IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Quantitative Characterization of Reconfigurable Transistor Logic Gates

MICHAEL RAITZA,[1] STEFFEN MÄRCKER,[1] Jens Trommer,[2] André Heinzig,[3] Sascha Klüppelholz,[4] Christel Baier,[4] and Akash Kumar[1]

[1]Chair for Processor Design, cfaed, Technische Universität Dresden, Dresden, Germany (e-mail: author.name@tu-dresden.de)
[2]Namlab gGmbH, Dresden, Germany (e-mail: author.name@namlab.com)
[3]Chair for Nanoelectronic Materials, Technische Universität Dresden, Dresden, Germany (e-mail: author.name@tu-dresden.de)
[4]Chair for Algebraic and Logic Foundations of Computer Science, Technische Universität Dresden, Dresden, Germany (e-mail: author.name@tu-dresden.de)

Corresponding author: Michael Raitza (e-mail: michael.raitza@tu-dresden.de).

The authors are supported by the DFG through the Excellence Initiative by the German Federal and State Governments (Cluster of Excellence cfaed).

ABSTRACT We present a new approach for early analysis of logic gates that is based on formal methods. As device technology research takes years and is very expensive, it is desirable to evaluate a technology's potential as early as possible, which is hard to do with current techniques. The actual impact of new devices on circuit design and their performance in complex circuits, are difficult to predict using simulation-based techniques. We propose a new approach that supplements simulation-based analysis and enables the development of standard cells alongside ongoing fundamental device research. Thereby, it potentially shortens the development cycle and time to market of a new technology. We develop a new discrete charge-transport model for electrical networks and a new flexible model of polarity-reconfigurable transistors as our formal basis. These models make circuit designs accessible to an analysis using probabilistic model checking and power our experiments. Besides worst-case analysis, we leverage measures hardly accessible to simulation such as average delay and average energy consumption per switching operation. We complement this with an automated design-space exploration that yields all reasonable implementations of a switching function built with reconfigurable transistors. After demonstrating the accuracy of our approach by comparison with finite element method analysis results, we undergo a comprehensive design-space exploration and analysis of the 3-minority function. The quantitative results are ranked with respect to various performance metrics, and we analyze the most promising circuit implementations in detail to derive a design guide that yields the best implementation for given statistics of the input patterns.

INDEX TERMS Circuit analysis, formal verification, nanoelectronics, probabilistic model checking, probability, quantitative analysis, reconfigurable logic, semiconductor device modeling

I. INTRODUCTION

DEVICE technologies with enhanced functionality or advantageous physical properties over established CMOS devices are in research with the goal to supplement or even replace current technologies. Such research includes reconfigurable transistors [1]–[5], graphene nano ribbon-based devices [6], [7] or mixed-dimensional heterostructure-based devices [8]. Alongside, alternative computation architectures move into focus, like asynchronous and approximate computing [9]. Meanwhile, conventional CMOS devices are no longer getting significantly smaller. The main concern in computational CMOS circuit designs has become their local power dissipation, requiring intricate dynamic voltage and frequency scaling schemes. While emerging technology devices promise to tackle these challenges, most of them are still in early development stages and it is not clear which of them are indeed viable candidates for integrated circuits.

Standard MOS devices have behaved largely the same for all technology generations, and the first decades in digital design were dominated by hand-designed circuits and standard cells. This design approach has been viable as no radical changes to the upper abstraction layers of logic gates, digital circuits and systems were to be expected. Thus, characterizing new devices beginning from technology readiness levels 4–5 did not result in a loss of time-to-market. With emerging devices, though, each transistor has enhanced functionality

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

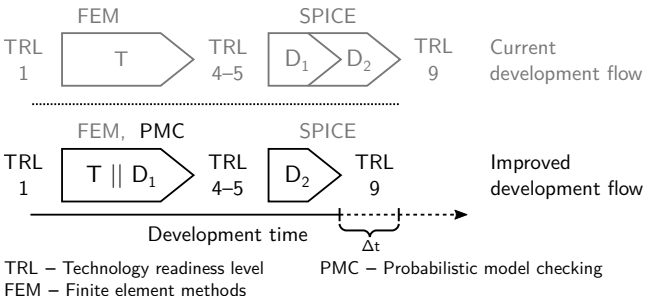


FIGURE 1. Current circuit development flow comprised of three consecutive steps, transistor device research (T), standard-cell design (D_1) and circuit design (D_2). Progressing from fundamental research (TRL 1), over technology development (TRL 4–5) to running production (TRL 9), our new approach features parallel device research and standard-cell design using probabilistic model checking (PMC), shortening overall development time.

that directly influences circuit design and characteristics. These influences should be investigated as early as possible as the insights might feedback on the development of the emerging device itself. For instance, if investigation shows that certain gate sizes or channel currents are in a bad relation to each other, device development focus can change early to address these identified issues. However, to evaluate a new technology node with state-of-the art modeling and analysis techniques, it has to reach *technology readiness level* (TRL) 4 or 5, known as technology development.

We propose an approach based on formal methods that allows us to start circuit modeling and performance analysis at TRL 1–2. Figure 1 shows the current development flow of a transistor technology alongside the improved development flow. Both start with fundamental device research at TRL 1–2 and mostly rely on the finite element method (FEM) [10], [11] to simulate the device and obtain insights into its basic properties. Currently, the design of standard cells (D_1) starts after reaching TRL 4–5, i. e., technology development. Our device models are simple enough such that early evaluation can start long before reaching TRL 4–5. As we demonstrate in this work, the toolset relies on very little device data that might come from FEM analysis or early measurements. In contrast to simulation methods like SPICE [12], we do not rely on detailed *compact* or *table models* which are hardly available at earlier TRLs. Our approach improves on the current development flow by parallelizing the design of standard cells (D_1) with the device research. By doing so, the overall development cycle and hence the time to market is potentially shortened.

This becomes possible, since we rely on a simple discrete charge-transport model for the physical network and a new flexible model of transistors. The abstract network and transistor models facilitate the application of *probabilistic model checking*, which provides radically improved capabilities that are infeasible in simulative approaches. For instance, extremal values, like the maximum power dissipation, can be computed directly. Expected values of metrics under stochastic input patterns that model application statistics, e. g., the average delay or the average energy consumption per switching operation, are computable as well. Additionally, our approach is particularly suitable to systematically exploit functional enhancements of new transistor devices. Here, we focus on *reconfigurable transistors* as an example technology, although the proposed quantitative characterization method is applicable to other emerging device technologies as well. The method's capabilities enable an *automatic design-space exploration* of the topology of a standard cell and quantification over multiple measures such as delay, power dissipation and energy consumption per operation.

Outline and Contribution

After discussing related methods from device research, circuit design, and formal methods in Section II, we cover our contributions in the following three sections: Section III introduces the models for *charge-transfer* and *reconfigurable transistors*. We demonstrate the accuracy of our approach and relate it to FEM and SPICE. Section IV explains the method and workflow in detail, how *circuits under test* are combined with *input patterns* and a *query* to form an experiment. Section V illustrates our method on a design space exploration of the 3-MIN logic gate over *power*, *energy* and *delay* measures. The developed tools, all models, and raw experimental data are available here:

`https://cfaed.tu-dresden.de/pd-downloads`

II. RELATED WORK

We target the design of standard-cell libraries for emerging devices with enhanced functionality. To showcase our method, we have chosen the Germanium-based reconfigurable field effect transistor (RFET) from [13] as an exemplary technology. *Transistor reconfiguration* switches a transistor between P- and N-type carrier transport dynamically [1]–[4], [13]. Additionally, *multiple gates* are used to intrinsically support the AND-functionality, as demonstrated for a similar technology based on Silicon [5], [14]. Reconfigurable devices are promising for applications in hardware security, where they can be applied to camouflage the circuit functionality [15], [16]. Furthermore, recent studies have shown that reconfigurable transistors, in particular devices with multiple independent gates, improve signal integrity in dynamic logic gates [17], [18]. They also have potential to foster non-traditional digital design styles, such as asynchronous and reversible logic [19]. The chosen Germanium-nanowire based RFETs are an example of lab technologies in early development phases for which just enough data is published to analyze first circuit models with our approach. Note, that in this early TRL stage, questions regarding physical layout and area cannot be answered seriously, yet. Though, estimations from [20]–[22] indicate that, depending on the individual benchmark circuit, reconfigurable transistor area requirements range from being competitive (17 % overhead) to even surpassing (41 % benefit) CMOS designs. For more insights into the physics and constraints of the reconfigurable technology, the reader is referred to the cited literature.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

We base our approach on a formally analyzable circuit and transistor abstraction, which mainly describes the charge transport. SPICE and FEM usually use differential equation models but charge-transport models have been shown in SPICE as well, see [23]. Our analysis approach contrasts to simulation-based approaches in that we can directly compute extremes and averages of various measures precisely, e. g., delay and energy.

Formal methods, such as theorem proving and model checking, have been proven to be valuable tools for hardware verification [24], computer-aided design in logic synthesis [25], and software verification [26]. Both methods have been successfully applied to verify the correctness of nontrivial circuits. *Theorem proving* [27], [28] was applied for the verification of complex circuits like floating-point units [29]. *Model checking* [30], [31], an algorithmic method to check transitions systems against temporal-logic specifications, is the dominant method in formal circuit verification of CMOS circuits, e. g., with respect to sequential specifications and functional correctness [32]. It is also used to analyze more complex circuits, e. g., a PowerPC CPU [33]. To integrate formal verification into the development flow, several approaches exist to automatically extract and verify formal models of circuits from hardware description languages such as VHDL [34] or Verilog [35]. The standard approach for timing analysis usually relies on gate-level simulation and static timing analysis [36]. *Timed automata* [37], [38] provide a formal approach to timing analysis that proved to be effective for complex circuits, e. g., a 4-Bit adder [39]. Timing analysis typically requires the circuit delay of standard cells to be known, e. g., from an Elmore delay model [40].

The formal approach we propose here, enables us to determine the circuit delay and other metrics at the transistor level using *probabilistic model checking* (PMC) [41], [42], an extension to model checking that incorporates stochastic transitions into a model which occur naturally, e. g., from input pattern statistics or production variations. Then, the analysis provides the extremal values, e. g., maximal circuit delay, as well as expectation of metrics under an input pattern. Although this work is the first application of PMC for quantitative analysis of performance characteristics of post-CMOS integrated circuits at the transistor level, the method has been successfully used in circuit analysis. In [43], the authors use PMC to verify several complex RTL designs against their timing specification, e. g., a H.264 decoder module. In the field of reliability analysis, PMC has been successfully applied to reason about NAND multiplexing [44]. For safety critical applications of SRAM-based FPGAs in the aerospace industry, PMC has been applied in [45] to analyze dependability in the presence of transient errors, e. g., cosmic radiation. In the scope of emerging materials, initial work on the development of fault models for testing [46] and the analysis of reliability tradeoffs [47] highlights the potential of formal methods. Also, BDD-based methods for the synthesis of XOR-circuits based on reconfigurable SiNW-transistors have been proposed in [48].

