# A Survey on Transistor-Level Electrical Rule Checking of Integrated Circuits

Bruno Ferres, Oussama Oulkaid, Matthieu Moy, Gabriel Radanne, Ludovic Henrio, Pascal Raymond, Mehdi Khosravian

▶ **To cite this version:**

Bruno Ferres, Oussama Oulkaid, Matthieu Moy, Gabriel Radanne, Ludovic Henrio, et al.. A Survey on Transistor-Level Electrical Rule Checking of Integrated Circuits. ACM Transactions on Design Automation of Electronic Systems, 2025, ⟨10.1145/3748327⟩. ⟨hal-05185119⟩

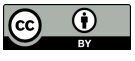
**HAL Id: hal-05185119**

**https://hal.science/hal-05185119v1**

Submitted on 24 Jul 2025

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons CC BY 4.0 - Attribution - International License

# A Survey on Transistor-Level Electrical Rule Checking of Integrated Circuits

BRUNO FERRES, Univ. Grenoble Alpes, CNRS, Grenoble INP‡, VERIMAG, 38000 Grenoble, France
OUSSAMA OULKAID, Université Claude Bernard Lyon 1, CNRS, ENS de Lyon, Inria, LIP, UMR 5668, 69342, Lyon cedex 07, France; Univ. Grenoble Alpes, CNRS, Grenoble INP‡, VERIMAG, 38000 Grenoble, France; Aniah, Grenoble, France
MATTHIEU MOY, GABRIEL RADANNE, and LUDOVIC HENRIO, CNRS, ENS de Lyon, Inria, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France
PASCAL RAYMOND, Univ. Grenoble Alpes, CNRS, Grenoble INP‡, VERIMAG, 38000 Grenoble, France
MEHDI KHOSRAVIAN GHADIKOLAEI, Aniah, Grenoble , France

Hardware verification is crucial to ensure the quality of Integrated Circuits, and prevent costly bugs down the manufacturing flow. *Electrical Rule Checking* (ERC) is a verification step used to assert that a circuit complies with some electrical rules, from the absence of short-circuits to dedicated constructor rules. In this survey, we provide a global overview of existing ERC techniques at transistor-level, where voltage values are explicit. We propose a new classification method to compare the existing approaches based on their semantic modeling of circuits. This survey precisely describes transistor-level ERC research challenges and existing solutions. We believe it will help structure this research domain by positioning existing approaches with respect to each other. Obviously, a survey should also facilitate technological transfer and this one should help CAD vendors identify the most relevant approaches to integrate in their tools. Finally, we highlight several promising directions to improve the existing solutions.

CCS Concepts: • **General and reference** → **Surveys and overviews**; **Verification**; • **Hardware** → **Electronic design automation**; **Design rule checking**.

Additional Key Words and Phrases: Electrical Rule Checking, Integrated Circuits, Electro-Static Discharge, Electrical OverStress, Static Verification

## 1 INTRODUCTION

Building an Integrated Circuit (IC) for a particular use case is a complex task that significantly relies on the expertise of hardware designers. During this process, they must consider both logic behavior and physical properties to produce electronic circuits that are suited for particular targeted environments. Indeed, while actually designing circuits is already difficult as it requires both expertise and time, it is even more difficult and time-consuming to verify that the circuit behaves as it should. As a result, most of the time required to build an IC is spent on circuit verification (50-60% in 2022 [14]), and easing this task is an efficient way to increase engineers productivity. More precisely, we are interested here in the verification of large ICs, like systems on chips (SoC) that can contain up to several billion circuits, where scalability of the approach is paramount.

In order to ensure the quality of the chips produced, verification steps must be taken at every level of abstraction in the flow. Functional properties are usually best checked at high level descriptions

‡Institute of Engineering Univ. Grenoble Alpes

Authors' addresses: Bruno Ferres, Univ. Grenoble Alpes, CNRS, Grenoble INP‡, VERIMAG, 38000 Grenoble, France, bruno.ferres@univ-grenoble-alpes.fr; Oussama Oulkaid, Université Claude Bernard Lyon 1, CNRS, ENS de Lyon, Inria, LIP, UMR 5668, 69342, Lyon cedex 07, France; Univ. Grenoble Alpes, CNRS, Grenoble INP‡, VERIMAG, 38000 Grenoble, France; Aniah, Grenoble, France, oussama.oulkaid@univ-grenoble-alpes.fr; Matthieu Moy, matthieu.moy@univ-lyon1.fr; Gabriel Radanne, gabriel.radanne@inria.fr; Ludovic Henrio, CNRS, ENS de Lyon, Inria, Université Claude Bernard Lyon 1, LIP, UMR 5668, 69342, Lyon cedex 07, France, ludovic.henrio@ens-lyon.fr; Pascal Raymond, Univ. Grenoble Alpes, CNRS, Grenoble INP‡, VERIMAG, 38000 Grenoble, France, pascal.raymond@univ-grenoble-alpes.fr; Mehdi Khosravian Ghadikolaei, Aniah, Grenoble , France, mehdi.khosravian@aniah.fr.

Author Version

like *Register-Transfer Level* (RTL). Electrical properties usually do not make sense at this level where voltage values are abstracted as logical values. However, complex circuits are composed of multiple power-domains with different voltages, which implies extra-functional properties to be checked on the circuit. For example, connection between different power-domains must be protected with so-called *level-shifters*. Their insertion can be automated [22], but human errors are possible in the configuration of automatic tools or in manually written parts of the design. Checking for missing level shifters cannot be done at RTL level, and requires going down to the transistor-level, which is the highest level of abstraction where voltages are explicit. More generally, in this survey, we consider a particular step of this verification flow, called *Electrical Rule Checking* (ERC).

In particular, we discuss ERC approaches that consider circuits topologies to perform analysis, and that can scale to analyze SoCs as a whole, or at least at a large scale (*i.e.,* from thousands to billions of transistors). This kind of checks can be designated as *Transistor-Level Electrical Rule Checking* (TL-ERC), as they operate at transistor-level to assert that the circuit is compliant with some electrical rules, which should protect it against particular issues. Section 2 details these electrical rules, and how issues can be introduced and detected. Consequently, we do not discuss verification approaches that aim at verifying complex properties, such as temporal properties, on the circuit. Neither do we discuss approaches that rely on simulation to model the exact behavior of the circuit. Indeed, simulation can provide a very precise model for small circuits but cannot scale to the billion component scale required for modern SoCs. The exact scope of this survey is further detailed in Section 3.

TL-ERC groups a large variety of techniques, tackling different electrical issues, with no common classification. This makes comparison of existing approaches difficult, and thus impede both the understanding and the improvement of such approaches.

In particular, while TL-ERC techniques are widely used in the industry (see Section 7 for an overview of industrial tools), as such checks are mandatory in the flow to avoid heavy additional costs after manufacturing, the research domain (in particular from the academic side) seems to be quite unorganized, and would benefit from a federated approach to highlight similarities in the various use-cases that exist in the literature. Moreover, it is also important to remark that TL-ERC is a domain that focuses in identifying issues in circuits that may be produced by different design flows. While some of those flows might ensure the absence of electrical errors using correct-by-construction approaches (*e.g.,* pure CMOS circuits), other design flows might rely on manual intervention and/or more complex electrical structures, resulting in circuits that might exhibit vulnerabilities after the design process. In this way, we are aware that most of the circuits should not exhibit electrical errors. However, for the remaining circuits, and independently of the design methodology that was used, it is important to provide guarantees that the circuit is safe, and to do so in a convenient way (*e.g.,* avoid raising too much false alarms, and provide meaningful reports for the engineers [2]). Such approaches should be usable as early as possible in the flow, because any bug found and fixed late in the flow requires re-running exhaustive and computationally intensive simulations to avoid regressions. Using early verification approaches also enables iterating more efficiently on the circuits, before running sign-off simulations at the end of the flow: ideally, such simulations should be run with a strong confidence in the fact that no electrical issues remain in the designs, in order to avoid time-consuming iterations at the end of the flow. Fast and fully automatic TL-ERC checks can even be run in continuous integration (*i.e.,* after each commit or push) during development [3].

The aim of this survey is hence to provide a global overview of the existing techniques for TL-ERC, their principles, their strengths and weaknesses, the properties they are able to ensure, and their ability to analyze large circuits. We also provide insights on how existing approaches could

Author version

be combined and enhanced to improve TL-ERC tools and allow them to check more properties and larger circuits. The main goals of this survey are:

- Give more visibility to the TL-ERC research challenges and existing solutions. Our experience is that researchers from the circuit verification communities are often unaware of the verification issues that appear at transistor-level, while researchers from the low-level hardware communities lack familiarity with static analysis techniques that could be used to verify their circuits. We want to foster collaborations between these research communities with a survey explaining both the problems and existing approaches.
- Structure the research domain of TL-ERC. The research approaches we consider in this survey are very diverse, and are often published without citing each other. By summarizing many approaches in a single document, and by proposing a new classification, we hope to help researchers working on TL-ERC position their work with respect to other existing approaches.
- Facilitate technological transfer to CAD tools. Many approaches presented in this document were developed either internally by circuit designers, or by academic researchers, but were not transferred to industrial tools. This survey should allow developers of circuit verification tools to identify the most relevant approaches to integrate in their tools.

As a side effect, the survey may also help circuit designers and verification engineers understand the state of the art in TL-ERC.

This paper is organized as follows. Section 2 introduces electrical rule checking, including the errors and properties of integrated circuits that we are interested in, and the integration in the design-flow. Section 3 delimits the perimeters of the research contributions we consider and how we classify them. Sections 4, 5 and 6 respectively compare the existing approaches for TL-ERC that correspond to the different classes of approaches that we discuss. Section 7 gives a rapid overview of the industrial tools that can be used to perform TL-ERC at transistor-level, and Section 8 concludes this work, presenting various perspectives that we consider promising for TL-ERC, and how they could be integrated in the community.

## 2 CHECKING ELECTRICAL PROPERTIES: DEFINITIONS AND PRINCIPLES

TL-ERC consists in verifying properties on a transistor-level description of the circuit, where power supplies are properly described. Properties involving voltage values can not be checked at higher level of abstraction, and would be far more difficult to check at lower-level, such as layout.

A transistor-level description of a circuit (also known as schematic-level) is a set of *devices* (transistors, diodes, resistors, etc.), interconnected with *nets* (or wires), which forms a *netlist*. Devices are connected to nets through several *terminals*, such as the source, drain and gate of a transistor. Some nets are internal, others are external *pads* (sometimes also called pins of the circuit), connected to the outside world, either *power supplies* (including the ground, which can be considered as a specific kind of supply), or inputs and outputs. A set of devices and nets connected to the same non-ground supply composes a *power domain*.

### 2.1 Properties of Interest

In order to apprehend the different approaches to TL-ERC, we define several properties of interest for netlist verification [3]. We are interested here in the properties that are related to the topology of the circuits being analyzed (including their power domains), and to specific electrical rules that the circuit must comply with to prevent damages and misfunctioning. Pure logical properties are out of the scope of this survey as they can be verified without low-level implementation aspects

Author Version

like power supplies and the precise netlist; this kind of properties can typically be verified at a higher level of abstraction such as Register Transfer Level.

Stating properties of an electrical system requires taking into account the different electrical states that a net can take, and reason on an abstraction of this set of states. The state of each net should relate to the different supplies of the circuit. A *connection* can be defined as an electrical path of low resistance, such as one made of wires, active transistors, diodes or resistors of low resistance. Two nets of a circuit are not connected if there is no path of low resistance connecting them, *i.e.,* every path between them is highly resistive. Of course the notion of high or low resistance paths depends on the context. Based on these simple notions of nets, states, and connectivity, we consider properties corresponding to the absence of the following potential faults.

