

BEHAVIORAL MODELING OF DRIVERS AND OSCILLATORS USING MACHINE LEARNING

A Dissertation
Presented to
The Academic Faculty

by

Huan Yu

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
December 2019

COPYRIGHT © 2019 BY HUAN YU

BEHAVIORAL MODELING OF DRIVERS AND OSCILLATORS USING MACHINE LEARNING

Approved by:

Dr. Madhavan Swaminathan, Advisor
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Saibal Mukhopadhyay
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Sung-Kyu Lim
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Dr. Suresh K. Sitaraman
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Arijit Raychowdhury
School of Electrical and Computer
Engineering
Georgia Institute of Technology

Date Approved: October 3, 2019

To my family and friends

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support of the people who have helped and inspired me during my Ph.D. life at Georgia Tech. I would like to thank all of them.

First, I would like to express my deepest gratitude to my advisor, Dr. Madhavan Swaminathan. All these accomplishments could not be possible without his valuable advice, insightful guidance and support. He is an outstanding scientist, mentor and leader. What I have learnt from him will be my lifetime treasure. I would also like to extend my gratitude to the committee members, Dr. Sung-Kyu Lim, Dr. Arijit Raychowdhury, Dr. Saibal Mukhopadhyay, and Dr. Suresh K. Sitaraman, for their time and effort in serving on my committee.

I would like to give my special thanks to current and former members of the research group, Ming Yi, Biancun Xie, David Zhang, Kyuhwan Han, Sung Joo Park, Colin Pardue, Hakki Torun, Majid Ahadi, Nahid Aslani Amoli, Sridhar Sivapurapu, Claudio Alvarez, Osama Waqar Bhatti, Venkatesh Avula, Kai Qi Huang, Serhat Erdogan, Seunghyup Han, Mutee ur Rehman, Xiaofan Jia, Chirag Mehta, Xiaotong Jia, Anoop Chidambar Kulkarni, Sebastian Muller, Anto K. Davis, Mohamed L.F Bellaredj, Mourad Larbi and Kallol Roy, and visiting scholars, Xiaojia Huang, Felipe Treviso, Hemanth Chalamalasetty, Sreenaesh Ramesh, Yongsheng Li, Wen-Sheng Zhao, Riccardo Trincherro and Carmine Gianfanga for their help and support. My thanks also extend to all the researchers and students I worked with, and all my friends.

I would like to especially thank my family. Without the endless love, guidance and support from my parents, I would not be here. They are always the source of my strength and motivation.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF SYMBOLS AND ABBREVIATIONS	xiv
SUMMARY	xvi
CHAPTER 1. Introduction	1
1.1 Background and Motivation	1
1.1.1 Behavioral Modeling and Neural Networks	1
1.1.2 Modeling of Oscillators	3
1.1.3 Modeling of I/O Drivers	5
1.2 Summary of Contributions	7
1.3 Organization of the Dissertation	8
CHAPTER 2. Steady-State Modeling of Oscillators	10
2.1 Introduction	10
2.2 Modeling of Oscillator with Constant Frequency	11
2.2.1 Mathematical Model	11
2.2.2 Augmented Neural Network for Oscillators	11
2.2.3 Neural Network Training	14
2.2.4 Modeling Examples	16
2.3 Modeling of Oscillator Including I/O behavior	21
2.3.1 Buffer Modeling	22
2.3.2 AugNN-based Model	24
2.3.3 Modeling Example	26
2.4 Modeling of VCOs	29
2.4.1 Mathematical Model	29
2.4.2 Augmented Neural Network for VCOs	29
2.4.3 AugNN Training	32
2.4.4 VCO Modeling Example	35
2.5 Modeling of VCO Including I/O behavior	38
2.5.1 AugNN-based Model	38
2.5.2 Modeling Process	40
2.5.3 Model Validation	42
2.6 Summary	48
CHAPTER 3. Behavioral Modeling of Tunable I/O Driver with Pre-emphasis	49
3.1 Introduction	49
3.2 Model Structure	51
3.3 Pre-emphasis Issue in Modeling	53

3.4	Modeling Method	56
3.4.1	Model Formulation with State-Aware Weighting Functions	56
3.4.2	Model Representations	58
3.4.3	Modeling Flow	62
3.4.4	Tunable Driver with Control Parameters	67
3.5	Modeling Examples	69
3.5.1	Pre-emphasis Driver with Power Supply Noise	69
3.5.2	Validation of State-Aware Weighting Functions with Pre-emphasis	73
3.5.3	Tunable Driver Modeling Test Cases	76
3.6	Summary	82
CHAPTER 4.	Modeling of I/O Drivers under Overclocking Conditions	83
4.1	Introduction	83
4.2	Model Overview	84
4.3	Overclocking Issue	86
4.4	Transition-Variational Model	91
4.4.1	Model Formulation with TVWFs	91
4.4.2	Modeling Process	94
4.5	Modeling Examples	96
4.5.1	Test Case under Normal Condition	97
4.5.2	Test Case under Overclocking Condition	100
4.6	Summary	103
CHAPTER 5.	Conclusion and Future Work	105
5.1	Contributions	106
5.2	Future Work	109
5.3	Publications	112
APPENDIX A.	OSCILLATOR MODEL IMPLEMENTATION	116
APPENDIX B.	OVERCLOCKED DRIVER MODEL IMPLEMENTATION	119
REFERENCES		124

LIST OF TABLES

Table 1	– Functions and value options of the driver control parameters.	67
Table 2	– Comparison of the eye diagram apertures of the behavioral models (reference: transistor-level model).	76
Table 3	– Comparison of the eye diagram apertures of the proposed behavioral model for tunable pre-emphasis driver in different test cases (reference: transistor-level model).	82
Table 4	– Eye diagram performance and efficiency evaluation for normal condition test case (reference: transistor-level model).	100
Table 5	– Eye diagram performance and efficiency evaluation for overclocking condition test case (reference: transistor-level model).	103
Table 6	– Model implementation in Verilog-A.	119

LIST OF FIGURES

Figure 1	– Behavioral modeling for time-domain analysis.	2
Figure 2	– Trapezoidal output waveform of an oscillator.	4
Figure 3	– Proposed augmented neural network for oscillators.	13
Figure 4	– Mod function: $y = x \bmod 1$.	15
Figure 5	– Schematic of a typical inverter ring oscillator.	17
Figure 6	– Test case 1 output waveforms for transistor-level oscillator model (straight line) and behavioral model (dotted line).	18
Figure 7	– Plot of the frequency parameter K during the training process.	18
Figure 8	– Test case 2 output waveforms for transistor-level oscillator model (straight line) and behavioral model (dotted line).	19
Figure 9	– Schematic of a quadrature oscillator.	20
Figure 10	– AugNN model for the quadrature oscillator.	20
Figure 11	– Output waveforms for the transistor-level model and the behavioral model of the quadrature oscillator.	21
Figure 12	– Oscillator circuit including output buffer.	22
Figure 13	– Buffer output current for input HIGH from transistor-level model (solid line) and from RNN model (dotted line).	23
Figure 14	– Proposed AugNN-based model for oscillators including output buffers.	25
Figure 15	– Ring oscillator circuit including an output buffer.	26
Figure 16	– Weighting functions extracted (solid line) and learnt by the AugNN (dotted line).	27
Figure 17	– Test setup for model validation.	28
Figure 18	– Output voltage waveform from transistor-level model (solid line) and from behavioral model (dotted line).	28
Figure 19	– Proposed AugNN model for VCOs.	30

Figure 20	– Recurrent node representation of the integral node in the periodic unit.	33
Figure 21	– Simplified schematic of the target VCO.	35
Figure 22	– AugNN with single output unit for VCO modeling.	36
Figure 23	– Training waveforms of control voltage (thick solid blue line), and the output of transistor-level (thin red line) and behavioral model (dotted line).	37
Figure 24	– Test waveforms of control voltage (thick solid blue line), and the output of transistor-level (thin red line) and behavioral model (dotted line).	37
Figure 25	– VCO circuit including output buffer.	38
Figure 26	– Proposed AugNN-based model for VCOs including output buffers.	39
Figure 27	– Test-benches used for the collection of training data required for model generation. (a) Transient analysis for the extraction of nonlinear dynamic sub-models. (b) Transient analysis for the extraction of weighting functions using step control voltage signal.	42
Figure 28	– Simplified schematic of the VCO circuit including output buffer.	43
Figure 29	– (a) Training waveform of control voltage. (b) Weighting function w_H extracted and learnt using AugNN. (c) Weighting function w_L extracted and learnt using AugNN.	44
Figure 30	– Tuning curve of the VCO obtained from the transistor-level model and the behavioral model.	45
Figure 31	– Model validation setup.	46
Figure 32	– (a) Control voltage waveform used in the test case. (b) Output waveforms of the transistor-level simulation and the behavioral model simulation.	47
Figure 33	– Generic driver electrical structure.	51
Figure 34	– Example output voltage waveform of a pre-emphasis driver.	54
Figure 35	– Example weighting function $w_{o,1}(t)$ for a pre-emphasis driver with possible down transitions (dashed lines) of different timings.	54
Figure 36	– Concatenation of steady-state switching timing signals for pre-	55

emphasis when the switched input logic states are shorter than the pre-emphasis duration.

Figure 37	– FSM graph for the proposed modeling method with state-aware weighting functions.	57
Figure 38	– RNN structure.	59
Figure 39	– (a) Piecewise-linear voltage waveform connected at the driver output. (b) Output current from a transistor-level model (solid line) and from RNN model (dashed line) for input high.	61
Figure 40	– FFNN structure.	62
Figure 41	– Test-benches used for the collection of driver responses required for the model generation. (a) Transient analysis for the extraction of nonlinear dynamic sub-models. (b) Transient analysis for the extraction of state-aware weighting functions using input transitions with different intervals.	64
Figure 42	– Example input waveforms with different intervals for the extraction of weighting functions for high-to-low transitions (dashed red down transition curves) with pre-emphasis effect.	66
Figure 43	– Tunable driver with control parameters.	67
Figure 44	– (a) Output waveforms of down transitions for different $Ron.pd$ and EQ settings, with $Ron.pu = 34$. (b) Output waveforms of up transitions for different $Ron.pu$, EQ and VOH settings, with $Ron.pd = 34$.	68
Figure 45	– Extracted output port weighting function $w_{o,0}$ with pre-emphasis for HIGH-to-LOW transitions.	70
Figure 46	– Simulation setup for model validation.	71
Figure 47	– Voltage waveform at the power supply port from the transistor-level model and from the behavioral model.	72
Figure 48	– Eye diagram at the far-end of the transmission line. Power supply variation is considered for the weighting functions of the behavioral model.	72
Figure 49	– Eye diagram at the far-end of the transmission line. Power supply variation is not considered for the weighting functions of the behavioral model.	73

Figure 50	– Simulation waveforms with and without state-aware weighting functions.	74
Figure 51	– Eye diagram at the near-end of the transmission line.	75
Figure 52	– Simulation setup under channel reflection and crosstalk scenarios.	77
Figure 53	– Voltage waveforms at (a) power supply port and (b) channel port 2B, in test case 1.	78
Figure 54	– Voltage waveforms at (a) supply port and (b) channel port 2B, in test case 2.	79
Figure 55	– Voltage waveforms at (a) supply port and (b) channel port 2B, in test case 3.	80
Figure 56	– Behavioral model structure for I/O drivers.	86
Figure 57	– Example driver output waveform under overclocking conditions.	87
Figure 58	– Generation of weighting coefficients based on the concatenation scheme.	88
Figure 59	– Transitions of the weighting function under overclocking conditions.	88
Figure 60	– (a) Input patterns with different transition separations. (b) Extracted weighting function w_1 .	90
Figure 61	– FSM graph for the proposed modeling method using TVWFs.	94
Figure 62	– Transient analysis setup used for the collection of data required for the extraction of weighting functions.	96
Figure 63	– Simulation setup for model validation using transmission line.	97
Figure 64	– Simulated voltage waveforms from the transistor-level model and the proposed model at the near-end of the transmission line.	98
Figure 65	– Eye diagrams from the transistor-level model and the proposed model at the near-end of the transmission line.	99
Figure 66	– Simulated voltage waveforms at the far-end of the transmission line from the transistor-level model, the proposed model (TVWFs) and the non-TVWFs behavioral model.	101
Figure 67	– Eye diagrams at the far-end of the transmission line from the transistor-level model and the proposed model (TVWFs).	102

Figure 68	– Eye diagrams at the far-end of the transmission line from the transistor-level model and the non-TVWFs behavioral model.	102
Figure 69	– Example transient waveform of an oscillator.	110
Figure 70	– Multi-level encoding and eye diagram in PAM-4 signaling.	111

LIST OF SYMBOLS AND ABBREVIATIONS

IC	Integrated Circuit
EDA	Electronic Design Automation
IP	Intellectual Property
ANN	Artificial Neural Network
PLL	Phase-Locked Loop
VCO	Voltage-Controlled Oscillator
I/O	Input/Output
ISI	Intersymbol Interference
SPI	Signal and Power Integrity
SI	Signal Integrity
IBIS	I/O Buffer Information Specification
EIA	Electronics Industry Alliance
I-V	Current-Voltage
V-t	Voltage-Time
CMOS	Complementary Metal Oxide Semiconductor
RNN	Recurrent Neural Network
AugNN	Augmented Neural Network
PI	Power Integrity
SSN	Simultaneous Switching Noise
PAM-4	Four-Level Pulse Amplitude Modulation
BP	Back Propagation
MSE	Mean Squared Error

RMSE	Root-Mean-Squared Error
FOM	Figure of Merit
FFNN	Feed-Forward Neural Network
BPTT	Back Propagation Through Time
PDN	Power Delivery Network
FSM	Finite State Machine
PRBS	Pseudorandom Bit Sequence
TVWF	Transition-Variational Weighting Function
NARX	Nonlinear Autoregressive Network with Exogenous Inputs
NRZ	Non-Return to Zero

SUMMARY

The objective of this dissertation is to develop time-domain behavioral models for I/O drivers and oscillators for fast simulation and IP protection. For oscillators, augmented neural networks (AugNNs) are proposed to capture the oscillatory behavior of fixed-frequency oscillators and VCOs. When output buffer is included as a part of the oscillator circuit, AugNN-based models are developed taking into account the I/O behavior of the oscillator. For tunable drivers with pre-emphasis, state-aware weighting functions are proposed, and the dynamic memory characteristics of the driver's output stage are captured using recurrent neural networks (RNNs). The behavior of the tunable control parameters is captured. Furthermore, a transition-variational model is discussed for the modeling of I/O drivers under overclocking conditions. The proposed models are compatible with Verilog-A.

CHAPTER 1. INTRODUCTION

The design process of integrated circuits (ICs) relies on the use of electronic design automation (EDA) tools which provide the capability of performing circuit simulations for functional verification. However, as complexity of IC designs increases, performing such analysis is becoming more and more challenging in terms of the CPU time required for transistor-level simulations of the circuits. Behavioral modeling plays an important role in the design process of ICs by reducing the CPU time for simulations. This black box modeling approach, which is always advantageous for intellectual property (IP) protection, is independent of the knowledge of the internal logic of the circuit components, and thus doesn't need to have the same complexity as the transistor-level circuit models. It's worth mentioning that the behavioral model developed will be of little use unless it is compatible with existing commercial circuit simulation environment (e.g., Spectre, HSPICE, etc.).

1.1 Background and Motivation

1.1.1 Behavioral Modeling and Neural Networks

The objective of behavioral modeling is to find suitable port relations similar to the original complex circuit such that the behavioral model is able to minimize the error compared to the response of the transistor-level model, as shown in Figure 1. At the same time, the behavioral model should be compatible with the existing circuit simulators, and should require less simulation time than the original circuit.

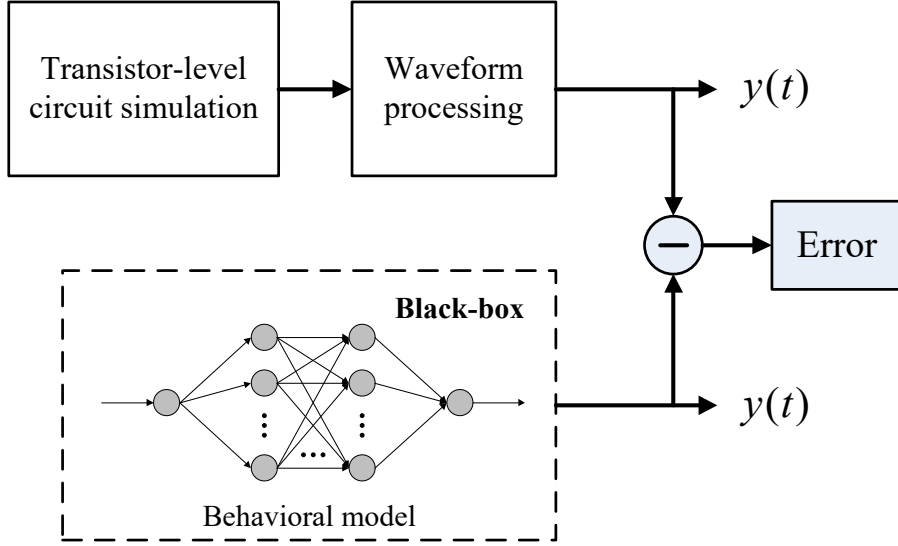


Figure 1 – Behavioral modeling for time-domain analysis.

Recently, machine learning has been widely explored in various areas of electronics such as modeling, optimization and inverse design [1]-[15]. Specifically, in behavioral modeling, artificial neural networks (ANNs) are shown to be capable of capturing the nonlinearities accurately [16]-[18]. Many techniques have been proposed in the past to develop behavioral models for nonlinear circuits using ANNs [17]-[19]. The advantages that ANNs provide include but are not limited to the following: Firstly, ANNs are universal approximators [16], [18] with bounded and monotone-increasing continuous nonlinear activation functions in the hidden neurons (in particular, the nonlinear function cannot be a polynomial, since the superposition of polynomials is still a polynomial), which ensures the effectiveness and flexibility for fitting the model components. Secondly, there are several well developed techniques (testing during training, regularization, etc.) devised for the training of ANNs to avoid overfitting the target functions, which provides the generalization capability of ANNs for multivariate modeling. Furthermore, compared to table-formatted models, ANNs, which can be

readily implemented in Verilog-A [20] using the common built-in mathematical functions and operators, are flexible to use and require less coefficients (just need network bias and weight values). Based on the above observations, ANNs are selected as the model architecture for the proposed work.

1.1.2 Modeling of Oscillators

In the design process of ICs, highly sophisticated transistor-level models are used. The entire mixed-signal circuit simulation sometimes can be dominated by certain circuit blocks like oscillators which is a key component in analog or digital phase-locked loops (PLLs). Behavioral modeling of oscillators plays an important role by reducing the CPU time for simulations.

In [21], a behavioral model is developed for a fixed-frequency oscillator using ANNs based on state space representation. The output of the oscillator and its time derivatives represent the state of the system. The time-domain behavior is embodied by an implicit formulation of the system differential equations which are solved using a circuit simulator. The formulation of the system differential equations represent the relationship between the output signal of the oscillator, its lower-order time derivatives and its higher-order time derivatives, which are learnt using ANNs. This state space approach, however, has its limitations. The shape of the oscillatory waveform is critical for the success of the state space model. Reference [22] shows an example of a trapezoidal oscillatory waveform, where the oscillator output stays invariant and all the derivatives go to zero in the plateau region as shown in Figure 2, and as a result, it's impossible for the model to trigger the transition of the system state.

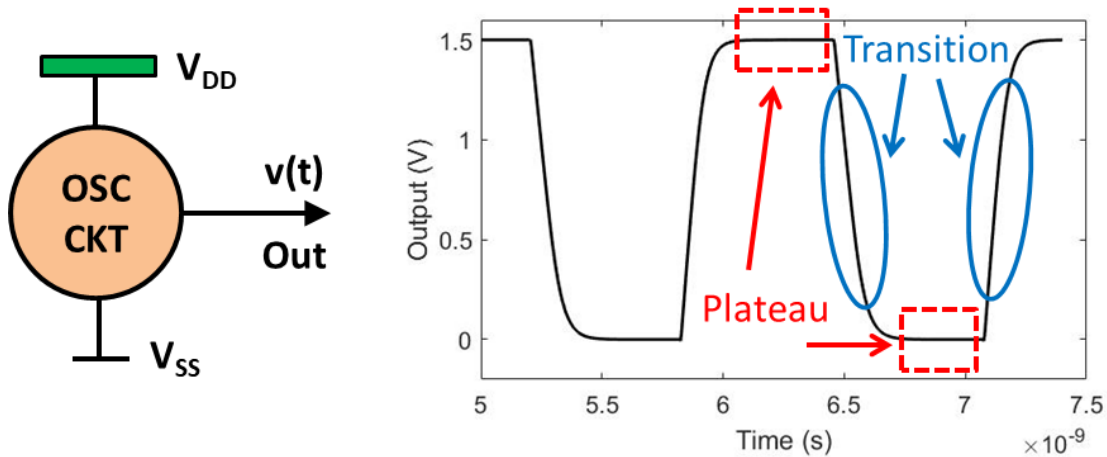


Figure 2 – Trapezoidal output waveform of an oscillator.

Using a similar approach, in [23], a behavioral model is developed for a voltage-controlled oscillator (VCO). The output of the VCO and its time derivatives are mapped to the higher-order time derivatives using ANNs. The difference for VCO modeling is that the system response is modified by a control voltage as well. Therefore, in the formulation of the system differential equations, the control signal is included as an additional input, and the ANNs need to be trained accordingly in order to capture the influence of the control signal on the mapping relation. However, as discussed above, this VCO modeling approach suffers from the same issue that it is limited by the shape of the target waveform, such as the trapezoidal oscillatory waveforms with flat regions. In addition, to properly train the ANNs for the state space equations, the training data needs to be generated carefully to cover all the possible area of the state space, since incomplete space coverage will result in the numerical solution of the differential equations deviating from the correct solution or failing to converge, which makes the approach difficult to use.

1.1.3 Modeling of I/O Drivers

The transfer of large amounts of data between different components in electronic systems relies on high-speed links, where mixed-signal input/output (I/O) drivers play a critical role in generating high quality communication signals in the channel. Pre-emphasis drivers are effective in driving signals through lossy transmission lines, and are useful in reducing intersymbol interference (ISI). Signal and power integrity (SPI) and timing analysis of digital systems in the design and optimization phase is becoming more and more time consuming due to the increasing complexity of driver designs. Therefore, being able to model pre-emphasis drivers is important in signal integrity (SI) analysis of digital systems.

There exist several techniques for modeling drivers [24]-[33]. The most popular approach is the I/O buffer information specification (IBIS) model [34] from the electronics industry alliance (EIA). It is a modeling technique that provides a simple table-based buffer model for semiconductor devices. IBIS models can be used to characterize current-voltage (I-V) output curves, rising/falling transition waveforms, and package parasitic information of the device. IBIS models are based on DC I-V curves along with a set of voltage-time (V-t) curves of the driver output voltage and packaging parasitic information of the I/O driver. However, the defined equivalent circuit in IBIS model decides a-priori the physical effects taken into account, thus leaving limited ability to capture advanced features of modern I/O drivers. Moreover, this table-based format has limitations in representing characteristics with multivariate dependency (e.g., port voltages, control parameters, etc.). IBIS models can also be challenged in accurately

capturing the dynamic characteristics of drivers at high data rates, since the dynamic characteristics of the driver become predominant when driver excitations become fast.

As alternative approaches for behavioral modeling, parametric and enhanced models have been proposed [35], [36], providing improved accuracy and variation of features for modeling I/O drivers of complementary metal oxide semiconductor (CMOS) technology. These parametric models suggest a two-piece structure, where two sub-models are used to capture the I-V behavior of the pull-up and pull-down devices respectively, and are multiplied by weighting coefficients which scale the transient contributions of the upper and lower devices' nonlinear current. The weighting coefficients, either in table format or represented using weighting functions, are generated corresponding to the bit sequence at the input of the driver, and containing the timing information of the transition events. However, these models are only applied to drivers without pre-emphasis.

Based on the above approach, [37]-[39] extend the modeling method to model pre-emphasis drivers. In these models, the same two-piece model structure are used, and the pre-emphasis behavior is captured using weighting signals that have the proper shape incorporating the multiple level states, i.e., normal high, normal low, strong high and strong low. These implementations assume state transitions are spaced sufficiently in time so that every new state transition only appears after the I/O driver has reached its steady state. However, these models do not address the modeling of pre-emphasis drivers when the bit duration is shorter than the pre-emphasis duration, where the above assumption is not satisfied.

In [40], a physics-based approach is adopted to construct a table-based empirical model for the simulation of digital drivers subject to overclocking conditions, where the identified nonlinear dynamic model operators of the input port replace the concatenated fixed step-input describing functions of the table-based IBIS model and of other parametric approaches. However, this modeling approach does not capture the pre-emphasis characteristics of the driver, and the signals on certain internal nodes of the transistor-level devices are revealed and manipulated, thereby leaving this gray-box model unfeasible and exposing IP. Moreover, it cannot be applied to tunable drivers with control parameters, since the identified nonlinear dynamic model operators are not suitable for parameterization.

1.2 Summary of Contributions

The objective of this dissertation is to develop time-domain behavioral models for drivers and oscillators to enable fast simulation and IP protection. For oscillators, augmented neural networks (AugNNs) are proposed to capture the oscillatory behavior of fixed-frequency oscillators and VCOs. For drivers, parametric modeling approaches are used, and the dynamic memory characteristics of the driver's output stage are captured using recurrent neural networks (RNNs). The proposed models are compatible with Verilog-A. In summary, the contributions of the dissertation are listed as follows:

1. Steady-state oscillator modeling. This work proposes time-domain steady-state behavioral models of fixed-frequency oscillators and voltage-controlled oscillators, which are not limited to the shapes of the target waveforms.

2. Modeling of oscillators including I/O behavior. With output buffer being part of an oscillator circuit, the behavior of the buffer needs to be incorporated into the modeling of the entire oscillator design. This work proposes the modeling of oscillators with output buffers.
3. Modeling of tunable drivers with pre-emphasis. SI analysis is important in the design of ICs. Similarly, power integrity (PI) analysis is critical considering that simultaneous switching noise (SSN) can have significant influence on the performance of circuits. Therefore, an accurate model for drivers with pre-emphasis that can be used for both SI and PI analysis is needed. On the other hand, in some of the modern driver designs, advanced features emerge such as tunable characteristics with control parameters. The driver model developed would be of little use unless the tunable characteristics are captured. The modeling of tunable drivers with pre-emphasis for SI and PI co-simulation is addressed in this work.
4. Driver modeling for overclocking analysis. When data rate becomes higher and higher, overclocking will occur when I/O drivers do not have enough time to reach steady state before the next transition. The existing approaches cannot accurately capture the overclocking behavior of the I/O drivers. This work proposes an accurate driver modeling method for overclocking analysis.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows: Chapter 2 describes the proposed AugNNs for the modeling of oscillators with constant frequency, multi-phase oscillators and VCOs. Also in this chapter, AugNN-based models are presented for the

modeling of oscillators with output buffers. In Chapter 3, behavioral modeling of tunable I/O driver circuits with pre-emphasis for SPI analysis is investigated, where the influence of the control parameters is taken into account. A modeling methodology for I/O drivers under overclocking conditions is demonstrated in Chapter 4. Finally, Chapter 5 presents summary of this work and discussion of some related future work.

CHAPTER 2. STEADY-STATE MODELING OF OSCILLATORS

2.1 Introduction

The design process of ICs relies on simulations using EDA tools. Performing these simulations for timing analysis, however, is becoming challenging in terms of the CPU time required. This is amplified for transistor-level simulations in the design and optimization phase due to increasing complexity of IC designs. The entire circuit simulation can be sometimes dominated by certain circuit blocks like oscillators which is a key component in analog or digital PLLs. Behavioral modeling techniques play an important role in reducing this complexity as they require less computing resources and still provide accurate numerical results without disclosing any internal circuitry information.

In [21] and [23], behavioral models are developed for oscillators using artificial neural networks based on state space representation. The output of the oscillator and its time derivatives represent the state of the system. The time-domain behavior is embodied by an implicit formulation of the system differential equations which are solved using a circuit simulator. This state space approach, however, has its limitations. The shape of the oscillatory waveform is critical for the success of the state space model. Figure 2 shows an example of a trapezoidal oscillatory waveform. In the plateau region, the oscillator output stays invariant and all the derivatives go to zero. As a result, it's impossible to trigger the transition of the system using the state space approach. In this chapter, a novel technique is presented for the time-domain behavioral modeling of oscillators using AugNNs. In the proposed method, a feed-forward neural network (FFNN) with a periodic unit is developed based on the mathematical formulation of oscillator behavior, where the periodic unit can capture the periodicity of the oscillatory output waveform, and the

neural network is able to learn the waveform shape properly. Time step values are used as an input to the neural network to learn the phase information of the oscillation. The resulting model is a black box that hides the details of a transistor circuit and thus protects IP.

2.2 Modeling of Oscillator with Constant Frequency

2.2.1 Mathematical Model

The output waveform of an oscillator with constant frequency in the time-domain can be expressed as [41]:

$$V_{out} = F\left(K_{OSC} \int dt\right), \quad (1)$$

where K_{OSC} is related to the oscillation frequency, and $F()$ is a periodic function that captures the shape of the waveform. In Equation 1, the argument of $F()$, $K_{OSC} \int dt$, is the total phase. The phase accumulates at a rate of K_{OSC} , and is mapped to the output using $F()$, which is periodic.

To derive a behavioral model, we reformulate Equation 1 in the discrete-time domain as:

$$V_{out}(n) = F\left(K_{OSC} \sum_{i=1}^n dt(i)\right), \quad (2)$$

where n is the time index in the discrete-time domain.

2.2.2 Augmented Neural Network for Oscillators

In behavioral modeling, ANNs are shown to be capable of capturing the nonlinearities accurately [16]-[18]. ANNs provide mappings which can approximate input-output nonlinear functions. Since the oscillator is a one-port system, it has no input signal. However, it can be seen from Equation 1 that the output of the oscillator depends on the total phase which is a function of time step dt . Therefore, time is an important input for modeling oscillators. Since the phase and time are related and the phase can be arbitrary, most neural networks have difficulty in capturing the relationship between phase information and output waveforms for oscillators. To capture the periodic dependence of the nonlinear behavior of the oscillator for an unbounded total phase, an AugNN is proposed as shown in Figure 3.

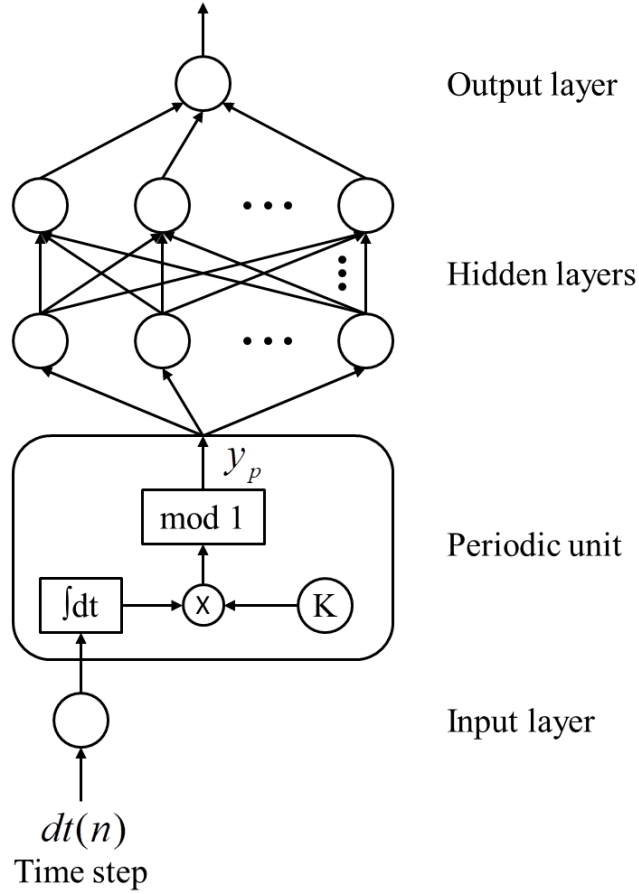


Figure 3 – Proposed augmented neural network for oscillators.

In the proposed neural network, a periodic unit is introduced and cascaded between the input layer and the first hidden layer of an FFNN. The input for the network consists of a chronologically-ordered sequence of time steps \mathbf{dt} which is fed into the periodic unit. Inside the periodic unit, the integral of \mathbf{dt} is multiplied by the frequency parameter K , after which the modulus operation is performed. The responses of the periodic unit can be computed as:

$$y_p(n) = \left(K \sum_{i=1}^n dt(i) \right) \text{mod } 1, \quad (3)$$

By doing so, the unbounded total phase is mapped to the folded and normalized phase within a range of $[0, 1)$. The output of the periodic unit, \mathbf{y}_p , is fed into the first hidden layer. The values of the nodes in the hidden and output layers are calculated using Figure 3 as:

$$y_h^j(n) = \tanh \left[\sum_{l=1}^{N_{prec}} y_{prec}^l(n) w_h^{l,j} + b_h^j \right], \quad (4)$$

where $\tanh(x) = (e^x - e^{-x}) / (e^x + e^{-x})$, y_h^j is the j th neuron in the current hidden layer, y_{prec}^l is the l th neuron of the preceding layer containing N_{prec} neurons, $w_h^{l,j}$ is the weight between them and b_h^j is the bias value of the current neuron. Furthermore,

$$y_{out}(n) = \sum_{l=1}^{N_{prec}} y_{prec}^l(n) w_{out}^l + b_{out}, \quad (5)$$

where y_{out} is the neuron in the output layer, w_{out}^l is the weight from the l th neuron of the precedent layer and b_{out} is the bias value of the output neuron, respectively.

2.2.3 Neural Network Training

Let the parameters of the augmented neural network be denoted as Φ , $\Phi = [\mathbf{w}^T \mathbf{b}^T K]^T$, where \mathbf{w} is the vector of weights, \mathbf{b} is the vector of biases for the neurons, K is the frequency parameter of the periodic unit, and T denotes transpose. When a set of training data is fed into the input layer, the output, $y_{out}(n)$, is compared with the desired output, $\hat{y}_{out}(n)$, to calculate the training error. The parameters are estimated to minimize the difference between the network output $y_{out}(n)$ and the transistor-level circuit data,

$\hat{y}_{out}(n)$. Let N_t be the number of time samples in the training dataset. The objective for training the AugNN model is given by:

$$\min_{\Phi} \frac{1}{2} \sum_{n=1}^{N_t} (y_{out}(n) - \hat{y}_{out}(n))^2. \quad (6)$$

In order to employ efficient gradient-based training algorithms, derivatives of the network output with respect to each parameter in Φ are required to form the Jacobian matrix \mathbf{J} . A training scheme based on back propagation (BP) is used to calculate the derivatives for each layer.

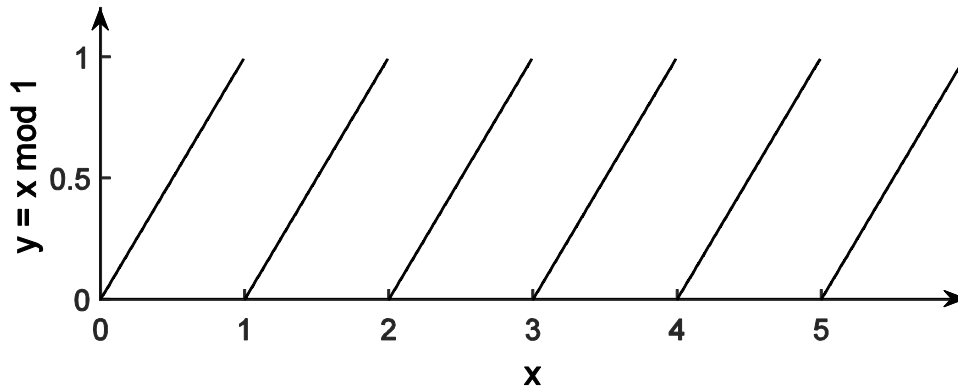


Figure 4 – Mod function: $y = x \bmod 1$.