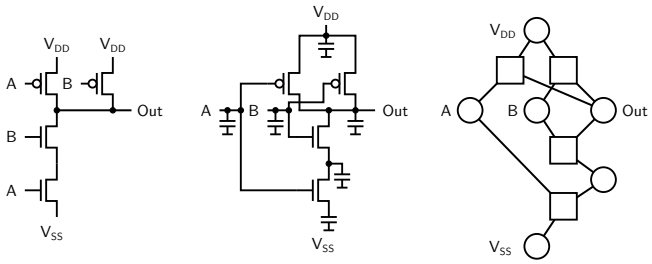


FIGURE 2. Conversion of a circuit diagram into the bipartite graph of our proposed charge transport network model. Devices are separated into ideal transistors (□) without capacitance and all capacitative behavior is collected into virtual capacitances (○) at the network connections.

III. PHYSICAL NETWORK AND TRANSISTOR MODEL

A. NETWORK MODEL

Our network model uses charge transport equations and describes a physical model that consists of *charge storages* and *charge transmitters*. Each electronic circuit is modeled as a bipartite network of these two fundamental elements, i. e., every node of the first type is connected to nodes of the second type only and vice versa. An example is shown in Figure 2. A well-known 2-NAND circuit is transcribed from its circuit diagram into its bipartite graph by adding virtual capacitances at the terminals, i. e., the drain and source contacts, and the gate of a transistor.

1) Charge storage nodes

These nodes represent the only points in the network that have a defined voltage potential. They are modeled such that all incoming and outgoing currents add up to zero. Each charge-storage node is tied to a certain capacitance which is charged and discharged with the current required to make the sum of all momentary currents at that node zero (Kirchhoff's current law). Hence, they can be regarded as ideal capacitances. Figure 2 shows them as circular nodes. We distinguish three types of charge storage nodes to allow flexible experimental setups. Passive nodes react to the attached contacts according to Kirchhoff's law (Out), constant nodes have a fixed voltage level to model power sources (V_{DD}, V_{SS}), and active nodes provide stimuli to the network by sweeping the voltage potential across the full range with a defined slope (A, B).

2) Charge transmitters

These nodes have complementary properties to charge storages. They are the only points in the network that have a defined current. By design, charge transmitters have at least two predefined terminals, each of which is connected to a charge storage node of the network. According to their specific model parameters and functional dependencies, the momentary current at the node is a result of its momentary resistance and the voltage differences between the attached charge storages. They can be seen as ideal resistors and are depicted as square nodes in Figure 2.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

3) Operational behavior

The electrical network's evolution over time arises entirely from the interaction of the charge storage and charge transmitter nodes. Neither capacitance nor resistance need to be constant properties of a device, which is crucial to model dynamic devices like transistors. The network model takes the momentary currents into account, as well as the momentary capacitances. Care must be taken when modeling new node subtypes, e. g., memristors, where both properties dependent on voltage differences to avoid discontinuities and jumps that do not occur in reality. The charge transport equations imply a sequence of (dis)charging processes according to the stimuli and the network topology, and starts with an initial assignment of concrete voltage potentials at the charge nodes.

To facilitate exhaustive analyses of the electrical networks, the model needs to have finitely many states. For this purpose, we treat voltage levels at charge storages and time as discrete values. Hence, the network's global state is fully determined by the voltage levels at the charge storages. This requires choosing minimal distinguishable voltage difference V_{scale} and a minimum time interval T_{scale}, which are used when evaluating the network. Within T_{scale}, we assume currents and voltage levels to be constant. Based on these values, we compute updated voltages in the charge storages that become visible to the attached charge transmitters in the next time step.

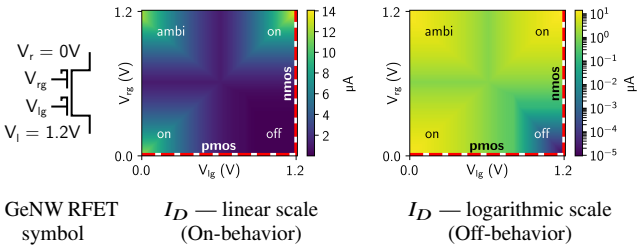
Neither the network model nor the device model make any assumptions about geometric properties of the circuit under test. However, the network model is generic enough to capture geometric influences, e. g., on signal propagation delays, using resistive elements (charge transmitters) and capacitive elements (charge storages). As we target early technology evaluation in this work, we focus on structural evaluation of circuit variants without taking these influences into account. At this TRL, questions of physical standard-cell layouts do not play a major role and any assumptions on the layouts, area requirements and implied capacitances would be highly speculative. But as soon as sophisticated PDKs are available, a detailed area analysis can be carried out based on the results of our method.

B. TRANSISTOR MODEL

The network model provides the necessary degrees of freedom to create almost arbitrary discrete time transistor models. These models characterize a transistor by functions in voltage, current, and capacitance at its contacts as well as finitely many internal states.

In this work, we focus on Schottky barrier-based reconfigurable transistor (SB RFET) technology [1], [13], [49]. This technology has interesting properties:

1) Transistors can be dynamically reconfigured between PMOS and NMOS characteristics. The polarity is controlled by a polarity control gate (PCG) which facilitates circuits to share transistors between P- and N-branches.
2) Transistors feature multiple independent gates per device (MIGFET), i. e., two SB gates and up to three inner gates.



GeNW RFET symbol | I_D — linear scale (On-behavior) | I_D — logarithmic scale (Off-behavior)

FIGURE 3. Heatmaps display the field of channel currents I_D, for the four *operation states* of the depicted GeNW RFET setup based on the device in [13]. Dashed lines in the I_D heat maps correspond to single curves in common I–D diagrams for CMOS. V_{lg} = 1.2 V configures NMOS and V_{rg} controls the channel (vertical). V_{rg} = 0 V configures PMOS and V_{lg} controls the channel (horizontal). The *ambipolar* state is assumed towards the upper left. Refer to [4] for a more detailed description.

Together they realize a wired-AND functionality.

Transistor reconfiguration is a standing term in the literature for the kind of device we use and model in this work. It must not be confused with *functional reconfiguration* known from programmable logic structures like FPGAs and PLAs, which have also been implemented using silicon nanowires [50]. For Boolean functions, both concepts have in common that a *reconfigurable circuit* is switched from performing a function α to performing another function β, but they differ how reconfiguration is implemented. Functional reconfiguration combines *independent* functional components into a larger circuit that can select between the functionality of its subcomponents using additional reconfiguration circuitry. In contrast, transistor reconfiguration is able to efficiently represent *self-dual* Boolean functions [51]. A self-dual function, such as the 3-XOR and the 3-MIN gate, yields the inverse output for an inverted set of inputs. The same transistors compute either of the two inverse sub-ordinate functions of a (partially) *self-dual* Boolean function, see also [52]. The only reconfiguration circuitry are additional inverters, as signals that reconfigure the circuit are required in direct and inverted form for proper polarity configuration.

The technology is in an early state of development, yet preliminary performance data and projections to competitive node sizes down to 24 nm gate lengths are available [4], [13]. Nevertheless, our method is also capable of modeling other emerging nanoelectronic devices with special properties like steep subthreshold slope tunnel FET's [53] or anti-ambipolar devices from stacked layers of graphene and MoS_2 [8].

1) GeNW MIGFET

The devices presented in [13] are representatives of a reconfigurable MIGFET technology at a gate length of 24 nm. Figure 3 illustrates the principal functionality using a two gate RFET. We name the contacts after their positions (l, lg, rg, r) as their functional rôles (src, drn, PCG and ctrl) change according to the attached voltages. The transistor's channel contacts are connected to 1.2 V and 0 V as depicted on the left. The SB gate voltages V_{lg} and V_{rg} are adjustable and mapped to the x-axis and y-axis of the heat maps. Given the fixed channel

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

TABLE 1. GeNW MIGFET model parameters [13]. Gate length: 24 nm, Channel length: $(48 + 40(n - 2))$ nm for $n \geq 2$ gates

	I_{D0} PMOS	I_{D0} NMOS	V_{th}	C_{ctrl}
SB gates ("PCG", "lg", "rg")	400 nA	120 nA	0.4 V	40 aF
Inner gates ("m_i")	12 nA	31 nA	0.2 V	40 aF
	I_D PMOS	I_D NMOS	V_{DD}	C_{chan}
Open channel	12 050 nA	14 060 nA	1.2 V	40 aF
Closed channel	50 pA	50 pA	1.2 V	40 aF

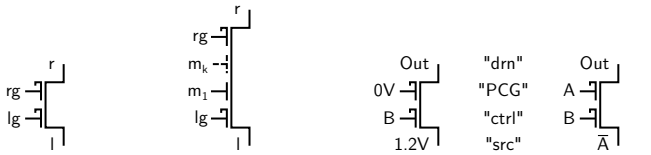
contact potentials, which gate assumes the role of the PCG depends on the voltage regime between these gates and the channel contacts. If $V_{lg} = 1.2$ V then *lg* is the PCG, i. e., the transistor is in NMOS configuration and *rg* steers it open or closed (Figure 3, vertical dashed lines). If $V_{rg} = 0$ V then *rg* is the PCG with *lg* opening and closing the transistor in PMOS mode (Figure 3, horizontal dashed lines). As these devices are completely symmetric, they exhibit the dual behavior if the channel contact voltages are swapped.

The heat maps visualize the currents I_D for all combinations of V_{lg} and V_{rg}. The linear scale heat map (left) highlights the on-behavior whereas the logarithmic heat map (right) zooms into the off-behavior. As long as either gate assumes the role of the PCG, the transistor is in a controlled on-state (NMOS or PMOS) or off. The device shows high currents in both controlled on-states as well as in a third, misconfigured ambipolar state (linear map, top left corner). This ambipolar behavior is different from standard MOSFET devices and must be taken into account when building pass-gate logic. As it only occurs when both SB gates are connected to the opposite voltage of their adjacent contact, it can be suppressed by connecting both SB gates to the same signal. If the reconfigurable device features (multiple) inner gates, the maximum current is always limited by the transmissibility of the controlling SB gate. The heat maps are the results of instantiating our transistor model with the basic device parameters extracted from FEM data and shown in Table 1. The depicted currents agree with the published data [13].

2) Dynamic transistor behavior

Figure 4 schematically depicts (1) a reconfigurable transistor (RFET) with two gates and (2) a multiple independent gate transistor (MIGFET) with k inner gates. The functional roles of the terminals, i. e., source, drain, control, and polarity-control, are determined by the actual voltages at the terminals. Subsequently, we denote the channel contacts l and r, and the gates from left to right lg, $m_1, \ldots, m_k$, and rg. Figure 4 shows two example RFET configurations where (3) puts the RFET in a static PMOS configuration, and (4) makes it dynamically reconfigurable, as the PCG and the source terminals are attached to an input signal A and its inverse $\overline{A}$, respectively.

We realize our generic reconfigurable MIGFET model by adapting the standard set of equations describing the transient behavior of MOSFET's as they are straightforward to integrate



(1) RFET: two SB gates. (2) MIGFET: two SB and k inner gates. (3) RFET configured to PMOS. (4) RFET reconfigurable in A.

