A Transistor Level Relational Semantics for Electrical Rule Checking by SMT Solving

Oussama Oulkaid[§†¶], Bruno Ferres[§†], Matthieu Moy[§], Pascal Raymond[†], Mehdi Khosravian[¶], Ludovic Henrio[§], Gabriel Radanne[§]
[§]Univ. Lyon, EnsL, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
[†]Univ. Grenoble Alpes, CNRS, Grenoble INP*, VERIMAG, 38000 Grenoble, France
[¶]Aniah, 38000 Grenoble, France

***Abstract*— We present a novel technique for Electrical Rule Checking (ERC) based on formal methods. We define a relational semantics of Integrated Circuits (IC) as a means to model circuits' behavior at transistor-level. We use Z3, a Satisfiability Modulo Theory (SMT) solver, to verify electrical properties on circuits — thanks to the defined semantics. We demonstrate the usability of the approach to detect current leakage due to missing level-shifter on large industrial circuits, and we conduct experiments to study the scalability of the approach.**

***Index Terms*—formal verification, integrated circuits, electrical rule checking, SMT solving**

I. Introduction

Modern circuits often count tens of billions of transistors [1]. Their design is a long process where verification plays a crucial rule. For Application Specific Integrated Circuit (ASIC), 50 %-60 % of median project time is spent in verification, according to the 2022 Wilson Research Group study [2].

One essential family of verification, called Electrical Rule Checking (ERC), verifies electrical properties via electronic design rules on Integrated Circuits (IC). Mostly, such rules protect the circuit against the so-called electrical errors. Examples of electrical errors include *floating* (*i.e.,* not connected to any supply) transistor gates, undesired current leakage, missing level-shifters between power domains, Electrical OverStress (EOS), ElectroStatic Discharge (ESD), etc.

ERC requires intervening at the transistor-level, when the circuit is abstracted as a net-list (*i.e.,* a set of transistors and *nets* representing wires connecting them). Indeed, in earlier stages of the circuit design flow — *e.g.,* Register-Transfer Level (RTL), the abstraction is purely logical; power domains, for example, are not available, and thus electrical properties cannot be expressed, and even less verified. Transistor net-lists are most of the time synthesized from the logical design, by adding electrical information, but can also be partially written or adapted by hand.

Some ERC approaches rely on transistor-level simulation, which is very precise but usually neither exhaustive nor scalable. Other approaches rely on performing *static voltage propagation*, which computes the set of supplies that are reachable for each net in the circuit. Since this approach yields a lot of false alarms, it is usually used together with *pattern-matching* to filter-out warnings raised in known-correct circuit patterns.

In this work, we propose a novel approach to ERC — that is sound, scalable and automatic — based on formal methods. For this purpose, we apply a Satisfiability Modulo Theory (SMT) solving based technique to tell whether an electrical configuration of a circuit can lead to an error [3]. This is achieved by (1) encoding the circuit into a logical formula according to some semantics defined in Section III, (2) encoding the property to verify (*e.g.,* absence of some error), and (3) checking the satisfiability of the conjunction of the circuit formula and the error formula. We delegate the formula solving task to an external solver. The latter gives an answer on whether the property is satisfiable. If so, the solver yields an error scenario. Else, we know that the error cannot occur. In Section IV, we show an application of this approach for the detection of missing level-shifters, a well known problem that can occur when circuits have several power domains [4].

II. Electrical Rule Checking Techniques

In this section, we present some of the most widely used approaches for ERC. We show how they do differ from each other, as well as how they compare to our approach.

A. Simulation

Simulation Program with Integrated Circuit Emphasis (SPICE) is a widely used electronic circuits simulator. It is an industry-standard tool, and is used in transistor-level verification flows. Because it is very precise, transistor-level simulation requires large computational resources, which makes it difficult to handle large designs [5]. Not only does SPICE use heavyweight mathematical models including differential algebraic equations [6], but each simulation is done for a given set of input voltages (called input vector). An exhaustive simulation requires to consider a number of input vectors that grows exponentially with the input size. Large circuits cannot be verified exhaustively in reasonable time, hence simulation provides no guarantee that all errors are detected. Another limitation of simulation is that it cannot easily explore the state-space of memory-related circuit states. Indeed the internal state of a circuit with memory does not only depend

This work is partially funded by region Auvergne-Rhône-Alpes as part of the EASYTECH program led by MINALOGIC.

*Institute of Engineering Univ. Grenoble Alpes

979-8-3503-4859-0/DATE24/© 2024 EDAA

on the input vector, but also depends on the previous state points, hence an exhaustive enumeration of inputs does not explore all internal states.

