Contents lists available at ScienceDirect

Integration, the VLSI Journal

journal homepage: www.elsevier.com/locate/vlsi

The Involution Tool for Accurate Digital Timing and Power Analysis[☆]

Daniel Öhlinger [a,*], Jürgen Maier [a], Matthias Függer [b], Ulrich Schmid [a]

[a] ECS Group, TU Wien, Treitlstraße 3, 1040 Wien, Austria
[b] CNRS & LSV, ENS Paris-Saclay, Université Paris-Saclay & Inria, 94235 Cachan, France

ARTICLE INFO

Keywords:
Digital timing simulation
Design tools
Delay models
Pulse degradation
Glitch propagation

ABSTRACT

We introduce the prototype of a digital timing simulation and power analysis tool for integrated circuits that supports the involution delay model (Függer et al. 2019). Unlike the pure and inertial delay models typically used in digital timing analysis tools, the involution model faithfully captures short pulse propagation and related effects. Our Involution Tool facilitates experimental accuracy evaluation of variants of involution models, by comparing their timing and power predictions to those from SPICE and standard timing analysis tools. The tool is easily customizable w.r.t. instances of the involution model and circuits, and supports automatic test case generation and parameter sweeping.

We demonstrate the capabilities of the Involution Tool by providing timing and power analysis results for three different circuits, namely, an inverter tree, the clock tree of an open-source processor, and a combinational circuit that involves multi-input NAND gates. Our evaluation uses two different technologies (15 nm and 65 nm CMOS), and three different variants of involution channels (Exp, Hill and SumExp-channels). It turns out that the timing and power predictions of all involution models are significantly better than the predictions obtained by standard digital simulations for the inverter tree and the clock tree, with the SumExp-channel channel clearly outperforming the others. For the NAND circuit, the performance of any involution model is generally comparable but not significantly better than that of standard models, however, which reveals some shortcomings of the existing involution channels for modeling multi-input gates.

1. Introduction

Modern digital circuit design relies heavily on state-of-the-art timing analysis tools like Synopsys Prime Time, Mentor Questa, Cadence NC-Sim or Synopsis VCS. These tools can accurately predict the signal propagation through a given circuit design, and thus identify setup/hold violations and other timing-related problems in synchronous designs, for example. Moreover, they facilitate a reasonably accurate power analysis at early design stages [1,2].

The "golden standard" for circuit analyses are fully-fledged analog simulations, e.g., by using *SPICE* [3], which are based on detailed physical models of all elements in a digital standard-cell library. Since the execution times of analog simulations on even moderately complex circuits are prohibitively excessive, however, digital timing analysis tools use *discrete-valued* (typically binary) circuit models augmented by continuous-time delays. The latter is determined by elaborate timing prediction models like CCSM [4] and ECSM [5], which characterize the delay of a cell via (typically manufacturer-supplied) technology data and massive analog simulations. The gate and wire delay estimates obtained via CCSM or ECSM are then used for parametrizing pure or inertial delay channels [6] (e.g., in VHDL Vital or Verilog timing libraries). The resulting executable HDL simulation models are finally used in subsequent simulation and timing analysis runs.

While pure delay channels with constant delay forward pulses without changing their width, inertial delay channels provide simple means to model pulse suppression. More accurate results can be expected from *dynamic* timing analysis techniques, which consider more elaborate signal trace-related effects. One example is pulse degradation, meaning that short input pulses usually get shorter when processed by a gate. The arguably simplest way to capture such dynamic effects are *single-history* channel models, which allow gate delays (modeled via the interconnecting channels) to vary depending on the *previous* transition in a trace. More specifically, single-history channels are characterized by a delay function δ that maps a transition occurring at the channel

☆ This research was partially funded by the Austrian Science Fund (FWF) projects DMAC (P32431), SIC (P26436), and RiSE (S11405), projects FREDDA (ANR-17-CE40-0013) and DEPEC MODE (Departement STIC), and by DigiCosme (working group HicDiesMeus).

* Corresponding author.

E-mail addresses: daniel.oehlinger@tuwien.ac.at (D. Öhlinger), jmaier@ecs.tuwien.ac.at (J. Maier), mfuegger@lsv.fr (M. Függer), s@ecs.tuwien.ac.at (U. Schmid).

https://doi.org/10.1016/j.vlsi.2020.09.007
Received 24 November 2019; Received in revised form 10 May 2020; Accepted 10 September 2020
Available online 22 September 2020

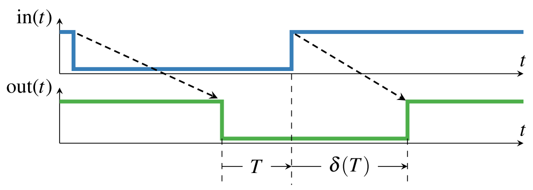


Fig. 1. Principle functionality of a single-history delay model. Based on the input-to-previous output transition time T, the delay $\delta(T)$ is determined.

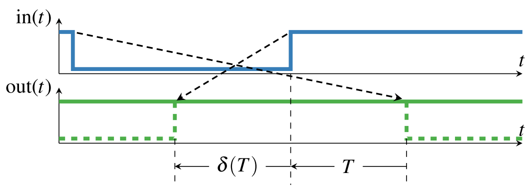


Fig. 2. The input pulse is so short that the transitions at the output appear in reverse order (dashed lines), i.e., cancel. Note that here $T < 0$ and $\delta(T) < 0$.

input at time t to its corresponding output transition at time $t + \delta(T)$, where T is the previous-output-to-input delay (cp. Fig. 1). If two succeeding input transitions would, according to $\delta(T)$, occur at the output in reversed order, they are said to cancel each other (shown in Fig. 2) and are removed. Note that single-history channels allow different rising and falling transition delays, specified by the delay functions $\delta_\uparrow$ and $\delta_\downarrow$, respectively.

The first proper single-history channel model was the *Degradation Delay Model (DDM)* introduced by Bellido-Díaz et al. [7,8]. However, it was proven by Függer et al. [9] that all existing delay models, including *DDM*, are not faithful: For the simple *short-pulse filtration* (SPF) problem, it turned out that the bounded resp. unbounded version is solvable in physical implementations but not within existing circuit models, or vice versa. In [10,11], the authors therefore introduced the *involution model* (*IM*), which is the only delay model known so far that does not share this problem. Its distinguishing property is that its delay functions form involutions, i.e., are self-inverse, in the sense that $-\delta_\downarrow(-\delta_\uparrow(T)) = T$ and $-\delta_\uparrow(-\delta_\downarrow(T)) = T$. Different variants of the *IM* differ in the particular instance of the delay function they use. In [12], the authors utilized both *SPICE* simulations and real measurements to demonstrate that even the most basic involution model (based on an Exp-channel, see Section 2.1) predicts the behavior of a real circuit, namely, an inverter chain, reasonably accurate.

Main contributions. In order to be able to apply the involution model to custom signal traces, we developed the *Involution Tool* (`invTool`) that is described in this paper.[1] It allows to include and run any *IM* in state-of-the-art digital circuit simulation tools (e.g. *Questa*), and is embedded in a comprehensive test infrastructure that allows to generate user-controlled random input vectors, to run different analog/digital simulations, and to generate various reports on the results. Thanks to its ability to process the output of other simulation tools, in particular, *HSPICE*- or *Spectre*-generated traces, it easily allows to compare timing and power predictions of the involution model vs. other models. As switching from standard to *IM* simulation is essentially achieved by loading a different library, existing infrastructure, such as test scripts and input vectors, can be reused without modification.

In more detail:

(i) We provide an overview of the features and some details of the implementation of the `invTool`, which facilitates digital timing simulation and power analysis of circuits composed of arbitrary Boolean functions connected via involution channels. We also describe the three variants of involution channels currently supported by our tool, namely, Exp-channels based on simple exponential switching waveforms, Hill-channels based on Hill functions that more closely match real switching waveforms, and SumExp-channels that were identified as a promising alternative thanks to the results of the earlier PATMOS'19 version [13] of this paper.

(ii) We demonstrate the utility of the `invTool` by conducting a timing and power analysis of three example circuits: an inverter tree, the clock tree of an open-source processor from [14], and a sample circuit that involves multi-input NAND gates, synthesized in two different technologies: 65 nm and 15 nm. It turns out that the timing and power predictions of all our involution models are significantly better than the predictions obtained by standard digital simulations for the inverter tree and the clock tree, albeit the Hill-channels surprisingly perform worse than the Exp-channels for short pulses. This observation inspired the definition of the SumExp-channel, which considerably outperforms the former two. For the NAND circuit, however, the predictions for any involution model turn out to be comparable but not significantly better than the ones of the standard models, sometimes even worse.

Overall, our experiments show that the *IM* is a viable approach for an accurate performance and power analysis of a circuit design. However, they also reveal some potential for improving the current model. In particular, we conclude that single-input single-output delay channels are not adequate for accurately modeling the behavior of multi-input gates.

Paper organization. In Section 2, we describe those features of the involution model that are instrumental for the `invTool`. Section 3 is devoted to an overview of how the *IM* is integrated into digital simulation tools. The architecture/implementation and the features provided by the `invTool` are described in Section 4, the experimental setup and the results obtained for our sample circuits are presented in Section 5. Finally, the paper closes with some conclusions in Section 6.

2. Involution model

We briefly summarize the most relevant properties of the involution model in Section 2.1, compare the utilized switching waveforms in Section 2.2, and explain the general principle of circuit simulations in this model in Section 2.3. The interested reader is referred to [11] for additional details.

2.1. Involution channels

When introducing the involution model in [10,11], Függer et al. have shown that its self-inverse delay functions arise naturally in a (generalized) standard analog model that consists of a pure delay component, a slew-rate limiter with generalized switching waveforms, and an ideal comparator, as shown in Fig. 3. First, the incoming, binary-valued input u_i is delayed by a pure delay T_p, which is necessary to assure causal channels, i.e., $\delta_{\uparrow/\downarrow}(0) > 0$. For every transition on u_d, the generalized slew rate limiter immediately switches to the corresponding waveform ($f_\downarrow$ for a falling and $f_\uparrow$ for a rising transition) in a way that the value at u_r, representing the analog output voltage, does not jump. Finally, the comparator generates the output u_o by discretizing the value of this waveform w.r.t. the threshold voltage V_{th}.

