Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Entwurfsautomatisierung

# Verification and Synthesis of Analog Power-Down Circuits

Michael Alois Zwerger

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

**Doktor-Ingenieurs (Dr.-Ing.)**

genehmigten Dissertation.

| | |
|---|---|
| **Vorsitzender:** | Prof. Dr. Andreas Herkersdorf |
| **Prüfer der Dissertation:** | 1. Prof. Dr.-Ing. Helmut Gräb |
| | 2. Prof. Dündar Günhan, Ph.D. |

Die Dissertation wurde am 25.10.2016 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 23.01.2017 angenommen.

This thesis was also published as book (ISBN 978-3-8439-3119-9) by Verlag Dr. Hut, Munich, Germany

## Abstract

This thesis investigates electronic design automation algorithms for the verification and synthesis of analog power-down circuits. The methods build upon a new structural graph model that models the static circuit behavior. The verification method detects floating nodes, leakage currents and reliability problems due to electrical stress. The synthesis method allows automatic computation of a complementary power-down circuit for a given circuit.

## Zusammenfassung

Diese Arbeit behandelt Entwurfsautomatisierungsalgorithmen zur Verifikation und Synthese analoger Power-Down-Schaltungen. Die Methoden bauen auf einem neuen strukturbasierten Graphmodell auf, welches das statische Verhalten der Schaltung modelliert. Die Verifikationsmethode erkennt Floating-Nodes, Leckströme und durch elekrischen Stress verursachte Zuverlässigkeitsprobleme. Die Synthesemethode berechnet automatisch die zu einer gegebenen Schaltung zugehörige Power-Down-Schaltung.

# Vorwort (Preface)

Diese Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Entwurfsautomatisierung, Technische Universität München. Eigentlich wollte ich mich mit analoger Layout-Synthese beschäftigen, doch es kam etwas dazwischen – analoge Power-Down-Schaltungen. Das Ergebnis sind die Algorithmen für Verifikation und Synthese in dieser Arbeit.

Ich möchte mich bei allen Kollegen des Lehrstuhls für die schöne Zeit am Institut und die hervorragende Arbeitsatmosphäre bedanken. Mein besonderer Dank gilt meinem Doktorvater Professor Helmut Gräb. Bedanken möchte ich mich auch bei allen Studenten, die im Rahmen einer Abschlussarbeit oder eines Praktikums einen Beitrag zu dieser Arbeit geleistet haben. Ganz besonderer Dank geht an Maximilian Neuner für seine unermüdliche und engagierte Unterstützung beim Verbessern und Implementieren der Algorithmen im Rahmen von Bachelorarbeit, Masterarbeit, Praktika und Werkstudententätigkeiten. Bedanken möchte ich mich auch bei meinem Vorgänger Dr. Michael Eick, dessen Software zur Strukturerkennung ich übernehmen konnte und dessen "Nachhilfestunden" ein sehr hilfreicher Beitrag zu einem zügigen Start des Projekts waren. Dank gilt auch allen Kollegen der Infineon Technologies AG mit denen ich im Rahmen des Forschungsprojektes "Rely" an der Verifikation analoger Power-Down-Schaltungen arbeiten konnte. Ganz besonders bedanken möchte ich mich bei Georg Georgakos für die wertvollen Diskussionen und die Möglichkeit die Algorithmen an Beispielen aus der Praxis zu testen und bei Professor Günhan Dündar von der Boğaziçi University Instanbul für die Begutachtung der Arbeit. Zu guter Letzt möchte ich mich bei meiner Familie für die Unterstützung bedanken, ohne die diese Arbeit sicherlich nicht möglich gewesen wäre.

Das Thema "analog power down" wurde bis dato in Wissenschaft und Praxis wenig unter dem Aspekt der Entwurfsautomatisierung (insbesondere Synthese) betrachtet. Ein solche Betrachtung bietet jedoch die Chance eine ständig wiederkehrende und an und für sich sehr systematische Aufgabe zu automatisieren. Ich hoffe, hierfür einen ersten Anstoß geleistet zu haben und würde mich freuen wenn der Leser in der vorliegenden algorithmischen Werkzeugkiste ein passendes Werkzeug findet.

München im März 2017

# Contents

# 1. Introduction

The continuous shrinking of circuits has led to the integration of more and more subsystems on a single chip. The complexity of modern systems on chip (SoCs) keeps on increasing while an extremely competitive market imposes strict time-to-market requirements. Computer-Aided Design (CAD) is used in order to keep up with the growing complexity. The design chapter of the International Technology Roadmap for Semiconductors 2011 (ITRS) states that the main threat to continuation of the fast progress is still the cost of design. In the 2007 edition of the roadmap [ITRS, 2007], the productivity gap between technology capabilities and design productivity was identified as a major challenge for the semiconductor industry. The productivity gap is still growing [ITRS, 2011; Medea+/Catrene Office, 2009; Lourenço and Horta, 2012].

Modern SoCs incorporate subsystems of different domains (analog, digital, mixed-signal) on a single chip. Thereby, verification of the analog/mixed-signal (AMS) systems is becoming a major bottleneck. The ITRS 2011 lists AMS verification as "a bottleneck that has now reached crisis proportions". Furthermore, it lists "automated methods for AMS design" as needs to cope with the challenge of exponentially increasing design costs. The percentage of synthesized analog content should have reached 30 % by 2016 and is forecasted to increase to 60 % in 2022 according to the ITRS 2011 roadmap. A worldwide survey under 561 attendees of a Tech on Tour event organized by Cadence Design Systems in 2011 [Balasubramanian, 2016; Wang, 2015] identified "AMS verification" as the biggest mixed-signal methodology challenge (36 % of mentions) followed by "SoC Verification" (19 % of mentions). It can be concluded that there is an urgent need for further improved tool support for verification and synthesis of analog/mixed-signal circuits. This thesis addresses one important aspect of verification which is power-down mode verification for analog/mixed-signal designs.

Driven by the need for energy efficiency of mobile applications, medical applications, automotive applications, devices for the Internet of Things (IoT) or energy-efficient modern data centers, the complexity of the power management increases in line with the complexity of the system itself. A prerequisite for power-efficient SoCs is the availability of power-manageable modules [Medea+/Catrene Office, 2009]. This means that all modules, including analog/mixed-signal modules, have one or several power modes. Low-power design techniques are applied as power efficiency is nowadays a design goal as important as speed and area. On technology and circuit level, the techniques comprise measures like reduction of operating voltages or reduction of leakage currents by using devices with different threshold voltages (e.g. [Keating et al., 2007; Lee, 2012]). On system level, power efficiency can be increased by switching subsystems into power-down

or low-power modes while they are not in use. Techniques like frequency and voltage scaling or power gating are applied in the digital domain (e.g. Appendix III of [ITRS, 2011], [Keating et al., 2007; Lee, 2012]). The aim of frequency and voltage scaling is the reduction of dynamic currents during operation as they are the dominant cause of power consumption. In contrast, power gating aims at the reduction of static leakage currents. They are the dominant cause of power losses while the digital system is not in use. In the analog domain, power consumption can be reduced by operating the transistors in the subthreshold region. Furthermore, analog circuits are equipped with power-down circuitry that puts the circuit in a defined power-down state and cuts off the bias currents [Juneja, 2015; Blieck and Janssens, 1996; Zwerger and Graeb, 2012a].

The subsystems of an SoC can be powered down individually. However, they remain physically connected. The complexity arising from the interaction of the power-modes of different subsystems leads to high complexity of verification [Hogan, 2014]. Managing this complexity in a system with a large number of connected subsystems is challenging. Furthermore, the voltages in power-down mode can lead to device degradation due to electrical stress [Chen et al., 2001; Michael et al., 1992; Maricau and Gielen, 2011; Chouard et al., 2011; Schlunder et al., 2003; Chouard, 2012]. Reliability and device aging considerations have to be taken into account for analog/mixed-signal power-down circuitry especially for newer technologies [Zwerger and Graeb, 2015; Schlünder, 2010; Georgakos, 2014].

The complexity of power management is another brick in the wall of exploding complexity of analog/mixed-signal verification. The situation can be improved by supporting the designers with tools that take care of the systematic verification of the power-down circuitry. Additionally, it is possible to go one step beyond verification by reversing the problem: Instead of checking the correctness of a given power-down circuit, the power-down circuit can be synthesized with given requirements for a correct power-down mode. Synthesis of power-down circuitry takes away the time consuming and repetitive manual task of adding power-down switches to a given design from the designers. Advances in power-down verification and synthesis enable faster design cycles and help to avoid expensive respins due to overlooked errors.

In this thesis, new methods for the verification and synthesis of power-down circuitry for analog/mixed-signal circuits are presented. Chapter 3 describes new methods for systematic verification of analog power-down circuitry. The methods cover the verification of the basic functionality of the power-down mode ("all currents off") and reliability considerations. A new method for automatic fault-free synthesis of power-down circuitry is developed in chapter 4. It builds on the the systematic developed for verification. The methods described in this thesis help to tackle the challenge of increasing complexity of power-down circuitry and to speed up the analog design process.

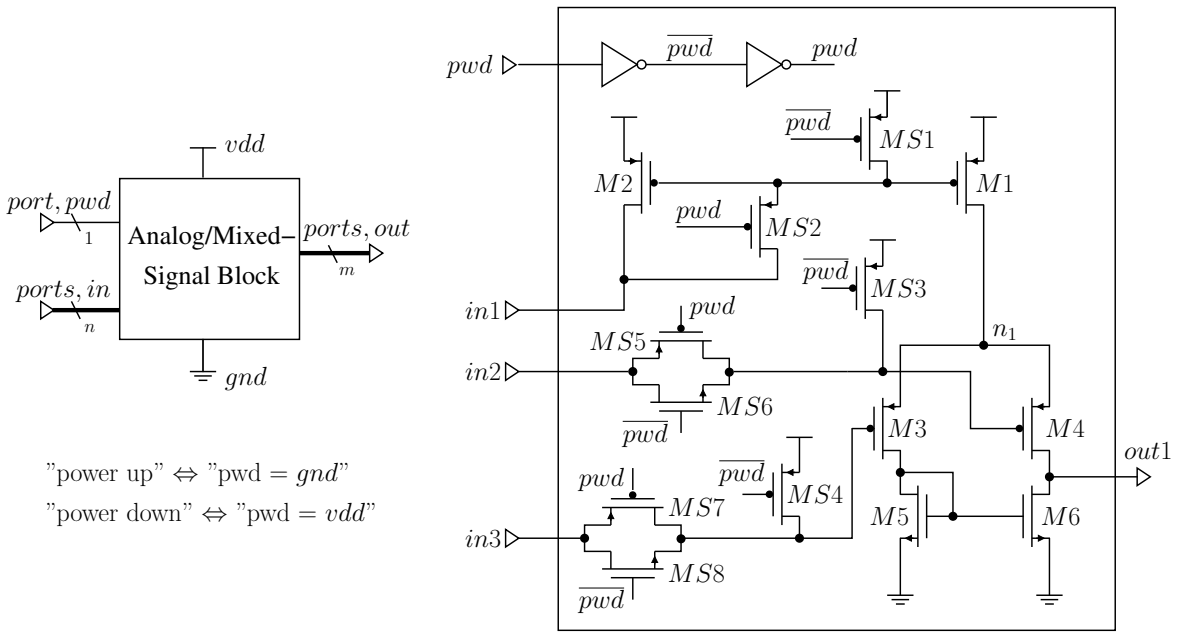



Figure 1.1.: Example of an analog/mixed-signal block with power-down circuit

power-down ($pwd = vdd$):

| | n | p |
|---|---|---|
| $pwd$ | conducting | off |
| $\overline{pwd}$ | off | conducting |

power-up ($pwd = gnd$):

| | n | p |
|---|---|---|
| $pwd$ | off | conducting |
| $\overline{pwd}$ | conducting | off |

Table 1.1.: State of the power-down switches in dependence of the signal on the gate (rows), the type of transistor (columns) and the power-mode (left or right table)

## 1.1. Power-Down Circuits

An example of an analog/mixed-signal circuit block with power-down circuitry is shown in Fig. 1.1. The symbol of a circuit block is shown on the left. A differential amplifier with power-down circuitry is shown as example on the right. The circuit block is connected to other circuitry over ports. The analog/mixed-signal circuit block has $n$ input ports and $m$ output ports. Furthermore, it has a port labeled $pwd$ that receives a power-down signal. This thesis follows the convention that $pwd = vdd$ represents the power-down mode and $pwd = gnd$ represents the power-up mode, i.e. the normal-operation mode of the circuit. The signal $pwd$ is a digital signal with high level denoted as $vdd$ and low level denoted as $gnd$. The symbol $vdd$ refers to the supply voltage and $gnd$ to the ground voltage. The inverted power-down signal is denoted as $\overline{pwd}$.

A simple amplifier circuit as shown on the right of Fig. 1.1 can be found in many designs. The example circuit receives a bias current over port $in1$. The differential input voltage between $in2$ and $in3$ is amplified to a single-ended output voltage between $out1$ and ground. The power mode is controlled by the signal on pin $pwd$.

The core part of the circuit in Fig. 1.1 is given by the transistors $M1$ to $M6$. They implement a simple differential amplifier, i.e. the power-up functionality of the circuit. In contrast, the power-down circuitry is formed by the power-down switches $MS1$ to $MS8$. This thesis follows the convention that the transistors forming the power-up part of a circuit are labeled with $Mx$ and the power-down switches are labeled with $MSx$ where $x$ is a number. Two inverters buffer the incoming $pwd$ signal and generate the inverted signal $\overline{pwd}$. The buffered $pwd$ signal as well as the inverted signal $\overline{pwd}$ are connected to the gates of the power-down switches. The switches can be conducting or off depending on the signal on the gate of the switches. There are n-MOSFETs and p-MOSFETs. They can either have the signal $pwd$ or $\overline{pwd}$ connected to their gates. This leads to four possible combinations. The states of the power-down switches for those four combinations for power-up and power-down mode are given in Table 1.1. The left table shows the states for the power-down mode and the right one for the power-up mode.

The power-down circuitry for the differential amplifier shown in Fig. 1.1 works as follows: Power-down switch $MS1$ is conducting in power-down mode and therefore pulls the gates of $M1$ and $M2$ to $vdd$. Consequently, the bias current through those two transistors is off. Switch $MS2$ is added in order to avoid current flow from $MS1$ over $in1$ over external circuitry to ground. The switch isolates the gates of $M1$ and $M2$ from external circuitry connected to $in1$. In power-up mode, $MS1$ is off and has nearly no effect on the differential amplifier. $MS2$ is conducting and the transistors $M1$ and $M2$ act as a current mirror to bias the differential amplifier. The switch $MS2$ has only to charge the gates of $M1$ and $M2$. The static bias current through $MS2$ is zero. Furthermore, no AC signal is transferred over $MS2$. Consequently, $MS2$ has nearly no effect on the operation of the differential amplifier in power-up mode. To sum up, $MS1$ and $MS2$

implement shutdown of all bias currents in power-down mode and do not significantly interfere with the circuit in power-up mode.

The switches $MS5/MS6$ and $MS7/MS8$ isolate the differential input of the amplifier from external circuitry during power-down mode. Furthermore, the switches $MS3$ and $MS4$ pull the gates of $M3$ and $M4$ to a defined voltage ($vdd$). This has two advantages: First, it makes the behavior of the circuit more predictable than it would be with floating gates of $MS3$ and $MS4$. Second, it avoids transistor aging of $MS3$ and $MS4$ due to negative temperature instability (NBTI) [Chouard et al., 2011; Schlünder, 2010; Schlunder et al., 2003].

There is one more aspect of aging that is illustrated by the differential amplifier: The switches $MS3$ and $MS4$ ensure matched power-down voltages for the p-MOS differential pair $M3/M4$. Matching is an important design principle of analog circuits. Matched devices should be as identical as possible. Mismatch can occur due to manufacturing variations and due to electrical stress. Differential pairs like $M3/M4$ are typical examples for mismatch sensitive building blocks. Mismatch caused by stress is called time dependent mismatch [Chen et al., 2001; Michael et al., 1992; Maricau and Gielen, 2011]. Time dependent mismatch can occur due to electrical stress in power-down mode [Zwerger and Graeb, 2015; Georgakos, 2014]. The likelihood of extreme stress voltages in power-down mode is even higher than during power-up mode as internal nodes of the circuit are pulled to the extreme voltages $gnd$ and $vdd$. The pattern implemented by the switches $MS3$ to $MS8$ avoids time dependent mismatch caused by electrical stress during power-down mode. Therefore, it increases the lifetime and reliability of the circuit. It is a design decision if the extra effort is necessary. The decision depends on the design-style, the aging behavior of the used used technology and the reliability/lifetime requirements of the application.

In power-down mode, internal nodes can be floating. A floating node is a node that has only high resistive connections to other nodes of the circuit. The voltages of floating nodes are hard to determine by numerical circuit simulation, e.g. SPICE [Nagel, 1975]. For instance, the internal node $n_1$ in Fig. 1.1 is completely isolated from the rest of the circuit in power-down mode. Its voltage depends on the subthreshold behavior of $M1$, $M3$ and $M4$ and on possible other leakage paths on the chip (e.g. through the substrate). Voltages of floating nodes can be very sensitive to manufacturing variations and on circuit properties that are not modeled with sufficient accuracy, e.g. the subthreshold region of the transistors or parasitic resistances. Hence, the voltage calculated for a floating node like $n_1$ by a numerical circuit simulator may not be trustworthy. It follows that verification of the circuit by numerical circuit simulation is problematic in presence of floating nodes. However, floating nodes in power-down mode can be determined by the circuit structure similarly to the way a designer would identify floating nodes by inspecting a circuit.

In this thesis, new methods for the verification and synthesis of analog power-down circuits are developed. The verification method takes into account the problems induced

by floating nodes as described in the previous paragraph. It further takes into account reliability considerations, i.e. transistor aging. The insights gained from formulating verification rules can be used as starting point for investigating the possibilities of automatic synthesis of power-down circuitry. Verification becomes obsolete if automatic synthesis is possible because the circuitry is systematically created by a computer program. Algorithms for both, verification and synthesis, should be easy to use. It is further desirable that they allow an interactive, semi-automatic design style. In the next section, the state of the art of analog power-down verification and synthesis is discussed and contributions of this thesis are highlighted.

## 1.2. State of the Art and Contributions

Verification of the analog power-down mode and synthesis of analog power-down circuitry was not subject of intensive research so far. However, there are plenty of published methods for verification and synthesis of analog circuits. Consequently, those methods will be surveyed in the this section with respect to suitability to solve the analog power-down verification or synthesis task. The relationship of the established methods to the methods prensented in this work is pointed out. Contributions of this thesis to the state of the art are highlighted for each topic area if applicable.

### 1.2.1. Verification of Analog Circuits

#### Numerical Circuit Simulation

Traditional analog verification builds on numerical circuit simulation. Numerical circuit simulation started with the SPICE simulator in the 1970s [Nagel, 1975]. Today, many commercial and mature simulation tools are available, e.g. [Cadence Design Systems, 2015b; Mentor Graphics, 2015b; Synopsys, 2015b]. Furthermore, there are nowadays simulators that are optimized for full-chip simulation. These simulators are usually referred to as "Fast Spice" simulators. They apply optimization techniques to speed up simulation in order to make full-chip simulation feasible [Rewienski, 2011]. There are also "Fast Spice variants" for each of the three commercial simulation tools mentioned above [Cadence Design Systems, 2015a; Mentor Graphics, 2015c; Synopsys, 2015a].

The behavior of the power-down mode can be characterized by the static behavior of the circuit. All nodes should settle at a defined value and the static current should be zero. Nodal analysis (DC analysis) as implemented by state-of-the-art circuit simulators can be considered for this verification task [Ho et al., 1975]. However, floating nodes cause values of node voltages that may not be trustworthy. This problem is especially significant if the power-down state of a circuit is examined. Many transistors will operate in the subthreshold region due to the nature of power-down mode: It cuts off the bias currents.

Modern simulators apply techniques like gmin-stepping or source stepping [Yilmaz and Green, 1999] in order to achieve convergence in presence of floating nodes. However, the problems mentioned in section 1.1 remain unresolved: The voltages of floating nodes can be highly dependent on manufacturing variances and effects like leakage currents that are very hard to model completely. An example of misleading DC simulation results is shown in section 3.2.7.