B. Topology based techniques

Some verification strategies consist in statically searching for potential errors, using some of the so-called static voltage propagation algorithms. This kind of approach checks for topologies where a net's static paths to supplies may lead to an error, without checking whether these paths are *dynamically* plausible. Some static voltage propagation algorithms use the simple abstraction of all transistors being switched-on, regardless of the voltage applied on their gate — which captures all reachable scenarios. Such abstraction provides sound results, but has the flaw of resulting in a lot of false alarms — since some unfeasible circuit states may arise [4], [7]. Other algorithms use *dynamic voltage propagation*: they take into consideration the dynamic states of transistors, where an iterative algorithm is applied on a graph representation of the circuit to annotate every node with the supply voltages it can reach [8]. Voltage propagation analyses are usually combined with some refinement approaches as a means to reduce the number of false alarms raised by voltage propagation [9]. These refinement techniques involve analyzing the structure of the circuit on which a warning was raised, to check whether it embeds some topology that is known to act as a protection block against the error of interest. These techniques are commonly referred to as *topology recognition* or pattern-matching techniques. Pattern-matching can be used to check the presence of ESD protection structures in multi-supply designs [10]. Other applications include the detection of floating transistor gates, missing level-shifters, and missing isolation cells [4]. Such applications require an exhaustive library of circuit topologies that are known to fulfill a specific function (*e.g.,* ESD protections, level-shifters, etc.). A search algorithm can be applied to look for those patterns within the design. This approach is useful to capture some types of errors, but may still lead to undetected false alarms.

C. Formal approaches

Formal methods are able to prove properties by reasoning on the state-space of a system without executing it. They are usually sound (*i.e.,* they provide a guarantee that all errors are detected), which makes them well-suited for the verification of safety-critical systems as well as RTL circuit verification, where proofs for correctness are highly demanded [11]. The use of formal methods for ERC remains very rare. Afonso and Monteiro [12] present an application of satisfiability (SAT) solving[1] for the analysis of short-circuit conditions at transistor-level in logic circuits. They encode a circuit into a Quantified Boolean Formula (QBF) that is derived from a graph based representation of the circuit, in order to find short-circuit configurations (*i.e.,* input valuations that result in an electrical path from the supply to the ground). The presented approach only applies to circuits with a single power supply that serves as the logical "1" for the whole circuit. This is a strong limitation, since modern circuits almost always have multiple power supplies. Also, the presented approach only handles one specific type of errors — that is short-circuits. Our work is closely related to their approach, yet our aim is to present a more modular approach that deals with different kinds of errors on multi-supply designs.

D. Our contribution

We apply SMT solving to check if errors are possible (*i.e.,* electrical steady states that violate some electrical rules). SMT supplements SAT with non-Boolean theories (*e.g.,* linear and non-linear arithmetic, integer numbers, real numbers, etc.). SMT is, therefore, able to check the satisfiability of formulas such as: $(x < y) \wedge (z + y \leq x) \wedge (\neg a \vee b)$, where a and b are Boolean variables and x, y and z are real number variables. The advantage of SMT over SAT is that it allows us to cover multi-supply circuits; SMT is more expressive and allows us to manipulate voltage values directly as rational numbers instead of being restricted to logical 0's and 1's like [12]. In contrast to simulation, our approach consists in symbolically, and therefore exhaustively, verifying electrical properties on circuits. Our approach is more precise than static voltage propagation, and unlike pattern-matching based approaches, it requires no prior knowledge on circuit topologies.

III. Integrated circuit semantics

In this section, we introduce the mathematical abstraction modeling circuits at transistor-level. We introduce our model, piece by piece, as conjoint SMT formulas. Such model makes it possible to capture the electrical steady states in a given supplies configuration.

A. Prerequisites

Let $\mathbf{N}$ denote the set of nets in a circuit, and let $\mathbf{S}$ denote the set of circuit supplies. $2^{\mathbf{S}}$ is the power set of $\mathbf{S}$. We consider a static function[2] ReachS, which returns the set of reachable supplies from a given net (Equation E_{ReachS}). We assume that this information is obtained from static voltage propagation.

$$\text{ReachS} : \mathbf{N} \longrightarrow 2^{\mathbf{S}} \qquad (E_{\text{ReachS}})$$

Let $\mathbf{I}$ denote the set of all inputs of the circuit — which are free variables of the system. From static voltage propagation, we build the set of inputs $\mathbf{I}$ as the circuit nets that can reach no supply (Equation $E_{\mathbf{I}}$). Both sets $\mathbf{S}$ and $\mathbf{I}$ are subsets of $\mathbf{N}$.

$$\mathbf{I} \triangleq \{i \in \mathbf{N} \mid \text{ReachS}(i) = \varnothing\} \qquad (E_{\mathbf{I}})$$

[1] SAT solving refers to checking whether a Boolean formula (*e.g.,* $a \wedge (b \vee \neg c)$, where a, b, and c are Boolean variables) is satisfiable. If so, the solver provides one solution as a plausible valuation of the formula variables.