However, there are points of discontinuity in the output of mod function in the periodic unit. As shown in Figure 4, the function, $y = x \bmod 1$, is discontinuous when x has integer values. But it is important to note that the derivative on the left side of the discontinuity is equal to that on the right side, which is 1 here. The same value can be assigned to the derivatives of the mod function at the discontinuous points to complete the Jacobian matrix for the periodic unit, which can be formulated as:

$$\begin{aligned}
\frac{\partial y_{out}(n)}{\partial K} &= \frac{\partial y_{out}(n)}{\partial y_p(n)} \frac{\partial y_p(n)}{\partial K} \\
&= \frac{\partial y_{out}(n)}{\partial y_p(n)} \sum_{i=1}^n dt(i).
\end{aligned} \tag{7}$$

Based on this gradient scheme, gradient-based training algorithms such as the Levenberg-Marquardt method can be used to train the augmented neural network model. Once the model is trained, it can be implemented in Verilog-A.

2.2.4 Modeling Examples

To quantify the accuracy of the behavioral model, a figure of merit (FOM) is defined as:

$$FOM = 100 \cdot \left[1 - \frac{\sum_{i=1}^N |V_i(ref) - V_i(DUT)|}{\Delta V \cdot N} \right], \tag{8}$$

where $V_i(ref)$ and $V_i(DUT)$ are the simulation output voltage data of the transistor-level model and the behavioral model, ΔV is the reference output voltage range, and N is the number of data points [42].

Test case 1: In order to demonstrate the accuracy of the proposed AugNN model, test cases are generated using transistor-level inverter ring oscillator models provided in [43]. The generic schematic is shown in Figure 5. The training data is generated using Spectre [43].

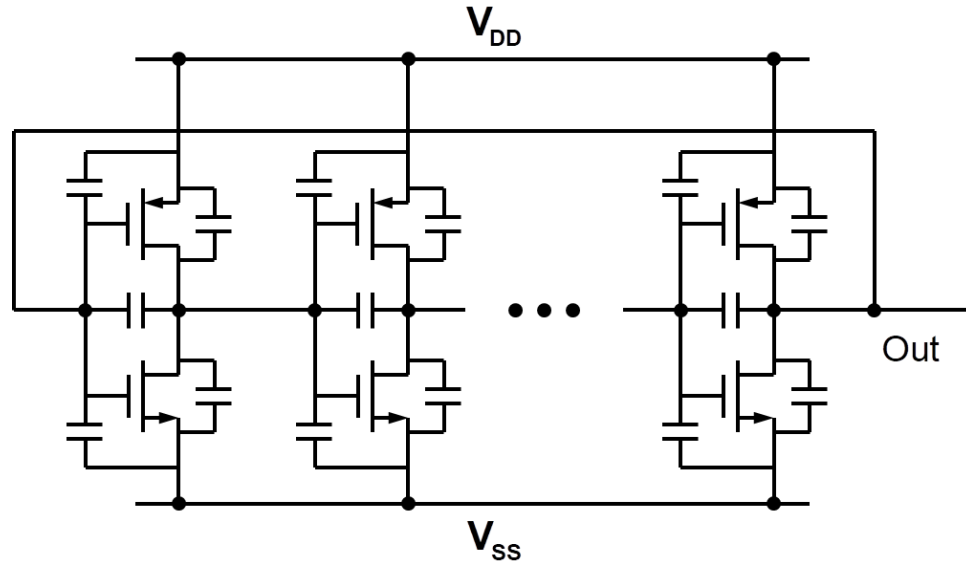


Figure 5 – Schematic of a typical inverter ring oscillator.

In the first test case, a 0.1689 GHz inverter ring oscillator was designed. The transistor-level simulation data is used to train the AugNN with random initializations. The one time training process costs 2~3 min. In Figure 6, the output waveform achieved using the behavioral model with a hidden layer consisting of 10 neurons is compared to the transistor-level model output. It can be seen that the behavioral model is accurate and matches well with the actual inverter ring oscillator voltage waveform, and a FOM value of 99.85 is achieved. Figure 7 shows the training results of the frequency parameter K , which converges to 0.1689×10^9 after several training iterations. The behavioral model achieves ~10X speed-up in simulation time compared to the transistor-level Spice oscillator circuit.

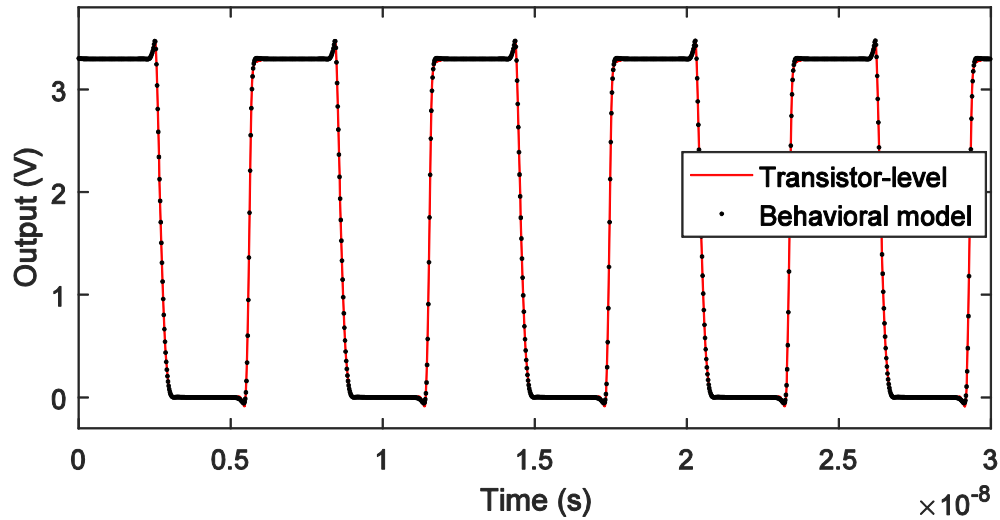


Figure 6 – Test case 1 output waveforms for transistor-level oscillator model (straight line) and behavioral model (dotted line).

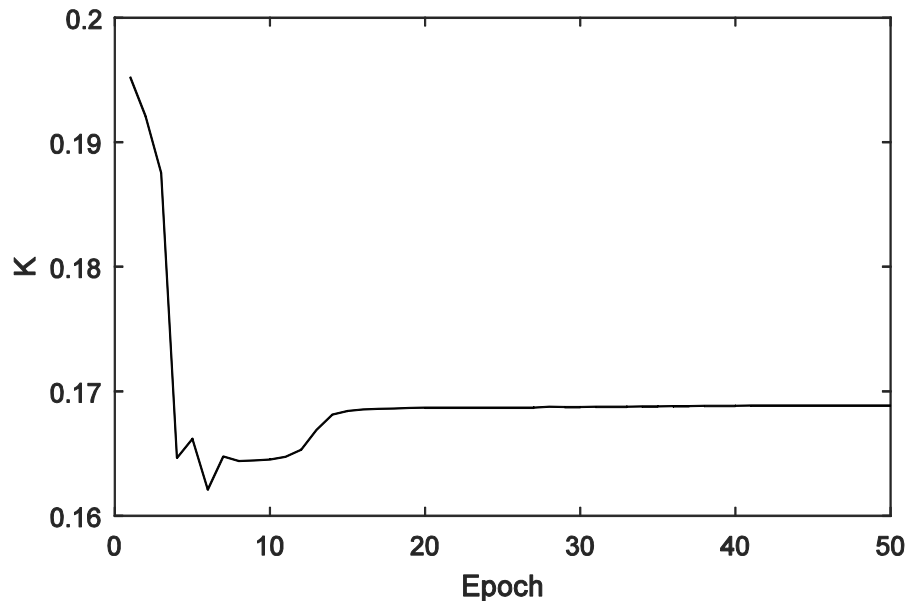


Figure 7 – Plot of the frequency parameter K during the training process.

Test case 2: In this test case, a behavioral model of a 0.5743 GHz inverter ring oscillator is generated. The network model has a hidden layer of 10 neurons. As shown in Figure 8, the output waveform of the behavioral model matches accurately with the transistor-level circuit output, and a FOM value of 99.90 is achieved.

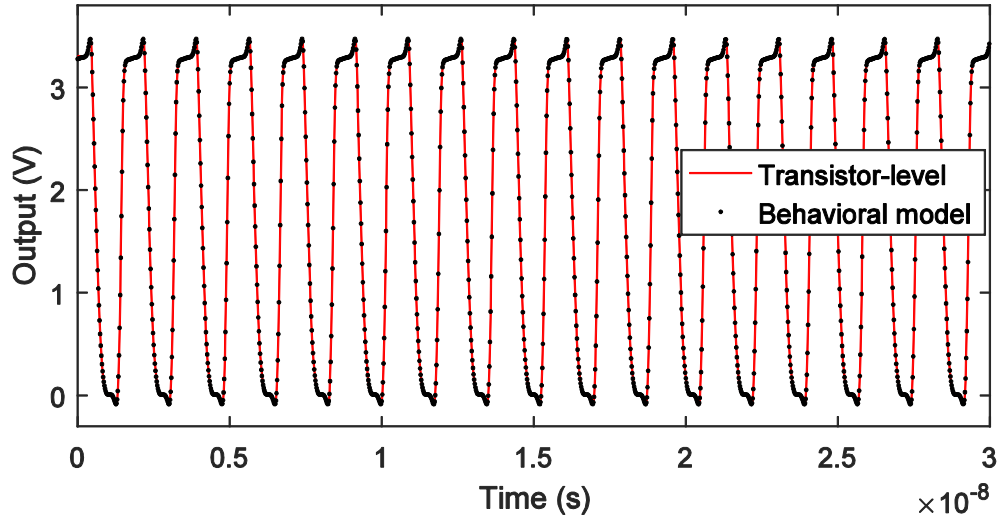


Figure 8 – Test case 2 output waveforms for transistor-level oscillator model (straight line) and behavioral model (dotted line).

Test case 3: In this test case, a modeling example of multi-phase oscillator is demonstrated. A quadrature ring oscillator as shown in Figure 9 is used for model validation [44], [45]. The oscillator is free running with a constant frequency and has four outputs. There is a phase shift of 90° between each successive output. Based on the proposed modeling methodology, for multi-output oscillators, an AugNN model is developed as shown in Figure 10. Since the outputs of the oscillator are highly correlated to each other with respect to the oscillation frequency, an AugNN architecture with the same periodic unit shared by the network outputs is necessary.

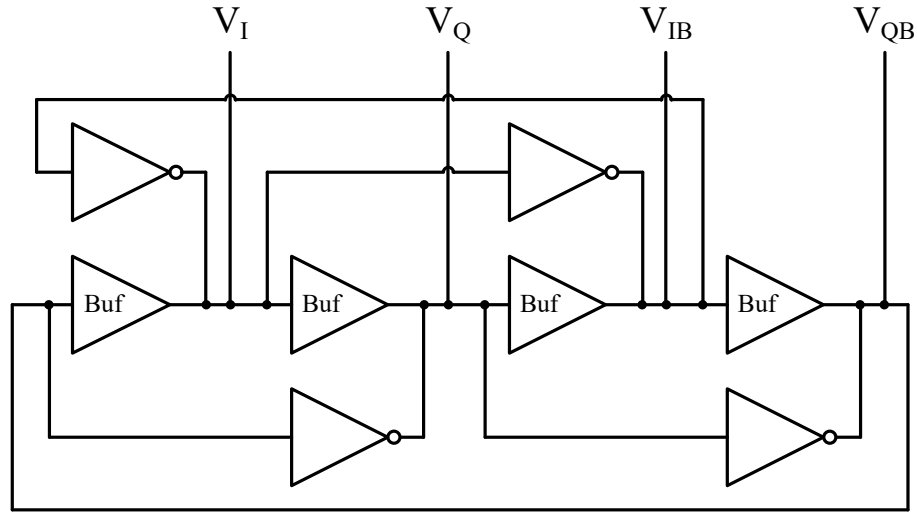


Figure 9 – Schematic of a quadrature oscillator.

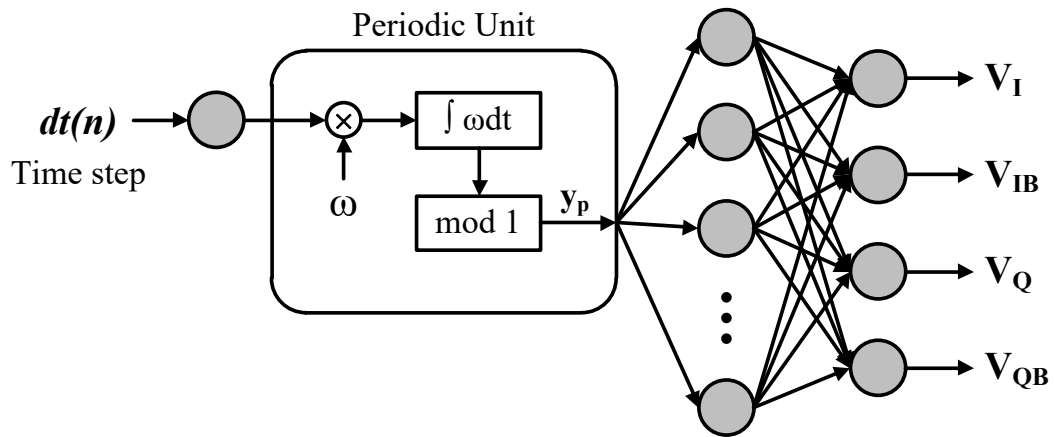


Figure 10 – AugNN model for the quadrature oscillator.

The network model has a hidden layer consisting of 15 neurons, and 4 output neurons corresponding to the quadrature outputs of the oscillator. The training data is generated from transistor-level simulation using Spectre. The network parameters are initialized randomly. A training mean squared error (MSE) of 6.25E-6 is achieved after several iterations, and the frequency parameter converges to 3.2249 GHz. After training, the model is implemented in Verilog-A, and simulated using Spectre. Figure 11 shows the output waveforms of the transistor-level model and the behavioral model, where a

good match can be observed with an MSE of 6.54E-6, and a FOM value of 99.82 is achieved.

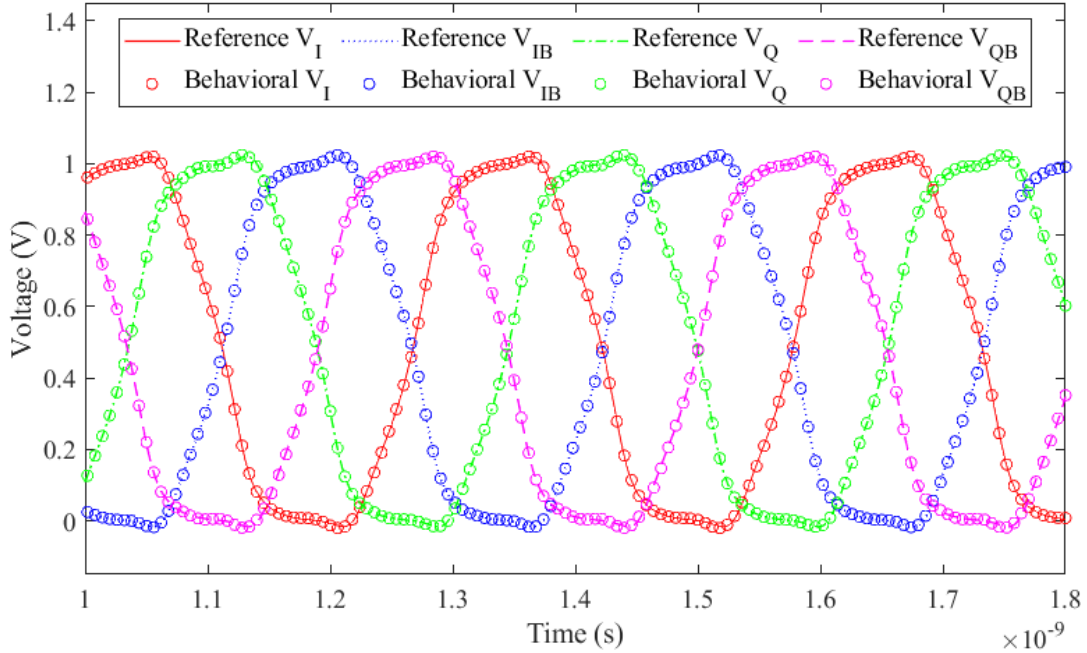


Figure 11 – Output waveforms for the transistor-level model and the behavioral model of the quadrature oscillator.

2.3 Modeling of Oscillator Including I/O behavior

In ICs, buffers are widely used as the interface between the internal circuit core and the load at the output. Since the buffers are unidirectional, the voltage and current at the output port have little influence on the internal activity. In oscillator designs, output buffers can be employed to minimize loading of the oscillator core and generate high quality signals with enhanced drive strength. Figure 12 shows a scenario where a buffer is included in the oscillator design.

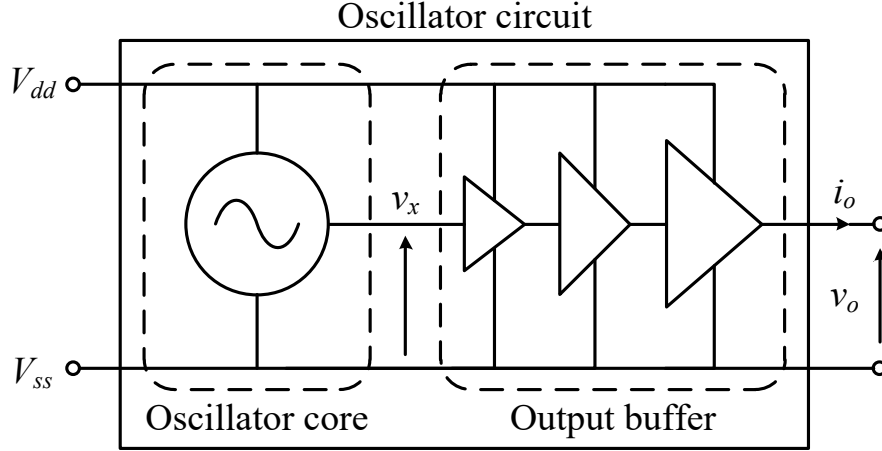


Figure 12 – Oscillator circuit including output buffer.

The model illustrated above, however, does not take into account output buffer behavior and the associated load variations. The model needs to be extended to address the modeling of oscillators with output buffers. In this section, an AugNN-based model is discussed, where the nonlinear dynamic behavior of the output buffer is taken into account using RNNs, and an AugNN with multiple output units is used to capture the oscillatory weighting functions controlling the transitions of the buffer.

2.3.1 Buffer Modeling

For a buffer circuit, the current at the output port can be expressed as [35], [37]:

$$i_o(t) = w_H(t)i_H(t) + w_L(t)i_L(t), \quad (9)$$

where $i_H(t)$ and $i_L(t)$ are nonlinear dynamic submodels accounting for the output current of the last stage of the buffer when the buffer input, v_x , is at the high (H) and low (L) voltage state respectively, and $w_H(t)$ and $w_L(t)$ are the weighting functions that help the

transitions between the two sub-models. In Equation 9, sub-model function, $i_n(t)$, can be defined using parametric relations as:

$$i_n(t) = i_n(v_o(t), \mathbf{D}), \quad n = H, L, \quad (10)$$

where $v_o(t)$ is the output voltage, and \mathbf{D} represents the dependence of sub-model current on the previous values of the signals, i_n and v_o .

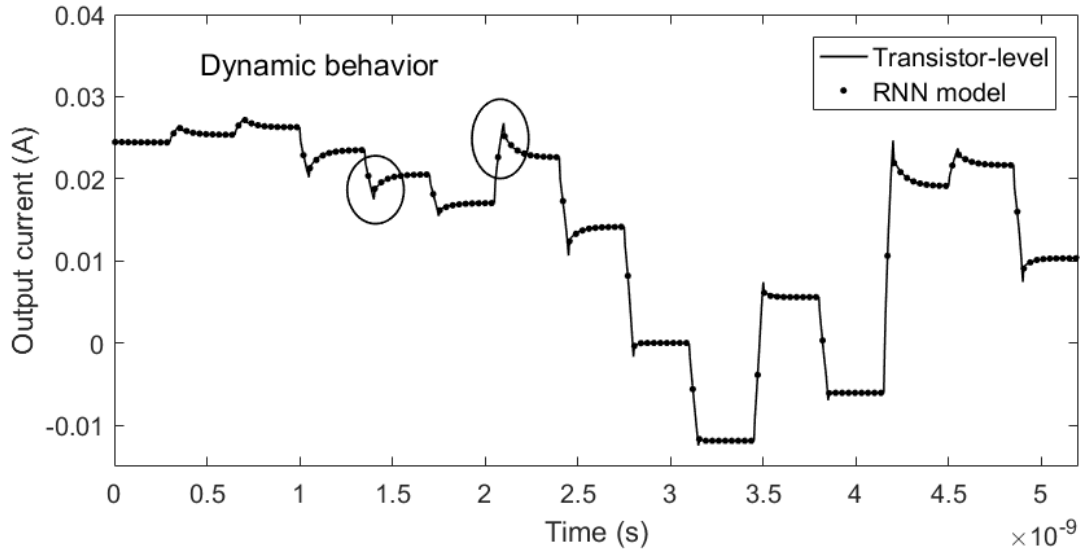


Figure 13 – Buffer output current for input HIGH from transistor-level model (solid line) and from RNN model (dotted line).

For a nonlinear buffer circuit, when the buffer operation is fast, the dynamic characteristics of the buffer become predominant, as shown in Figure 13 where a piecewise-linear voltage source is connected at the buffer output port and the buffer input is set high. In such cases, sub-models need to be capable of capturing the nonlinearity associated with the dynamic response of the driver. RNNs, which include feedback paths allowing for taking into account the previous samples, are shown to be capable of modeling nonlinear dynamic circuits [19], [37]. To efficiently capture the memory effects

and the nonlinear dynamic behavior of the buffer, the previous time instances of the buffer output voltage and current are considered and the sub-models can be learnt as RNN models:

$$i_n(t) = i_n^{RNN} \begin{pmatrix} v_o(t), v_o(t-h), \dots, v_o(t-rh) \\ i_n(t-h), \dots, i_n(t-rh) \end{pmatrix}, \quad (11)$$

$$n = H, L,$$

where v_o is the output voltage, h is the sampling time step, and r is referred to as the dynamic order of the model, which usually has a value of 1 or 2 for typical buffers [35]. It can be seen from Figure 13 that with the inclusion of two previous time instances, the output current values learnt by the RNN sub-model match accurately with the transistor-level simulation results.

For the weighting functions, $w_H(t)$ and $w_L(t)$, the models in [35] and [37] assume time concatenation of the successive transients triggered by the different input state transitions. This approach, however, does not apply to oscillators since they do not have any input signals. On the other hand, the approach of directly connecting the oscillator model to the buffer model may not work well, when the oscillation frequency is high where there is not enough time for the buffer to reach steady state before each transition. Therefore, a different method of extracting and modeling the weighting functions is required.

2.3.2 AugNN-based Model

Based on the above discussion, an AugNN-based model is proposed for oscillators with buffers, as shown in Figure 14, where the periodicity of the oscillatory weighting functions is captured using an AugNN with multiple output units and the nonlinear dynamic behavior of the output stage is captured using RNN sub-models. The input for the AugNN consists of a chronologically-ordered sequence of time steps which is mapped to the folded and normalized phase within a range of $[0, 1)$ by the periodic unit [22]. At the output layer, the AugNN generates the weighting coefficients, $w_H(t)$ and $w_L(t)$, which scale the transient contributions of the upper and lower devices' sub-model current sources, $i_H^{RNN}(t)$ and $i_L^{RNN}(t)$, at the buffer output port. It is important to note that the use of one AugNN with multiple output units is necessary to ensure the two weighting functions oscillate at the same frequency by sharing the same periodic unit.

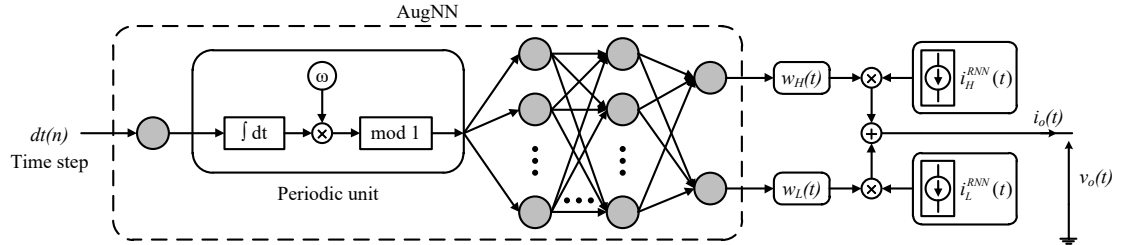


Figure 14 – Proposed AugNN-based model for oscillators including output buffers.

The training of the proposed model consists of two parts:

1) *RNN sub-models*: The training of the RNN sub-models can be carried out by connecting a multilevel piecewise-linear voltage source at the buffer output port to generate the excitation identification signals as shown in Figure 13, with the buffer input, v_x , fixed at HIGH and LOW respectively [35], [37]. Once the data is collected, RNNs in (11) are trained accordingly to learn the sub-models.

2) *AugNN*: After the sub-models are obtained, the oscillatory weighting coefficients need to be extracted for AugNN training. This can be done by running the oscillator with the buffer output port connected to two different loads Z_1 and Z_2 , e.g., a 50- Ω resistor for load Z_1 and the series connection of a V_{dd} voltage source and a 50- Ω resistor for load Z_2 . The weighting coefficients can be extracted using the method of matrix inversion [35], [37]. Once the weighting functions are extracted, the AugNN can be trained using the gradient-based training algorithms in [22] modified for multiple output neurons. After the training, the behavioral model is implemented in Verilog-A.

2.3.3 Modeling Example

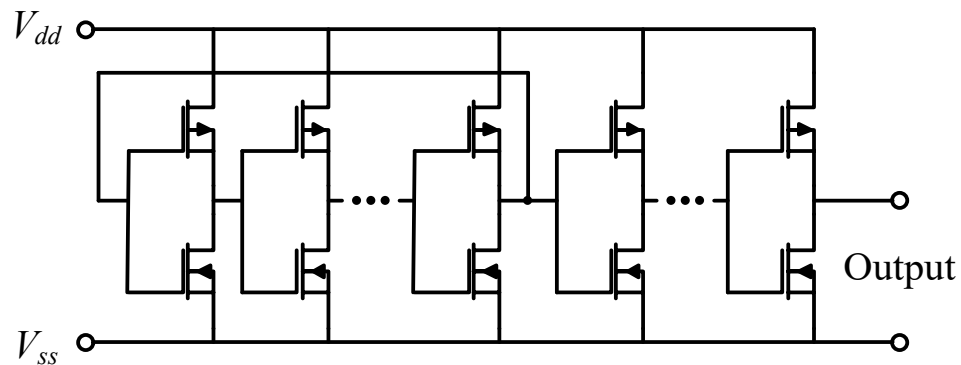


Figure 15 – Ring oscillator circuit including an output buffer.

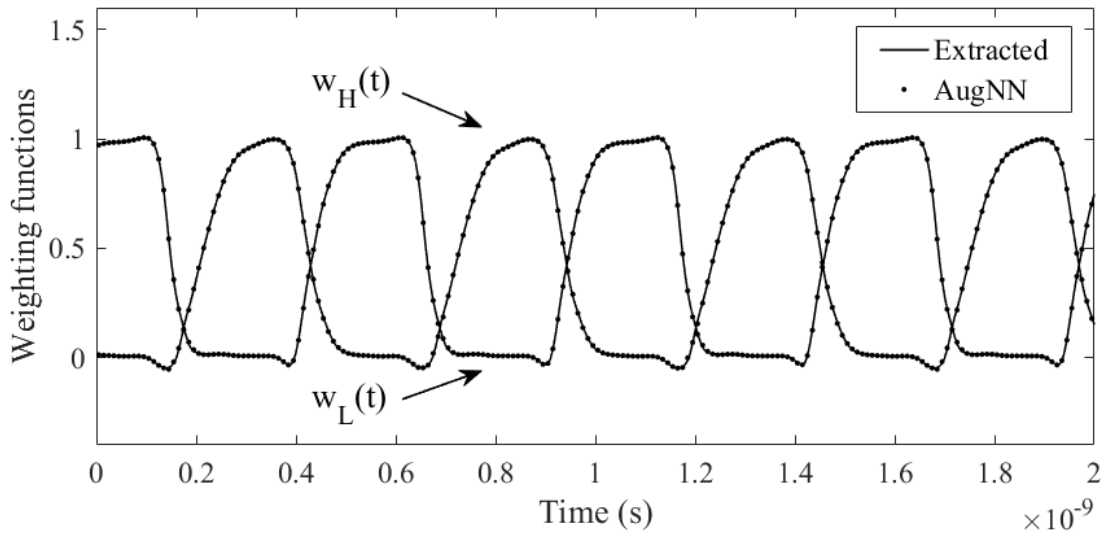


Figure 16 – Weighting functions extracted (solid line) and learnt by the AugNN (dotted line).

To demonstrate the accuracy and efficiency of the proposed model, a modeling example is generated using a transistor-level ring oscillator circuit including an output buffer as shown in Figure 15. The transistor-level simulation data generated using Spectre is used to train the model. The RNNs for sub-models contain 10 neurons in the hidden layer. Based on the method described in the previous section, the weighting coefficients are extracted as shown in Figure 16. A training root-mean-squared error (RMSE) of 0.0034 is achieved using an AugNN with a hidden layer consisting of 15 neurons. The oscillation frequency is learnt by the AugNN as the parameter, ω , in the periodic unit, which is 1.9481 GHz. The RNNs and AugNN are then implemented as a Verilog-A model.

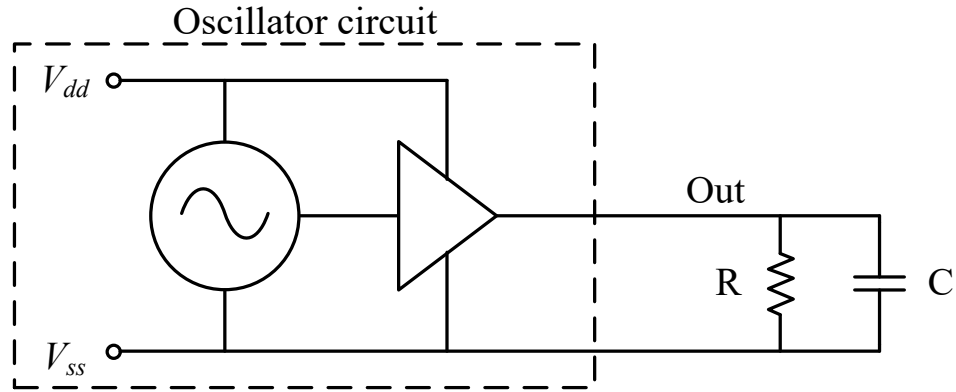


Figure 17 – Test setup for model validation.

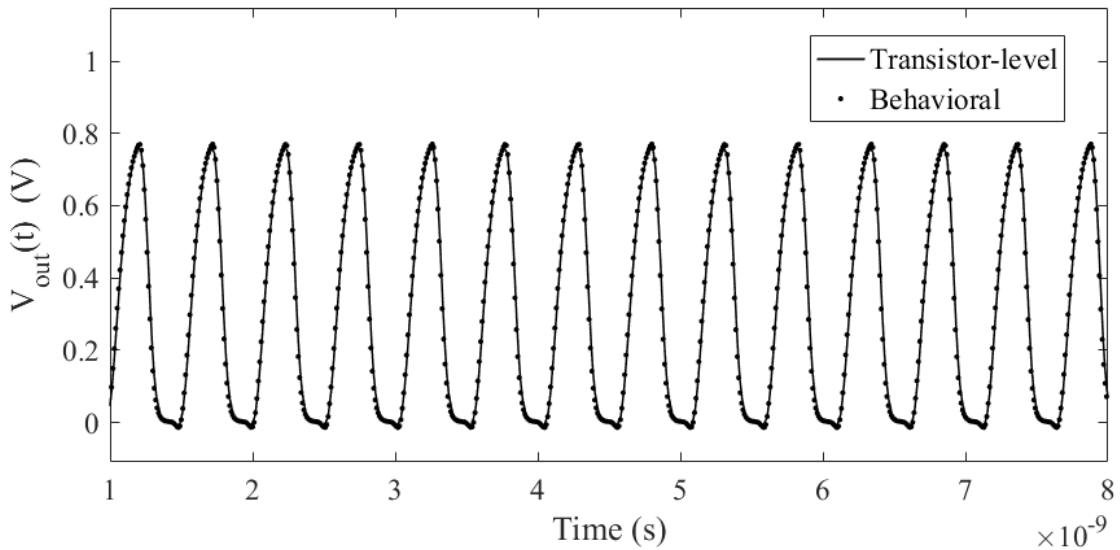


Figure 18 – Output voltage waveform from transistor-level model (solid line) and from behavioral model (dotted line).

To test the performance of the trained model, the oscillator circuit is connected to a load resistor of 75Ω and a capacitor of 1.5 pF in parallel as shown in Figure 17, and simulated in Spectre. It can be seen from Figure 18 that the simulation results of the behavioral model match well with the transistor-level model. A FOM value of 98.047 is achieved in Figure 18, and the behavioral model achieves a reduction in simulation time of up to 96%.

2.4 Modeling of VCOs

2.4.1 Mathematical Model

The steady-state time-domain output waveforms of a VCO with multiple output ports can be expressed as:

$$[V_{out,1}(t), V_{out,2}(t), \dots, V_{out,N_{out}}(t)] = F\left(\int \omega(t)dt\right), \quad (12)$$

where $\omega(t)$ is the instantaneous frequency, N_{out} is the number of outputs, and $F()$ is a periodic function that captures the shape of the waveforms. Specifically, for an oscillator with fixed frequency, $\omega(t)$ is a constant, whereas, for a VCO, $\omega(t)$ is a function of the frequency control voltage, i.e., $\omega(t) = G(v_c(t))$ [41]. In Equation 12, the function argument, $\int \omega(t)dt$, is the total oscillation phase, which accumulates at a rate of $\omega(t)$. The phase is mapped to the output through $F()$, a periodic function with respect to phase.

To derive a behavioral model, we reformulate Equation 12 in the discrete-time domain as:

$$[V_{out,1}(n), V_{out,2}(n), \dots, V_{out,N_{out}}(n)] = F\left(\sum_{i=1}^n \omega(i)dt(i)\right), \quad (13)$$

where n is the time index in the discrete-time domain.

2.4.2 Augmented Neural Network for VCOs

From Equation 13, we can see that the output of a VCO is a function of the total oscillation phase, which can be obtained by integrating the product of the instantaneous frequency ω and the time step dt . Therefore, time is an inherent input for modeling oscillators. However, the total oscillation phase is unbounded, since the phase can be arbitrarily large along with the increase of time. As a result, most neural networks will fail to capture the phase-output relation of oscillators, since the unbounded input values cannot be properly handled. Moreover, for oscillators with multiple ports, the outputs are highly correlated with each other in terms of phase. This correlation must be strictly retained in the behavioral model to avoid any mismatch in their instantaneous frequencies. To overcome these difficulties, a multi-output AugNN is proposed as shown in Figure 19, where a periodic unit is introduced. In the following discussion, it can be seen that, given the model formulations for VCOs, the model for fixed-frequency oscillators is a subset with the frequency $\omega(t)$ being constant.

