, shown in Figure 2.17, suffices these criteria and maps (squashes) the sum of weighted inputs to the range $g(u) \in (0, 1)$. The advantage of the logistic function is that it provides distinct gradients for small $|u|$, beneficial for gradient based training, such as gradient descent. An alternative to the logistic function is tangens hyperbolicus, providing a similar differentiable and monotonous slope. Although worse in terms of computational effort, tanh is zero-centered, which improves convergence during training [140]. An issue which both activation functions, logistic function and tanh, introduce is the problem of vanishing gradients, where early parameters in deep networks experience only little change during training due to the diminishing derivatives [141]. The tanh function is less prone to the vanishing gradient problem, as its derivative is greater in magnitude and therefore typically declines more slowly towards earlier layers of the MLP than the derivative of the logistic function [131]. More prominent as an activation function today is the computationally efficient ReLU function, shown in Figure 2.17, introduced by Hahnloser and Seung in 2000 [142]. Although the strategy of introducing a piece wise linear function seems counterintuitive for gradient based learning due to the fact that ReLU is not differentiable in $u = 0$, the constant slope of $\frac{du}{dx} = 1$ for $u > 0$ allows strong gradients to propagate back to early layers during backpropagation. Still, there are circumstances in which ReLU is not optimal: If the combination of trained weights (positive/negative) and cell inputs (negative/positive) lead to $u < 0$ and forces the output to $\mathrm{ReLU}(u) = 0$, the unit is called *dead neuron* and does not contribute to the overall MLP [131].

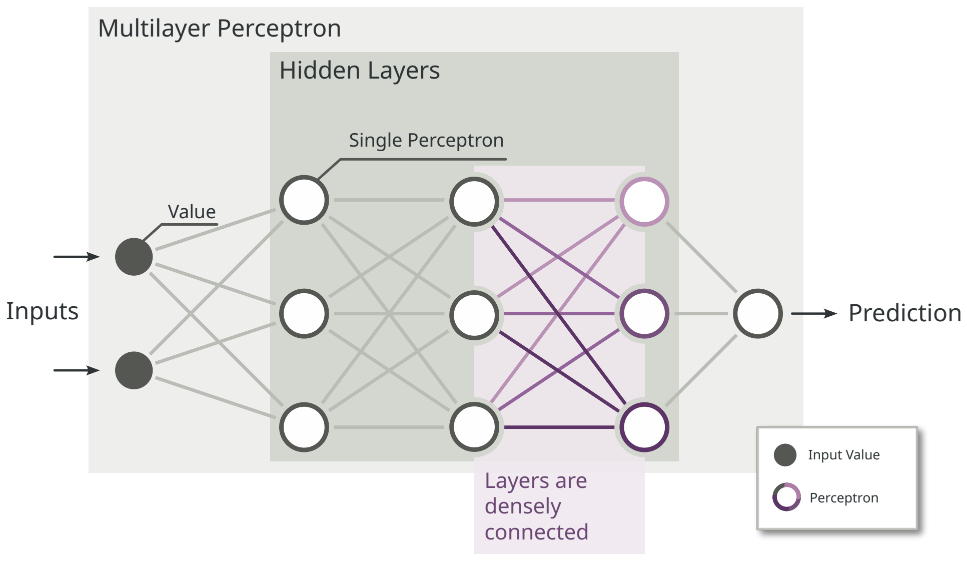


Figure 2.18: The MLP is a feed-forward network with of densely connected layers.

The MLP, as shown in Figure 2.18, is a feedforward neural network where all outputs of perceptrons of a layer are inputs to all perceptrons of the subsequent layer [128]. There are no loops and the flow is unidirectional. Input values of the MLP are provided directly without computation. The performance of an MLP comes mainly from the deep architecture, which introduces multiple hidden processing layers consisting of elementary perceptron units. While size of input and output layer, i. e. number of features and predictions, are constrained by the application the hidden layer count and the size of each individual layer are freely selectable. The densely connected layers require each perceptron unit of index n within layer l to compute their respective output $x_{l,n}$ by

$$x_{l,n} = g\Big(\sum_{i=1}^{N_{u,l-1}} w_{l,n,i} \cdot x_{l-1,i} + b_{l,n} \Big), \tag{2.13}$$

where the previous layer contains $N_{u,l-1}$ units. Each unit contains an invariant bias $b_{l,n}$ and each input i of each unit is attributed with an individual weight $w_{l,n,i}$. The detailed structure of an MLP is shown in Figure 2.19.

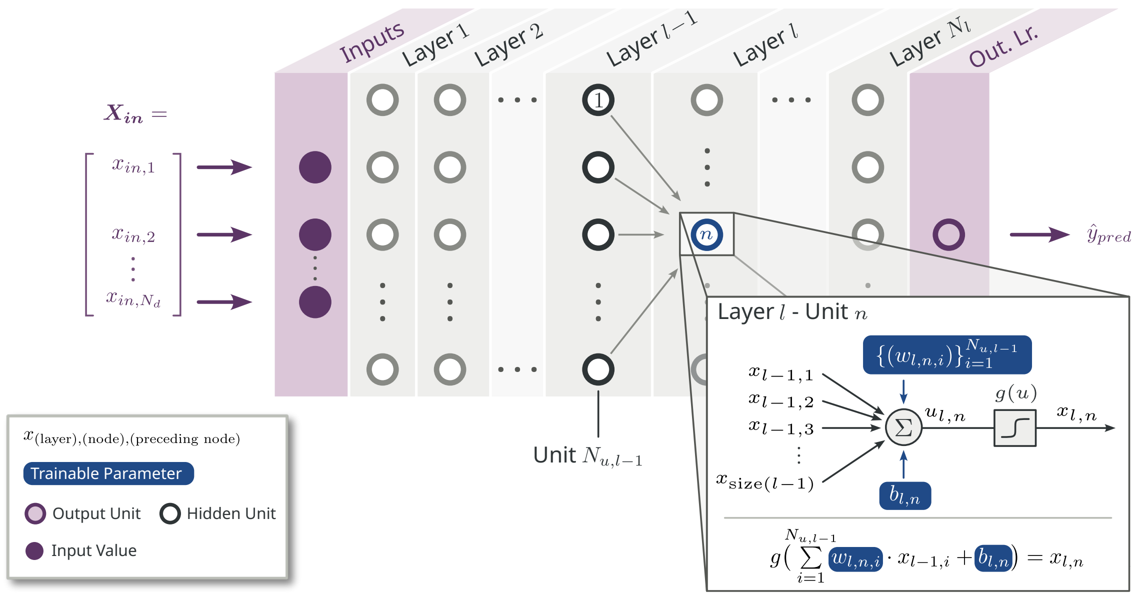


Figure 2.19: Each unit n in layer l of the MLP features trainable parameters $w_{l,n,i}$ to weigh every of its inputs i and $b_{l,n}$ to provide a general offset for the summation stage. After weighted summation, the activation function $g(u)$ is applied.

The forward pass through the MLP, which requires each unit to compute Equation 2.13, allows formulation as matrix multiplication for each layer l as

$$\boldsymbol{X_l} = g(\boldsymbol{W_l X_{l-1}}) = g\left(\begin{bmatrix} w_{l,1,1} & w_{l,1,2} & w_{l,1,3} & b_{l,1} \\ w_{l,2,1} & w_{l,2,2} & w_{l,2,3} & b_{l,2} \\ w_{l,3,1} & w_{l,3,2} & w_{l,3,3} & b_{l,3} \end{bmatrix} \begin{bmatrix} x_{l-1,1} \\ x_{l-1,2} \\ x_{l-1,3} \\ 1 \end{bmatrix}\right) = \begin{bmatrix} x_{l,1} \\ x_{l,2} \\ x_{l,3} \end{bmatrix}, \tag{2.14}$$

where $x_{l=0,i} = x_{in,i}$, $\boldsymbol{X_l} \in \mathbb{R}^{N_{u,l}}$ and $\boldsymbol{W_l} \in \mathbb{R}^{N_{u,l} \times (N_{u,l-1})+1}$. The addition of the invariant bias $b_{l,n}$ can be included into the matrix multiplication, to spare one operation. The advantage of the formulation as matrix multiplication is that implementations can leverage parallel computational capabilities such as single instruction/multiple data (SIMD) instructions to speed up computation.

Weight and bias parameters are subject to algorithmic optimization during training of the MLP. Training MLPs typically relies on backpropagation [135], [136] to optimize the weight and bias parameters as described in Section 2.3.1. Training data are taken from a dataset $\mathcal{D} = \{(X_{in}^{(i)}, y^{(i)})\}_{i=1}^{N_{samples}}$ with a total sample count of $N_{samples}$. Backpropagation relies on two phases with opposite propagation directions [136]. The first phase is a forward pass of an input X_{in} in order to compute the loss $L(X_{in}, y)$. In the second phase, the gradients of $L(X_{in}, y)$ with respect to all weight and bias parameters are determined using the Leibniz chain rule. These derivatives are then used to find the optimum with respect to all $w_{l,n,i}$ and $b_{l,n}$, i. e. the minimum of $L(X_{in}, y)$. Due to the deep architecture of the model, the learned weight and bias parameters are highly incomprehensible for humans and therefore typically left untouched by the user.

Hyperparameter Tuning

What the user can, in turn, configure are the hyperparameters and this section describes methods to find them. Hyperparameters are algorithmic parameters, which influence the training process, such as choice of optimizer, learning rate and MLP structure, i. e. number of layers and sizes of layers. Typically, there are fewer hyperparameters than model parameters and their causal relationship to the deep learning approach is comprehensible [128]. Hyperparameters are specified before training, either by making an educated guess about the structural and algorithmic constraining leading to an acceptable solution, or by employing a tuning algorithm. Figure 2.20 shows a hyperparameter tuning loop for two hyperparameters $A \in \mathbb{R}$ and the discrete valued $B \in \{B_0, B_1, ..., \}$. The loop trains with the current hyperparameter set using the training set. The performance is then evaluated using the validation set and a termination criterion, such as a maximum number of iterations or a target accuracy is checked. Based on this evaluation, tuner either returns or generates a new set of hyperparameters to continue in the tuning loop. Each trial, i. e. training a model for a respective set of hyperparameters λ, is costly, so that the hyperparameter search space for algorithmic tuning has to be as small as possible. Even for a small set of hyperparameters to be tuned, the optimization is costly because of the partly discrete and partly continuous nature of parameters such as layer count and learning rate. Often, for the task of hyperparameter tuning, gradient based optimization is inefficient because of the high cost for training of the model.

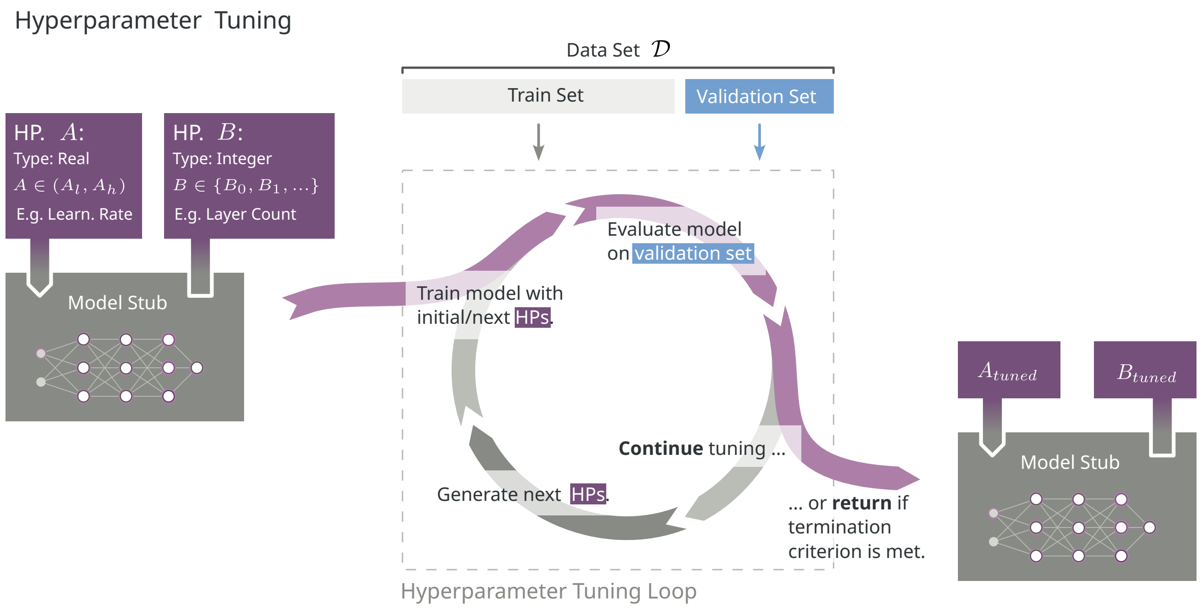


Figure 2.20: Hyperparameter tuning is typically performed in a loop of generating new hyperparameter, training the according model and evaluating the new model on the validation set. Once a termination criterion is met, the tuning loop returns.

The most basic tuning approach when it comes to derivative-free optimization is grid search, which exhaustively tests combinations of the hyperparameter search space, discretizing the continuous hyperparameters. Grid search is feasible if the number of hyperparameters to be fitted is low, as the number of trials increases exponentially with the number of hyperparameters

[143]. Typically, random search is more efficient, as decisions about hyperparameter for future trials are made based on the result of the previous trial [144], instead of a predefined grid. This work, however, focuses on more recent algorithms, namely hyperband tuner and Bayesian optimization.

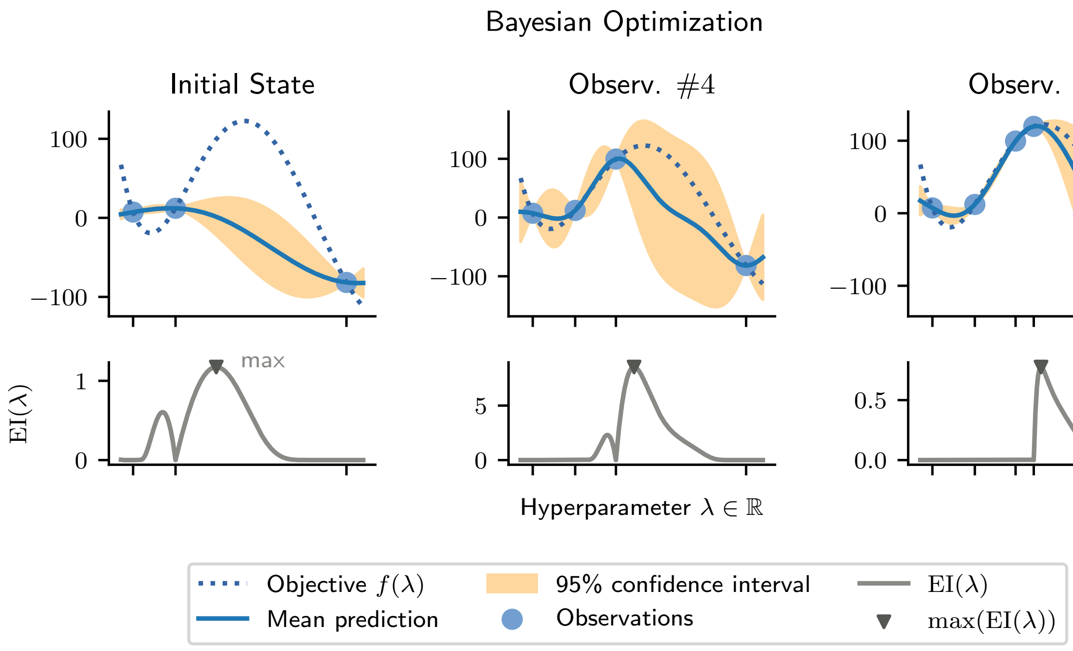


Figure 2.21: Bayesian optimization approximates an objective function $f(\lambda)$ using a surrogate model. An acquisition heuristic determines which value λ to probe next, in order to improve the overall approximation.

Bayesian optimization is a derivative-free black-box approach proposed by Kushner [145] in 1964, which gained popularity as a hyperparameter tuner for machine learning models in the last decades [146]–[148]. The underlying concept of Bayesian optimization is to construct a surrogate model to predict an objective function over the hyperparameter search space, modeling the conditional probability $p(f|\lambda)$, with objective function f and hyperparameter (-set) λ [149]. This surrogate model is typically a Gaussian process, which provides continuous mean and variance of the hyperparameter prediction by postulating a certain smoothness of the unknown function with respect to the hyperparameters. As can be seen in Figure 2.21 with a single hyperparameter $\lambda \in \mathbb{R}$, the confidence intervals show little correlation with the unknown function $f(\lambda)$ when the number of observations is low (e. g. 3). The optimization principle of Bayesian optimization involves an acquisition heuristic. The acquisition function makes a decision based on the current surrogate model (called *prior*) which hyperparameter combination to sample next in order to reduce the confidence interval around $f(\lambda)$ and to improve the function approximation [150]. A commonly used acquisition heuristic is the expected improvement (EI) by Mockus et al. [151], [152], which places the subsequent λ to be observed at the point where the improvement of the surrogate model is expected to be highest, by selecting the maximum of

$$\mathrm{EI}(\lambda) = \mathbb{E}[\max(0, f(\lambda) - f(\hat{\lambda}))], \tag{2.15}$$

where $\mathbb{E}[h]$ is the expected value of h, and $\hat{\lambda}$ is the hyperparameter with the highest observed value $f(\hat{\lambda})$. Figure 2.21 shows that the surrogate model improves by the additional observations placed on the maximum of $\mathrm{EI}(\lambda)$ through Bayesian inference, which leads to a good estimate of $f(\hat{\lambda}) = max(f(\lambda))$ and therefore $\mathrm{argmax}_\lambda(f(\lambda))$. Bayesian optimization based on Gaussian processes is efficient in noisy processes, as the surrogate model allows observations to be afflicted with uncertainty. However, Bayesian optimization is typically used for a low number of hyperparameters (< 20), owed to sharp acquisition functions in higher dimensional optimization problems [148], [153].

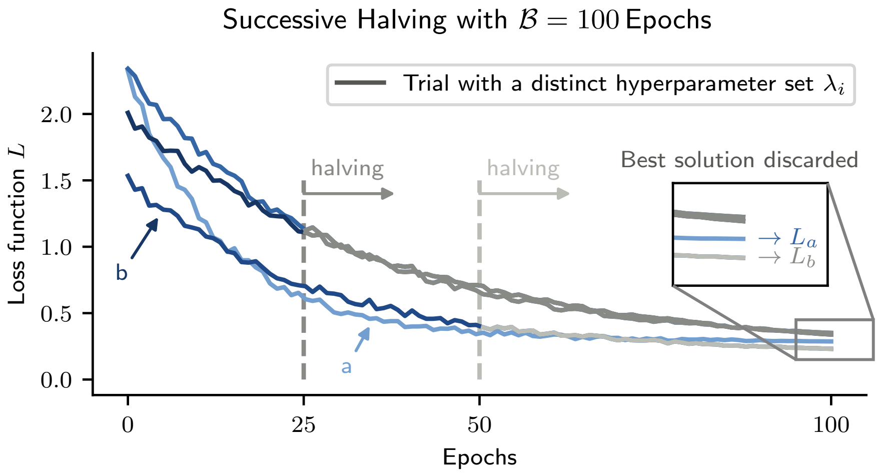


Figure 2.22: A single successive halving step does not necessarily expose the best model. In the constructed example, the halving step at epoch 50 drops the model (b), although it would turn out best at epoch 100. Hyperband therefore conducts a system of successive halving steps with distinctive numbers of trials.

A different approach is taken in Hyperband. Hyperband is a multi-fidelity tuning algorithm by Li et al. [149], which allocates an available resource budget $\mathcal{B}$ (e. g. number of epochs) to an exploratory problem. Hyperparameters can be found by the successive halving method [154], which uniformly allocates $\mathcal{B}$ (e. g. training epochs) to a set of N_λ different hyperparameter combinations $\lambda = \{(\lambda_i)\}_{i=1}^{N_\lambda}$, resulting in budget $b_i = \mathcal{B}/N_\lambda$ for each trial of λ_i, $i \in 1, ..., N_\lambda$. When reaching the maximum number of allowed epochs during a trial, the trial is stopped (early stopping). Successive halving then means to keep the best 50% of the trials and their respective λ_i, allocating the same budget $\mathcal{B}$ to the now $N_\lambda/2$ combinations of λ_i. The resulting budget per trial b_i doubles after every halving step and only the best performing hyperparameter combinations of λ_i are kept a single candidate remains with a tuned hyperparameter set. A drawback of using the plain successive halving for tuning is that different trials are likely to have different convergence properties, so that the limited budget per trial b_i is not necessarily informative about the overall performance of the respective hyperparameter set with a larger budget. Figure 2.22 shows that a hyperparameter set that converges quickly to a loss L_a can be preferred over one that converges to a lower loss $L_b < L_a$, but at a flatter descent, which results in a less than optimal result. The choice of N_λ, which influences the amount of budget allocated to each trial before the respective halvings, is therefore highly relevant for an optimal outcome. However, it is often unknown how early the quality of the trials stands out from each other for a certain deep learning model, and a suitable N_λ is therefore difficult to select manually. Hyperband aims at mitigating this issue by conducting multiple successive halving runs with a different amount of combinations N_λ in a grid search fashion. In the beginning a maximum budget $\mathcal{B}$ for any succesive halving step within a specific value of N_λ, called a *bracket*, and a factor η are specified manually. The first bracket, is performed with a maximum N_λ that is possible for $\mathcal{B}$ epochs, resulting in $N_\lambda = \mathcal{B}$ combinations with one epoch each. Every subsequent bracket $b \in \{b_{max}, ..., 1, 0\}$ permits $N_{\lambda,b} = N_\lambda/\eta^b$ combinations with a respective epoch budget of $\mathcal{B}_b = \mathcal{B}/N_\lambda \cdot \eta^b$. With every bracket the number of halving steps are reduced by one, until the last bracket $b = 0$ allocates the maximum allowed budget $\mathcal{B}$ without halving steps, which resembles a random search.

The question which hyperparameter tuning method to use for a given machine learning model is typically answered with reserved advice, if at all. Even the computational effort of each method depends on the number of the hyperparameters and the size of the search space. A general principle that this thesis follows is to prefer Hyperband when discrete valued hyperparameters such as layer count and units per layer are involved. Bayesian optimization is designed for continuous value hyperparameters, although rounding methods allows optimization of discrete values [155]. Still, to converge towards an optimal solution when tuning a small number (< 20) of continuous hyperparameters, Bayesian optimization is often the tuner of choice.

2.3.2 Symbolic Regression

Neural networks, as explained in Section 2.3.1, require structural constraints, e. g. in form of model architecture. During training the parameters are fitted to the underlying and unknown distribution of the data set $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_1^{N_s}$, so the structural constraints are crucial for model performance as much as the structure of a polynomial equation is for a curve fitting problem. And the complexity of an MLP, i. e. the number of hidden layers and their respective units, can only partially make up for the fact that the distribution of $\mathcal{D}$ is only approximated by a large combination of multiplication-summation-activation. This limitation becomes apparent when training an MLP with a periodic distribution, which typically leads to poor generalization behavior [156], [157]. So the question if an MLP can really grasp and represent a mathematical principle sampled in $\mathcal{D}$ is at times not so much a question about the complexity of the described principle, but rather about the expressiveness introduced by activation functions. A very basic example by Minsky et al. [132] demonstrates this principle early on: when using only linear activation functions, the MLP can only represent linear distributions. It is common to choose the activation function mainly based on convergence and performance during training, as the single computations of the respective units are not comprehensible to humans anyhow. The resulting model provides a deep network of solitary computations that fit the training data well and ideally all data in close proximity [158]. However, if the true physical or mathematical principle behind the distribution is apparent, e. g. because it represents a harmonic function within a waveguide or the exponential decay of an unstable material, it is preferable to obtain a simple and short equation which models the general causality. This path can lead from deep learning back to curve fitting, if the underlying principle is known ahead and well understood, so the only computational task is to perform regression with respect to manually introduced parameters. But in case it is too costly to manually analyze and characterize $\mathcal{D}$, which is usually the main motivation to lean towards deep learning, symbolic regression is a promising candidate.

Symbolic regression is the task of optimizing structure and parameters of an analytical model with respect to a given distribution

$$y(X) = \phi(X, \theta) + \epsilon, \tag{2.16}$$

with variables $X = (x_1, x_2, ..., x_{N_d})$, space of analytical expressions ϕ and space of parameters θ and white noise ϵ [159], [160]. The result of symbolic regression on observations from Equation 2.16 is a closed form analytical expression

$$\hat{y}(X) = \hat{\phi}(X, \hat{\theta}) : \mathbb{R}^d \to \mathbb{R}, \tag{2.17}$$

with feature count N_d and $X \in \mathbb{R}^{N_d}$ [159]. One advantage of optimizing the structure along with its parameters is that the bias introduced by human constraining, e. g. in form of a neural network architecture, is avoided. Further, the interpretation of the resulting analytic equation can provide insights into underlying principles of the sampled process [159]. The value of interpretable models is particularly high if predictions lead to decisions which impact human life, e. g. in health care or justice [161]. Another finding by Wilstrup et al. [162] indicates that symbolic regression often outperforms other machine learning methods, such as lasso regression, gradient boosting, random forest and decision trees, when the available data set $\mathcal{D}$ is small.

There are several approaches to symbolic regression, which rely on different optimization techniques. A traditional approach to symbolic regression is genetic programming, which generates a population of solutions and defines genetic operations such as mutation and recombination among them [163], [164]. The general aim is to increase the fitness of the population by selective breeding, which results in a generation of novel solutions. Ideally, selective breeding provides one or more acceptable solution after multiple generations. Typically, the outcome of symbolic regression through genetic programming can show good fitness, however a drawback is that complexity of the promising candidates is high [165]. As a consequence, computation effort for prediction rises, but more importantly, overly complex models are difficult to interpret. Recently, approaches aim for minimum complexity and the possibility to incorporate prior knowledge such as Bayesian Symbolic Regression by Jin et al. [166] and AI Feynman by Udrescu et al. [167], which exploits structural assumptions in physical equations. Operon by Burlacu et al. [168] is based on genetic programming incorporating gradient based optimization. Deep Symbolic Regression by Petersen, Landajuela et al. [169], [170] exploits reinforcement learning and recurrent neural networks. Fast Function Extraction by [171] generates linear and non-linear base functions, extracts promising candidates using pathwise regularized learning and combines them in a linear model.

Two assessments facilitate the choice of method and implementation for this work. Cava et al. [159] propose SRBench in 2021, a benchmark set with 252 regression problems and compares 14 contemporary symbolic regression implementations against 7 traditional machine learning implementations, such as MLP and random forest. The two types of problems for which the candidates are evaluated are black box regression, where $y(X)$ is unknown, and ground truth regression, for with the sample set $\mathcal{D}$ is created from an analytic function $y(X)$. As the best performing implementations for black box regression, Operon stands out with high accuracy. Among the approaches that do not rely on genetic programming, Deep Symbolic Regression and fast function extraction share the first place. For ground truth regression, AI Feynman shows low accuracy when the target function is afflicted with noise, rendering its use with real world data, e. g. measurements, questionable. Even in the presence of noise, Deep Symbolic Regression significantly outperforms Operon and AI Feynman and often recovers the exact solution $y(X)$.

In 2022, a new competitor enters the SRBench Competition [172], a showdown of symbolic regression implementations at the Genetic and Evolutionary Computation Conference in Boston, MA. The synthetic track rewards among others the discovery of the true $y(X)$, extrapolation and sensitivity to noise. QLattice [173], [174] relegates Deep Symbolic Regression to the third place and significantly outperforms Operon. A second track deals with generation of models for COVID-19 prediction, where Deep Symbolic Regression outscores QLattice by little. Being among the first two/three places in both disciplines, QLattice and Deep Symbolic Regression are both deemed promising candidates for technology modeling. In early evaluations, which aimed at modeling the channel current of the planar RFET from simulation data, QLattice consequently outperformed Deep Symbolic Regression. Therefore, this section – and this work in general – focuses on QLattice.

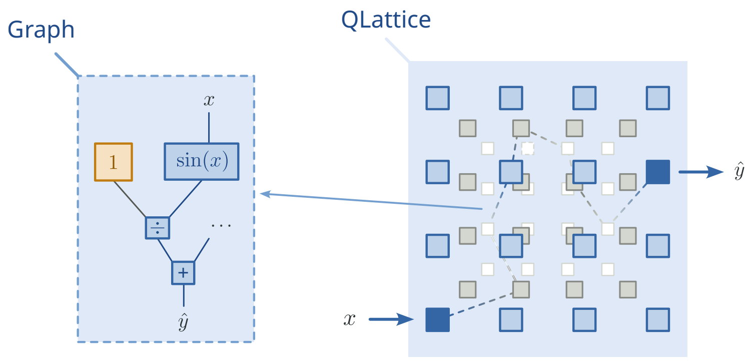


Figure 2.23: QLattice finds symbolic equations by optimizing pathes through a structure, where the nodes represent arithmetic operations, called *interactions*. The symbolic equations are represented by graphs.

It has to be noted that Qlattice is a commercial product by Abzu and that source code and detailed working principles are not publicly available. The description is therefore based on a US patent about the derivation of correlations [173] by Abzu founder and CEO Caspar Wilstrup, a publication about QLattice by Broløs et al. [174] and documentation of its application programming interface (API) [175]. QLattice treats analytic equations as unidirectional acyclic graphs where the analytic operations are represented by the nodes. Nodes therefore describe the interactions between terms or variables, which can range from simple functions, such as addition/subtraction to more complex functions such as exponential, logarithmic and trigonometric functions. Inspiration for QLattice comes from Richard Feynman's perspective on quantum mechanics: Similar to the infinite number of trajectories that are summed up in Feynman's path integral formulation [176] a large amount of paths, [174] proposes more than 1000, are created through the lattice with the aim of fitting the given distribution $y(X)$ represented by the set of samples $\mathcal{D}$. Convergence towards the most useful path, in analogy to the convergence to the path of the least action that leads to Newtonian mechanics, then results in sound solutions. The singular solution is a unidirectional and acyclic graph which represents a symbolic equation as shown in Figure 2.23. As the interactions are sampled from a probability distribution, typically more than one solution exist and a set of useful solution graphs is available in a structure called QGraph. As shown in Figure 2.24 with the reconstruction of the $\mathrm{sinc}(x)$ function, the QGraph proposes a ranking of solutions. The solution ranking is based on a loss function in combination with information criteria such as Bayesian information criterion (BIC) or Akaike information criterion (AIC), which penalize model complexity. A useful feature of QLattice is to further allow the user to manually select promising candidates from the QGraph, based on which the lattice probabilities adapt for further optimization runs. This manual selection allows incorporating prior assumptions into the optimization.

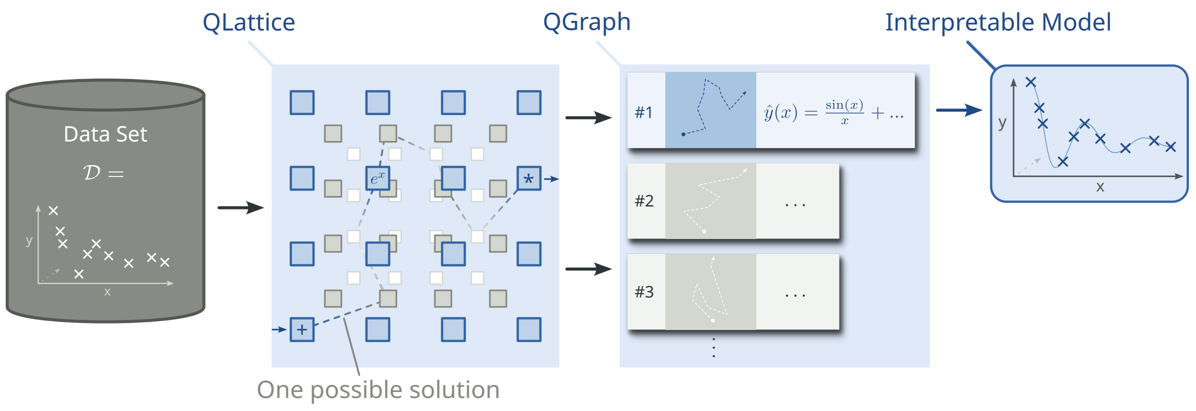


Figure 2.24: Symbolic regression using QLattice generates multiple model candidates in form of graphs, accumulated and ranked in the QGraph structure. Models, which are symbolic equations, can be interpreted to gain insights into underlying causalities and mechanisms in the distribution of $\mathcal{D}$.

2.3.3 Ensemble Models

If one model performs subpar, an ensemble of multiple models can lead to improvement. On the one hand this approach seems intuitive, as increased overall model size can be related to better fitting capability and possibly better generalization, given that the distribution behind the data is complex. On the other hand, there is Ockham's Razor:

"Nunquam ponenda est pluralitas sin necesitate."
– William Ockham, 1285-1347, [177], [178]

William Ockham claims in a philosophical context that the multitude of entities is not to be adopted without necessity. If the simpler theory shows the same accuracy, it is the preferable theory. Ockham's Razor is found frequently throughout machine learning literature as a plea to stick to the minimal possible model complexity [128], [138], [179]–[181]. Domingos [178] extracts a specific guideline with respect to modeling and machine learning: Knowing that the generalization error of two models is equal, the simpler model is to be preferred. Still, there are situations in which it is beneficial to combine the predictions of multiple individual models to provide a more robust overall prediction and the result is called *ensemble model* [182].

Ensemble models often come into play when generalization ability of individual learners is not sufficient for a classification task. There are two criteria for success [183]: The individual classifiers need to have a better accuracy of prediction than random guessing when it comes to generalization, i. e. prediction on unseen data. Further, the individual classifiers are required to be diverse, meaning that they show different errors when generalizing. Increasing the generalization ability of the model is often difficult, which is why an ensemble model approach is selected in the first place [184]. The question of how to generate diversity between the set of models remains.

Li et al. [184] divides the task of introducing diversity between individual neural networks into *transform training data*, *change neural network characteristics* and lastly *individual neural network optimization*. Transformation of training data refers to providing a different set of training data to each individual learner. An intuitive and broadly used example is the *random forest* classifier, which combines the predictions of multiple decision trees. Each decision tree is trained on a specific subset of the complete training set $\mathcal{D}_{train}$. For regression problems, common strategies are variants of *boosting* [185], attributing a weight to each learning sample. After training the model, the weights are then updated depending on their learning effect and all obtained models are combined to create strong predictions. Changing neural network characteristics can be as simple as changing initial weights of the model [184], but can also mean to substantially change model structure [186] or loss function [187]. Individual neural network optimization refers to the task of selecting the ideal individual models to contribute to the ensemble model, which is not part of this work.

After having established an accurate and diverse set of individual models, the individual predictions are combined to obtain the ensemble prediction. A commonly used strategy for merging N_{ens} individual predictions $\hat{y}_i$ with $i \in \{1, ..., N_{ens}\}$ into an ensemble prediction $\hat{y}$ is averaging [128], [184]

$$\hat{y} = \frac{1}{N_{ens}} \sum_{i=1}^{N_{ens}} \hat{y}_i \ . \tag{2.18}$$

Further, weighted averaging in form of

$$\hat{y} = \sum_{i=1}^{N_{ens}} w_i \cdot \hat{y}_i \tag{2.19}$$

allows to attribute a weight $w_i \in \mathbb{R}$ to each individual prediction and therefore adapt the individual influence of each learner to the prediction. This concept can be driven even further by assigning dynamic weights, as shown by Adhikari et al. [188], who assign time-varying weights for time series forecasting such as stock prizes and inflation rate.

2.3.4 The Importance of the Data Source

Machine learning methods in general rely on data sets, which express the characteristic of a process. The data set $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^{|\mathcal{D}|}$ represents the available information for the modeling approach to learn, and optimal learning imposes certain quality requirements on the data. A baseline quality requirement for a data set is balance. In classification problems, balance corresponds to the similar distribution of samples over all classes. Likewise, regression models benefit from uniform sampling as prediction quality depends on the sample density of the respective area, if the importance of prediction is distributed uniformly over the sample space [158], [189], [190]. However, data sets are often byproducts of real world processes, or in general processes where observability is limited, and therefore a balanced sample distribution may not be available. In case of imbalanced sample distributions, the mitigation strategies can be threefold [191]: One approach is modifying the sample set $\mathcal{D}$. A mitigation strategy especially in chemical and biomedical settings is to resample the data in order to balance their distribution over target classes [192]. Resampling is also useful when it comes to regression methods and there are various strategies such as random over-/under-sampling in combination with the addition of noise [193]. The second approach targets algorithm level, which balances existing biases within the data set during regression and the third approach combines advantages of both methods in a hybrid manner. In this thesis the focus is placed on modifications of the sample set, as it is preferred to alter the design of experiments and conduct post-processing to tackle the root cause of the imbalance.

Learning requires testing, and if the entire available information is seen during training, evaluation of the learning result is restricted to check if the candidate can reproduce what is known. In the spirit of keeping new exam questions undisclosed ahead of the exam to evaluate a candidate's transfer knowledge, machine learning models are typically not tested with the same data set which they are trained on. Instead, the available information $\mathcal{D}$ is split up into a training set and at least one other set on which the model is evaluated to assess its generalization ability. This additional set, in this work called *test set* $\mathcal{D}_{test} \subset \mathcal{D}$, is unseen during training and allows the comparison of generalization between distinct models, e. g. by comparing loss or other metrics on the test set, as shown in Figure 2.25.

A typical pitfall when training machine learning models is overfitting. Machine learning models are optimized with respect to a training set, but the goal is to achieve good generalization on data unseen during training, characterized by a low test error [194]. The training error, being the optimization target, is typically lower than the test error and the discrepancy indicates a failure to generalize. The model then *overfits*. Overfitting may occur if the complexity of the model structure, and therefore its ability to fit complex functions, exceeds the complexity of the distribution behind the available data. It is possible to detect overfitting during training by monitoring the training loss along with the loss on an additional set, on which the model is not trained. Especially deep learning approaches therefore benefit from further splitting to create a third set, the *validation set* $\mathcal{D}_{val}$, while QLattice only works on training- and test set. As QLattice is efficient on small training sets and does not require hyperparameter tuning or early stopping, the test set can make up a larger share of the available data, as shown in Figure 2.25. The situation to be avoided is a decreasing training loss without decreasing validation loss and the best generalization is achieved at the minimum of the validation loss. When suspecting a minimum in the validation loss, the optimization can be stopped early and the parameter values leading to a minimal validation loss are returned as result. The validation set is further used for evaluation of hyperparameter combinations, as explained in Section 2.3.1. This leaves the test set for final assessment of model performance on data which are neither seen during training, unlike the training set, nor used to steer modeling decisions, such as the validation set. Splitting the data set prior to training is called *hold-out*, and there are other methods such as *cross-validation*, where the data set is split up into multiple groups which change between training and validation role during fitting [128]. Hold-out is a popular choice for deep learning, as it is simple to implement and efficient [131].

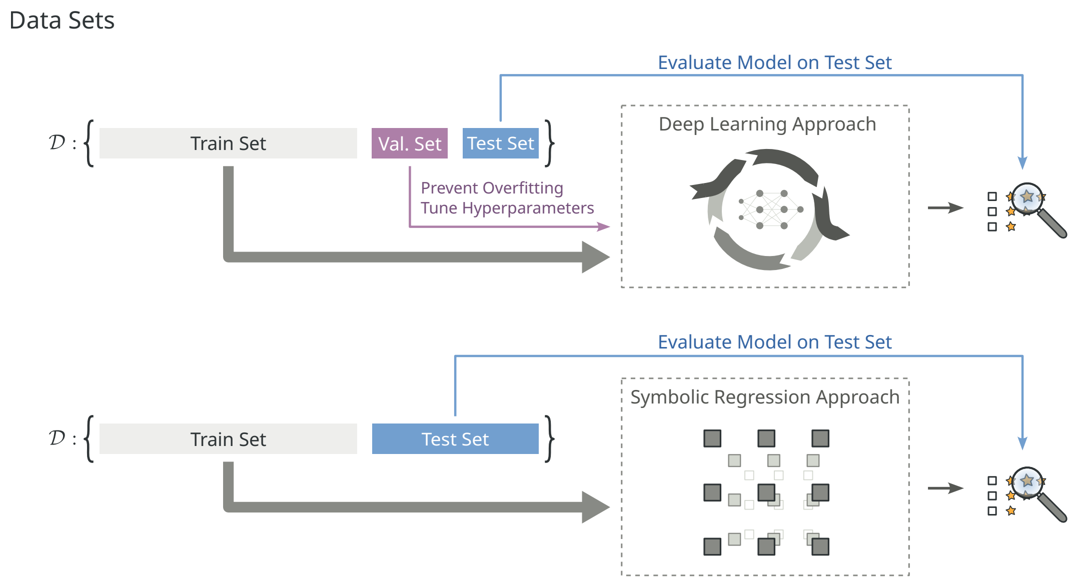


Figure 2.25: Deep Learning typically features a split of three sets: A validation set is evaluated during training to allow early stopping and to tune hyperparameters. Symbolic regression in general does not feature hyperparameters, and QLattice in particular does not allow early stopping. A validation set is therefore not required.

The design of machine learning based methods involves many decisions based on experience, such as model selection and hyperparameter constraints. The ideal split between training, validation and test set is another. Typical splits between train and test set found throughout literature are 20%/80% and 30%/70% [128], [194]. The goal of splitting is to preserve the statistical properties of the original data set within train, validation and test set and a typical way for large datasets is a randomized strategy [195]. A more sophisticated but intuitive splitting strategy is proposed by Kennard and Stone [196]. Kennard and Stone propose to define a distance metric, the Euclidean distance

$$D^2_{\nu\mu} = ||X_\nu - X_\mu||^2 = \sum_{i=1}^{d}(x_{\nu,i} - x_{\mu,i}), \tag{2.20}$$

where $X_i \in \mathbb{R}^d$, and interpret each tuple X_α in $\mathcal{D}$ as vectors to a point P_α. An empty set $\mathcal{D}_*$ is created and the first two points to be appended are either two a priori selected points, or the points P_β and P_γ with the highest Euclidean distance $D^2 = \max_{\beta,\gamma}(||X_\beta - X_\gamma||^2)$. Each additional point is then the point with the highest Euclidean distance to any member of $\mathcal{D}_*$. The results of Kennard-Stone splitting are two data sets, where a set $\mathcal{D}_* \subset \mathcal{D}$ shows high uniformity of sample distribution with respect to the Euclidean distance and is therefore well suited for training. The set of remaining samples $\mathcal{D}_{remaining} = \mathcal{D} - \mathcal{D}_*$ is held out for test and validation.

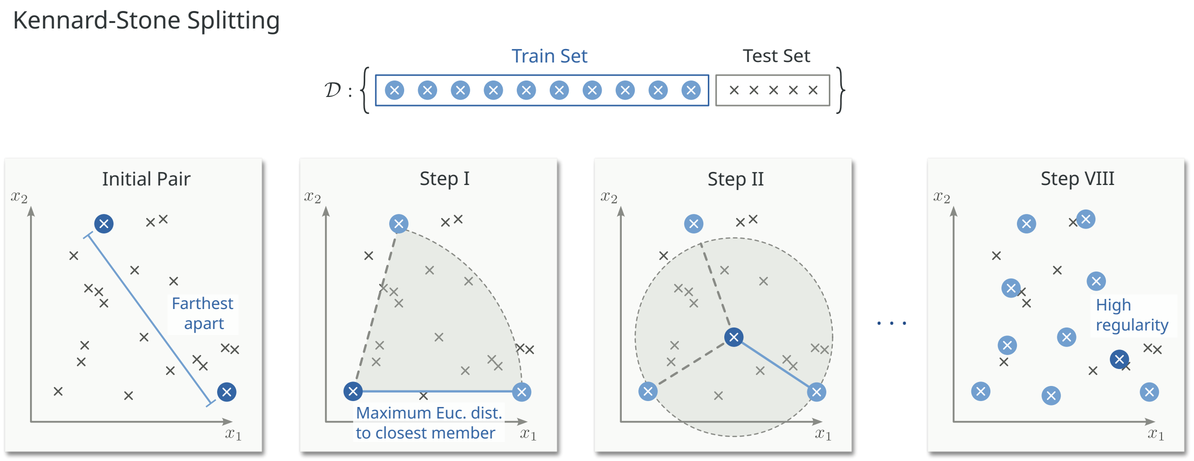


Figure 2.26: The algorithm of Kennard and Stone [196] next picks the sample which shows the highest euclidean distance to all samples in the present set. The resulting data set shows high regularity with respect to the input x.

Machine learning methods develop their potential with the amount of data available to learn from – at least to a certain extent. There are saturating effects which diminish the benefit of additional samples, for example the increasing information overlap which Cui et al. [197] show for classification through deep learning. However, it is typical for machine learning approaches to work on a limited amount of data, be it due to the cost of each data point, e. g. because simulation requires resources, or the inherent limitations of available data, e. g. when predicting health related issues from a low incidence. A cheap way of obtaining more data is to append a transformation of the existing data set $\mathcal{D}$, which is called *augmentation* [194]. Especially in classification, where the classifier is expected to be invariant regarding a specific transformation $t(X)$, this transformation $t(X)$ can be applied to all $(X_i, y) \in \mathcal{D}$, improving generalization of the model [131]. The vast majority of augmentation methods is traditionally centered around classification tasks, in particular image classification. Sietsma and Dow [198] propose to afflict input vectors with Gaussian noise for training of a classifier and discover that generalization for noisy inputs improves. Yaeger et al. [199] augment the data with stroke variations of handwritten characters. But regression methods can benefit from data augmentation as well, as recently shown by Hwang and Whang [200]: RegMix creates mixtures between samples after identifying the nearest neighbors, respectively. Noise augmentation is used by Raju et al. [201], who improve generalization of a linear regression model for failure analysis and reverse engineering of semiconductor models [202]. Raju et al. replicate the training set and add white

Gaussian noise to the labels of the replica, achieving a higher robustness in generalization. Hutchins et al. [112] use Gaussian noise augmentation for compact modeling of a memristor.

2.3.5 Metrics and Loss Functions

An integral part of optimization is the calculation of errors. The intention is to either minimize a certain loss function or to evaluate a model with respect to a ground truth data set of size n. A basic loss function is the mean absolute error (MAE)

$$MAE = \frac{1}{n}\sum_{i=1}^{n} \left| y_i - \hat{y}_i \right|, \tag{2.21}$$

which is intuitive to interpret, as it preserves the unit. When using the MAE as a loss function, the errors are included linearly and the function is not smooth or differentiable because it considers the respective absolute errors. A commonly utilized error metric therefore is the MSE

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{2.22}$$

which is smooth, differentiable and sensitive to outliers due to their squared contribution. Both MAE and MSE can be considered absolute metrics, because they depend on the magnitude of the deviation. The R^2 score

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (\hat{y}_i - y_i)^2}{\sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2}, \tag{2.23}$$

with average ground truth result $\bar{y}$ reduces the influence of the magnitude of the deviation because it references to the total variance. The R^2 score is a commonly used metric to compare accuracy of multiple models which are not necessarily trained on the same data set [112], [116], [203].

Situations where the relative error is of interest require the deviation to be referenced to the magnitude of the ground truth result. The mean absolute percentage error (MAPE)

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \tag{2.24}$$

is an extension of the MAE, which divides by the respective ground truth result. A drawback of the MAPE is that the division by y_i is not defined for $y_i = 0$ and enables a large contribution for sumands with y_i that are low in magnitude, such as leakage current. Following on from the MAPE [204], the symmetric mean absolute percentage error (sMAPE)

$$\text{sMAPE} = \frac{1}{n}\sum_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{(\hat{y}_i + y_i)/2} \right| \tag{2.25}$$

divides by the average of prediction and ground truth, which avoids the singularity at $y_i = 0$. The disadvantage of the sMAPE with respect to MAPE is that interpretation is less intuitive.

3 Generating Device Data

Fast and accurate modeling requires information about the key characteristics of a device. For technology-aware modeling of conventional semiconductor devices these key characteristics can be extracted by measurement or simulation and then be introduced as model parameters, as described in Section 2.2.2. In the domain of emerging devices, however, often either conventional compact models or detailed physical insights into operating principles are unavailable, so that a more general approach is required.

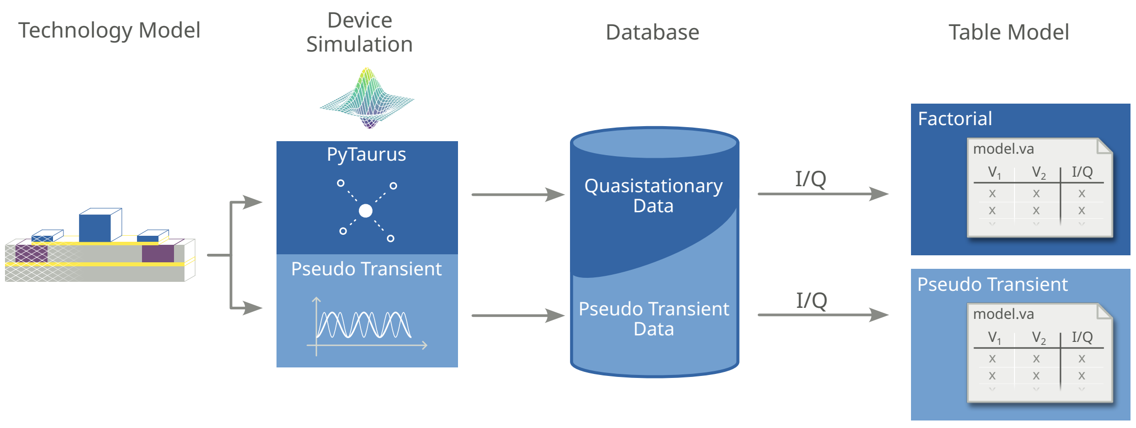


Figure 3.1: In this work, characteristic data are generated either by the factorial simulation cluster PyTaurus, or by the pseudo transient simulation approach. The simulation results, currents and charges, are stored in a database and lead to two table models, derived from the same technology model.