[2] A static function is a function that returns information that can be extracted statically from the circuit net-list. We use SmallCaps font to write those functions. In contrast, dynamic functions ($\mathcal{O}n$ and $\mathcal{V}$ below) encode the logical constraints to be solved by an SMT solver.

B. Voltage abstraction

A function $\mathcal{V}$ defines the voltage value, for each net, as a non-negative rational number (Equation $E_{\mathcal{V}}$).

$$\mathcal{V}\colon \mathbf{N} \longrightarrow \mathbb{Q}^{+} \qquad (E_{\mathcal{V}})$$

In the following section, we define a transistors' model, which we formalize using the $\mathcal{V}$ function. Later on, we define a set of constraints that $\mathcal{V}$ and other functions of the system must satisfy. Our tool then uses an SMT solver to check whether there exists a valuation of these functions such that these constraints are satisfied.

C. Transistor modeling

We now describe the physical behavior of transistors, and our abstract model that represents it. We base our modeling of transistors on MOSFET (Metal-Oxide-Semiconductor Field-Effect Transistor) devices. There exists two complementary MOSFET device types (Fig. 1).

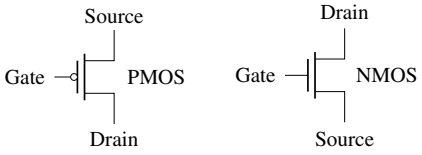

Fig. 1: PMOS (left) and NMOS (right) device symbols

PMOS and NMOS devices have characteristic curves $I_d = f(V_{GS})$ that can be obtained from running a SPICE simulation, as shown by the blue line in Fig. 2. I_d is the outgoing current from the transistor drain, and V_{GS} is the voltage difference between the gate and source nets (*i.e.,* $\mathcal{V}(\text{Gate}) - \mathcal{V}(\text{Source})$). V_S denotes $\mathcal{V}(\text{Source})$. Both PMOS and NMOS devices are characterized with a threshold voltage V_{th}. An NMOS device starts to be active (*i.e.,* current flows between its source and drain) as the voltage difference V_{GS} exceeds $|V_{th}|$. The current is maximal when V_{GS} reaches the saturation voltage V_{sat}. On the other hand, a PMOS device is in the saturation region when V_{GS} is minimal, and as soon as V_{GS} reaches $-|V_{th}|$ it enters the cut-off region — disconnecting thus source from drain.

We use the *switch* abstraction to model transistor states; a transistor is considered to be either switched-on or switched-off. Since we don't model voltage values precisely, a precise value for V_{th} would not make sens. We choose $V_{th} = 0\,\text{V}$ as a simplification, *i.e.,* consider that transistors are switched-on as soon as a voltage difference is applied on their gate. The corresponding abstract model is shown in Fig. 2 with a dashed red line, where the PMOS is switched-on for $V_{GS} < 0\,\text{V}$, and the NMOS is switched-on for $V_{GS} > 0$. The switch abstraction is widely used when dealing with digital circuits [13], [14]. It also serves as a good choice for an abstract modeling that can apply beyond digital circuits. Albeit this abstraction does not provide the precise computation of voltage values given transistor states, it is possible to estimate voltage values with intervals.

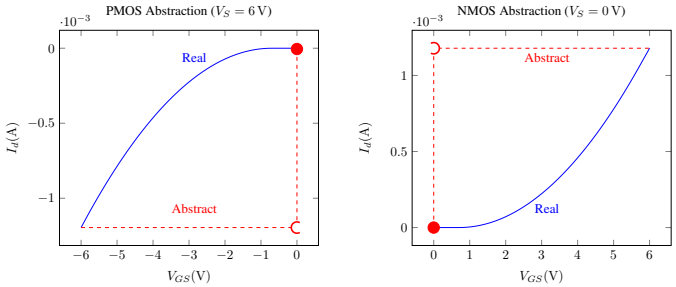

Fig. 2: PMOS (left) and NMOS (right) characteristic curves (obtained from SPICE with $V_{th} = 0.7\,\text{V}$) and their abstraction

We consider the example of an inverter circuit (Fig. 3), where three configurations are possible, as shown in Table I. When the ground voltage (0 V) applies on the input pin (A), the M1 PMOS device is switched-on and the M2 NMOS device is switched-off, which makes the output pin (Z) connected to the 5 V supply. Similarly, when A is of value 5 V, Z connects to the ground through M2. Finally, for all voltage values of A that are situated between the supply and ground, we associate an identical configuration: both M1 and M2 are switched-on, thus Z connectes to both supply and ground pins.