FIGURE 4. Schematics of SB reconfigurable transistors, as in [13]. (1) depicts a two-gate reconfigurable transistor (RFET) and (2) a multiple independent gate transistor (MIGFET). Two possible configurations are shown on the right. Connecting PCG to 0 V and *src* to 1.2 V yields a fixed PMOS configuration (3). Connecting PCG to signal A and *src* to its inverse $\overline{A}$ (4) reconfigures the RFET between PMOS and NMOS via A.

in our discrete time and voltage modeling approach. For simplicity, we do not distinguish between PMOS and NMOS here, as the equations are similar for both polarities. We implemented the set of MOSFET equations as follows:

$$I_D = \begin{cases} I_{D0} + \frac{\beta}{2}\,(V_{GS} - V_{th})^2 & \text{Saturation} \\ I_{D0} + \beta\left((V_{GS} - V_{th})\,V_{DS} - \frac{{V_{DS}}^2}{2}\right) & \text{Linear} \\ I_{D0}\; e^{\frac{V_{GS} - V_{th}}{SS \cdot V_T}} & \text{Subthreshold} \end{cases}$$

These equations differ slightly from the literature. Here, the linear and the saturation mode are fitted to meet in (V_{th}, I_{D0}), where I_{D0} denotes the drain current when the gate voltage reaches the threshold voltage $V_{GS} = V_{th}$. This avoids discontinuities, which otherwise occur close to the threshold voltage with $I_D = 0$ A, i. e., neither mode gives meaningful results for the threshold voltage. The transistor-specific conductance parameter β can be used to parametrize the equations according to the characteristics of an actual device. Although it is an approximation, Section III-D shows that the model fits well with data from FEM and SPICE simulations and creates precise predictions while having the benefit of avoiding discontinuities around V_{th}.

3) Channel polarity control

As shown in Figure 4, we focus on the example of emerging Schottky barrier-based polarity-controllable transistors, which feature a second or more gates at a single transistor. As polarity-controllable transistors do not necessarily possess a fixed polarity-control gate, we describe the transistor's operation mode purely in terms of the voltages at the left channel contact l and left Schottky gate lg, and their right-hand side counterparts r and rg, cf. Figure 4. Then, the operation modes are:

PMOS: Open channel for h^+ charge carriers
if $V_l > V_{rg}$ and $V_l > V_r$ or
if $V_r > V_{lg}$ and $V_l < V_r$,

NMOS: Open channel for e^- charge carriers
if $V_l < V_{rg}$ and $V_l < V_r$ or
if $V_r < V_{lg}$ and $V_l > V_r$,

ambipolar: Open channel for both charge carriers
if $V_l > V_{lg}$ and $V_{rg} > V_r$ and $V_l > V_r$ or
if $V_l < V_{lg}$ and $V_{rg} < V_r$ and $V_l < V_r$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

4) Ambipolar behavior

One important conclusion from the set of equations is, that both gates may open the channel for the opposite type of charge carriers at the same time, e. g., in the voltage regime $V_l = V_{rg} = V_{DD}$ and $V_r = V_{lg} = 0\,\mathrm{V}$. Neither gate properly configures the channel to a particular charge carrier type and the device is in an *ambipolar* state. As this state is not of immediate interest for CMOS circuits, and as it is not yet sufficiently quantified in the literature, we implement ambipolar behavior as I_D following the NMOS and PMOS I_D curves from saturation down to I_{D0} and back up to saturation again.

This can be seen in the heat maps in Figure 4 when tracing the current along the top edge and the left edge, respectively. Our current calculation of I_D in the ambipolar case must be seen as an upper bound. Once experimental data becomes available, we will specify the ambipolar behavior in our equations more precisely.

5) Drain current computation

In a unipolar NMOS/PMOS voltage regime, the SB gate opposite the polarity-control gate, and all inner gates $m_1, \ldots, m_k$ in between, act as control gates that steer the channel open or close. Using the MOSFET Equations and the reconfigurable transistor's operations modes, we introduce the following equations to calculate I_D:

$$I_D = \max\left(I_{l\to r}, I_{l\leftarrow r}, I_{ambi}\right)$$

$$I_{l\to r} = \begin{cases} \min\left(I_{D,lg}, I_{D,m_1}, \ldots, I_{D,m_k}\right) & rg \text{ is } PCG \\ 0\,\mathrm{A} & \text{otherwise} \end{cases}$$

$$I_{l\leftarrow r} = \begin{cases} \min\left(I_{D,m_1}, \ldots, I_{D,m_k}, I_{D,rg}\right) & lg \text{ is } PCG \\ 0\,\mathrm{A} & \text{otherwise} \end{cases}$$

$$I_{ambi} = \begin{cases} \min\left(I_{D,lg}, I_{D,rg}\right) & \text{neither } lg \text{ nor } rg \text{ are } PCG \\ 0\,\mathrm{A} & \text{otherwise} \end{cases}$$

For each direction ($I_{l\to r}$ or $I_{l\leftarrow r}$), the minimum drain current I_D over all control gates is computed according to the standard transistor equations.

As a special case, I_{ambi} is calculated as the minimum directional drain current in the ambipolar voltage regime. This overestimates the ambipolar current but provides an upper bound that matches the still incomplete experimental data for this mode best.

Finally, the maximum of the directional and ambipolar currents is the output drain current. For lack of experimental data for I_{off} over V_{DS}, we assume $I_{off} = 0\,\mathrm{A}$ for all measurements, neglecting I_D for the *closed channel* from Table 1.

For this work, we assume the channel parasitics C_{chan} and gate capacitances C_{gate} to be constant for all voltage regimes and transients. This simplifies the model further and avoids estimating a function over varying capacitance that can hardly be backed up by actual experimentation. Evaluation will show that the model still matches well in the intended usage scenarios.

In summary, we provide a transistor model that is:

- Analytical and stateless,
- Fast to compute, avoiding differential equations,
- Simple enough, to not require much experimental data or a high technology readiness level,
- Parametric, to allow adaptation of individual devices, or to adapt to new technology nodes or projections,
- Accessible to an analysis using formal methods, and
- Precise enough to model digital circuits.

C. COMPARISON TO SIMULATION METHODS

Established simulation techniques, like SPICE and FEM, rely on stateful models of single elements, e. g., transistors, and characterize the state evolution over time by differential equations, e. g., see [54]. All elements are connected within a network, forming a single big set of equations that describes the global state evolution of the network. A principle benefit of how these approaches are usually implemented is, that the model resolution is bounded only by the amount of history (i. e., state) that can be handled practically. The biggest drawback is that this model cannot be exhaustively explored but must be evaluated piecewise. Also, variable model resolution often causes convergence problems. This implies the necessity to choose a reasonable starting parameters and to estimate after how many simulation steps a reasonable result is achieved. Usually, this forces the designer to iteratively explore both, the input domain and the output range, to achieve meaningful results.

The approach we propose here handles modeling in a very different way. Each network node is a piecewise abstraction of its transient characteristics which depends only on the adjacent nodes and the node's own internal state, e. g., the voltage level at a charge storage. This simplifies the transfer functions significantly as the direct influence of a single node is localized to the immediately attached network nodes. The networks global behavior arises from the local interactions of the network nodes instead of a global transfer function. As a consequence, convergence problems as for the aforementioned simulation methods do not occur. However, because the network model is discrete in voltage levels and time, we have to choose suitable discretization parameters V_{scale} and T_{scale} for a meaningful analysis, cf. Section III-A.

The immediate benefit of the proposed modeling approach is, that all possible network states can be explored which makes a network's behavior accessible to a formal analysis. Then, probabilistic model checking allows us to compute measures hardly accessible in simulation methods, e. g., average delays in the presence of stochastically modeled inputs, the energy of single switching operations, or optimization of multiple goal functions, see also [55], [56]. Generally, unlike simulation-based approaches, PMC yields the measured value and the offending input pattern(s) directly without requiring user knowledge about suitable simulation lengths or suspicious input patterns.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

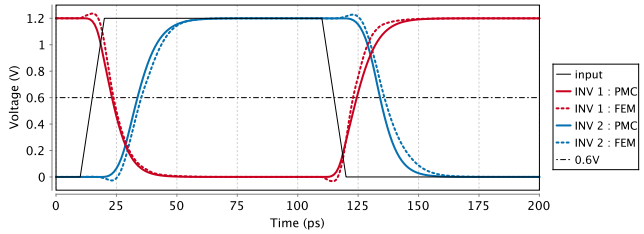
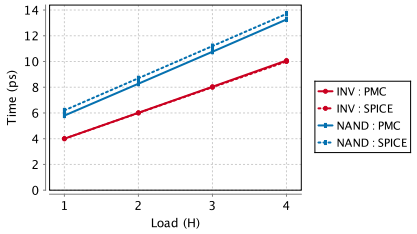


FIGURE 5. Fitting of time response of inverters built with the GeNW MIGFET, controlled via SB gate. Input slope 10 ps × $V_{DD}{}^{-1}$, output load $H = 1$ ($C_{\text{out}} = 80$ aF), calculated delays $\Delta t_{\text{PMC}} \approx 10$ ps, $\Delta t_{\text{FEM}} \approx 12$ ps.



$I_{on} = 109\,\mu\text{A}$
$I_{D0} = 0.8\,\mu\text{A}$
$V_{DD} = 1.0\,\text{V}$
$V_{th} = 0.3\,\text{V}$
$C_{gate} \approx 127\,\text{aF}$
$C_{chan} \approx 127\,\text{aF}$
$1\,\text{H} = 2 \times C_{gate} \approx 254\,\text{aF}$

FIGURE 6. Comparison of 1/2 buffer delays depending on load factor H between a SPICE simulation and the proposed PMC-based method for a CMOS 32 nm process [57]. The PMC model was fitted to the SPICE-results for the inverter and, then, validated with a comparison of two NAND circuits.

D. MODEL ACCURACY COMPARED TO SPICE AND FEM

To assess the accuracy of our modeling approach, we will show that the charge transport model and the transistor model—though abstract—are sufficiently detailed to yield predictions that are close to the results obtained from simulations. First, we instantiate an experiment with the parameters of the GeNW RFET device presented in [13] and described in Section III-B. We compare our PMC-based predictions of the circuit delay to the results of an FEM simulation of the equivalent experiment. Although our method targets at emerging technologies at low TRL levels, it is important to show, that our approach also delivers correct results for state-of-the-art CMOS processes. Thus, we second instantiate the transistor model with the process parameters of an established CMOS process from [57]. In this experiment, we compare the circuit delays for a concrete circuit calculated with PMC to a SPICE simulation using the same technology node.

1) 24 nm GeNW MIGFET transistor

The basic parameters of the GeNW RFET transistor at 24 nm gate and 48 nm channel lengths, were retrieved from FEM simulations and are shown in Table 1. Mixed-mode circuit simulations have been carried out as reference, as no suitable table nor compact models exist, yet.

The reference circuit for this comparison is a buffer, for which we computed the delay between the outputs of the first and the second inverter and a load of $C_{\text{out}} = 80$ aF. Figure 5 shows the input and output transients in a V–t plot as well as a horizontal line at 0.6 V, the threshold used for the circuit delay. Using the estimated gate capacitance extracted from the FEM simulation of ≈ 40 aF, we already achieve a close fit.