The modeling approach in this work is data driven, in order to be versatile and generally applicable for novel device concepts. Data driven approaches have the advantage of being mostly technology-agnostic and can therefore, after definition of key parameters, be carried out as a fully automated tool flow. The data source this work is based on physical simulation, which is costly to conduct, as described in Section 2.2.1. For this reason, this chapter is dedicated to efficiently generate characteristic device data.

This section describes the generation of representative data sets for predictive circuit simulation of emerging semiconductor devices by sampling a DUT, as depicted in Figure 3.2. The purpose of the presented data generation methods is to obtain

1. I_{DC}, being the DC drive current of the planar RFET, and
2. $Q_{e,\{0,\ldots,N_e\}}$, being the electric charge of the N_e actively driven electrodes and a passive reference electrode E_0.

Description of the methodology in this chapter is centered around the DC drive current I_{DC}, for sake of simplicity, but extends to the electrode charges. The generated data sets also satisfy the constraints for Verilog-A table models discussed in Section 2.2.3 and can directly be used for circuit simulation in Cadence SPECTRE. The methods are built around Sentaurus TCAD, to carry out the respective electrical device simulations of the planar RFET, but are designed to be generally applicable to other TCAD suites and other semiconductor devices. With availability of structured device data, the path for table model simulation is open.

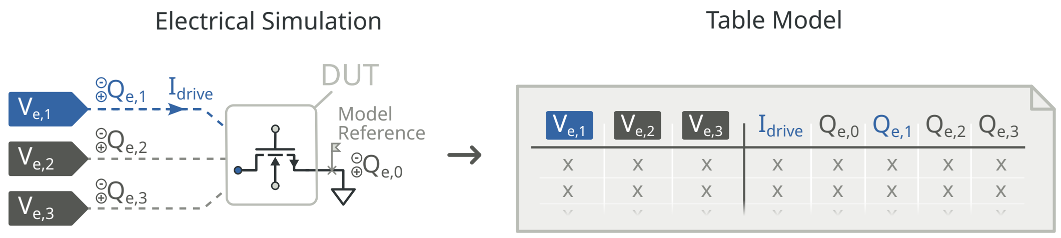


Figure 3.2: Device characterization is done by sampling the DUT over a predefined structure of bias point and extracting the respective DC drive current and electrode charges. The resulting data points can be used as table model for circuit simulation.

The requirements for a data set which accurately represents a transistor in logic cell simulation are discussed in Section 3.1. PyTaurus, a Python tool suite to set up, conduct and evaluate a factorial setup of quasistationary physical simulations for generation of a table model, is presented in Section 3.2. A fundamentally different approach is proposed in Section 3.3: A single slowly proceeding transient simulation with nested harmonic functions returns a dataset, which can directly be extracted as a table model.

3.1 Data Driven Models for Logic Cell Simulation

The modeling process of a semiconductor device for circuit simulation starts with the assessment of input variables, which are typically the input voltages. Although other methods are possible, it is common to select a single reference potential as shown in Figure 3.2 and then define the voltage of each electrode with respect to this reference. In case of targeting static logic cell simulation the range of DC input voltages is constrained to $[V_{SS}, V_{DD}]$. As the potential of the reference electrode $V_{e,0}$ lies within the supply voltages, so does the magnitude of all remaining electrode voltages $\left|V_{e,\{1,2,\ldots\}}\right|$, referencing to $V_{e,0}$. The use case of the model then determines the input voltage range that is required to be represented by the model. In the case of conventional static CMOS, such as the inverter cell shown in Figure 3.3, a dedicated PMOS model operates with the electrode voltages $V_{GS}, V_{DS} \in [-V_{DD}, 0\,V]$, using source as reference. Complementary to the PMOS, all NMOS electrode voltages are positive, and so it is sufficient for the NMOS model to support input voltages within quadrant I. The strict structural requirements of static CMOS allow the input voltages of each of the two device models to be confined to their respective quadrant.

RFETs, however, provide p-type, n-type and ambipolar conduction modes, which a model is required to represent accurately. After various approaches of composing RFET models from a dedicated p- and n-type model such as [101], [107] described in Section 2.2.2, this work aims at creating monolithic RFET models for digital cell simulation. The example of the *XOR* in Figure 3.3, as introduced in Section 2.1.3, shows that when the same model is instantiated for

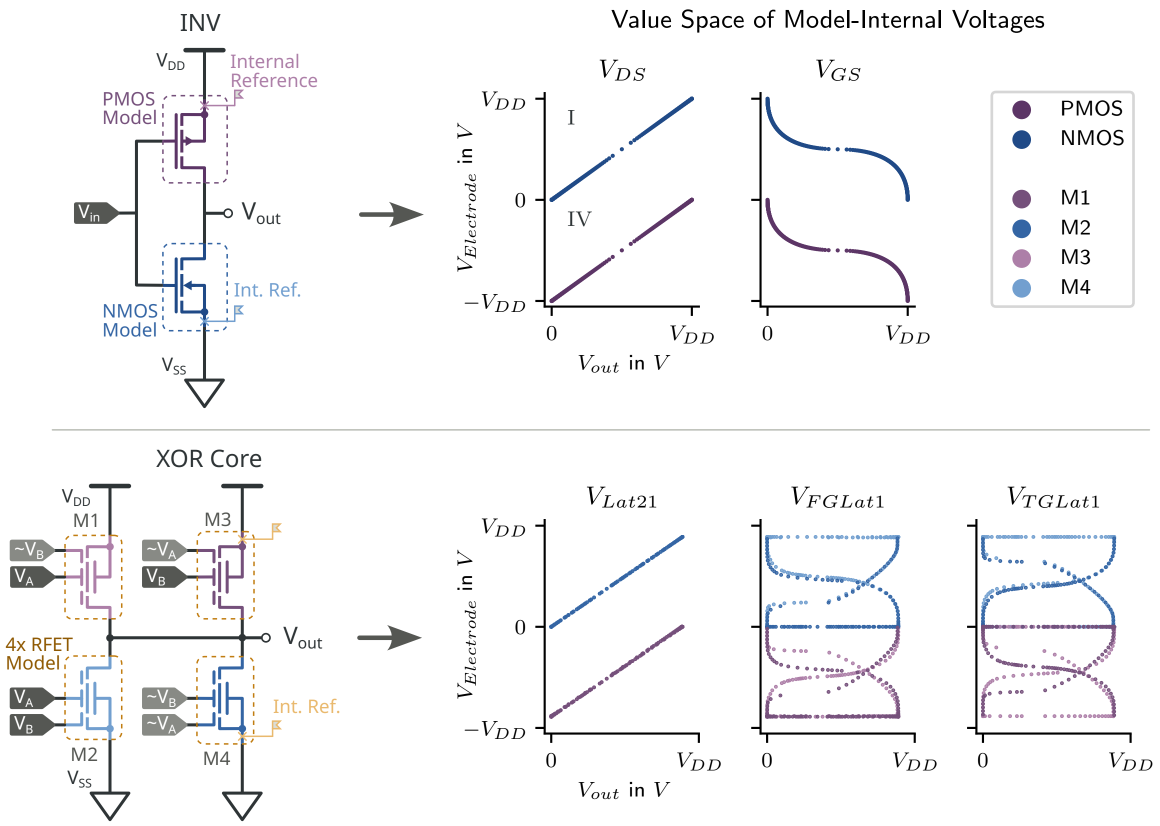


Figure 3.3: CMOS devices feature a limited range of input voltage, as they are constrained to their respective pull branch (-up, or -down). The versatility of RFETs requires characterization over a wider range, e. g. over two quadrants for the *XOR* cell.

all devices $M1$ to $M4$, the range of input voltages V_{TGLat1}, V_{FGLat1}, V_{Lat21} spans quadrants I and IV. DC simulation confines all electrode voltages within a static CMOS circuit to $[0, V_{DD}]$ with respect to V_{SS}. However, over/undershoots and other transient effects can exceed the supply rails, which is why a robust model ideally features an input range, which exceeds $[-V_{DD}, +V_{DD}]$. Estimation of transient effects is not part of this work, so a lump sum of $V_{DD} \pm 0.2\,V$ is assumed. This leads to a basic requirement regarding the range of input voltages for a versatile compact model of the RFET:

$$V_{e,\{1,\ldots,N_e\}} \in [-V_{DD} - 0.2\,V, +V_{DD} + 0.2\,V]. \tag{3.1}$$

3.2 PyTaurus – A Factorial Simulation Cluster for TCAD

PyTaurus is a tool suite developed in the scope of this work, to facilitate generation, execution and evaluation of structured, i. e. factorial, experiments for Sentaurus TCAD. There are three main goals, which justify the development of a custom solution, specifically dedicated to table model generation: Firstly, with the multitude of technology simulations required for to generate a table model, easy setup of the model becomes important. Ideally, only constraints are formulated, handing generation and execution of the respective technology simulations over to the tool suite. Especially with a rising number of gate electrodes, the complexity of

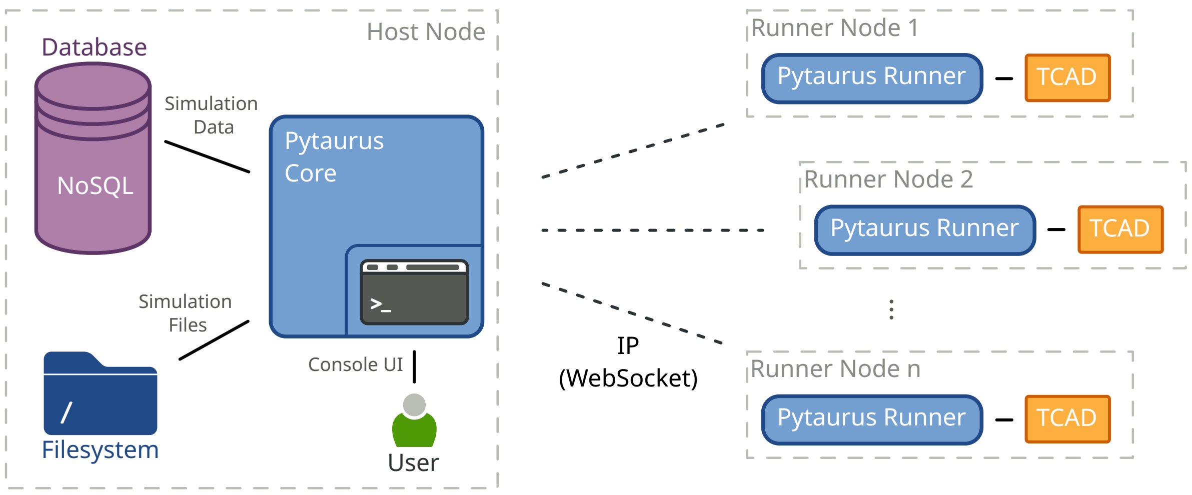


Figure 3.4: The PyTaurus simulation cluster generates the data structure for factorial simulation and distributes the simulation deck to available runners over WebSocket.

table model building cannot efficiently be handled manually. Secondly, if a table model is to be extended, e. g. by adding custom voltage steps to an arbitrary dimension, the tool suite should reuse as many data points as possible and propagate the change with a minimum number of additional simulations. It is emphasized, that the extension of the table model should happen automatically at a change of the underlying table model building constraints. Thirdly, technology simulation is prone to convergence issues. Due to model complexity, convergence aids should be available to the user which require minimal user action to alter numeric parameters, when necessary.

In PyTaurus, simulations are set up as DC parameter sweeps to fulfill the table model constraints for Verilog-A, as described in Section 2.2.1. Multiple parameter sweeps are set up and conducted to cover the bias space of the model with respect to its N_E electrodes in the tuple of electrodes E, excluding the reference electrode, with the purpose of exploring the bias space $\mathbb{R}^{N_E}$. For $N_E > 2$ and a sweep electrode E_{N_E}, the design of experiments can be described as a full factorial experiment for $E_1, ..., E_{N_E-1}$, where each obtained data tuple $(V_{e1}, V_{e2}, ..., V_{e(N_E-1)})$ results in a unique OFAT experiment, a parameter sweep of E_{N_E}. The discrete values of each electrode voltage step within the full factorial design can be chosen individually.

To generate and manage large data sets, it is imperative to develop versatile generation functions and flexible data structures. PyTaurus Figure 3.4 is designed for systematic setup and simulation of bias tables from TCAD models according to the conventional quasistationary sweep method explained in Section 2.2.1. The numeric parameters of a table model, e. g. operating range and granularity of each electrode, can be specified in detail along with all technology parameters of the respective TCAD device model. The resulting set of TCAD simulations is then scheduled to available computing nodes to execute simulations in parallel. All simulation files, e. g. results and log files, are stored on the file system to maintain database performance. All obtained bias point are then united in a Pandas [205] DataFrame object and finally exported to *.csv* tables to be referenced as Verilog-A table models.

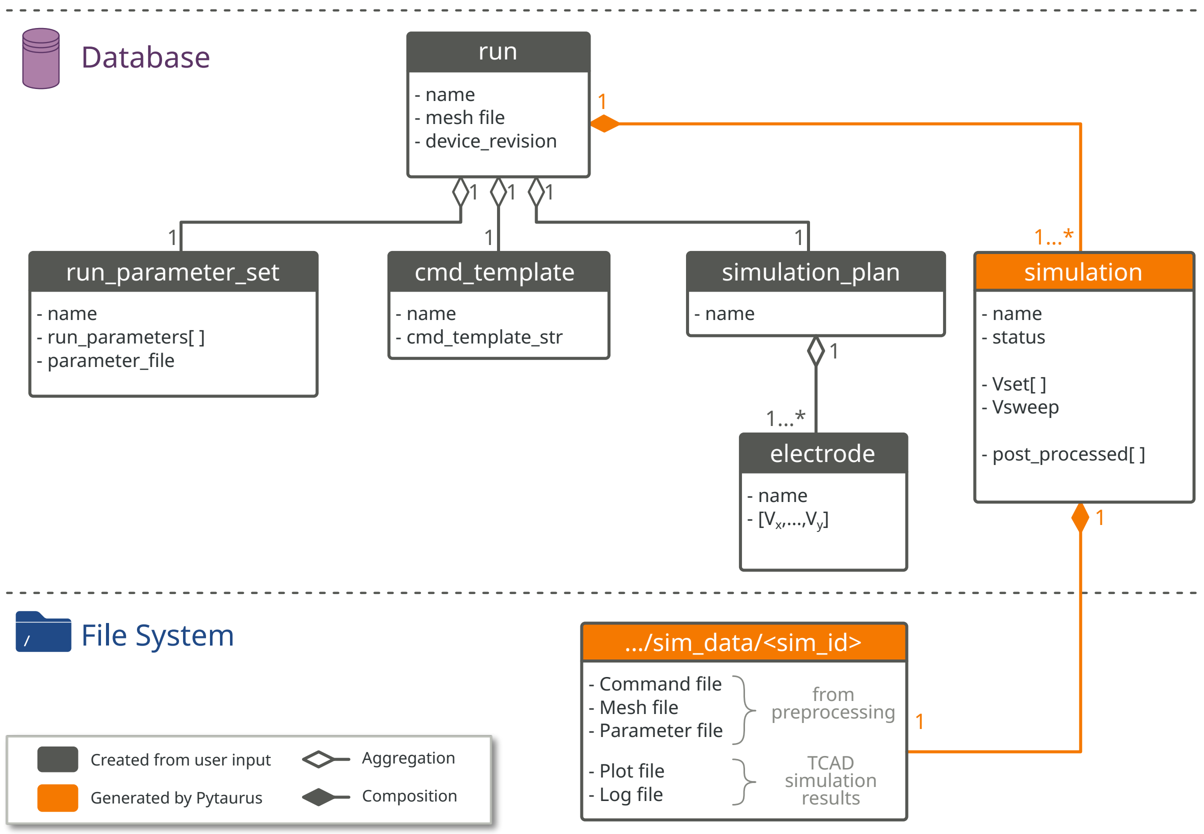


Figure 3.5: A *run* aggregates all data structures to generate a table model in a NoSQL database. The actual simulation data, e. g. input- and result files, are placed on the file system for performance. The proposed structure facilitates reuse of definitions and, most importantly, of existing simulation results in subsequent characterization runs.

PyTaurus Program- and Data-Structure

The PyTaurus host application interfaces with a Not only SQL (NoSQL) database, to store internal data structures and, after post-processing, the electric results of each TCAD simulation. A non-relational database architecture is preferred over a tabular architecture, because it allows expansion of all data objects ("documents") with additional fields at any time. A collection of a NoSQL database can therefore contain documents which differ in content without the need to propagate changes over an entire table, as would be required for relational databases.

The main entity of PyTaurus' internal data structure, shown in Figure 3.5, is named *run* and represents all data required to generate a complete table model in TCAD, respecting the requirements set out in Section 2.2.3. The data structure distinguishes between entities that are created from user input over the console user interface (UI) and structures that PyTaurus generates internally – the Sentaurus Device input files. The underlying principle of the program structure is to reuse user defined entities like *cmd_template* and *run_parameter_set* for multiple runs in order to minimize user interaction required for the creation of a new run with altered parameters. In fact, a new table model run can be created and simulated from similar and already partially existing dependencies without redefining any of the aggregated entities. In this chapter, all references to entities of the NoSQL database structure are emphasized with italic font-decoration, such as *run*.

Every TCAD simulation features the run parameter set of its respective run, shared by all generated simulations. Run parameters are typically device specific parameters, e. g. Schottky barrier height, material work functions, and solver specific parameters, e. g. minimal-/ maximal-step and step increment. The *run_parameter_set* document further includes the content of the parameter file which defines material parameters for Sentaurus Device simulation.

Each run aggregates a command template, which contains the string of a Sentaurus Device command file. In order to restrict adaptions of a Sentaurus Workbench originated Sentaurus Device command file to a minimum, PyTaurus supports the available @-parameter notation for run specific parameters stored in *run_parameter_set*. A second type of preprocessor parameters is supported with $-notation as shown in Listing 3.1. Unlike @-parameters, $-parameters are simulation specific parameters and distinguish simulations from each other within a table model run. $-parameters typically contain goals of set- and sweep-electrodes and their respective voltages, further explained in Section 2.2.3, and are therefore determined individually by PyTaurus for every generated simulation.

Listing 3.1: A command file for PyTaurus continues the Sentaurus TCAD @-notation for run specific paramters and introduces an additional $-notation for simulation specific-parameters.

```
...
Quasistationary(
    DoZero
    InitialStep = @initial_step@    Increment = @step_increment@
    Minstep =     @min_step@        MaxStep =   @max_step@

    Goal{ Name = "$pyt_electrode_sweep$" Voltage = $pyt_Velectrode_sweep_goal$}
        Plot {Range = (0 1) Intervals = 0}
...
```

The third user defined entity within a PyTaurus run is the simulation plan, which consists of the numeric table model parameters. The simulation plan aggregates electrodes, which defines a set of explicit voltage samples $V_x, ..., V_y$ for the dimension of the respective electrode within the biasing space of the table model. Voltage samples of different electrodes do not have to be correlated, nor are the sample sets required to have the same cardinality. The main benefit of this electrode-wise definition of bias samples is that granularity can be adapted to the circuit simulation target of the resulting table model. A table model for a MOSFET targeting a digital circuit simulation of an inverter can therefore be set up with a single bulk electrode voltage of $S_{e,Bulk} = \{V_B\} = \{V_S\} = \{0\,V\}$, if source potential is used as reference for the TCAD simulation. The remaining electrodes i can then independently feature larger sample sets $S_{e,i}$.

PyTaurus produces all Sentaurus Device simulations required for a complete table model. The first step is to generate each prospective Sentaurus Device simulation of a run as database document. Each *simulation* document contains mainly simulation parameters, status and, after successful TCAD simulation, the simulation results I and Q_e along with their respective tuple of input voltages V_e. In the further course of the run, described in Section 3.2.1, PyTaurus writes all files of a respective *simulation* to the file system, identifiable by the folder name matching the MongoDB object ID. The simulation files required to run the TCAD simulation are stored on the host file system to achieve high database performance at queries over a large document count and to obey the document size limit of MongoDB ($16\,MB$). In general, all simulation-related information, which is intended to be queried by PyTaurus, is placed into the *simulation* document, while large (device structure related results) and raw files (plot files, log files) are kept on the file system.

3.2.1 Setting up a Table Model Simulation Run

Setting up a table model simulation run starts with creating all user defined data structures and ends when preprocessed simulation data for every required TCAD simulation are available in the database and on the file system, as shown in Figure 3.5. User defined data structures are created through the console UI, which provides dedicated commands for each user defined entity. The command template can be read directly from the file system and the simulation plan along with the electrode voltage sets are given as console command arguments. Specification of run parameters from the run parameter set along with their respective values is uncomfortable with the console UI, as the planar RFET TCAD model already features 20 run parameters. Instead, as shown in the sequence diagram Figure 3.6, the run parameter set (RPS) can be extracted directly from an already provided command (CMD) template. After extraction of all run specific parameters, the obtained parameter list is written to a temporary Java Script Object Notation (JSON) file. The JSON format, an easy-to-read text file format, allows the user to edit all parameter values in the temporary file, before triggering PyTaurus console UI to load the manually provided parameters into the run parameter set database document.

After all user defined data structures shown in Figure 3.5 become available in the database, simulation setup is divided into three sequential steps: generate, plan and preprocess. For explanation of the internal functionality of PyTaurus, it is assumed that the considered DUT exhibits $N_e + 1$ electrodes $E_0, E_1, ..., E_{N_e}$. The respective voltage-step sets $S_{e,i} = \{V_1, V_2, V_3, ...\}$ for all E_i with $i \in \{1, 2, ..., N_e\}$ are aggregated in the simulation plan document. There is no voltage-step set $S_{e,0}$, as E_0 is the reference electrode with constant $V_{e,0} = 0\,V$. The sweep electrode is assumed to be E_{N_e}.

Generation Step

Generating run simulations means creating the *simulation* database documents shown in Figure 3.5. PyTaurus always generates simulations that consist of an initial phase and a sweep phase, as explained in Section 2.2.1. Each run can aggregate simulations to create N_e table models in total, each table model with a different electrode E_{sweep}. Typically, however, only one table model with a single sweep electrode E_{sweep} is desired, so the generation command receives the name of the desired sweep electrode E_{sweep} as parameter. A user can decide to further generate simulations for table models based on other sweep electrodes to a specified run at any time without producing duplicates or overwriting existing simulations. The feature of allowing a joint table model, which consists of sub-models with different sweep electrode is available and tested, but left out of scope of this work.

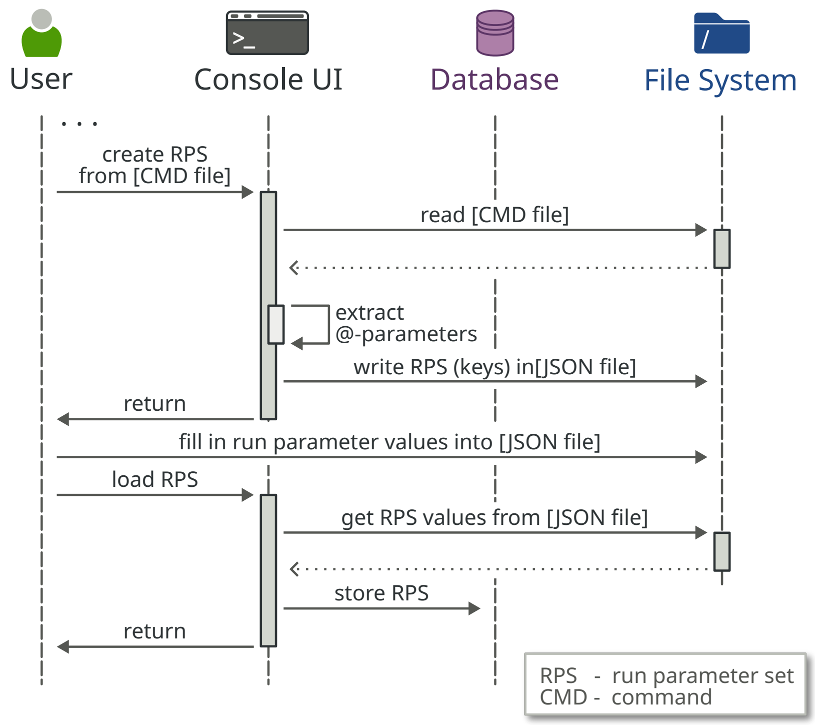


Figure 3.6: The setup of all user defined data structures involves the console UI, which writes the respective data to file system and database. Input of large data structures, such as simulation parameters, is optimized by supporting JSON files, which the user can edit.

The first step towards the simulation deck, described by Equation 2.2 in Section 2.2.1, is to generate a simulation document for each element in the Cartesian product of the all $i \in 1, ..., N_e$ electrode voltage sets $S_{e,i}$, resulting in a set of

$$S_{bias,min} = \prod_{i=1}^{N_e} S_{e,i} = S_{e,1} \times S_{e,2} \times ... \times S_{e,N_e}, \tag{3.2}$$

where $\times$ is the Cartesian product and set $S_{bias,min}$ has the cardinality

$$|S_{bias,min}| = \prod_{i=1}^{N_e} |S_{e,i}| \,. \tag{3.3}$$

In the generation step PyTaurus creates a separate TCAD simulation for each tuple in $S_{bias,min}$ as shown in Figure 3.7.

Each member of $S_{bias,min}$ is at this point available as a separate *simulation* document. However, not all of the generated *simulation* documents are required to be conducted in order to obtain a working table model: As described in Section 2.2.1 and illustrated in Figure 3.8, it is sufficient to conduct a parameter sweep for each input tuple

$$S_{table,min} = \prod_{i=1}^{N_e-1} S_{e,i}, \tag{3.4}$$

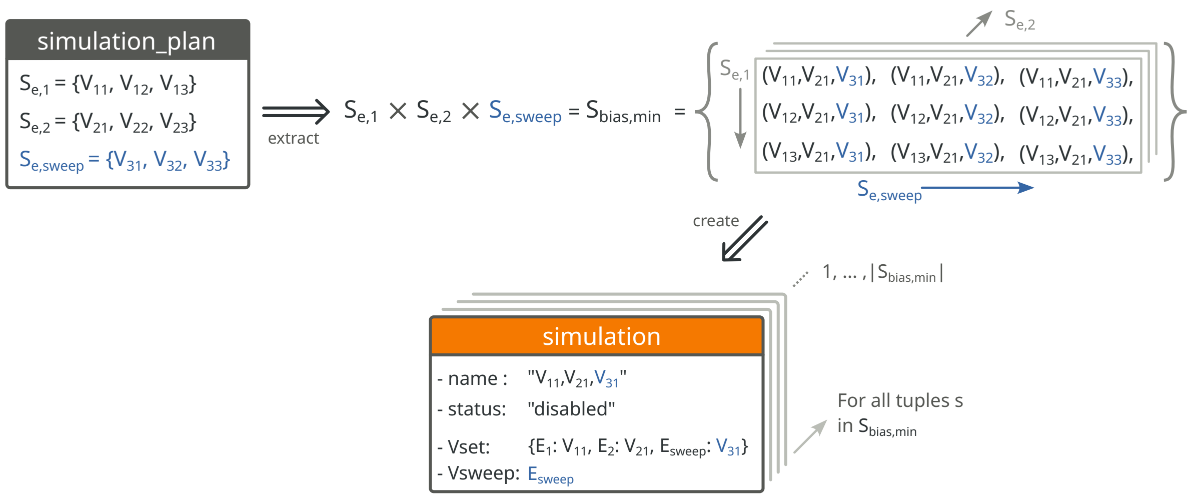

simulation_plan
S_e,1 = {V_11, V_12, V_13}
S_e,2 = {V_21, V_22, V_23}
S_e,sweep = {V_31, V_32, V_33}
extract
S_e,1 × S_e,2 × S_e,sweep = S_bias,min =
S_e,2
S_e,1
(V_11,V_21,V_31), (V_11,V_21,V_32), (V_11,V_21,V_33),
(V_12,V_21,V_31), (V_12,V_21,V_32), (V_12,V_21,V_33),
(V_13,V_21,V_31), (V_13,V_21,V_32), (V_13,V_21,V_33),
S_e,sweep
create
1, ... , |S_bias,min|
simulation
- name : "V_11,V_21,V_31"
- status: "disabled"
- Vset: {E_1: V_11, E_2: V_21, E_sweep: V_31}
- Vsweep: E_sweep
For all tuples s in S_bias,min

Figure 3.7: The simulation plan is translated to a set of tuples, consisting of the Cartesian product of the respective electrode sample sets $S_{e,i}$. For each tuple, a *simulation* document is set up.

and then drive the respective parameter sweep of V_{e,N_e} to $V_{goal} = \max(S_{e,N_e})$. The sweep dimension is then covered in the range of $V_{e,N_e} \in [0\,V, \max(S_{e,N_e})]$ and $|S_{e,N_e}| - 1$ simulations are avoided. The implemented algorithm allows easy adaptation of a data set, e. g. by adding voltage steps in every dimension, even after a table model run has completed. For that purpose, the user is allowed to change the sample sets $S_{e,i}$ in the database, e. g. by adding a voltage V_j to $S_{e,i}$ in entity *electrode*, and the re-execution of the generation step sets up the required additional simulations. It is therefore explicitly necessary in the generation step to iterate over all existing *simulation* documents of the respective run, instead of directly querying for $x_{N_e} = \max(S_{e,N_e})$. The simulations, which are not required for the table model are flagged accordingly.

Planning Step

Planning is the step to select which *simulation* documents will, in the course of the table model generation, be conducted as TCAD simulations. These simulations form the simulation deck $S_{bias,planned}$. These *simulation* documents are therefore edited to contain a status field `status : 'planned'`. The flags for already contained *simulation* and therefore avoided documents from the generation step are evaluated and each document flagged as such is left untouched. Documents that are not flagged, are staged for preprocessing by setting the according status to `status : 'planned'`, as shown in Figure 3.8.

Preprocessing Step

Preprocessing simulations means to interpret the *simulation* document and assemble the files which Sentaurus Device requires for simulation of the respective parameter sweep. For every planned simulation, PyTaurus creates a directory with the respective NoSQL database document identifier as name and writes parameter file (from *run parameter set*), command file (from *simulation*), and mesh file (from *run*), as shown in Figure 3.5. Each formerly planned

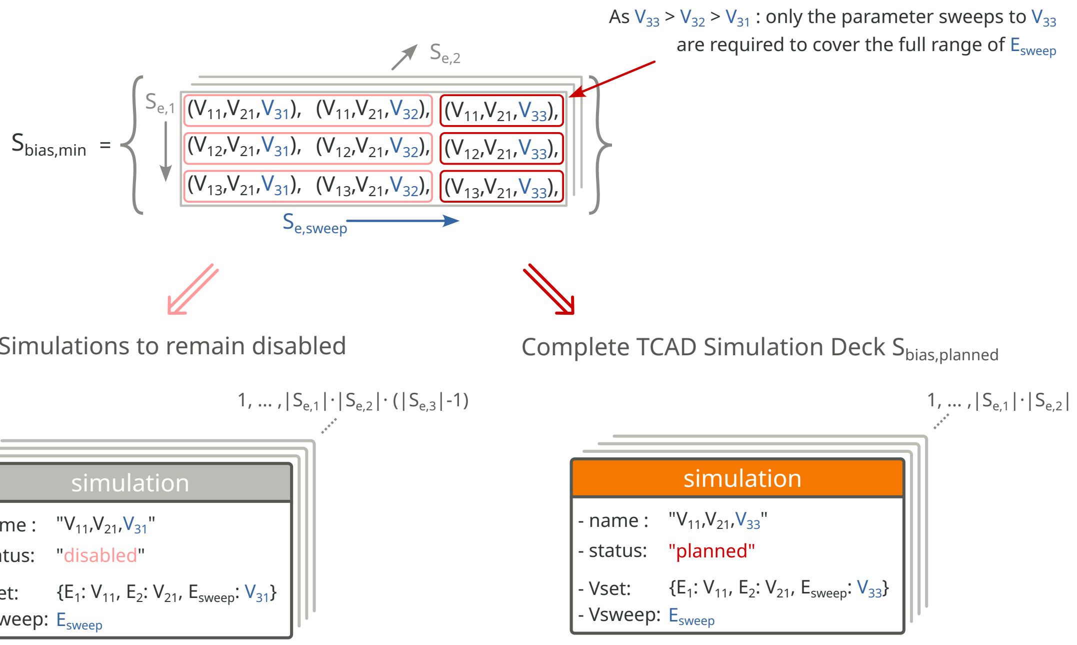

As $V_{33} > V_{32} > V_{31}$: only the parameter sweeps to V_{33} are required to cover the full range of E_{sweep}
$S_{e,2}$
$S_{bias,min}$ =
$S_{e,1}$
(V_{11},V_{21},V_{31}), (V_{11},V_{21},V_{32}), (V_{11},V_{21},V_{33}),
(V_{12},V_{21},V_{31}), (V_{12},V_{21},V_{32}), (V_{12},V_{21},V_{33}),
(V_{13},V_{21},V_{31}), (V_{13},V_{21},V_{32}), (V_{13},V_{21},V_{33}),
$S_{e,sweep}$
Simulations to remain disabled
Complete TCAD Simulation Deck $S_{bias,planned}$
$1, \ldots, |S_{e,1}| \cdot |S_{e,2}| \cdot (|S_{e,3}|-1)$
simulation
- name : "V_{11},V_{21},V_{31}"
- status: "disabled"
- Vset: $\{E_1: V_{11}, E_2: V_{21}, E_{sweep}: V_{31}\}$
- Vsweep: E_{sweep}
$1, \ldots, |S_{e,1}| \cdot |S_{e,2}|$
simulation
- name : "V_{11},V_{21},V_{33}"
- status: "planned"
- Vset: $\{E_1: V_{11}, E_2: V_{21}, E_{sweep}: V_{33}\}$
- Vsweep: E_{sweep}

Figure 3.8: Not all created *simulation* documents are required. Only those, which drive a parameter sweep to the maximum entry of the sweep dimension, are selected for the final simulation deck.

simulation document is finally annotated with `status : 'preprocessed'`. After the preprocessing step, the simulation run is set up and ready for execution.

3.2.2 From Start to Finish – Simulating the Run

PyTaurus provides a cluster computation interface, which distributes TCAD simulations to available hosts on the networks. The executing counterpart is the PyTaurus Runner, which accepts jobs, conducts the associated TCAD simulation and reports back the simulation result. The runner node requires access to the executable TCAD binaries.

PyTaurus Cluster Host

The host node provides a server, which accepts incoming WebSocket requests. Once the server is activated via the console UI, a specified simulation run can be started. The simulation run is started by enabling a scheduler, which distributes simulation jobs to runner nodes.

The scheduler iterates over available runner nodes and queries the runner status. When an idle runner is found, the scheduler fetches the next available preprocessed *simulation* document of the active run from the NoSQL database. The document is provided in form of a Binary Java Script Object Notation (BSON) structure, to which the scheduler adds all file contents of the corresponding file-system folder, as shown in Figure 3.9. This BSON structure, consisting of simulation file contents and *simulation* document is further called job object. The job object at this point contains the *simulation* document, created during the generation step, and the content of the simulation files, created during the preprocessing step. The job object is then sent to the respective idle runner. The underlying principle of data handling between host

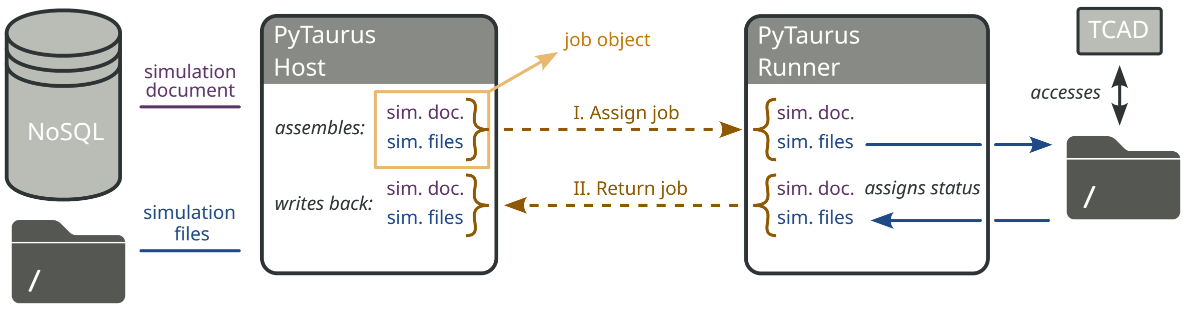


Figure 3.9: The data flow of simulation starts with PyTaurus host assembling the job object from the simulation files and the respective database *simulation* object. The job object is sent to a runner, which carries out the simulation. The complete or failed simulation job is sent back to the host and written back to database and file system.

and runner determines that the host sends the entire available data regarding a preprocessed simulation to the runner, and all data manipulation, e. g. appending of simulation results or setting the field *simulation.status*, is performed on the runner's side. A job object returned from the runner is therefore directly split into files and database document and written back to the respective destination.

PyTaurus Runner

The runner node connects to a PyTaurus Host via WebSocket and goes to superstate *Available*, as shown in Figure 3.10. During Available, the runner awaits status requests and simulation assignments. All status requests are contested with the runner state and the current central processing unit (CPU) load.

When a simulation job is assigned, the runner receives a job object containing simulation files and *simulation* document. The simulation files are written to a temporary folder and the TCAD simulation is started as a subprocess. There are three possible outcomes of a TCAD simulation:

1. The simulation finishes with a complete curve trace,
2. the simulation fails to converge, or
3. the simulation stalls.

Finished simulations are annotated with `status: 'done'` and failed simulations are annotated with `status: 'failed_returncode'` as soon as the subprocess ends.

Mitigation of stalling effects is important, as runners occupied with stalling simulations are blocked and cannot conduct further simulations. Detecting a stalling simulation, however, requires monitoring of the TCAD simulation. Concurrent with the TCAD simulation, a stall detection thread monitors the simulation progress, as shown in Figure 3.10. The stall detection mechanism works on a user defined stall detection window t_{stall}, which specifies the time in minutes in which the simulation is allowed to execute without progress. If the log file does not show progress within t_{stall}, the TCAD simulation is stopped and the respective *simulation*

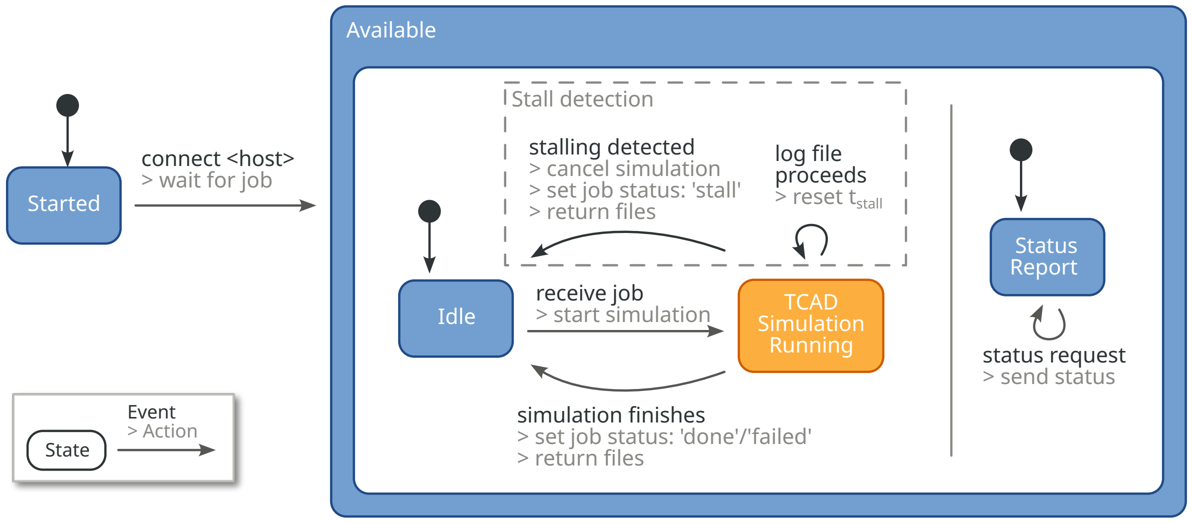


Figure 3.10: The runner conducts the simulation and monitors the progress. If a simulation finishes, fails or stalls, the job is returned with the respective files and annotation. Further, the runner responds to status requests by the host.

document is annotated with `status: 'failed_stall'`. PyTaurus can, however, not detect if the stalling simulation is in a non-recoverable deadlock or only takes exceptionally long for the further Newtonian iteration.

After simulation and subsequent annotation, the simulation files are stored in the job object and returned to the host along with the annotated *simulation* document. The runner then returns to substate *idle* and stands by for further job assignments.

During the scheduling table model simulation run, the share of completed simulation (status `'done'`) increases. These simulations can be post-processed at any time during the runtime of PyTaurus. In a post-processing step, PyTaurus iterates over all `'done'` simulations and reads the plot file of the respective simulation. From the plot files, all electrode voltages, -currents and -charges data points are extracted and stored into the respective *simulation* document. Post-processed simulations are annotated with `'post-processed'`. At this point, all individual samples generated within the parameter sweep of a simulation are stored within the NoSQL database.

3.2.3 Mitigating Convergence- and Stall Issues

The goal of PyTaurus is to generate a complete table model, which features all planned simulations, as described in Section 3.2.1 to run to completion. If simulations fail due to stalling or do not converge, the resulting table model misses data points which can make the table fail to represent the iso line criterion explained in Section 2.2.3.

Stalling simulations, i. e. simulations that do not progress within the stall window t_{stall}, are found by querying `status: 'failed_stall'`. Deadlocks cannot be told from slow executing simulations, as explained in Section 2.2.1, so the strategy for stall mitigation is to reschedule the respective simulations with a larger stall window to increase the chance of completion without causing congestion of the scheduler. After a table model simulation run finishes

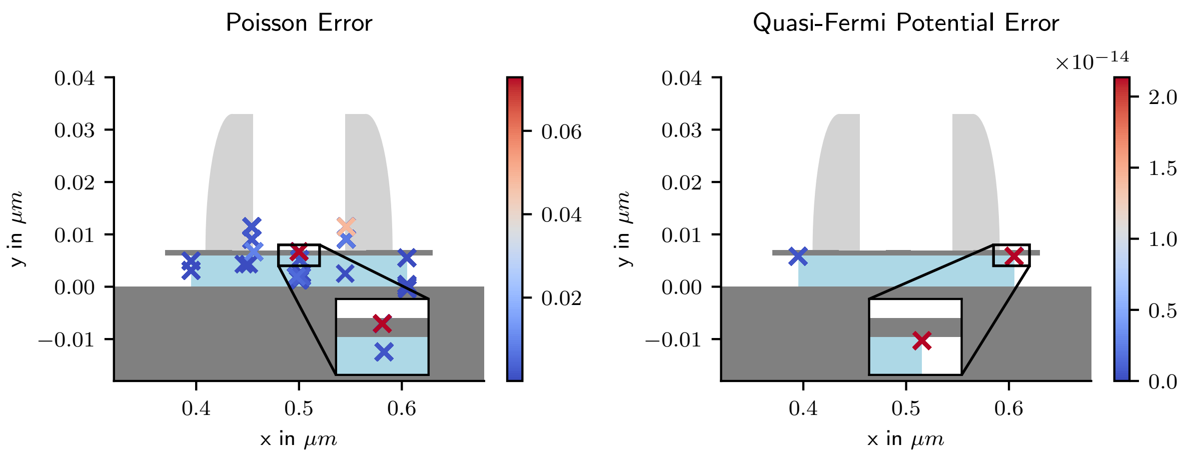


Figure 3.11: PyTaurus offers a meshing diagram, which visualizes the locations of all rejected solutions among the failed simulations, along with their respective magnitude for the different semiconductor equations. Based on the diagrams, the user can quickly make decisions about mesh refinements.

and all simulations are conducted, the scheduler increases t_{stall} by $60\,min$ and resets the simulation status of all queried simulations to `'preprocessed'`. This causes simulations, which are problematic with respect to stalling, to be scheduled again with a larger stall window after all regularly executing simulations are done. Congestion of runners is avoided and regularly executing simulations are preferred. The user can interrupt the table model simulation run with increased t_{stall} at any time when chances of completion are deemed to be too low to continue. In this situation, a user typically resorts to convergence aids.

Convergence of a Sentaurus Device simulation depends mostly on the meshing of the technology model, solver parameters and electrical boundary conditions. Simulations that show convergence issues typically require mesh refinement or increase of solver steps, which in turn increases computational effort. A practical approach to efficiently simulate a table model run is to constrain a mesh with a reasonable amount of vertices and the solver with a low number of iterations, with the intention of generating low computational effort for the vast majority of simulations. For refinement of the failed simulations, it is important to analyze the root cause of the convergence issues. As PyTaurus stores all simulation data, i. e. the preprocessed simulations and the simulation results, every individual simulation log is accessible. The PyTaurus tool suite includes a script based mesh analysis, shown in Figure 3.11, which evaluates the log files of all simulations and gathers all critical vertices. Position, equation type and respective error are plotted, so that the user can identify areas with large accumulated errors and make refinements, accordingly. The exemplary mesh analysis plot shown in Figure 3.11 exposes multiple critical vertices across body and spacer regions with a high error regarding the Poisson equation in the middle of the FG oxide. The quasi-fermi potential has two critical vertices at the corner of electrodes lat2/lat1, body area and FG oxide. This information supports the user in development of a refinement strategy for meshing and solver parameters.

For the remaining simulations, which fail to converge in the original run, a parameter variation strategy has to be found. As scheduling of simulations always considers a *run* entity with all available `'preprocessed'` simulations, the principle of parameter variation is to derive a new table model run, as shown in Figure 3.12. The derived run contains only the failed and stalled simulations from the original run. The separation between generation, planning and preprocessing step allows the user to derive a new run, e. g. *run_derived1*, modify one or multiple parameters and then set the derived run up for scheduling by performing the preprocessing step. Each parameter which is present in a user defined structure shown in Section 3.2 can be varied and is then considered during the preprocessing step for the derived run. As shown with the example in Figure 3.12, multiple derivations with variation of different parameters can be necessary until all simulations of the original run are completed.

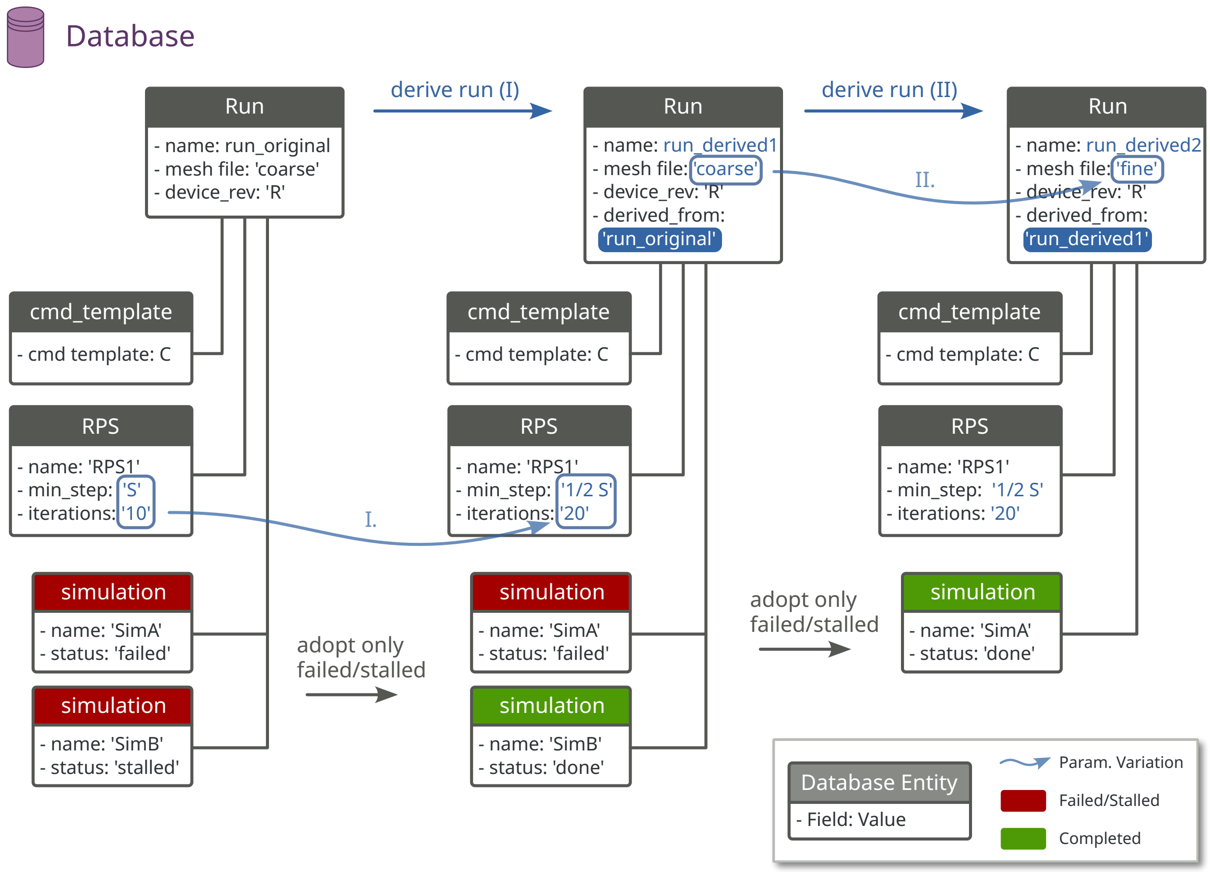


Figure 3.12: Refinement of simulations is important in technology simulation, as convergence issues are common. PyTaurus offers the simple refinement of all failed simulations of a respective run, by generating a derived run with one or multiple altered parameters. Refinement can be carried out for all parameters, including mesh file or numeric solver parameters.

Exporting the Table Model from the PyTaurus Database

After successful simulation and post-processing of a simulation run in PyTaurus, all data required for the table model, namely all nested parameter sweeps, are present in the NoSQL database. For the extraction flow of the table model data, a script based approach is preferred over the console UI, as it provides flexible processing and monitoring of the data extraction.