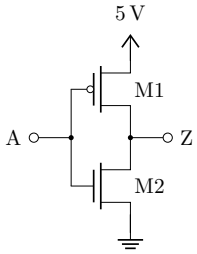

Fig. 3: Inverter circuit

A	M1	M2	Z
0 V	on	off	5 V
5 V	off	on	0 V
$\in\,]0\,\text{V}, 5\,\text{V}[$	on	on	$\in\,]0\,\text{V}, 5\,\text{V}[$

Table I: Inverter circuit states obtained from the switch abstraction of transistors

In our formalism, we denote $\mathbf{D}_{\texttt{NMOS}}$ the set of NMOS devices in a circuit, and $\mathbf{D}_{\texttt{PMOS}}$ the set of PMOS devices. We denote $\mathbf{D}$ the set of all transistors: $\mathbf{D} \triangleq \mathbf{D}_{\texttt{NMOS}} \uplus \mathbf{D}_{\texttt{PMOS}}$. We use the notation M_x to denote a transistor device, where $M_x \in \mathbf{D}$. Transistor nets can be retrieved with some static functions that we denote GATE, SRC and DRN, that respectively return the gate, source and drain nets of a device. A predicate $\mathcal{O}n\colon \mathbf{D} \longrightarrow \mathbb{B}$, indicates whether a transistor is switched-on. A MOSFET transistor is a symmetrical device (*i.e.,* its source and drain nets may be swapped in some contexts of use): its state depends on both the voltage difference between gate and source, and the voltage difference between gate and drain. The corresponding semantics rules for NMOS and PMOS device types are shown, respectively, by Rule $R_{\texttt{NMOS}}$ and Rule $R_{\texttt{PMOS}}$.

$$\bigwedge_{M_x \in \mathbf{D}_{\texttt{NMOS}}} \left(\begin{array}{ll} \mathcal{O}n(M_x) & \triangleq \mathcal{V}(\text{GATE}(M_x)) > \mathcal{V}(\text{SRC}(M_x)) \\ & \vee\ \mathcal{V}(\text{GATE}(M_x)) > \mathcal{V}(\text{DRN}(M_x)) \end{array} \right) \qquad (R_{\texttt{NMOS}})$$

$$\bigwedge_{M_x \in \mathbf{D}_{\texttt{PMOS}}} \left(\begin{array}{ll} \mathcal{O}n(M_x) & \triangleq \mathcal{V}(\text{GATE}(M_x)) < \mathcal{V}(\text{SRC}(M_x)) \\ & \vee\ \mathcal{V}(\text{GATE}(M_x)) < \mathcal{V}(\text{DRN}(M_x)) \end{array} \right) \qquad (R_{\texttt{PMOS}})$$

These rules only define the condition for a device to be switched-on, but assign no value to devices' source and drain nets. We define other rules to constrain these values, as we explain in the following section.

D. Local dynamic voltage

We denote $\Pi(n)$ the set of static neighbors of a net n. Elements of $\Pi(n)$ are triples of the form (n, M_x, p), where n is connected to some other net p through some device M_x.

We define nets' voltage values with respect to their neighboring nets (*i.e.,* nets at the other end of switched-on devices). Rule $R_{\substack{local\\voltage}}$ encodes the constraints to locally restrict a net's voltage value. It ensures that, for every net in the circuit, some current enters the net (*i.e.,* there is a strictly positive voltage difference across a switched-on device connected to the net) if and only if some current exits the net. This is an abstraction of Kirchhoff's current law.

$$\bigwedge_{n\in\mathbf{N}\setminus(\mathbf{S}\cup\mathbf{I})}\left(\begin{array}{l}\bigvee\limits_{(n,M_x,p)\in\Pi(n)} \mathcal{O}n(M_x)\wedge\big(\mathcal{V}(p)<\mathcal{V}(n)\big)\\ \Leftrightarrow \bigvee\limits_{(n,M_x,p)\in\Pi(n)} \mathcal{O}n(M_x)\wedge\big(\mathcal{V}(n)<\mathcal{V}(p)\big)\end{array}\right) \qquad (R_{\substack{local\\voltage}})$$

By construction, for each net, the formula evaluates to $true$ either by (1) having both sides evaluate to $true$ (*i.e.,* $true \Leftrightarrow true$) — which corresponds to the case where current flows through the net — or (2) having both sides evaluate to $false$ — which corresponds to the case where no current flows through the net. In case (1), the rule enforces that the voltage value of the net is strictly between the voltage values of its neighbouring nets, in other words, it ensures a strict voltage monotony along the nets that are part of some *active* electrical path (*i.e.,* a path composed of dynamically switched-on devices). In case (2), the rule enforces that the voltage value of the net is equal to that of all of its neighbouring nets.