The transitions between power-up and power-down mode are characterized by the transient behavior of the circuit. The settling times of all nodes have to be appropriate for a given application. Floating nodes appear in the result waveforms of transient simulation [Cadence Design Systems, 2015b; Mentor Graphics, 2015b; Synopsys, 2015b] as nodes with extremely large settling times. It is non-trivial to write assertions that systematically discover those occurrences. An example is discussed in section 3.2.7. More importantly than the problem of defining adequate assertions, the problem of incomplete modeling and high sensitivity to manufacturing variances also applies for transient simulation. Additionally, transient simulation depends on the input stimuli. A complete verification by exhaustive transient simulation taking into account all (or approximately all) input stimuli is impossible due to exploding computational effort.

### Symbolic Analysis

Other circuit models than those used for numerical simulation, e.g. symbolic analysis, can be considered in order to avoid the above mentioned problems. Research on symbolic analysis was particularly active in the late 1980s and beginning of the 1990s. An overview is given in [Rutenbar et al., 2002]. Symbolic analysis does not solve equation systems or differential equations as it is done by numerical simulation. Instead, it tries to approximate the circuit behavior with symbolic expressions. A model of the the circuit is generated that gives insight in the dependencies between performances and parameters. The term symbolic analysis usually refers to methods that approximate AC transfer functions of the circuit, e.g. [Fernández et al., 1991; Starzyk and Konczykowska, 1986; Gielen et al., 1989; Li and Gu, 1992; Wambacq et al., 1995; Yu and Sechen, 1995; Wambacq et al., 1999; Shi and Tan, 2000]. There are also methods that approximate the time-domain (transient) behavior of the circuit, e.g. [Kobylarz, 1969; Hassoun and Ackerman, 1990; Yu et al., 2014; Liu et al., 2012; Grabowski et al., 2006; Yang et al., 2005; Zhang and Shi, 2011]. Transient symbolic methods can be used for efficient characterization of the impact of process variations on the transient behavior [Yu et al., 2014; Liu et al., 2012; Grabowski et al., 2006]. Two of the mentioned papers claim that the method is also capable of doing symbolic DC analysis in their abstract. However, in [Kobylarz, 1969] it is stated that currently only transient analysis is available and no DC results are shown. The method in [Hassoun and Ackerman, 1990] suggests that the frequency should be set to zero in AC analysis for a DC characterization. However, the presented results do not explicitly give insight in the DC behavior of the used example circuits.

It can be summarized that symbolic analysis mainly focuses on frequency domain behavior. The frequency domain methods are suitable for power-up mode characterization but not for power-down mode characterization. Methods for static and transient symbolic analysis can be considered to solve the power-down verification task. However, static methods are barely available and the available transient methods aim at a higher accuracy of characterization than actually needed to describe the power-down mode behavior. It will be shown in section 3.2 of this thesis that the power-down mode behavior of analog circuits can be meaningfully approximated by using discrete (digital) voltages and a static approach. The approximation uses values of vdd and ground to describe the power-down mode behavior. It turns out that a discrete approximation of the behavior is robust and does not generate misleading results as other methods may do in presence of floating nodes.

### State-Space Oriented Verification and Formal Methods

Above, the problem of exploding computational effort of transient analysis of large systems with many inputs and input stimuli was addressed. In analog verification, methods for state-space oriented verification are an active field of research (e.g. [Steinhorst and Hedrich, 2012c; Dastidar and Chakrabarti, 2005]). Those methods are subject to the state-space explosion problem, i.e., it is practically impossible to cover the whole state-space with transient simulation. Consequently, those methods focus on intelligent measures to circumvent the problem, i.e. by intelligent stimuli generation [Steinhorst and Hedrich, 2010, 2012b; Ahmadyan et al., 2013; Yin et al., 2013]. On the other hand, if adequate coverage of the state-space is ensured, the remaining (non trivial) problem is checking if the circuit fulfills the specified (and sometimes non-specified but essential) requirements. The problem is addressed by formal verification methods for analog circuits. A survey is given in [Zaki et al., 2006]. It categorizes the methods into "equivalence checking", "model checking" and "reachability analysis", "runtime verification methods" and "deductive methods". In equivalence checking, the circuit is checked for equivalent behavior with a reference circuit or an abstract behavioral model. In model checking and reachability analysis, the reachable set of the state space is approximated. Those methods are suitable for ensuring that the system does not enter into a bad state. However, they suffer from the state-space explosion problem.

Runtime verification tries to overcome this problem by omitting the step of creating a computational model. The required properties of the circuit have to be monitored / asserted on the result waveforms as no model for comparison is available. The list of publications is long and the methods are diverse. For example, in [Steinhorst and Hedrich, 2012a] a method that applies verification in the state-space using the Analog Specification Language (ASL) is presented. In [Bonfini et al., 2006], a framework that helps to reuse techniques from state-of-the-art digital verification (e.g. automatic results collection, coverage elaboration, stimuli generation) is presented. In [Dastidar and Chakrabarti, 2005], an "Analog Conventional Temporal Logic (Ana CTL)" which

can be used to specify complex properties of the transient behavior of analog systems is presented. In [Maler and Nickovic, 2004], a temporal logic that is used for specifying the behavior of time discrete systems is adapted for the continuous case. More examples can, e.g., be found in [Zaki et al., 2006].

The last category of the analog formal verification methods mentioned in [Zaki et al., 2006] are the deductive methods. Deductive methods try to describe the circuit behavior with logic equations and use formal deduction to prove that the circuit implies the specified behavior. For instance, Hanna describes a predicate logic based on rectilinear specifications of the device behavior as a conservative approximation of the real device behavior [Hanna, 2000]. In [Ghosh and Vemuri, 1999], an approach for formal verification of the DC behavior of analog designs containing linear and piecewise linear components is presented. It checks the circuit against a user given specification using the higher order logic proof checker PVS. The principle of the logic description of the devices is similar to [Hanna, 2000]. However, the device equations on currents and voltages are used instead of rectangular approximations.

A deductive approach like [Ghosh and Vemuri, 1999] or [Hanna, 2000] can be considered for formal verification of the power-down mode behavior. A method for automatic deduction of the proposed logic and for automatic checking against the specification needs to be developed for this purpose. However, the method in [Hanna, 2000] is limited to the description of the theory. A case study showing the automated applicability of the rectilinear logic to larger circuits is missing. Such a case study is given for circuits containing up to five op amps in [Ghosh and Vemuri, 1999]. The examples are telephone receiver and transmitter circuits as well as a not further specified circuit named “Neural Contr”. The runtimes are “of the order of a few minutes” on a Sun workstation of this era.

A formal specification of the power-down mode behavior is required if formal verification is used. Defining the specification is a non trivial task. To the best of my knowledge, there is no prior publication on this topic. The research for this thesis investigated if voltage propagation (see section 3.2) can be replaced by a description of the circuit in terms of logic equations [Vlachas, 2014]. Those equations should then be used to prove that no current is flowing and no floating nodes are present in power-down mode using a constraint program solver [Christian Schulte and Z.Lagerkvist, 2013]. Four different constraint programs have been investigated as alternative to the voltage propagation method described section 3.2. The investigations have shown that this approach is feasible, but finding the correct set of logic equations is a challenging task and the approach has the following limitations: The answer of the constraint-program solver is “false” if the circuit has a design error. No further information for diagnosis is available at this point and it is hard to locate the design error or to get further insight into the circumstances that caused the error. While working on this thesis, it was further investigated if this problem can be solved by adding further conditions for locating short-circuit paths in power-down mode. The results show that this is principally possible [Vlachas,

2014]. However, the automatic creation of the logic equations tends to become very complicated. The simplest, most robust and fastest constraint program was published in [Zwerger et al., 2015b]. The other three constraint programs documented in [Vlachas, 2014] can be seen as steps towards formal verification of the power-down mode. However, the voltage propagation method described in section 3.2 has proven to be the superior solution in terms of practicability and robustness. The main advantage of voltage propagation and short-circuit search as described in section 3.2 is the fact that intermediate results can be used for diagnosis in the case design errors are detected.

### Macromodeling

Macromodeling tries to abstract the circuit behavior by leaving out details in order to speed up the evaluation of the system performance. Macromodels can, e.g., be divided into black box models (e.g. [Boyle et al., 1974]) that model the dependence of performance attributes on design parameters and structural models [Ferent and Doboli, 2013; Wei and Doboli, 2008]. There are many diverse approaches for model generation. Following the introduction of [Ferent and Doboli, 2013], there are methods for modeling a large variety of performance attributes including AC performances, weakly nonlinear performances or large-signal nonlinear-related attributes. The models can be grouped by the used modeling techniques, e.g. regression analysis, symbolic analysis (as discussed above), neural networks, piecewise-linear approximation, optimization-based symbolic descriptions, support vector machines, system identification or model-order reduction. References to example methods are given in [Ferent and Doboli, 2013].

For power-down mode verification, we are mainly interested in methods that model the static behavior of the circuit without simulation. In [Wei and Doboli, 2008], it is concluded that "present modeling methods are mostly for black-box modeling in the frequency domain". Those models are not suitable for power-down verification. However, structural models are candidates for power-down mode modeling. A structural model for systematic comparison of circuits is presented in [Ferent and Doboli, 2013]. The comparison focuses on "DC gain, bandwidth, noise, CMRR and sensitivity", hence, on small signal features. In [Wei and Doboli, 2008], a structural model in the frequency domain is presented. To the best of my knowledge, there is no publication on macromodeling of static circuit performances that is not simulation based. However, there are other non-simulation based approaches for static analysis. Those are switch-level simulation and voltage propagation. Both approaches rely on some kind of static modeling. The approaches are described in the next two paragraphs.

### Switch-Level Simulation

"Modeling a MOS circuit as a network of simple switches began in 1979 with the development of the Mossim simulator. Since that time, dozens of switch-level simulators have

been implemented" [Bryant, 1987]. The survey of switch level algorithms in [Bryant, 1987] mentions "logic and fault simulation", "formal hardware verification", "timing analysis" and "automatic test pattern generation" as applications. Research on switch level simulation was particularly active during the late 1980s and early 1990s (e.g. [Adler, 1991; Meyer and Camposano, 1993; Blaauw et al., 1991; Vandris and Sobelman, 1990; Huang and Bryant, 1993; Dahlgren, 1995; Jones and Blaauw, 1994; Kravitz et al., 1991; Asai and Suzuki, 1992; Ramachandran, 1986; Svensson and Tjarnstrom, 1988; Hajj, 1990; Bryant, 1988; Yuan and Svensson, 1988]) The research focused on achieving speed-up / efficiency [Adler, 1991; Meyer and Camposano, 1993; Blaauw et al., 1991; Vandris and Sobelman, 1990; Jones and Blaauw, 1994; Kravitz et al., 1991; Asai and Suzuki, 1992; Bryant, 1988], improving the switch-level model [Huang and Bryant, 1993; Dahlgren, 1995; Ramachandran, 1986; Svensson and Tjarnstrom, 1988], investigating theoretic aspects [Hajj, 1990; Adler, 1991] or investigating other applications like speed optimization of CMOS circuits (e.g. [Yuan and Svensson, 1988]).

Switch level simulators model the static behavior of the circuit by a structural approach. Usually, the circuit node voltages are represented by ternary logic values, i.e. 0, $X$, and 1. The value $X$ defines an undefined state (see e.g. [Bryant, 1984; Adler, 1991]). Furthermore, timing behavior can be modeled by taking into account node capacitances and a repeated evaluation of the static switch-level model. The ideas of switch-level simulation are candidate ideas to be adopted for power-down mode verification. There are similarities to the graph model presented in section 3.2. However, the difference between the models developed for switch level simulation and the graph model from section 3.2 are the properties that are abstracted. Switch level simulators abstract the logic behavior (including timing) of digital circuits whereas the graph model developed in section 3.2 abstracts the static DC behavior of an analog circuit. The commonly used approach to abstract the DC behavior of analog circuits used in state-of-the-art tools is voltage propagation. It is described in the next paragraph.

#### Voltage Propagation

Voltage propagation is used in analog circuit verification for electrical rule checks. For instance, electrical rule checks can comprise checks for correct power-domain transitions, e.g. [Hogan, 2014; Medhat, 2009, 2012; Lescot et al., 2012; Pompl et al., 2006], or for correct protection against electro static discharge (ESD), e.g. [Darfeuille, 2012; Medhat, 2009; Ker et al., 2011; Lu and Bell, 2010; Pompl et al., 2006]).

Static checks and voltage propagation are implemented in commercial analog/mixed-signal verification tools. The tool "CustomSim Circuit Check (CCK)" from Synopsys implements, among other features, "Design and Electrical rule checks" and "Low-power design and leakage detection" [Synopsys, 2015]. The description of the "Design and Electrical rule checks" claims that the tool can, among other plausibility checks, detect floating gates before simulation. Furthermore, it monitors excessive currents during

simulation. The description of the "Low-power design and leakage detection" claims detection of leakage current paths, monitoring of standby current and identification of high-Z nodes *during* simulation. According to the product description, CCK's features are comparable to the voltage propagation and short-circuit detection presented in section 3.2 of this thesis. However, to the best of my knowledge, no public description of the details of the used methods is available. Furthermore, the tool still partially relies on numerical circuit simulation which can be problematic.

The tool PERC ("Programmable Electrical Rule Check") from Mentor graphics [Mentor Graphics, 2015a] features methods, among others, to detect electrical overstress (EOS) at thin-oxide devices, to verify ESD protection rules and electrical rule checks for multi power-domain designs [Mentor Graphics, 2016]. Applications of PERC are, e.g., described in [Lu and Bell, 2010; Hogan, 2014; Lescot et al., 2012]. PERC can propagate voltage in a vectorless mode as well as in a vectored mode. The vectored mode is comparable to the voltage propagation described in section 3.2 of this thesis. As PERC is programmable, the propagation rules are user input. To the best of my knowledge, no description of a rule set with similar completeness as the graph model described in section 3.2 is publicly available. It would be possible to implement the propagation rules of this thesis within the PERC framework. Evaluating potential runtime improvements of this approach compared to the experimental implementation in this thesis is left for future work. Furthermore, vectorless propagation was not investigated in this thesis. It is a candidate to solve the voltage enumeration problem described in section 3.3.6 in future work.

To the best of my knowledge, there is only one prior academic publication that describes a voltage propagation approach [Blieck and Janssens, 1996]. The algorithm was developed for power-down mode verification. The paper describes a basic version and an elaborated version of the algorithm. The basic version is limited to resistors and MOS devices, i.e., bulk diodes cannot be considered. Furthermore, there is no guarantee that all potential short-circuit paths are found. These limitations are solved by the elaborated version. It additionally supports diodes and guarantees the detection of all potential short-circuits. Compared to the elaborated version, the voltage propagation method described in section 3.2 of this thesis makes the following new contributions:

- A precise, unified, graph-based description of the algorithm is given.
- Propagation rules for all common analog components are given including diodes and bipolar transistors.
- Improved modeling of diode-connected and off-connected transistors.
- The runtime complexity is analyzed.
- The detected short-circuit paths are categorized into definite paths, potential paths and induced path in order ease locating the root cause of the problem for diagnosis.
- Experimental results for larger, hierarchical designs are given.

Voltage propagation is a fundamental basis for the verification of power-down reliability constraints as described in section 3.3. Voltage propagation and short-circuit detection

were originally published in [Zwerger and Graeb, 2012a,b, 2013]. Section 3.2 presents an extended version.

### Constraint-Driven Verification and Constraint Generation

State-of-the-art design tools support constraint-driven design. An overview over this design paradigm can, e.g., be found in the introduction of [Meyer zu Bexten et al., 2015]. Constraint-driven design is supported by state-of-the-art design flows. For instance, the Cadence Virtuoso Design Framework that is one of the most popular development frameworks for full custom IC design provides functions to store, manage, create and verify design constraints. There are many different types of constraints, e.g. electrical constraints, placement constraints or routing constraints. A typical class of constraints are matching constraints which ensure matching between circuit elements. Matching constraints can be written down for sizing, layout, routing, etc. In this thesis, matching constraints for reliability of the power-down mode are developed. They are described in section 3.3 along with a method for automatic verification.

Constraints can be annotated manually by the designer or automatically generated. Symmetries in the circuit need to be computed for automatic generation of matching constraints. Basic symmetries can be computed for pairs of devices, e.g. for differential pairs. Therefore, basic building blocks (e.g. differential pairs) need to be detected in a given netlist. Pattern matching can be used for this purpose. It finds given subcircuits, i.e. building blocks, in a given netlist, e.g. [Lescot et al., 2012; Meyer zu Bexten et al., 2015; Pelz and Roettcher, 1994; Rubanov, 2006; Ohlrich et al., 1993; Ebeling, 1988]. Pattern matching usually faces the problem of ambiguities while detecting subcircuits. A specialized method that resolves ambiguities for different types of current mirrors and differential pairs was published in [Massier et al., 2008; Massier, 2010]. The original intent of this method was the automatic generation of sizing constraints. Nevertheless, the algorithms can also be used for automatic generation of other classes of matching constraints. The algorithm was extended for mixed-signal applications and improved in terms of efficiency in [Eick, 2013]. The development of the algorithm was continued for this thesis. In chapter 2 an improved version is presented. It meets the requirements of automatic constraint generation for power-down mode reliability for hierarchical designs.

The key contributions to structure recognition can be summarized as follows:

- The ambiguity arbitration mechanism of [Massier et al., 2008; Massier, 2010] and [Eick, 2013] is simplified/generalized without deteriorating recognition quality.
- A new library for detecting components forming the power-down signal path is developed.
- The implementation was improved. It was integrated into an industrial design flow using the Open Access Database [Si2 Consortium, 2013]. For the first time, the

efficiency of the method published in [Eick, 2013; Massier et al., 2008; Massier, 2010] is demonstrated by experimental results for industrial, hierarchical circuits.

The improved structure recognition and the voltage propagation from section 3.2 form the basis for the power-down reliability verification in section 3.3.

Hierarchical pattern matching is also subject of [Milošević, 2009]. The PhD thesis proposes "Virtually Flattened Views" that make hierarchical data accessible as it would be after flattening. The approach is a candidate for further speed-up of structure recognition in future work. The implementation of the structure recognition approach discussed in chapter 2 is able to read hierarchical circuits. The designs are internally flattened during a prepossessing step. Consequently, the implementation approach presented in [Milošević, 2009] is a candidate approach to be investigated for further speedup.

### 1.2.2. Power-Down Verification

Fig. 1.2 shows a typical state-of-the-art design flow for analog power-down circuit design. First, the core part of the circuit that implements the basic analog functionality of the circuit block is designed. The power-down circuitry is manually added towards the end of the design process. The functionality of the power-down mode is then checked by a manual design review or by a tool supported flow using tools like Mentor PERC [Mentor Graphics, 2015a] or Synopsys Circuit Check [Synopsys, 2015]. The practices may vary from company to company and are not publicly documented. To the best of my knowledge, the only publication that explicitly deals with power-down verification is [Blieck and Janssens, 1996]. It describes a very basic voltage propagation approach as mentioned above. Furthermore, no previous publication on the verification of power-down reliability matching constraints is known to the author. Power-down reliability verification as described in section 3.3 was developed as part of a joint research project together with Infineon Technologies. It was first published in [Zwerger and Graeb, 2015]. The approach was patented in [Georgakos, 2014]. Section 3.3 presents an extended version of [Zwerger and Graeb, 2015].

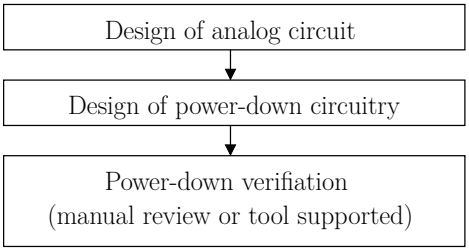



Figure 1.2.: Typical design flow for power-down circuitry

The state-of-the-art design flow shown in Fig. 1.2 will be extended in this thesis (see Figs 3.1 and 4.1). Improved tool support for power-down verification is subject of chapter 3 (see Fig 3.1).

The main contributions of this thesis to power-down verification can be summarized as follows:

- A new and complete method for floating node and short-circuit detection based on voltage propagation is presented
- The first method for reliability verification of the power-down mode is presented.
- The efficiency, effectiveness and practicability of the method is demonstrated on industrial example circuits.