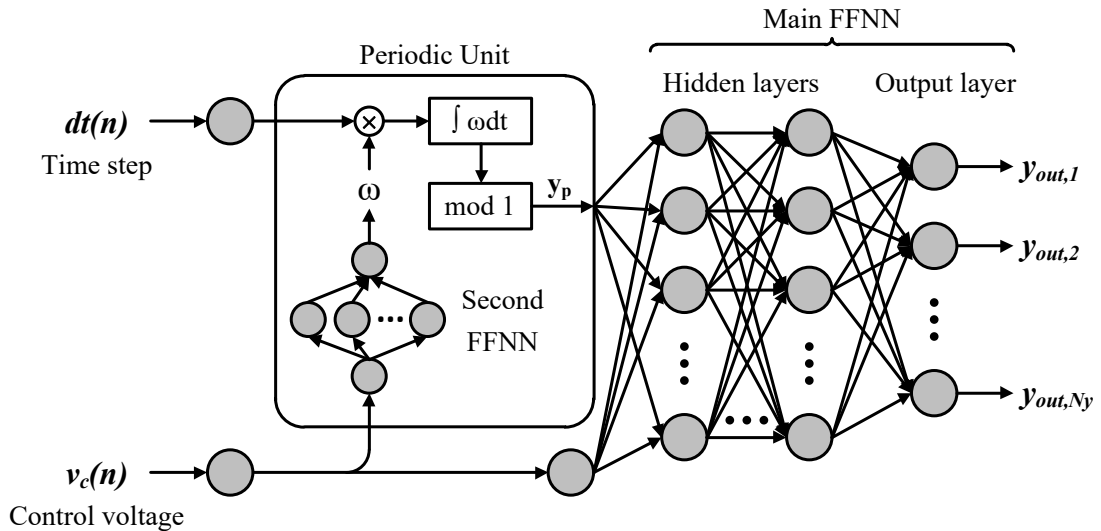


Figure 19 – Proposed AugNN model for VCOs.

The input for the proposed AugNN consists of the chronologically-ordered sequence of time steps \mathbf{dt} and the control signal which are fed into the periodic unit. The function of the periodic unit is to learn the instantaneous frequency and generate bounded oscillation phase. Since the frequency is a function of the control voltage without time dependency, this can be done by first capturing ω using an FFNN as:

$$\omega(n) = g_{PU}^{FFNN}(v_c(n)), \quad (14)$$

where $g_{PU}^{FFNN}()$ represents the formulation of the FFNN inside the periodic unit. Then the product of the instantaneous frequency ω and the time step dt is integrated to obtain the total oscillation phase, after which the modulus operation is performed to map the unbounded total oscillation phase to the folded phase that is normalized within a range of $[0, 1)$. The responses of the periodic unit can be computed as:

$$y_p(n) = \left(\sum_{i=1}^n \omega(i)dt(i) \right) \text{mod } 1. \quad (15)$$

The output of the periodic unit, \mathbf{y}_p , and the control signal, \mathbf{v}_c , are used as the input of the main FFNN. For an AugNN with multiple outputs, let N_y be the total number of the output neurons, and $y_{out,k}(n)$ be the k th output signal at the n th time sample. The values of the nodes in the output layer of the main FFNN are calculated using Figure 19 as:

$$[y_{out,1}(n), y_{out,2}(n), \dots, y_{out,N_y}(n)] = f_{main}^{FFNN}(y_p(n), v_c(n)), \quad (16)$$

where $f_{main}^{FFNN}()$ represents the formulation of the main FFNN. For the main and second FFNNs, the hyperbolic tangent function, $\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$, is used as the activation function for the hidden neurons.

2.4.3 AugNN Training

The training of the network is the process of tuning the network parameters to minimize the difference between the transistor-level circuit data $\hat{\mathbf{y}}_{out}(n)$ and the network output $\mathbf{y}_{out}(n)$, which is defined as the squared error here. For training the multi-output AugNN, the objective function is given by:

$$\min_{\Phi} \frac{1}{2} \sum_{n=1}^{N_t} \sum_{k=1}^{N_y} (y_{out,k}(n) - \hat{y}_{out,k}(n))^2, \quad (17)$$

where N_t is the number of time samples in the training dataset. It's important to note that the training data fed into the network must be in the chronological order to make sure the phase integration is calculated correctly. Let the parameters of the AugNN be denoted as Φ , $\Phi = [\mathbf{w}_M^T \mathbf{b}_M^T \mathbf{w}_{PU}^T \mathbf{b}_{PU}^T]^T$, where \mathbf{w}_M and \mathbf{b}_M are the vectors of the weights and biases of the main FFNN, \mathbf{w}_{PU} and \mathbf{b}_{PU} are the vectors of the weights and biases of the second FFNN inside the periodic unit, and T denotes transpose. Gradient-based training algorithms can be used to optimized the network parameters, where the Jacobian matrix \mathbf{J} is computed using the BP method to calculate the derivatives of the network outputs with respect to each parameter in Φ . The derivatives of the FFNNs can be computed by normal BP [16].

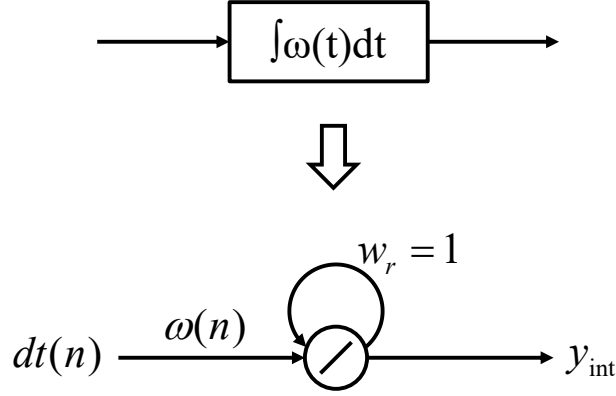


Figure 20 – Recurrent node representation of the integral node in the periodic unit.

Particularly, the integral operation in the periodic unit can be represented equivalently using a recurrent node with a fixed recurrent weight of one, as shown in Figure 20. Based on this representation, the integral output, y_{int} , is iteratively computed as:

$$y_{int}(n) = y_{int}(n - 1) + \omega(n)dt(n). \quad (18)$$

However, there are discontinuity points brought by the modulus operation $y = x \bmod 1$ in the periodic unit, where x has integer values. In order to overcome this discontinuity difficulty, the value of one can be used as the derivatives of the *mod* function at the discontinuous points, which is equal to the derivatives on both the left and the right sides of the discontinuity. The derivatives for the periodic unit can be formulated as:

$$\frac{\partial y_{out,k}(n)}{\partial \Phi_{PU}} = \sum_{m=1}^n \frac{\partial y_{out,k}(n)}{\partial \omega(m)} \frac{\partial \omega(m)}{\partial \Phi_{PU}}, \quad (19)$$

where $\Phi_{PU} = [\mathbf{w}_{PU}^T \mathbf{b}_{PU}^T]^T$ is the vector of the parameters of the periodic unit, $\partial \omega(m) / \partial \Phi_{PU}$ can be computed for the second FFNN using BP method, and

$$\begin{aligned}
\frac{\partial y_{out,k}(n)}{\partial \omega(m)} &= \frac{\partial y_{out,k}(n)}{\partial y_p(n)} \frac{\partial y_p(n)}{\partial \omega(m)} \\
&= \frac{\partial y_{out,k}(n)}{\partial y_p(n)} dt(m), 1 \leq m \leq n,
\end{aligned} \tag{20}$$

where $\partial y_{out,k}(n)/\partial y_p(n)$ can be computed for the main FFNN using BP method. After this procedure, the Jacobian matrix of the k th output can be formulated as:

$$\mathbf{J}_k = \begin{bmatrix} \frac{\partial y_{out,k}(1)}{\partial \Phi_{\mathbf{M}}} & \frac{\partial y_{out,k}(2)}{\partial \Phi_{\mathbf{M}}} & \dots & \frac{\partial y_{out,k}(N_t)}{\partial \Phi_{\mathbf{M}}} \\ \frac{\partial y_{out,k}(1)}{\partial \Phi_{\mathbf{PU}}} & \frac{\partial y_{out,k}(2)}{\partial \Phi_{\mathbf{PU}}} & \dots & \frac{\partial y_{out,k}(N_t)}{\partial \Phi_{\mathbf{PU}}} \end{bmatrix}^T, \tag{21}$$

where $\Phi_{\mathbf{M}} = [\mathbf{w}_{\mathbf{M}}^T \mathbf{b}_{\mathbf{M}}^T]^T$ is the vector of parameters of the main FFNN. The entire Jacobian matrix is then given by:

$$\mathbf{J} = [\mathbf{J}_1^T \quad \mathbf{J}_2^T \quad \dots \quad \mathbf{J}_{N_y}^T]^T. \tag{22}$$

Using this gradient scheme, the gradient-based training algorithms, the Levenberg-Marquardt method, can be applied to train the AugNN model, and the corresponding step Δ for updating the network parameters in each training iteration can be calculated as [46]:

$$(\mathbf{J}^T \mathbf{J} + \mu \mathbf{I}) \Delta = -\mathbf{J}^T \mathbf{q}, \quad \mu \geq 0, \tag{23}$$

where \mathbf{I} is the identity matrix, μ is the damping parameter, and

$$\mathbf{q} = [\mathbf{q}_1^T \quad \mathbf{q}_2^T \quad \dots \quad \mathbf{q}_{N_y}^T]^T, \tag{24}$$

where

$$\mathbf{q}_k = \begin{bmatrix} y_{out,k}(1) - \hat{y}_{out,k}(1) \\ y_{out,k}(2) - \hat{y}_{out,k}(2) \\ \vdots \\ y_{out,k}(N_t) - \hat{y}_{out,k}(N_t) \end{bmatrix}. \quad (25)$$

After training, the AugNN model can be implemented as a Verilog-A model.

2.4.4 VCO Modeling Example

To demonstrate the accuracy of the AugNN model, a modeling example is generated using a transistor-level VCO circuit model that has the lowest frequency of 0.216 GHz and the highest frequency of 0.306 GHz with the control voltage v_c ranging from 1 V to 3 V. Its simplified schematic is shown in Figure 21.

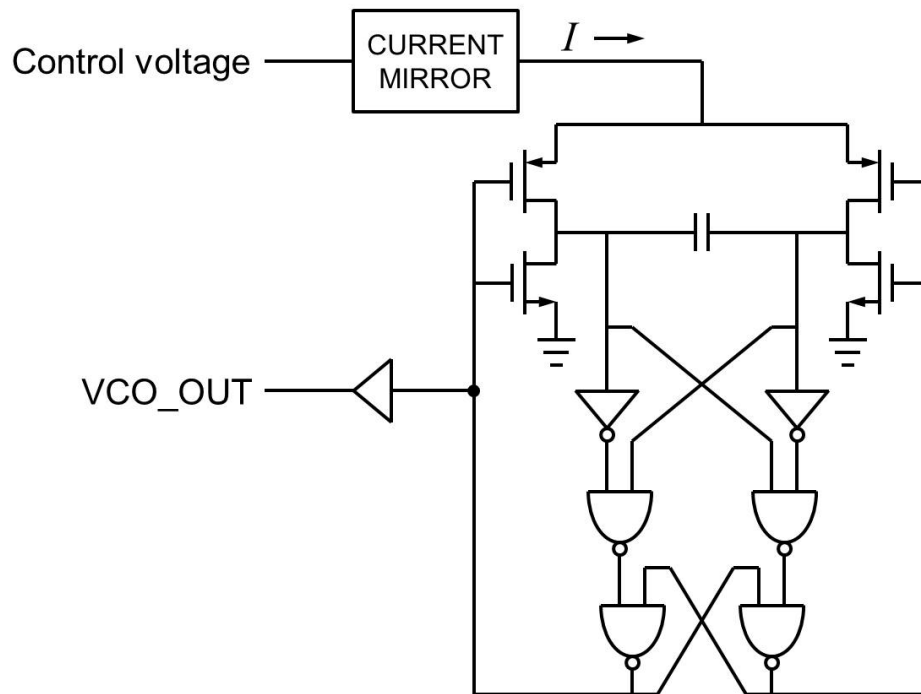


Figure 21 – Simplified schematic of the target VCO.

As can be seen that the VCO circuit has single output port. Therefore, the proposed AugNN can be simplified with single output unit as shown in Figure 22. The training data is generated using Spectre with v_c varying from 3 V to 1 V in steps of -0.2 V as shown in Figure 23. When the root-mean-squared error is expressed as a percentage of the peak-to-peak amplitude, a training RMSE/p-p of 0.61% is achieved using an AugNN containing 20 neurons in the first and 10 neurons in the second hidden layer of the original FFNN, and 5 neurons in the single hidden layer of the second FFNN in the periodic unit. The one time training process costs around 5 min.

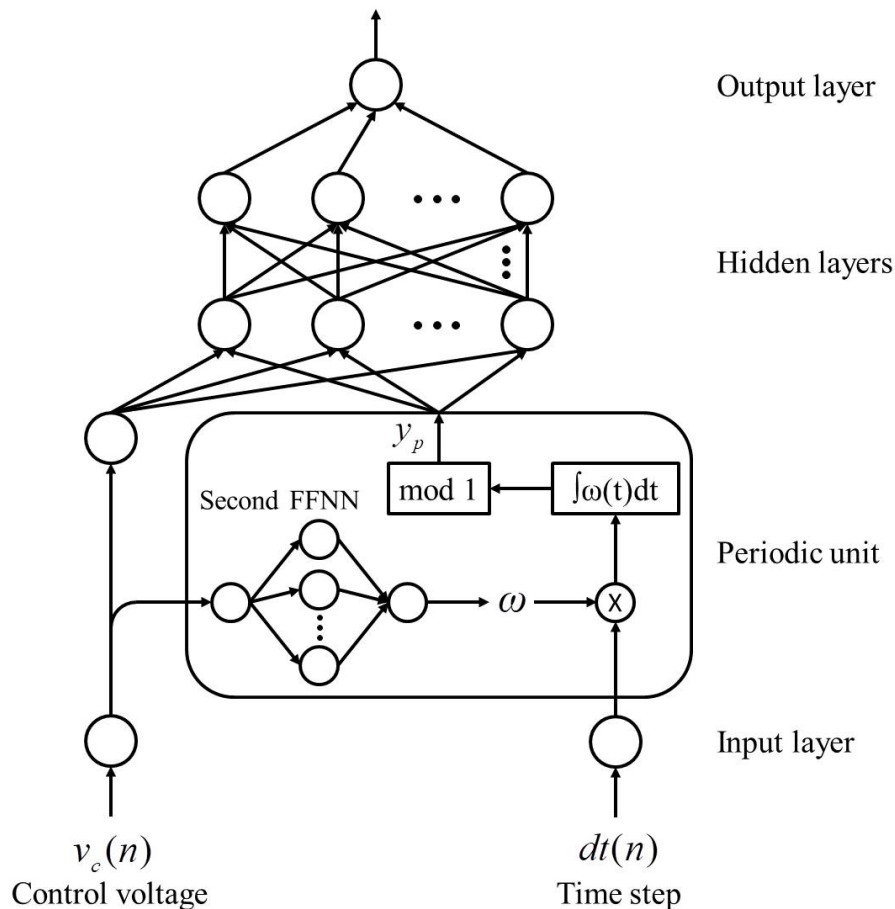


Figure 22 – AugNN with single output unit for VCO modeling.

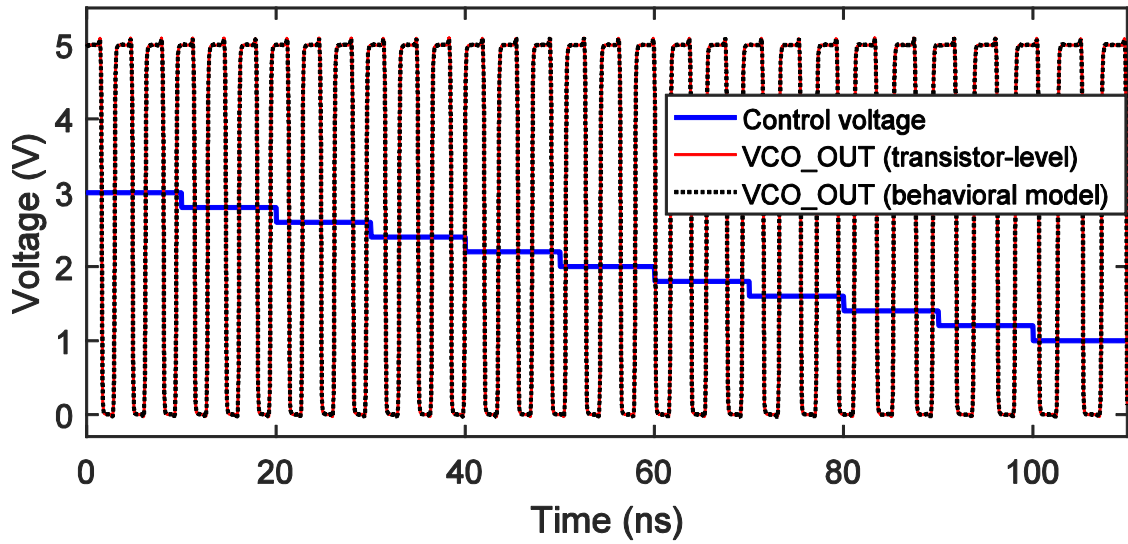


Figure 23 – Training waveforms of control voltage (thick solid blue line), and the output of transistor-level (thin red line) and behavioral model (dotted line).

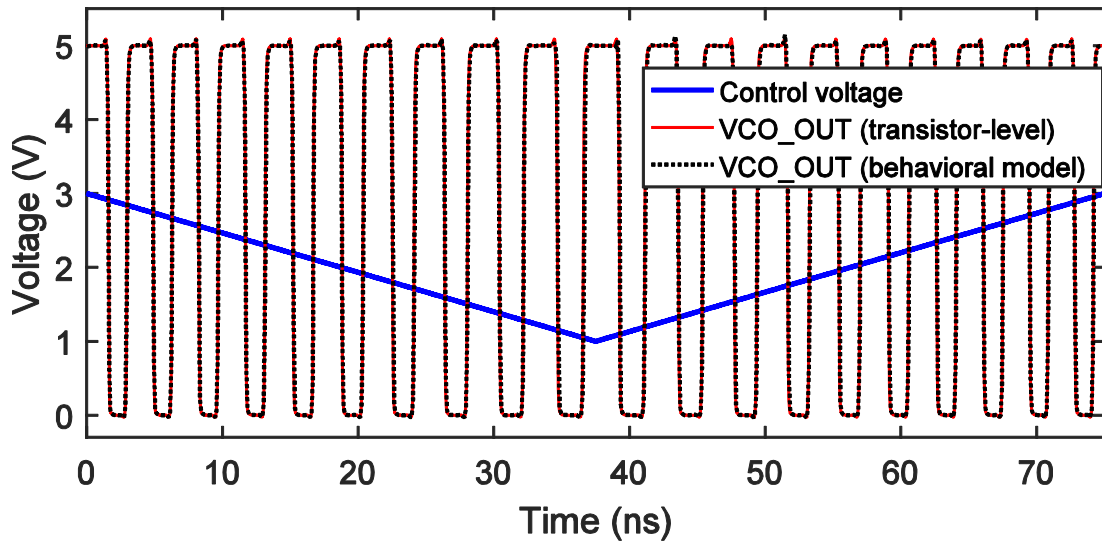


Figure 24 – Test waveforms of control voltage (thick solid blue line), and the output of transistor-level (thin red line) and behavioral model (dotted line).

To test the performance of the trained model, a frequency sweep is applied by continuously changing v_c between its maximum and minimum as shown in Figure 24. Good agreement between the output of the behavioral and transistor-level model can be

observed, while the behavioral model achieves a reduction in simulation time of up to 93%. A FOM value of 99.15 is achieved.

2.5 Modeling of VCO Including I/O behavior

As a key role of the interface between the internal circuit core and the load at the output, buffers can be included in the VCO designs as shown in Figure 25. This brings extra nonlinear dynamics to the voltage-controlled oscillatory port behavior, which must be accurately captured.

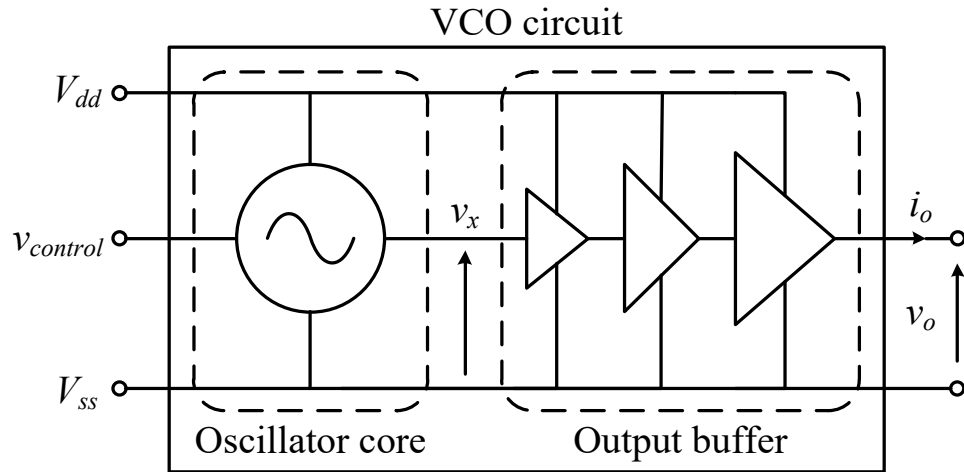


Figure 25 – VCO circuit including output buffer.

2.5.1 AugNN-based Model

From the previous discussion, it can be seen that the behavior of the I/O circuit can be decompose into two parts, where sub-models for HIGH and LOW states and the corresponding weighting functions embody the nonlinear dynamic behavior of the output port, as shown in Equation 9. For output buffers with VCOs, however, the previous modeling approaches cannot be applied directly, since these weighting functions are not

simply concatenated fixed-pattern signals but functions of control voltage. Therefore, methods of extracting and modeling the weighting functions are required.

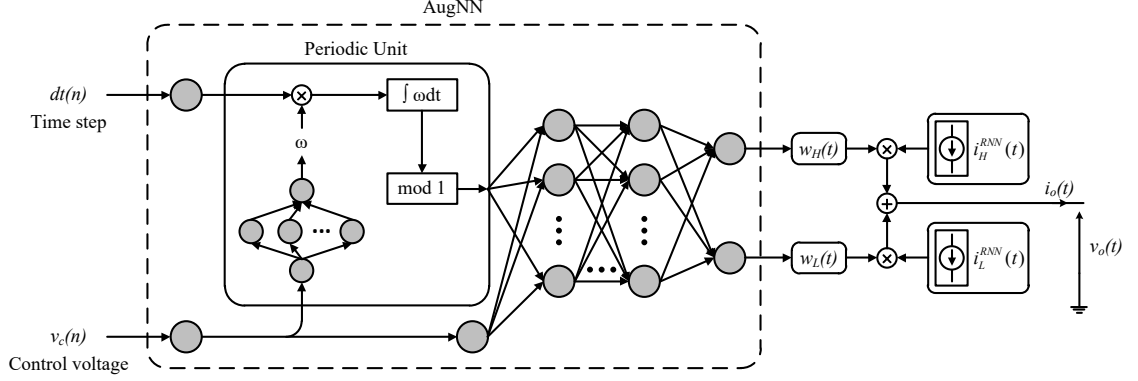


Figure 26 – Proposed AugNN-based model for VCOs including output buffers.

Based on above discussion, for VCOs including output buffers, an AugNN-based model in Figure 26 is proposed. In the proposed model, an AugNN with two output neurons is used to capture the voltage-controlled oscillatory weighting functions, which control the transitions of the buffer output stage. Two RNNs are used to capture the sub-models that take into account the nonlinear dynamic current-voltage relation at the output port. The output current can be expressed as:

$$i_o(t) = w_H^{AugNN}(t)i_H^{RNN}(t) + w_L^{AugNN}(t)i_L^{RNN}(t). \quad (26)$$

From Figure 26 it can be seen that the periodic unit of the AugNN maps the total oscillation phase to the folded phase normalized within $[0, 1)$ periodically for the weighting functions, $w_H(t)$ and $w_L(t)$, to oscillate in a controlled manner. It is worth mentioning that it is necessary to use one single AugNN with multiple output units to retain the phase correlation between the two weighting functions and to ensure the two

weighting functions oscillate at the same instantaneous frequency by sharing the same periodic unit.

2.5.2 Modeling Process

The proposed behavioral model can be obtained by carrying out the following key steps.

1) *Dynamic sub-model characterization*: In this step, the buffer's output port is driven to obtain the transient voltage and current signals that contain the information on its nonlinearities and dynamics. Figure 27 (a) shows the setup required to collect the data for sub-model characterization, where the buffer is manually set to HIGH or LOW state, and the buffer's output port is excited using piecewise-linear voltage signals. These signals must be carefully designed in order to excite the nonlinear dynamic behavior at the output port. Such driving waveforms can be multilevel signals containing different slopes and different transition heights. More details regarding identification signals can be found in [35] and [37]. For better excitation of the dynamics of the buffer's output port, small signal noise can be added as suggested in [35]. It is important to note that these waveforms need to sufficiently cover the range of operating voltages. Therefore, transient analysis should be performed for buffer in HIGH and LOW states respectively to obtain the corresponding output current signals. Once the data are collected, RNNs are trained accordingly to learn the sub-models, $i_H^{RNN}(t)$ and $i_L^{RNN}(t)$.

2) *Voltage-controlled oscillatory weighting function extraction*: After the sub-models are obtained, the voltage-controlled oscillatory weighting coefficients are extracted to train the AugNN. Figure 27 (b) shows the setup required to collect the data

for weighting coefficients extraction. In this step, the frequency control port of the VCO is fed with step voltage signal to obtain the transient voltage and current signals at the output port, which carry information on its voltage-controlled frequency characteristics. The step control signal performs a sampling of the control voltage, and covers the range of control voltage. It is worth mentioning that each step interval should be at least longer than one full oscillation cycle. The output port is connected to a load Z_i . Since two unknowns exist, weighting functions, w_H and w_L , can be extracted using two different loads Z_1 and Z_2 . A good combination is a resistor for load Z_1 and the series connection of a V_{dd} voltage source and a resistor for load Z_2 [37]. Matrix inversion can be used to extract the weighting functions as:

$$\begin{bmatrix} w_H(t) \\ w_L(t) \end{bmatrix} = \begin{bmatrix} i_H^{Z_1}(t) & i_L^{Z_1}(t) \\ i_H^{Z_2}(t) & i_L^{Z_2}(t) \end{bmatrix}^{-1} \begin{bmatrix} i_o^{Z_1}(t) \\ i_o^{Z_2}(t) \end{bmatrix}. \quad (27)$$

Using the extracted weighting functions, the training algorithms proposed can be applied to train the AugNN. The trained behavioral model can be implemented in Verilog-A. Details of model implementation are discussed in APPENDIX A.

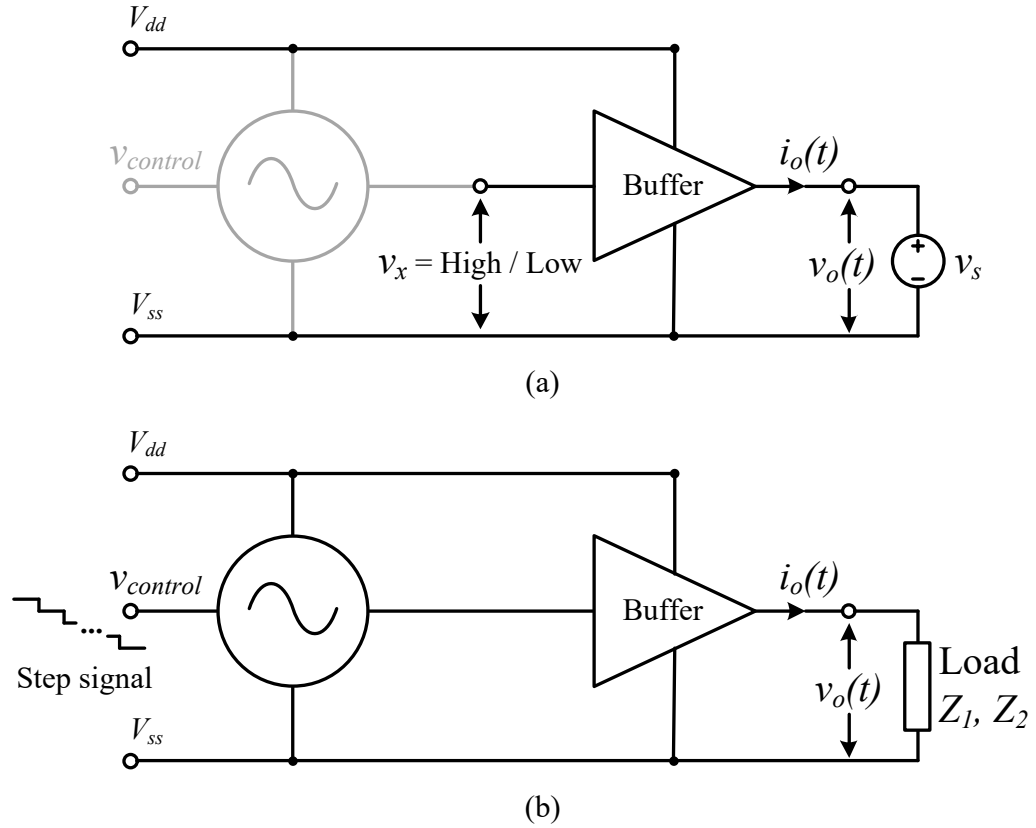


Figure 27 – Test-benches used for the collection of training data required for model generation. (a) Transient analysis for the extraction of nonlinear dynamic sub-models. (b) Transient analysis for the extraction of weighting functions using step control voltage signal.

2.5.3 Model Validation

In this modeling example, a VCO circuit that includes the output buffer is considered, as shown in Figure 28. The VCO has the lowest frequency of 0.83 GHz and the highest frequency of 1.27 GHz with the control voltage $v_{control}$ ranging from 0.2 V to 0.8 V.

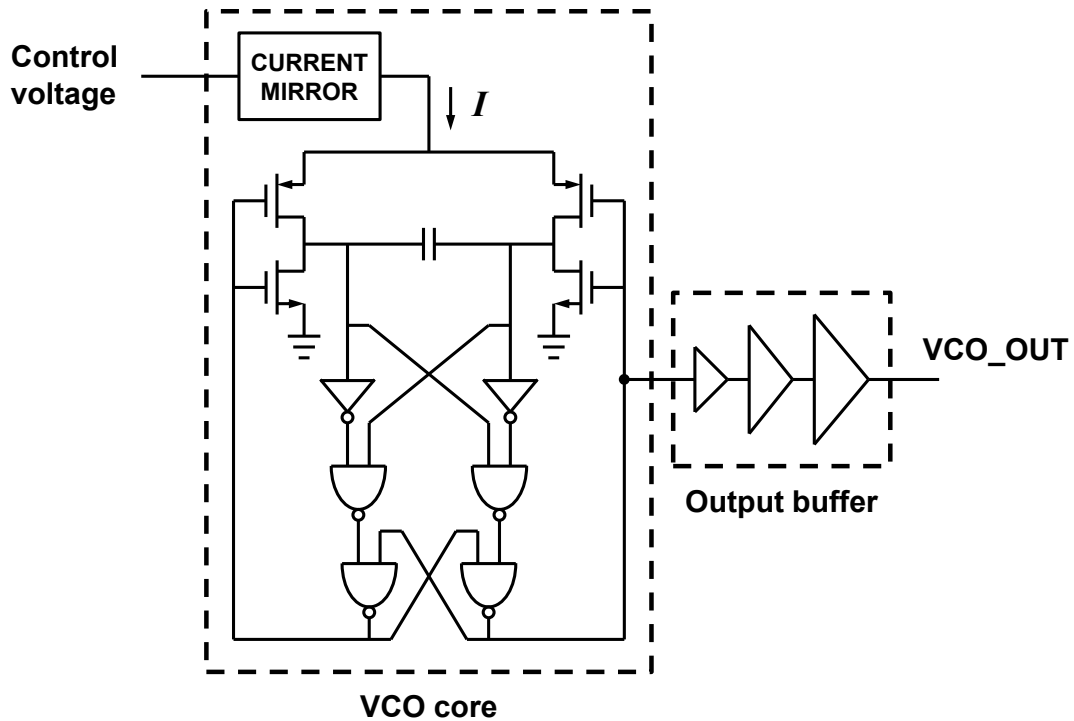


Figure 28 – Simplified schematic of the VCO circuit including output buffer.

The training data is generated from simulations using Spectre to train the model shown in Figure 26, based on the modeling process in the previous section. For sub-models, different dynamic orders are tested, and no further improvement in accuracy was found for dynamic orders larger than two. Therefore, a dynamic order of $r = 2$ was adopted. The sampling time step is 5 ps. Each RNN contains a hidden layer consisting of 10 neurons. The back propagation through time (BPTT) algorithm with the Levenberg-Marquardt method is used to train the RNNs [47].

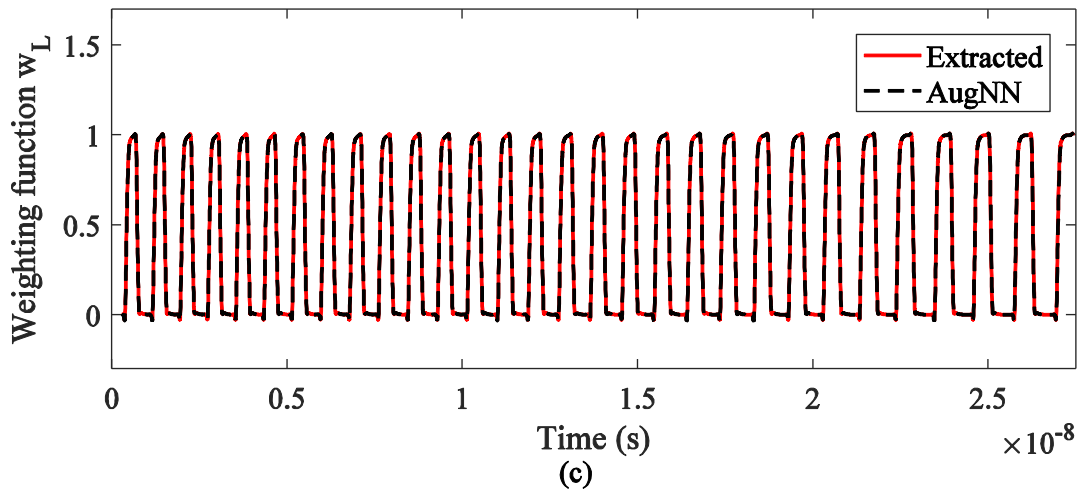
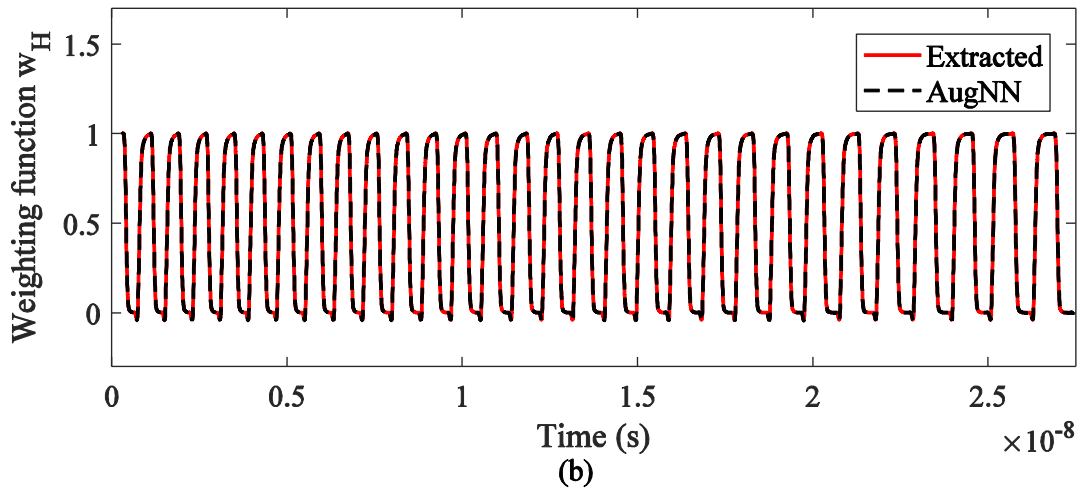
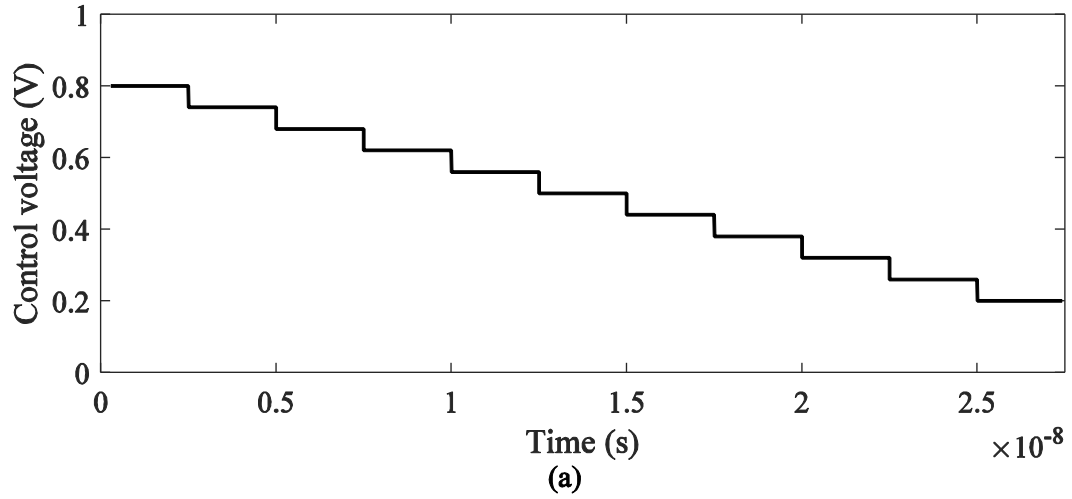


Figure 29 – (a) Training waveform of control voltage. (b) Weighting function w_H extracted and learnt using AugNN. (c) Weighting function w_L extracted and learnt using AugNN.