Rule $R_{\substack{local\\voltage}}$ only applies to circuit nets that are neither supplies nor inputs. Both supplies and inputs do not depend on the internal state of the circuit, but are constrained by the environment. Hence, we define special rules to restrict supplies and inputs, as we present in Section III-F.

E. Static voltage range

Some circuit nets may be floating — in which case Rule $R_{\substack{local\\voltage}}$ doesn't constrain their value. To ensure the integrity of the voltage value of a floating net, the voltage value $\mathcal{V}$ of a net n (that is neither a supply nor an input) is bounded by a static interval defined by the supplies it can reach. We obtain such supplies statically from function ReachS (Equation E_{ReachS}) as shown in Rule $R_{\substack{static\\domain}}$.

$$\bigwedge_{n\in\mathbf{N}\setminus(\mathbf{S}\cup\mathbf{I})}\left(\begin{array}{l}\mathcal{V}(n)\geq \min\limits_{s\in\text{ReachS}(n)}\mathcal{V}(s)\\ \wedge\ \mathcal{V}(n)\leq \max\limits_{s\in\text{ReachS}(n)}\mathcal{V}(s)\end{array}\right) \qquad (R_{\substack{static\\domain}})$$

F. Building circuit formula

The presented circuit semantics allows us to write, for any circuit description, a formula that models it. Circuits are, in general, intended to function within a specific context of power supplies and input values. In order to obtain meaningful results, we need to define such context. We present, in the following, clauses to restrict circuit supplies and inputs. Afterwards, we show how a circuit formula is built, and we present its use for error verification.

We denote $\mathbf{V}$ the set of all existing supply values in a circuit, where: $\mathbf{V}\subset\mathbb{Q}^+$. For every supply, we assume that a set of possible voltage values is given.

1) Supply constraint: We define a static function Suppl that, given a supply pin, returns a set of potential voltage values (Equation E_{Suppl}).

$$\text{Suppl}\colon \mathbf{S}\longrightarrow 2^{\mathbf{V}} \qquad (E_{\text{Suppl}})$$

We simply constrain each of the circuit supplies' voltage value to the set of possible voltages they are given (Rule $R_{supplies}$).

$$\bigwedge_{s\in\mathbf{S}}\left(\bigvee_{v\in\text{Suppl}(s)}\mathcal{V}(s)=v\right) \qquad (R_{supplies})$$

2) Input constraint: We consider inputs to be restricted to the context of the circuit supply voltage values; every input can have the same $\mathcal{V}$ value as any available supply (Rule R_{inputs}).

$$\bigwedge_{i\in\mathbf{I}}\left(\bigvee_{v\in\mathbf{V}}\mathcal{V}(i)=v\right) \qquad (R_{inputs})$$

3) Formula building: Let $\mathcal{F}$ denote the circuit formula that is built using the local relational semantics (Equation $E_{\mathcal{F}}$). $\mathcal{F}$ is defined as the logical conjunction of all semantics rules.

$$\mathcal{F}\triangleq R_{\texttt{NMOS}}\wedge R_{\texttt{PMOS}}\wedge R_{\substack{local\\voltage}}\wedge R_{\substack{static\\domain}}\wedge R_{supplies}\wedge R_{inputs} \qquad (E_{\mathcal{F}})$$

4) Error specification and verification: Given a circuit and its corresponding formula $\mathcal{F}$, we consider an error as a relation between the variables appearing in $\mathcal{F}$ (most of the ERC errors can be expressed as such a relation). Let $\mathcal{E}$ be such a relation. We can use an SMT solver, like Z3, to check whether the conjunction $\mathcal{F}\wedge\mathcal{E}$ is satisfiable or not. If it is satisfiable, the corresponding solution gives a diagnosis of the error as a valuation of nets' voltages and transistors' states. It is important to note that our approach only considers electrical steady states (Section III) and cannot check for errors that involve a succession of different circuit states. If it is not satisfiable, it means that, with the transistor abstraction we have chosen, the error cannot appear in the considered circuit.

IV. Application: missing level-shifters detection

In this section, we demonstrate how the proposed approach can be used to tackle industrial problems, in particular the identification of missing level-shifters in integrated circuits.

A level-shifter is a circuit that is used to connect circuit cells belonging to different power domains, in a way that prevents

undesired current leakage on transistors at the interface of power domains. There exists several conventional level-shifter circuits. They are, in general, supplied with voltage values from the two power domains to connect, and are made of some switch circuitry to select one of the supply voltage values as the output, depending on the input value.