### 1.2.3. Analog Circuit Synthesis

In chapter 4 (see Fig. 4.1), methods for power-down synthesis are discussed. Therefore, this section summarizes the state of the art in analog circuit synthesis.

Synthesis of analog circuits is a field of research that was particularly active in the 1990s. An overview over publications of this era, e.g. [Degrauwe et al., 1987; Harjani et al., 1989; Gielen et al., 1990; Harvey et al., 1992; Maulik et al., 1995; Antao and Brodersen, 1995; Kruiskamp, 1995; Plas et al., 2001; Crols et al., 1995; Medeiro et al., 1994; Ochotta et al., 1996; Phelps et al., 2000; Antreich et al., 2000; Hershenson et al., 2001; Koh et al., 1990], is given in [Rutenbar et al., 2002].

There are knowledge based approaches, optimization based approaches and approaches that are a mixture of both categories. Knowledge based approaches formalize expert knowledge in order to exploit it for automatic synthesis, e.g. [Degrauwe et al., 1987; Harjani et al., 1989]. The optimization based approaches can be classified with respect to the used optimization method and the used circuit model. Optimization approaches can be, e.g., simulated annealing, geometric programming, genetic algorithms or deterministic optimization methods. The circuit models can be, e.g., symbolic models, behavioral models or numerical models (see also section 1.2.1).

Research on analog circuit synthesis is still ongoing. For example: In [Jiao et al., 2015], new mining techniques for analog circuit design metaknowledge are proposed. A new reasoning-based method is proposed in [Ferent et al., 2012]. A new method based on evolutionary optimization and numerical simulation is presented in [Lourenço and Horta, 2012]. The introduction of the paper gives a comprehensive overview over previously published analog synthesis approaches including many of the above cited examples. A fully deterministic, rule-based, constructive synthesis method is proposed in [Meissner and Hedrich, 2015].

All the mentioned methods for analog circuit thesis have in common that they synthesize the power-up behavior of the circuit. None of the methods is dealing with analog power-down circuit synthesis.

### 1.2.4. Power-Down Synthesis

Power optimization is incorporated into state-of-the-art digital synthesis tools [Keating et al., 2007]. However, digital power-*optimization* techniques require different approaches and should not be confused with power-*down* techniques. In the digital world, circuit blocks are usually powered down by power gating or clock gating. Retention cells are added at the block boundary in order to avoid problems in other blocks that are connected to the block that is powered-down.

The analog power-down mode has completely different requirements. Power gating would introduce head or food switches and reduce the available voltage swing. This is not feasible in times of ever shrinking supply voltages. Furthermore, special requirements like the reliability considerations from section 3.3 have to be taken into account.

To the best of my knowledge, no prior work on power-down circuit synthesis for analog circuits was published so far. The ideas presented in chapter 4 were originally published in [Zwerger et al., 2015a]. Chapter 4 presents an extended and improved version of the method.

The main contribution of this thesis to power-down synthesis can be summarized as follows:

- The first method for the synthesis of analog power-down circuits is presented.
- Experimental results demonstrate the effectiveness and efficiency of the method for amplifier circuits (CMOS and BiCMOS), an oscillator circuit and an industrial LVDS driver circuit.
- The practicability of the method is demonstrated. It is shown that the method can easily be adopted to augment state-of-the-art industrial design flows by adding a schematic backannotation step.

## 1.3. Previous Publications

Parts of the methods and ideas of this thesis were previously published. The voltage propagation based floating node and short-circuit verification from section 3.2 was originally published in [Zwerger and Graeb, 2012a,b, 2013]. A formal approach was investigated in [Zwerger et al., 2015b] as an alternative to voltage propagation. The method for power-down reliability verification described in section 3.3 was partially published in [Zwerger and Graeb, 2015]. The power-down synthesis approach described in chapter 4

was partially published in [Zwerger et al., 2015a]. Simulation results for power-down synthesis were published in [Zwerger et al., 2016]. Student theses [Ferraria, 2013; Neuner, 2015; Shrivastava, 2016; Neuner, 2013; Schillmaier, 2014; Vlachas, 2014] and internship projects [Leitao, 2012b,a; Meyer-Jungclaussen, 2014; Mlayeh, 2012] contributed to the results presented in this work.

## 1.4. Overview over this Thesis

This thesis is structured as follows: Power-down verification is subject of chapter 3. The chapter deals with the following two main topics:

1. A method to detect floating nodes and short circuits in power-down mode is presented in section 3.2. It is based on a voltage-propagation method and circumvents the problems of numerical simulation. It uses a graph-based model to model the static behavior of the circuit in power-down mode.
2. A method for detecting asymmetric device stress in power-down mode is described in section 3.3. It requires a voltage propagation method (e.g. section 3.2) and a structure recognition method (e.g. chapter 2).

Chapter 4 presents the new method for power-down synthesis. The graph model from voltage propagation (section 3.2), structure recognition (chapter 2) and the rules developed for detecting asymmetric device stress (section 3.3.5) are the basis of the power-down synthesis method.

Experimental results for each of the mentioned methods are presented at the end of the corresponding sections/chapters. The experimental results for structure recognition are given in section 2.4. The experimental results for floating-node and short-circuit detection are described in section 3.2.7. Section 3.3.7 presents the results of the detection of asymmetric stress. The experimental results of power-down synthesis are discussed in section 4.4.

The thesis concludes with chapter 5.

# 2. Structure Recognition

## 2.1. Motivation and Problem Description

Structure recognition plays an important role in design automation for analog circuits. Typical applications are the automatic generation of constraints for a constraint-driven design style or for automatic sizing methods, e.g., [Massier et al., 2008]. Furthermore, structure recognition methods can be found in commercial verification tools (e.g. [Mentor Graphics, 2015a]) to support verification tasks like electrical rule checking for reliability, ESD protection, voltage domain checks and other tasks. In this thesis, structure recognition is required for the detection of asymmetric voltages in power-down mode as described in section 3.3 and for power-down synthesis as described in chapter 4. For the detection of asymmetric voltages, common analog building blocks and blocks forming the power-down signal path need to be identified. Current mirrors need to be detected for the current mirror rip-up heuristic used in power-down synthesis (see section 4.3.3). In general, any structure identification method can be used for those applications. An algorithm based on the sizing rules method [Massier et al., 2008; Massier, 2010; Eick, 2013] is used in order to generate the experimental results presented in this thesis. The structure recognition method was refined in order to meet the requirements for large hierarchical circuits. Experimental results for circuits with sizes up to 20.000 devices are presented in section 2.4. Hence, this is the first work to present experimental results that demonstrate the efficacy and efficiency of the sizing rules method for industrial analog cells.

## 2.2. Enhanced State-of-the-Art Algorithm

In this chapter, the structure recognition algorithm used in this thesis is described. It is based on the sizing rules method. The sizing rules method was originally published in [Massier et al., 2008]. A more detailed description can be found in [Massier, 2010]. The original algorithm was refined in [Eick, 2013]. The algorithm presented in this section is based on the refined version. This chapter outlines the main ideas of structure recognition with emphasis on the enhancements that were made compared to the state-of-the-art versions.

**Algorithm 1** StructureRecognition

**Require:** circuit netlist with devices $D$, supply rail information, Library $L$

```
// the building blocks of level 0 are the devices
B := D

// find arrays of devices on level 1
B_1 := {}
for all array types t_array in hierarchy level L_1 do
   B_1 := B_1 ∪ FindArrays(t_array, B) // see [Eick, 2013], section 3.2.2
end for
B := B ∪ B_1

// find pairs on level 2 and above
for l = 2 to maxLevel(L) do
   B_l := {}
   for all pair types t_pair in hierarchy level L_l do
      B_l := B_l ∪ FindPairs(t_pair, B) // see [Eick, 2013], section 3.2.1
   end for
   B := B ∪ B_l

   // arbitration enhanced and simplified compared to [Eick, 2013]:
   B := SelfArbitration(B, l) // Algorithm 2
   B := CrossCoupledPairArbitration(B, l) // Algorithm 3
end for

// final removal of uncertain building blocks
B := SelfArbitration(B, maxLevel(L) + 1)

return B
```

The main enhancements can be summarized as follows:

- The removal of uncertain building blocks described in [Eick, 2013] and the dominance relation introduced in [Massier et al., 2008; Massier, 2010] have been combined to a self arbitration step and a minimalistic dominance rule for the cross coupled pair.
- A new building block library (see Figs. 2.2 and 2.3) for detecting building blocks forming the power-down signal path was developed.
- The implementation has been improved in terms of suitability for industrial sized hierarchical circuits. Experimental results demonstrate the efficiency of the method for industrial sized circuits.

The structure recognition algorithm is given in Algorithm 1. It finds the building blocks

given in Figs. 2.1, 2.2 and 2.3. Fig. 2.1 shows the library $L_{analog}$ to detect typical analog building blocks, i.e., different types of current mirrors and differential pairs. The library is based on the library presented for the sizing rules method [Massier et al., 2008]. An additional library $L_{pwd}$ was developed for the detection of asymmetric voltages in power-down mode (section 3.3). It is illustrated in Figs. 2.2 and 2.3 and can be used to detect typical building blocks that can be found on the power-down signal path in an analog circuit. The libraries are organized in levels. Each building block only consists of building blocks from lower levels.

Algorithm 1 requires the circuit netlist with devices $D$, the supply rail information and a library $L$ as input. The library $L$ is either $L_{analog}$ or $L_{pwd}$. The supply rail information is described later in this section. Building blocks are added to a set of building blocks $\mathcal{B}$ during the recognition process. The algorithm starts with building blocks on the lowest level, i.e. on level $l = 0$. The building blocks in level 0 are given by the devices of the circuit $D$, i.e., they are resistors and transistors (line 2).

The algorithm continues with the detection of the array building blocks on level $l = 1$. (see lines 4 to 9). Array building blocks are building blocks which are formed by $n$ children of the same type. For both, $L_{analog}$ and $L_{pwd}$, level $l = 1$ consists of array building blocks only. For each array type $t_{array}$ on level 1, i.e. for normal arrays ($na$) and diode arrays ($da$) in case of $L_{analog}$ and for normal arrays ($na$) and resistor arrays ($ra$) in case of $L_{pwd}$, matching structures in the circuit are detected by the function FindArrays($t_{array}$, $\mathcal{B}$) (see line 7). The details of the efficient rule checking procedure that is performed FindArrays($t_{array}$, $\mathcal{B}$) are described in [Eick, 2013]. The function returns the recognized arrays. They are first added to the set of recognized building blocks on level 1, i.e. to $\mathcal{B}_1$, and then accumulated in $\mathcal{B}$.

On the levels $l = 2$ and above, the libraries $L_{analog}$ and $L_{pwd}$ contain only pair building blocks. Pair building blocks are building blocks that consist of exactly two children. The pair building blocks of type $t_{pair}$ are recognized for level 2 and above (lines 11 to 22). The algorithm is iterating from level 2 up to the highest level in the library. On each level, for each type $t_{pair}$ on that level (as given in Figs. 2.1, 2.2 and 2.3), the matching structures in the circuit are detected by a function FindPairs($t_{pair}$, $\mathcal{B}$) (line 15). The details for the efficient rule checking that is performed in FindPairs($t_{pair}$, $\mathcal{B}$) are described in [Eick, 2013]. The function FindPairs($t_{pair}$, $\mathcal{B}$) returns the matching pair building blocks. They are first added to the set for the current level $\mathcal{B}_l$ and then accumulated in $\mathcal{B}$.

So far, we have seen that the algorithm distinguishes between device, array and pair building blocks. Device building blocks do not have children as they are on the lowest hierarchy level. All building blocks on the hierarchy levels 1 and above, have only children of lower levels than their own. The arrays on level 1 have children that are devices from level 0. The pairs on level 2 and above have children from any level below, i.e. devices, arrays or pairs.

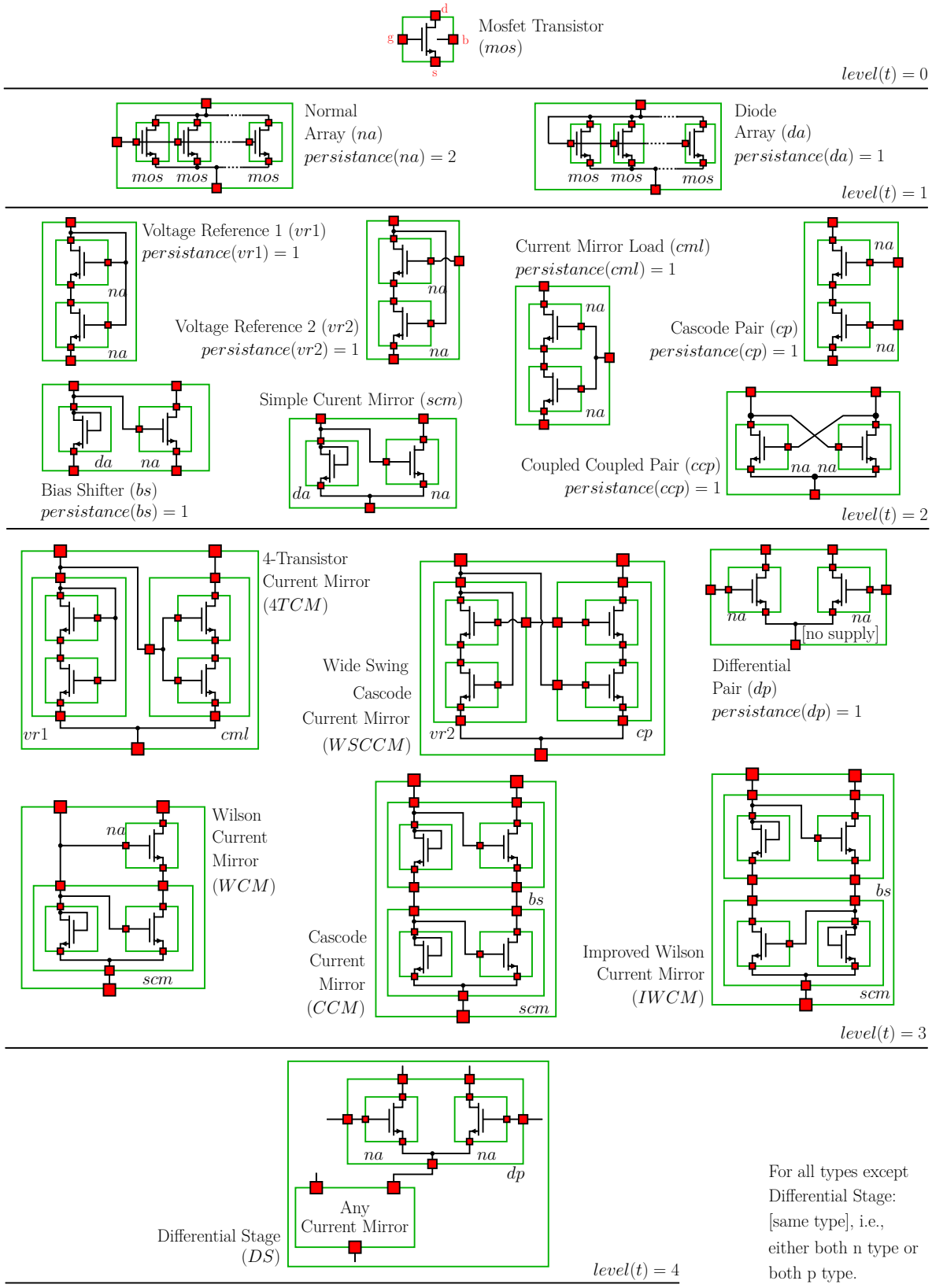



Figure 2.1.: Library $L_{analog}$ [Massier et al., 2008]

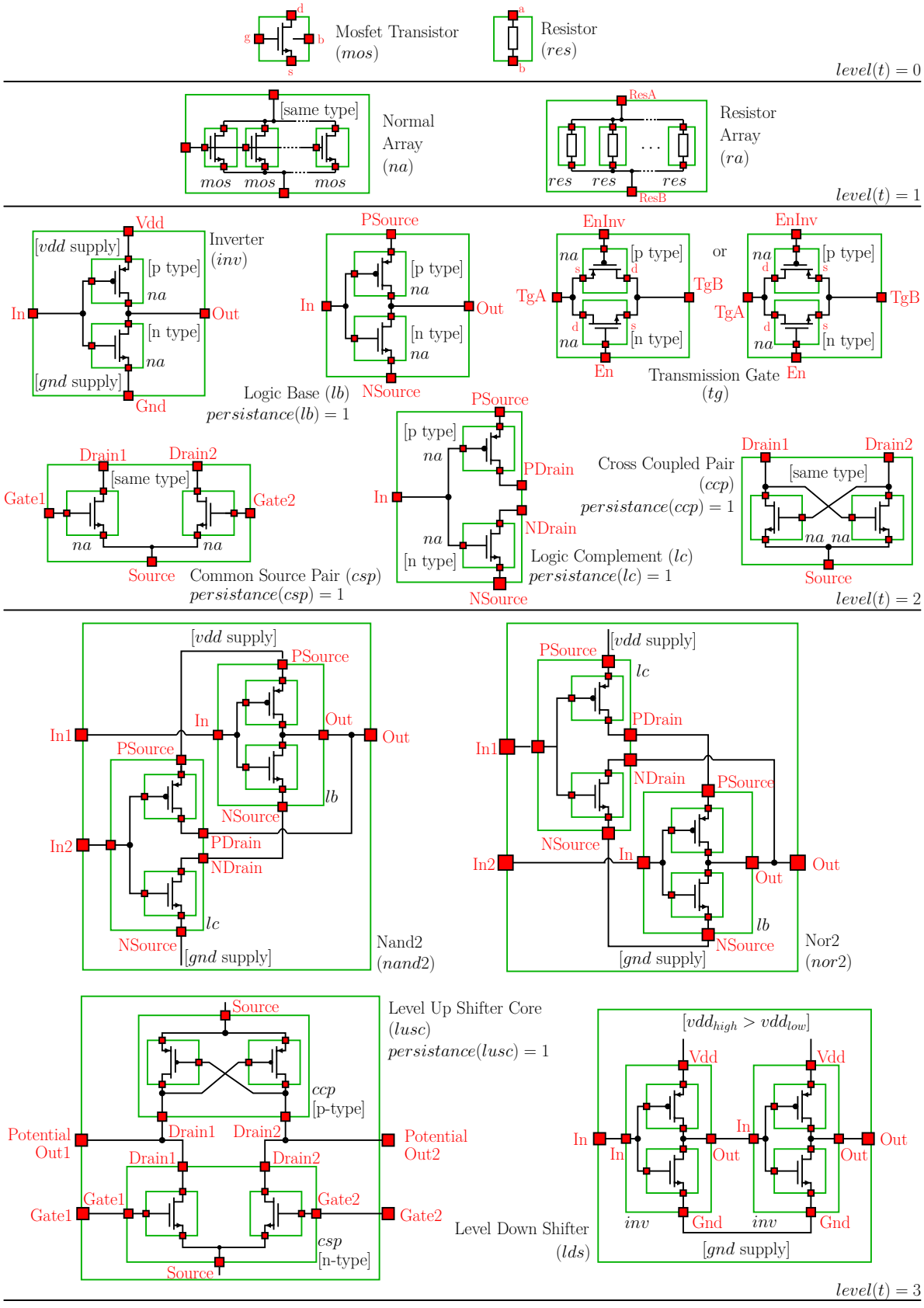
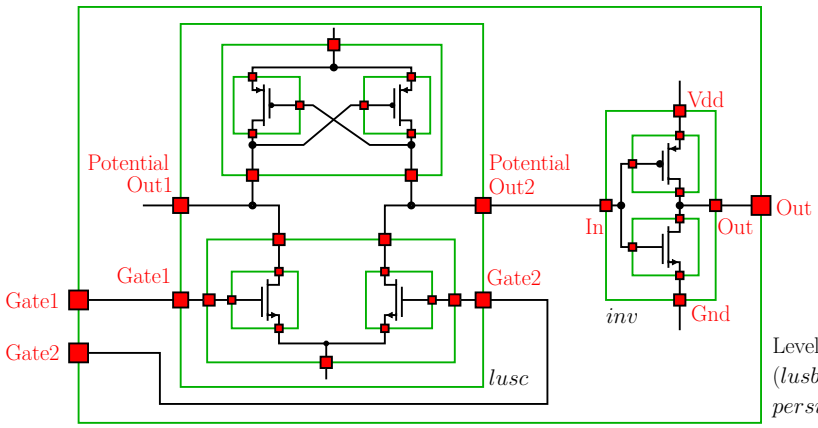
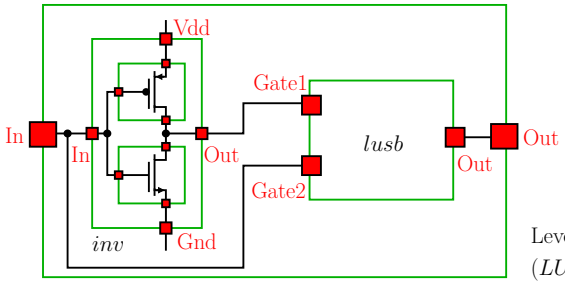



Figure 2.2.: Library $L_{pwd}$ (continued on page 24)

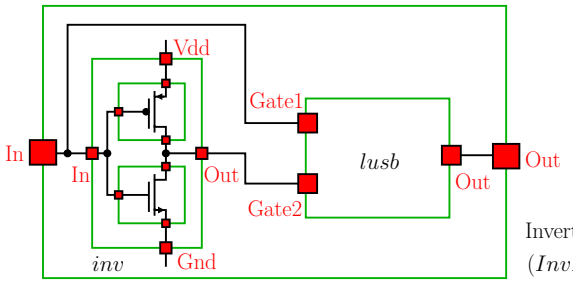



Level Up Shifter Base
(*lusb*)
$persistance(lusb) = 1$

$level(t) = 4$




Level Up Shifter
(*LUS*)




Inverting Level Up Shifter
(*InvLUS*)

$level(t) = 5$

Figure 2.3.: Library $L_{pwd}$ (continued from page 23)

In Figs. 2.1, 2.2 and 2.3, recognition rules to be checked by the functions FindArrays($t_{array}$, $\mathcal{B}$) and FindPairs($t_{pair}$, $\mathcal{B}$) are given graphically. The borders of building blocks are drawn as green boxes. The pins of building blocks are drawn as red squares and are labeled in red.

