Technische Universität München
Lehrstuhl für Entwurfsautomatisierung

Accurate Waveform-based Timing Analysis with Systematic Current Source Models

Christoph Knoth

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technische Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:	Univ.-Prof. Dr. sc. techn. Gerhard Kramer
Prüfer der Dissertation:	
1:	Univ.-Prof. Dr.-Ing. Ulf Schlichtmann
2:	Univ.-Prof. Dr.-Ing. Klaus Hofmann, Technische Universität Darmstadt

Die Dissertation wurde am 17.04.2012 bei der Technische Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 27.08.2012 angenommen.

Die Dissertation ist als gebundenes Buch im isle Verlag erschienen.
A book version of this thesis is available from the publisher isle. ISBN: 978-3-938843-75-8

Prephase

This book is the outcome of my research work done at the Institute for Electronic Design Automation. Consequently, I first thank Univ.-Prof. Dr.-Ing. Ulf Schlichtmann for providing me with this opportunity, for his support, and for the trust which allowed me to freely explore my ideas. Furthermore, I thank Univ.-Prof. Dr.-Ing. Klaus Hofmann and Univ.-Prof. Dr. sc. techn. Gerhard Kramer for their work in the committee.

I further thank our partners at Infineon Technologies AG. Undoubtedly at first Petra Nordholz, who contributed much CSM experience and always was an advocate for the project. Then, of course, the Titan group around Reinhart Schultz and the team around Manfred Haberl. My thanks further go to everyone involved in providing us with licences, especially in tough times.

At the institute, I would like to express special gratitude to Manuel Schmidt and Michael Eick. Similar to Petra, their research directly influenced my work. I thank Husni Habal for helping me to avoid the most evil language pitfalls.

None of this would have been possible without the support of helping hands from students. In chronological order these are: Veit Kleeberger, Irina Eichwald, Christoph Maßmann, Sebastian Kiesel, Carsten Uphoff, and Hela Jedda. Thank you.

Concerning this work and other publications, I thank the following people for reading and suggesting improvements: Hela Jedda, Shushanik Karapetyan, Carsten Uphoff, Daniel Platte, Adrian Knoth, Reinhold Herschel, and Bing Li.

Finally I thank my family, my girlfriend Nadine, and my friends for motivation and constant support.

München, November 25, 2012

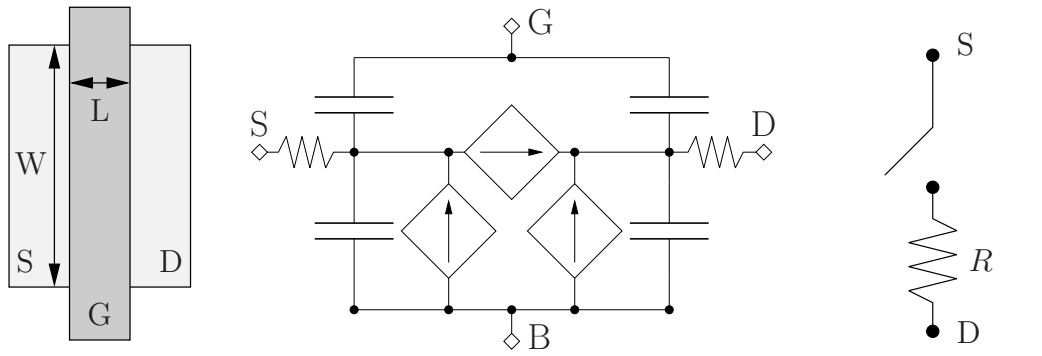
Contents

Time waits for nobody.

Freddy Mercury

1 Introduction

The first integrated circuit (IC) has been presented by Kilby in 1956 [Kil76]. It was a sinewave generator consisting of a single transistor and a few passive components. Since then, a rapid growth of IC complexity has been observed [Mac11]. In 2011, ICs with more than one million transistors are common. A total number of about 10^{16} transistors are manufactured every day [Han11], and several hundred new designs are released every year. A KPMG survey showed that 50 percent of the IC manufacturers start three or more new projects every year, and expect first revenues after a time span of 18–21 months [KPM08]. This corresponds to a typical design cycle of about 9–12 months, even for complex designs with millions of transistors [All07]. Such development times are only possible through electronic design automation (EDA), which offers tools to support the designer by means of analyses and synthesis. First EDA tools have been developed in the 1960s with the availability of digital computers [KK66, MP71]. The IC simulators aimed to predict the behaviour of a new design before its fabrication. This became necessary because building prototypes for ICs requires significantly more time and money than making prototypes using discrete components. Since then, many things have changed. More and more tools have been developed at various levels of abstraction to assist the IC designer. Nonetheless, one thing never changed: Simulation speed and capability are never satisfactory. The simulator "Time" in 1971 allowed to analyse circuits with more than 50 transistors, and the authors already complained about the poor simulation speed of earlier programs [Jen71]. A simple analysis of five transistors took 12 seconds. In 1973, the then newly developed "Spice" was capable of handling circuits of up to 200 elements, of which 100 could be nonlinear. A short simulation of 22 bipolar transistors required 16.7 seconds. Without being more specific, in 1990 Merten describes a simulator that handles "some hundreds of transistors in reasonable computation time" [Mer90]. By comparing these examples, one general trend can be inferred: The improvements in simulation tools and the large increase in computational capability



(a) Physical (b) Analog (transient) (c) Digital switch

Figure 1.1.: Different models for MOS transistor

are immediately compensated by even larger designs which have to be simulated. The only solution to this complexity problem is in abstraction, hierarchy, and re-use. More abstract models are developed which describe the relevant aspects in a simpler way and completely neglect features that are not important for a particular analysis. Of course, there is a trade-off between accuracy and simulation speed. For example, Fig. 1.1 shows different models of a transistor. The physical model describes geometries. This is relevant to device simulation from which electrical properties can be derived [SLM06]. For analog circuit design, only these electrical characteristics matter, which are therefore described in complex transistor models. For most digital applications, the transistor is only used in conduction or isolating state. Consequently, a simple switch with constant channel resistance might be sufficient.

Further abstraction, from transistors to logic cell, allows to introduce hierarchy and re-use. Designers working on gate level use standard cells as building block instead of transistors. The same design of a logic cell can be used multiple times and in different designs. Model complexity is reduced by describing each cell by a Boolean equation and a delay value. This simple representation allows to quickly construct and analyse very complex IC designs, and it is the key to the automated synthesis of digital circuits.

The more abstract the model, the larger the difference to the real hardware. Analyses which are carried out in early design phases lack many information, e.g., physical positions of the components. Consequently, there are errors and uncertainties in the estimation. The aim of each analysis is to provide a reasonably accurate performance prediction which aligns well with the next, more precise, analysis. This ensures that there are no insolvable problems in late design phases. Unfortunately, the continued transistor scaling has partly invalidated this separation of modelling layers. Modern integrated components must consider layout information to ensure sufficient yield. Similarly, analyses of digital components must account for analog effects to be accurate [KO95], e.g., the challenges arising from metal interconnects, which are strongly coupled to neighbouring nets. Consequently, signals influence each other and hence change path delays. Performing timing analysis for such nets is very difficult since traditional digital models

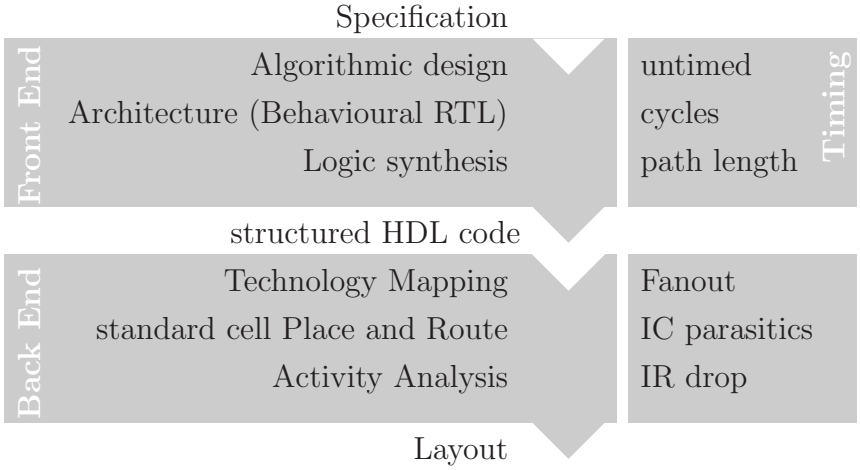


Figure 1.2.: Simplified digital design flow

cannot handle the occurring arbitrary waveforms. On the other side, analog modelling and simulation cannot cope with the enormous complexity. Thus, there is again the need for a new model trade-off as well as a simulator suited to digital circuits. Before refining the objectives of the research project in Sec. 1.2, a short introduction on designing integrated circuits is provided. It also includes a brief review on current problems and trends.

1.1. Designing Digital Integrated Circuits

Designing integrated circuits (ICs) is a structured process of continued refinement of the IC model until finally mask data for fabrication can be produced. Figure 1.2 shows a simplified design flow. It starts with specifying the intended functions and performances of the new IC. Once the required algorithms are found, the IC project is usually broken into several modules which implement specific parts. For each module, there is a behavioural description, written in a hardware description language (HDL), to express *what* the module does. Logic synthesis transforms this behavioural register transfer level (RTL) code into an equivalent set of Boolean equations. This step is yet technology independent, but the equations can be written as a netlist of generic logic cells. Denoted as structured RTL, it describes *how* the behaviour can be achieved, and it is the final product of the front end design phase. During physical synthesis, back end designers implement this design with the available technology. They must ensure that the final IC meets design constraints such as area, power consumption, and operating frequency. Therefore, technology mapping makes an optimal selection of logic cells and typically restructures the cell netlist. During Place and Route (PnR) these cells are assigned to locations on the chip layout and their electrical connection is defined. Provided that the performances are met, each cell instance is replaced by the real polygons for transistors and contacts to generate the mask data for fabrication.

Each step of the design flow is accompanied by several analyses and checks, of which

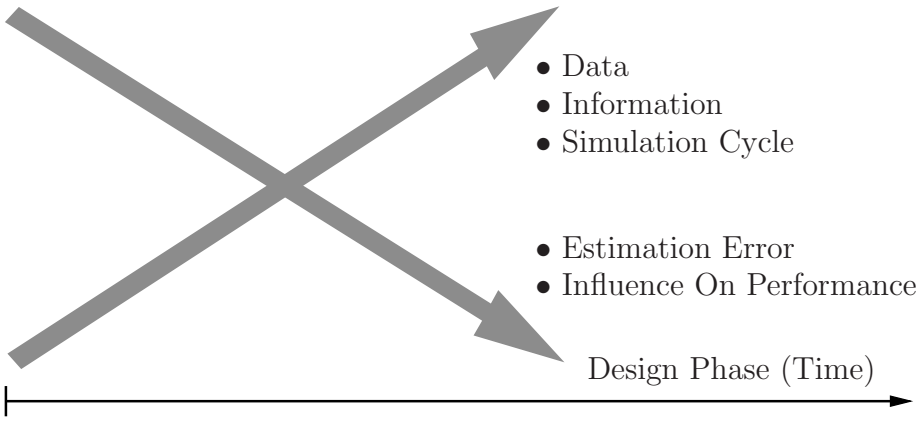


Figure 1.3.: Lower simulation speed but higher accuracy in late design phases

only timing analysis shall be discussed here. It is first performed with structured RTL to ensure that logic operations are completed within the intended clock frequency. Using the maximum number of logic stages and a fixed delay value for each cell gives a rough estimate for maximum path delay. However, as the design itself is still rather abstract, this value cannot be very accurate. The estimate is more realistic after technology mapping when the cells' output loads are known. A quick trail PnR is performed to get estimates for length and shape of interconnects. This adds to the capacitive loads and allows a more realistic timing analysis. If it is likely that timing closure will be successful, a detailed PnR is performed. Based upon the final cell locations and interconnect shapes, a detailed parasitic layout extraction yields a precise electrical model for the interconnect, which is written in the Standard Parasitic Exchange Format (SPEF) [BC09]. It is the basis for most time consuming and most accurate analyses during sign-off. It includes crosstalk analysis to ensure signal integrity as well as to consider noise-on-timing effects [Das+11, RLB00, ARP00, HGB02]. An activity analysis is performed to compute current consumption for typical operations [SLM06]. Together with the model for the supply net, this yields information on the local lowering of supply voltage. This IR drop influences the cell delay and is also considered in the timing analysis. Due to the long turnaround times, timing closure can take several months [Wak05].

Recent back end developments

Semiconductor companies encounter high economical pressure, and being able to deliver the product at the right moment is crucial for the success of a project. To meet tough time-to-market constraints, it is essential that analyses at each step of the design flow provide reasonably accurate performance estimations within short turnaround times. At the same time, the continued reduction of transistor sizes has led to increased vulnerability of circuit performance with respect to parameter variation, crosstalk, and wear out phenomena over lifetime [ARA08]. Moreover, tighter requirements on power consumption limit the applicability of countermeasures. Consequently, modern design tools

must be variation-aware [SSR08], which implies massive challenges in terms of complexity and standard cell characterisation. Unfortunately, many effects are only analysable during the late design phases. This is shown in Fig. 1.3, where information, data, and simulation time increase as the IC project is more and more refined. On the other hand, decisions made in early design phases generally have larger influence on system performances. Consequently, new EDA tools often internally perform a quick design flow to obtain the required information. For instance, starting at 130 nm, the effect of noise on timing is so severe that it must be considered during static timing analysis (STA) [KTK08]. This implies that STA engines must almost provide sign-off quality even for the optimisation loops of technology mapping. For timing analysis during design closure, the impact of interconnects on timing is growing. One factor is additional capacitive loading. Moreover, since via resistance does not scale with technology, the resistive nature of interconnects is growing and poses new challenges on the accuracy of cell delay models. The "undesired analog behaviour" of logic cells at smaller technology nodes and at higher operation speed becomes more pronounced [KO95]. The delay errors of traditional timing models for standard cells are growing, to a large extend because of the incapability to handle realistic waveforms. According to [KO95], generally there are two approaches: (a) more precise timing models for logic simulation, and (b) reducing complexity of circuit models and improving circuit simulation.

1.2. Problem formulation and Contributions

Aim of the work is to provide new possibilities to analyse the timing behaviour during late stage of the design flow. This includes the development of a new model for logic cells that can accurately predict voltage waveforms but is significantly faster than transistor level simulation. This current source model (CSM) must be sufficiently accurate for a wide range of logic cells, and parameter variations must be included in a waveform-accurate static timing analysis. Model generation must be fast and automated, simulation accuracy close to Spice, and simulation performance close to STA tools. If possible, the new CSM is a holistic model that can be used for different analyses [MKA08]. This would not only dramatically reduce the characterisation effort, but as the authors point out, an accurate model should account for the signal waveform, IR drops, crosstalk, multiple-input switching, nonlinearity of the receiver capacitance, and parameter variation.

This work makes three contributions. The first is a systematic modelling approach for standard cells. The identification of root causes of observable cell behaviour leads to a simple model structure and a very efficient, but also accurate, characterisation scheme. Based upon this transparent modelling, two new models, Systematic Current Source Model (SCSM) and Pull-up/Pull-down Model (PXM), are presented. SCSMs are parametric to account for variations of process parameters, and to enable statistical analyses. Although primarily intended for timing analysis, the PXM is a holistic yet simple cell model for (combined) timing, noise, and power analysis. Its name refers to a separate modelling of PMOS and NMOS transistors. The second contribution is the

integration of the new CSMs into existing circuit simulators. This reduces simulation time of sophisticated analyses and allows the user to continue using his well-known program. The third contribution is SWAT, a dedicated simulator for waveform-accurate timing. It is optimised for using the new SCSMs and provides Monte Carlo simulations as well as a sensitivity propagation analysis.

Several publication resulted from this work: [Kno+08, Kno+09, KKNS09a, NK09, KN09, KKNS09b, KENS10, KUKS11, KS12, KJS12]. The following master and bachelor theses were created in the course of this project: [Kle09, Eic09, Mas09, Kan09, Jed11, Hub11]

1.3. Structure of the Book

The problem formulation outlines the three issues of model derivation, model extension to parameter variations, and model application. Generally, it is not possible to consider each of the problems individually. Analysis methods depend on the simulator which is used as well as on the models upon which the simulator is built. Similarly, it is not possible to discuss cell models without considering the way in which they are generated or how they are later used in a simulation. Over the years, many models and simulators have been derived for different purposes. Some handle parameter variations, others do not. For each issue, there has been many work published. Putting all this into a single review chapter results in a labyrinth of different aspects which is more likely to confuse than to enlighten. That is why this book does not obey the typical structure of problem formulation, review on existing methods, own contribution, and results. Instead, it is structured into the following five chapters. Chapter 2 introduces standard cells and different modelling aspects, including remarks on standard cell characterisation. Section 2.3 specifically focuses on the evolution of delay models. Thereafter, Chapter 3 presents several ways of simulating (digital) integrated circuits with the focus on timing verification. Section 3.1 explains analog circuit simulation in more detail. This knowledge is needed for understanding current source models in Chapter 4 and the CSM simulator in Sec. 5.3. Once the related simulation methods and latest timing models have been presented, the new current source models are discussed in Chapter 4. Section 4.5 is devoted to own contributions of the new systematic modelling approach as well as the new CSMs. The application of CSMs are described in Chapter 5. While Sec. 5.2 presents own experience with integrating CSMs into existing circuit simulators, Sec. 5.3 presents a newly developed CSM timing analyser. The results for the new CSMs and the simulator are given in Chapter 6.

Reality is frequently inaccurate.

Douglas Adams, "The Restaurant at the End of the Universe"

2 Standard Cell Modelling

2.1. Standard Cells

Standard cells (SCs) consist of a small number of transistors to implement basic logic operations like AND, XOR, MULTIPLEXER, or, FLIP-FLOP. They are, together with macro cells and IO cells, the building blocks of semi-custom ICs. SCs are layouted individually and are provided to IC design teams to realise digital modules in IC products. SCs have a common height which allows abutting of cells as shown in Fig. 2.3. To implement complex logic functions, only the electrical connection between the SCs must be done. Typically, several SCs exist for the same basic logic function. They differ in terms like area, power, or delay and hence allow synthesis tools to find an optimal selection to meet design constrains.

Figure 2.1 shows a detailed layout view of a small CMOS NOR cell. PG pins, *dd* and *ss*, at top and bottom are used to connect the cell to power and ground rail, respectively. A pull-up network (PUN) of PMOS transistors provides a charging path from the positive supply rail, *dd*, to the output pin, *o*. Similarly, a pull-down network (PDN) of NMOS transistors provides a path for discharging the output pin into ground, *ss*. PUN and PDN with one common output form a channel connected block (CCB). It is also called stage or inverting block. SCs can consist of several abutted stages, as in the case of ANDs or BUFFERs. Usually, a CCB contains source-to-drain connected transistors, either in the PUN or in the PDN. This serial connection is denoted as transistor stack.

2.2. Cell Views and Libraries

SCs are provided to the IC designer in the form of standard cells libraries. To reduce memory requirements and allow for abstraction, only specific cell features are stored

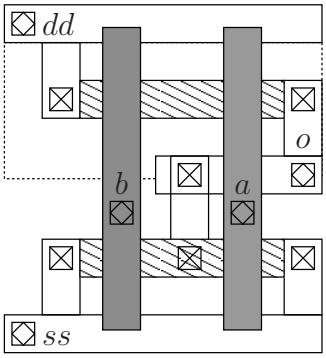


Figure 2.1.: Layout of NOR cell

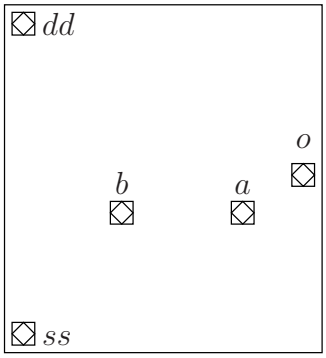


Figure 2.2.: NOR Abstract view

```
1 .SUBCKT NOR a b o dd ss
2 M1 net2 b dd dd pmos
3 M2 o a net2 dd pmos
4 M3 o b ss ss nmos
5 M4 o a ss ss nmos
6 .ENDS
```

Listing 2.1: Electrical netlist

```
1 architecture anor of nor2 is
2 signal aorb:std_logic;
3 begin
4     aorb <= a OR b;
5     o <= NOT aorb;
6 end anor;
```

Listing 2.2: Behavioural netlist

in the corresponding library. The physical view is identical to Fig. 2.1. It is used for the final mask data and hence has all geometric information on every wire segment, contact, and doped area. A typical format which conveys this layout information is the Library Exchange Format (.lef). An example of this human-readable format is given in the appendix at page 99 in Sec. A.1. It additionally has an abstract view [Bha05]. Figure 2.2 shows a graphical representation of the contained data. It is reduced to cell shape and pin locations because this is the required information for Place and Route.

Similarly, further views with much higher abstraction are created. Verification and library characterisation engineers require an electrical cell view. This is a transistor netlist as shown in Listing 2.1. This view can also contain passive elements that model parasitic capacitances and resistances. The more abstract logical view of Listing 2.2, intended for gate level simulation, explicitly contains the logic function and reduces the number of pins to the signal pins. It is written using HDLs like Verilog or VHDL.

At many stages of the design flow even an electrical view is not suitable because there is no electrical simulation. At those levels, cell performances such as delay or logic function are required but they are only implicitly modelled by a transistor netlist. Therefore, these characteristic values of SCs are measured in an electrical simulation of the transistor netlist, and are stored in the Liberty format (.lib) [Liu]. It contains information on cell type, logic function, area, delay, as well as switching and leakage power. An example is given in Listing 2.3. For each pin the direction and type is given. This distinction cannot be made for electrical pins in a transistor netlist. For output pins, the logic function as well as the timing and power consumption are provided. Delay and switching power are related to the corresponding active input pin. It might also be dependent on the logic values of other pins, denoted as side inputs.

This .lib modelling with input and output pins reflects the directed signal propagation through a logic cell, and eventually through a digital circuit. The means to express this propagation is the timing arc. According to [Isf], a timing arc is defined as

Definition 1. *"[A timing arc is an] abstract representation of a measurement of an interval between two points in time during operation of a library cell."*

```
cell (NOR2) {
  cell_leakage_power : 0.084 ;
  area : 2.5621 ;
  pin(a)  {
    direction : input;
    capacitance : 0.0034;
  }
  pin(b)  {
    direction : input;
    capacitance : 0.0036;
  }
  pin(o)  {
    direction : output;
    max_capacitance : 0.0851;
    function : "(!(a+b))";
    timing() {
      related_pin : "a";
      timing_sense : negative_unate;
      cell_fall(delay_template_7x7) {
        index_1 ("0.0052, 0.0411, 0.0568, 0.0846, 0.4701, 0.5188, 0.7056");
        index_2 ("0.0046, 0.0059, 0.0081, 0.0475, 0.0557, 0.0720, 0.4116");
      values("0.0090, 0.0427, 0.0498, 0.0556, 0.0640, 0.4457, 0.2250", \
          ...
          "0.0089, 0.0218, 0.0508, 0.0945, 0.4555, 0.2164, 0.5810");
      }
      fall_transition(delay_template_7x7) {
        ...
      }
      cell_rise(delay_template_7x7) {
        ...
      }
      rise_transition(delay_template_7x7) {
        ...
      }
    }
  }
}
```

Listing 2.3: Example of .lib standard cell

In [Liu], a different formulation is chosen:

> "A combinational timing arc describes the timing characteristics of a combinational element. The timing arc is attached to an output pin, and the related pin is either an input or an output."

This formulation expresses more clearly the possibility to define different propagation paths for different combinations of input pins. Figure 2.4 visualises this concept by overlaying rising and falling timing arcs onto the NOR cell. The timing arc $\langle \text{a0}\rightarrow\text{o}\rangle^r$ denotes the propagation from input a to a rising signal at output o, when input b is zero. For each timing arc, there is an attached timing model. This separation allows to exchange the timing model while the software interface for accessing library cells and timing arcs remain untouched. An overview on existing timing models is given in Sec. 2.3.

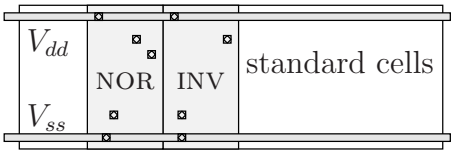


Figure 2.3.: Abutted standard cells are placed in rows

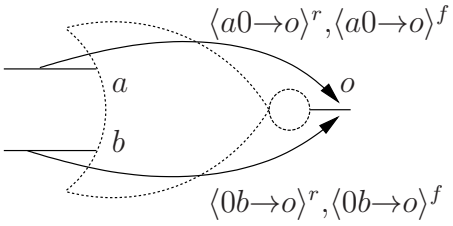


Figure 2.4.: Combinational timing arcs for NOR cell

2.3. Delay Models

Delay models are needed to express the time which is required for a signal to propagate through a standard cell or a digital circuit. The abstract logic signals correspond to physical voltage levels. A net x has a logic value of "1" when its voltage, $V(x)$, is greater than typically 70 percent of the supply voltage V_{dd} [Wak05]. It is logically "0" when it is less than 30 percent of V_{dd}. In the case of a signal change, the net voltage makes a transition from V_{ss} to V_{dd} or vice versa.

Definition 2. *Arrival time T_x is a simple model of a value change of signal x. It is usually defined as the point in time when the node voltage V_x equals 50 percent of the supply voltage, V_{dd}.*

Based upon this definition, the delay D is defined.

Definition 3. *The delay D_{xy} is the difference of the arrival times T_y and T_x.*

The delay can be defined between any two electrical nodes. Typical nodes are input and output pins of standard cells. This allows to store a delay value for a particular timing arc of a SC in the timing library. Algorithms for the delay calculation, which are discussed in Sec. 3.5, can simply add these delay values to obtain the overall delay of a digital circuit.

Physically, the SC delay has internal and external reasons. The first internal cause is the time needed by transistors to switch from conducting to isolating state and vice versa. The second is internal as well as external. After opening a conducting path through PUN or PDN, it takes a finite amount of time to charge/discharge the output net. Figure 2.5 displays input and output waveform of a logic cell. Clearly visible is the charging behaviour, which is similar to the step response of an RC lowpass. There is also a voltage undershoot for V_o, which is caused by charges from the quickly rising input signal [Hua+10].

Different approaches to model the SC delay evolved over time and will be presented in the remainder of this section. It is pointed out in [BJV06], that there is a constant need of improving timing models to be sufficiently reliable, which usually means to have errors of less than 10 percent compared to transistor level simulation. Consequently, the presentation mostly follows a chronological order. The majority of delay models are voltage response models (VRMs) [KTK08]. They explicitly yield the output voltage

waveform, mostly in a very abstract form, as a result of input voltage and load conditions. According to [BJV06], they are "heuristic" models, which means that they are obtained through a pre-phase in which the required model data is gathered. Typically, this pre-phase consists of many electrical simulations[1].

Fixed Cell Delays

In early designs, path delays were dominated by cell delays and interconnect delay was negligibly small. Furthermore, cell delays were roughly equal and hence the number of stages in the longest (sensitisable) path defined the overall delay [SLM06]. Consequently, a single delay value was sufficient to describe the whole cell library. Despite the existence of more sophisticated delay models today, such simple models which assign a single fixed delay value to each cell type is still in use in early design phases. Of course, the resulting accuracy is low. [KO95] reported typical errors of 20–50 percent.

Load Dependent Delay Models

Changing design objectives of standard cells allowed to provide more area and power efficient solutions. Consequently, cell delays were no longer constant but now depend on cell type as well as the size of their driven output load. The "logical effort" is a reasonable model for a quick delay estimation [SSH99]. It takes into account the cell structure as well as internal and external capacitances. The load capacitance was dominated by input capacitances of fanout cells. [BJV06] denotes these models as "static", because each cell delay depends only on the above-mentioned parameters, whose values only depend on fixed topological information. Since these models are independent of shape of the input waveform, step functions were used to model the voltage signals.

Slew rate – Load – Models

The continued downscaling also lowered the supply voltage to mitigate the increase in field strength at the transistors. This led to the abandonment of step functions as signal models. The new waveform model uses the slew rate, also denoted as transition time, to describe the finite time needed for switching the logic value.

Definition 4. *Transition time, τ, is the difference of time points at which a signal crosses two voltage levels.*

As shown in Fig. 2.5, typical values for the voltage levels are 30 and 70 percent of V_{dd} [Wak05]. Thresholds for transition time and delay must be chosen carefully to avoid negative delays, since STA is based upon finding longest and shortest paths in a graph. Negative edge weights complicate the calculations significantly [DP96, Bla05]. Cell delays might be negative in the case of very slow input transitions and small output loads. The resulting fast transitions at the cell output can cross the delay voltage threshold prior to the input.

[1] The alternative is an analytical model which does not require simulation data.

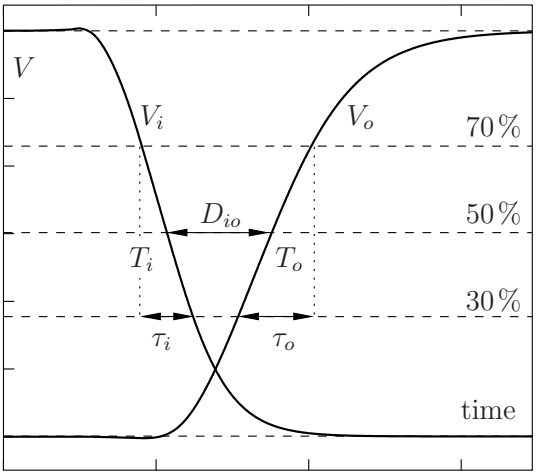


Figure 2.5.: Input and output waveforms with voltage thresholds, delay, and transition time

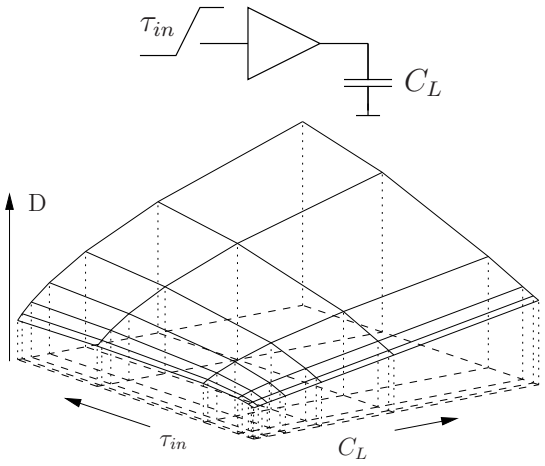


Figure 2.6.: NLDM lookup table for delay index by output load, C_L and input transition time, τ_{in}

This input transition time affects how quickly the transistors are turned on. It also influences the signal overshoot through dynamic coupling. Such waveform-sensitive models are denoted as dynamic models [BJV06]. The most widely used is the nonlinear delay model (NLDM). It consists of two-dimensional lookup tables (LUTs) which store cell delay and output transition time [Koe69]. Figure 2.6 shows an example. The LUT indexes, output load, C_L, and input transition time, τ_{in}, are usually not equally spaced. A truncated textual example is given in Listing 2.3.

NLDM libraries require a significant amount of memory to store the LUT data for many timing arcs. The scalable polynomial delay model (SPDM) therefore approximates the LUTs by suitable polynomials. The SPDM consists of a sum of base functions, ϕ, which are weighted by model coefficients, k_i.

$$D = \sum_{i}^{n_k} k_i \phi_i(\mathbf{x}) = \sum_{i}^{n_k} k_i \prod_{j=0}^{n_x} x_j^{j_i} \quad \text{with} \quad j_i \leq m_x \tag{2.1}$$

These base functions are polynomials in terms of the n_x model parameters, x_j, with a maximum order of m_x [Abe03]. This framework of (2.1) also allows to model the delay dependence on process parameters by including them into the vector of model parameters.

Effective load capacitance

The described delay models assume the presence of a purely capacitive load. However, naive scaling of feature sizes increases the wire resistance. While this is mitigated by increasing the interconnect height, via resistance does not scale well. In sum, interconnect resistance is not negligible and creates large challenges for delay models. One large concern is resistive shielding, which reduces the total capacitance "seen" by the driving cell.

The concept of the effective capacitance, C_{eff}, accounts for this effect [QPP94, CPO04, AP03, NL05, HKIM05]. It matches the average current injection into the output net such that original output waveform and effective waveform have identical arrival times. However, effective capacitance and driver waveform are mutually dependent, which leads to an iterative process to determine stage delays and output waveforms. Moreover, it only matches the cell delay but not the output waveform [BC09]. [FA08] identifies the gap between delay model characterisation and delay calculation during application as the main problem. For them, the effective capacitance is not suitable for accurate timing analysis: "It is clear that traditional effective capacitance based analysis is not able to capture the waveform accurately enough and introduces significant timing errors" [FA08]. However, due to the high investments in training and tools, C_{eff} is widely used as a patch to avoid significant changes.

Effective Current Source Model and Composite Current Source Model

The main accuracy problem of the increased interconnect resistance is that the driver impedance is no longer significantly larger than the load impedance [Kez06]. Therefore, new current-based cell models, composite current source (CCS) and effective current source model (ECSM), are proposed by EDA vendors to replace NLDM. They account more naturally for the current source behaviour of transistors than voltage sources do for driver modelling [Syn06, Cad07, GK05]. They produce more precise output voltage waveforms from current waveforms which are stored in lookup tables. Figures 2.7–2.8 show examples of such LUTs. The LUT indexes are output load and transition time, which makes CCS and ECSM compatible to the existing NLDM and the .lib format. In fact, ECSM does not even store current waveforms but the time points of crossing several voltage levels [Fel+08], which are converted to output currents through

$$i_o \approx C_L \cdot \frac{V(t_{n+1}) - V(t_n)}{t_{n+1} - t_n}. \tag{2.2}$$

While this allows more precise driver modelling, the waveform at the far end of the interconnect is then reduced to a simple ramp signal for the LUT. This results in severe timing errors [LVFA09]. Furthermore, these models still rely on the effective capacitance model [MKA08].

Academia proposed waveform-independent models, also named CSMs. [MKA08] denote this "second generation" of CSMs as "truly electrical". They abandon the concept of forward propagation of waveform parameters, as in the case of VRMs. Instead, CSMs provide an output current for every combination of input voltage(s) and output voltage. Output waveform and cell delay are obtained implicitly during delay calculation. Hence, CSMs require some kind of Spice-like simulation (see Sec. 3.1). The waveform can be modelled accurately by a series of time voltage tuples. A detailed description and overview on different approaches is provided in Chapter 4.

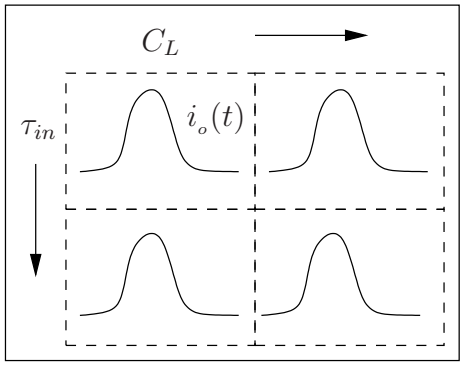


Figure 2.7.: CCS lookup table

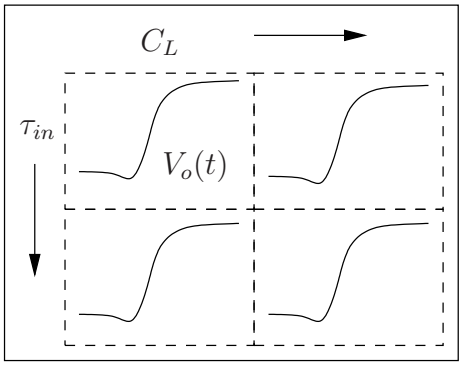


Figure 2.8.: ECSM lookup table

Analytical Models

In contrast to simulating individual cells and storing measured delay values, analytical models are based upon simplified transistor models from which delay expressions are derived [BJV06]. The principle is to build a formula such as (2.3), and solve it for the cell delay, D_{io}.

$$|V_o(T_i) - V_o(T_i + D_{io})| = \frac{1}{2}V_{dd} = \int_{T}^{T+D_{io}} f\left(C_{load}, \tau_{in}, \mathbf{p}\right)dt \tag{2.3}$$

The model parameters, $\boldsymbol{p}$, are obtained through Spice simulations of individual transistors or reference cells. The concept of most approaches is to model the voltage trajectory across the cell's output current [CPZ97]. Usually the coefficients of simple MOS transistor current equations, which are based upon a power law, are fitted to the cell under consideration [SN90]. Provided there is a linear ramp input signal and a purely capacitive load, explicit delay equations can be derived. Nonetheless, these approaches usually only work for simple cells like inverters without parasitics [Hua+10, RS04]. More complex cells have to be mapped to an effective inverter [HOT98]. Analytical cell delay models never gained much popularity in industry. This is due to their complexity, limited applicability to industrial cells, and additional inaccuracy compared to NLDM.