2) 32 nm CMOS transistor

We used the device from [57] and parametrized the transistor model from Section III-B with I_{on} and I_{D0} as well as V_{DD} and V_{th} and adjusted β to match the published device transients. However, the gate capacitances C_{gate} and channel parasitics C_{chan} of the SPICE reference model are neither publicly available nor directly measurable. Thus, we estimated their values as follows: We simulated a reference circuit in SPICE and adjusted C_{gate} and C_{chan} of our abstract transistor model such that the circuit delays predicted by PMC agreed with the SPICE simulation for various load factors. Figure 6 lists the derived set of parameters we used to model the 32 nm CMOS transistor. As reference circuit, we used a chain of three inverters and determined the 1/2 *buffer delay*, i. e., the average of the delays of the first and the second inverter. This experiment setup compensates imbalances in PMOS and NMOS drive characteristics and make the results comparable to measurements using a single inverter delay. Figure 6 (red curves) shows that we succeeded to parametrize our transistor model to achieve a perfect fit of the PMC and SPICE-based delays for the INV-chain reference circuit that stays congruent with increasing output load.

To confirm that the transistor model is not only fitted to the INV-chain but delivers predictions close to SPICE for other circuits as well, we carried out a similar experiment with a chain of three NAND gates, using the capacitance values obtained from the INV simulations. The NAND gates were set to be fixed to V_{DD} on input A and sensitive input B. Again, we computed the 1/2 buffer delay of the chain and compared the results from PMC to SPICE in Figure 6 (blue curves). The results support our claim that the PMC-based analysis of the formal model predicts the behavior of a classical circuit with sufficient accuracy: First, the slopes match precisely (2.5 ps/H) and second, the curves are very close to each other. We attribute the remaining difference to the fact that the transistor and network model is comparatively abstract and parameterized with rough estimates for the capacitances, since the precise values in the SPICE simulation of that CMOS technology are not directly accessible to us.

IV. WORKFLOW OF OUR PROPOSED METHOD

Figure 7 gives an overview of the workflow that a designer would follow in order to analyze a circuit using our proposed approach. It involves three conceptual steps, of which only the first one on the left requires user input whereas all others are fully automatic. These are, from left to right: the experiment setup, the construction of a physical model and its mapping to the model-checker's input language, and the push-button quantitative analysis using PMC. To facilitate this workflow, we developed the tool *prism-gen*. It provides an extensible circuit-description language that it translates according to the charge-transfer model into the modeling language of the probabilistic model checker Prism [58].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

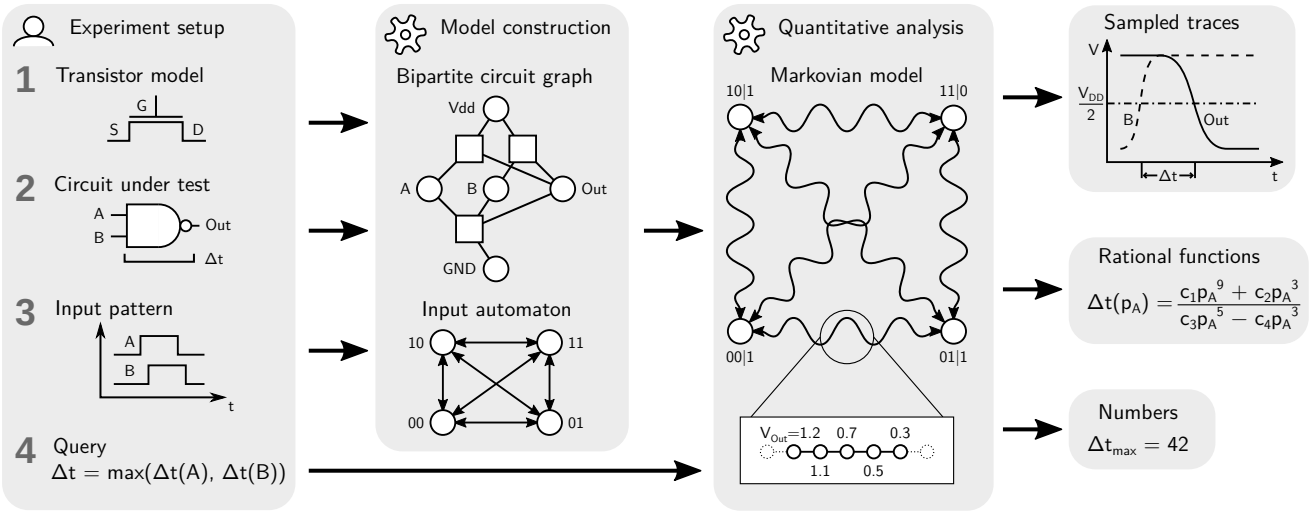


FIGURE 7. Workflow of our proposed method. The designer provides three artifacts to prism-gen, the tool that formalizes the experiment; a transistor model, a circuit under test and an input pattern. The fourth artifact, the query, is a formalization of the actual "measurement" to be performed (e. g., worst-case delay). Quantitative analysis computes on a generated model delivering all possible results of the query. These can be numeric, rational functions or sampled traces.

A. EXPERIMENT SETUP

To evaluate a specific circuit, the designer provides four artifacts as shown in Figure 7. The first three, a *transistor model* (or multiple) from which the circuit shall be implemented, the specification of the *circuit under test* itself, and the specification of an *input pattern*, define how the experiment is set up. The artifacts interface via the transistor terminals and the inputs and outputs of the circuit under test. This design facilitates reusing existing components to derive new experiments. For example, all experiments in this paper require only two different input pattern specifications. It is possible to parametrize any of the components, e. g., transistor's beta ratios, gate capacitances, output loads or correlations between two inputs. This allows the designer to adapt them to the specific conditions of the experiment by reusing the composable units. The fourth artifact, a *query* (or multiple), defines a goal function that shall be computed for a given model. It is specified in the input language of the model checker but refers to variables from the model by a common naming convention.

1) Transistor model

Transistor models are built using the MOSFET equations as described in Section III-B and usually stored in a library. When a transistor device is instantiated in a circuit, each instance can be customized via parameters such as the capacitance at its terminals. Device libraries are written in prism-gen language, too. This allows a designer to create a new library for an emerging device on his own, as long as the electrical behavior can be expressed as a (piecewise) function of an electrical current over voltages at the contacts and an optional internal state of the device, cf. Section III-A.

2) Circuit under test

The circuit under test is specified in prism-gen's netlist-like input language. The network is automatically transformed to a bipartite circuit graph as shown in Figure 7. The two basic elements of a circuit graph, charge transport and charge storage nodes, are drawn as squares and circles. Transistors are instantiated as charge transport nodes since they are dynamic resistors. Their terminal capacitances are added to the corresponding charge storage nodes. The prism-gen language allows abstracting circuit topologies into a library of circuit blueprints. When instantiated to compose a more complex circuit, each sub-circuit can be parametrized with the transistor technology, e. g., the transistors of a generic inverter can be replaced by scaled variants in a concrete circuit.

3) Input pattern

An input pattern defines the sequence of stimuli over time of the experiment. It is special in that it may appear only once in a circuit. Since it is likely to reference most of the underlying circuit components, the input pattern is normally defined in the top-level component. Nevertheless, input patterns can be equally kept in a separate module and instantiated at the top level to provide reusable input patterns that can be shared between various experiments. For instance, each scenario in our experiments in Section V-C uses a single input pattern description.

An input pattern defines *how* and *when* the charge storage nodes of the experiment's inputs should change. Since time is modeled as discrete steps, finite automata can be used to specify the input pattern. Alternatively, the designer can use a step counter to model the chronology of input changes straightforwardly.

Depending on the goals of the quantitative analysis, an input pattern belongs to one of three basic types or any combination of them. First, a pattern may consist of a *fixed sequence of*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

change events at predefined points in time, optionally put into a loop. This resembles a very specific situation, e. g., the transient behavior in a V–t plot as depicted in Figures 5 and 7. The change events can be triggered after a predefined number of steps, as in simulation or also, if a condition on the model state is met, like reaching a voltage threshold at the output or the convergence of currents.

Second, a pattern does not have to specify how and when the inputs change but can provide a *set of possible choices* instead. This allows every possible input sequence to occur and hence, the analysis will cover all of them, e. g., worst-case delay analysis exploits this type of pattern. A complete coverage like this is usually hard to accomplish using testing and its (necessarily incomplete) set of stimuli.

Third, a pattern can integrate statistical knowledge on the likelihood of specific input changes by means of a *probabilistic choice*. This implies a probability measure on the possible sequences. Consequently, we can treat metrics, e. g., the delay, as random variables and reason about their expectations, long-run averages and other statistical measures.

4) Query

A query formalizes the goal-function to be computed in an experiment, i. e., a performance or quality measure such as circuit delay or functional correctness. Therefore, it provides a generic characterization of the set of traces that have to be taken into account. During the analysis, PMC can reason about the (possibly infinite) *set of all traces meeting the criteria* without generating them explicitly. This is in stark contrast to simulation-based approaches and measurements, which have to enumerate the relevant traces explicitly.

The formalization requires the designer to specify relevant *model states* and to impose constraints on the *temporal sequence* of those states in a measurement. Logic propositions over the model variables, such as effective voltage levels or currents, suffice as filters. To specify temporal sequences, we rely on *temporal logics* (LTL, CTL [59]–[61]) which offer suitable temporal operators like "*next* state" and "*eventually* some state". We restrain ourselves to the input-output delay in this work, but temporal logics enable us to reason about many more complex events. For instance, we have already characterized the occurrence of hazards and output oscillation.

As an example, the input-output delay can be defined as the delay between those states where the rising input transient B crosses $1/2V_{DD}$ and those states where, say, a declining output Out crosses $1/2V_{DD}$ as an effect. The V–t plot in Figure 7 (right) depicts an example trace meeting this criterion. More formally, we are interested in the paths that

1) *start* in a state where $V_B < 1/2V_{DD}$ and $V_{Out} > 1/2V_{DD}$,
2) *next* transition to a state where $V_B \geq 1/2V_{DD}$ and
3) *eventually* end once in a state where $V_{Out} \leq 1/2V_{DD}$.

The delay is the number of model steps of a path matching this specification and the worst-case delay is the maximum over all those paths. If the input pattern is probabilistic, we can treat the delay as random variable and may ask for the expectation or the probability that it stays below a certain threshold. Apart from this example, the measure can be any (positive) cost function over states and transitions in the model, e. g., to reason about power and energy.

Moreover, the temporal constraints in a query can also be used to pick very specific input sequences while keeping the actual input pattern of the model generic and underspecified. Our queries used in Section V express a general measurement, independently of a specific circuit or of transistor device parameters, i. e., the delay of all 3-input circuits.

B. MODEL CONSTRUCTION

prism-gen translates the aforementioned artifacts *transistor*, *circuit* and *input specification* into a description of a *discrete time Markov chain* or a *Markov decision process*. These formal model types cover the whole range of input patterns and are well supported by model checking tools. prism-gen derives the bipartite graph from the *circuit specification*, which yields the topology, the input drive strength and the output load(s). The transistor nodes are parametrized according to the *transistor specification*, which provides the necessary equations as well as the terminal capacitances. The *input specification* is translated into an automaton which controls the voltage potential of the input charge storage nodes. This internal representation of the experiment is then mapped to the Prism modeling language, yielding the *experiment formalization*. Please note, that the use of *Markovian models* enables us to verify the functional correctness of circuit designs without additional costs, as well. Although, that's not in focus here, as the circuits in the experiment section are correct by construction.