The rules for the types of the child building blocks can be derived from the annotations next to the green boxes. Furthermore, connection rules can be derived from the wires that connect the pins. In addition to the type and connection rules, further rules for techtype ($n$-type or $p$-type) and for supplies can be specified. These rules are annotated in square brackets. Child building blocks can be required to be of $n$-type, i.e., n-MOS or npn transistors, or of $p$-type, i.e., p-MOS or pnp transistors. The rules [$n$-type] and [$p$-type] require a child to be of the corresponding type. The rule [same type] requires all children to be of either $n$- or $p$ type. The techtype is propagated up the building block hierarchy as follows: A building block is techtype $n$ or $p$, if all children are of type $n$ or $p$, respectively. If the children are of mixed techtype, the building block's techtype is $undefined$. An $undefined$ techtype can not fulfill any [$n$-type], [$p$-type] or [same type] rule.

The structure recognition takes supply net information as input. Additional supply net rules are specified in the library using this information. This is an enhancement over the original algorithms published in [Massier et al., 2008; Massier, 2010] and [Eick, 2013]. The supply net rules are used in $L_{pwd}$ to detect basic logic gates and level shifters. Furthermore, experimental results show that the [no supply] rule for the source of differential pairs in $L_{analog}$ drastically reduces the number of candidate pairs for differential pairs. The addition of this rule leads to a drastic speed-up for larger circuits.

The supply net information that is taken as input has the following format: Each circuit net that acts as a ground net is tagged with $gnd$. Each circuit net that corresponds to a supply rail is tagged with $vdd_n$, where $n = 1, 2, \ldots, n_{supplies}$ and $n_{supplies}$ is the number of different supply rails in the circuit. The supply rails as well as the voltage tags can be sorted according to their magnitude, e.g. a 5 $V$ rail should get a tag with a higher index than a 3 $V$ rail. Typically in this case, the 5 $V$ rail would be assigned the tag $vdd_2$ and the 3 $V$ rail the tag $vdd_1$. In this way, the qualitative ordering of the supply voltages is made available as additional information to structure recognition while the quantitative nominal voltage values are abstracted. The following rules using the supply net information are used in $L_{analog}$ and $L_{pwd}$:

- The rule [$vdd$ supply] is fulfilled, if a net has a $vdd_n$ tag.
- The rule [$gnd$ supply] is fulfilled, if a net has a $gnd$ tag.
- The rule [$vdd_{high} > vdd_{low}$] with $high > low$ is fulfilled if the left net has a tag with higher index than the right net.

To sum up, a library $L$ contains type rules, connection rules, tech-type rules and supply rules. The rules are illustrated by Figs 2.1, 2.2 and 2.3. More detailed definitions of the rules are published in [Eick, 2013].

*2. Structure Recognition*

In the following, definitions are made that are used to describe the structure recognition algorithm. A library is a set of types with corresponding recognition rules:

$$L = \{t_1, t_2, \ldots, t_n\} \tag{2.1}$$

A library can be partitioned into subsets $L_l$ for each hierarchy level $l$ as follows:

$$L = \bigcup_{l=0,1,\ldots,\text{maxLevel}(L)} L_l \tag{2.2}$$

The function maxLevel($L$) returns the highest level in the library. For the two libraries discussed in this section, the highest levels are given as follows:

$$\text{maxLevel}(L_{analog}) = 4 \quad \text{and} \quad \text{maxLevel}(L_{pwd}) = 5 \tag{2.3}$$

According to Fig. 2.1, the subsets $L_{analog,l}$ for the analog library $L_{analog}$ are given as follows:

$$\begin{aligned}
L_{analog,0} &= \{mos\} \\
L_{analog,1} &= \{na, da\} \\
L_{analog,2} &= \{vr1, vr2, cml, cp, bs, scm, ccp\} \\
L_{analog,3} &= \{dp, 4TCM, WSCCM, WCM, CCM, IWCM\} \\
L_{analog,4} &= \{DS\}
\end{aligned}$$

Similarly and according to Figs. 2.2 and 2.3, the subsets $L_{pwd,l}$ for the power-down signal library $L_{pwd}$ can be written down as given below:

$$\begin{aligned}
L_{pwd,0} &= \{mos, res\} \\
L_{pwd,1} &= \{na, ra\} \\
L_{pwd,2} &= \{inv, tg, csp, lb, lc, ccp\} \\
L_{pwd,3} &= \{nor2, nand2, lusc, lds\} \\
L_{pwd,4} &= \{lusb\} \\
L_{pwd,5} &= \{LUS, InvLUS\}
\end{aligned}$$

A level function can be defined which assigns a level to each type using the partitioning given above. It is defined as follows:

$$level : L \to \{0, 1, \ldots, maxLevel(L)\} : t \mapsto level(t) \tag{2.4}$$

The level function is used in the following to describe the function SelfArbitration (Algorithm 2).

**Algorithm 2** SelfArbitration

**Require:** $\mathcal{B}$, $l$

**repeat**
  $removed := 0$
  **for all** $b$ in $\mathcal{B}$ **do**
    **if not** hasParent($b$) **and** $l >$ level($type(b)$) + persistence($type(b)$) **then**
      $\mathcal{B} := \mathcal{B} \setminus \{b\}$ `// changes hasParent() of other building blocks`
      $removed := 1$
    **end if**
  **end for**
**until** $removed = 0$
**return** $\mathcal{B}$

**Algorithm 3** CrossCoupledPairArbitration

**Require:** $\mathcal{B}$

**for all** $b$ in $\mathcal{B}$ **do**
  **if** "a child of $b$ is also child of a cross coupled pair ($cc$)" **then**
    $\mathcal{B} := \mathcal{B} \setminus \{b\}$
  **end if**
**end for**
**return** $\mathcal{B}$

Each recognized building block $b \in \mathcal{B}$ is of one of the types given above. Therefore, a function can be defined that maps the building block to its type. It is defined as follows:

$$type : \mathcal{B} \rightarrow L : b \mapsto type(b) \tag{2.5}$$

The function is used in this thesis to refer to the type of a building block.

Ambiguities between building blocks can occur during the recognition process. Ambiguity means that a building blocks can be part of different building bocks without any rule violation. Those ambiguities need to be resolved, i.e., the correct type among all types with matching rules needs to be chosen and the wrong building blocks need to be removed from $\mathcal{B}$. The sizing rules method uses a predefined dominance relation in order to resolve ambiguities in the circuit [Massier et al., 2008; Massier, 2010]. Furthermore, rules for so called uncertain structures are used. These are structures that are frequently subject to ambiguities. Typically, uncertain structures are building blocks where the children have a low number of connected pins, e.g., the differential pair ($dp$) and the cascode pair ($cp$) of the library $L_{analog}$. For $dp$ and $cp$, the children are only connected via one pin pair. In [Massier, 2010], uncertain building blocks are marked for manual review by the circuit designer for a final decision. In [Eick, 2013], uncertain building blocks are removed after the main loop if they are not part of another building

block. In order to simplify the rules for ambiguity arbitration and to decrease the likelihood of false recognitions, ambiguity arbitration has been enhanced compared to the sizing rules method. The new ambiguity arbitration scheme is described by the function SelfArbitration (Algorithm 2). Additionally, the function CrossCoupledPairArbitration (Algorithm 3) implements a minimalistic dominance relation: The cross coupled pair (*ccp*) dominates all other building blocks. Both functions together define the arbitration scheme.

Self arbitration generalizes the principle of the removal of uncertain building blocks by introducing a persistence value for uncertain building blocks. The persistence values are annotated in Figs 2.1, 2.2 and 2.3. The function

$$persistence : L \rightarrow \{1, 2, \ldots\ maxLevel(L)\} \qquad t \mapsto persistence(t) \tag{2.6}$$

assigns a persistence value to each type of the library. The default value is the highest level number. Only for types where the persistence value is lower, it is annotated in Figs 2.1, 2.2 and 2.3.

The persistence value controls the lifetime of a building block in $\mathcal{B}$ while iterating up the hierarchy levels $l$. The function SelfArbitration (Algorithm 2) is called (line 20) in each iteration of the main loop in Algorithm 1. It filters out all building blocks $b \in \mathcal{B}$ which were recognized more than $persistence(type(b))$ iterations ago and do not have a parent yet, i.e., have not become part of another building block yet. In this case, it is very likely that the building block is a false recognition. The persistence values in the libraries $L_{analog}$ and $L_{pwd}$ have been carefully chosen in order to resolve ambiguities. Experimental results show that a sophisticated predefined dominance relation as used in [Massier et al., 2008; Massier, 2010] is not required when the approach described in this section is used.

A hierarchy of building blocks is built up during the structure recognition process as described by Algorithm 1. The hierarchy can be described by child/parent relationships between the recognized building blocks. A building block $b_{child}$ that is part of another parent building block $b_{parent}$ is said to be a child of $b_{parent}$. Based on the child/parent relationships of the building blocks, the following logic predicate can be defined:

$$\text{hasParent(b)} = 1 \quad \Longleftrightarrow \quad \text{“}b \text{ is part of another building block } b_{parent}\text{”} \tag{2.7}$$

The function is used in function SelfArbitration (Algorithm 2).

The algorithm of this section is used for detecting asymmetric voltages in power-down mode (section 3.3) and for power-down synthesis (chapter 4). For the detection of asymmetric voltages in power-down mode, the library $L_{pwd}$ is used to split the power-down signal path away from the rest of the circuit such that it can be analyzed for analog matched pairs with $L_{analog}$. For power-down synthesis, current mirrors are detected using $L_{analog}$.

## 2.3. New Power-Down Signal Path Library

As the library $L_{pwd}$ (see Figs 2.2 and 2.3) was newly developed, a more detailed explanation of the library is given in this section. The goal of $L_{pwd}$ is to provide a library to detect building blocks forming the power-down signal path as required by the method in section 3.3. A set of basic building blocks that covers all elements on the power-down signal path of the industrial circuits examined in section 3.3.7 has been identified. It can be listed as follows:

- Digital inverter circuits
- Nor/Nand gates with two inputs
- Transmission gates
- Signal level shifters for shifting the digital signal levels up or down (example see Fig. 3.11)
- ESD-protection as shown in Fig. 3.11

In general, the structure recognition method has to be replaced or completed by available other methods (e.g. [Lescot et al., 2012; Meyer zu Bexten et al., 2015; Pelz and Roettcher, 1994; Rubanov, 2006; Ohlrich et al., 1993; Ebeling, 1988; Eick, 2013]) if this set of structures is not sufficient for the given application.

The library $L_{pwd}$ recognizes the building blocks mentioned above. The recognized building blocks are input for a power-down signal tracing method. The method will be described later in section 3.3.3. Based on the detected building blocks, a signal tracing graph is built. Fig. 3.13 shows the types from $L_{pwd}$ that are input of the signal-tracing method method. In the following, those types are called top level power-down signal blocks. They can be be listed as follows:

1. Resistor Array ($ra$)
2. Transmission Gate ($tg$)
3. Inverter ($inv$)
4. Nand2 ($nand2$) and Nor2 ($nor2$)
5. Level Down Shifter ($lds$)
6. Level Up Shifter ($LUS$) and Inverting Level Up Shifter ($InvLUS$)

The top level power-down signal blocks can also be found in $L_{pwd}$ (see Figs 2.2 and 2.3). All other building blocks in $L_{pwd}$ are required to recognize the top level power-down signal blocks. The relationships between the types in $L_{pwd}$ are described in the following. The description is going through the list above from top to down.

1. The resistor array ($ra$) is formed by resistors ($res$).
2. The transmission gate ($tg$) is a pair two normal arrays ($na$), one of tech-type $p$ and the other of techtype $n$. Two variants exist: “source and source connected” and “source and drain connected”.

3. The inverter ($inv$) is a pair of two normal arrays ($na$), one of tech-type $p$ and the other of techtype $n$. It is equivalent to the the logic base ($lb$) except the [gnd supply] and [vdd supply] rules.
4. The Nand2 ($nand2$) and Nor2 ($nor2$) building blocks are pairs of a logic base ($lb$) and a logic complement ($lc$) building block. They differ in serial and parallel connections from the output to the supply/ground of the normal arrays ($na$) that form the logic base ($lb$) and the logic complement ($lc$).
5. The level down shifter $lds$ is a pair of two inverters ($inv$). The left one has to be connected to a higher supply rail than the right one as shown by the [$vdd_{high} > vdd_{low}$] rule.
6. The level up shifter ($LUS$) and the inverting level up shifter ($InvLUS$) are pairs of an inverter ($inv$) and the level up shifter base ($lub$) block. Level up shifter ($LUS$) and inverting level up shifter ($InvLUS$) differ in the connection pattern: The connections of the pins $Gate1$ and $Gate2$ are swapped. The level up shifter base ($lub$) block is a pair of level up shifter core ($lusc$) and an inverter ($inv$). The level up shifter core ($lusc$) is a pair of a cross coupled pair ($ccp$) and a common source pair ($csp$).

The persistence values of non top level power-down signal blocks in $L_{pwd}$ have been chosen such that the lifetime of the building blocks during the recognition process is limited to the next higher hierarchy level where the respective parent(s) of the block are found.

The library $L_{pwd}$ is used later in section 3.3. There, further examples for the recognition process and experimental results can be seen. The next section presents experimental results for industrial size circuits and the library $L_{analog}$.

## 2.4. Experimental Results for Industrial-Size Analog Blocks

In previous publications, e.g. [Massier et al., 2008; Eick, 2013; Massier, 2010], experimental results for structure recognition following the principle of the sizing rules method [Massier et al., 2008] can be found. The presented examples are non-hierarchical circuits of sizes below 100 devices or nets. In this work, the first experimental results for hierarchical industrial size analog designs are presented. Table 2.1 summarizes the results. The runtime of structure recognition was measured for the five circuits given in the table. The circuits are briefly described in the following. The acronym LVDS denotes a “Low Voltage Differential Signaling” circuit. The LVDS transmitter circuit (LVDS Tx) drives a LVDS signal to a pair of pins. The LVDS receiver circuit (LVDS Rx) senses a LVDS signal on a pair of pins. Furthermore, combined LVDS / GPIO blocks were analyzed. They combine the aforementioned LVDS Tx and LVDS Rx circuits with general purpose input/output (GPIO) blocks. The combined circuits alternatively drive/sense a LVDS

| Circuit | $\|D\|$ | $\|N\|$ | $\|\mathcal{B}\|$ | runtime |
|---|---|---|---|---|
| LVDS Tx [(1)] | 901 | 354 | 102 | 13 $s$ |
| LVDS Rx [(1)] | 620 | 265 | 87 | 5.2 $s$ |
| LVDS Tx / GPIO [(1)] | 3032 | 1067 | 171 | 18 $min$ |
| LVDS Rx / GPIO [(1)] | 3385 | 1192 | 186 | 22 $min$ |
| ANALOG TOP [(2)] | 20929 | 5929 | 1061 | 35 $min$ |

[(1)] Intel(R) Xeon(R) CPU X5670 @ 2.93GHz; missing [no supply] rule for *dp*

[(2)] Intel(R) Xeon(R) CPU X5650 @ 2.67GHz; added [no supply] rule for *dp*

Table 2.1.: Experimental results for industrial analog blocks using $L_{analog}$

signal or digital signal on the same pair of pins. The two combined circuit blocks are denoted as LVDS Tx / GPIO and LVDS Rx / GPIO.

All the four LVDS circuit blocks contain amplifiers, other analog circuitry and some amount of digital circuitry. The digital circuitry translates signals from the chip's core voltage domain to the analog voltage domain and controls different signaling modes, sleep modes with reduced power consumption, a power-down mode which turns off the block completely, start-up behavior and the enabling/disabling of LVDS/GPIO for the combined blocks.

Table 2.1 shows the circuit size, i.e., the number of nets $|N|$ and the number of devices $|D|$ in the circuit. Furthermore, the number of detected building blocks $|B|$ and the measured runtime is given in the rightmost column. The runtimes are in the order of magnitude of a few seconds for the smaller LVDS Tx and LVDS Rx circuits and around 20 minutes for the larger combined blocks. The first four circuits were analyzed on an Intel(R) Xeon(R) CPU X5670 2.93GHz CPU. For those experiments, the [no supply] rule for the differential pair (*dp*, see Fig. 2.1) was not part of the library. It was added later while analyzing the largest circuit in our row of experiments. The circuit is denoted as ANALOG TOP and comprises more than 20.000 devices. It is an analog top-cell, i.e., the complete analog part of a chip. The experiment for this circuit was executed on a different machine, i.e., on a machine with a Intel(R) Xeon(R) CPU X5650 2.67GHz processor. Due to the high number of candidate differential pairs along the supply rails, the algorithm did not terminate in a practicable amount of time. However, the runtime could be reduced to 35 *min* by adding the [no supply] rule for the differential pair *dp*. It was not possible to repeat the four LVDS experiments with [no supply] rule for the differential pair in place due to access restrictions to the industrial circuits. Further experiments need to be conducted in future work in order to generate a more consistent row of runtimes.

The results presented in this section demonstrate that an algorithm based on the sizing rules method is capable to analyze complete analog designs in a practicable amount of time. To the best of my knowledge, there are no previously published experimen-

tal results for the sizing rules method for circuits consisting of up to 20.000 devices. The enhanced structure recognition algorithm is used to detect asymmetric voltages at matched structures in power-down mode (section 3.3) and for power-down circuit synthesis (chapter 4).

# 3. Power-Down Verification

In this chapter, methods for automatic verification of the correctness of the power-down circuitry are presented. First, the design flow supported by the methods of this chapter is briefly discussed in section 3.1. Then, a voltage propagation method is presented. It can estimate power-down node voltages and detect floating nodes and leakage currents based on a structural graph model. The graph model is the basis for the power-down synthesis method presented in chapter 4. In section 3.3, the voltage propagation method and the structure recognition method from the previous chapter are combined in order to detect asymmetric aging critical voltages in power-down mode. Furthermore, voltage matching rules for automatic detection of asymmetric voltages are developed. The voltage-matching rules will also be used for power-down synthesis.