For voltage-controlled oscillatory weighting functions extraction, the control voltage signal varies from 0.8 V to 0.2 V in steps of 0.06 V, as shown in Figure 29 (a). A resistor of 50 Ω is used for loads Z_1 and Z_2 . Using matrix inversion in Equation 27, the weighting functions are extracted and learnt using an AugNN, as shown in Figure 29 (b) and (c). The main FFNN of the AugNN contains 20 neurons in the first hidden layer and 15 neurons in the second hidden layer. And the second FFNN in the periodic unit has a single hidden layer of 5 neurons. A training MSE of 2.40E-5 was achieved. After training, the RNNs and AugNN are implemented as a behavioral VCO model in Verilog-A.

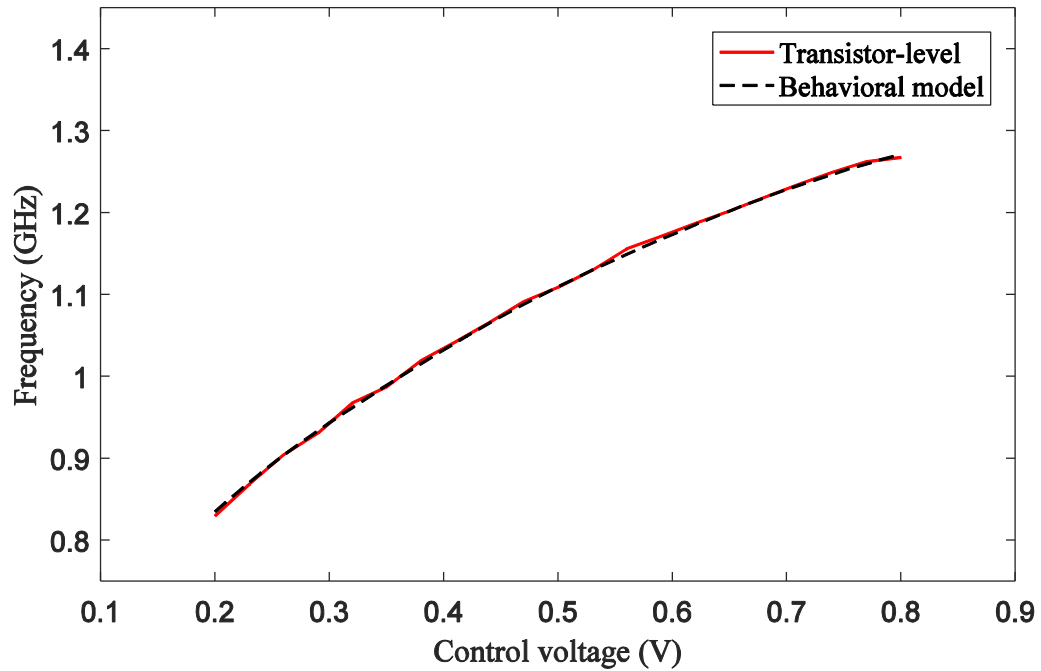


Figure 30 – Tuning curve of the VCO obtained from the transistor-level model and the behavioral model.

To assess the accuracy of the trained model, the tuning curve is shown in Figure 30. Good agreement can be seen, showing that the nonlinearity of the tuning curve is captured accurately using the behavioral model.

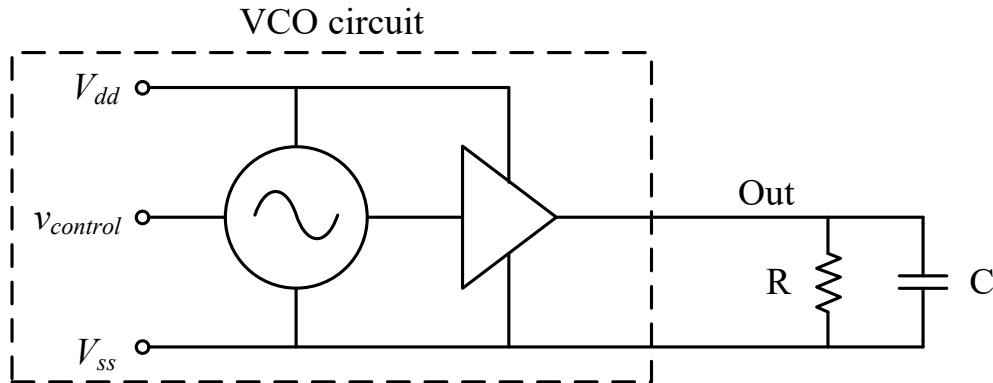


Figure 31 – Model validation setup.

Furthermore, to test the time-domain performance, the VCO circuit is connected to a load resistor of 75Ω and a capacitor of 0.5 pF in parallel, as shown in Figure 31, which is different from the load used for model extraction. A frequency sweep is applied by continuously changing the control voltage between the maximum and minimum of the tuning range. The simulations are performed using Spectre. Figure 32 shows the simulation waveforms, where the results of the behavioral model and the transistor-level model match well. An MSE of $8.47\text{E-}4$ is achieved in Figure 32 with a FOM value of 98.62 . To assess the speed-up, a simulation for a period of 10 ms was tested. The CPU time for the transistor-level model was $8 \text{ m } 43 \text{ s}$, whereas the corresponding time for the behavioral model was 33 s using the same computer. The behavioral model achieves a simulation time reduction of 94% . An even higher speed-up would be possible for more complex oscillator circuits, as the computational time reduction using the behavioral

model compared to the transistor-level model increases with increase in the complexity of the circuit.

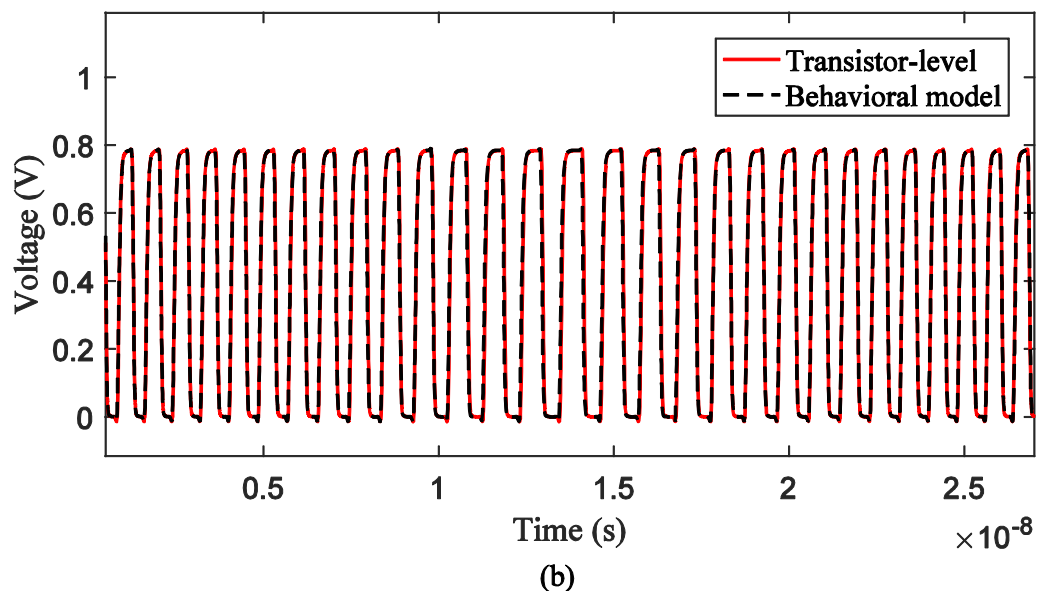
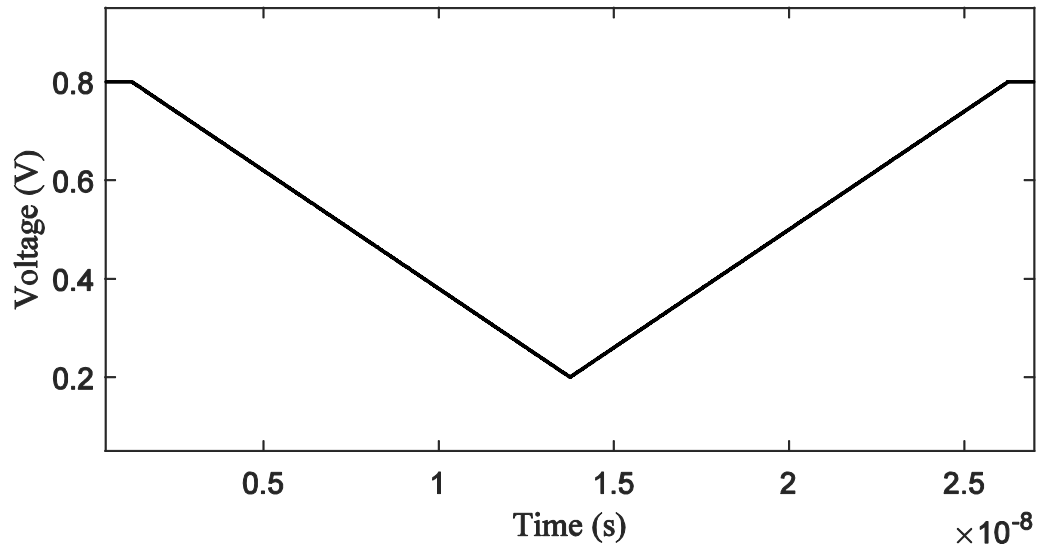


Figure 32 – (a) Control voltage waveform used in the test case. (b) Output waveforms of the transistor-level simulation and the behavioral model simulation.

2.6 Summary

In this chapter, multi-output AugNNs are proposed for the modeling of steady-state multi-phase oscillators in time-domain with the corresponding gradient scheme and training methodology. For VCOs including output buffers, an AugNN-based model is proposed, where RNNs are used to capture the nonlinear dynamic behavior of the buffer, and the relation between the frequency and control voltage is learnt explicitly. As a black-box approach, the resulting model protects IP. The Verilog-A implementation makes the model compatible with existing commercial tools. The fidelity and efficiency of the proposed model have been demonstrated with examples using transistor-level oscillator circuits.

CHAPTER 3. BEHAVIORAL MODELING OF TUNABLE I/O DRIVER WITH PRE-EMPHASIS

3.1 Introduction

In ICs, I/O drivers of CMOS technology are widely used as the interface between the internal circuit core and the load at the output. Mixed-signal drivers play a critical role in generating high quality communication signals in the channels connecting different IC blocks. The design and optimization of such high-speed links rely on iterative simulations and accurate timing analysis using commercial EDA tools. As the circuit complexity increases, however, it is becoming more challenging because of the large amount of CPU time required for transistor-level circuit simulations. In these scenarios, behavioral models can provide significant speed-up as they reduce the complexity of the original transistor-level circuit models. At the same time, behavioral models are independent of the knowledge of the internal design of the circuit components, and thus protect IP, which is an important advantage.

Among the existing techniques for modeling drivers, IBIS model is one of the most popular approaches [34]. In this approach, however, the physical effects taken into account are decided by the pre-defined equivalent circuit, thus leaving limited ability to capture advanced features of modern I/O drivers. Moreover, the table-based format has limitations in representing characteristics with multivariate dependency. For PI related simulations, while some properly prepared models have shown some level of ability to

capture power delivery network (PDN) voltage and current behaviors, they are not necessarily accurate in capturing the PDN voltage induced timing behavior (jitter).

As alternative approaches for behavioral modeling, enhanced and parametric models have been proposed [35]–[37], [39], [40], [48]–[54], providing improved accuracy and variation of features for modeling I/O drivers of CMOS technology. In [35], [39], [40], [48]–[50], various driver models have been proposed as well. These models, however, do not take into account power supply variation and can be used for SI simulations only. In [36], [37], [51], [52], modeling of driver circuits including power supply is demonstrated. These approaches, however, do not consider the effects of power supply voltage variation on the transition behavior, which can lead to non-negligible timing inaccuracy. In [53] and [54], enhanced models are proposed with the inclusion of power supply effects on both state switching and sub-models. These models, however, do not address the modeling strategy when switched input logic states are shorter than the pre-emphasis duration under power supply variation. Moreover, as I/O circuits keep evolving, advanced features are emerging in modern driver designs, including control parameters which need to be tuned to control the behavior of the driver circuit. However, the modeling of these important tunable characteristics has not been discussed in the previous models.

In this chapter, the multi-port behavioral model for tunable drivers with pre-emphasis including power supply noise is discussed. The formulation of the proposed state-aware weighting functions and the finite state machine (FSM) of the model algorithm are presented, along with the corresponding modeling procedure.

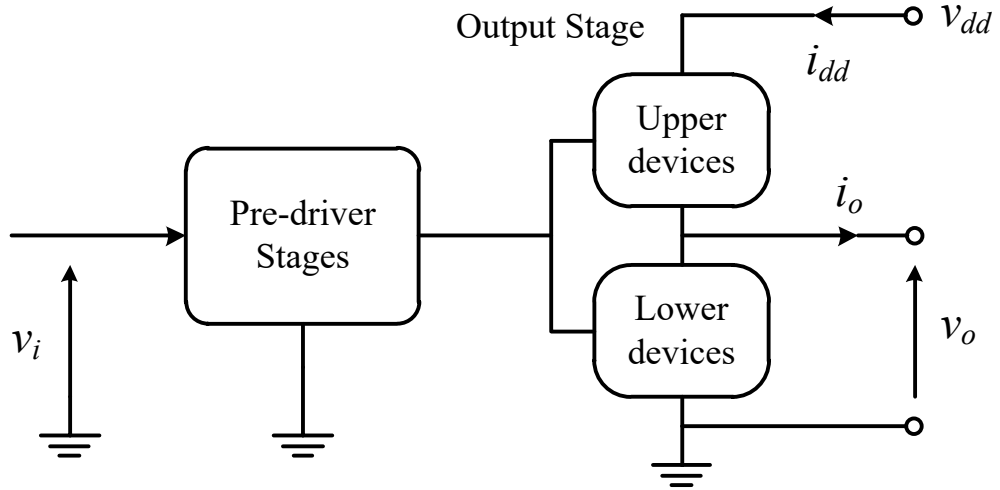


Figure 33 – Generic driver electrical structure.

3.2 Model Structure

Figure 33 shows a general driver electrical structure. For a driver circuit, the current at the output port at a certain time t , can be expressed as the summation of two current components [34]–[40], [48]–[54]:

$$i_o(t) = w_{o,1}(t)i_{o,1}(t) + w_{o,2}(t)i_{o,2}(t), \quad (28)$$

where $i_{o,1}(t)$ and $i_{o,2}(t)$ represent the nonlinear dynamic sub-models accounting for the output current of the last stage of the driver when the driver input is at the high (H) and low (L) voltage states, respectively. Similarly, the current at the supply port can be expressed as:

$$i_{dd}(t) = w_{dd,1}(t)i_{dd,1}(t) + w_{dd,2}(t)i_{dd,2}(t) + w_{dd,3}(t), \quad (29)$$

where $i_{dd,1}(t)$ and $i_{dd,2}(t)$ are the nonlinear dynamic current sub-models of the supply port for driver input high and low. In Equation 28 and Equation 29, the sub-model functions, $i_{v,n}(t)$, can be defined using parametric relations as:

$$\begin{aligned} i_{x,n}(t) &= i_{x,n}(v_o(t), v_{dd}(t), \mathbf{D}); \\ x &= o, dd; n = 1, 2, \end{aligned} \quad (30)$$

where $v_o(t)$ is the output voltage, $v_{dd}(t)$ is the supply voltage, and \mathbf{D} represents the dependence of sub-model current on the previous time instances of the signals, $i_{x,n}$, v_o and v_{dd} . Sub-model functions can be extracted using the data from measurements or transistor-level simulations. In Equation 28 and Equation 29, $w_{x,n}(t)$ are the weighting functions that help the transitions of sub-models from one state to the other, and particularly $w_{dd,3}(t)$ represents the crowbar supply current component during logic switching. The occurrence of events relating to up and down transitions is controlled by the driver input signal. The extraction of weighting functions can be done using inversion of Equation 28 and Equation 29 for two different loads Z_1 and Z_2 connected to the output port when the driver is performing up and down transitions, as discussed in [37], where for the output port:

$$\begin{bmatrix} i_{o,1}^{Z_1}(t) & i_{o,2}^{Z_1}(t) \\ i_{o,1}^{Z_2}(t) & i_{o,2}^{Z_2}(t) \end{bmatrix} \begin{bmatrix} w_{o,1}(t) \\ w_{o,2}(t) \end{bmatrix} = \begin{bmatrix} i_o^{Z_1}(t) \\ i_o^{Z_2}(t) \end{bmatrix}, \quad (31)$$

and for the supply port:

$$\begin{bmatrix} i_{dd,1}^{Z_1}(t) & i_{dd,2}^{Z_1}(t) & 1 \\ i_{dd,1}^{Z_2}(t) & i_{dd,2}^{Z_2}(t) & 1 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} w_{dd,1}(t) \\ w_{dd,2}(t) \\ w_{dd,3}(t) \end{bmatrix} = \begin{bmatrix} i_{dd}^{Z_1}(t) \\ i_{dd}^{Z_2}(t) \\ 1 \end{bmatrix}. \quad (32)$$

Once the weighting functions are obtained, the signals representing the input bit pattern are processed by concatenation and scaling of the sub-model currents at the output and the supply ports.

3.3 Pre-emphasis Issue in Modeling

Being able to model pre-emphasis drivers is important for the simulation of digital systems. Pre-emphasis drivers are useful in reducing ISI in high-speed links by adding extra high-frequency components to the initial signal to compensate for lossy interconnects. Pre-emphasis drivers usually incorporate two additional voltage levels: strong high and strong low, in addition to the normal high and normal low voltage levels. Pre-emphasis comes into effect during a short period when the input signal transitions from one logic state to the other. When the input logic state switches, the output signal is first boosted to a strong high (for low to high transition) or strong low (for high to low transition) voltage level and stays at this level for a short period. When the input logic state continues being the same, the output signal returns to the normal high or normal low voltage level. A graphical illustration is shown in Figure 34, where an example of the output waveform of a pre-emphasis driver is plotted.

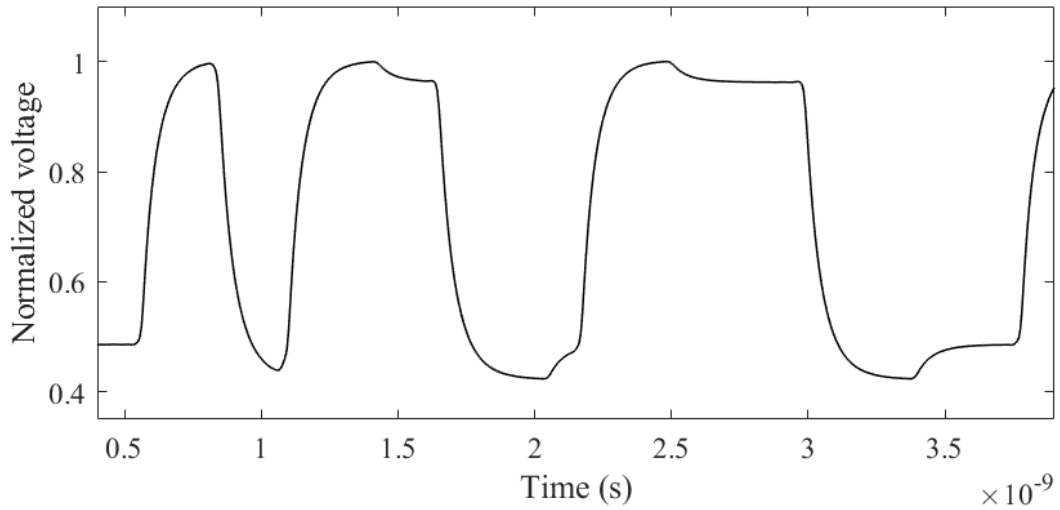


Figure 34 – Example output voltage waveform of a pre-emphasis driver.

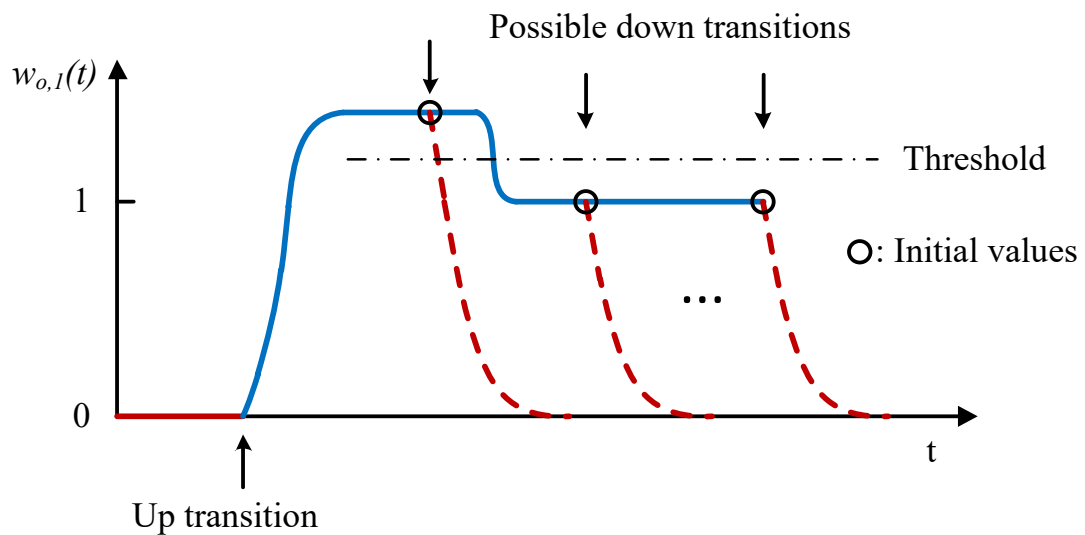


Figure 35 – Example weighting function $w_{o,1}(t)$ for a pre-emphasis driver with possible down transitions (dashed lines) of different timings.

Using the above-mentioned modeling approaches, the timing signals are concatenated based on the successive input state transitions. These implementations assume state transitions are spaced sufficiently in time so that every new state transition only appears after the I/O driver has reached its steady state. This condition, however, is not satisfied when switched input logic states are shorter than the pre-emphasis duration,

due to the presence of multiple high and low levels. A graphical illustration of this modeling issue is shown in Figure 35, where the weighting function $w_{o,1}(t)$ is plotted as an example. After an up transition, the weighting function $w_{o,1}(t)$ first enters the strong high state. The following down transition may start during the strong high state or the normal high steady state, since the pre-emphasis effect can last longer than the bit duration. As a result, the driver is not guaranteed to be in its steady state at the normal high or low level when the new transition appears. As an example, Figure 36 shows a graphical illustration of the concatenated timing signal, where the switching timing signal is incomplete when the switched input logic states are shorter than the pre-emphasis duration and thus the models fail to produce accurate results.

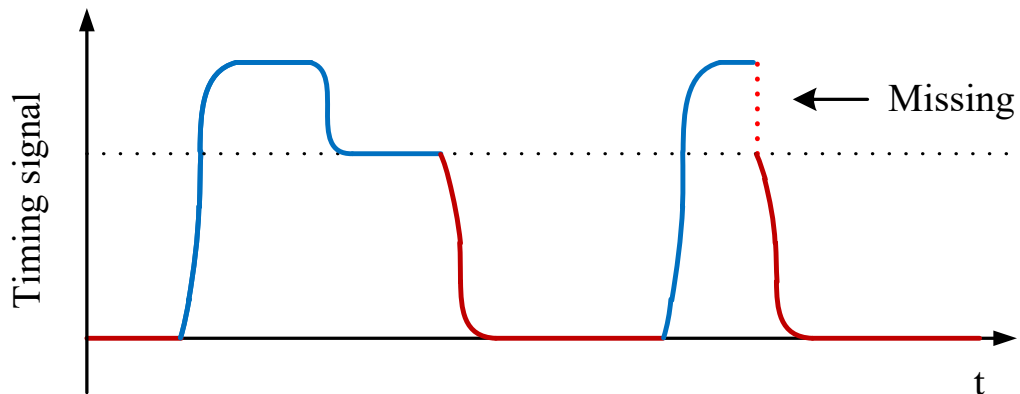


Figure 36 – Concatenation of steady-state switching timing signals for pre-emphasis when the switched input logic states are shorter than the pre-emphasis duration.

Based on the above observations and analysis, it can be concluded that the modeling methodologies based on concatenation can suffer from inaccuracy and convergence issues when modeling pre-emphasis drivers, which is mainly due to the single fixed-pattern weighting coefficients or functions adopted. In the following section, we present a black-box behavioral model formulation and extraction procedure for SI and

PI analysis, which does not rely on fixed-pattern timing signals but rather uses parametric state-aware weighting functions, and is therefore capable of correctly accounting for successive input transitions with shorter time separations than pre-emphasis durations. Using the proposed model, the modeling of tunable drivers with control parameters is addressed as well.

3.4 Modeling Method

3.4.1 Model Formulation with State-Aware Weighting Functions

As concluded above, for pre-emphasis drivers the main issue with previous modeling techniques is caused by the limited capability of using single fixed-pattern timing signals for capturing the nonlinear dynamics of the transitions between driver states with shorter logic switch separations. Therefore, in the proposed model, based on the model structure in Section 3.2, the parametric state-aware weighting functions $w_{x,n}(t)$ are formulated as:

$$w_{x,n}^m(t) = w_{x,n}^m(s_x^{m,state}, v_{dd}(t), t_{tran}), \quad (33)$$

where

$$\begin{cases} n = 1,2 \text{ for } x = o \\ n = 1,2,3 \text{ for } x = dd \end{cases} \quad (34)$$

$$m = up, down,$$

m denotes the transition direction, t_{tran} is the time since the transition event starts, and $s_x^{m,state}$ is the state variable of the weighting function $w_{x,n}^m(t)$, which represents the pre-

emphasis state of the driver (strong or normal) at the time point when a certain transition happens (this is also the time point at the end of the precedent transition). As an example in Figure 35, for transitions occurring during different states, the corresponding switching weighting functions have different magnitudes and shapes. Therefore, the state variable needs to uniquely indicate the driver state, which can be used by the weighting functions to generate the transition weighting signals with proper shapes. On the other hand, as can be seen from Figure 35, when logic switching starts during different levels, the transitions of the weighting function start from different initial values $w_{x,init}^m$, which are also the values of the weighting function at the end of the previous logic event. Hence, the value of the weighting function at the logic switching point can be used as the state variable $S_x^{m,state}$.

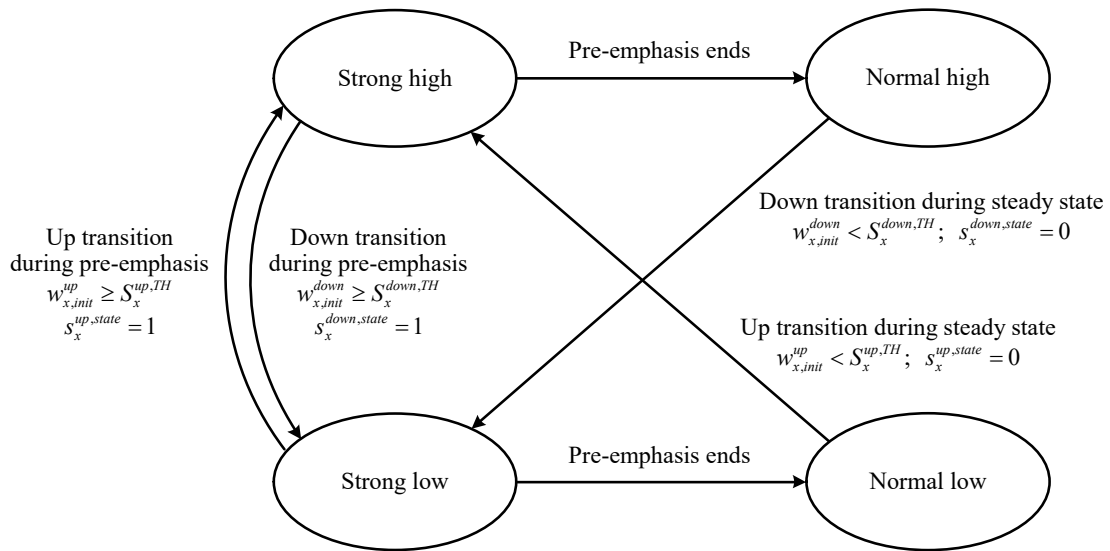


Figure 37 – FSM graph for the proposed modeling method with state-aware weighting functions.

Another important observation is that the strong level region and the normal steady-state level region of the weighting function curves are both flat, and the transition

time from strong to normal level is very short (negligible). This allows us to classify the transitions into two scenarios (i.e., transition during pre-emphasis and transition during steady-state) by comparing the initial value with a threshold. The average of the initial values of the two scenarios can be used as the threshold value $S_x^{m,TH}$, as shown in Figure 35. In this way, the state variable reduces to a binary-valued variable:

$$S_x^{m,state} = \begin{cases} 1, & \text{if } w_{x,init}^m \geq S_x^{m,TH} \\ 0, & \text{if } w_{x,init}^m < S_x^{m,TH} \end{cases} \quad (35)$$

Based on the new formulations in Equation 33 – Equation 35, the weighting functions are no longer single fixed-pattern timing signals. The state variable has a control of the shapes of the weighting functions according to the driver state. The resulting driver model can be implemented using a FSM graph, as shown in Figure 37.

3.4.2 Model Representations

There exist several candidates for the selection of the model representations, including piecewise-linear table-formatted models, and parametric models such as artificial neural networks. Considering the rich nonlinear dynamic characteristics and memory effects exhibited in the driver circuit and the multivariate dependency of the tunable driver behavior on the control parameters (discussed below), neural networks are adopted as the model representation for this work. In the past, many techniques have been proposed to develop behavioral models for nonlinear circuits using neural networks, showing the capability of capturing the nonlinearities accurately [16]-[19], [22], [55]-[57].

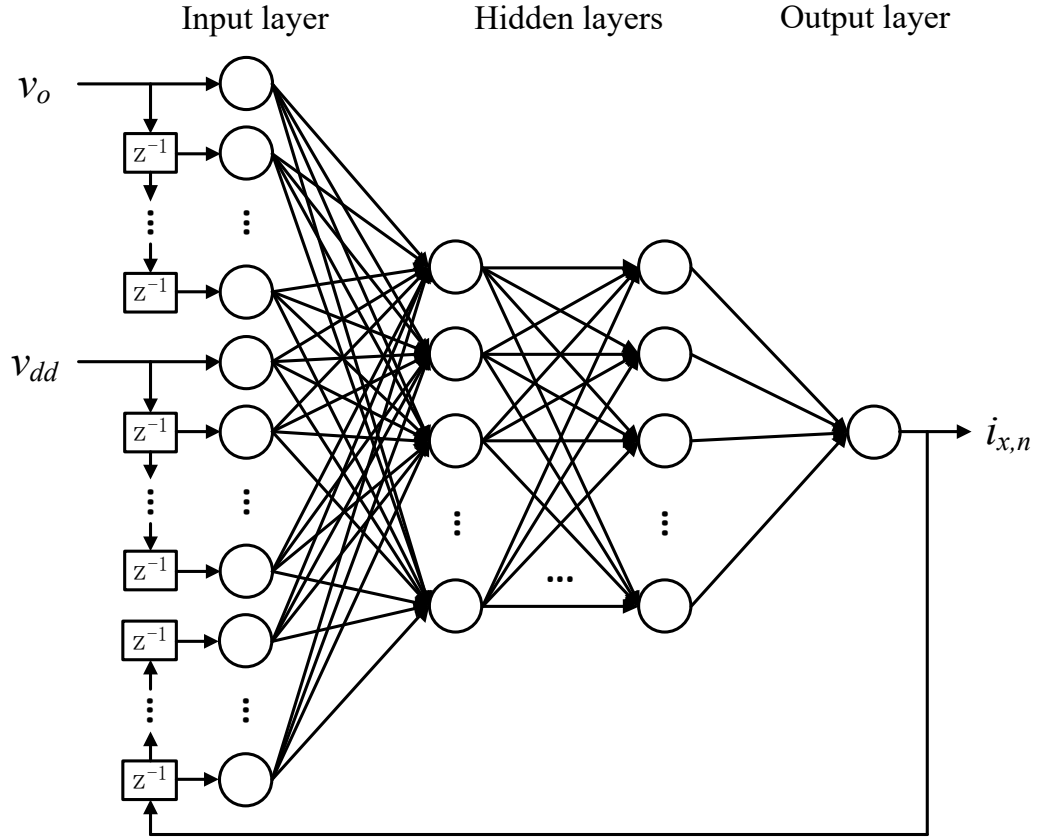


Figure 38 – RNN structure.

Specifically, for the current sub-models of the ports, RNNs are preferred to capture the nonlinear memory effects, as they include feedback paths allowing for better emulation of the interaction between the output and input samples of the networks. The structure of an RNN including input, hidden and output layers is shown in Figure 38. When previous time instances of the ports' voltage and current are considered, the sub-models in Equation 30 can be learnt as RNN models:

$$i_{x,n}(t) = i_{x,n}^{RNN} \left(\begin{array}{c} v_o(t), v_o(t-h), \dots, v_o(t-rh), \\ v_{dd}(t), v_{dd}(t-h), \dots, v_{dd}(t-rh), \\ i_{x,n}(t-h), \dots, i_{x,n}(t-rh) \end{array} \right), \quad (36)$$

where h is the RNN sampling time step, and r is referred to as the dynamic order of the model, which indicates the number of previous samples to be considered as represented by \mathbf{D} in Equation 30 which usually has a value of 1 or 2 for typical drivers. An example is shown in Figure 39 where piecewise-linear voltage sources with small amplitude noise voltages are connected at the driver's output port and power supply port, and the driver input is set high. As shown in Figure 39 (b), when two previous time instances are included, the output sub-model current values learnt by the RNN match accurately with the transistor-level simulation results. For supply current sub-models, similar results can be achieved using RNNs.