Receiver Models

Almost all delay models use the parameter output load, C_L. Since C_L accounts for the attached standard cells, their input capacitances must be estimated. Similar to the delay models, also the receiver models has continuous accuracy improvements. An early and simple approach is static and topology-based. It consists of adding all capacitances at the cell input [KTK08]. However, this ignores the Miller effect [BHSA03]. The problem is, that the sum of capacitances includes gate-drain-capacitances, which account for $2CV_{dd}$ charges flowing into the gate, since there is also a full swing voltage difference at the output node. Furthermore, this method ignores the influence of parasitic elements. Consequently, the most common way to determine the input capacitance is based upon

simulation. In the load-matching method, a driving cell is attached to the input pin and its delay is measured. Then, the driven cell is replaced by a capacitor and it is sized to have equal delay. The other method is to apply a ramp voltage directly to the input pin and to integrate its current over time.

$$C_i = \frac{\Delta Q}{\Delta V} = \frac{1}{\Delta V} \cdot \int i_{V_{IN}}(t)dt \tag{2.4}$$

The discussion above implies that the input capacitance is not independent of the cell's output load. Due to capacitive coupling inside a cell and the nonlinearity of the transistor capacitances, the input capacitance further depends on the input slew rate. This leads to similar NLDM lookup tables for C_i. Nonetheless, this dependence is weaker than the one for the cell delay. Therefore, only minimum and maximum values are provided in the libraries. ECSM and CCS allow to define two voltage- or time-dependent input capacitances [Kez06]. Therefore, the voltage source is replaced by a current source which charges the cell input and the resulting voltage difference is measured.

2.4. Characterisation

The process of characterisation starts after the required standard cells haven been identified and designed [SMSB08]. It consists of generating different views, VHDL or Verilog descriptions, electrical netlists with and without parasitics, as well as libraries for power, timing, and noise models. The last step typically extensively uses analog electrical simulation. It can be very time consuming, since each cell must be simulated for various scenarios to obtain reasonable accurate modelling data. Typically this means that a small testbench must accurately represent the occurring signals and environments for the cell in a real integrated circuit. Consequently, although the NLDM in Sec. 2.3 models waveforms only by transition time, this does not imply that characterisation should by conducted using a linear ramp voltage source. Instead, a smoothed transition is the better approximation of the signals that occur in an IC [Joh09]. A longer discussion with more information on highly accurate waveform models is given on page 31 in Sec. 3.5.

One constantly faces the challenge of designing and verifying tomorrow's ever more complex and ever faster hardware with the aid of today's computers.

Pillage, Rohrer, and Visweswariah

3 Simulation of Integrated Circuits

Due to the high cost and long turn around times of fabricating hardware prototypes, designing integrated circuits (ICs) heavily relies on simulation of IC models. At each design phase, different analyses and optimisations are performed to ensure that design constraints are met. This also includes verifying by simulation that the design is functionally correct, which means that the designed logic function fulfils the specification. Furthermore, simulations are used to ensure that the IC also operates correctly. This includes verifying that a target frequency can be reached, that power consumption is within specified limits, or that crosstalk of neighbouring wires does not cause malfunction. Many of these concerns are only analysable, or of interest, at some design phases. Consequently, there are many different methods and models for the IC simulation. This chapter provides an insight into various analyses. After introducing the general electrical simulation, the focus is on analysis methods for the timing properties of digital ICs.

3.1. Transient Electrical Simulation

Electrical simulation calculates physical quantities like node potentials, currents through elements, charges, or fluxes in a circuit as functions of time. Based upon their values, other quantities like power consumption or path delay are derived. It is the lowest level of abstraction used by circuit designers, and therefore its results are usually regarded as golden references [BJV06]. Simulators of this kind are typically named "Spice" simulators, although or because it is the name of one of the first implementations [NP73, MP71].

The typical representation of an electrical circuit is the netlist as shown in Fig. 3.1. It describes the connection of network elements. Each element is described by its model which expresses the voltage and current relations of its pins. For example, a simple

```
1 V1 (in 0)     PULSE (0 2.5 1n)
2 R1 (in a)     100
3 C1 (a 0)      10e-13
4 G2 (b 0 a 0)  -0.01
5 C2 (b 0)      100e-15
6 R2 (b 0)      100
```

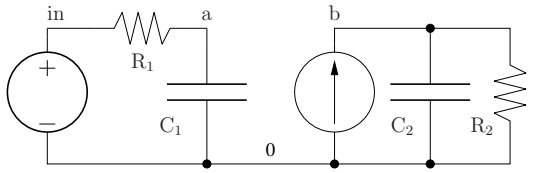


Figure 3.1.: Spice netlist ans schematic for electrical simulation

model for the resistor R_1 in Fig. 3.1 would be

$$I_{res,in} = \frac{1}{R_1} \cdot (V_{in} - V_a) \tag{3.1}$$

$$I_{res,a} = -\frac{1}{R_1} \cdot (V_{in} - V_a) \tag{3.2}$$

More complex devices like diodes or transistors might consist of several basic elements like resistors, capacitors, or voltage-controlled current sources (VCCSs). Many of these components are modelled by nonlinear functions of device parameters and node voltages. Compact transistor models consist of many equations to calculate internal variables such as surface potentials, but return only the resulting currents and charges [SH68, Bmm].

The electrical simulator uses the circuit topology to combine the model equations into a system of nonlinear differential algebraic equations (NLDAEs). Different possible formulations exist, of which the compact modified nodal analysis (MNA) is widely used [DK69]. Each equation represents one electrical node for which Kirchhoff's current law (KCL) requires the net current sum to be zero. Further equations are added for currents through voltage sources and inductances.

$$\boldsymbol{F}(\boldsymbol{V}, t) = \boldsymbol{0} \tag{3.3}$$

Since MNA uses an admittance formulation, the vector of circuit variables, $\boldsymbol{V}$, contains node potentials and currents through voltage sources and inductances. Equations 3.4–3.7 are the NLDAEs for the example of Fig. 3.1.

$$V_{in} - V_1(t) = 0 \qquad \text{(from pulse voltage source V1)} \tag{3.4}$$

$$\frac{1}{R_1}(V_{in} - V_a) + i_{V_1} = 0 \qquad \text{(KCL in)} \tag{3.5}$$

$$\frac{1}{R_1}(V_a - V_{in}) + C_1 \frac{d}{dt} V_a = 0 \qquad \text{(KCL a)} \tag{3.6}$$

$$-0.01 V_a + C_2 \frac{d}{dt} V_b + \frac{1}{R_2} V_b = 0 \qquad \text{(KCL b)} \tag{3.7}$$

The remaining task is the mathematical problem of solving a first order NLDAE.

Solving nonlinear differential algebraic equations

Solving an NLDAE consists of two nested loops as shown in Figs. 3.2 and 3.3. In the outer loop, the continuous simulation time is broken into discrete time points. The

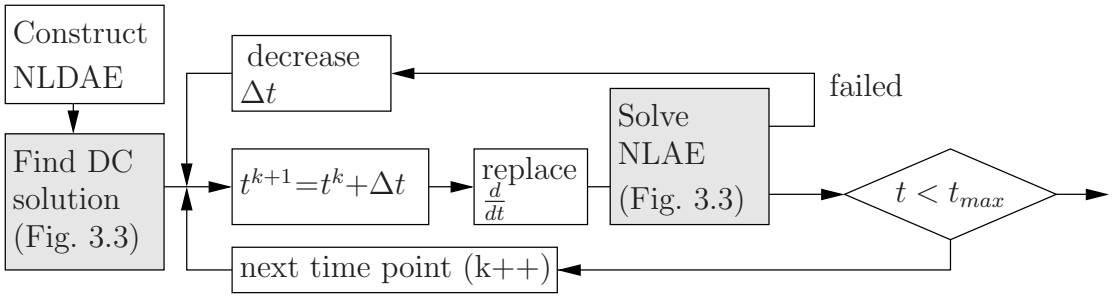


Figure 3.2.: Transient electrical (analog) circuit simulation

time derivatives are replaced by approximations which only use these time points. The most common integration methods are Forward Euler, Backward Euler, and trapezoidal integration. Most Spice simulators use one-step methods, which calculate the time derivatives based upon the values at two time points and the time step $h = t_{i+1} - t_i$. Typically, Forward Euler integration is used at the beginning of the transient simulation and is then changed to trapezoidal approximation.

$$\text{Forward Euler} \qquad \dot{V}_i \approx \frac{1}{h} \cdot (V_{i+1} - V_i) \tag{3.8}$$

$$\text{Backward Euler} \qquad \dot{V}_{i+1} \approx \frac{1}{h} \cdot (V_{i+1} - V_i) \tag{3.9}$$

$$\text{Trapezoidal} \qquad \dot{V}_{i+1} \approx \frac{2}{h} \cdot (V_{i+1} - V_i) - \dot{V}_i \tag{3.10}$$

By replacing the derivative operator, $\frac{d}{dt}$, in the circuit equations (3.4)–(3.7) with the appropriate calculation of (3.8)–(3.10), the NLDAE is converted to a nonlinear algebraic equation (NLAE). This NLAE must be solved for each time point. Different methods exist for this purpose. The most common is the Newton-Raphson-Algorithm (NR), which is shown in Fig. 3.3. It is a gradient-based method, which repetitively solves the linearised algebraic equations (LAE) for a correction of the approximated solution, $\Delta \boldsymbol{V}$, of the NLAE [Ste95].

$$\frac{d\boldsymbol{F}(\boldsymbol{V})}{d\boldsymbol{V}} \cdot \Delta \boldsymbol{V} = \boldsymbol{J}(\boldsymbol{V}) \cdot \Delta \boldsymbol{V} = -\boldsymbol{F}(\boldsymbol{V}) \tag{3.11}$$

The solution is accepted when the error for the circuit equations is less than a tolerance value. If this cannot be reached within a number of iterations, the time point is rejected and the time step must be reduced. NR is widely used for Spice simulators because calculating the Jacobian matrix, $\boldsymbol{J}(\boldsymbol{V})$, consisting of the partial derivatives, can be avoided. Instead, each network component inserts a specific stamp into the matrix and the right hand side (RHS) of (3.11) [VS94].

In brevity, these are the required steps to perform transient analog circuit simulation. Nonetheless, many detailed issues must be considered to make it work. This includes further essential functionality such as output, post process, consistency checking, exception handling, parameter variation, and maybe distributed computing. Only a simulator

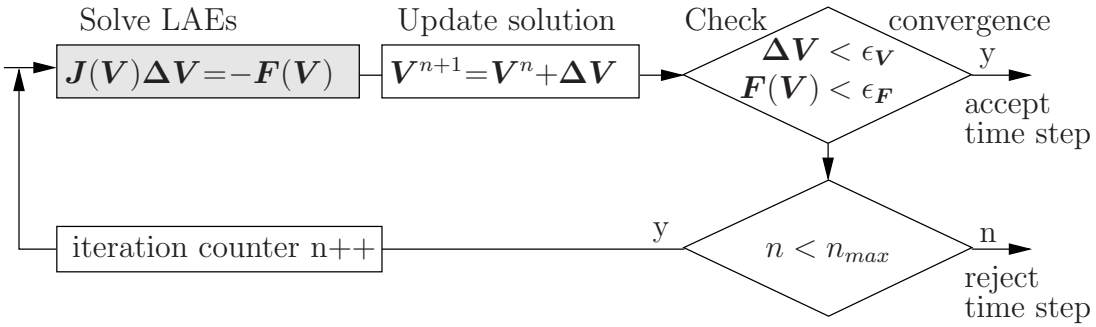


Figure 3.3.: Solving NLAEs by Newton-Raphson-Algorithm

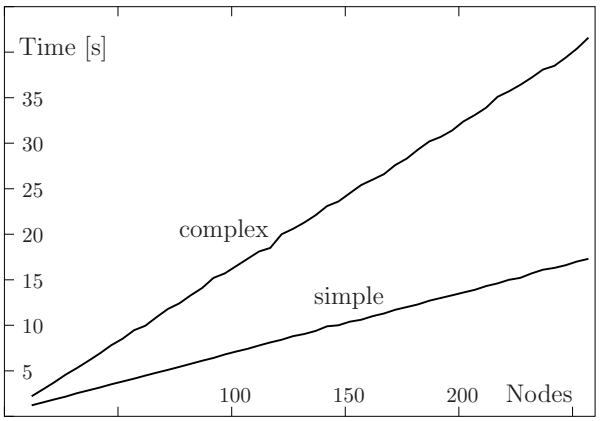


Figure 3.4.: Comparison of Spice runtimes for different transistor models

with this capabilities can be used in a design flow. It is therefore highly beneficial to extend existing tools by including new models or algorithms, if possible.

Spice's largest drawback is its complexity. For early simulators, the maximal number of nodes and transistors was restricted by the available computing resources, especially the available memory. This problem is practically solved for modern computers. Instead, the required times for component evaluation and solving the NLDAE pose practical limits on the applicability. Simulating a few milliseconds of a small design with about 2300 transistors can already take several dozens of hours.

3.2. Timing Simulators

Spice circuit simulation, as described in Sec. 3.1, is not appropriate to estimate the path delays in complex digital circuits. Its strength is the high accuracy which is usually far beyond the requirements for timing simulation. Moreover, the price to pay is the unacceptable runtimes for transient simulation. The general problem is in the large dimension of the equation system and the large number of required time steps. Consequently, electrical simulators have been invented which can provide delay estimates by simpler device models and modified simulation algorithms.

Various analyses show that model evaluation accounts for about 75 percent of simulation time [GCKS09]. Hence, it is reasonable to replace complex transistor models by much simpler ones. For instance, a large number of equations, containing the calculation of logarithms or exponentials, are replaced by lookup tables, piecewise linear, or piecewise constant functions [PRV95]. Nonlinear floating capacitors are approximated by grounded linear capacitors. Figure 3.4 compares the runtimes of simulating inverter chains of different lengths using a complex and a simpler transistor model. Despite obvious performance gains, the speedup is only around 2.4 x. Since model simplifications affect only one part of the simulation algorithm, this measure has an asymptotic gain of 4 x, which is not sufficient [GCKS09]. Therefore, additionally to accelerating model evaluation, exploiting spatial and temporal latency allows to reduce the number of component evaluations and the system size [PRV95]. Spatial latency refers to the fact that at any given time, most regions of the circuit have constant signal values and are therefore not active [MME86]. Moreover, these portions themselves will create signal changes in a small part of the system. Hence, the circuit can be partitioned into different subcircuits which are analysed in a decoupled manner [PRV95]. Temporal latency refers to the model perspective and denotes time-variant accuracy requirements. That means, a less accurate but faster cell model can be used when the cell instance is not active.

The partitioning of large circuits leads to two further benefits [Cad04b]. First, solving equations systems of fewer dimensions is faster because of the superlinear scaling factors. Second, each subcircuit can use an individual time step whereas normally the smallest RC determines the time step for all components.

Different timing simulators, which are briefly discussed, have been developed on the basis of these concepts. A detailed dissection can be found in [PRV95] or [KO95].

MOTIS [CGK75] uses simplified voltage-to-current characteristics, stored in a 2D lookup table (LUT) of (3.12b), together with a 1D LUT for the threshold voltage.

$$V_{th} = \mathcal{F}_1(V_{gb}) \tag{3.12a}$$

$$i_{ds} = \frac{W}{L} \cdot \mathcal{F}_2(V_{gs} - V_{th}, V_{ds}) \tag{3.12b}$$

Current tables from pull-up network and pull-down network are merged before the event driven simulation.

SAMSON [SD85] defines two accuracy levels for MOSFET models. A transistor in alert state is currently switching and hence an accurate Spice device model is used. In dormant state, model evaluation is changed to extrapolation of the previous solution.

ELogic [KHN89] is an event-driven simulator with a set of discrete voltages and piecewise linear MOSFET model. Instead of solving the voltages for a number of time points, ELogic computes the time points for transitions from one state to another.

SPECS [VW93] uses piecewise constant transistor currents, which are implemented as lookup tables. It also solves circuit equations for new events. An event happens when the transistor voltage moves from one table entry to the next, resulting in different currents. The time point of the new event is calculated as the time when this table entry will be left.

[Cad04b] provide an overview of the achievable performance gains. According to their studies, event driven simulation gains 2–100 x, whereas table models result in 5–20 x speedup. These empirical values are greater than the predicted limit of 4 x, because they include secondary effects. For instance, simplified transistor models often lump and linearise capacitances, which results in larger time steps and hence fewer model evaluations, convergence checks, matrix operations, and data output.

Devgan and Rohrer developed the adaptively controlled explicit simulation (ACES), which is the basis for commercial FastSpice simulators [DR93, DR94]. ACES operates in the derivative space, i.e. the system variables are dV/dt. Based upon the state space representation, it determines the next time step such that at least one variable reaches a quiescent state. That means, the time point for which the time derivative of this state variable is zero. This method leads to a significant reduction of time points.

Finally, using standard cells implies that a circuit contains several instances of the same subcircuit. The newer high-capacity simulators exploit this fact in an isomorphism hierarchy simulation. If different instances observe about the same port voltages, they are denoted as being in the same state. Consequently, these cells will behave equally, and it is legal to simulate only one instance and reuse the results for the other instances.

The approach of [ADP02] is more similar to a normal Spice simulator. Table models are used for MOSFETs currents and nonlinear MOS capacitances are set to a constant value. They employ Quasi NR method to skip evaluation time of Jacobian entries. Speedup for the large RC interconnects is obtained by reduced order interconnect model, and latency is exploited by a stagewise simulation.

3.3. Switch Level Simulation

Continued simplification of the transistor model resulted in less accuracy for voltages and currents. Consequently, also more abstract signal models are used. Starting with switch level simulation, no longer voltages and currents, but logic values and delays are calculated. At switch level, transistors are modelled as ideal switches with a series resistor. The transistor is either open, closed, or indeterminate during the switching event (see Figure 1.1). Consequently, nodes have logic values of “1”, “0”, or “X” [Mic87]. The value of an output node of a CCB is determined by comparing pull-up and pull-down strengths of the resistors [BJV06]. One switch level simulator is MOSSIM [Bry81]. The authors point out, that “no attempt is made to model the actual speed of the circuit”. Other simulators include timing estimations in one of two ways. The first is to perform Spice simulation on the simplified small subblocks, and then annotate the measured delay values. The other is in using the Elmore delay for the resulting circuit, which consists of grounded capacitors connected by resistors [Elm48].

The resistor values to model the transistors are obtained from LUTs storing the effective resistance as a function of rise time ratio. The latter is the quotient of the cell’s intrinsic rise time, that is the delay for a step input, to the intrinsic rise time of its load [Ous84]. Each transistor type has a separate LUT for rising and falling.

Timing analysers like Crystal [Ous85] then calculate the delay of a CCB by the lumped

RC model of (3.13) or the Elmore delay of (3.14).

$$D \approx \sum R \times \sum C \tag{3.13}$$

$$D \approx \sum_i R_i C_i \tag{3.14}$$

More information on switch level simulation can be found in [KO95].

3.4. Logic Simulation

Integrated circuits can be simulated at higher levels, which are at least partially decoupled from the physical hardware. The modelling levels cover gate level, RTL, system level and finally algorithmic level. In this book only gate level simulation is discussed. More information on higher levels can be found in [SLM05].

For logic simulation, each cell is modelled by a Boolean function and a propagation delay [BJV06]. Consequently, the basic signal values are "1", "0", and "X". This modelling suits the two main reasons for performing logic simulation at gate level [MME86]. The first is to verify functional correctness of a design which is given as a synthesised digital component. It is written using an HDL to represent the standard cells in their logical connection. The HDL is compiled into C-code which is then executed. Since the intention is to verify the logic, usually zero delay cell models are used [KO95].

The second purpose is in dynamic timing analysis (DTA). Here, each SC has a delay value, and the IC is simulated for a set of input vectors. Listing 3.1 shows a Verilog delay model. Different values are given to model best, typical, and worst case delays for rising and falling transitions.

```
1 xor #(2:3:4,1:2:4) DUT (o,a,b);
```

Listing 3.1: Verilog delay model

More accuracy is achieved by backannotating the results from a static timing analysis to cells in the design using the specify statement. This is shown in Listing 3.2 where pin-to-pin delays depend on direction of transition and side inputs. While the delay model in Listing 3.1 typically has separate values for each cell type, backannotation results in different values for each cell instance.

Logic simulation is done in either a compiled, event-driven, or a cycle-based method [SLM05]. In a compiled logic simulation, each cell instance is evaluated at each time step. In contrast, the concept of event-driven is to evaluate only those cell instances that have changing input values. If this leads to a value change at the cell's output, a new event is scheduled which defers this change by the cell delay. This can lead to a large number of events. To increase simulation performance, cycle-based simulation is used. The principle is that everything that really matters in synchronous circuits is the logic value at the registers at the time point of the clock edge. Consequently, if a static

timing analysis ensures that timing is met for a combinational block, it is sufficient to simulate two events; one at the clock edge and one just after to produce the new logic values. Of course, the precise times of the switching events of internal nets are removed.

```
1 xor(o, a, b);
2 specify
3   if ((b == 0'b1))  (a => o) = (2,3);  // delay (rise/fall) for a->o when b=0
4   if ((b == 1'b1))  (a => o) = (4,5);  // delay (rise/fall) for a->o when b=1
5   if ((a == 0'b1))  (b => o) = (6,7);  // delay (rise/fall) for b->o when a=0
6   if ((a == 1'b1))  (b => o) = (8,9);  // delay (rise/fall) for b->o when a=1
7 endspecify
```

Listing 3.2: Verilog delay model through *specify* statement

Simulation at higher levels is very similar. The main difference is the communication between modules. Instead of using bit vectors with specific width, more abstract data types like integers or messages are used. This reduces the amount of events and data [SLM05].

3.5. Static Timing Analysis

Static timing analysis (STA) is a vectorless analysis to obtain a conservative estimation for the maximum operating frequency of a digital circuit. The gate level description is transformed into a timing graph [Sap04]. It contains no logic functions but only timing information.

Working on timing graphs naturally exploits spatial and temporal latency of digital circuits and allows for massively parallel processing [Kal11]. Practical limitations are discussed in [BC09].

Definition 5. *A timing graph is a directed acyclic graph (DAG) in which edges represent cell and/or wire delays and nodes correspond to important locations in the circuits.*

Such important locations are IO pins of blocks or standard cells. Usually, source and sink nodes are connected to the module ports and can represent external timing constraints. Figure 3.5 shows a digital circuit in which dashed lines indicate electrical nets. Its timing graph is overlaid. It has a virtual sink and source node. Each timing arc represents a delay value which is retrieved from the delay model. It fits very well for designs using standard cells because for those this timing information can be provided (see Section 2.2).

According to [KTK08], an STA consists of three parts: The first consists of the delay models which predict the signal delay along one timing arc. Typically, these are stored in the .lib file (see Sec. 2.3). The second are algorithms for the delay calculation (DC) that work on the delay models. Finally, there are graph algorithms allowing efficient application of DC on a large circuit.

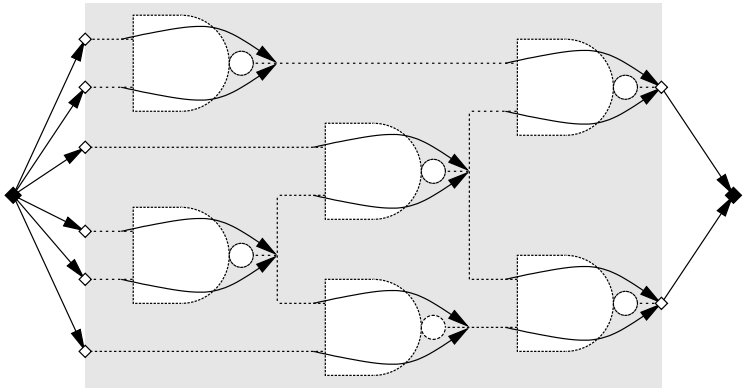


Figure 3.5.: C17 circuit and its timing graph

The task is to analyse timing properties such as the maximum path delay from source to sink. This information is used to check that timing constraints at ports are met, and that there is no hold and setup violation at the flip-flops. Two algorithms exist: In the path-based approach all paths are enumerated and each path delay is calculated as the sum of all edge delays. The critical path has maximum delay [BJV06]. The complexity of this method is exponential in number of nodes.

The alternative is a block-based approach, which is linear in circuit size. At each node only the maximum arrival time is stored and finally yields the latest arrival time at the output pins. A backward traversal computes the slack for each net and identifies the critical path [Hit82].

STA works on timing graphs and hence finds critical paths without consideration of the logic function. This guarantees that the real path delay of any combination of digital input vectors is less than or equal to this critical path delay. Hence, STA yields conservative results. However, ignoring the logic functions also implies that there might be no input pattern such that the signal propagates along the critical path. A false path detection is used after the STA to exclude such spurious path delays.

The accuracy of STA strongly depends on delay and waveform models, which are discussed in Sec. 2.3. A major concern is the abstraction of signal waveform. Therefore, tools like NanoTime [Syn12] or the path delay calculator [SKS08] allow to resimulate paths or blocks on transistor level. Nonetheless, if the circuit information is incomplete, the real accuracy gains are limited. Simulating a digital design on transistor level without any interconnect models mainly increases simulation time rather than accuracy.

Waveform-Accurate Timing

Many authors identified the waveform model as significant contributor to timing error. While undisturbed signal transitions are fairly linear in the midrange of the voltage swing, they exhibit overshoots at beginning and long charging tails at the end. “Due to the low-pass nature of RC loads, the ’digital’ signals exhibit non-digital behaviour, and the performances, especially the delay, become largely affected by the waveshapes.” [DMP96] The authors of [ADI03] reported up to 19 percent mismatch of

cell delays for a real signal and the linear ramp with identical slew rate. Obviously linear ramps are not suited for .lib-characterisation. It is during STA, where only the slew rate parameter is available, that the linear ramp is the only possible waveform. During characterisation more realistic waveforms, such as shown in Fig. 3.6, with the same slew rate can and must be used. EDA vendors suggest to use Sine-Square signal instead of linear ramps [LVFA09]. An alternative is the pre-driver method, shown in Fig. 3.6(a), for which the cell input is attached to the output of another cell [Kez06]. The slew rate can be varied through a grounded input capacitor. Using these waveforms still allows to model a signal by arrival time and slew rate. More precise waveform models have been suggested to better model differences in beginning and end of the transition. [MAO95] for instance presented a signal model with four parameters.

$$V(t) = V_{const} + k_1 e^{p_1 t} + k_2 e^{p_2 t} \tag{3.15}$$

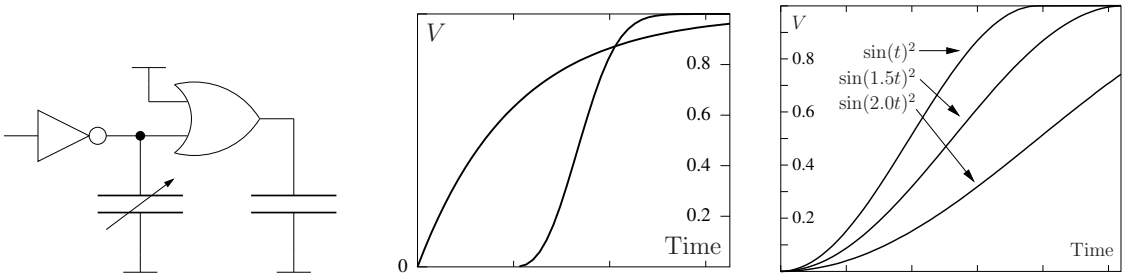
The first exponential term in (3.15) dominates the entire transition whereas the second implements a correction near t=0. Nonetheless, a four parameter waveform model infers cell characterisation with respect to four parameters. Over almost two decades large investments have been made on characterisation flows and companies are not willing to change this [HYO04]. Moreover, moving from two parameters to four results in a significant increase of timing libraries and prohibitive simulation effort for characterisation.

A more suitable waveform model which has been used for many other approaches is the Weibull function of (3.16), shown in Fig. 3.6(b) [ADI03, VSB08]. It requires only two parameters.

$$f(t, \alpha, \beta) = 1 - e^{-\left(\frac{t}{\beta}\right)^{\alpha}} \quad \forall t \geq 0 \tag{3.16}$$

[ADI03] used this model and characterised logic cells in terms of Weibull parameters α and β. Since they are not really intuitive to use, the authors suggest handy parameters transition time and shape factor and convert internally. α and β of real waveforms are obtained through linear regression. The Weibull function originates from statistics where it models a cumulative distribution function (CDF). The similarity of CDF and logic signal has been recognised by several authors. [NA04] showed that usually two parameters (moments) are sufficient to model inverter waveforms. [LVFA09] used four moments.

While CDFs model signal transitions more realistically, they cannot handle overshoots or noise bumps. [KTK08] further demanded for models that can deal with non-monotonic transitions or glitches to account for noise on timing in STA below 130 nm. [JBZ05] therefore suggest to extend the characterisation by using several "typical" base waveforms. They are found by simulating observed waveforms for each cell. Thereafter, these waveforms are approximated by the base waveforms using stretching and time shift. If a certain error bound cannot be met, the set of base waveforms is extended. Finally, for each cell time stretching and time shift values are stored in LUTs which are indexed by base waveforms. During timing analysis, the most suitable output base waveform is found and the factors for shift and stretching are calculated through mean squares. However, [HYO04] showed that such a least square minimum can result in very large errors for delay and resulting output waveform. As alternative, they introduce



(a) Pre-driver (b) Weibull waveforms (c) Sin-squared waveforms

Figure 3.6.: More realistic characterisation waveforms

weights that are based upon the impact of input waveform on the output waveform and thereby find the equivalent input waveform that produces best output waveform.

The procedure in [MS02] determines the worst-case input vectors for each cell in a design. It uses Spice to simulate each cell instance with this vector and then propagates the output waveform. A similar approach, which also accounts for parameter variations, is proposed in [Sch08]. The authors of [BSE08] avoid simulations inside the STA. Instead, they perform waveform-accurate simulation by first simulating various scenarios for each SC and storing not only delay, but also output currents and waveforms. During the circuit analysis, the simulator combines these information of different cells to perform a circuit simulation. All SCs which are not in the library are approximated by finding an equivalent SC based upon the logical effort.

3.6. Varied Parameter Analysis

For every IC, there is a range of operating conditions for which correct functionality must be guaranteed. This means that operating frequency and power consumption must be met for higher or lower supply voltages and for a given temperature range. Additionally to these variations of environmental parameters, there is device variability. This is due to imperfections during manufacturing such as line edge roughness, tolerances for mask alignment, non-uniform illumination, or chemical-mechanical-polishing [Bar+11]. However, there is also unavoidable systematic process variability. Dopant concentration, for instance, fluctuates randomly because implantation is a statistical process. Parameter variation affects circuit performance and can make a chip fail to meet the specification. The products can, if even, be only sold for a lower price. However, achieving high yield depends on the circuit design, and hence methods and tools are needed to analyse the influence of parameter variations. This section, therefore, presents four common and emerging varied parameter analyses.

Monte Carlo Analysis

The Monte Carlo (MC) method is a general algorithm that repeatedly uses randomised input samples to compute desired results [BSMM01]. It is applied to the simulation of ICs by describing distributions of process parameters, voltage, and temperature (PVT) or delay values and then generating random values. Each electrical or timing simulation for one set of parameters is an MC run, and the obtained results represent the feature distribution, for instance path delay. No restrictions are made on the parameter distributions, and the accuracy depends only on the number of MC runs, denoted as n. Equations 3.17–3.18 approximate standard deviation of the estimation errors for mean value, $\sigma_{\overline{x}}$, and standard deviation σ_{s_x} [Qim08]. s_x is the estimated standard deviation for the measurement x.

$$\sigma_{\overline{x}} = \frac{s_x}{\sqrt{n}} \tag{3.17}$$

$$\sigma_{s_x} = \frac{s_x}{\sqrt{2(n-1)}} \tag{3.18}$$

$$n = \left(\frac{2.58 \cdot s_x}{\epsilon_\mu}\right)^2 \tag{3.19}$$

Equation 3.19 yields the required number of runs to achieve a confidence level of 99 percent that the MC approximation of the sample mean has an error of less than ϵ_μ.

The obvious advantages of the Monte Carlo method are its accuracy, generality, and simplicity. No approximations are done for modelling, and no restrictions are made on the parameter distributions. The simulations can be distributed on different computers and run in parallel to reduce simulation time. Furthermore, the number of required MC runs is independent of the numbers of parameters. Instead, the number of MC runs directly relates to the accuracy of the estimate. However, usually designers are interested in the tails of distributions but these samples are rarely drawn. Hence, it requires a large number of simulation runs to achieve sufficient accuracy. Methods to reduce the number of runs are of two natures: importance sampling [OZ00] directly generates samples at the tails of the distribution and then scales the results to the overall distribution. Latin hypercube sampling and Quasi Monte Carlo discretise the parameter space to ensure better coverage and weigh each of the results by its probability [KK99, SR10, VCBS11, Sob67].

Corner Analysis

MC analysis is very time consuming. Furthermore, it requires parametrised libraries to express the cell delay as a function of parameters. Generating these libraries also requires a significant amount of time. However, it has been observed that cell delay is generally a monotonic function of each parameter. Consequently, extreme values of parameters result in an extreme value of cell delay. Therefore, satisfying timing constraints at these corners ensures that there is no violation for any combination of parameters. Historically, there were five delay corners: typical, slow-slow, fast-fast, slow-fast, and fast-slow. To

each corresponds a set of values for process parameters, voltage, and temperature. A separate timing library is characterised for each corner. Additional corners appeared with the increasing importance of interconnects and their variation. A reduced wire cross section leads to higher resistance but also to smaller capacitances. For newer technologies, the monotony of delay dependency is no longer true for all parameters. Especially inverse time dependency requires a separate corner. That is, usually delay increases with temperature as carrier mobility decreases. However, higher temperature also lowers the threshold voltage which results in decreased delay [DH06]. Which effect dominates depends on the supply voltage and driver strength of a cell. In any case, finding the real worst case corner is very complex [SSP07].

Statistical static timing analysis

The corner analysis models a global, omnipresent parameter variation, in which all devices on a die share the same parameter values. This picture is rather unlikely because local parameter variation causes different delays for different instances of the same cell within one die [BM11]. Hence, by setting all cell instances to the worst case values, the corner analysis overestimates the expectable path delay. A more realistic model separates the variation into the aforementioned global and a local part. While still modelling a worst case scenario, local averaging effects are considered. In practice, scaling factors are applied to path delay to account for local on-chip variation (OCV) and hence to reduce pessimism [Web02]. Newer approaches derive the scaling factors from path length and geometrical spacing [Aot]. This level-based and location-based OCV further considers the number of shared cell instances between two paths. The typical application scenario is the clock tree to launch and capture flip-flops. Any delay variation in the first cells will equally affect both paths. Hence, the variability of the clock differences is less than the sum of variability of all cells in the paths. Accounting for this effect is denoted as a reduction of common path pessimism.

In contrast to modelling local parameter variation after the real timing analysis, statistical static timing analysis (SSTA) directly includes the variation model. SSTA is conceptionally identical to STA except that delays, arrival time, and slacks are no fixed numbers but random variables [BCSS08]. Each cell delay is related to the parameter distributions by a sensitivity model such as in (3.20).

$$D = D_{nom} + \sum \mathcal{S}_p^D \cdot RV_p \tag{3.20}$$

The nominal value is amended by a function of random variables (RVs) with zero mean value. The random variables can model global, local, and cell internal random variations of parameters [Vis+04]. Although higher order sensitivities are possible, most SSTA approaches use a linear sensitivity model. Similar to the NLDM sensitivity model, a statistical extension for ECSM is defined in [Ini07]. Each quantity, such as delay, slew, input capacitance, or time points in the output waveform, can be modelled using linear sensitivities as in (3.20). The standard further allows to use second or third order polynomials to capture nonlinearity and interdependence of parameters.