*Floating nets.* A net is considered *floating* (or "in a *high impedance* state") if it is not connected to any supply. Note that a net can be floating for three distinct reasons. The first two ones come from the netlist itself: the net is not *physically* connected to a supply. This can happen because there is no wire leading to any supply, or because such wires cannot force a value on the net (*e.g.,* a net connecting only the gates of two transistors — no voltage can be forced on it, as voltages are only propagated from source to drain or drain to source in transistors). Such nets will always be floating, independently of the electrical state, and can be considered a *static floating net.* Static floating nets in the design can straightforwardly be found with a reachability analysis of the netlist. They can also be the result of a manufacturing open-circuit fault, in which case they have to be detected by testing on each individually manufactured chip [61] and cannot be found by ERC. Such testing techniques are therefore out of the scope of this survey.

In contrast, a net is *dynamically floating* when the circuit electrical state is such that every path from the net to any supply crosses a device in highly resistive state. In most cases, a floating net is an error. Connecting a floating net to the gate of a transistor leads to non-deterministic behavior. Simulation results in the presence of floating gates are often not trustworthy as devices are operating in the sub threshold region, which is often not modeled accurately enough by the available device models [64]. The actual behavior of the circuit is technology-dependent and even layout-dependent [48]. Floating nets are therefore both undesirable in the final circuit, and hard to spot or debug. However, dynamically floating nets can be legitimate in particular circumstances — *e.g.,* high impedance states are commonly used to build communication buses.

*Short circuits and leakages.* A net is in *short circuit* state if it is connected to two (or more) different supplies, leading to a continuous path of low resistance between two supplies. A path of almost null resistance is clearly an error — *i.e.,* an undesirable short-circuit — but in some contexts a path of medium resistance implying some leakage current may be used on purpose, *e.g.,* to propagate information through electrical current instead of voltage level. A leakage — *i.e.,* an electrical current between two supplies, which increases the power consumption of a circuit — can thus be either legitimate or illegitimate, based on the *intention* of the designer.

Both short-circuit and floating properties of the nets can lead to damages and/or malfunctioning of circuits. It is hence required to check for the absence of such faults during TL-ERC [1, 62]. However, since both can be legitimate in particular cases, properties to be checked on the circuit are usually more specific than the complete absence of both. For example, one could prove that only some nets of interest (such as transistors gates) cannot be floating, or that no leakage is possible at the interface between power-domains.

*Missing level-shifters between power domains.* When it comes to complex circuits with multiple supplies, more complex properties might be required. One typical case of potential failure of a circuit appears at the interface between power domains. If a device has its terminals connected to

Author version

different power domains, it might cause misfunctioning of the circuit. For instance, current can leak through a transistor at the interface between power domains, as the voltage applied on their gate might cause them to be partly conductive. To prevent this problem, specific components, known as *level-shifters*, should be used at these interfaces [29], and a proper verification methodology should assert that each interface is correctly protected [30, 64].

*Over-voltage on thin oxide.* The technological details of the devices used in the circuit may also require additional verification. For example, when thin oxide transistors are used, one needs to assert that the voltages applied on the terminals of those devices cannot damage them. This particular fault, called *over-voltage on thin oxide*, requires a proper analysis to identify the vulnerable devices, as well as the potential voltages applied to their terminals [30].

*Electro-Static Discharges (ESD) and Electrical OverStresses (EOS).* Other vulnerabilities of the circuit may not come from their design, but from external events which may cause damages if not properly handled. For example, circuits can be exposed to *Electro-Static Discharge* (ESD) events, where undesired current flows within the devices. This can happen when an excessive voltage is applied on the pads of the circuits, typically because a human touched the device (known as the *Human-Body Model*, or HBM), or when a device builds up internal voltage and release it in the circuit (known as the *Charged-Device Model*, or CDM). Moreover, ESD events are becoming even more critical with the development of new manufacturing technologies, as reducing the size of the transistors also reduces the breakdown voltage for ESD [32]. Specific protections must be introduced in the circuit to prevent induced damages, should an ESD event occur, and it is thus necessary to check that critical parts of the circuits are protected against ESD, to ensure their robustness [33, 59].

*Electrical OverStress* (EOS) is a similar kind of electrical event that may damage circuits. Electrical OverStress occurs when undesirable voltage or current is applied to a given component for a longer period of time. Some design rules can be used to ensure protection against EOS, and should once again be verified to improve reliability [21].

ESD and EOS events are critical to ensure the robustness of circuits. The definition of design rules that can prevent such events and the verification of the presence of protections against ESD and EOS constitute a wide domain of research [11]. Moreover, those checks must be performed at various steps during the design flow, from cell level to full-chip level [25]: some of those verifications require considering very low-level details of the circuit (*e.g.,* physical characteristics of the devices), while other approaches might only consider the topology of the circuits under verification.

*Power-down modes and their consequences.* While the properties introduced previously are interesting to check in a general context, other verifications may be required, depending on the context of use of the designed circuits. A highlighting example can be found with the integration of a *power-down mode* in a given circuit. Such functioning mode makes it possible to save energy by dynamically cutting power supplies in parts of the circuits when these parts are not used, by adding extra-functional circuitry to the initial design. However, with such mode comes new verification challenges: modifying the circuit to add this functionality can induce new errors, such as new short circuits and floating nets. It can also impact some crucial properties of analog circuits, such as *symmetry* [10, 27, 35]. The absence of symmetry in an analog circuit can cause aging [64], reducing its lifespan and increasing the risk of failures. Ensuring symmetry properties when adding a power-down mode to a circuit is hence required to improve its robustness.

*Beyond transistor-level properties.* In this survey, we focus on the verification of properties directly related to the transistor-level. However, the verification of some properties linked to other levels of abstraction may require to also consider the transistor-level to be performed efficiently. In particular,

Author version

some verification approaches combine *transistor-level* and *layout level* verifications to improve the feedback for users, with approaches such as *Logic-Aware Physical checks* [31], *Logic-Driven Layout Analysis* [16, 26] or *Design Rule Check driven geometrical checks* [21]. Those approaches focus on verifying some *geometrical properties* of the layout, but uses information extracted from the transistor netlist to identify the area where geometrical checks should be performed. This kind of approach demonstrates that combining information from different levels of abstraction can help providing meaningful verification tools. However, as will be stated in Section 3, in this survey we focus on *transistor-level verification approaches*; we thus do not study such combined approaches.

## 2.2 Verification at Different Levels of Abstraction

Checking the logic properties of a circuit can usually be done at a high level of abstraction, regardless of the exact implementation of state-machines and basic cells such as gates and registers. These properties can typically be checked on the RTL description [5, 18, 24], or even at a higher-level [13, 55]. The verification at RTL level is a widely explored domain: it can rely on different techniques, such as high-level simulations, or state-space exploration techniques that can reason directly with more abstract concepts, *e.g.,* exploring the states of a Finite State Machine without considering the detailed implementation. This question has hence been widely addressed in the literature, with methods and tools enabling the verification of a lot of properties — including temporal properties — from hardware descriptions written in languages such as VHDL or Verilog. This can be used to verify that a device is compliant with a given specification, by proving properties on the behavior of the system such as the reachability of a given state. The correctness properties should then be preserved when synthesizing the netlist from the high-level description.

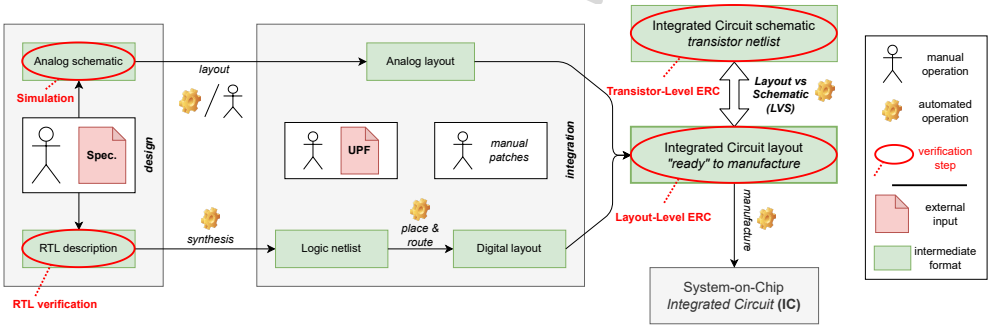



Fig. 1. High level view of a typical design and integration flow including both digital and analog parts: a particular focus is put on *transistor-level verification*.

## 2.3 Integrated Circuit Design Flow

However, the verification task should occur at different steps in the design process of an IC, considering different levels of abstraction to verify that the circuit can be sent to manufacturing, not only from a functional point of view, but also concerning extra-functional aspects — *e.g.,* verifying some electrical rules to ensure that there is no current leakage. Nevertheless, many different design flows can be used to produce a circuit from a set of specifications, depending on the usage of analog and digital parts, the target technology, the design methodology, or even the company expertise — and the verification steps must be adapted to these different design flows. These flows are complex, and any error introduced in the design can result in heavy costs to fix the circuits after manufacturing, making it important to check the circuit properties all along the flow.

Author version

An example of a typical design process is shown in Figure 1. In this flow, the circuits are built and assembled by expert designers, from a combination of RTL descriptions and analog patterns, derived from a specification of the target circuit. It results in a global layout of the circuit to be sent to the foundry for manufacturing. Important verification steps are highlighted in red, they occur all the way from high-level representations to low-level details, as manual interventions may occur at every step of the flow (see Section 2.4).

In such process, the concept of "netlist" is used at different levels of abstraction: logic- and transistor-levels. As a matter of fact, a netlist is only a description of the interconnections between base components, where these components can be logic gates, transistor devices, or any other relevant component depending on the level of abstraction.

When considering logic netlists, the base components are logic gates, which can be mapped to *standard cells*: for a given technology of manufacturing, the foundry provides a list of primitives (the so-called standard cells), each physically implementing a logic gate or a set of logic gates into a layout. These standard cells map a logic representation to a set of characteristics — such as layout size and path delay — to be used in the manufacturing flow. This enables the efficient realization of the *place and route* step, *i.e.,* generating a circuit layout by physically mapping each logic gate to a set of physical transistors — which are basically sets of geometrical shapes on the circuit layout — at a particular location on the chip, and adding the wires to connect them.

On the other hand, at the end of the design flow, we need to reason at low-level of abstraction and consider transistor-level netlists. Indeed, before running the manufacturing step, the circuit layout should be checked, especially to ensure that some design rules — required by the foundry — have been respected in the design process. However, circuit layouts are representations built to fit the needs of semiconductor manufacturing processes, and are hence not adapted to the verification of some functional or electrical properties on it. To cope with this issue, a usual step of the verification flow is to consider both the circuit layout and its schematic in a step called *Layout vs. Schematic*[1] (LVS). Doing so, some properties can be checked on the schematic representation, and subsequently ensured on the layout representation. This schematic representation is actually the *transistor-level netlist.*

### 2.4 Sources of Errors in the Flow

For digital designs, the complete flow from RTL, *i.e.,* a "synthesis" step followed by "place and route", is potentially automatic. RTL can either be handwritten or generated from a high-level synthesis tool. When this is the case, no manual intervention can introduce bugs in the flow. However, in practice, many human interventions may be needed in the flow, to go from the design phase to the actual layout. For analog designs, the transistor-level netlist is usually generated from a manually designed schematic, hence it might contain functional and/or electrical human-introduced errors. Moreover, as illustrated in Figure 1, complex Systems-on-Chip are often composed of both digital and analog parts, which can either be custom-made, or external IPs. In this context, the process of integrating the different parts can also introduce errors, even if each individual subcircuit is correct. In particular, such integration requires the designers to explicitly consider the power domains of each component to properly interface each part, and this must be thoroughly checked to avoid errors that can result in severe damages on the produced chips [30, 59]. While the integration of power-domains can be automated, in particular with tools based on the *Universal Power Format* (UPF), the process remains complex, allowing the user to customize the integration. Figure 1 shows a simplified version, more detailed versions can be found in *e.g.,* [17]. Doing so, the integration

[1]Sometimes also called *Layout vs. Source* in the literature.