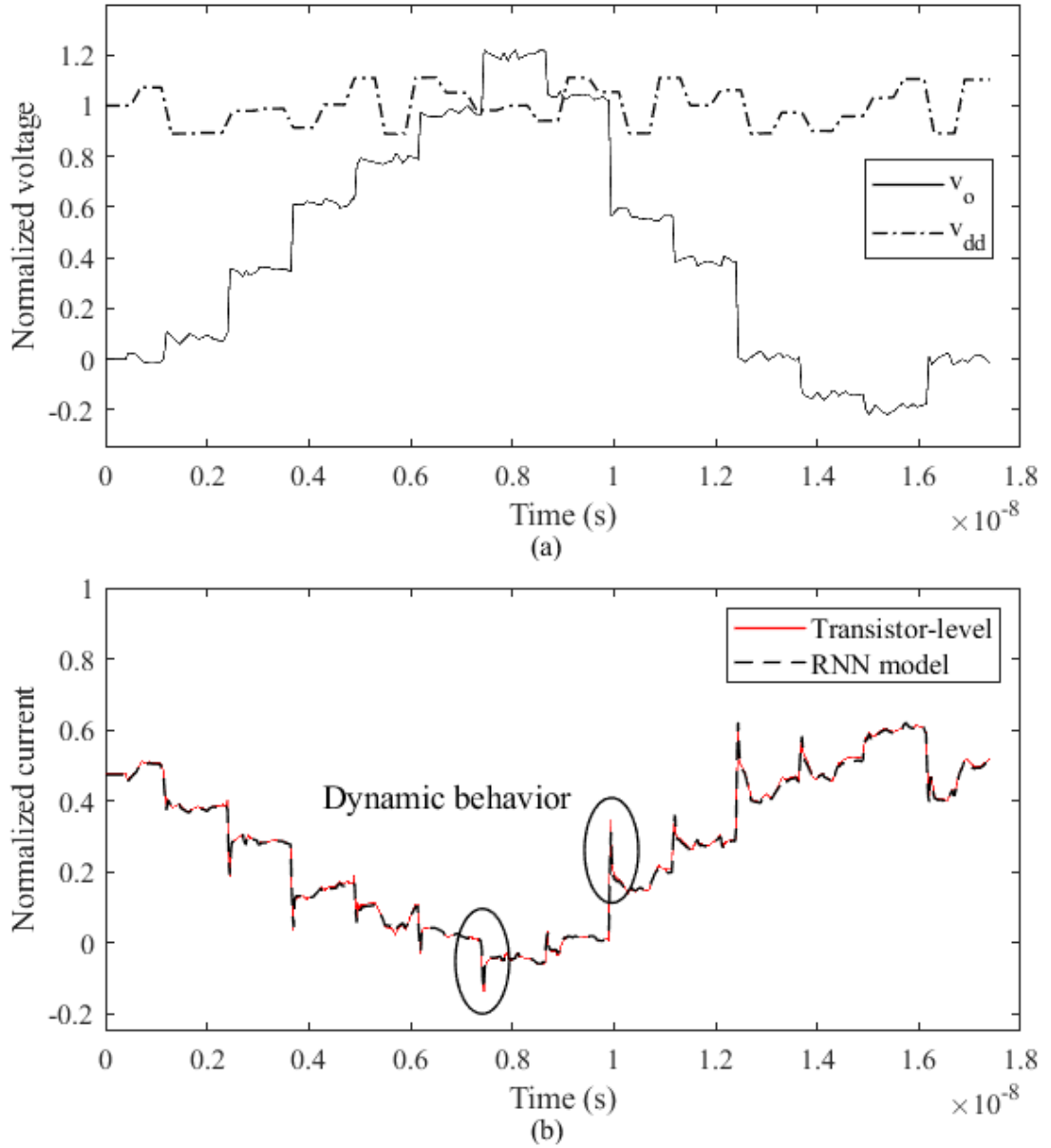


Figure 39 – (a) Piecewise-linear voltage waveform connected at the driver output. (b) Output current from a transistor-level model (solid line) and from RNN model (dashed line) for input high.

On the other hand, for the state-aware weighting functions in Equation 33, since $w_{x,n}^m(t)$ does not rely on the previous time instances, there are no feedback paths required for the interaction between the input and output samples. Therefore, the use of RNNs is not necessary. In this case, FFNNs can provide for effective and flexible modeling

capability in capturing the weighting functions. The structure of an FFNN including input, hidden and output layers is shown in Figure 40. To capture the nonlinear pre-emphasis behavior and the influence of the supply voltage, the weighting functions can be learnt as FFNN models:

$$w_{x,n}^m(t) = w_{x,n}^{m,FFNN}(s_x^{m,state}, v_{dd}(t), t_{tran}). \quad (37)$$

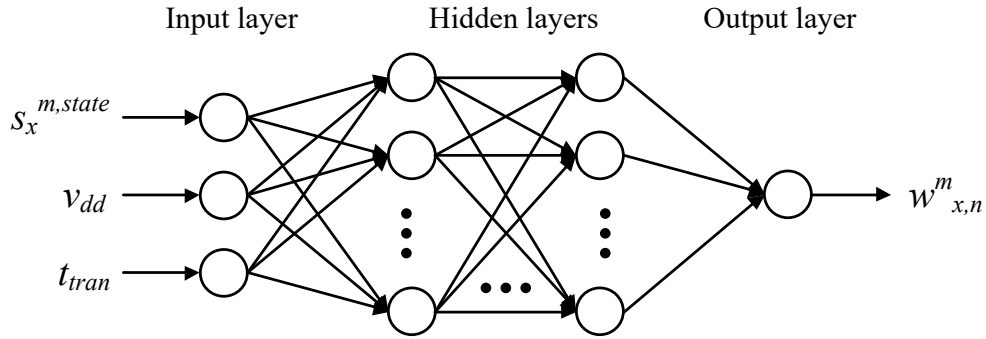


Figure 40 – FFNN structure.

3.4.3 Modeling Flow

The proposed model can be obtained by carrying out the following key steps.

1) *Dynamic sub-model characterization*: In this step, the driver output port and supply port are driven properly to obtain the transient voltage and current signals that contain the information on the nonlinearities and dynamics of the ports. Figure 41 (a) shows the setup required to collect the data for sub-model characterization. Piecewise-linear voltage sources are connected at the driver output and supply ports at the same time to generate the excitation signals. Such driving waveforms can be multilevel signals containing different slopes and different transition heights with small amplitude noise

voltages added, as shown in Figure 39 (a) [35], [37]. The piecewise-linear excitation can mainly provide two types of information: multiple voltage levels capture DC and low-frequency I-V relation of the driver's ports; Transition edges capture the fast changing dynamic behavior. The small amplitude noise voltages mimic the small voltage fluctuation that can be encountered in the simulations. For good characterization, 5-10 randomly generated voltage levels can be used for the voltage waveform connected at the output port, and the slopes of the transition edges should fall in the slope range of the switching output waveforms of the driver in the rising and falling direction. At the same time, the power supply port characterization waveform contains random voltage levels covering the target V_{dd} variation range (10% of the nominal V_{dd} value in this work), with the transition times similar to the rise and fall times. It is important to note that these waveforms need to sufficiently cover the range of operating voltages including the pre-emphasis effects. Transient analysis is performed for driver input high and low respectively to obtain the corresponding current signals at the output and supply ports. Once the data are collected, RNN models are trained to learn the sub-models $i_{x,n}^{RNN}$.

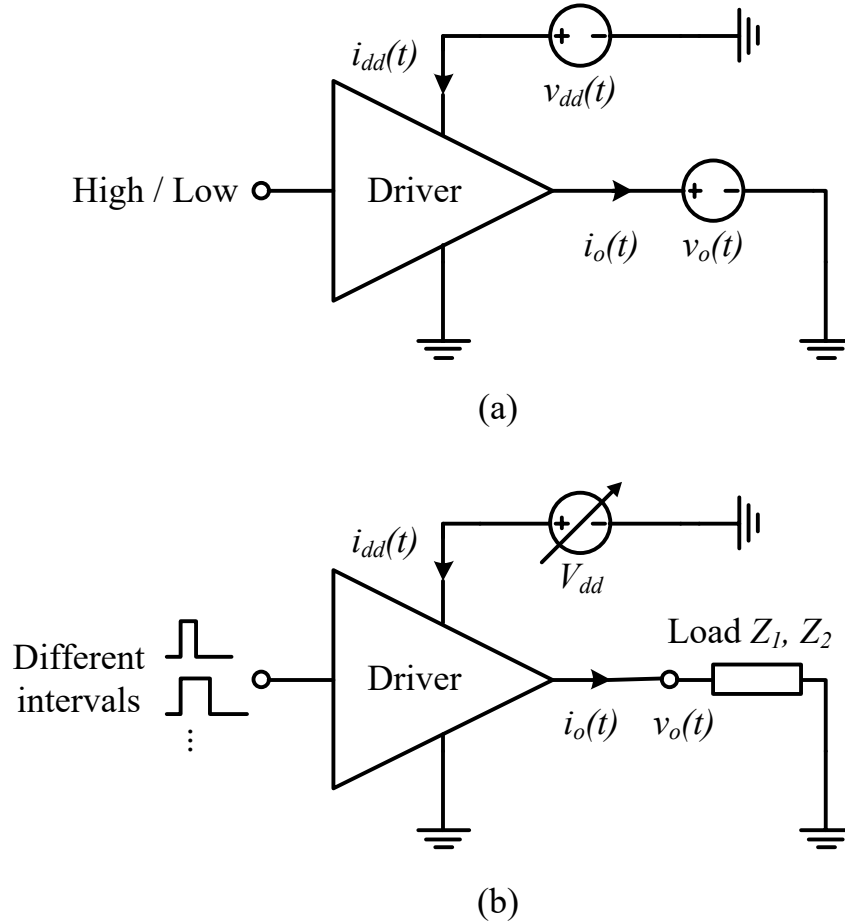


Figure 41 – Test-benches used for the collection of driver responses required for the model generation. (a) Transient analysis for the extraction of nonlinear dynamic sub-models. (b) Transient analysis for the extraction of state-aware weighting functions using input transitions with different intervals.

2) *State-aware weighting function extraction:* In this step, the driver input port is driven by digital waveforms to obtain the transient voltage and current signals at the output and supply ports, which carry information on the input associated switching characteristics under pre-emphasis conditions. Figure 41 (b) shows the setup required to collect the data for weighting function extraction. The driver output port is connected to a load Z_i . Using matrix inversion in Equation 31 and Equation 32, weighting functions are extracted using two different loads Z_1 and Z_2 . A good combination is a resistor for load

Z_1 and the series connection of a resistor and a V_{dd} voltage source for load Z_2 [37]. For each input pattern, this transient analysis is performed for both Z_1 and Z_2 . Also, in order to capture the influence of supply voltage variation on the transition and the pre-emphasis behavior, this extraction procedure is performed for multiple V_{dd} values sampled within the maximum dynamic operation range of the supply voltage. In this work, 7 uniformly sampled power supply voltage levels within $\pm 10\%$ nominal V_{dd} value are used. The range of V_{dd} may be changed depending on the amount of power supply variation required to be modeled, and the sampled levels can be adjusted accordingly. It is important to note that the input digital waveforms used here are not single fixed bit patterns. As discussed previously, due to the pre-emphasis effect, transitions occurring in different pre-emphasis states start with different initial values, resulting in different weighting function shapes. Therefore, the input driving bit patterns consist of two different transition intervals, such that one of the intervals is shorter than the pre-emphasis duration to start the transition during pre-emphasis and the other interval is longer than the pre-emphasis duration to start the transition during steady state. To better illustrate this, an example for capturing high-to-low transitions is shown in Figure 42, where shorter and longer switching patterns are used as the input. The same approach is applied to low-to-high transitions as well. It is worth mentioning that the extraction window length should be large enough for the driver to reach its steady state (normal high or low). After the driver has reached the steady states, the values of the weighting functions will remain the same. When the weighting functions are extracted, their initial values at the logic switching point are recorded, and the average of the two initial values are used as the threshold value $S_x^{m,TH}$.

Once the data are collected, FFNN models are trained to learn the state-aware weighting functions $w_{x,n}^{m,FFNN}$.

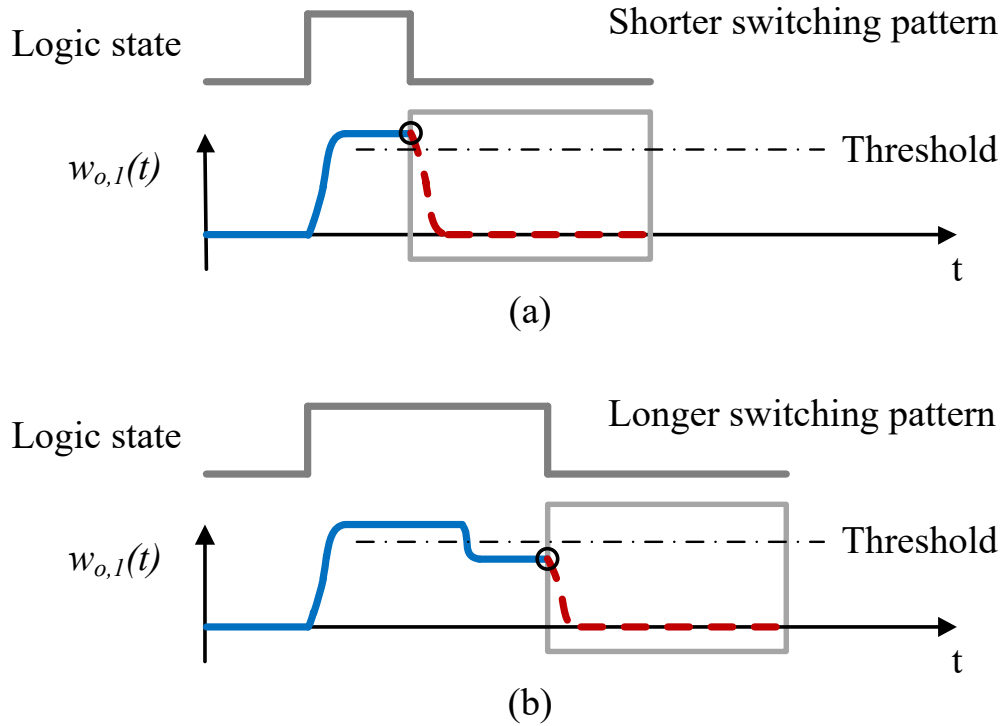


Figure 42 – Example input waveforms with different intervals for the extraction of weighting functions for high-to-low transitions (dashed red down transition curves) with pre-emphasis effect.

More voltage levels in the sub-model characterization waveforms will result in better accuracy of the trained model in capturing DC and low-frequency behavior, and adequate transition edges and small amplitude noise can improve the accuracy in capturing high-frequency component. More sampled V_{dd} levels for weighting function extraction will result in better accuracy in capturing the influence of power supply voltage on the timing behavior and pre-emphasis. There is a trade-off between the characterization accuracy and the amount of data. The example waveforms for sub-models characterization, and 7 V_{dd} levels within $\pm 10\%$ nominal value for weighting

functions extraction used in this work are good balance. It's worth mentioning that when the target voltage range changes, re-characterization with the corresponding voltage range will be needed.

3.4.4 Tunable Driver with Control Parameters

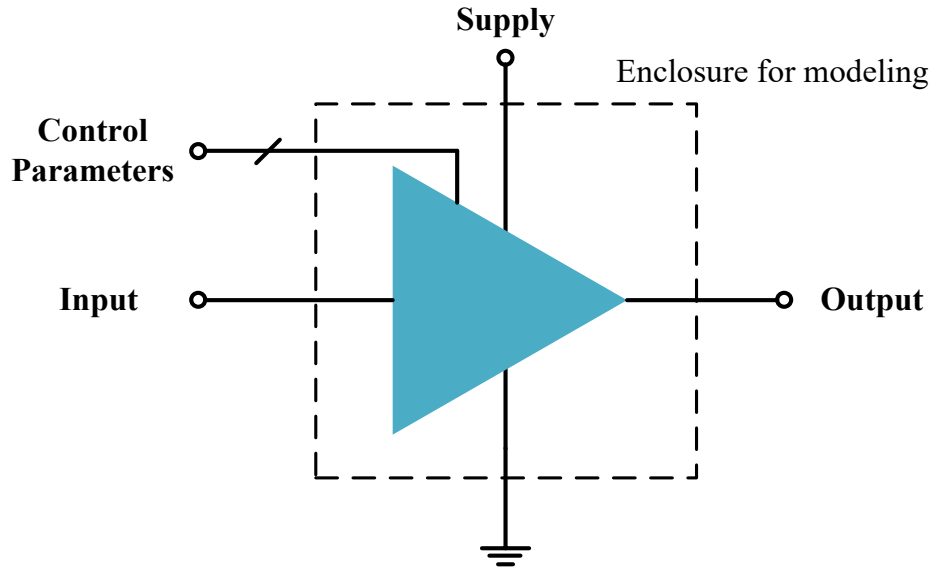


Figure 43 – Tunable driver with control parameters.

Table 1 – Functions and value options of the driver control parameters.

Parameter	Function	Values
$R_{on.pd}$	Control the turn-on impedance of the pull-down device	34, 40, 48, 60, 80, 120
$R_{on.pu}$	Control the turn-on impedance of the pull-up device	34, 40, 48, 60, 80, 120
EQ	Control the rising and falling slew rate and pre-emphasis strength (larger number means steeper edge and stronger pre-emphasis)	0, 1, ..., 7
VOH	Control the output swing magnitude	0.5, 0.6

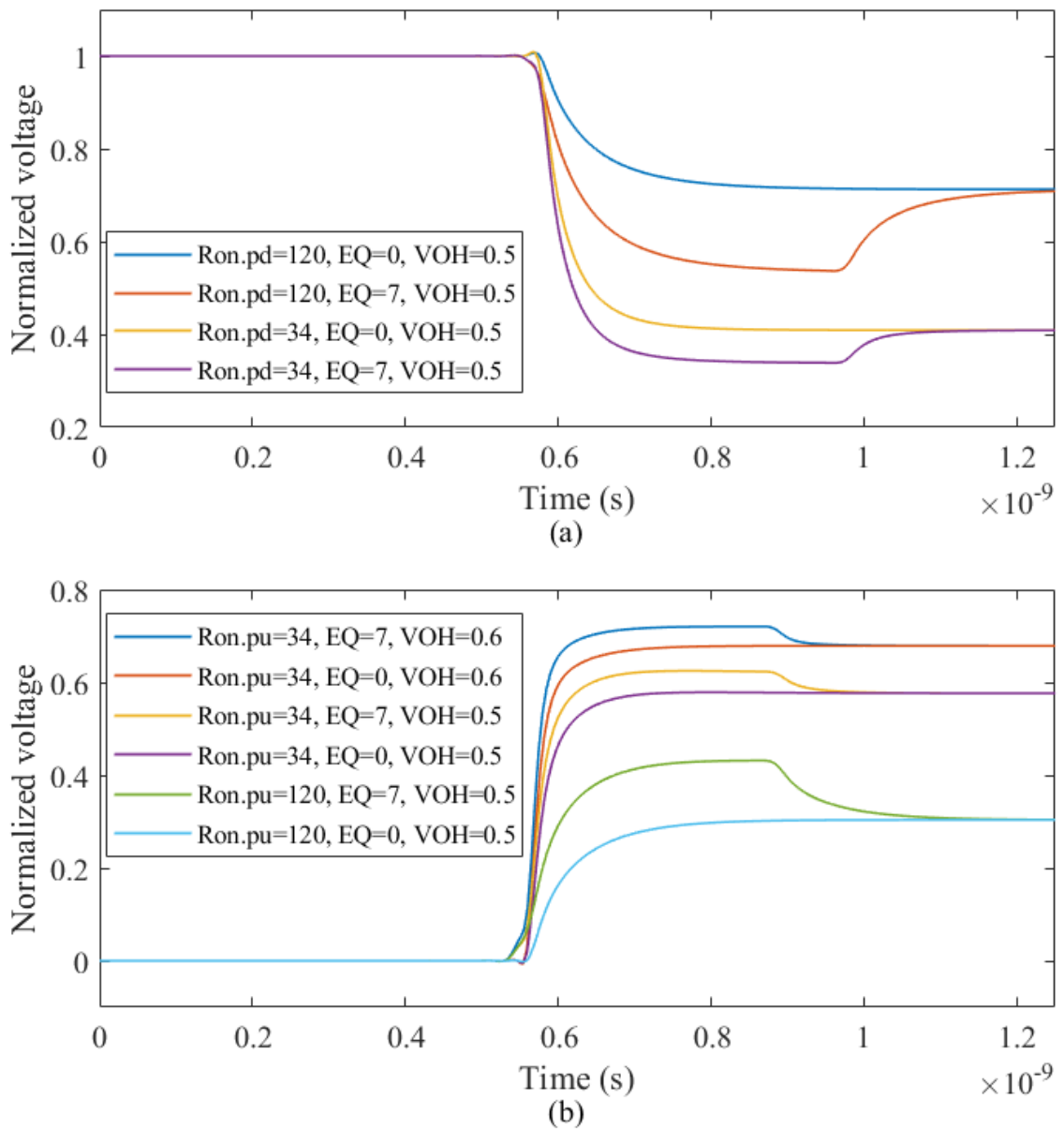


Figure 44 – (a) Output waveforms of down transitions for different $R_{on.pd}$ and EQ settings, with $R_{on.pu} = 34$. (b) Output waveforms of up transitions for different $R_{on.pu}$, EQ and V_{OH} settings, with $R_{on.pd} = 34$.

In many state-of-the-art driver designs, behavior of the driver is no longer confined to a fixed pattern, but instead is regulated using control parameters. Figure 43 shows a scenario where, besides the conventional signal ports, the driver has additional inputs of control parameters. These parameters control driver behavior, such as pull-up

and pull-down characteristics, and pre-emphasis. In this work, an industrial tunable driver circuit (provided by Qualcomm Technologies, Inc.) with four control parameters is considered. The four control parameters are $Ron.pd$, $Ron.pu$, EQ and VOH . Their functions and value options are shown in Table 1, which lead to a total of 576 parameter combinations. A graphical illustration of their effects is shown in Figure 44.

To model tunable pre-emphasis drivers, the method discussed in the previous sections can be extended to incorporate control parameters. In these cases, the current sub-models are not only functions of the current and voltage but also functions of the control parameters, Φ , which can be formulated as:

$$i_{x,n}(t) = i_{x,n}^{RNN} \left(\begin{array}{c} v_o(t), v_o(t-h), \dots, v_o(t-rh), \\ v_{dd}(t), v_{dd}(t-h), \dots, v_{dd}(t-rh), \\ i_{x,n}(t-h), \dots, i_{x,n}(t-rh), \\ \Phi \end{array} \right), \quad (38)$$

and similarly, the state-aware weighting functions can be written as:

$$w_{x,n}^m(t) = w_{x,n}^{m,FFNN} (S_x^{m,state}, v_{dd}(t), t_{tran}, \Phi). \quad (39)$$

Based on Equation 38 and Equation 39, the modeling flow described can be modified to include control parameters.

3.5 Modeling Examples

3.5.1 Pre-emphasis Driver with Power Supply Noise

In this test case, the influence of power supply variation is demonstrated. During weighting function extraction, the procedure is performed for multiple v_{dd} values

sampled within the maximum dynamic operation range in real case simulations. Figure 45 shows an example of a set of extracted output port weighting function $w_{o,0}$ with pre-emphasis for HIGH-to-LOW transitions, with a supply variation of $\pm 10\%$ nominal V_{dd} value. It can be seen that the power supply variation has an influence on both the transition delay and the pre-emphasis behavior.

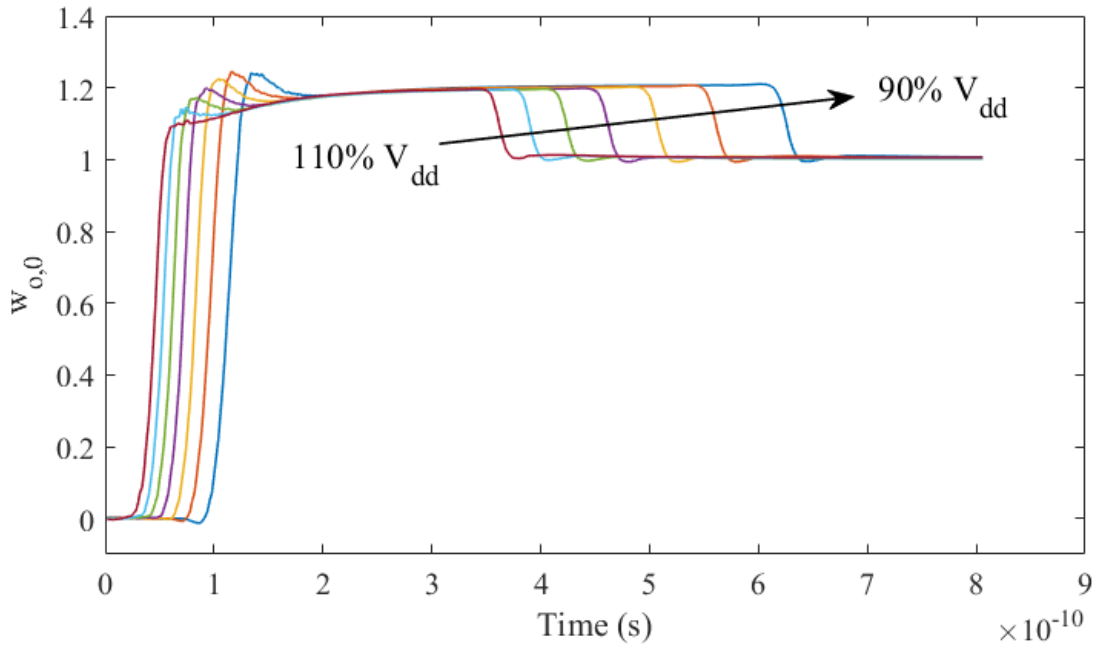


Figure 45 – Extracted output port weighting function $w_{o,0}$ with pre-emphasis for HIGH-to-LOW transitions.

In order to demonstrate the accuracy and efficiency of the proposed model, a modeling example is generated using a state-of-the-art industrial driver circuit with pre-emphasis. The setup for the test cases is shown in Figure 46, where a pre-emphasis driver is connected to a $50\text{-}\Omega$ transmission line with a length of 50 mm. The transmission line is terminated with a load resistor of $60\ \Omega$ and a capacitor of $0.7\ \text{pF}$. An extracted PDN model in form of R, L and C is used to provide non-ideal power supply. In the analyses, the driver is driven by a 500-bit long pseudorandom bit sequence (PRBS) with 268-ps bit

period. The transistor-level SPICE simulation data are used to train the model. After training, the model is implemented in Verilog-A

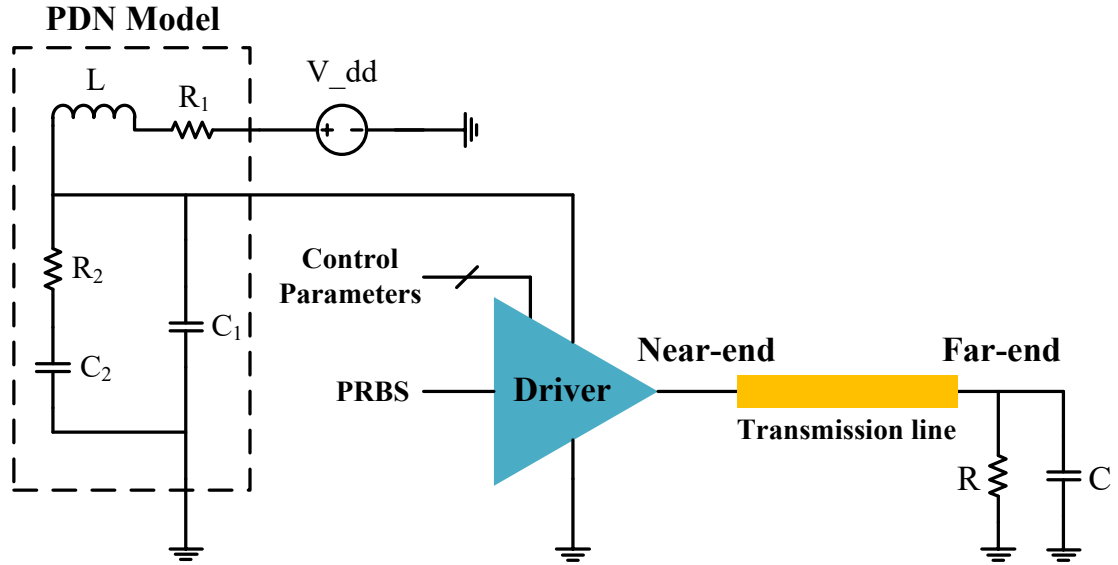


Figure 46 – Simulation setup for model validation.

In the test, the voltage waveforms at the supply port and the far-end of the transmission line are observed. Figure 47 shows the transient simulation results of the behavioral model at the supply port, compared to the responses of the corresponding transistor-level model. It can be seen that the power supply variation is captured accurately, and a FOM value of 98.35 is achieved. The eye diagram at the far-end of the transmission line is shown in Figure 48, where an excellent correlation can be observed with a FOM value of 99.83. The non-negligible influence of the supply variation on the timing behavior of the pre-emphasis driver is demonstrated using an example shown in Figure 49, where the supply variation is not considered for the weighting functions of the behavioral model (static weighting functions), and as a result it can be seen that the timing accuracy degrades significantly with a reduced FOM value of 99.45. In the simulation, the developed behavioral model achieves a speed-up of ~300X.

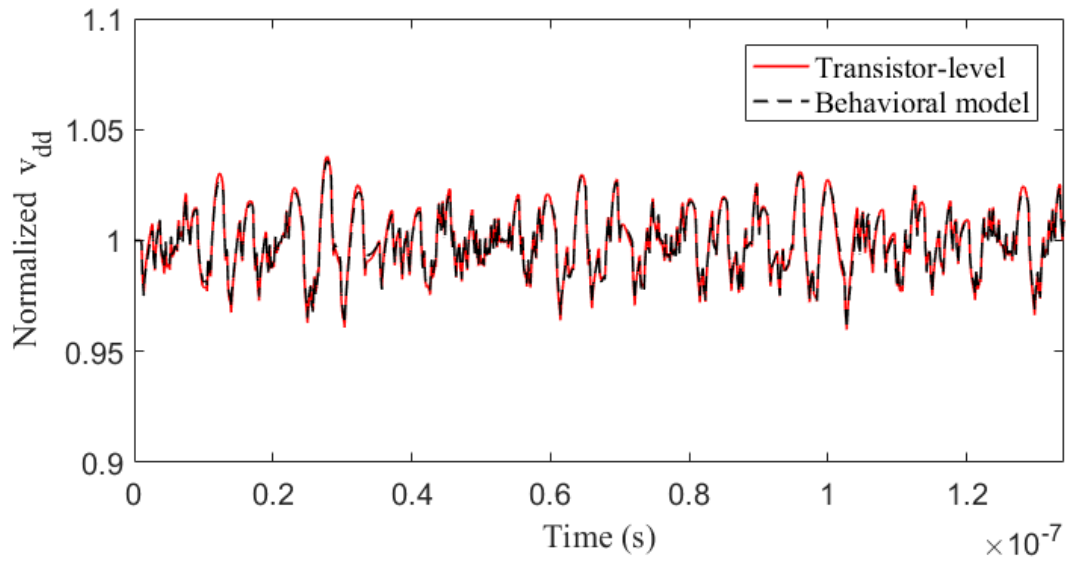


Figure 47 – Voltage waveform at the power supply port from the transistor-level model and from the behavioral model.

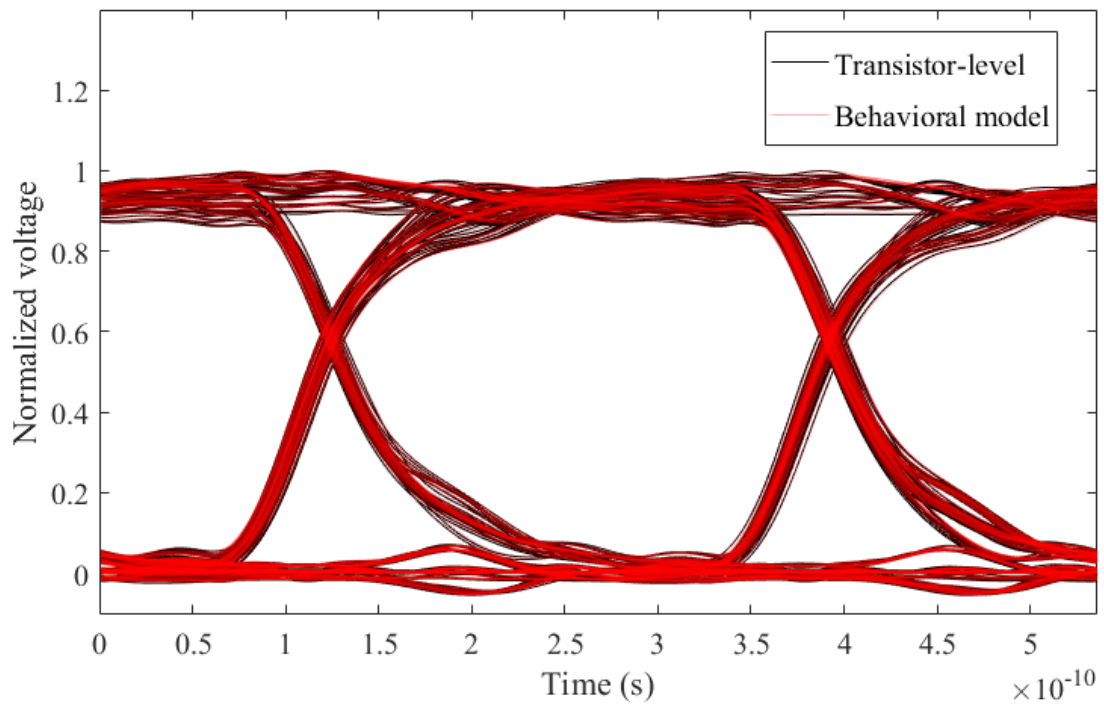


Figure 48 – Eye diagram at the far-end of the transmission line. Power supply variation is considered for the weighting functions of the behavioral model.

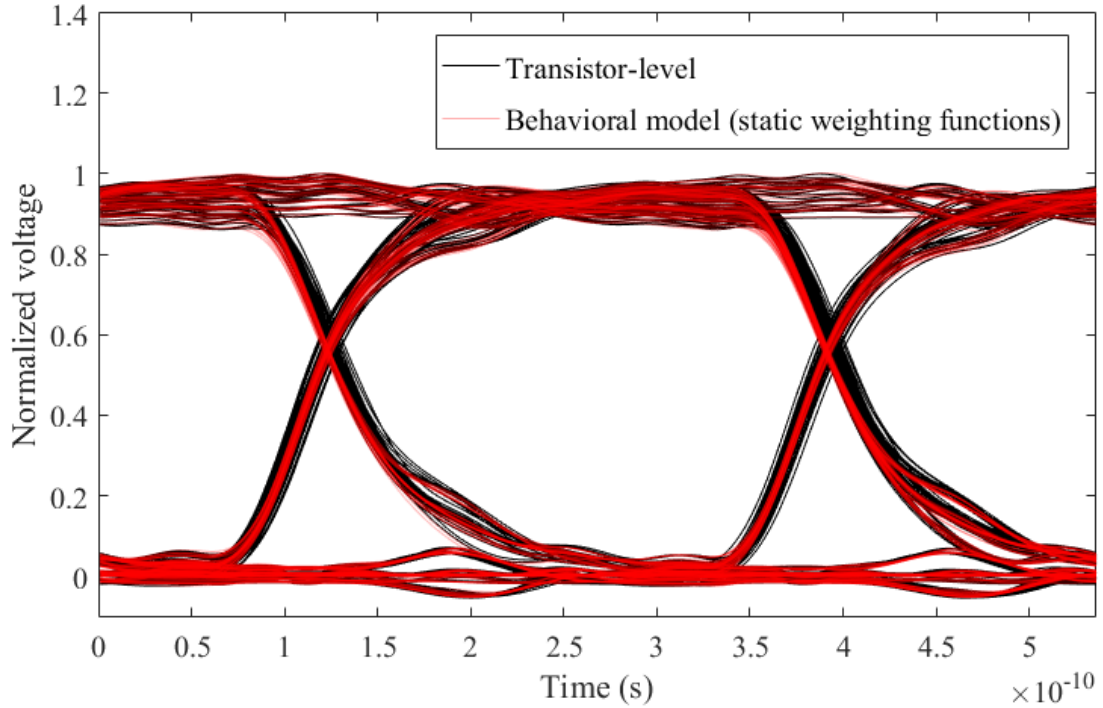


Figure 49 – Eye diagram at the far-end of the transmission line. Power supply variation is not considered for the weighting functions of the behavioral model.