To calculate the delay function $\delta_\downarrow(T)$, as detailed in [10], one has to determine the value of u_r as the falling transition on u_d arrives and the

[1] The `invTool` can be found on GitHub: https://github.com/oehlinscher/InvolutionTool.

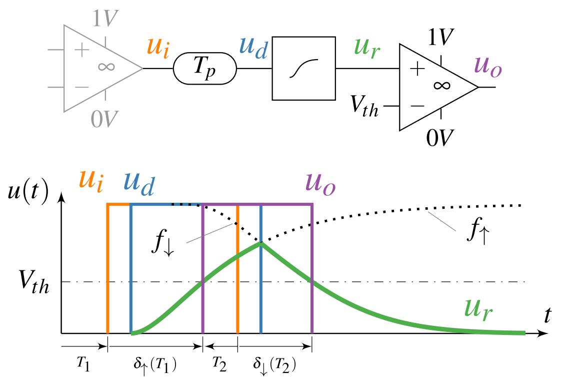


Fig. 3. Simple analog channel model (upper part) with a sample execution (bottom part) taken from [10]. At a transition on u_d (blue) the transitions waveforms are immediately switched (green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

time it takes from there onwards to return to V_{th}. For this purpose, we compute the delay of a perfectly idle channel $\delta_\infty^\uparrow = \lim_{T\to\infty} \delta_\uparrow(T)$ and $\delta_\infty^\downarrow = \lim_{T\to\infty} \delta_\downarrow(T)$ from a transition on u_i to reaching V_{th} on u_r as

$$\delta_\infty^\uparrow = T_p + f_\uparrow^{-1}(V_{th}) \quad \text{and} \quad \delta_\infty^\downarrow = T_p + f_\downarrow^{-1}(V_{th}). \tag{1}$$

For a time difference of T between the last transition on u_0 and the current one on u_d, the value of u_r can now be expressed as $f_\uparrow(T+\delta_\infty^\uparrow)$. To finally get $\delta_\downarrow(T)$, the time it takes for $f_\downarrow$ to reach u_r has to be subtracted from $\delta_\infty^\downarrow$, i.e.,

$$\begin{aligned}\delta_\uparrow(T) &= \delta_\infty^\uparrow - f_\uparrow^{-1}(f_\downarrow(T+\delta_\infty^\downarrow)) \quad \text{and}\\ \delta_\downarrow(T) &= \delta_\infty^\downarrow - f_\downarrow^{-1}(f_\uparrow(T+\delta_\infty^\uparrow)).\end{aligned} \tag{2}$$

If the slew rate limiter is implemented as a first-order RC low pass filter, for example, we obtain what is called an *Exp-channel*: The switching waveforms are $f_\downarrow(t) = 1 - f_\uparrow(t) = e^{-t/\tau}$ here, with τ being the RC time constant that determines its steepness. Inserting these functions and their inverses into Eqs. (2) and (1), we obtain

$$\begin{aligned}\delta_\uparrow(T) &= T_p - \tau\ln(1-V_{th}) + \tau\ln(1-e^{-(T+T_p-\tau\ln(V_{th}))/\tau})\\ \delta_\downarrow(T) &= T_p - \tau\ln(V_{th}) + \tau\ln(1-e^{-(T+T_p-\tau\ln(1-V_{th}))/\tau}).\end{aligned} \tag{3}$$

Recall that $T_p > 0$ is required to ensure causality of the Exp-channel, i.e., $\delta_\uparrow(0) > 0$, $\delta_\downarrow(0) > 0$.

The `invTool` also supports involution channels based on the well-known *Hill function* [15], which matches real switching waveforms better than the exponential function; they are called *Hill-channels* in this paper. Their switching waveforms are

$$f_\downarrow(t) = 1 - f_\uparrow(t) = \frac{t^n}{k^n + t^n}, \tag{4}$$

the parameter k basically determines when the threshold is reached and thus primarily depends on $\delta_\infty^\uparrow$ resp. $\delta_\infty^\downarrow$, T_p and V_{th}. The parameter n (the Hill coefficient) can be chosen freely to adjust the actual switching speed. By using

$$\begin{aligned}\delta_\infty^\uparrow &= T_p + k_\uparrow \sqrt[n_\uparrow]{\frac{1-V_{th}}{V_{th}}} \quad \text{and}\\ \delta_\infty^\downarrow &= T_p + k_\downarrow \sqrt[n_\downarrow]{\frac{V_{th}}{1-V_{th}}}\end{aligned} \tag{5}$$

and again inserting these functions and their inverses into Eq. (2), we obtain

$$\begin{aligned}\delta_\uparrow(T) &= \delta_\infty^\uparrow - k_\uparrow\left(\frac{k_\downarrow}{T+\delta_\infty^\downarrow}\right)^{\frac{n_\downarrow}{n_\uparrow}} \quad \text{and}\\ \delta_\downarrow(T) &= \delta_\infty^\downarrow - k_\downarrow\left(\frac{k_\uparrow}{T+\delta_\infty^\uparrow}\right)^{\frac{n_\uparrow}{n_\downarrow}}.\end{aligned} \tag{6}$$

Another variant of an involution channel can be obtained by using the slew-rate limiter to model the electrical behavior of interconnecting wires. This is quite reasonable, since nowadays wire delays are more dominant than gate delays. A popular model here is the *Π* model [16], which can be reduced to a second-order system consisting of two RC low pass filters in series. Calculating the switching waveform of such a system yields a linear combination of two exponential functions, with different[2] positive time constants τ_1, τ_2:

$$\begin{aligned}f_\uparrow(t) &= 1 - x_1 e^{-\frac{t}{\tau_1}} - (1-x_1)e^{-\frac{t}{\tau_2}} \quad \text{and}\\ f_\downarrow(t) &= 1 - f_\uparrow(t) = x_1 e^{-\frac{t}{\tau_1}} + (1-x_1)e^{-\frac{t}{\tau_2}}\end{aligned} \tag{7}$$

Eq. (7) has more degrees of freedom than Exp and Hill-channels, which makes fitting to real switching waveforms easier: τ_1 and τ_2 are used to parameterize the speed of each exponential function, whereas x_1 is used to control the ratio between the two exponential functions. Like for all other involution channels, T_p can be chosen freely. Since Eq. (1) imposes constraints on the switching waveform, however, we effectively lose one degree of freedom: In particular, it is possible to compute x_1 out of the other parameters such that the constraints for Eq. (1) are fulfilled. Compared to the other two channel types, empirically parametrizing the SumExp-channel is a lot more challenging, as choosing unsuitable values for the time constants quickly leads to unreasonable waveforms, for example ones that go above 1 respectively below 0.

Therefore, we augmented Eq. (7) by means of an additional time scaling factor c that allows to stretch or compress the waveform. The additional degree of freedom due to c compensates for the loss caused by fulfilling the constraints for Eq. (1). This is particularly beneficial for empirical parametrization, as it is possible to first define the general shape of the waveform (using x_1, τ_1 and τ_2) and subsequently use c to cross V_{th} at the intended point in time. The corresponding involution channel is called *SumExp-channel* in this paper:

$$\begin{aligned}f_\uparrow(t) &= 1 - x_1 e^{-c\frac{t}{\tau_1}} - (1-x_1)e^{-c\frac{t}{\tau_2}} \quad \text{and}\\ f_\downarrow(t) &= 1 - f_\uparrow(t) = x_1 e^{-c\frac{t}{\tau_1}} + (1-x_1)e^{-c\frac{t}{\tau_2}}.\end{aligned} \tag{8}$$

Note that, for each parameter set, i.e., for each gate's characteristic δ_∞ (see Section 3 for details), the scaling factor c has to be computed individually. Since there is no closed-form solution for c, we employ the Newton–Raphson method for this purpose. Note that, since there is no closed form for the delay function in Eq. (2) either, the `invTool` must calculate the actual delay values during simulation numerically as well.

2.2. Channel parametrization

Sample switching waveforms of the utilized *IM* channels are shown in Fig. 4; we only depict one switching waveform ($f_\uparrow$) for better readability. Our channels differ not only in the shape of the corresponding waveforms, but also in the effort for empirical characterization, which will be sketched below. Note that the problem of how to *systematically* determine the parameters of, say, a SumExp-channel in order to match the delay function of a given gate in a given technology, which is already difficult for the simple DDM model [17,18], is outside the scope of this paper.

The first complication, which we neglected on purpose so far for simplicity, is that in general not only $\delta_\infty^\uparrow \neq \delta_\infty^\downarrow$ but also $f_\uparrow(t) \neq 1 - f_\downarrow(t)$. Consequently, rising and falling switching waveforms must be characterized separately.

The first and foremost property that has to be guaranteed here is that the switching waveform hits V_{th} exactly at time $\delta_\infty^\uparrow - T_p$ resp. $\delta_\infty^\downarrow - T_p$. Then, by varying the parameters, the waveform is tuned such that the

[2] Actually, a SumExp-channel degenerates to an Exp-channel when $\tau_1 = \tau_2$.