## 3.1. Design Flow with Automatic Verification

Fig. 3.1 shows a design flow with manual review on the left side and a design flow with automatic verification on the right side. In both flows, the power-down circuitry is added manually to the analog design after the design of the analog core part is finished. The analog core part is the part of the circuit that implements the power-up functionality. In the manual flow shown on the left, the correctness of the added power-down circuitry is verified by a manual design review. In contrast, tool supported automatic verification is used in the flow shown on the right side. The methods presented in this chapter support the design flow shown on the right.

For modern systems on chip, tool support for power-down mode verification is becoming more and more important. This is due to an increasing complexity of power management

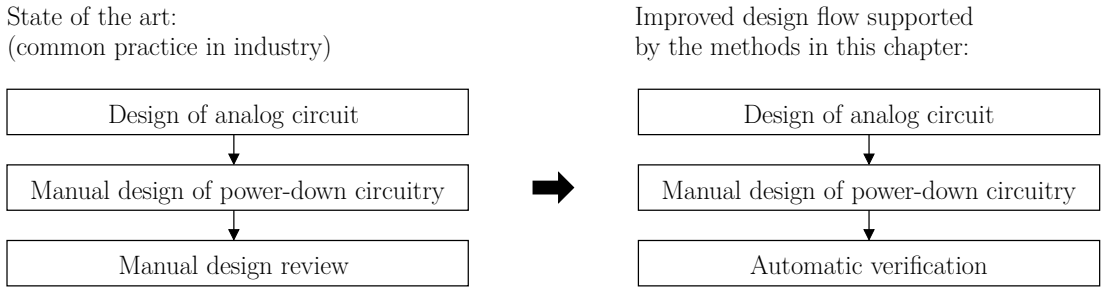



Figure 3.1.: Common practice industrial design flow and improved design flow supported by the methods in this chapter

and interaction between the digital and analog chip parts. For example, the following reasons for increasing complexity of power-down mode verification can be identified: First, there can be many different power-down or sleep modes. Second, other (non power-down related) digital signals may interact with the power-down circuit in power-down mode. Third, more and more measures have to be taken to ensure that aging in power-down mode does not cause reliability problems as newer technologies with smaller feature sizes tend to be more sensitive to aging effects. The systematic approaches described in this chapter contribute to cope with the complexity. They can be used to systematically reduce design errors.

## 3.2. Detection of Floating Nodes and Short Circuits

In this section, a method for automatic detection of floating nodes and short circuits in power-down mode is described. First, the problem is described more detailed in section 3.2.1. Then, an overview over the method is presented in section 3.2.2. The method combines a voltage propagation approach (see section 3.2.4) that propagates voltages on a graph model of the circuit (see section 3.2.3) and a short-circuit-path search that searches leakage currents on this graph model (see section 3.2.5). The complete algorithm is formulated in section 3.2.6. The section concludes with the presentation of the experimental results (see section 3.2.7).

### 3.2.1. Problem Description

As described in the introduction of this thesis, analog circuit blocks are usually not powered down by driving down the supply voltages as this would introduce head and/or foot switches which cost voltage headroom. Instead, switches are added to pull the internal nodes of the circuit to one of the supply rails or to ground such that all currents are turned off. If this principle is implemented incompletely, internal nodes may stay floating. A floating node in power-down mode is defined as follows [Zwerger and Graeb, 2012a, 2013]:

> *Floating node:* A node that is only connected to the supplies via high impedance paths in power-down mode.

The actual voltages that will appear on floating nodes are hard to predict. The voltages of these nodes depend on sub threshold characteristics of the devices, on leakage-current paths and on external signals. This poses a challenge to the verification of the power-down mode, as accurate simulation is difficult under these conditions. The sub threshold characteristics of the devices are often not completely captured in models used for numerical circuit simulation. Furthermore, leakage paths depend on the circuit layout and process variations. Therefore, they are hard to predict and control. Last but not least, the signals on the input and output pins of a circuit are often not specified by the

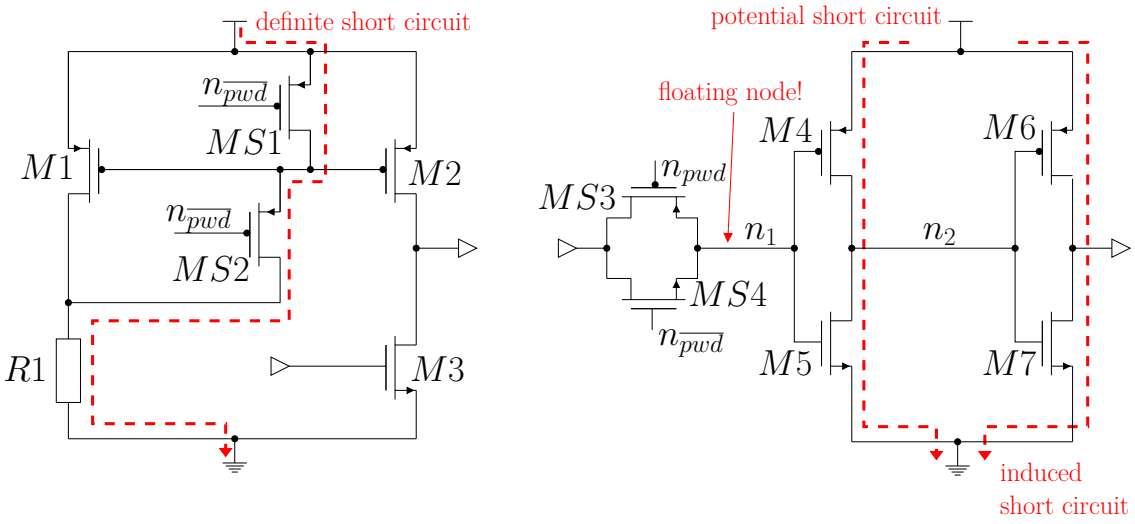



Figure 3.2.: Examples of different types of short-circuit paths

designer of the circuit. For a complete coverage of the verification, each possible combination of external signals needs to be simulated in the worst case. Incomplete models limit the validity of results gained from DC or transient simulation, whereas unknown external signals lead to excessive simulation times. All in all, numerical simulation of the power-down mode is problematic. However, the accuracy of numerical simulation is not needed to describe the static behavior in power-down mode, as the transistor states are in a defined conducting or non-conducting state and not in between. It should be considered as a good design style to avoid floating nodes in power-down mode. A floating node can easily be eliminated by adding a transistor with minimum size that pulls the node to a defined level, i.e., to one of the positive supplies or to ground.

The power-down circuitry should suppress all currents, i.e., it should drive the power consumption to zero while the supply voltage is still connected. Therefore, all paths through the circuit allowing current flow in power-down mode are design errors. In the following, those paths are called short-circuit paths. In this section, a method to detect those short-circuit paths as well as floating nodes is described. The method avoids the problems of numerical simulation by adopting a structural approach.

Fig. 3.2 shows an example for a floating node and examples for different types of short-circuit paths. The floating node in Fig. 3.2 appears on the gate of the inverter that is formed by $M4$ and $M5$. Net $n_1$ is disconnected from the other parts of the circuit by a pass gate $MS3$, $MS4$. Therefore, net $n_1$ is floating. Furthermore, examples of different types of short-circuit paths are shown in Fig. 3.2. Three types of short-circuit paths are distinguished: Definite short-circuit paths, potential short-circuit paths and induced short-circuit paths [Zwerger and Graeb, 2013, 2012a]. They are described in the following.

> *Definite short-circuit path:* A path from the positive supply to the negative supply along conducting transistor channels or other conducting circuit elements which will definitely conduct significant currents in power-down mode.

For the amplifier circuit shown on the left in Fig. 3.2, a definite short-circuit path occurs because the power-down switch $MS2$ was wrongly connected to $n_{\overline{pwd}}$ instead of $n_{pwd}$. Therefore, both, $MS1$ and $MS2$ are conducting and allowing current flow over the marked path.

> *Potential short-circuit path:* A path from the positive supply to the negative supply along conducting transistor channels or other conducting circuit elements and across transistors with a floating gate or base.

In Fig. 3.2, it cannot be guaranteed that either $M4$ or $M5$ is properly turned off due to the floating input of the inverter. This is a subtle error that is hard to detect by numerical simulation.

> *Induced short-circuit path:* A short-circuit path that is caused by a potential or definite short-circuit path.

The voltages of the nodes on a definite short-circuit path have a defined voltage, i.e., a voltage that can reliably be found out by numerical circuit simulation. The voltages of the nodes on a potential short-circuit path will have a voltage between *vdd* and *gnd* that depends on the charge that is on the floating gate. As the voltage of the floating gate is problematic to determine by numerical simulation, the node voltages on the potential short-circuit path are also problematic to predict. This can induce further short-circuit paths if further gates are connected to the nodes on a short-circuit path.

It can be observed that once there is one definite or potential short-circuit path, a large number of other paths that are not safely shut off are often induced. In order to limit the number of reported errors to the root causes, the induced paths should not be reported in the first place. They will be fixed by fixing all definite and potential short-circuit paths. An example can be seen in Fig. 3.2. An induced short-circuit path is running over the inverter consisting of $M6$ and $M7$. The current of this path is only off if the voltage of $n_2$ is either near *vdd* or near *gnd*. However, the voltage of $n_2$ depends on the voltage of the floating node $n_1$. Consequently, the current in the path over $M6$ and $M7$ is potentially not turned off. Pulling the floating node $n_1$ to a defined voltage (*vdd* or *gnd*) will remove the potential short-circuit path and as a further consequence also the induced short-circuit. Hence, it is sufficient to report only the floating node and the potential short circuit in an error report. The induced short-circuit path can be omitted. Nevertheless, it can be used for diagnosis.

### 3.2.2. Overview

The algorithm proposed in this section verifies that there are no short-circuit paths and no floating nodes in power-down mode. It was first published in [Zwerger and Graeb, 2012a]. An extended version was published in [Zwerger and Graeb, 2013]. The algorithm combines a voltage propagation algorithm and an algorithm for computing short-circuit paths. The voltage propagation algorithm estimates voltages of nodes that are not part of a short-circuit path. It needs all short-circuit paths as input. The short-circuit algorithm calculates these short-circuit paths. However, it needs the voltages calculated by voltage propagation as input. This circular dependency is resolved by iteratively executing voltage propagation and short-circuit algorithm until no more short-circuit paths are found. The voltage propagation algorithm is described in section 3.2.4. The short-circuit algorithm is described in section 3.2.5. In section 3.2.6, these two algorithms are combined to form the verification algorithm.

The verification algorithm reports errors in the following cases [Zwerger and Graeb, 2012a, 2013]:

| | |
|---|---|
| Definite SC: | There is a definite short-circuit path. |
| Potential SC: | There is a potential short-circuit path. |
| Floating Node: | There is a floating node. |

The items above are referenced by Tables 3.2 and 3.6 in section 3.2.7. Highlighting occurrences of the above problems in the schematic of the circuit helps the designer to quickly identify errors and potential spots of trouble in large circuits.

### 3.2.3. Definitions for the Graph Model

The voltage propagation algorithm described in this section estimates the steady-state node voltages of a circuit in power-down mode. The circuit is transformed to a graph $G = (N, E, \phi^+, \phi^-, \tau)$. In the following, the definitions of the graph model are given [Zwerger and Graeb, 2012a, 2013]. The sets of nodes and edges are defined as follows:

$$N = \{n_1, n_2, \ldots, n_{|N|}\} \quad \text{(set of nodes)} \tag{3.1}$$

$$E = \{e_1, e_2, \ldots, e_{|E|}\} \quad \text{(directed edges)} \tag{3.2}$$

The incidence mappings are defined as follows[1]:

$$\phi^+ \colon E \to N \text{ where } \phi^+(e) = n \Leftrightarrow \text{“}e \text{ starts at } n\text{”}, \tag{3.3}$$

$$\phi^- \colon E \to N \text{ where } \phi^-(e) = n \Leftrightarrow \text{“}e \text{ ends at } n\text{”}. \tag{3.4}$$

[1]The indices + and − refer to the voltage drop along an edge if current is flowing, i.e for current flow, $\phi^+(e)$ has to have a higher potential than $\phi^-(e)$.

Each device of the circuit is mapped to a set of edges as shown by Fig. 3.3. The mapping is discussed in detail in section 3.2.4. In this section, the definitions for the graph model are given.

The edges are labeled with an edge type. An edge can either be of diode-type (`d`), n-switch type (`n`), or p-switch type (`p`). The type mapping $\tau$ assigns one of these labels to each edge. It is defined as follows:

$$\tau\colon E \to T,\ e \mapsto \tau(e) \text{ with } T = \{\mathtt{d}, \mathtt{n}, \mathtt{p}\} \tag{3.5}$$

Diode edges are non switched edges. In contrast, `n` and `p` edges are switched edges. A set of switched edges can be defined as follows:

$$S = \{e \in E \mid \tau(e) = \mathtt{n} \vee \tau(e) = \mathtt{p}\} \tag{3.6}$$

Each net, $n \in N$, has an associated voltage. The voltage of net $n$ is denoted as $n.V$. The voltage can be $gnd$, $float$, $sc$ or a supply rail voltage $vdd_i$. The voltage $gnd$ indicates a voltage of zero volt, i.e., the voltage of the ground supply rail. The voltage $float$ indicates a floating node. Nodes on short-circuit paths are marked with voltage $sc$. The supply rail voltages $vdd_i$ are used to model supply rails with different voltages similarly to the supply net information for structure recognition described in section 2.2. Supply rails with lower voltages are assigned a voltage $vdd_i$ with a lower index $i$. For instance,the 3.3 $V$ rail can be represented by voltage $vdd_1$ and the 5 $V$ rail can be represented by $vdd_2$. In this way, the qualitative ordering of the voltages is preserved without having to refer to the actual values. The set of all possible voltages is defined as follows:

$$V = \{gnd, float, sc\} \cup \Big( \bigcup_{i=1\ldots n_{sup}} \{vdd_i\} \Big) \qquad (\text{with } n.V \in V) \tag{3.7}$$

The number $n_{sup}$ is the total number of supply rails for a given circuit.

The voltage propagation algorithm (Algorithm 4) requires the sets of nets that are initialized with a given voltage as input. Those nets are denoted as $N_{gnd}$, $N_{sc}$ and $N_{vdd_i}$. The set $N_{gnd}$ contains nodes which have a fixed voltage of $gnd$. The set $N_{sc}$ contains nodes located on short-circuit paths which have been computed by the short-circuit algorithm described in section 3.2.5. They are initialized with voltage $sc$. Furthermore, a set $N_{vdd_i}$ for each supply rail $vdd_i$ is taken as input, i.e., the algorithm requires a tuple of sets $(N_{vdd_1}, \ldots, N_{vdd_{n_{sup}}})$ as input. The nets in those sets are initialized with the corresponding supply voltage. Typically, each of those sets contains the corresponding supply net plus additional nets, e.g., nets driven by external signals. The positive supply rail nets can be summarized in a set $N_{vdd}$ as follows:

$$N_{vdd} = \bigcup_{i=1\ldots n_{sup}} N_{vdd_i} \tag{3.8}$$

| | $s(e).V = gnd$ | $s(e).V = float$ | $s(e).V = sc$ | $isVdd(s(e).V)$ |
|---|---|---|---|---|
| $\tau(e) = \texttt{n}$ | $off$ | $float$ | $sc$ | $on$ |
| $\tau(e) = \texttt{p}$ | $on$ | $float$ | $sc$ | $off$ |
| $\tau(e) = \texttt{d}$ | always $on$, $s(e)$ not defined | | | |

Table 3.1.: State mapping $\sigma(e)$ [Zwerger and Graeb, 2012a, 2013]

The sets $N_{gnd}$, $N_{sc}$ and $N_{vdd_i}$ describe the boundary conditions for voltage propagation.

Furthermore, a logic predicate to verify that a voltage is one of the given positive supply voltages is defined:

$$isVdd(v) \quad \Longleftrightarrow \quad (v = vdd_1) \vee (v = vdd_2) \vee \cdots \vee (v = vdd_{n_{sup}}) \tag{3.9}$$

It returns true, if a voltage $v$ is one of the given supply voltages.

Each edge is assigned a state based on the node voltages. The set of edge states is defined as follows:

$$ES = \{on, float, sc, off\} \tag{3.10}$$

The state $on$ denotes a conducting edge and $off$ represents a non-conducting edge. The states $float$ and $sc$ indicate that the state of a transistor is not known. Two different states are introduced to separate the reason for the unknown state: floating node or another short-circuit path.

The state of each switched edge is determined by the voltage of the corresponding gate or base node. A switching function $s(e)$ is defined as follows:

$$s\colon S \to N \text{ where } s(e) = n \Leftrightarrow \text{“}e \text{ is switched by } n\text{”} \tag{3.11}$$

A state is assigned to each edge by a state-mapping function. It is defined as follows:

$$\sigma : E \to ES, e \mapsto \sigma(e) \tag{3.12}$$

with

$$\sigma(e) = \begin{cases} on & \text{for } \begin{array}{l} \tau(e) = \texttt{d} \vee \\ \big( (\tau(e) = \texttt{n} \wedge isVdd(s(e).V)) \vee \\ \quad (\tau(e) = \texttt{p} \wedge s(e).V = gnd) \big) \end{array} \\ off & \text{for } \begin{array}{l} \big( (\tau(e) = \texttt{n} \wedge s(e).V = gnd) \vee \\ \quad (\tau(e) = \texttt{p} \wedge isVdd(s(e).V)) \big) \end{array} \\ float & \text{for } (\tau(e) = \texttt{n} \vee \tau(e) = \texttt{p}) \wedge s(e).V = float \\ sc & \text{for } (\tau(e) = \texttt{n} \vee \tau(e) = \texttt{p}) \wedge s(e).V = sc \end{cases} \tag{3.13}$$

Table 3.1 summarizes the definition of $\sigma(e)$. An edge state is assigned to the edge in dependency of the edge type $\tau(e)$ and the voltage of the controlling node. The assigned edge state for `n` and `p` edges is given in the corresponding lines of Table 3.1. The voltage of the controlling node of `n` and `p` edges is denoted as $s(e).V$. For `d` edges, there is no controlling node. Hence, the switching function $s(e)$ is not defined. Edges of type `d` always have state *on* according to the last line of Table 3.1.

### 3.2.4. Voltage Propagation

Voltage propagation estimates the node voltages of the circuit without using circuit simulation. The voltage propagation algorithm is listed in Algorithm 4. It takes the graph model $G = (N, E, \phi^+, \phi^-, \tau)$ as input. The graph model as well as the algorithm are explained in the following [Zwerger and Graeb, 2013, 2012a]. At the end of this section, the runtime complexity of the algorithm is discussed.

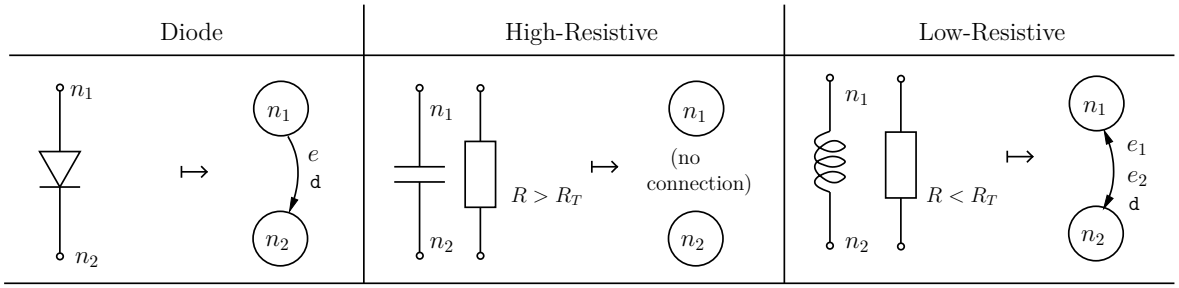
#### Graph Model

The devices of the circuit are mapped to the graph according to Fig. 3.3. The upper part of the figure shows the mapping of two-terminal devices. The lower part of the figure shows the mapping of transistors.