Replacing fixed numbers by random variables implies severe challenges for the delay calculation and timing analysis. First, there is no critical path. Instead, every path has a probability of being critical. This significantly increases complexity [BVB03]. Second, calculating the sum of two random variables is done by convolution of their joint probability density functions (PDFs). To avoid this integral, cell delays are assumed to be Gaussian. Nonetheless, even in this case the correlations between the variables are needed. The RVs are correlated because cells are in physical proximity and the within-die variation causes spatial correlation. Another source of correlation are reconverging paths since both arrival times share delay contributions from identical cells. Third, and most importantly, calculating the maximum of two Gaussian variables results in a non-Gaussian distribution. Consequently, higher moments must be computed and propagated. However, this requires calculating sum and maximum for arbitrary distributions. Alternatively, the resulting distributions can be approximated by Gaussian distributions at the expense of new errors. Finally, compared to MC analysis, SSTA runtime does depend on the number of parameters. To avoid these problems, some authors propose the MC-SSTA algorithm [GK09, Con+10, SSR08]. Despite the name, it is a normal MC analysis using STA. Usually the sample generation is improved by algorithms to cover the parameter space with a reduced number of samples (see page 34). Additionally, the independence and simplicity of the algorithm is exploited on powerful hardware like FPGAs or graphic processors.

Sensitivity analysis

Sensitivity analysis (SA) is a method to determine the influence of parameter changes on some quantities of interest, denoted as measurements. Usually this influence is approximated by a linear sensitivity, S. Consequently, the validity of these sensitivity values is typically restricted to a parameter deviation of about 10 percent. It is always possible to obtain linear or quadratic sensitivities by the perturbation method. For this, a parameter is positively and negatively deflected by some value and the analysis is rerun for these values. Based upon the three parameter values and the three values for the measurement, a linear or quadratic response model is generated. In the method, each parameter influence is analysed individually and cross dependencies are ignored. Generating more complex sensitivity models without simulating every parameter combination can be achieved by methods described in [Mor91]. For the SA of electronic circuits, special methods have been developed. The direct method is efficient in calculating the sensitivities of all measurements with respect to one parameter whereas the adjoint method efficiently calculates the sensitivities of one measurement with respect to all parameters. Most Spice simulators only support DC and AC SA. The authors of [KJA07] point out that the principles of calculating transient sensitivity have been published several decades ago [BS80, DR69, HYTE85]. However, they are hardly available in commercial tools due to their computational costs.

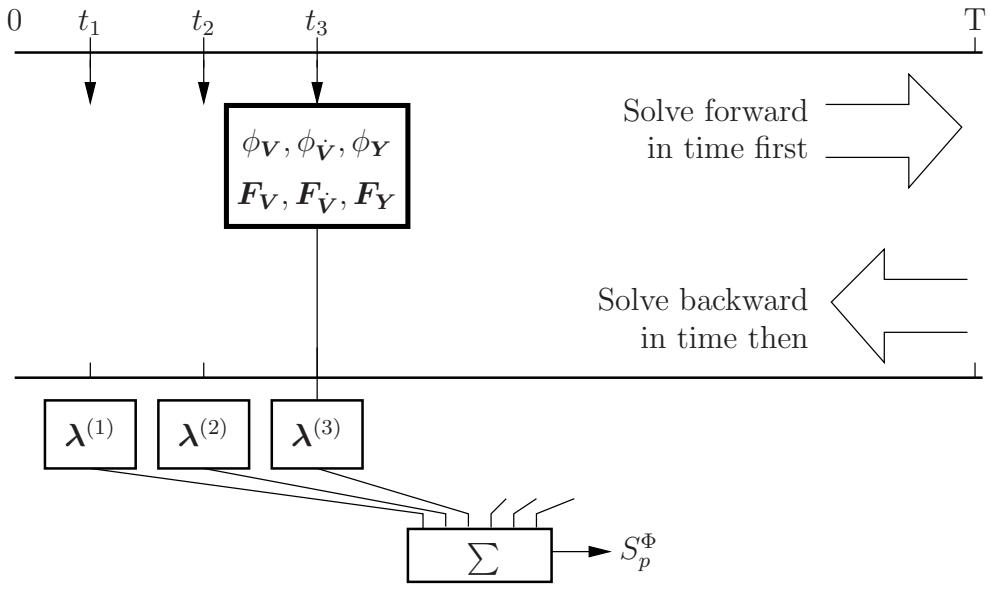


Figure 3.7.: Adjoint method for transient sensitivity calculation

Transient Adjoint method

The method is based upon the application of Tellegen's theorem on electronic circuits [PSD70]. The implementation is described in [BS80] and is shown in Fig. 3.7. Let there be a measurement Φ according to (3.21), which is obtained by a transient simulation.

$$\Phi = \int_0^T \phi(\boldsymbol{V}, \dot{\boldsymbol{V}}, \boldsymbol{Y}, p, t)dt \tag{3.21}$$

$\boldsymbol{V}$ denote the state variables, and $\boldsymbol{Y}$ the remaining variables whose time derivatives are not in the circuit equations, $\boldsymbol{F}$. The transient sensitivity of Φ with respect to parameter p, S_p^{Φ}, is found by solving (3.22).

$$S_p^{\Phi} = \int_0^T \left(\phi_p - \boldsymbol{F}_p^T \boldsymbol{\lambda}\right) dt \tag{3.22}$$

Vector $\boldsymbol{\lambda}$ contains the sensitivities of circuit variables, which are the solutions of the adjoint sensitivity network. The original NLDAE is solved forward in time first. At each time point solutions for system variables and sensitivity information are stored. The latter are the derivatives of measure function, ϕ, and circuit equations, $\boldsymbol{F}$, with respect to $\boldsymbol{V}$, $\dot{\boldsymbol{V}}$, and $\boldsymbol{Y}$. Then, the sensitivity network, which has the same time-depended Jacobians, is simulated backward in time and the convolution of (3.22) is calculated. The latter must be repeated for every measurement. There might be accuracy problems when forward time steps are not suited to the backward integration tolerances [NOW99, DGQ99]. Also, the sensitivity matrices must be stored for each time step or regenerated from the state variables. On the other hand, solving the linear sensitivity system for a large number of parameters can be done very efficiently.

Transient Direct method

The direct computation of transient sensitivities is efficient when the number of parameters is small compared to the number of measurements [PRV95, HYTE85, NOW99, DGQ99]. For each parameter the circuit equations are differentiated to obtain the corresponding sensitivity networks.

$$\boldsymbol{F}(V, \frac{d}{dt}V, p, t) = 0 \tag{3.23a}$$

$$\frac{d}{dp}\boldsymbol{F}(V, \frac{d}{dt}V, p, t) = 0 \tag{3.23b}$$

$$\boldsymbol{F_V} \cdot \frac{\partial \boldsymbol{V}}{\partial p} + \boldsymbol{F}_{\frac{d}{dt}\boldsymbol{V}} \cdot \frac{d}{dt}\frac{\partial \boldsymbol{V}}{\partial p} + \frac{\partial \boldsymbol{F}}{\partial p} = 0 \tag{3.23c}$$

$$\boldsymbol{F_V} \cdot S_p^{\boldsymbol{V}} + \boldsymbol{F}_{\frac{d}{dt}\boldsymbol{V}} \cdot \frac{d}{dt}S_p^{\boldsymbol{V}} + S_p^{\boldsymbol{F}} = 0 \tag{3.23d}$$

Using the Euler approximation for the time derivatives and sorting the resulting terms yields.

$$\underbrace{\left(\boldsymbol{F_V} + \frac{1}{h}\boldsymbol{F}_{\frac{d}{dt}\boldsymbol{V}}\right)}_{Jacobian} \cdot S_p^{\boldsymbol{V}} = -S_p^{\boldsymbol{F}} + \frac{1}{h} \cdot \boldsymbol{F}_{\frac{d}{dt}\boldsymbol{V}} \cdot S_p^{\boldsymbol{V}}(t-1) \tag{3.24}$$

Hence, calculating the linear sensitivities, $S_p^{\boldsymbol{V}}$, reuses the Jacobian matrix at each time step after convergence. No additional storage of matrices is needed as the sensitivity networks are directly solved during the transient analysis.

The transition from VRM to CSM is taking place today. This transition is by no means cheap nor simple as it requires substantial changes in library standards and characterisation methodologies.

Keller, Tam, and Kariat

4 Current Source Models

It has been pointed out in Sec. 2.3 that cell delay models are needed which provide a more accurate modelling of the driving behaviour for increasingly resistive interconnects. According to [KTK08], this class of delay models is denoted as current source models:

Definition 6. *A current source model (CSM) uses at least two parameters to predict the effective currents for one or more pins of a logic cell.*

For the case of CCS and ECSM, the parameters are input slew rate and output load. For CSMs which are developed in academia, the parameters are input and output voltage.

$$\hat{i} = f(V_i, V_o) \tag{4.1}$$

Therefore, these models are truly electrical models, whereas ECSM and CCS belong to the class of voltage response models (VRMs). Because CSMs provide the cell current as functions of port voltages, the output waveform, and hence delay and output slew rate, are obtained through a Spice-like simulation. Therefore, these CSMs can handle almost arbitrary signal waveforms and impose no restrictions on the load model. Consequently, CSMs are holistic models which can be used for timing and for noise analysis, in some cases even for power analysis.

This chapter provides an overview of different CSMs. For each approach, the model and its characterisation scheme are discussed. The new SCSM current source model, which is the essential part of this work, is described in Sec. 4.5. The discussion includes the generation of parametric models, and leads to the derivation of a combined power-timing model named PXM in Section 4.6. Implementation and utilisation of CSMs, including SCSM and PXM, are discussed in detail in Chapter 5.

4.1. Equation-Based Models

Equation-based models formulate the circuit equations for a single logic cell. These equations are then solved within the model, instead of the Spice simulator, and only the port current is returned. This reduces complexity of the overall simulation. It also allows to use simpler Spice-like simulators which are usable for timing simulation because a complex netlist of transistors is converted into a circuit of SC models with signal propagation direction.

Types of this CSM are presented in [TP07], [DDR07], and [Kno+08], of which the latter is discussed briefly. The model exploits that Spice simulators (see Sec. 3.1) effectively always solve linear circuit equations for $\boldsymbol{\Delta V}$.

$$\boldsymbol{J}(\boldsymbol{V}^{(k,n)}) \cdot \boldsymbol{\Delta V}^{(n+1)} = (\boldsymbol{G} + s\boldsymbol{C}) \cdot \boldsymbol{\Delta V}^{(n+1)} = -\boldsymbol{F}(\boldsymbol{V}^{(k,n)}) \tag{4.2}$$

s denotes the complex parameter in the frequency domain (Laplace transformation). Replacing $\boldsymbol{\Delta V}^{(n+1)}$ by $\boldsymbol{V}^{n+1} - \boldsymbol{V}^{n}$ for the equations of linear/linearised circuits leads to another common formulation of (4.2) using the excitation vector, $\boldsymbol{E}$.

$$\boldsymbol{J} \cdot \boldsymbol{V} = \boldsymbol{E} \tag{4.3}$$

For the special case of a linearised logic cell with attached voltage sources, the vector of circuit variable, $\boldsymbol{V}$ contains internal node potentials and the port currents.

$$\boldsymbol{V} = [i_i, i_o, V_i, V_o, \dots]^T \tag{4.4}$$

The principle of the CSM is to construct Jacobian and excitation vector in (4.3) with symbolic references to the circuit elements instead of using their numerical values. Then, applying Cramer's rule to (4.3) yields a symbolic solution for i_o.

$$i_o = \frac{\begin{vmatrix} \boldsymbol{J} & \boldsymbol{E} \\ -\mathbf{d}_o^t & 0 \end{vmatrix}}{\begin{vmatrix} \boldsymbol{J} & \mathbf{0} \\ -\mathbf{d}_o^t & 1 \end{vmatrix}} = \frac{\left|\mathbf{T}^*\right|}{\left|\mathbf{T}\right|} = \frac{N(V_i, V_o, s)}{D(s)} \tag{4.5}$$

The selection vector $\mathbf{d}_o = [0, 1, 0, 0, \dots]^T$ is a zero vector with a one at the position of i_o. The symbolic solution (4.5) consists of nominator and denominator which are linear combinations of symbolic s-polynomials of the port voltages. They are converted to a system of first-order time derivatives to be compatible with Spice solvers. To account for nonlinearity of the logic cell, the polynomial coefficients are functions of port voltages. They are measured in DC Spice simulations and are stored in lookup tables. The application of the model in simulation is shown in Fig. 4.1. The input voltage is known and the output voltage is calculated in a Spice-like manner. At each time point, $V_o^{(k-1)}$ and $V_i^{(k-1)}$ determine the coefficients of the linear transfer equation (4.5). The largest drawbacks of this method are the high complexity of solving (4.5) symbolically and the numerical instability of higher order time derivatives.

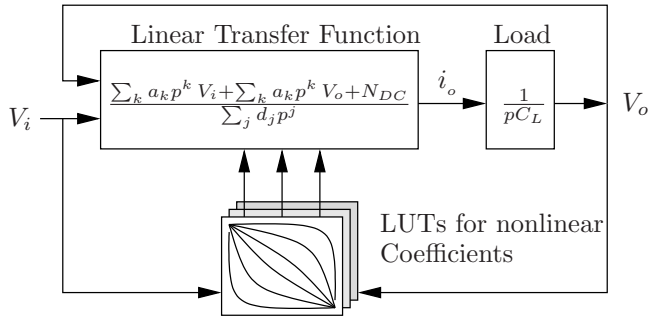


Figure 4.1.: Simulation with Transfer System Model [Kno+08]

4.2. Current Source Models by Error Minimisation

The majority of CSMs consist of a small number of voltage-controlled or linear circuit elements as shown in Fig. 4.3. The values or functions of these model components are determined by a number of Spice simulations and by minimising the error in port current, cell delay, or output voltage waveform.

Every CSM uses at least one VCCS, $\mathbb{I}_o$, which models the static output current as a function of port voltages. For all approaches, its values are found by the DC simulation of Fig. 4.2. DC voltage sources are attached to the ports, and input and output voltages are swept from V_{ss} to V_{dd}. The port currents are measured for each combination of port voltages.

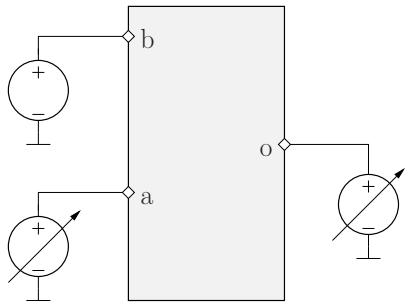
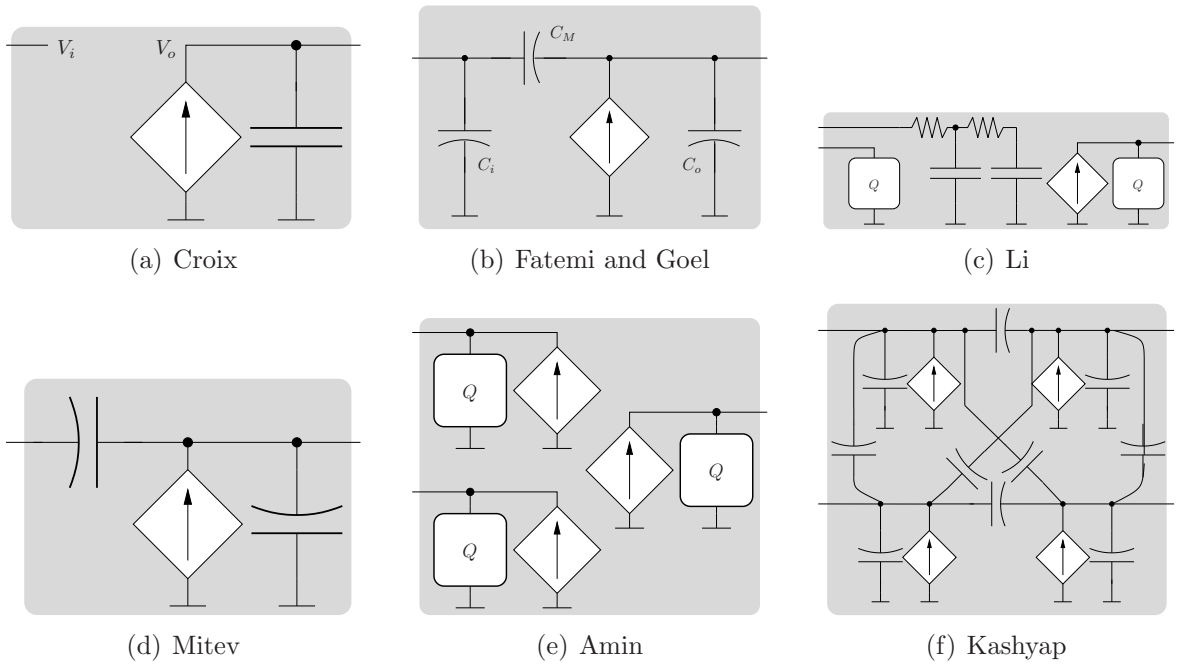


Figure 4.2.: Characterisation setup for static port current

Further elements like capacitors, voltage-controlled charges, or resistors are needed to account for the dynamic behaviour of the logic cell. Their selection and characterisation makes the differences of the CSM approaches.

The CSM according to [CW03] is shown in Fig. 4.3(a). Dynamic effects are modelled by a linear capacitor and a time shift of the output voltage waveform. The capacitor value is determined by applying a fast ramp signal to the cell and minimising the difference between original and CSM output waveform. For cells with large delays, the entire output waveform is shifted in time by an amount t_d, which is also determined by matching the output waveforms. Due to a missing time shift element in Spice, this CSM is not directly applicable to accelerate Spice simulations but requires post processing [BJV06].



(a) Croix (b) Fatemi and Goel (c) Li

(d) Mitev (e) Amin (f) Kashyap

Figure 4.3.: Different current source models

Figure 4.3(b) shows the CSM of [FNP06]. It has an input and a Miller capacitor, which models the capacitive coupling from input to output pin. All capacitors are voltage-controlled (VCC) and their lookup tables, shown in Fig. 4.4, are obtained through a sequence of ramp signals and (4.6)–(4.7).

$$i_i \stackrel{!}{=} \hat{i}_i = C_i \cdot \frac{d}{dt} V_i + C_M \cdot \frac{d}{dt} (V_i - V_o) \tag{4.6}$$

$$i_o \stackrel{!}{=} \hat{i}_o = \mathbb{I}_o + C_o \cdot \frac{d}{dt} V_o + C_M \cdot \frac{d}{dt} (V_o - V_i) \tag{4.7}$$

Equations 4.6–4.7 are derived from Kirchhoff's current law for input and output pin of the CSM in Fig. 4.3(b). By keeping V_o constant while ramping V_i, the VCCs C_i and C_M are characterised. Thereafter, C_o can be measured by applying the ramp to V_o and keeping V_i constant. The same method is used to generate models of latches, SRAM cells, and transmission gates [NFP11], and for modelling the short circuit current of combinational cells [FNP07].

The CSM of [GV08a] also has the structure of Fig. 4.3(b), but the characterisation is based upon step functions similar to [Ami+06] (described shortly). The lookup tables are approximated by third order Legendre polynomials, and hence simulation for characterisation is only performed at the approximation nodes. Parameter variation is modelled as linear or quadratic function.

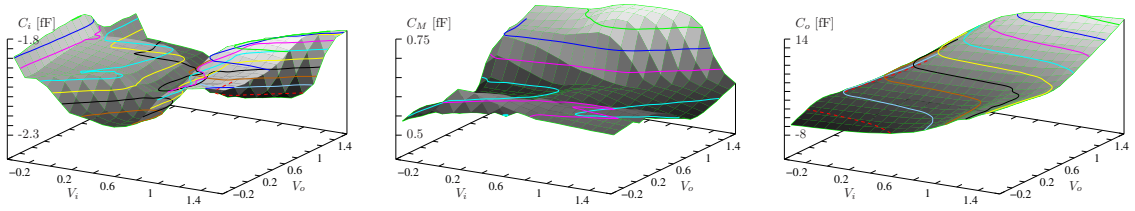
The authors of [CKSB06] and [KLX06] avoid any additional simulation by deriving the CSMs from existing ECSM/CCS libraries (see page 19 in Sec. 2.3). The model is similar to Fig. 4.3(b) but consists of linear capacitors. Again, KCL for the output node

is the basis to derive the model components.

$$\hat{i}_o + C_M \frac{d}{dt}(V_o - V_i) + (C_o + C_L)\frac{d}{dt}V_o = \mathbb{I}_o(V_i, V_o) \tag{4.8}$$

C_L in (4.8) is the attached output capacitance. Its values, as well as input and output waveforms, are directly retrieved from the library. The transient output current of table entries with slow transitions is the best approximation of the DC current. Equation 4.8 implies that for these cases the output current is maximal. Hence, table segments of maximal output current are found and the current is expressed as $\mathbb{I}_o(V_i, V_o)$. Similarly, approximations for C_M and C_O are found from segments of minimal output currents. Finally the CSM input capacitance is calculated by $C_i = C_{lib} - C_M$, with C_{lib} being the capacitance value stored in the library. The authors of the Imodel use a similar method to derive CSM noise models from NLDM libraries [DSNZ09].

The CSM in [LFA07], shown in Fig. 4.3(c), uses basically the same characterisation method of ramp signals. The three voltage-controlled capacitors are merged into two voltage-controlled charges (VCQs). In addition, it has a second order lowpass filter to model a cell internal delay for the input voltage. Its parameters are found by minimising the errors in the output voltage waveform. Finally, the CSM is tuned by optimising its components to match typical input and output waveforms. Despite the high complexity of characterisation, the authors provide a parametric CSM by generating different CSMs for different parameter values, and compute a first order model through finite differences.



(a) C_i (b) C_M (c) C_o

Figure 4.4.: Capacitor LUTs for Fatemi's model

The authors of [Mit+07] derive the CSM in Fig. 4.3(d) by first modelling pull-up network and pull-down network individually, and then combine these models. The voltage-controlled elements are not lookup tables but interpolation functions for a small number of points. Similar to approximations of transistor currents, these points have special meanings like the borders between linear and saturation region. For these points also the sensitivities with respect to parameter variation are constructed. Both, model parameters and variation model, are obtained through error minimisation for different input and output waveforms and for a number of Monte Carlo simulations.

CSMs for Multiple-Input Switching

The CSM according to [Ami+06] shown in Fig. 4.3(e) is the most general form of a CSM. Each port is modelled by a VCCS and a VCQ. Cell internal nodes can be treated as

additional virtual ports to increase model accuracy. To characterise a port charge, $\hat{Q}_x$, a step function is applied to one port while the others remain constant at the bias point. With the previously determined static port current, the additional dynamic current is integrated to obtain the charge value.

$$\hat{Q}_x = \int_T \left(i_x(t) - \hat{I}_x(t) \right) dt \tag{4.9}$$

This measurement is repeated for every combination of (virtual) port voltages to obtain lookup tables as in Fig. 4.5. This leads to very large characterisation times, especially when internal nodes increase LUTs dimension.

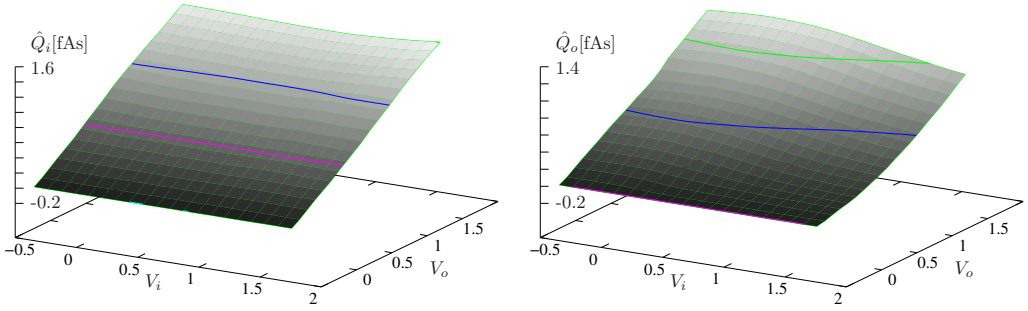


Figure 4.5.: Charge LUTs for Amin's model

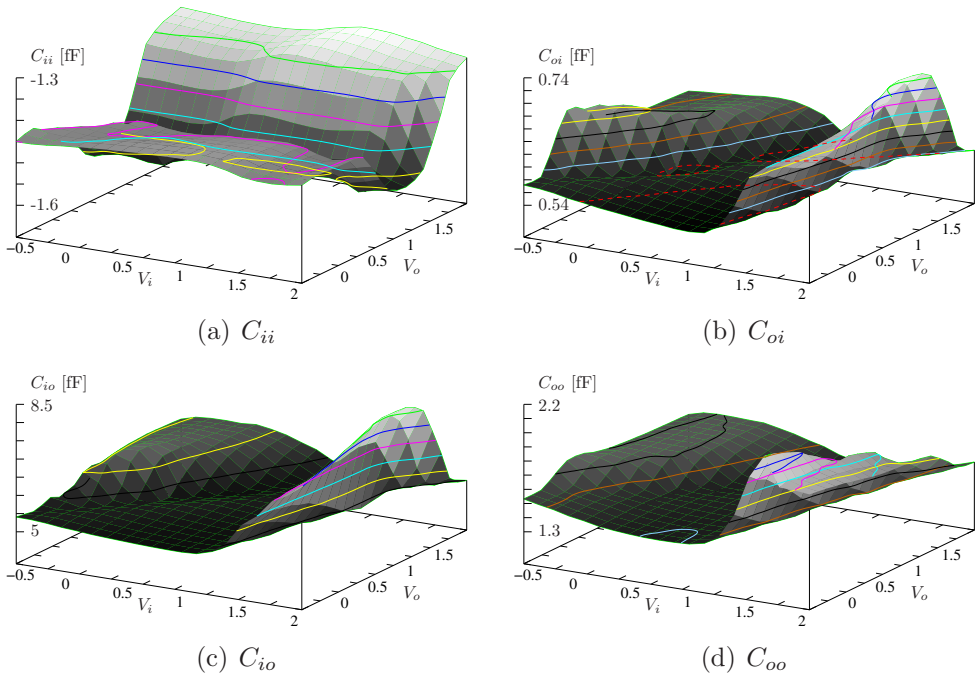
4.3. Current Source Models through Direct Characterisation

The CSM generation based upon error minimisation has at least two drawbacks. The first is a sequential characterisation which requires to first build a DC model before the dynamic components can be generated. The second is the large number of transient simulations which are needed for characterisation. Therefore, the same authors of Fig. 4.3(e) replace the VCQs by a number of voltage-controlled capacitors (VCCs) as shown in Fig. 4.3(f) [KAMC07]. Their values are obtained in an AC analysis for which an AC voltage source with amplitude k and frequency f is attached to cell pin y. The imaginary part of the measured current at pin x yields the transcapacitance through

$$\hat{C}_{xy} = \frac{\Im(i_x)}{2\pi f \cdot k_y} \tag{4.10}$$

The lookup tables for a minimal inverter are shown in Fig. 4.6. The drawback of this CSM is the high model complexity due to the voltage-controlled cross-capacitances.

In contrast to these black-box models, there are a few transistor-oriented approaches. The authors of [NB07] presented a simple CSM-like model which is derived from the individual transistors. Each drain source resistance is modelled as simple exponential expression. Furthermore, the total resistance of series and parallel transistor chains is



(a) C_{ii} (b) C_{oi}

(c) C_{io} (d) C_{oo}

Figure 4.6.: Capacitor LUTs for Kashyap's model

approximated as of minimum and maximum of the exponent terms. In [RVBG08], the transistors are modelled by a lookup table for the drain current and piecewise linear capacitors. Multi-finger transistors are collapsed before simulation.

4.4. Qualitative comparison of existing CSMs

A compact overview on the different approaches, including characterisation schemes and reported applications, is provided in Appendix B.1 at page 101. In this section, the focus is on a more qualitative comparison of the methods. Table 4.1 therefore compares CSMs which have been discussed in the previous sections. Approaches that replace individual transistors by CSMs are excluded because of their high complexity. Similarly, very simple approaches such as in [NB07] or [CKSB06] ignored as they cannot meet the tough accuracy requirements. Equation-based models are not applicable to a wide range of cells. The table shows that almost every CSM requires a substantial characterisation effort, which makes CSM library generation very costly. Furthermore, model complexity varies significantly, partly due to the support of different modelling features such as multiple-input switching. Nonetheless, model complexity directly correlates to memory requirements and indirectly influences simulation speed. Table 4.1 further shows that there is no simple CSM that supports modelling of supply currents. Similarly, parameter variation is not supported by all models which might be due to the high characterisation efforts.

CSM	Fig.	Characterisation effort	Model complexity	Supply currents	Parameter variation
Croix	4.3(a)	moderate	low	–	–
Fatemi	4.3(b)	high	low	–/✓	–
Goel	4.3(b)	high	low	–	✓
Li	4.3(c)	high	moderate	–	✓
Mitev	4.3(d)	moderate	moderate	–	✓
Amin	4.3(e)	very high	high	✓	–
Kashyap	4.3(f)	low	very high	✓	–

Table 4.1.: Qualitative comparison of existing CSMs

4.5. Systematic Current Source Model

It has been pointed out in the last section that CSM characterisation usually causes a significant simulation effort or results in complex CSMs. The number of simulations is even larger when parametric CSMs have to be generated which account for variations of process or environmental parameters. Consequently, the objective is to derive a simple CSM which allows a much faster characterisation. Ideally, the new CSM is holistic, which means it supports various analyses such as timing, noise, and power analysis. The automated model generation for logic cells must furthermore be applicable to many different combinational cell types. In addition, the characterisation must be adjustable by user input but not requiring complex settings. Most important, cell characterisation must be quick, and CSM accuracy must be close to Spice.

In contrast to existing phenomenological approaches, the components of the new CSM are not determined by matching port currents which can be observed from outside the cell. In this approach, the model components replace original netlist elements. That's why, each component has an explicit relation to original netlist elements, and the model at higher abstraction level is physical consistent to the underlying models. Consequently, the transistor netlist of the standard cell is the basis for a systematic white-box characterisation.

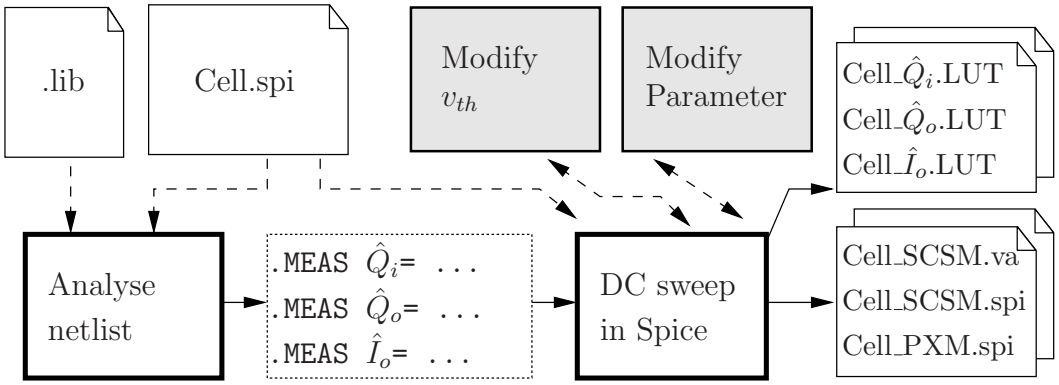


Figure 4.7.: SCSM characterisation flow. Gray processes generate parametric models

Figure 4.7 shows the full characterisation flow. Each SCSM is directly derived from the transistor netlist and the corresponding entry in the cell library file (.lib). If the library file is not available, the transistor netlist is used to determine logic function and timing arcs for the cell. The SCSM targets single-input switching. Consequently, separate CSMs are generated for the timing arcs ⟨a0→o⟩ and ⟨0b→o⟩ of an NOR cell. A separation in rising and falling timing arc is not needed because the CSM is waveform-independent. The main task in SCSM generation is to analyse the netlist. It defines the SCSM structure and identifies all netlist elements that contribute to the model components. These contributions are written as Spice measurement statements. During a DC simulation, their numerical values are obtained and stored in lookup tables. These are then used by the cell models, which are written as Spice subcircuits or Verilog-A modules. The grey boxes in Fig. 4.7 indicate processes for generating parametric

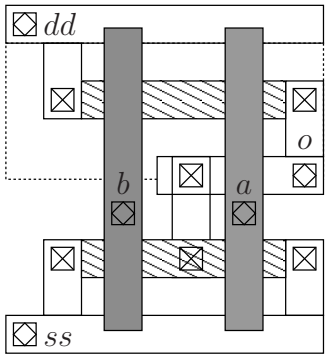


Figure 4.8.: NOR layout

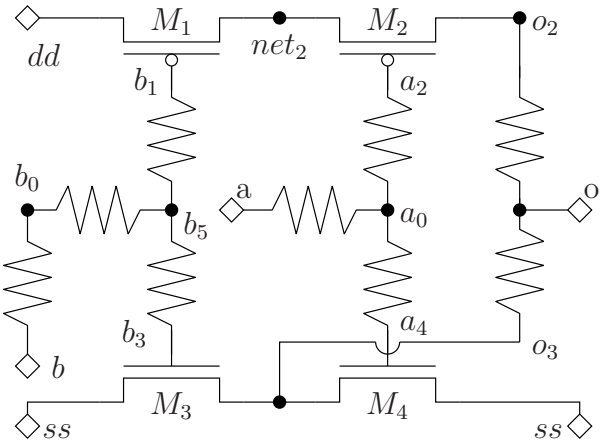


Figure 4.9.: NOR schematic

```
1  .SUBCKT R_SNR2X010 a b o dd ss
2  M1 net2 b_1 dd   dd pmos
3  M2 o_2  a_2 net2 dd pmos
4  M3 o_3  b_3 ss   ss nmos
5  M4 o_3  a_4 ss   ss nmos

7  R1  b_0 b_5  34.3453
8  R2  b_3 b_5  4.1887
9  R3  b_1 b_5  6.0232
10 R11 b   b_0  6.0176

12 R4  a_2 a_0  6.6944
13 R5  a_0 a_4  5.8194
14 R12 a   a_0  35.9332

16 R13 o_2 o    6.6425
17 R14 o   o_3  8.4617

19 C1 a    o_2  1.541E-17
20 C2 a    net2 2.514E-17
21 C3 a_0  o_2  1.514E-17
22 C4 a_4  ss   3.214E-17
23 C5 a_0  o    1.514E-17
24 C8 b_1  a_0  1.076E-17
25 C9 b_1  b_3  3.372E-17
26 C10 b_3 o    1.076E-17
27 C11 b_1 dd   2.088E-17
28 C12 ss  b    1.444E-17
29 C13 o   dd   6.956E-18
30 .ENDS
```

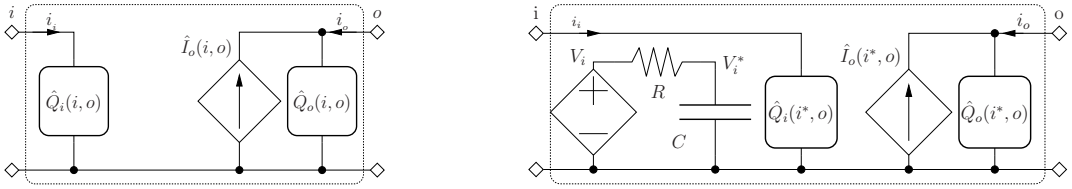
Listing 4.1: NOR netlist

models. They are explained at the end of this section, beginning on page 60. The same characterisation of Fig. 4.7 is used to generate holistic timing/noise/power models. These Pull-up/Pull-down Models (PXMs) are explained in Section 4.6.

The complete SCSM model derivation and model generation is explained using a NOR standard cell. Figure 4.8 shows the mask layout of NOR and Fig. 4.9 the simplified electrical view after parasitic extraction. Capacitances, which can be attached to every node, are not shown to improve readability. The netlist in Listing 4.1 presents the same information in a textual format. It contains the transistors as well as parasitic capacitors and resistors. The resistors stem from contacts and longer poly silicon wire segments.