The first step of the extraction is to gather all simulations from a specified run and regarding a specific sweep electrode, which have not been flagged to be disabled during the generation step. This set of simulations contains all required simulations for the table model. The original run typically contains a share of uncompleted simulations, due to failing or stalling, as described in Section 3.2.3, which are therefore neither disabled nor post-processed. For each uncompleted, i. e. failed or stalled, simulation in the original run, a post-processed variant has to be found, contained in the scope of one of the derived runs. There is, however, no need to specify all derived runs manually. Any simulation which features the same boundary conditions and the same device model, specified by the field *run.device_revision*, serves as replacement regardless of the respective run it is aggregated to.

In case the set of incomplete simulations cannot be replaced entirely, linear grid interpolation is used to generate all data points in $S_{table,min}$ from Equation 3.2, as required by the `$table_model` function. The resulting data set is split up into lookup table files for I_{DC} and the electrode charges $Q_{e,n}$ with $n \in \{0, ..., N_e\}$, which are later referenced by `$table_model` .

Short Summary

In conclusion, PyTaurus is a tool for cluster simulation of factorial data models, which targets technology simulation. The focus is on easy setup, reuse and refinement of table model simulation decks, as typically the high number of simulations is difficult to manage manually. This section introduces the integral components, data structures and intended use of PyTaurus. In the course of this work, PyTaurus is used to generate a factorial table model for the planar RFET.

3.3 The Pseudo Transient Method

In technology simulation, true DC data can be created by conducting quasistationary simulations. In an initial step, the boundary conditions of the DUT are ramped from $0\,V$ to the specified electrode biases. Afterwards the parameter sweep is conducted, creating a set of true DC data points. In a system of parameter sweeps, such as a table model, ramp-ups to the starting conditions of the parameter sweep significantly contribute to the overall simulation time of the data set [206]. Compared to real measurements, the characterization time technology simulation is typically substantially larger than for technology simulation of an equivalent data set.

The goal of this section is to establish a simulation method, which mitigates the recurring ramp-ups of the conventional nested parameter sweep setups. Inspired by the inherent time affliction of all real world measurements, all data points are reached in a single Sentaurus Device simulation. The method is named pseudo transient because it represents the setup of a transient method, while aiming to suppress transient effects [206]. The means of suppressing transient effects comprise driving excitation signals over a large time base and a method for estimation and mitigation of remaining transients are presented in Section 3.3.2.

The basic principles of the pseudo transient method extend to an arbitrary number of electrodes N_e, therefore all mathematic formulations are generalized for electrodes $E_1, ..., E_{N_e}$. Visualization of the method, however, restricts to $N_e = 3$ for sake of clarity and readability.

Parts of this section were developed within the scope of the master's thesis by Dakyung Lee and published in [206] and [207].

3.3.1 Fundamental Functions to Generate the Bias Space

The boundary conditions of device electrodes within table models are formulated as N_e-tuples $V_e = (V_{e,1}, V_{e,2}, ..., V_{e,N_e})$ within the bias space $\mathbb{R}^{N_e}$, as explained in Section 2.2.3. In general, a good simulation setup covers the spatial structure of multidimensional table data efficiently, e. g. with high regularity and minimizing redundancies. Another important goal is to omit any simulation steps, which do not result in extractable information, such as the initial ramp up to specified boundary conditions for a parameter sweep. Still, stepping through a regular grid in multiple dimensions requires a form of nesting. The pseudo transient method is best introduced by portraying a real world measurement in time domain: In time domain and without concurrency, nesting of excitatory parameters is based on periodic, or at least non-monotonic functions, repeating each $V_{e,i}(t) \in S_{e,i}$, stepping through all boundary conditions V_e to conduct all targeted parameter sweeps for V_{e,N_e}.

Typically, electrical simulation tools, e. g. SPICE and Sentaurus Device, provide periodic excitation sources in form of pulse- and harmonic functions for transient simulation. These pulse sources allow configuration of rectangular, trapezoid and triangular functions and allow shaping through various parameters. In a first attempt to replicate the procedure of performing real measurements, a triangular function, similar to a series of parameter sweeps described in Section 3.2, comes into focus. Assuming that each voltage $V_{e,i}$ of electrode i covers a range of $V_{e,i} \in [-V_a, V_a]$, a suitable function for the most outer and therefore slowest changing, excitation function $V_{e,1}$ is a single ramp, as shown in Figure 3.13. The single ramp represents half a period of a triangle signal within a simulation time of $t \in [0, T]$. In general, the frequency of subsequent electrodes E_{i+1} for $i \in [1, ..., N-1]$ increases, in order to obtain a spatial distribution of all N_e-tuples $V_e(t)$ in $\mathbb{R}^{N_e}$. Figure 3.13 shows in a), b) and c) a basic setup of 3 electrodes with each E_{i+1} doubling the signal frequency of the previous electrode. In this particular case of doubling frequencies, the coverage of the bias space in $\mathbb{R}^3$, shown in d), is low. For practical implementations, it is useful to have the electrode voltage frequencies increase with integer multiples for ideal coverage of the bias space.

The triangle function provides linear ramps, ideal for equidistant stepping of a factorial simulation setup and similar to quasistationary simulation. When ramps are nested in form of triangle functions, however, turning points that occur at $n \cdot T_{p,triangle}/2,\ n \in \mathbb{N}^*$ cause high transient currents and reduce chance of convergence of the simulation. As convergence is crucial for a characterization method that relies on a single simulation, the ideal function is smooth and continuously differentiable in every point. This brings harmonic functions into focus. Figure 3.13 shows an implementation with triangle functions and cosine functions, respectively. Although the piece-wise linear triangle function promises higher regularity of the generated data set, the trade-off between both fundamental function goes in favor of

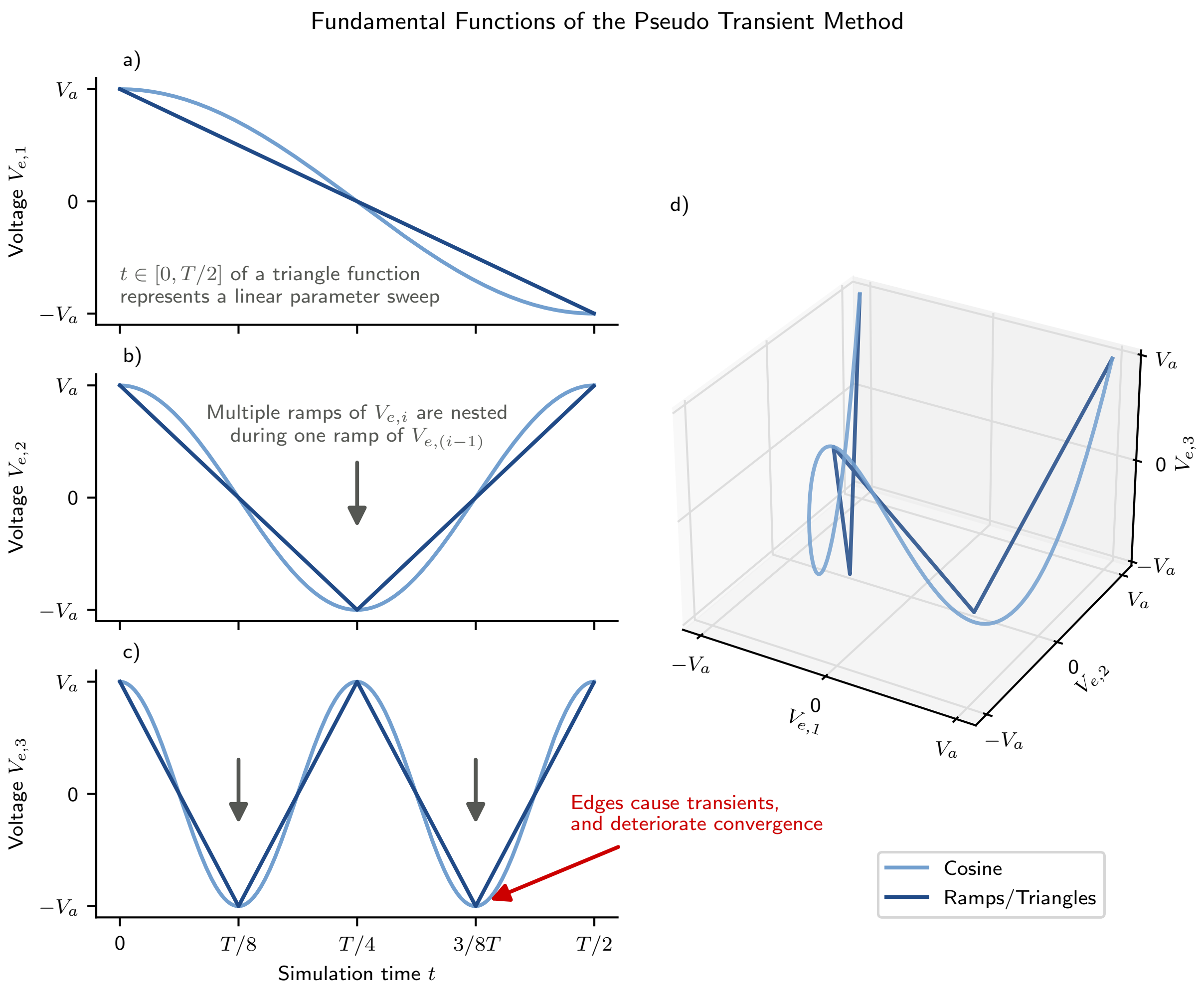


Figure 3.13: Triangle- and harmonic curves can be used as fundamental functions to generate a bias space in $\mathbb{R}^{N_e}$ by nesting their respective periods. In general, nested simulations or nested electrode voltages are a typical pattern in table model generation.

the harmonic function for allowing better convergence. The electrode voltages waveforms $V_{e,\{1,2,3\}}(t)$ are generated by

$$V_{e,i} = V_a \cdot \cos\left(2\pi \cdot k_i \cdot \frac{1}{T} \cdot t\right), \text{ with } k_1 = \frac{1}{2} \text{ and } k_{i+1} = 2 \cdot k_i. \tag{3.5}$$

The approach generalizes to an arbitrary number N_e of electrodes, where the electrodes $E_i \in \{E_1, ..., E_{N_e}\}$ are in order with the magnitude of their respective frequencies k_i/T with $T = const$. In general, the nesting factor between the frequency of an electrode function and the frequency of the subsequent electrode function can be expressed as

$$\frac{f_{e,i+1}}{f_{e,i}} = m_i, \text{ with } m_i \in \mathbb{N}^* \backslash \{1\} \tag{3.6}$$

The nesting factor m_i directly affects the granularity of the data set with respect to dimension i. If nesting is intended to be constant throughout the dimensions, a single nesting factor m can be used. In analogy to PyTaurus' *electrode* structure within a *simulation_plan* in Section 3.2, a pseudo transient simulation can therefore be conducted with an arbitrary granularity for each electrode voltage.

Nested cosine functions cover the full range of the bias space $V_{e,i} \in [-V_a, V_a]$ within the first half of the period, where the phase of the most outer electrode $\varphi_{e,1} = \omega_1 t \in [0, \pi]$. Still, an initial ramp-up to the boundary conditions $V_{e,i} = V_a$ at phase $\varphi_{e,1} = 0$ is required before time step $t = 0\,s$ of the pseudo transient simulation. Instead of ramping to the initial boundary conditions of the cosine function at $t = 0\,s$, a phase shift of $\pi/2$, effectively transforming the fundamental function to a sine function, naturally constrains a time step $t = 0\,s$ with $V_{e,i} = 0\,V$. The first interval is then a preamble before the table data range and the generated data set is ignored. Figure 3.14 shows that the most outer electrode E_1 within the nested configuration is excited with a falling edge between phase $\varphi_{e,1} = (\frac{\pi}{4}, 3\frac{\pi}{4})$. Within this falling edge of E_1, the simulated table data cover the full range of $V_{e,i} \in [-V_a, V_a]$, so the interval provides data for a complete table model. The most important argument for the use of sine functions, however, is that there are simulation environments where cosine functions, or in general a phase shifted sine function, are not available as excitation source. In Sentaurus Device, for example, the possibility to use a cosine function was added by introducing a phase shift parameter for the pulse source compact model in version S-2021.06. Previous versions do not allow phase shifts of the sinusoidal source. In this work, the pseudo transient method is therefore based on sine functions for broad applicability in characterization tasks and model building. The general form of the fundamental sine function as source for electrode E_i is therefore

$$V_{e,i}(t) = V_{a,i} \cdot \sin(2\pi \cdot f_{e,i} \cdot t). \tag{3.7}$$

The simulation principle can then be adapted individually to the features of the respective simulation environment.

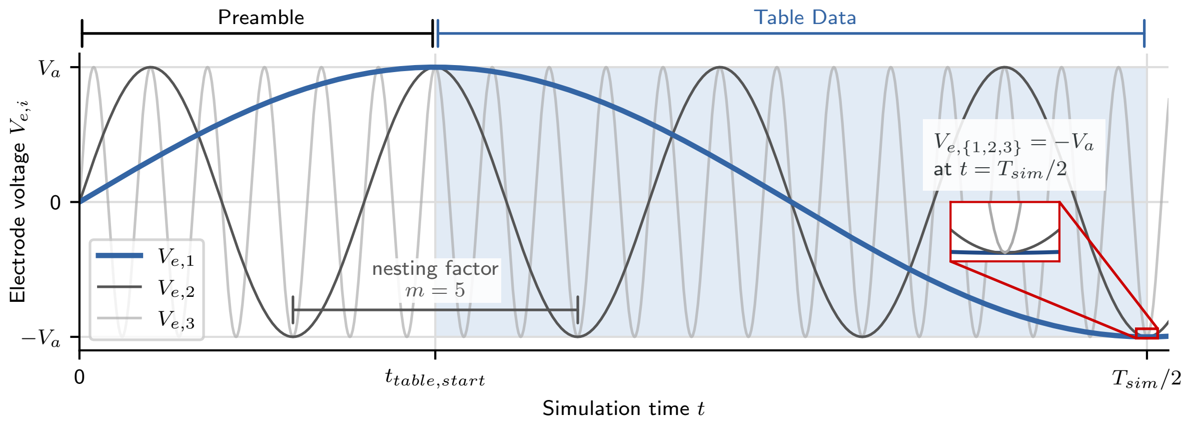


Figure 3.14: Using a sine, instead of cosine, as fundamental function avoids initial ramps, as all pulse sources feature $V_{e,i}(t = 0\,s) = 0\,V$. The shown pseudo transient setup features a nesting factor of $m = 5$. A table model data set can be sampled from $t \in [t_{table,start}, T_{sim}/2]$.

3.3.2 Estimating and Suppressing Remaining Transients

The pseudo transient simulation method is based on harmonic excitation functions, which simulation tools provide for time domain simulation. The resulting data are therefore afflicted with transient currents, depending on the respective time-related setup, i. e. excitation frequencies. Quasistationary simulation environments, on the other hand, typically do not provide excitation functions which are periodic with respect to a simulation variable, such as a sweep parameter. Time, as a parameter, is not available in quasistationary simulation and the resulting data set consists of stationary operating points. This section describes how to combine the benefits of both simulation types: the harmonic excitation functions in time domain are used to generate a nested data set, which is then transformed to approximately quasistationary data, as required for accurate table models. First, it is described how to set up simulations where the influence of capacitive elements is attenuated during simulation. In a second step, the remaining transient part of the respective current samples is estimated. Finally, a mathematical transformation for canceling out transient currents in linear RC networks is presented.

Before simulating, a decision has to be made which physical electrode is assigned to which position in the nesting chain of $E_{1,...,N_e}$. Capacitive currents increase with the frequency of the applied excitation, and the frequencies of the electrode sources $V_{e,1}$ to V_{e,N_e} increase as an inherent property of the harmonic nesting. Electrodes are ideally assigned $E_1, ..., E_{N_e}$ with the descending order of their estimated capacitance, in order to drive high capacitance electrodes with low frequencies and vice versa. For a conventional MOSFET, for example, it is reasonable to define the gate electrode as E_1, because it is a valid assumption that the gate features a higher capacitance with respect to the source electrode, than the drain electrode.

The quasistationary context restricts the impact on the characteristic current to the momentary values of the electrode voltages $V_{e,\{1,...,N_e\}}$ and disregards a steepness of the exciting source voltage, as time derivatives are not defined. Within a transient simulation, however, the terms that depend on time derivatives decrease in magnitude with increasing excitation signal periods $T_{p,e,\{1,...,N_e\}}$ and therefore T_{sim}. Terms with time derivatives, such as capacitive currents $I_c(t) = C\frac{dV(t)}{dt}$, are then attenuated when increasing T_{sim}. These, with the time base monotonically decreasing, transients eventually reach (numeric) noise floor level and therefore become negligible. If this condition holds true for every extracted data point, the simulated device characteristic can be considered quasistationary and the dataset can be used to produce an accurate table model.

Following the recommendations for electrode order and increasing simulation time helps to reduce transient currents, but the question arises if the transient component of one or even all obtained samples can be estimated. A performance metric for the pseudo transient method is aspired, which allows for assessment of the resulting data set.

Within a quasistationary context, a stateless device without hysteretic effects is expected to generate the same characteristic current for two points in time t_x and t'_x, where

$$V_{e,\{1,...,N_e\}}(t = t_x) = V_{e,\{1,...,N_e\}}(t = t'_x) \tag{3.8}$$

and

$$\frac{dV_{e,\{1,...,N_e\}}(t = t_x)}{dt} \neq \frac{dV_{e,\{1,...,N_e\}}(t = t'_1)}{dt}, \tag{3.9}$$

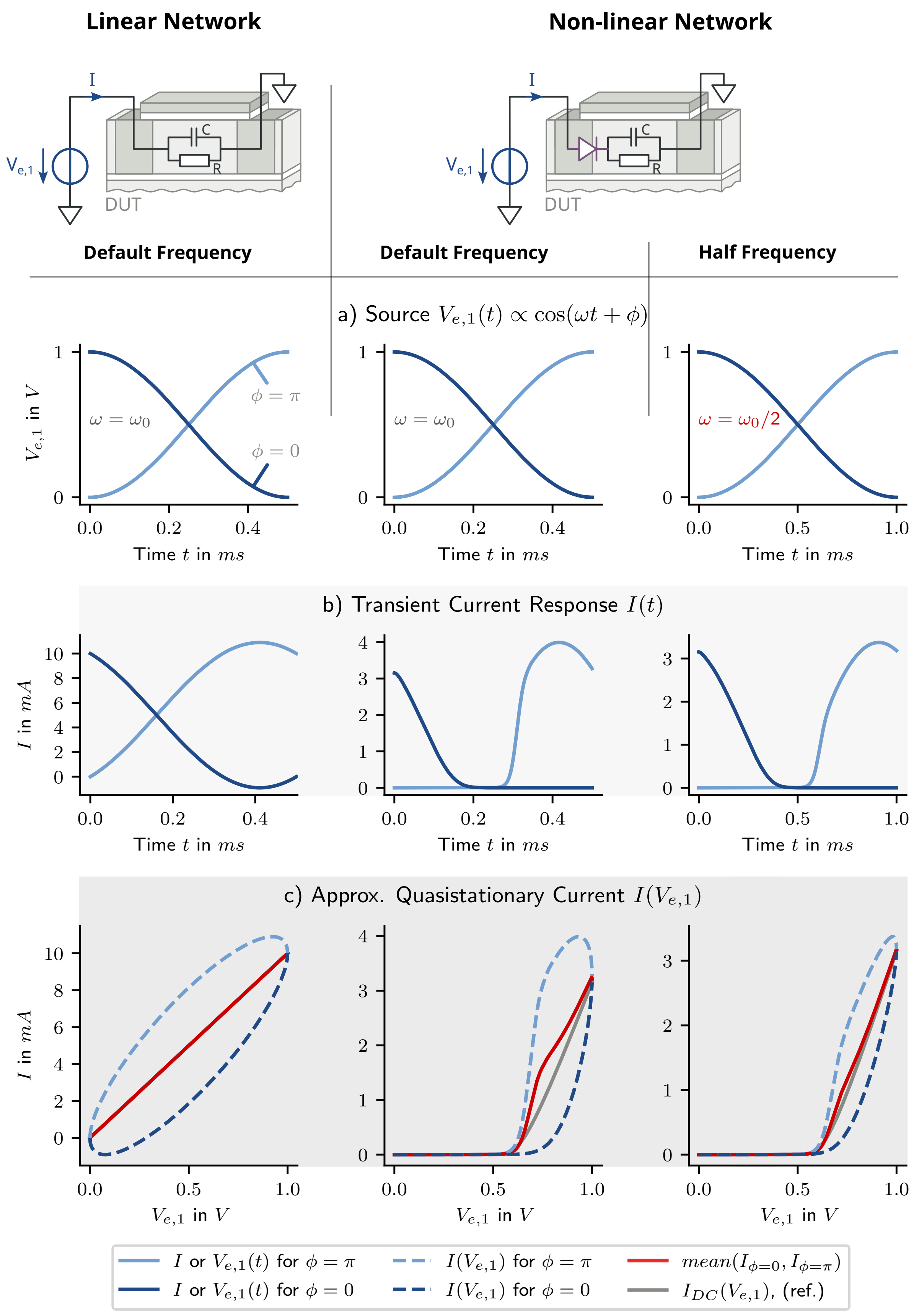


Figure 3.15: In linear networks the mean current that is excited by two harmonic voltage sources with $\phi = 0$ and $\phi = 0$ is the exact DC current. In non-linear networks, the DC current is only approximated.

as time derivatives are neglected in general. In turn, a transient simulation setup, where the structure of the data set allows to cancel out the terms that depend on time derivatives, is expected to result in a model without transient components and therefore equally accurate as a quasistationary table model approach. A strategy for quasistationary verification of a sample at t_x in pseudo transient simulation is therefore to provide a second sample with electrode voltages V'_e at time t_x with $V'_e(t_x) = V_e(t_x)$, but respectively opposite time related slopes

$$\frac{dV'_{e,i}(t = t_x)}{dt} = -\frac{dV_{e,i}(t = t_x)}{dt}, \tag{3.10}$$

for every $i \in \{1, 2, ..., N_e\}$. The difference in the resulting current- and charge values which stem from $V_e(t_x)$ and $V'_e(t_x)$ can then be used for estimation of transient effects.

The principle behind this transient suppression strategy can be demonstrated when reducing the simulation setup to a single harmonic source which drives characteristic current through a DUT, as shown in Figure 3.15. All described simulations are performed in LTSpice by Analog Devices [208]. The DUT model features either a linear RC network, or includes non-linear effects. In both cases two transient simulations are conducted, with

$$V_{e,1}(t) = 0.5V + 1\,V \cdot \cos(\omega t + \phi) \tag{3.11}$$

as sinusoidal source, including a phase shift ϕ, which is to be defined. The parameters $R = 100\,\Omega$, $C = 2\,nF$ and $\omega_0 = 2\pi \cdot 2\,kHz$ are selected for demonstration, to ensure that the resulting current $I(t)$ shows a significant transient component. Both DUTs are simulated for half a period of the cosine source ($t \in [0\,ms, 0.5\,ms]$) with $\phi = 0$, which results in a falling edge for $V_{e,1,\phi=0}$ in Figure 3.15 a). A second simulation is performed in Figure 3.15 a), where $\phi = \pi$ and therefore the voltage source $V_{e,1,\phi=\pi}$ drives a rising edge. In case of the non-linear DUT model, the same simulations are conducted once more for $\omega = \omega_0/2$. The current response $I_{\phi=0}(t)$ and $I_{\phi=\pi}(t)$ in Figure 3.15 b) both expose transfer characteristics of the respective DUT, but the resulting data sets differ due to transient currents caused by the capacitive element C. When ignoring the time annotation and expressing the DUT current as a function of the input voltage $I(V_{e,1})$, as done in Figure 3.15 c), it becomes evident that depending on the slope of the voltage source within the transient information, the reference current $I_{DC}(V_{e,1})$ is either over- or underestimated. For the linear DUT, the arithmetic mean

$$I_{mean}(V_{e,1}) = \text{mean}(I_{\phi=0}(V_{e,1}), I_{\phi=\pi}(V_{e,1})) = I_{DC}(V_{e,1}) \tag{3.12}$$

replicates the exact DC current I_{DC}, that is obtained by a true DC analysis. The non-linear DUT similarly shows over- and underestimation of the true reference current, depending on the input slope. However, it is observed in Figure 3.15 c) that the true DC current is not recovered by calculating the arithmetic mean. Unlike for the linear DUT, the arithmetic mean deviates from the reference for $\omega = \omega_0$ with a non-linear DUT, but it is noteworthy that the error is still significantly lower than for $I_{\phi=0}(V_{e,1})$ or $I_{\phi=\pi}(V_{e,1})$. The remaining option to further increase correlation between $I_{mean}(V_{e,1})$ and $I_{DC}(V_{e,1})$ at this point is to reduce the source frequency ω to $\omega = \omega_0/2$. As a consequence, transient currents are further suppressed. $I_{\phi=0}(V_{e,1})$, $I_{\phi=\pi}(V_{e,1})$ and especially $I_{mean}(V_{e,1})$ then show higher correlation with $I_{DC}(V_{e,1})$ and the accuracy of the current samples increases. The example of Figure 3.15 focuses on

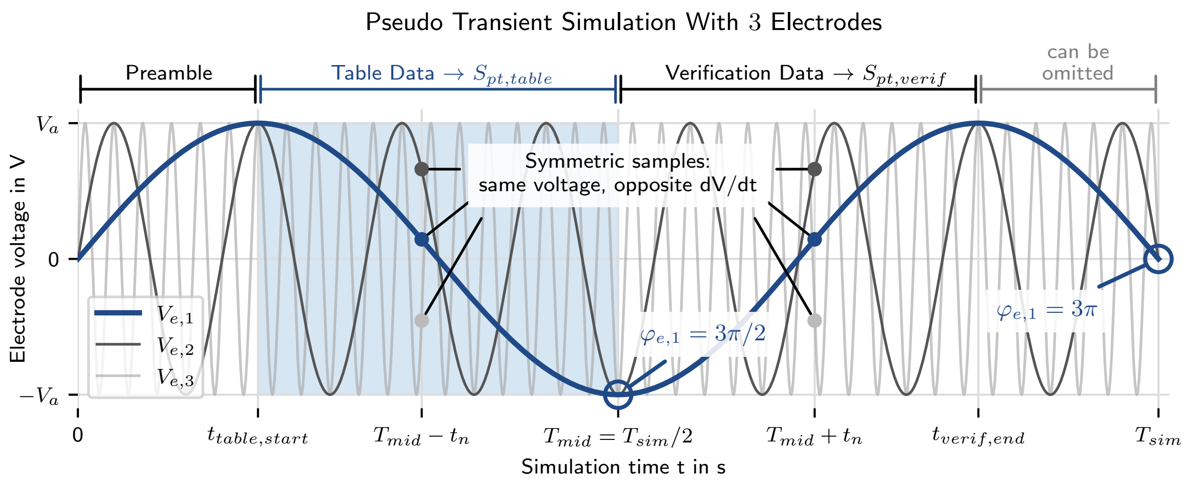


Figure 3.16: The voltage sources $V_{e,i}$ of a pseudo transient simulation of $N = 3$ electrodes are driven with a nesting factor $m = 5$ and amplitudes $V_{a,i} = V_a$ for all electrodes E_i, $i \in [1, 2, 3]$. The pseudo transient simulation can be constrained to create symmetry axis at $t = T_{sim}/2$, which provides two symmetrical intervals with same biases but opposite time-related slope in order to estimate the transient component of the current.

transient current, but the principles extend to electrode charges, due to the linearity of the integral operation $Q_{e,i} = \int_{t_1}^{t_2} I_{e,i}(t)dt$, with two arbitrary points in time t_1 and t_2, electrode charge $Q_{e,i}$ and electrode current $I_{e,i}$.

In the next step, a pseudo transient simulation is constructed, which fulfills the requirement of providing for each sample $I(V_e)$ a second sample $I_{ref}(V_e)$ with the same electrode voltage tuple V_e but opposite slope, and therefore obeying Equation 3.8, Equation 3.9 and Equation 3.10. One solution can be to drive a second simulation, similar to the two opposite cosine edges simulated in Figure 3.15. A disadvantage is that an initial phase parameter has to be supported for sinusoidal sources, possibly limiting the application scope. A more elegant solution comes from the periodic nature of the sine function. While the interval $[t_{table,start}, T_{sim}/2]$, as shown in Figure 3.14, covers a full table data set, the subsequent interval $[T_{sim}/2, t_{verif,start}]$ replicates this set but provides the opposite slope for each electrode voltage sample in the respective mirror sample. Figure 3.16 shows that $T_{mid} = T_{sim}/2$ represents a mirror axis, enabled by the constraint

$$V_{e,i}(t = T_{mid}) \quad = \quad -V_{a,i}, \tag{3.13}$$

which provides symmetry between two points

$$V_{e,i}(T_{mid} - t_n) \quad = \quad V_{e,i}(T_{mid} + t_n), \tag{3.14}$$

$$\frac{d}{dt}V_{e,i}(t)\bigg|_{t=T_{mid}-t_n} \quad = \quad -\frac{d}{dt}V_{e,i}(t)\bigg|_{t=T_{mid}+t_n}, \tag{3.15}$$

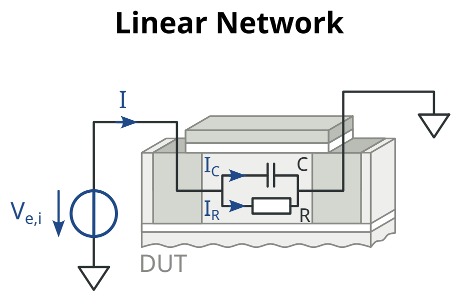


Figure 3.17: The compensation of transient effects is shown with a parallel circuit of R and C.

with $t_n \in (0, t_{table,start}]$. The frequency constraints for all frequencies $f_{e,i}$ of all electrodes E_i can then be formulated as

$$V_{a,i} \cdot \sin(2\pi f_{e,i} \cdot T_{mid}) = -V_{a,i}. \tag{3.16}$$
$$\tag{3.17}$$

For $V_{a,i} \neq 0$, this equation is fulfilled if and only if the following condition is true

$$2\pi f_{e,i} \cdot T_{mid} = \frac{4k-1}{2}\pi, \tag{3.18}$$
$$\tag{3.19}$$

and therefore

$$\Rightarrow f_{e,i} = \frac{1}{T_{mid}} \cdot (k - \frac{1}{4}), \tag{3.20}$$

with $k \in \mathbb{N}^*$.

Having constrained the electrode waveforms $V_{e,i}(t)$, the cancellation of transient effects in linear circuits by taking the arithmetic mean between opposing slope samples can be shown. Due to superposition in linear systems, it is sufficient to show this property for a simple equivalent circuit with a single voltage source. For the parallel RC equivalent circuit illustrated in Figure 3.17, the current $I(t)$ is driven by the pseudo transient voltage source from Equation 3.7 with the frequency constraint of Equation 3.20, resulting in

$$V_{e,i}(t) = V_{a,i} \cdot \sin\left(\overbrace{2\pi \cdot \frac{1}{T_{mid}} \cdot (k - \frac{1}{4})}^{=:\omega_i} \cdot t \right) = V_{a,i} \cdot \sin(\omega_i \cdot t). \tag{3.21}$$

The total current in the time domain, with $I(t)$, $I_C(t)$ and $I_R(t)$ defined in Figure 3.17, can be formulated as

$$I(t) = I_C(t) + I_R(t) \tag{3.22}$$
$$= C\frac{dV_{e,i}(t)}{dt} + \frac{V_{e,i}(t)}{R}, \tag{3.23}$$

and the arithmetic mean between two symmetric points $T_{mid} - t_n$ and $T_{mid} + t_n$ in discrete time has to be shown to be equal to the resistive current for all $t_n \in \mathbb{R}^+$, following

$$I_{mean}(t_n) = \frac{I(T_{mid} - t_n) + I(T_{mid} + t_n)}{2} \overset{!}{=} I_R(T_{mid} \pm t_n). \tag{3.24}$$

The explicit current equation can then be formulated as

$$\begin{aligned}\frac{V_{e,i}(T_{mid}\pm t_n)}{R} &\overset{!}{=} \frac{C\frac{dV_{e,i}(t)}{dt}\Big|_{t=T_{mid}-t_n} + \frac{V_{e,i}(T_{mid}-t_n)}{R}}{2}\\ &+ \frac{C\frac{dV_{e,i}(t)}{dt}\Big|_{t=T_{mid}+t_n} + \frac{V_{e,i}(T_{mid}+t_n)}{R}}{2}. \end{aligned} \tag{3.25}$$

Here, the equality of resistive currents $I_R(T_{mid} \pm t_n)$ is assumed due to symmetry of $V_{e,i}$ with respect to T_{mid}, as constrained in Equation 3.14 and Equation 3.24, and the general time invariance of the resistive impedance R. Equation 3.25 therefore further reduces to

$$\frac{V_{e,i}(T_{mid}\pm t_n)}{R} = \frac{C}{2}\frac{dV_{e,i}(t)}{dt}\Big|_{t=T_{mid}-t_n} + \frac{C}{2}\frac{dV_{e,i}(t)}{dt}\Big|_{t=T_{mid}+t_n} + \frac{V_{e,i}(T_{mid}\pm t_n)}{R} \tag{3.26}$$

$$0 = \frac{dV_{e,i}(t)}{dt}\Big|_{t=T_{mid}-t_n} + \frac{dV_{e,i}(t)}{dt}\Big|_{t=T_{mid}+t_n}. \tag{3.27}$$

Deriving the source voltage $V_{e,i}$ with respect to time results in

$$\frac{d}{dt}V_{e,i} = V_{a,i}\cdot\omega_i\cdot\cos(\omega_i t), \tag{3.28}$$

to be inserted into Equation 3.27 for

$$\frac{dV_{e,i}(t)}{dt}\Big|_{t=T_{mid}-t_n} = -\frac{dV_{e,i}(t)}{dt}\Big|_{t=T_{mid}+t_n}, \tag{3.29}$$

$$V_{a,i}\cdot\omega_i\cdot\cos(\omega_i(T_{mid}-t_n)) = -V_{a,i}\cdot\omega_i\cdot\cos(\omega_i(T_{mid}+t_n)). \tag{3.30}$$

In the further course of the transformation, the cosine angle addition theorem for $\cos(\alpha\pm\beta)$ is applied to obtain

$$\begin{aligned}&\overbrace{\cos(\omega_i T_{mid})}^{\cos(2\pi\cdot(k-\frac{1}{4}))=0}\cos(\omega_i t_n) + \overbrace{\sin(\omega_i T_{mid})}^{\sin(2\pi\cdot(k-\frac{1}{4}))=-1}\sin(\omega_i t_n)\\ =-&\underbrace{\cos(\omega_i T_{mid})}_{\cos(2\pi\cdot(k-\frac{1}{4}))=0}\cos(\omega_i t_n) - \underbrace{\sin(\omega_i T_{mid})}_{\sin(2\pi\cdot(k-\frac{1}{4}))=-1}\sin(\omega_i t_n).\end{aligned} \tag{3.31}$$

For $k \in \mathbb{N}^*$ the constant harmonic functions are evaluated at T_{mid}, which results in 0 for cosine and -1 for sine and leads to the tautology

$$\sin(\omega_i t_n) = \sin(\omega_i t_n), \tag{3.32}$$

which holds true for all $t_n \in \mathbb{R}^+$.

FVM simulation, however, operates on discrete time steps, returning time-descrete samples. With transient simulations, the placement of time steps typically depends on multiple factors during simulation and some bias situations require shorter time steps than others to converge. In order to obtain sample pairs, which allow evaluation of Equation 3.24, the sampling points are required to be distributed with mirror axis T_{mid}, likewise. In Sentaurus Device the CurrentPlot feature explained in Section 2.2.1 can be used to define the number of plot intervals and therefore enforce a specific sampling rate. If regular sampling or in general sampling with symmetry axis T_{mid} is unavailable in the respective simulation environment, resampling to a custom grid is a viable option as a fall back strategy to still obtain a transient current estimation. The pseudo transient sampling strategy is illustrated in Section 3.3.3.

For a linear approximation of the DUT, the arithmetic mean is shown to cancel out transient currents for continuous time. For DUT of non-linear nature the transients do not cancel out, yet the difference in current of two sample pairs

$$\Delta I(t_n) = I(T_{mid} - t_n) - I(T_{mid} + t_n) \tag{3.33}$$

is proportional to the transient components of the DUT current and can therefore be used as performance metric. A current mismatch of $\Delta I(t_n) < I_{noise}$ is the goal of transient suppression and leaves little room for improvement. With the availability of an estimator for transient currents, mitigation strategies beyond increasing T_{sim} and electrode order optimization can be designed.

3.3.3 Sampling Strategy

Goal of the pseudo transient method is to provide one or more data sets from which data tuples can be extracted to construct a table model. The data set to be extracted typically comes from the table data region, which is denoted by the time interval $[t_{table,start}, T_{mid}]$ as shown in Figure 3.16. In this segment, the set of electrode voltages $V_e(t)$ with $t \in [t_{table,start}, T_{mid}]$ is further denoted as $S_{pt,table}$, while the data set that stems from the interval $[t_{mid}, T_{verif,end}]$ is named $S_{pt,verif}$.

A fundamental requirement comes from the transient estimation and cancellation approach described in Section 3.3.2. The sample distribution is required to be symmetrical to T_{mid}, in order to allow comparison between data points in $S_{pt,table}$ and $S_{pt,verif}$. Uniform sampling with

$$f_s = \frac{a}{T_{sim}}, \tag{3.34}$$

and $a \in \mathbb{N}^* \backslash \{1\}$, which includes the points $t = 0\,s$, T_{mid} and T_{sim}, fulfills this criterion.

Verilog-A's `$table_model` function requires the data to be structures as parameter sweeps over iso lines, as described in Section 2.2.3. However, unlike with the parameter sweep based approach of PyTaurus in Section 3.2, the condition to obtain at least two samples of V_{e,N_e} for each $(V_{e,1}, V_{e,2}, ..., V_{e,(N_e-1)})$ is not fulfilled for the N_e electrodes $E_{\{1,...,N_e\}}$ sorted for ascending $f_{e,i}$. The strict monotonicity of $V_{e,1}$ within $S_{pt,table}$ and $S_{pt,verif}$ prevents the existence of iso lines with respect to $V_{e,\{1,...,N_e-1\}}$, as each sample features a unique voltage $V_{e,1}$. One way of solving this issue is to interpolate and resample the data set to a regular grid with respect to $V_{e,\{1,...,N_e\}}$, where E_{N_e} can then be interpreted as sweep variable. However, this interpolation and resampling introduces errors, which depend in magnitude on the targeted interpolation grid. In search for a more elegant approach, a way of creating the required iso lines, in resemblance of a parameter sweep, is developed. $V_{e,1}$ is the only (strictly) monotonic function within the respective data sets and is therefore selected as independent sweep voltage. The electrode voltages $V_{e,i}$, with $i \in [2, N_e]$, are then used to construct the iso lines, as they sample the same set of voltages $S_{e,i}$ in every period of the respective $V_{e,i}$. For demonstration, Figure 3.18

shows a period of a voltage $V_{e,i}$ with their discrete samples. The iso line criterion implies that the voltage samples $S_{e,i}$ of $V_{e,i}(t)$ have to be equal within every period of $V_{e,i}(t)$. As every dimension $V_{e,\{2,\dots,N_e\}}$ provides then a fixed set of $S_{e,i}$ which holds true for every period of the pseudo transient simulation, the set of iso lines is the Cartesian product

$$S_{iso\_lines} = \prod_{i=2}^{N_e} S_{e,i} = S_{e,2} \times S_{e,3} \times \; ... \; \times S_{e,N_e}. \tag{3.35}$$

The sample rate f_s has to be constrained with the targeted value range $[-V_{a,i}, V_{a,i}]$ of each electrode $V_{e,i}$ with $i \in [2, N_e]$ in mind. A minimum requirement for f_s is therefore to place at least one sample in every peak of $V_{e,i}$. Along with the samples in the turning points of the harmonic functions, the minimum sampling frequency can be formulated as

$$f_{s,min} = 4 \cdot f_{e,N_e}. \tag{3.36}$$

To increase the precision of the obtained data set with respect to the dimension of E_{N_e}, the sampling frequency f_s can be further increased by introducing an oversampling factor $N_{os} \in \mathbb{N}^*$ for

$$f_s = N_{os} \cdot 4 \cdot f_{e,N_e}. \tag{3.37}$$

The number of uniform sample intervals to be enforced during simulation by Sentaurus Device CurrentPlot feature is therefore calculated as

$$N_{intervals} = f_{e,N_e} \cdot T_{sim} \cdot N_{os}. \tag{3.38}$$

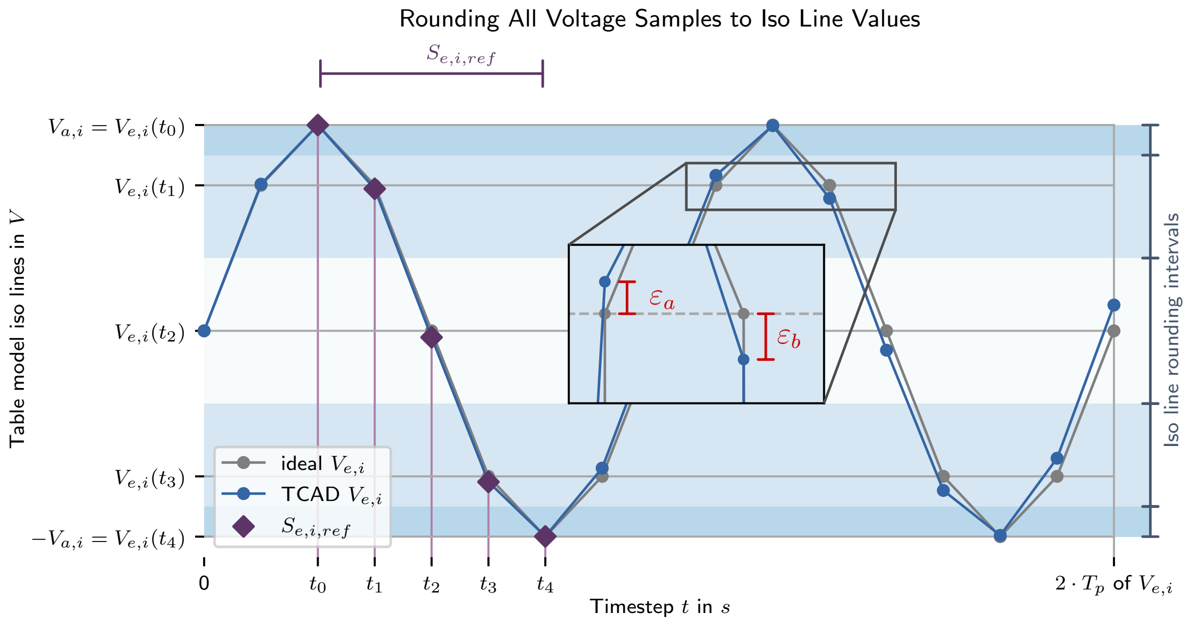


Figure 3.18: Limited numeric precision of functions and constants leads to deviations with respect to the periods $T_{p,i}$ and therefore the voltages $V_{e,i}$ at the respective samples throughout the simulation. As the voltage samples form iso lines of the table model, rounding is required.

Figure 3.18 shows an oversampling of $N_{os} = 2$, which results in 8 samples per period of $V_{e,i}$ and 5 samples per rising or falling edge when $i = N_e$ is assumed. Due to symmetry between rising and falling edge of the harmonic functions $V_{e,i}(t)$, the voltage samples of rising and falling edge are equal. The resulting sample set $S_{e,i}$ is therefore restricted to the voltage samples of a single edge, demonstrated as $t_0, t_1, ..., t_k$, where

$$k = \frac{4 \cdot N_{os}}{2} + 1 = 2 \cdot N_{os} + 1. \tag{3.39}$$

The sampling of all other edges of $V_{e,i}(t)$ during the pseudo transient simulation results in the same voltage samples $V_{e,i}(t_0), V_{e,i}(t_1), ..., V_{e,i}(t_4)$. Having specified all parameters of the pseudo transient voltage sources along with the sampling constraints, the total number of samples of a respective pseudo transient simulation constrained by m, N_{os} and N_e is

$$\begin{aligned} |\mathcal{D}_{pstrans,total}| &= f_{e,N_e-1} \cdot T_{sim} \cdot 4 \cdot N_{os} \\ &= f_{e,1} \cdot m^{N_e-1} \cdot \frac{1.5}{f_{e,1}} \cdot 4 \cdot N_{os} \\ &= 6 \cdot m^{N_e-1} \cdot N_{os}, \end{aligned} \tag{3.40}$$

which includes the first sample at $t = 0\,s$. However, the actual table model size is only

$$|\mathcal{D}_{pstrans}| = 1/3 \cdot |\mathcal{D}_{pstrans,total}|\,, \tag{3.41}$$

because the extracted data come only from the arithmetic mean of table data range and verification data range, as shown in Figure 3.16.

With analytic expressions, the period of a harmonic functions such as $V_{e,\{1,...,N_e\}}$ can be constrained exactly. The sample frequency f_s can be set to obtain respective sets $S_{e,i}$ which are exactly equal for every period. In simulation, however, finite precision of operations, such as $\cos(x)$, and numeric constants, such as π, leads to deviations from the proposed analytic expressions. The periods $T_{p,i}$ of the respective electrode voltages lack precision and as a consequence, the harmonic voltage sources exhibit drift in time domain. As shown in Figure 3.18, this leads to unique sample sets $S_{e,i}$ in every period of $V_{e,i}$ with $i > 1$, drifting further with progress of time t. The rounding errors ε are typically small compared to source voltages in the simulation, rendering ε_a and ε_b negligible. But the iso line criterion requires numeric exactness between the voltage samples in $S_{e,i}$ of every $V_{e,i}(t)$ in every period. In order to emulate sampling of $V_{e,i}(t)$ at the numerically equal $S_{e,i}$ throughout the whole pseudo transient simulation, a reference edge is selected and the resulting set $S_{e,i,ref}$ is denoted as a reference set. Figure 3.18 shows this reference set extracted from $t \in [\frac{1}{4}T_{p,i}, \frac{3}{4}T_{p,i}]$. Each sample $V_{e,i}(t)$ within the entire pseudo transient simulation is then rounded to the respectively nearest reference sample $V_{e,i}(t_{\{0,...,k\}})$, distributing the value range in $k-1$ rounding intervals. During this rounding step only the voltage tuples $V_e(t) \in S_{pt}$ are modified. Unlike with true resampling, the associated current- and charge samples $I(V_e(t))$ and $Q_{e,i}(V_e(t))$ remain unchanged.

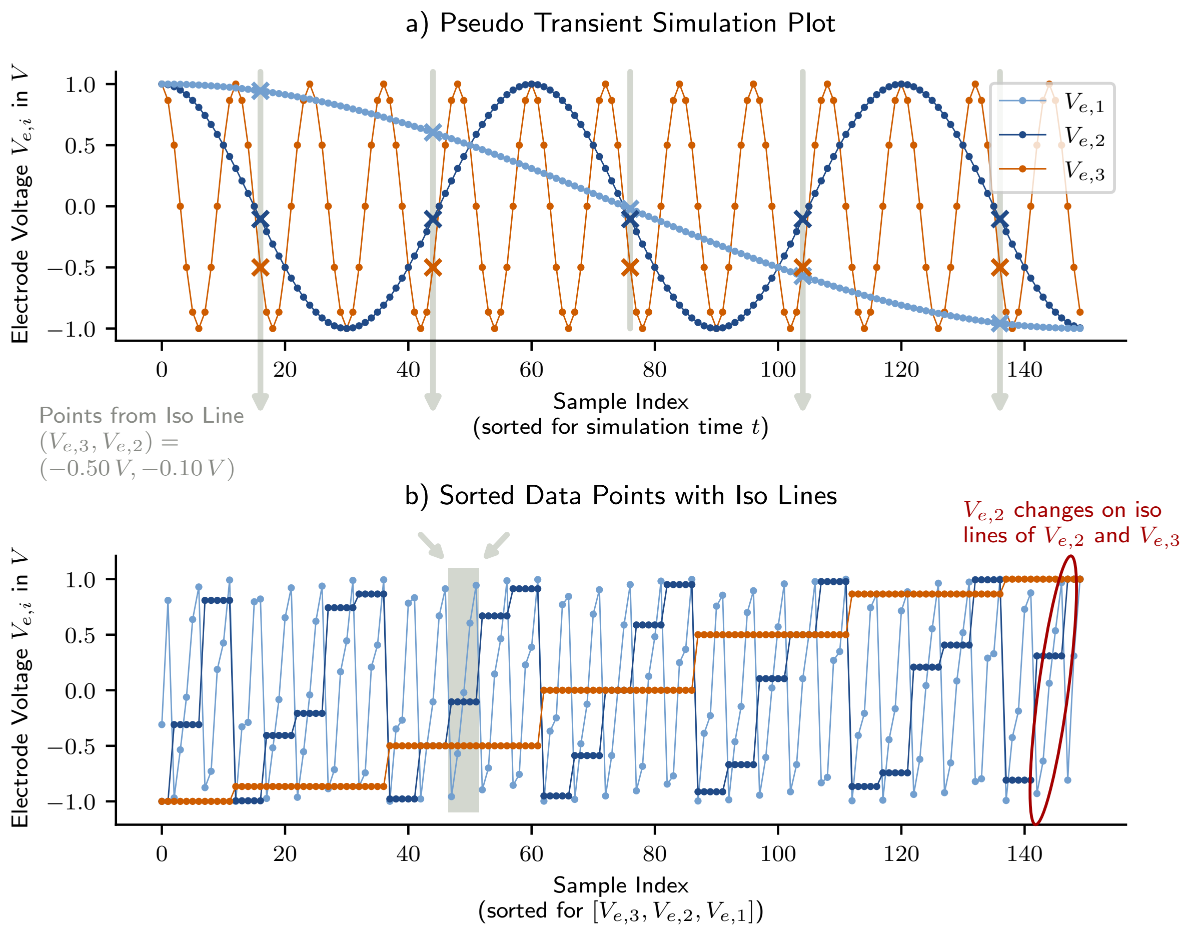


Figure 3.19: Sorting the samples from pseudo transient simulation by $V_{e,N_e}, ..., V_{e,1}$, exposes the iso lines of the obtained data set. Unlike a full factorial data set, the pseudo transient simulation shows less regularity.