Author version

phase might introduce errors when the tools are mis-configured, or when the UPF source file is erroneous.

Last but not least, even a generated transistor-level netlist may be manually modified for various reasons. For example, the detection of a bug late in the flow may require to patch the generated netlist, as patching the initial RTL and starting again the flow from scratch would be too costly. Such late patches have led to the apparition of engineering processes, such as *Engineering Change Order* (ECO) [9], which can be used to fix bugs very late in the flow, adjusting the circuit to a slightly modified functioning environment or modifying the interface between the components. This may also introduce new bugs.

### 2.5 Preventing Electrical Issues

Preventing electrical issues as the ones mentioned above is indeed crucial to ensure that circuits are reliable. As this problem is critical in the manufacturing domains, solutions have been used for years to gain confidence in the reliability of the circuits, before the manufacturing process ends.

As far as we are aware of, there exist three different (and complementary) approaches to ensure that no electrical issues are left in a design [7]:

(1) Low-level (SPICE) simulations covering all interesting cases. With the growing complexity of circuits, the number of configurations to simulate grows exponentially with time, and full coverage in simulation is no longer achievable.
(2) Architectural constraints on the circuit, imposing each sub-circuit (usually called IP) to include the necessary protections. This leads to needlessly duplicate some parts of the circuit, and is hardly achievable when the IPs themselves include complex power-management logic for example.
(3) Silicon debug tools, where electrical issues are measured physically on a first prototype. While post-silicon validation is anyway required, waiting for this stage to debug electrical errors is only applicable when at least two tape-outs (*i.e.,* masks manufacturing) are planned, which is less and less manageable with the increasing cost of mask sets [45]. Obviously, this should only be used as a last resort solution.

### 2.6 Transistor-Level Electrical Rule Checking

All these reasons motivate the need for efficient verification processes at the transistor-level, especially to verify properties that cannot be checked at a higher level of abstraction. The transistor netlist can be extracted automatically from the layout (LVS step in Fig. 1), hence properties checked on this netlist also hold on the layout. Among these properties, the electrical properties of the netlist are crucial to avoid misfunctioning of the chips or damaging them.

In this survey, we call this particular verification step *Transistor-Level Electrical Rule Checking* (TL-ERC), *i.e.,* ensuring that a design is compliant with a set of electrical rules. Such rules, which can be checked by considering the topology of the netlists, can vary from simple and naive ones (*e.g.,* "there is no wire between two supplies") to complex statements (*e.g.,* "this part of the chip includes a protection against ESD"), resulting in a need for various verification tools. Moreover, for complex circuits such as Systems-on-Chip (SoC), TL-ERC must be performed at the scale of billion transistors, which requires a trade-off between the exhaustiveness of the verification, and its speed. Such checks are difficult and may take time but, as mentioned earlier, the cost of fixes after circuits have been produced by the foundry [58] are high enough to make these verifications valuable.

Compared to solutions mentioned in Section 2.5, TL-ERC is not meant to replace simulation completely, as simulation is anyway needed to check other properties. It can however reduce considerably the number of simulations to be run in the complete design flow, while keeping a full

Author version

coverage with respect to some properties. It also makes it possible to lift the constraints imposed on the architecture, giving designers more freedom to optimize the circuit finely, instead of solely relying on protected IPs. Finally, TL-ERC reduces the risk of finding issues during post-silicon validation, hopefully allowing the first tape-out to be correct.

The various approaches for TL-ERC that exist in the literature tend to be extremely specific for a particular application, and the positioning is usually focused on the applicative domain, and do not compare with the existing approaches for seemingly different problems. Indeed, some aspects of the existing verification solutions depend on the circuits' functionality, context of use, and reliability. However, in this work, we aim at providing a unified vision of the TL-ERC problem, comparing verification approaches that may differ on their applicative context, but use similar techniques and models. Crucially, in order to compare existing methodologies, we do not classify on faults. Instead, we classify the approaches for verification tasks based on the level of abstraction used to represent transistors. Our classification criteria are precisely defined in Section 3.2.

## 3 SCOPE AND METHODOLOGY OF THE SURVEY

### 3.1 Scope of the Survey

As mentioned in the previous section, standard design flows are complex processes that operates at various abstraction levels, from a high level specification to the physical manufacturing of the resulting chip, and verifications must be performed at various steps of these processes to ensure the quality of the final product.

RTL verification is a well-known verification step that has been widely discussed in the literature [5, 24]. Nevertheless, it cannot be used to verify electrical properties, which usually requires information on power supplies that are not yet present at RTL level. For all these reasons we do not discuss RTL verification in this survey.

Another approach for circuit verification that is frequently used to model electrical properties is to use simulation, which allow designers to ensure the quality of a device for specific electrical scenarios, by providing *input vectors*. This makes it possible to model details of the electrical behavior of a circuit in an accurate way via analytic models rooted in physics principles — *e.g.,* to analyze the leakage of a given transistor in a given electrical state. However, simulation-based approaches are very costly in terms of computation needs, meaning that they cannot scale to full-chip analysis [50]. Moreover, as these approaches rely on input vectors to make the electrical scenario explicit for the verification task, they lack exhaustiveness: a simulation is usually run for every input vector, and the only way to be exhaustive is hence to consider every possible vector in the simulation plan. Naturally, this grows exponentially with the number of inputs and is intractable in realistic use cases at industrial scale. Last but not least, in some specific cases, the model used to simulate the behavior of a circuit may not be accurate under certain assumptions: for example, Zwerger et al. state that, in power-down mode, the behavior of transistors may not be modeled as accurately as it is in "normal" functioning mode [64]. For all of those reasons, we will not discuss simulation-based verification approaches in this survey.

As a matter of fact, both RTL and simulation-based verification processes highlight an inherent challenge of the verification of any complex system: there is an implicit trade-off between the accuracy of the verification tools, and the complexity it requires. The more a verification method uses a realistic model to approximate the real behavior of a system, the more complex is the verification process, hence impacting the scalability of the approach. For example, to correctly model the behavior of an IC, simulation-based models usually rely on differential equation systems to simulate the behavior of the components and their interactions, and this approach indeed requires more computational resources than higher-level abstractions, such as RTL. On the other hand, by

Author Version

abstracting the low-level details, RTL verification methods can be used to verify more complex properties, such as the temporal behavior of a state machine, as the inherent model is simpler. Both RTL verifications and simulation-based methods are considered out of the scope of this survey.

This work focuses on verification processes that operate at transistor-level to verify electrical properties on the circuits. As stated by Zwerger et al. [64], such verification tools are growing in popularity, with initiatives to detect issues like floating gates, leakages, electrical overstress at thin-oxide devices or missing level shifters at the border between voltage domains [20, 56]. In particular, these processes must consider not only the transistor netlists of the circuits, but also the various power supplies that are driving them, in order to model electrical behavior in an accurate way. This survey covers all the design methodologies that, as long as they only consider the components that were introduced in Section 2 (transistors, diodes and resistors). A particular focus will be put on the handling of hierarchy in each approach, as it is a practical way to scale up verification processes to full-chip analysis, by benefiting from the modular design of the circuits.

This survey also covers some of the approaches used for ESD verification in the literature. In particular, considering the ESD approach taxonomy as exposed by Trivedi et al. [57], static ESD checks can be performed either at transistor-level or at layout level [15]. While the layout level is out of scope here, some of the static ESD checks fall in our scope. They are called *topology checks* and consider the transistor topology and their patterns to identify vulnerabilities to ESD events.

To summarize, in this survey, we consider a specific kind of verification that operates at transistor-level, called *Transistor-Level Electrical Rule Checking* (TL-ERC). TL-ERC is used to check that some electrical rules are followed on a given netlist; these rules avoid some well-identified misfunctioning and/or damage causes that were described in Section 2. In the next sections, we detail existing approaches for TL-ERC, explain their principles, and compare them.

### 3.2 Classification

As the goal of this article is to provide a comprehensive analysis of the different approaches that exist for Transistor-Level Electrical Rule Checking in the literature, it is important to propose adequate classification criteria for comparing the contributions.

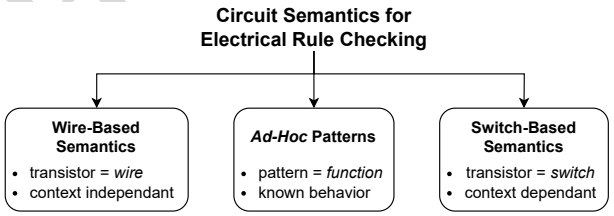



Fig. 2. Proposed taxonomy for Transistor-Level Electrical Rule Checking (TL-ERC) approaches, based on the semantics used to model the components' behavior.

The different approaches presented in this survey are split with respect to how they model the semantics of the circuit they analyze. More specifically, we consider three different semantics categories, as introduced in Figure 2.

**Wire-Based Semantics** (Section 4) model transistors as wires: the analysis considers that current may flow between drain and source independently of the voltage applied to the gate terminal. This is a coarse approximation of the circuit's real behavior, but its simplicity has benefits for TL-ERC, notably scalability.

Author Version

**Pattern recognition approaches** (Section 5) aim to identify, in a given circuit, some structures with a specific behavior (for example, correct interfaces between power domains). Instead of modeling the behavior of each component in the circuit, these approaches only focus on some identified structures and thus only model parts of the circuit. It can be used for specific behaviors which require details on the low-level implementation and can not be modeled precisely by combining the individual behaviors of the components.
**Switch-Based Semantics** (Section 6) model the switching behavior of transistors, making it possible to model more precisely the voltages at each net of a circuit. In these approaches, a transistor is modeled either as a wire — *i.e.,* is *switched on* — or as an open circuit — *i.e., switched off* — depending on the voltage applied to its gate.

In each of these sections, we present the different techniques that have been proposed for TL-ERC, not only their functioning principles, but also the several criteria that can be used to compare them. In particular, we will consider the following comparison criteria: *properties and errors* that are considered, *usability* of the approach on realistic use cases, *scalability* of the methods, and identified *limitations*.

## 4 WIRE-BASED SEMANTICS APPROACHES

This section discusses approaches to the TL-ERC problem that use a simple abstraction of the transistors' behavior, by modeling them as wires.

*Principles.* The main shared characteristics of these approaches is the abstraction that transistors can always be switched on, no matter the voltage applied to their gate. In other words, a transistor is modeled as a wire between the source and drain terminals. Even though naive, this approach can be used to analyze all the possible scenarios of voltage propagation in circuits, by considering the reachability of the circuit nets in a graph formed by those wire-like transistors. However, such approximation also leads to a lot of "false alarms", as the considered electrical states of a circuit are a coarse over-approximation of the actually reachable states. For example, the wire-based abstraction of an inverter considers that it may connect the supply to the ground. The actual circuit, however, contains two transistors in series that cannot be "on" at the same time in normal conditions.

*Analyses.* Performing an analysis with a wire-based semantics is often called *voltage propagation* (or *supply propagation*), and mainly relies on traversing a graph that is an abstraction of the circuit to annotate every net with the possible supplies that can reach it [30, 63]. If a given supply can reach the source (resp. drain) of a transistor, then voltage propagation considers that it can also reach its drain (resp. source). Hogan et al. [21] use a similar approach in one of their analysis modes, namely *vectorless mode*, where no input vector is provided to the tool. Their other approach to supply propagation, called *vectored mode*, is discussed in Section 6. Srinivasan et al. [54] demonstrate how a property can be propagated throughout all the resistors in a hierarchical SoC, without flattening its structure thanks to a symbolic algorithm. The kind of properties they study are path-dependent, such as point-to-point resistance in a circuit. As resistors are simple dipoles, it is modeled as a wire in their approach, making it similar to the wire-based TL-ERC approaches.

Note that other approaches use supply propagation with different semantics (see Section 6). In the following parts, we will hence denote the process of propagating voltage information through transistors independently of the voltages applied to their gates as *naive voltage propagation*.