Deriving the SCSM structure

According to [HRR78], the task of defining a model is "the process of obtaining a simplified circuit, a set of mathematical equations, a multidimensional table, or some symbolic expressions which usually represent a more complex circuit". The aim of the SCSM is to produce voltage waveforms that are identical to those of the NOR's transistor netlist description for any given input signal and any attached load. This will be realised when for both, input and output port, the SCSM current, $\hat{i}(t)$, equals the original current,



(a) SCSM for small and medium sized cells

(b) SCSM for large cells

Figure 4.10.: SCSM with and without lowpass filter to account for parasitic delay

$i(t)$, for any combination of port voltages, $\boldsymbol{V}$(t).

$$i(\boldsymbol{V}(t), \frac{d}{dt}\boldsymbol{V}(t)) = \hat{i}(\hat{\boldsymbol{V}}(t), \frac{d}{dt}\hat{\boldsymbol{V}}(t)) \tag{4.11}$$

A voltage-controlled current source, $\mathbb{I}$, is required to handle the special case of constant port voltages.

$$i(\boldsymbol{V}, 0) = \hat{i}(\hat{\boldsymbol{V}}, 0) = \mathbb{I}(\hat{\boldsymbol{V}}) \tag{4.12}$$

In case of voltage changes of port nodes, i and o, additional dynamic currents result from charging the parasitic capacitors and transistor capacitances. Each dynamic current is modelled by an associated port charge, $\hat{Q}$, and is implemented as a voltage-controlled charge, $\mathbb{Q}$.

$$\hat{i}(\hat{\boldsymbol{V}}(t), \frac{d}{dt}\hat{\boldsymbol{V}}(t)) = \mathbb{I}(\hat{\boldsymbol{V}}(t)) + \frac{d}{dt}\mathbb{Q}(\hat{\boldsymbol{V}}(t)) \tag{4.13}$$

Figure 4.10(a) shows the corresponding SCSM. Notably the VCCS at the input pin is removed. As it will become clear in the course of this section, this VCCS would only model gate leakage currents, which are much smaller than the dynamic currents. The remaining input charge, $\hat{Q}_i$, is also denoted as receiver charge. Similarly, the output components $\hat{Q}_o$ and $\hat{I}_o$ are also named driver charge and driver current.

In very large cells the passive parasitic network causes a notable signal delay at the input. A lowpass filter is therefore inserted into the SCSM which produces a delayed input voltage in $\hat{\boldsymbol{V}}^* = [V_{i^*}, V_o]^T$. The final topology of the SCSM is shown in Fig. 4.10(b).

$$i(\boldsymbol{V}, \frac{d}{dt}\boldsymbol{V}(t)) = \mathbb{I}(\hat{\boldsymbol{V}}^*(t)) + \frac{d}{dt}\mathbb{Q}(\hat{\boldsymbol{V}}^*(t)) \tag{4.14}$$

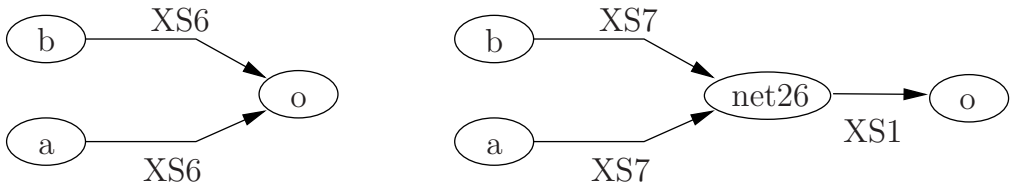
Once the CSM structure has been derived, the next step is to define the functions of $\hat{I}$ and $\hat{Q}$.

Topology Analysis

The SCSM components are derived from netlist elements. This requires to automatically understand and structure the transistor netlist. This essential step is not part of own research but employs the results from [SS09] and [You10]. Key idea in the work of [SS09]

Name	Type	Devices		Inputs		Outputs
XS6	INVERTER	XS5	Serial Chain	[in]	b	[out] o = $!a!b$
			M_1 PMOS	[in2]	a	
			M_2 PMOS			
		XS3	Parallel Array			
			M_4 NMOS			
			M_3 NMOS			

Table 4.2.: Structural information from DCAT analysis



(a) NOR (b) OR

Figure 4.11.: Examples of recognised cell structures

is to build a bipartite graph representing the topology of transistors. Therefore, in a first step all parasitics are removed and their electrical nodes are merged to one logical net. Thereafter, this graph is analysed to find isomorphisms which are defined in a library of basic structures. Such basic structures are parallel structure, serial structure, and gate structure which consists of structures containing NMOS and PMOS transistors. The library elements are recursive; each structure can contain any other structure. This introduces hierarchy levels to the graph. The left part of Tab. 4.2 shows this information for the NOR cell. At lowest level, it finds a serial connection of PMOS transistors M_1 and M_2, as well as a parallel connection of NMOS transistors M_3 and M_4. These two structures form an inverting stage or gate, XS6. It also finds "a" and "b" as the inputs to this stage, and node "o" as the logical output. [You10] finally extracts the logic function from this structural information. This information can be used to define the timing arcs of the logic cell for which a SCSM will be generated. Figure 4.11(a) finally shows the derived signal-flow graph for the NOR cell. It defines that SCSMs are generated for input pins "a", $\langle$a0→o$\rangle$, and "b", $\langle$0b→o$\rangle$. The topology analysis works for complex cell types including tristate buffers or transmission gates. Figure 4.11(b) shows the signal flow graph for an OR, consisting of two inverting stages.

Identifying Relevant Contributors

The information about the topology of the logic cell is used to identify the functions of the SCSM model components. It allows to systematically trace the physical origins of observed port behaviour. A careful look at Listing 4.1 reveals that each cell consists of active and passive parts. The active parts are channel connected blocks (CCBs), which

Algorithm 4.1 Construction of resistive tree

```
RT = { pin } ;
repeat
  for all resistors not yet in RT do
    if resistor shares node with RT then
      add resistor and new node to RT;
until RT does not change
```

consist of transistors and implement the logic function. The passive parts are parasitic resistors and capacitors. It turned out that these resistors form trees which connect the CCBs to the cell pins. Hereby tree refers to the definition from graph theory.

Definition 7. *A tree, T, is an undirected connected graph of N nodes which are connected by $N-1$ edges.*

Definition 7 enforces that trees do not contain loops [BSMM01]. Consequently, there is exactly one path between any two nodes. It is possible to freely select one node, x, of this tree and denote it as root node. The tree is then called rooted tree.

Definition 8. *A resistive tree, RT, is a tree of which the nodes represent electrical nodes and each edge represents a resistor.*

The parasitic network of the logic cell can be regarded as a graph, in which the resistive trees can be found. Algorithm 4.1 is used to construct the resistive tree for input and output node of the logic cell. The practical importance of these resistive trees for the SCSM model is due to the following property.

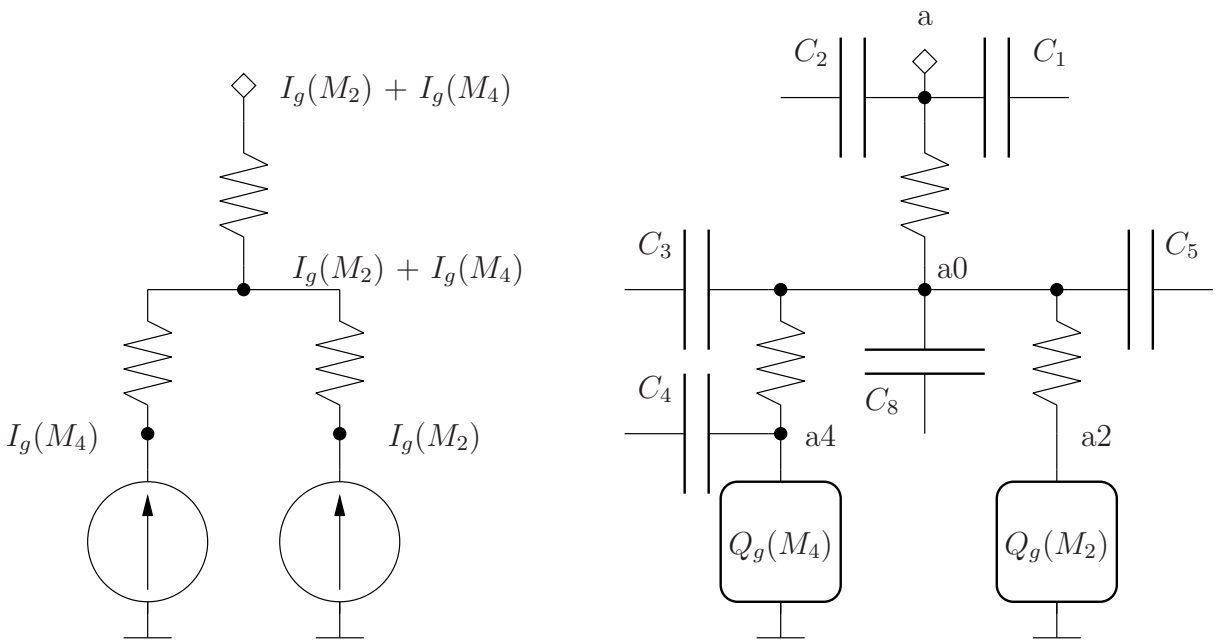
Theorem 1. *Let RT_n be a resistive, rooted tree with current sources I_i attached to its nodes i, and V_n a grounded voltage source attached to root node n. The current through V_n equals the sum of injected currents from I_i.*

Proof: KCL forces the net current to be zero for all nodes in RT_n, including the leaf nodes. Since each current source has a fixed value, the required compensation currents must be drawn from neighbouring nodes. Furthermore, the voltage source V_n is the only element that can provide a non-fixed current to fulfil KCL at node n and all other nodes. ■

In other words, the resistive tree with root node n is an expression tree for the net current at n, consisting of summations of current sources.

$$I(V_n) = \sum_{k}^{RT_n} I_k \tag{4.15}$$

For the application of Theorem 1, active input pin and output pin are taken as root nodes and resistive trees are found by Alg. 4.1. The sum of all DC current sources then yields the static port current.



(a) Port current from RT_a (b) Port charge from RT_a

Figure 4.12.: Resistive tree for cell pin a with currents and charges

Definition 9. *The static port current $\hat{I}_n$ is the sum of static current sources which are attached to each node in the resistive tree RT_n.*

Applying this algorithm to the NOR circuit in Listing 4.1 on page 48 yields the port current of Fig. 4.12(a). For obtaining the transistor contributions, a compact transistor model is used [KAMC07]. It models each pin by a voltage-controlled current source (VCCS) and a voltage-controlled charge (VCQ), and all physical interna are hidden in the complex functions of these elements. Figure 4.12(a) shows very clearly that the static port current, $\hat{I}_x$, of a logic cell equals the sum of the resistively connected transistor pins. Consequently, the derived model equations for the static port currents $\hat{I}_a$ and $\hat{I}_o$ are:

$$\hat{I}_a = I_g(M_2(V_{o_2}, V_{a_2}, V_{net_2})) + I_g(M_4(V_{o_3}, V_{a_4}, V_{ss})) \tag{4.16}$$

$$\hat{I}_o = I_d(M_2(V_{o_2}, V_{a_2}, V_{net_2})) + I_d(M_3(V_{o_3}, V_{b_3}, V_{ss})) + I_d(M_4(V_{o_3}, V_{a_4}, V_{ss})) \tag{4.17}$$

Beside the static current sources, there is another type of current contributor inside a cell. Each capacitance, which can be nonlinear, causes a dynamic current if the amount of its charge is altered.

$$i^k_{dyn} = \frac{d}{dt} Q^k(\boldsymbol{V}^k(t)) = \frac{dQ^k(\boldsymbol{V}^k(t))}{d\boldsymbol{V}^k(t)} \cdot \frac{\boldsymbol{V}^k(t)}{dt} \tag{4.18}$$

Similar to the static port current of Fig. 4.12(a), a dynamic port current is defined as

Algorithm 4.2 Deriving measurements for input and output port components

```
RT_{i/o} = findResistiveTree(i/o, netlist);
for all Node n in RT_{i/o} do
  for all Capacitor C connected to n do
    Q_{i/o} += C · V(C);
  for all Transistor M connected to n do
    I_{i/o} += I_n(M);
    Q_{i/o} += Q_n(M);
```

the sum of all dynamic current sources (4.18) which are attached to the resistive tree.

$$\hat{i}_x = \sum_k^{RT_x} \frac{d}{dt} Q^k(\boldsymbol{V}^k(t)) \tag{4.19}$$

There is a fundamental problem with the formulation of (4.19). The dynamic currents of (4.18) depend on the absolute voltages, that is the operating point, as well as the time derivatives of these voltages. This introduces ambiguities during model generation of selecting valid values for the time derivatives, or significantly increases model and characterisation complexity. Therefore, the proposed modelling method changes summation and time derivatives in (4.19).

$$\hat{i}_x = \frac{d}{dt} \sum_k^{RT_x} Q^k(\boldsymbol{V}^k(t)) = \frac{d}{dt} \hat{Q}_x(\boldsymbol{V}) \tag{4.20}$$

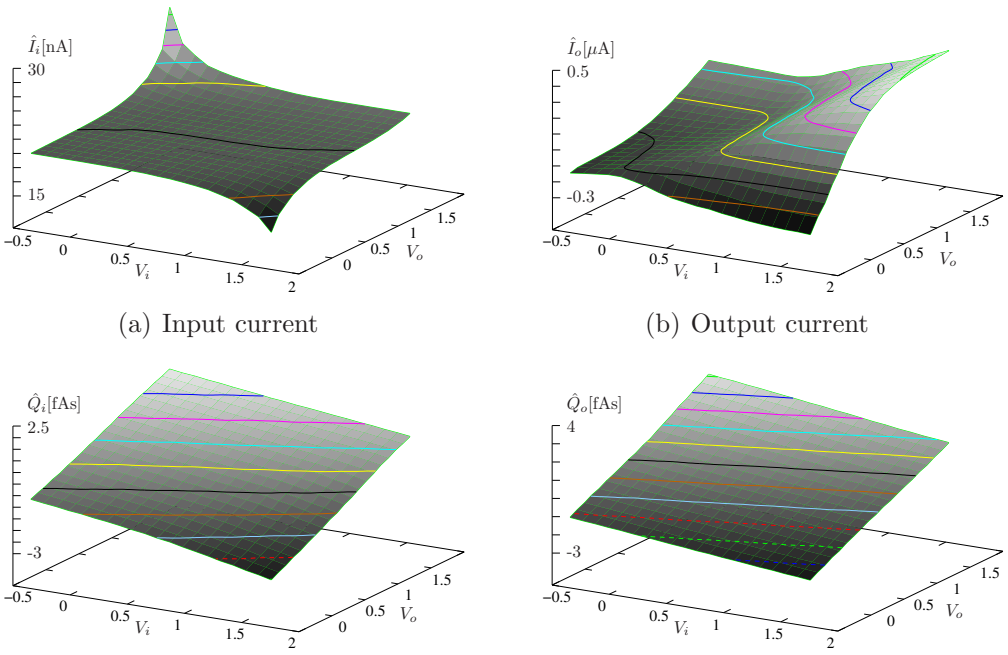
Hence, the dynamic output current results from the time derivative of a single charge, denoted as port charge.

Definition 10. *Let RT_n be a resistive tree with root node n. The port charge $\hat{Q}_n$ is the sum of charges that are stored at the nodes of RT_n.*

Figure 4.12(b) shows the resistive tree with attached capacitors and charges. Due to the similarity with the static port current, Alg. 4.2 is the general way of deriving the model equations for port charge and port current. However, this is only possible due to an important property of the port charge: $\hat{Q}_x$ is a static quantity, that means its value only depends on the operating point [Bmm]. This algebraic dependency is also expressed in the corresponding model equations (4.21)–(4.22) of the NOR example of Fig. 4.9, which do not contain time derivatives.

$$\begin{aligned}\hat{Q}_a = {} & Q_d(M_2(V_{o_2}, V_{a_2}, V_{net_2})) + Q_d(M_4(V_{o_3}, V_{a_4}, V_{ss}))+ \\ & + C_1 V_{ao2} + C_2 V_{anet_2} + C_3 V_{a0o2} + C_4 V_{a4ss} + C_5 V_{a0o} + C_8 V_{a0b1}\end{aligned} \tag{4.21}$$

$$\begin{aligned}\hat{Q}_o = {} & Q_d(M_2(V_{o_2}, V_{a_2}, V_{net_2})) + Q_d(M_3(V_{o_3}, V_{b_3}, V_{ss})) + Q_d(M_4(V_{o_3}, V_{a_4}, V_{ss}))+ \\ & + C_1 V_{o2a} + C_3 V_{o2a0} + C_5 V_{oa0} + C_{10} V_{ob3} + C_{13} V_{odd}\end{aligned} \tag{4.22}$$



(a) Input current (b) Output current

(c) Input charge (d) Output charge

Figure 4.13.: Static port currents and port charges

There is the special case when both nodes of a capacitor belong to the same resistive tree, for example capacitor C_9. This leads to two charge contributions which are identical but of opposite sign. Consequently, this capacitor does not contribute to the overall port charge and could be excluded by the algorithm.

Numerical Values for CSM Elements

After deriving the function for each SCSM component, port charges and DC port current, their numerical values are determined through a Spice simulation. Since all component functions are algebraic, the DC characterisation setup of Fig. 4.2 is used. Moreover, since the components are mutually independent, they are characterised in the same simulation. The attached DC voltage sources are set to the constant value of the side input. In the case of timing arc ⟨a0→o⟩, that is V_b equals zero. A nested sweep of the DC voltage sources at the active pins, a and o, is performed to cover a range of possible port voltages. For every operating point, equations (4.16, 4.17, 4.21, and 4.22) are evaluated and their values are stored. These equations physically still depend on many internal voltages. To reduce the complexity and to provide a cell model, these internal node potentials are approximated by an algebraic function of the port potentials, V_a, V_b and V_o. Since V_b is constantly zero for the timing arc ⟨a0→o⟩, all SCSM components are two-dimensional functions of V_a and V_o.

Figures 4.13(a)–4.13(b) show input and output current for the NOR. The output current is a typical composition of PMOS and NMOS drain currents. Depending on the input voltage, one of both is conducting and dominating. The input current models only

gate leakage and thus is several orders smaller. Compared to the dynamic current, which results from the input port charge, is is negligible. Hence, the final SCSM has only a VCCS at the output port. The port charges in Figure 4.13(c)–4.13(d) are almost linear due to contributions from parasitic capacitors. Nonetheless, stronger nonlinearity has been observed for other SCs, and a linear approximation significantly reduces accuracy.

The total time needed to obtain the SCSM and LUTs (50x50) for a minimum inverter is 3.39 s. This is even less than the 6.22 s which are required for generating a NLDM with 7×7 LUT.

Quasi Static Model Components

The model components of (4.17) and (4.21–4.22) are approximated by algebraic functions of port potentials instead of internal node potentials. However, internal capacitances introduce internal states which lead to an differential algebraic dependency. This conflict is resolved by an approximation, which exploits the principles of electrical simulation. Section 3.1 introduced how Spice simulators perform transient simulation for a number of discrete time points. Most often, time derivatives are approximated using Euler or Trapezoid method, which employ two time points [VS94]. Assuming that a time step of 3 ps is used to simulate the dynamic voltage changes in Fig. 4.14, the simulator does not recognise a difference between the waveforms ① and ②. For both cases, the time derivative is approximated by (4.23), and the effective signal waveform is a straight line from V_1 to V_2 [MP71].

$$\frac{d}{dt}V = \dot{V} = \frac{V(3\,\mathrm{ps}) - V(0\,\mathrm{ps})}{3\,\mathrm{ps}} = \frac{V_2 - V_1}{3\,\mathrm{ps}} \tag{4.23}$$

Applying this voltage waveform to X, a circuit element with arbitrary function, results in the model evaluation of X for only these two voltage-time points:

$$X(t = 0\,\mathrm{ps}, V = V_1) \tag{4.24}$$

$$X(t = 3\,\mathrm{ps}, V = V_2) \tag{4.25}$$

Consequently, X behaves as if it is driven by a piecewise-*linear* voltage source. If it is further assumed, that X is implemented by algebraic equations, it is independent of the time derivative (4.23). Therefore, X behaves as if it is driven by a piecewise-*constant* voltage source, which is shown in Fig. 4.15.

This systematic limitation of simulation accuracy is the basis for the SCSM characterisation. As described earlier in this section, only DC simulations are used, and SCSM components are built upon algebraic measurement statements. This assumes, that the time steps during a transient simulation are sufficiently large, such that all cell internal node potentials decay to the values which are identical to the DC operating point when DC voltage sources are attached the ports.

Figure 4.16 validates this assumption for the NOR cell of Listing 4.1. It displays the voltage differences between typical transient simulations and the DC potentials for internal nodes when being simulated with different input slew rates and output loads.

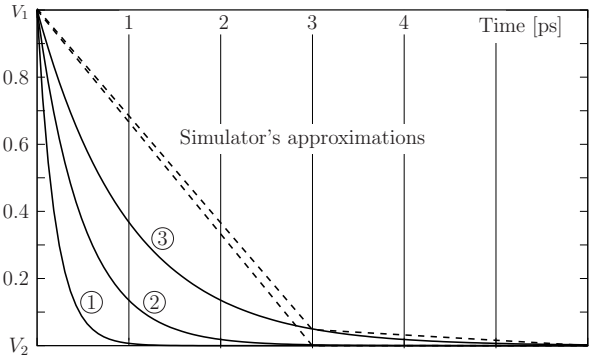


Figure 4.14.: Fast decaying particulate solutions not visible to the simulator

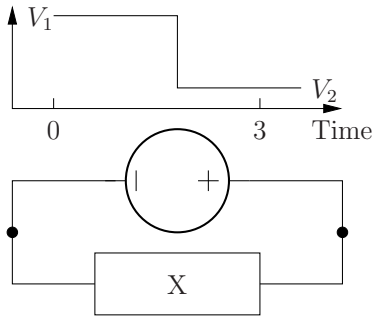
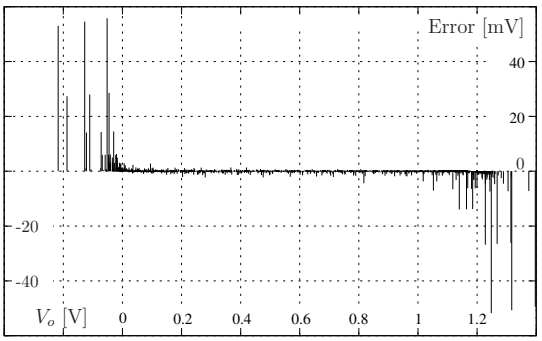
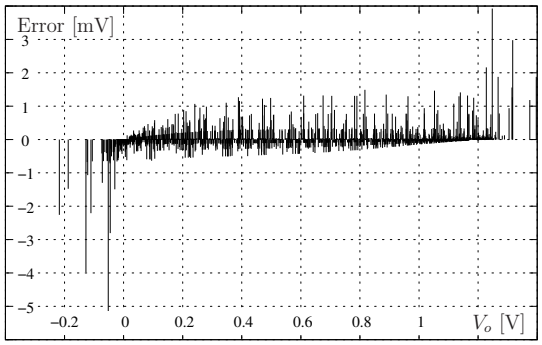


Figure 4.15.: Equivalent piecewise constant waveform for algebraic component X



(a) Node in RT_a of input "a" of a small cell



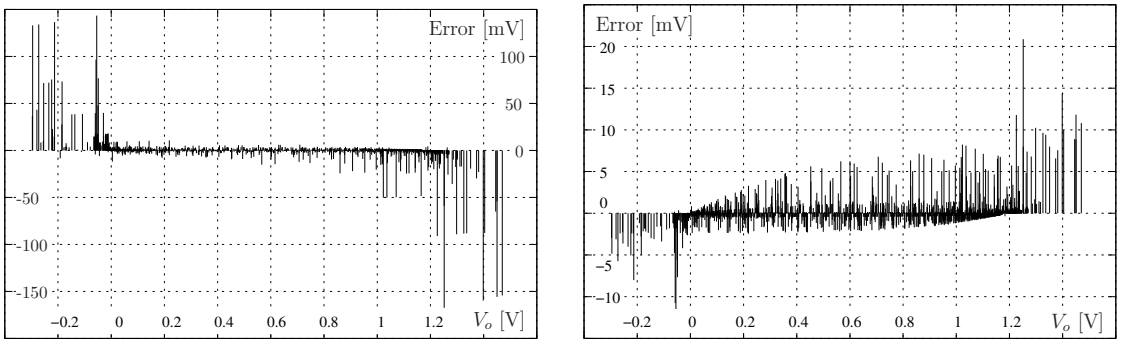
(b) Node in RT_o of output "o" of a small cell

Figure 4.16.: Very small voltage differences for internal node potentials

For the vast majority of time points the voltage errors are less than 10 mV. Only very sharp inputs of less than 1 ps and loads of 1–2 fF cause temporal errors of 55 mV.

Quasi Static Correction

Standard cells with high driver strengths consist of multiple parallel transistors to increase the effective channel width. Connecting all transistors to the cell ports creates large resistive trees. Especially for the input pins, the tree is so deep that there is a noticeable signal delay. Figure 4.18 shows the step responses for the transistor gate pins in a large INVERTER cell. While the input changes immediately from 0 V to 1.2 V, this change takes about 6 ps for internal nodes. Therefore, Fig. 4.17 shows differences of node potentials between transient and DC simulation which differ by up to 167 mV for sharp transitions. Obviously the spurious potentials result in wrong values of currents, and hence in derivations of voltage waveforms and delay errors. Figures 4.17(a) and 4.17(b) also show that the parasitic delay is significant for the input pin, but not for the cell output. This corresponds to the netlists in which the input pins have deep resistive



(a) Node in RT_a of input "a" of a large cell

(b) Node in RT_o of output "o" of a large cell

Figure 4.17.: Large voltage difference at input for internal node potentials

Algorithm 4.3 createLowpassFilter

```
RT = findResistiveTree(input pin, netlist)
for all Node n in RT do
  if Transistor is attached to n then
    add measurement for cutoff frequency f_n^3dB
Run AC analysis
Build average cutoff frequency f^3dB
```

trees, whereas the trees of output nets are wide.

To account for this dynamic behaviour, a lowpass filter can be added automatically to the SCSM. Its output voltage V_i^*, represents ta delayed input voltage, and it is used to control the SCSM model components. Algorithm 4.3 is used to determine its cutoff frequency by an AC analysis. If the resulting frequency is greater than a user defined threshold, as it is the case for small cells, the filter is not inserted.

Stacked Transistors

So far the explanation discussed the model generation for the timing arc ⟨a0→o⟩. For this case, the switching transistors, M_2 and M_4, are directly attached to the cell output. Using Algs. 4.1 and 4.2 to generate a SCSM for ⟨0b→o⟩ results in the same model equation for output current of (4.17) and output charge of (4.22) because the resistive trees, RT_o, are identical. This implies that the active transistor is not part of the corresponding measurement. However, since the DC drain current of a conducting transistor equals its negative source current, there is no problem for the output current. The inaccuracy arises from neglecting the charging and discharging of the internal transistor stack node, net_2. To include all charges at net_2 for the port charge, Alg. 4.1 is modified to consider conducting transistors as resistors iff their channels are in the path between switching transistor and output node. The decision is made automatically. It is based upon the identified CMOS structure and also works for complex stacks with parallel and serial combinations. With this modification made to the algorithm, (4.27) now contains all

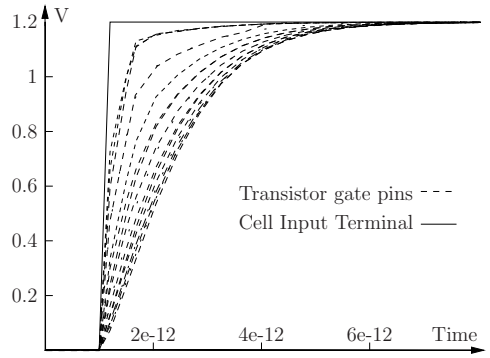


Figure 4.18.: Inverter input step response

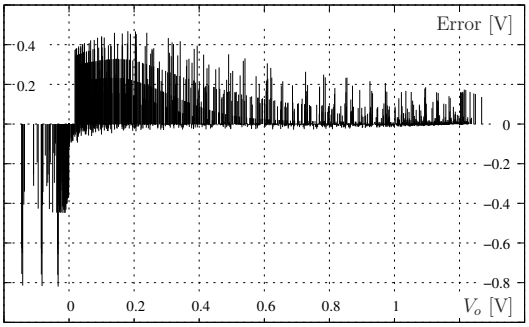


Figure 4.19.: Large transient voltage differences for stack node net_2

necessary charge contributions.

$$\begin{aligned}\hat{I}_o &= I_d(M_1) + \overbrace{I_s(M_2) + I_d(M_2)}^{\text{cancel out}} + I_d(M_3) + I_d(M_4) \\ &= I_d(M_1) + I_d(M_3) + I_d(M_4) \end{aligned} \tag{4.26}$$

$$\begin{aligned}\hat{Q}_o = &\overbrace{Q_d(M_1) + Q_s(M_2)}^{\text{new}} + Q_d(M_2) + Q_d(M_3) + Q_d(M_4) + \\ &+ C_1 V_{o2a} + \underbrace{C_2 V_{net_2a}}_{\text{new}} + C_3 V_{o2a0} + C_5 V_{oa0} + C_{10} V_{ob3} + C_{13} V_{odd} \end{aligned} \tag{4.27}$$

$$\begin{aligned}\hat{Q}_b = &Q_g(M_1) + Q_g(M_3) + \\ &+ C_8 V_{b1a0} + \underbrace{C_9 V_{b1b3} + C_9 V_{b3b1}}_{\text{cancel out}} + C_{10} V_{b3o} + C_{11} V_{b1dd} + C_{12} V_{bss} \end{aligned} \tag{4.28}$$

Accounting for the additional charges of stacked nodes substantially improves accuracy. Nonetheless, the more transistors are between the output node and the active transistor, the larger the violation on the quasi static relationship between internal node potentials and port voltage. Figure 4.19 shows the voltage errors of up to 800 mV for net_2. Moreover, such large errors exist for a large number of time points. To restore the algebraic dependency of model components, internal stack nodes must be added to the model. Investigations performed by [Mas09] on splitting the stack and modelling relevant transistors individually showed that high accuracy can be obtained at the expense of more simulation time.

Multi-Stage Cells

Noninverting cells like BUFFERs or ANDs consist of multiple channel connected blocks (CCBs). Similar to inner nodes of stacked transistors, it is crucial to model charging and discharging of cell internal nodes between CCBs [MKA08]. Modelling each CCB by a separate SCSM, which is shown in Fig. 4.21, allows to reuse the discussed algorithms and to avoid lookup tables of higher dimension. Identifying the transistor contributors for

each block is straight forward, but ambiguities exist for separating the parasitic elements. While the logical net between two CCBs consists of several electrical nodes, the DCAT topology analysis focuses on the logical connection and randomly selects the name of one electrical node. To achieve balanced trees for each CCB, the logical net is analysed and split at the electrical node with largest distance to all resistively connected transistors. Thereafter, resistive trees for driving CCB and for receiving CCB are constructed by reusing Alg. 4.1. Finally Alg. 4.2 is applied to create the measurement statements. However, there are several points to discuss for the charge components. The first is that the internal charges are connected in parallel to the same electrical node, as shown in Fig. 4.21. Consequently, if a linear capacitor is connected to both resistive trees, its charge contribution is always zero. This is similar to a capacitor that is connected to one resistive tree by its two pins. Consequently, charge contributions are only added if the capacitor is connected to only one of the trees by exactly one pin.

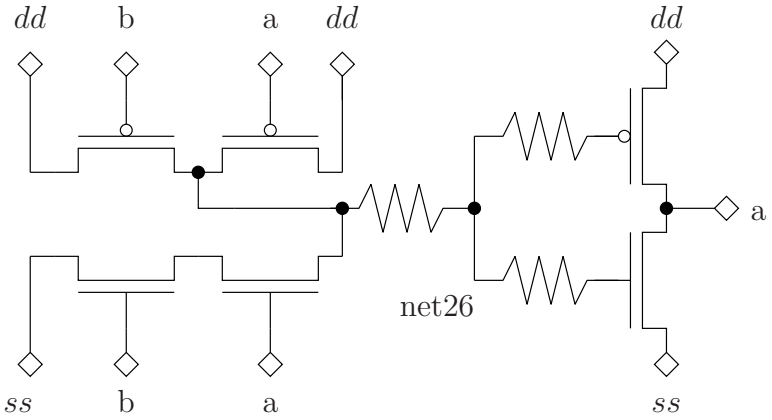


Figure 4.20.: Resistive network of AND

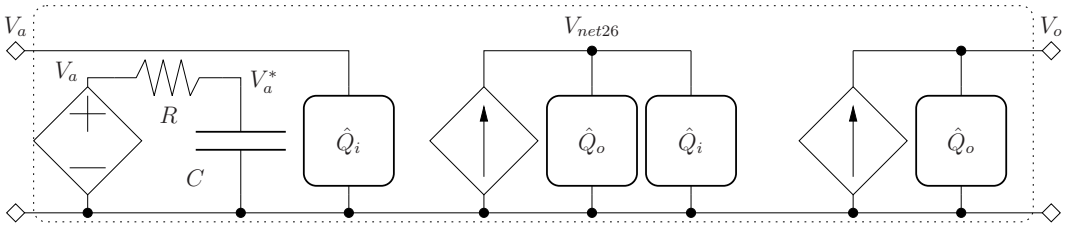


Figure 4.21.: SCSMs for non-inverting cells with two stages

A second concern are capacitors that are not connected to input and output nets of one CCB. There is no problem if the other pin remains at constant potential, which is the case for V_{dd}, V_{ss}, or a constant side input pin. However, capacitors whose other pin connects to another switching net require separate consideration. Since the created charge then depends on more than two voltages, characterisation must be adjusted to perform nested sweeps to create lookup tables of higher dimension. Alternatively, these contributions are grouped in additional 2D VCQs [Mas09]. Moreover, since these

charges exclusively stem from linear capacitors, a better solution is to simply include these capacitors in the model. It should be noted that this problem of wide-spanning capacitors was never observed for any of the industrial standard cells used for this work.

Parameter Variation

The cost of generating current source models which support analyses of parameter variation is directly related to the simulations required for the nominal CSM. This is because sensitivity calculation in Spice simulators is usually limited to DC and AC analyses [BJV06]. All CSM methods which use transient simulations require a re-characterisation for a different parameter value, followed by calculating the finite difference (4.29) for each model component.

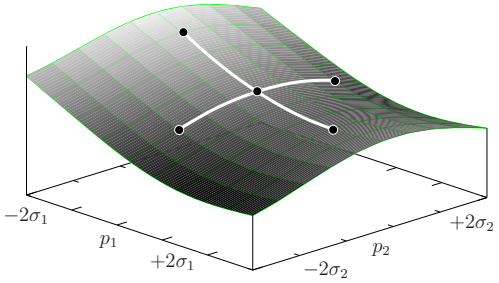
$$S_p^X \approx \frac{X(p + \Delta p) - X(p)}{\Delta p} \tag{4.29}$$

The same holds for this approach because sensitivity computations can only be performed for expressions which are based upon circuit variables. The measurement statements (4.17) and (4.21)–(4.22) contain transistor internal quantities, and hence sensitivity computation is not supported by the Spice simulator. Nonetheless, because all components are obtained in a single DC simulation, characterisation times are small. This allows the calculation of finite difference. For each parameter, two additional simulations are performed with a deflection of $\pm 2\sigma$. Using this method, linear or quadratic sensitivities can be generated. However, as Fig. 4.22(a) implies, also such a quadratic model neglects cross terms. This method is not limited to process parameters but includes temperature and supply voltage drop. Figures 4.22(b)–4.22(d) show typical sensitivity tables for the output current.

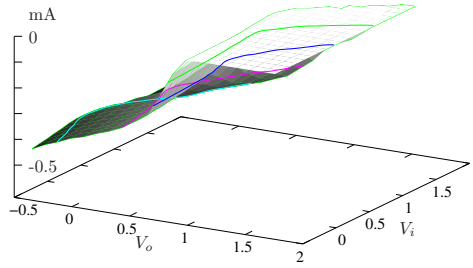
Transistor Aging

Traditionally, ensuring reliable operation of devices over a fixed period of time focused on catastrophic failures due to electromigration or dielectric breakdown. For advanced technology nodes also wear-out phenomena, denoted as aging, become more severe. These effects cause an increase in cell delay and hence can lead to malfunction. The dominating aging effect is negative bias temperature instability (NBTI) [Hua+09, Wan+10]. NBTI affects PMOS transistors. When the drain-gate voltage of a transistor is positive, the electric field causes the generation of charge traps at the Si/SiO_2 interface. The effect increases the threshold voltage, $v_{th} + \Delta v_{th}$, which leads to a decreased output current and hence an increased cell delay. Its extent depends on various stress conditions like switching activity, temperature, or stress time. The precise value is calculated using vendor specific degradation equations such as (4.30). The required values for operating conditions (OC) and workload (WL) over time can be obtained for each cell in a design by using tools such as the one described in [LGS10].