To illustrate the structure of a pseudo transient data set, the simulation result of a setup with $T_{sim} = 1\,s$, $m = 5$ and $N_{os} = 3$ is shown in Figure 3.19. With respect to simulation time, the data points are the time-discrete points on harmonic waves, as shown in a). Sorting for $V_{e,3}, V_{e,2}, V_{e,1}$, Figure 3.19 b) exposes the iso lines formed by $V_{e,3}$ and $V_{e,2}$, which are the subsets with constant $V_{e,3}$ and $V_{e,2}$. The number of voltage steps in every dimension $V_{e,i}$ for $i \in 1, ..., N_e - 1$ varies with $V_{e,i+1}$. Further, the respective voltage steps are not uniformly distributed on their respective iso line.

Short Summary

To conclude this section, the pseudo transient simulation is a novel simulation to cover a parameter space $\mathbb{R}_e^N$ using frequency-nested harmonic signals in a single transient simulation. This section describes the setup for a general fundamental function, sine, and lays the mathematical foundation for selection of basic parameters, which then allow the attenuation of transient components in the data samples. The proposed approach is used to generate a table model of the planar RFET in this work.

4 Data Driven Compact Modeling

Typically, data driven device modeling approaches stop at a point where a characteristic data set is obtained and Verilog-A's table model functionality takes over interpolation during circuit simulation. Limitations of table models, such as the strict structural constraints, difficult convergence and high computational cost, however, justify exploration of post-processing methods. This section goes one step further and presents methods to transform characteristic data sets into analytic equation based models, i. e. compact models, to achieve higher robustness and performance.

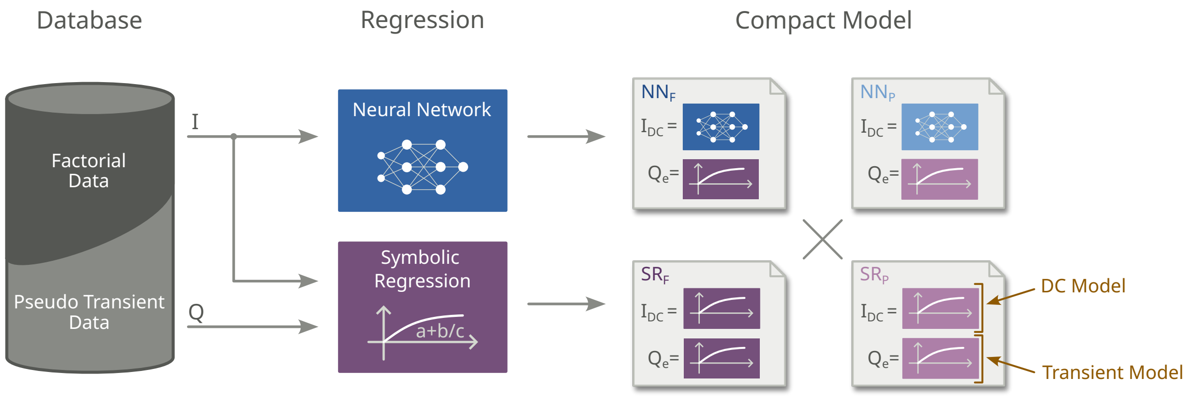


Figure 4.1: Two types of data sets, factorial data and pseudo transient data, along with two types of two modeling approaches for I_{DC} generate a total of 4 compact models, NN_F, NN_P, SR_F and SR_P. The charge models are all generated by symbolic regression.

This chapter explains on the methodology of model building using machine learning methods following the fundamentals in Section 2.3. The goal of the proposed methods is to form a versatile characterization platform for semiconductor device characterization and model building, where little domain knowledge is available. The underlying data set does not have structural constraints, such as the iso line criterion, and every data sample $I_{DC}(V_{e,1}, ..., V_{e,N_e})$ and $Q_{e,i}(V_{e,0}, ..., V_{e,N_e})$ of with $i \in 0, ..., N_e$ can be used for training. Parameters of the regression flow are selected with the planar RFET in mind and the designated target environment is digital cell simulation. In particular, this chapter describes the transformation from either quasistationary- or pseudo transient data, presented in Chapter 3, to equation based Verilog-A device models. The modeling flow is shown in Figure 4.1. The regression methods employed to form the respective equations are deep learning and symbolic regression. The DC drive current is therefore either modeled as a neural network or as a symbolic equation. All electrode charge behavior for the transient model is generated using symbolic regression. Considering that the source data set is either generated by a factorial approach or a pseudo transient approach, 4 different compact models, $\mathsf{NN_F}$, $\mathsf{NN_P}$, $\mathsf{SR_F}$ and $\mathsf{SR_P}$ are formed.

Parts of this chapter have been developed within the scope of the master's thesis by Johannes Wilm and published in [43].

4.1 The Compact Model Architecture

Although the presented techniques are developed with an unrestricted number of independent gates in mind, this work restricts the number of independent gates to two, to focus on the methodology. The methods are shown with the planar RFET as an example and the electrodes of interest are top gate (TG), front gate (FG) and the two lateral channel electrodes lateral terminal 1 (Lat1) and lateral terminal 2 (Lat2). The back gate (BG) electrode is assumed to be hard-wired to Lat1, so that $V_{BGLat1} = 0$. With the BG tied to Lat1, the device is optimized for operation in digital circuits [45], which is the target application in this work. However, the BG charge Q_{BG} is relevant for the transient behavior, so that a charge model for BG is required.

The architecture of the compact model consists of two levels, as shown in Figure 4.2: The electric model distinguishes between gates and channel adjacent terminals. For the DC model, the conduction of the channel is represented by a single current source, while gates are only regarded as infinite impedance inputs. The transient model adds a capacitive current to every electrode. Following the good practice for Verilog-A compact modeling from Section 2.2.2, the capacitive currents are obtained by deriving the respective electrode charges with respect to time.

The parameters of the sources on the electric level stem from the machine learning level, which provides the numeric predictions of DC drive current and electrode charges. The DC current model is an ensemble of two models. The linear model is trained on the data set, where the only allowed transformation is linear scaling. Within the high dynamic range of the RFET currents, the linear model is expected to perform well for high currents and lack accuracy at low currents. To compensate for the lack of accuracy with low currents, the logarithmic model is trained on logarithmically transformed data with the expectation of good overall performance over the entire dynamic range. A dynamic weight function then combines both predictions I_{lin} and I_{log} into a single ensemble prediction I_{DC}, which then serves as the DC current parameter for the current $I_{Lat12} = I_{DC}$. The electrode charges are predicted individually by 4 symbolic regression models, which are analytic equations. The features and therefore the inputs of all machine learning models are identically: The currents and charges are solely predicted from the electrode voltages with respect to the reference electrode Lat1.

4.2 Data Set Operations

Properties of the available data sets, such as the distribution of samples within the sample space have to be taken into account when designing regression methods. The density of samples is treated in Section 4.2.1. The split between the training-, test- and validation set is elaborated on in Section 4.2 and augmentation of the data set is described in Section 4.2.3.

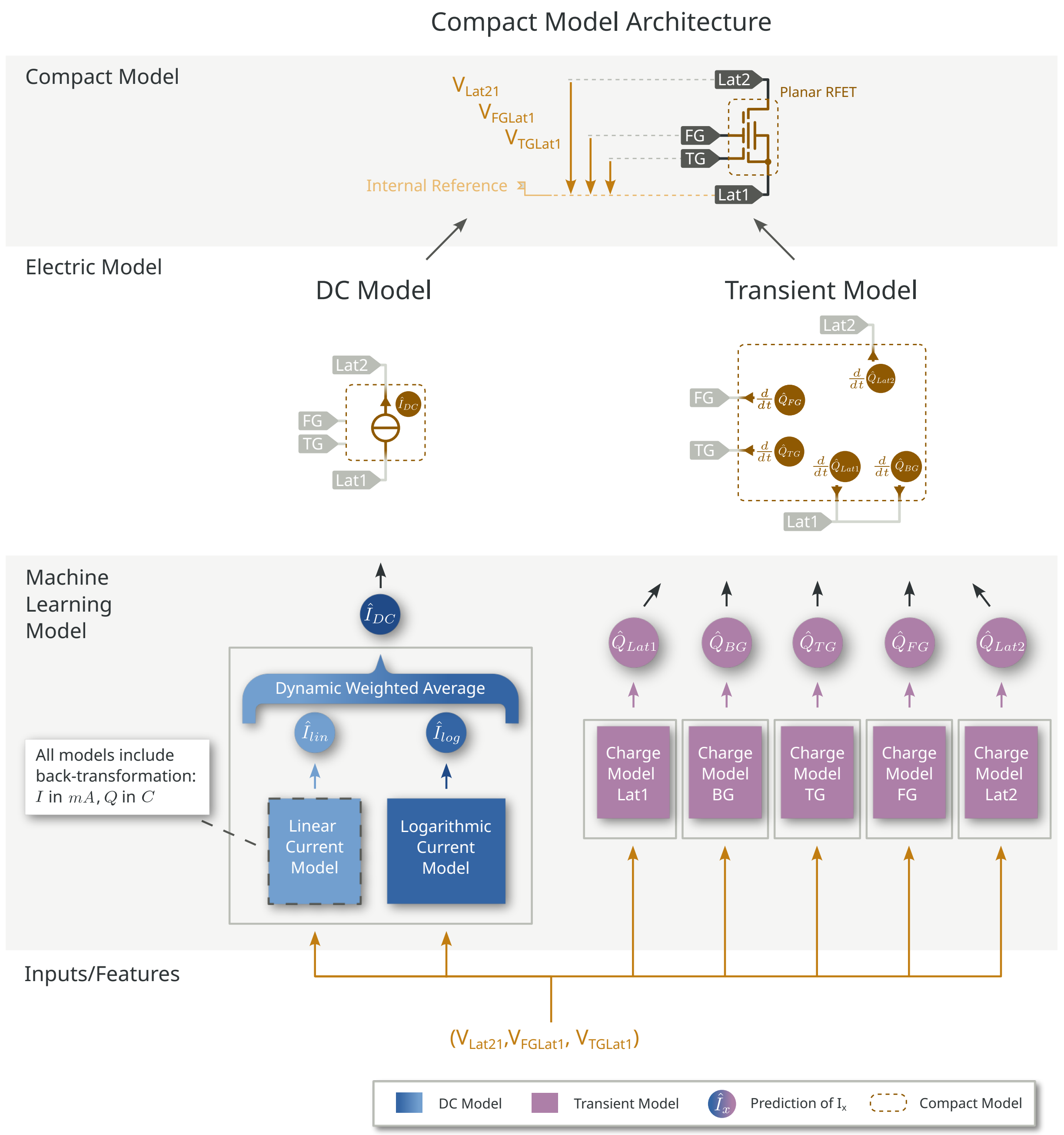


Figure 4.2: The compact model architecture features a superposition of a DC- and a transient model. All models receive the same input features, which are the electrode voltages with reference to V_{Lat1}.

4.2.1 Sample Density of Quasistationary Sweeps

The internal sample structure within Sentaurus Device, be it from quasistationary- or transient simulation, shows variable step size. Internally, the steps on which a solution is available depend on convergence or discretization of the transient solver, as explained in Section 2.2.1. A typical sample distribution over a parameter- or transient sweep features a low initial step size, which increments towards the maximum constrained step size. If operating ranges with low convergence are crossed, the step size can drop to the constrained minimum. The consequence of this adaptive step size approach is that the sample density is generally higher at the beginning of a sweep and in low convergence regions, providing more data samples in these regions. When regression algorithms execute on uneven sample distributions, the imbalanced loss leads to reduced performance compared to a data set of same size and uniform distribution [158], [197]. Further, focusing the precision on high sample density regions, i. e. initial sweep region and low convergence regions, does not have benefits in circuit simulation. Convergence depends on other factors in circuit simulation, as the underlying equation systems are different from the semiconductor equations of a technology simulation. Therefore, the approaches proposed in this chapter target uniform model precision over the full operating range. In this work, it is decided to follow a data driven approach to reduce imbalances in the source data, as described in Section 2.3.4, because processing of input data is deemed simpler and more generally applicable than algorithmic adaptions.

The data from Sentaurus Device can generally be exported on a regular grid, uniformly distributed with respect to the sweep parameter, using the CurrentPlot feature described in Section 2.2.1. However, if a regular grid is not strictly required, the explicit intervals that can be specified for the CurrentPlot cause loss of available information. The steps, constrained by the respective intervals, are exported, while the solutions of the remaining steps, placed through adaptive stepping, are discarded. As a consequence, efficiency, interpreted in terms of information outcome per simulation, is reduced by constraining a regular grid export.

Imbalance in the sample distribution, as proposed in [191], does not have to be eliminated entirely. It is not an algorithmic requirement for the machine learning methods employed in this work, to have an exactly regular structure, such as the iso line constraint in Verilog-A's `$table_model` function. The goal of an introduced preprocessing step is rather to reduce the granularity in ranges where quasistationary technology simulation produces a high sample density due to a low step size. To achieve this, each parameter sweep is pruned in a post-processing step until a specified minimum step size $\Delta V_{sweep,prune}$ is reached. The prune algorithm removes the sample step n with the minimum preceding step size

$$\Delta V_{sweep}(n) = V_{sweep}(n) - V_{sweep}(n-1), \tag{4.1}$$

for $n > 0$, and repeats until no more samples with

$$\Delta V_{sweep}(n) \leq \Delta V_{sweep,prune} \tag{4.2}$$

are found. Figure 4.3 demonstrates how a parameter sweep is pruned to a step size of $V_{sweep,prune} = 25\,mV$ within 369 iterations in total. The imbalance in the distribution of samples is reduced without introducing resampling errors from interpolation. From the perspective of the machine learning modeling approach, this is a preprocessing step and results in the dataset $\mathcal{D}_{pp}$.

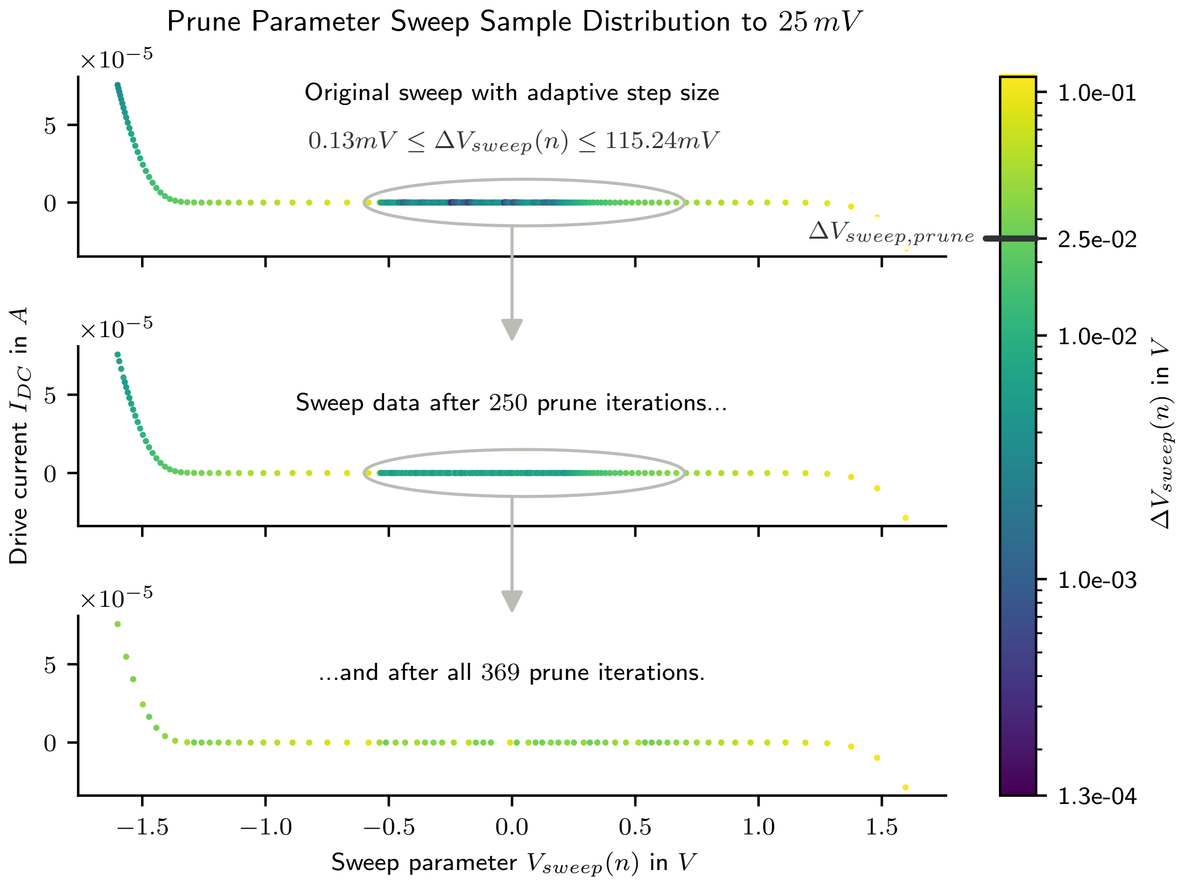


Figure 4.3: In order to improve balance of a data set from a parameter sweep with adaptive step size, iterative pruning removes all samples which contribute to a granularity with a step size of less than ΔV_{sweep}. Resampling is avoided.

The pruning step is only performed for the factorial data set, as the pseudo transient dataset inherently features a higher regularity due to the uniform sampling of the harmonic functions in time domain.

4.2.2 Splitting into Training-, Validation- and Test Set

Having reduced imbalances in the sample distribution in Section 4.2.1, the question arises which samples are selected for model training, leaving the remaining samples for testing and validation.

Acceptable balance assumed, deep learning is usually performed on as many data points as available and the question is typically which part of the data can be spared to allow validation and testing. Symbolic regression methods on the other hand do not rely as strongly on the size of the data set as neural networks (see Section 2.3.2). In fact, the benefit of additional data points are expected to diminish at increasing cost in form of computational time of the regression method as well as the cost of generating the data points through technology simulation or measurement. Unlike for neural network approaches, there are no commonly used splits to be found throughout literature, and it is likely that what is considered a good split is highly dependent of the specific symbolic regression method used. Therefore, and for the sake of comparability, the training-, test- and validation sets for symbolic regression and neural network methods are chosen to be equal in this work. For both machine learning current models, the available preprocessed data set $\mathcal{D}_{pp}$ is divided threefold into $10\%/13.5\%/76.5\%$ for $\mathcal{D}_{test}/\mathcal{D}_{val}/\mathcal{D}_{train}$ as shown in Figure 4.4. The symbolic regression approach does not require a validation set, as neither hyperparameter training, nor early stopping take place. For that reason, the validation set of the symbolic regression approach is only used for hyperparameter training of the respective ensemble backends of $\mathrm{SR_F}$ and $\mathrm{SR_P}$. Symbolic regression based charge models feature neither hyperparameters to be tuned nor is there early stopping available. The split ratio is, however, kept even for the charge models, in order to establish comparability between all machine learning approaches. In these cases, the validation set is unused.

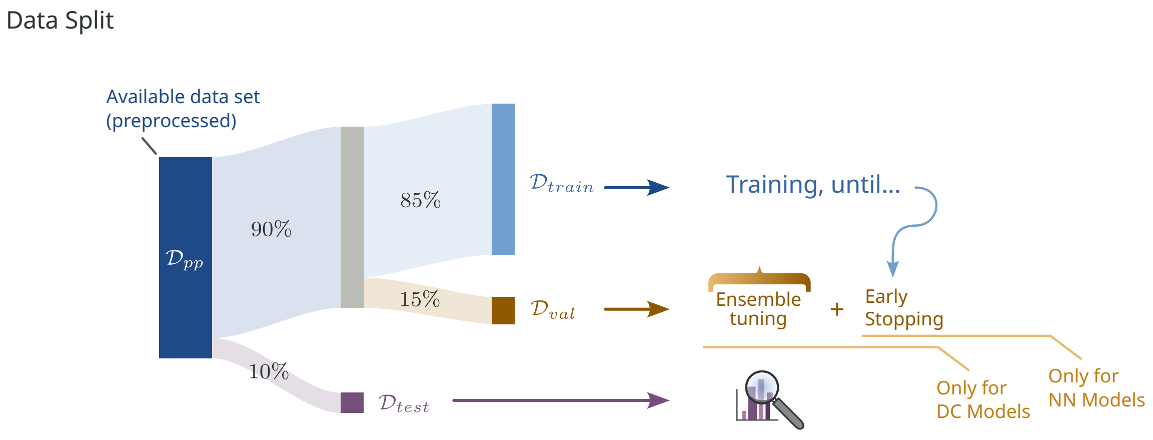


Figure 4.4: The split of the data set is equal for all proposed machine learning models, in order to establish comparability. Models that do not require the validation set, such as symbolic regression models, ignore $\mathcal{D}_{val}$.

4.2.3 Data Augmentation

As described in Section 2.3.4, the addition of noise to the training set, can improve the generalization ability of a neural network model [201]. The idea of data augmentation, to apply a mathematical transformation for which the model is intended to be invariant, to a copy of the data set, is appealing to increase the size of the available data and can be adapted for transistor modeling. Particularly noise augmentation comes into focus, when considering that both, generation of characteristic data and also circuit simulation, introduce numeric noise to model evaluation. The noise model does not have to be derived from expected influences for the targeted use-case, still the, in electronic circuits commonly found, Gaussian noise model is expected to be a reasonable choice [209], [210]. Gaussian noise, characterized by the probability density

$$\varphi(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}, \tag{4.3}$$

is adapted by specifying expected value μ and standard deviation σ.

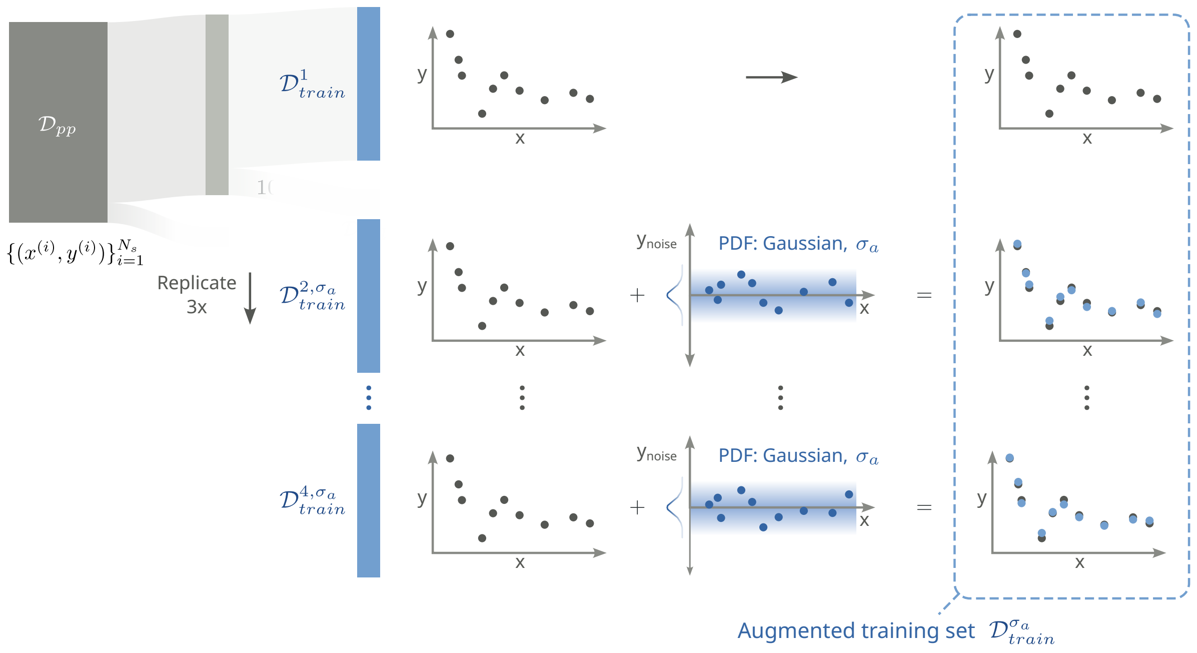

$\mathcal{D}_{pp}$
$\{(x^{(i)}, y^{(i)})\}_{i=1}^{N_s}$
Replicate 3x
$\mathcal{D}^1_{train}$
$\mathcal{D}^{2,\sigma_a}_{train}$
$\mathcal{D}^{4,\sigma_a}_{train}$
y
x
y_{noise}
PDF: Gaussian, σ_a
+
=
Augmented training set $\mathcal{D}^{\sigma_a}_{train}$

Figure 4.5: The data set on which the logarithmic model is trained, is augmented with Gaussian noise. 3 copies of the original set are each superimposed with a noise set of $\mu = 0\,A$ and a reasonable σ_a.

After selecting a suitable noise model, the training set is replicated 3 times, in order to obtain 4 identical sets $\mathcal{D}^{\{1,\ldots,4\}}_{train}$. Figure 4.5 shows that 3 individual sets with length $|\mathcal{D}_{train}|$ are sampled from the Gaussian probability density function Equation 4.3 with a specific σ_a. To each ground truth result y_i of the training sets $\mathcal{D}^{\{2,\ldots,4\}}_{train}$ the respective individual noise samples are added.

The introduction of a bias into the training data has to be avoided, leading to the constraint $\mu = 0\,A$ for the Gaussian probability density function $\varphi(x)$. The selection of a reasonable standard deviation σ is key and needs to be sufficiently high to improve generalization but low enough to not disturb the characteristic of the device. Figure 4.6 shows an excerpt from an exemplary training set $\mathcal{D}_{train}$, which is a $V_{Lat21} \in [-1.6\,V, 1.6\,V]$ sweep at $V_{FGLat1} = -1\,V$ and $V_{TGLat1} = 1\,V$. Two figures can be taken into account: Firstly, the DC drive current I_{DC} is expected to be monotonic with respect to the lateral channel voltage V_{Lat21}. I can be seen in Figure 4.6 that this relationship holds true down to $|I_{DC}| \approx 10^{-17}\,A$ rendering it a reasonable estimation of the noise floor. And secondly, the lowest expected current to occur in a simulated digital circuit can be considered. In the scope of this work, digital cells with a maximum of two inputs are considered, and therefore the lowest current is expected to be within a decade of the minimum current in an inverter voltage transfer characteristic (VTC). Using the same technology model, the inverter simulation with two statically configured planar RFET devices in [43], results in $I_{INV,min} = 2.12 \cdot 10^{-12}\,A$. Further, Figure 4.6 shows the effect of different σ_a at and above $I_{INV,min}$, which introduce high disturbance of the characteristic of the device. It is therefore safe to place the noise augmentation closely to the observed noise floor and a standard deviation of $\sigma_a = 10^{-17}\,A$ ensures that 99.7 % of introduced Gaussian noise samples lie within $3\sigma_a = 3 \cdot 10^{-17}\,A$.

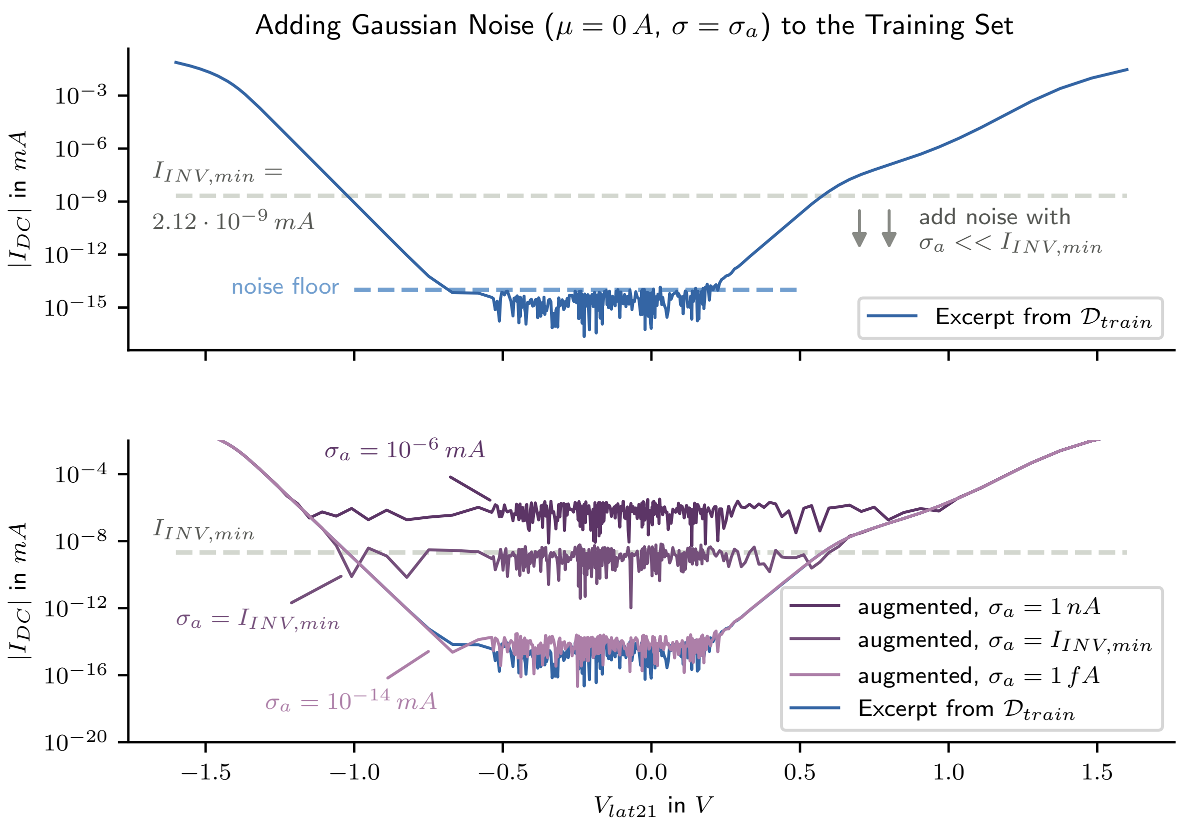


Figure 4.6: Screening the parameter sweeps performed in technology simulation exposes numeric noise below $1 \cdot 10^{-14}\,mA$. This figure shows the effects of different σ_a on the augmented data set.

Not all modeling approaches benefit from the described noise augmentation strategy. In the presence of logarithmic models, the linear models are only consulted for peak currents. Further, symbolic regression methods do not necessarily benefit from increased training set size, as described in Section 2.3.2. This work therefore considers the noise augmentation strategy only for the logarithmic neural network model.

4.2.4 The Ensemble Backend

The objective of the ensemble backend is to provide a prediction of the DC drive current I_{DC} based on the linear- and the logarithmic model. The logarithmic model is expected to outperform the linear model for low currents and vice versa. A practical solution is therefore to prefer the logarithmic model for low currents and the linear model for high currents. Regardless of what is considered *high-* and what is considered *low* current, a handover between both models has to be designed for a threshold $I_{\beta,th}$.

The choice of the right model to emphasize for the overall prediction resembles a classification task. But instead of training a separate classifier, a robust estimation can be derived from the logarithmic prediction: Although lacking accuracy for high currents, in experiments with planar RFET data the logarithmic model has shown to provide more overall relative accuracy over the whole dynamic range than the linear model. It is therefore possible to use the logarithmic current prediction to dynamically weigh both predictions accordingly. The decision where to place the threshold $I_{\beta,th}$, the current which marks the handover between both models, depends on the relative accuracy between the models. With the accuracy of the linear model expected to decrease with the DC drive current prediction, $I_{\beta,th}$ then marks the threshold at which the logarithmic model outperforms the linear model. A straightforward optimization method is to construct $I_{\beta,th}$ from hyperparameters in the continuous domain and tune them accordingly.

Accuracy of the ensemble is not the only criterion for a good compact model, because the resulting ensemble predictions I_{DC} are interpreted as a physical process within numeric simulation. As explained in Section 2.2.2, robust Verilog-A models are continuous and differentiable and these criteria therefore apply to the ensemble backend. Instead of an instantaneous handover between linear and logarithmic model at $I_{\beta,th}$, a smooth transition is required. The transition function consists of a weighted average function with a dynamic weight. The general weighted average function from Equation 2.19 for $N_{ens} = 2$ reduces to

$$\hat{I}_{DC} = \sum_{i=1}^{2} w_i \cdot \hat{y}_i = w_{lin}\hat{I}_{lin} + w_{log}\hat{I}_{log}. \tag{4.4}$$

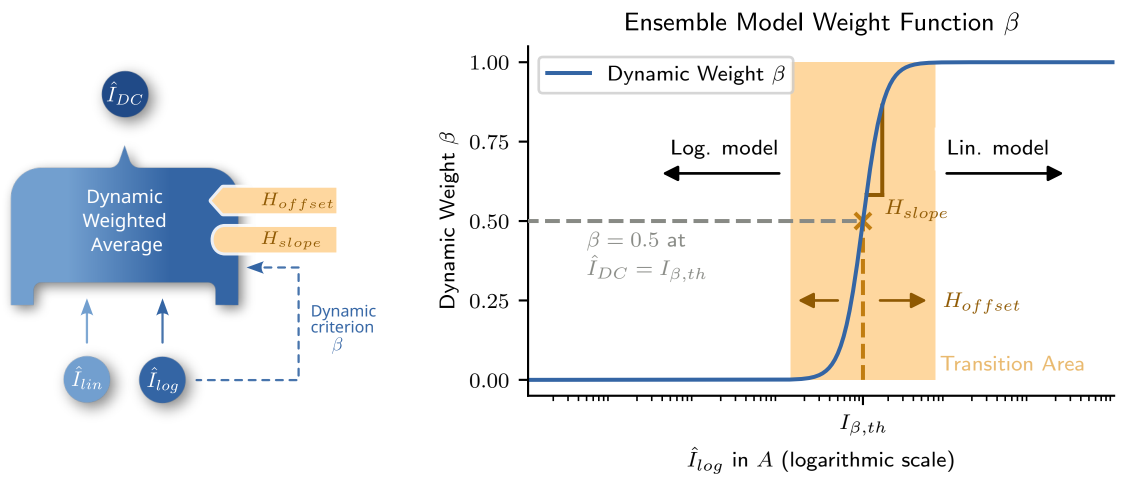


Figure 4.7: The dynamic weight function $\beta(\hat{I}_{log})$ provides a smooth transition between linear- and logarithmic model. The width of the transition region and the threshold current, at which both models are weighted equally, are found during hyperparameter training of H_{slope} and H_{offset}.

Both weight factors are required to add up to 1, which allows their formulation with a single parameter $\beta \in (0, 1)$

$$w_{lin} = \beta, \tag{4.5}$$
$$w_{log} = 1 - \beta. \tag{4.6}$$

The overall ensemble function can then be formulated as

$$\hat{I}_{DC} = \beta(\hat{I}_{log}) \cdot \hat{I}_{lin} + (1 - \beta(\hat{I}_{log})) \cdot \hat{I}_{log}. \tag{4.7}$$

A function that is commonly used whenever the application demands a smooth monotonic transition is the sigmoid function. A special type of sigmoid function, tanh, is in fact provided by Verilog-A, and is therefore employed for the ensemble model. To provide monotonic transition between for $\beta \in (0, 1)$, the dynamic parameter then equates to

$$\beta(\hat{I}_{log}) = \frac{1}{2}(1 + \tanh(H_{slope} \cdot u(\hat{I}_{log}) + H_{offset})), \tag{4.8}$$

with the hyperparameters H_{slope} and H_{offset}, and the transformation $u(\hat{I}_{log})$. The hyperparameters H_{slope} and H_{offset} constrain the width of the transition region and the location of $I_{\beta,th}$, as shown in Figure 4.7. The definition of $u(\hat{I}_{log})$ remains, which is required to follow the magnitude of $\hat{I}_{log}$. The transition is designed to be sensitive to the orders of magnitude of $\hat{I}_{log}$, and it is therefore suitable to transform the relevant criterion $\hat{I}_{log}$ logarithmically. The function $u(\hat{I}_{log})$ then follows

$$u(\hat{I}_{log}) = \frac{1}{2}\log_{10}((\hat{I}_{log})^2 + \varepsilon), \tag{4.9}$$

where $\varepsilon = 10^{-100} A^2$ is introduced to avoid the evaluation of the logarithm at $\hat{I}_{log} = 0\,A$, while the occurrence of $(\hat{I}_{log})^2 + \varepsilon = 0\,A$ is expected to be less likely within the circuit simulation environment. The inverse function β^{-1} to obtain the characteristic current-related values, such as the location and width of the transition region shown in Figure 4.7, is

$$\beta^{-1} = \hat{I}_{log}(\beta) = 10^{\frac{\tanh^{-1}(2\beta-1)-H_{offset}}{H_{slope}}}. \tag{4.10}$$

The result is a dynamic weight function $\beta(\hat{I}_{log})$, which provides a smooth and monotonic transition between logarithmic and linear model, where slope and offset can be optimized.

4.3 The Neural Network Models

The choice of the deep learning library is expected to play a minor role in model performance and is therefore based on convenience. In this work, the neural network approach is implemented in TensorFlow [211], partially by making use of Keras API [212] with TensorFlow as backend. Neural networks are exclusively used for I_{DC} prediction, consisting of a linear and a logarithmic model, respectively. This work does not consider neural network implementations for charge prediction as part of the transient model, because the relation between electrode charges and the electrode voltages is expected to be less complex to fit and therefore a good use-case for symbolic regression as described in Section 4.4.

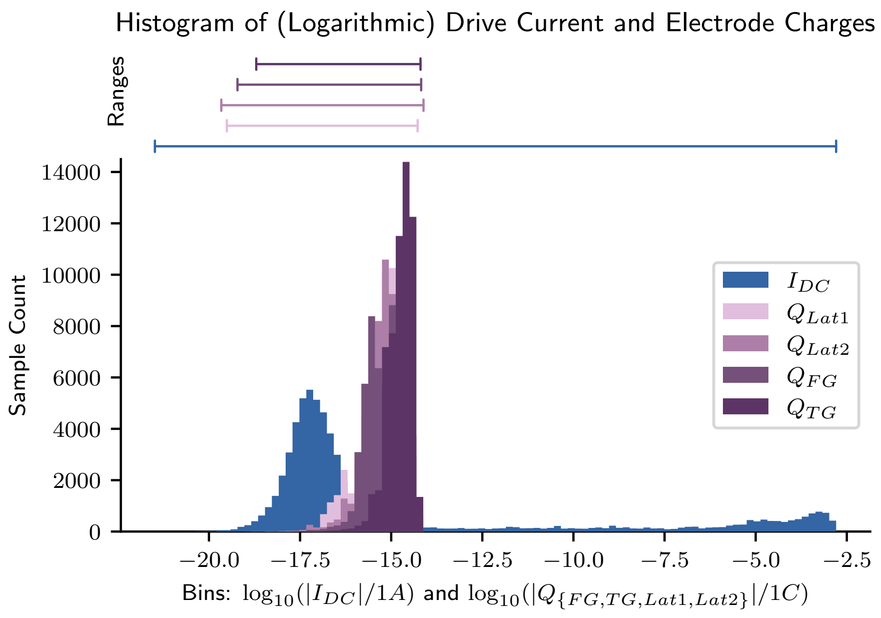


Figure 4.8: The histograms of samples in logarithmic bin size shows that the distribution of current and charges is greatly different. Instead of covering a wide range, as seen for I_{DC}, the electrode charges are concentrated around $10^{-15}\,C$.

An important issue when training machine learning models is to achieve accuracy over the value space of the respective use-case. In the use-case of circuit simulation for digital cells, I_{DC} covers a high dynamic range, as active devices cross multiple operating regions during switching. Considering the simulation of a basic logic cell, the CMOS-style inverter, the drive current of the planar RFET typically lies within $5.43 \cdot 10^{-6} A$ and $2.12 \cdot 10^{-12} A$ in DC analysis, spanning 6 decades [43]. For the planar RFET technology model, the dynamic of the distribution differs between DC drive current I_{DC} and electrode charges Q_e, as can be seen in factorial table model of the planar RFET on a regular grid. Figure 4.8 shows that, in the biasing range of $V_{\{FGLat1,TGLat1,Lat21\}} \in [-1.6\,V, 1.6\,V]$ and $V_{BGLat1} = 0\,V$, I_{DC} spans the range of $(3.29 \cdot 10^{-22}\,A, 1.61 \cdot 10^{-3}\,A)$. There are two regions where samples accumulate: The highest concentration of samples is around $I_{DC} \approx 10^{-17}\,A$. Further, many samples show drive current in the range of $I_{DC} \in (10^{-5}\,A, 10^{-2.5}\,A)$. The modeling approach has to account for both regions, including the transition in between. The ranges of the electrode charges, on the other hand, are more narrow and the vast majority of sample lie around $10^{-15}\,C$, as depicted in Figure 4.8. The lower dynamic range of the electrode charge values lead to the assumption that charge modeling is more straightforward than modeling I_{DC}.

It is essential for a compact model to project the low off currents as well as the high on currents, to enable accurate simulation of digital cells. Problems arise, when training a model with a loss function that depends on the absolute error, e. g. the MSE. A higher relative error would be accepted for low currents, biasing optimization towards the prediction of high currents. This is less than ideal, as the relative error of device characteristics is expected to be important for simulation of digital circuits, where the steady state output depends on the prediction of low currents of the complementary devices, for either low channel voltage or low gate voltage. While an absolute error of $\varepsilon = \pm 1\,\mu A$ in on-current prediction can possibly be neglected, an absolute error $\varepsilon = \pm 1\,\mu A$ error for a switched off device likely leads to a wrong logic level, for example. One way of achieving a constant relative accuracy over the entire current range is to utilize a relative loss function, such as MAPE or sMAPE as presented in Section 2.3.5. Of course, this requires the respective machine learning model to have the expressiveness to fit the respective data accurately, in the first place. Another way is to transform the training data, compressing the dynamic range, and then utilize an absolute loss function, such as the MSE. In this work logarithmic transformation is used to compress the DC drive currents and then train the logarithmic model on them. A second model, the linear model, is trained on the plain data set without prior transformations, to complement possible inaccuracies of the compressed logarithmic model with respect to high currents. The decision is based mainly on two factors: Firstly, splitting the coverage of the dynamic range of the DC drive current into two models increases expressiveness. The compression, secondly, allows the use of the MSE, which is sensitive to outliers and cheap to compute, as a loss function for both models. Although evaluation of different loss functions is not part of this work, preliminary tests on data sets of the planar RFET evaluated in favor of the MSE, when evaluated on linear model and logarithmic DC drive current model. After training, percentage error metrics are applied for performance evaluation of the obtained models.

Logarithmic- and linear neural network models are two separate MLPs with distinct structure, following the basic concepts presented in Section 2.3.1. Both models consist of 3 densely connected hidden layers, inspired by Hutchins et al. [112] and Tung et al. [203], who deem 3 hidden layers to provide a good trade-off between complexity and accuracy for transistor models. The structural distinction comes from the different size of the respective hidden layers of each model. Within a model, all 3 hidden layers have the same size. To find the optimal hidden layer size the hyperparameter $H_{hl-size,<model>}$ is introduced and optimized, as described in Section 4.5. However, the optimal $H_{hl-size}$ is not necessarily the one which causes the best score on the validation set, as increasing complexity of a model reduces its generalization power according to Aggarwal [131]. In addition, computation effort of the compact model rises with the layer size. A good practice is therefore to screen the results of the best $H_{hl-size}$ make a trade-off between score and layer size. If the second-best model achieves similar performance to the best model at a fraction of the hidden layer size $H_{hl-size}$, it is reasonable to prefer the second-best model, for instance.

Two mathematical properties make MLPs suitable for compact models: they can be designed to be continuous and differentiable with respect to their inputs. Differentiability is provided, according to Leibniz' chain rule, if the MLP consists solely of differentiable functions. The linear combinations of weighted inputs fulfill this criterion naturally, so the choice of activation function is the decisive factor. As explained in Section 2.3.1, the commonly used activation function ReLU is not differentiable. Further, at a hidden layer count of 3, the MLPs are on the shallow side of deep learning and therefore vanishing gradients, which would be a strong argument for the use of ReLU, are not expected to be an issue. Instead, logistic function and tanh come into focus. Similar to the ensemble model presented in Section 4.2.4, the decision is made in favor of tanh, as its implementation is provided in Verilog-A. All hidden layer neurons use tanh as activation function, while the output neuron features linear activation.

4.3.1 The Linear Neural Network Model

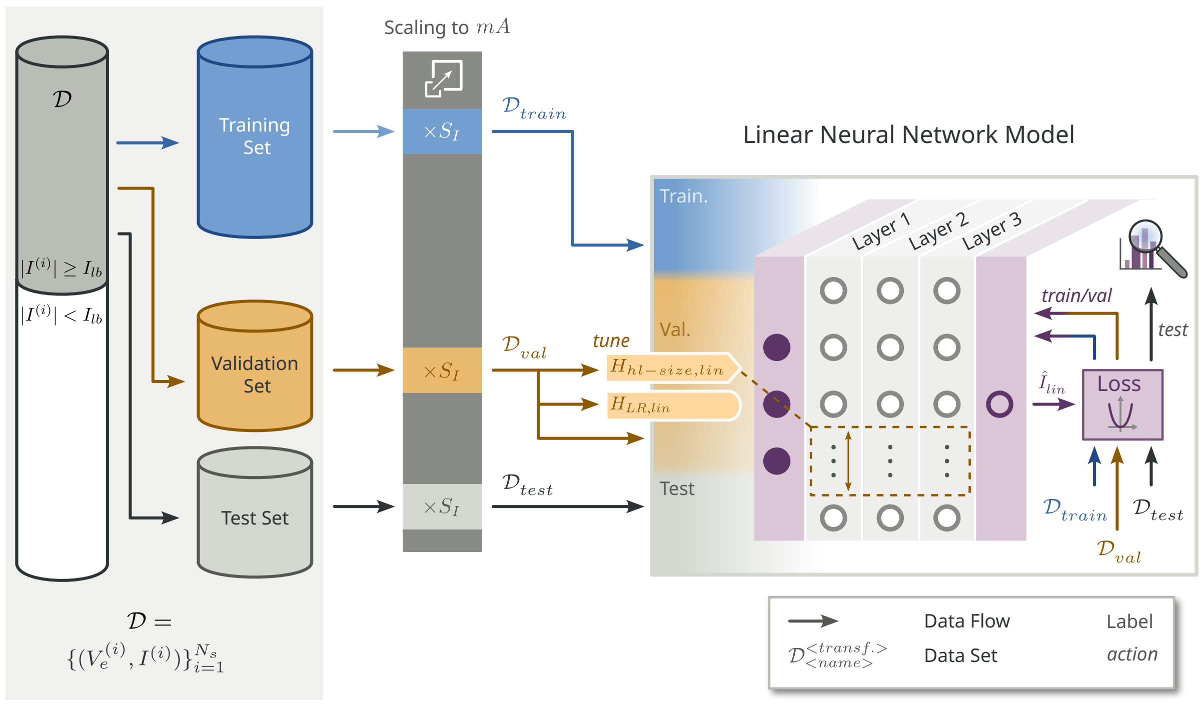


Figure 4.9: A lower bound of the current samples improves accuracy in the prediction of high currents. Before training of the linear neural network model, the data sets are scaled to mA.

The training method of the linear model is shown in Figure 4.9. Unlike the logarithmic model, the linear model is not trained on the entire available data set, because the ensemble backend renders its predictions irrelevant for low currents. The logarithmic model is expected to outperform the linear model for low $|I_{DC}|$, and therefore receives the dynamic weight for low current regions. It is therefore reasonable to design the linear model to ignore low currents at all by removing them from the data set, effectively introducing a lower bound I_{lb} before performing the split. Accuracy of the linear model improves, as a large share of ground truth values for irrelevant bias regions is left out. The lower bound $I_{lb} = 10^{-11}\,A$ is set well below the lower bound of the transition region for the ensemble hyperparameter corner of $H_{offset} = 40$ and $H_{slope} = 8$, described in Section 4.5.

The remaining data set is split according to Section 4.2. The distribution of the DC drive current has a peak of $1.61\,mA$ in magnitude, as described in Section 4.3, and a typical approach is to scale output values to the input feature ranges to aid convergence during neural network optimization. The input features all have ranges of $V_{e,i} \in [-1.6\,V, 1.6\,V]$ for all electrodes E_i. Accordingly, training, validation and test set are scaled linearly by a factor $S_I = 1000$, which corresponds to a transformation from A to mA. The linear model is then trained on the training set $\mathcal{D}_{train}$, allowing a maximum of 5000 training epochs. The validation set is used for early stopping and interrupts training when the loss on validation data reaches a plateau. Further, the validation set is used for tuning of the layer size hyperparameter $H_{hl-size,lin}$ along with the learning rate $H_{LR,lin}$ as further described in Section 4.5.