3.5.2 Validation of State-Aware Weighting Functions with Pre-emphasis

In this modeling example, the effectiveness of the proposed model with state-aware weighting functions is validated using the test case in Figure 46. The driver output port is connected to a $50\text{-}\Omega$ transmission line of 20 mm. At the far-end, the transmission line is terminated using a $60\text{-}\Omega$ resistor and a 0.7-pF capacitor in parallel. The driver supply port is connected to an extracted PDN model which is represented using the equivalent resistors, capacitors and inductor to provide non-ideal supply voltage. The input signal is a 500-bit long PRBS with 268-ps bit period and with equal rise and fall time of $t_r = t_f = 10.72$ ps. The driver circuit introduced in the previous section is used with the control parameters set as: $Ron.pd = 34$, $Ron.pu = 60$, $EQ = 4$ and $VOH = 0.5$. In this study, the bit cycle is shorter than the pre-emphasis duration. For the driver sub-models, a

dynamic order of $r = 2$ is used. Each RNN contains a hidden layer consisting of 20 neurons. The RNN sampling time step is 2 ps. The BPTT algorithm with the Levenberg-Marquardt method is used to train the RNN models [47]. For weighting functions, each FFNN contains a hidden layer consisting of 50 neurons. The BP algorithm with the Levenberg-Marquardt method is used to train the FFNN models. For all the neural networks, hyperbolic tangent function $\tanh()$ is chosen to be the nonlinear activation function. The transistor-level simulation data is used as the training data. To demonstrate the effectiveness of the proposed approach, another model is developed without the use of state-aware weighting functions. These models are implemented in Verilog-A and simulated using a circuit simulator. For details of the implementation of RNNs and FFNNs in Verilog-A, refer to APPENDIX A.

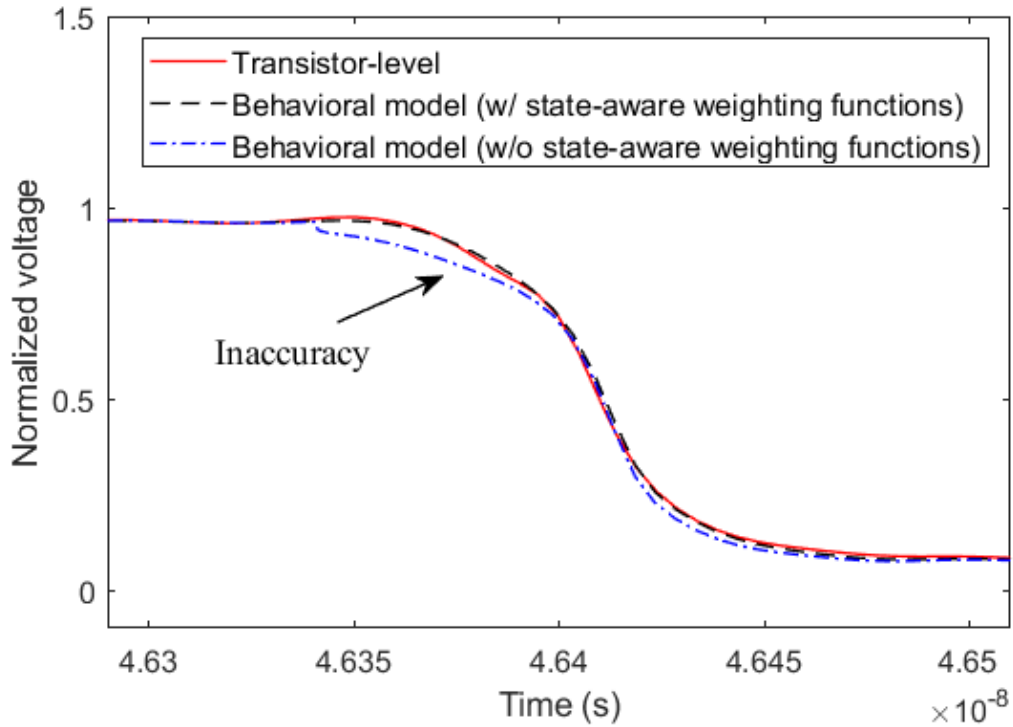


Figure 50 – Simulation waveforms with and without state-aware weighting functions.

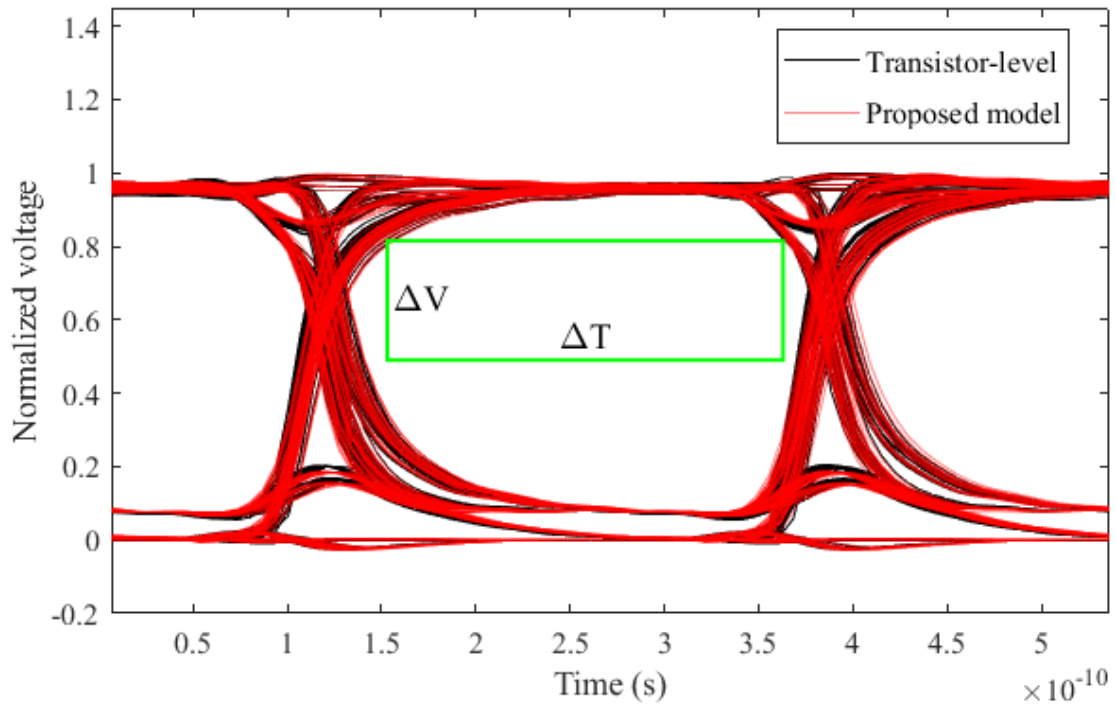


Figure 51 – Eye diagram at the near-end of the transmission line.

Figure 50 shows the voltage waveforms at the near-end of the transmission line. From the figure, mismatch can be observed for the model without state-aware weighting functions, which can result in timing inaccuracy of more than 10 ps (3.7% UI). In contrast, the results are accurate using the proposed model with state-aware weighting functions. Figure 51 shows the eye diagram at the near-end of the transmission line. The eye diagram aperture height ΔV and width ΔT are calculated and the errors are computed comparing to the transistor-level results as shown in Table 2, which further demonstrates the accuracy of the proposed model.

Table 2 – Comparison of the eye diagram apertures of the behavioral models (reference: transistor-level model).

Model	FOM	ΔV (V)	ΔT (ps)	ΔT Error	
				(ps)	(UI)
Transistor-level	-	0.32	207.1	-	-
Behavioral model (state-aware)	99.24	0.32	206.1	1.0	0.37 %
Behavioral model (non-state-aware)	97.78	0.32	197.4	9.7	3.62 %

3.5.3 Tunable Driver Modeling Test Cases

To test the fidelity and simulation speed for tunable driver modeling with control parameters, a more complicated simulation setup under channel reflection and crosstalk scenarios is used as shown in Figure 52. In this simulation, multiple identical drivers are connected to the channels terminated using capacitors of 0.7 pF and resistors in parallel, which bring more significant switching noise at the power supply port and degrade the timing performance of the signals. The tunable driver in the previous section is used here. The supply ports of the drivers are connected to the PDN model as used above. Each driver is driven by a different 900-bit long PRBS.

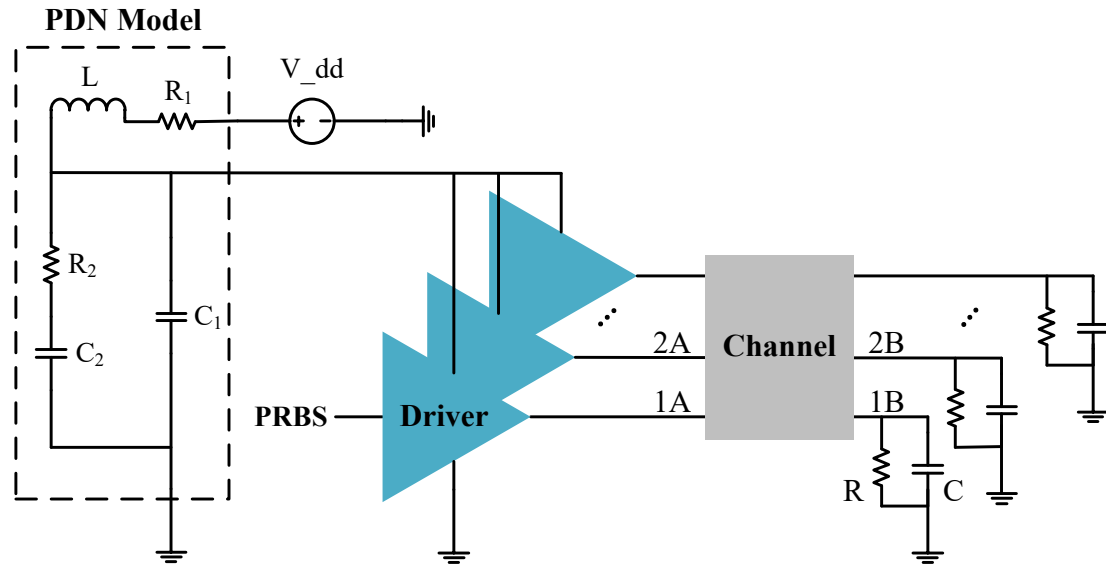


Figure 52 – Simulation setup under channel reflection and crosstalk scenarios.

For validation, test cases are generated using three different settings of the control parameters. These parameter combinations are selected among the combinations with the highest training errors. The corresponding parameter values and loading conditions are listed as follows:

- 1) In case 1, the control parameters of the drivers are set as: $Ron.pd = 34$, $Ron.pu = 34$, $EQ = 0$ and $VOH = 0.6$. The load resistor is 34Ω .
- 2) In case 2, the control parameters of the drivers are set as: $Ron.pd = 120$, $Ron.pu = 60$, $EQ = 7$ and $VOH = 0.6$. The load resistor is 60Ω .
- 3) In case 3, the control parameters of the drivers are set as: $Ron.pd = 34$, $Ron.pu = 34$, $EQ = 5$ and $VOH = 0.6$. The load resistor is 34Ω .

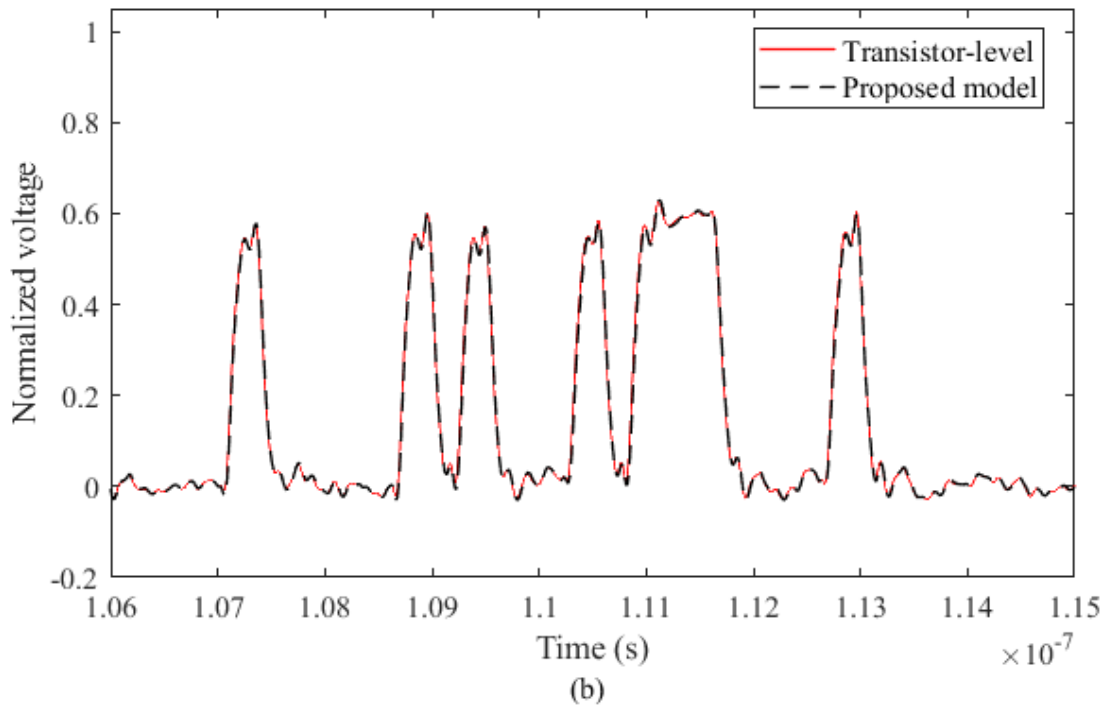
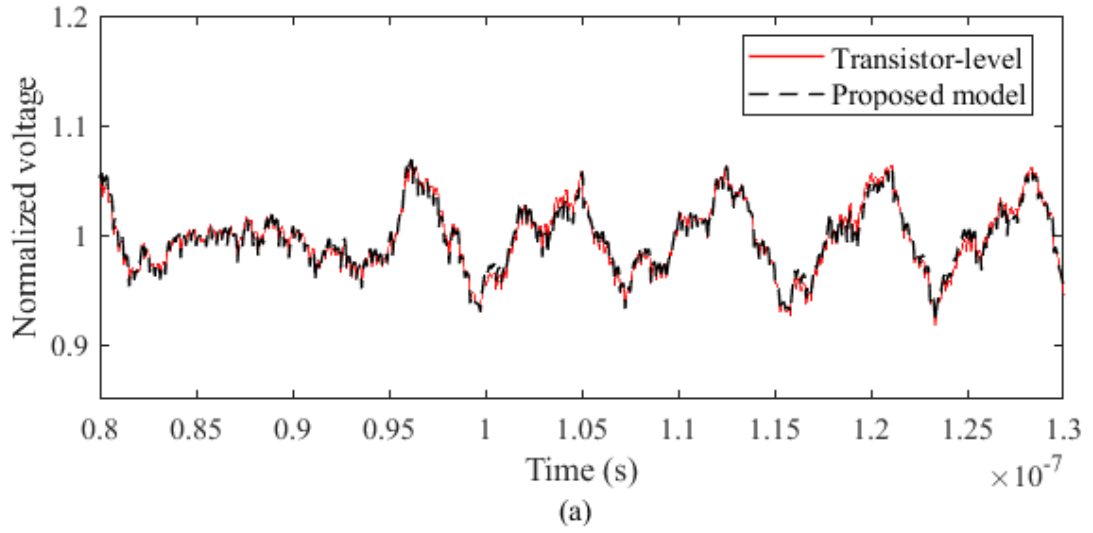


Figure 53 – Voltage waveforms at (a) power supply port and (b) channel port 2B, in test case 1.

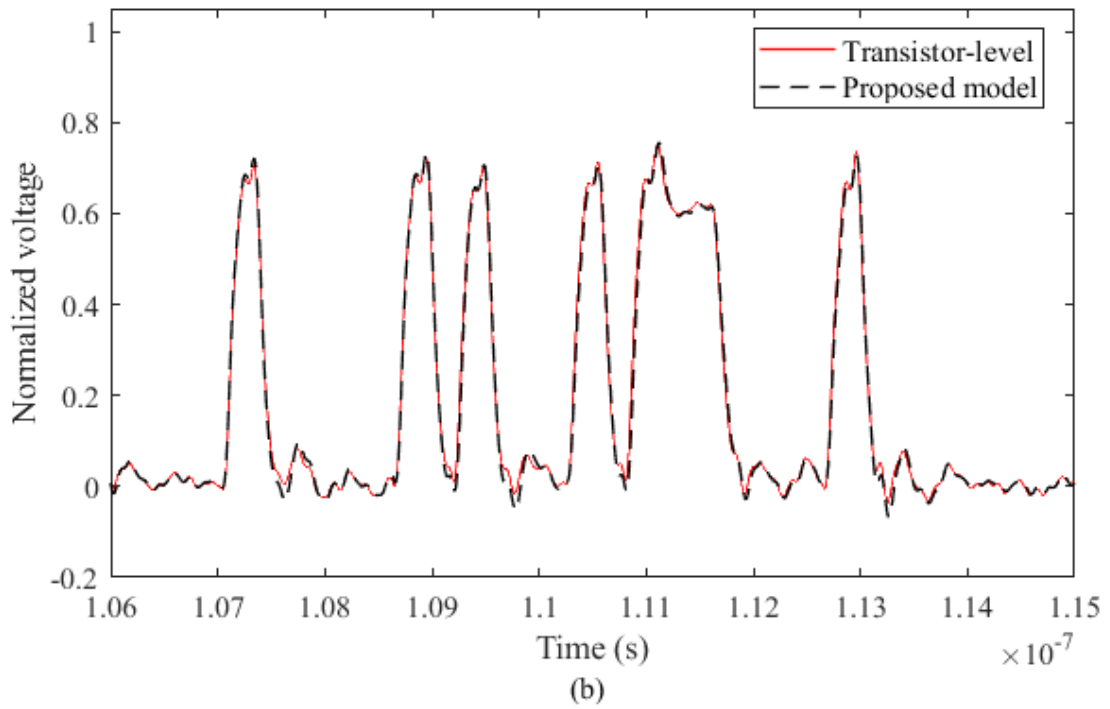
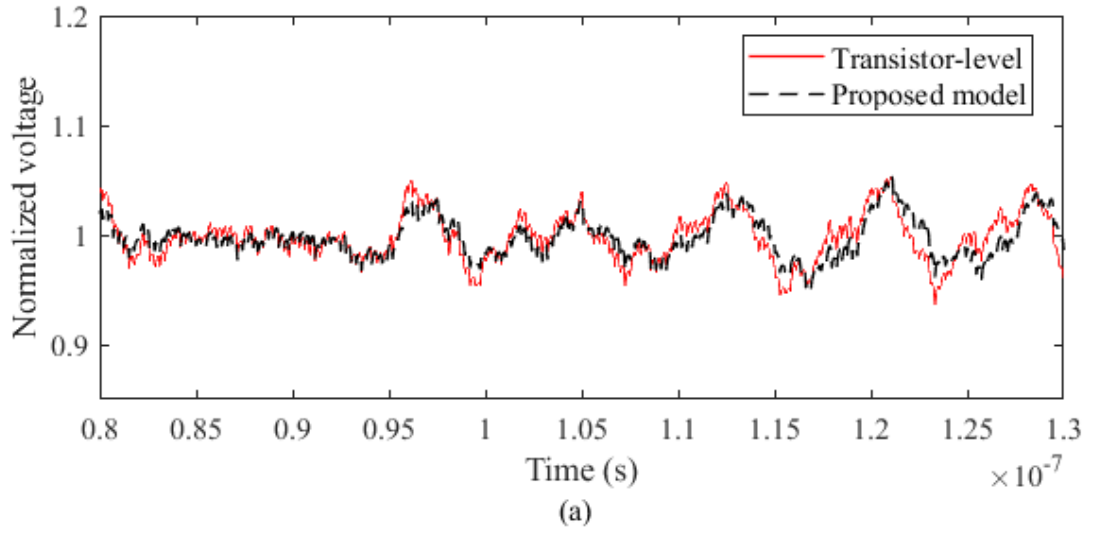


Figure 54 – Voltage waveforms at (a) supply port and (b) channel port 2B, in test case 2.

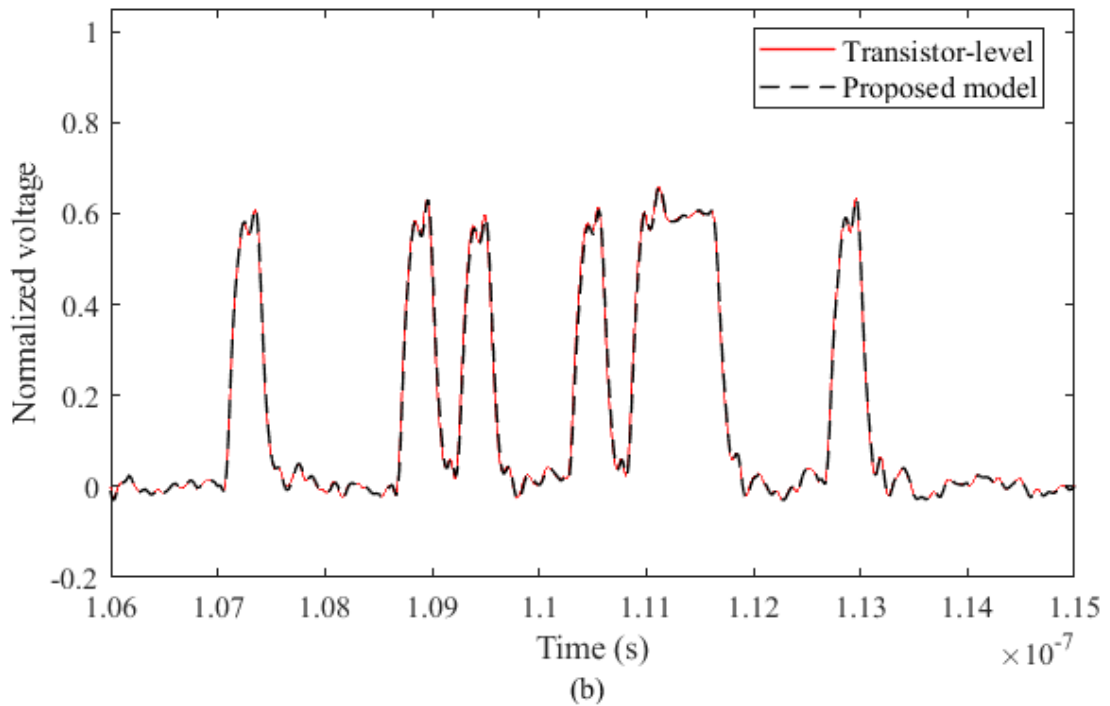
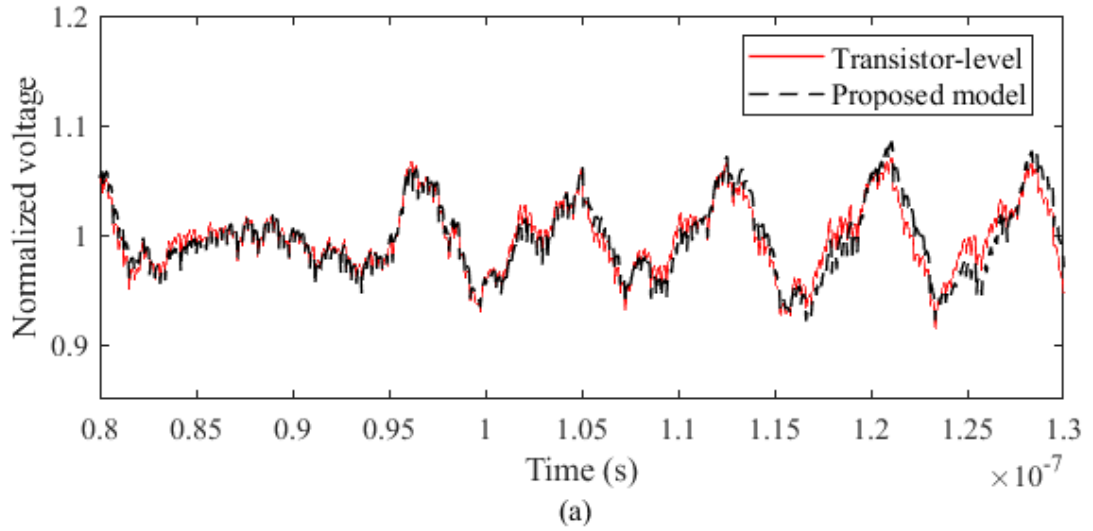


Figure 55 – Voltage waveforms at (a) supply port and (b) channel port 2B, in test case 3.

The voltage waveforms at the supply port and the channel port 2B are plotted. Figure 53, Figure 54 and Figure 55 show the transient simulation results of the behavioral model, compared to the responses of the transistor-level model. The corresponding eye

diagram performance is measured and the aperture height and width are shown in Table 3. In these cases, good agreement can be observed between the behavioral model and the transistor-level model. To assess speed-up, the simulations were done using the same computer. For each test case, the CPU time for the transistor-level model was around 3150 min, whereas the corresponding time for the proposed model was around 9 min. A simulation speed-up of ~350X has been achieved.

For sub-model characterization, 1152 (576 combinations * 2 sub-models) sets of waveform were generated. For weighting function extraction, 32256 (576 combinations * 2 loads * 4 transitions * 7 V_{dd} levels) sets of waveform were generated. With the large amount of training data for tunable driver (5040000 samples for each RNN and 3233664 samples for each FFNN), the training of each RNN and FFNN costs 10 ~ 20 hours using Intel 2.2 GHz Xeon E5-2699 v4 CPU without GPU acceleration (With GPUs the training time can be reduced effectively). The training processes for the output port and power supply port are independent, and can be done in parallel. Considering the long simulation time of the transistor-level model (2.2 days for a test case with tunable drivers), the training time of this one-time model generation process is acceptable, since the developed model can be used repetitively in various simulations. Training strategies such as gradually increasing data size during training and data reduction in preprocessing can further reduce the training time effectively.

Table 3 – Comparison of the eye diagram apertures of the proposed behavioral model for tunable pre-emphasis driver in different test cases (reference: transistor-level model).

Model	Test case	FOM	ΔV (V)	ΔT (ps)	ΔT Error		CPU time (min)
					(ps)	(UI)	
Transistor-level	Case 1	-	0.2	167.3	-	-	~ 3150
	Case 2	-	0.2	157.7			
	Case 3	-	0.2	171.8			
Proposed	Case 1	99.00	0.2	168.4	1.1	0.41 %	~ 9
	Case 2	98.62	0.2	153.4	4.3	1.60 %	
	Case 3	99.06	0.2	170.0	1.8	0.67 %	

3.6 Summary

In this chapter, a technique is presented for the time-domain behavioral modeling of tunable I/O drivers with pre-emphasis including power supply noise. The modeling method using state-aware weighting functions has been proposed to facilitate the concatenation modeling scheme when switched input logic states are shorter than the pre-emphasis duration. FFNNs and RNNs are used to capture the nonlinear dynamic behavior associated with driver state transitions and ports' current-voltage relations. The modeling of tunable driver circuits with control parameters has been demonstrated. The accuracy and efficiency of the proposed model have been validated using practical industrial driver examples.

CHAPTER 4. MODELING OF I/O DRIVERS UNDER OVERCLOCKING CONDITIONS

4.1 Introduction

In high-speed link design, behavioral models of I/O drivers are widely used for faster simulation. As the data rate keeps increasing, being able to accurately simulate drivers under overclocking conditions is becoming an emerging requirement for behavioral driver models.

In the past, different driver models have been developed. In these behavioral models, the I-V relations are captured for HIGH and LOW logic states separately, and timing signals or weighting functions are generated to mimic the switching behavior of the driver circuit from digital to analog signals. Among the existing models, the IBIS model is a popular industry standard behavioral model for driver simulations. However, the IBIS model suffers from significant inaccuracy issue when the driver is overclocked as demonstrated in [40], [58] and [59]. To address the overclocking issues, approaches in [40], [58] and [59] assume a low-pass filter approximation of driver's input stages to generate timing coefficients. However, this idealized scheme can bring in inherent approximation inaccuracy. Furthermore, the fidelity of the model based on this approximation scheme is limited by the data rate. When the data rate is higher than a certain level, a new model must be developed as discussed in [59]. In [60], a simple mechanism is introduced to generate the necessary weighting functions under

overclocking conditions through realignment. However, as shown later in this chapter, this method suffers from accuracy degradation when data rate is high.

In this chapter, a transition-variational behavioral model of I/O drivers is discussed for overclocking simulation. The driver behavior under overclocking conditions is investigated. The corresponding weighting function response surface is presented, which demonstrates the insufficiency of the previous approaches in [34]-[39], [60] that use fixed timing signals extracted under normal operating conditions only. The presented model addresses the modeling of overclocking behavior by using transition-variational weighting functions (TVWFs) along with a transition variable. The weighting coefficients can be well captured for different overclocking scenarios. The corresponding model extraction flow is presented. Using the proposed TVWFs, the driver input signal is processed during run-time by a FSM algorithm which can be implemented in the widely supported hardware description languages such as Verilog-A. Modeling examples using commercial driver circuit show that the proposed model is able to capture the driver behavior accurately under both normal operation and overclocking conditions, and reduce the simulation time. This chapter is focused on the two-port driver modeling for SI analysis but can be extended to a multi-port model including power supply noise.

4.2 Model Overview

The design of I/O driver circuits of CMOS technology is generally based on cascaded structure of multi-stage logic gates. The output stage is the interface between the inner circuit parts (e.g. input stages) and the load outside (e.g. transmission line channels), which generates pull-up and pull-down currents with upper and lower

transistor devices. The switching of the output stage is controlled by the precedent stages based on the input logic patterns. The pull-up and pull-down admittances exhibit nonlinear dynamic behavior, and can be represented as:

$$\begin{cases} i_0(t) = F_0(v_{out}(t)) \\ i_1(t) = F_1(v_{out}(t)) \end{cases} \quad (40)$$

where i_0 and i_1 are the output current sub-models of the driver at LOW ('0') and HIGH ('1') logic states, respectively. The sub-models can be expressed as functions of the output voltage v_{out} . These functions, $F_0(\cdot)$ and $F_1(\cdot)$, can be implemented as look-up tables or captured using spline functions or artificial neural networks. Note that for two-port models, the power supply voltage remains constant, and thus is not included in Equation 40.

At any time point during the simulation, the total output current is a weighted summation of the two sub-models:

$$i_{out}(t) = w_0(t) \times i_0(t) + w_1(t) \times i_1(t), \quad (41)$$

where w_0 and w_1 are the weighting functions that capture the switching dynamics of the input stages. To obtain the weighting functions, the driver output can be connected to two different loads Z_1 and Z_2 while the driver input is fed with LOW-to-HIGH and HIGH-to-LOW switching patterns [36], [37]. The extraction of the weighting coefficients at each time point is done through linear inversion of Equation 41:

$$\begin{bmatrix} i_0^{Z_1}(t) & i_1^{Z_1}(t) \\ i_0^{Z_2}(t) & i_1^{Z_2}(t) \end{bmatrix} \begin{bmatrix} w_0(t) \\ w_1(t) \end{bmatrix} = \begin{bmatrix} i_{out}^{Z_1}(t) \\ i_{out}^{Z_2}(t) \end{bmatrix}. \quad (42)$$

A state-machine algorithm can be implemented to generate adequate weighting coefficients in real-time using the concatenation scheme based on the input switching event occurrence. Once the sub-models and weighting functions are obtained, the driver model is assembled as shown in Figure 56.

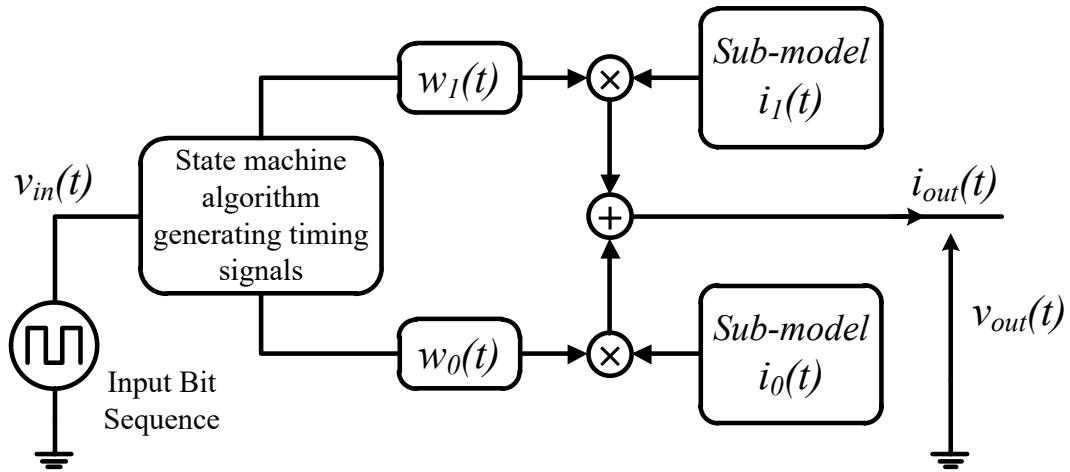


Figure 56 – Behavioral model structure for I/O drivers.

4.3 Overclocking Issue

As the demand for fast data transfer increases, the data rate of high-speed links is becoming higher and higher, which results in a shorter bit duration. On the other hand, after each logic switching event occurs, the I/O driver circuit needs a certain time interval to reach the steady state. A next logic switching event appearing beyond this time interval allows a proper transition on the analog side of the driver. However, when the bit duration is shorter than this interval, the driver does not have enough time to reach steady state, which leads to overclocking. Figure 57 shows an example of the output voltage

waveform simulated under overclocking conditions, where the Micron EDY4016AABG driver is loaded with a 50-Ω resistor. When the driver is overclocked, the rising and falling edges of the output signal are distorted as well as the voltage level. These effects will further degrade the SI performance of the high-speed link such as eye height and width. Therefore, it is beneficial to be able to accurately modeling the overclocking behavior.

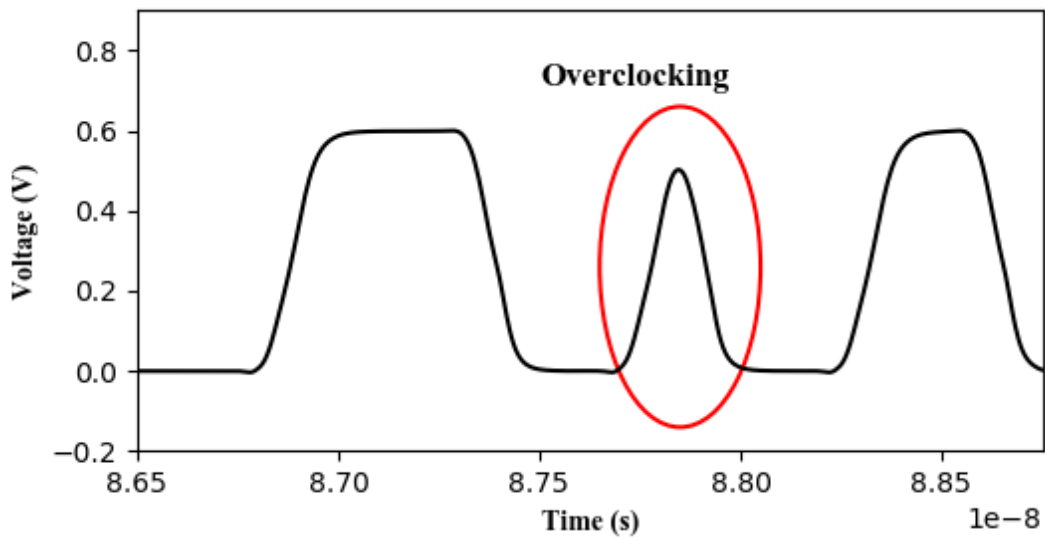


Figure 57 – Example driver output waveform under overclocking conditions.