After a naive voltage propagation step, basic TL-ERC analysis can be performed by traversing the annotated graph and use voltage propagation information to check some connectivity properties on the nets (short-circuit, floating nets, . . . ).

Author Version

Several techniques rely on custom type systems to propagate more information on the nets' connectivity, such as stating if the net is a drain, a source and/or a gate [33, 59]. In particular, it can be used to identify power domain interfaces, where specific design rules must be used to protect the circuits — notably against ESD events — or ensure a correct behavior — *e.g.,* with level-shifters. Such interfaces can be identified using a type system, with the following rule: if a transistor's source and drain have voltage $v_1$ its gate has voltage $v_2$, then this transistor is on a power domain crossing interface. For example, Viale et al. [59] use a specific net type propagation algorithm as a prerequisite for further analysis of ESD paths. They use naive voltage propagation to initialize the nets in their representation of the circuit with specific types and voltage labels, which can then be used for further processing. Custom type system can be used for other checks, as the nature of a net (*i.e.,* source, gate, net, . . . ) has to be considered in error scenario. For example, if a net is not a gate, it can not be concerned by a *floating gate* error.

Another example of wire-based semantics approach can be found within the graph-based approach for ESD vulnerability identification from Liu et al. [32]. In this approach, hierarchical circuits are modeled as weighted graphs, where weights are used to define current paths between I/O pads, which may lead to ESD vulnerability. The proposed algorithm can hence be used to identify the pad-to-pad paths to protect against ESD events, considering that only a small fraction of pad-to-pad paths are vulnerable to such events (*i.e.,* one only needs to protect paths where there are few MOS transistors, as the other paths are not vulnerable to ESD events due to their higher resistance).

*Properties and errors considered.* Several properties may be considered with wire-based semantics, which can be seen as a fast and reliable first filter to eliminate parts of the circuit where no error can occur.

Lescot et al. [30] propose an integrated solution to identify various electrical errors, namely floating MOS gates, over-voltage on thin-oxide transistors and missing level shifters. Their approach is based on supply detection — using a heuristic on bulk-connections to identify the supplies at the different levels of hierarchy — and naive voltage propagation, demonstrating that naive voltage propagation can be useful in many use cases.

Other approaches demonstrate the usability of this abstraction for EOS ([21] in *vectorless mode*) and ESD identification [32, 33, 54], even though most approaches rely on a refining step — called *pattern recognition* — to filter out "false alarms" from real ones. Naive voltage propagation is hence mainly used as a first step for other approaches that will be detailed more thoroughly in Sections 5 and 6.

*Usability.* The abstraction at stake in the wire-based semantics approaches is quite naive, and is hence rarely used as a standalone method. Instead, it is generally used as a first step of further analyses — see Sections 5 and 6 for a comprehensive study of how *naive voltage propagation* can be used in more advanced TL-ERC techniques.

*Scalability.* However, performing naive voltage propagation on an industrial-scale circuit — which can include up to billions of transistors — is an expansive task, which needs to be efficient to be integrated in standard toolchains. Those approaches led to the development of industrial tools to perform this task, such as Calibre PERC [51], which can be used for supply propagation [16, 26, 30], and also provides *pattern recognition features*, that will be introduced in Section 5. Calibre PERC is designed to handle circuits hierarchy (and in particular hierarchical supply propagation), in order to scale efficiently to realistic industrial designs.

*Limitations.* Wire-based semantics are a way to over-approximate the behavior of an integrated circuit in a coarse-grain manner. This abstraction identifies more or less trivial errors in a given netlist and as such its results may include many false alarms.

Author version

Overall, wire-based analysis raises too many false alarms to be usable as a standalone approach — as its results must be filtered out by experienced designers. However, because it is simple, this approach scales and can be used as an effective prerequisite for other more precise analyses.

## 5 AD-HOC PATTERNS APPROACHES

A classic complement to a wire-based semantics is to rely on *pattern recognition*, which aims to ensure some properties of a circuit by identifying specific patterns within it.

*Principles.* The main principles behind the ad-hoc pattern approaches is to consider specific blocks of the circuit which presents interesting properties with respect to TL-ERC. Those approaches are generally used to add behavioral information to wire-based semantics. For instance, a pattern can be used to identify existing circuitry for ESD protection on a path that was identified as vulnerable by a previous wire-based analysis. Patterns can be provided by both the designers of the circuit and the engineers in charge of the verification task, and can be used to specify the behavior of specific blocks with respect to the errors models under analysis. These approaches hence rely on pre-defined pattern libraries, which provide all the variations of the blocks to identify to ensure the given property. These libraries must be provided by experienced engineers, with sufficient warranties on the properties of each pattern. In order to identify these patterns in the circuit, the different approaches use *graph isomorphism* algorithms to detect pre-selected patterns in a given circuit [43, 46].

Pattern-based approaches allows checking properties that is difficult to build by considering each component individually, instead the behavior of the identified patterns is based on designers' expertise and time-tested design rules.

*Analyses.* Ad-hoc pattern approaches are most of the time used to check extra-functional considerations, which does not act on the circuit's functionality, but are design rules that shall be used to ensure a correct functioning and prevent damages. Typically, ad-hoc pattern approaches can check that a specific feature exists in a given circuit — *e.g.,* a protection pattern against ESD events or a level-shifter topology.

Combining a first analysis based on wire-based semantics with a pattern-based refinement step is hence generally a good approach, as it is a way to filter out potential errors that have been identified with voltage propagation. Pattern-based analyses are often used to improve the wire-based semantics of a circuit with some more accurate details for specific parts, based on some expertise-based libraries of patterns.

*Properties and errors considered.* Pattern recognition-based approaches are used to identify level shifter topologies, typically to avoid raising a false *missing level shifter* alarm when naive propagation techniques raise a warning within a level-shifter component, at the interface of two power domains [30]. Patterns are also used to identify ESD protections, in the context of ESD vulnerability analysis [15, 30, 31, 33, 57]. In particular, those approaches focus on verifying that cross-domain interfaces (*i.e.,* transistors for which gate and source/drain terminals belong to different power domains) are correctly protected against ESD events, as such interfaces are particularly vulnerable. These first approaches essentially consist in asserting electrical rules based on the existence of relevant patterns.

Protection against ESD deserves a particular discussion. On one side, ESD protection is a canonical case for pattern matching as ESD protection circuits are quite standard circuits that provide a local protection at very specific points. We thus clearly consider them as a particular form of TL-ERC. On the other side, while checking the presence of ESD protections can be relatively easy using pattern-matching, identifying the places of the circuit where such protection is needed is

Author version

more difficult: ESD protections are usually added *in parallel* to the path to be protected, to give an alternative, easier path to high-voltage discharges. To protect against any external ESD (see *Human-Body Model* in Section 2.1), one protection should be needed for any pair of external pads (I/O or supplies) of the circuit [59]. However, some ESD protections should also be integrated inside the circuit itself to protect it against inner ESD events (such as *Charged-Device Model*).

Overall, ESD analysis covers a wide range of techniques: some being local to I/O ports regions, others requiring a more global view of the circuit. Even if the points of verification are in general well-identified, ESD analyses are already a good source of inspiration for more global pattern-based approaches. Conversely, we will see that some efforts exist for ESD analysis using different (*i.e.,* less local) verification methods.

Structure recognition techniques are also used to assert properties on the patterns themselves, for specific use cases such as analog designs and power-down mode. For instance, Zwerger and Graeb [64] and Neuner and Graeb [39] rely on pattern recognition can to assert symmetry properties in analog structures, such as differential pairs and current mirror, which is crucial to limit the aging of the designs. Similarly, Kunz et al. [28] use pattern matching to identify structures that are known to be vulnerable against ESD events. Pattern-based analysis can also provide a qualitative analysis of such structures — *e.g.,*, to characterize the quality of all the ESD protections in a design [59, 60], making it possible to quickly identify the most vulnerable parts.

*Scalability.* Building pattern recognition tools is a non-trivial task, which has already been addressed in the graph analysis domain. In order to avoid implementing such tool from scratch, TL-ERC approaches based on pattern recognition tend to use existing tools, and to adapt graph isomorphism techniques to the context of transistor-level circuits. Notably, Calibre PERC [51] is commonly used for this particular feature [19, 30, 59, 60].

To scale up to billion-scale SoCs, pattern recognition techniques need to handle hierarchy. Neuner and Graeb [39] demonstrate how a hierarchical approach can use pattern-matching techniques, in particular by matching module patterns for symmetry assertions. Other optimizations have been proposed to improve the performances of pattern recognition on realistic use cases, for example by using pattern reduction [59] to reduce the complexity of the approach.

*Usability.* TL-ERC approaches and tools should be easily integrable in design flows, in order to ease their usage by designers in their daily task, especially so that verification processes are run whenever necessary. To enable such integration, Viale and Allard [59] propose an innovative process to characterize and visualize the quality of ESD protection patterns. In their framework, the quality of each ESD protection is quantified, and can be visualized through a visualization matrix, where each cell represents the vulnerability of the path between two I/O pads. Users can hence use this visualization feature to quickly apprehend which parts of the circuit are the most vulnerable, and where the design efforts should be put to improve the circuit resilience. This is particularly useful during the revision process, *i.e.,* the task of modifying a circuit when the verification process highlighted some design rule violations. By using such approach, designers can hence reduce the effort required for revision by focusing on the actual vulnerabilities of the circuit, as highlighted by the analysis tool. Other pattern-based TL-ERC approaches also provide guidelines for revising the designs based on the results of the TL-ERC analysis [39], to help designers understanding the impact of those analysis results, and using them correctly for the debug task.

*Limitations.* One of the major issues of pattern-based approaches lies in the patterns themselves. As a matter of fact, relying on a library of patterns for the structure recognition can lead to erroneous conclusions during the analysis. On one hand, a pattern library cannot be exhaustive, and some structures which should be identified by the pattern recognition algorithms might actually be

Author Version

missed. When pattern recognition is used as a refinement step after an analysis yielding false alarms, some warnings which should have been filtered out will not. Conversely, if pattern recognition is used to identify vulnerability patterns, some might not be found by the tool, as they do not correspond precisely to the ones in the library. On the other hand, libraries are provided by the engineers, which leverages their expertise to identify relevant patterns, along with the properties of those patterns. An engineer may introduce an erroneous pattern in the library, which would make the analysis unsound (real errors from a prior analysis may be mistakenly filtered out). Additionally, some functional blocks with interesting properties might replace the usual patterns and not be detected by a standard pattern-recognition technique. For example, functional parts might be considered as secondary protections against ESD events [59], but cannot be identified by pattern recognition in the general case.

Moreover, while pattern recognition techniques can be used to ensure properties that cannot be detected by a simpler approach such as the ones described in Section 4, it cannot be used to accurately model the dynamic behavior of the designs. It is hence limited to identifying structures where the interesting behavior (*e.g.,* the ability to prevent damages during an ESD event) does not depend on the electrical state of the circuit. In this way, the pattern recognition algorithms search for structures with a specific "static behavior", but cannot consider the "dynamic behavior" of those structures. For example, Viale et al. [60] and Viale and Allard [59] use pattern recognition to identify both static and dynamic ESD IPs, *e.g.,* respectively structures that are always a protection against ESD events, and structures that must be triggered to efficiently protect the design. However, as patterns do not model the electrical state of the circuit (which mean it cannot determine the value of the trigger signal), they specify that for dynamic ESD IPs, they can only assume that the pattern is properly controlled — *i.e.,* that the trigger signal is correctly set. This results in potential flaws in the analysis, as their assumption cannot be enforced by the verification process. In this example, dynamic ESD IPs may never be enabled, while the verification process considers they are.