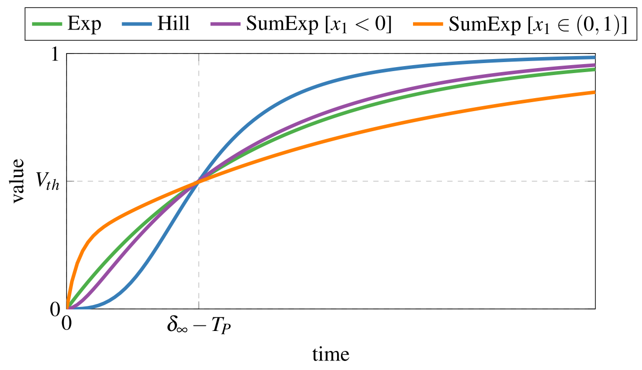


Fig. 4. Sample switching waveforms for the utilized *IM* channels (only one direction shown). Recall that the SumExp-channel provides more degrees of freedom than the other ones. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

shape of the resulting delay function matches the desired one more closely (see Fig. 5 for some examples). Note that the Exp-channel does not provide sufficiently many parameters to allows this: By fixing $\delta_\infty^\uparrow$, $\delta_\infty^\downarrow$ and T_p (which is identical for both), the time constant τ and thus the shape of the waveforms and, consequently, the shape of the delay functions is fully determined according to

$$\tau_\downarrow = -\frac{\delta_\infty^\downarrow - T_p}{\ln(V_{th})}$$
$$\tau_\uparrow = -\frac{\delta_\infty^\uparrow - T_p}{\ln(1 - V_{th})}. \tag{9}$$

Similarly, for the Hill-channel, the parameters $k_\downarrow$ and $k_\uparrow$ can be computed as

$$k_\downarrow = (\delta_\infty^\downarrow - T_p) \cdot \left(\frac{V_{th}}{1 - V_{th}}\right)^{\frac{1}{n_\downarrow}}$$
$$k_\uparrow = (\delta_\infty^\uparrow - T_p) \cdot \left(\frac{1 - V_{th}}{V_{th}}\right)^{\frac{1}{n_\uparrow}}. \tag{10}$$

In contrast to the Exp-channel, however, Hill-channels have an additional degree of freedom: The parameters $n_\uparrow$ and $n_\downarrow$ can be freely chosen, whereat higher values result in steeper switching waveforms. Note that only their ratio is relevant for the resulting delay functions, as revealed by Eq. (6).

The most flexible *IM* channel is the SumExp-channel given in Eq. (8), which is parametrized via x_1, τ_1, τ_2 and c. Similar to the Hill-channel, the corresponding delay functions only depend on the ratio $A = \tau_2/\tau_1$, whereat we assume without loss of generality that $A > 1$. Depending on the choice for A, the value of x_1 then determines the shape of the waveform. As pointed out, these choices are very delicate, because inappropriate values lead to over/undershooting. Waveforms resembling real signals (that look similar to Hill functions, i.e., start with a derivative of zero, getting steeper, and finally flat again, like the purple example in Fig. 4) are achieved for $x_1 = 1/(1 - A)$, as a short calculation reveals. For our assumption $A > 1$, we end up with $x_1 < 0$ and thus $1 - x_1 > 1$, i.e., with a difference of two exponentials. Unfortunately, however, such traces did not lead to satisfactory fittings of the delay functions.

Consequently, we had to treat x_1 as a freely adjustable parameter as well. Good results w.r.t. matching the desired delay functions have eventually been obtained for $x_1 \in (0, 1)$, which leads to $1 - x_1 \in (0, 1)$ and thus to a sum of exponentials again. Note carefully, however, the corresponding switching waveforms looks very unrealistic (see the orange curve in Fig. 4). However, a nice feature of such waveforms is the ability to easily read-off suitable parameter values from the desired shape: A determines the ratio between the steepness at beginning and end of the curve, while $(1-)x_1$ defines the approximate value where the transition from the part dominated by τ_1 to the one dominated by τ_2 occurs. Finally, by using $\delta_\infty^\uparrow$ resp. $\delta_\infty^\downarrow$, the appropriate value of c can be determined numerically, as no closed-form expression exists.

2.3. Simulations

Viewed at the level of digital signals, the behavior of an involution channel is defined by the channel simulation Algorithm 1, which maps channel input signal s (with event list Input) to channel output signal $f_C(s)$ (with event list Output). Event lists hold the transitions (t, s) of a signal, where t is the occurrence time and $s \in \{0, 1\}$ the signal value after the transition. Signals contain an initial transition at time $-\infty$, potentially followed by transitions with increasing non-negative times and alternating values. Since output signal transitions of an involution channel may cancel each other, the simulation algorithm also maintains an internal event list Pending: An output transition is only fixed, i.e., moved from Pending to Output, when it cannot be canceled later on.

Algorithm 1 Channel algorithm, up to time τ.

```
1: Pending ← [ ]; Output ← [ ]
2: $(-\infty, x_{-\infty})$ ← initial event in Input
3: add $(-\infty, x_{-\infty})$ to Output
4: Prev ← $(-\infty, x_{-\infty})$
5: for $(t, x)$ in Input with $0 \le t \le \tau$, ascending in time $t$ do
6:   $(t', x')$ ← Prev
7:   if $x = 1$ then $\delta \leftarrow \delta_\uparrow(t - t')$ else $\delta \leftarrow \delta_\downarrow(t - t')$ endif
8:   Prev ← $(t + \delta, x)$
9:   if $t + \delta \le t'$ then
10:      remove $(t', x')$ from Pending
11:   else
12:      if exists move $(t', x')$ from Pending to Output
13:      add $(t + \delta, x)$ to Pending
14:   end if
15: end for
16: if exists $(t, x)$ in Pending with $t \le \tau$, add it to Output
17: return Output
```

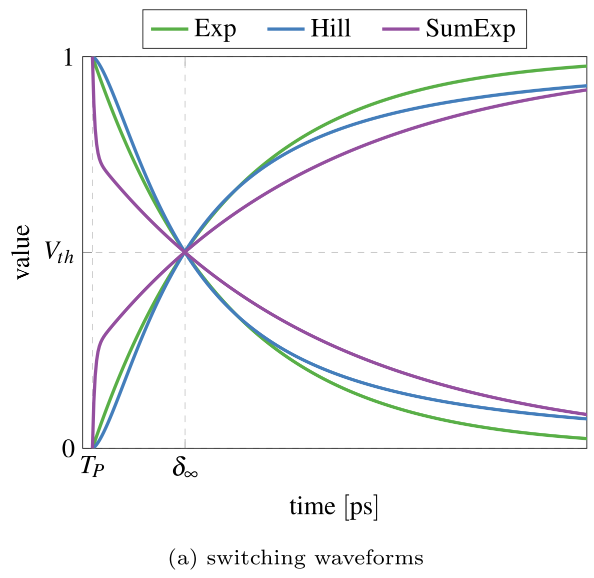
When being implemented in a digital simulation tool (in our case *Questa*), Algorithm 1 can be simplified dramatically, since transitions in the wrong temporal order are dropped automatically. Therefore, it suffices to calculate $\delta(T)$ and delay the input transition by exactly that amount, leaving it to *Questa* to cancel wrongly ordered transitions. Note that we did not verify this convenient property for alternative simulation suites.

3. Incorporating *IM* in *Questa* in practice

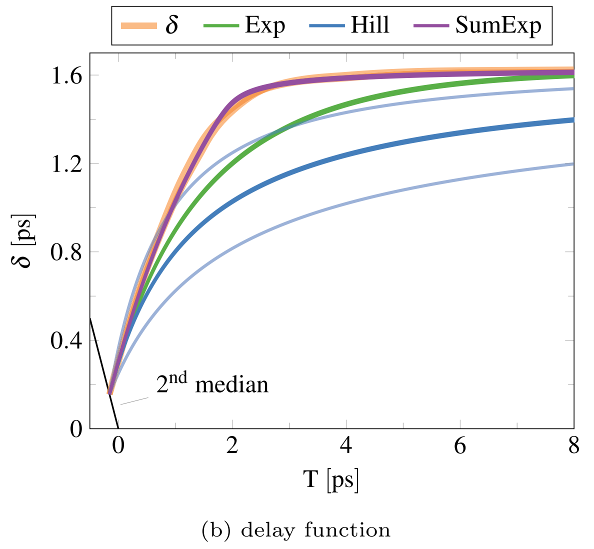
One of the main reasons for developing the `invTool` was our desire to perform circuit simulations using the *IM* without the need to install and utilize some non-standard software tool. For that reason, we used VHDL Vital as our guidance for the development of the `invTool`. As a result, changing between our *IM* implementation and the former is achieved simply by switching libraries, which facilitates code and test setup re-using.

Consequently, our solution not only has the same structure as VHDL Vital, but also responds to the same variables and is also written in VHDL. Simulations in the `invTool` are completely controlled by *Questa*, which makes it possible to use all its features without restrictions: Based on the next input transition time at the channel input, the algorithm determines T and the resulting $\delta(T)$, and adds the transition to the channel's output. This is done separately for each channel, as their parameters can differ.

For simulations using the `invTool`, one hence needs exactly the same input files as for any standard post-layout simulation: the circuit,



(a) switching waveforms



(b) delay function

Fig. 5. Fitting results for an inverter (INV_X1) using Exp-channel, Hill-channel and SumExp-channel. The latter has a very artificial shape of the switching waveforms, which however achieves an almost perfect match of the experimentally determined delay function of a real gate. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

a testbench, and the timing characteristics stored in *.sdf* files. The latter contain the static delay of each gate ($\delta_\infty^\uparrow$ and $\delta_\infty^\downarrow$) in the circuit and the interconnects in between.

While VHDL Vital uses essentially pure/inertial delays with a priori given fixed delay values, the channels used for the *IM* need to be parametrized to evaluate the delay functions $\delta_\uparrow$ and $\delta_\downarrow$. In this paper, we rely on the educated guesses and empirical fits sketched in Section 2.2 for all those parameters that are not fully determined by the delay characteristics in the *.sdf* files, namely, the pure delay T_p for all our channels, the values for $n_\uparrow$ and $n_\downarrow$ for Hill-channels, and (the ratio of) τ_1 and τ_2 and x_1 for SumExp-channels. Note that the parameter sweeping capabilities of the `invTool` also allow to experimentally determine the impact of a parameter like T_p, as we demonstrate in Section 5.