4.3.2 The Logarithmic Neural Network Model

Figure 4.10 illustrates the data flow for building the logarithmic model. The logarithmic model is trained on the entire available data set, as the dynamic weight parameter β depends on the prediction of the logarithmic model $\hat{I}_{log}$. Accuracy over the entire dynamic range of available data is therefore required. Augmentation is used to increase the set of available training data and therefore improve the overall model performance. The augmentation strategy of choice in this work is Gaussian noise augmentation, as described in Section 4.2.3. In addition to the increase of training data, the transformation of adding noise to the training samples can

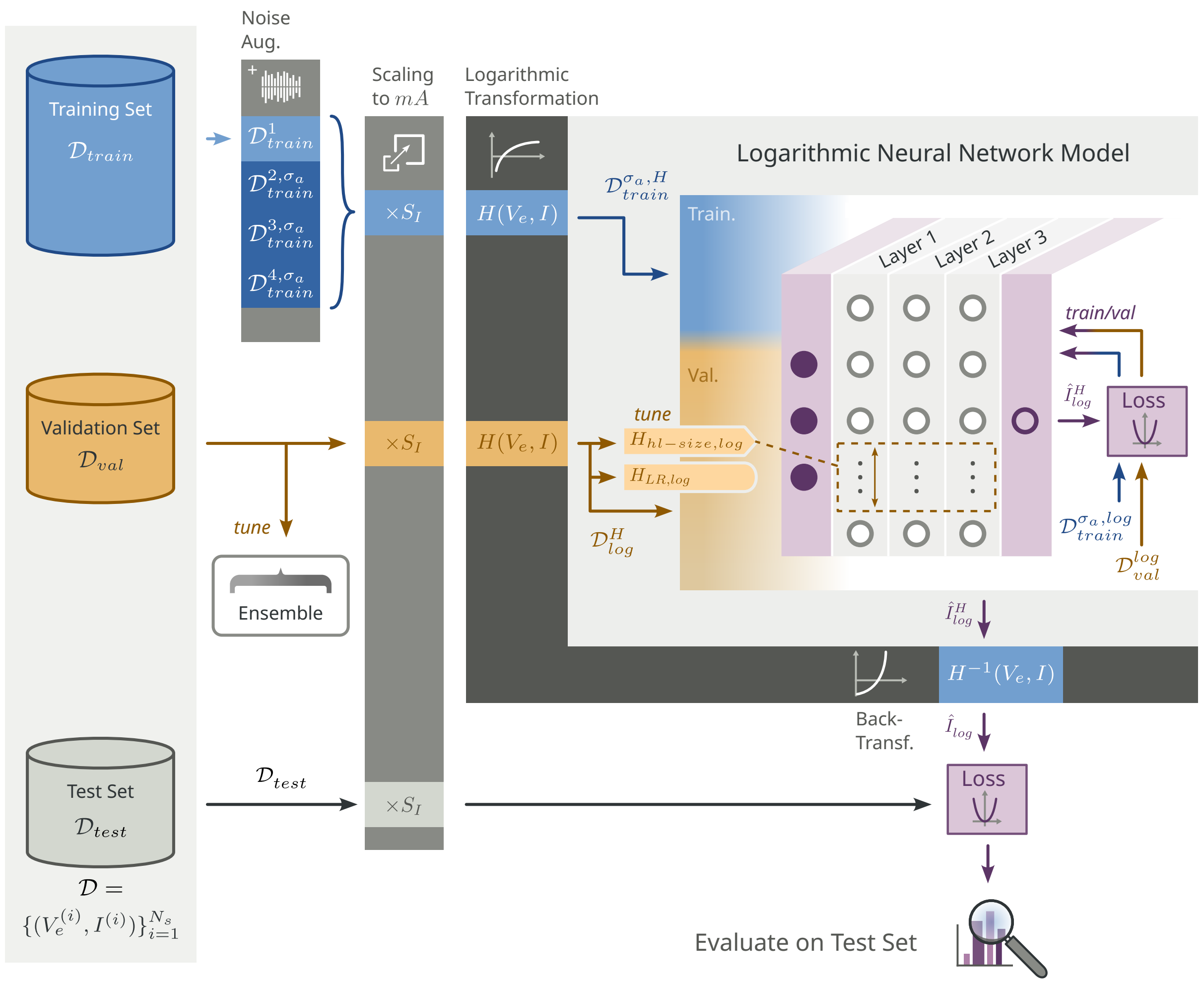


Figure 4.10: Training of the logarithmic model involves noise augmentation of the training set and the actual logarithmic transformation. The validation set is, further used for tuning of the hyperparameters of the ensemble model.

increase robustness of the model against noise, which can occur in circuit simulation. As the excerpt from the factorial data set in Figure 4.6 shows numeric noise with amplitudes in the range of $10^{-17}\,A$, the introduction of further noise is deemed uncritical. After splitting the available data into training-, validation- and test set, the training set is therefore augmented with Gaussian noise of $\mu_a = 0\,A$ and $\sigma_a = 10^{-17}\,A$. The noise augmentation step replicates the available training set 3 times and appends random noise to the replicas.

The resulting 4 training sets $\mathcal{D}^1_{train}$ and $\mathcal{D}^{\{2,3,4\},\sigma_a}_{train}$ are then transformed logarithmically. In order to compress the dynamic range of the drive current, the transformation is performed according to Tung et al. [203] as

$$I^H_{log} = H(I) = \ln\left(\frac{I_{DC}}{V_{Lat21}}\right), \tag{4.11}$$

where I_{DC} is the ground truth DC drive current and V_{Lat21} the according channel voltage. The natural logarithm is defined for arguments in $\mathbb{R}^+$, and it is expected that I_{DC} always has the same sign as the channel voltage V_{Lat21}, that causes it. The division by V_{Lat21} forces the argument of the natural logarithm to be positive. During measurement or simulation, noise can cause the sign of voltage V_{Lat21} and current I_{DC} to differ for certain samples. These samples cannot be transformed and are excluded from training. Further, division by $V_{Lat21} = 0\,V$ has to be avoided, and in order to preserve as many samples as possible a lower bound of $|V_{Lat21}| = 1\,mV$ is implemented by modification of the respective samples. After the described preprocessing steps the argument of the natural logarithm in Equation 4.11 is forced to $\mathbb{R}^+$.

Table 4.1: The QLattice composes symbolic models from various arithmetic functions, called *interactions* [174].

Name	**Interaction**
Addition	$a + b$
Multiply	$a \cdot b$
Squared	$a \cdot a$
Linear	$a \cdot \text{weight} + \text{bias}$
Tanh	$\tanh(a)$
Single-legged Gaussian	e^{-a^2}
Double-legged Gaussian	$e^{-(a^2+b^2)}$
Exponential	e^a
Logarithmic	$log(a)$
Inverse	$\frac{1}{a}$

4.4 The Symbolic Regression Models

Similar to the neural network approach, the symbolic regression approach is treated as a black box flow in this work. In addition, it is possible to interpret the resulting model equations and prefer physically reasonable candidates for further optimization, as described in Section 2.3.2. However, the scope of this work is limited to conduct a data driven modeling approach with as little domain knowledge as possible and therefore the physical sense of the generated equations is not inspected. Instead, the only criterion for the quality of the symbolic regression model is its performance on a test set and the performance in digital circuit simulation.

Of the symbolic regression implementations described in Section 2.3.2, QLattice shows convincing performance and is therefore chosen for the compact modeling approach conducted in the scope of this work. The symbolic regression models are obtained by using the available data set to train the QLattice through the Feyn API [175]. The set of arithmetic interactions which QLattice provides for symbolic regression is shown in Table 4.1. It is possible to restrict the set of interactions if preliminary knowledge about the distribution to be modeled is available. In this work, however, all interactions are allowed, in order to avoid the introduction of bias through constraining without specific domain knowledge.

The computational effort to execute a symbolic regression model in Verilog-A cannot be constrained as finely as the execution of a neural network model, where all node operations are known ahead of training. However, it is possible to influence the complexity of a symbolic regression model by limiting the allowed number of edges of the solution graphs. As the solution of non-linear models in a SPICE-like solver requires an a priori unknown number of iterations, the constrained complexity of the model does not necessarily reflect the performance in simulation. Preliminary trials, show that solution graphs for the planar RFET have not exceeded 50 edges. In this work a maximum complexity of 100 edges is permitted.

4.4.1 The Symbolic Regression-Based Logarithmic Current Model

Similar to the neural network modeling approach presented in Section 4.3, the symbolic regression based DC drive current model consists of a logarithmic and a linear model. The linear modeling approach can be explained together with the charge models, as they feature high similarity. Both training approaches are therefore described together in Section 4.4.2. The logarithmic model training, outlined in Figure 4.11, does provide a validation set for tuning the hyperparameters of the ensemble model, similar to the neural network modeling approach. However, in training of the symbolic regression model, the validation set cannot be used and is therefore ignored. The logarithmic symbolic regression approach shows another fundamental difference to its neural network counterpart, as there is no noise augmentation of the training data. The reason why the symbolic regression model does not rely on noise augmentation is that symbolic regression methods are expected to improve little with the amount of training samples, if the original training data set already expresses the key characteristics of the device, as presented in Section 2.3.2. Therefore the artifical augmentation of the training set is not expected to increase accuracy of the prediction. The logarithmic transformation of the training data $H(V_e, I)$ is performed for the symbolic regression approach equally as for the neural network-based logarithmic model, described as Equation 4.11 and Equation 4.13. Evaluation of the model performance is done on the test set after backtransformation of the model.

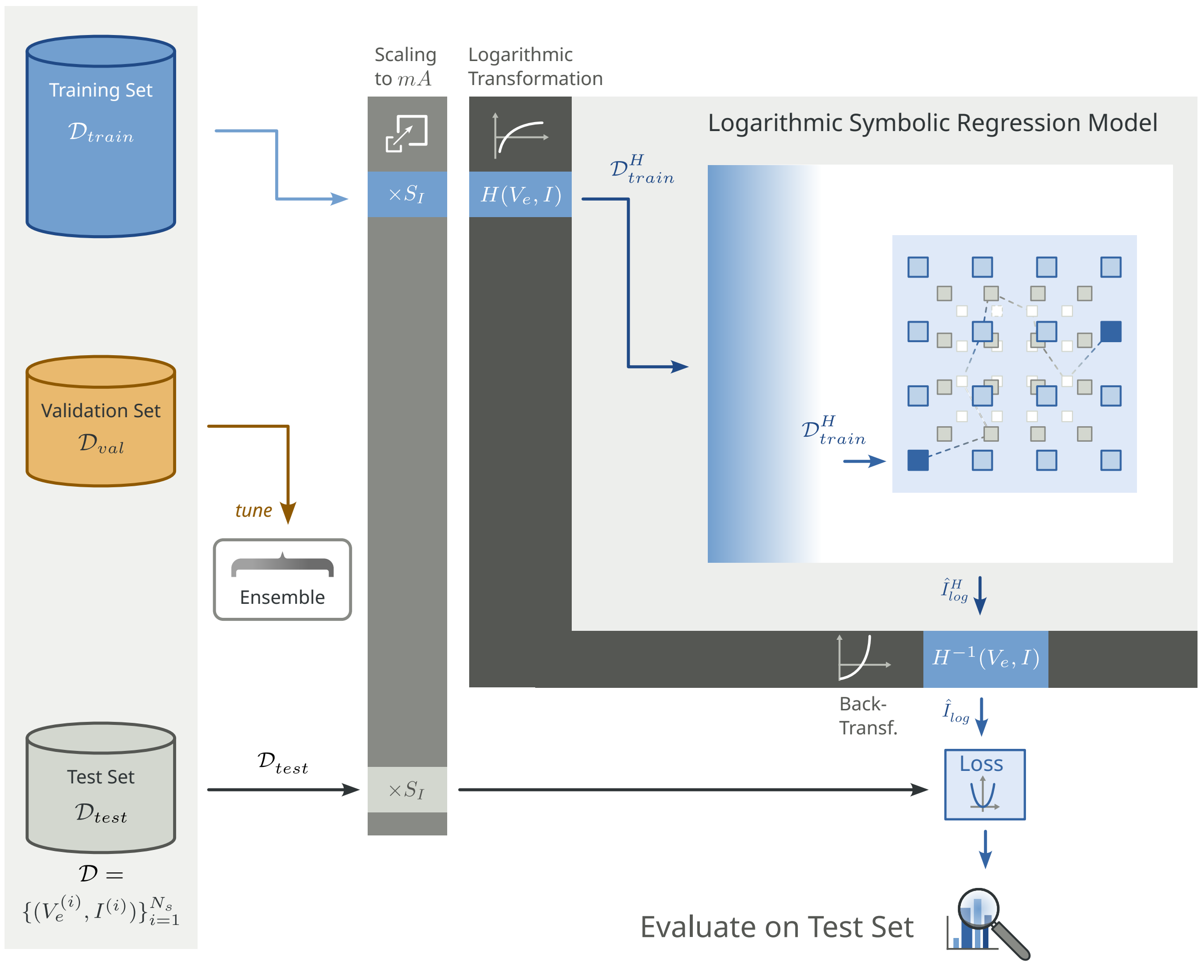


Figure 4.11: The logarithmic model generated by symbolic regression does not feature noise augmentation, as symbolic regression is less variant to the size of the training set. The validation set is only used to train the hyperparameters of the respective ensemble model.

4.4.2 Linear Current Model and Charge Model with Symbolic Regression

The linear DC drive current model training and the charge model training for the transient model are strongly related and shown in Figure 4.12. There is also high similarity to the linear neural network modeling approach. The main difference between the symbolic regression approach and the neural network approach lies in the lack of a validation set, as user-implemented early stopping is not available and no hyperparameter tuning of learning rate or hidden layer size is required. Further, the symbolic regression electrode charge models do not have a lower bound in contrast to the linear model. The dynamic range of electrode charges is narrow compared to I_{DC}, and the charge models are therefore designed to cover the entire range directly.

There are two different scale parameters S_I and S_Q, because the value ranges of DC drive current I_{DC} and electrode charges $Q_{e,j}$ of electrodes E_j with $j \in \{0, ..., N_e\}$ are distinct. In accordance to the linear neural network approach, a linear scaling factor of $S_I = 1000$ is chosen to obtain the same train set for the symbolic regression model. The charge values of the planar RFET lie in the range of $|Q_{e,j}| \in [9.5 \cdot 10^{-21}\,C, 7.7 \cdot 10^{-15}\,C]$, so that $S_Q = 10^{15}$ is selected as scaling factor, bringing electrode charges into the approximate range of $[-10\,C, 10\,C]$.

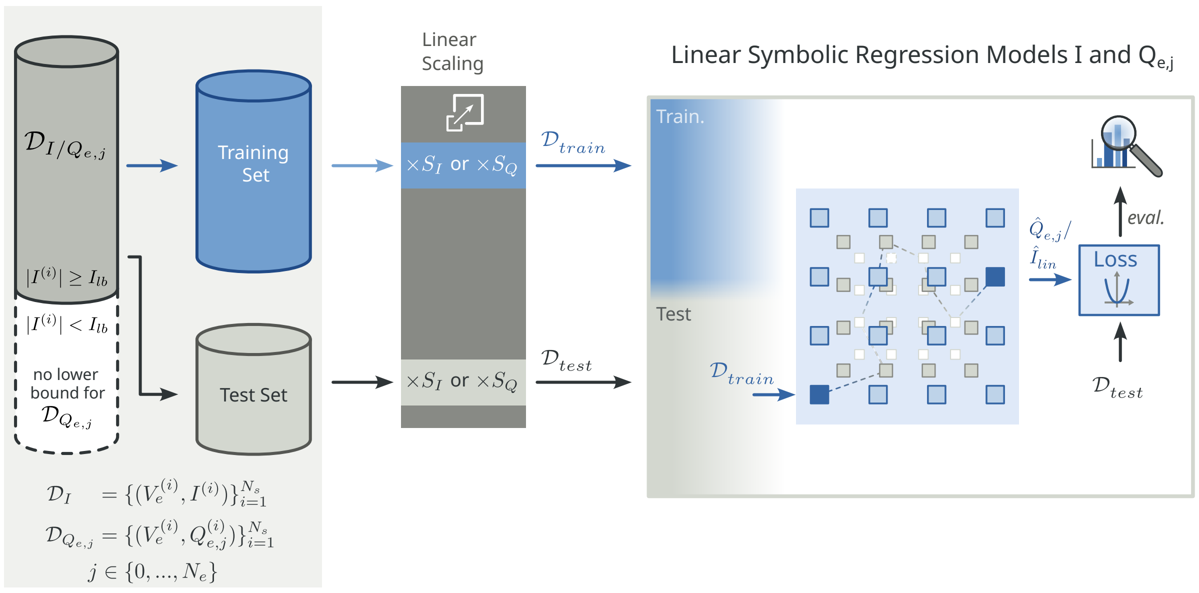

$\mathcal{D}_{I/Q_{e,j}}$
$|I^{(i)}| \geq I_{lb}$
$|I^{(i)}| < I_{lb}$
no lower bound for $\mathcal{D}_{Q_{e,j}}$
Training Set
Test Set
Linear Scaling
$\times S_I$ or $\times S_Q$
$\times S_I$ or $\times S_Q$
$\mathcal{D}_{train}$
$\mathcal{D}_{test}$
Linear Symbolic Regression Models I and $Q_{e,j}$
Train.
Test
$\mathcal{D}_{train}$
$\hat{Q}_{e,j}/\hat{I}_{lin}$
Loss
eval.
$\mathcal{D}_{test}$
$\mathcal{D}_I = \{(V_e^{(i)}, I^{(i)})\}_{i=1}^{N_s}$
$\mathcal{D}_{Q_{e,j}} = \{(V_e^{(i)}, Q_{e,j}^{(i)})\}_{i=1}^{N_s}$
$j \in \{0, ..., N_e\}$

Figure 4.12: Symbolic regression is used for one of the linear I_{DC} model candidates and for all charge models. Model building is similar in both cases, only that the I_{DC} model applies a lower bound, similar to the linear neural network-based current model.

4.5 Hyperparameter Training

Hyperparameter tuning in this work comprises two distinct sets of hyperparameters, which are tuned using different methods. Table 4.2 lists the hyperparameters along with their datatype and a respective tuning range. The hyperparameters of the linear and logarithmic neural network-based current models $H_{hl-size,\{lin,log\}}$ and $H_{LR,\{lin,log\}}$ are tuned during the respective model training, as shown in Figure 4.13. From Section 4.2.2 stem two different splits for linear and logarithmic model, which result in distinct validation- and test sets for the respective models.

The hidden layer sizes $H_{hl-size,\{lin,log\}}$ are integer parameters and therefore this work resorts to Hyperband for drive current model hyperparameter tuning. The neural network models are expected to provide acceptable accuracy with a maximum of 50 units per hidden layer. Hyperband depends on the convergence rate of the loss function, as explained in Section 2.3.1 and therefore typically benefits from a higher learning rate. However, the faster convergence of the initial epochs does not necessarily lead to the optimal solution at the maximum epoch count. Due to this interdepence the learning rate is only tuned within one decade. During early tests of the modeling approach a learning rate of $1.6 \cdot 10^{-3}$ lead to acceptable accuracy and the tuning range for $H_{LR,\{lin,log\}}$ is therefore selected to be $[5 \cdot 10^{-4}, 5 \cdot 10^{-3}]$. All current model specific hyperparameters tuned using Hyperband are sampled linearly within their respective tuning range, listed in Table 4.2. The computational budget is selected to be $\mathcal{B} = 2000$ training epochs and a factor of $\eta = 3$. The Hyperband implementation of choice in this work is provided by Keras Tuner [212], which tunes the hyperparameters on the respective validation sets of linear and logarithmic neural network model.

Table 4.2: Hyperparameters of the linear- and logarithmic neural network-based models are trained with Hyperband. Ensemble model parameters are continuous, which justifies the use of Bayesian optimization.

Tuner	Hyperparameter	Type	Range
Hyperband	$H_{hl-size,\{lin,log\}}$	Integer	$[10, 40]$
	$H_{LR,\{lin,log\}}$	Float	$[5 \cdot 10^{-4}, 5 \cdot 10^{-3}]$
Bayesian Optimization	H_{offset}	Float	$[-1, 40]$
	H_{slope}	Float	$[2, 8]$

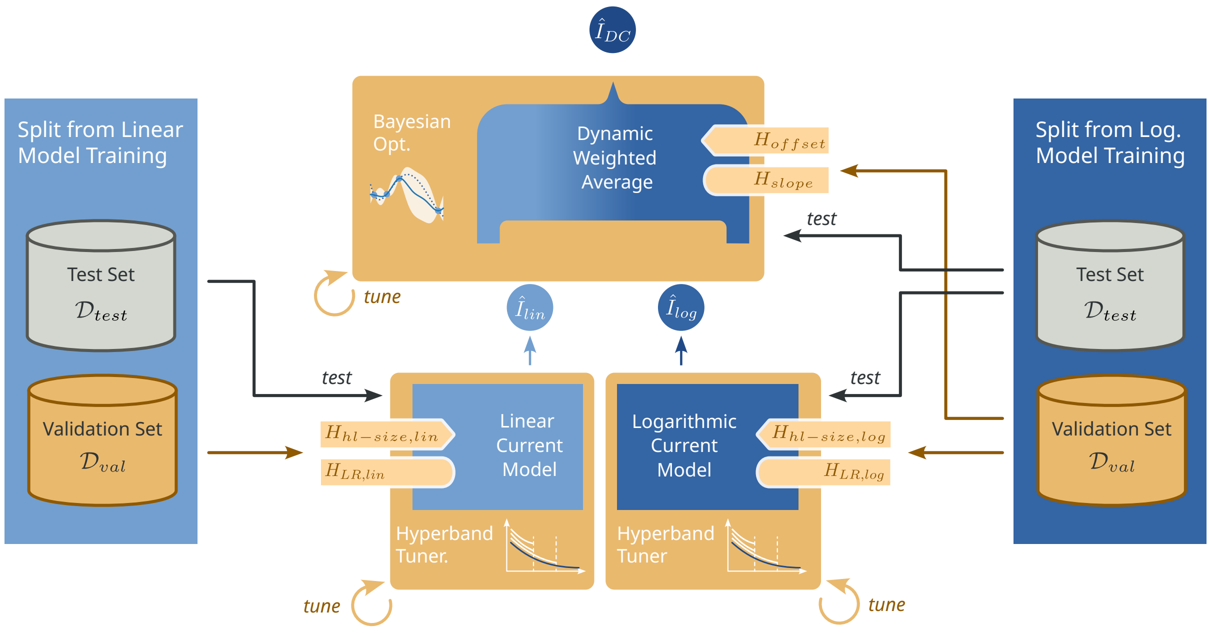


Figure 4.13: The sets for validation and testing during and after hyperparameter tuning come from the splits of the logarithmic models.

The second set $\{H_{slope}, H_{offset}\}$ constrains the dynamic weighted average function β in its slope and location of the threshold current $I_{\beta,th}$. The ensemble backend hyperparameters are trained after the respective model parameters, as the dynamic weighted average function is required to adapt to the performance of the current models which is determined by the model specific hyperparameters. Both ensemble backend hyperparameters are tuned within a continuous domain, and the unknown objective function with respect to H_{slope} and H_{offset} is expected to be smooth. Therefore, Bayesian optimization with a Gaussian process as surrogate function, is selected for hyperparameter tuning. The specific implementation used in this work is BayesSearchCV, provided by scikit-optimize [213].

Slope and offset of the ensemble backend are tuned within the continuous intervals of $H_{offset} \in [-1, 40]$ and $H_{slope} \in [2, 8]$. The corner cases drawn in Figure 4.14 show that with a sharp transition caused by a high slope, the offset shifts the threshold current $I_{\beta,th}$ between $1.33\,mA$ and $10.00\,nA$. The effect of the slope parameter has to be considered linearly, which allows the transition to vanish in favor of the linear model, as shown for $H_{offset} = 40$, $H_{slope} = 2$. Towards the maximum current of $I_{DC,max} = 1.63\,mA$, the combination of $H_{offset} = -1$, $H_{slope} = 2$ allows significant influence of the logarithmic current model into the mA range. The optimal threshold current location at the optimal slope is to be found during hyperparameter tuning, using `n_iter` $= 32$ samples.

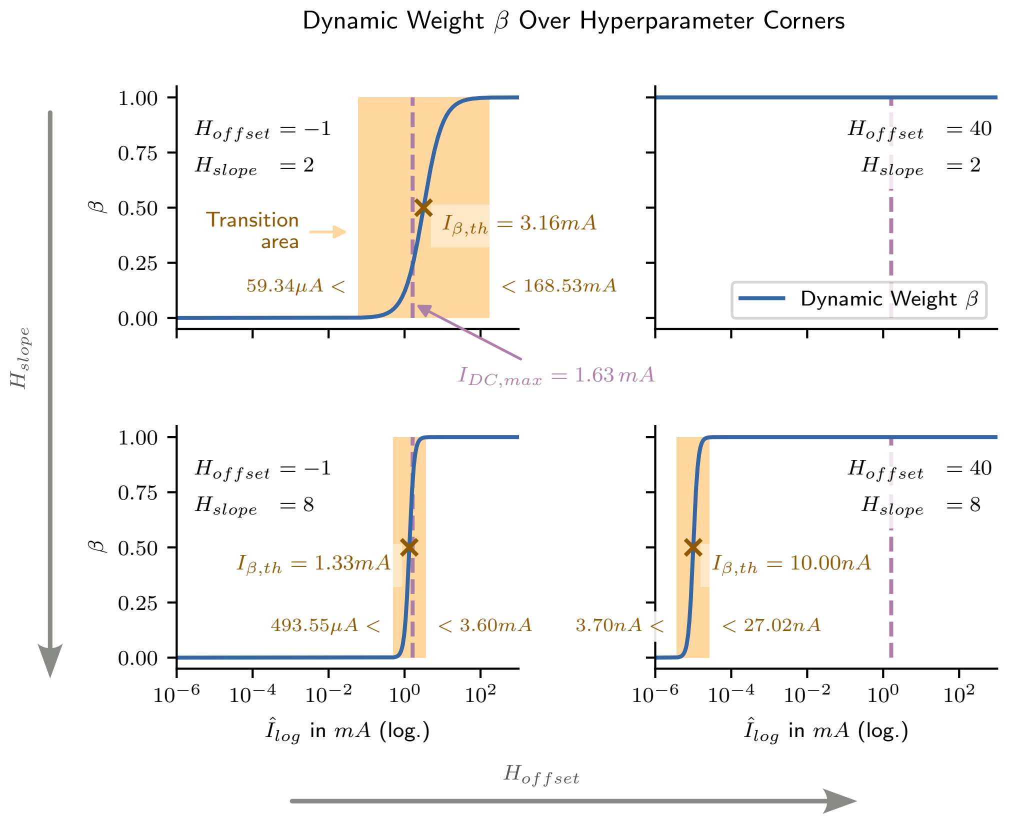


Figure 4.14: The corner cases of the dynamic weight function β cover the entire range of possible values. Width and position of the threshold point $I_{\beta,th}$ is tuned during hyperparameter training of the ensemble.

4.6 Model Deployment

In order to deploy the machine learning models to the compact model, the transformations, which are applied before training of the respective model, have to be reversed, as shown in Figure 4.15. The linear model is trained on linearly scaled data and the back transformation is

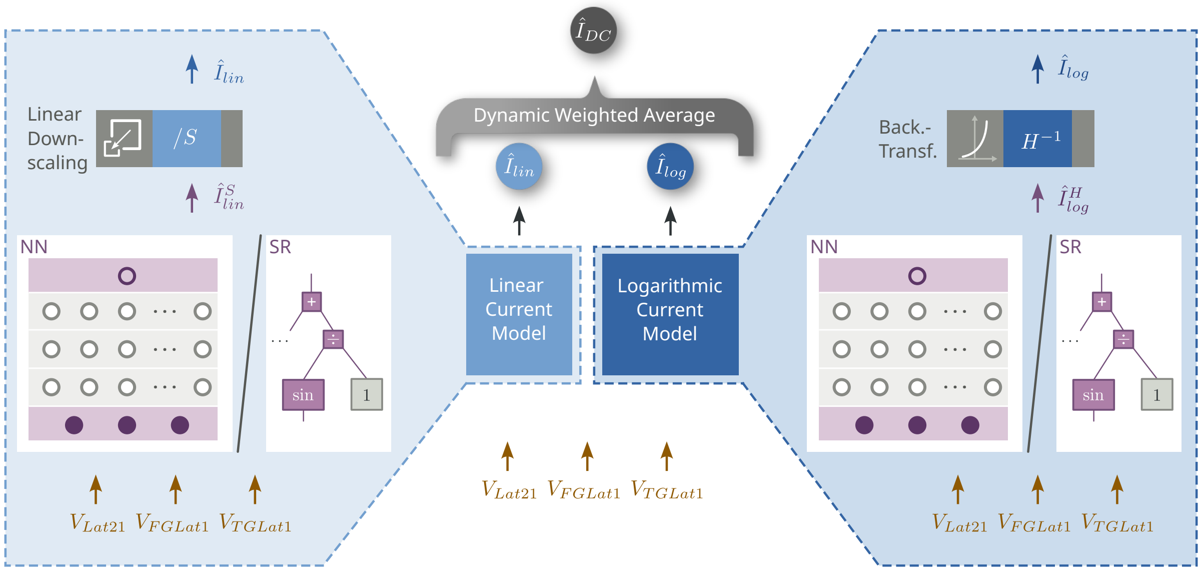


Figure 4.15: The concrete data flow within the Verilog-A model is similar between neural network- and symbolic regression-based DC models. After evaluation of the respective models, backtransformations are applied and the overall predictions are weighted by the ensemble model function.

performed by a division by the scale factor S_I. The current prediction of the linear model is therefore expressed as

$$\hat{I}_{lin} = \frac{\hat{I}^S_{lin}}{S_I}, \tag{4.12}$$

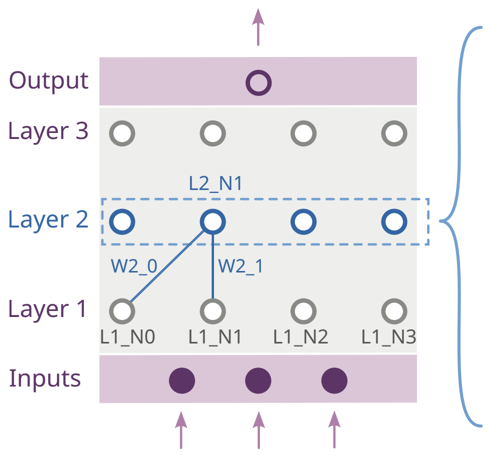
where $\hat{I}^S_{lin}$ is the linearly scaled prediction. Back transformation of the logarithmic model is performed as

$$\hat{I}_{log} = H^{-1}(\hat{I}^H_{log}) = V_{Lat21} \cdot e^{\hat{I}^H_{log}}, \tag{4.13}$$

where V_{Lat21} is evaluated from the biasing of the respective simulation step.

Implementation of the symbolic regression model means translating the graph of the selected model into symbolic equations, which can then directly be implemented in Verilog-A. This is straightforward, as QLattice provides an interface to the symbolic computing Python library SymPy [214]. The model graph object can be extracted as a string, which contains the symbolic equation and satisfies the syntax requirements of Verilog-A.

Deployment of the neural networks is more cumbersome, because Verilog-A does not support matrix multiplication, as described in Section 2.3.1. A fall-back solution in environments that are not optimized for deep learning can be the implementation of every node equations of the MLP as a standalone expression. Figure 4.16 shows an exemplary model with 3 hidden layers along with a listing of the equations implemented for layer 2. Computation of the symbolic perceptron equation Equation 2.12 for each node within a layer is performed within the scope of a `for` loop for each layer. Lines 4-9 perform the weighted sum of inputs and bias, while line 10 applies the activation function tanh. Weight and bias parameters are provided with



```
...
// --- Hidden Layer 2
for (j=0;j<4;j=j+1) begin
    L2_N_temp =
        W2_0[j]*L1_N[0] +
        W2_1[j]*L1_N[1] +
        W2_2[j]*L1_N[2] +
        W2_3[j]*L1_N[3] +
        B2[j];
    L2_N[j] = tanh(L2_N_temp);
end
// --- Hidden Layer 3
...
```

Figure 4.16: As Verilog-A does not support matrix operations, all equations of the computational units have to be implemented separately.

4 significant digits. The computational effort for a single solution using the neural network approach depends on the structure of the model, which is predetermined before training. It is therefore possible to target a specific computational budget and constrain a model structure accordingly. The operation count for the MLP implementation with $H_{hl-size}$ neurons per hidden layer and 3 hidden layers can be calculated as

$$N_{ops,nn} = N_{ops,hl1} + N_{ops,hl2} + N_{ops,hl3} + N_{ops,out}, \tag{4.14}$$

$$\begin{aligned} &= \overbrace{H_{hl-size}}^{\text{current layer nodes}} \cdot \Big(\big(\overbrace{1}^{\text{ADD}} + \overbrace{1}^{\text{MUL}} \big) \cdot \overbrace{|V_e|}^{\text{previous layer nodes}} + \overbrace{1}^{\text{activation func.}} \Big) \\ &\quad + H_{hl-size} \cdot (2 \cdot H_{hl-size} + 1) \\ &\quad + H_{hl-size} \cdot (2 \cdot H_{hl-size} + 1) \\ &\quad + 2 \cdot H_{hl-size} + 2, \end{aligned} \tag{4.15}$$

for example. Here, each node in its respective layer performs one addition and one multiplication for each preceding layer, or in case of the first hidden layer each input from V_e, along with a bias constant and the respective activation function. The computational effort required for the entire solution of a time step, however, also depends on the number of required iterative steps that the solver requires to converge to a solution.

Short Summary

This chapter describes the machine learning-based methods to transform a data set, e. g. a table model, into a equation based compact model. The DC drive current, depending on the electrode voltages, is represented by an ensemble model of a linear and a logarithmic prediction. This ensemble model is implemented with neural networks and, as a reference, in symbolic regression. As the dynamic range of electrode charges is observed to be more restricted for the planar RFET, every electrode charge is modeled with a single, individual model. The ground truth distribution of the charge samples is expected to be less complex to model than the drive current, so that all charge models are formed by symbolic regression. This chapter further shows the training, tuning and the implementation of the proposed models.

5 Evaluation

Two perspectives are considered in this evaluation: The device modeling perspective typically seeks accuracy and low modeling effort, saving computational- and human resources. At the core of the evaluation are therefore the accuracy that the presented modeling approaches provide and the simulation times of the data generation methods. In comparison with the TCAD-based data generation, resources for training the machine learning models can be neglected. The perspective of the circuit designer further demands, aside from accuracy, high computational performance of the device model in order to conduct large scale circuit simulation. This chapter therefore analyzes the simulation times for the proposed models in DC and transient simulation of digital cell.

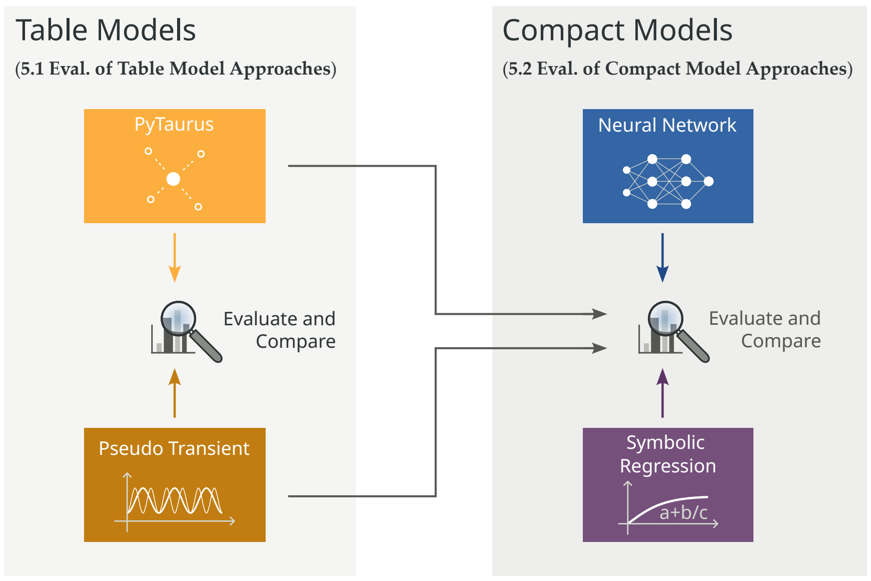


Figure 5.1: Evaluation is first performed for the table models, and then for the compact models. The performance of the compact model approaches is then compared to the table models.

From the two table model generation approaches, factorial and pseudo transient, and two machine learning approaches, neural network and symbolic regression, stem 2 table model candidates and 4 compact model candidates. The candidates are referred to with nomenclature $<\text{ARCHITECTURE}>_{<datasource>}$. The model candidates are TAB_F, TAB_P, NN_F, NN_P, SR_F, SR_P, emphasized in the shown diagrams by the indicated color scheme.

The evaluation of the proposed methods is tailored to the planar RFET as DUT and targets digital cell simulation on circuit level. The two table model approaches, the PyTaurus table model simulation cluster proposed in Section 3.2 and the pseudo transient method proposed in

Section 3.3, are evaluated with respect to their performance and modeling effort in Section 5.1, as shown in Figure 5.1. The subsequent step after obtaining the table model is to transform the underlying database into compact models, as described in Chapter 4. The respective accuracy and performance of the compact modeling approaches are then compared to the table models and evaluated in Section 5.2.

5.1 Evaluation of Table Model Approaches

This section evaluates the generation of a table model using PyTaurus against the pseudo transient method. Firstly, the required resources and the structure of the resulting data sets are evaluated in Section 5.1.1 and Section 5.1.2. Both models are then used as table models and their circuit level performance is evaluated in Section 5.1.3 in order to obtain baseline models to which the compact models are then compared in Section 5.2.

Each data generation method is set up with the planar RFET technology model. The lateral channel electrode Lat1 is used as reference electrode and the remaining electrode voltages cover the range of $V_{e,\{Lat2,FG,TG\}} \in [-V_{DD} - 0.2\,V, +V_{DD} + 0.2\,V]$, with $V_{DD} = 1.4\,V$, to account for transient overshoots. The BG electrode potential is set to the model internal reference potential, i. e. the potential of Lat1. Electrical quantities of interest are drive current $I_{DC} = I_{Lat2}$ and electrode charges $Q_{\{Lat1,Lat2,BG,FG,TG\}}$.

5.1.1 Generation Flow: PyTaurus

The configuration of the PyTaurus simulation flow requires the selection of a sweep electrode. The choice of reference electrode is arbitrary, but the choice of sweep electrode influences circuit simulation accuracy, as the sweep electrode typically features finer granularity than the remaining electrode voltages within the factorial simulation design. For the planar RFET it is practical to have the independent gates FG and TG feature similar granularity, in order to achieve similar accuracy when using them interchangeably as signal input in digital cells. In this work, the sweep variable is therefore selected to be Lat2. The gate voltage steps are arranged in a regular grid with equal $S_{\{FG,TG\}} = \{-1.6\,V, -1.4\,V, ..., 1.6\,V\}$ and therefore $|S_{\{FG,TG\}}| = 17$. A voltage step size of $200\,mV$ is deemed an acceptable trade off between model accuracy and data generation effort. The full factorial simulation design with respect to the gates therefore leads to a simulation deck of

$$\begin{aligned} S_{deck} =& S_{Lat1} \times S_{FG} \times S_{TG} \qquad (5.1)\\ =&\{(0\,, -1.6\,V, -1.6\,V), \qquad (5.2)\\ &(0\,V, -1.6\,V, -1.4\,V),\\ &(0\,V, -1.6\,V, -1.2\,V),\\ &...\} \end{aligned}$$

with $S_{Lat1} = \{0\,V\}$. The size of the simulation deck after generation and planning step, as described in Section 3.2.1 is therefore

$$|S_{deck}| = |S_{Lat1}| \cdot |S_{FG}| \cdot |S_{TG}| = 289. \qquad (5.3)$$

The step size of the quasistationary solver for the sweep electrode Lat1 is constrained to a maximum of $\Delta V_{Lat1} = 50\,mV$, and allowed to decrease to $\Delta V_{Lat1} = 1\,nV$ for bias regions with convergence difficulties.

The simulation run is conducted on a simulation cluster, as described in Section 3.2.2. As typical for technology simulation, not all simulations converge in the originally created simulation run. Figure 5.2 shows that out of the 289 scheduled simulations only 80% finish right away The refinement function of PyTaurus, as described in Section 3.2.3, can then be employed. In order to bring the remaining 59 simulations to convergence, the first refinement attempt increases the allowed Newton-iterations of the solver from 35 to 60, which provides convergence for further 26 simulations. After analyzing the maximum errors of the failed simulations using the mesh plot feature introduced in Section 3.2.3, the second refinement increases the density of the simulation mesh. The 3 following refinement steps alter the solver step increment parameter within $[1.15, 1.21]$ to provide different seeding to the numeric simulation, which eventually leads to convergence for the remaining $9 + 1 + 1$ simulations. Since all simulations from the simulation deck converge after refinement, all data points of the factorial simulation grid are provided and interpolation steps are not required.

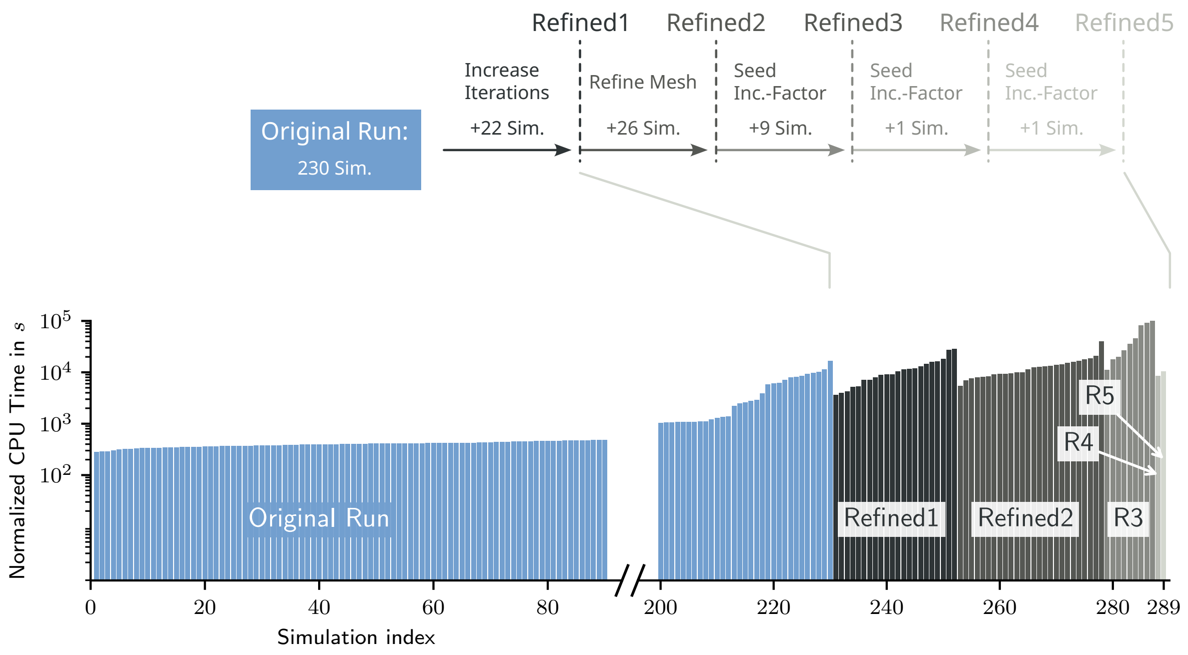


Figure 5.2: Out of the simulation deck of size $17 \cdot 17 = 289$, only 230 simulations finish successfully in the first attempt. The remaining 59 simulations are gradually refined within 5 refinement steps.

The simulations are distributed over various hosts in the simulation cluster, which feature different computational resources, and therefore the CPU times of the respective simulations are not directly comparable. In this work the CPU times are therefore normalized to the computational resources of a specific cluster runner with an Intel i9-9900K CPU and $64\,GB$ memory. The normalization is achieved by computing a performance score from randomly selecting 10 simulations of the original run and executing these simulations on all cluster hosts.

The respective CPU time $t_{CPU,r,s}$ on each runner r for a simulation s is then referenced to the respective CPU time $t_{CPU,ref,s}$ of the reference runner, to obtain a performance score

$$\eta_r = \frac{1}{10} \sum_{s=1}^{s=10} \frac{t_{CPU,r,s}}{t_{CPU,ref,s}} \tag{5.4}$$

for each runner. The normalized CPU time $t_{CPU,norm,r,s}$ of every simulation s is then obtained through

$$t_{CPU,norm,r,s} = \frac{t_{CPU,r,s}}{\eta_r}. \tag{5.5}$$

All simulation times in this work are either CPU times of the reference runner or normalized, to be directly comparable.

As a result, the total CPU time of all completed simulations is $1.23 \cdot 10^6\, s$ and the simulation with the overall longest CPU time takes $9.4 \cdot 10^4 s$ to run to completion. The outcome of the completed simulation run is a total of $|\mathcal{D}_{fact}| = 58,474$ data points, each representing drive current and all electrode charges of the respective bias points.

5.1.2 Generation Flow: Pseudo Transient Approach

In this section the model candidate generated through pseudo transient simulation TAB_P is evaluated. Unlike the factorial simulation approach followed by PyTaurus, where the novelties lie in usability and practicality, the pseudo transient simulation is a fundamentally new concept. Therefore, this section ends with a study about the influence of the basic parameterization on the results of the pseudo transient method.

While PyTaurus generates table models from a factorial simulation setup, where a regular grid can be constrained, the pseudo transient approach results in variable granularity within the bias space $V_e \in \mathbb{R}^{N_e}$. For sake of comparability with the PyTaurus table model, the pseudo transient approach is constrained for the same divisions per dimension and therefore $m = 17$ is specified. The order of electrodes, from lowest to highest frequency, is TG, FG, Lat1, to account for the high capacitance estimation of TG, due to the 50% overlap at Lat1/Lat2. To match the size of the PyTaurus generated data set $|\mathcal{D}_{fact}| = 58,474$, the required oversampling factor is obtained using Equation 3.40 and Equation 3.41 in form of

$$N_{os} = \lceil \frac{|3 \cdot \mathcal{D}_{fact}|}{6 \cdot m^{N_e - 1}} \rceil = \lceil \frac{58,474}{6 \cdot 17^2} \rceil = \lceil 33.77 \rceil = 102, \tag{5.6}$$

to obtain a positive integer, as specified in Section 3.3.3. The resulting count of extracted samples, i. e. the amount of samples in the table model $\mathcal{D}_{pstrans}$, is then $|\mathcal{D}_{pstrans}| = 58,956$, as described by Equation 3.40 and the total number of samples of the pseudo transient simulation from $t \in [0, T_{sim}]$ is $176,869$.

With the selection of $T_{sim} = 100\, s$, which empirically showed sufficiently low transients in test runs with the planar RFET, the selection of basic parameters $V_a = 1.6\, V$, $N_{os} = 108$, $m = 17$ and $T_{sim} = 100\, s$ is complete.

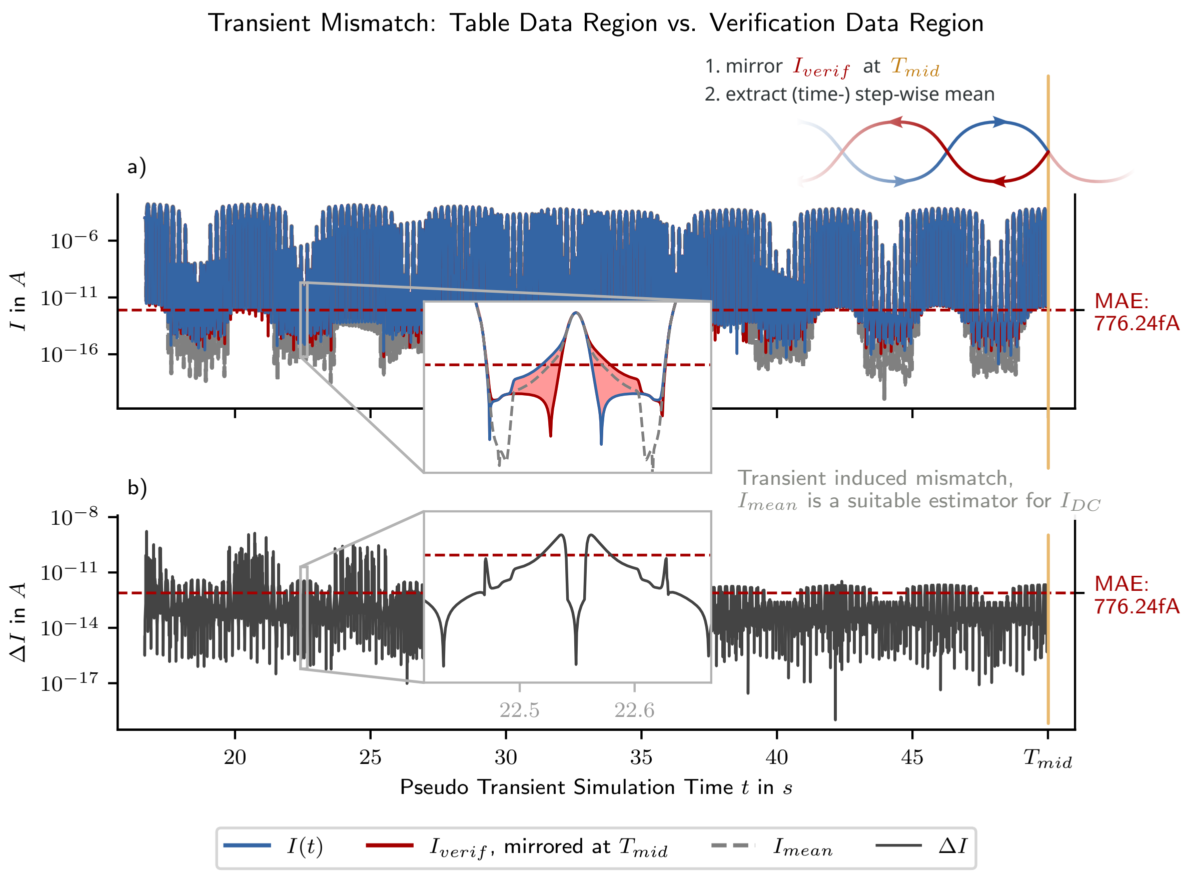


Figure 5.3: The transient mismatch ΔI becomes evident when plotting the difference between the table data region and the (mirrored) verification data region. The sample-wise arithmetic mean between the currents in both regions features a attenuated transient component [207].

Performance and Limitations of the Table Model Candidate

Since the data points are extracted from transient simulation, it is necessary to check the samples for transient error to decide if intrinsic accuracy, characterized by the size of transient mismatch, is acceptable and the data set can be considered for a device model. Figure 5.3 a) displays the magnitude of the simulated current of the table model region $S_{pt,table}$ (blue), overlaid with the magnitude of the mirrored simulated current from the verification region $S_{pt,verif}$ (red), in order to provide the similar electrode bias $V_e(t)$ for both traces at each point in simulation time, respectively. In addition to both raw simulation data traces, the respective mean absolute current I_{mean} is drawn for each point in time, which serves as the estimator of a corresponding DC current I_{DC}, as explained in Section 3.3.2. The respective magnitude of the transient mismatch ΔI can be extracted from diagram Figure 5.3 b). The key figure for intrinsic accuracy is the MAE over all samples, which is found to be $\varepsilon_{transient} = 7.76 \cdot 10^{-13}\,A$. The individual $\Delta I(t)$ of every sample shows high fluctuation between $\Delta I_{min} = 2 \cdot 10^{-18}\,A$ and $\Delta I_{max} = 2.04 \cdot 10^{-9}\,A$, with the majority of samples, 80%, below the mean. As $\varepsilon_{transient}$ lies below the minimum inverter current $I_{INV,min} = 2.12 \cdot 10^{-12}\,A$ from Section 4.2.3 it is reasonable to accept this pseudo transient simulation result as a model candidate and transform I_{mean} into a table model.