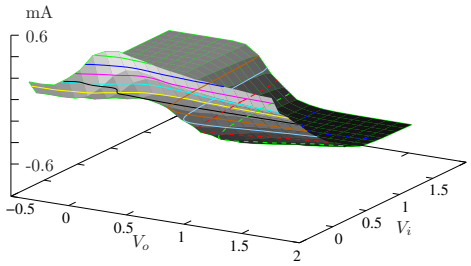
$$\Delta v_{th} = f(OC, t, WL) \tag{4.30}$$



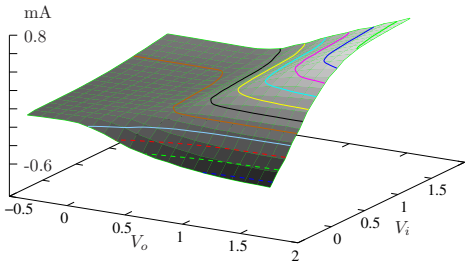
(a) Finite differences to generate linear sensitivities



(b) Supply voltage V_{dd}



(c) NMOS threshold voltage



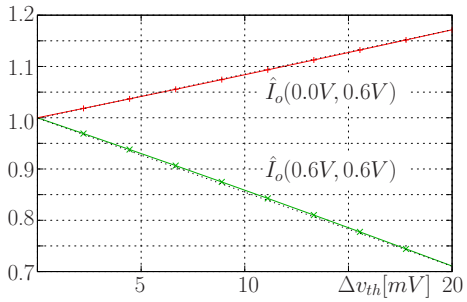
(d) Channel width

Figure 4.22.: Sensitivity lookup tables for the output current

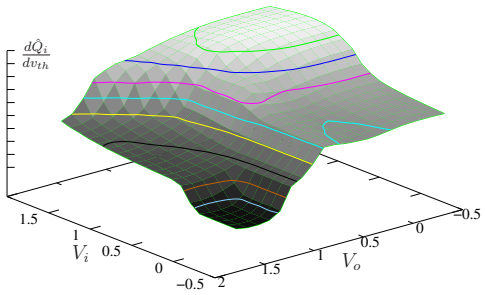
Since each value of Δv_{th} is different for each cell instance in a design, the SCSM must be parametric in Δv_{th}. The influence of NBTI on the SCSM is modelled, similar to process variation, as LUTs with linear sensitivities of model components $\hat{I}_o$, $\hat{Q}_o$, and $\hat{Q}_i$ with respect to Δv_{th}. After running the Spice simulations to obtain the nominal LUTs, the threshold voltage of all PMOS transistors are modified automatically. Their values are deflected by some reasonable amount Δv_{th}. The precise value of Δv_{th} is not crucial since degradation is almost linear (see Fig. 4.23(a)). With this modification made to the original cell description, another Spice simulation is made for this "aged" cell, and thereafter the linear sensitivities of model components with respect to Δv_{th} are obtained through finite differences of (4.29). Figures 4.23(b)–4.23(d) show the corresponding lookup tables.

4.6. PXM – the SCSM Power Model

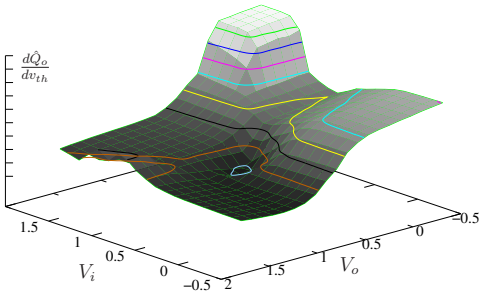
It has been pointed out that library generation causes significant efforts and must be repeated for obtaining noise, power, and timing models. The SCSM is a timing model. Since it supports arbitrary waveforms, it can also be used as noise model. It cannot model power consumption because PG pins are not part of the SCSM. A very accurate cell model would combine power and timing to account for their mutual dependencies [BJV06]. The SCSM at least models the static supply voltage drop of up to 10 % sufficiently accurate by LUTs of linear sensitivities. It is described in this section how also dynamic voltage variations are modelled by the newly developed Pull-



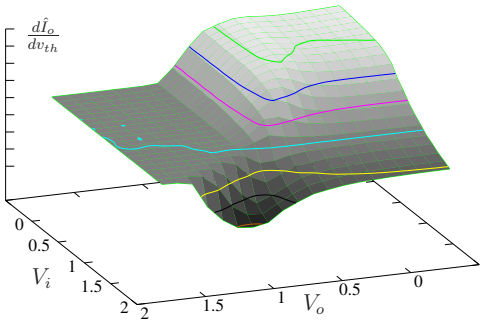
(a) Normalised output current for different NBTI drifts and operating points



(b) Δv_{th}-Sensitivity of $\hat{Q}_i$



(c) Δv_{th}-Sensitivity of $\hat{Q}_o$



(d) Δv_{th}-Sensitivity of $\hat{I}_o$

Figure 4.23.: Sensitivities of SCSM components with respect to Δv_{th}

up/Pull-down Model (PXM). It is based upon the same principles as the SCSM, but is a combined power and delay model. Hence, the PXM accounts for the influence of supply voltage variations on cell delay and accurately models the supply currents drawn by a logic cell.

CSMs that include PG pins typically model *dd* and *ss* as general output pins. This leads to additional CSM components and increased LUT dimension. The systematic modelling method avoids these problems by generating a physically consistent model for each CCB. The new model is shown in Fig. 4.24. Similar to the SCSM of Fig. 4.10(a), there are input and output charge, $\hat{Q}_i$ and $\hat{Q}_o$. However, there are now separate components for pull-up network and pull-down network. Consequently, the model is called Pull-up/Pull-down Model (PXM). The voltage-controlled current sources model the static current for each of the PG pins. Their sum yields the output current of the stage, and hence no additional VCCS is needed. The charges, $\hat{Q}_{dd}$ and $\hat{Q}_{ss}$, account for transistor and parasitic charges at *dd* and *ss*. They yield a dynamic current into the supply nets.

Finding the model equation for each PXM component is identical to the SCSM components. Hence, the Algs. 4.1 and 4.2 are used for signal and PG pins. The resulting PUN and PDN components for the NOR example on page 48 are listed in (4.31)–(4.34).

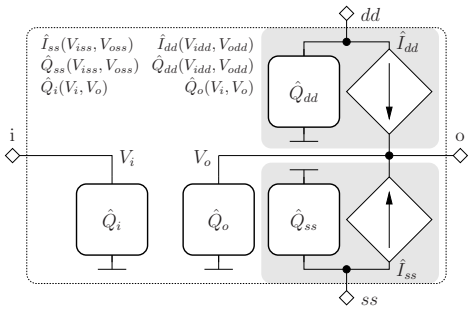
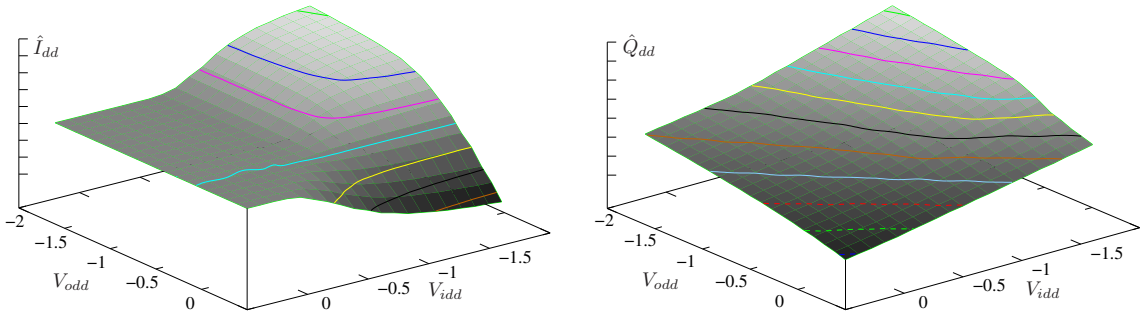


Figure 4.24.: PXM model for CMOS logic cells



(a) Typical $\hat{I}_{dd}$ lookup table

(b) Typical $\hat{Q}_{dd}$ lookup table

Figure 4.25.: PXM lookup tables

The other charges, $\hat{Q}_i$ and $\hat{Q}_o$, are identical to $\hat{Q}_a$ and $\hat{Q}_o$ in (4.21) and (4.22).

$$\hat{I}_{dd} = I_s(M_1(V_{net_2}, V_{b_1}, V_{dd})) \tag{4.31}$$

$$\hat{I}_{ss} = I_s(M_4(V_{o_3}, V_{a_4}, V_{ss})) + I_s(M_3(V_{o_3}, V_{b_3}, V_{ss})) \tag{4.32}$$

$$\hat{Q}_{dd} = Q_s(M_1(V_{net_2}, V_{b_1}, V_{dd})) + C_{11}V_{ddb1} + C_{13}V_{ddo} \tag{4.33}$$

$$\hat{Q}_{ss} = Q_s(M_4(V_{o_3}, V_{a_4}, V_{ss})) + Q_s(M_3(V_{o_3}, V_{b_3}, V_{ss})) + C_4V_{ssa4} + C_{12}V_{ssb} \tag{4.34}$$

The second import step is to avoid 3D or 4D lookup tables for (4.31)–(4.34). This is achieved by recognising that all quantities for the PUN depend on V_i, V_o, and V_{dd}, but are independent of V_{ss}. Similarly, the PDN components are functions of V_i, V_o, and V_{ss}, but not of V_{dd}. Furthermore, the transistor quantities are functions of voltages instead of absolute potentials. Hence, each voltage-controlled element in the PUN is a function of input-to-rail and output-to-rail voltage, $V_{idd} = V_i - V_{dd}$ and $V_{odd} = V_o - V_{dd}$. Figure 4.25 shows typical lookup tables whose indexes span from $-1.6 = -0.4 - 1.2$ to $0.5 = 1.7 - 1.2$. Using voltages instead of potentials not only allows to model dynamic variations of supply voltages, but also generates models that are independent of V_{dd}. Therefore, no re-characterisation must be done for different supply voltages.

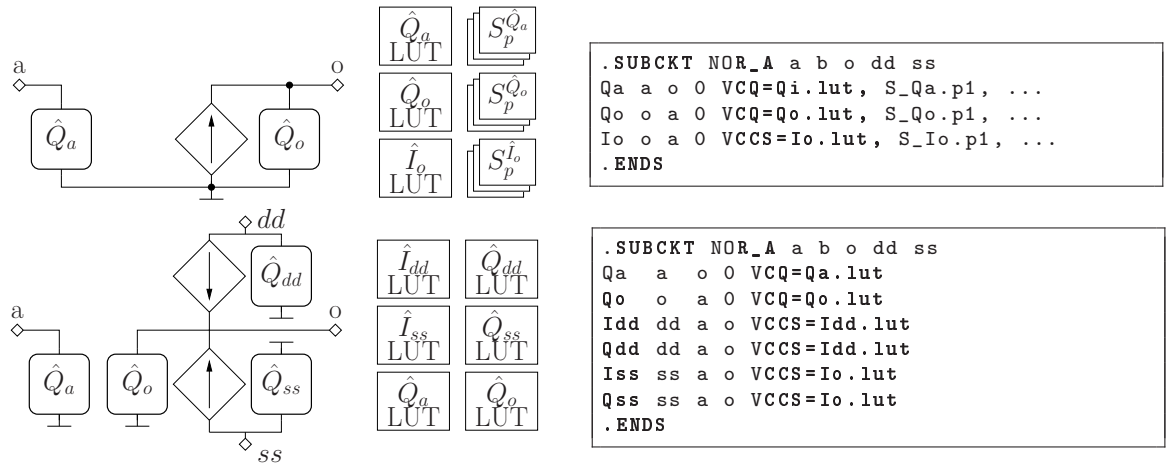


Figure 4.26.: Generated SCSM and PXM with Spice netlist and lookup tables for nominal values and linear sensitivities

4.7. Summary

After reviewing different CSM approaches in Sections 4.1 to 4.3, Sections 4.5 and 4.6 presented two new current source models, SCSM and PXM. Their principle is to identify and model the origins of port currents in a systematic, fully automated way. Figure 4.7 at the beginning of Section 4.5 summarises the characterisation flow. All necessary information is obtained from the transistor netlist of a logic cell and the timing arcs in the .lib-file. The topology is analysed first to identify CCBs and to recognise the transistor structure. Thereafter, measurement statements for SCSM or PXM components in each of the CCBs are derived automatically from the transistor netlist. DC voltage sources are attached to the CCB and swept from V_{ss} to V_{dd} to execute the measurements. This generates the nominal data for the LUTs as shown in Fig. 4.26. To obtain LUTs of linear sensitivities of model components with respect to process parameters, each parameter is deflected individually and the Spice simulation is repeated. Finite difference is used to generate the sensitivities. To obtain a parametric model in terms of NBTI, the threshold voltages of the PMOS transistors are modified, the cell is resimulated and the sensitivity is calculated. The lookup tables are written to ASCII files and are referenced by the SCSM/PXM components in the subcircuit model. Depending on the targeted application, the new subcircuits can be written to easily replace original cell instances in a larger netlist.

Implementing an efficient and accurate STA capability is one of the major challenges in developing an EDA system.

Keller, Tam, and Kariat

5 Realisation and Application of Current Source Models

Despite the number of different current source models, their application has been hardly reported. In addition to characterisation complexity, efficient CSM implementations are required to deliver simulation speedup to the designer. In this chapter, different CSM realisations and applications are discussed. After reviewing existing approaches, Sec. 5.2 focuses on the integration of SCSM and PXM into existing SPICE simulators. Section 5.3 then describes a dedicated statistical timing analyser using SCSMs.

5.1. Existing Approaches and Applications

Since CSMs are primarily intended for accurate timing analysis, most work is published in this field. [CW03] presented the corresponding simulation framework for their simple model. Similar to an STA engine, each cell is simulated separately. A simulator with variable time step and secant-based NLAE solver predicts the waveform at the output of a cell. Thereafter, the voltages at the far end of the interconnect are calculated by recursive convolution. The results are limited to simple single cells. [Ami+06] also mention a prototype of a stage-based solver which was used for timing analysis of a microprocessor block. The cell types were limited to single-stage cells. One CSM application is in simulating clock meshes [VFHL10]. Such structures cannot be handled with ordinary timing models since multiple cells drive a common net. The authors replaced buffers and inverters with CSMs and used hspice for simulation. [KAMC07] propose an STA-like simulator which uses the NR method for cell evaluation. The Jacobian is evaluated at most once per time step to reduce computational costs. The results contained only single cells. Later in [MKA08], the authors demonstrated the capability of the cell to

simulate current consumption. [VSB08] is almost an STA implementation to propagate Weibull waveforms. The simulator uses simple a CSM with a VCCS implemented as bicubic spline interpolation. Capacitors are time-dependent to model two parts of the output transition. Thereafter, the output waveform is shifted in time.

Because CSMs support arbitrary signal waveforms, they are also used for noise analysis. Compared to existing linear noise models, CSMs reflect the nonlinearity of holding resistance through the voltage-controlled current source. [GDTB09] employed a model with separate CSMs for pull-up network and pull-down network for the noise analysis. To preserve efficiency, this accurate model is only used for large induced voltage spikes, for which traditional linear noise models are too inaccurate. [CKSB06] use CSMs of AOIs, INVERTERs, and NANDs for worst case alignment of crosstalk noise. They employ the analytical expressions in Mathematica. [Das+11] used their Imodel for a crosstalk-aware timing analysis.

Dedicated simulators have been developed to address statistical analysis. [LK06] present a simulator to propagate mean and standard deviation of each voltage point in a waveform. The results are based upon INVERTERs and NANDs. Unfortunately, no propagation is shown. [FNP06] showed results for two single gates. Time points of crossing several voltage levels are modelled as Markovian process. The same authors applied their CSM to simulation of power nets in [FNP07]. The example consists of ten cells. [Zol+07] used a very simple CSM with constant capacitors to derive an analytical expression for the variational waveform. The idea is to shift the waveform in time to model delay variation, and to model slew rate variation by stretching the waveform. The examples consist of ten stages using INVERTERs, NORs, and NANDs. [GV08b] used a more complex CSM. Time points of crossing voltage levels are modelled as linear or quadratic functions of process parameters. The simulator is implemented in Matlab and uses single gates. [TZBM10] uses Matlab to calculate variation waveforms based upon simplified transistor models. The examples consisted of INVERTERs and NANDs.

[RVBG08] presented a fast statistical timing simulator using simplified transistor models instead of CSMs. The key component to achieve fast simulation is an efficient threading algorithm which allows effective parallelisation.

5.2. Current Source Models for Spice simulators

Performing delay calculation with CSMs requires the same algorithms which are used in Spice simulation, and a CSM can be regarded as transistor model for logic cells. Consequently, it is reasonable to employ CSMs in already existing simulators. By replacing the logic cells in a transistor netlist, simulation times for mixed signal circuits can be reduced. Furthermore, these tools provide sophisticated analyses and methods, and the designer is usually content to stay with familiar tools.

Implementing the nonlinear, voltage-controlled components is the most challenging part. Usually, the available controlled network primitives only allow to express dependencies as polynomials of one variable [Qim08]. More modelling functionality is provided by HDLs for analog components such as Verilog-A and VHDL-AMS. Additionally, most

```
1  module CSM(in, out, gnd);
2      inout in, out, gnd;
3      electrical in, out, gnd;
4      real id, qd, qr;

6      analog begin
7          id = $table_model(V(in), V(out), "id.tbl", "L");
8          qd = $table_model(V(in), V(out), "qd.tbl", "L");
9          qr = $table_model(V(out), V(in), "qr.tbl", "L");
10         I(out, gnd) <+ id;
11         I(out, gnd) <+ ddt(qd);
12         I(in, gnd)  <+ ddt(qr);
13     end
14 endmodule
```

Listing 5.1: Verilog-A implementation of a CSM

Model	time steps	iterations	CPU time [s]
BSIM	845	2511	0.726
Verilog-A	838	2015	2.901

Table 5.1.: CPU for transient analysis using Verilog-A Models

simulators provide interfaces to extend the component library through compiled models (CMs). Both possibilities are discussed in this section.

VHDL-AMS Models

VHDL has been extended to model analog and mixed signal components [KZ04]. Across and through quantities of component pins are declared by their electrical properties, and the circuit equations are formulated inside the component [Pla08]. In practice however, most simulators do not provide the full functionality of the VHDL-AMS reference manual. Additionally, there is no efficient interpolation for CSM lookup tables.

Verilog-A Models

Verilog-A is less equation-focused than VHDL-AMS. Instead, a module can directly source currents to nodes or provide output voltages [Vlr]. This component-based modelling is well suited for CSMs. Listing 5.1 shows the CSM as a Verilog-A module. Data for $\hat{I}$ and $\hat{Q}$ can be provided as ASCII files. The command *$table_model* is used to perform bilinear interpolation on the LUTs. Unfortunately simulation speed of this modelling is not sufficient since it seems to be more intended for modelling lager blocks rather than standard cells. Table 5.1 compares the CPU time spent on a transient simulation for an inverter chain of ten minimal inverters. BSIM stands for the simulation with transistor models [Bmm]. Clearly, the minimal inverter with only two transistors is the hardest performance test case. The simulations with CSMs and BSIM consist of about the same number of time points, for which the CSMs even require 20 percent

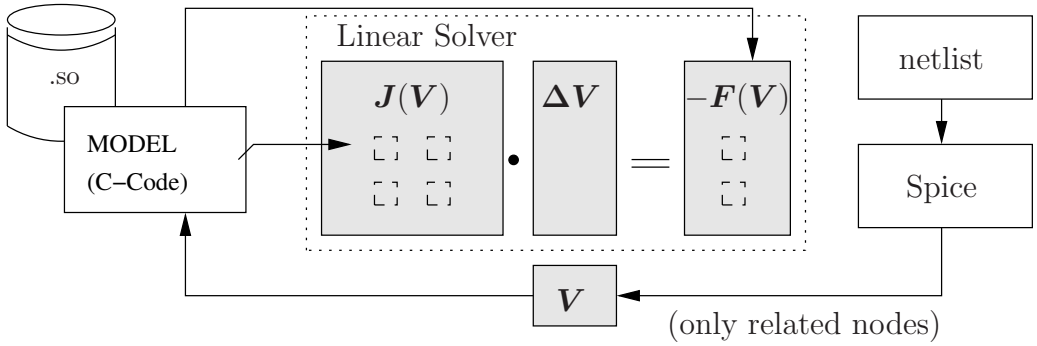


Figure 5.1.: Interaction between compiled model (CM) and simulator

fewer iterations. However, it is four times slower than BSIM. Reasons might be the interaction to simulator or inefficient interpolation.

Compiled Models

Most Spice simulators provide application programming interfaces to allow the integration of user-defined models for circuit components. For Cadence tools, this is the compiled model interface (CMI) [Cad04a]. The Synopsys equivalent is named user model interface (UMI) and for Titan that is Z-element model specification (ZMS). In all cases, the model is a C function which is compiled into a shared library. The latter is dynamically linked to the simulator. CMs are similar to analog HDLs. However, they go beyond describing voltage and current relations. Instead, CMs have direct access to the Jacobian matrix and simulator flags and hence interact much closer with the simulator.

A CM consists of several setup functions which are needed to register the new model to the simulator. This includes number and nature of pins, internal electrical nodes, and other internal variables. It further includes a dependency description of the model equations that are known to the simulator. It is the base for reserving the required matrix elements in the Jacobians. Additionally, each CM provides an *evaluate*-function. It is called every iteration for each model instance and contains the real model equations.

Figure 5.1 visualises the interaction of CM and simulator kernel. If the netlist contains instances of compiled models, the simulator calls the required model functions, which are available in a shared object. For model evaluation, all necessary data is passed to the model. These are flags to indicate the analysis type, simulation time, temperature, and instance parameters. Most importantly, it provides the vector of circuit variables, $\boldsymbol{V}$. Similar to Verilog-A modules, the response quantities are calculated. For the CSM, these are static and dynamic port currents. It is returned to the simulator which adds it to the corresponding node equation. More precisely, the current contributions are added into one element in the RHS of the equation system. Additionally, the CM calculates the entries for static and dynamic Jacobian matrices.

CMs offer most flexibility for modelling since virtually everything can be done within the C-functions. On the other hand, more effort must be spent on developing a proper

CM that allows efficient time step control and has good convergence. Some of the issues related to CSMs are discussed now.

Lookup tables in Spice-like simulators

As described in Sec. 3.1, most Spice simulators use the Newton-Raphson-Algorithm (NR) to solve the NLAE. It is a gradient-based method and requires function derivatives. The derivatives do not necessarily have to be exact but must point towards the final solution of the NLAE. The function should be continuously differentiable to allow convergence. That is, the function must be differentiable and its derivative must be continuous over the defined range. A function, $f(V)$, is continuously differentiable over the interval, R, if the limit exists for every point in R. This requires more explicitly that the limit from left always equals the limit from right.

$$f'(V) = \lim_{\Delta \to 0} \frac{f(V-\Delta)}{\Delta} = \lim_{\Delta \to 0} \frac{f(V+\Delta)}{\Delta} \quad \forall V \in R \tag{5.1}$$

A lookup table is a set of tuples $(\boldsymbol{V}_i, f(\boldsymbol{V}_i))$. The $\boldsymbol{V}_i$s define a discrete space. For all values of $\boldsymbol{V}$ that are not in this space, a LUT-function, $\mathcal{F}(\boldsymbol{V})$, must calculate an approximated value.

The simplest approach with least computation is the Snap-to-Grid method. The value of $\mathcal{F}(\boldsymbol{V})$ is the stored value $f(\boldsymbol{V}_i)$ at the closest point $\boldsymbol{V}_i$ in the LUT. This approach is not suited for NR for two reasons. First, the function $\mathcal{F}$ is not continuous over the defined LUT range.

$$\exists \boldsymbol{V} | \lim_{\Delta \to 0} \mathcal{F}(\boldsymbol{V}-\Delta) \neq \lim_{\Delta \to 0} \mathcal{F}(\boldsymbol{V}+\Delta) \tag{5.2}$$

Second, even in ranges where the function is continuous, its derivative is always zero. This causes singular Jacobian matrices. Approximating the LUT by a globally defined function solves these problems. Typical approaches are spline interpolations of higher order or polynomial approximation by for instances Legendre polynomials. A further advantage of the latter is that simple analytical expressions are usually faster to compute than accessing several memory entries. On the other hand, polynomials have the trade-off problem of approximation errors for the table points, $\mathcal{F}(\boldsymbol{V}_i)$, for low orders, and oscillation between the $\mathcal{F}(\boldsymbol{V}_i)$ if higher orders are used. This issue holds especially for the DC operating points [SW01]. An alternative are radial base functions, which consist of the superposition of globally defined functions that substantially affect only small areas in the LUT [SW01, Bin08]. Radial base functions are well suited for NR because they are differentiable and are globally defined. If the NR iteration exceeds the range of the lookup table, this method still yields reasonable values to ensure convergence. Due to the superposition of locally effective functions also very rough surfaces can be approximated.

For the implementation of SCSM and PXM, bilinear interpolation is used. It offers good accuracy at low computational cost. In fact, most of the time is spent on the four memory accesses. Bilinear interpolation also ensures accurate approximation for values at the table entries. Finally, it is well suited to handle parameter variations.

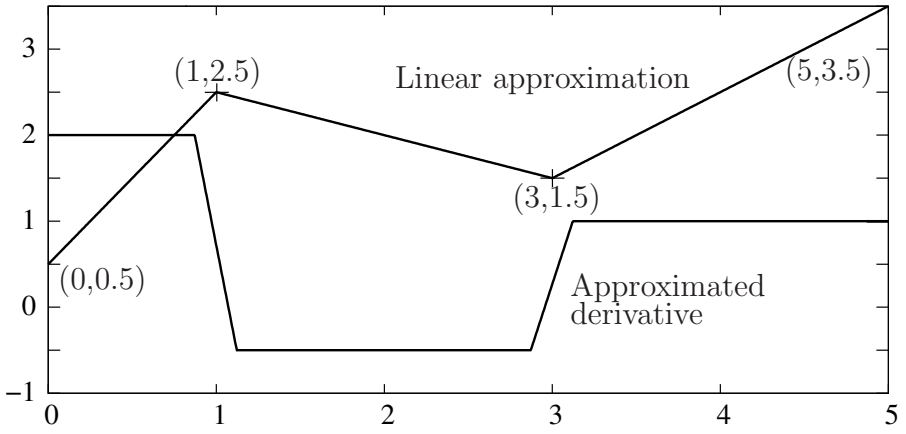


Figure 5.2.: Derivative approximation for bilinear interpolation

As shown in Fig. 5.2, bilinear interpolation can ensure steadiness of $\mathcal{F}$ and its derivatives. While the first is obvious, the steadiness of the derivatives requires an approximation since a continuous, piecewise-linear function is not differentiable in the break points. However, when calculating the derivatives as the RHS of (5.1), the deviation, Δ, is set to a reasonably large value. Of course, this causes the estimate for $\mathcal{F}$' to be not exact in points close to $\boldsymbol{V}_i$. However, NR requires exact values for the function evaluation, not the derivatives. Instead, this approximation replaces the discontinuity of the exact derivative in $\mathcal{F}'(\boldsymbol{V}_i)$ by a smooth transition as shown in Fig. 5.2. In practice, the SCSM LUTs are much smoother than this example, and hence the differences in derivatives are much smaller. (See Fig. 4.13 on page 54).

The disadvantage of bilinear interpolation is the large amount of memory for storing the LUT. Nonuniform grid may be used to reduce the amount of data, especially for 3D and 4D LUTs [GS10]. On the other hand, more calculations are required to access values during model evaluation [Nad+03]. For the SCSM the observation has been made, that reasonable table sizes do not significantly affect the overall runtime.

Table 5.2 compares simulation times for different CMs against transistor simulation of a minimal inverter. The transient simulation was conducted with fixed time steps and constant voltages to avoid other influences such as cache misses. The model evaluation of an empty CM, that is a CM consisting of functions without any calculation or data, provides a practical upper bound for the achievable speedup of 34 x. Since the SCSM consists of three voltage-controlled elements, the simulation is repeated for three empty CMs. The gain decreases to 27 x. The current SCSM implementation uses 50x50 lookup tables. The required memory footprint reduces the speedup to 17 x. This is already half of the maximum and yet there is no computation for model evaluation. If only the resulting currents are computed, the speedup drops to 10 x. Due to the simplicity of bilinear interpolation, this time is dominated by memory accesses. Consequently, the additional computation of derivatives is relatively cheap. The final performance for replacing a minimal inverter by a SCSM is around 7 x.

Simulation	Time [s]	Speedup [x]
Empty compiled model, no model data, no computation	1.47	34.76
3 instances of above test case	1.86	27.47
Full SCSM with LUT data but no computations	2.91	17.56
Full SCSM but only RHS computation, no derivative computation for Jacobian	5.38	9.50
Full SCSM with all computations	7.05	7.25
Transistor netlist of minimal inverter	51.10	ref.

Table 5.2.: Expectable speedup with ZMS interface

Modelling Parameter Variations

The SCSM models parameter variations by lookup tables of linear sensitivities. Parameter values are either defined in the netlist or a MC sample is generated during the initialisation phase of a single transient analysis. If a parameter differs from its nominal value, it creates additional charges or currents according to

$$\mathcal{F}(\boldsymbol{V}, \Delta p) = \mathcal{F}^{nom}(\boldsymbol{V}) + \sum \Delta p_i \cdot \mathcal{F}^{p_i}(\boldsymbol{V}) \tag{5.3}$$

At circuit level, this can be modelled as a number of additional current sources. The implementation of (5.3) in the compiled model is straight forward. However, it requires to calculate the LUT interpolation, $\mathcal{F}^{p_i}$, for each parameter in every iteration. Similarly, the parametric contributions to the Jacobian entries must be calculated. Hence, the alternative of (5.4)–(5.5) is to update the complete nominal lookup table during initialisation. The effort of scaling and adding several matrices in the beginning avoids any additional interpolation in each iteration.

$$LUT^{*} = LUT^{nom} + \sum \Delta p_i \cdot LUT^{p_i} \tag{5.4}$$

$$\mathcal{F}(\boldsymbol{V}, \Delta p) = \mathcal{F}^{*}(\boldsymbol{V}) \tag{5.5}$$

The second method is obviously beneficial for higher numbers of iterations. The precise point of breaking even strongly depends on the LUT size and the computer's memory. These factors influence the times for coherent memory access, for copying entire LUTs, and almost random access into different LUTs. If local parameter variation is considered, the second method requires to store a separate LUT for each cell instance whereas the first methods only stores the parameter vector.

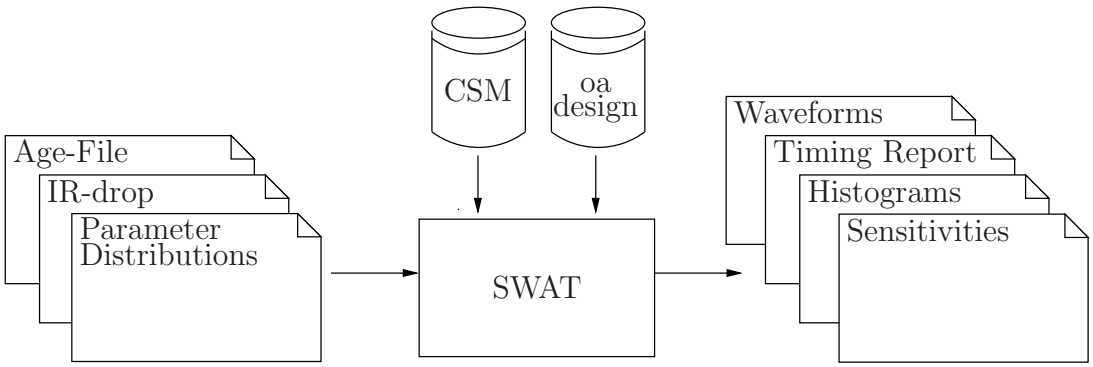


Figure 5.3.: SWAT application overview

5.3. Statistical Timing Analyser for Current Source Models

The integration of CSMs into Spice and FastSpice simulators allows mixed signal simulation and timing verification by path simulation. Despite significant performance gains, the simulation times are still too large for many use cases. The fundamental problem is that electrical simulation with CSMs, as described in Sec. 5.2, is still significant slower than STA tools. More optimisation can be done by exploiting circuit topology and by adapting the simulator engine to current source models.

The Simulator for Waveform-Accurate Timing (SWAT) has been developed as an STA that allows fast and waveform-accurate timing analysis. It supports MC analysis and temporal variation due to aging. Further, it provides a sensitivity propagation mode to analyse the effect of parameter variations without performing a high number of MC simulations. SWAT allows block-based and path-path timing analysis. Basis for the DC are SCSMs that drive interconnect and load models. Each cell is simulated individually. The output waveform is calculated by a highly optimised solver. The key idea is to exploit the known and simple circuit structure.

The SWAT usage in shown in Figure 5.3. SWAT is equipped with interactive command shell and scripting interface. The designs, including interconnect parasitics, are obtained from Cadence OpenAccess design database. This allows easy integration to tools that are already in use. Auxiliary information on parameter variation, v_{th} drifts, or local IR drops can be provided by the user. SWAT then performs the requested analyses and generates timing reports, waveform plots, histograms, or sensitivity information.

SWAT solver

Figure 5.4 shows the typical small circuit for which the output waveform has to be calculated. The driver consist of one or more abutted SCSMs. For the receiver cells, only the input charges are considered. The interconnect model can be a single grounded capacitor, a CRC Π model, or any arbitrary RCL topology.

A normal Spice simulator would create the corresponding system equation and Jaco-

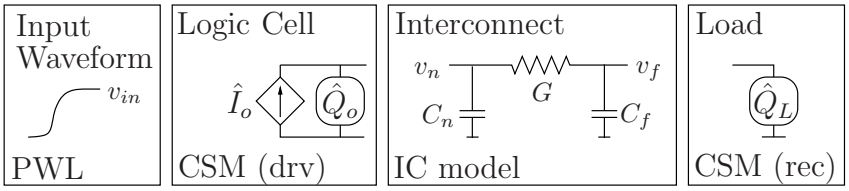


Figure 5.4.: Circuit to be simulated for delay calculation

bian during the initialisation phase. Matrix reordering might be done to avoid numerical problems. In case of large circuits, sparse matrices are used. This generality requires simulation time but is not required for the intended application. Due to the SCSMs and the stagewise simulation, circuit topology and therefore also circuit equations are known. Analysing the SCSM netlists revealed that 65 percent of the cells consist of one stage, and further 33 percent consist of two stages. This motivated to provide dedicated solvers for one- and two-stage SCSMs. The remaining two percent of cells with more than two stages are solved by a general solver, which is very similar to normal Spice.

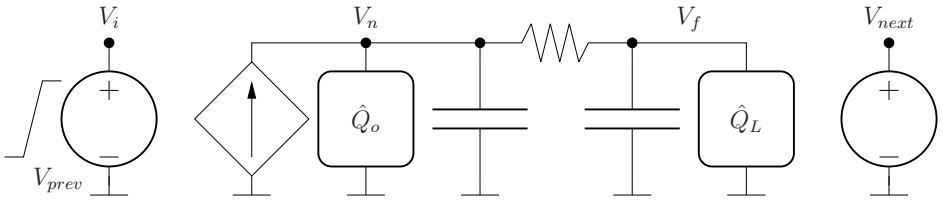


Figure 5.5.: Circuit for single SCSM and Π interconnect model

Simulating a single-stage cell (e.g. INVERTER) with a Π wire model, as shown in Fig. 5.5, results in the 3x3 nonlinear differential algebraic equation system of (5.6).

$$\boldsymbol{F}(\boldsymbol{V}(t+1,\boldsymbol{p}),\boldsymbol{p},t) = \begin{bmatrix} V_i - V_{prev} \\ \hat{I}_o(V_i,V_n) + G\cdot(V_n - V_f) + \frac{d}{dt}\left(C_n V_n + \hat{Q}_o(V_n,V_i)\right) \\ G\cdot(V_f - V_n) + \frac{d}{dt}\left(C_f V_f + \hat{Q}_L(V_f,V_{next})\right) \end{bmatrix} \tag{5.6}$$

V_{next} is the constant voltage of the node following the receivers. The value is known since every SCSM models an inverting stage. V_i is identical to the output waveform of the previous cell and would not be required. However, it is needed for the sensitivity propagation mode, which will be introduced later. The nonlinear algebraic equations are solved for near and far output node, V_n and V_f, iteratively by using NR method (5.7-5.8).

$$\boldsymbol{J}(\boldsymbol{V})\cdot\boldsymbol{\Delta V} = \boldsymbol{J}(\boldsymbol{V})\cdot\begin{bmatrix}\Delta V_i \\ \Delta V_n \\ \Delta V_f\end{bmatrix} = -\boldsymbol{F}(\boldsymbol{V}) \quad \text{and} \quad V_x = V_x + \Delta V_x \tag{5.7}$$

Solver	CPU time
Optimised	0.56 s
General	14.69 s

Table 5.3.: Solving times for small LAE system

with

$$\boldsymbol{J}(\boldsymbol{V}) = \begin{bmatrix} 1 & 0 & 0 \\ \frac{d\hat{I}_o}{dV_i} + \frac{d}{dt}\frac{d\hat{Q}_o}{dV_i} & \frac{d\hat{I}_o}{dV_n} + G + \frac{d}{dt}\left(C_n + \frac{d\hat{Q}_o}{dV_n}\right) & -G \\ 0 & -G & G + \frac{d}{dt}\left(C_f + \frac{d\hat{Q}_L}{dV_f}\right) \end{bmatrix} \tag{5.8}$$

Solving (5.7) for the Newton correction vector, $\boldsymbol{\Delta V}$, is required more than once for every time point of the waveform. The Jacobian of (5.8) always has this structure with $J_{11} = 1$ and $J_{12} = J_{13} = J_{31} = 0$, because every simulated single-stage cell instance results in the same NLDAE system. The solution for (5.7) is given in (5.9) and is directly implemented as C-Code. Further optimisation is done at the C-Code by identifying common subexpressions [ASU86].

$$\boldsymbol{\Delta V} = \begin{bmatrix} -r_1 \\ (J_{33} \cdot J_{21} \cdot r_1 - J_{33} \cdot r_2 + r_3 \cdot J_{23})/(-J_{32} \cdot J_{23} + J_{22} \cdot J_{33}) \\ -(J_{32} \cdot J_{21} \cdot r_1 - J_{32} \cdot r_2 + J_{22} \cdot r_3)/(-J_{32} \cdot J_{23} + J_{22} \cdot J_{33}) \end{bmatrix} \tag{5.9}$$

Equation 5.6 allows to simplify (5.9) further since $r_1 = V_{prev} - V_{in} = 0$ holds for most of the iterations. Within the code, the amount of calculations is small and hence efficient memory access is crucial. Usually general purpose solvers are optimised for larger matrices, resulting in large overhead for this tiny system. Instead, employing the circuit information allows to use very simple and fast data structures. Table 5.3 compares the time needed for updating the numerical values in the linear system and solving. Using the tailored solver accelerates this crucial part for the overall performance by a factor of 26 x.