Fig. 5 shows a typical fitting result for a specific value of T_p. Note that some delay functions are so close together that they appear as a single line in the figure. Clearly, the SumExp-channel achieves the best results, followed by the Exp and Hill-channel. While the Exp-channel

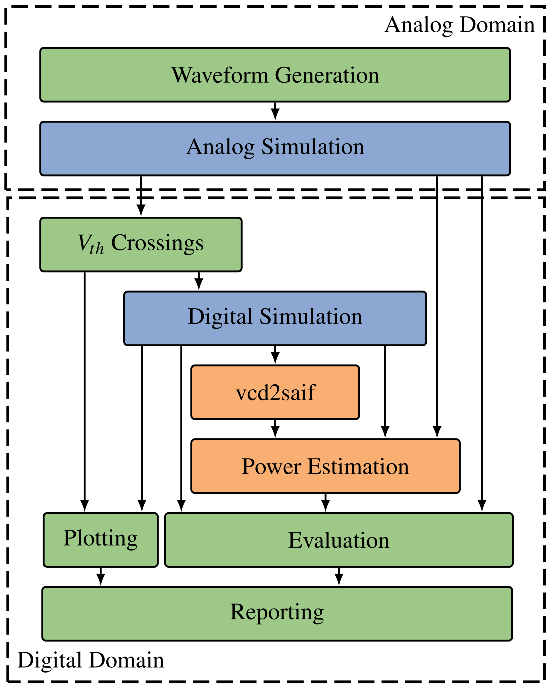


Fig. 6. Workflow in the `invTool`. The green parts had to be implemented from scratch. The blue parts have been available, albeit the available resources had to be extended significantly. Orange parts could be used almost out of the box. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

is fully determined by T_p and $\delta_\infty^\uparrow$ resp. $\delta_\infty^\downarrow$ (cp. Section 2.2), the Hill-channel has two free parameters ($n_\uparrow$ and $n_\downarrow$) that can be adjusted: By varying their ratio (remember that this is the only important parameter for the delay function), the delay functions $\delta_\uparrow(T)$ and $\delta_\downarrow(T)$ that are the same for $n_\uparrow/n_\downarrow = 1$ can be made different, as shown for $n_\uparrow/n_\downarrow = 1.5$ by opaque blue lines in Fig. 5b. Unfortunately, tuning the ratio typically increases the matching for one direction while decreasing it for the other. For NAND gates, we even observed such effects for SumExp-channels. One important avenue of future research is hence definitely a systematic approach for characterizing the parameters of a channel model in order to match the delay function of a given real gate.

Baseline delay models. Besides checking our results against fully fledged analog simulation results, we also compare them to state-of-the-art digital delay models. Depending on the used gate library, either VHDL Vital [19, Chapter 9] or the Verilog delay model [20, Chapter 14] is applied. Both implementations offer the possibility to use pure and inertial delay channels.

- Verilog uses two parameters (rejection: *pulse_r* and error: *pulse_e*) to control the behavior of the delay model. If both are set to 0, pulses are never rejected and hence a pure delay model is configured. If both parameters are set to 100, all pulses with a width smaller than δ_∞ are rejected. In the simulations performed in Section 5, we used solely the inertial delay implementation of Verilog for comparison.
- In Vital, the type of delay model is configured already in the library, and cannot be overridden via command line parameters. It offers a simple inertial (*VitalInertial*) and pure delay (*VitalTransport*) model. Note carefully, however, that the default setting in the libraries we used for simulation was *OnDetect*, a mode which outputs “unknown” (‘X’) values in the case of a glitch (the parameters of Verilog allow the user to configure a similar behavior). To ensure a fair comparison with the *IM*, this needs to be unset.

Note that we verified our channel simulation framework by also modeling pure delay channels and comparing the results to the ones achieved by using VHDL Vital and Verilog pure delay models. Since no difference could be observed, we can reasonably assume that our involution channel implementations work properly.

Another crucial task for making the `invTool` useful in practice is to provide a well-populated set of available basic gates, which consist of Boolean functions interconnected by *IM* channels. In order to incorporate a new gate, it suffices to model the Boolean functionality in VHDL and to connect inputs and outputs via suitable *IM* channels. While single-input single-output gates are easy to handle, things get slightly more complicated for multi-input gates, as there are different possible locations for placing the (single-input single-output) *IM* channels: As they can be placed either at the output of a gate or at its input(s), which may result in different timing behaviors of the overall gate, this sometimes needs careful consideration.

4. Involution Tool (`invTool`)

Our Involution Tool `invTool`, which was originally developed in [21], is a complete framework for the systematic and automatic evaluation of different delay prediction methods for several power and timing metrics. It allows to generate user-controlled random input vectors, to run different analog/digital simulations, to automatically sweep the ranges of user-defined parameters, and to generate various reports on the results. In Section 4.1, we outline the workflow of using the `invTool`, which also provides a glimpse on its overall architecture and its core features. In Section 4.2, we explain how the tool supports parameter sweeping and multiple simulation runs.

4.1. Workflow of the `invTool`

The overall information flow in our tool is shown in Fig. 6. Around the central delay estimation method (termed *Digital Simulation* in our figure, see Section 3 for details), which is implemented in *Questa* (also called QuestaSim), we developed a complete framework that handles everything from waveform generation to evaluation and reporting autonomously. The only required inputs are (i) a *SPICE*/Verilog description of the circuit under test as well as (ii) the corresponding timing file, which can be created using *Cadence Encounter* [14], for example. Due to its modular structure, each part of the toolchain can be substituted, provided that the interfaces do not change. Note that this feature became crucial in the course of the experiments described in this paper, as we encountered numerical issues with *HSPICE* that forced us to switch to *Spectre* for some circuits/technologies.

Waveform generation. The first task of `invTool` is to generate the test stimuli for the circuit. Since suitable test vectors largely depend on the actual circuit, we resorted to randomly generating transitions on the input(s) here. More specifically, for each input, the time until the next transition follows a Gaussian distribution, whose parameters (μ, σ) can be set by the user. Besides the obvious restriction to non-negative inter-transition times, it is also possible to specify an optional worst case bound $\beta > 0$. If set, it prohibits inter-transition times outside $\mu \pm \beta$ (by causing a re-sampling).

This way, desired constellations, such as pulse cancellations in the circuit, can be made more probable than others, and even ruled out for sure via β. The user can furthermore choose between two modes of operation: In one the randomly determined delay is added to the last global transition time, while in the other it is added to the last transition at the corresponding input. By grouping multiple inputs, the user can increase the probability for transitions in a short period of time for the signals of the group, which can be useful for multi-input gates. All these choices can be saved, together with the parameters (μ, σ), in a configuration file, which is finally read by a Python script that ultimately determines the actual transition times.

Analog simulation. To evaluate the accuracy of our predictions, we need a "golden reference", which are currently analog simulations using *HSPICE* or *Spectre*. Initially, the randomly generated input transition times are transformed to an analog curve using a piecewise linear (PWL) source with a rise/fall time of 1 ps. This source is then added to a template file, which also imports the circuit under test and is used for the subsequent analog simulations. Since the input transitions are very steep (mostly likely too steep for certain circuits), it is possible to employ inverter chains in front of each input to shape the signal. For the technologies used in our experiments, two inverters proved to be sufficient to generate realistic signals. Note that we provided a separate supply voltage source to these shaping circuits to prevent any effect on our power measurements.

Our analog simulations provide us with both (i) a reference for the actual power consumption and (ii) reference switching traces (i.e., threshold crossings) at different nodes within the circuit. In the next step of our tool chain, these *SPICE* switching traces are fed to a Python script (termed V_{th} *Crossings* in Fig. 6), which generates input files that can be interpreted by the succeeding digital simulation tool, in our case, *Questa*.

Digital simulation. All digital simulations, as described in Section 3, are performed by *Questa*. The input files, which are extracted from the analog simulation switching traces, are read by the testbench and applied to the circuit under test as inputs. Different delay models can be used in these simulations: some provided VHDL Vital or Verilog library, denoted as `STA` in the sequel, or our *IM* library (`INV`).

Of course, as pointed out in Section 3, every gate used in the circuit under test has to be present in the respective library. In the case of *IM*, the `invTool` is capable of automatically generating a corresponding entry for simple gates, i.e., those consisting of a single combinational function only. The parameters of the gates (in particular, channel type, pure delay, location and specific parameters) can be specified in a configuration file. For more complex gates, the user has to enter the description of the gate in VHDL manually, which gives full control on channel types and locations. The `invTool` is also capable of generating a complete gate library, which can be used as drop-in replacement in already existing simulation environments.

Power estimation. In addition to the *SPICE*-generated analog power estimation, the `invTool` allows to use multiple digital power estimation tools, currently *Design Compiler* and *PrimeTime*, where for the latter two different modes (average and time-based) can be chosen. Note that in our simulations these two differed quite significantly. As the reference power estimation, the `invTool` allows to choose between the following two alternatives: (i) the *SPICE* analog power estimation and (ii) the *Design Compiler* and *PrimeTime* power estimation generated for the *SPICE* switching traces (obtained from the *SPICE* waveforms). For power comparison, the results of the *Questa* simulation (`STA`, `INV`) are fed into the same tools as in (ii) and compared to the reference value. Note that the *Design Compiler* and the average-based simulation mode of *PrimeTime* only use the switching activity information file (.saif), whereas the time-based simulation utilizes a value change dump file (.vcd), which also contains information about the time of the transitions.

Evaluation. As a first step, the tool converts all the results from the previous stages into a unified format. The following four metrics are supported:

- *Power deviation*: As already mentioned, the results of the digital simulation of the involution gates (`INV`) and the standard gates (`STA`) are compared to the two types of reference values (where the second type actually produces three values, one for each tool and option). This way, we can identify deviations and bias caused by the different power estimation techniques.