As illustrated in Equation 41, the switching behavior is captured using weighting functions, and the generation of continuous weighting coefficients relies on the concatenation scheme based on the input signal, as shown in Figure 58. In the previous approaches in [34]-[39], [60], the implementation of the concatenation assumes that the separation of two successive transitions are long enough in time such that posterior switching events only occur when the I/O driver circuit is in steady state. However, under overclocking conditions, this assumption is not satisfied, since transitions can occur

before steady state, as illustrated in Figure 59. Thus, any modeling approaches based on the concatenation of fixed-pattern weighting coefficients would fail to accurately model overlocked drivers.

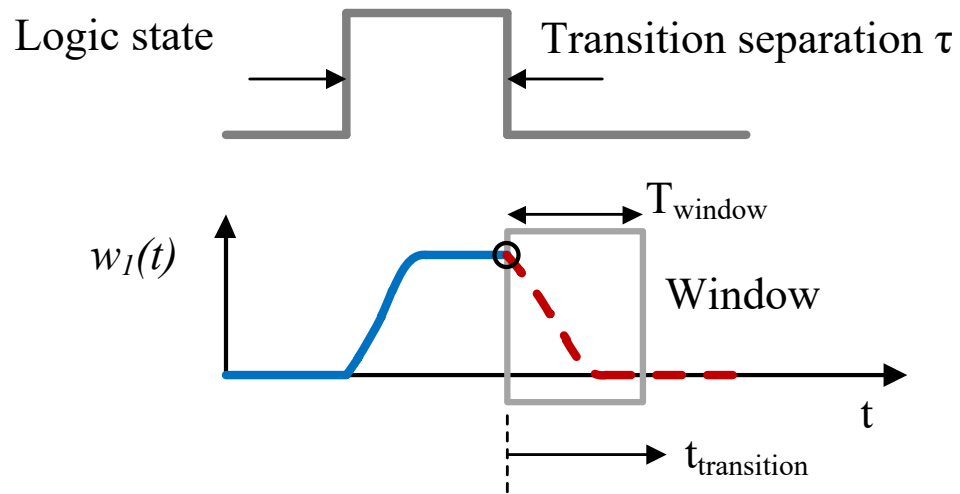


Figure 58 – Generation of weighting coefficients based on the concatenation scheme.

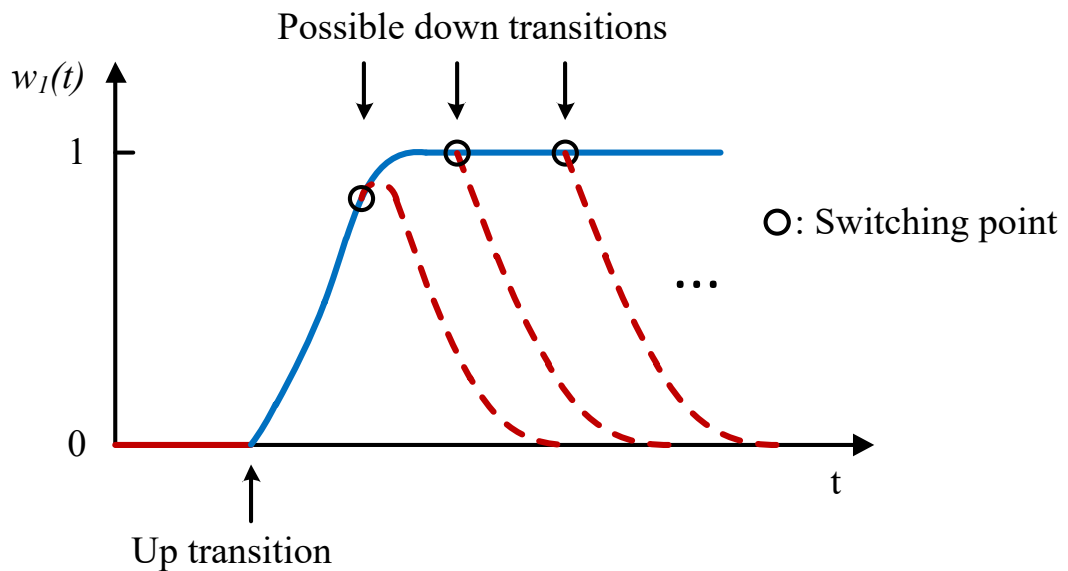


Figure 59 – Transitions of the weighting function under overclocking conditions.

In order to overcome this limitation, it is necessary to investigate the behavior of the weighting functions when the driver is overclocked. For this purpose, weighting function w_1 is extracted and plotted as shown in Figure 60, where the input of a Micron EDY4016AABG driver is excited using input patterns with different transition separations ranging from 100 ps to 500 ps. Note that here the delay between the driver input and last stage is removed, and the input waveform and w_1 waveform are realigned. It can be seen from Figure 60 (b) that the driver enters overclocking conditions when the transition separation is less than 250 ps. In the overclocking region, the weighting function is no longer a fixed-pattern signal. The peak value of the weighting function is reduced due to the fact that the driver does not have enough transition time to reach the steady state. Moreover the transition edge of the weighting function is shifted resulting in variable I/O delay. A similar behavior can be observed for weighting function w_0 . The degree of distortion increases with the overclocking data rate. The modeling of overclocked drivers will not be successful, unless this distortion is correctly captured.

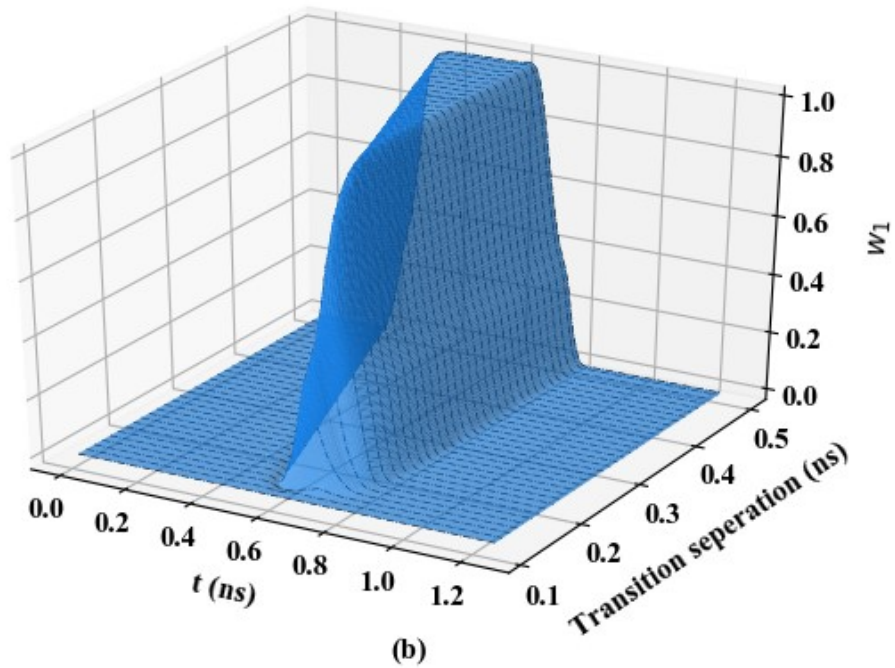
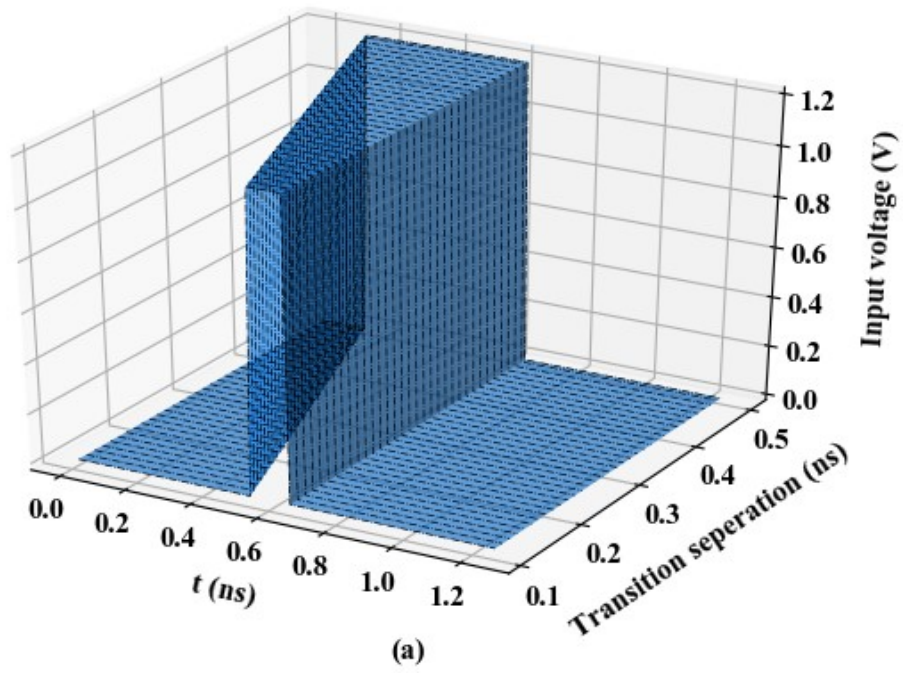


Figure 60 – (a) Input patterns with different transition separations. (b) Extracted weighting function w_1 .

In the following section, a transition-variational model is presented for the modeling of I/O drivers under overclocking conditions. Instead of fixed-pattern timing signals, the proposed model relies on the use of TVWFs, allowing for the modeling of variably distorted transition behavior.

4.4 Transition-Variational Model

4.4.1 Model Formulation with TVWFs

Based on the above discussion, the driver input stages exhibit much more dynamics under overclocking conditions. The existing parametric modeling approaches suffer from inaccuracy issues due to the incapability of their weighting coefficients generation mechanisms. In order to overcome this modeling difficulty, a dynamic weighting functions generation mechanism is proposed, based on the model structure in Equation 41. In the new model, the formulation of the proposed TVWFs $w_{m,n}(t)$ can be expressed as:

$$w_{m,n}(t) = G_{m,n}(t_{transition}, u_{m,n}), \quad (43)$$

where

$$\begin{cases} m = r, f \\ n = 0, 1 \end{cases}, \quad (44)$$

$t_{transition}$ is the transition time since the current switching event occurs, $u_{m,n}$ is the transition variable, and m denotes the transition direction (i.e., r for LOW-to-HIGH and f for HIGH-to-LOW transitions).

In Equation 43, transition variable $u_{m,n}$ is a key parameter that introduces flexibilities to the weighting functions for capturing the additional dynamics of the overclocked driver behavior. The transition variable records the overclocking status every time a logic switching event happens. As shown in Figure 59 and Figure 60, in different overclocking status, the corresponding weighting functions start from different switching points, and their transition shapes, peak values and delays vary as well. Therefore, in order to generate the weighting functions of the correct forms, the transition variable must uniquely indicate the overclocking status at the current switching point. Based on this analysis, good candidates for transition variable can be the weighting function values at the switching points (if the transition edges of the weighting functions are monotonically increasing or decreasing until steady states) or the transition separation τ between the current and precedent switching events (as shown in Figure 58). In this work, the transition separation τ is used as the transition variable.

Let T_{window} denote the maximum length of the window for the extraction of the weighting functions. T_{window} should be longer than the time needed to complete the transition. After T_{window} , the driver reaches steady state and the weighting functions remain constant, as shown in Figure 58. The behavioral model of the driver output port using the proposed TVWFs is expressed as:

$$i_{out}(t) = \begin{cases} [i_0(t) \ i_1(t)] \begin{bmatrix} w_{r,0}(t_{transition}) \\ w_{r,1}(t_{transition}) \end{bmatrix}, & \text{if LOW-to-HIGH \& } t_{transition} < T_{window} \\ [i_0(t) \ i_1(t)] \begin{bmatrix} w_{r,0}(T_{window}) \\ w_{r,1}(T_{window}) \end{bmatrix}, & \text{if LOW-to-HIGH \& } t_{transition} \geq T_{window} \\ [i_0(t) \ i_1(t)] \begin{bmatrix} w_{f,0}(t_{transition}) \\ w_{f,1}(t_{transition}) \end{bmatrix}, & \text{if HIGH-to-LOW \& } t_{transition} < T_{window} \\ [i_0(t) \ i_1(t)] \begin{bmatrix} w_{f,0}(T_{window}) \\ w_{f,1}(T_{window}) \end{bmatrix}, & \text{if HIGH-to-LOW \& } t_{transition} \geq T_{window} \end{cases}. \quad (45)$$

The corresponding weighting function concatenation algorithm is illustrated using a FSM graph as shown in Figure 61. Using the proposed method, the timing signals are no longer restricted to the fixed transition patterns between steady states. The use of the transition variable in the TVWFs along with the proposed concatenation scheme enables the behavioral model to identify the overclocking status of the driver and generate correct weighting coefficients, and thus allows for flexible modeling of different transition trajectories.

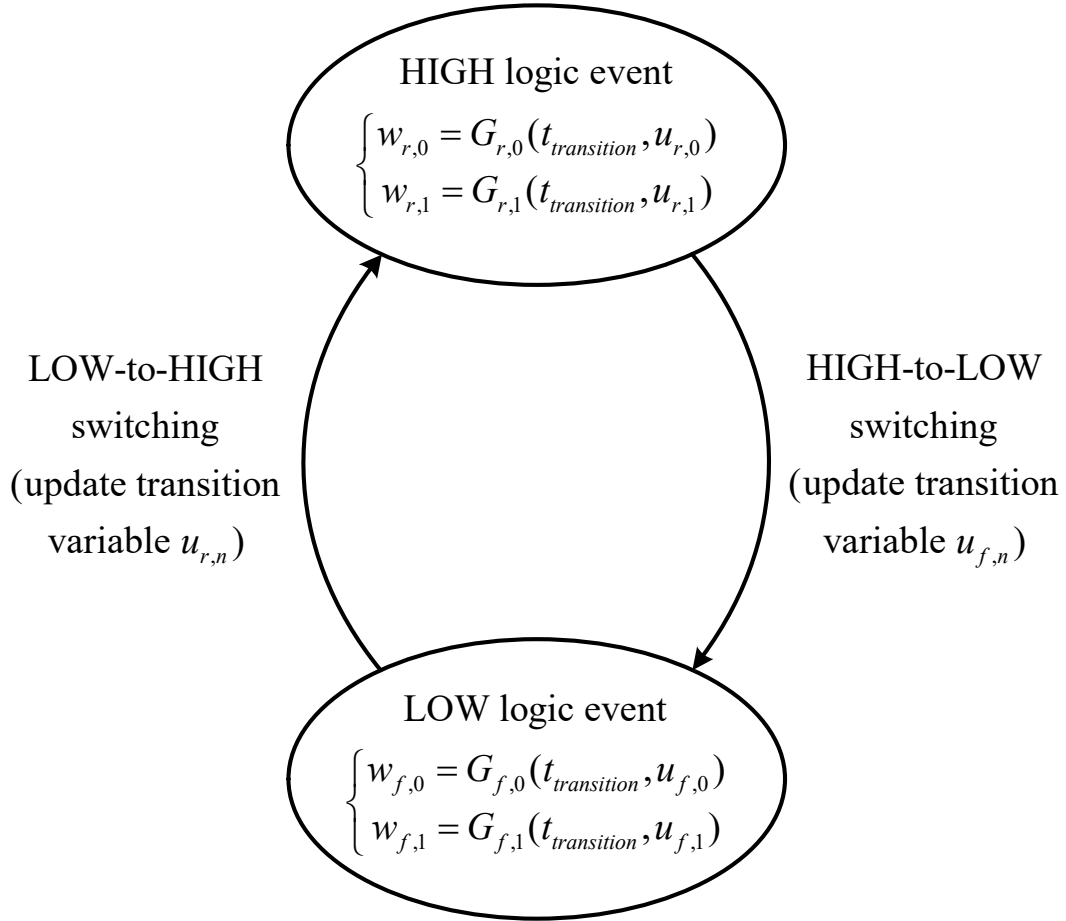


Figure 61 – FSM graph for the proposed modeling method using TVWFs.

Based on the proposed model formulation, appropriate mathematical model representations can be selected for the implementation of sub-models and weighting functions, such as table-formatted models, and parametric models including artificial neural networks. It is important to note that the proposed model is a general approach that is independent of the form of model representations. Details of the FSM algorithm and model implementation in Verilog-A are discussed in the APPENDIX B.

4.4.2 Modeling Process

The key steps to generate the proposed behavioral model are listed as follows:

1) *Dynamic sub-model characterization*: This step is designed to capture the I-V relations of the sub-models for HIGH and LOW logic states, respectively. A preferred approach is to drive the output port of the driver properly to obtain the output current and voltage waveforms using a piecewise-linear voltage source, with the driver's input set to HIGH or LOW voltage. The driver's nonlinearities and dynamics are embedded in these time-domain signals which should be captured by the generated sub-models. The driving waveforms of the voltage source can be multi-level signals containing different transition slopes and heights, which can sufficiently cover the possible voltage range. More details about the driving signals and the setup required for data collection can be found in [36] and [37].

2) *TVWFs extraction*: After the sub-models are characterized, the weighting functions should be extracted to capture the switching behavior of the driver circuit. Figure 62 shows the transient analysis setup required to collect the data for weighting coefficients extraction. In this step, the driver's input port is driven by switching patterns and the current and voltage waveforms at the output port are recorded. In order to extract the weighting functions through inversion of Equation 42, for each input pattern, the transient analysis should be performed using two different loads Z_1 and Z_2 connected to the output port, as discussed in [36] and [37]. The input waveforms used in this step are not fixed patterns but patterns with different transition separations instead. To properly excite the transition dynamics of the driver circuit under overclocking conditions, the input waveforms should perform a sampling of the input switching pattern space (as shown in Figure 60 (a) for HIGH-to-LOW transition). The transition separations should cover the range from the shortest possible bit duration to the time needed for a complete

transition. Separations larger than this range are not required to be considered, since the weighting functions remain the same for the transitions occurring after the driver has reached steady states. The same process applies to both LOW-to-HIGH and HIGH-to-LOW transitions. When extracting, the propagation delay between the driver input and output should be calculated, and the extraction window should be shifted accordingly to start at the switching point of the weighting functions, as shown in Figure 58. The extracted weighting coefficients along with the recorded transition variable values (e.g., transition separation values) are used to generate the weighting functions in Equation 43.

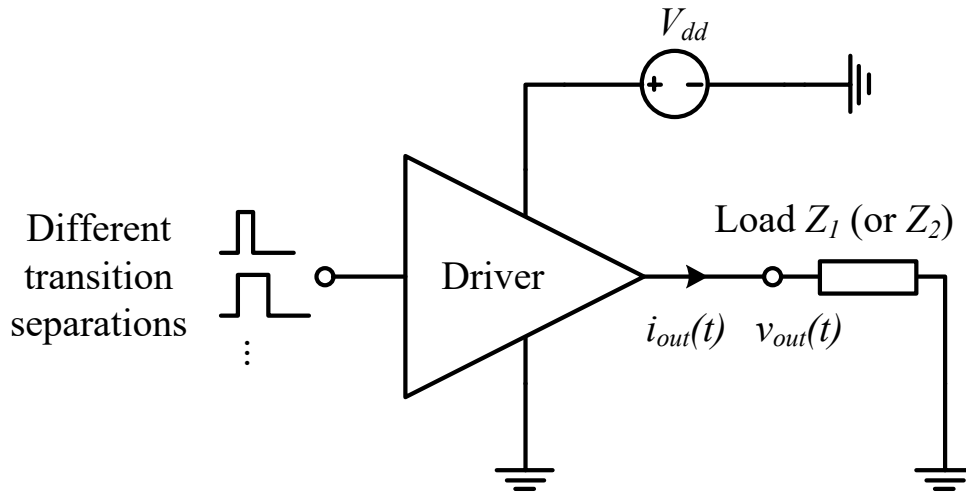


Figure 62 – Transient analysis setup used for the collection of data required for the extraction of weighting functions.

4.5 Modeling Examples

In this section, the accuracy and efficiency of the proposed model are validated under normal condition and overclocking condition using the Micron EDY4016AABG driver circuit (nominal $V_{dd} = 1.2 V$). As demonstrated above, the driver enters overclocking conditions when the bit cycle duration is less than 250 ps. The simulation data from the transistor-level driver model is used to generate the behavioral model and

used as the reference waveforms for model validation. The simulations are done using an Intel 2.4 GHz Xeon CPU.

4.5.1 Test Case under Normal Condition

A behavioral model with TVWFs is developed for the target driver circuit following the formulation and modeling procedure in Section 4.4. Considering the rich nonlinear dynamics and memory effects in the I-V relations of the driver's output port, RNNs are selected as the model representations for the sub-models. Specifically in this work, the RNNs of the nonlinear autoregressive network with exogenous inputs (NARX) type are used [54]. A time step of 2 ps is used for the post-processing of the simulation waveforms for RNN training. Each RNN has a single hidden layer of 10 neurons. For the extraction of weighting functions, a 50- Ω resistor is used as Z_1 , and the series connection of a V_{dd} voltage source and a 50- Ω resistor is used as Z_2 . The extracted TVWFs are implemented using look-up tables. The one time modeling process is done in 10 minutes. The generated model is implemented in Verilog-A.

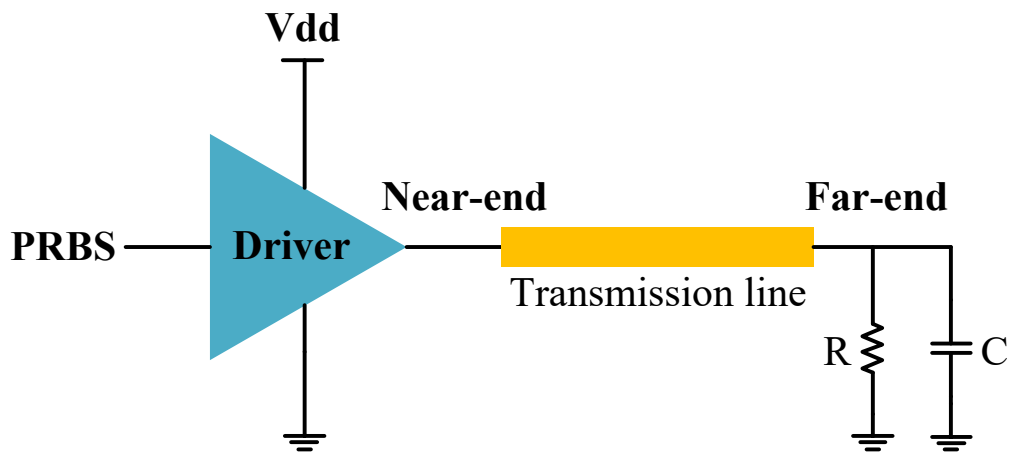


Figure 63 – Simulation setup for model validation using transmission line.

The simulation setup for model validation is shown in Figure 63, where the driver circuit is connected to a 50- Ω transmission line of 50 mm. The far-end of the transmission line is terminated using a 60- Ω resistor and a 1-pF capacitor in parallel. The input port of the driver is driven by a 500-bit long PRBS. The bit period is 500 ps, which allows the driver to work under normal condition. The rising and falling time is 10 ps.

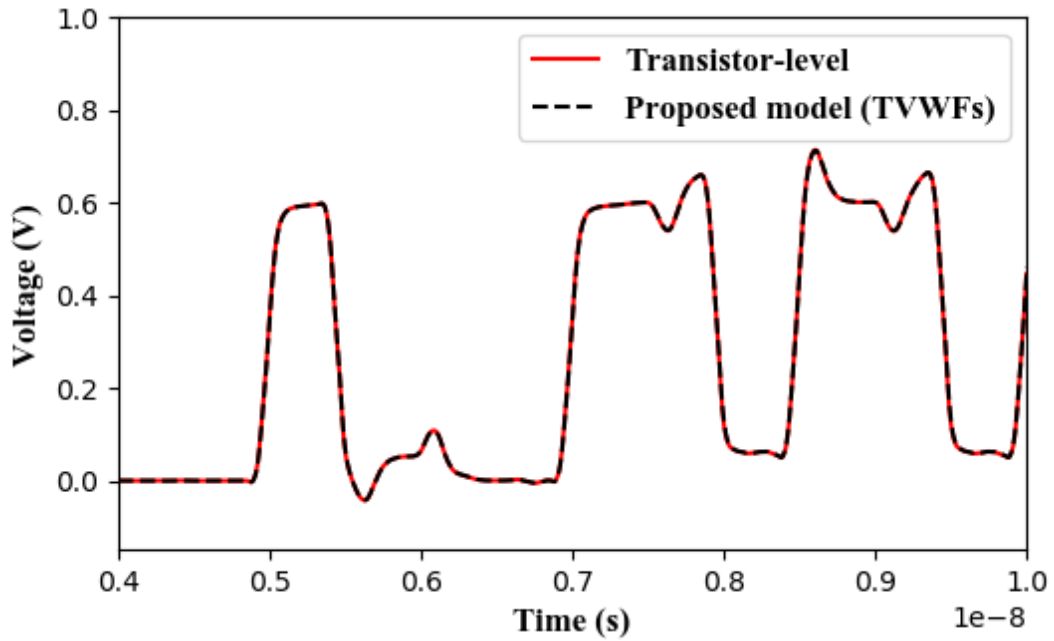


Figure 64 – Simulated voltage waveforms from the transistor-level model and the proposed model at the near-end of the transmission line.

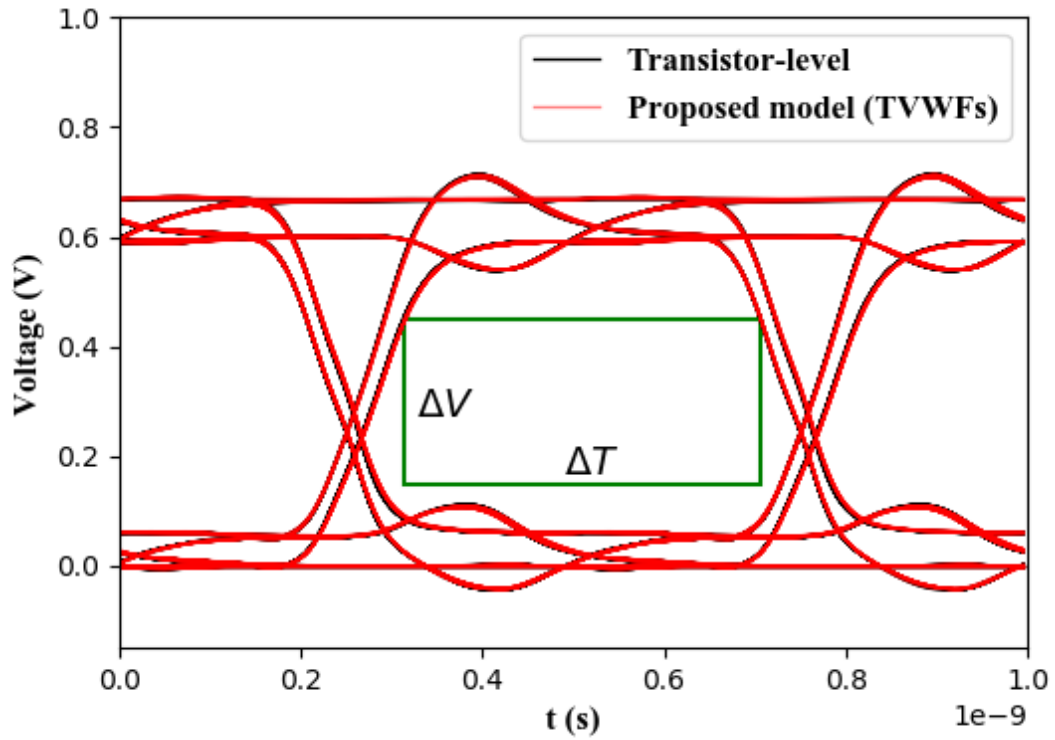


Figure 65 – Eye diagrams from the transistor-level model and the proposed model at the near-end of the transmission line.

Figure 64 shows the time-domain voltage waveform at the near-end of the transmission line. It can be seen from the figure that the waveforms from the transistor-level model and the proposed model match very well. The near-end eye diagrams are plotted as shown in Figure 65. The aperture height (ΔV) and width (ΔT) are measured and compared as shown in Table 4, which further demonstrates the fidelity and efficiency of the proposed model for driver simulations under normal condition.

Table 4 – Eye diagram performance and efficiency evaluation for normal condition test case (reference: transistor-level model).

Model	FOM	ΔV (V)	ΔT (ps)	ΔT Error		CPU Time (s)
				(ps)	(UI)	
Transistor-level	-	0.3	392.4	-	-	615.03
Proposed model (TVWFs)	99.87	0.3	391.2	1.2	0.2 %	9.19

4.5.2 Test Case under Overclocking Condition

In this test case, the driver modeling performance is validated under overclocking condition. The same simulation setup in the previous test case is used. The driver input is driven by a 500-bit long PRBS. Here the bit duration is set to 180 ps, which is much shorter than the normal case, making the driver circuit work under overclocking condition.

To demonstrate the effectiveness of the proposed model, another driver model is developed without the use of TVWFs. This non-TVWFs model is generated using the method in [60], where overclocking is handled through timebase realignment with fixed-pattern weighting functions extracted under normal transition conditions.

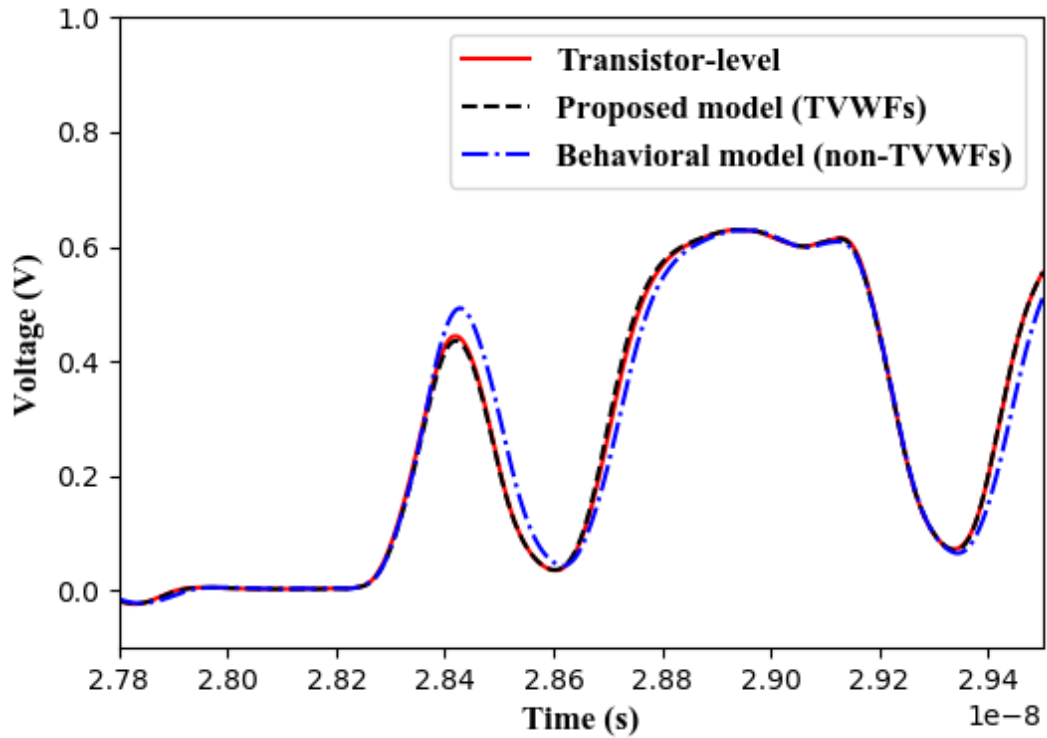


Figure 66 – Simulated voltage waveforms at the far-end of the transmission line from the transistor-level model, the proposed model (TVWFs) and the non-TVWFs behavioral model.

Both of these two behavioral models are simulated and compared to the transistor-level driver model using the same setup. The voltage waveforms at the far-end of the transmission line are shown in Figure 66. In the figure, the simulation results from the proposed model and the transistor-level model are in good agreement, whereas the non-TVWFs model fails to accurately predict the waveform in the overlocking region.

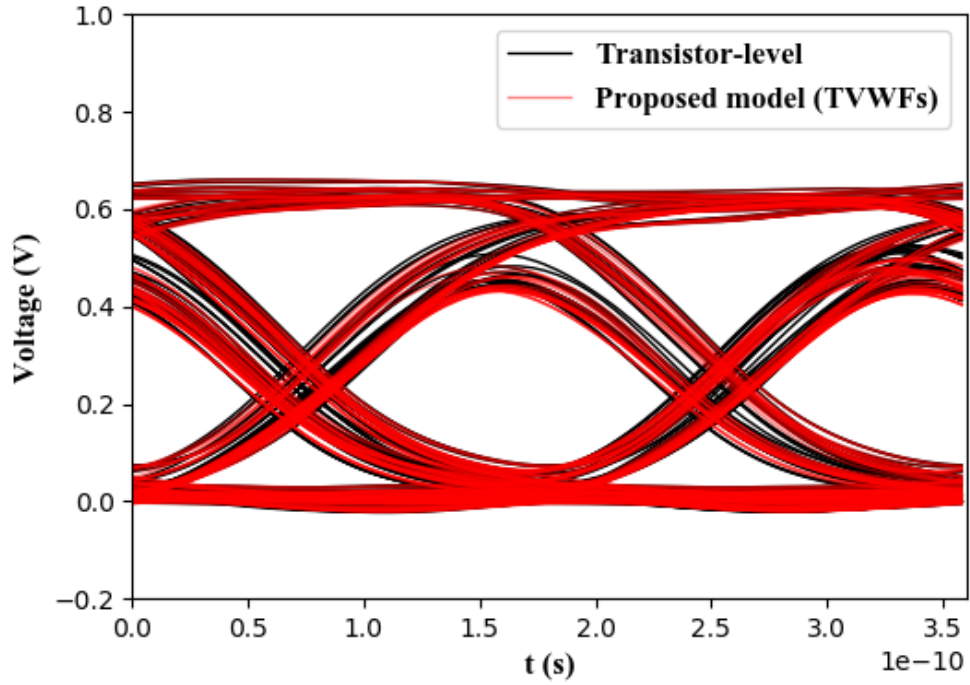


Figure 67 – Eye diagrams at the far-end of the transmission line from the transistor-level model and the proposed model (TVWFs).

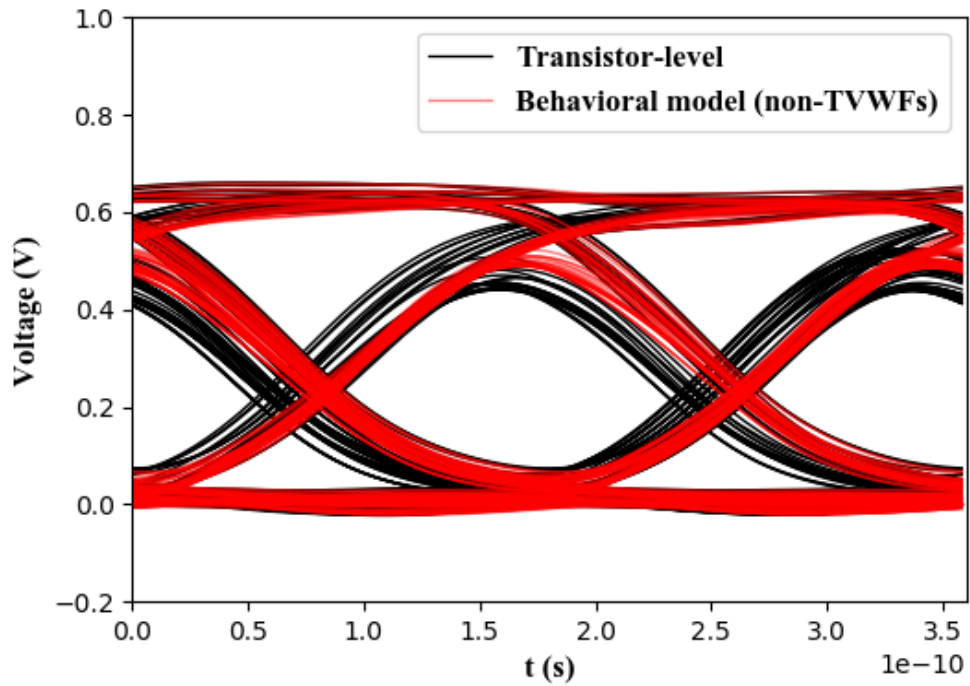


Figure 68 – Eye diagrams at the far-end of the transmission line from the transistor-level model and the non-TVWFs behavioral model.