The variable granularity of the pseudo transient data set in the time domain is emphasized in Figure 5.4, where $V_{e,1}$ changes with the lowest frequency $f_{e,1} = \frac{1.5}{T_{sim}}$ and $V_{e,3}$ changes with the highest frequency $f_{e,3} = \frac{1.5 \cdot m^2}{T_{sim}}$. The time steps are uniform, which causes the voltage resolution to be more coarse around $V_{e,i} = 0\,V$. The voltage resolution increases towards the turning points of the harmonic functions.

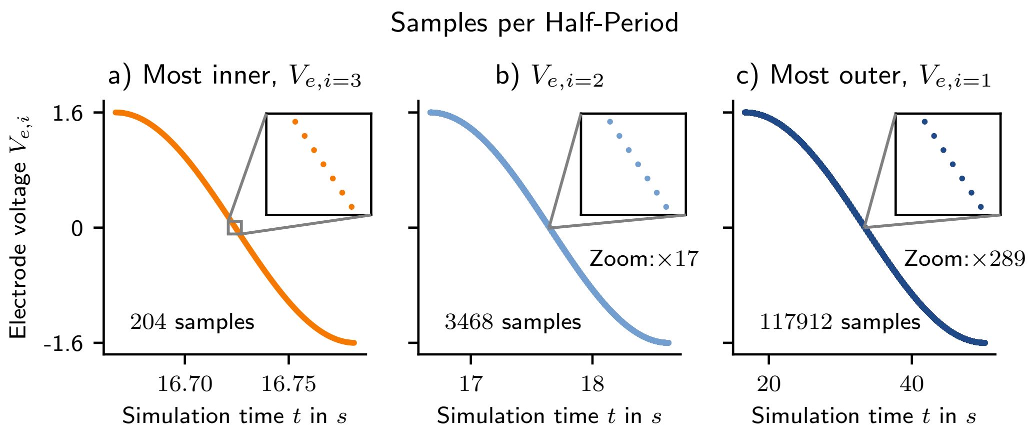


Figure 5.4: From the inner to the outer electrode of the nested setup, the number of samples per half-period increases by the nesting factor $m = 17$.

After forming the table model through sorting by the inverted sequence of electrodes, i. e. $V_{e,3}, V_{e,2}, V_{e,1}$, the granularity of the respective dimensions is screened in the resolution histogram in Figure 5.5. In order to analyze the distribution of granularity within all dimensions, Figure 5.5 shows the step sized, i. e. the resolution, of each dimension. The step sizes are accumulated and divided by the average number of divisions m in the previous dimension $V_{e,i+1}$ for normalization. This normalization step allows to compare the resolution of all dimensions by cancelling out the increase in samples, which is introduced by the parameter nesting.

Screening the described normalized resolution of samples in each dimension then confirms that although the number of divisions matches the full factorial PyTaurus approach, the resolution of $V_{e,1}$ and $V_{e,2}$ varies between $31.33\,\mu V$ and $585.49\,mV$, as shown in Table 5.1. As $V_{e,3}$ is the most outer dimension in the table model and creates $2 \cdot N_{os} + 1 = 205$ samples. The respective voltage steps, i. e. the resolution, are shown in the histogram of Equation 3.39. The voltage steps of $V_{e,3}$ are fine granular due to the high N_{os} and show a mean resolution of $16\,mV$. Accumulating the step sizes throughout the remaining dimensions $V_{e,2}$ and $V_{e,1}$, increases the respective step counts by m^i for $V_{e,i}$ and $i < 3$. In order to compare the distribution of the respective step sizes, it is therefore suitable to normalize the histogram accordingly. After normalization, the step sizes show equal distribution with mean resolution of $200\,mV$, corresponding to the uniform resolution of the iso lines in the factorial table model. This shows, that control of resolution is only partially possible as the granularity of the table model depends on the respective bias region and can either be finer or coarser than a corresponding factorial design. The mean resolution, however, can be influenced by m and N_{os}.

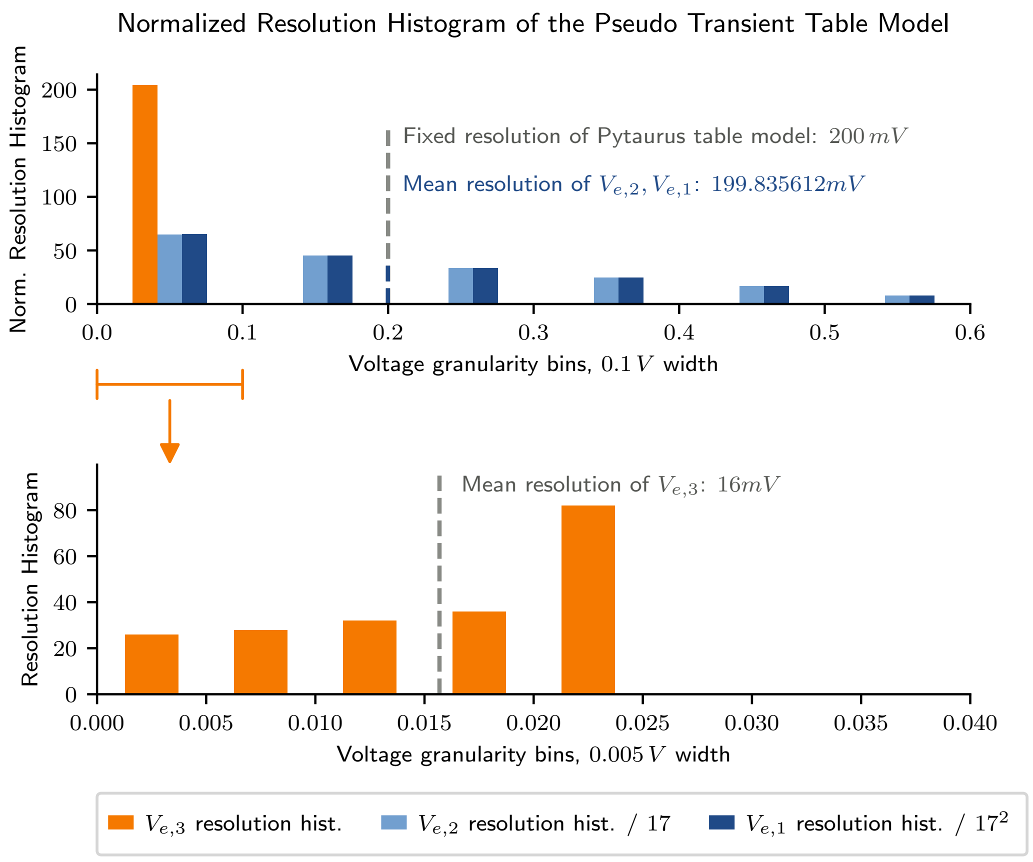


Figure 5.5: The mean voltage step of the pseudo transient data set is equal to the factorial model ($200\,mV$), but the individual steps vary throughout the bias space. The granularity of the most outer function of the nested setup $V_{e,3}$ is highest.

Table 5.1 further shows that the generation time of the pseudo transient table model is a factor of $2.5\times$ higher than the generation time of the factorial model through PyTaurus. This can be attributed to the fact that the pseudo transient simulation computes $3\times$ the sample count of the actually extracted data, expressed by Equation 3.41. It is, however, possible to set up a pseudo transient simulation using cosine as fundamental function, which only simulates the table data range, as explained in Section 3.3. This reduces the sample count by a factor of $3\times$, which, in extrapolation, shifts the advantage of computational efficiency from the factorial method to the pseudo transient method for the shown simulation environment and technology model. In general the relative performance between both methods depends substantially on the respective performance of the employed quasistationary solver and transient solver.

An important finding from this evaluation is that even though the resolution of a pseudo transient table model can be influenced, and a mean resolution can be constrained, it is not possible to set up continuous iso lines for specific dimensions, which exist in all divisions of that respective dimension. Where the full factorial simulation design of PyTaurus can explicitly

Table 5.1: The proposed pseudo transient data set is not structured as regularly as the factorial set. A higher generation time is observed, if the the entire range $t \in [0\,s, T_{sim}]$ is simulated.

Characteristic	Factorial	Pseudo Transient
Resolution per (fact.) dimension ($V_{e,\{1,2\}}$)	$200\,mV$	$[31.33\,\mu V, 585.49\,mV]$, mean: $199.84\,mV$
Resolution of sweep dimension ($V_{e,3}$)	$[11.82\,pV, 160\,mV]$, mean: $15.89\,mV$	$[189.72\,\mu V, 24.64\,mV]$, mean: $15.69\,mV$
Sample count	58,474	Table: 58,956 Total: 176,869
Generation time (CPU)	$1.23 \cdot 10^6\,s$	$3.10 \cdot 10^6\,s$

provide data points for the electrode voltages $V_{e,i} = V_{DD}$, the pseudo transient can only target a mean resolution. Introducing arbitrary electrode voltages into the simulation design is not supported. Further, the complete pseudo transient approach generates $3\times$ the sample count of a factorial and quasistationary approach, when executed as described in Section 3.3. The generic approach with sine as the fundamental function and a verification range can be reduced in size: If the simulation environment supports cosine voltage sources and a verification range is not required, due to low capacitance of the DUT, the simulation time can be reduced to $t \in [0, T_{sim}/3]$, reducing the number computed samples accordingly.

Influence of Basic Parameters

After evaluating transient mismatch and table model structure from the pseudo transient generation approach, the influence of the basic parameters is analyzed in this section. The most important parameters to constrain the pseudo transient simulation method are the nesting factor m, the oversampling factor N_{os} and the simulation time T_{sim}, as described in Section 3.3. The resources of interest are the time t_{CPU}, the CPU time which the respective simulation requires, and quality of the data set, measured in the transient mismatch.

For evaluation of the respective influences a basic example of $m = 9$, $N_{os} = 10$ and $T_{sim} = 1\,s$ is assumed, which is complex enough to deliver an accurate table model of the planar RFET but does not feature the full complexity of the table model candidate TAB$_{\text{P}}$. From the basic example on, the parameters m, N_{os} and T_{sim} are varied and the respective change in t_{CPU} and transient mismatch are shown in Figure 5.6.

The variation of m in Figure 5.6 a) shows that a higher nesting factor increases the transient mismatch and leads to higher computational effort manifested through rising CPU time. Both observed effects go approximately linear with m. The correlation of nesting parameter m and transient mismatch $\varepsilon_{transient}$ matches with the expectation, as the excitation frequencies of all $V_{e,\{2,\ldots\}}$, which drive the total currents with DC and transient component, are proportional to m.

In turn, increasing the oversampling factor N_{os} in Figure 5.6 b) leads to a decrease in mean transient mismatch, proportional to $\frac{1}{N_{os}}$. This trend supports the assumption that the mean transient mismatch is dominated by minor regions of the bias space, which feature high capacitive current. However, increasing only N_{os} to reduce $\varepsilon_{transient}$ is not recommended, if the relevant electrical parameters that cause the transient component, such as frequency and

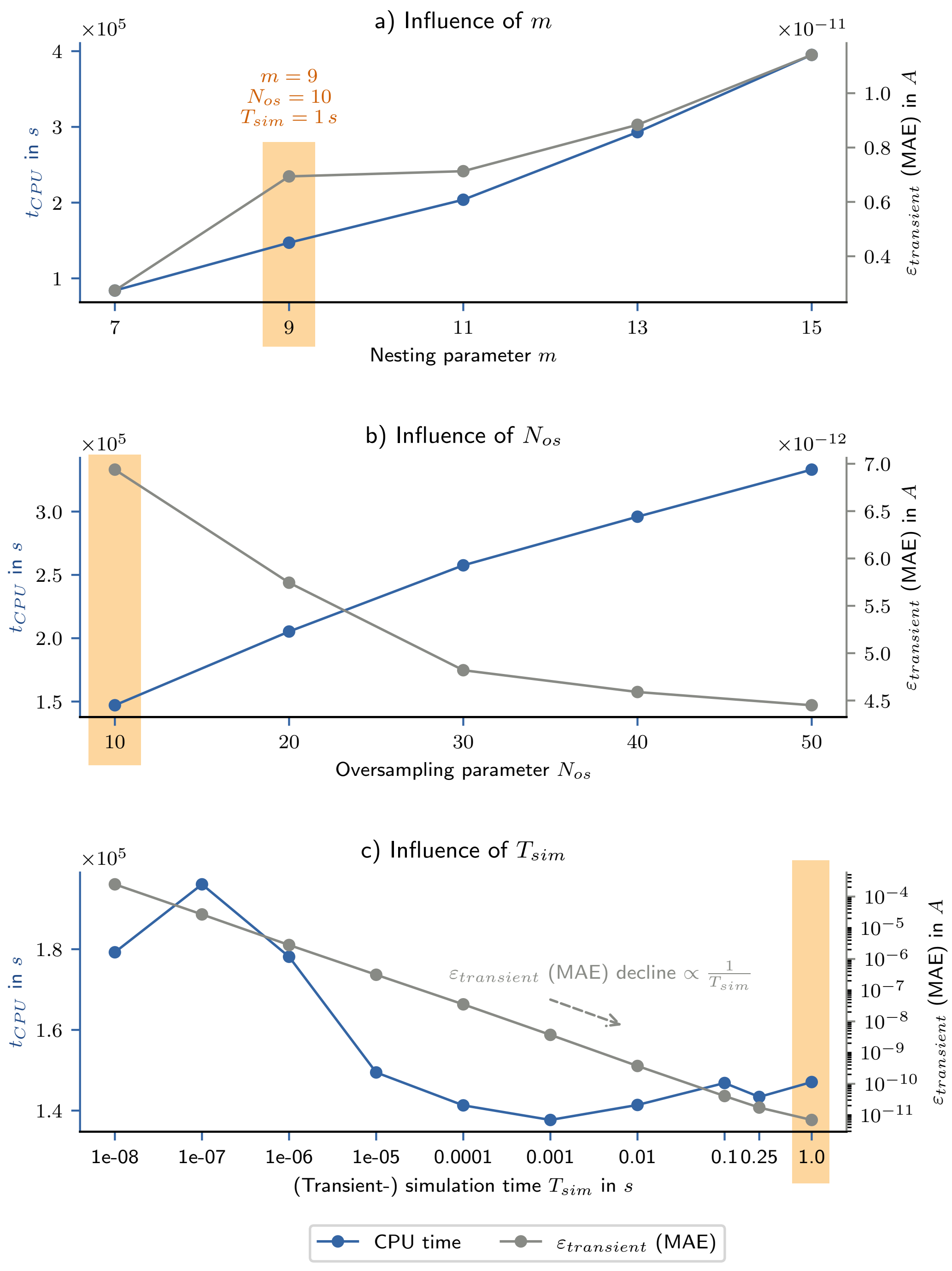


Figure 5.6: The influence of the parameters m, N_{os} and T_{sim} are analyzed [207]. Higher nesting without increasing T_{sim} in a) leads to higher generation time and higher transient components. Oversampling relates about linearly to generation time and transient mismatch in b). The analysis c) confirms that a larger T_{sim} does not necessarily require more resources, but causes a $\approx \frac{1}{T_{sim}}$ drop of MAE.

time base remain unchanged. Only the change in sample distribution leads to a change in the mean-related figure of merit. The CPU time for simulation increases with N_{os}, as the number of inserted time steps to be solved increases.

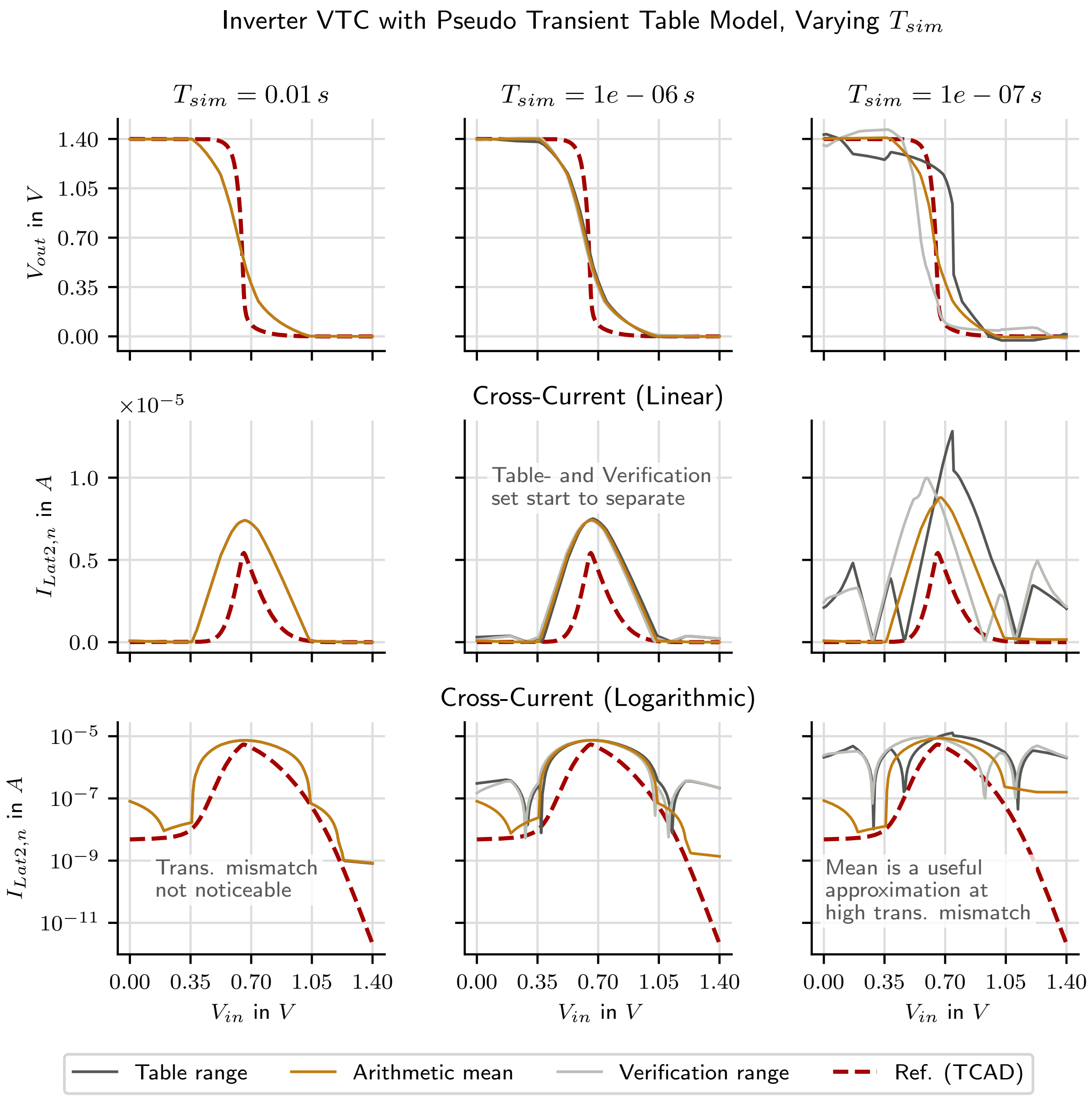


Figure 5.7: The transient suppression method is evaluated by comparing 3 simulations with a decreasing time base. The arithmetic mean remains a good estimate even in situations where a single sided (table data range or verification range) data export leads to heavy distortions in circuit simulation.

A possibility to influence only the transient mismatch leads through T_{sim}: The increase of T_{sim} leads to a decrease of $\varepsilon_{transient}$ with a slope approximately proportional to $\frac{1}{T_{sim}}$. The inverse proportionality of the transient mismatch is consistent with the simplification of Section 3.3.2, where the example of a linear capacitance is introduced to describe the occurrence and cancellation of transient currents. The magnitude of the current of a linear capacitive element $I_C(t) = C \cdot \frac{dV}{dt}$, or in the frequency domain $I_C(j\omega) = V_c \cdot j\omega C$, is proportional to $\frac{1}{dt}$ or $\omega \propto \frac{1}{T}$. The declining presence of transient currents in Section 3.3.2 c) therefore approximates the behavior of the described linear elements. With respect to CPU time a systematic correlation with T_{sim} is not depicted, rendering the increase of T_{sim} a suitable measure to increase accuracy of the data points without penalty for the observed range. However, as transient simulations require step size constraints, which are described in Section 2.2.1, further increasing T_{sim} with constant step size constraints eventually leads to more time steps to be solved and the CPU time is expected to increase.

The effect of variation of T_{sim} in combination with the proposed transient suppression strategy can be seen in Figure 5.7. Three pseudo transient simulations are conducted with the technology model of the planar RFET, using $T_{sim} \in \{10^{-2}\,s, 10^{-6}\,s, 10^{-7}\,s\}$. The resulting table models, one for table range, verification range and the arithmetic mean current, are employed in a VTC simulation of a CMOS inverter. For $T_{sim} = 10^{-2}\,s$ table range (left), verification range (right) and the mean current, the estimator for the dc drive current, are visually identical, as the transient components of the table model data is low. With increasing T_{sim}, the separation between the three extracted table models increases substantially. However, it can be clearly seen that in the given example, the arithmetic mean remains a useable estimator for the DC characteristic of the DUT.

5.1.3 Circuit Level Evaluation

In the final step of table model evaluation, circuit simulation results, which instantiates $\mathsf{TAB_P}$ and $\mathsf{TAB_F}$, are analyzed in detail. This evaluation focuses on the performance in fundamental digital cells *INV* and *XOR*. A perfectly accurate table model from TCAD data is expected correlate exactly with a separately conducted simulation of the identical circuits in TCAD. Therefore, model accuracy in this work references to mixed-mode simulation of the respective circuit in TCAD.

Inverter - DC and Transient Analysis

The inverter cell is a suitable candidate for basic model evaluation, because it features a low transistor count on one hand, while still being able to expose convergence issues and inaccuracies, on the other hand. A single voltage (V_{out}) and a single current (cross current $I_{Lat2,n}$) characterize the transition. The VTC of the inverter in Figure 5.8 is drawn for the two table models, the $\mathsf{TAB_F}$ and $\mathsf{TAB_P}$, along with the reference from TCAD mixed-mode simulation. The switching threshold is predicted accurately by both table models. At the onset of switching, however, both table models deviate from the reference. The reason for these deviations become evident with the cross current $I_{Lat2,n}$: The accuracy of the factorial model is tied to a regular grid with $200\,mV$, which is the voltage step size of the FG voltage $V_{FG} = V_{in}$. Between the table data points, where the factorial model correlates closely with the reference, the linear interpolation of the table model causes deviations from the reference current.

The same principle holds true for the pseudo transient model. Here, however, the distance of the iso lines with respect to V_{FG} is not uniform, so that the width of the interpolation artifacts vary with up to approximately $400\,mV$ for the inverter VTC. The low granularity of the pseudo transient data set with respect to V_{in} is especially high before and after the transition region, which is why at these points the largest deviation from the reference VTC can be seen. Overall, the higher granularity of the factorial table model leads to better correlation with the reference in the transition region. The output swing, however shows a better fit with the pseudo transient

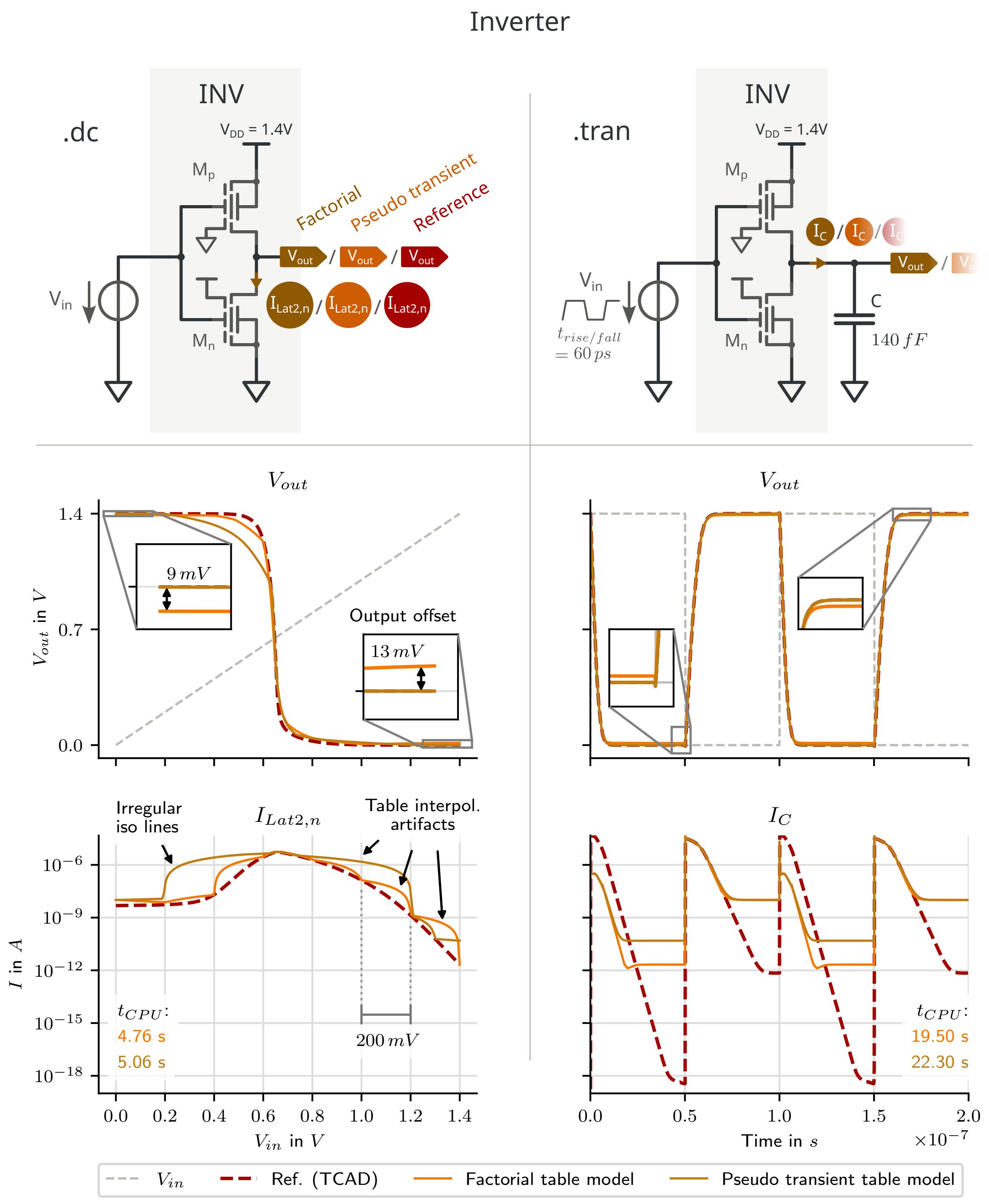


Figure 5.8: Inverter simulation featuring the table models exposes the artifacts that come from linear interpolation. Although the factorial model has iso lines at V_{DD}, the output swing is slightly reduced.

model, while the factorial model shows output offsets of $9\,mV/13\,mV$ for high/low output. Although, V_{DD} is not an iso line of the pseudo transient model, the granularity of the data set increases towards $V_{in} = 0\,V$ and $V_{in} = V_{DD}$, while the factorial model shows more uniform granularity.

Further, Figure 5.8 shows a transient analysis of the inverter with an input pulse of $t_{rise} = t_{fall} = 0.06\,ns$, and an output capacitor of $140\,fF$. In this work, all transition times are given for a transition between 20% and 80% of V_{DD}. Again, the factorial table model shows output offset, while the pseudo transient model has full output swing. The transition region is matched closely by both table models. However, the current trace shows substantial lack of dynamic range in both table models, which fail to predict currents below $1\,pA$ (factorial model) and $10\,pA$ (pseudo transient model), while the reference reaches a minimum absolute current of $3.52 \cdot 10^{-19}\,A$. The falling transition, with a rising edge of V_{in}, the table models both show higher deviation than for the rising transition. Especially the high mismatch in pull-down operation implies that the table models show better accuracy of the p-type behavior than the n-type behavior in the inspected bias region. The slightly reduced output swing of $\mathsf{TAB_F}$ is exposed, similarly to the VTC simulation.

DC simulations with both table models compute 200 equidistant simulation steps, each and the resulting simulation time in Cadence SPECTRE is similar for both models, with a minimal advantage of $\mathsf{TAB_F}$. For the DC simulation in Figure 5.11, $\mathsf{TAB_P}$ requires 6% more CPU time, compared to the factorial model. The sample count is left unconstrained for the transient simulation, resulting in $132/134$ samples for pseudo transient/factorial model. Computing a similar sample count, the pseudo transient model again exceeds the CPU time of the factorial model by 14%. Over all, the pseudo transient model shows up to 14% higher simulation time at a close to equal table size with less than 1% difference amount of samples, and an equal/similar amount of computed simulation steps for DC/transient simulation. This trend indicates that the structure of the factorial table model contributes to a slightly faster simulator performance.

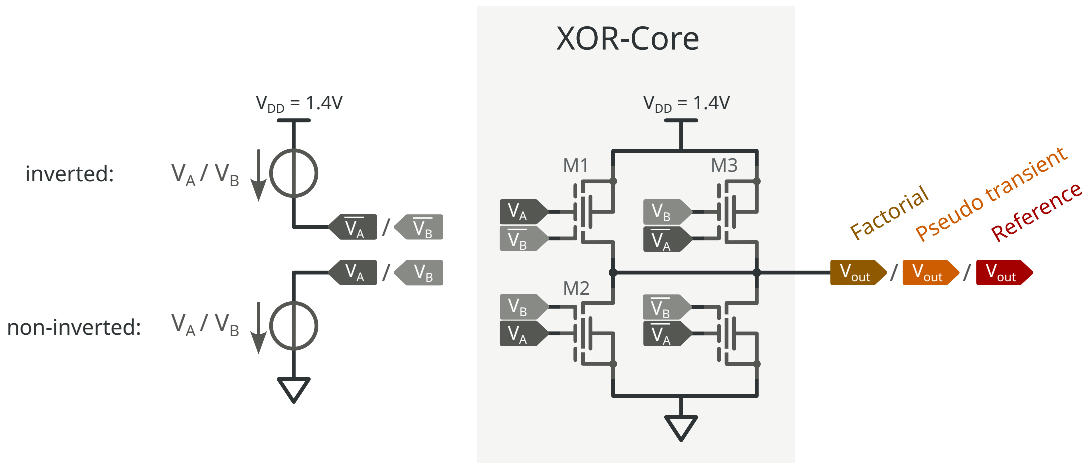
XOR Core - DC Analysis

As illustrated in Section 2.1.3, RFETs allow the area efficient implementation of logic cells, such as the *XOR* gate, by exploiting the polarity gate as a logic input. The analysis of the core of the *XOR* gate, i. e. the logic stage after the input inverters, is presented in this work, and demonstrates the capabilities of dynamic reconfiguration of the proposed models. Analysis without the input inverters exposes the detailed switching characteristic depending on the linear input sweep, as the transition region becomes wider. Figure 5.9 shows a simulation setup with two DC voltage sources for the non-inverted inputs A/B and the inverted inputs $\overline{A}/\overline{B}$, and the simulation is again performed for the factorial- and the pseudo transient table model, respectively.

In general, both models show close correlation with the reference in most parts and mostly succeed in predicting switching thresholds. Similarly as for the inverter, the deviations are largest with the pseudo transient table model, as pronounced in case d), where V_B is swept with $V_A = V_{DD}$. The lack of smoothness, especially of the pseudo transient table model curves, compared to the reference is, again, a result of interpolation. The simulation time in Figure 5.9 confirms the findings of the inverter simulation, namely that the pseudo transient model requires approximately 15% more simulation time for the computation of 200 simulation steps than the factorial model at also slightly reduced accuracy.

5.2 Evaluation of Compact Model Approaches

The purpose of the machine learning based compact model approach is to improve on existing data sets, such as the factorial and the pseudo transient table models, which are evaluated in Section 5.1. First the hyperparameter tuning result is analyzed in Section 5.2.1. Then, the regression results of the machine learning models are analyzed in Section 5.2.2, where structure and accuracy scores are presented for each model. This section further answers the question of what are the benefits and drawbacks in transforming the table models into machine learning based compact models. Consequently, detailed circuit level analysis is performed on *INV* and *XOR* cell in Section 5.2.3. Timing analysis of a set of standard cells, consisting of *NAND*, *NOR*, and *XOR*, all resorting to the proposed planar RFET compact models, is shown in Section 5.2.4.



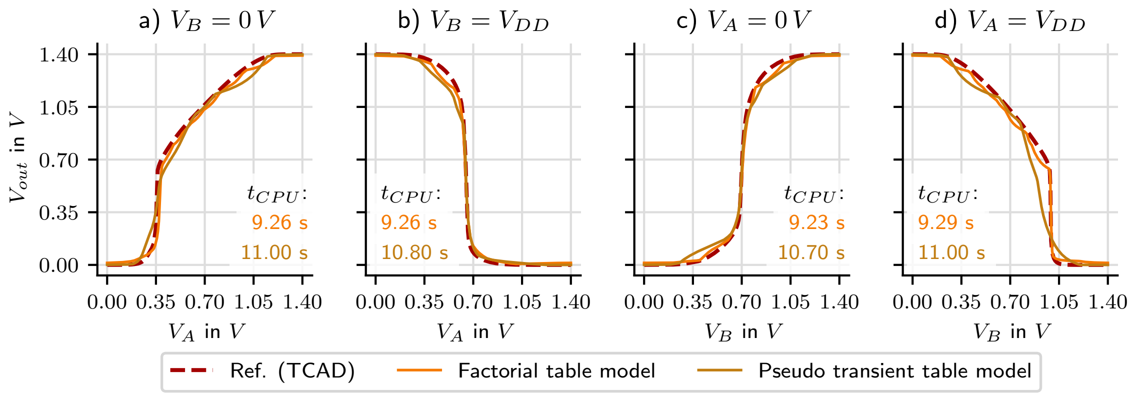


Figure 5.9: Simulation of the *XOR* core shows correct logic behavior, while interpolation artifacts are clearly visible. The factorial table model shows better correlation with the reference than the pseudo transient table model.

This comparison of logic cell timings between simulations with the proposed models gives an estimate of their respective accuracy.

For the compact model from factorial data, a pruning step is conducted with a minimum step size $\Delta V_{sweep,prune} = 0.02\,V$, according to Section 4.2.1. This results in a sample count of $|\mathcal{D}_{pp,fact}| = 7370$. For sake of comparability, the pseudo transient data set is pruned similarly, by extracting only every $8th$ sample. This approach is equal to reducing N_{os} by a factor of 8 and leads to a sample count after preprocessing of $|\mathcal{D}_{pp,pstrans}| = 7941$.

5.2.1 Ensemble Model Hyperparameters

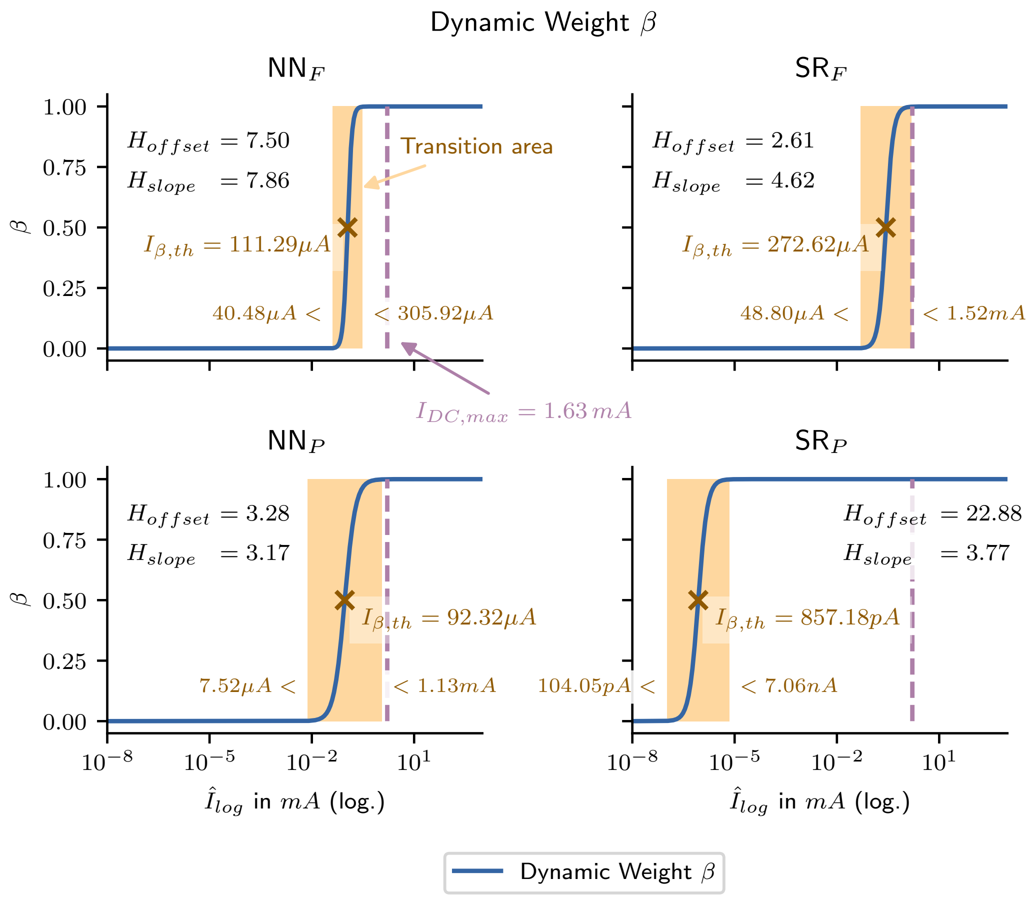


Figure 5.10: Hyperparameter training of the ensemble model exposes that the logarithmic models outperform the linear models over a wide range, except for SR_P.

The hyperparameters H_{offset} and H_{slope} specify the dynamic weight function β, which provides the transition of the linear ($\beta = 1$) and the logarithmic current models ($\beta = 0$), as described in Section 4.2.4. H_{offset} and H_{slope} are tuned for accuracy of the ensemble prediction of the DC drive current $\hat{I}_{DC}$. The location of the transition in Figure 5.10 therefore represents the absolute magnitude of current for which the transition is optimal, according to the tuning result. This transition offset can be interpreted as the relative performance between the respective linear and logarithmic models. If the optimal ensemble is achieved with a high

H_{offset}, the transition is shifted to the left, as seen for $\mathsf{SR_P}$ with a transition threshold of $I_{\beta,th} = 857.18\,pA$. This can either be explained with poor performance of the logarithmic model or outstanding performance of the linear model well into the low current region. Similarly, the second symbolic regression-based model $\mathsf{SR_F}$ shows a relatively high upper bound of transition area, which almost includes the entire current range of the $\mathcal{D}_f act$. Tuning of the neural network-based models results in a transition threshold between $I_{\beta,th,NN_P} = 92.32\,\mu A$ and $I_{\beta,th,SR_F} = 111.29\,\mu A$. Consequently, the contribution of the linear model lies only in the peak current region for $\mathsf{NN_P}$, $\mathsf{NN_F}$ and $\mathsf{SR_F}$.

5.2.2 Structure and Accuracy of Individual Models

A baseline estimate of model accuracy and the benefits of the ensemble model technique over the individual machine learning models can be gained by applying the error metrics from Section 2.3.5 along with ground truth data unseen during training. To provide a common ground for comparison between the individual models, especially between linear and logarithmic models, all metrics are applied after back transformation to mA and C, respectively. The training sets for linear and logarithmic current model are initially transformed to mA, using a scaling factor of $S_I = 10^3$, as described in Section 4.3 and Section 4.4. While this already sets the linear model up for evaluation, the logarithmic model requires a back transformation of H^{-1} (Equation 4.13) to be scaled to mA. The charge models are scaled to C using the linear back transformation $1/S_Q = 10^{-15}$. Further, 4 different test sets are used for evaluation in this section: the split for the linear model is taken from a data set, where a lower bound is enforced, as explained in Section 4.3.1. The resulting test set is used to evaluate the linear models only. The other test set, resulting from a split of the entire available data set, is then used to evaluate all logarithmic models and ensemble models. Test models from the linear and the logarithmic split are available for both data sources, factorial and pseudo transient, which results in a total of 4 test sets.

Current Models

Depending on the employed regression method, the resulting model structure is distinct. Table 5.2 lists the required operations with their respective count for the symbolic regression-based drive current models. The two data sources for each model, factorial and pseudo transient, result in a similar involvement of arithmetic operations for the respective logarithmic models. During training of the symbolic regression-based current models the edge count is constrained to a maximum of 100 edges, as described in Section 4.4. The resulting models feature a maximum of 18 operations and 31 graph edges in case of the logarithmic models, hence the available complexity of 100 edges is far from being exploited.

Table 5.2: All symbolic regression based models show similar complexity, which is below the constrained maximum of 100 edges.

Arch.	**Mdl.**	**Data**	Edges	OPs	$+-$	$\times$	$\frac{1}{a}$	e^a	$\sqrt{a}$	a^2	Lin.	Gauss.	tanh
SR	$\hat{I}_{lin}$	Fact.	29	15	5	3	0	0	1	0	0	5	1
		P.T.	30	17	6	3	0	1	0	0	2	3	2
	$\hat{I}_{log}$	Fact.	31	18	6	3	0	0	0	0	2	3	4
		P.T.	31	18	6	3	1	0	0	1	0	3	4

Table 5.3: The operation count distinguishes neural network- and symbolic regression approach clearly, as matrix operations are not supported in Verilog-A.

Arch.	**Mdl.**	**Data**	$H_{hl-size}$	OPs	$+-$	$\times$	$\frac{1}{a}$	e^a	$\sqrt{a}$	a^2	Lin.	Gauss.	tanh
NN	$\hat{I}_{lin}$	Fact.	35	5286	2590	2590	0	0	0	0	0	0	106
		P.T.	35	5286	2590	2590	0	0	0	0	0	0	106
	$\hat{I}_{log}$	Fact.	34	4999	2448	2448	0	0	0	0	0	0	103
		P.T.	40	6841	3360	3360	0	0	0	0	0	0	121

The neural network-based current model show substantially larger model size than the symbolic regression-based models, as shown in Table 5.3. Hyperparameter tuning of the hidden layer sizes results in $H_{hl-size}$ between 34 and 40 units per hidden layer. The logarithmic model from pseudo transient data stands out as it exploits the maximum available hidden layer size, which is constrained during hyperparameter tuning, as described in Section 4.5. The proposed neural network-based current models are all similar in size, with a minimum operation count of 4999 in total for the logarithmic model from factorial data and a maximum of 6841 operations for the logarithmic model from pseudo transient data. The respective addition, multiplication and $\tanh$-activation, required for sequential computation of forward propagation, are calculated using Equation 4.14. Other functions are not used for implementation of the MLP. At a similar count of iterations per solution and a similar amount of simulator steps, $\mathsf{NN_F}$ and $\mathsf{NN_P}$ are expected to result in substantially higher resource requirements than $\mathsf{SR_F}$ and $\mathsf{SR_P}$.

Having performed all back transformations to mA and C, the individual models are evaluated using R^2, MAE, MSE, and sMAPE, introduced in Section 2.3.5. The R^2 score abstracts from the magnitude of the data samples and is therefore adequate to compare all models – current and charge – with each other. Similarly, the sMAPE is chosen to provide a relative error figure. Further, the MAE is used for its intuitive expression of the average error and the MSE represents the loss after training of the neural network models, both after back transformation to mA.

Table 5.4 shows the individual current model scores with respect to the data source on which they are trained and evaluated. It becomes evident that the neural network-based models show a substantially higher accuracy with respect to symbolic regression-based models in all employed metrics. Especially the logarithmic symbolic regression model shows high deviation from the ground truth when predicting on the respective test set. This confirms that the high offset of the dynamic weight function β for SRP, evaluated in Section 5.2.1, stems from an inaccurate logarithmic model. In the set of symbolic regression models, the pseudo transient data set results in slightly better relative accuracy, expressed by a lower sMAPE. However, the overall quality of the symbolic regression models is low and the distribution of the drive current samples appears to be difficult to fit with the proposed symbolic regression approach. Considering that the available complexity of 100 edges for the symbolic regression has only been used by approximately 30%, it can be concluded that either the set of available functions in QLattice does not suffice or the regression algorithm does not allow accuracy comparable to the neural network approach.

Table 5.4: Evaluation of the individual models confirms that the ensemble strategy improves the relative error (sMAPE) for all models but SRP. Symbolic regression current models show substantially lower accuracy than the neural network-based models.

Arch.	**Model**	**Data**	R^2 ≤ 1	**MAE**	**MSE/loss** (mA^2)	**sMAPE** $\in [0\%, 200\%]$
NN	$\hat{I}_{lin}$	Fact.	0.99988	$1.75\,\mu A$	$7.38 \cdot 10^{-6}$	72.16
		Ps. Trans.	0.99998	$1.44\,\mu A$	$4.48 \cdot 10^{-6}$	62.01
	$\hat{I}_{log}$	Fact.	0.99870	$2.05\,\mu A$	$4.64 \cdot 10^{-5}$	34.22
		Ps. Trans.	0.99938	$3.66\,\mu A$	$9.68 \cdot 10^{-5}$	8.96
	$\hat{I}_{DC}$	Fact.	0.99996	$482.42\,nA$	$1.49 \cdot 10^{-6}$	33.87
		Ps. Trans.	0.99999	$685.00\,nA$	$9.68 \cdot 10^{-5}$	8.51
SR	$\hat{I}_{lin}$	Fact.	0.99772	$7.34\,\mu A$	$1.44 \cdot 10^{-4}$	91.20
		Ps. Trans.	0.99921	$8.19\,\mu A$	$1.61 \cdot 10^{-4}$	76.19
	$\hat{I}_{log}$	Fact.	0.67312	$31.91\,\mu A$	$1.17 \cdot 10^{-2}$	87.78
		Ps. Trans.	0.94313	$44.88\,\mu A$	$8.94 \cdot 10^{-3}$	72.89
	$\hat{I}_{DC}$	Fact.	0.93666	$16.76\,\mu A$	$2.27 \cdot 10^{-3}$	84.24
		Ps. Trans.	0.99925	$5.93\,\mu A$	$8.94 \cdot 10^{-3}$	79.65

Neural network-based models expose how the ensemble model improves the overall accuracy. The rationale for using the ensemble model technique is to compensate the high relative error for low currents, that the linear model accepts during training, with a logarithmic model. Although the logarithmic model is expected to show less absolute accuracy, expressed by the

MAE, the accuracy can be more consistent throughout the entire value space, compared to the linear model, which is in turn expressed by the sMAPE. This is confirmed in Table 5.4, as the linear models consequently show a higher sMAPE and a lower MAE than the logarithmic models. It can therefore be assumed that that substantial contributors to this high relative error of the linear models lie, in fact, in the low current regime, which emphasizes the relevance of the ensemble model approach for the DC drive current of $\mathsf{NN_F}$ and $\mathsf{NN_P}$.

Table 5.5: The operation count of the charge models is almost identical throughout the electrodes and data sets.

Model	**Data**	Edges	Total	$+-$	$\times$	e^a	$\sqrt{a}$	a^2	Lin.	Gauss.	tanh
$\hat{Q}_{FG}$	Fact.	29	14	7	3	0	0	0	0	4	0
	Ps. Trans	31	16	5	4	0	0	0	1	5	1
$\hat{Q}_{TG}$	Fact.	29	14	7	2	0	0	0	0	5	0
	Ps. Trans	31	16	8	1	0	1	0	0	5	1
$\hat{Q}_{BG}$	Fact.	31	16	5	6	0	0	0	1	4	0
	Ps. Trans	31	16	3	5	1	1	0	0	6	0
$\hat{Q}_{Lat1}$	Fact.	30	15	5	3	0	0	1	0	6	0
	Ps. Trans	31	15	8	2	0	0	0	0	5	0
$\hat{Q}_{Lat2}$	Fact.	31	15	8	2	0	0	0	0	5	0
	Ps. Trans	31	15	5	2	0	0	0	0	8	0

Charge Models

Another assumption on which the compact modeling approach is designed, is that the mathematical distribution of charge of each electrode with respect to the electrode voltages is less complex, in the sense of suitability for regression-based fitting, than the distribution of the drive current. Hence, the charge models are exclusively obtained by symbolic regression, as described in Section 4.4.2. In fact, the charge models show a similar structural complexity as the symbolic regression-based current models, as shown in Table 5.5. The number of edges per graph and the types of operations used for charge modeling are highly similar throughout electrodes and data sets of the charge samples, and also to the current models in Table 5.2.

However, evaluating the score on their respective test set in Table 5.6, exposes a substantially lower sMAPE than the symbolic regression-based current models in Table 5.4. Hence, even though the resource requirement between current models and charge models are similar, the better fit renders the symbolic regression approach appropriate for charge modeling. The best relative accuracy is seen with the TG charge model, which features a 50% overlap at Lat1 and Lat2, respectively. $\hat{Q}_{FG}$ appears as slightly more difficult to fit, as it features the highest relative error and the lowest R^2 of all charge models. However, the relative error (sMAPE) is still considerably lower than all symbolic regression-based DC current models in Table 5.4.

Table 5.6: The symbolic regression based charge models show significantly higher accuracy than the symbolic regression-based current models.