- The *number of transitions* in the digital simulation trace, for the signal at every node, is calculated and compared to the corresponding *SPICE* switching traces. Both the average deviation and the maximum deviation are computed.
- The deviation between a digital simulation trace and the corresponding *SPICE* switching trace is measured via the *area under the deviation trace*, which can be computed with and without induced resp. suppressed glitches. Note that the `invTool` actually computes signed areas, normalized to $V_{DD} = 1$ and per transition (i.e., divided by the number of relevant transitions), with the sign depending on whether the transition of the reference signal comes first (negative, representing a *trailing* transition) or not (positive, representing a *leading* transition). Note that the normalized area per transition effectively represents the (average) time a transition happens before or after the reference signal. This feature is extremely useful for determining a bias in the delays, e.g., caused by inaccurate information in the *.sdf* file. Fig. 7 shows an example trace with leading (blue) and trailing (green) transitions. The average time that the reference transition happens before the digital trace is $\frac{0.2+0.2+0.1+0.1+0.1}{5} = 0.14$ time units. The result for the trailing metric is $\frac{0.1+0.1}{2} = 0.1$ time units per transition. Note that we ignored the deviation caused by original glitches here, since their influence is considered in a separate metric.
  Due to the design of our involution channels, they will always perform better compared to a pure delay channel for the *trailing* category, as the maximum delay δ_∞ is the one that the pure delay channel uses at every transition. On the other hand, with respect to *leading* transitions, the *IM* predictions will always be worse for the same reasons. Some caveat is in order when comparing numerical values of the leading (or trailing) metric per transition for different models, however: Even if the underlying *SPICE* trace is the same, the number of relevant transitions for different models is likely to be different, due to suppressed and/or induced glitches. Since such glitches barely change the total area under the deviation trace, this leads to a possibly significantly different normalized area per transition.
- During the comparison of a *SPICE* switching trace and the corresponding digital simulation trace, the tool checks whether a pulse (= two subsequent transitions, starting from and returning to the current level of the other trace) happens in one trace without any transition in the other one: If such a pulse occurs in the *SPICE* switching trace, we call it an *original suppressed glitch*, otherwise an *original induced glitch*. The `invTool` also evaluates whether a pulse in one trace properly contains a pulse in the other trace; we call such a pulse an *inverted suppressed glitch* resp. an *inverted induced glitch*. It outputs the number of those glitches divided by the total number of transitions in the corresponding signal. Fig. 7 shows an example trace depicting a suppressed glitch (s) on the *SPICE* trace, and an induced glitch (i) on the digital trace, i.e., the trace obtained by using *Questa*. In the example trace, the suppressed original glitch percentage and the induced glitch percentage are both $\frac{1}{9} \approx 11\%$

Section 5 will show that these relatively simple measures provide useful information on the influence of certain model parameters.

Reporting. The final step of the toolchain allows to automatically generate a LaTeX report. The default report shows (i) information about the simulation environment, (ii) waveform generation settings, (iii) power consumption, (iv) trace comparison results, (v) plots and (vi) the schematic of the circuit. Detailed information about the trace comparison results, for every node, is also stored in a .csv file. The report can hence easily be customized by the user: format and content can be configured by means of a template, which allows to incorporate all values extracted and calculated during the evaluation.

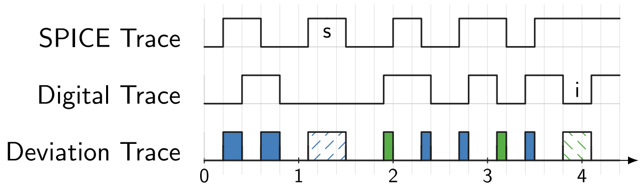


Fig. 7. Categorization of the deviation between analog and digital simulation. Shown are pulses (dashed areas) caused by original induced (i) or original suppressed glitches (s) and the leading (blue) and trailing metric (green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.2. Multi-execution and parameter sweeping

The `invTool` is capable of automatically performing multiple simulation runs, by invoking the toolchain several times. Since the waveforms are generated randomly, this is an important feature for obtaining reasonable results. Furthermore, sweeping over different channel parameters and waveform generation settings is supported by the tool. In the following, we denote a complete sweep over all configured parameters as *simulation run*, whereas *simulation* denotes a single execution with a certain configuration.

The following channel parameters are supported:

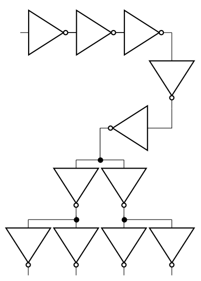
- *Pure delay* T_p.
- *Channel location*: Decides whether the channels are placed at the inputs or the outputs of the gates.
- *Channel-specific parameters*: For Hill-channels, $n_\uparrow$ and $n_\downarrow$ can be specified. For SumExp-channels, τ_1, τ_2 and x_1 can be set.

An important feature of our multi-execution is that the generated waveforms can be retained between subsequent simulations, as long as only the channel parameters are modified. Since these parameters only influence the involution channel parts of the simulation, which are solely handled by *Questa*, a lot of simulation time (up to 2/3) can be saved. After all, the analog simulation needed for waveform generation is the most time-consuming part of the toolchain. Moreover, starting from the same input waveforms increases reproducibility and comparability of the simulation results.

After finishing all simulation runs in multi-execution, the tool aggregates (i.e., averages) the results from all simulations and generates a report, again based on a LaTeX template. Moreover, the results are exported to a .csv file again, which allows further post-processing. Using the `invTool`, in particular the multi-execution feature, enables easy comparison between different channels and also experimenting with new channel types to check hypotheses.

5. Results

To demonstrate the utility of the `invTool` and to validate/reject some conjectures w.r.t. the existing involution model, we present the results of the evaluation of three different circuits, namely, an inverter tree, the clock tree of an open-source MIPS processor [14] and a custom NAND circuit. For our simulations we used a standard 65 nm UMC library ($V_{DD} = 1.1$ V) and a Nangate Open Cell Library with FreePDK15^{TM} 15 nm FinFET models [22] ($V_{DD} = 0.8$ V) to investigate the accuracy for different technologies. Note that we used *HSPICE* in combination with the 65 nm library, while we had to resort to *Spectre* for the 15 nm one, since we ran into numerical issues using *HSPICE* in this setup. However, due to the modular design of the `invTool`, this switch of tools was easily achievable. The inverter tree and the NAND circuit were analyzed for both technologies while for the MIPS clock tree solely the more modern 15 nm library was used. Below, we describe the results of the evaluation of these circuits and draw some conclusions from our findings.


(a) inverter tree

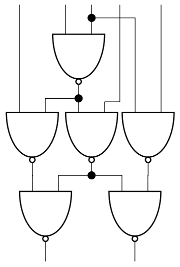

(b) NAND circuit

Fig. 8. Schematic of circuits used for simulation.

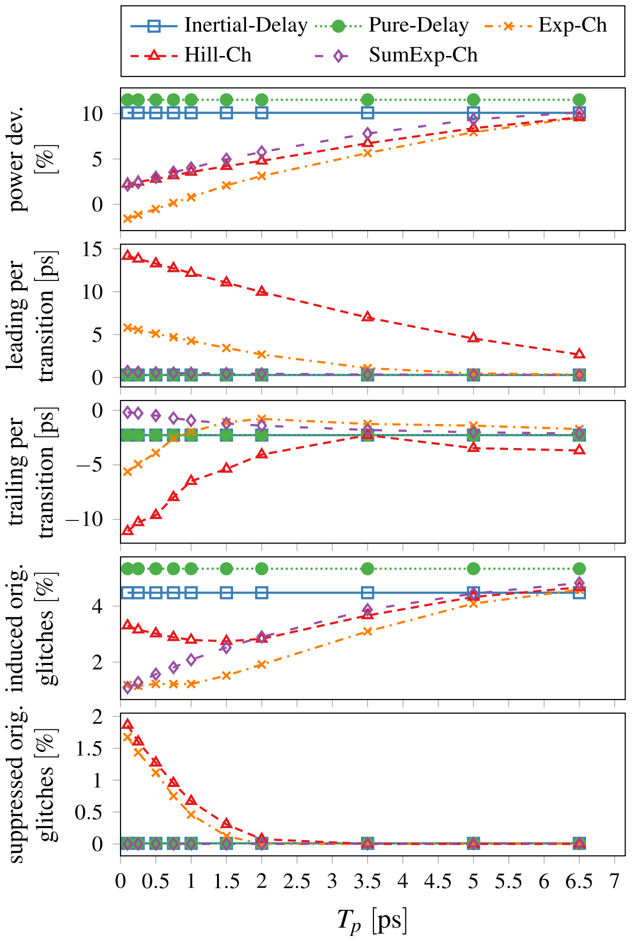


Fig. 9. Inverter tree simulation results (65 nm) for $\mu = 45$ ps.

Among all the metrics extractable by the `invTool`, as described in Section 4.1, we selected the following

- Power deviation: We focused on the comparison between *SPICE* and the *PrimeTime* time-based simulation mode. In Section 5.5 we discuss the differences between the used power estimation tools, and why focusing on one tool is reasonable.
- Leading and trailing normalized area under the deviation trace per transition (without glitches): This metrics give insights how accurate the transition time is estimated, compared to *SPICE*, which is useful when examining properties of different channels.
- Percentage of original induced and suppressed glitches: The glitch metrics can be used to find out how susceptible different channels are to glitches. Especially for clock networks and asynchronous logic, accurate modeling of glitches is a major concern.