The two-terminal devices are categorized into low-resistive devices, high-resistive devices and diode devices. Two anti-parallel edges of type `d` are inserted into the graph for each low-resistive device. The algorithm takes a threshold resistance $R_T$ as input in order to categorize resistors into low- and high-resistive. For example, a suitable value is $R_T = 10\ M\Omega$. It is guaranteed that each path from $N_{vdd}$ to $N_{gnd}$ will have a resistance of at least $R_T$ if no short-circuit path is found by the method. No edge is inserted into the graph for high-resistive devices. For diode devices, a single directed diode edge is inserted.

Transistors are categorized into diode-connected transistors, off-connected transistors and transistors with no self connection. The three categories correspond to the column headings of the lower table in Fig. 3.3. For diode-connected and off-connected transistors, a single diode edge is added between drain and source or between emitter and collector, respectively. The direction of the edge is same for all n-type (npn, n-MOS) and all p-type (pnp, p-MOS) transistors. In the diode-connected case, the diode is inserted in the opposite direction compared to the off-connected case. This pattern reflects the fact that the qualitative behavior of MOS and bipolar devices is the same in forward and reverse direction. Drain and source can be flipped according to the polarity of the drain-source voltage. However, bipolar devices are usually optimized for operation in forward direction. The quantitative behavior in forward and reverse direction is different but the qualitative behavior remains the same. This behavior is correctly captured by

**Graph Model for Two-Terminal Devices:**




**Graph Model for Transistors:**

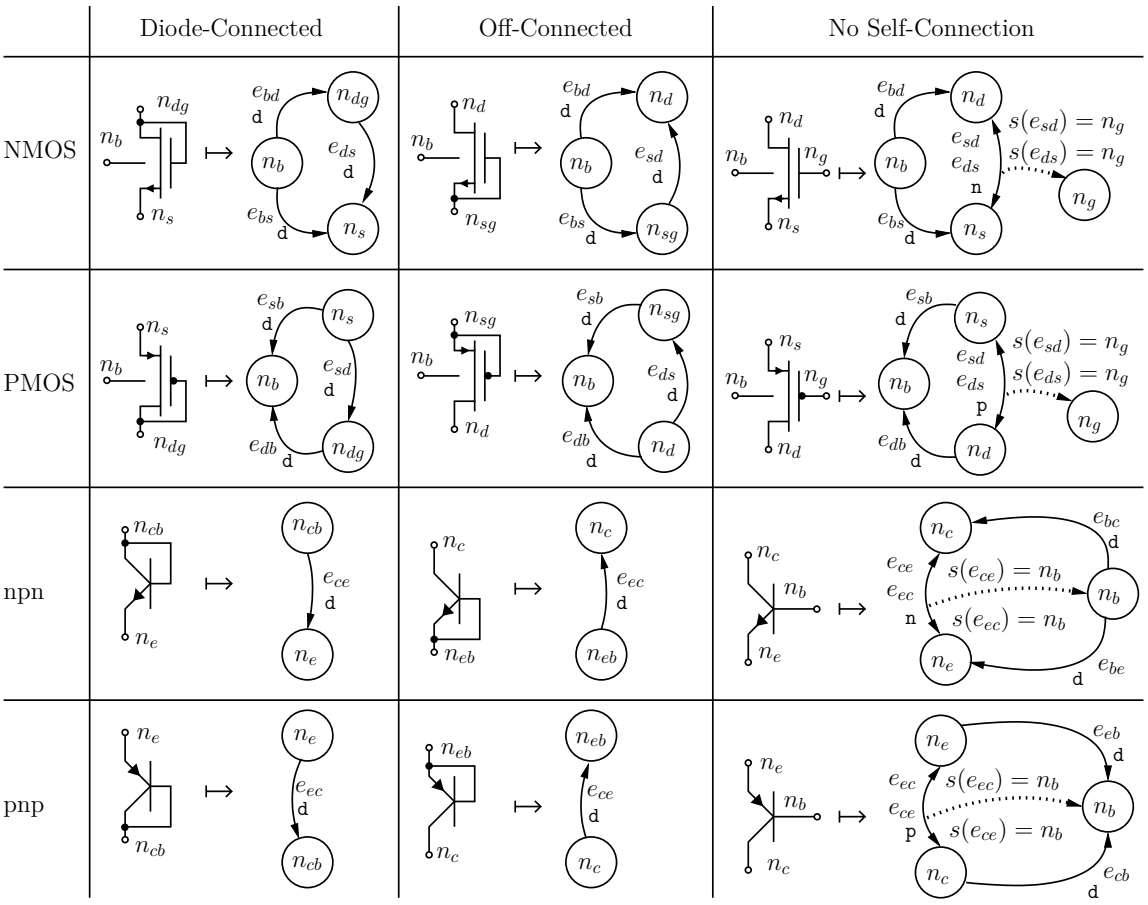



Figure 3.3.: Mapping of devices to the graph $G = (N, E, \phi^+, \phi^-, \tau)$ and switching function $s(e)$ [Zwerger and Graeb, 2012a, 2013]

**Algorithm 4** VoltagePropagation [Zwerger and Graeb, 2012a, 2013]

**Require:** $G$, $N_{gnd}$, $(N_{vdd_1}, \ldots, N_{vdd_{n_{sup}}})$, $N_{sc}$

1: `// initialize all node voltages:`
2: **for all** $n \in N$ **do** $n.V := float$
3: **for all** $n \in N_{gnd}$ **do** $n.V := gnd$
4: **for all** $i \in \{1, 2, \ldots, n_{sup}\}$ **do**
5: **for all** $n \in N_{vdd_i}$ **do** $n.V := vdd_i$
6: **end for**
7: **for all** $n \in N_{sc}$ **do** $n.V := sc$
8:
9: `// propagate voltages:`
10: $E_{prop} = \{\}$
11: **repeat**
12: $E_{cond} := \{e \in E \setminus E_{prop} \mid prop(e)\}$
13: **for all** $e \in E_{cond}$ **do**
14: **if** $\phi^+(e).V = float$ **then** $\phi^+(e).V := \phi^-(e).V$
15: **if** $\phi^-(e).V = float$ **then** $\phi^-(e).V := \phi^+(e).V$
16: **end for**
17: $E_{prop} = E_{prop} \cup E_{cond}$
18: **until** $E_{cond} = \{\}$
19:
20: **return** $G$ `// Graph with modified node voltages`

the graph model. Two anti parallel `n` or `p` edges are added between drain and source (or emitter and collector) if no self connection is present. These edges are controlled by the gate or base nodes, respectively. The resulting switching function is visualized with dotted arrows.

Diode edges between bulk and drain as well as between bulk and source are added to model the bulk diodes of MOS transistors. Diode edges between base and emitter and between base and collector are added for bipolar transistors to capture possible leakage paths due to base current. In contrast to the gate current of MOS transistors, the base current of bipolar transistors is non-zero.

### Voltage Propagation Algorithm

The graph is constructed as first step of the verification method (see Algorithm 7, line 2). The graph is input data of the voltage propagation algorithm (Algorithm 4). All node voltages are first initialized with $float$ (line 2). Then, all nodes in the sets $N_{gnd}$, $N_{vdd_i}$ and $N_{sc}$ are initialized to their corresponding voltages (lines 3 - 7). The voltages are iteratively propagated along the edges starting from the nodes that have been initialized with $gnd$ or any positive supply voltage $vdd_i$. The propagation is carried out by the

main loop in lines 11 to 18. A propagation condition $prop(e)$ is evaluated for each edge that has not been propagated yet (line 12) to check if a voltage can be propagated along that edge. An edge that is *on* propagates any positive supply voltage $vdd_i$ in direction of the edge and *gnd* counter the direction of the edge if the other side has the voltage $float$. The propagation condition can be formalized as follows:

$$prop(e) \iff (\sigma(e) = on) \wedge \Big[ \big(isVdd(\phi^+(e).V) \wedge \phi^-(e).V = float\big) \\ \vee \big(\phi^-(e).V = gnd \wedge \phi^+(e).V = float\big) \Big] \tag{3.14}$$

In each iteration, this condition is evaluated (line 12) and all edges along which propagation is possible are stored in the set $E_{cond}$. The propagation is executed in lines 14 and 15 for all edges in $E_{cond}$. The $float$ voltage on one side of the edge is replaced by the voltage on the other side. This leads to a changed edge state of the edges controlled by affected nodes. Therefore, further propagation can occur in the next iteration. All edges along which a voltage has already been propagated are collected in the set $E_{prop}$ (line 17). The main loop terminates if there are no more edges along which propagation can occur (line 18). Termination is guaranteed as propagation occurs at most once per edge due to the following reason: Neither the end nor the start of an edge has voltage $float$ if propagation occurred along that edge. However, this is necessary for propagation. Thus, each edge is only propagated once and the algorithm is guaranteed to terminate.

#### Complexity

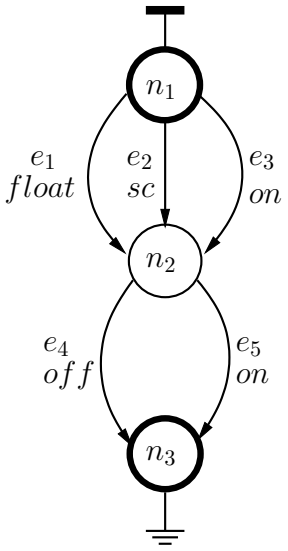
The worst-case complexity of the voltage propagation algorithm (Algorithm 4) is quadratic in the number of edges, i.e., it is $O(|E|^2)$. The following analysis leads to this result [Zwerger and Graeb, 2013, 2012a]: Every edge is propagated at most once. Therefore, the main loop (lines 11 to 18) is repeated at most $|E|$ times. This happens if in each iteration exactly one edge is propagated. Then, the propagation condition has to be checked $|E|$ times in the first iteration, $|E| - 1$ times in the second iteration and once in the last iteration (line 12). This yields

$$\sum_{i=1}^{|E|} i = \frac{1}{2}|E|^2 + \frac{1}{2}|E| \tag{3.15}$$

evaluations of the propagation condition in the worst case. Thus, the number of evaluations for a large number of edges is bounded by $|E|^2$.

### 3.2.5. Short-Circuit-Path Search

The graph model can be analyzed for short-circuit paths if the node voltages calculated from voltage propagation are given. In this section, the different types of short-circuit




$$N_{vdd} = \{n_1\},\ N_{gnd} = \{n_3\}$$

$$\begin{aligned}\mathscr{P} = \{&(), (e_1), (e_2), (e_3), (e_4), (e_5),\\ &(e_1, e_4), (e_1, e_5), (e_2, e_4), (e_2, e_5),\\ &(e_3, e_4), (e_3, e_5)\}\end{aligned}$$

$$\begin{aligned}\mathcal{P}_{vdd\to gnd} = \{&(e_1, e_4), (e_1, e_5), (e_2, e_4),\\ &(e_2, e_5), (e_3, e_4), (e_3, e_5)\}\end{aligned}$$

$$\begin{aligned}\mathcal{P}_{SC,def} &= \{(e_3, e_5)\}\\ \mathcal{P}_{SC,pot} &= \{(e_1, e_5)\}\\ \mathcal{P}_{SC,sc} &= \{(e_2, e_5)\}\end{aligned}$$

Figure 3.4.: Example of different types of short-circuit paths [Zwerger and Graeb, 2013]

paths that were introduced in section 3.2.1 are defined in terms of edges and nodes of the graph model. The short-circuit path search algorithm is described using those definitions [Zwerger and Graeb, 2013]. Furthermore, the runtime complexity of the short-circuit-path algorithm is determined at the end of this section.

### Definitions of Short-Circuit Paths

A short-circuit path is a path along conducting or potentially conducting edges. In section 3.2.1 three types of short-circuit paths were introduced: definite, potential and induced short-circuit paths. Those paths are defined mathematically in the following. Therefore, the set of all paths in the graph $G = (N, E, \phi^+, \phi^-, \tau)$ is defined. It is then constrained to the respective short-circuit paths. Fig. 3.4 shows an example graph as well as examples for the following definitions. The nodes in $N_{vdd}$ and $N_{gnd}$ are marked with a supply and a ground symbol, respectively.

The set of all paths is defined as follows:

$$\mathscr{P} = \Big\{()\Big\} \cup E \cup \bigcup_{n=2\ldots|E|} \Big\{(e_1, e_2, \ldots, e_n) \in E^n \;\Big|\; \underset{i=1\ldots n-1}{\forall} \phi^-(e_i) = \phi^+(e_{i+1})\Big\} \tag{3.16}$$

The symbol () denotes an empty path. The set of all paths can be further constrained

to the set of all paths from $N_{vdd}$ to $N_{gnd}$ as follows:

$$\mathcal{P}_{vdd\to gnd} = \Big\{(e_1, e_2, \ldots, e_n) \in \mathscr{P} \;\Big|\; \phi^+(e_1) \in N_{vdd} \wedge \phi^-(e_n) \in N_{gnd} \quad \text{(3.17)}$$
$$\text{“Path has no loops.” } \wedge \text{ “No intermediate node is in } N_{vdd} \text{ or } N_{gnd}.\text{”}\Big\}$$

A path in $\mathcal{P}_{vdd\to gnd}$ is a short-circuit path if it is not containing any edge with state $off$. The set of short-circuit paths can be partitioned into a set of definite short-circuit paths $\mathcal{P}_{SC,def}$, a set of potential short-circuit paths $\mathcal{P}_{SC,pot}$ and a set of induced short-circuit paths $\mathcal{P}_{SC,sc}$. The set of definite short-circuit paths is defined as follows:

$$\mathcal{P}_{SC,def} = \Big\{(e_1, e_2, \ldots, e_n) \in \mathcal{P}_{vdd\to gnd} \;\Big|\; \underset{i=1\ldots n}{\forall} \sigma(e_i) = on\Big\} \quad \text{(3.18)}$$

A definite short-circuit path has only edges with state $on$.

The set of potential short-circuit paths is defined as follows:

$$\mathcal{P}_{SC,pot} = \Big\{(e_1, e_2, \ldots, e_n) \in \mathcal{P}_{vdd\to gnd} \;\Big|$$
$$\Big(\underset{i=1\ldots n}{\forall} \sigma(e_i) = on \vee \sigma(e_i) = float\Big) \wedge \Big(\underset{i=1\ldots n}{\exists} \sigma(e_i) = float\Big)\Big\} \quad \text{(3.19)}$$

A potential short-circuit path has only edges with state $on$ or $float$. At least one edge must have the state $float$. Otherwise it would be a definite short-circuit path.

The set of induced short-circuit paths is defined follows:

$$\mathcal{P}_{SC,sc} = \Big\{(e_1, e_2, \ldots, e_n) \in \mathcal{P}_{vdd\to gnd} \;\Big|$$
$$\Big(\underset{i=1\ldots n}{\forall} \sigma(e_i) = on \vee \sigma(e_i) = float \vee \sigma(e_i) = sc\Big) \wedge \Big(\underset{i=1\ldots n}{\exists} \sigma(e_i) = sc\Big)\Big\} \quad \text{(3.20)}$$

An induced short-circuit path has only edges with state $on$, $float$ or $sc$. At least one edge has to have state $sc$. Otherwise it would be a definite or a potential short-circuit path.

The set of all short-circuit paths is the union of the three sets above:

$$\mathcal{P}_{SC} = \mathcal{P}_{SC,def} \cup \mathcal{P}_{SC,pot} \cup \mathcal{P}_{SC,sc} \quad \text{(3.21)}$$

The set $\mathcal{P}_{SC}$ is computed by the short-circuit path algorithm (Algorithm 5). Equations (3.18), (3.19) and (3.20) are evaluated in line 15 of Algorithm 7 to classify the computed short-circuit paths.

**Algorithm 5** ShortCircuitPathSearch [Zwerger and Graeb, 2013]

**Require:** ($G$, $N_{gnd}$, $N_{vdd}$)
1: $\mathcal{P}_{SC} := \{\}$
2: **for all** $n_{start} \in N_{vdd}$ **do**
3: $N_{dfs} := \{\}$
4: $P_{dfs} := ()$
5: $\mathcal{P}_{result} :=$ DepthFirstSearch($G$, $N_{gnd}$, $N_{vdd}$, $n_{start}$, $N_{dfs}$, $P_{dfs}$) // *Algorithm 6*
6: $\mathcal{P}_{SC} := \mathcal{P}_{SC} \cup \mathcal{P}_{result}$
7: **end for**
8: **return** $\mathcal{P}_{SC}$

A push operation for paths is used by the depth-first-search algorithm (Algorithm 6). The push operation appends an edge $e' \in E$ to an existing path $P = (e_1, e_2, \ldots, e_n) \in \mathscr{P}$. It is defined as follows:

$$\text{push}(P, e') = (e_1, e_2, \ldots, e_n, e') \tag{3.22}$$

The new path has to be a valid path again.

### Short-Circuit-Path Search

The short-circuit-path search is listed in Algorithm 5. It computes the short-circuit paths defined in the previous section. Algorithm 5 requires the graph $G = (N, E, \phi^+, \phi^-, \tau)$ initialized with the voltage from voltage propagation and the sets $N_{gnd}$ and $N_{vdd}$ as input. Paths from nodes in $N_{vdd}$ to nodes in $N_{gnd}$ are searched. Therefore, a recursive depth-first-search algorithm is started from each node in $N_{vdd}$ (line 2). The depth-first search is listed in Algorithm 6 and called in line 5 of Algorithm 5. The set $N_{dfs} \subseteq N$ and the path $P_{dfs} \in \mathscr{P}$ are initialized empty in lines 3 and 4 of Algorithm 5. $N_{dfs}$, $P_{dfs}$ and the current visited node $n_{dfs} \in N$ are passed on by value during the depth-first search, i.e., each recursive call has a copy of the data. The depth-first-search algorithm returns a set of short-circuit paths $\mathcal{P}_{result}$ for each starting node in $N_{vdd}$. All detected short-circuit paths are collected in the set $\mathcal{P}_{SC}$ (line 6 of Algorithm 5).

The depth-first search (Algorithm 6) recursively calls itself (line 15) for each edge that starts at the current node and which has a state that is not $off$ (line 13). The graph $G$, $N_{gnd}$, $N_{vdd}$, the next node $\phi^-(e)$, $N_{dfs}$ and the new current path $P'_{dfs}$ are passed on to the recursive call. The new current path $P'_{dfs}$ is the old current path $P_{dfs}$ extended with the current edge. The recursion stops descending if the current node $n_{dfs}$ is in the set of currently visited nodes $N_{dfs}$ (line 2). This condition avoids infinite recursions along cycles in the graph. A path from a node in $N_{vdd}$ to another node in $N_{vdd}$ has been found if the current node is in $N_{vdd}$. An empty set is returned in line 5 as this cannot be short-circuit path. A short-circuit path has been found if the current visited node is in

**Algorithm 6** DepthFirstSearch [Zwerger and Graeb, 2013]

**Require:** ($G$, $N_{gnd}$, $N_{vdd}$, $n_{dfs}$, $N_{dfs}$, $P_{dfs}$)

1: **if** $n_{dfs} \in N_{dfs}$ **then**
2: **return** {} // *cycle found*
3: **end if**
4: **if** $n_{dfs} \in N_{vdd}$ **then**
5: **return** {} // *path from* $N_{vdd}$ *to* $N_{vdd}$
6: **end if**
7: **if** $n_{dfs} \in N_{gnd}$ **then**
8: **return** $\{P_{dfs}\}$ // *short circuit path found*
9: **end if**
10:
11: $N_{dfs} := N_{dfs} \cup \{n_{dfs}\}$
12: $\mathcal{P}_{result} = \{\}$
13: **for all** $e \in \{e \in E \mid \phi^+(e) = n_{dfs} \wedge \sigma(e) \neq off\}$ **do**
14: $P'_{dfs} := \text{push}(P_{dfs}, e)$ // *see* (3.22)
15: $\mathcal{P}_{new} := \text{DepthFirstSearch}(G, N_{gnd}, N_{vdd}, \phi^-(e), N_{dfs}, P'_{dfs})$
16: $\mathcal{P}_{result} := \mathcal{P}_{result} \cup \mathcal{P}_{new}$
17: **end for**
18:
19: **return** $\mathcal{P}_{result}$

$N_{gnd}$. The found path is returned in line 8. The returned paths are collected in $\mathcal{P}_{result}$ (line 16) and returned (line 19).