The accuracy of SWAT is controlled by setting the fixed time step and tolerances for system variables. If latter are violated for any time point, the time step is reduced for the entire waveform and the delay calculation is repeated.

A common approach to reduce the computational cost of Spice-like simulations is to use simplified or constant Jacobians [Ste95]. This skips the computation of derivatives at the expense of more iterations to converge. This method is not employed in SWAT for two reasons. First, fixed time steps and the small dimensions result in quick convergence. Second, the sensitivity analysis, which is discussed later, requires the exact derivatives.

Waveform Model and Waveform Truncation

SWAT simulates the output waveforms for each waveform at the input pins. No approximation is made for the signal. Instead, as shown in Listing 5.2, the waveform is

```
1 class Waveform {
2     double      arrival_time;
3     double      time_offset;
4     double      timestep;
5     double*     voltage;
6     int         length;
7 };
```

Listing 5.2: Waveform Model

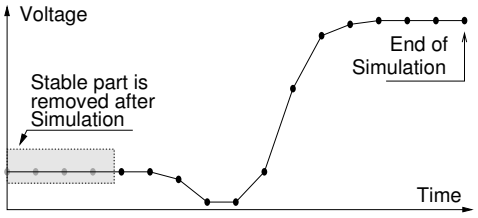


Figure 5.6.: Waveform truncation removes time points with constant voltage

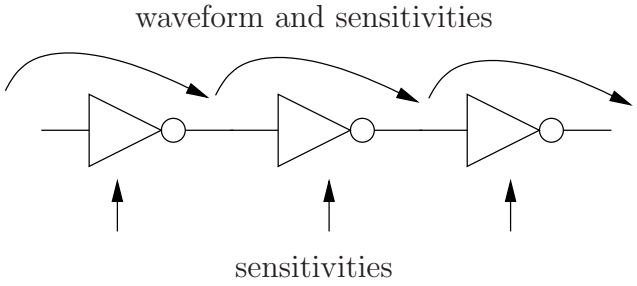


Figure 5.7.: SWAT sensitivity propagation

a vector of voltage points for fixed time steps. The latter can be different for different waveforms. Since there is significant latency in integrated circuits, the waveform is unchanged for most of the time. The important part of the signal is the transition, and hence only these time points are simulated. Once the output voltage settles at the final value, the simulation is stopped. Nonetheless, the waveform will contain a large number of points with constant voltage before the transition. As shown in Fig. 5.6, all these waveform entries in the shaded area are removed from the signal. The corresponding amount of time is added to the time_offset variable, which defines the absolute beginning of the waveform. The arrival time is computed as the time point of crossing $0.5V_{dd}$ plus time_offset.

Sensitivity Propagation

SWAT uses the parametric SCSMs. This allows to provide an individual set of parameters for each cell instance. Global and local parameter variations can be described in the statistic file. Also deterministic parameter changes for Δv_{th} drifts or lowered supply voltages are included. The STA is then rerun with the new parameters and histograms are generated. As alternative to repeating STA with varied parameters, SWAT also provides a sensitivity propagation mode. Here, the influence of parameter variations is determined within a single simulation by solving the sensitivity network. For example, this allows to quantity the impact of local V_{dd} drops or aged cells on the path delays. Consequently, the waveform model must be extended to express sensitivities. This is realised by replacing all floating point values in the normal waveform of Listing 5.2 by

```
class Waveform {
    SensVec     arrival_time;
    SensVec     time_offset;
    SensVec     timestep;
    SensVec*    voltage;
    int         length;
};
```

Listing 5.3: Variational Waveform Model

```
class SensVec {
    double         nominal;
    double[NUM_G]  gamma; // global
    double*        lambda; // local
    double         num_lambda;
};
```

Listing 5.4: Auxiliary Sensitivity Information

the sensitivity vectors of Listing 5.4. The resulting variational waveform model is shown in Listing 5.3. Each entry consists of a nominal value and two vectors which contain the linear sensitivities of the nominal value with respect to parameters.

During the simulation, the sensitivities in the output waveform must be calculated. Figure 5.7 shows that variation of voltage entries in a waveform result from two sources. First, the simulated cell instance introduces variation because their SCSM components are parametric. The second source is the input waveform whose voltage entries also contain sensitivities.

To compute the output waveform sensitivities, direct sensitivity computation is used (see Sec. 3.6 or [PRV95]). It is more efficient than the adjoint network method since the number of measures (i.e. points in output waveform) is large compared to the number of parameters. The sensitivity network is constructed by differentiating the circuit equations (5.6) with respect to the parameter. Original and sensitivity network use the same Jacobian matrix (5.8); only the RHS of (5.7) must be updated. This avoids a large amount of recomputation and memory accesses. Most importantly, the optimised code for solving the linear equations can be reused. The new vector of the RHS contains the sensitivities of CSM components with respect to the parameter, provided by LUTs. As Figure 5.7 shows, at each cell new waveform variation is introduced and variation of the input waveform is propagated. Calculating the sensitivity of the output waveform with respect to voltage variations of the input waveform employs the same algorithm. Since the current cell is not affected by local parameters of its predecessors, the sensitivity with respect to input waveform is calculated first. Thereafter, the variation is propagated from input to output by (5.10).

$$\frac{dV_o}{dp} = \frac{dV_o}{dV_i} \cdot \frac{dV_i}{dp} \tag{5.10}$$

Replacing the time derivatives in (5.6) by the Euler approximation yields the NLAE of (5.12).

$$0 = \boldsymbol{F}(\boldsymbol{V}(t+1,p),p,t) \tag{5.11}$$

$$\begin{bmatrix}0\\0\\0\end{bmatrix} = \begin{bmatrix} V_i - V_{prev} \\ \hat{I}_o(V_i,V_n) + G(V_n - V_f) + \frac{1}{h}(V_n - V_n^o) + \frac{1}{h}(\hat{Q}_o(V_n,V_i) - \hat{Q}_o(V_n^o,V_i^o)) \\ G(V_f - V_n) + \frac{1}{h}(V_f - V_f^o) + \frac{1}{h}(\hat{Q}_L(V_f,V_o) - \hat{Q}_L(V_f^o,V_o^o)) \end{bmatrix} \tag{5.12}$$

The equation system of (5.12) is differentiated with respect to p. Thereby, the chain rule dependencies must be obeyed. The resulting system is given in (5.13).

$$\frac{d\boldsymbol{F}}{dp} = \begin{bmatrix} S_p^{V_i} - S_p^{V_{prev}} \\ \frac{\partial \hat{I}_o}{\partial V_i} S_p^{V_i} + \frac{\partial \hat{I}_o}{\partial V_n} S_p^{V_n} + \mathcal{S}_p^{\hat{I}_o}(V_i, V_n) + G \cdot (S_p^{V_n} - S_p^{V_f}) \\ G \cdot (S_p^{V_f} - S_p^{V_n}) \end{bmatrix} + \\ + \frac{1}{h} \begin{bmatrix} 0 \\ (C_n + \frac{\partial \hat{Q}_o}{\partial V_n}) \cdot (S_p^{V_n} - S_p^{V_n,o}) + \frac{\partial \hat{Q}_o}{\partial V_i} \cdot (S_p^{V_i} - S_p^{V_i,o}) + \mathcal{S}_p^{\hat{Q}_o} - \mathcal{S}_p^{\hat{Q}_o,o} \\ (C_f + \frac{\partial \hat{Q}_L}{\partial V_f})(S_p^{V_f} - S_p^{V_f,o}) + \mathcal{S}_p^{d\hat{Q}_L} - \mathcal{S}_p^{\hat{Q}_L,o} \end{bmatrix} \tag{5.13}$$

The first equation introduces the propagated sensitivity from the input voltage. Second and third equations contain direct influences of parameter variation on the SCSM components. Their values are obtained from the sensitivity lookup tables, $\mathcal{S}_p^{\cdot}$. Additionally, the equations contain the sensitivities of node potentials. Whereas the sensitivities from the last time point, $S_p^{\cdot o}$, are known, the current sensitivities must be calculated. As shown in Sec. 3.6, the sensitivity system (5.13) reuses the Jacobian matrix of (5.7). Hence, in (5.14) only the RHS and system variables are changed.

$$\boldsymbol{J}(\boldsymbol{V}) \begin{bmatrix} S_p^{V_i} \\ S_p^{V_n} \\ S_p^{V_f} \end{bmatrix} = \begin{bmatrix} S_p^{V_{prev}} \\ -\mathcal{S}_p^{\hat{I}_o}(V_i, V_n) \\ 0 \end{bmatrix} + \\ + \frac{1}{h} \begin{bmatrix} 0 \\ +(C_n + \frac{\partial \hat{Q}_o}{\partial va}) \cdot S_p^{V_n,o} + \frac{\partial \hat{Q}_o}{\partial vin} \cdot S_p^{V_i,o} - (\mathcal{S}_p^{\hat{Q}_o} - \mathcal{S}_p^{\hat{Q}_o,o}) \\ +(C_f + \frac{\partial \hat{Q}_L}{\partial vb}) \cdot S_p^{V_f,o} - (\mathcal{S}_p^{\hat{Q}_L} - \mathcal{S}_p^{\hat{Q}_L,o}) \end{bmatrix} \tag{5.14}$$

Each considered parameter, p, introduces new sensitivities. It only requires to calculate the RHSs. The corresponding calculations for a two stage SCSM are provided in App. C.

Figure 5.8 shows the computation of the full output waveform. In every time point, the new nominal output voltage, $V_o(t)$, is found. Thereafter, its sensitivities with respect to global parameters, local parameters, and input voltage are calculated. The latter is used to propagate the local sensitivities that are already in the input waveform. Thereafter the simulation continues with the next time point.

Calculating the voltage sensitivities and obtaining a new waveform such as in Fig. 5.9(b) is valid if the resulting voltage changes are sufficiently small. The new arrival time can be obtained from the new waveform. For voltage waveforms of digital circuits, parameter variations ultimately result in delay variations or a time shift of the waveform. For the signal delay along several logic cells, this results in voltage variations which are as large as the full voltage swing. The obtained results are spurious.

To keep voltage sensitivities small, (5.15) calculates the sensitivity of the arrival time by scaling the voltage sensitivity [Sch08]. The minus sign indicates that a positive voltage

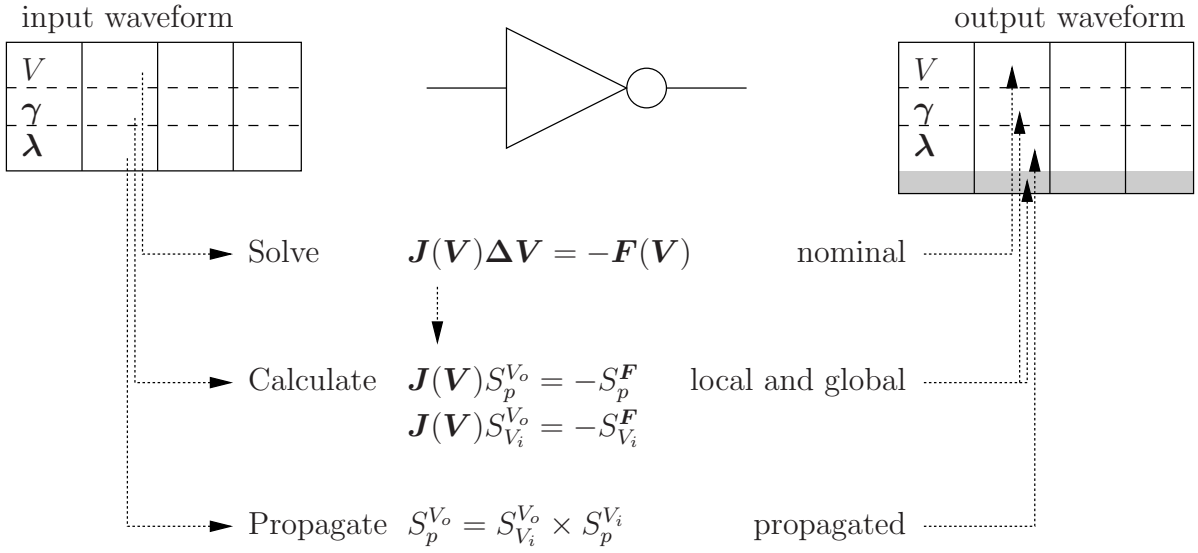
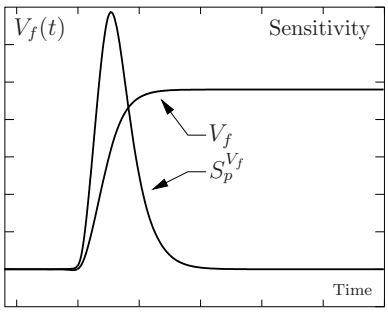
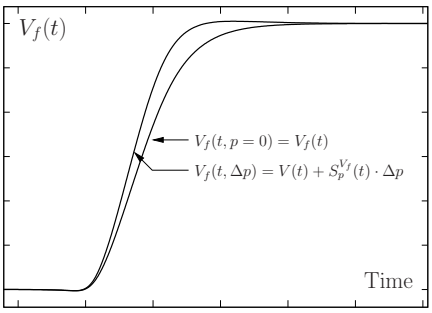


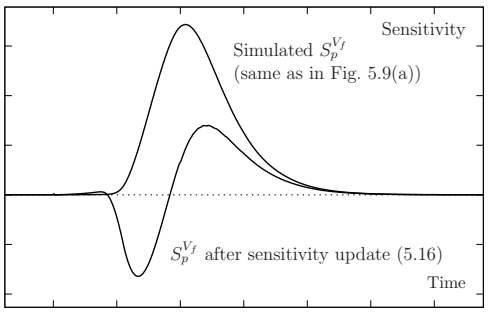
Figure 5.8.: Computation of output waveform that includes sensitivities



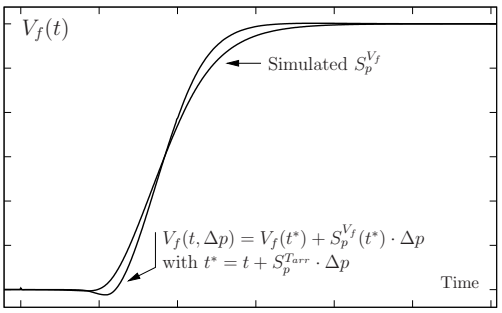
(a) Simulated output and sensitivity waveforms



(b) Nominal and varied waveform



(c) Modified voltage sensitivity after time shift sensitivity of whole waveform



(d) Varied waveform from updated sensitivities, time shift not shown

Figure 5.9.: Nominal and sensitivity waveforms after simulation and after time shift update

change in a rising/falling transition results in a smaller/larger arrival time. The arrival time sensitivity is used to express a time shift of the entire waveform.

$$S_p^{T_{arr}} = -S_p^{V(T_{arr})} \cdot \frac{dt}{dV(T_{arr})}\bigg|_{V=0.5V_{dd}} \tag{5.15}$$

The waveform data type of Listing 5.3 is amended to account for sensitivities of voltages and arrival times. The SensVec of Listing 5.4 consists of the nominal value and two vectors. $\boldsymbol{\gamma}$ contains a fixed number of voltage sensitivities with respect to global parameters. $\boldsymbol{\lambda}$ accumulates voltage sensitivities with respect to parameters that affect each cell individually. Its size increases with every cell in the path.

Since the sensitivity of arrival time expresses the time shift of the entire waveform, voltage sensitivities of each waveform entry must be updated according to (5.16).

$$S_p^{V(t)} := S_p^{V(t)} - \left(-\frac{dV(t)}{dt} \cdot S_p^{T_{arr}}\right) \tag{5.16}$$

The updated sensitivity waveform is shown in Fig. 5.9(d). Figure 5.9(c) shows the change in sensitivities more clearly. For the arrival time, the voltage sensitivity is zero since this voltage is entirely shifted in time. The other points now seem to raise or lower the voltage. Hence, the total modification of the waveform is realised by shifting each entry by the same amount in time, and applying an individual voltage shift.

There are two possible outcomes: if the result confirms the hypothesis, then you have made a discovery. If the result is contrary to the hypothesis, then you have made a discovery.

Enrico Fermi

6 Results

This chapter focuses on comparing accuracy and speedup of the current source models. In Sec. 6.1, the minimal inverter provides the base for an accuracy study of different CSM approaches. The new SCSM is analysed in more detail in Sec. 6.2. Most simulations were done using the Titan simulator. It also provides the references values for delay measurements using the transistor models (BSIM), as well as the required simulation time. Section 6.3 shows the accuracy for power model extension, and Sec. 6.4 investigates accuracy and simulation performance of SWAT.

6.1. Accuracy studies for different CSMs

Every CSM must be able to predict output voltage waveforms for typical input waveforms and output loads. The investigations are carried out using a minimal sized inverter, which is the simplest standard cell in a digital library. If a CSM is inaccurate for this cell, it is unlikely to be a good model for more complex cells.

Two kinds of simulations are performed for the analysis. The first is a transient simulation with a very noisy input waveform. Although this waveform is unrealistic, it reveals how versatile the CSM really is. The second test is a standard delay characterisation simulation. The inverter is fed by various input ramp signals and drives output loads of different sizes. These simulations have been done for the CSMs by Fatemi [FNP06], Amin [Ami+06], Kashyap [KAMC07], and the new SCSM and PXM. In addition to the delay and waveform errors, also the simulation performance is analysed.

Fatemi's CSM

Figure 6.1(a) shows the noisy input waveform and two output waveforms, CSM and BSIM reference. Both output waveforms have similar shapes. The CSM seems to be too

slow for the rising input whereas it is too fast for a falling input signal. These obvious differences translate to delay errors of 9–14 ps or up to 58 percent, listed in Tab. 6.1. Moreover, the relative errors are less than 10 percent only in the cases of rather large values of transition time and load.

Amin's CSM

Figure 6.1(b) shows the output waveform for the CSM by Amin. Again, the general trend is predicted but obvious deviations exist. The model seems to overreact and always produces to much current. This means the charges do not sufficiently compensate the DC current and hence, the delay is always underestimated. On the other hand, the model does not correctly reflect the input-output coupling. Over- and undershoots are too small. The achievable delay accuracy is about 3–5 ps, which correspond to 1.5–30 percent. The error is largest for normal loads and sharp signal transition.

Kashyap's CSM

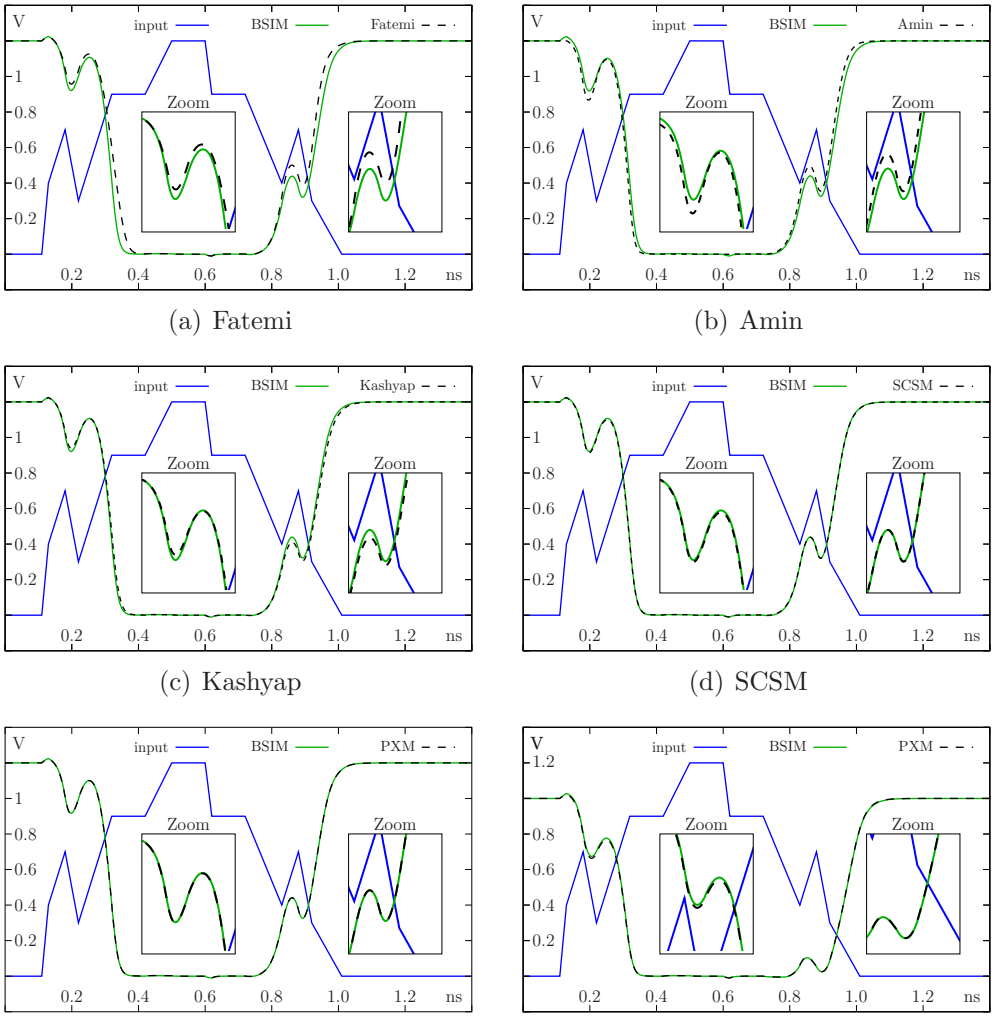
Model generation according to Kashyap uses AC analyses instead of transient simulations. The modification and separation of static and dynamic model components seem to pay off. The output waveforms in the noise test of Fig. 6.1(c) overlap almost perfectly. Deviations are only observable at the noise bumps during the transitions. Absolute delay errors are 1–3 ps and hence almost only half as large as for the other models. This translates to relative errors of less than 3 percent for most cases. However, delay error is also up to 14 percent for very sharp signal transitions.

SCSM and PXM

The same simulations are repeated for the new CSMs, SCSM and PXM. Almost no deviations can be seen in Figs. 6.1(d)–6.1(f). The models react just like the original circuit. Consequently, the delay errors in are less than 1 ps or less than 1.13 percent. The PXM further demonstrates to be accurate for lowered supply voltage.

Performance

The testcase of the minimal inverter is less suited to draw comprehensive conclusions for the overall performance. Nonetheless, a small comparison shall be made. All model components are two-dimensional LUTs, hence the model complexity only depends on the number of voltage-controlled elements. The CSMs of Fatemi and Amin, as well as the SCSM, consist of three elements, whereas the PXM and Kashyap's CSM have six and four, respectively. By comparing the simulation times of SCSM and the more complex PXM in Table 6.2, it can be concluded that the number of instances has only a minor influence for this small circuit. The table further compares other performance measures such as number of iterations and time steps. The CSMs of Fatemi and Kashyap have small problems with time step control and non-convergence, leading to an increased



(a) Fatemi (b) Amin

(c) Kashyap (d) SCSM

(e) PXM with $V_{dd} = 1.2\,\mathrm{V}$ (f) PXM with $V_{dd} = 1.0\,\mathrm{V}$

Figure 6.1.: Comparison of different CSMs to BSIM for noisy input

Testcase		Relative and absolute delay errors for different CSMs									
C_L	τ_{in}	Fatemi		Amin		Kashyap		SCSM		PXM	
$[fF]$	$[ps]$	[%]	[ps]	[%]	[ps]	[%]	[ps]	[%]	[ps]	[%]	[ps]
100	100.00	6.03	14.22	−1.51	−3.55	0.50	1.17	−0.14	−0.33	−0.05	−0.12
100	75.25	7.63	14.04	−1.92	−3.52	0.64	1.18	−0.16	−0.30	−0.06	−0.11
100	50.50	10.38	13.70	−2.65	−3.48	0.90	1.19	−0.20	−0.26	−0.08	−0.11
100	25.75	15.98	12.79	−4.24	−3.36	1.50	1.20	−0.27	−0.22	−0.13	−0.10
100	1.00	58.22	11.12	−30.37	−5.71	14.17	2.71	−1.13	−0.21	−0.64	−0.12
40	10.00	39.85	13.76	−9.67	−3.34	3.52	1.22	−0.37	−0.13	−0.24	−0.08
130	10.00	20.21	10.49	−7.05	−3.64	2.86	1.48	−0.45	−0.23	−0.22	−0.11
220	10.00	15.62	9.91	−6.68	−4.22	2.99	1.89	−0.56	−0.36	−0.23	−0.15
310	10.00	13.18	9.47	−6.53	−4.68	3.08	2.21	−0.66	−0.47	−0.25	−0.18
400	10.00	11.62	9.11	−6.51	−5.09	3.15	2.47	−0.78	−0.61	−0.27	−0.21

Table 6.1.: Absolute and relative delay errors for CSMs of minimal inverter for different combinations of slew rate, τ_{in}, and output load, C_L

	BSIM	Fatemi	Amin	Kashyap	SCSM	PXM
Time steps	1500	1587 /1.06 x	1500 /1.00 x	1582 /1.05 x	1500 /1.00 x	1500 /1.00 x
Iterations	2806	3309 /1.18 x	2768 /0.99 x	3113 /1.11 x	2864 /1.02 x	3789 /1.35 x
Runtime	0.18 s	0.06 s/2.85 x	0.18 s/0.98 x	0.06 s/3.00 x	0.06 s/3.05 x	0.07 s/2.39 x

Table 6.2.: Simulation performance for minimal inverter of different CSMs compared to BSIM. Less efficient VCQs implementation for Amin's CSM which causes the reduced speedup.

number of time step and iterations. This might be due to the very complex functions for the capacitors (see Fig. 4.4 and Fig. 4.6). Also the PXM shows bad convergence, which is likely due attaching two similar VCCSs to the same node. Despite these shortcomings, simulation speedup is achieved. Amin's CSM is not so much slower than the other CSMs, but a different implementation of the VCQs had to be used. The real speedup is expected to be similar to the SCSM. However, this example shows the importance of a good model implementation.

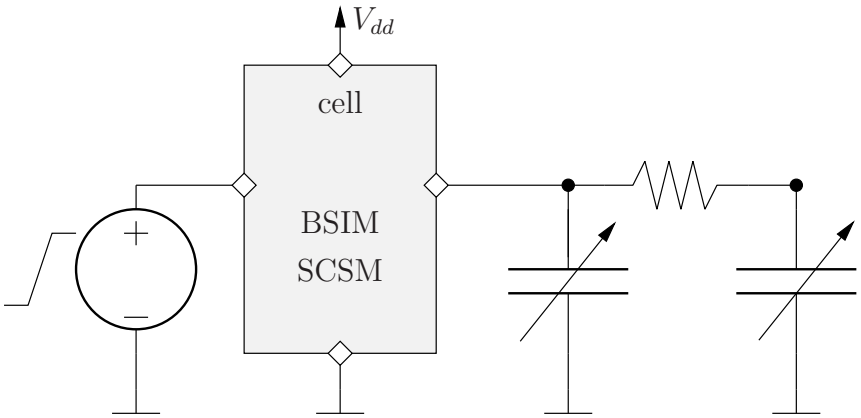


Figure 6.2.: Circuit for single cell tests

6.2. Detailed Analysis for SCSM

An SCSM library has been generated for 293 industrial 90 nm CMOS standard cells with extracted parasitics. Out of totally 535 SCs, this subset of combinational cells has no reconverging arcs in the signal flow graph. The cells have 1 to 8 input pins, and implement standard logic functions of different driver strengths and sometimes with partially inverted inputs. In total, there were 19 INVERTERs, 18 BUFFERs, 34 NAND, 27 NORs, 69 ANDs, 49 ORs, and 77 COMPLEXs cells, i.e. AND-OR-INVERT, OR-NAND combinations. The corresponding CSMs consist of one to three stages. The influences of the six most dominant process parameters, static supply voltage drop, and NBTI have been considered.

The SCSM have been tested using the setup of Fig. 6.2. The input voltage is a smoothed ramp signal of different slew rates. Each SCSM drives the typical CRC Π interconnect model.

Testing each cell individually

The SCSMs for every timing arc of every cell have been tested individually by performing 50 Monte Carlo (MC) runs for different combinations of input waveforms, CRC Π-loads, and process parameters. The histograms in Fig. 6.3 show relative delay and slew errors for these tests. For the majority of testcases the CSM delay prediction matches the BSIM reference. In 93.18 % of the testcases the delay error is less than 2 %, 99.58 % are within 5 % of BSIM. The error of output slew was less than 2 % for 96.54 % and less than 5 % for 99.86 % of all tests. The results have been analysed further in terms of input pin and cell type. The latter as well as the number stages have no significant influence on the delay error. Instead, model accuracy strongly depends on the number and position of the input pin. Cells with one input pin, i.e. inverters and buffer, have least errors. Accuracy decreases for cells with more pins. The important factor is the location of the switching transistors. The more transistors in the charging path between switching transistors and output node, the larger the delay mismatch. Also, the error is slightly

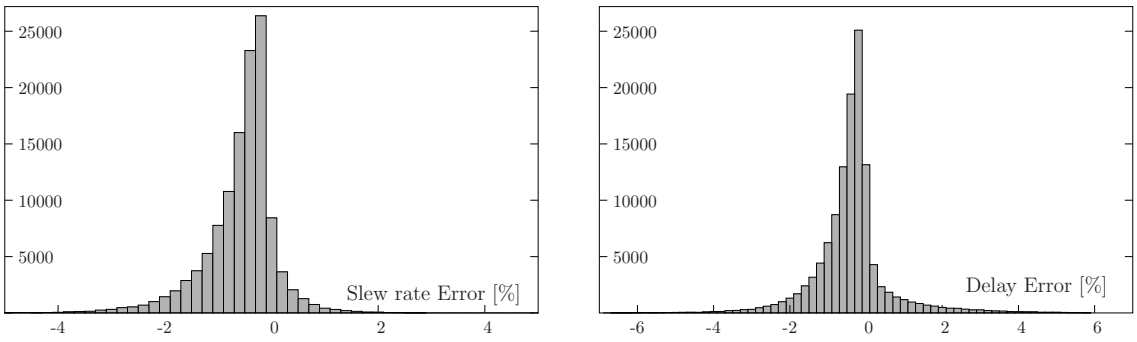


Figure 6.3.: Relative delay and slew errors for all cells with different CRC Π-loads, inputs slews, and parameter variation (50 MC runs per timing arc)

larger for NOR-type stages with PMOS-stacks than for NAND topologies. However, no conclusion could be drawn about systematic positive or negative delay deviations. The modelling error decreases with increasing output load. Hence, the SCSMs are best suited for simulation with realistic interconnect models.

Figure 6.4 demonstrates the capability of handling noisy input signals. In addition to Sec. 6.1, also random variations of one parameter, plot B, all parameters, plot C, and additional static V_{dd}-drop, plot D, are shown. Again, for all cases the waveforms overlap almost completely. This also visualises that first order sensitivities are suitable to capture parameter variations for the SCSM components. The same analysis is done to validate the NBTI sensitivity. Figure 6.5 plots the normalised delays of various types of logic cells for different combinations of slew rates and loads. The plot contains four measurements for each of those combinations. First, the new cell is simulated using the BSIM transistors and using the SCSMs. Thereafter, the cell is randomly aged and the simulations are repeated, again using BSIM and SCSMs. Figure 6.5 shows clearly how cell delays increase for the aged cell. The overlapping of measured delay values indicate the high accuracy of SCSM for both, new and aged cell.

Simulating Paths

The application of SCSMs in a path-based timing analysis is studied in this subsection. Therefore, a standard STA tool was used to identify critical paths in the benchmark circuits. Each path has been simulated with Spice (Titan) and FastSpice (UltraSim). The cell instances are modelled using transistors (BSIM) or using SCSMs.

Table 6.3 compares the predicted path delays and simulation times for 50 MC runs in Titan. Good accuracy is achieved with most mean errors being less than 1 %. The simulations could be accelerated by factors of 82–175 x. For the circuit c6288 this means a reduction from 3 days and 11 hours to 30 minutes. The correlation plot of path delays for c1355 in Fig. 6.6 shows that most errors are within 5 % while the maximum error is 8.9 %. Similar results are obtained for the other circuits.