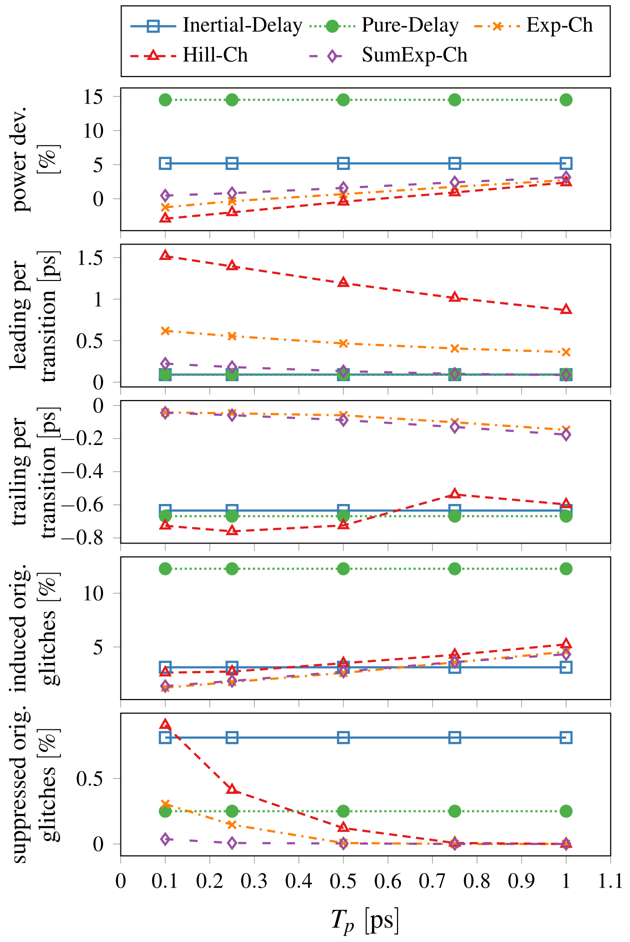


Fig. 10. Inverter tree simulation results (15 nm) for $\mu = 15$ ps.

In all simulations, we use Exp-channels, Hill-channels and SumExp-channels and compare them to the default Verilog inertial delay model and a pure delay model. For every parameter setting, we averaged the results of 10 randomly generated traces, which was found to be sufficient to reasonably average out the stochastic variations in all our experiments.

Using the multi-execution feature of our tool, we carried out simulations for multiple values of T_p and varying channel parameters: For the Hill-channel we swept over the ratio $\frac{n_\downarrow}{n_\uparrow}$, and for the SumExp-channel over x_1, τ_1, τ_2. Since the results for the Hill-channel were best when using $\frac{n_\downarrow}{n_\uparrow} = 1$, the following plots omit results with different ratios. For the SumExp-channel, $x_1 = 0.25, \tau_1 = 30$ fs and $\tau_2 = 3000$ fs have been chosen for the inverters, while changing them to $x_1 = 0.42, \tau_1 = 20$ fs and $\tau_2 = 7000$ fs for the NAND gates.

5.1. Inverter tree

Fig. 8a shows the inverter tree used for our simulation experiments. Note that the minimum values of $\delta_\infty^\downarrow$ and $\delta_\infty^\uparrow$ can be as low as 7.7 ps in the 65 nm and 1.5 ps in the 15 nm technology. In order to facilitate a direct comparison of the results between those two, we chose $\mu = 45$ ps for the 65 nm library (Fig. 9), and $\mu = 15$ ps for the 15 nm one (Fig. 10). This is reasonable, since the latter is about three times faster compared to the former, with an even larger ratio for the respective minimum values of $\delta_\infty^\downarrow$ and $\delta_\infty^\uparrow$ (see above).

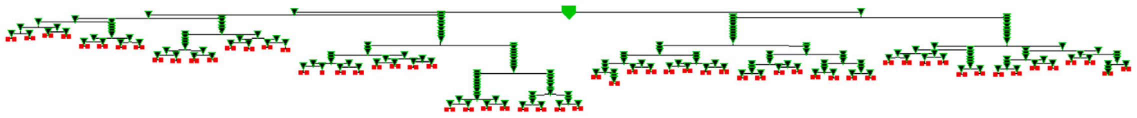


Fig. 11. Schematic of MIPS clock tree.

With respect to the power estimation accuracy,[3] for both technologies, the Exp-channel performs best, immediately followed by the SumExp-channel and the Hill-channel. The increase in the power deviation with increasing T_p is a result of the higher amount of induced glitches resp. the lower amount of suppressed glitches. The cause can easily be identified by taking a look at the analog channel model (see Fig. 3): By increasing T_p, signal u_d and thus the switching between the up and down waveform is more and more delayed. Consequently, the original waveform is followed for a longer duration, which causes pulses that formerly did not just reach the threshold, i.e., were canceled and hence counted as a suppressed glitch, to reach it, i.e., become decanceled and thus create an additional pulse at the output.

In our results, we also observe a decrease of leading (= leading normalized area under the deviation trace per transition) with increasing T_p. As explained in the previous paragraph, one possible cause is the increasing number of relevant transitions caused by less suppressed glitches. However, as the leading keeps decreasing when all suppressed glitches have disappeared (for 65 nm at around 2 ps), there must also be another cause. To spot it, recall the shape of the delay functions shown in Fig. 5b. By increasing T_p, one effectively moves the starting point of $\delta(T)$ along the 2nd median towards larger delay values, while the end point (the maximum delay $\delta_\infty^\uparrow$ or $\delta_\infty^\downarrow$) remains fixed. In other words, the closer T_p gets to the minimum $\delta_\infty^\uparrow$ and $\delta_\infty^\downarrow$, the more involution channels behave like pure delays, i.e., the less is the dependence on the switching waveform. This effectively leads to an overestimation of the delay for small values of T, which causes transitions to be scheduled later in our simulations. Thus, transitions that had been scheduled before their corresponding reference transition would be pushed closer to it (or even beyond it), thereby decreasing leading (or even setting leading to zero and increasing trailing). This effect increases with increasing T_P, which is clearly visible as the leading metric approaches zero for all channels. Note carefully that this also explains the slight increase in trailing with increasing T_p for larger values of T_p, e.g., for Exp-channel and SumExp-channel in both Figs. 9 and 10.

Somewhat counterintuitive is the behavior of the Hill-channel and, moderately so, also of the Exp-channel with respect to the trailing metric. As we explained in Section 4.1, pure delay channels are supposed to be the worst here, as they always use the maximum delay causing their transitions to be scheduled latest. And indeed, we verified this property for individual simulation traces. The reason why some involution channels are sometimes worse in the per transition trailing metric in our figures can be explained by means of the delay functions shown in Fig. 5b: Despite the fact that the Hill-channel uses a realistic switching waveform, it underestimates the delay considerably (the same is true for the Exp-channel in certain regions). This is primarily a consequence of the continuous but nevertheless instantaneous switching between up- and down waveforms, recall Section 2.1: If the rising and falling waveforms are very steep in some range, as is the case for the Hill-channel close to the threshold voltage, in particular, this creates a sharp peak in the combined waveform, which in turn causes the corresponding *IM* to underestimate the delay. Due to the underestimated delays, the number of suppressed glitches is large and the number of relevant transitions for the trailing metric is small. Dividing the total area by the latter hence causes the per transition metric to sometimes increase beyond the value for pure delay channels.

[3] Note that we did not use a *.spef* file for the power estimation with *Design Compiler* and *PrimeTime* when using the 65 nm library.

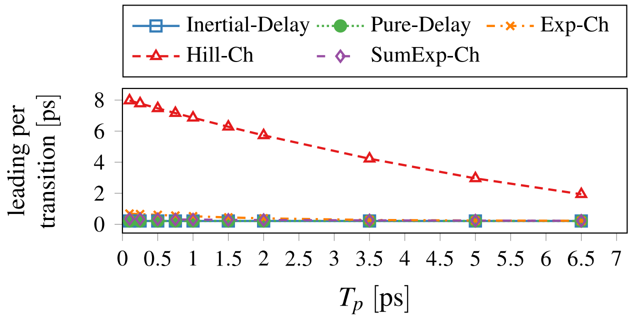


Fig. 12. Inverter tree simulation results (65 nm) for $\mu = 100$ ps.

Overall, it turns out that, surprisingly, waveforms that match real switching waveforms better, like the Hill-channel, perform considerably worse, as is shown by the leading and trailing metric both in Figs. 9 and 10.

Comparing our results for 65 nm (Fig. 9) and 15 nm (Fig. 10) indicate a quite reasonable technology independence: The SumExp-channel performs best, followed by the Exp-channel. However, in terms of absolute numbers, the results differ significantly, with lower values achieved by the 15 nm technology. This is not surprising, of course, given the fact that the maximum delay values are considerably smaller (δ_∞ of 7.7 ps versus 1.5 ps). The influence of T_p on both trailing and leading metric is also a lot less pronounced for the 15 nm technology, which is again due to the fact that we experienced larger variations of $\delta_\infty^\uparrow$ and $\delta_\infty^\downarrow$ for different gates in the first place there. In fact, only the Hill-channel shows a significant dependence on T_p, which can be traced back to the effects already explained.

For broader pulses ($\mu = 100$ ps), we found that power is estimated accurately and no glitches are induced or suppressed. As revealed by Fig. 12, however, the Hill-channel still substantially underestimates delays. Fig. 5b shows that, even for large T, the deviation between the real delay function and the Hill-channel is large, which explains its poor performance. The other involution channels, as well as the baseline delay models, perform very well. With respect to trailing, involutions now always outperform Verilog.

In summary, except for suppressed glitches and the leading metric, the Verilog inertial delay model and especially the pure delay model perform poorly compared to any involution model. Among the latter, the SumExp-channel is clearly superior.

5.2. MIPS clock

The clock tree was synthesized in 15 nm technology for a MIPS [14] with Cadence Encounter. It comprises of 227 inverters (strengths X1, X2, X4, and X8) which drive 123 Flipflops (see Fig. 11). Fig. 13 shows the results of our evaluation. Since $\delta_\infty^\downarrow$ and $\delta_\infty^\uparrow$ can be as low as 1.2 ps here, we had to restrict the range for T_p appropriately.