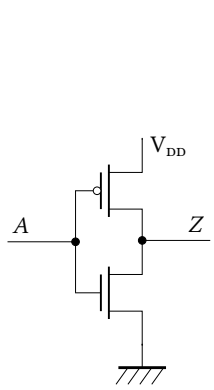
## 6 SWITCH-BASED SEMANTICS APPROACHES

The last family of TL-ERC approaches model the switching behavior of the transistors in the design under analysis. In contrast with the wire-based semantics approaches introduced in Section 4, switch-based semantics considers more realistic electrical behavior of the circuits, reducing the abstraction level at which the analysis is performed.

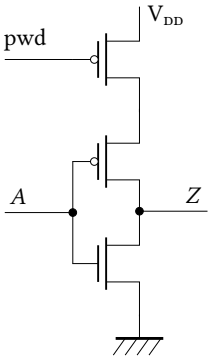
*Principles.* The approaches presented in this section model transistors as programmable switches: depending on the voltage on the gate, an electrical connection is made between source and drain. This refinement over the wire-based semantics allows pruning the electrical state space from unstable/unreachable states. For example, when considering the inverter circuit from Figure 3a, such approach can model that the trigger signals of both transistors — *i.e.,* the voltage applied on their gates — are not independent. In this case, a switch-based analysis is able to state that at least one of the transistors must be switched on and thus $Z$ cannot be floating. Such behavior cannot be captured by the previously described approaches, meaning that both wire-based and pattern-based semantics would consider some unreachable states in their verification process, leading to numerous "false alarms".

Switch-based approaches for TL-ERC are still emerging initiatives, implemented using numerous analysis techniques. The following subsections consider three proposed approaches which seem promising both in the accuracy of their results and their potential of scalability.

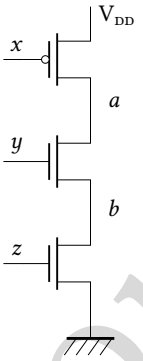
Author Version




(a) CMOS-based inverter circuit




(b) Inverter circuit with a power-down mode




(c) Circuit with a short-circuit path

Fig. 3. Integrated circuits schematics as examples

### 6.1 Iterative voltage propagation

*Iterative voltage propagation* is an improvement of the naive voltage propagation technique introduced in Section 4, where the switching behavior of transistors is considered during the voltage propagation phase. The technique mainly relies on marking the voltage at each net and the state of each transistor of a circuit as initially unknown. Then, an input vector is used to start an iterative traversal of the nets. For each gate for which a voltage can be defined, the transistor is either switched on or off, and this information can be used for further propagation at the next iteration. At the end of this propagation phase, every net has been annotated with either a voltage, or an unknown value if the input vector was not sufficient to propagate information to this net. This phase is sometimes called *static voltage propagation* in the literature [62, 64]. In the scope of this survey, we use the clearer denomination of *iterative voltage propagation*, to contrast with naive voltage propagation approaches described in Section 4.

*Analyses.* A first approach for iterative voltage propagation was introduced by Blieck et al. [6], which used a simple algorithm and type system to iteratively propagate the voltage values in the circuit nets to check for undesired currents. This approach has been generalized by Zwerger et al. [62, 64], making it possible to efficiently propagate supplies in analog circuits with power-down modes. They provide various models for the basic components (*i.e.,* transistors, resistors and capacitors), and use them to build a graph representation of the circuits, where vertices are the nets and edges are connection between the nets, through the devices. Doing so, they can define an *ad-hoc* propagation algorithm that iteratively scans this graph to propagate the supply values — as well as a given power-down signal value — throughout the circuit. Edges may be considered as on or off switches, depending on the voltages already propagated in the circuit. The complexity of this iteration-based supply propagation algorithm is $O(n^2)$, with $n$ being the number of connections in the circuit.

Hogan et al. [21] propose a similar technique with a two-mode analysis: the *vectorless mode* that corresponds to naive voltage propagation and has been presented in Section 4, and the *vectored mode*. In *vectored mode*, users must provide an input vector, to define the value of each input signal of the circuit (making this approach quite similar to simulation-based approaches). The voltage propagation algorithm can then use this input vector to model transistors either as on or off switches (while *vectorless mode* models every transistor as a switched on transistor), iteratively

Author Version

propagating the supplies from source to drain in the design. This can hence be used to define the global electrical state of the design — *i.e.,* the voltage at every net — with respect to an input vector.

*Considered properties.* Iterative voltage propagation has been used in multiple verification tools for various error properties. By design, it can estimate the voltages of the different nets of a circuit in a more accurate manner than naive voltage propagation. It can even be used to propagate some information on the electrical status (*i.e.,* short-circuit or floating) of the nets [6]. For those reasons, these approaches have been used to check for undesired short-circuits, current leakage or floating nets [6, 62]. This approach has also been used in the context of asymmetric voltage conditions in power-down mode [64]. As stated in Section 2, asymmetry in the voltages applied to some analog structures may lead to critical aging of devices. Using a proper modeling of the components' behavior, iterative voltage propagation can be used to check for symmetry rules in analog designs, making it possible to identify potential vulnerabilities, like for example asymmetry resulting from the integration of a power-down mode. Last but not least, iterative voltage propagation can also be used for more complex error identification in combination with other techniques (notably pattern recognition), such as identifying EOS [21] or ESD [37, 39] vulnerabilities in analog circuits.

*Scalability.* Iterative voltage propagation mainly rely on a custom modeling of the transistor behavior and a propagation algorithm. To improve the performance of these algorithms, Neuner et al. [37, 39] proposed a generalization of the voltage propagation algorithm from Zwerger and Graeb [62], to make it possible to handle hierarchical circuits. This is probably a required step for a further adoption in industrial-scale processes.

*Usability.* Iterative voltage propagation should be usable in standard verification processes, to be integrated in most workflows. The possibility to use input vectors for dynamic supply propagation [21] hence seems suited for industrial usage, as it complies with standard simulation processes that already use such vectors.

*Limitations.* While iterative supply propagation approaches seem to be a promising improvement of naive supply propagation techniques, it currently suffers from some limitations.

The first limitation is related to I/O value management. In most of the approaches, the value of inputs is not modeled [6, 62], and the iterative voltage propagation mechanism actually only propagates the different supplies in the circuit. This means that if an input net (with an unknown voltage) is applied to the gate of a transistor, the current status of this transistor is undefined — it is hence considered as being switched on, for correctness reasons. On the other hand, in the "vectored" approach from Hogan et al. [21], the user must provide an input vector to specify the voltage of each input net, meaning that each propagation is valid only for this vector. In the first case, the voltage propagation over-approximates the reachable electrical states, indeed the reachability of some states depends on the input values. In the second case, supply propagation must be run on every interesting input vectors [64], which can be an exhaustive combination of all possible vectors, or a subset of it if defined by the user (either based on previous simulations, or users' knowledge), but this is in general too inefficient. Additionally, if only some scenarios are explored, this under-evaluates the set of reachable states.

Moreover, in the current state of the art, iterative supply propagation is also limited due to its computational complexity [62]. As it is an emerging approach, there is no standard, optimized tool to perform this propagation, and algorithmic/implementation optimizations are still to be proposed to improve its efficiency.

Author Version

### 6.2 Power-up transformation

*Power-up transformation* is an operation proposed by Zwerger et al. [64] as a prerequisite for TL-ERC verification on analog circuits with a built-in power-down mode. Power-up transformation uses an identified *power-down signal* as input, which is the trigger signal for the power-down mode (for example, the *pwd* signal in Figure 3b). This analysis can hence be seen as a particular case of iterative voltage propagation, where the input vector consists in the *pwd* signal. In addition, a library of patterns is also used to identify digital blocks which may be implied in the power-down circuitry.

*Analyses.* Power-up transformation is often a necessary first step because standard verification tools are not built to work with the power-down circuitry, and may fail to do so — for example, analyses may use pattern recognition to identify analog patterns that should be symmetrical, but such patterns may be different from the ones in the pattern recognition libraries due to the power-down circuitry. Indeed, once the power-down circuitry has been identified, it can be removed (*i.e.,* the corresponding transistors will be replaced either by a wire, if power-up mode requires them to be switched on, or will be removed in the other case). The resulting circuit can then be checked using standard techniques (such as pattern recognition).

For example, consider the inverter circuit from Figure 3b, we can use power-up transformation to remove the power-down circuitry (*i.e.,* the PMOS transistor controlled by the *pwd* signal), in order to retrieve the original inverter circuit from Figure 3a. This makes it possible to model the circuit behavior during its "normal" operating mode.

In [64], power-up transformation is followed by another supply propagation algorithm that uses an input vector to propagate voltages to the nets of the circuit.

*Considered properties.* Zwerger et al. [64] used power-up transformation for symmetry assertions, using two complementary pattern libraries: a first library to identify and remove power-down circuitry, and a second one to identify analog patterns in which symmetry rules must be asserted. For each of those patterns, and for each input vector, they identify symmetry violations and provide a report error that can be used to verify that no mismatch in the analog design may lead to premature aging of the circuit.

Based on this approach, Neuner et al. [37, 39] proposed a generalized approach to identify not only symmetry rules violations, but also undesired short circuits and floating nets.

*Usability.* Neuner et al. also address a more practical concern by providing guidelines to revise designs after analysis [39]. The technique has shown to be general enough to be integrated in more complex approaches for verification and synthesis. Based on the same techniques, initiatives have also been proposed for synthesis of power-down circuitry [65], in order to ensure the electrical properties of the circuits while adding a power-down mode.

*Scalability.* Power-up transformation techniques are meant to be a necessary first-step of the analysis. As such, they must be efficient and scalable. By nature, the analysis is not much precise and thus relatively scalable. Neuner et al. also propose a hierarchical optimization of these techniques in order to improve the performance on large scale circuits [39].

*Limitations.* Power-up transformation approaches suffer from the limitations due to the techniques used.

First of all, they rely on pattern recognition techniques, which lack exhaustiveness and correctness, as detailed in Section 5. Moreover, the voltage propagation phase relies on input vectors to identify power-down signals and are thus sensible to the fact that these signals are exhaustively identified.

Author version

### 6.3 SAT solving approaches

A Boolean SATisfiability solver (SAT for short) is a tool that analyzes a logic formula to check whether there exists a valuation of the formula's variables such that the formula evaluates to true.

SAT solvers can be used to analyze a circuit. The approach consists in building a logical formula from a given netlist, where each term either encodes the behaviour of the circuit, or encodes an error condition. A standard SAT solver (*e.g.,* CryptoMiniSat [53]) can then be used to efficiently find a condition on the circuit inputs that satisfies this formula, somehow propagating the error condition to check if it's satisfiable.

*Analyses.* Afonso et al. [1] use a SAT solver to identify sufficient conditions for a short-circuit. To build the formula, the algorithm iteratively scans the netlist, and, for each device, adds the condition for a short-circuit to happen on this device to the existing short-circuit formula. The existing formula is simplified if an existing term in the formula is in contradiction with the new term to add. In this approach, the dynamic behavior of the circuit is modeled by encoding the switching conditions of the transistors directly in the short-circuit formula. Resistors and inductors are modeled as wires, and capacitors are modeled as open circuit, as the approach only considers the steady states of the circuits. Consequently, if no input combination is found, it also means that no short-circuit is possible in a given circuit. For example, their method can be used to identify the short-circuit conditions in the circuit from Figure 3c, where the input combination $\neg x \wedge y \wedge z$ leads to a direct path from *Vdd* to the ground.

In [40], a method to encode the semantics of a circuit is presented in the form of a patent. The encoding considers not only 0/1 logical values for each net, but also high-impedance — *i.e.,* floating values. It uses a one-hot encoding of net states, assigning one Boolean variable for "net has value 0", "net has value 1" and "net is floating". The encoding is applied to floating net detection [41]. The method relies on a prior search for suspect floating gates. A logical tree representation is then built for every suspect gate net, taking into account the immediate sub-circuit containing the gate net. An iterative process is run to check the satisfiability of the error on predecessor circuits, which repeats until either no predecessor circuit that can cause the gate to float is found, or until a circuit causing the gate to float is found. The analysis is performed on a subset of gate nets representing floating gate candidates, it then follows a Counter Example Guided Abstraction Refinement (CEGAR) over circuit portions on which a candidate floating gate net is found. The net is only identified to be actually floating when a root cause is encountered. The authors also discuss the possibility of reducing the number of reported floating gates — *e.g.,* by dismissing floating gates when they correspond to powered off devices.