#### Complexity

The complexity of Algorithm 5 is linear, i.e., it is $O(|E|)$. The following reasoning leads to this result [Zwerger and Graeb, 2013]: The complexity of the depth-first search is linear in the number of edges, i.e., $O(|E|)$ [Sedgewick, 1988]. Consequently, the complexity of Algorithm 6 is linear in the number of edges. The number of iterations of the loop in Algorithm 5, i.e. the number of calls of Algorithm 6, is defined by the size of $N_{vdd}$. It can be assumed that $|N_{vdd}|$ is constant (and not proportional to the circuit size). This is a reasonable assumption as the number of pins for a chip is limited. It follows that the overall complexity of Algorithm 5 is linear, i.e., it is $O(|E|)$.

### 3.2.6. Floating Node and Short-Circuit Verification

In this section, the voltage propagation algorithm described in section 3.2.4 and the short-circuit-path search described in section 3.2.5 are combined to the complete verification algorithm. The complete algorithm for floating node and short-circuit verification

**Algorithm 7** FloatingNodeAndShortCircuitVerification [Zwerger and Graeb, 2013]

**Require:** circuit netlist, $N_{gnd}$, $(N_{vdd_1}, \ldots, N_{vdd_{n_{sup}}})$, $R_T$

1: $N_{vdd} = N_{vdd_1} \cup \cdots \cup N_{vdd_{n_{sup}}}$
2: Build graph $G = (N, E, \phi^+, \phi^-, \tau)$ according to Fig. 3.3.
3: $\mathcal{P}_{SC} := \{\}$
4: $\mathcal{P}_{SC,old} := \{\}$
5: $N_{sc} := \{\}$ *// nodes with state sc*
6:
7: **repeat**
8: $\quad \mathcal{P}_{SC,old} := \mathcal{P}_{SC}$
9: $\quad G :=$ VoltagePropagation($G$, $N_{gnd}$, $(N_{vdd_1}, \ldots, N_{vdd_{n_{sup}}})$, $N_{sc}$) *// Algorithm 4*
10: $\quad \mathcal{P}_{SC} :=$ ShortCircuitPathSearch($G$, $N_{gnd}$, $N_{vdd}$) *// Algorithm 5*
11: $\quad N_{sc} :=$ nodesOfPathSet($\mathcal{P}_{SC}$) *// see* (3.24)
12: $\quad N_{sc} := N_{sc} \setminus (N_{gnd} \cup N_{vdd})$
13: **until** $\mathcal{P}_{SC} = \mathcal{P}_{SC,old}$
14:
15: Partition the set $\mathcal{P}_{SC}$ into $\mathcal{P}_{SC,def}$, $\mathcal{P}_{SC,pot}$ and $\mathcal{P}_{SC,sc}$ according to formulas (3.18), (3.19) and (3.20)
16:
17: **return** $G$, $\mathcal{P}_{SC,def}$, $\mathcal{P}_{SC,pot}$, $\mathcal{P}_{SC,sc}$

is listed in Algorithm 7. A description of the verification algorithm and definitions used by the algorithm are provided in the following [Zwerger and Graeb, 2013]. At the end of this section, the runtime complexity of the overall verification method is determined. The analysis of the complexity builds upon the runtime complexities for voltage propagation and short-circuit-path search that were determined in sections 3.2.4 and 3.2.5, respectively.

### Definitions

The nodes that are located on a short-circuit path need to be computed in order to mark them with state *sc*. Therefore, a nodesOfPath() operation is defined. It returns the set of nodes which are located on path $P$ and is defined as follows:

$$\text{nodesOfPath}(P) \Longleftrightarrow \{n \in N \mid \text{“Node } n \text{ is located on path } P\text{”}\} \tag{3.23}$$

This definition is extended for sets of paths. The nodesOfPathSet() operation applies nodesOfPath() to each path $P_i$ in a set of paths $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$. It is defined as follows:

$$\text{nodesOfPathSet}(\mathcal{P}) \Longleftrightarrow \bigcup_{i=1\ldots n} \text{nodesOfPath}(P_i) \tag{3.24}$$

All nodes on all short-circuit paths need to be marked with the voltage *sc*. Therefore, the nodesOfPathSet() operation is used in line 11 of Algorithm 7 to compute all nodes on all detected short-circuit paths.

### Verification Algorithm

The floating node and short-circuit verification algorithm listed in Algorithm 7 is described in the following. The circuit netlist, the sets $N_{gnd}$ and $N_{vdd,i}$ and the threshold resistance $R_T$ are input data of the algorithm. As first step, the graph $G = (N, E, \phi^+, \phi^-, \tau)$ is built according to Fig. 3.3 (line 2). The set $N_{sc}$ tracks all nodes which are located on a short-circuit path. It is initialized in line 5. The main loop of the algorithm executes voltage propagation (Algorithm 4) and short-circuit-path search (Algorithm 5) until no more new short-circuit paths are found (see lines 7 to 13).

The short-circuit-path search called in line 10 returns the set of short-circuit paths $\mathcal{P}_{SC}$. The result depends on the current state of the node voltages and edges in the graph $G = (N, E, \phi^+, \phi^-, \tau)$. All nodes on short-circuit paths, except the nodes in $N_{vdd}$ and $N_{gnd}$, are added to the set $N_{sc}$ (line 12). They are initialized with voltage *sc* in the next iteration of voltage propagation. Furthermore, the voltage *sc* will never be overwritten with any other voltage. This is ensured by the the propagation condition (3.14). It only allows $float$ to be overwritten with $gnd$ or $vdd_i$.

The main loop terminates if no more short-circuit paths are found (line 13). The set $\mathcal{P}_{SC}$ of the last iteration is partitioned into $\mathcal{P}_{SC,def}$, $\mathcal{P}_{SC,pot}$, $\mathcal{P}_{SC,sc}$ according to formulas (3.18), (3.19) and (3.20). The verification report is generated based on $\mathcal{P}_{SC,def}$, $\mathcal{P}_{SC,pot}$, $\mathcal{P}_{SC,sc}$ and the node voltages of the last iteration.

### Complexity

The complexity of the overall method can be determined based on the complexities of voltage propagation and short-circuit search. The worst-case complexity of Algorithm 7 is $O(|E|^3)$ due to the following reasoning [Zwerger and Graeb, 2013]: The largest number of iterations occurs if one short-circuit path is detected in the first iteration which causes an induced short-circuit path. This induced short-circuit path is detected in the next iteration and causes another induced short-circuit path so that in each iteration one new induced short-circuit path is detected. Each short-circuit path consists of at least one edge. Thus, the loop will be repeated less than $|E|$ times. The effort in the loop is dominated by the quadratic worst-case complexity of voltage propagation. Thus, the overall worst-case complexity for the verification algorithm is $O(|E|^3)$.

| | Definite SC | Potential SC | Floating Node |
|---|---|---|---|
| Miller OTA without $MS4$ | none | ($MS3$, $M7$) | $n_1$, $n_3$ |
| Miller OTA with $MS4$ | none | none | $n_1$ |
| Oscillator without $MS3$ and $MS4$ | none | ($M7$, $M8$) | $n_3$, $n_4$, $n_5$ |
| Oscillator with $MS3$ and $MS4$ | none | none | none |
| BiCMOS OTA without $MS2$ | none | none | $n_{out}$ |
| BiCMOS OTA with $MS2$: | ($M4$, $M12$) | none | $n_{in}$ |

Table 3.2.: Verification reports [Zwerger and Graeb, 2012a, 2013]

| | power-up | power-down |
|---|---|---|
| total power | 729 $\mu W$ | 804 $pW$ |
| gate voltage $M7$ | 617 $mV$ | 80.7 $mV$ |
| drain current $M7$ | 118 $\mu A$ | 132 $pA$ |

Table 3.3.: DC simulation results of the Miller OTA (without $MS4$) [Zwerger and Graeb, 2012a, 2013]

### 3.2.7. Experimental Results

In this section, the floating node and short-circuit path verification is demonstrated for a Miller OTA, an oscillator circuit and a symmetrical BiCMOS OTA [Zwerger and Graeb, 2012a, 2013]. The results for the Miller OTA demonstrate the difficulties of numerical simulation in presence of floating nodes. The results for the oscillator circuit are an example of how the method distinguishes between potential short-circuit paths and induced short-circuit paths. The BiCMOS OTA demonstrates the ability of the method to handle bipolar devices. Table 3.2 summarizes the verification reports for the experiments.

#### Miller OTA

The top of Fig. 3.5 shows the schematic of the Miller OTA. The bulk pins of n-MOSFETs and p-MOSFETs are connected to $n_{gnd}$ and $n_{vdd}$, respectively. Those default bulk connections are not drawn in the schematic. However, they are taken into account during the computation. The amplifier is configured as voltage follower, i.e., negative input and output are connected. It has the power-down switches $MS1$, $MS2$, $MS3$ and $MS4$. Two experiments were carried out. The circuit is analyzed without switch $MS4$ in the first experiment. In a second experiment, $MS4$ is included.

Net $n_{\overline{pwd}}$ is pulled to $gnd$ in power-down mode. Consequently, the power-down switches $MS1$, $MS2$ and $MS3$ are conducting. It follows that the net $n_{bias}$ is pulled to $vdd$ by $MS2$ and the bias currents through $M1$ and $M6$ are switched off. The intended functionality of the power-down switches $MS1$ and $MS2$ is to pull the input ($n_{in}$) and

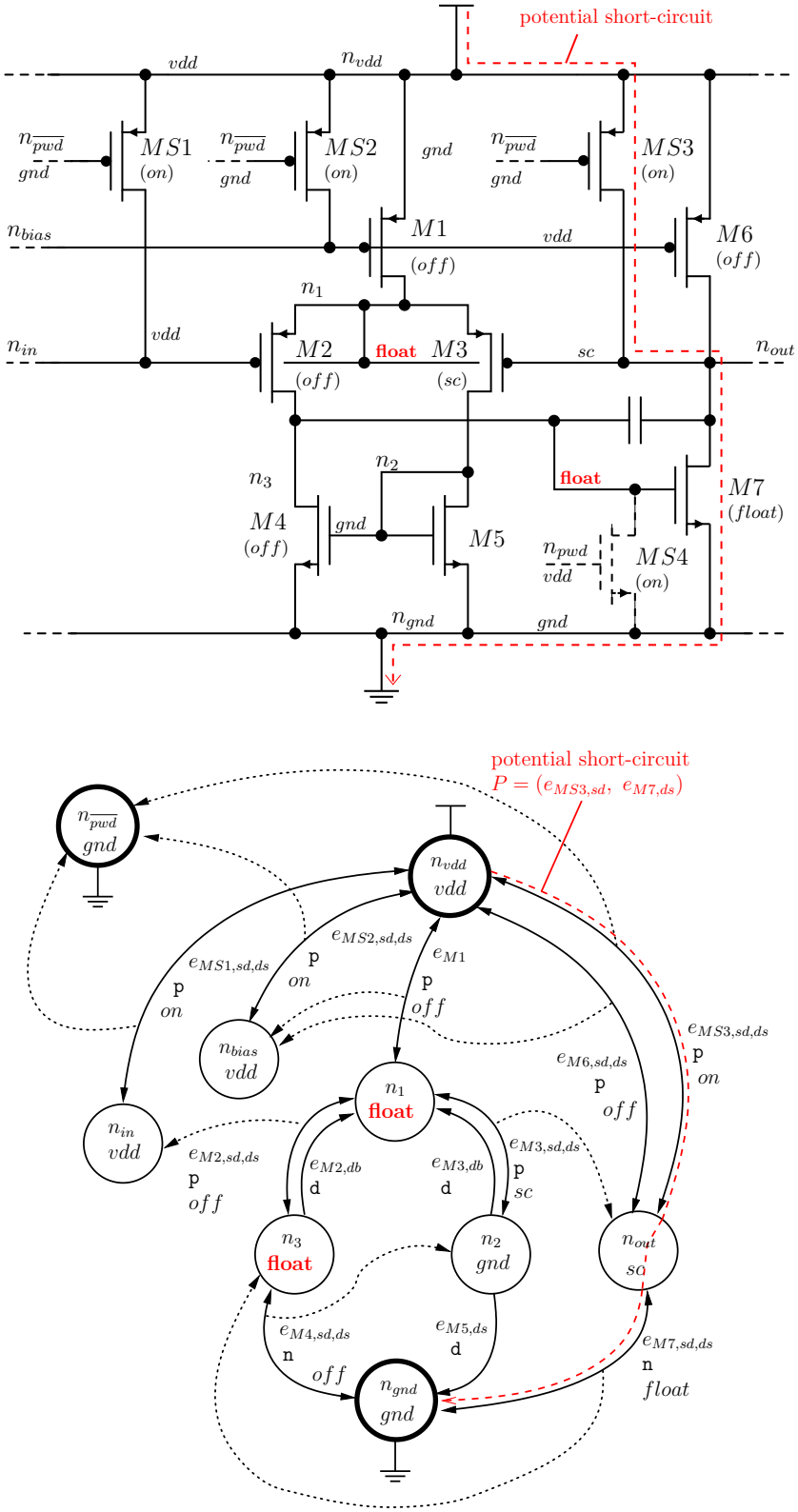



Figure 3.5.: Miller OTA with power-down circuitry and corresponding graph model [Zwerger and Graeb, 2012a, 2013]

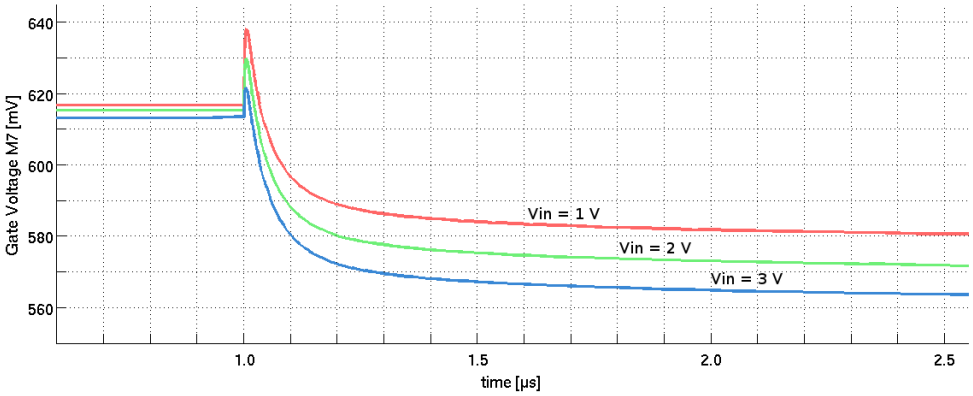



Figure 3.6.: Transient voltage at gate of $M7$ [Zwerger and Graeb, 2013]

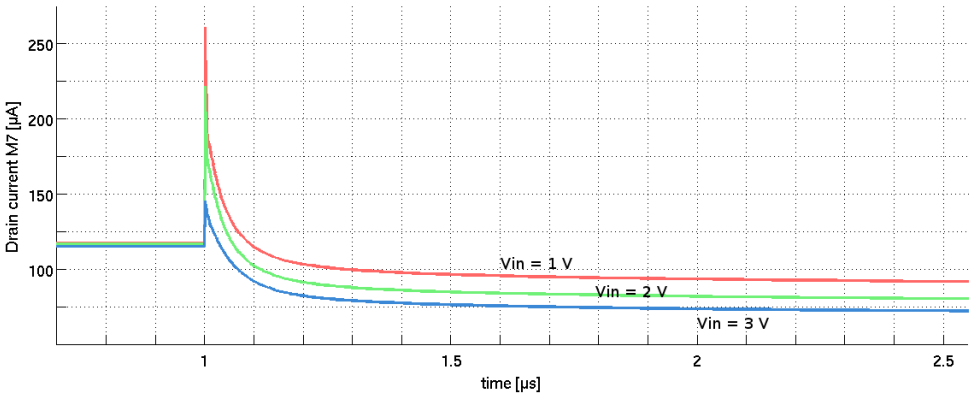



Figure 3.7.: Transient drain current through $M7$ [Zwerger and Graeb, 2013]

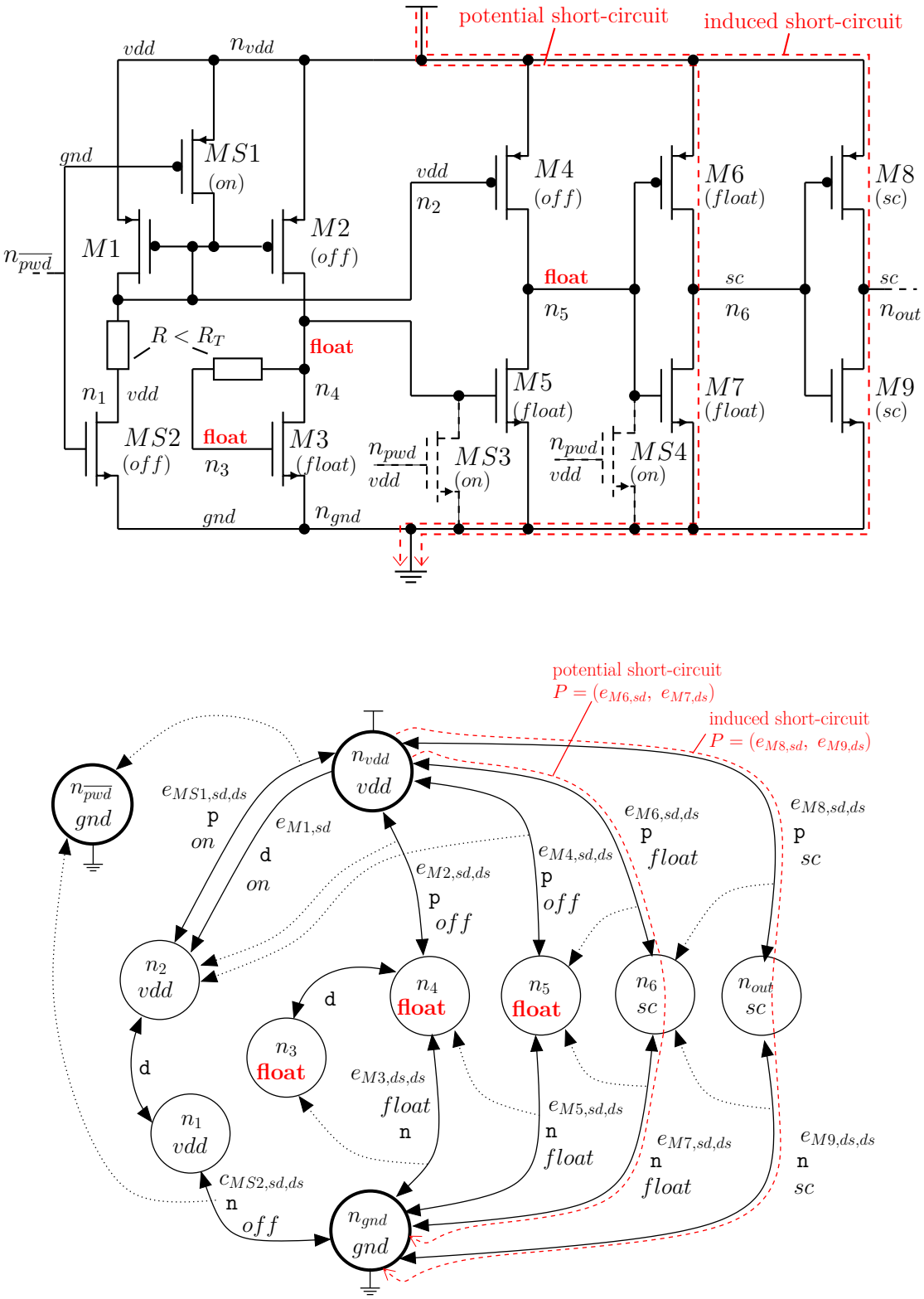



Figure 3.8.: Crystal oscillator with corresponding graph [Zwerger and Graeb, 2012a, 2013]