In this work, we study the case of level-up shifters — which translate signals from low to high power domains [15]. Fig. 4 illustrates the use of a level-up shifter to connect two inverters in series, from a $1.2\,\mathrm{V}$ domain to a $3.3\,\mathrm{V}$ domain. The level-shifter transforms the value of $\mathrm{net1}$ to a suitable value on $\mathrm{net2}$, from $1.2\,\mathrm{V}$ to $3.3\,\mathrm{V}$, or repeats $0\,\mathrm{V}$ on $\mathrm{net2}$ when $\mathrm{net1}$ is grounded. In the absence of a level-shifter (*i.e.,* directly connecting $\mathrm{net1}$ to $\mathrm{net2}$) $\mathrm{net2}$ may be at $1.2\,\mathrm{V}$, which will result in a possibly undesired current leakage on $\mathrm{M3}$ and $\mathrm{M4}$.

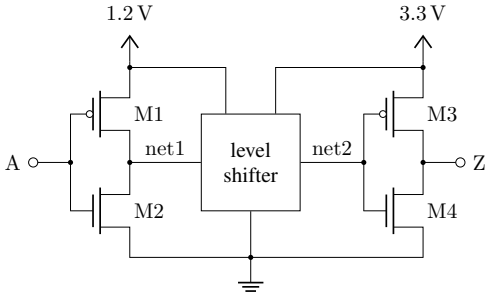

Fig. 4: Use of a level-up shifter to connect two inverters

A. Error specification

For a given circuit, we formalize a simple condition to detect a missing level-shifter with the function $\mathcal{E}_{\texttt{MLS}}$ defined by Equation $E_{\mathcal{E}_{\texttt{MLS}}}$. It consists in checking whether a specific PMOS device belongs to two different power domains in a way that results in a current leakage across the transistor. A dual clause can be expressed on NMOS device types. We say that a PMOS transistor M_x is at a low-to-high power domain interface when its gate voltage is lower than its source (or drain) voltage, without being equal to the minimal reachable value from source and drain nets (which is usually the ground). Since we are dealing with PMOS device types, this is equivalent to the device being switched-on (Rule $R_{\texttt{PMOS}}$). There is a current leakage if the transistor is switched-on and $\mathcal{V}(\textsc{Src}(M_x)) \neq \mathcal{V}(\textsc{Drn}(M_x))$ as long as the gate voltage does not correspond to the minimal reachable value from source and drain. There may be a current leakage that is not due to a missing level-shifter if $\mathcal{V}(\textsc{Gate}(M_x)) = \min_{\substack{s \in \textsc{ReachS}(\textsc{Src}(M_x)) \\ \cup \textsc{ReachS}(\textsc{Drn}(M_x))}} \mathcal{V}(s)$.

$$\mathcal{E}_{\texttt{MLS}} : \mathbf{D}_{\texttt{PMOS}} \to \mathbb{B}$$
$$M_x \mapsto \left(\begin{array}{l} \mathcal{O}n(M_x) \\ \wedge\ \mathcal{V}(\textsc{Src}(M_x)) \neq \mathcal{V}(\textsc{Drn}(M_x)) \\ \wedge\ \mathcal{V}(\textsc{Gate}(M_x)) > \min\limits_{\substack{s \in \textsc{ReachS}(\textsc{Src}(M_x)) \\ \cup \textsc{ReachS}(\textsc{Drn}(M_x))}} \mathcal{V}(s) \end{array} \right) \quad (E_{\mathcal{E}_{\texttt{MLS}}})$$

Finally, we check the absence of an error on a specific PMOS device (M_x): if $\mathcal{F} \wedge \mathcal{E}_{\texttt{MLS}}(M_x)$ is satisfiable, then we consider that M_x is indeed at the interface of two power domains where a level-shifter is missing — as we do in Section IV-B. Alternatively, we check whether there exists at least one missing level-shifter in the whole circuit, by checking the satisfiability of $\mathcal{F} \wedge \left(\bigvee_{M_x \in \mathbf{D}_{\texttt{PMOS}}} \mathcal{E}_{\texttt{MLS}}(M_x) \right)$ — as we do in Section IV-C. Note that the error formula can be tweaked to fulfill specific needs, in a fully compositional way (*e.g.,* search for missing level-shifters between specific power domains).

B. Industrial case-study

Our case-study consists of a set of real-life circuits and a set of PMOS devices which we obtained from the Aniah company, as part of a research partnership. The circuits database is made of net-lists that are part of a 10-bit Analog-to-Digital Converter (ADC). It represents a large variety of topologies that can be found in modern CMOS designs, while exhibiting a strong analog content. We analyze a total of $11\,459$ circuit instances (*i.e.,* distinct supplies configurations) of 197 unique circuits, among which 20 distinct supplies are used. The complexity of analyzed circuits depends on both the number of transistors and their connectivity. We represent the latter with the connectivity ratio $\frac{\#\text{transistors}}{\#\text{nets}}$. Table II presents a summary of the analyzed circuits.