Arch.	**Model**	**Data**	R^2 ≤ 1	**MAE** (C)	**MSE/loss** (C^2)	**sMAPE** $\in [0\%, 200\%]$
SR	$\hat{Q}_{FG}$	Fact.	0.99517	$1.06 \cdot 10^{-16}$	$2.03 \cdot 10^{-32}$	26.37
		Ps. Trans.	0.99481	$1.69 \cdot 10^{-16}$	$5.67 \cdot 10^{-32}$	22.30
	$\hat{Q}_{TG}$	Fact.	0.99991	$2.10 \cdot 10^{-17}$	$6.81 \cdot 10^{-34}$	2.30
		Ps. Trans.	0.99993	$2.22 \cdot 10^{-17}$	$7.95 \cdot 10^{-34}$	2.41
	$\hat{Q}_{BG}$	Fact.	0.99723	$7.12 \cdot 10^{-18}$	$9.28 \cdot 10^{-35}$	8.66
		Ps. Trans.	0.99643	$1.22 \cdot 10^{-17}$	$3.00 \cdot 10^{-34}$	14.94
	$\hat{Q}_{Lat1}$	Fact.	0.99692	$6.23 \cdot 10^{-17}$	$6.46 \cdot 10^{-33}$	15.80
		Ps. Trans.	0.99605	$1.05 \cdot 10^{-16}$	$2.09 \cdot 10^{-32}$	15.24
	$\hat{Q}_{Lat2}$	Fact.	0.99819	$7.00 \cdot 10^{-17}$	$8.59 \cdot 10^{-33}$	13.35
		Ps. Trans.	0.99816	$1.07 \cdot 10^{-16}$	$2.26 \cdot 10^{-32}$	15.50

5.2.3 Circuit Level Analysis

Similarly to the evaluation of the table models in Section 5.1.3, *INV* and *XOR* cells are presented for a detailed performance analysis of the machine learning based compact models. Evaluation of an unstable feedback circuit exposes the accuracy of the models in transient simulation. For this purpose, a 5-stage ring oscillator circuit is simulated with all proposed models.

Inverter - DC and Transient Analysis

Figure 5.11 shows the characteristic of an inverter for the 4 compact model candidates. Neural network- and symbolic regression based simulations are separated for better visual distinction. The prediction of the VTC in a) and the according cross current in e), demonstrate the high accuracy of the neural network models for both data sources. Correlation with the TCAD reference is close and interpolation artifacts, seen in the table model simulation of Figure 5.8 are avoided entirely. The neural network approach appears as an adequate method to replace interpolation for non-polynomial distributions in DC simulation.

When the inverter circuit is extended with an output capacitance of $C = 140\,fF$, transient simulation shows close correlation with the reference for V_{out}. The reduced output swing, shown by TAB_F in Figure 5.8, is compensated. Further, the neural network approach allows the pseudo transient data set to show a higher accuracy after the falling transition in Figure 5.11 g). In general the difference between both data sources, factorial method and pseudo transient

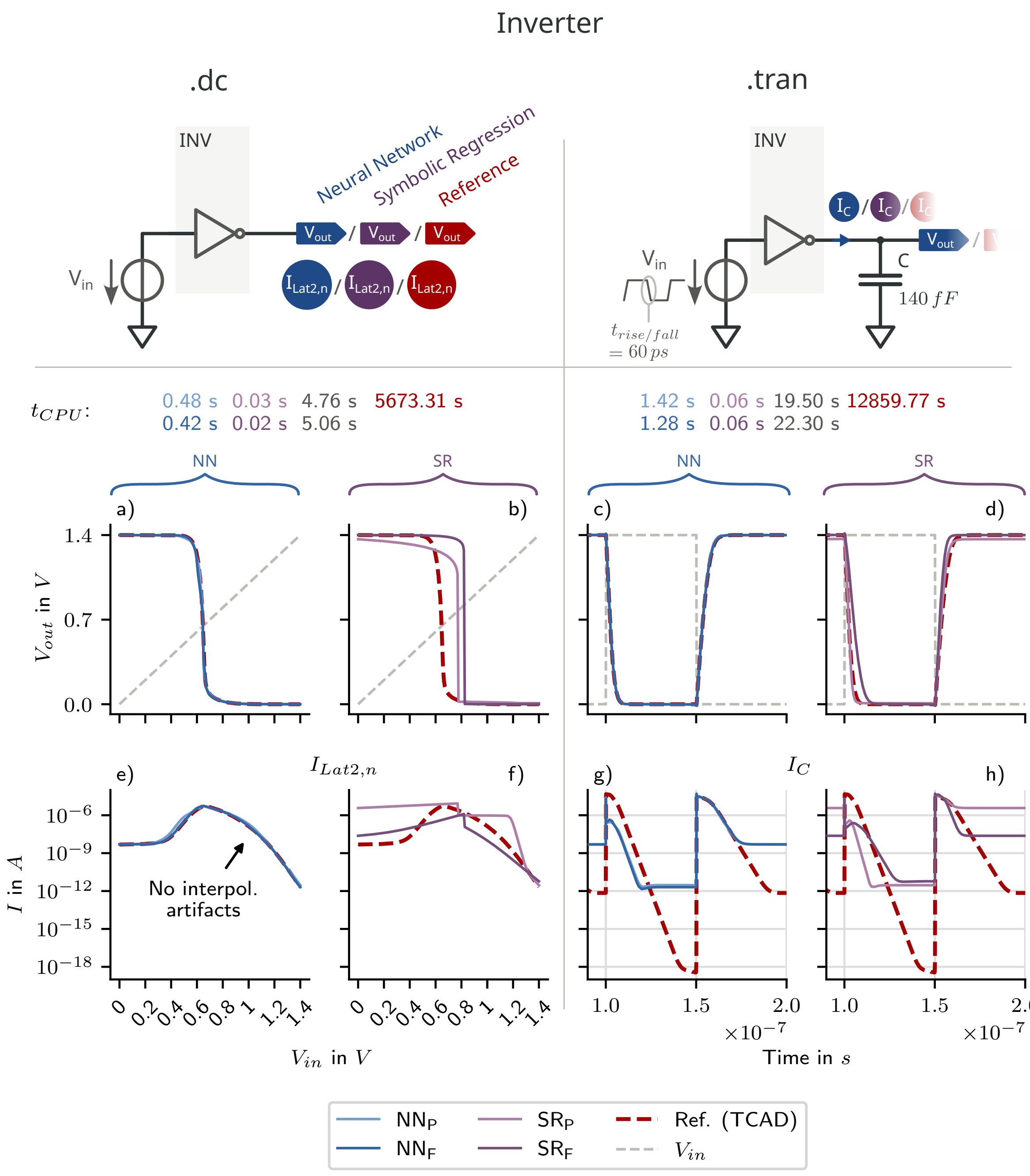


Figure 5.11: The neural network-based models show excellent accuracy in DC simulation, while SR_F and SR_P show high deviations. Symbolic regression, although with higher deviation, is substantially faster than neural network-based models or table models.

method, shrinks after transformation to a neural network based compact model and in case of DC simulation, the accuracy improves substantially.

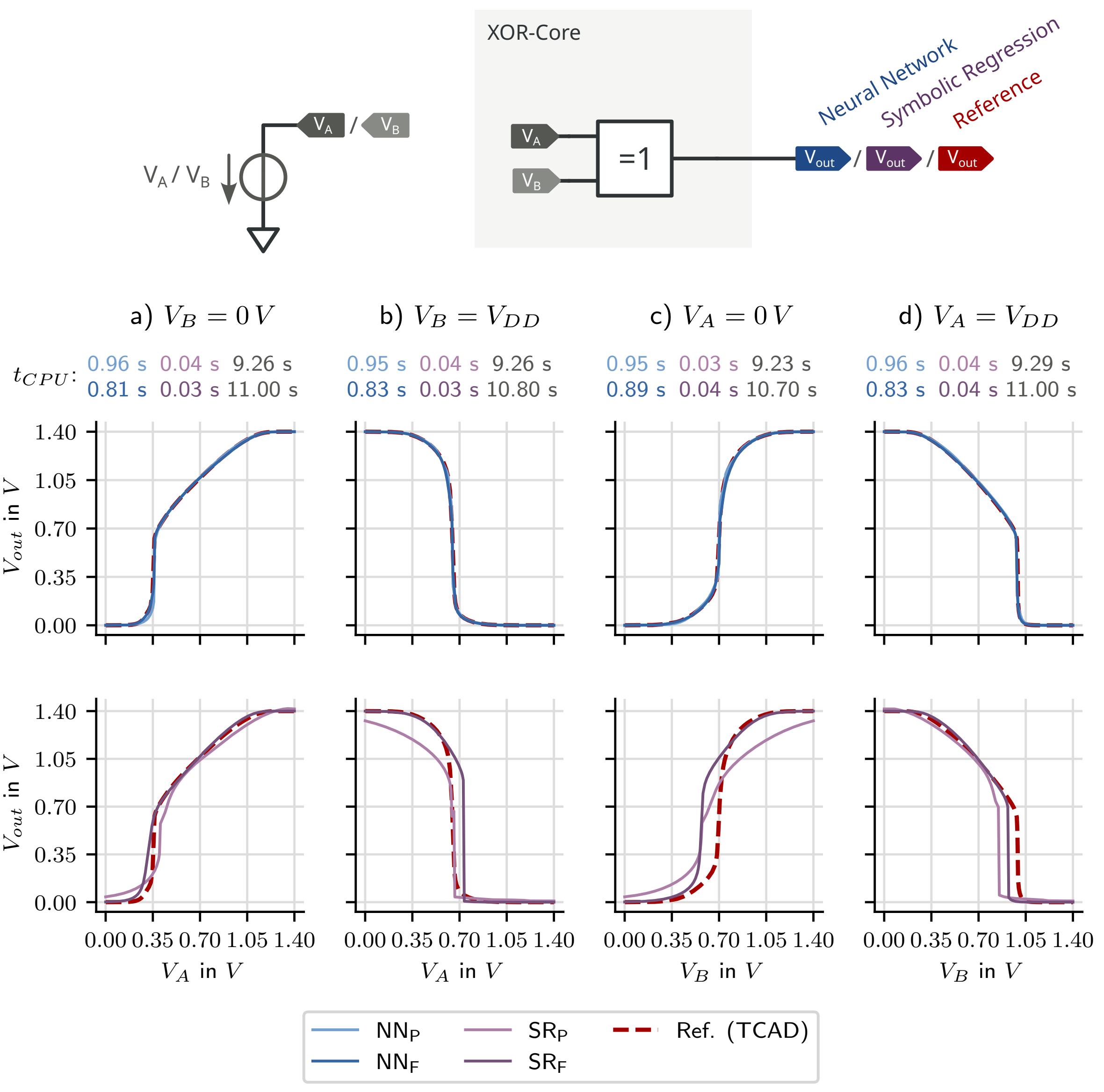


Figure 5.12: The neural network based approach performs accurately when dynamic reconfiguration is simulated. Similar to the inverter demonstration, the symbolic regression approach fails at providing the full output swing.

However, the symbolic regression-based compact models fail to predict the transition region and introduce a shift towards higher V_{in}, as shown for the DC simulation in Figure 5.11 b). The logarithmic cross current diagram of SR_F and SR_P in f) shows that the correlation with the reference is poor in most parts and sharp transitions between the respective linear and logarithmic models cause an overly steep VTC. Further, SR_P does not reach full swing output as it shows considerable offset to V_{DD} for $V_{in} = 0\,V$. Transient simulation of SR_P shows substantial deviation from the reference, accordingly. Along with clear shifts of rising and falling transition, this prevents the recommendation of $\mathsf{SR_P}$ and $\mathsf{SR_F}$ for predictive circuit simulation.

When analyzing the simulation times (t_{CPU}), however, the ranking changes. All proposed models speed up circuit simulation, with respect to the TCAD reference. The machine learning-based compact models are in general more efficient than the table models. $\mathsf{SR_F}$ brings an acceleration of up to $25\times$ with respect to $\mathsf{NN_P}$ while a factor of up to $17\times$ distinguishes the simulation time of $\mathsf{TAB_P}$ and $\mathsf{NN_P}$. This result reflects the low operation count of the symbolic regression models, with respect to the neural networks.

XOR Core - DC Analysis

The simulation of the *XOR* core cell in Figure 5.12 agrees with the findings of the inverter simulation and provides, with the neural network-based compact models, close to perfect correlation with the reference transitions for both inputs A and B. Similar to the inverter VTC in Figure 5.11, interpolation artifacts are suppressed, and the traces are visibly smooth.

Again, $\mathsf{SR_P}$ and $\mathsf{SR_F}$ show large deviations the transition region and, especially $\mathsf{SR_P}$, fails to provide a full swing output in transition a), b) and c). The transitions, especially in b), c) and d) are overly steep and lack smoothness.

The simulation time with 4 compact model instances confirms the advantage of neural network- and symbolic regression architecture. The symbolic regression-based models extend their lead in computational efficiency of to up to $27\times$, when comparing $\mathsf{NN_P}$ with $\mathsf{SR_F}$ for the transition in a). The neural network based models have their peak acceleration of $13\times$ with $\mathsf{NN_F}$, compared to the $\mathsf{TAB_P}$ in a).

A 5-Stage Ring Oscillator

The simulation of a ring oscillator imposes high transient accuracy on the instantiated compact models. A typical method to prevent a static solution at $V_{out} = 0.5 \cdot V_{DD}$ for all inverters of the unstable feedback circuit is to either constrain an initial condition or to insert another FET to release one of the switching nodes with a pulse waveform at the start of the simulation. While both of these methods are available for RFETs, controllable polarity enables an additional way of starting and stopping the oscillator: When flipping the polarity of both RFETs of an inverter, the cell transforms into a buffer, which decrements the number of inverters in the ring and stops oscillation. In Figure 5.13 reconfiguration from buffer to inverter is used to start the 5-stage ring oscillator. Before the start of the oscillation at $t \approx 30\,ns$, the steady state solution can be seen to deviate from the reference, with $\mathsf{TAB_P}$, $\mathsf{SR_F}$ and $\mathsf{SR_P}$.

The waveform of the ring oscillator in Figure 5.13 shows that the table models and both neural network-based compact models and $\mathsf{SR_P}$ show a similar period T_p between $4.69\,ns$ and $4.96\,ns$. The outlier is $\mathsf{SR_F}$ with $T_p = 7.43\,ns$, and while the former models deviate from the reference period with a factor of less than 1.3, the ring oscillator with $\mathsf{SR_F}$ models almost doubles the period of the reference. The similar waveforms of the table models and the neural network compact models indicate that the underlying quasistationary data sets limit the accurate prediction of frequency dependent capacitances in transient simulation, as described in Section 2.2.3.

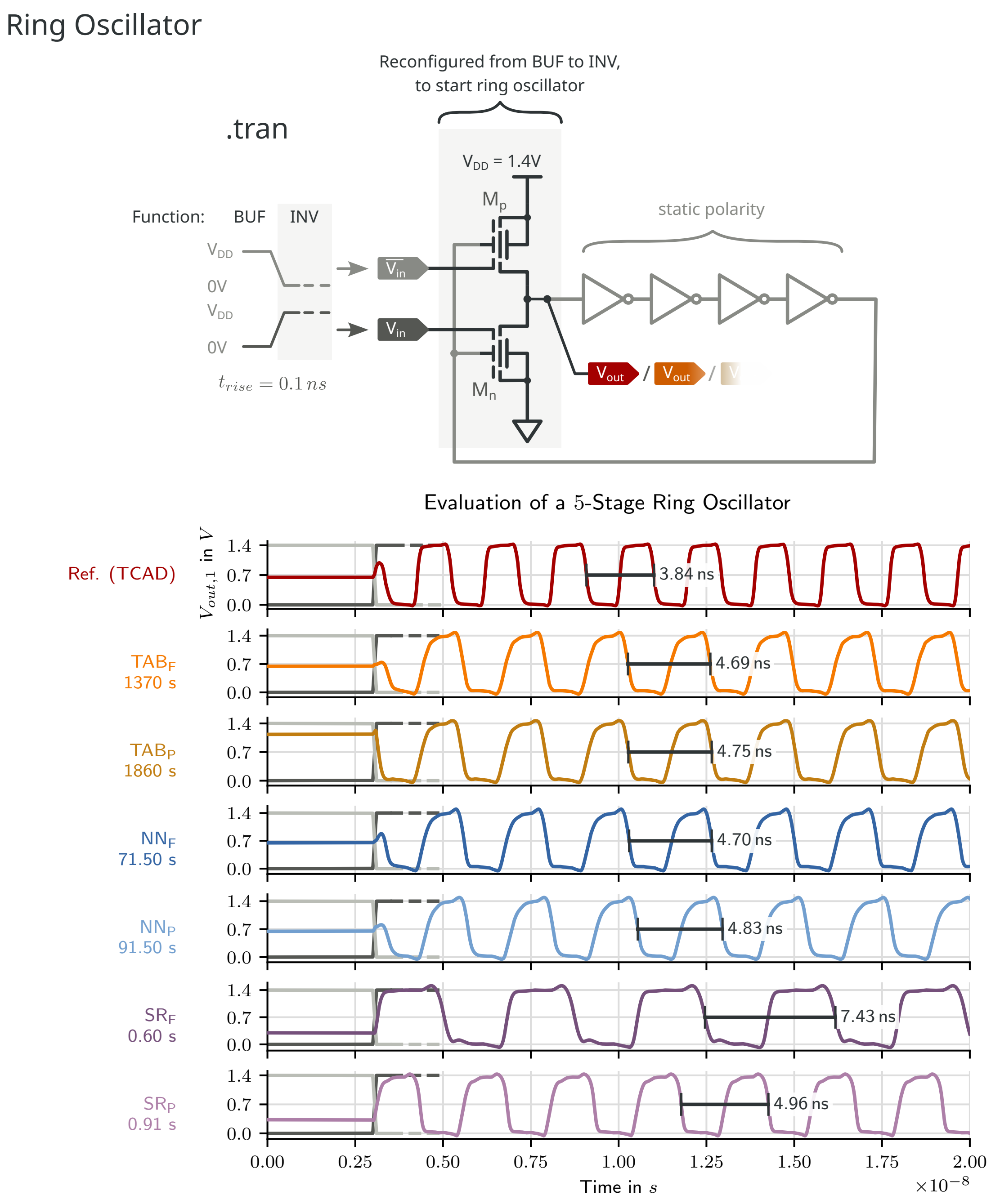


Figure 5.13: Simulation of a 5 stage ring oscillator expose limitations of the data driven methods. The ring oscillator is enabled by reconfiguring one cell in the chain from buffer to inverter.

5.2.4 Cell Timing Analysis

During the detailed model evaluation with respect to accuracy figures and circuit simulation, SR_F and SR_P turn out to provide insufficient accuracy for reliable circuit simulation. Although the symbolic regression approach shows excellent efficiency, i. e. simulation time, in its current state the evaluation in form of cell characterization is not considered useful. This section therefore focuses on the neural network-based models $\mathsf{NN_F}$ and $\mathsf{NN_P}$ along with the proposed table models $\mathsf{TAB_F}$ and $\mathsf{TAB_P}$. The extracted characterization values are put in comparison with the respective TCAD reference.

The principal objective of this work is to obtain models, which unite accuracy and efficiency for the simulation of digital circuits. In the final part of the evaluation, this objective is evaluated for a set of logic cells: *NAND*, *NOR* and *XOR*. The respective cells are terminated with $C_{out} = 140\,fF$ towards V_{SS} and the linear input edges have a transition time of $t_{tr,rise} = t_{tr,fall} = 60\,ps$ – all equal to the detailed analyses of circuit simulation in this chapter. These constraints approximately correspond to a fan-out of 10 *XOR* gates, each with an input capacitance of $C_{in,xor} \approx 10\,fF$ and a line capacitance of $40\,fF$ according to [45]. The input transition is exemplary to provoke a representative transient characteristic in the output voltage. The respective simulation time for each cell, input transition and model are not subject of this section and can be found in Appendix A.

The characterization tables, Table 5.7 to Table 5.10 show a characterization of the respective digital cell for input transitions with two inputs A and B. 4 input transitions are possible: a transition of A with $B = 0$, a transition of A with $B = 1$, a transition of B with $A = 0$ and a transition of B with $A = 1$. The transitions are conducted with rising- and falling edge, respectively. *NAND* and *NOR* cell only switch for two out of the four possible input transitions of A and B. For each characteristic value of t_{tr} and t_d the best prediction and the worst prediction are displayed with green and red cell background. The resulting waveforms are time-discrete and, in order to obtain accurate characteristic values at 20%, 50% and 80% of V_{DD}, the respective samples are obtained by linear interpolation. The relative deviation from the TCAD reference of the worst predicting model in percent is listed in the bottom row.

Table 5.7: Transition and delay figures of the *NAND* cell favor the table models $\mathsf{TAB_F}$ and $\mathsf{TAB_P}$.

Cell	**NAND**							
Param.	**Transition t_{tr} in s**				**Delay t_d in s**			
Edge	**A01_B1**		**A1_B01**		**A01_B1**		**A1_B01**	
Dir (In)	**Rise**	**Fall**	**Rise**	**Fall**	**Rise**	**Fall**	**Rise**	**Fall**
Ref.	1.36e-8	7.87e-9	1.36e-8	7.96e-9	7.44e-9	3.63e-9	7.51e-9	3.70e-9
$\mathsf{TAB_F}$	🟩 1.37e-8	7.94e-9	🟩 1.37e-8	8.03e-9	🟩 7.39e-9	🟥 3.50e-9	🟩 7.46e-9	🟥 3.57e-9
$\mathsf{TAB_P}$	1.32e-8	🟩 7.88e-9	1.32e-8	🟩 7.96e-9	7.26e-9	🟩 3.61e-9	7.34e-9	🟩 3.69e-9
$\mathsf{NN_F}$	🟥 1.31e-8	7.74e-9	🟥 1.31e-8	7.82e-9	🟥 7.16e-9	3.57e-9	🟥 7.23e-9	3.63e-9
$\mathsf{NN_P}$	1.38e-8	🟥 7.60e-9	1.38e-8	🟥 7.65e-9	7.71e-9	3.58e-9	7.77e-9	3.64e-9
$\Delta \leq$	3.8 %	3.4 %	3.8 %	3.8 %	3.7 %	3.5 %	3.8 %	3.6 %

The *NAND* cell, characterized in Table 5.7, shows clear patterns with respect to the best prediction. Over all, the table models provide the best predictions for all transitions, while the underlying data set depends on the direction of the transition: A rising transition at the input is accurately predicted by the factorial data set, while the falling transition exposes the pseudo transient data set to predict best. For all transition characteristics of the *NAND* cell, the same data set that results in the best prediction when used as a table model, leads to the highest deviation when used with the neural network approach. While the pattern of the best prediction extends to the delay figures, the $\mathsf{TAB_F}$ and $\mathsf{TAB_P}$ deviate by 3.6% and 3.5% with a falling input transition.

Table 5.8: Transition and delay figures of the *NOR* cell favor $\mathsf{TAB_P}$, consequently.

Cell	NOR							
Param.	**Transition t_{tr} in s**				**Delay t_d in s**			
Edge	**A01_B0**		**A0_B01**		**A01_B0**		**A0_B01**	
Dir (In)	**Rise**	**Fall**	**Rise**	**Fall**	**Rise**	**Fall**	**Rise**	**Fall**
Ref.	5.03e-9	1.58e-8	4.92e-9	1.58e-8	2.33e-9	8.06e-9	2.27e-9	7.88e-9
$\mathsf{TAB_F}$	5.05e-9	1.63e-8	4.93e-9	1.63e-8	2.30e-9	7.96e-9	2.23e-9	7.79e-9
$\mathsf{TAB_P}$	5.03e-9	1.59e-8	4.91e-9	1.59e-8	2.36e-9	8.05e-9	2.29e-9	7.88e-9
$\mathsf{NN_F}$	5.00e-9	1.56e-8	4.89e-9	1.56e-8	2.40e-9	7.92e-9	2.33e-9	7.75e-9
$\mathsf{NN_P}$	5.09e-9	1.50e-8	4.98e-9	1.50e-8	2.37e-9	7.74e-9	2.30e-9	7.58e-9
$\Delta \leq$	1.2 %	5.1 %	1.3 %	5.1 %	2.8 %	4.0 %	2.6 %	3.8 %

In general, the highest deviations (Δ) are all from consequent underestimation of the transition time by a relative amount of 3.4% to 3.8%. The accuracy of the best predictions is, however, well within 1%.

The *NOR* cell in Table 5.8 shows an even more clearly recognizable pattern. $\mathsf{TAB_P}$ always predicts best. And, similar to the transition figures of the *NAND* cell, the same data set – the pseudo transient data set – is responsible for the highest deviation, which reaches 5.1% for the transition times with a falling input edge. The falling input, in case of the *NOR* gate, corresponds to a rising edge at the output and the charging process is dominated by the p-type behavior of the planar RFET. The higher deviation in these cases confirms the findings of Figure 5.11, in which $\mathsf{NN_P}$ shows slightly reduced correlation in the pull-up region of the VTC. For pull-down (n-type) behavior, the transition figures all correlate closely with the reference. Delay figures for the *NOR* cell are more balanced between 2.6% and 4.0% of maximum deviation. The models with the largest deviations, respectively, overestimate transition- and delay timings with a rising input edge and underestimate with a falling input edge.

The demonstrator of dynamic reconfiguration, the *XOR* cell, shows less structure in its characteristic figures. All evaluated models shine in at least two characteristics, as shown in Table 5.9 and Table 5.10. In case of the falling input transition of A with $B = 0$ the maximum deviation is less than 1%. Responsible for the largest deviation of 4.6% is the neural network-based model from the pseudo transient data set for a rising edge of A with $B = 0$.

The take-away from the evaluation of timing characteristics is that all models perform within a maximum relative error of 5.1%, for the given input transition time and output capacitance, so that all interpretation of causality is made based on a low overall deviation, in the first place. Unlike with the 5-stage ring oscillator, even the largest deviations are not constantly over- or underestimation of transition- and delay characteristics. A major systematic error, e. g. from a consequently overestimated inversion capacitance, is therefore difficult to conclude. Quite the

Table 5.9: Transition and delay figures of the *XOR* cell expose accuracy in all evaluated timing figures.

Cell	**XOR**							
Param.	**Transition** t_{tr} in s				**Delay** t_d in s			
Edge	**A01_B0**		**A0_B01**		**A01_B0**		**A0_B01**	
Dir (In)	**Rise**	**Fall**	**Rise**	**Fall**	**Rise**	**Fall**	**Rise**	**Fall**
Ref.	7.79e-9	4.30e-9	7.62e-9	4.13e-9	2.94e-9	1.25e-9	2.91e-9	1.13e-9
TAB$_\text{F}$	7.91e-9	4.35e-9	7.80e-9	4.26e-9	2.99e-9	1.25e-9	2.92e-9	1.15e-9
TAB$_\text{P}$	7.64e-9	4.25e-9	7.59e-9	4.16e-9	2.82e-9	1.26e-9	2.82e-9	1.17e-9
NN$_\text{F}$	7.61e-9	4.15e-9	7.55e-9	4.05e-9	2.88e-9	1.25e-9	2.85e-9	1.15e-9
NN$_\text{P}$	7.43e-9	4.26e-9	7.40e-9	4.17e-9	2.92e-9	1.24e-9	2.92e-9	1.15e-9
$\Delta \leq$	4.6 %	3.5 %	2.8 %	3.2 %	4.0 %	0.89 %	3.1 %	3.6 %

Table 5.10: A clear causality of the origin of the maximum deviation is not apparent, as the relative error of models over all evaluated cells is $\leq 5.1\%$.

Cell	**XOR**							
Param.	**Transition** t_{tr} in s				**Delay** t_d in s			
Edge	**A01_B1**		**A1_B01**		**A01_B1**		**A1_B01**	
Dir (In)	**Rise**	**Fall**	**Rise**	**Fall**	**Rise**	**Fall**	**Rise**	**Fall**
Ref.	3.91e-9	7.60e-9	3.90e-9	7.77e-9	1.15e-9	3.20e-9	1.13e-9	3.15e-9
TAB$_\text{F}$	3.93e-9	7.81e-9	3.92e-9	7.90e-9	1.15e-9	3.20e-9	1.13e-9	3.25e-9
TAB$_\text{P}$	3.92e-9	7.62e-9	3.91e-9	7.69e-9	1.17e-9	3.12e-9	1.15e-9	3.13e-9
NN$_\text{F}$	3.82e-9	7.58e-9	3.82e-9	7.64e-9	1.15e-9	3.19e-9	1.13e-9	3.22e-9
NN$_\text{P}$	3.95e-9	7.42e-9	3.94e-9	7.45e-9	1.16e-9	3.22e-9	1.14e-9	3.23e-9
$\Delta \leq$	2.2 %	2.7 %	2.1 %	4.0 %	1.3 %	2.5 %	2.0 %	3.2 %

opposite is the case, as the maximum error of 5.1% demonstrates the confidence and suitability of the proposed methods for table model and neural network-based compact models.

6 Summary, Conclusion and Outlook

In the course of this work, data driven methods for modeling of emerging semiconductor devices are proposed, which lead to the first compact models of the planar RFET. The first step is to generate data sets for DC drive current and electrode charges, which requires efficient handling of resources in technology simulation. Two approaches are proposed:

- **The factorial cluster simulator PyTaurus** allows efficient setup, reuse and then distributed simulation of parameter sweeps. The proposed tool provides functionalities to mitigate convergence issues in large simulation decks, which is an important asset when going off the beaten track of conventional materials and widely explored conduction mechanisms.
- **Pseudo transient simulation** is a fundamentally different approach, which does not rely on quasistationary parameter sweeps. Instead, all samples are generated by conducting a single slowly proceeding transient simulation. While PyTaurus generates a factorial set through a nested setup of parameter sweeps, pseudo transient simulations realize nesting by applying voltages with harmonic waveforms and distinct frequencies to the electrodes of the DUT.

For comparison of both characterization approaches, the average amount of voltage steps per dimension and the input voltage space are set equal. Both simulation approaches fulfill formal requirements to be used as table models in Verilog-A, as shown in Figure 6.1.

The obtained characteristic values of DC channel current and electrode charges are then transformed into equation based models, i. e. compact models. The DC drive current is represented by an ensemble model, which combines the predictions of two machine learning models. While the linear model is trained on channel current samples in mA, a logarithmic model compensates the decreasing accuracy towards low currents. To obtain the respective linear and logarithmic models, two approaches are demonstrated for the DC drive current:

- **The deep learning approach** provides models in form of MLPs, where the respective predictions for the ensemble model are obtained by forward propagation. Hidden layer sizes and learning rate are treated as hyperparameters and tuned accordingly. The computations of each neuron have to be conducted sequentially, as Verilog-A does not support matrix operations.
- **Symbolic regression** using QLattice provides a model in form of an analytic and closed form expression, which can directly be implemented in Verilog-A. Hyperparameter tuning is not required.

For the charge models, analytic equations are obtained solely by symbolic regression. Four compact models result from a permutation of the two data sets, factorial and pseudo transient,

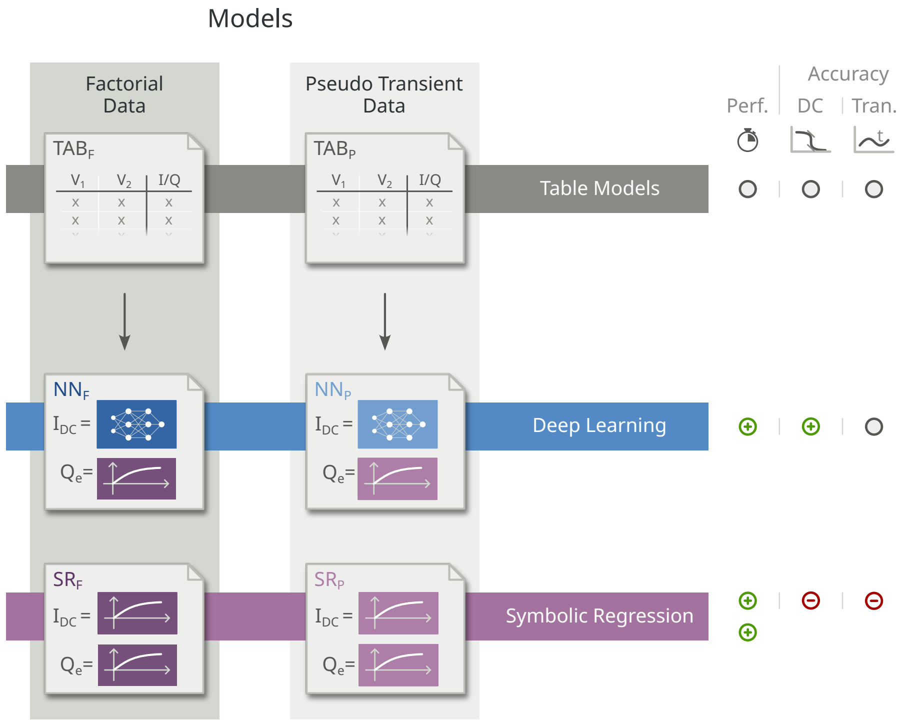


Figure 6.1: Two data generation approaches are proposed in this work. The permutation of factorial- and pseudo transient data with deep learning and symbolic regression results in four models $\mathsf{NN_F}$, $\mathsf{NN_P}$, $\mathsf{SR_F}$ and $\mathsf{SR_P}$.

and two machine learning approaches for the DC drive current, deep learning and symbolic regression, as shown in Figure 6.1.

6.1 Conclusion

The two data generation approaches for a factorial and a pseudo transient data set produce table models, which both satisfy the structural constraints of Verilog-A and show similar accuracy and computational efficiency in circuit simulation. PyTaurus allows distributed simulation and the convergence aids are employed to run all required simulation to completion. Pseudo transient simulation is in its general implementation less resource efficient as the generation of a factorial set through PyTaurus. Perspectively, however, by choosing according parameters such as a high T_{sim}, transient estimation can be omitted and simulation time can be reduced. With further adaptions in implementation, such as the use of cosine as fundamental harmonic function, the simulation time can be reduced further. In this case, it can be extrapolated that The pseudo transient simulation has the advantage in terms of resource efficiency for the considered simulation environment and technology model.

Raised in Section 1.1 and clarified in this work is the key question: Is the machine learning-based compact model an improvement over the direct use of the original table model? From a detailed evaluation, it can be concluded that compact models with a symbolic regression-based DC drive current modeling show subpar performance with respect to error figures and lead to substantial deviations from the reference in circuit simulation. Their use is discouraged. However, the remaining models, the table models $\mathsf{TAB_P}$ and $\mathsf{TAB_F}$, and the models with neural network-based DC drive current modeling $\mathsf{NN_F}$ and $\mathsf{NN_P}$, provide sufficient accuracy for predictive circuit simulation. In a detailed analysis, these models show correct logic behavior of the inspected digital cells *INV* and the *XOR*, which exploits dynamic reconfiguration of the planar RFET. Of all proposed models, $\mathsf{NN_F}$ and $\mathsf{NN_P}$ show the best correlation with the TCAD reference in simulation of the respective VTCs. Compared to the TCAD reference, the relative error in a characterization of transition time and propagation delay is $\leq 5.1\%$, which confirms the suitability, especially of the neural network-based compact models, for predictive simulation of digital cells.

With respect to computational efficiency, the symbolic regression-based compact models show the most efficient simulations. $\mathsf{SR_F}$ and $\mathsf{SR_P}$ speed up circuit simulation time by a factor of up to $27\times$ in the demonstrated simulations of *INV* and *XOR*, compared to the neural network-based models. Table models show the lowest computational efficiency and require are up to $17\times$ more CPU time for circuit simulation than neural network-based models in the analyzed simulations.

To wrap up the evaluation of all proposed models, a reasonable recommendation is to clearly prefer the neural network-based compact modeling approach for DC simulation of digital cells, as accuracy and computational efficiency improve. When it comes to simulations with a focus is on transient behavior, such as ring oscillators, the inherent limitations of data driven modeling are exposed and the accuracy of neural network-based modeling and the plain table models is similar. But as the neural network-based approach speeds up simulation substantially, it should still be the preferred approach. When characterizing digital cells, the accuracy of table models and neural network-based models both show sufficient accuracy for reliable prediction of timing characteristics. Overall, the findings confirm that the straightforward neural network approach, without introducing further effort in manual characterization, leads to a better predictive circuit model with respect to accuracy and computational performance, than using the table model directly.

6.2 Outlook

The methods shown in this work form a versatile foundation for data driven modeling of emerging semiconductor devices, which are demonstrated with a planar RFET. The methodology itself is designed to be generic and applicable when domain knowledge is limited. Further work can extend the proposed modeling flow for other devices, that not necessarily have to be stateless.

As domain knowledge gradually becomes available during technology optimization, either based on TCAD or on fabrication of prototypes, generalization can be traded for accuracy by introducing further constraints. One way of introducing domain knowledge is to use symbolic regression in an iterative manner, by selecting model candidates which resemble physical processes. The model building flow then becomes interactive and allows the user to bias the outcome towards an expected model structure. Further, custom specification of initial symbolic regression models is possible.

The table model approaches are conducted with a single sweep electrode in structured experiment designs, which is required to fulfill constraints of the interpolation algorithm. With the machine learning based compact model approaches it is no longer necessary to satisfy structural constraints, as training samples only have to be in the same feature space. This enables selective refinement of the compact models by appending arbitrary characteristic data to the training set.

The substantial advantages in computational efficiency justify further optimization of the symbolic regression approach, e. g. either with a different algorithm, iterative candidate selection or feature transformation. With the neural network approach, further generalization, e. g. in form of hyperparameters for the number of hidden layers and the amplitude of noise augmentation, can lead to better model accuracy. It is imaginable, that the introduction of convolution into the neural networks leads to a better representation of patterns in the conduction characteristics of a device.

The state of the art already comprises various machine learning approaches for compact modeling. This work joins these ranks, and it is likely that, with similar pace as the introduction of machine learning into more and more areas of our life, machine learning based compact models will experience a boost and bring predictive VLSI simulation to early phases of technology optimization.

In the end, it is questionable, if in the 21st century Gordon Moore is right about the trend of semiconductors. Remains the hope that George Box and Norman Draper are, and all models in this work are "wrong, but some are useful" [1].

Bibliography

[1] G. E. P. Box and N. R. Draper, *Empirical Model-Building and Response Surfaces* (Wiley Series in Probability and Mathematical Statistics). New York: Wiley, 1987, ISBN: 978-0-471-81033-9.

[2] Y. S. Chauhan, D. D. Lu, V. Sriramkumar, *et al.*, *FinFET Modeling for IC Simulation and Design: Using the BSIM-CMG Standard*. London, UK: Academic Press Inc, 2015, ISBN: 978-0-12-420031-9.

[3] G. Moore, "Cramming More Components Onto Integrated Circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, Jan. 1998. DOI: 10.1109/JPROC.1998.658762.

[4] T. Mikolajick, G. Galderisi, S. Rai, *et al.*, "Reconfigurable Field Effect Transistors: A Technology Enablers Perspective," *Solid-state Electronics*, pp. 108 381–108 381, May 2022. DOI: 10.1016/j.sse.2022.108381.

[5] S. Cristoloveanu, *Fully Depleted Silicon-on-Insulator: Nanodevices, Mechanisms and Characterization*. Place of publication not identified: Elsevier, 2021, ISBN: 978-0-12-823165-4.

[6] S. Cristoloveanu, K. H. Lee, H. Park, *et al.*, "The concept of electrostatic doping and related devices," *Solid-State Electronics*, vol. 155, pp. 32–43, May 2019. DOI: 10.1016/j.sse.2019.03.017.

[7] F. Balestra, S. Cristoloveanu, M. Benachir, *et al.*, "Double-gate silicon-on-insulator transistor with volume inversion: A new device with greatly enhanced performance," *IEEE Electron Device Letters*, vol. 8, no. 9, pp. 410–412, Sep. 1987. DOI: 10.1109/EDL.1987.26677.

[8] S. K. Saha, *FinFET Devices for VLSI Circuits and Systems*, 1st ed. CRC Press, Jul. 2020, ISBN: 978-0-429-50483-9. DOI: 10.1201/9780429504839.

[9] J.-P. Colinge, J. Alderman, Weize Xiong, *et al.*, "Quantum-mechanical effects in trigate SOI MOSFETs," *IEEE Transactions on Electron Devices*, vol. 53, no. 5, pp. 1131–1136, May 2006. DOI: 10.1109/TED.2006.871872.

[10] G. Gupta, B. Rajasekharan, and R. J. Hueting, "Electrostatic doping in semiconductor devices," *IEEE Transactions on Electron Devices*, vol. 64, no. 8, pp. 3044–3055, Aug. 2017. DOI: 10.1109/TED.2017.2712761.

[11] R. J. E. Hueting and G. Gupta, "Electrostatic Doping and Devices," in *Springer Handbook of Semiconductor Devices*, M. Rudan, R. Brunetti, and S. Reggiani, Eds., Cham: Springer International Publishing, 2023, pp. 371–389, ISBN: 978-3-030-79826-0 978-3-030-79827-7. DOI: 10.1007/978-3-030-79827-7_11.

[12] P. S. Peercy, "The drive to miniaturization," *Nature*, vol. 406, no. 6799, pp. 1023–1026, Aug. 2000. DOI: 10.1038/35023223.

[13] T. Mikolajick, G. Galderisi, M. Simon, *et al.*, "20 Years of reconfigurable field-effect transistors: From concepts to future applications," *Solid-State Electronics*, vol. 186, p. 108 036, Dec. 2021. DOI: 10.1016/j.sse.2021.108036.

[14] R. Ranjith, R. S. Komaragiri, and K. J. Suja, "Reconfigurable tunnel field effect transistor exhibiting reduced ambipolar behaviour," in *2016 IEEE Annual India Conference (INDICON)*, Bangalore, India: IEEE, Dec. 2016, pp. 1–5, ISBN: 978-1-5090-3646-2. DOI: 10.1109/INDICON.2016.7838995.

[15] P. N. Hai, S. Sugahara, and M. Tanaka, "Reconfigurable Logic Gates Using Single-Electron Spin Transistors," *Japanese Journal of Applied Physics*, vol. 46, no. 10R, p. 6579, Oct. 2007. DOI: 10.1143/JJAP.46.6579.

[16] Y.-M. Lin, J. Appenzeller, and P. Avouris, "Novel carbon nanotube FET design with tunable polarity," in *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.*, IEEE, 2004, pp. 687–690.

[17] Y.-M. Lin, J. Appenzeller, J. Knoch, *et al.*, "High-Performance Carbon Nanotube Field-Effect Transistor With Tunable Polarities," *IEEE Transactions On Nanotechnology*, vol. 4, no. 5, pp. 481–489, Sep. 2005. DOI: 10.1109/TNANO.2005.851427.

[18] W. M. Weber, L. Geelhaar, A. P. Graham, *et al.*, "Silicon-nanowire transistors with intruded nickel-silicide contacts," *Nano Letters*, vol. 6, no. 12, pp. 2660–2666, 2006. DOI: 10.1021/nl0613858. eprint: https://doi.org/10.1021/nl0613858.

[19] W. M. Weber, L. Geelhaar, L. Lamagna, *et al.*, "Tuning the polarity of si-nanowire transistors without the use of doping," in *2008 8th IEEE Conference on Nanotechnology*, 2008, pp. 580–581. DOI: 10.1109/NANO.2008.171.

[20] M. De Marchi, D. Sacchetto, S. Frache, *et al.*, "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire FETs," in *2012 International Electron Devices Meeting*, San Francisco, CA, USA: IEEE, Dec. 2012, pp. 8.4.1–8.4.4, ISBN: 978-1-4673-4871-3 978-1-4673-4872-0 978-1-4673-4870-6. DOI: 10.1109/IEDM.2012.6479004.

[21] F. Wessely, T. Krauss, and U. Schwalke, "CMOS without doping: Multi-gate silicon-nanowire field-effect-transistors," *Solid-State Electronics*, vol. 70, pp. 33–38, Apr. 2012. DOI: 10.1016/j.sse.2011.11.011.

[22] F. Wessely, T. Krauss, and U. Schwalke, "Reconfigurable CMOS with undoped silicon nanowire midgap Schottky-barrier FETs," *Microelectronics Journal*, vol. 44, no. 12, pp. 1072–1076, Dec. 2013. DOI: 10.1016/j.mejo.2012.08.004.

[23] T. Krauss, F. Wessely, and U. Schwalke, "Dual Metal-Gate Planar Field-Effect Transistor for Electrostatically Doped CMOS Design," *arXiv: Materials Science*, May 2014.

[24] T. Krauss, F. Wessely, and U. Schwalke, "An electrostatically doped planar device concept," in *2014 9th IEEE International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS)*, Santorini, Greece: IEEE, May 2014, pp. 1–4, ISBN: 978-1-4799-4972-4. DOI: 10.1109/DTIS.2014.6850650.

[25] T. Krauss, F. Wessely, and U. Schwalke, "Dual Metal-Gate Planar Field-Effect Transistor for Electrostatically Doped CMOS Design," 2014. DOI: 10.48550/ARXIV.1405.7562.

[26] T. A. Krauss, F. Wessely, and U. Schwalke, "Novel Electrostatically Doped Planar Field-Effect Transistor for High Temperature Applications," *ECS Transactions*, vol. 64, no. 12, pp. 11–24, Aug. 2014. DOI: 10.1149/06412.0011ecst.

[27] T. A. Krauss, "Planar electrostatically doped reconfigurable schottky barrier FDSOI field-effect transistor structures," Ph.D. dissertation, Technische Universität Darmstadt, Darmstadt, Jun. 2019.

[28] A. Heinzig, S. Slesazeck, F. Kreupl, *et al.*, "Reconfigurable Silicon Nanowire Transistors," *Nano Letters*, vol. 12, no. 1, pp. 119–124, Jan. 2012. DOI: 10.1021/nl203094h.

[29] A. Heinzig, T. Mikolajick, J. Trommer, *et al.*, "Dually Active Silicon Nanowire Transistors and Circuits with Equal Electron and Hole Transport," *Nano Letters*, vol. 13, no. 9, pp. 4176–4181, Sep. 2013. DOI: 10.1021/nl401826u.

[30] T. Baldauf, A. Heinzig, J. Trommer, *et al.*, "Tuning the tunneling probability by mechanical stress in Schottky barrier based reconfigurable nanowire transistors," *Solid-State Electronics*, vol. 128, pp. 148–154, Feb. 2017. DOI: 10.1016/j.sse.2016.10.009.

[31] V. Sessi, M. Simon, S. Slesazeck, *et al.*, "S2–2 back-bias reconfigurable field effect transistor: A flexible add-on functionality for 22 nm FDSOI," in *2021 Silicon Nanoelectronics Workshop (SNW)*, 2021, pp. 1–2.

[32] Maik Simon, Halid Mulaosmanovic, Violetta Sessi, *et al.*, "Three-to-one analog signal modulation with a single back-bias-controlled reconfigurable transistor," *Nature Communications*, vol. 13, no. 1, Nov. 2022. DOI: 10.1038/s41467-022-34533-w.

[33] P. Magnone, F. Crupi, M. Alioto, *et al.*, "Understanding the Potential and the Limits of Germanium pMOSFETs for VLSI Circuits From Experimental Measurements," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 9, pp. 1569–1582, Sep. 2011. DOI: 10.1109/TVLSI.2010.2053226.

[34] J. Trommer, A. Heinzig, U. Mühle, *et al.*, "Enabling Energy Efficiency and Polarity Control in Germanium Nanowire Transistors by Individually Gated Nanojunctions," *ACS Nano*, vol. 11, no. 2, pp. 1704–1711, Feb. 2017. DOI: 10.1021/acsnano.6b07531.

[35] R. Böckle, M. Sistani, K. Eysin, *et al.*, "Gate-Tunable Negative Differential Resistance in Next-Generation Ge Nanodevices and their Performance Metrics," *Advanced Electronic Materials*, vol. 7, no. 3, p. 2 001 178, Mar. 2021. DOI: 10.1002/aelm.202001178.

[36] M. Sistani, R. Böckle, D. Falkensteiner, *et al.*, "Nanometer-Scale Ge-Based Adaptable Transistors Providing Programmable Negative Differential Resistance Enabling Multi-valued Logic," *ACS Nano*, vol. 15, no. 11, pp. 18 135–18 141, Nov. 2021. DOI: 10.1021/acsnano.1c06801.

[37] T. Krauss, F. Wessely, and U. Schwalke, "Electrostatically Doped Planar Field-Effect Transistor for High Temperature Applications," *ECS Journal of Solid State Science and Technology*, vol. 4, no. 5, Q46–Q50, 2015. DOI: 10.1149/2.0021507jss.

[38] T. A. Krauss, F. Wessely, and U. Schwalke, "Favorable Combination of Schottky Barrier and Junctionless Properties in Field-Effect Transistors for High Temperature Applications," *ECS Transactions*, vol. 75, no. 13, pp. 57–63, Aug. 2016. DOI: 10.1149/07513.0057ecst.

[39] M. Schwarz, A. Kloes, and D. Flandre, "Temperature-dependent performance of Schottky-Barrier FET ultra-low-power diode," *Solid-State Electronics*, vol. 184, p. 108 124, Oct. 2021. DOI: 10.1016/j.sse.2021.108124.

[40] S. M. Sze, K. K. Ng, and Y. Li, *Physics of Semiconductor Devices*, Fourth edition. Hoboken, NJ, USA: Wiley, 2021, ISBN: 978-1-119-42911-1.

[41] G. Galderisi, C. Beyer, T. Mikolajick, *et al.*, "Insights into the Temperature-Dependent Switching Behavior of Three-Gated Reconfigurable Field-Effect Transistors," *physica status solidi (a)*, vol. 220, no. 13, p. 2 300 019, Jul. 2023. DOI: 10.1002/pssa.202300019.

[42] Y. Han, J. Sun, F. Xi, *et al.*, "Cryogenic characteristics of UTBB SOI Schottky-Barrier MOSFETs," *Solid-State Electronics*, vol. 194, p. 108 351, Aug. 2022. DOI: 10.1016/j.sse.2022.108351.

[43] M. Reuter, J. Wilm, A. Kramer, *et al.*, "Machine Learning-Based Compact Model Design for Reconfigurable FETs," *IEEE Journal of the Electron Devices Society*, vol. 12, pp. 310–317, 2024. DOI: 10.1109/JEDS.2024.3386113.