Normally, one would of course drive a clock tree with a frequency that is low enough in order not to cause any pulse cancellations. However, our experiments with large values of $\mu = 50$ ps and bounded variation $\beta = 5$ ps (recall Section 4.1) revealed that involution channels and standard channels do not show any significant difference in this setting. For a more aggressive value of $\mu = 15$ ps, the qualitative results for the MIPS clock are quite similar to the ones for the inverter tree. The

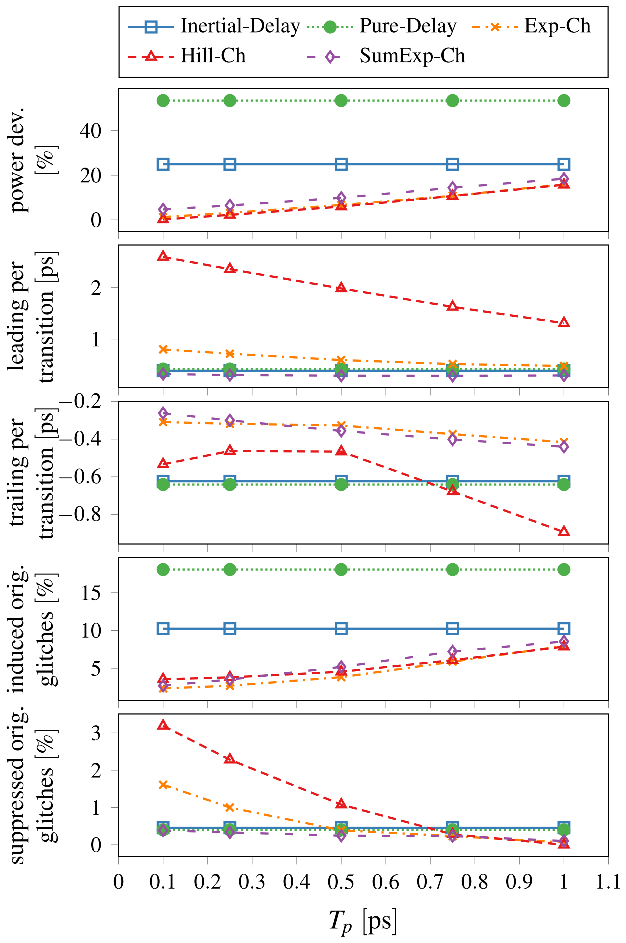


Fig. 13. MIPS clock circuit simulation results (15 nm) for $\mu = 15$ ps.

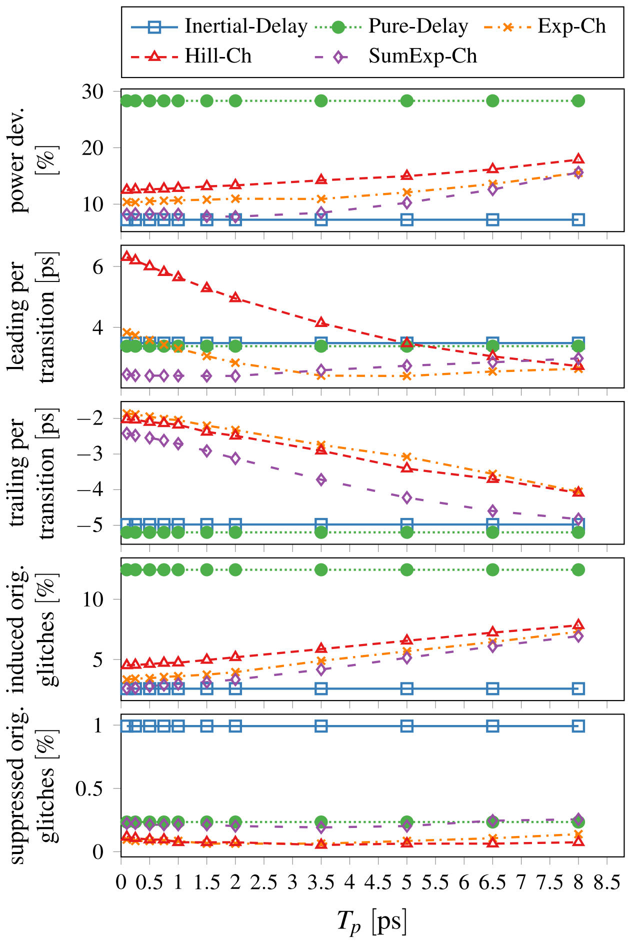


Fig. 14. NAND circuit simulation results (65 nm) for $\mu = 45$ ps, with channels placed at the output.

SumExp-channel perform particularly well: It accurately models the transition times, and also in terms of glitches it is among the best. The increasing number of induced glitches can be explained when bearing in mind that $\delta_\infty^\downarrow$ and $\delta_\infty^\uparrow$ can be as small as 1.2 ps, which results in a gradual degeneration of the different channels to a pure-delay channel. This effect causes more transitions on the digital simulation trace, and therefore the power deviation metric increases.

The trailing metric for the Hill-channel again nicely demonstrates both the influence of an increasing number of relevant transitions caused by a decreasing number of suppressed glitches (for $T_p \leq 0.25$ ps) and the effect of shifting transitions more into the future with increasing T_p causing both leading to drop and trailing to increase, i.e., the former gets better while the latter worse. Note that the leading of the SumExp-channel being below the inertial and pure delay channel is again an artifact caused by less relevant transitions, causing a sometimes excessive quotient.

5.3. NAND circuit

The NAND circuit shown in Fig. 8b was synthesized both in the 65 nm (see Fig. 14 for the results) technology and in the 15 nm technology (Fig. 15), which resulted in $\delta_\infty^\downarrow$ resp. $\delta_\infty^\uparrow$ being as low as 9.2 ps resp. 2.7 ps. The involution channels were placed either at each input or at the output of a gate in our multi-execution setting.

Since the circuit has multiple inputs, we used the local waveform generation feature of the `invTool`, which results in a higher overall density of transitions. Interestingly, placing the channels at the input was better in the 15 nm technology, whereas channels at the outputs provided better results for 65 nm. This already suggests that single-input single-output channels are not fully adequate for accurately modeling the delay of multi-input gates.

The simulation results for both technologies are comparable in the sense that the behavior of the involution channels is similar; again, the SumExp-channel outperforms the others, however. The only apparent differences are observable for the leading and trailing metric: While for the 65 nm technology inertial and pure delay are very close together compared to the accuracy of the involution channels, for 15 nm the predictions are in between. We conjecture that this is due to the fact that the pure delays are placed at the inputs of a gate for this simulation, whereas the Verilog inertial delays are tied to the gate outputs.

Consequently, it is apparent from the results in Figs. 14 and 15 that the involution models do not outperform the default Verilog inertial delay model as significantly as for the inverter tree. Another explanation for the loss in accuracy is logical masking, which makes the NAND circuit less susceptible to propagating glitches.

In addition, we strongly conjecture that any model that accurately models multi-input gates needs to consider multiple inputs *together* for computing the gate delay. For example, for our 2-input NAND gate, an accurate delay function should depend on *two* parameters, namely, the previous-output-to-input delays for both inputs. Developing suitable delay functions is of course way beyond the scope of this paper, and actually a pivotal part of our current work on extensions of the involution model.

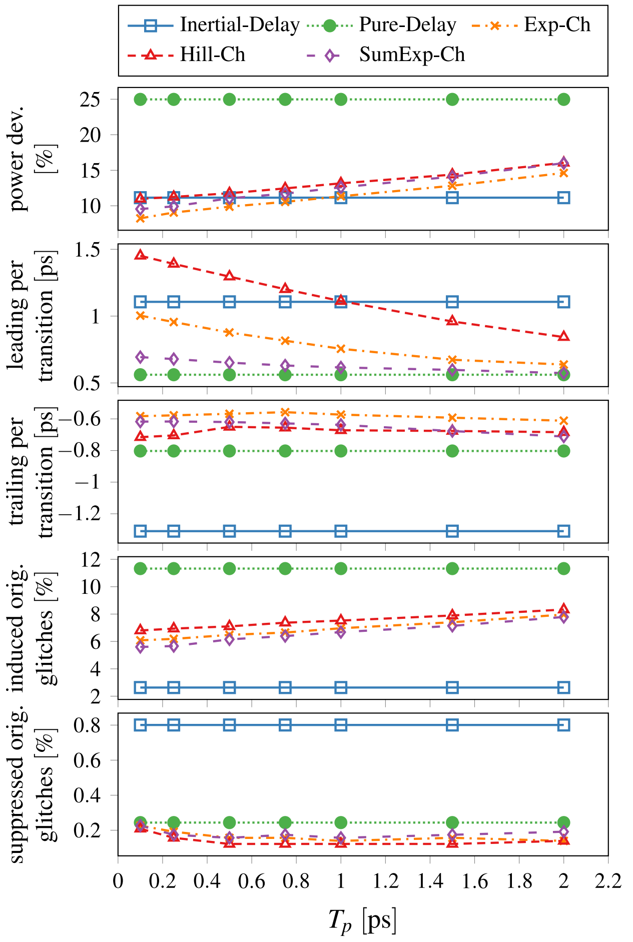


Fig. 15. NAND circuit simulation results (15 nm) for $\mu = 15$ ps, with channels placed at the input.

Table 1
Simulation time for the 15 nm MIPS clock tree ($\mu = 15$ ps, $\sigma = 10$ ps, 50 transitions at the input).

SPICE (Spectre)	Inertial (Verilog)	Pure-delay	Exp-channel	Hill-channel	SumExp-channel
576.6 s	2.122 s	2.345 s	2.357 s	2.347 s	2.364 s

5.4. Performance

A very important figure of merit for digital simulations is running time. More specifically, it is the main reason for choosing a digital simulator rather than its much more accurate analog counterpart, e.g., *SPICE*, which takes a prohibitive amount of simulation time even for circuits of small size. Hence, we also compared the simulation times of all our approaches on a server (2 Intel Xeon X5650, 1600 MHz, 32 GB RAM, CentOS 6.10) for the MIPS clock tree (see Table 1), which is the largest circuit in our test set. The results clearly reveal that the overhead introduced by using the computationally more demanding involution channels is only minimal and does not blow up for circuits of a certain size. For the MIPS clock tree, our involution channel implementations increase the simulation time only by approximately 10 % compared to the standard delay models. Interestingly, despite involving numerical computations, the performance of the SumExp-channel is only marginally lower than that of the other involution channels.