	Min	Med	Ave	Max
Number of transistors	2	14	46.2	1303
Connectivity ratio	0.5	1.27	1.46	5.5

Table II: Statistics related to the case-study

Each pair of circuit instance and transistor represents a case of specific interest for our industrial partner. Our industrial database therefore has a set of $22\,168$ cases that we proceed to check using our SMT based approach.

Experiments are ran on a machine with $2720\,\mathrm{GiB}$ system memory, using a single core running at $3.4\,\mathrm{GHz}$. The CPU model used is Intel® Xeon® Processor E7-4890 v2. The analysis consists in building the SMT problem (*i.e.,* logical conjunction of the circuit formula and the error formula) and checking its satisfiability with Z3. Fig. 5 shows a scatter plot of the analysis time, which takes $16\,\mathrm{min}\ 12\,\mathrm{s}$ in total. $49.51\,\%$ of analyzed cases were found not to be erroneous, and for the remaining $50.49\,\%$ an error scenario is found.

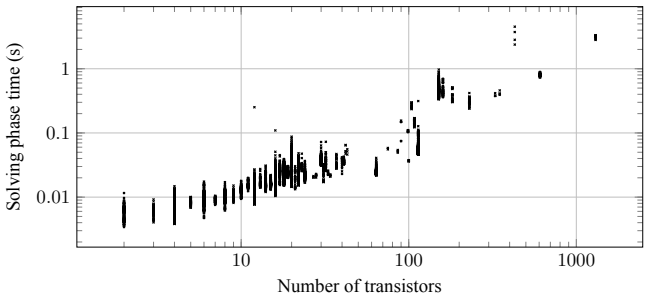

Fig. 5: Time spent in the solving phase for each analyzed case

Fig. 6 shows a box-plot of the time spent in each of the analysis steps. Given the relatively narrow interquartile range, we may say that $50\,\%$ of circuits instances are analyzed in the order of hundredths of a second. For more complex circuits,

the analysis may require a few seconds. Note that the static voltage propagation, used in REACHS (Equation E_{REACHS}), is not considered a contribution of this paper, therefore not taken into account. We use an unoptimized prototype but much faster industrial tools exist for this particular task.

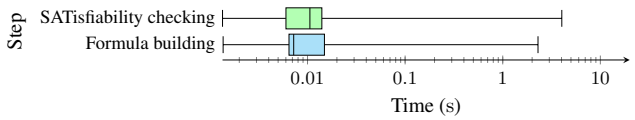

Fig. 6: Distribution of time spent in each of the analysis steps

C. Performance evaluation on synthetic benchmarks

To study the scalability of our approach, we run benchmarks on generated full-adder circuits of variable size. A 2-bit full-adder is composed of 9 NAND gates, and each NAND gate is made of 4 transistors (2 PMOS and 2 NMOS). The full-adder is therefore composed of 36 transistors. We generate N-bit adders as a composition of 2-bit full-adder blocks. Each block is assigned a random voltage value ranging from 1 V to $N-1$ V. Benchmarks consist in running the analysis to check for missing level-shifters, using the same experimental setup as in Section IV-B, where the checked formula is $\mathcal{F} \wedge \left(\bigvee_{M_x \in \mathbf{D}_{\text{PMOS}}} \mathcal{E}_{\text{MLS}}(M_x) \right)$ — with $\mathcal{F}$ being the circuit formula.

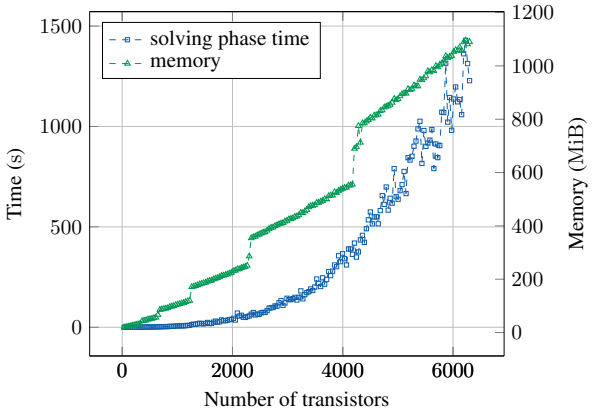

Fig. 7: Full-adder analysis benchmarking results

Obtained results demonstrate that our approach scales up to thousands of transistors (Fig. 7). It is however difficult to compare these performances against existing works, as our approach differs from the state-of-the-art. Yet similar metrics were studied in [12] on a different ERC problem, where a SAT solving based solution is proposed to check the possibility of short-circuits in single-supply designs. Experimental results of [12] show that for a 500 nets and 30 inputs circuit, search for short-circuits could be done in 3329 s. In our synthetic benchmarks, checking for level-shifters in a multi-supply 30-bit full-adder circuit containing 611 nets is done in only 19 s (among which 12 s is spent in the unoptimized static voltage propagation phase). Besides, our approach clearly separates the circuit encoding and the property (or error) of interest, which makes it applicable to other ERC problems including the analysis of short-circuits.