*Considered properties.* SAT solving have only been used for short-circuit and floating gates identification, but in principle it could be generalized to identify sufficient conditions for any path to be active in a given circuit. This could identify other errors and could be composed with other approaches, for example to perform voltage propagation by modeling the conditions for each net to be connected to the various supplies of a design.

*Scalability.* In their work, Afonso et al. [1] propose an approach where the different terms of the formula are built independently from different parts of the circuit. This phase can be parallelized before merging the different terms for solving [49], leading to a reasonable scalability (circuits with 10,000 internal nodes are analyzed in less than a minute on average with 16 threads).

CEGAR-based approaches [41] run the SAT-based analysis only on very small portions of the circuit, avoiding the scalability issue due to the NP-completness of SAT and allowing them to scale linearly in the size of the circuit. They can also use parallelism to independently analyze several

Author Version

floating gates and discover whether each of them is a real error or disappears when enlarging the boundaries of the analyzed circuit.

*Limitations.* Afonso et al. only consider circuits with a single supply, making the voltage propagation phase useless, while in [41] two power levels are considered ("low" or "high"). To generalize these approaches to more realistic use cases, multiple supplies should be supported.

Not many results exist in the literature analyzing the scalability of SAT-based approaches. Moreover, while both of the presented approaches consider exploiting parallelism to improve the performances, hierarchy considerations are not discussed, hence limiting the potential of scalability.

Overall, this field of researches highlight a promising usage of SAT solvers, long known to software verification engineers for symbolic model-checking, in the context of Transistor-Level Electrical Rule Checking.

*Usability.* The restriction to a single supply in the SAT-based approach from Afonso et al. [1, 49] is, in practice, an important limitation for an integration in standard design flows. For example, short-circuits are usually avoided by construction on CMOS designs synthetized from *e.g.,* RTL, but they can appear by mistake on circuits with multiple supplies, for example as a consequence of a missing-level shifter.

### 6.4 SMT solving approaches

A more general approach, based on Satisfiability Modulo Theory (or SMT) instead of SAT, has been proposed by Oulkaid et al. [12, 44]. SMT solvers make it possible to consider numerical variables to represent voltage values. This makes it possible to represent multi-supply circuits and to express properties related to voltage values. The objective of this work is to provide a generic framework for TL-ERC, based on the use of an SMT solver. They demonstrate how it can be used to identify missing level shifters in integrated circuits.

*Analyses.* The approach is based on the proposal of semantics of integrated circuits at transistor-level. Doing so, they provide a way to encode the behavior of any circuit into a logical formula $\mathcal{F}$. $\mathcal{F}$ actually encodes all the constraints that are used to define the voltage of every net of a circuit, resulting in a large formula for which any solution represents a steady-state of the circuit. This formula can then be checked against any property $\mathcal{P}$ that can be expressed as an SMT formula too, *e.g.,* any formula that reasons on the voltages of the nets in a circuit. If $\mathcal{F} \wedge \neg\mathcal{P}$ is UNSATisfiable, then it is proved that $\mathcal{P}$ holds for any valuation of $\mathcal{F}$, *i.e.,* for any electrical state of the circuit.

*Considered properties.* This approach is demonstrated on an industrial use case, in order to identify missing level shifters in integrated circuits. Moreover, they also state that both short-circuit and floating states are encoded in the logical formula, implying that this method could be adapted to identify undesirable short-circuit or floating gates.

*Scalability.* The performance of this approach is tightly linked to the performance of SMT solvers, which may be a limitation for scaling. The approach is shown to scale to subcircuits of thousands of transistors on artificial benchmarks and real-life examples, but not to full-circuit scale. This can be used to successfully analyze subcircuits and identify electrical issues at the leaf or at least the bottom of the circuit hierarchy, reducing the number of issues to identify with other approaches (*e.g.,* simulation) in the remaining of the flow. Moreover, the authors propose *modular verification* as prospects of their work, to make it possible to scale to realistic, industrial-size circuits.

*Usability.* The use of an SMT solver instead of a SAT solver makes the approach applicable to multi-supply circuits or sub-circuits, hence more widely usable than *e.g.,* [1]. The proposed encoding is modular in the sense that each device is translated independently to a set of SMT

Author Version

clauses, and the property to be verified is also encoded separately. As a consequence, the same circuit encoding can be used to verify properties other than missing-level shifters.

*Limitations.* To demonstrate their approach, Oulkaid et al. [12, 44] focus on the problem of missing level shifter identification, and the applicability of the method to other TL-ERC is still to be proved. Moreover, they also state that the semantics that they propose is limited, and cannot be used to analyze any circuit. Last but not least, the scalability of this approach is yet to be studied, with respect to industrial-size circuits with up-to-billions of transistors.

There exists some more precise SAT-based approaches that aim at reasoning about analog circuit parameters in steady-states — as done by Miller et al. [36]. They introduce a circuit encoding technique which considers a bounded behavior model of transistors, so that different sorts of device variations can be captured. Such approach can be seen as complementary to TL-ERC since it proposes an alternative to simulation to explore the state-space exhaustively, but doesn't fit in the scope of this survey as the scalability is limited to a few transistors only.

## 7 INDUSTRIAL TOOLS

The goal of this survey is to present the various methods and approaches for TL-ERC that exist in the literature. Indeed, besides the several articles that were discussed in this paper, numerous industrial tools exist that operate on circuit descriptions to perform TL-ERC related tasks, such as ESD characterization or short-circuit identification.

In this section, we hence present some of the most used tools for TL-ERC. By essence, it is difficult to understand exactly how industrial tools work — it would require several benchmarks to understand the various approaches and properties that each tool considers. As such benchmarking effort is out-of-scope for this survey, in this section we only describe the purpose of each tool, as well as how it relates to the taxonomy we introduced in Section 3.2, based on publicly available information.

Some of these tools are dedicated to the identification and characterization of ESD vulnerabilities in circuit design. Among them, one can cite Magwel ESDi [34] and Ansys PathFinder [4], which provide way to find the causes of design issues related to ESD, or Cadence ESD Cross Domain Checker [8], which focuses on ESD protection at cross-domain interfaces. Those tools can be used to identify ESD protection networks, characterize their properties, and identify related vulnerabilities to assert the quality of the chips.

On the other hand, some of the industrial tools target non ESD-related ERC, such as identifying short-circuits, floating gates or missing level shifters. Among them, the tool that is the most mentioned in the literature seems to be Calibre PERC. Calibre PERC [19, 51] (for "Programmable ERC") is a tool from Siemens that can be used to define custom ERC rules, and check them automatically. In particular, it offers *hierarchical supply propagation* and *pattern recognition* features that have been used in several of the works that we considered [26, 30, 59, 60]. It can also be used for context-aware geometrical checks, *i.e.,* performing geometrical checks on the layout for a subset of devices that have been identified at transistor-level [16, 26, 31]. Siemens recently acquired the company InsightEDA, which commercializes the tool Insight Analyzer [23] that performs a wide variety of checks on a transistor-level netlist, including short-circuit checks, floating gate checks [41], and missing level-shifter detection [42]. They position their tool as *designed to find issues that are often missed by simulation and other traditional sign-off tools.* Synopsys also proposes its own tool, called Synopsys CustomSim [56], which can be used for Analog/Mixed Signal (AMS) verification. It embeds SPICE simulation as well as Design Rule Checking, and makes it possible to perform static, dynamic or even programmable checks on integrated circuits, in order to identify vulnerabilities such as mismatched power domains, excess leakage paths or floating gates. More

Author version

recently, Aniah proposed a new solution to perform efficient ERC [2], with a particular focus on how to control the trade-off between accuracy and efficiency of the verification phase.

*Limitations.* Even though these industrial tools are widely used in manufacturing processes, we highlight a few of their limitations, mostly linked to their opacity, *i.e.,* the lack of internal documentation and/or access to the source code of the tools. We believe their opacity limits their adoption, both from the point of view of the users and from the point of view of researchers. As the approaches used by industrial tools are in general not (or lightly) documented, it is difficult for the users (whether researchers or engineers) to (1) understand which approaches could be adapted to each use case; (2) characterize the limitations of the tools, in particular, which case may lead to false warnings or to errors being missed (see the discussion on soundness and completeness in conclusion). These limitations raise significant issues in terms of reliability of the proposed solutions, since claims about the absence of false negatives are not verifiable. For example, scientific research could easily back-up the soundness of a well-defined approach by publishing their results, while this is not possible if the approach is not well-documented nor formalized. These limitations also impact the development of new TL-ERC approaches, as they make it difficult to reproduce and improve the existing methods.

## 8 CONCLUSION AND PROSPECTS

This survey proposes a new analysis of existing techniques for the problem of *Transistor-Level Electrical Rule Checking* (TL-ERC). More specifically, we introduce a new taxonomy to classify existing approaches with respect to how they model the behavior of the components, in particular of transistors. This taxonomy is based on how the behavior of the transistors is abstracted away: some approaches model transistors as wires (Section 4), other approaches identify specific patterns of components to model a known behavior (Section 5), and finally some approaches model the switching behavior of transistors (Section 6).

### Complementarity of Wire-, Ad-hoc- and Switch-based Approaches for TL-ERC

Based on the considerations exposed in Sections 4, 5, and 6, we highlight several points of interest in the analysis of TL-ERC approaches. The following paragraphs introduce a discussion on those keypoints, which can be used to understand what is the common ground for the different approaches and how they can be combined and/or compared.

*Voltage estimation as a prerequisite.* Most of the approaches that we discuss are based on estimation of the voltages at the nodes of the circuit. Techniques used to perform this estimation can be very simple, as seen with *naive voltage propagation* (Section 4), or more precise, with approaches such as *iterative voltage propagation* or even SMT encoding of the constraints on the voltages (Section 6). This highlights a trade-off between the performance of the voltage estimation method and the accuracy of the estimation. Indeed, *naive voltage propagation* is fast, it can be done in a few seconds, but leads to a rough overestimation of the voltage sets for each node of a circuit, eventually resulting in a lot of "false alarms" in the diagnosis. On the other side, while more advanced estimation techniques can be used to reduce the number of "false alarms" at the cost of performance. For example SMT encoding seems to be limited to a few thousands of transistors [44].

*Generality vs ad-hoc approaches.* Some of the approaches that we discuss are limited by design to specific use-cases — *e.g.,* Afonso et al. [1, 49] use a SAT encoding of circuits dedicated to short-circuit detection — while some others are generalizable to more use cases — *e.g.,* Oulkaid et al. [12, 44], where the error condition can be any formula considering the voltages of the nodes. It can be remarked that, in this setup, adding error conditions, like trying to identify missing level-shifters or

Author Version

floating gates instead of missing level-shifters only, should not impact the scalability substantially, as the encoding of the voltages of the circuit remains the same.

On the other hand, when it comes to ad-hoc patterns approaches, scalability might be affected when trying to generalize the approach. As exposed in Section 5, pattern matching approaches are widely used, either to filter out "false alarms" raised by other approaches, *e.g.,* by *naive voltage propagation*, or to identify vulnerable patterns in a design. However, by design, those approaches are limited in scaling by the number of patterns to be searched: if we want to add a new class of patterns to detect, either for detecting a new kind of errors or for supporting a new protection pattern, the matching algorithm must consider more patterns, hence impacting their performance.

This highlights a difference in the scalability potential between ad-hoc patterns and voltage estimation approaches (either with wire- or switch-based semantics). Indeed, most approaches (both in industrial and academic worlds) tend to mix techniques from Sections 4, 5, and 6 to provide adapted tools that can be integrated in existing workflows. Fast and imprecise approaches like voltage propagation are commonly used as a first step to focus more precise analyses on smaller parts of the circuit, where less scalable approaches can be used.