Not surprisingly, all involution channels outperform the analog *SPICE* simulations by a factor of 250. To be fair, however, we should add that our *SPICE* simulations also contain the estimation of the power consumption. Estimating the power consumption based on the results of the digital simulation adds another 2.2 s to the simulation time (*PrimeTime*, time-based mode). Note that for larger values of μ, the *SPICE* simulation time almost scales up linearly, while the digital simulation time is mostly influenced by the overall number of transitions. This is a consequence of the implemented simulation methods: *SPICE* determines all signal values at every point throughout the simulated time period. Digital simulations, however, are only triggered by a transition, causing them to be computationally much more efficient.

5.5. Tool accuracy

To ensure that the used tools yield accurate results, we checked the transition time accuracy and the power estimation accuracy.

Transition time accuracy. During simulation, we observed that the tool-generated delay estimation files (*.sdf*) are very inaccurate: They led to significant deviations between analog and digital simulations already for very broad pulses (i.e., large μ). Therefore, we eventually decided to create custom delay files based on our analog simulations. By doing so, we achieved precise delay predictions,[4] with an accuracy of ± 1 ps. These custom delay files serve the additional purpose of incorporating the interconnect delay into the gate delay. This is mandatory, as our model does not yet support separate interconnect delays. By using these custom delay files, we ensure that the leading and trailing metric, as well as the glitch percentages, yield meaningful results.

Power estimation accuracy. As mentioned in Section 4.1, the `invTool` uses different tools for estimating power. All simulation results show the deviation of the power estimation between the *SPICE* trace and the traces obtained with *Questa* when using the time-based mode in *PrimeTime* for both. Since we are interested in the relative behavior of different channels and configurations, this is a reasonable approach.

Nevertheless, we also compared the power estimation tools against *SPICE*: First, we simulated our designs in *SPICE*, logging power and signal traces. We then discretized the *SPICE* traces and fed them back into the *Design Compiler* and *PrimeTime*, obtaining power estimates. The results can be seen in Table 2. While for 65 nm simulations the tools perform similar, the time-based mode of *PrimeTime* tends to perform better for 15 nm simulations. We conjecture that the absolute values of the time-based mode are more accurate due to the different input data that the different simulations use. While the *Design Compiler* and the average-based mode of *PrimeTime* only use a switching activity file (which contains no information about time), the time-based approach has information about the switching times. One interesting observation from the results of Table 2 is that, for the simulation of the 65 nm inv_tree with $\mu = 100$ ps, the deviation between *SPICE* and the power estimation tools is quite large, while all the metrics we use for comparison show good results. For the simulation of the mips_clock, the deviation is also quite large. However, these observations are not an issue, since we are not interested in absolute values, but rather in the relative behavior of different channels and configurations. By consistently using the time-based mode of *PrimeTime*, the validity of the power deviation results is guaranteed.

6. Conclusions

In this paper, we presented an overview of the features and the internal architecture of the Involution Tool, a custom simulation environment for the involution delay model. Thanks to its embedding into the state-of-the-art digital simulation tool *Questa*, and its compatibility with

[4] As a consequence, the overall area under the deviation trace has been significantly decreased. For example, for $\mu = 100$ ps (the case shown in Fig. 12), from over 30 to below 3.5.

Table 2
Comparison of the different power estimation tools (*Design Compiler*, *PrimeTime* average based mode and *PrimeTime* time based mode) against the *SPICE* baseline in percent.

		SPICE vs.		
		DC	PT avg	PT time
Fig. 9	inv_tree, 65 nm, $\mu = 45$ ps	−4.5	−4.5	−4.6
Fig. 10	inv_tree, 15 nm, $\mu = 15$ ps	14.8	14.8	3.3
Fig. 12	inv_tree, 65 nm, $\mu = 100$ ps	−10.5	−10.5	−10.5
Fig. 13	mips_clock, 15 nm, $\mu = 15$ ps	23.0	23.1	22.6
Fig. 14	c17_slack, 65 nm, $\mu = 45$ ps	5.5	5.5	4.6
Fig. 15	c17_slack, 15 nm, $\mu = 15$ ps	16.9	18.3	7.6

analog simulation tools like *HSPICE* and *Spectre*, existing circuits can be easily simulated in the involution model and its performance compared to other prediction methods. Complemented by automatic waveform generation, parameter sweeping capabilities, and automatic report generation facilities, it allows the systematic experimental evaluation of different circuits in different model variants.

We demonstrated its capabilities by means of analyzing several circuits, in different technologies, using different involution channels. Whereas our experiments confirmed the superiority of the involution model in general, they also revealed two unexpected facts. First, it turned out that Hill-channels, which are based on more realistic switching waveforms, provide worse delay predictions than the simple Exp-channels for short pulses. Second, the considerably less superior predictions of the involution model for our NAND circuit suggest that the existing involution channels are not fully adequate for accurately modeling multi-input gates. Developing more accurate extensions of the involution model for multi-input gates is hence an important part of our current and future work.

CRediT authorship contribution statement

Daniel Öhlinger: Methodology, Software, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Jürgen Maier:** Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Visualization. **Matthias Függer:** Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing - review & editing. **Ulrich Schmid:** Conceptualization, Methodology, Formal analysis, Writing - original draft, Writing - review & editing, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] F. Najm, A survey of power estimation techniques in VLSI circuits, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 2 (4) (1994) 446–455.
[2] M. Favalli, L. Benini, Analysis of glitch power dissipation in CMOS ICs, in: Proceedings of the 1995 International Symposium on Low Power Design, in: Ser. ISLPED '95, ACM, New York, NY, USA, 1995, pp. 123–128.
[3] L.W. Nagel, D. Pederson, SPICE (Simulation Program with Integrated Circuit Emphasis), Tech. Rep. UCB/ERL M382, EECS Department, University of California, Berkeley, 1973.
[4] CCS Timing Library Characterization Guidelines, Synopsis Inc., 2016, version 3.4.
[5] Effective Current Source Model (ECSM) Timing and Power Specification, Cadence Design Systems, 2015, version 2.1.2.
[6] S.H. Unger, Asynchronous sequential switching circuits with unrestricted input changes, IEEE Trans. Comput. 20 (12) (1971) 1437–1444.
[7] M.J. Bellido-Díaz, J. Juan-Chico, A.J. Acosta, M. Valencia, J.L. Huertas, Logical modelling of delay degradation effect in static CMOS gates, IEE Proc., Circuits Devices Syst. 147 (2) (2000) 107–117.
[8] M.J. Bellido-Díaz, J. Juan-Chico, M. Valencia, Logic-Timing Simulation and the Degradation Delay Model, Imperial College Press, London, 2006.
[9] M. Függer, T. Nowak, U. Schmid, Unfaithful glitch propagation in existing binary circuit models, IEEE Trans. Comput. 65 (3) (2016) 964–978.
[10] M. Függer, R. Najvirt, T. Nowak, U. Schmid, Towards binary circuit models that faithfully capture physical solvability, in: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, in: Ser. DATE'15, EDA Consortium, San Jose, CA, USA, 2015, pp. 1455–1460.
[11] M. Függer, R. Najvirt, T. Nowak, U. Schmid, A faithful binary circuit model, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. (2019) 1.
[12] R. Najvirt, U. Schmid, M. Hofbauer, M. Függer, T. Nowak, K. Schweiger, Experimental validation of a faithful binary circuit model, in: Proceedings of the 25th Edition on Great Lakes Symposium on VLSI, in: Ser. GLSVLSI'15, ACM, New York, NY, USA, 2015, pp. 355–360.
[13] D. Öhlinger, J. Maier, M. Függer, U. Schmid, The involution tool for accurate digital timing and power analysis, in: 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2019.
[14] J.C. Ferreira, Physical synthesis with encounter (cadence), 2015/16, https://paginas.fe.up.pt/~jcf/ensino/disciplinas/mieec/pcvlsi/2015-16/tut_encounter/tut_encounter.html.
[15] A.V. Hill, The possible effects of the aggregation of the molecules of haemoglobin on its dissociation curves, J. Physiol. (London) 40 (1910) 4–7.
[16] N. Weste, D. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, fourth ed., Addison-Wesley Publishing Company, USA, 2010.
[17] A. Millan, J. Juan, M.J. Bellido, P. Ruiz-de Clavijo, D. Guerrero, Characterization of normal propagation delay for delay degradation model (DDM), in: Integrated Circuit Design, in: Ser. LNCS 2451, Springer, 2002, pp. 477–486.
[18] J.-C.B. Acosta, J. Juan-chico, M.J. Bellido, A.J. Acosta, A.B. riga, M. Valencia, S. Centro, N. Microelectrónica, Delay degradation effect in submicronic CMOS inverters, in: Proc. FODO'93, Springer, 1997, pp. 215–224.
[19] IEEE Standard for VITAL ASIC (Application Specific Integrated Circuit) Modeling Specification, IEEE Computer Society, 2001, IEEE Std 1076.4-2000.
[20] IEEE Standard Verilog Hardware Description Language, IEEE Computer Society, 2001, IEEE Std 1264-2001.
[21] D. Öhlinger, Involution tool, in: E191 - Institut für Computer Engineering, Tech. Rep. TUW-278633, Technische UniversitäT Wien, 2018, [Online]. Available: https://publik.tuwien.ac.at/files/publik_278633.pdf.
[22] M. Martins, J.M. Matos, R.P. Ribas, A. Reis, G. Schlinker, L. Rech, J. Michelsen, Open cell library in 15 nm freepdk technology, in: Proceedings of the 2015 Symposium on International Symposium on Physical Design, in: Ser. ISPD '15, ACM, New York, NY, USA, 2015, pp. 171–178.