V. CONCLUSION AND PERSPECTIVES

In this work, we introduce a novel transistor-level semantics that makes it possible to use formal verification techniques to perform Electrical Rule Checking (ERC). The presented case-study shows the applicability of the approach on a particular error: missing level-shifters. As experimental results show, circuits with a few hundreds of transistors are analyzed almost instantaneously. Our approach is, therefore, convenient for analyzing small assemblies of basic circuit cells. Furthermore, circuits with thousands of transistors were analyzed in a reasonable amount of time and memory usage. The presented technique showed interesting results and has been implemented in the Aniah OneCheck solution. Future works include studying the applicability of classical techniques for scaling up, such as counter-example guided abstraction refinement, to our approach. Other future works include taking into account physical parameters of transistors (*e.g.,* threshold voltage V_{th}, width-to-length ratio W/L) and improving the transistor modeling to be less abstract, to provide more precise and quantitative results. This would notably be useful for reproducibility of error scenarios in simulation based setups.

REFERENCES

[1] K. Rupp, "Microprocessor Trend Data," https://github.com/karlrupp/microprocessor-trend-data.

[2] H. Foster, "Part 8: The 2022 Wilson Research Group Functional Verification Study," https://blogs.sw.siemens.com/verificationhorizons/2022/12/12/part-8-the-2022-wilson-research-group-functional-verification-study.

[3] B. Ferres, O. Oulkaid, L. Henrio, M. Khosravian, M. Moy, G. Radanne, and P. Raymond, "Electrical Rule Checking of Integrated Circuits using Satisfiability Modulo Theory," in *DATE*, Antwerp, Belgium, Apr. 2023.

[4] J. Lescot, V. Bligny, D. Medhat, D. Chollat-Namy, Z. Lu, S. Billy, and M. Hofmann, "Static Low Power Verification at Transistor Level for Soc Design," in *ISLPED*. California, USA: ACM Press, 2012.

[5] L. Lang, "Case Study: Power-aware IP and Mixed-Signal Verification," *DVCON (San Jose, Calif)*, 2010.

[6] K. Nichols, T. Kazmierski, M. Zwolinski, and A. Brown, "Overview of spice-like circuit simulation algorithms," *IEE Proceedings-Circuits, Devices and Systems*, vol. 141, no. 4, pp. 242–250, 1994.

[7] M. Zwerger and H. Graeb, "Verification of the power-down mode of analog circuits by structural voltage propagation," *Analog Integrated Circuits and Signal Processing*, vol. 78, no. 1, pp. 177–189, Jan. 2014.

[8] S. Blieck and E. Janssens, "Software Check for Power-down Mode of Analog Circuits," p. 4, 1996.

[9] M. Zwerger and H. Graeb, "Detection of Asymmetric Aging-Critical Voltage Conditions in Analog Power-Down Mode," in *DATE*. Grenoble, France: IEEE Conference Publications, 2015, pp. 1269–1272.

[10] J. Lescot, P. Dehan, W. Boujarra, D. Medhat, and S. Billy, "A Comprehensive ESD Verification Flow at Transistor Level For large Soc Designs," in *EOS/ESD*. Reno, NV, USA: IEEE, Sep. 2015.

[11] T. Grimm, D. V. Lettnin, and M. Hübner, "A survey on formal verification techniques for safety-critical systems-on-chip," *Electronics*, vol. 7, no. 6, Article 81, pp. 81–1 – 81–27, 2020.

[12] J. Afonso and J. Monteiro, "Analysis of short-circuit conditions in logic circuits," in *DATE*. IEEE, 2017, pp. 824–829.

[13] Randal E Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems," *IEEE Transactions on Computers*, Feb. 1984.

[14] M. Zwerger and H. Graeb, "Short-Circuit-Path and Floating-Node Verification of Analog Circuits in Power-Down Mode," in *SMACD*. Seville, Spain: IEEE, Sep. 2012, pp. 241–244.

[15] K.-H. Koo, J.-H. Seo, M.-L. Ko, and J.-W. Kim, "A New Level-Up Shifter for High Speed and Wide Range Interface in Ultra Deep Sub-Micron," in *ISCAS 2005*, 2005, pp. 1063–1065 Vol. 2.