**Comparison of the presented approaches**

The different approaches analyzed are summarized in Table 1. In particular, the table details how the different aspects of each analysis under study can be classified in three classifying categories. The table also exhibits the errors and properties that are considered by the various approaches, highlighting that several properties may be asserted with similar approaches. The detailed description of the approach in the table has been proposed in the survey, but this table allows us to draw a few high-level conclusions. Note that pattern-based analyzes (Section 5) are a bit special as they mostly do not reason on the net voltages in the circuit to identify errors. Moreover, please note that, in Table 1, we do not include the synthesis approaches such as [38, 65], as we focus on verification techniques rather than correct-by-construction methods that are safer but not adapted to some of the industrial practices.

One can first observe that many approaches are very similar and analyze similar errors and properties. For example, similar approaches, based on iterative supply propagations, have been proposed to diagnose undesired short-circuits in various functioning conditions [6, 37, 62]. Another example lies in identifying vulnerabilities to ESD events, with similar methods based on both supply propagation and structure recognition [31, 59, 60], highlighting how the various techniques summarized in this work can be combined for efficient, industrial-fitted tools. In particular, it demonstrates how TL-ERC approaches are suitable to perform a subset of the static ESD checks, *i.e.,* the *topology checks* [57]. However, the proposed solutions are not well positioned relatively to the similar approaches in the literature. Our first contribution in this survey is to propose a common setting for comparing the different existing approaches. We hope that this will help build a more consistent research community around the topic. A further step would consist in providing benchmarks for TL-ERC, making it possible to effectively compare the various approaches in a quantitative manner. Such benchmarks would be a big step forward in an attempt to unify the community working on TL-ERC, providing not only a theoretical common setting to discuss TL-ERC approaches, but also an experimental setup to compare their performances.

**Performance and usability**

The approaches presented here work in practice, and many of them have been experimented on real-world large circuits. The approaches that can analyze real-life circuits all rely on ad-hoc combinations of the key steps we identified. The most effective industrial tools (such as Calibre PERC) combine several different steps, they generally perform both naive voltage propagation

Author version

Table 1. Summarizing the approaches to Transistor-Level Electrical Rule Checking

| Article(s) | Analysis Technique | Circuit Semantics | | | Errors Considered[2] |
|---|---|---|---|---|---|
| | | *Wire-based (Section 4)* | *Ad-Hoc Patterns (Section 5)* | *Switch-based (Section 6)* | |
| [6] | Type based & dynamic supply propagation | - | - | supply propagation | SC |
| [21] | Custom rule checking | supply propagation in "vectorless mode" | - | supply propagation in "vectored mode" | EOS |
| [30] | Custom rule checking and error filtering | supply detection & supply propagation | pattern identification for error filtering | - | FN, MLS, ESD, EOS |
| [31] | ESD verification using cross domain checks and pattern matching | supply detection & supply propagation | ESD structure recognition | - | ESD |
| [59, 60] | ESD pattern recognition | supply detection & supply propagation | ESD network identification | - | ESD |
| [33] | Cross-domain interface verification with custom rule checking | topology aware net type for supply propagation | *ad-hoc type-based ESD protections (anti-parallel diodes and GGNMOS)* | - | ESD |
| [32] | ESD path identification with graph decomposition | ESD path identification | - | - | ESD |
| [28] | ESD sensitive topology identification | - | identification of topology patterns that are vulnerable against ESD events | - | ESD |
| [62] | Graph-based dynamic supply propagation | - | - | supply propagation | SC, FN |
| [64] | Asymmetry voltage conditions in analog power-down mode | - | power-down mode structures & symmetrical pattern recognition | supply propagation & power-up transformation | Sym |
| [37, 39] | Power-up transformation, dynamic supply propagation and symmetry assertions | - | power-down mode structures & symmetrical pattern recognition | supply propagation & power-up transformation | SC, FN, Sym |
| [1, 49] | Identifying short-circuit conditions using a SAT solver | *single supply* | - | encode short-circuit conditions as logical formula | SC |
| [12, 44] | Identifying missing level shifters using a SMT solver | - | - | encode both circuit semantics and error conditions as logical formula | MLS |
| [40, 41] | Checking the Satisfiability of floating gate candidates, by iterative propagation of error conditions | - | - | Refinement based analysis for identifying floating gates | FN |

[2]SC: Short-Circuits; ESD: ElectroStatic Discharges (in particular, *Human Body Model*); EOS: Electrical OverStress; FN: Floating Nets; MLS: Missing Level Shifters; Sym: Symmetry properties (see Section 2).

Author Version

and pattern recognition. However, a verification tool that would consider not only the topology of the circuit, but also its semantics could perform a precise analysis without the need to maintain a library of patterns. Such an approach would in general be a bit more computationally intensive, but we believe that, by combining with other approaches, it could scale to large circuits.

Concerning scalability, this survey also highlighted the fact that enabling TL-ERC at industrial scale requires managing efficiently the hierarchy of circuits. Indeed, a lot of recent initiatives have adapted previously designed approaches to handle hierarchy [39, 59]. This demonstrates how existing approaches can be adapted for industrial-scale circuits of billions of components and this way, be integrated in realistic design flows.

Concerning precision, we believe that switch-based semantic approaches is an interesting and under-exploited direction for improving the performances and reliability of TL-ERC approaches. *Switch-based semantic approaches* model the fact that transistors can dynamically switch from on to off states, given the electrical state of a circuit, while operating at transistor-level (*i.e.,* without considering complex details such as differential equations to model the real behavior of the transistor). Formal methods [1, 12, 49] can effectively model the switch-based semantics of circuits. In particular *Counter Example Guided Abstraction Refinement* (CEGAR) and *Satisfiability Modulo Theory* (SMT) techniques seem particularly adapted to the reasoning on electrical circuits and TL-ERC. As a matter of fact, recent initiatives based on formal methods have tackled TL-ERC. For example, Plassan et al. [47] use CEGAR to assert that clock domain crossing parts of circuits are correctly protected. While this check is performed at RTL abstraction level, the problem itself is very similar to asserting that level-shifter structures are correctly inserted in a circuit at transistor-level; TL-ERC could hence benefit from such initiatives. More generally, we believe that the formal-method community could bring efficient and innovative approaches to TL-ERC.

### Reproducibility and comparison of TL-ERC approaches

In this work, we consider various approaches to TL-ERC, with each approach seemingly validated on custom benchmarks, composed of non public circuits. This makes it difficult to compare the different approaches. It also makes it difficult to design new solutions, especially for researchers who might not have access to realistic circuits through industrial partnerships. While out of the scope of this survey, we believe that providing an open-source benchmark of circuits that could be used to evaluate new approaches to TL-ERC would be a crucial contribution to the TL-ERC research community. Such a benchmark would benefit to both academic and industrial actors. Designing such a benchmark is however particularly difficult in a context where there was no common view of the existing approaches. We hope that the present survey will provide to the researcher and industrial actors an overview of the approaches a public benchmark should target.

### Discussion on soundness and completeness

When it comes to verification, strong — i.e., formal — guarantees on the result provided by the verification tool play a major role. In the formal method community, this is often defined using two complementary concepts: soundness and completeness [52]. To put it in a nutshell, a verification approach that is sound should identify all possible errors (*i.e.,* no error is missed but false warning may be raised), while a complete approach should identify only errors that appear in feasible scenarios (*i.e.,* no false warning can be raised, but some errors may be missed). In other words, sound approaches can only prove properties that are actually true, while complete approaches can prove all true properties. Obviously, an ideal TL-ERC approach should hence be both complete and sound, meaning that the approach should identify real electrical violations, and only those. However, for most interesting analyses, finding precisely the errors is an undecidable problem, and each verification approach can only provide guarantees of either soundness or completeness, while limiting the number of false alarms or missed errors for the approach to be usable.

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

## REFERENCES

[1] Joao Afonso and Jose Monteiro. 2017. Analysis of short-circuit conditions in logic circuits. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, Lausanne, Switzerland, 824–829. https://doi.org/10.23919/DATE.2017.7927102

[2] Aniah. 2020. ERC: a Trade-Off Between Coverage and False Positives? https://aniah.fr/2020/10/12/erc-a-trade-off-between-coverage-and-false-positives/. [Accessed 08-July-2024].

[3] Aniah. 2024. ERC: an exhaustive classification of false errors. https://aniah.fr/2024/04/11/erc-an-exhaustive-classification-of-false-errors/. [Accessed 08-July-2024].

[4] Ansys. 2023. Ansys PathFinder-SC. https://www.ansys.com/fr-fr/products/semiconductors/ansys-pathfinder-sc. [Accessed 08-July-2024].

[5] P. Ashar, S. Bhattacharya, A. Raghunathan, and A. Mukaiyama. 1998. Verification of RTL generated from scheduled behavior in a high-level synthesis flow. In *1998 IEEE/ACM International Conference on Computer-Aided Design. Digest of Technical Papers (IEEE Cat. No.98CB36287)*. 517–524. https://doi.org/10.1145/288548.289080

[6] S Blieck and Er Janssens. 1996. Software Check for Power-down Mode of Analog Circuits. *ESSCIRC'96: Proceedings of the 22nd European Solid-State Circuits Conference* (1996), 404–407.

[7] Vincent Bligny. 2024. Personal communication. CEO of Aniah, https://aniah.fr/ [Accessed 22-July-2025].

[8] Cadence. 2025. ESD Cross Domain Checker. https://www.cadence.com/zh_CN/home/resources/datasheets/esd-cross-domain-checker-ds.html. [Accessed 22-July-2025].

[9] ChipEdge. 2022. What is ECO in VLSI Physical Design? https://chipedge.com/what-is-eco-in-vlsi-physical-design/ [Accessed 22-July-2025].

[10] Michael Eick and Helmut E. Graeb. 2012. MARS: Matching-Driven Analog Sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 8 (2012), 1145–1158.

[11] ESD Association. 2025. ESD/EOS Symposium. https://esda.events. [Accessed 22-July-2025].

Author version

[12] Bruno Ferres, Oussama Oulkaid, Ludovic Henrio, Mehdi Khosravian, Matthieu Moy, Gabriel Radanne, and Pascal Raymond. 2023. Electrical Rule Checking of Integrated Circuits using Satisfiability Modulo Theory. In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1–2.

[13] Luca Ferro and Laurence Pierre. 2009. ISIS: Runtime verification of TLM platforms. In *2009 Forum on Specification & Design Languages (FDL)*. 1–6.

[14] Harry Foster. 2022. Part 8: 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. [Accessed 24-October-2023].

[15] Eleonora Gevinti, Lorenzo Cerati, Leonardo Di Biccari, Giuseppe Ballarin, Antonio Andreini, Mauro Fragnoli, and Antonio Bogani. 2015. Schematic-Level and Layout-Level ESD EDA check methodology applied to smart power IC's - initialization and implementation. In *2015 37th Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, Reno, NV, USA, 1–10. https://doi.org/10.1109/EOSESD.2015.7314770

[16] Patrick Gibson, Ziyang Lu, Fedor Pikus, and Sridhar Srinivasan. 2010. A Framework for Logic-Aware Layout Analysis. In *2010 11th International Symposium on Quality Electronic Design (ISQED)*. IEEE, San Jose, CA, USA, 171–175.

[17] Venkatesh Gourisetty, Hamid Mahmoodi, Vazgen Melikyan, Eduard Babayan, Rich Goldman, Katie Holcomb, and Troy Wood. 2013. Low power design flow based on Unified Power Format and Synopsys tool chain. In *2013 3rd Interdisciplinary Engineering Design Education Conference*. IEEE, 28–31.

[18] Aarti Gupta. 1992. Formal hardware verification methods: A survey. *Formal Methods in System Design* 1, 2 (1992), 151–238.