The far-end eye diagram is plotted in Figure 67 for the proposed model, which visually matches well with the transistor-level results. On the contrary, significant mismatch can be observed for the non-TVWFs model, as shown in Figure 68. To quantify the modeling performance, the apertures and simulation time are measured as shown in Table 5. Using the proposed model, the ΔT error is reduced significantly from 24.6 ps (13.7% UI) to just 1.1 ps (0.6% UI), which confirms the effectiveness of the proposed TVWFs model for overclocked driver modeling. At the same time, the simulation is accelerated by a factor of ~ 72 compared to the transistor-level model.

Table 5 – Eye diagram performance and efficiency evaluation for overclocking condition test case (reference: transistor-level model).

Model	FOM	ΔV (V)	ΔT (ps)	ΔT Error		CPU Time (s)
				(ps)	(UI)	
Transistor-level	-	0.22	73.3	-	-	313.23
Proposed model (TVWFs)	99.54	0.22	72.2	1.1	0.6 %	4.32
Behavioral model (non-TVWFs)	97.24	0.22	98.0	24.6	13.7 %	4.28

4.6 Summary

In this chapter, a transition-variational behavioral model is proposed for the time-domain I/O driver modeling under overclocking conditions. TVWFs are proposed to capture the overclocked transition behavior of the driver’s input stages. The weighting coefficients can be generated appropriately for different overclocking scenarios. The corresponding modeling procedure is presented. As a black-box approach, the resulting

model also protects the intellectual property of the driver circuit design. The proposed model can be implemented as a Verilog-A module, which makes it compatible with many existing simulation tools. Modeling examples using commercial driver circuit demonstrated the accuracy and speed-up of the proposed model for both normal operation and overclocking simulations.

CHAPTER 5. CONCLUSION AND FUTURE WORK

With the increasing complexity of circuit designs, behavioral models are in demand for replacing transistor-level circuit models to achieve faster simulations. Behavioral models have reduced complexity compared to the original circuit, and thus require less CPU time. The simulation speed-up allows circuit designer to extensively explore the circuit design space during the initial development phase for optimized solutions, and reduces the time-to-market. On the other hand, behavioral models developed using black-box approaches do not reveal any internal design information of the circuit which is an advantage in terms of IP protection.

As discussed in this dissertation, the challenges for modeling oscillators and I/O drivers lie in the following aspects: First, a modeling method that can accurately capture the frequency-related information is required for oscillators, especially with trapezoidal waveform shapes. Second, modern I/O driver design with emerging features such as tunable control parameters and with pre-emphasis need to be considered, and I/O drivers with high data rates require improved modeling accuracy. Third, models suffer from degraded accuracy under overclocking conditions when the I/O drivers are excited with even higher data rates. To address these challenges, several behavioral modeling techniques are developed in this dissertation. Specifically, in Chapter 2, AugNNs are presented, which use periodic units proposed for oscillators. The proposed AugNNs can accurately mimic the oscillatory behavior. Furthermore, AugNN-based models are discussed which take into account the I/O behavior when buffers are included in oscillator circuits. In Chapter 3, behavioral model for tunable I/O drivers with pre-

emphasis is discussed, where the nonlinear dynamics are captured using RNNs and FFNNs. A behavioral model is developed for I/O drivers under overclocking conditions where the transition distortion is captured using transition-variational weighting functions in Chapter 4. All the proposed models are validated using transistor-level models. The simulation results indicate the good accuracy and efficiency of the proposed modeling methods.

5.1 Contributions

The contributions of the dissertation are summarized as follows.

- **Steady-State Model of Oscillator with Constant Frequency** A novel technique is presented for the steady-state time-domain behavior modeling of fixed-frequency oscillators using AugNN. In the proposed method, a FFNN with a periodic unit is developed to capture the periodicity of the oscillatory output waveform. Time step values are used as an input to the neural network to learn the phase information of the oscillation. The formulation of the corresponding training method is developed. The proposed model is not limited to the shapes of the target waveforms. The modeling of multi-phase oscillator is also addressed.
- **Steady-State Model of VCO** This work proposes time-domain steady-state behavioral model of VCOs. An AugNN with voltage-controlled periodic unit is developed which can accurately capture the voltage-frequency relation of the time-domain waveform. Inside the periodic unit, a second FFNN is used to map the control voltage to the instantaneous frequency. The associated training method is proposed. As opposed to the state space model, the proposed model generates

the output of a VCO explicitly using the AugNN and avoids solving system of differential equations.

- **Modeling of Oscillators with Output Buffers** With output buffer being part of an oscillator circuit, the behavior of the buffer needs to be incorporated into the modeling of the entire oscillator design. For this purpose, AugNN-based models are proposed for fixed-frequency oscillators and VCOs, where the oscillatory current-voltage relation of the output port can be accurately captured. In the proposed model, the nonlinear dynamic behavior of the output buffer is taken into account using RNNs, and an AugNN with multiple output units is used to capture the oscillatory weighting functions controlling the transitions of the buffer. Modeling examples show the fidelity of the reproduced waveforms from the model when different loads are used.
- **Behavioral Modeling of Pre-emphasis Drivers** The nonlinear behavioral modeling of pre-emphasis drivers including power supply noise is proposed, where state-aware weighting functions are used to accurately capture the transitions between pre-emphasis and steady states. The effects of power supply variation on the pre-emphasis behavior are demonstrated. The proposed model relies on parametric power-aware weighting functions that control the transitions of driver's output stage. Using the model extraction method introduced, the dynamics and nonlinearities of the driver's output and power supply ports are captured using RNNs, and the power-aware weighting functions are implemented using FFNNs. The model takes into account power supply induced jitter and thus can be used for SI and PI co-simulation.

- **Tunable Driver Modeling** In modern driver designs, advanced features emerge including the tunable characteristic with control parameters. The modeling of tunable drivers with pre-emphasis is addressed, where the behavior of tunable parameters are captured in a single model using neural networks. By doing so, the predominant nonlinear dynamic characteristics and memory effects of the driver circuit are modeled accurately, and the multivariate dependency (i.e., output and supply voltages, control parameters, etc.) of the driver behavior can be captured properly, making use of the advantages of neural networks. This also provides a more convenient approach compared to separately modeling a large number of control parameter combinations.
- **Overclocked Driver Modeling** A transition-variational behavioral model of I/O drivers is proposed for overclocking simulation. The driver behavior under overclocking conditions is investigated. The corresponding weighting function response surface is presented, which demonstrates the insufficiency of the previous modeling approaches that use fixed timing signals extracted under normal operating conditions only. The proposed model addresses the modeling of overclocking behavior by using TVWFs along with a transition variable. The weighting coefficients can be well captured for different overclocking scenarios. The corresponding model extraction flow is presented. Using the proposed TVWFs, the driver input signal is processed during run-time by a FSM algorithm which can be implemented in the widely supported hardware description languages such as Verilog-A. The proposed model is able to capture the driver

behavior accurately under both normal operation and overclocking conditions, and reduce the simulation time.

5.2 Future Work

1. Power-Aware Modeling of Oscillators

When power supply is non-ideal, the supply voltage is no longer at a constant level, but can vary. This supply voltage variation will affect not only the shape of the output waveform but also the oscillation frequency. To capture this influence, a model that takes into account the variation of supply voltage is needed.

The modeling approaches can be based on the AugNNs proposed in this dissertation. Specifically, for single-frequency oscillators, the frequency parameter inside the periodic unit of the AugNN should be a function of the supply voltage, which can produce the normalized oscillation phase under supply variation. The phase and supply voltage can then be used as the input to the FFNN to generate the corresponding output waveform. For VCOs, the instantaneous frequency should be controlled by both control voltage and supply voltage, which can be used as the input to the second FFNN inside the periodic unit. The phase, control voltage and supply voltage can then be used as the input to the main FFNN to generate the output waveform. Based on these modeling approaches, the corresponding training data generation scheme, training method and modeling procedure should be developed.

2. Transient Behavioral Modeling of Oscillators

When an oscillator is activated, there can be transient section before the oscillator goes into steady state. In the transient part, the amplitude of oscillation changes gradually from zero to the steady-state amplitude as shown in an example in Figure 69.

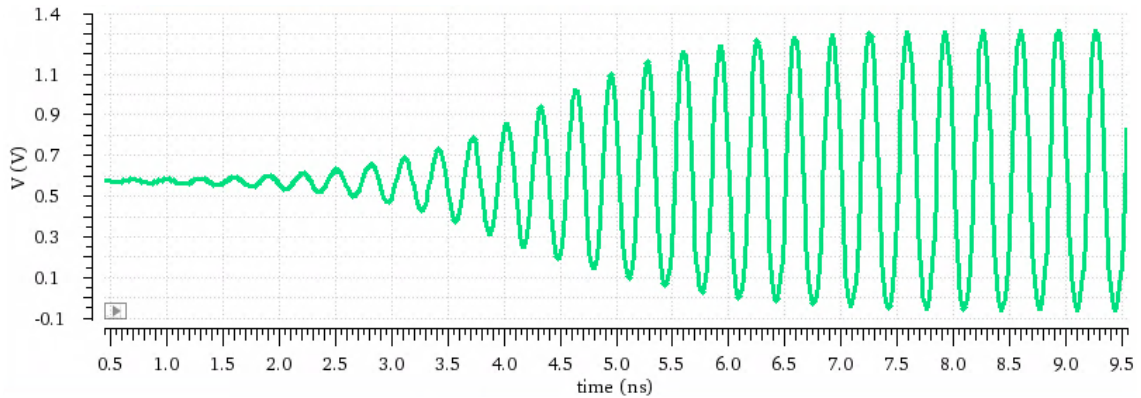


Figure 69 – Example transient waveform of an oscillator.

Because of the changing amplitude, the models developed for steady-state oscillators cannot be applied directly to transient behavioral modeling, since the steady-state models assume constant oscillation amplitude. To address the transient behavioral modeling of oscillators, possible approaches can be: 1. In the transient section, the output of the oscillator depends on not only the oscillation phase but also the previous output samples. To model the transient behavior of oscillators, previous time instances can be considered. 2. Decompose the oscillator waveform into transient and steady-state domains, and develop sub-models for different domains, respectively.

3. PAM-4 Driver Modeling

For high speed data transmission, four-level pulse amplitude modulation (PAM-4) is overtaking non-return to zero (NRZ) because of the bandwidth advantage. In the earlier work, it has been shown that state transitions and nonlinear output characteristics of I/O drivers can be captured quite well using proper mathematical model representations. However, what has been demonstrated in the literature cannot be directly applied to driver circuits with differing switching schemes as in PAM-4 signaling. The problem arises because the state transitions for PAM-4 drivers are quite different with multiple output voltage levels as shown in Figure 70.

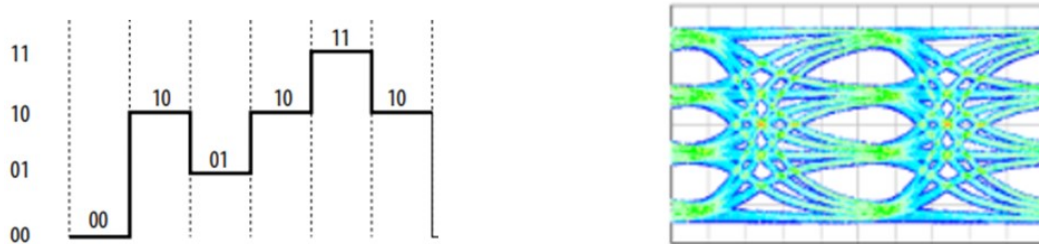


Figure 70 – Multi-level encoding and eye diagram in PAM-4 signaling.

However, it has been demonstrated in the previous section that parametric models can be flexible and quite powerful in modeling port current behavior. Therefore a modeling framework for PAM-4 drivers can be developed by capturing all the state transitions using proper mathematical representations and model components starting from its fundamental behavior. Before an appropriate model structure can be proposed, the general circuit structures of PAM-4 drivers must be investigated and analyzed. A modeling method that can accurately

capture the transitions between multi-level signals is needed and the corresponding parametric modeling framework can be developed.

5.3 Publications

Journal publications:

- **H. Yu** and M. Swaminathan, “A Transition-Variational Model of I/O Drivers for Overclocking Analysis,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019. (submitted)
- **H. Yu**, T. Michalka, M. Larbi, and M. Swaminathan, “Behavioral Modeling of Tunable I/O Drivers with Pre-emphasis Including Power Supply Noise,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2019. (accepted)
- **H. Yu**, H. Chalamalasetty, and M. Swaminathan, “Modeling of Voltage-Controlled Oscillators Including I/O Behavior Using Augmented Neural Networks,” IEEE Access, vol. 7, pp. 38973–38982, 2019.
- Y.-S. Li, **H. Yu**, H. Jin, T. E. Sarvey, H. Oh, M. S. Bakir, M. Swaminathan, and E.-P. Li, “Dynamic Thermal Management for 3-D ICs With Time-Dependent Power Map Using Microchannel Cooling and Machine Learning,” IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 9, no. 7, pp. 1244–1252, 2019.

- W.-S. Zhao, P.-W. Liu, **H. Yu**, Y. Hu, G. Wang, and M. Swaminathan, “Repeater Insertion to Reduce Delay and Power in Copper and Carbon Nanotube-Based Nanointerconnects,” *IEEE Access*, vol. 7, pp. 13622–13633, 2019.
- C. Gianfagna, **H. Yu**, M. Swaminathan, R. Pulugurtha, R. Tummala, and G. Antonini, "Machine-Learning Approach for Design of Nanomagnetic-Based Antennas," *Journal of Electronic Materials*, vol. 46, pp. 4963-4975, Aug 2017.
- M. Yi, S. S. Li, **H. Yu**, W. Khan, C. Ulusoy, A. Vera-Lopez, J. Papapolymerou, M. Swaminathan, "Surface Roughness Modeling of Substrate Integrated Waveguide in D-Band," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, pp. 1209-1216, Apr 2016.

Conference publications:

- H. M. Torun, **H. Yu**, et al., “A Spectral Convolutional Net for Co-Optimization of Integrated Voltage Regulators and Embedded Inductors,” the International Conference on Computer-Aided Design (ICCAD), 2019. (accepted)
- **H. Yu**, J. Shin, T. Michalka, M. Larbi, and M. Swaminathan, “Behavioral Modeling of Tunable I/O Drivers with Pre-emphasis Using Neural Networks,” 20th International Symposium on Quality Electronic Design (ISQED), 2019.
- **H. Yu**, J. Shin, T. Michalka, M. Larbi, and M. Swaminathan, “Behavioral Modeling of Pre-emphasis Drivers Including Power Supply Noise Using Neural Networks,” 10th IEEE Latin American Symposium on Circuits and Systems (LASCAS), 2019.

- M. Ahadi Dolatsara, **H. Yu**, J. Hejase, W. Becker, M. Swaminathan, “Polynomial Chaos modeling approach for jitter estimation in high-speed links”, 2018 IEEE International Test Conference (ITC), Oct 2018.
- **H. Yu**, H. Chalamalasetty, and M. Swaminathan, “Behavioral Modeling of Steady-State Oscillators with Buffers Using Neural Networks,” in Electrical Performance of Electronic Packaging and Systems (EPEPS), 2018 IEEE 27th Conference on. IEEE, 2018. (**Best student paper finalist**)
- **H. Yu**, M. Swaminathan, C. Ji, and D. White, “A nonlinear behavioral modeling approach for voltage-controlled oscillators using augmented neural networks,” 2018 IEEE MTT-S International Microwave Symposium (IMS), pp. 551–554, 2018.
- Y.-S. Li, E.-P. Li, **H. Yu**, H. Oh, M. Bakir, and M. Swaminathan, "Machine Learning for 3D-IC Electric-Thermal Simulation and Management," in 2018 IEEE International Conference on Computational Electromagnetics (ICCEM), 2018, pp. 1-3.
- **H. Yu**, M. Swaminathan, C. Ji, and D. White, “A method for creating behavioral models of oscillators using augmented neural networks,” in Electrical Performance of Electronic Packaging and Systems (EPEPS), 2017 IEEE 26th Conference on. IEEE, 2017.
- S. Li, M. Yi, S. Pavlidis, **H. Yu**, M. Swaminathan, and J. Papapolymerou, "Investigation of surface roughness effects for D-band SIW transmission lines on

LCP substrate," in Radio and Wireless Symposium (RWS), 2017 IEEE, 2017, pp. 121-124.

- S. J. Park, **H. Yu**, and M. Swaminathan, "Preliminary application of machine-learning techniques for thermal-electrical parameter optimization in 3-D IC," in Proc. IEEE Int. Symp. Electromagn. Compat. (EMC), pp. 402–405, July 2016.

APPENDIX A. OSCILLATOR MODEL IMPLEMENTATION

The oscillator behavioral model can be implemented as a Verilog-A module with three ports, i.e., p_in, p_out and p_gnd. Ports p_in and p_out represent the VCO's input (control voltage) and the circuit's output, respectively, and p_gnd represents the ground port which is used as the reference port when calculating the port voltages. In the declarations of the module, constants such as the weights and biases of the neural networks and the normalization constants are defined. The body of the module's behavioral description between the begin and end keywords contains the definition of the equations describing the neural networks, as well as the associated normalization and de-normalization. For neural networks that use hyperbolic tangent function tanh() as the activation function, the nodes in a hidden layer can be implemented as follows:

$$\begin{aligned} h_node[i] = & \tanh(w_node_i[1]*x1+ \\ & w_node_i[2]*x2+ \\ & \quad \vdots \\ & w_node_i[n]*x_n+ \\ & b_node_i); \end{aligned}$$

where x_1, x_2, \dots, x_n are the values of the nodes in the precedent layer, $w_node_i[1:n]$ is the vector containing the weights of the connections to the i_{th} node, $h_node[i]$, in the current layer and b_node_i is its bias. The output layer can be implemented in a similar format.

For AugNN, to obtain the folded phase, the integral of the total oscillation phase and the modulus operation in the periodic unit can be implemented using the Verilog-A built-in `idtmod` function:

```
folded_phase = idtmod(inst_f, p0, 1);
```

where `inst_f` is the instantaneous frequency, and `p0` is the initial phase.

For output current sub-models, RNN models are used. In order to obtain the previous time samples, internal nodes are defined in the module declaration part as:

```
electrical vout_d0, vout_d1, ..., vout_dr;  
electrical iH, iH_d0, iH_d1, ..., iH_dr;  
electrical iL, iL_d0, iL_d1, ..., iL_dr;
```

Here, `vout_dr`, `iH_dr` and `iL_dr` represent the previous time samples of the output voltage and sub-model currents with a delay of `r` sampling steps of `delta_t`. A good approach to acquire their values can be implemented using the Verilog-A built-in `ddt` function in a concatenated manner:

```

V(vout_d0)<+V(p_out, p_gnd);
V(vout_d1)<+V(vout_d0)-delta_t*ddt(V(vout_d1));
V(vout_d2)<+V(vout_d1)-delta_t*ddt(V(vout_d2));
      :
V(iH_d0)<+V(iH);
V(iH_d1)<+V(iH_d0)-delta_t*ddt(V(iH_d1));
V(iH_d2)<+V(iH_d1)-delta_t*ddt(V(iH_d2));
      :
V(iL_d0)<+V(iL);
V(iL_d1)<+V(iL_d0)-delta_t*ddt(V(iL_d1));
V(iL_d2)<+V(iL_d1)-delta_t*ddt(V(iL_d2));
      :

```

Once the neural networks for the weighting functions and the sub-models are defined, the behavioral model can be implemented based on Figure 26.

APPENDIX B. OVERCLOCKED DRIVER MODEL

IMPLEMENTATION

The proposed transition-variational model can be implemented as a Verilog-A module. In this work, RNNs are used as the mathematical representation of the output current sub-models. Details on implementing neural networks in Verilog-A can be found in [57]. The FSM algorithm for the weighting functions that process the input signals in real-time can be implemented as shown in Table 6.

Table 6 – Model implementation in Verilog-A.

```
module driver(in, out, gnd);  
  
- Declare ports, variables and nodes  
  
analog begin  
  
    V(in_delay, gnd) <+ absdelay(V(in, gnd), io_delay);  
  
    @(initial_step) begin  
  
        if (V(in_delay, gnd)<vin_threshold) begin  
  
            driver_state = -1;  
  
        end else begin  
  
            driver_state = 1;  
  
        end  
  
    end  
  
endmodule
```

```
end

end

@(cross(V(in_delay, gnd)-vin_threshold, +1)) begin

    t_cross = $abstime;

    if (driver_state == -2) begin

        transition_variable = t_cross - t_pre_cross;

    end else begin

        transition_variable = t_window;

    end

    t_pre_cross = t_cross;

    driver_state = 2;

end

@(cross(V(in_delay, gnd)-vin_threshold, -1)) begin

    t_cross = $abstime;

    if (driver_state == 2) begin

        transition_variable = t_cross - t_pre_cross;

    end else begin
```

```

        transition_variable = t_window;

    end

    t_pre_cross = t_cross;

    driver_state = -2;

end

case (driver_state)

    -2 : begin // HIGH-to-LOW transition

        t_transition = $abstime - t_cross;

        - calculate: w0 = Gf,0(t_transition,
transition_variable)

        - calculate: w1 = Gf,1(t_transition,
transition_variable)

    end

    -1 : begin // Initial state LOW

        - calculate: w0 = Gf,0(t_window,
transition_variable)

        - calculate: w1 = Gf,1(t_window,
transition_variable)

```

```

    end

    1 : begin // Initial state HIGH

        - calculate: w0 = Gr,0(twindow,
transition_variable)

        - calculate: w1 = Gr,1(twindow,
transition_variable)

    end

    2 : begin // LOW-to-HIGH transition

        ttransition = $abstime - tcross;

        - calculate: w0 = Gr,0(ttransition,
transition_variable)

        - calculate: w1 = Gr,1(ttransition,
transition_variable)

    end

endcase

- calculate: i0 = F0(V(out, gnd))

- calculate: i1 = F1(V(out, gnd))

I(out) <+ -(w0 * i0 + w1 * i1);

```

end

endmodule

REFERENCES

- [1] P.-H. Arnoux, P. Caillard, and F. Gillon, "Modeling Finite-Element Constraint to Run an Electrical Machine Design Optimization Using Machine Learning," *IEEE Transactions on Magnetics*, vol. 51, no. 3, pp. 1–4, 2015.
- [2] J. Xu, M. Yagoub, R. Ding, and Q. J. Zhang, "Exact adjoint sensitivity analysis for neural-based microwave modeling and design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 51, no. 1, pp. 226–237, 2003.
- [3] Y. Cao and Q.-J. Zhang, "State-Space Dynamic Neural Network Technique for High-Speed IC Buffer Modeling," *2007 International Symposium on Signals, Systems and Electronics*, 2007.
- [4] C. Zhang, S. Yan, Q.-J. Zhang, and J.-G. Ma, "Behavioral modeling of power amplifier with long term memory effects using recurrent neural networks," *2013 IEEE International Wireless Symposium (IWS)*, 2013.
- [5] S. Yan, C. Zhang, and Q.-J. Zhang, "Recurrent neural network technique for behavioral modeling of power amplifier with memory effects," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 25, no. 4, pp. 289–298, 2014.
- [6] V. Devabhaktuni, M. Yagoub, and Q.-J. Zhang, "A robust algorithm for automatic development of neural-network models for microwave applications," *IEEE Transactions on Microwave Theory and Techniques*, vol. 49, no. 12, pp. 2282–2291, 2001.
- [7] V. Devabhaktuni, B. Chattaraj, M. Yagoub, and Q. Zhang, "Advanced microwave modeling framework exploiting automatic model generation, knowledge neural networks and space mapping," *2002 IEEE MTT-S International Microwave Symposium Digest (Cat. No.02CH37278)*.
- [8] F. Wang and Q. Zhang, "Knowledge based neural models for microwave design," *1997 IEEE MTT-S International Microwave Symposium Digest*.
- [9] I. Lagaris, A. Likas, and D. Fotiadis, "Artificial neural networks for solving ordinary and partial differential equations," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 987–1000, 1998.

- [10] I. Lagaris, A. Likas, and D. Papageorgiou, "Neural-network methods for boundary value problems with irregular boundaries," *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1041–1049, 2000.
- [11] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative Adversarial Networks. [online] arXiv.org. Available at: <https://arxiv.org/abs/1406.2661>.
- [12] Liu, Z., Zhu, D., Rodrigues, S.P., Lee, K.T. and Cai, W., 2018. Generative model for the inverse design of metasurfaces. *Nano letters*, 18(10), pp.6570-6576.
- [13] H. M. Torun and M. Swaminathan, "High-Dimensional Global Optimization Method for High-Frequency Electronic Design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 6, pp. 2128–2142, 2019.
- [14] H. M. Torun, J. A. Hejase, J. Tang, W. D. Beckert, and M. Swaminathan, "Bayesian Active Learning for Uncertainty Quantification of High Speed Channel Signaling," 2018 IEEE 27th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2018.
- [15] R. Trincherro, M. Larbi, H. M. Torun, F. G. Canavero, and M. Swaminathan, "Machine Learning and Uncertainty Quantification for Surrogate Models of Integrated Devices With a Large Number of Parameters," *IEEE Access*, vol. 7, pp. 4056–4066, 2019.
- [16] S. S. Haykin, *Neural networks: a comprehensive foundation*. Prentice- Hall, Inc., 1998.
- [17] J. Xu, M. C. Yagoub, R. Ding, and Q.-J. Zhang, "Neural-based dynamic modeling of nonlinear microwave circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 50, no. 12, pp. 2769–2780, 2002.
- [18] F. Mkadem and S. Boumaiza, "Physically inspired neural network model for rf power amplifier behavioral modeling and digital predistortion," *IEEE Transactions on Microwave Theory and Techniques*, vol. 59, no. 4, pp. 913–923, 2011.
- [19] Y. Cao, X. Chen, and G. Wang, "Dynamic behavioral modeling of nonlinear microwave devices using real-time recurrent neural network," *IEEE Transactions on Electron Devices*, vol. 56, no. 5, pp. 1020–1026, 2009.
- [20] Cadence Verilog-A Language Reference. Cadence Design Systems, Inc., 2015.

- [21] M. Kraemer, D. Dragomirescu, and R. Plana, "Nonlinear behavioral modeling of oscillators in vhdl-ams using artificial neural networks," in Radio Frequency Integrated Circuits Symposium, 2008. RFIC 2008. IEEE. IEEE, 2008, pp. 689–692.
- [22] H. Yu, M. Swaminathan, C. Ji, and D. White, "A method for creating behavioral models of oscillators using augmented neural networks," in Electrical Performance of Electronic Packaging and Systems (EPEPS), 2017 IEEE 26th Conference on. IEEE, 2017, pp. 1–3.
- [23] M. Kraemer, D. Dragomirescu, and R. Plana, "A nonlinear order-reducing behavioral modeling approach for microwave oscillators," IEEE Transactions on Microwave Theory and Techniques, vol. 57, no. 4, pp. 991–1006, 2009.
- [24] I. Stievano and I. Maio, "Behavioral models of digital IC ports from measured transient waveforms," IEEE 9th Topical Meeting on Electrical Performance of Electronic Packaging (Cat. No.00TH8524).
- [25] I. Stievano, D. Becker, F. Canavero, Z. Chen, G. Katopis, and I. Maio, "Behavioral modeling of IC ports including temperature effects," 2002 IEEE 11th Topical Meeting on Electrical Performance of Electronic Packaging, 2002.
- [26] I. Stievano, I. Maio, F. Canavero, and C. Siviero, "Parametric macromodels of differential drivers and receivers," IEEE Transactions on Advanced Packaging, vol. 28, no. 2, pp. 189–196, 2005.
- [27] Q. Zhang, Y. Cao, and I. Erdin, "Fast IO buffer modeling using neural network methods," 2010 11th International Conference on Electronic Packaging Technology & High Density Packaging, 2010.
- [28] W. Dghais, T. R. Cunha, and J. C. Pedro, "Reduced-Order Parametric Behavioral Model for Digital Buffers/Drivers With Physical Support," IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 2, no. 12, pp. 2071–2079, 2012.
- [29] W. Dghais and F. H. Bellamine, "Discrete controlled pre-driver FIR model for hybrid IBIS model AMS simulation," 2016 IEEE 20th Workshop on Signal and Power Integrity (SPI), 2016.
- [30] C. Diouf, M. Telescu, N. Tanguy, I. S. Stievano, and F. G. Canavero, "Robust nonlinear models for CMOS buffers," 2016 IEEE 20th Workshop on Signal and Power Integrity (SPI), 2016.

- [31] I. Stievano, F. Canavero, Z. Chen, G. Katopis, and I. Maio, "Parametric macromodels of drivers for SSN simulations," 2003 IEEE Symposium on Electromagnetic Compatibility. Symposium Record (Cat. No.03CH37446).
- [32] Signorini, Siviero, Telescu, and Stievano, "Present and future of I/O-buffer behavioral macromodels," IEEE Electromagnetic Compatibility Magazine, vol. 5, no. 3, pp. 79–85, 2016.
- [33] Y. Cao, I. Erdin, and Q.-J. Zhang, "Transient Behavioral Modeling of Nonlinear I/O Drivers Combining Neural Networks and Equivalent Circuits," IEEE Microwave and Wireless Components Letters, vol. 20, no. 12, pp. 645–647, 2010.
- [34] (2015, September) IBIS (I/O Buffer Information Specification), version 6.1. [Online]. Available: <https://ibis.org/ver6.1/ver6.1.pdf>
- [35] I. S. Stievano, I. A. Maio, and F. G. Canavero, "Parametric macromodels of digital I/O ports," IEEE Transactions on Advanced Packaging, vol. 25, no. 2, pp. 255–264, 2002.
- [36] I. S. Stievano, I. A. Maio, and F. G. Canavero, "M π log, macromodeling via parametric identification of logic gates," IEEE Transactions on Advanced Packaging, vol. 27, no. 1, pp. 15–23, 2004.
- [37] B. Mutnury, M. Swaminathan, and J. P. Libous, "Macromodeling of nonlinear digital I/O drivers," IEEE Transactions on Advanced Packaging, vol. 29, no. 1, pp. 102–113, 2006.
- [38] B. Mutnury, M. Swaminathan, M. Cases, N. Pham, D. de Araujo, and E. Matoglu, "Macro-modeling of non-linear pre-emphasis differential driver circuits," in Microwave Symposium Digest, 2005 IEEE MTT-S International. IEEE, 2005, pp. 4–pp.
- [39] I. S. Stievano, I. A. Maio, F. G. Canavero, and C. Siviero, "Parametric macromodels of differential drivers with pre-emphasis," IEEE Transactions on advanced packaging, vol. 30, no. 2, pp. 238–245, 2007.
- [40] W. Dghais, T. R. Cunha, and J. C. Pedro, "A novel two-port behavioral model for i/o buffer overclocking simulation," IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 3, no. 10, pp. 1754–1763, 2013.
- [41] B. Razavi, RF Microelectronics, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2011.

- [42] (2000, April) I/O Buffer Accuracy Handbook, revision 2.0. [Online]. Available: <https://ibis.org/accuracy/handbook.pdf>
- [43] Virtuoso Spectre Circuit Simulator and Accelerated Parallel Simulator User Guide. Cadence Design Systems, Inc., 2016.
- [44] M. Grozing, B. Phillip, and M. Berroth, "CMOS ring oscillator with quadrature outputs and 100 MHz to 3.5 GHz tuning range," ESSCIRC 2004 - 29th European Solid-State Circuits Conference (IEEE Cat. No.03EX705), pp. 679–682, 2003.
- [45] E. J. Pankratz and E. Sanchez-Sinencio, "Multiloop High-Power-Supply-Rejection Quadrature Ring Oscillator," IEEE Journal of Solid-State Circuits, vol. 47, no. 9, pp. 2033–2048, 2012.
- [46] K. Madsen, H. B. Nielsen, and O. Tingleff, Methods for non-linear least squares problems 2nd ed. 2004 [Online]. Available: <http://orbit.dtu.dk/files/2721358/imm3215.pdf>, 2004
- [47] Y. Fang, M. C. Yagoub, F. Wang, and Q.-J. Zhang, "A new macromodeling approach for nonlinear microwave circuits based on recurrent neural networks," IEEE Transactions on Microwave Theory and Techniques, vol. 48, no. 12, pp. 2335–2344, 2000.
- [48] P. Tehrani, Y. Chen, and J. Fang, "Extraction of transient behavioral model of digital I/O buffers from IBIS," 1996 Proceedings 46th Electronic Components and Technology Conference, pp. 1009–1015, 1996.
- [49] T. Zhu, M. B. Steer, and P. D. Franzon, "Accurate and scalable io buffer macromodel based on surrogate modeling," IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 1, no. 8, pp. 1240–1249, 2011.
- [50] H. Yu, J. Shin, T. Michalka, M. Larbi, and M. Swaminathan, "Behavioral modeling of tunable i/o drivers with pre-emphasis using neural networks," 2019 20th International Symposium on Quality Electronic Design (ISQED), 2019.
- [51] A. Varma, M. Steer, and P. Franzon, "Improving behavioral IO buffer modeling based on IBIS," IEEE Transactions on Advanced Packaging, vol. 31, no. 4, pp. 711–721, 2008.
- [52] W. Dghais and J. Rodriguez, "New Multiport I/O Model for Power-Aware Signal Integrity Analysis," IEEE Transactions on Components Packaging and Manufacturing Technology, vol. 6, pp. 447–454, Mar 2016.

- [53] G. Signorini, C. Siviero, S. Grivet-Talocia, and I. S. Stievano, "Macromodeling of I/O buffers via compressed tensor representations and rational approximations," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 6, no. 10, pp. 1522–1534, 2016.
- [54] H. Yu, J. Shin, T. Michalka, M. Larbi, and M. Swaminathan, "Behavioral modeling of pre-emphasis drivers including power supply noise using neural networks," 2019 IEEE 10th Latin American Symposium on Circuits & Systems (LASCAS), 2019.
- [55] S. Yan, C. Zhang, and Q.-J. Zhang, "Recurrent neural network technique for behavioral modeling of power amplifier with memory effects," *International Journal of RF and Microwave Computer-Aided Engineering*, vol. 25, no. 4, pp. 289–298, 2014.
- [56] Z. Chen, M. Raginsky, and E. Rosenbaum, "Verilog-a compatible recurrent neural network model for transient circuit simulation," in *Electrical Performance of Electronic Packaging and Systems (EPEPS)*, 2017 IEEE 26th Conference on. IEEE, 2017.
- [57] H. Yu, H. Chalamalasetty, and M. Swaminathan, "Modeling of voltage-controlled oscillators including i/o behavior using augmented neural networks," *IEEE Access*, 2019.
- [58] C. Diouf, M. Telescu, I. S. Stievano, N. Tanguy, and F. G. Canavero, "Simplified topology for IC buffer behavioural models," *IET Circuits Devices Syst.*, vol. 11, no. 2, pp. 183–187, 2017.
- [59] W. Dghais, M. Souilem, and M. Alam, "Mixed-Signal Overclocked I/O Buffers Model Abstraction for Signal Integrity Assessment," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 3, pp. 691–699, 2019.
- [60] G. Signorini, C. Siviero, S. Grivet-Talocia, and I. S. Stievano, "Macromodeling of I/O buffers via compressed tensor representations and rational approximations," *IEEE Trans. Compon., Packag., Manuf. Technol.*, vol. 6, no. 10, pp. 1522–1534, Oct. 2016.