C. QUANTITATIVE ANALYSIS

The final step is performed by a probabilistic model checker. It takes the *experiment formalization* and the *query formalization* as inputs and computes the requested performance or quality measure automatically without further user interaction. We use a Prism version tailored to our needs, here, but alternative tools with different feature sets are available, such as Storm [62] or The Modest Toolset [63].

First, the model checker expands the *experiment formalization* into a single Markovian model that encompasses all states that the model may ever assume as well as all input sequences permitted by the input pattern. Its general structure is depicted in Figure 7 (Markovian model in box *Quantitative Analysis*). Each circle represents a state where the input module is about to trigger a new stimulus. Hence, the general structure is similar to the input automaton's graph in box *Model construction*, which is, in fact, a *minor* of the Markov model's graph structure.

The curly arrows, however, resemble the physical processes transducing the model between these states. They consist of finite sequences of intermediate states produced by the step-wise voltage-level updates to the charge storages in accordance to the charge transport model and transistor equations. The magnification shows the transition of the output V_{Out} from

the voltage level representing logic '1' towards logic '0' in response to the input B transitioning in the opposite direction while keeping input A at logic '1'. The state labels show the inputs and the corresponding output: $AB|Out \leftrightsquigarrow AB|Out = 10|1 \leftrightsquigarrow 11|0$.

The magnification corresponds to the result trace shown right next to it. If the input module features probabilistic choices, the sequences would be labeled with the respective probabilities. Since the model is comprehensive, we possess perfect knowledge about the model variables in all states and about the transitions between the states including their associated probabilities, if any. In particular, we know whether voltages will stabilize or whether they will change again under the current input conditions. This feature enables us to directly refer to those higher-level conditions in our queries instead of approximating or guessing.

After the Markovian model is built, multiple measurement tasks provided as *query formalizations* can be computed in a batch. This happens fully automatically by identifying the states that a query refers to and applying a set of well-known graph algorithms and solution methods for linear equations and linear programs (cf. [42]). We use this to directly calculate extremal and average delays over all states where an input change is triggered without resorting to sampling, e. g., $\Delta t_{\max} = 42$ in Figure 7 (result column in the right). It is also possible to treat probabilities as parameters instead of assigning specific values. Then, the results are computed as rational functions in these parameters like the $\Delta t(p_A)$ in Figure 7.

V. EXPERIMENTS

In the previous sections, we established how our proposed modeling and analysis method works and that it indeed yields realistic results for known samples. We now demonstrate its application in a design space exploration (DSE) of standard-cell implementations and a more detailed analysis of the most promising implementation candidates. As an example we use the 3-input minority (3-MIN) function. 3-MIN has NAND and NOR as subordinate functions and hence is functional complete, i. e., it suffices to realize every Boolean function. Therefore, it can be regarded as the bread-and-butter device for reconfigurable circuits. The DSE consists of a worst-case analysis of circuit delay and power dissipation and an average-case analysis with respect to delay, and energy-per switching operation. For the worst-case analysis, the input signals switch nondeterministically, whereas we assume a probability distribution on the switching events for the average-case analysis. Furthermore, we distinguish two application scenarios. In the first scenario (3-MIN), no information is known about the input patterns and the output is computed over all input combinations. In the worst case we consider all input combinations, and in the average case we assume that each input is equally likely to switch. The second scenario (NAND-NOR) targets an application that select between 3-MIN's subordinate functions NAND and NOR via control input A and treats the inputs B and C as data inputs. Therefore, we zoom in to the computation of these functions and consider only input combinations of the data inputs while the control input is fixed to either 0 V or 1.2 V for the worst case. For the average-case we assume a lower probability that the control input switches.

A. EXPERIMENT SETUP

As shown in Figure 7, we supplied four artifacts as inputs to the DSE. The first artifact, the device, is the MIGFET from [13] as modeled in Sect. III-B1. The second artifact, the circuit under test, is a 3-MIN circuit. The third artifact, the input pattern, depends on the experiment, we conducted, but is independent of the circuit under test. For the worst-case experiments, they are completely non-deterministic with the constraint, that an input switch is only allowed after the circuit has come to rest, such that each new input pattern is independent of the previous one except for the current output value. Likewise, the experiments examining the average performance assign each input a switching probability to change its current value. Without loss of generality, we connect the control input to the PCGs of the circuit variants that are reconfigured by a single input signal.

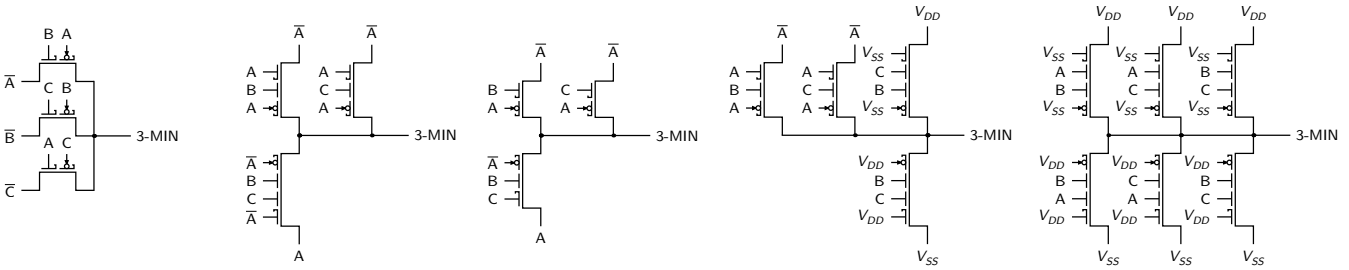
Electrically, we provide all three input signals directly and in negated form. All signals are generated by non-scaled inverters that are fully modeled with the MIGFET device according to Table 1. We use transistors with three gates and connect the input to the inner gates and the SB-gates to V_{SS} for the PMOS branch, respectively V_{DD} for the NMOS branch. While the negated signals are driven by the direct signals, these are, in turn, driven by artificial inputs with a linear slope of $0.83\,\mathrm{ps\,V^{-1}}$. Each circuit under test drives a load of one standard inverter, i. e., a fanout of $H = 1$ which is $C_{\mathrm{out}} = 80\,\mathrm{aF}$. These standard inverters, which also drive the inputs, are fully modeled and built from unscaled transistors with three gates. According to Table 1, they have a channel length of 88 nm.

Usually, delay is measured starting from a $1/2 V_{DD}$ input change, but this would skew results towards either the fastest or the slowest input transition in cases where multiple inputs change simultaneously depending on whether the delay measurements starts with the first input change or the last. In the following experiments, delay is measured from the moment the artificial inputs start to change to a new pattern until the output reaches $1/2 V_{DD}$. Using the moment the inputs start to change is overly pessimistic but removes the dependency on input signal speed and makes results comparable across topologies. Due to complete knowledge about the modeled circuit, delays are calculated until the final output crossing. So, they include all spurious output transitions that might occur due to hazards.

Computation of metrics

A consequence of using PMC is that the results in the following experiments are guaranteed to be correct with respect to the model and are not obtained from potentially noisy measurements. In our discrete charge-transfer model, we know voltage and current at each charge transmitter in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access



(1s) Fully reconfigurable, reconfiguration balanced over all inputs.

(2i) Fully reconfigurable, reconfiguration restricted to input *A*.

(2s) Same as (2i) but with reduced number of gates.

(4i) Semi-static, reconfiguration restricted to input *A*.

(8i) Fully static, all inputs balanced.

FIGURE 8. Competitive 3-MIN variants as evaluated in this section. Reconfigurable inputs attached to transistor's drain contact and source SB gate as pairwise inverted signals. Variants suffixed with *s* use SB gates (with curls) for both, reconfigurable inputs and normal inputs. Variants suffixed with *i* restrict normal inputs to inner gates.

each time step. According to Kirchhoff's circuit laws, we can compute the current power dissipation over all transistors connected to a charge storage as the sum over the individual electrical powers.

Equally, we can compute average energy per operation and average delay. On the long-run, i. e., after an initial phase, the expected performance is determined by the statistics of how the inputs switch. These measures are generally hard to capture using simulation, but PMC provides methods to compute expectations, long-run averages and related measures efficiently.

B. AUTOMATIC EXPLORATION OF CIRCUIT VARIANTS

The DSE starts with an automatic exploration of all reasonable topologies for the 3-MIN circuit. We consider circuit variants that are a single logic stage and whose inputs may appear in direct and negated form. As noted in Section III-B, multiple gates on a transistor act as a wired AND function. So, for the exploration algorithm, each transistor can be regarded as a minterm of a Boolean function. We modified the well-known Quine-McCluskey algorithm [64] to generate all minimal representations of the input Boolean function and to filter them by certain criteria in order to remove duplication caused by symmetries.

In the context of reconfigurable devices, up to three branches constitute an implementation: the static N- and P-branches as well as a reconfigurable R-branch. Each transistor in the R-branch is driven by an input signal at the source contact and covers only minterms that would otherwise appear in both, the N- and P-branch. We call circuits without R-branch *static*, circuits consisting only of an R-branch *reconfigurable*, and circuits that feature both, an R-branch and N- and P-branches *partially* reconfigurable or semi-static. We configure all transistors such that the polarity control gate is always complementary to the source signal, e. g., if the transistor is driven by V_{DD} then $PCG = 0\,\mathrm{V}$ or if signal A drives the transistor then $PCG = \overline{A}$. Static N-/P-branches connect inputs only to inner gates, to exploit their performance advantage over SB gates. Circuit reconfiguration is driven either by one or more reconfiguration inputs starting with input A. Our algorithm generates all these variants with minimized R-, N- and P-branches and discards variants that only rename inputs.

3-MIN circuit – 8 topologies with 2 types each

For 3-MIN, our exploration yields eight distinct topologies, of which only a few have been previously known from manually crafted designs. We further discriminated all (partially) reconfigurable implementations into two types suffixed either *s* or *i*, e. g., (2s) and (2i), respectively. They either make use of the source-side SB gate to connect inputs (s type) or restrict all inputs except reconfiguration to the inner gates (i type). The reason for this distinction is, that we observed, that on the one hand, using SB gates may be considerably slower. But on the other hand, restricting inputs to the inner gates and, thus, forcing that the source SB gate must also be driven by the reconfiguration input, increases the load on that input. Results will show that either effect may weigh negative or positive against the other in various scenarios (see Figures 10 and 11).