The above studies focused on verifying the SCSM accuracy for long paths and with

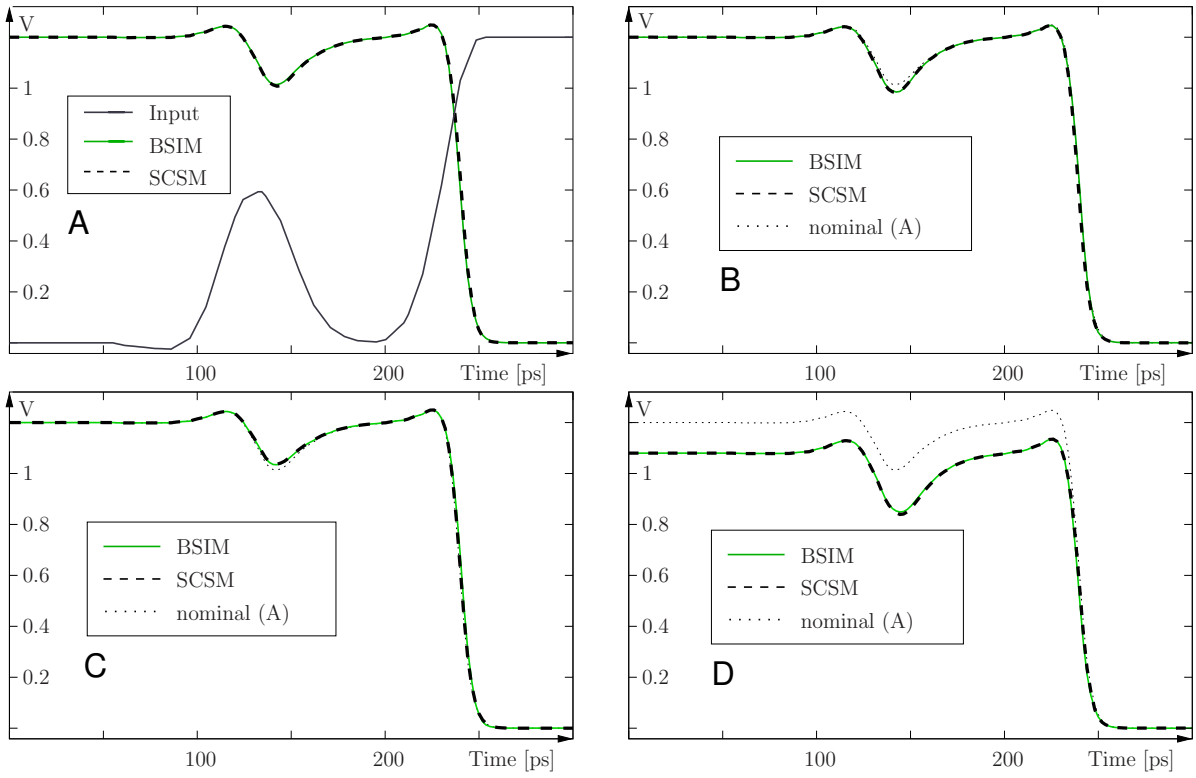


Figure 6.4.: Accurate waveform prediction in the presence of noise for nominal conditions (A), one altered parameter (B), all altered parameters (C), additional V_{dd}-drop (D)

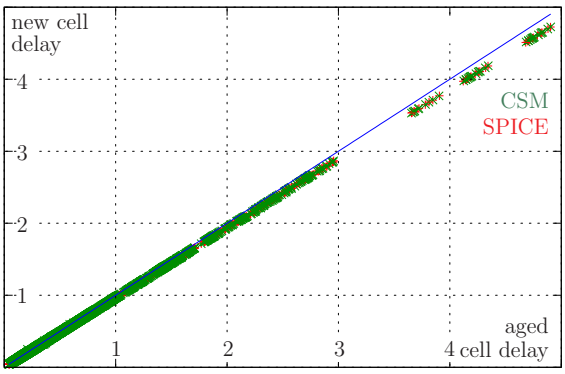


Figure 6.5.: Delay values for new and aged standard cells predicted by BSIM and SCSM

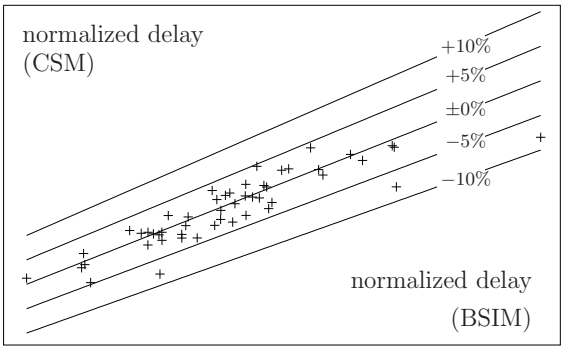


Figure 6.6.: Correlation plot of path delay variation for c1355

Circuit	Delay Error [%]		CPU-Time		
	mean	max	BSIM [s]	SCSM [s]	Speedup
c17	0.044	-1.762	151.32	1.21	125.06
c1355	0.753	8.883	14332.32	107.18	133.72
c880	0.076	8.339	15343.50	121.27	126.52
c1908	-0.499	8.642	23176.59	202.32	114.55
c2670	-0.519	5.542	14838.57	180.74	82.10
c5315	-0.159	9.472	16739.05	226.45	73.92
c6288	-3.159	-9.215	299763.30	1715.30	174.76

Table 6.3.: Simulation time and path delay errors of 50 MC runs in Spice (Titan) using transistors and CSMs

Circuit	Titan Runtime [s]			UltraSim Runtime [s]			Relative delay error [%]		
	BSIM	SCSM	BSIM/ SCSM	BSIM	SCSM	BSIM/ SCSM	Titan SCSM	UltraSim BSIM	UltraSim SCSM
c17	41.05	0.70	58.6	3.06	0.32	9.6	0.00	-1.46	-1.46
c880	2180.24	27.01	80.7	42.37	5.22	8.1	-0.31	-2.02	-2.02
c1355	2008.78	22.96	87.5	40.95	4.62	8.9	0.26	-2.71	-2.71
c1908	3473.82	41.60	83.5	35.67	7.46	4.8	-1.49	-2.33	-2.33
c2670	2197.10	39.33	55.9	32.12	7.35	4.4	-0.84	-2.95	-3.01
c5315	2742.89	47.27	58.0	38.09	9.34	4.2	-1.08	-3.07	-2.90
c6288	30865.54	140.04	220.4	1725.58	17.57	98.0	-2.86	-2.56	-2.56

Table 6.4.: Simulation times and delay errors of critical paths simulated in Spice (Titan) and FastSpice (UltraSim) using BSIM transistors and SCSMs

parameter variation. It has been further investigated if the SCSMs can improve existing tools used for timing analysis. FastSpice simulators provide the necessary functions for timing verification with transistor level accuracy [Cad03]. They apply circuit partitioning, use simpler device models and adaptively controlled explicit simulation (see Sec. 3.2). SCSMs further reduce the computational effort by combining several transistors of a logic cell into three lookup tables. Table 6.4 compares the simulation times and speedup factors for different models and simulators. As expected, Titan with BSIM models is prohibitively time consuming. Replacing the cells by SCSMs causes a significant acceleration of 60–80 x (circuit c6288 is discussed below). Simulation times are now of the same order as the UltraSim simulator with transistor models. These times can be further reduced by 4–10 x by using CSMs as cell models in UltraSim. Specially remarkable are simulation times and speedup for c6288. This circuit consists of many identical gates. Hence, in contrast to other circuits only a few CSMs must be held in memory during simulation, resulting in fewer cache misses and higher speedup. This effect can be illustrated by reducing the circuit size. Truncating the path to 50 % or 25 % decreases the speedup to 62.99 x and 38.14 x, respectively.

Table 6.4 further lists the relative path delay errors compared to Spice with BSIM models. Using a FastSpice simulator has caused more error than using SCSMs in Spice. Furthermore, using SCSMs in a FastSpice simulator did not result in noticeable additional errors.

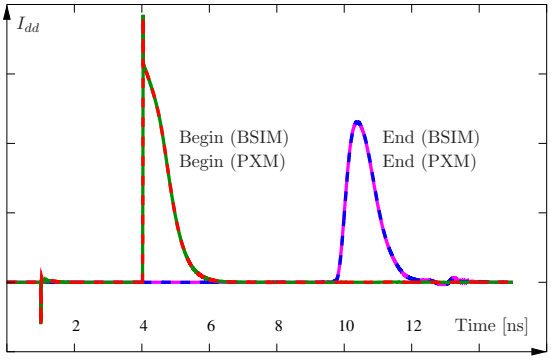
Cell	V_{dd}	Rel. Err. [%]: Delay		Rel. Err. [%]: $\int I_{dd}dt$		Rel. Err.[%]: $\int I_{ss}dt$	
		Mean	$P_{98\%}$	Mean	$P_{98\%}$	Mean	$P_{98\%}$
INVS	1.2	0.2	1.0	0.0	0.1	0.0	0.1
	1.0	0.3	1.1	0.1	0.3	0.3	1.0
	0.8	0.4	1.1	0.3	0.6	0.5	1.4
BUFS	1.2	−0.0	1.3	0.1	1.6	−0.1	1.9
	1.0	−0.1	1.8	−0.0	0.8	−0.2	0.6
	0.8	−0.2	2.8	−0.3	0.3	−0.4	1.2
NANDS	1.2	−1.2	0.3	−0.0	0.2	0.2	0.7
	1.0	−0.4	0.5	0.1	0.5	0.4	1.8
	0.8	1.9	8.4	0.1	0.8	0.6	2.6
NORS	1.2	0.0	0.6	0.4	1.9	−0.1	0.6
	1.0	0.5	3.1	0.7	4.3	0.1	1.4
	0.8	0.9	3.5	0.8	4.7	0.1	1.8
ANDS	1.2	0.2	6.8	−0.5	0.3	−0.5	4.3
	1.0	−0.2	1.5	0.1	0.5	−0.2	0.1
	0.8	−0.6	0.9	0.2	0.6	−0.0	0.2
ORS	1.2	−0.5	0.4	0.2	5.1	−0.9	0.6
	1.0	−0.6	1.1	0.2	9.6	0.3	3.6
	0.8	−1.0	0.7	−0.3	0.1	1.1	5.1
COMPLEX	1.2	−0.5	0.5	−0.2	6.4	−0.1	5.0
	1.0	−0.3	3.5	−0.6	2.4	−0.1	4.1
	0.8	−0.1	6.9	−0.0	1.1	0.0	2.4

Table 6.5.: Relative errors for delay and total supply currents. Each cell is simulated individually with different values of transition time, load, V_{dd}, and is compared with transistor simulation.

6.3. Detailed Analysis for SCSM Power model

Similar to the SCSM models, a library of PXMs for industrial 90 nm standard cells of different sizes and types has been generated with V_{dd}=1.2V. Their accuracy is studied only in the Titan simulator. Again, first each cell has been simulated individually with different input slew rates and output loads. In addition to cell delay also the total supply currents, $\int I_{ss/dd}dt$, have been measured. Table 6.5 lists mean and 98 percentile of the relative errors in percent, sorted by supply voltage. Very small average and maximum errors have been observed for most simulations. Nonetheless, for a few cases delay errors are close to 10 percent. It has been noted that accuracy is lower for stacked switching transistors and very small output loads. This is due to a notable signal delay between port voltage and voltages at the switching transistors. Consequently, there are differences in voltages and currents between DC characterisation and transient simulation.

In the next test, PXM accuracy and performance are analysed for simulating generic paths. Therefore, cell chains, powered by RC chains as PG model, have been simulated and compared for delay, energy consumption, and simulation speed. The chains consist of 20 cells to cover various values of V_{dd} and V_{ss} and different signal waveforms. Figure 6.7 shows typical supply current waveforms for $\hat{I}_{dd}$ and $\hat{I}_{ss}$ for the first and the last cell in a chain. PXM and BSIM reference simulation overlap almost completely. It is



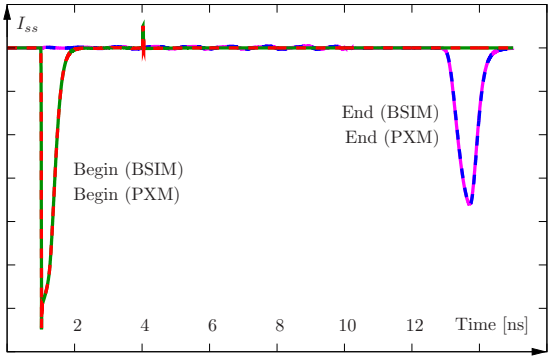


Figure 6.7.: Supply currents

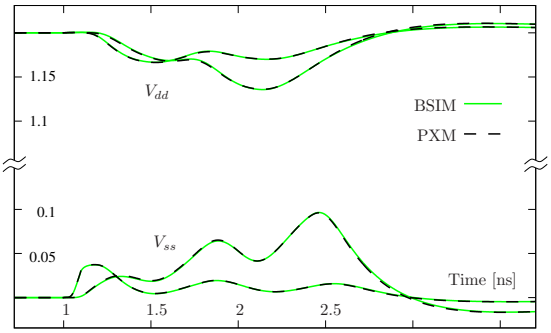


Figure 6.8.: Correct prediction of noise on RLC supply nets

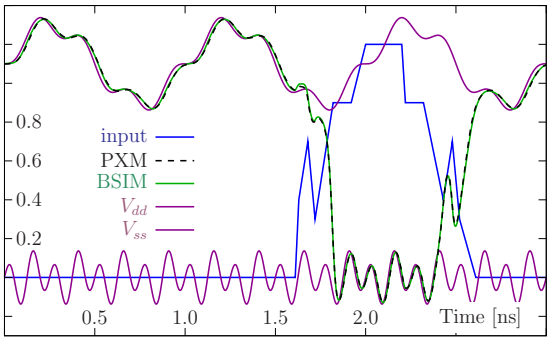


Figure 6.9.: Propagation of noise from V_{dd} and V_{ss} to the cell output correctly predicted

important to note the differences in PG current waveforms between the first and the last cell. They result from IR drop along the chain. CSMs that ignore this dependency would overestimate the peak currents but underestimate the path delay. This visual PXM validation corresponds to the very small errors for delay and energy consumption which are shown in Tab. 6.6. Depending on the gate complexity, Titan simulations are accelerated by up to 53 x, while the maximum errors in delay and energy do not exceed 1 and 3 percent, respectively.

Figure 6.8 finally shows the potentials for different nodes in the supply rails when using an RLC model for PG. Since the PXM correctly predicts supply and output currents, also the noise on supply nets is modelled with high accuracy. Similarly Fig. 6.9 shows that the PXM correctly models the influence of PG noise on the cell output. This testcase further included a noisy input signal as well as a lowered supply voltage. The model characterisation has been done with V_{dd}=1.2 V.

CELL	Speedup	Delay Error [%]	Energy Error [%]
INVS	32.24	≤2.62	≤0.92
BUFS	29.20	≤1.38	≤0.75
NANDS	27.24	≤0.56	≤0.67
NORS	52.81	≤0.82	≤1.90
ANDS	14.61	≤0.08	≤0.51
ORS	23.21	≤5.34	≤0.56
COMPLEX	31.50	≤1.19	≤1.89

Table 6.6.: Speedup and relative errors for delay and energy for simulating cell chains with supply networks (different driver strengths and cell types)

6.4. Accuracy and Runtimes of SWAT

This section presents accuracy and performance analysis of the Simulator for Waveform-Accurate Timing (see Sec. 5.3). Due to different target applications of SWAT and Titan, no runtime comparisons are made. Instead, the influence of different SWAT features on simulation time will be analysed.

First however, Tab. 6.7 compares arrival times of an inverter chain simulated in SWAT against the BSIM Titan simulation. For this, and for all other SWAT simulations, a time step of 1 ps is used. The nominal delay values have errors of less than 1 %. In addition, the linear sensitivities are correctly predicted. Their errors mainly result from the LUT approximation, not from the sensitivity propagation.

Net	T_{arr} [1e-10 s]			$^{dT}/_{dL}$ [1e-11]			$^{dT}/_{dTOX}$ [1e-12]		
	Spice	SWAT	Err [%]	Spice	SWAT	Err [%]	Spice	SWAT	Err [%]
5	4.01	3.98	0.70	1.98	1.94	−1.82	3.42	3.36	−1.87
4	3.18	3.15	0.76	1.64	1.57	−4.06	2.60	2.59	−0.52
3	2.49	2.47	0.72	1.17	1.12	−3.76	2.14	2.11	−1.57
2	1.59	1.58	0.79	0.75	0.72	−4.27	0.13	1.24	−0.92
1	0.90	0.90	0.22	0.30	0.29	−2.95	0.80	0.78	−2.26

Table 6.7.: SWAT sensitivity propagation compared with Spice

Table 6.8 shows simulation times for sensitivity mode and for 200 MC simulations for different circuits. The runtimes are influenced by the number of output pins and the lengths of corresponding paths. The longer the path, the more computations are necessary for the sensitivity propagation of local parameters. However, even for c6288 with 16 paths longer than 97 stages, the simulation completed within 28 seconds. The table further lists the relative errors of standard deviations for path delays. The average error is 4.1 %, and the sensitivity with respect to channel width is least accurate. The accumulated errors for other parameters do not exceed 10 % even for very long paths.

Circuit		Paths	Runtime [s]		Rel. Error of std. dev. for arrival times T [%]									
					TOX		VTH		L		W		RDSW	
Name	Size	#/Max./Avg./Med.	Sens	MC	Avg	Max	Avg	Max	Avg	Max	Avg	Max	Avg	Max
c880	435	26/ 30/ 13/ 7	3.8	103.1	2.9	6.1	2.8	3.9	1.3	3.1	4.5	10.4	4.0	5.7
c1355	590	32/ 29/ 29/ 29	6.6	134.7	0.2	0.2	2.0	2.0	3.3	3.3	11.4	11.4	5.3	5.3
c1908	1057	25/ 50/ 40/ 37	9.6	213.8	1.2	1.7	1.4	1.9	2.5	2.8	9.1	10.2	5.0	5.4
c3540	1983	22/ 57/ 32/ 39	13.5	368.7	1.2	3.0	2.6	3.4	0.4	0.4	7.5	11.8	3.7	4.9
c6288	2416	32/ 125/ 79/ 97	27.5	563.4	1.0	9.0	4.5	4.7	5.2	5.9	12.3	13.4	7.2	7.8

Table 6.8.: Runtime and delay error for SWAT sensitivity propagation vs. 200 MC runs

Table 6.9 analyses the performance gains of waveform truncation and optimised linear solvers. Column two lists the simulation times using the general solver and no waveform truncation. By only simulating the waveform transitions, simulation times are reduced by 5 x. This value depends on the number of skipped stable points. Hence, the longer the path, the higher the speedup. On the other hand, the optimised solvers accelerate solving every simulated time step. It further reduces simulation times by about 5 x. Together, both methods lead to performance gains of 18–34 x without sacrificing accuracy.

Ciruit	Simulation time [s] GS, no WT	GS, WT	OS, no WT	OS, WT	Speedup WT	OS	OS, WT
c880	6.7	1.6	1.4	0.4	4.1	4.9	18.0
c1355	9.2	2.0	1.6	0.4	4.7	5.9	24.1
c1908	14.7	2.7	2.7	0.6	5.5	5.4	24.1
c3540	29.0	5.5	6.2	1.3	5.3	4.7	22.5
c6288	52.0	8.3	8.7	1.5	6.3	6.0	34.0

Table 6.9.: Speedup contributions

Was ist der langen Rede kurzer Sinn?
Friedrich Schiller, "Die Piccolomini"

7 Summary

Accurate and fast timing analysis is a central component of every sign-off and optimisation tool. Over the years, various models and simulators have been developed, each finding a new trade-off between accuracy, versatility, complexity, and speed. With this regard, the emerging current-based timing models for standard cells are yet another trade-off solution. Current source models (CSMs) hereby span from analog to digital modelling and analysis. Despite being mainly developed as logic cell models for accurate timing analysis of ICs with resistive interconnects, CSMs are general enough to support crosstalk and even power analysis. In fact, CSMs can be used not only for static timing/noise/power analysis, but also as subcircuit models for a Spice simulation. Hence, a CSM is a versatile cell model that allows runtime reduction of Spice simulations and waveform-accurate timing analysis of digital integrated circuits.

A successful realisation of such analyses which are based upon current source models has four prerequisites.

- A conceptually simple cell model. The CSM complexity is the key factor to both characterisation effort and analysis performance. Unfortunate choices result in an unacceptably large number of complex simulations which are required for model generation. Furthermore, such CSMs result in large library sizes and complicate the delay calculation.

- A fast, automated, transparent, and adjustable characterisation method. The increasing number of process corners requires the generation of different model libraries for the same set of standard cells. Hence, usually CSM generation is no one time effort. If the characterisation is too complicated to be set up, or requires too many simulations, the cost of library generation is too high.

- Backward compatibility and integration. The new model shall be compatible to

existing tools and should not require a fundamental change of tool set and analysis methodology.

- Benefits. The real value of the CSM is measured during application. It must deliver benefits to the user such as reduced simulation times, the ability of handling larger designs, avoiding to change the simulator, higher accuracy, or completely new analyses which are only possible because of the CSM. These gains are compared against characterisation effort and modelling error.

This work is devoted to each of the above-mentioned points. All research is done with the focus on obtaining an industrially applicable current source model. This goal is reached, and the following contributions are made.

- Derivation of a physically consistent modelling method. Each component of the generated Systematic Current Source Model (SCSM) has explicitly known contributors in the standard cell transistor netlist. This results in simple yet accurate CSM structures without artificial dependencies or complexity.

- A highly efficient characterisation method is developed for model generation. It treats the standard cell as a white box and obtains required cell information by analysing the transistor netlist. This allows to create each model component individually and with minimal simulation effort. Still, the method is transparent and can be adjusted easily. The whole characterisation is fully automated and requires, besides the configuration, no user input. It is parallelisable even for a single cell, and characterisation is not only significantly faster compared to other CSMs, but in some cases even faster than the characterisation of current industrial standard, NLDM.

- Despite – or better because of – its simplicity, SCSM is a versatile model. Additional to supported timing and noise analyses, a SCSM power model, PXM, is derived using the same modelling principles. Hence, instead of generating separate models for each analysis, a single PXM can be used. The SCSM models are accompanied with sensitivity information to handle the effects of PVT variations. It is the first CSM which can model transistor aging due to NBTI. Similarly, the PXM is the first CSM with low complexity that can be used for different supply voltages without re-characterisation.

- SCSM and PXM are efficiently integrated into different Spice and FastSpice simulators, as well as a timing reference tool. A robust model implementation crucially affects achievable performance gains and usability. By supporting standard tools for analog and digital domain, SCSM and PXM are widely usable during back-end verification. The seamless integration provides new options to reduce simulation times, also for mixed signal designs. Different components can be modelled either by CSMs, transistors, analog HDLs, or other subcircuit models. Simulation times typically can be reduced by 60–80 x for Spice and 4–10 x for FastSpice simulators. The path delay errors typically are less than 5 percent.

- A separate statistical static timing analyser for waveform-accurate timing, SWAT, is developed. It is specially optimised for SCSMs and performs block-based or path-based analysis. The tool allows to assign individual parameters to each cell instance, including process parameters, supply voltage, and aging profile. In addition to Monte Carlo simulation, also the propagation of local and global sensitivities are supported. This allows to investigate the sensitivity of paths delays with respect to each cell. The runtimes for ISCAS benchmark circuits are only 0.5–3 s for purely nominal simulations and 4–27 s including sensitivity propagation.

In total, the work covers every step from model generation to application development. This wide range is needed to provide a consistent solution with low characterisation effort and fast models which support various analyses. Of course, further research for improvements can be done at any of the steps that were covered. So far, the SCSM is limited to combinational cells without reconverging signal paths. Lifting this restricting as well as an extension to sequential cells is both desirable and possible. More effort can be spent on very efficient implementation of lookup tables. Due to the link between SCSM/PXM lookup table and transistor models, there might be simple analytical approximation functions which could result in even higher speedup. Further tools can be built upon the SCSM and PXM, and, of course, accuracy should be increased where needed. Not to mention, there is still no commercial standard that allows CSM integration to EDA tools.

In sum, this work presented a modelling method, two models, a dedicated tool, and model implementations for electrical simulation. Based upon their application as well as due to changing requirements, there will arise both need and space for constant improvements.

A

Examples of standard cell libraries

A.1. Library Exchange Format

```
1  MACRO NOR2
2      CLASS CORE ;
3      ORIGIN 0.000 0.000 ;
4      SIZE 1.240 BY 2.520 ;
5      SYMMETRY x y ;
6      PIN z
7          ANTENNADIFFAREA 0.4600 ;
8          DIRECTION OUTPUT ;
9          PORT
10         LAYER M1 ;
11         RECT  0.910 0.710 1.050 2.050 ;
12         RECT  0.420 0.510 0.910 0.840 ;
13         ...
14         END
15     END z
16     PIN b
17         ANTENNAGATEAREA 0.1470 ;
18         DIRECTION INPUT ;
19         PORT
20         LAYER M1 ;
21         RECT  0.640 0.770 0.870 1.670 ;
22         END
23     END b
24     PIN a
25         ANTENNAGATEAREA 0.1470 ;
26         DIRECTION INPUT ;
27         PORT
28         LAYER M1 ;
29         RECT  0.440 0.970 0.490 1.550 ;
30         END
31     END a
32     ...
33 END NOR2
```

Listing A.1: LEF exmaple

A.2. Nonlinear delay model (NLDM) in Liberty format

```
cell (NOR2) {
  cell_leakage_power : 0.084 ;
  area : 2.5621 ;
  pin(a)  {
    direction : input;
    capacitance : 0.0034;
  }
  pin(b)  {
    direction : input;
    capacitance : 0.0036;
  }
  pin(o)  {
    direction : output;
    max_capacitance : 0.0851;
    function : "(!(a+b))";
    timing() {
      related_pin : "a";
      timing_sense : negative_unate;
      cell_fall(delay_template_7x7) {
        index_1 ("0.0052, 0.0411, 0.0568, 0.0846, 0.4701, 0.5188, 0.7056");
        index_2 ("0.0046, 0.0059, 0.0081, 0.0475, 0.0557, 0.0720, 0.4116");
      values("0.0090, 0.0427, 0.0498, 0.0556, 0.0640, 0.4457, 0.2250", \
          ...
          "0.0089, 0.0218, 0.0508, 0.0945, 0.4555, 0.2164, 0.5810");
      }
      fall_transition(delay_template_7x7) {
        ...
      }
      cell_rise(delay_template_7x7) {
        ...
      }
      rise_transition(delay_template_7x7) {
        ...
      }
    }
  }
}
```

Listing A.2: .lib exmaple

B

Overview on CSM approaches

B.1. Existing approaches

Paper	Model	Characterisation	Application	Notes
Croix and Wong, [CW03]	$\mathbb{I}_o$, C_o, time shift	DC, typical waveforms, LSF	separate simulators; SPICE-like simulator for cell with secant method; Interconnect simulator; AWE recursive convolution; deterministic solution several cell types	problem with multi-stage
Amin et al., [Ami+06]	$\mathbb{I}$ and $\mathbb{Q}$ per port	DC, step signals and current integration, LSF	own simulator, stage-based propagation, single-stage cells, analysis of micro processor	
Chopra, Kashyap, Su, and Blaauw, [CKSB06]	tanh current approximation, C_i, C_M, C_o	derived from ECSM/CCS lib	worst case alignment of noise from analytical hold resistance, Mathematica AOI, Inv, Nand	
Das, Scott, Nazarian, and Zhou, [DSNZ09]	Scaling function of I_{sat}	derived from NLDM lib	coupling aware timing analysis	
Liu and Kahng, [LK06]	[CW03]		propagation of μ and σ for time of each voltage point	INVERTER and NAND, no propagation showed
Fatemi, Nazarian, and Pedram, [FNP06]	$\mathbb{I}_o$, $\mathbb{C}_i$, $\mathbb{C}_M$, $\mathbb{C}_o$	DC, ramp functions	own simulator, variational waveform modelled as Markov chain, NAND, AOI	MC runs, [NFP11] sequential cells, [FNP07] short current

Paper	Model	Characterisation	Application	Notes
Mitev et al., [Mit+07]	finite points for interpolation for PUN and PDN current, then superposition, $\mathbb{C}$s	DC, typical waves and regression for $\mathbb{I}$	NAND(2–4), NOR(4), XOR, AOI	MC runs for sensitivity model; regression
Li, Feng, and Acar, [LFA07]	$\mathbb{I}_o$, $\mathbb{Q}_o$, $\mathbb{Q}_i$, lowpass filter	DC, ramp signals, LSF, fine tuning with typical waveforms	own simulator, clock mesh simulation	FD around full characterisation
Kashyap, Amin, Menezes, and Chiprout, [KAMC07]	$\mathbb{I}$ per port, $\mathbb{C}_{xy}$ across all ports	DC, AC, potentially in parallel	own simulator, single gates	power grid analysis in [MKA08]
Zolotov et al., [Zol+07]	$\mathbb{I}_o$, $\mathbb{C}_o$	nothing mentioned	variational waveform model of shifting and stretching; INVERTER, NAND, NOR, 10 stages	
Goel and Vrudhula, [GV08a]	$\mathbb{I}_o$, $\mathbb{C}_i$, $\mathbb{C}_M$, $\mathbb{C}_o$	step functions, current integration	Matlab simulator for variational waveforms, single gates, LUTs approximated by Legendre polynomials	linear or quadratic function of parameters
Raja, Varadi, Becer, and Geada, [RVBG08]	replace transistors, 3D LUTs, piecewise linear capacitors	DC	variational waveform simulator, own tool, threading	commercialised in clkda.com
Tang, Zjajo, Berkelaar, and Meijs, [TZBM10]	replace each transistor	DC, parameters from BSIM equations	Matlab simulator for variational waveforms, INVERTER, NAND(2)	
Gupta and Sapatnekar, [GS10]	$\mathbb{I}_o$, $\mathbb{Q}_o$	same as [Ami+06]	body bias, LUT reduction, sensitivity waveform for influence of body bias	

B.2. Newly developed methods

Paper	Model	Characterisation	Application	Notes
Knoth et al., [Kno+08]	symbolic transfer function	DC	Matlab simulator for single cells, INVERTER	no statistic
Knoth, Kleeberger, Nordholz, and Schlichtmann, [KKNS09b]	$\mathbb{I}_o$, $\mathbb{Q}_i$, $\mathbb{Q}_o$	DC	Spice	no statistic
Knoth, Eichwald, Nordholz, and Schlichtmann, [KENS10]	$\mathbb{I}_o$, $\mathbb{Q}_i$, $\mathbb{Q}_o$, lowpass	DC, AC	Spice, FastSpice, single paths of ISCAS85 circuits	FD, V_{dd}
Knoth, Uphoff, Kiesel, and Schlichtmann, [KUKS11]	$\mathbb{I}_o$, $\mathbb{Q}$	DC	SWAT simulator for sensitivity propagation, optimised solvers, ISCAS85 circuits	FD, V_{dd}, NBTI
Knoth, Jedda, and Schlichtmann, [KJS12]	$\mathbb{I}$ and $\mathbb{Q}$ for PUN and PDN, $\mathbb{Q}_i$, $\mathbb{Q}_o$	DC	Spice, timing, power, noise on power	no statistic

C

Sensitivity Calculation for two stage circuit

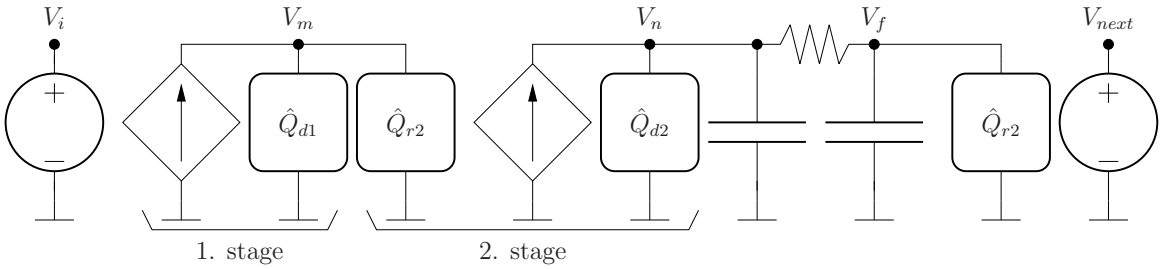


Figure C.1.: Simulation circuit for two-staged SCSM

The NLAE for a two stage cell is in (C.1)-(C.4)

$$0 = V_i - V_{prev} \tag{C.1}$$

$$0 = \hat{I}_1 + \frac{d}{dt}(\hat{Q}_{d1} - \hat{Q}^o_{d1}) + \frac{d}{dt}(\hat{Q}_{r2} - \hat{Q}^o_{r2}) \tag{C.2}$$

$$0 = \hat{I}_o + \frac{d}{dt}(\hat{Q}_{d2} - \hat{Q}^o_{d2}) + \frac{d}{dt}C_n(V_n - V^o_n) + G \cdot (V_n - V_f) \tag{C.3}$$

$$0 = G \cdot (V_f - V_n) + \frac{d}{dt}C_f(V_f - V^o_f) + \frac{d}{dt}(\hat{Q}_L - \hat{Q}^o_L) \tag{C.4}$$

(C.1)–(C.4) are differentiated with respect to p. For the first equation, this yields (C.5) and introduces the variation input voltage.

$$0 = S_p^{V_i} - S_p^{V_{prev}} \tag{C.5}$$

Differentiating (C.2) yields and includes chain rule terms as well as dynamic contributions.

$$
\begin{aligned}
0 &= \mathcal{S}_p^{\hat{I}_1} + \mathcal{S}_{\partial V_i}^{\hat{I}_1} S_p^{V_i} + \mathcal{S}_{\partial V_m}^{\hat{I}_1} S_p^{V_m} + \\
&\quad + \frac{1}{h}\left[(\mathcal{S}_p^{\hat{Q}_{d1}} + \frac{\partial \hat{Q}_{d1}}{\partial V_i} S_p^{V_i} + \frac{\partial \hat{Q}_{d1}}{\partial V_m} S_p^{V_m}) - (\mathcal{S}_p^{\hat{Q}_{d1},o} + \frac{\partial \hat{Q}_{d1}}{\partial V_i} S_p^{V_i,o} + \frac{\partial \hat{Q}_{d1}}{\partial V_m} S_p^{V_m,o}) \right] \\
&\quad + \frac{1}{h}\left[(\mathcal{S}_p^{\hat{Q}_{r2}} + \frac{\partial \hat{Q}_{r2}}{\partial V_n} S_p^{V_n} + \frac{\partial \hat{Q}_{r2}}{\partial V_m} S_p^{V_m}) - (\mathcal{S}_p^{\hat{Q}_{r2},o} + \frac{\partial \hat{Q}_{r2}}{\partial V_n} S_p^{V_n,o} + \frac{\partial \hat{Q}_{r2}}{\partial V_m} S_p^{V_m,o}) \right] \\
&= \mathcal{S}_p^{\hat{I}_1} + \mathcal{S}_{\partial V_i}^{\hat{I}_1} S_p^{V_i} + \mathcal{S}_{\partial V_m}^{\hat{I}_1} S_p^{V_m} + \\
&\quad + \frac{1}{h}\left[\frac{\partial \hat{Q}_{d1}}{\partial V_i}(S_p^{V_i} - S_p^{V_i,o}) + \frac{\partial \hat{Q}_{d1}}{\partial V_m}(S_p^{V_m} - S_p^{V_m,o}) + (\mathcal{S}_p^{\hat{Q}_{d1}} - \mathcal{S}_p^{\hat{Q}_{d1},o}) \right] \\
&\quad + \frac{1}{h}\left[\frac{\partial \hat{Q}_{r2}}{\partial V_n}(S_p^{V_n} - S_p^{V_n,o}) + \frac{\partial \hat{Q}_{r2}}{\partial V_m}(S_p^{V_m} - S_p^{V_m,o}) + (\mathcal{S}_p^{\hat{Q}_{r2}} - (\mathcal{S}_p^{\hat{Q}_{r2},o}) \right]
\end{aligned}
\tag{C.6}
$$

The equivalent differentiation and reordering is done for the remaining equations.

$$
\begin{aligned}
0 &= \mathcal{S}_p^{\hat{I}_o} + \mathcal{S}_{\partial V_n}^{\hat{I}_o} S_p^{V_n} + \mathcal{S}_{\partial V_m}^{\hat{I}_o} S_p^{V_m} + \\
&\quad + \frac{1}{h}\left[(\mathcal{S}_p^{\hat{Q}_{d2}} + \frac{\partial \hat{Q}_{d2}}{\partial V_n} S_p^{V_n} + \frac{\partial \hat{Q}_{d2}}{\partial V_m} S_p^{V_m}) - (\mathcal{S}_p^{\hat{Q}_{d2},o} + \frac{\partial \hat{Q}_{d2}}{\partial V_n} S_p^{V_n,o} + \frac{\partial \hat{Q}_{d2}}{\partial V_m} S_p^{V_m,o}) \right] + \\
&\quad + \frac{1}{h} C_n (S_p^{V_n} - S_p^{V_n,o}) + G \cdot (S_p^{V_n} - S_p^{V_f}) \\
0 &= \mathcal{S}_p^{\hat{I}_o} + \mathcal{S}_{\partial V_n}^{\hat{I}_o} S_p^{V_n} + \mathcal{S}_{\partial V_m}^{\hat{I}_o} S_p^{V_m} + G \cdot (S_p^{V_n} - S_p^{V_f}) \\
&\quad + \frac{1}{h}\left[\frac{\partial \hat{Q}_{d2}}{\partial V_n}(S_p^{V_n} + S_p^{V_n,o}) + \frac{\partial \hat{Q}_{d2}}{\partial V_m}(S_p^{V_m} - S_p^{V_m,o}) + (\mathcal{S}_p^{\hat{Q}_{d2}} - \mathcal{S}_p^{\hat{Q}_{d2},o}) \right] + \\
&\quad + \frac{1}{h} C_n (S_p^{V_n} - S_p^{V_n,o})
\end{aligned}
\tag{C.7}
$$

$$
\begin{aligned}
0 &= G \cdot (S_p^{V_f} - S_p^{V_n}) + \frac{d}{dt} C_f (S_p^{V_f} - S_p^{V_f,o}) + \frac{d}{dt}[(\mathcal{S}_p^{\hat{Q}_L} + \frac{\partial \hat{Q}_L}{\partial V_f} S_p^{V_f}) - (\mathcal{S}_p^{\hat{Q}_L,o} + \frac{\partial \hat{Q}_L}{\partial V_f} S_p^{V_f,o})] \\
0 &= G \cdot (S_p^{V_f} - S_p^{V_n}) + \frac{d}{dt}[(C_f + \frac{\partial \hat{Q}_L}{\partial V_f}) \cdot (S_p^{V_f} - S_p^{V_f,o}) + (\mathcal{S}_p^{\hat{Q}_L} - \mathcal{S}_p^{\hat{Q}_L,o})]
\end{aligned}
\tag{C.8}
$$

Equations C.5–C.8 are resorted to group the sensitivity vector at the left hand side.

$$\boldsymbol{J}(\boldsymbol{V})v_p^{i+1} = \begin{bmatrix} S_p^{V_{prev}} \\ -\mathcal{S}_p^{\hat{I}_1} \\ -\mathcal{S}_p^{\hat{I}_o} \\ 0 \end{bmatrix} + \frac{1}{h}\cdot$$

$$\begin{bmatrix} 0 \\ \frac{\partial \hat{Q}_{d1}}{\partial V_i}\cdot S_p^{V_i,o} + \frac{\partial \hat{Q}_{d1}}{\partial V_m}\cdot S_p^{V_m,o} - (\mathcal{S}_p^{\hat{Q}_{d1}} - \mathcal{S}_p^{\hat{Q}_{d1},o}) + \frac{\partial \hat{Q}_{r2}}{\partial V_n}\cdot S_p^{V_n,o} + \frac{\partial \hat{Q}_{r2}}{\partial V_m}\cdot S_p^{V_m,o} - (\mathcal{S}_p^{\hat{Q}_{r2}} - \mathcal{S}_p^{\hat{Q}_{r2},o}) \\ +(C_n + \frac{\partial \hat{Q}_{d2}}{\partial V_n})\cdot S_p^{V_n,o} + \frac{\partial \hat{Q}_{d2}}{\partial V_m}\cdot S_p^{V_m,o} - (\mathcal{S}_p^{\hat{Q}_{d2}} - \mathcal{S}_p^{\hat{Q}_{d2},o}) \\ +(C_f + \frac{\partial \hat{Q}_L}{\partial V_f})\cdot S_p^{V_f,o} - (\mathcal{S}_p^{\hat{Q}_L} - \mathcal{S}_p^{\hat{Q}_L,o}) \end{bmatrix} \tag{C.9}$$

One original thought is worth a thousand mindless quotings.