[44] T. Krauss, F. Wessely, and U. Schwalke, "Reconfigurable electrostatically doped 2.5-gate planar field-effect transistors for dopant-free CMOS," in *2018 13th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS)*, Taormina: IEEE, Apr. 2018, pp. 1–4, ISBN: 978-1-5386-5291-6. DOI: 10.1109/DTIS.2018.8368567.

[45] M. Reuter, J. Pfau, T. A. Krauss, *et al.*, "From MOSFETs to Ambipolar Transistors: Standard Cell Synthesis for the Planar RFET Technology," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 1, pp. 114–125, Jan. 2021. DOI: 10.1109/TCSI.2020.3035889.

[46] M. Reuter, T. Krauss, M. Fathipour, *et al.*, "From MOSFETs to Ambipolar Transistors: A Static DeFET Inverter Cell for SOI," *Asia Pacific Conference on Circuits and Systems*, pp. 113–116, Nov. 2019. DOI: 10.1109/apccas47518.2019.8953083.

[47] F. Gerfers, N. Lotfi, E. Wittenhagen, *et al.*, "Body-Bias Techniques in CMOS 22FDX® for Mixed-Signal Circuits and Systems," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Genoa, Italy: IEEE, Nov. 2019, pp. 466–469, ISBN: 978-1-72810-996-1. DOI: 10.1109/ICECS46596.2019.8964676.

[48] J. Pfau, J. Hernandez, M. Reuter, *et al.*, "Co-Simulating Region-Based Dynamic Voltage Scaling for FPGA Architecture Design," in *2023 IEEE Nordic Circuits and Systems Conference (NorCAS)*, Aalborg, Denmark: IEEE, Oct. 2023, pp. 1–7, ISBN: 9798350337570. DOI: 10.1109/NorCAS58970.2023.10305486.

[49] J. Trommer, A. Heinzig, S. Slesazeck, *et al.*, "Elementary Aspects for Circuit Implementation of Reconfigurable Nanowire Transistors," *IEEE Electron Device Letters*, vol. 35, no. 1, pp. 141–143, Jan. 2014. DOI: 10.1109/LED.2013.2290555.

[50] I. O'Connor, J. Liu, F. Gaffiot, *et al.*, "CNTFET Modeling and Reconfigurable Logic-Circuit Design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 11, pp. 2365–2379, Nov. 2007. DOI: 10.1109/TCSI.2007.907835.

[51] S. Rai, S. Patnaik, A. Rupani, *et al.*, "Security Promises and Vulnerabilities in Emerging Reconfigurable Nanotechnology-Based Circuits," *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1, 2020. DOI: 10.1109/TETC.2020.3039375.

[52] M. Reuter, J. Pfau, T. Krauss, *et al.*, "Towards Ambipolar Planar Devices: The DeFET Device in Area Constrained XOR Applications," *2020 IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*, p. 9 069 043, Feb. 2020. DOI: 10.1109/lascas45839.2020.9069043.

[53] M. Ben Jamaa, K. Mohanram, and G. De Micheli, "Novel library of logic gates with ambipolar CNTFETs: Opportunities for multi-level logic synthesis," in *2009 Design, Automation & Test in Europe Conference & Exhibition*, Nice: IEEE, Apr. 2009, pp. 622–627, ISBN: 978-1-4244-3781-8 978-3-9810801-5-5. DOI: 10.1109/DATE.2009.5090742.

[54] P.-E. Gaillardon, M. De Marchi, L. Amaru, *et al.*, "Towards structured ASICs using polarity-tunable Si nanowire transistors," *Design Automation Conference*, p. 123, May 2013. DOI: 10.1145/2463209.2488886.

[55] J. Trommer, A. Heinzig, T. Baldauf, *et al.*, "Reconfigurable nanowire transistors with multiple independent gates for efficient and programmable combinational circuits," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 169–174.

[56] C. Cakirlar, M. Simon, G. Galderisi, *et al.*, "Cross-shape reconfigurable field effect transistor for flexible signal routing," *Materials Today Electronics*, vol. 4, p. 100 040, Jun. 2023. DOI: 10.1016/j.mtelec.2023.100040.

[57] C. Pan, C.-Y. Wang, S.-J. Liang, *et al.*, "Reconfigurable logic and neuromorphic circuits based on electrically tunable two-dimensional homojunctions," *Nature Electronics*, vol. 3, no. 7, pp. 383–390, Jun. 2020. DOI: 10.1038/s41928-020-0433-9.

[58] A. Rupani, S. Rai, and A. Kumar, "Exploiting Emerging Reconfigurable Technologies for Secure Devices," in *2019 22nd Euromicro Conference on Digital System Design (DSD)*, Kallithea, Greece: IEEE, Aug. 2019, pp. 668–671, ISBN: 978-1-72812-862-7. DOI: 10.1109/DSD.2019.00107.

[59] Maximilian Reuter, Andreas Kramer, Tillmann A. Krauss, *et al.*, "Reconfiguring an RFET Based Differential Amplifier," *2022 IEEE 40th Central America and Panama Convention (CONCAPAN)*, 2022. DOI: 10.1109/concapan48024.2022.9997726.

[60] S. Bobba, M. De Marchi, Y. Leblebici, *et al.*, "Physical synthesis onto a Sea-of-Tiles with double-gate silicon nanowire transistors," in *Proceedings of the 49th Annual Design Automation Conference*, San Francisco California: ACM, Jun. 2012, pp. 42–47, ISBN: 978-1-4503-1199-1. DOI: 10.1145/2228360.2228369.

[61] S. Rai, A. Rupani, D. Walter, *et al.*, "A physical synthesis flow for early technology evaluation of silicon nanowire based reconfigurable FETs," *Design, Automation and Test in Europe*, pp. 605–608, Mar. 2018. DOI: 10.23919/date.2018.8342080.

[62] A. Krinke, S. Rai, A. Kumar, *et al.*, "Exploring Physical Synthesis for Circuits based on Emerging Reconfigurable Nanotechnologies," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, Munich, Germany: IEEE, Nov. 2021, pp. 1–9, ISBN: 978-1-66544-507-8. DOI: 10.1109/ICCAD51958.2021.9643439.

[63] *Virtuoso Custom IC Design Environment, Version IC6.1.8-64b.500.21*, Cadence, 2022.

[64] N. Lophitis, A. Arvanitopoulos, S. Perkins, *et al.*, "TCAD Device Modelling and Simulation of Wide Bandgap Power Semiconductors," in *Disruptive Wide Bandgap Semiconductors, Related Technologies, and Their Applications*, Y. K. Sharma, Ed., InTech, Sep. 2018, ISBN: 978-1-78923-668-2 978-1-78923-669-9. DOI: 10.5772/intechopen.76062.

[65] *Sentaurus™ Device User Guide U-2022.12*, Dec. 2022.

[66] G. Lehner, *Elektromagnetische Feldtheorie*. Berlin/Heidelberg: Springer-Verlag, 2006, ISBN: 978-3-540-26550-4. DOI: 10.1007/3-540-29428-7.

[67] I. N. Toptygin, *Electromagnetic Phenomena in Matter: Statistical and Quantum Approaches*, 1st ed. Wiley, Mar. 2015, ISBN: 978-3-527-41178-8 978-3-527-69347-4. DOI: 10.1002/9783527693474.

[68] O. Schenk, K. Gärtner, and W. Fichtner, "Efficient Sparse LU Factorization with Left-Right Looking Strategy on Shared Memory Multiprocessors," *BIT Numerical Mathematics*, vol. 40, no. 1, pp. 158–176, Mar. 2000. DOI: 10.1023/A:1022326604210.

[69] O. Schenk, "Scalable parallel sparse LU factorization methods on shared memory multiprocessors," Ph.D. dissertation, ETH Zurich, 2000, 131 S. DOI: 10.3929/ETHZ-A-003876213.

[70] R. Bank, W. Coughran, W. Fichtner, *et al.*, "Transient Simulation of Silicon Devices and Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 4, no. 4, pp. 436–451, Oct. 1985. DOI: 10.1109/TCAD.1985.1270142.

[71] A. Jankovic, G. Chaudhary, and F. Goia, "Designing the design of experiments (DOE) – An investigation on the influence of different factorial designs on the characterization of complex systems," *Energy and Buildings*, vol. 250, p. 111 298, Nov. 2021. DOI: 10.1016/j.enbuild.2021.111298.

[72] B. Durakovic, "Design of experiments application, concepts, examples: State of the art," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 5, no. 3, Dec. 2017. DOI: 10.21533/pen.v5i3.145.

[73] R. A. Fisher, *The Design of Experiments*, 9. ed. New York: Hafner Press, 1974, ISBN: 978-0-02-844690-5.

[74] Y. S. Chauhan, S. Venugopalan, M.-A. Chalkiadaki, *et al.*, "BSIM6: Analog and RF Compact Model for Bulk MOSFET," *IEEE Transactions on Electron Devices*, vol. 61, no. 2, pp. 234–244, Feb. 2014. DOI: 10.1109/TED.2013.2283084.

[75] C. C. McAndrew, “Compact Models for MOS Transistors: Successes and Challenges,” *IEEE Transactions on Electron Devices*, vol. 66, no. 1, pp. 12–18, Jan. 2019. DOI: 10.1109/TED.2018.2849943.

[76] D. Root, “Future Device Modeling Trends,” *IEEE Microwave Magazine*, vol. 13, no. 7, pp. 45–59, Nov. 2012. DOI: 10.1109/MMM.2012.2216095.

[77] Y. Singh Chauhan, S. Venugopalan, M. A. Karim, *et al.*, “BSIM - Industry standard compact MOSFET models,” in *2012 Proceedings of the European Solid-State Device Research Conference (ESSDERC)*, Bordeaux, France: IEEE, Sep. 2012, pp. 46–49, ISBN: 978-1-4673-1708-5 978-1-4673-1707-8 978-1-4673-1706-1. DOI: 10.1109/ESSDERC.2012.6343330.

[78] M. S. Lundstrom and D. A. Antoniadis, “Compact Models and the Physics of Nanoscale FETs,” *IEEE Transactions on Electron Devices*, vol. 61, no. 2, pp. 225–233, Feb. 2014. DOI: 10.1109/TED.2013.2283253.

[79] G. Coram, “How to (and how not to) write a compact model in Verilog-A,” in *2004 IEEE International Conference on Cluster Computing (IEEE Cat. No.04EX935)*, San Jose, CA, USA: IEEE, 2004, pp. 97–106, ISBN: 978-0-7803-8615-0. DOI: 10.1109/BMAS.2004.1393990.

[80] C. C. McAndrew, G. Coram, K. K. Gullapalli, *et al.*, “Best Practices for Compact Modeling in Verilog-A,” *IEEE Journal of the Electron Devices Society*, vol. 3, no. 5, pp. 383–396, Jul. 2015. DOI: 10.1109/jeds.2015.2455342.

[81] L. Lemaitre, G. Coram, C. McAndrew, *et al.*, “Extensions to Verilog-A to support compact device modeling,” in *Proceedings of the 2003 IEEE International Workshop on Behavioral Modeling and Simulation*, San Jose, CA, USA: IEEE, 2003, pp. 134–138, ISBN: 978-0-7803-8135-3. DOI: 10.1109/BMAS.2003.1249872.

[82] L. Zhou, B. Hu, B. Wan, *et al.*, “Rapid BSIM model implementation with VHDL-AMS/Verilog-AMS and MCAST compact model compiler,” in *IEEE International [Systems-on-Chip] SOC Conference, 2003. Proceedings.*, Portland, OR, USA: IEEE, 2003, pp. 285–286, ISBN: 978-0-7803-8182-7. DOI: 10.1109/SOC.2003.1241523.

[83] M. Rewieński, “A Perspective on Fast-SPICE Simulation Technology,” in *Simulation and Verification of Electronic and Biological Systems*, P. Li, L. M. Silveira, and P. Feldmann, Eds., Dordrecht: Springer Netherlands, 2011, pp. 23–42, ISBN: 978-94-007-0148-9 978-94-007-0149-6. DOI: 10.1007/978-94-007-0149-6_2.

[84] Chung-Wen Ho, A. Ruehli, and P. Brennan, “The modified nodal approach to network analysis,” *IEEE Transactions on Circuits and Systems*, vol. 22, no. 6, pp. 504–509, Jun. 1975. DOI: 10.1109/TCS.1975.1084079.

[85] G. Gildenblat, Ed., *Compact Modeling*. Dordrecht: Springer Netherlands, 2010, ISBN: 978-90-481-8613-6 978-90-481-8614-3. DOI: 10.1007/978-90-481-8614-3.

[86] S. K. Saha, *Compact Models for Integrated Circuit Design: Conventional Transistors and Beyond*. CRC Press, 2019, ISBN: 978-1-138-82740-0.

[87] B. Sheu, D. Scharfetter, P.-K. Ko, *et al.*, “BSIM: Berkeley short-channel IGFET model for MOS transistors,” *IEEE Journal of Solid-State Circuits*, vol. 22, no. 4, pp. 558–566, Aug. 1987. DOI: 10.1109/JSSC.1987.1052773.

[88] H. Shichman and D. Hodges, “Modeling and simulation of insulated-gate field-effect transistor switching circuits,” *IEEE Journal of Solid-State Circuits*, vol. 3, no. 3, pp. 285–289, Sep. 1968. DOI: 10.1109/JSSC.1968.1049902.

[89] Yuhua Cheng, Min-Chie Jeng, Zhihong Liu, *et al.*, “A physical and scalable I-V model in BSIM3v3 for analog/digital circuit simulation,” *IEEE Transactions on Electron Devices*, vol. 44, no. 2, pp. 277–287, Feb. 1997. DOI: 10.1109/16.557715.

[90] J. Watts, C. McAndrew, C. Enz, *et al.*, “Advanced compact models for MOSFETs,” in *Workshop on Compact Models–Nanotech 2005*, 2005, pp. 9–12.

[91] G. Gildenblat, X. Li, W. Wu, *et al.*, “PSP: An Advanced Surface-Potential-Based MOSFET Model for Circuit Simulation,” *IEEE Transactions on Electron Devices*, vol. 53, no. 9, pp. 1979–1993, Sep. 2006. DOI: 10.1109/TED.2005.881006.

[92] Y. S. Chauhan, M. A. Karim, S. Venugopalan, *et al.*, “BSIM6: Symmetric bulk MOSFET model,” in *Nanotechnology 2012*, ser. Technical Proceedings of the 2012 NSTI Nanotechnology Conference and Expo, NSTI-nanotech 2012, 2012, pp. 724–729, ISBN: 978-1-4665-6275-2.

[93] T. Chen and G. Gildenblat, “Analytical approximation for the MOSFET surface potential,” *Solid-State Electronics*, vol. 45, no. 2, pp. 335–339, Feb. 2001. DOI: 10.1016/S0038-1101(00)00283-5.

[94] C. Sah and H. Pao, “The effects of fixed bulk charge on the characteristics of metal-oxide-semiconductor transistors,” *IEEE Transactions on Electron Devices*, vol. ED-13, no. 4, pp. 393–409, Apr. 1966. DOI: 10.1109/T-ED.1966.15702.

[95] G. Gildenblat, Hailing Wang, Ten-Lon Chen, *et al.*, “SP: An advanced surface-potential-based compact MOSFET model,” *IEEE Journal of Solid-State Circuits*, vol. 39, no. 9, pp. 1394–1406, Sep. 2004. DOI: 10.1109/JSSC.2004.831604.

[96] R. Rios, S. Mudanai, Wei-Kai Shih, *et al.*, “An efficient surface potential solution algorithm for compact MOSFET models,” in *IEDM Technical Digest. IEEE International Electron Devices Meeting, 2004.*, San Francisco, CA, USA: IEEE, 2004, pp. 755–758, ISBN: 978-0-7803-8684-6. DOI: 10.1109/IEDM.2004.1419282.

[97] R Van Langevelde, AJ Scholten, and DBM Klaassen, “Recent enhancements of MOS model 11,” in *Workshop on Compact Modeling, NSTI-Nanotech 2004*, vol. 2, 2004, pp. 60–65.

[98] M. Miura-Mattausch, H. Ueno, M. Tanaka, *et al.*, “HiSIM: A MOSFET model for circuit simulation connecting circuit performance with technology,” in *Digest. International Electron Devices Meeting,*, San Francisco, CA, USA: IEEE, 2002, pp. 109–112, ISBN: 978-0-7803-7462-1. DOI: 10.1109/IEDM.2002.1175790.

[99] M. Bucher, A. Bazigos, F. Krummenacher, *et al.*, “EKV3.0: An advanced charge based MOS transistor model.A design-oriented MOS transistor compact model,” in *TRANSISTOR LEVEL MODELING FOR ANALOG/RF IC DESIGN*, W. Grabinski, B. Nauwelaers, and D. Schreurs, Eds., Dordrecht: Springer Netherlands, 2006, pp. 67–95, ISBN: 978-1-4020-4555-4 978-1-4020-4556-1. DOI: 10.1007/1-4020-4556-5_3.

[100] S. Khandelwal, Y. S. Chauhan, D. D. Lu, *et al.*, "BSIM-IMG: A Compact Model for Ultrathin-Body SOI MOSFETs With Back-Gate Control," *IEEE Transactions on Electron Devices*, vol. 59, no. 8, pp. 2019–2026, Aug. 2012. DOI: 10.1109/TED.2012.2198065.

[101] S. Martinie, O. Rozeau, T. Poiroux, *et al.*, "L-UTSOI: A compact model for low-power analog and digital applications in FDSOI technology," in *2020 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Kobe, Japan: IEEE, Sep. 2020, pp. 311–314, ISBN: 978-4-86348-763-5. DOI: 10.23919/SISPAD49475.2020.9241684.

[102] M. Miura-Mattausch, U. Feldmann, Y. Fukunaga, *et al.*, "Compact modeling of SOI mosfets with ultrathin silicon and BOX layers," *IEEE Transactions on Electron Devices*, vol. 61, no. 2, pp. 255–265, 2014. DOI: 10.1109/TED.2013.2286206.

[103] P. Kushwaha, G. Pahwa, H. Agarwal, *et al.*, *BSIM-IMG103.0.0 Independent multi-gate MOSFET compact model: Technical Manual*, 2020.

[104] D. D. Lu, M. V. Dunga, C.-H. Lin, *et al.*, "A computationally efficient compact model for fully-depleted SOI MOSFETs with independently-controlled front- and back-gates," *Solid-State Electronics*, vol. 62, no. 1, pp. 31–39, Aug. 2011. DOI: 10.1016/j.sse.2010.12.015.

[105] A. Roy, J. Sallese, and C. Enz, "A closed-form charge-based expression for drain current in symmetric and asymmetric double gate MOSFET," *Solid-State Electronics*, vol. 50, no. 4, pp. 687–693, Apr. 2006. DOI: 10.1016/j.sse.2006.03.021.

[106] D. D. Lu, "Compact models for future generation CMOS," Ph.D. dissertation, University of California, Berkeley, 2011.

[107] J. Goguet, S. Fregonese, C. Maneux, *et al.*, "Compact Model of a Dual Gate CNTFET: Description and Circuit Application," in *2008 8th IEEE Conference on Nanotechnology*, Arlington, Texas, USA: IEEE, Aug. 2008, pp. 388–389, ISBN: 978-1-4244-2103-9. DOI: 10.1109/NANO.2008.120.

[108] C. Roemer, G. Darbandy, M. Schwarz, *et al.*, "Uniform DC Compact Model for Schottky Barrier and Reconfigurable Field-Effect Transistors," in *2021 IEEE Latin America Electron Devices Conference (LAEDC)*, Mexico, Mexico: IEEE, Apr. 2021, pp. 1–4, ISBN: 978-1-66541-510-1. DOI: 10.1109/LAEDC51812.2021.9437954.

[109] C. Roemer, G. Darbandy, M. Schwarz, *et al.*, "Physics-Based DC Compact Modeling of Schottky Barrier and Reconfigurable Field-Effect Transistors," *IEEE Journal of the Electron Devices Society*, vol. 10, pp. 416–423, 2022. DOI: 10.1109/JEDS.2021.3136981.

[110] W. Ni, Z. Dong, B. Huang, *et al.*, "A Physic-Based Explicit Compact Model for Reconfigurable Field-Effect Transistor," *IEEE Access*, vol. 9, pp. 46 709–46 716, 2021. DOI: 10.1109/ACCESS.2021.3064961.

[111] V. Litovski, J. Radjenovié, Ž. Mrčarica, *et al.*, "MOS transistor modelling using neural network," *Electronics Letters*, vol. 28, no. 18, p. 1766, 1992. DOI: 10.1049/el:19921124.

[112] J. Hutchins, S. Alam, A. Zeumault, *et al.*, "A Generalized Workflow for Creating Machine Learning-Powered Compact Models for Multi-State Devices," *IEEE Access*, vol. 10, pp. 115 513–115 519, 2022. DOI: 10.1109/ACCESS.2022.3218333.

[113] J. Xu, D. Gunyan, M. Iwamoto, *et al.*, "Drain-Source Symmetric Artificial Neural Network-Based FET Model with Robust Extrapolation Beyond Training Data," in *2007 IEEE/MTT-S International Microwave Symposium*, Honolulu, HI, USA: IEEE, Jun. 2007, pp. 2011–2014, ISBN: 978-1-4244-0687-6 978-1-4244-0688-3. DOI: 10.1109/MWSYM.2007.380244.

[114] J. Wang, Y.-H. Kim, J. Ryu, *et al.*, "Artificial Neural Network-Based Compact Modeling Methodology for Advanced Transistors," *IEEE Transactions on Electron Devices*, vol. 68, no. 3, pp. 1318–1325, Mar. 2021. DOI: 10.1109/TED.2020.3048918.

[115] Y. Zhao, Z. Xu, H. Tang, *et al.*, "Compact Modeling of Advanced Gate-All-Around Nanosheet FETs Using Artificial Neural Network," *Micromachines*, vol. 15, no. 2, p. 218, Jan. 2024. DOI: 10.3390/mi15020218.

[116] F. Klemme, J. Prinz, V. M. Van Santen, *et al.*, "Modeling emerging technologies using machine learning: Challenges and opportunities," in *Proceedings of the 39th International Conference on Computer-Aided Design*, Virtual Event USA: ACM, Nov. 2020, pp. 1–9, ISBN: 978-1-4503-8026-3. DOI: 10.1145/3400302.3415770.

[117] H. Habal, D. Tsonev, and M. Schweikardt, "Compact Models for Initial MOSFET Sizing Based on Higher-order Artificial Neural Networks," in *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, Virtual Event Iceland: ACM, Nov. 2020, pp. 111–116, ISBN: 978-1-4503-7519-1. DOI: 10.1145/3380446.3430632.

[118] M. Li, O. Irsoy, C. Cardie, *et al.*, "Physics-Inspired Neural Networks for Efficient Device Compact Modeling," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 2, pp. 44–49, Dec. 2016. DOI: 10.1109/JXCDC.2016.2636161.

[119] Dong Wang, J.-M. Muller, N. Brisebarre, *et al.*, "(M,p,k)-Friendly Points: A Table-Based Method to Evaluate Trigonometric Function," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 9, pp. 711–715, Sep. 2014. DOI: 10.1109/TCSII.2014.2331094.

[120] H. Erdem, "Implementation of software-based sensor linearization algorithms on low-cost microcontrollers," *ISA Transactions*, vol. 49, no. 4, pp. 552–558, Oct. 2010. DOI: 10.1016/j.isatra.2010.04.004.

[121] V. Saripalli, S. Datta, V. Narayanan, *et al.*, "Variation-tolerant ultra low-power heterojunction tunnel FET SRAM design," in *2011 IEEE/ACM International Symposium on Nanoscale Architectures*, San Diego, CA, USA: IEEE, Jun. 2011, pp. 45–52, ISBN: 978-1-4577-0993-7. DOI: 10.1109/NANOARCH.2011.5941482.

[122] X. Li, Fan Yang, Fan Yang, *et al.*, "MOS Table Models for Fast and Accurate Simulation of Analog and Mixed-Signal Circuits Using Efficient Oscillation-Diminishing Interpolations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 9, pp. 1481–1494, Mar. 2015. DOI: 10.1109/tcad.2015.2413392.

[123] S. Rai, S. Srinivasa, P. Cadareanu, *et al.*, "Emerging reconfigurable nanotechnologies: Can they support future electronics?" In *Proceedings of the International Conference on Computer-Aided Design*, San Diego California: ACM, Nov. 2018, pp. 1–8, ISBN: 978-1-4503-5950-4. DOI: 10.1145/3240765.3243472.

[124] *Verilog-AMS Language Reference Manual Version 2.4*, Jun. 2014.

[125] W. H. Burnham, “Memory, Historically and Experimentally Considered. I. An Historical Sketch of the Older Conceptions of Memory,” *The American Journal of Psychology*, vol. 2, no. 1, p. 39, Nov. 1888. DOI: 10.2307/1411406. JSTOR: 1411406.

[126] H. Wang and B. Raj, “On the Origin of Deep Learning,” 2017. DOI: 10.48550/ARXIV.1702.07800.

[127] N. R. Draper and H. Smith, *Applied Regression Analysis* (Wiley Series in Probability and Statistics), 1st ed. Wiley, Apr. 1998, ISBN: 978-0-471-17082-2 978-1-118-62559-0. DOI: 10.1002/9781118625590.

[128] Zhou, Zhi-Hua, *Machine Learning*. Springer Nature, 2021, ISBN: 978-981-15-1967-3. DOI: 10.1007/978-981-15-1967-3.

[129] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, Dec. 1943. DOI: 10.1007/BF02478259.

[130] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.,” *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. DOI: 10.1037/h0042519.

[131] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham: Springer International Publishing, 2018, ISBN: 978-3-319-94462-3 978-3-319-94463-0. DOI: 10.1007/978-3-319-94463-0.

[132] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, Expanded ed. Cambridge, Mass: MIT Press, 1988, ISBN: 978-0-262-63111-2.

[133] S. Linnainmaa, “Taylor expansion of the accumulated rounding error,” *BIT*, vol. 16, no. 2, pp. 146–160, Jun. 1976. DOI: 10.1007/BF01931367.

[134] J. Martens, “Deep learning via Hessian-free optimization,” in *International Conference on Machine Learning*, 2010.

[135] P. J. Werbos, “Applications of advances in nonlinear sensitivity analysis,” in *System Modeling and Optimization*, R. F. Drenick and F. Kozin, Eds., vol. 38, Berlin/Heidelberg: Springer-Verlag, 1982, pp. 762–770, ISBN: 978-3-540-11691-2. DOI: 10.1007/BFb0006203.

[136] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986. DOI: 10.1038/323533a0.

[137] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, Dec. 1989. DOI: 10.1007/BF02551274.

[138] T. M. Mitchell, *Machine Learning* (McGraw-Hill Series in Computer Science), Nachdr. New York: McGraw-Hill, 2013, ISBN: 978-0-07-115467-3 978-0-07-042807-2.

[139] A. Apicella, F. Donnarumma, F. Isgrò, *et al.*, “A survey on modern trainable activation functions,” *Neural Networks*, vol. 138, pp. 14–32, Jun. 2021. DOI: 10.1016/j.neunet.2021.01.026.

[140] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, Sep. 2022. DOI: 10.1016/j.neucom.2022.06.111.

[141] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994. DOI: 10.1109/72.279181.

[142] R. Hahnloser and H. S. Seung, "Permitted and forbidden sets in symmetric threshold-linear networks," in *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, and V. Tresp, Eds., vol. 13, MIT Press, 2000.

[143] R. Elshawi, M. Maher, and S. Sakr, "Automated Machine Learning: State-of-The-Art and Open Challenges," 2019. DOI: 10.48550/ARXIV.1906.02287.

[144] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 10, pp. 281–305, 2012.

[145] H. J. Kushner, "A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, Mar. 1964. DOI: 10.1115/1.3653121.

[146] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, *et al.*, Eds., vol. 25, Curran Associates, Inc., 2012.

[147] A. Klein, S. Falkner, S. Bartels, *et al.*, "Fast bayesian optimization of machine learning hyperparameters on large datasets," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, A. Sign and J. Zhu, Eds., ser. Proceedings of Machine Learning Research, vol. 54, PMLR, Apr. 2017, pp. 528–536.

[148] P. I. Frazier, "A Tutorial on Bayesian Optimization," 2018. DOI: 10.48550/ARXIV.1807.02811.

[149] L. Li, K. G. Jamieson, G. DeSalvo, *et al.*, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *Journal of Machine Learning Research*, vol. 18, 185:1–185:52, 2016.

[150] J. T. Wilson, F. Hutter, and M. P. Deisenroth, "Maximizing acquisition functions for Bayesian optimization," 2018. DOI: 10.48550/ARXIV.1805.10196.

[151] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," *Towards Global Optimization*, vol. 2, no. 117-129, p. 2, 1978.

[152] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998. DOI: 10.1023/A:1008306431147.

[153] S. Rana, C. Li, S. Gupta, *et al.*, "High dimensional Bayesian optimization with elastic Gaussian process," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, 2017-08-06/2017-08-11, pp. 2883–2891.

[154] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, A. Gretton and C. C. Robert, Eds., ser. Proceedings of Machine Learning Research, vol. 51, Cadiz, Spain: PMLR, 2016-05-09/2016-05-11, pp. 240–248.

[155] P. Luong, S. Gupta, D. Nguyen, *et al.*, "Bayesian Optimization with Discrete Variables," in *AI 2019: Advances in Artificial Intelligence*, J. Liu and J. Bailey, Eds., vol. 11919, Cham: Springer International Publishing, 2019, pp. 473–484, ISBN: 978-3-030-35287-5 978-3-030-35288-2. DOI: 10.1007/978-3-030-35288-2_38.

[156] L. Ziyin, T. Hartwig, and M. Ueda, "Neural Networks Fail to Learn Periodic Functions and How to Fix It," 2020. DOI: 10.48550/ARXIV.2006.08195.

[157] P. Belcák and R. Wattenhofer, "Periodic Extrapolative Generalisation in Neural Networks," 2022. DOI: 10.48550/ARXIV.2209.10280.

[158] M. Steininger, K. Kobs, P. Davidson, *et al.*, "Density-based weighting for imbalanced regression," *Machine Learning*, vol. 110, no. 8, pp. 2187–2211, Aug. 2021. DOI: 10.1007/s10994-021-06023-5.

[159] W. L. Cava, P. Orzechowski, Bogdan Burlacu, *et al.*, "Contemporary Symbolic Regression Methods and their Relative Performance," *NeurIPS Datasets and Benchmarks*, 2021.

[160] J. R. Koza, *Genetic Programming. 1: On the Programming of Computers by Means of Natural Selection* (Complex Adaptive Systems). Cambridge, Mass.: MIT Press, 1992, ISBN: 978-0-262-52791-0.

[161] C. Rudin, "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead," 2018. DOI: 10.48550/ARXIV.1811.10154.

[162] C. Wilstrup and Jaan Kasak, "Symbolic regression outperforms other models for small data sets," *arXiv.org*, 2021. DOI: 10.48550/arXiv.2103.15147.

[163] J. R. Koza and R. Poli, "Genetic Programming," in *Search Methodologies*, E. K. Burke and G. Kendall, Eds., Boston, MA: Springer US, 2005, pp. 127–164, ISBN: 978-0-387-23460-1. DOI: 10.1007/0-387-28356-0_5.

[164] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, 1975, ISBN: 978-0-262-27555-2. DOI: 10.7551/mitpress/1090.001.0001.

[165] M. F. Korns, "Accuracy in Symbolic Regression," in *Genetic Programming Theory and Practice IX*, R. Riolo, E. Vladislavleva, and J. H. Moore, Eds., New York, NY: Springer New York, 2011, pp. 129–151, ISBN: 978-1-4614-1769-9 978-1-4614-1770-5. DOI: 10.1007/978-1-4614-1770-5_8.

[166] Y. Jin, W. Fu, J. Kang, *et al.*, "Bayesian Symbolic Regression," 2019. DOI: 10.48550/ARXIV.1910.08892.

[167] S.-M. Udrescu and M. Tegmark, "AI Feynman: A Physics-Inspired Method for Symbolic Regression," 2019. DOI: 10.48550/ARXIV.1905.11481.

[168] B. Burlacu, G. Kronberger, and M. Kommenda, "Operon C++: An efficient genetic programming framework for symbolic regression," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, Cancún Mexico: ACM, Jul. 2020, pp. 1562–1570, ISBN: 978-1-4503-7127-8. DOI: 10.1145/3377929.3398099.

[169] B. K. Petersen, M. Landajuela, T. N. Mundhenk, *et al.*, "Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients," 2019. DOI: 10.48550/ARXIV.1912.04871.

[170] M. Landajuela, C. S. Lee, J. Yang, *et al.*, "A unified framework for deep symbolic regression," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, *et al.*, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 33 985–33 998.

[171] T. McConaghy, "FFX: Fast, Scalable, Deterministic Symbolic Regression Technology," in *Genetic Programming Theory and Practice IX*, R. Riolo, E. Vladislavleva, and J. H. Moore, Eds., New York, NY: Springer New York, 2011, pp. 235–260, ISBN: 978-1-4614-1769-9 978-1-4614-1770-5. DOI: 10.1007/978-1-4614-1770-5_13.

[172] *SRBench Competition 2022*, https://cavalab.org/srbench/competition-2022.

[173] C. Wilstrup, "Method of deriving a correlation," pat. US 11,537,686, Dec. 2022.

[174] K. R. Broløs, M. V. Machado, C. Cave, *et al.*, *An Approach to Symbolic Regression Using Feyn*, 2021. DOI: 10.48550/ARXIV.2104.05417.

[175] *QLattice/Feyn API Documentation*, https://docs.abzu.ai/docs/api_reference/feyn.

[176] R. P. Feynman, *Quantum Mechanics and Path Integrals*. New York : McGraw-Hill, 1965.

[177] S. C. Tornay, *Ockham: Studies and Selections*, William, Ed. La Salle, Ill.: The Open court publishing company, 1938.

[178] P. Domingos, "Occam's two razors: The sharp and the blunt," in *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, ser. KDD'98, New York, NY: AAAI Press, 1998, pp. 37–43.

[179] Anselm Blumer, A. Blumer, Andrzej Ehrenfeucht, *et al.*, "Occam's razor," *Information Processing Letters*, vol. 24, no. 6, pp. 377–380, Apr. 1987. DOI: 10.1016/0020-0190(87)90114-1.

[180] K. Sun and F. Nielsen, "A Geometric Modeling of Occam's Razor in Deep Learning," 2019. DOI: 10.48550/ARXIV.1905.11027.

[181] Y. Wang, Z. Huang, and X. Hong, "S-prompts learning with pre-trained transformers: An occam's razor for domain incremental learning," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, *et al.*, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 5682–5695.

[182] L. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, Oct./1990. DOI: 10.1109/34.58871.

[183] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Multiple Classifier Systems*, G. Goos, J. Hartmanis, and J. Van Leeuwen, Eds., vol. 1857, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 1–15, ISBN: 978-3-540-67704-8 978-3-540-45014-6. DOI: 10.1007/3-540-45014-9_1.

[184] K. Li, W. Liu, K. Zhao, *et al.*, "A Novel Dynamic Weight Neural Network Ensemble Model," *International Journal of Distributed Sensor Networks*, vol. 11, no. 8, p. 862 056, Aug. 2015. DOI: 10.1155/2015/862056.

[185] R. E. Schapire, "The strength of weak learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, Jun. 1990. DOI: 10.1007/BF00116037.

[186] Z. Zhao, Y. Zhang, and H. Liao, "Design of ensemble neural network using the Akaike information criterion," *Engineering Applications of Artificial Intelligence*, vol. 21, no. 8, pp. 1182–1188, Dec. 2008. DOI: 10.1016/j.engappai.2008.02.007.

[187] Yong Liu and Xin Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, vol. 29, no. 6, pp. 716–725, Dec./1999. DOI: 10.1109/3477.809027.

[188] R. Adhikari and G. Verma, "Time Series Forecasting Through a Dynamic Weighted Ensemble Approach," in *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics*, A. Nagar, D. P. Mohapatra, and N. Chaki, Eds., vol. 43, New Delhi: Springer India, 2016, pp. 455–465, ISBN: 978-81-322-2537-9 978-81-322-2538-6. DOI: 10.1007/978-81-322-2538-6_47.

[189] P. Branco, L. Torgo, and R. P. Ribeiro, "Pre-processing approaches for imbalanced distributions in regression," *Neurocomputing*, vol. 343, pp. 76–99, May 2019. DOI: 10.1016/j.neucom.2018.11.100.

[190] E. Vladislavleva, G. Smits, and D. Den Hertog, "On the Importance of Data Balancing for Symbolic Regression," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 2, pp. 252–277, Apr. 2010. DOI: 10.1109/TEVC.2009.2029697.

[191] B. Krawczyk, "Learning from imbalanced data: Open challenges and future directions," *Progress in Artificial Intelligence*, vol. 5, no. 4, pp. 221–232, Nov. 2016. DOI: 10.1007/s13748-016-0094-0.

[192] G. Haixiang, L. Yijing, J. Shang, *et al.*, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, May 2017. DOI: 10.1016/j.eswa.2016.12.035.

[193] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, Jun. 2004. DOI: 10.1145/1007730.1007735.

[194] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning). Cambridge, Massachusetts: The MIT Press, 2016, ISBN: 978-0-262-03561-3.

[195] V. R. Joseph and A. Vakayil, "SPlit: An Optimal Method for Data Splitting," *Technometrics*, vol. 64, no. 2, pp. 166–176, Apr. 2022. DOI: 10.1080/00401706.2021.1921037.

[196] R. W. Kennard and L. A. Stone, "Computer Aided Design of Experiments," *Technometrics*, vol. 11, no. 1, pp. 137–148, Feb. 1969. DOI: 10.1080/00401706.1969.10490666.

[197] Y. Cui, M. Jia, T.-Y. Lin, *et al.*, "Class-Balanced Loss Based on Effective Number of Samples," 2019. DOI: 10.48550/ARXIV.1901.05555.

[198] J. Sietsma and R. J. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67–79, Jan. 1991. DOI: 10.1016/0893-6080(91)90033-2.

[199] L. Yaeger, R. Lyon, and B. Webb, "Effective training of a neural network character classifier for word recognition," in *Advances in Neural Information Processing Systems*, M. Mozer, M. Jordan, and T. Petsche, Eds., vol. 9, MIT Press, 1996.

[200] S.-H. Hwang and S. E. Whang, "RegMix: Data Mixing Augmentation for Regression," 2021. DOI: 10.48550/ARXIV.2106.03374.

[201] S. S. Raju, B. Wang, K. Mehta, *et al.*, "Application of Noise to Avoid Overfitting in TCAD Augmented Machine Learning," in *2020 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Kobe, Japan: IEEE, Sep. 2020, pp. 351–354, ISBN: 978-4-86348-763-5. DOI: 10.23919/SISPAD49475.2020.9241654.

[202] Y. S. Bankapalli and H. Y. Wong, "TCAD Augmented Machine Learning for Semiconductor Device Failure Troubleshooting and Reverse Engineering," in *2019 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Udine, Italy: IEEE, Sep. 2019, pp. 1–4, ISBN: 978-1-72810-940-4. DOI: 10.1109/SISPAD.2019.8870467.

[203] Chien-Ting Tung, M.-Y. Kao, and Y.-H. Liao, "Neural Network-Based and Modeling With High Accuracy and Potential Model Speed," *IEEE Transactions on Electron Devices*, vol. 69, no. 11, pp. 6476–6479, Nov. 2022. DOI: 10.1109/ted.2022.3208514.

[204] S. K. Morley, T. V. Brito, and D. T. Welling, "Measures of Model Performance Based On the Log Accuracy Ratio," *Space Weather*, vol. 16, no. 1, pp. 69–88, Jan. 2018. DOI: 10.1002/2017SW001669.

[205] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51–56. DOI: 10.25080/Majora-92bf1922-00a.

[206] Maximilian Reuter, Dakyung Lee, David Riehl, *et al.*, "Quick Compact Model Development Through Slow Transient Simulation: An Alternative Approach to Table Models for Emerging Nanodevices," *IEEE International New Circuits and Systems Conference*, Jun. 2022. DOI: 10.1109/newcas52662.2022.9842125.

[207] M. Reuter, A. Kramer, D. Lee, *et al.*, "Generating Predictive Models for Emerging Semiconductor Devices," *IEEE Journal of the Electron Devices Society*, vol. 12, pp. 56–64, 2024. DOI: 10.1109/JEDS.2023.3347306.

[208] *LTSpice, Version: 17.1.15*, Jun. 2024.

[209] A. K. Boyat and B. K. Joshi, "A Review Paper: Noise Models in Digital Image Processing," 2015. DOI: 10.48550/ARXIV.1505.03489.

[210] S. O. Rice, "Mathematical Analysis of Random Noise," *Bell System Technical Journal*, vol. 23, no. 3, pp. 282–332, Jul. 1944. DOI: 10.1002/j.1538-7305.1944.tb00874.x.

[211] M. Abadi, A. Agarwal, P. Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, 2015.

[212] F. Chollet *et al.*, *Keras*, 2015.

[213] T. Head, M. Kumar, H. Nahrstaedt, *et al.*, *Scikit-optimize/scikit-optimize*, [object Object], Oct. 2021. DOI: `10.5281/ZENODO.5565057`.

[214] A. Meurer, C. P. Smith, M. Paprocki, *et al.*, "SymPy: Symbolic computing in Python," *PeerJ Computer Science*, vol. 3, e103, Jan. 2017. DOI: `10.7717/peerj-cs.103`.

A Appendix

A.1 Simulation Times of Timing Characterization Simulations

Table A.1: Simulation times (CPU) in s

Cell	**NAND**	
Edge	**A01_B1**	**A1_B01**
Ref.	4.15e+4	4.51e+4
$\mathsf{TAB_F}$	7.19e+01	1.01e+02
$\mathsf{TAB_P}$	8.75e+01	1.02e+02
$\mathsf{NN_F}$	5.47e+00	6.22e+00
$\mathsf{NN_P}$	7.07e+00	7.52e+00

Table A.2: Simulation times (CPU) in s

Cell	**NOR**	
Edge	**A01_B0**	**A0_B01**
Ref.	4.32e+4	4.57e+4
$\mathsf{TAB_F}$	9.48e+01	8.01e+01
$\mathsf{TAB_P}$	1.01e+02	8.79e+01
$\mathsf{NN_F}$	7.58e+00	7.15e+00
$\mathsf{NN_P}$	8.82e+00	8.08e+00

Table A.3: Simulation times (CPU) in s

Cell	**XOR**	
Edge	**A01_B0**	**A0_B01**
Ref.	1.05e+5	1.05e+5
$\mathsf{TAB_F}$	1.78e+02	1.78e+02
$\mathsf{TAB_P}$	2.03e+02	2.10e+02
$\mathsf{NN_F}$	1.26e+01	1.25e+01
$\mathsf{NN_P}$	1.52e+01	1.55e+01

Table A.4: Simulation times (CPU) in s

Cell	**XOR**	
Edge	**A01_B1**	**A1_B01**
Ref.	8.85e+4	8.53e+4
TAB_F	1.74e+02	1.83e+02
TAB_P	2.09e+02	2.14e+02
NN_F	1.27e+01	1.30e+01
NN_P	1.53e+01	1.57e+01

Publications

Publications in Peer-Reviewed Journals

M. Reuter, J. Wilm, A. Kramer, N. Bhattacharjee, C. Beyer, J. Trommer, T. Mikolajick and K. Hofmann. "Machine Learning-Based Compact Model Design for Reconfigurable FETs". In: *IEEE Journal of the Electron Devices Society, vol. 12*, pp. 310-317, (2024).
DOI: 10.1109/JEDS.2024.3386113.

M. Reuter, A. Kramer, D. Lee, J. Trommer, N. Bhattacharjee, G. Galderisi, T. Mikolajick and K. Hofmann. "Generating Predictive Models for Emerging Semiconductor Devices". In: *IEEE Journal of the Electron Devices Society, vol. 12*, pp. 56-64, (2023).
DOI: 10.1109/JEDS.2023.3347306.

M. Reuter, J. Pfau, T. A. Krauss, J. Becker and K. Hofmann. "From MOSFETs to Ambipolar Transistors: Standard Cell Synthesis for the Planar RFET Technology". In: *IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 68, no. 1*, pp. 114-125, (2021).
DOI: 10.1109/TCSI.2020.3035889.

Conference Contributions

M. Reuter, D. Lee, D. Riehl and K. Hofmann. "Quick Compact Model Development Through Slow Transient Simulation: An Alternative Approach to Table Models for Emerging Nanodevices." *20th IEEE Interregional NEWCAS Conference (NEWCAS)*, Quebec City, QC, Canada, pp. 485-489, (2022).
DOI: 10.1109/NEWCAS52662.2022.9842125.

M. Reuter, A. Kramer, T. Krauss, J. Pfau, J. Becker and K. Hofmann. "Reconfiguring an RFET Based Differential Amplifier." *IEEE 40th Central America and Panama Convention (CONCAPAN)*, Panama, pp. 1-6, (2022).
DOI: 10.1109/CONCAPAN48024.2022.9997726.

M. Reuter, J. Pfau, T. Krauss, M. Moradinasab, U. Schwalke, J. Becker and K. Hofmann. "Towards Ambipolar Planar Devices: The DeFET Device in Area Constrained XOR Applications." *IEEE 11th Latin American Symposium on Circuits & Systems (LASCAS)*, San Jose, Costa Rica, 2020, pp. 1-4, (2020).
DOI: 10.1109/LASCAS45839.2020.9069043.

M. Reuter, T. Krauss, M. Moradinasab, J. Pfau, U. Schwalke, J. Becker and K. Hofmann. "From MOSFETs to Ambipolar Transistors: A Static DeFET Inverter Cell for SOI." *IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*, Bangkok, Thailand, 2019, pp. 113-116, (2019).
DOI: 10.1109/APCCAS47518.2019.

J. Trommer, **M. Reuter**, N. Bhattacharjee, Y. He, V. Sessi, M. Drescher, M. Zier, M. Simon, K. Ruttloff, K. Li, A. Zeun, A.-S. Seidel, C. Metze, M. Grothe, S. Jansen, G. Galderisi, V. Havel, S. Slesazeck, J. Hoentschel, K. Hofmann, and T. Mikolajick. "Speeding-Up Emerging Device Development Cycles by Generating Models via Machine-Learning directly from Electrical Measurements". *European Solid-State Electronics Research Conference (ESSERC)*, Bruges, Belgium, pp. 217-220, (2024).
DOI: 10.1109/ESSERC62670.2024.10719591.

N. Bhattacharjee, **M. Reuter**, K. Hofmann, T. Mikolajick and J. Trommer. "Single Transistor Analog Building Blocks: Exploiting Back-Bias Reconfigurable Devices." *21st IEEE Interregional NEWCAS Conference (NEWCAS)*, Edinburgh, United Kingdom, pp. 1-5, (2023).
DOI: 10.1109/NEWCAS57931.2023.10198128.

J. Pfau, J. Hernandez, **M. Reuter**, K. Hofmann and J. Becker. "Co-Simulating Region-Based Dynamic Voltage Scaling for FPGA Architecture Design." *IEEE Nordic Circuits and Systems Conference (NorCAS)*, Aalborg, Denmark, pp. 1-7, (2023).
DOI: 10.1109/NorCAS58970.2023.10305486.

J. Pfau, **M. Reuter**, K. Hofmann and J. Becker. "Designing Universal Logic Module FPGA Architectures for Use With Ambipolar Transistor Technology." *International Conference on Field-Programmable Technology (ICFPT)*, Maui, HI, USA, pp. 165-173, (2020).
DOI: 10.1109/ICFPT51103.2020.00031.

J. Pfau, **M. Reuter**, T. Harbaum, K. Hofmann and J. Becker. "A Hardware Perspective on the ChaCha Ciphers: Scalable Chacha8/12/20 Implementations Ranging from 476 Slices to Bitrates of 175 Gbit/s." *32nd IEEE International System-on-Chip Conference (SOCC)*, Singapore, pp. 294-299, (2019).
DOI: 10.1109/SOCC46988.2019.1570548289.

Patents

M. Reuter, K. Hofmann. "Stromspiegel", patent DE 10 2021 121 573.0.

Supervised Theses

N. Dudeja, "Development of a Sub-1V Voltage Reference for High-Speed ADCs", Master's thesis, 2023.

R. Dwarakanath, "Evaluation of Portable Test and Stimulus Standard for Intellectual Property Testbench", Master's thesis, 2023.

H. Korn, "Synapse Circuit Design for Spiking Neural Networks using Reconfigurable FETs", Master's thesis, 2022.

F. Lalla, "Capacitance Modeling for a Planar Reconfigurable Field Effect Transistor", Bachelor's thesis, 2023.

Md. A. Rahman, "Rekonfigurierbare Zellen in Synthese und Technologie Mapping", Bachelor's seminar, 2023.

B. Wiedekind, "Standard Cell Characterization From Technology Simulation", Master's thesis, 2023.

J. Wilm, "Machine Learning Based Compact Models for Reconfigurable FETs", Master's thesis, 2023.

V. Weinelt, "In-Memory Computing Based on Reconfigurable FETs", Master's thesis, 2022.

B. Wiedekind, "Ein Planarer RFET Differenzverstärker", Master's seminar, 2022.

W. Xing, "Efficient Implementation of Table Models for Semiconductor Devices", Bachelor's thesis, 2022.

D. Lee, "Compact Modelling of a Multiple Independent Gate FET", Master's thesis, 2021.

A. Löser, "Design of Digital Circuits Using a Planar Reconfigurable Three-Independent-Gate-FET", Bachelor's thesis, 2021.

D. Riehl, "Equation Based Modelling of an Ambipolar Transistor", Master's thesis, 2020.

K. Karageorgos, "Literature Review on Phase-Shift-Keying circuits", Bachelor's Seminar, 2019.

S. Pichl, "Reconfigurable Standard Cells Using Ambipolar Transistors", Bachelor's seminar, 2019.

S. Mourelatos, "Literature Review On CMOS Characterization", Bachelor's seminar, 2019.

D. Voutyrakou, "Literature Review on Analog Integrated Circuits Incorporating Ambipolar Transistors", Bachelor's seminar, 2019.

Colophon

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

`https://bitbucket.org/amiede/classicthesis/`