Five circuits variants stand out in the DSE for various reasons explained below. They are shown in Figure 8 and actually cover the four corners of topological possibilities to implement the circuit with reconfigurable transistors. They can be either (partially) *reconfigurable* or *static*; they can also be reconfigurable in *a single input* or *balanced over all inputs*. Variant (8i) on the right, is the static CMOS implementation exploiting MIGFETs. To its left is a representative of a partially reconfigurable, or semi-static, variant named (4i), which reconfigures only those transistors for which the reconfiguration input A actually occurs in the corresponding minterm. The branches connected to inputs B and C are implemented as static N-/P-branches. The second from the left variant (2i) is the fully reconfigurable circuit from [20], which is reconfigurable in input A only. Up to now, all circuits are of the i type, optimizing the non-reconfiguration inputs B and C. The innermost circuit variant (2s) is also known from [20] and differs from (2i) only by using the source-side SB gate to connect inputs, too. Leftmost of Figure 8 is variant (1s), which reconfigures its transistors equally in all three inputs. It is also the smallest possible implementation of 3-MIN. Due

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

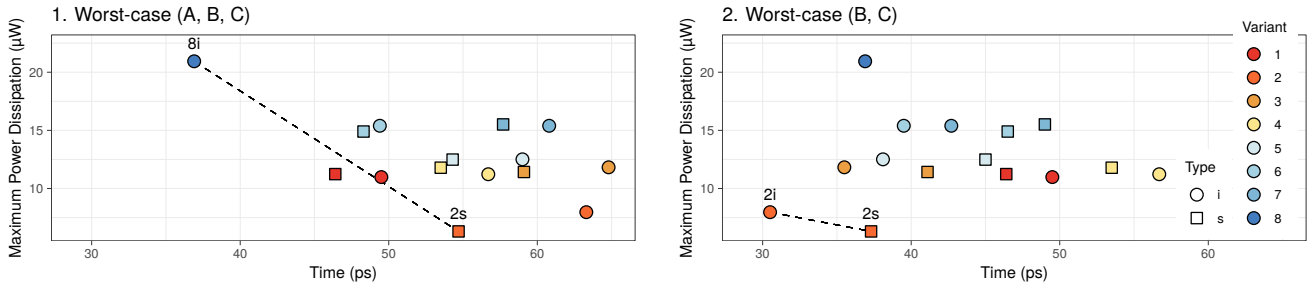


FIGURE 9. Comparison of fundamental 3-MIN designs with respect to maximum dynamic power dissipation over 1) worst-case delay with all three inputs freely switching, 2) worst-case delay with input A fixed to either input value. Transistor model MIGFET (Sect. III-B). Input slope 1 ps full swing. Output load $H = 1$ ($C_{\text{out}} = 80$ aF). Inverters after artificial inputs. If applicable, inputs are used for reconfiguration in the order A, B, then C.

to its simple structure, saving transistor gates outperforms (in terms of delay) its i type counterpart.

Classic CMOS implementations and two-stage circuits are unfavorable with the MIGFET device, as their performance suffers from long serial paths. All in all, our algorithm yields 15 circuits for the eight different topology variants (numbered 1–8) and two types (suffixed s and i).

C. DSE – WORST-CASE DELAY AND POWER

Figure 9 depicts the tradeoff between worst-case delay and worst-case power dissipation for all 15 circuit variants in two scenarios. On the left, the results for the first scenario 3-MIN are shown, where all three inputs may switch in arbitrary combinations. On the right, we consider scenario two where input A selects between the subordinate functions NAND and NOR. In both experiments, the transition that exhibits the worst-case delay need not be the same as the one that causes the maximum power dissipation, so both values independently represent the absolute worst case for each circuit under test.

1. Scenario 3-MIN: Worst case (A, B, C)

The left diagram in Figure 9 shows that, without knowledge about the switching behavior of the inputs, the static variant (8i) and the reconfigurable variant (2s) mark the extremal points of the design space on either dimension. Both are connected by a dashed tradeoff line, which marks the hypothetical ability to linearly trade power dissipation against delay during circuit design.

Opting for the most power-efficient variant (2s), we would pay with a delay that is 1.5 times longer than that of (8i) with 36 ps. Nevertheless, we trade (8i)'s speed with 3.5 times more power dissipation compared to (2s), which is ranging at 6 µW. The variants (1i/s) (red) are valuable candidates to trade power against delay, should the need arise. Under these conditions, all other variants fall behind the performance of those four variants.

2. Scenario NAND-NOR: Worst case (B, C)

The results change dramatically for a scenario where input A is reconfiguring the circuit between NAND and NOR functionality but is otherwise not engaged during normal operation. The right diagram in Figure 9 shows that the variants (2i) and (2s) are clearly the best implementation candidates. Neither the static variant (8i), which has only slightly better delay (< 1 ps) than the most power-efficient variant (2s), nor the variants (1i/s) are competitive in this scenario.

The variants (2i/s) use half the number of transistors (and gates) as variant (8i) and hence induces minimal load on the inputs B and C. This is a considerable advantage of this topology in a scenario where input A selects between NAND and NOR. This reflects in a difference in power dissipation of $\Delta P \approx 13$ µW between (8i) and (2s) which is a factor up to 3.5.

Comparing the delays between the left and right diagrams, it turns out that variants (1i/s) and (8i) are symmetric topologies. The worst-case values they exhibited on the left were achieved by changing two inputs simultaneously. Which one does not matter, since all are combinations equivalent. On one hand, by fixing input A, symmetric circuits have nothing to gain. On the other hand, as Section V-E will show, they are not susceptible to changes of the switching probabilities of the input signals.

D. DSE – AVERAGE DELAY AND DYNAMIC ENERGY

Figure 10 displays two diagrams depicting the tradeoff between the average *dynamic* energy consumption per operation and the average circuit delay for two scenarios. On the left, scenario 1 is shown, where all inputs switch with probability $p_A = p_B = p_C = 1/2$, which means, every time the circuit has stabilized, each input is equally likely to switch to its opposite value or to stay as it is. On the right, scenario 2 is shown, i. e., only inputs B and C switch with probability $p_B = p_C = 1/2$ and input A is 10 000 times less likely to switch with probability $p_A = 10^{-4}$, which resembles the second scenario of the worst-case experiments. Quantifying the average energy per operation is particularly interesting, if the circuit is designed for applications with a limited energy budget. Average delay is most important for asynchronous circuits that, despite being hard to design, are known to be very energy efficient.

3. Scenario 3-MIN: Average ($p_A = p_B = p_C = 1/2$)

Similarly to the first worst-case analysis, the DSE in the first average-case scenario is bounded by the variants (8i) and (2s), as the left diagram in Figure 10 shows. However, all other variants are distributed along the tradeoff line now. This

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

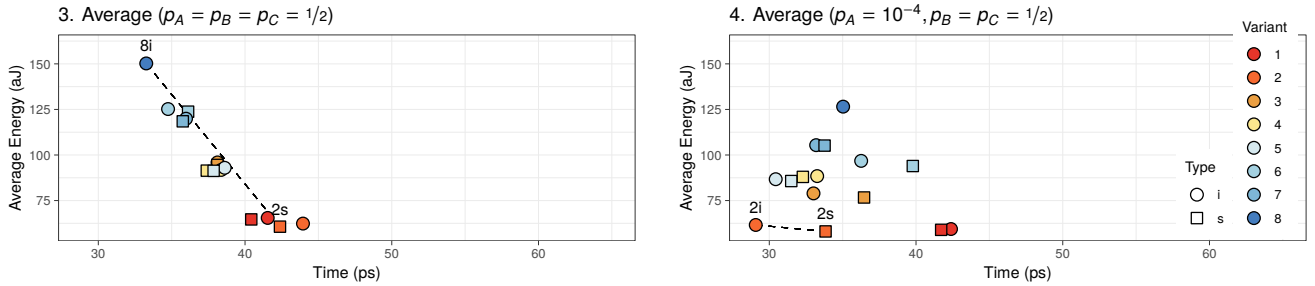


FIGURE 10. Comparison of fundamental 3-MIN designs with respect to average dynamic energy consumption per operation over 3) average delay with equal switching probability for all three inputs and 4) average delay with switching probability for input A reduced to $p_A = 10^{-4}$. Experiment parameters as in Figure 9.

means, if nothing is known about the input distribution and if worst-case performance is not critical, the average dynamic energy per operation can be directly traded against the average delay. The delay distributes over a range of 33 ps to 44 ps, with the second-best circuit (6i) being only 3 % slower and the slowest circuit (2i) being 30 % slower than the fastest (8i). The energy per operation distributes over 60 aJ to 150 aJ. The second-best circuit (2i) uses 3 % more energy and the worst (8i) 2.47 times more energy than the most efficient (2s). The variants fall into four clusters that correlate with the number of static N-/P-branches in their topology from 0 (all transistors reconfigurable) to 3 (fully static).

4. Scenario NAND-NOR: Average ($p_A = 10^{-4}, p_B = p_C = 1/2$)

Reducing the switching probability of input A impacts the results in the right diagram of Figure 10 in a similar way as fixing input A in the second worst-case scenario emphasized the structural benefits of variants (2i/s). However, all circuits are closer together under these assumptions, with surprising improvements of the average delay performance of variants (4i/s) (yellow) from last places (Figure 9, right) to the front of the field.

With respect to the average delay, the fully static variant (8i) performs below the majority of the field. Comparing the left and right diagrams, many variants shift down and to the left with variants (8i), (6i/s) and (1i/s) being notable exceptions. This means, they all improve in that scenario, even though the designer can no longer trade energy against delay linearly. Furthermore, variants (8i) and (1i/s) show almost the same average delay in both scenarios due to their symmetric topology.

Regarding the energy per operation, variants (2i/s) and (1i/s) differ only about 1 aJ on average. Hence, (1i) and (1s) might be eligible candidates for being even smaller circuits. Referring back to the left diagram or the worst-case scenarios, though, these are highly affected by changes in the input distribution.

5) Observations

The four experiments from this section show that the (partially) reconfigurable implementations are competitive w. r. t. worst-case delay if used to select between NAND and NOR. The five topologies, for which the circuit diagrams are depicted in Figure 8, particularly stand out:

(1s) Reconfigurable, balanced inputs. Slower, but shows as balanced a performance as static variant (8i),

(2i) Reconfigurable via input A. Performs best in the NAND-NOR scenario,

(2s) Same topology as (2i). Most power / energy efficient circuit but sensitive to input distribution,

(4i) Semi-static, reconfigurable via input A. Might be interesting, if both 3-MIN and NAND/NOR scenarios overlap,

(8i) Static, balanced inputs. Performs best in the 3-MIN scenario.

A rule of thumb, that can be distilled from comparing experiments 1 and 2, is: *Inputs to sub-ordinate functions should be connected to inner gates only (i types left of s types in Figure 10, right), whereas inputs to the higher-order function 3-MIN should also use the SB gates, too (s types left of i types in Figure 10, left).*

The average delay experiments make an even stronger point in favor of (partially) reconfigurable circuits. Experiments 3 and 4 gave a hint that energy and delay performance may strongly depend on the input distribution. If the switching probability of a single input (without loss of generality, input A) is at least somewhat skewed towards zero, the reconfigurable variants make a viable option for implementing 3-MIN.

E. IN-DEPTH COMPARISON OF FOUR 3-MIN-DESIGNS

We take a closer look on four of the five circuit, variant (2i) is very similar to (2s) and is left out. If the logic gate is treated as reconfigurable element between the subordinate functions NAND and NOR, (2i) and (4i) outperform the fully static implementation (8i). If the logic gate is not on the critical path, (1s) can be a promising candidate to lower the overall circuits complexity. Here, we focus on average-delay and quantify the variants in order to answer which implementation performs best for certain input switching probability ranges.