Banksy

Bibliography

[Abe03] U. Abelein. "Numerische Laufzeitmodellierung digitaler Bibliothekszellen". MA thesis. Technische Universität München, 2003.

[ADI03] C. S. Amin, F. Dartu, and Y. I. Ismail. "Weibull Based Analytical Waveform Model". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2003, pp. 161–168.

[ADP02] E. Acar, F. Dartu, and L. T. Pileggi. "TETA: transistor-level waveform evaluation for timing analysis". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 21.5 (2002), pp. 605–616.

[All07] G. S. Alliance. "Innovation and Productivity Survey Results". In: (2007).

[Ami+06] C. Amin et al. "A multi-port current source model for multiple-input switching effects in CMOS library cells". In: *ACM/IEEE Design Automation Conference (DAC)*. 2006, pp. 247–252.

[Aot] *Advanced On-chip-variation Timing Analysis*. Tech. rep. Incentia Design Systems, Inc., 2007.

[AP03] S. Abbaspour and M. Pedram. "Calculating the effective capacitance for the RC interconnect in VDSM technologies". In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2003, pp. 43–48.

[ARA08] M. H. Abu-Rahma and M. Anis. "A Statistical Design-Oriented Delay Variation Model Accounting for Within-Die Variations". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.11 (2008), pp. 1983–1995.

[ARP00] R. Arunachalam, K. Rajagopal, and L. T. Pileggi. "TACO: timing analysis with coupling". In: *ACM/IEEE Design Automation Conference (DAC)*. ACM, 2000, pp. 266–269. URL: `http://doi.acm.org/10.1145/337292.337415`.

[ASU86] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers Principles, Techniques and Tools*. Addison-Wesley Publishing Company, 1986.

[Bar+11] C. Barker et al. "Factors Affecting Thickness Variation of SiO_2 Thin Films Grown by Wet Oxidation". In: *IEEE Transactions on Semiconductor Manufacturing* 24.2 (2011), pp. 348–357.

[BC09] J. Bhasker and R. Chadha. *Static Timing Analysis for Nanometer Designs*. Springer, 2009.

[BCSS08] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer. "Statistical Timing Analysis: From Basic Principles to State of the Art". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 27th ser. 4 (2008), pp. 589–607.

[Bha05] J. Bhasker. *The Exchange Format Handbook: A DEF, LEF, SDF, SPEF, VCD Primer*. Star Galaxy Publishing, 2005.

[BHSA03] C. Bittlestone, A. M. Hill, V. Singhal, and N. V. Arvind. "Architecting ASIC libraries and flows in nanometer era". In: *ACM/IEEE Design Automation Conference (DAC)*. 2003, pp. 776–781.

[Bin08] F. Binder. "Approximation of the static behavior of cells in analog circuit simulation for the use in current source models". Diplomarbeit. Universität Hamburg, 2008.

[BJV06] M. J. Bellido, J. Juan, and M. Valencia. *Logic-Timing Simulation and the Degradation Model*. Imperial College Press, 2006.

[Bla05] P. E. Black. "Bellman-Ford algorithm". In: *Dictionary of Algorithms and Data Structures [online]*. U.S. National Institute of Standards and Technology, 2005. URL: `http://www.nist.gov/dads/HTML/bellmanford.html`.

[BM11] M. Bashir and L. Milor. "Determining the Impact of Within-Die Variation on Circuit Timing". In: *Semiconductor Manufacturing, IEEE Transactions on* 24.3 (2011), pp. 385–391.

[Bmm] *BSIM 4.6.2 MOSFET Model: Users Manual*. Department of Electrical Engineering, University of California at Berkeley. `http://www-device.EECS.Berkeley.EDU/~bsim3/`, 2008.

[Bry81] R. E. Bryant. "MOSSIM: A switch-level simulator for MOS LSI". In: *ACM/IEEE Design Automation Conference (DAC)*. 1981, pp. 786–790.

[BS80] R. K. Brayton and R. Spence. *Sensitivity and Optimization*. New York, NY, USA: Elsevier Science Inc., 1980.

[BSE08] D. Bountas, G. Stamoulis, and N. Evmorfopoulos. "A macromodel technique for VLSI dynamic simulation by mapping pre-characterized transitions". In: *IEEE International Conference on Computer Design (ICCD)*. 2008, pp. 450–456.

[BSMM01] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. 5th ed. Verlag Harri Deutsch, 2001.

[BVB03] S. Bhardwaj, S. Vrudhula, and D. Blaauw. "τAU: Timing analysis under uncertainty". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2003, pp. 615–620.

[Cad03] Cadence. *UltraSim User's Manual*. 2003.

[Cad04a] Cadence. *Compiled-Model Interface Reference*. 2004.

[Cad04b] Cadence. *Using Hierarchy and Isomorphism to accelerate circuit simulation*. White paper. 2004.

[Cad07] Cadence. *ECSM - Effective Current Source Model*. `http://www.cadence.com/Alliances/languages/Pages/ecsm.aspx`. 2007.

[CGK75] B. Chawla, H. Gummel, and P. Kozak. "MOTIS-An MOS timing simulator". In: *IEEE Transactions on Circuits and Systems* 22.12 (1975), pp. 901–910.

[CKSB06] K. Chopra, C. Kashyap, H. Su, and D. Blaauw. "Current Source Driver Model Synthesis and Worst-case Alignment for Accurate Timing and Noise Analysis". In: *ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*. 2006, pp. 45–50.

[Con+10] J. Cong et al. "Accelerating Monte Carlo based SSTA using FPGA". In: *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. 2010, pp. 111–114. URL: `http://doi.acm.org/10.1145/1723112.1723132`.

[CPO04] M. Celik, L. Pileggi, and A. Odabasioglu. *IC Interconnect Analysis*. Kluwer Academic Publishers, 2004.

[CPZ97] P. Cocchini, G. Piccinini, and M. Zamboni. "A comprehensive submicrometer MOST delay model and its application to CMOS buffers". In: *Solid-State Circuits, IEEE Journal of* 32.8 (1997), pp. 1254–1262.

[CW03] J. Croix and M. Wong. "Blade and razor: cell and interconnect delay analysis using current-based models". In: *ACM/IEEE Design Automation Conference (DAC)*. 2003, pp. 386–389.

[Das+11] D. Das et al. "FA-STAC: An Algorithmic Framework for Fast and Accurate Coupling Aware Static Timing Analysis". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19.3 (2011), pp. 443–456.

[DDR07] S. Dabas, N. Dong, and J. Roychowdhury. "Automated Extraction of Accurate Delay/Timing Macromodels of Digital Gates and Latches using Trajectory Piecewise Methods". In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2007, pp. 361–366.

[DGQ99] L. Daldoss, P. Gubian, and M. Quarantelli. "Transient sensitivity computation in circuit simulation". In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. 1999, pp. 302–305.

[DH06] A. Dasdan and I. Hom. "Handling inverted temperature dependence in static timing analysis". In: *ACM Transactions on Design Automation of Electronic Systems* 11.2 (2006), pp. 306–324.

[DK69] C. Desoer and E. Kuh. *Basic circuit theory*. McGraw Hill international editions: Electrical and electronic engineering series. McGraw-Hill, 1969.

[DMP96] F. Dartu, N. Menezes, and L. T. Pileggi. “Performance computation for precharacterized CMOS gates with RC loads”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15.5 (1996), pp. 544–553.

[DP96] F. Dartu and L. T. Pileggi. “Modeling signal waveshapes for empirical cmos gate delay models”. In: *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. 1996.

[DR69] S. W. Director and R. A. Rohrer. “The generalized adjoint network and network sensitivities”. In: *IEEE Transactions on Circuit Theory* 16 (1969), pp. 318–323.

[DR93] A. Devgan and R. A. Rohrer. “ACES: A Transient Simulation Strategy for Integrated Circuits”. In: *IEEE International Conference on Computer Design (ICCD)*. 1993, pp. 357–360.

[DR94] A. Devgan and R. A. Rohrer. “Adaptively controlled explicit simulation”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 13.6 (1994), pp. 746–762.

[DSNZ09] D. Das, W. Scott, S. Nazarian, and H. Zhou. “An efficient current-based logic cell model for crosstalk delay analysis”. In: *IEEE International Symposium on Quality Electronic Design (ISQED)*. 2009, pp. 627–633.

[Eic09] I. Eichwald. “Evaluation of Structure-based Current Source Models for Path Based Timing Analysis”. Bachelorarbeit. Technische Universität München, 2009.

[Elm48] W. C. Elmore. “The transient response of damped linear networks with particular regard to wide-band amplifiers”. In: *Journal of Applied Physics* 19.1 (1948), pp. 55–63.

[FA08] P. Feldmann and S. Abbaspour. “Towards a more physical approach to gate modeling for timing, noise, and power”. In: *ACM/IEEE Design Automation Conference (DAC)*. 2008, pp. 453–455.

[Fel+08] P. Feldmann et al. “Driver waveform computation for timing analysis with multiple voltage threshold driver models”. In: *ACM/IEEE Design Automation Conference (DAC)*. 2008, pp. 425–428.

[FNP06] H. Fatemi, S. Nazarian, and M. Pedram. “Statistical Logic Cell Delay Analysis Using a Current-based Model”. In: *ACM/IEEE Design Automation Conference (DAC)*. 2006, pp. 253–256.

[FNP07] H. Fatemi, S. Nazarian, and M. Pedram. “A current-based method for short circuit power calculation under noisy input waveforms”. In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2007, pp. 774–779.

[GCKS09] K. Gulati, J. F. Croix, S. P. Khatri, and R. Shastry. "Fast circuit simulation on graphics processing units". In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. ASP-DAC '09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 403–408.

[GDTB09] R. Gandikota, L. Ding, P. Tehrani, and D. Blaauw. "Worst-case aggressor-victim alignment with current-source driver models". In: *ACM/IEEE Design Automation Conference (DAC)*. New York, NY, USA: ACM, 2009, pp. 13–18.

[GK05] R. Goyal and N. Kumar. "Current Based Delay Models: A Must For Nanometer Timing". In: *CDNLive*. 2005.

[GK09] K. Gulati and S. P. Khatri. "Accelerating statistical static timing analysis using graphics processing units". In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2009, pp. 260–265.

[GS10] S. Gupta and S. S. Sapatnekar. "Current source modeling in the presence of body bias". In: *IEEE/ACM Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2010, pp. 199–204.

[GV08a] A. Goel and S. Vrudhula. "Current source based standard cell model for accurate signal integrity and timing analysis". In: *Design, Automation and Test in Europe (DATE)*. 2008, pp. 574–579.

[GV08b] A. Goel and S. Vrudhula. "Statistical waveform and current source based standard cell models for accurate timing analysis". In: *ACM/IEEE Design Automation Conference (DAC)*. 2008, pp. 227–230.

[Han11] W. Hansch. "Halbleiterproduktionstechnik". In: *Technik in Bayern* (2011).

[HGB02] I.-D. Huang, S. Gupta, and M. Breuer. "Accurate and efficient static timing analysis with crosstalk". In: *IEEE International Conference on Computer Design: VLSI in Computers and Processors*. 2002, pp. 265–272.

[Hit82] R. B. Hitchcock. "Timing Verification and the Timing Analysis Program". In: *ACM/IEEE Design Automation Conference (DAC)*. 1982, pp. 594–604.

[HKIM05] Z. Huang, A. Kurokawa, Y. Inoue, and J. Mao. "A Novel Model for Computing the Effective Capacitance of CMOS Gates with Interconnect Loads". In: *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 10 (2005), pp. 2562–2569.

[HOT98] A. Hirata, H. Onodera, and K. Tamaru. "Proposal of a timing model for CMOS logic gates driving a CRC *pi* load". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1998, pp. 537–544.

[HRR78] H. Hsieh, N. Rabbat, and A. Ruehli. "Macromodelling and macrosimulation techniques". In: *IEEE Int Sym. on Circuit and Systems*. 1978, pp. 336–339.

[Hua+09] V. Huard et al. “CMOS device design-in reliability approach in advanced nodes”. In: *IEEE International Reliability Physics Symposium (IRPS)*. 2009, pp. 624–633.

[Hua+10] Z. Huang et al. “Modeling the Overshooting Effect for CMOS Inverter in Nanometer Technologies”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29.2 (2010), pp. 250–260.

[Hub11] M. Huber. “Automatische Generierung von Interconnect Modellen für die Static Timing Analysis”. Studienarbeit. Technische Universität München, 2011.

[HYO04] M. Hashimoto, Y. Yamada, and H. Onodera. “Equivalent waveform propagation for static timing analysis”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 23.4 (2004).

[HYTE85] D. A. Hocevar, P. Yang, T. N. Trick, and B. D. Epler. “Transient sensitivity computation for MOSFET circuits”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 4 (1985), pp. 609–620.

[Ini07] S. I. Initiative. *Si2 Effective Current Source Model (ECSM) Statistical Extensions Specification*. 2007.

[Isf] *IEEE Standard for an Advanced Library Format (ALF) Describing Integrated Circuit (IC) Technology, Cells, and Blocks*. IEEE Std 1603-2003. 2004.

[JBZ05] A. Jain, D. Blaauw, and V. Zolotov. “Accurate delay computation for noisy waveform shapes”. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2005, pp. 947–953.

[Jed11] H. Jedda. “Power Current Source Model”. Bachelorarbeit. Technische Universität München, 2011.

[Jen71] F. Jenkins. “TIME-a nonlinear d.c. and time-domain circuit-simulation program”. In: *IEEE Journal of Solid-State Circuits* 6.4 (1971), pp. 182–188.

[Joh09] H. Johnson. “Real signals”. In: *EDN* (2009). URL: `http://www.edn.com/article/458264-Real_signals.php`.

[Kal11] K. Kalafala (IBM). *Static timing analysis sign-off challenges and opportunities at 22nm and below*. Keynote at PATMOS. 2011.

[KAMC07] C. Kashyap, C. Amin, N. Menezes, and E. Chiprout. “A nonlinear cell macromodel for digital applications”. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2007, pp. 678–685.

[Kan09] K. Kanbas. “Tool conception for the classification of standard cell libraries simulation results”. Bachelorarbeit. Technische Universität München, 2009.

[KENS10] C. Knoth, I. Eichwald, P. Nordholz, and U. Schlichtmann. "White-Box Current Source Modeling Including Parameter Variation and Its Application in Timing Simulation". In: *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. 2010, pp. 200–210.

[Kez06] R. C. Kezer. *Characterization Guidelines for ECSM Timing Libraries*. Silicon Integration Initiative, Inc. 2006.

[KHN89] Y. H. Kim, S. H. Hwang, and A. Newton. "Electrical-logic simulation and its applications". In: *IEEE Transactions o Computer-Aided Design of Integrated Circuits and Systems* 8.1 (1989), pp. 8–22.

[Kil76] J. S. Kilby. "Invention of the integrated circuit". In: *IEEE Transactions on Electron Devices (TED)* 23.7 (1976), pp. 648–654.

[KJA07] J. Kim, K. D. Jones, and M. A.Horowitz. "Fast, non-Monte-Carlo estimation of transient performance variation due to device mismatch". In: *ACM/IEEE Design Automation Conference (DAC)*. DAC '07. 2007, pp. 440–443.

[KJS12] C. Knoth, H. Jedda, and U. Schlichtmann. "Current Source Modeling for Power and Timing Analysis at Different Supply Voltages". In: *Design, Automation and Test in Europe (DATE)*. 2012, pp. 923–928.

[KK66] F. Kuo and J. Kaiser. *System analysis by digital computer*. Wiley, 1966.

[KK99] M. Keramat and R. Kielbasa. "Modified Latin Hypercube Sampling Monte Carlo (MLHSMC) Estimation for Average Quality Index". In: *Analog Integrated Circuits and Signal Processing* 19 (1999), pp. 87–98.

[KKNS09a] C. Knoth, V. B. Kleeberger, P. Nordholz, and U. Schlichtmann. "Characterization and Implementation of Nonlinear Logic Cell Models for Analog Circuit Simulation". In: *International Symposium on Integrated Circuits (ISIC)*. 2009, pp. 97–100.

[KKNS09b] C. Knoth, V. B. Kleeberger, P. Nordholz, and U. Schlichtmann. "Fast and Waveform Independent Characterization of Current Source Models". In: *IEEE/VIUF International Workshop on Behavioral Modeling and Simulation (BMAS)*. 2009, pp. 90–95.

[Kle09] V. B. Kleeberger. "Generation of Nonlinear Dynamic Gate Models for Logic Cells Using DC Analysis and Structural Information". MA thesis. Technische Universität München, 2009.

[KLX06] A. B. Kahng, B. Liu, and X. Xu. "Constructing Current-Based Gate Models Based on Existing Timing Library". In: *IEEE International Symposium on Quality Electronic Design (ISQED)*. 2006, pp. 37–42.

[KN09] C. Knoth and P. Nordholz. *Extended Modeling Approaches for the Characterization of the Variation-Behavior of Standard-Cells*. Tech. rep. Technische Universität München, 2009.

[Kno+08] C. Knoth et al. “Transfer System Models of Logic Gates for Waveform-based Timing Analysis”. In: *International Workshop on Symbolic and Numerical Methods, Modeling and Applications to Circuit Design (SM2ACD)*. 2008, pp. 247–252.

[Kno+09] C. Knoth et al. “Waveform-based Timing Analysis for Digital Circuits using Current Source Models and Model Order Reduction”. In: *edaWorkshop*. VDE Verlag, 2009, pp. 19–24.

[KO95] J.-T. Kong and D. Overhauser. *Digital Timing Macromodeling for VLSI Design Verification*. Kluwer Academic Publishers, 1995.

[Koe69] D. Koehler. “Computer modeling of logic modules under consideration of delay and waveshaping”. In: *Proceedings of the IEEE* 57.7 (1969), pp. 1294–1296.

[KPM08] KPMG. *The Consumer Electronics Boom*. 2008. URL: `http://www.gsaglobal.org/publications/cestudy/`.

[KS12] C. Knoth and U. Schlichtmann. “Characterization of Standard Cells”. In: *Process Variations and Probabilistic Integrated Circuit Design*. Ed. by M. Dietrich and J. Haase. first. Springer, 2012. Chap. 4.1, pp. 93–106.

[KTK08] I. Keller, K. H. Tam, and V. Kariat. “Challenges in gate level modeling for delay and SI at 65nm and below”. In: *ACM/IEEE Design Automation Conference (DAC)*. 2008, pp. 468–473.

[KUKS11] C. Knoth, C. Uphoff, S. Kiesel, and U. Schlichtmann. “SWAT: Simulator for Waveform-Accurate Timing including Parameter Variations and Transistor Aging”. In: *Integrated Circuit and System Design, Power and Timing Modeling, Optimization and Simulation (PATMOS)*. Springer, 2011, pp. 193–203.

[KZ04] K. Kundert and O. Zinke. *The Designer’s Guide to Verilog-AMS*. Springer, 2004.

[LFA07] P. Li, Z. Feng, and E. Acar. “Characterizing Multistage Nonlinear Drivers and Variability for Accurate Timing and Noise Analysis”. In: *IEEE Transactions on VLSI Systems* 15.11 (2007), pp. 1205–1214.

[LGS10] D. Lorenz, G. Georgakos, and U. Schlichtmann. “Aging-aware Timing Analysis of Combinatorial Circuits on Gate Level”. In: *it - Information Technology* 4 (2010). Ed. by Hellebrand.

[Liu] *Liberty User Guide*. 2008.

[LK06] B. Liu and A. B. Kahng. “Statistical Gate Level Simulation via Voltage Controlled Current Source Models”. In: *IEEE International Behavioral Modeling and Simulation Workshop*. 2006.

[LVFA09] D. D. Ling, C. Visweswariah, P. Feldmann, and S. Abbaspour. “A moment-based effective characterization waveform for static timing analysis”. In: *ACM/IEEE Design Automation Conference (DAC)*. 2009, pp. 19–24.

[Mac11] C. Mack. “Fifty Years of Moore’s Law”. In: *Semiconductor Manufacturing, IEEE Transactions on* 24.2 (2011), pp. 202–207.

[MAO95] Y. Miki, M. Abe, and Y. Ogawa. “PCHECK: a delay analysis tool for high performance LSI design”. In: *IEEE Custom Integrated Circuits Conference (CICC)*. 1995, pp. 267–270.

[Mas09] C. Massmann. “Accuracy Enhancements for Current Source Models Considering Stacked Transistors”. Bachelorarbeit. Technische Universität München, 2009.

[Mer90] K. Merten. “Circuit simulation in the semiconductor industry - state of the art, requirements and future development”. In: *International Series of Numerical Mathematics* 93 (1990), pp. 1–10.

[Mic87] A. Miczo. *Digital Logic Testing and Simulation*. Wiley, 1987.

[Mit+07] A. Mitev et al. “A robust finite-point based gate model considering process variations”. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2007, pp. 692–697.

[MKA08] N. Menezes, C. Kashyap, and C. Amin. “A true electrical Cell Model for Timing, Noise, and Power Grid Verification”. In: *ACM/IEEE Design Automation Conference (DAC)*. 2008, pp. 462–467.

[MME86] S. Murai, H. Matsushita, and K. Enomoto. “Logic Simulation Programs”. In: *Advances in CAD for VLSI*. Ed. by E. Hoerbst. Vol. 2. Elseviers Science Publishers, 1986. Chap. 5, pp. 135–163.

[Mor91] M. D. Morris. “Factorial sampling plans for preliminary computational experiments”. In: *Technometrics* 33 (1991), pp. 161–174.

[MP71] W. McCalla and D. Pederson. “Elements of Computer-Aided Circuit Analysis”. In: *Circuit Theory, IEEE Transactions on* 18.1 (1971), pp. 14–26.

[MS02] L. McMurchie and C. Sechen. “WTA - Waveform-Based Timing Analysis for Deep Submicron Circuits”. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2002.

[NA04] S. R. Nassif and E. Acar. “Advanced waveform models for the nanometer regime”. In: *ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*. 2004.

[Nad+03] D. Nadezhin et al. “SOI Transistor Model for Fast Transient Simulation”. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2003, pp. 120–127.

[NB07] S. R. Nassif and R. V. H. Booth. “A Simple MOSFET Model Suitable for Fast Timing Analysis”. In: *IEEE International Conference on Integrated Circuit Design and Technology (ICICDT)*. 2007, pp. 1–6.

[NFP11] S. Nazarian, H. Fatemi, and M. Pedram. "Accurate Timing and Noise Analysis of Combinational and Sequential Logic Cells Using Current Source Modeling". In: *IEEE Transactions on VLSI Systems* 19.1 (2011), pp. 92–103.

[NK09] P. Nordholz and C. Knoth. *Erweiterte Modellierungsansätze zur Charakterisierung des Schwankungsverhaltens von Standardzellen im Semi-Custom-Bereich*. Tech. rep. Technische Universität München, 2009.

[NL05] S. R. Nassif and Z. Li. "A More Effective C_{EFF}". In: *IEEE International Symposium on Quality Electronic Design (ISQED)*. 2005, pp. 648–653.

[NOW99] T. Nguyen, P. O´Brien, and D. Winston. "Transient Sensitivity Computation for Transistor Level Analysis and Tuning". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1999.

[NP73] L. W. Nagel and D. Pederson. *SPICE (Simulation Program with Integrated Circuit Emphasis)*. Tech. rep. UCB/ERL M382. EECS Department, University of California, Berkeley, 1973. URL: `http://www.eecs.berkeley.edu/Pubs/TechRpts/1973/22871.html`.

[Ous84] J. K. Ousterhout. "Switch-level delay models for digital MOS VLSI". In: *ACM/IEEE Design Automation Conference (DAC)*. DAC. 1984, pp. 542–548.

[Ous85] J. K. Ousterhout. "A Switch-Level Timing Verifier for Digital MOS VLSI". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 4.3 (1985), pp. 336–349.

[OZ00] A. Owen and Y. Zhou. "Safe and effective importance sampling". In: *Journal of the American Statistical Association* 95.449 (2000), pp. 135–143.

[Pla08] D. Platte. "Simulation Efficiency of Analog Behavioral Models". PhD thesis. Gottfried Wilhelm Leibniz Universität Hannover, 2008.

[PRV95] L. T. Pillage, R. A. Rohrer, and C. Visweswariah. *Electronic Circuit and System Simulation Methods*. McGraw-Hill, Inc., 1995.

[PSD70] P. Penfield, R. Spence, and S. Duinker. *Tellegen's Theorem and Electrical Networks*. The M.I.T. Press, 1970.

[Qim08] Qimonda. *Titan 7.4 - User Manual*. 2008.

[QPP94] J. Qian, S. Pullela, and L. T. Pillage. "Modeling the "Effective capacitance" for the RC interconnect of CMOS gates". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 13.12 (1994).

[RLB00] M. Ringe, T. Lindenkreuz, and E. Barke. "Static Timing Analysis Taking Crosstalk into Account". In: *Design, Automation and Test in Europe (DATE)*. 2000, pp. 451–451.

[RS04] J. Rossello and J. Segura. "An analytical charge-based compact delay model for submicrometer CMOS inverters". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 51.7 (2004), pp. 1301–1311.

[RVBG08] S Raja, F Varadi, M Becer, and J Geada. "Transistor Level Gate Modeling for Accurate and Fast Timing, Noise, and Power Analysis". In: *ACM/IEEE Design Automation Conference (DAC)*. 2008, pp. 456–461.

[Sap04] S. S. Sapatnekar. *Timing*. Kluwer Academic Publishers, 2004.

[Sch08] M. Schmidt. "Waveform Based Statistical Timing Analysis of Integrated Digital Circuits". PhD thesis. Technische Universität München, 2008.

[SD85] K. A. Sakallah and S. W. Director. "SAMSON2: An Event Driven VLSI Circuit Simulator". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 4.4 (1985), pp. 668–684.

[SH68] H. Shichman and D. Hodges. "Modeling and Simulation of Insulated-Gate Field-Effect Transistor Switching Circuits". In: *IEEE Journal of Solid-State Circuits SC* 3 (1968), pp. 285–289.

[SKS08] M. Schmidt, H. Kinzelbach, and U. Schlichtmann. *More Accurate Statistical Timing Analysis by Considering Waveform Variations*. Tech. rep. Technische Universität München, 2008.

[SLM05] L. Scheffer, L. Lavagno, and G. Martin, eds. *Electronic design automation for integrated circuit system design, verification, and testing*. Electronic Design Automation for Integrated Circuits Handbook. CRC/Taylor & Francis, 2005.

[SLM06] L. Scheffer, L. Lavagno, and G. Martin, eds. *EDA for IC implementation, circuit design, and process technology*. CRC Press, 2006.

[SMSB08] J. sun Seo, I. L. Markov, D. Sylvester, and D. Blaauw. "On the Decreasing Significance of Large Standard Cells in Technology Mapping". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2008, pp. 116–121.

[SN90] T. Sakurai and A. R. Newton. "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas". In: *IEEE Journal of Solid-State Circuits SC* 25.2 (1990), pp. 584–594.

[Sob67] I. M. Sobol. "On the distribution of points in a cube and the approximate evaluation of integrals". In: *Computational Mathematics and mathematical physics* 7.4 (1967), pp. 86+.

[SR10] A. Singhee and R. Rutenbar. "Why Quasi-Monte Carlo is Better Than Monte Carlo or Latin Hypercube Sampling for Statistical Circuit Analysis". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 29.11 (2010), pp. 1763–1776.

[SS09] L. Stolberg-Stolberg. "Automatische Gatterextraktion für digitale CMOS Schaltungen". Studienarbeit. Technische Universität München, 2009.

[SSH99] I. Sutherland, B. Sproull, and D. Harris. *Logical effort: designing fast CMOS circuits*. Morgan Kaufmann Publishers Inc., 1999.

[SSP07] L. G. e Silva, L. M. Silveira, and J. R. Phillips. "Efficient computation of the worst-delay corner". In: *Design, Automation and Test in Europe (DATE)*. 2007, pp. 1617–1622.

[SSR08] A. Singhee, S. Singhal, and R. A. Rutenbar. "Practical, Fast Monte Carlo Statistical Static Timing Analysis: why and how". In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2008, pp. 190–195.

[Ste95] G. W. Stewart. *Afternotes on Numerical Analysis*. SIAM, 1995.

[SW01] R. Schaback and H. Wendl. "Characterization and construction of radial basis functions". In: *Multivariate Approximation and Applications*. Cambridge University Press, 2001, pp. 1–24.

[Syn06] Synopsys. *Composite Current Source*. `http://www.synopsys.com/products/solutions/galaxy/ccs/cc_source.html`. 2006.

[Syn12] Synopsys. *Static Timing Analysis Solution for Custom Designs with Embedded Memories*. `http://www.synopsys.com/Tools/Implementation/SignOff/Pages/NanoTime.aspx`. 2012.

[TP07] S. K. Tiwary and J. R. Phillips. "WAVSTAN: Waveform Based Variational Static Timing Analysis". In: *Design, Automation and Test in Europe (DATE)*. 2007.

[TZBM10] Q. Tang, A. Zjajo, M. Berkelaar, and N. van der Meijs. "RDE-Based Transistor-Level Gate Simulation for Statistical Static Timing Analysis". In: *ACM/IEEE Design Automation Conference (DAC)*. 2010, pp. 787–792.

[VCBS11] V. Veetil, K. Chopra, D. Blaauw, and D. Sylvester. "Fast Statistical Static Timing Analysis Using Smart Monte Carlo Techniques". In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 30.6 (2011), pp. 852–865.

[VFHL10] G. Venkataraman, Z. Feng, J. Hu, and P. Li. "Combinatorial algorithms for fast clock mesh optimization". In: *IEEE Transactions on VLSI Systems* 18.1 (2010), pp. 131–141.

[Vis+04] C. Visweswariah et al. "First-Order Incremental Block-Based Statistical Timing Analysis". In: *ACM/IEEE Design Automation Conference (DAC)*. 2004, pp. 331–336.

[Vlr] *Verilog-AMS Language Reference Manual version 2.3*. http://www.eda.org/verilog-ams/. 2008.

[VS94] J. Vlach and K. Singhal. *Computer Methods for Circuit Analysis and Design*. 2nd ed. Van Nostrand Reinhold, 1994.

[VSB08] V. Veetil, D. Sylvester, and D. Blaauw. "Fast and Accurate Waveform Analysis with Current Source Models". In: *IEEE International Symposium on Quality Electronic Design (ISQED)*. 2008, pp. 53–56.

[VW93] C. Visweswariah and J. Wehbeh. “Incremental Event-Driven Simulation of Digital FET Circuits”. In: *ACM/IEEE Design Automation Conference (DAC)*. 1993, pp. 737–741.

[Wak05] J. F. Wakerly. *Digital Design Principles and Practices*. 4th ed. Prentice Hall, 2005.

[Wan+10] W. Wang et al. “The Impact of NBTI Effect on Combinational Circuit: Modeling, Simulation, and Analysis”. In: *IEEE Transactions on VLSI Systems* 18.2 (2010), pp. 173–183.

[Web02] M. Weber. “My Head Hurts, My Timing Stinks, and I Don’t Love On-chip Variation”. In: *SNUG*. 2002.

[You10] A. Youssef. “Automatische Strukturanalyse extrahierter Standardzellen”. MA thesis. Technische Universität München, 2010.

[Zol+07] V. Zolotov et al. “Compact modeling of variational waveforms”. In: *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2007, pp. 705–712.

List of Figures

List of Tables

Acronyms

.lef	Library Exchange Format
.lib	Liberty format
AC	alternating current
ACES	adaptively controlled explicit simulation
API	application programming interface
CCB	channel connected block
CCS	composite current source
CDF	cumulative distribution function
CM	compiled model
CMI	compiled model interface
CSM	current source model
DAE	differential algebraic equation
DAG	directed acyclic graph
DC	delay calculation
DC	direct current
DCAT	digital circuit analysis tool
DTA	dynamic timing analysis
ECSM	effective current source model
EDA	electronic design automation
FD	finite difference
HCI	hot carrier injection
HDL	hardware description language
IC	integrated circuit
IR	voltage drop on power nets due to supply currents and wire resistance
KCL	Kirchhoff's current law

LAE	linear algebraic equation
LSF	least square fit
LUT	lookup table
MC	Monte Carlo
MNA	modified nodal analysis
NBTI	negative bias temperature instability
NLAE	nonlinear algebraic equation
NLDAE	nonlinear differential algebraic equation
NLDM	nonlinear delay model
NR	Newton-Raphson-Algorithm
OCV	on-chip variation
PDF	probability density function
PDN	pull-down network
PnR	Place and Route
PUN	pull-up network
PVT	process parameters, voltage, and temperature
PXM	Pull-up/Pull-down Model
RHS	right hand side
RTL	register transfer level
RV	random variable
SA	sensitivity analysis
SC	standard cell
SCSM	Systematic Current Source Model
SPDM	scalable polynomial delay model
SPEF	Standard Parasitic Exchange Format
SSTA	statistical static timing analysis
STA	static timing analysis
SWAT	Simulator for Waveform-Accurate Timing
UMI	user model interface
VCC	voltage-controlled capacitor
VCCS	voltage-controlled current source
VCQ	voltage-controlled charge
VRM	voltage response model
ZMS	Z-element model specification

List of symbols

$\dot{x}$	time derivative of any quantity x
$\hat{x}$	model equivalent of any quantity x
$\langle \mathrm{a0}{\rightarrow}\mathrm{o} \rangle^r$	timing arc from input pin a to rising output pin o and second input pin is low
$\boldsymbol{C}$	dynamic part of Jacobian matrix
$\mathbb{C}$	voltage-controlled capacitor (network element)
D	delay
$\boldsymbol{E}$	excitation vector
$\boldsymbol{F}(\boldsymbol{V}, t)$	circuit equations
$\boldsymbol{G}$	DC part of Jacobian matrix
h	time step during transient analysis
I_d	static driver current (output of inverting block)
$\mathbb{I}$	voltage-controlled current source (network element)
$\hat{i}$	model current
$\boldsymbol{J}(\boldsymbol{V})$	Jacobian matrix
M	transistor (network element)
PG	power/ground
Q	charge at a node, capacitor, or transistor pin
$\hat{Q}$	port charge of current source model
$\mathbb{Q}$	voltage-controlled charge (network element)
S_y^x	linear sensitivity of quantity x with respect to y

$\mathcal{S}_y^x$	interpolated linear sensitivity of component x with respect to y obtained from lookup table
τ	transition time or slew rate of a signal
$\mathcal{F}$	approximation function for lookup table
t	time
T	time of a signal change
v_{th}	threshold voltage of transistor
$\boldsymbol{V}$	vector of circuit variables
V	circuit variable, usually voltage
$\dot{\boldsymbol{V}}$	time derivative of circuit variables
$\hat{\boldsymbol{V}}$	(port) voltage vector of the CSM

Index