[19] Sherif Hany and Matthew Hogan. 2022. *Beyond geometry checks: Context-aware design verification.* Technical Report. Siemens EDA.

[20] M Hogan. 2023. Reduce Verification Complexity in Low/Multi-Power Designs. Mentor Graphics White Paper, online: http://www.mentor.com. [Accessed 01-September-2023].

[21] Matthew Hogan, Sridhar Srinivasan, Dina Medhat, Ziyang Lu, and Mark Hofmann. 2013. Using static voltage analysis and voltage-aware DRC to identify EOS and oxide breakdown reliability issues. *2013 35th Electrical Overstress/Electrostatic Discharge Symposium* (2013), 1–6.

[22] Jiwoo Hong, Sunghoon Kim, and Dongsuk Jeon. 2022. An Automatic Circuit Design Framework for Level Shifter Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 12 (Dec. 2022), 5169–5181.

[23] Insight EDA. 2024. Insight Anlayzer. https://insighteda.com/. [Accessed 08-July-2024].

[24] Ahmed Irfan, Alessandro Cimatti, Alberto Griggio, Marco Roveri, and Roberto Sebastiani. 2016. Verilog2SMV: A Tool for Word-Level Verification. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 1156–1159.

[25] Michael G. Khazhinsky, Shuqing Cao, Harald Gossner, Gianluca Boselli, and Melanie Etherton. 2012. Electronic Design Automation (EDA) Solutions for ESD-Robust Design and Verification. In *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference* (2012-09). IEEE, 1–8.

[26] Kishore Kollu, Trey Jackson, Farhad Kharas, and Anant Adke. 2012. Unifying design data during verification: Implementing Logic-Driven Layout analysis and debug. In *2012 IEEE International Conference on IC Design & Technology*. IEEE, Austin, TX, USA, 1–5. https://doi.org/10.1109/ICICDT.2012.6232874

[27] Kishor Kunal, Jitesh Poojary, Tonmoy Dhar, Meghna Madhusudan, Ramesh Harjani, and Sachin S. Sapatnekar. 2020. A General Approach for Identifying Hierarchical Symmetry Constraints for Analog Circuit Layout. In *Proceedings of the 39th International Conference on Computer-Aided Design*. ACM, Virtual Event USA, 1–8. https://doi.org/10.1145/3400302.3415685

[28] Hans Kunz, Gianluca Boselli, Jonathan Brodsky, Minas Hambardzumyan, and Ryan Eatmon. 2010. An Automated ESD Verification Tool for Analog Design. In *Electrical Overstress/Electrostatic Discharge Symposium Proceedings 2010*.

[29] Luke Lang. 2010. Case Study: Power-aware IP and Mixed-Signal Verification. *DVCON* (2010).

[30] Jérôme Lescot, Vincent Bligny, Dina Medhat, Didier Chollat-Namy, Ziyang Lu, Sophie Billy, and Mark Hofmann. 2012. Static low power verification at transistor level for SoC design. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design - ISLPED '12*. ACM Press, Redondo Beach, California, USA, 129. https://doi.org/10.1145/2333660.2333694

[31] Jerome Lescot, Patrice Dehan, Wahbi Boujarra, Dina Medhat, and Sophie Billy. 2015. A comprehensive ESD verification flow at transistor level for large SoC designs. In *2015 37th Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, Reno, NV, USA, 1–6. https://doi.org/10.1109/EOSESD.2015.7314740

[32] C.-H. Liu, H.-Y. Liu, C.-W. Lin, S.-J. Chou, Y.-W. Chang, S.-Y. Kuo, S.-Y. Yuan, and Y.-W. Chen. 2008. An Efficient Graph-Based Algorithm for ESD Current Path Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 8 (Aug. 2008), 1363–1375.

[33] Ziyang Lu and David Averill Bell. 2010. Hierarchical verification of chip-level ESD design rules. *Electrical Overstress/Electrostatic Discharge Symposium Proceedings 2010* (2010), 1–6.

[34] Magwel. 2023. Magwel ESDi. https://www.magwel.com/about/esdoverview/. [Accessed 08-July-2024].

Author Version

[35] Tobias Massier, Helmut Graeb, and Ulf Schlichtmann. 2008. The Sizing Rules Method for CMOS and Bipolar Analog Integrated Circuit Synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27, 12 (2008), 2209–2222.

[36] Merritt Miller and Forrest Brewer. 2013. Formal verification of analog circuit parameters across variation utilizing SAT. In *2013 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1442–1447.

[37] Maximilian Neuner and Helmut Graeb. 2019. Power-Down Mode Verification for Hierarchical Analog Circuits. In *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, Lausanne, Switzerland, 125–128. https://doi.org/10.1109/SMACD.2019.8795264

[38] M. Neuner and H. Graeb. 2020. Hierarchical Analog Power-Down Synthesis. In *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, Glasgow, UK, 1–4. https://doi.org/10.1109/ICECS49266.2020.9294889

[39] Maximilian Neuner and Helmut Graeb. 2020. Verification and revision of the power-down mode for hierarchical analog circuits. *Integration* 73 (July 2020), 1–9. https://doi.org/10.1016/j.vlsi.2020.02.009

[40] Jesse Conrad Newcomb. 2012. Circuit states. https://patents.google.com/patent/US8225251B2/en?inventor=Jesse+Conrad+Newcomb [Accessed 22-July-2025].

[41] Jesse Conrad Newcomb. 2012. Method of predicting electronic circuit floating gates. https://patents.google.com/patent/US20120110528A1 [Accessed 22-July-2025].

[42] Jesse Conrad Newcomb and Govinda Keshavdas. 2013. Method and system of automatically identifying level shifter circuits. https://patents.google.com/patent/US8595660B2 [Accessed 22-July-2025].

[43] Miles Ohlrich, Carl Ebeling, Eka Ginting, and Lisa Sather. 1993. Subgemini: Identifying Subcircuits using a Fast Subgraph Isomorphism Algorithm. In *Proceedings of the 30th International Design Automation Conference*. 31–37.

[44] Oussama Oulkaid, Bruno Ferres, Matthieu Moy, Pascal Raymond, Mehdi Khosravian, Ludovic Henrio, and Gabriel Radanne. 2024. A Transistor Level Relational Semantics for Electrical Rule Checking by SMT Solving. In *Design, Automation and Test in Europe Conference*. Valencia, Spain. https://hal.science/hal-04527225 Available at https://hal.science/hal-04527225/file/date2024.pdf..

[45] Dylan Patel. 2022. The Dark Side Of The Semiconductor Design Renaissance – Fixed Costs Soaring Due To Photomask Sets, Verification, and Validation. https://www.semianalysis.com/p/the-dark-side-of-the-semiconductor [Accessed 08-July-2024].

[46] Georg Pelz and Uli Roettcher. 1994. Pattern Matching and Refinement Hybrid Approach to Circuit Comparison. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 13, 2 (1994), 264–276.

[47] Guillaume Plassan, Hans-Jörg Peter, Katell Morin-Allory, Shaker Sarwary, and Dominique Borrione. 2017. Improving the Efficiency of Formal Verification: The Case of Clock-Domain Crossings. In *VLSI-SoC: System-on-Chip in the Nanoscale Era – Design, Verification and Reliability*, Thomas Hollstein, Jaan Raik, Sergei Kostin, Anton Tšertov, Ian O'Connor, and Ricardo Reis (Eds.). Springer International Publishing, 108–129.

[48] M Renovell and G Cambon. 1986. Topology dependence of floating gate faults in MOS integrated circuits. *Electronics Letters* 3, 22 (1986), 152–153.

[49] Rafael Santos, Joao Afonso, and Jose Monteiro. 2020. Short-circuit Analysis using a Parallel QBF Solver. In *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*. IEEE, Segovia, Spain, 1–6. https://doi.org/10.1109/DCIS51330.2020.9268636

[50] Guoyong Shi. 2013. A Survey on Binary Decision Diagram Approaches to Symbolic Analysis of Analog Integrated Circuits. *Analog Integrated Circuits and Signal Processing* 74, 2 (Feb. 2013), 331–343.

[51] Siemens EDA. 2023. Calibre PERC. https://eda.sw.siemens.com/en-US/ic/calibre-design/reliability-verification/perc/. [Accessed 08-July-2024].

[52] Peter Smith. 2010. Types of proof system. http://www.logicmatters.net/resources/pdfs/ProofSystems.pdf. [Accessed 22-July-2025].

[53] Mate Soos, Karsten Nohl, and Claude Castelluccia. 2009. Extending SAT Solvers to Cryptographic Problems. In *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings (Lecture Notes in Computer Science)*, Oliver Kullmann (Ed.). Springer, 244–257.

[54] Sridhar Srinivasan, Ellis Cohen, and Mark Hofmann. 2014. A New Approach Using Symbolic Analysis To Compute Path-Dependent Effective Properties Preserving Hierarchy. In *2014 27th IEEE International System-on-Chip Conference (SOCC)* (Las Vegas, NV, USA, 2014-09). IEEE, 404–408. https://doi.org/10.1109/SOCC.2014.6948963

[55] S. Swan. 2006. SystemC transaction level models and RTL verification. In *2006 43rd ACM/IEEE Design Automation Conference*. 90–92. https://doi.org/10.1145/1146909.1146937

[56] Synopsys. 2023. Synopsys CustomSim. https://www.synopsys.com/content/dam/synopsys/verification/datasheets/customsim-ds.pdf. [Accessed 08-July-2024].

[57] N. Trivedi and D. Alvarez. 2015. Essential - Integration of ESD Verification Methodologies. In *2015 37th Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, Reno, NV, USA.

Author version

[58] VentureBeat. 2011. A billion-dolar mistake: Intel recalls a supporting chip for popular Sandy Bridge platform. https://venturebeat.com/2011/01/31/a-billion-dollar-mistake-intel-recalls-a-supporting-chip-for-popular-sandy-bridge-platform/. [Accessed 08-July-2024].

[59] Benjamin Viale and Bruno Allard. 2020. Scalable and Versatile Design Guidance Tool for the ESD Robustness of Integrated Circuits - Part I. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 10 (Oct. 2020), 3067–3080. https://doi.org/10.1109/TCAD.2019.2962120

[60] Benjamin Viale, Mathieu Fer, Lionel Courau, Philippe Galy, Blaise Jacquier, Jerome Lescot, and Bruno Allard. 2016. An automated tool for chip-scale ESD network exploration and verification. In *2016 38th Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*. IEEE, Garden Grove, CA, USA, 1–10. https://doi.org/10.1109/EOSESD.2016.7592551

[61] H. Xue, C. Di, and J.A.G. Jess. 1994. Probability Analysis for CMOS Floating Gate Faults. In *Proceedings of European Design and Test Conference EDAC-ETC-EUROASIC*. 443–448. https://doi.org/10.1109/EDTC.1994.326838

[62] Michael Zwerger and Helmut Graeb. 2012. Short-circuit-path and floating-node verification of analog circuits in power-down mode. In *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, Seville, Spain, 241–244. https://doi.org/10.1109/SMACD.2012.6339384

[63] Michael Zwerger and Helmut Graeb. 2014. Verification of the power-down mode of analog circuits by structural voltage propagation. *Analog Integrated Circuits and Signal Processing* 78, 1 (Jan. 2014), 177–189. https://doi.org/10.1007/s10470-013-0107-x

[64] Michael Zwerger and Helmut Graeb. 2015. Detection of Asymmetric Aging-Critical Voltage Conditions in Analog Power-Down Mode. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*. IEEE Conference Publications, Grenoble, France, 1269–1272. https://doi.org/10.7873/DATE.2015.0140

[65] Michael Zwerger, Maximilian Neuner, and Helmut Graeb. 2017. Analog Power-Down Synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36, 12 (Dec. 2017), 1954–1967. https://doi.org/10.1109/TCAD.2017.2702615

Author Version