In contrast to previous experiments and other simulation techniques, we now keep the switching probability p_A a parameter [65], [66] and compute the results as *rational functions* that define the average delay precisely in p_A instead of sampling the delay for specific values. Those rational functions can then be subject to further mathematical

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access

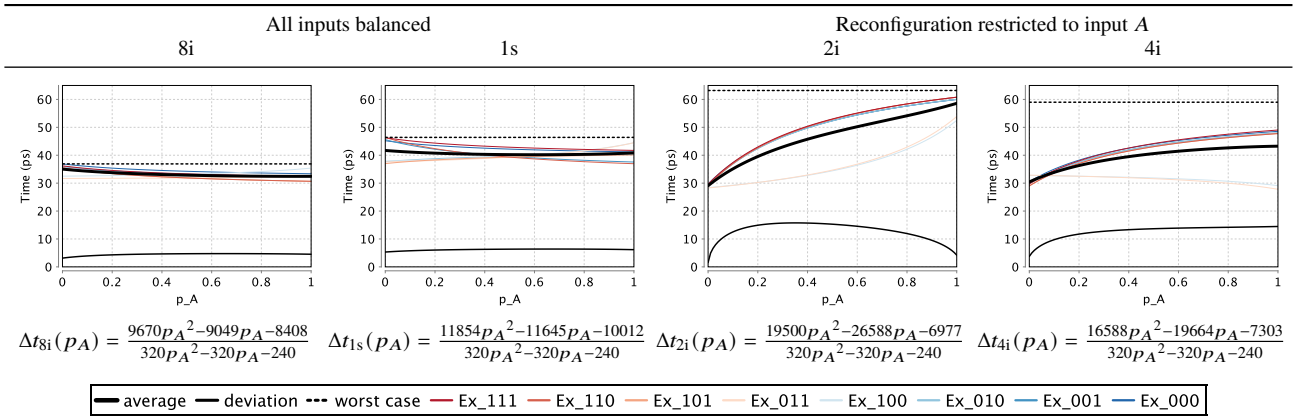


FIGURE 11. Average delays and standard deviation of four 3-MIN variants based on the reconfigurable MIGFET from Sect. III-B1. The two columns on the left show the "balanced" implementations (8i) and (1i) and the two columns on the right show variants (2i) and (4i) which restrict reconfiguration to input A. Additionally, the plots show the worst-case delay as well as the average delays from all states where the logic gate is about to switch away from, e. g., Ex_111 is the average delay if switching from $A = B = C = V_{DD}$. Below each plot is the rational function describing the respective average delay. Input slope 1 ps, output load $H = 1$ ($C_{\text{out}} = 80$ aF). Artificial inputs fed through inverters before use.

analysis and could be used in synthesis tools to select suitable implementations according to the application scenario. For simplicity, we consider only p_A a parameter here and keep the probabilities of the other inputs fixed at $1/2$.

1) Parametric average delays

The left side of Figure 11 shows the long-run average delay and the standard deviation for the four selected circuit variants. On the right side, we show the fastest implementations according various intersection points. We refer the reader back to Section V-A for an explanation of long-run average delay. It turns out that the variants can be categorized in balanced (1s, 8i) and restricted (2i, 4i) implementations.

The bold black curves give the average delay on the long run and the thin black line at the bottom shows the standard deviation σ. The respective worst-case delays are depicted as dotted thresholds on the top. They may not be reached by their corresponding delay curves, as these are averages. The other colored curves are the average delays for all possible input combinations the gate is about to switch away from, e. g., Ex_111 is the average delay if switching from $A = B = C = V_{DD}$. The rational function of each long-run average delay is given below the respective plot.

The average delay of the minimal reconfigurable implementation (1s) depends only slightly on p_A, which confirms our previous experiments. Though clearly slower, its characteristics are quite similar to the static implementation (8i) with the same standard deviation of only about 5 ps. This indicates that (1s) might indeed be a suitable choice on non-critical paths. As expected, the reconfigurable variant (2i) shows a significant dependence on p_A as its delay grows rapidly in p_A. For probabilities less than $p_A \approx 0.01$, though, the delay drops below the static variant (8i) in both, the average delay as well as the standard deviation. Interestingly, the standard deviation decreases if p_A approaches either 0 or 1 and reaches its maximum around $p_A = 0.3$. This is explained by the fact that input A drives more gates than the other inputs, see Figure 8. Hence, close to 0, only the faster inputs are relevant and towards 1, input A dominates the delays. Variant (4i) is almost as fast as (2i) if p_A is small but features a shallower slope towards $p_A = 1$. Though, its standard deviation is a bit higher close 0 and increases steadily in p_A.

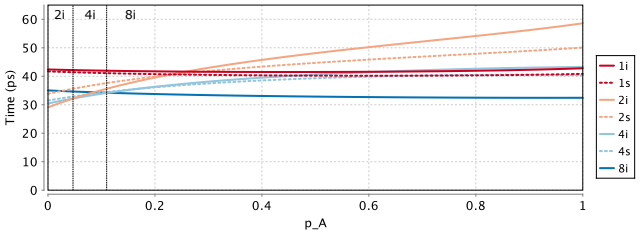
2) Observations

In the previous section, we improved over sampling and guessing probable worst-case-delay scenarios by directly computing worst-case and average-case delays for all circuit variants and then picking the best circuit variant. In this section on average performance on the long-run, we abstracted from picking particular input switching probabilities for input A and compare circuit variants for the best average delay performance over all switching probabilities of input A.

Therefore, we compute the rational function in the probability p_A that specifies the delay for each circuit variant directly. This allows us to determine the precise ranges of the input switching probabilities in which each of the modeled circuit variants perform best. Figure 12 plots the long-run average delays for each circuit variant from Figure 11 including their respective compressed $*$s or expanded $*$i circuit variants. Though clearly slower than the other variants, (1s) and (4s) are interesting options as they allow a designer to trade complexity and energy efficiency for speed effectively. For load factors other than $H = 1$, the circuit application ranges will of course differ from the table above. But as the delays turned out to be linear in H indeed, we can make the load a second parameter of the obtained rational functions straightforwardly by calculating a circuit variants slope in H.

These results demonstrate that a detailed circuit analysis uncovers the potential of reconfigurable transistors and allows us to pick the optimal solution for the targeted application scenario. Directly calculating the exact average delay rational functions considerably extends the circuit designer's possibilities to judge the use of particular implementations while eliminating the overhead of a repetitive analysis with samples

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3001352, IEEE Access



Fastest implementation depending on p_A			
Variant	2i	4i	8i
$p_A \in$	$[0, 0.0480)$	$[0.048, 0.1123)$	$[0.1123, 1]$
∗s variants are faster than ∗i variants above certain threshold			
$\Delta t_{2s} < \Delta t_{2i}$ for		$p_A > 0.2206$	
$\Delta t_{4s} < \Delta t_{4i}$ for		$p_A > 0.1601$	
Variant (4s) is faster than (1s) within two distinct intervals of p_A			
$\Delta t_{4s} < \Delta t_{1s}$ for $p_A \in$		$[0, 0.6836]$	$[0.9023, 1]$

FIGURE 12. Comparison of average delays between the variants (1i/s, 2i/s, 4i/s and 8i). Left side, graph of delay over input probability. Right side, fastest implementations according to various intersection points, for which the first row is also shown in the plot.

from the parameter's domain at this design stage.

To fully analyze the 3-MIN circuit, as shown throughout this section, our tool generated 135 experiments from 4 different queries and 4 different experiment setups. They took about 32 CPU h to compute in total. Each circuit consisted of 15–18 fully modeled transistors (with 2–4 gates each) for the function itself and the inverters driving its inputs.

VI. CONCLUSION

In this paper, we proposed a practical method that enables us to progressively develop standard cells, based on new transistor devices starting as early as technology readiness level 1. By this we were able to assess a new technology early and to retrieve results earlier in the overall development cycle.

We contributed a modular modeling framework that supports the integration of new devices by providing either the basic parameters of a reconfigurable multiple-gate transistor (MIGFET) device or a custom analytical description. In this work, we provided a new model for Germanium-based reconfigurable nanowire transistors and demonstrated its accuracy of our approach by a comparison with FEM results. We used a discrete charge-transport model that is tailored to a formal quantitative analysis by probabilistic model checking. This enabled us to compute a range of performance and quality measures, some of which are hard to obtain from simulation-based approaches. These are, for instance, the average circuit delay, which is most-relevant for asynchronous circuits, the average energy per switching operation, and the maximum power dissipation. We computed these results as rational functions that can be evaluated later with probabilities obtained from statistics over the actual input sequences and thus avoiding the necessity of reevaluation.

In our accompanying experiments, we demonstrated that our approach enabled us to do a comprehensive early technology evaluation regarding digital circuit designs. Long before prototypical circuits become available at relevant features sizes, we were able to quantify how the technology is about to perform time- and energy-wise. This provides useful insights into how to drive technology development to take full advantage of the emerging technology's capabilities.

References

[1] A. Heinzig, S. Slesazeck, F. Kreupl, T. Mikolajick, and W. M. Weber, "Reconfigurable silicon nanowire transistors," *Nano Letters*, vol. 12, no. 1, pp. 119–124, 2011.

[2] M. De Marchi *et al.*, "Polarity control in double-gate, gate-all-around vertically stacked silicon nanowire fets," in *Electron Devices Meeting, IEDM '12, 2012 IEEE Int.*, 2012, pp. 8–4.

[3] J. Zhang, P.-E. Gaillardon, and G. De Micheli, "Dual-threshold-voltage configurable circuits with three-independent-gate silicon nanowire fets," in *Circuits and Systems, ISCAS '13, 2013 IEEE Int. Symp.*, 2013, pp. 2111–2114.

[4] A. Heinzig, T. Mikolajick, J. Trommer, D. Grimm, and W. M. Weber, "Dually active silicon nanowire transistors and circuits with equal electron and hole transport," *Nano Letters*, vol. 13, no. 9, pp. 4176–4181, 2013.

[5] J. Trommer *et al.*, "Reconfigurable nanowire transistors with multiple independent gates for efficient and programmable combinational circuits," in *Proc. 2016 Design, Automation & Test in Europe Conf. & Exhib., DATE '16*, Mar. 2016, pp. 169–174.

[6] G. Fiori and G. Iannaccone, "Simulation of graphene nanoribbon field-effect transistors," *IEEE Electron Device Letters*, vol. 28, no. 8, pp. 760–762, Aug. 2007.

[7] B. Radisavljevic, A. Radenovic, J. Brivio, V. Giacometti, and A. Kis, "Single-layer MoS2 transistors," *Nature Nanotechnology*, vol. 6, no. 3, pp. 147–150, 2011.

[8] D. Jariwala, T. J. Marks, and M. C. Hersam, "Mixed-dimensional van der waals heterostructures," *Nature Materials*, vol. 16, pp. 170–181, Aug. 2016.

[9] S. Ullah, S. S. Murthy, and A. Kumar, "SMApproxLib: Library of FPGA-based approximate multipliers," in *55th ACM/ESDA/IEEE Design Automation Conference, 2018, DAC '18*, 2018, pp. 1–6.

[10] G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*. Wellesley-Cambridge Press, 1988.

[11] M. G. Larson and F. Bengzon, *The Finite Element Method: Theory, Implementation, and Applications*. Springer, 2013.

[12] L. W. Nagel and D. Pederson, "SPICE (simulation program with integrated circuit emphasis)," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M382, Apr. 1973.

[13] J. Trommer *et al.*, "Enabling energy efficiency and polarity control in germanium nanowire transistors by individually gated nanojunctions," *ACS Nano*, vol. 11, no. 2, pp. 1704–1711, 2017.

[14] M. Simon *et al.*, "A wired-and transistor: Polarity controllable fet with multiple inputs," in *2018 76th Device Research Conf. (DRC'18)*, Jun. 2018, pp. 1–2.

[15] Y. Bi, P.-E. Gaillardon, X. S. Hu, M. Niemier, J.-S. Yuan, and Y. Jin, "Leveraging emerging technology for hardware security-case study on silicon nanowire FETs and graphene SymFETs," in *2014 IEEE 23rd Asian Test Symp.*, IEEE, 2014, pp. 342–347.

[16] J. Knechtel, "Hardware security for and beyond CMOS technology: An overview on fundamentals, applications, and challenges," in *Proc. 2020 Int. Symp. on Physical Design, ISPD '20*, 2020, pp. 75–86.

[17] D. Vana, P.-E. Gaillardon, and A. Teman, "C^2TIG: Dynamic C^2MOS design based on three-independent-gate field-effect transistors," *IEEE Transactions on Nanotechnology*, vol. 19, pp. 123–136, 2020.

[18] J. Trommer, M. Simon, S. Slesazeck, W. M. Weber, and T. Mikolajick, "Inherent charge-sharing-free dynamic logic gates employing transistors with multiple independent inputs," *IEEE Journal of the Electron Devices Society*, 2020.

[19] P.-E. Gaillardon *et al.*, "Three-independent-gate transistors: Opportunities in digital, analog and RF applications," in *17th Latin-American Test Symposium, 2016, LATS '16*, IEEE, 2016, pp. 195–200.

[20] M. Raitza *et al.*, "Exploiting transistor-level reconfiguration to optimize combinational circuits," in *Proc. 2017 Design, Automation & Test in Europe Conf. & Exhib., DATE '17*, IEEE, 2017.

[21] S. Rai *et al.*, "A physical synthesis flow for early technology evaluation of silicon nanowire based reconfigurable fets," in *Proc. 2018 Design, Automation & Test in Europe Conf. & Exhib., DATE '18*, 2018, pp. 605–608.

[22] G. Gore, P. Cadareanu, E. Giacomin, and P.-E. Gaillardon, "A predictive process design kit for three-independent-gate field-effect transistors," in *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*, IEEE, 2019, pp. 172–177.

[23] V. Vaidya, J. Kim, J. N. Haddock, B. Kippelen, and D. Wilson, "SPICE optimization of organic FET models using charge transport elements," *IEEE Trans. Electron Devices*, vol. 56, no. 1, pp. 38–42, 2009.

[24] T. Kropf, Ed., *Formal Hardware Verification - Methods and Systems in Comparison*, ser. Lecture Notes in Computer Science. Springer, 1997, vol. 1287.

[25] G. D. Hachtel and F. Somenzi, *Logic Synthesis and Verification Algorithms*. Springer, 2006.

[26] V. D'Silva, D. Kroening, and G. Weissenbacher, "A survey of automated techniques for formal software verification," *IEEE Trans. CAD of Integrated Circuits and Systems*, vol. 27, no. 7, pp. 1165–1178, Jul. 2008.

[27] D. W. Loveland, *Automated Theorem Proving: A Logical Basis*, ser. Fundamental studies in computer science. North-Holland, 1978, vol. 6.

[28] Y. Bertot and P. Castéran, *Interactive Theorem Proving and Program Development - Coq'Art: The Calculus of Inductive Constructions*, ser. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2004.

[29] J. Harrison, "Floating-point verification using theorem proving," in *Formal Methods for Hardware Verification, 6th Int. School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM '06, Adv. Lectures*, vol. 3965, 2006, pp. 211–242.

[30] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Logics of Programs, Workshop, 1981*, vol. 131, 1981, pp. 52–71.

[31] J.-P. Queille and J. Sifakis, "Specification and verification of concurrent systems in CESAR," in *Int. Symp. on Programming, 5th Colloq., 1982 Proc.*, vol. 137, 1982, pp. 337–351.

[32] L. Fix, "Fifteen years of formal property verification in intel," in *25 Years of Model Checking: History, Achievements, Perspectives*, O. Grumberg and H. Veith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 139–144.

[33] A. Biere, E. M. Clarke, R. Raimi, and Y. Zhu, "Verifiying safety properties of a power PC microprocessor using symbolic model checking without BDDs," in *Computer Aided Verification, 11th Int. Conf., CAV '99, 1999 Proc.*, vol. 1633, 1999, pp. 60–71.

[34] D. Borrione, P. Georgelin, and V. M. Rodrigues, "Symbolic simulation and verification of VHDL with ACL2," in *System-on-Chip Methodologies & Design Languages*, P. J. Ashenden, J. P. Mermet, and R. Seepold, Eds., Springer, 2001, pp. 59–69.

[35] M. Raffelsieper, J.-W. Roorda, and M. Mousavi, "Model checking verilog descriptions of cell libraries," in *2009 9th Int. Conf. on Application of Concurrency to System Design, ACSD '09*, Jul. 2009, pp. 128–137.

[36] T. I. Kirkpatrick and N. R. Clark, "Pert as an aid to logic design," *IBM Journal of Research and Development*, vol. 10, no. 2, pp. 135–141, Mar. 1966.

[37] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[38] O. Maler and A. Pnueli, "Timing analysis of asynchronous circuits using timed automata," in *Correct Hardware Design and Verification Methods*, 1995, pp. 189–205.

[39] Q. Ain and O. Hasan, "Formal timing analysis of digital circuits," in *6th Int. Workshop, FTSCS '18, 2018*, Jan. 2019, pp. 84–100.

[40] W. C. Elmore, "The transient response of damped linear networks with particular regard to wideband amplifiers," *Journal of Applied Physics*, vol. 19, pp. 55–63, 1948.

[41] J.-P. Katoen, "The probabilistic model checking landscape," in *Proc. 31st Annu. ACM/IEEE Symp. on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, 2016, pp. 31–45.

[42] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[43] J. A. Kumar, L. Liu, and S. Vasudevan, "Scaling probabilistic timing verification of hardware using abstractions in design source code," in *2011 Formal Methods in Computer-Aided Design, FMCAD '11*, Oct. 2011, pp. 196–205.

[44] G. Norman, D. Parker, M. Kwiatkowska, and S. K. Shukla, "Evaluating the reliability of NAND multiplexing with PRISM," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 10, pp. 1629–1637, Oct. 2005.

[45] K. Hoque, O. Ait Mohamed, Y. Savaria, and C. Thibeault, "Probabilistic model checking based DAL analysis to optimize a combined TMR-blind-scrubbing mitigation technique for FPGA-based aerospace applications," in *12th ACM/IEEE Int. Conf. on Methods and Models for System Design*, Oct. 2014.

[46] H. G. Mohammadi, P.-E. Gaillardon, and G. De Micheli, "Fault modeling in controllable polarity silicon nanowire circuits," in *Proc. 2015 Design, Automation & Test in Europe Conf. & Exhib., DATE '15*, 2015, pp. 453–458.

[47] D. Bhaduri and S. Shukla, "Tools and techniques for evaluating reliability trade-offs for nano-architectures," *Nano, quantum and molecular computing*, pp. 157–211, 2004.

[48] L. Amarú, P.-E. Gaillardon, and G. De Micheli, "Biconditional BDD: A novel canonical BDD for logic synthesis targeting XOR-rich circuits," in *Proc. 2013 Design, Automation & Test in Europe Conf. & Exhib., DATE '13*, Mar. 2013, pp. 1014–1017.

[49] J. Zhang, M. De Marchi, D. Sacchetto, P.-E. Gaillardon, Y. Leblebici, and G. De Micheli, "Polarity-controllable silicon nanowire transistors with dual threshold voltages," *Electron Devices, IEEE Trans.*, vol. 61, no. 11, pp. 3654–3660, Nov. 2014.

[50] M. H. Ben-Jamaa *et al.*, "Silicon nanowire arrays and crossbars: Top-down fabrication techniques and circuit applications," *Science of Advanced Materials*, vol. 3, no. 3, pp. 466–476, 2011.

[51] J. C. Bioch and T. Ibaraki, "Decompositions of positive self-dual Boolean functions," *Discrete Mathematics*, vol. 140, no. 1-3, pp. 23–46, 1995.

[52] S. Rai, M. Raitza, and A. Kumar, "Discern: Distilling standard-cells for emerging reconfigurable nanotechnologies," in *Proc. 2020 Design, Automation & Test in Europe Conf. & Exhib., DATE '20*, IEEE, 2020, xx–yy.

[53] A. M. Ionescu and H. Riel, "Tunnel field-effect transistors as energy-efficient electronic switches," *Nature*, vol. 479, pp. 329–337, Nov. 2011.

[54] N. Arora, *MOSFET Modeling for VLSI Simulation Theory and Practice*. World Scientific, 2007.

[55] V. Forejt, M. Kwiatkowska, and D. Parker, "Pareto curves for probabilistic model checking," in *Proc. 10th Int. Symp. on Automated Technology for Verification and Analysis, ATVA '12*, vol. 7561, 2012, pp. 317–332.

[56] J. Klein *et al.*, "Advances in symbolic probabilistic model checking with PRISM," in *Proc. 22nd Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, TACAS '16*, vol. 9636, 2016, pp. 349–366.

[57] P. Packan *et al.*, "High performance 32nm logic technology featuring 2nd generation high-k + metal gate transistors," in *2009 IEEE Int. Electron Devices Meeting, IEDM '09*, Dec. 2009, pp. 1–4.

[58] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proc. 23rd Int. Conf. on Computer Aided Verification, CAV '11*, vol. 6806, 2011, pp. 585–591.

[59] J. A. W. Kamp, "Tense logic and the theory of linear order," Ph.D. dissertation, Univ. of California, Los Angeles, 1968.

[60] A. Pnueli, "The temporal logic of programs," in *18th Annu. Symp. on Foundations of Computer Science, SFCS '77*, Oct. 1977, pp. 46–57.

[61] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic," in *Logic of Programs, Workshop*, 1982, pp. 52–71.

[62] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A storm is coming: A modern probabilistic model checker," in *Computer Aided Verification, 29th Int. Conf., CAV '17, 2017 Proc., Part II*, vol. 10427, 2017, pp. 592–600.

[63] A. Hartmanns and H. Hermanns, "The modest toolset: An integrated environment for quantitative modelling and verification," in *Tools and Algorithms for the Construction and Analysis of Systems, 20th Int. Conf., TACAS '14, 2014 Proc.*, 2014, pp. 593–598.

[64] E. J. McCluskey, "Minimization of boolean functions," *The Bell System Technical Journal*, vol. 35, no. 6, pp. 1417–1444, Nov. 1956.

[65] R. Lanotte, A. Maggiolo-Schettini, and A. Troina, "Parametric probabilistic transition systems for system design and analysis," *Formal Aspects of Computers*, vol. 19, no. 1, pp. 93–109, 2007.

[66] E. M. Hahn, H. Hermanns, and L. Zhang, "Probabilistic reachability for parametric markov models," in *Model Checking Software, 16th Int. SPIN Workshop, 2009 Proc.*, 2009, pp. 88–106.