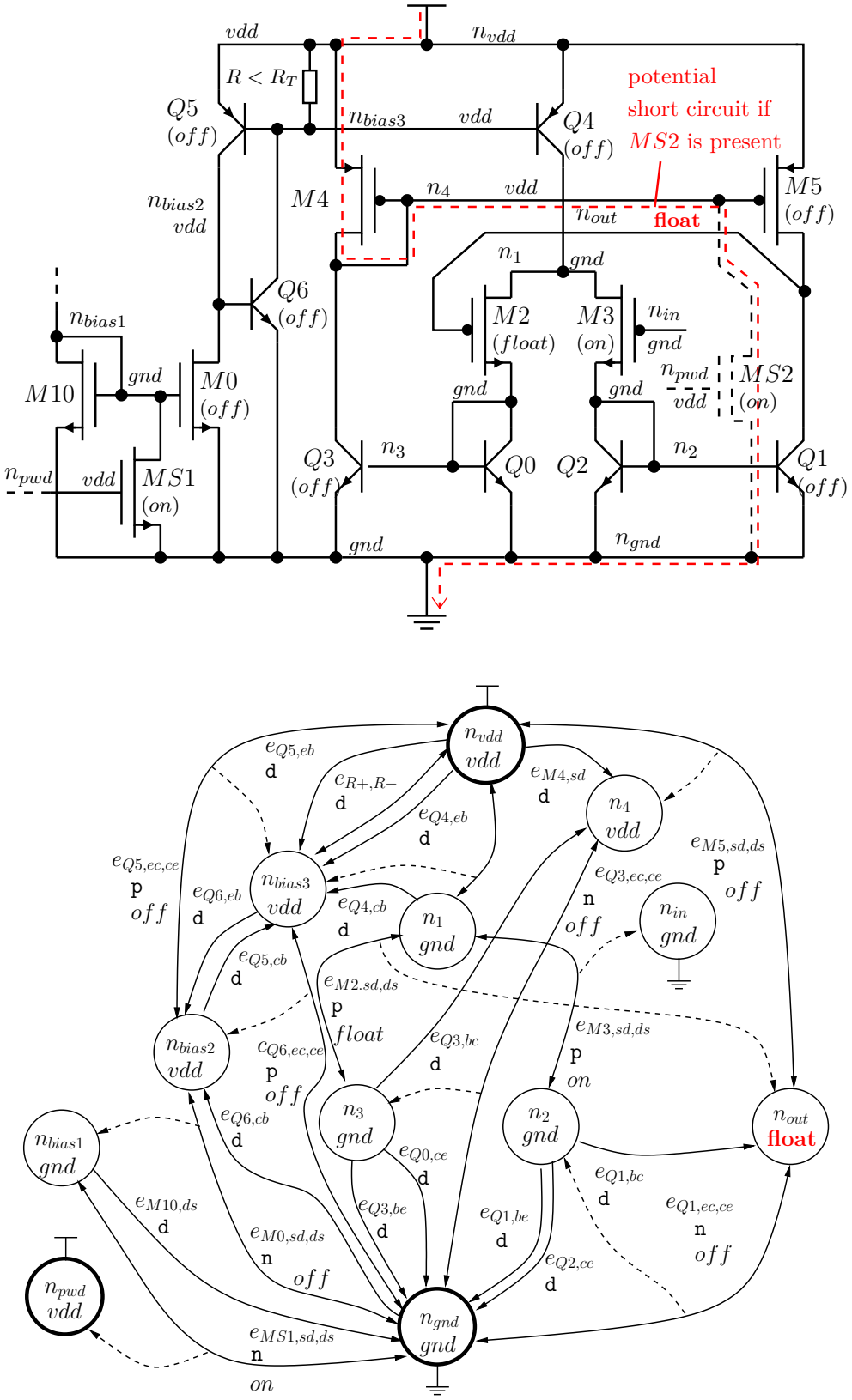



Figure 3.9.: Symmetrical BiCMOS OTA with corresponding graph [Zwerger and Graeb, 2012a, 2013]

the output ($n_{out}$) to $vdd$ in power-down mode to avoid device aging due to NBTI at the differential pair $M2, M3$ [Schlünder, 2010; Schlunder et al., 2003].

The bottom of Fig. 3.5 shows the graph model and the switching function of the Miller OTA without $MS4$. The diode edges modeling the bulk diodes with default connections to $n_{gnd}$ or $n_{vdd}$ are not drawn for reasons of better readability. They never propagate and can be omitted in the presentation. The bulk pins of $M2$ and $M3$ are connected to their own well, i.e., to net $n_1$. The source to bulk edges of $M2$ and $M3$ are shorted. They are left out in the illustration. The drain to bulk edges are drawn as they might propagate voltages.

The node voltages and edge states after termination of Algorithm 7 are annotated in the circuit and the graph. The line "Miller OTA without $MS4$" in Table 3.2 shows the verification report for the first experiment. There is one potential short-circuit path from $n_{vdd}$ over $MS3$ to $n_{out}$ over $M7$ to $n_{gnd}$. This short-circuit path is drawn as a (red) dashed line in the schematic as well as in the graph. Furthermore, there are two floating nodes $n_1$ and $n_3$. The potential short circuit is caused by the floating gate of $M7$. Due to the floating gate it is not ensured that $M7$ is turned off properly.

The results of voltage propagation can be compared to numerical simulation. Cadence's simulator Spectre [Cadence Design Systems, 2015b] together with transistor models from a 350 $nm$ technology of Austria Microsystems were used for the following experiment. The results of DC simulation of the Miller OTA (without $MS4$) are given in Table 3.3. The total power consumption in power-up mode is 729 $\mu W$. The power-consumption drops to 804 $pW$ in power-down mode, i.e., the circuit seems to be turned off properly. The gate voltage is reduced to a voltage below the threshold voltage. Consequently, the drain current of $M7$ is reduced to approximately zero. The power-down mode seems to work correctly according to the DC simulation results.

Figs. 3.6 and 3.7 show the results of transient simulation of the Miller OTA without $MS4$. The gate voltage of $M7$ is plotted over the time axis in Fig. 3.6. Fig. 3.7 plots the drain current of $M7$ over time. There are three waveforms in each plot. Each waveform corresponds to a different initial value of the input signal $n_{in}$. The power-down mode is activated, i.e., $n_{\overline{pwd}}$ is driven from $vdd$ to $gnd$ at $t = 1\ \mu s$.

The transient simulation results reveal the problem caused by the floating gate of $M7$. The gate voltage of $M7$ is not significantly reduced in power-down mode according to the transient simulation results. It approaches the steady state with a very large time constant. The gate voltage is translated to the drain current shown in Fig. 3.7. The drain current through $M7$ also decreases with a very large time constant. Consequently, it is not turned off properly.

Transient analysis has revealed that there is a problem. Following a simulation approach, the unwanted current flow has to be traced down manually towards $M7$. This effort can be saved by using the structural voltage propagation method of this thesis. The presented

algorithm automatically detects and *locates* the problem. Algorithm 7 has calculated the exact location of the root cause.

For a second experiment, the transistor $MS4$ is added to the amplifier. The additional switch $MS4$ should pull the gate of $M7$ to *gnd* in order to eliminate the floating node and turn off the current properly. The line "Miller OTA with $MS4$" of Table 3.2 shows the verification report of this experiment. The potential short-circuit is gone and $n_3$ is no longer floating. There is one floating node $n_1$ remaining.

### Crystal Oscillator

The schematic of the crystal oscillator [Blieck and Janssens, 1996] is shown in the upper part of Fig. 3.8. Net $n_{\overline{pwd}}$ is driven to *gnd* in power-down mode. This should turn off all the currents through the oscillator. The lower part of Fig. 3.8 shows the graph model of the oscillator without $MS3$ and $MS4$. Bulk edges are omitted analogously to the Miller OTA described in the previous section. Two experiments for the oscillator are discussed in the following. Only the power-down switches $MS1$ and $MS2$ are present in the circuit during the first experiment. The switches $MS3$ and $MS4$ are added to the circuit in the second experiment.

The line "Oscillator without $MS3$ and $MS4$" in Table 3.2 presents the verification report of the first experiment. There are three floating nodes, one potential short circuit from $n_{vdd}$ over $M6$ to net $n_6$ over $M7$ to net $n_{gnd}$ and one induced short circuit from $n_{vdd}$ over $M8$ to net $n_{out}$ over $M9$ to net $n_{gnd}$. The short circuits are drawn with (red) dashed lines in the schematic as well as in the graph.

The potential short circuit is caused by the floating node $n_5$. A short circuit from $n_{vdd}$ over $M8$ to net $n_{out}$ over $M9$ to net $n_{gnd}$ is induced by this potential short circuit. The voltage of net $n_6$ depends on the voltage of the floating node $n_5$ which is causing the potential short circuit. Consequently, it is not ensured that the current through $M8$ and $M9$ is turned off. However, the induced short-circuit path will most likely disappear once the root cause is eliminated. The root cause is the potential short-circuit path over $M6$ and $M7$. Algorithm 7 discovers the same potential short-circuit paths as the method in [Blieck and Janssens, 1996]. Additionally, it distinguishes between cause of the problem and symptom by distinguishing between potential and induced short-circuit paths.

### BiCMOS OTA

The the schematic for the BiCMOS OTA is given in the upper part of Fig. 3.9. It is configured as voltage follower, i.e. input and output are connected. The BiCMOS OTA has one power-down switch $MS1$. It turns off the bias current through $M0$. Two configurations are discussed in the following. The first experiment is carried out without $MS2$. The additional power-down switch $MS2$ is added in the second experiment. The

| | | without $MS2$ | with $MS2$ |
|---|---|---|---|
| power-up | total power | 184 $\mu W$ | 184 $\mu W$ |
| power-down | total power | 255 $pW$ | 195 $\mu W$ |
| | total current | 122 $pA$ | 93 $\mu A$ |
| | drain-current through $MS2$ | – | 93 $\mu A$ |

Table 3.4.: DC simulation of the BiCMOS OTA [Zwerger and Graeb, 2012a, 2013]

lower part of Fig. 3.9 shows the graph model of the BiCMOS OTA without $MS2$. Bulk connections are omitted in the illustration but taken into account in the computation.

The verification report of the first experiment is given in line "BiCMOS OTA without $MS2$" of Table 3.2. It shows that one single power-down switch ($MS1$) is sufficient to turn off all currents. No short-circuit path is reported. The output net $n_{out}$ is floating in power-down mode. Adding transistor $MS2$ as additional power-down switch is one possible attempt to eliminate the floating node. The line "BiCMOS OTA with $MS2$" of Table 3.2 shows the verification report of this experiment. It reveals that adding $MS2$ introduces another unintended short-circuit path from $n_{vdd}$ over diode-connected transistor $M4$ to net $n_4$ over $MS2$ to $n_{gnd}$.

A DC simulation has been performed for the BiCMOS OTA. The results are shown in Table 3.4. The amplifier consumes 184 $\mu W$ during power-up state (with and without $MS2$). The total current as well as the total power decrease to approximately zero in power-down mode for the configuration without $MS2$. In contrast, the total consumed power in power-down mode even increases to 195 $\mu W$ for the configuration with $MS2$. The total consumed current of 93 $\mu A$ flows entirely through the drain of $MS2$, i.e., along the detected short-circuit path.

### Experiments on Larger Circuits and Runtimes

Algorithm 7 was implemented in C++. With this implementation the runtimes were measured. Runtimes include the total time of the program run, including reading in the circuit, running the algorithm and writing the results to a text file. The runtimes for the small examples (MillerOTA, Crystal Oscillator, BiCMOS OTA) were all negligible ($< 10\ ms$). The algorithm was additionally tested on four industrial analog IO blocks: The algorithm was executed for a low voltage differential signaling transmitter (LVDS Tx), a low voltage differential signaling receiver (LVDS Rx), a combined blocks. The combined blocks combine a LVDS interface with a general purpose input/output (GPIO) interface. The combined circuits are denoted as LVDS/GPIO Tx and LVDS/GPIO Rx. The combined cells also include a significant portion of digital control logic which

| Circuit | $\|D\|$ | $\|N\|$ | $\|N_{gnd} \cup N_{vdd}\|$ | runtime[(1)] |
|---|---|---|---|---|
| LVDS Tx | 901 | 354 | 17 | 3.37 $s$ |
| LVDS Rx | 620 | 265 | 15 | 1.09 $s$ |
| LVDS/GPIO Rx | 3032 | 1067 | 38 | 8.15 $s$ |
| LVDS/GPIO Tx | 3385 | 1192 | 42 | 57.1 $s$ |

(1) Intel(R) Xeon(R) CPU X5670 @ 2.93GHz

Table 3.5.: Runtimes for larger industrial circuits

| | Definite SC | Potential SC | Floating Nodes | Floating Gates |
|---|---|---|---|---|
| LVDS Tx | 0 | 0 | 23 | 0 |
| LVDS Rx | 0 | 0 | 16 | 0 |
| LVDS/GPIO Tx | 0 | 0 | 75 | 2 |
| LVDS/GPIO Rx | 0 | 0 | 152 | 12 |

Table 3.6.: Verification reports for industrial circuits

controls different modes of the block. The circuits are the same circuits as used for the experiments for structure recognition (section 2.4).

Table 3.5 summarizes the circuit sizes and the total runtimes of the experiments. Column $|N|$ denotes the number of nets, column $|D|$ denotes the number of devices and column $|N_{gnd} \cup N_{vdd}|$ denotes the number of supply and input signal nets. All runtimes are below one minute. It was also analyzed if a relation of those figures to the theoretically computed runtime complexities can be seen. However, based on the given four experiments no solid conclusion can be drawn. A comprehensive study on a bigger set of circuits of a wider range of sized would be needed which is a task for future research. Nevertheless, the results demonstrate that the runtimes are very practicable for realistic circuit sizes.

The verification reports for the four industrial circuits are given in Table 3.6. No definite or potential short-circuit path was found. Consequently, it is verified that the all currents are safely turned off in power-down mode.

The column "Floating Nodes" lists the number of nodes with voltage level $float$ as additional information. As stated in section 3.2.1, it should be considered as a good design style to avoid floating nodes. However, floating nodes are not automatically an design error. The decision if it is worth to add an additional power-down switch to avoid the floating node depends on many aspects. A hint that a floating node might be critical is that it is connected to a transistor gate, i.e., that there is a floating transistor gate. The number of floating gates is listed in the corresponding column in Table 3.2. A further study if floating nodes are critical with respect to device aging is done in section 3.3.

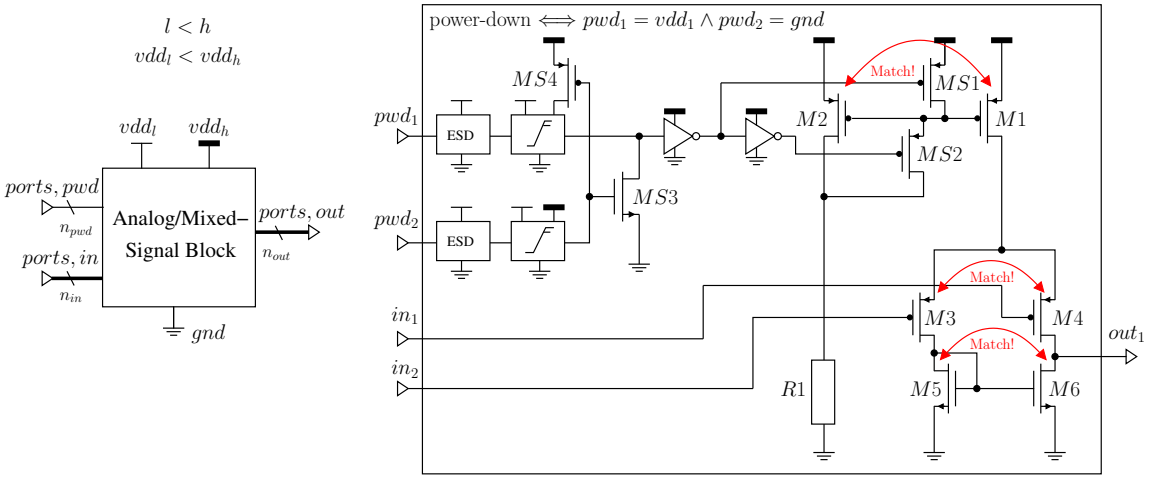



Figure 3.10.: Analog/mixed-signal block with two supply voltages and power-down mode on left side and differential amplifier with matched device pairs as example on right side

## 3.3. Detection of Asymmetric Aging-Critical Voltages

### 3.3.1. Problem Description

Analog circuits rely heavily on so called matching. Matched devices either have to be as identical as possible, or their properties, e.g. their resistances, have to have an exact ratio. On one hand, a lot of effort in analog circuit design is made to achieve good matching. The commonly applied approaches are sizing rules for robustness [Massier et al., 2008] and sophisticated layout structures [Hastings, 2006]. Furthermore, in order to determine the lifetime of circuits, comprehensive aging simulations [Cadence Design Systems, 2013] are carried out. On the other hand, less attention is paid on verifying that the matching is not impaired and that there are no reliability problems induced by the power-down circuitry. The method described in this section contributes to closing this gap.

In the previous section, a method for verifying the basic functionality, i.e., the detection of short-circuits and floating nodes in power-down mode was described. The solution approach is based on voltage propagation. If there are no short circuits in power-down mode, voltage propagation computes an estimate for the power-down voltage of each node in the circuit. In power-down mode, internal nets of the circuit are pulled to the positive supply ($vdd$) or to ground ($gnd$). Consequently, transistors are exposed to different stress voltages than during normal operation. As the nodes are pulled to one of the extremes ($gnd$ or $vdd$) the stress voltages are usually higher than in normal operation, i.e., device degradation due to aging effects like hot carrier injection (HCI) [Chouard, 2012; Cadence Design Systems, 2013; Schlünder, 2010] or negative bias in-

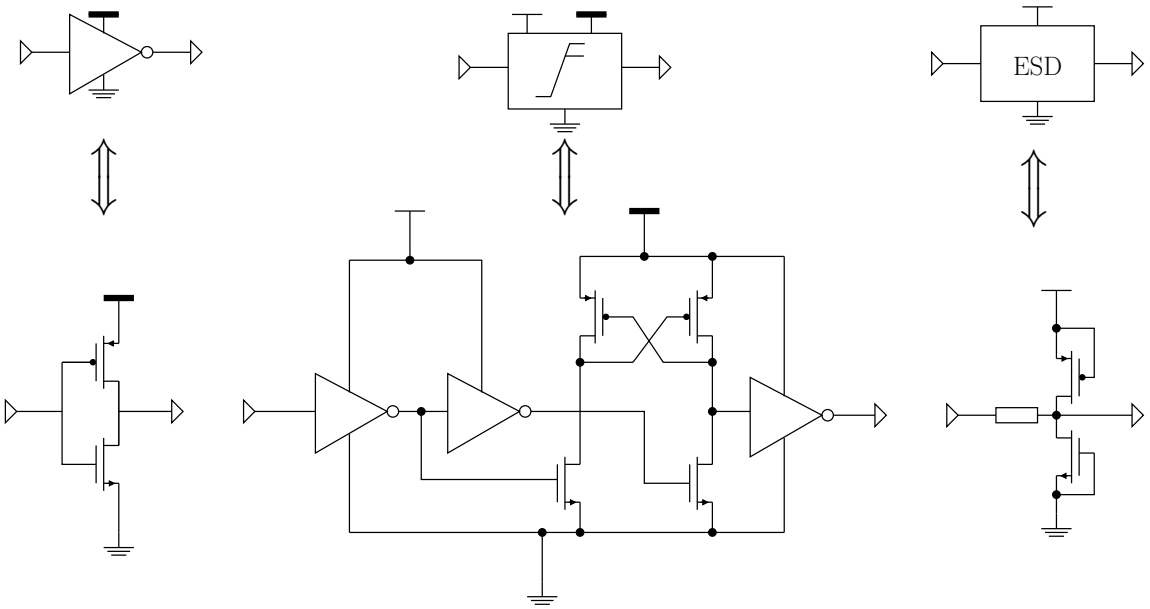



Figure 3.11.: Examples for level-up shifter and ESD protection cells

stability (NBTI) [Maricau and Gielen, 2011; Cadence Design Systems, 2013; Schlunder et al., 2003] can be significant. Therefore, the matching can degrade during power-down mode [Michael et al., 1992; Chen et al., 2001], i.e., there is time dependent mismatch in the circuit. Once the mismatch reaches a critical level, the circuit fails when it is powered up again.

Time dependent mismatch can be avoided by ensuring symmetrical power-down voltage conditions at symmetrical (matched) structures. The task of finding critical conditions can be partitioned into determining critical symmetrical structures on the one hand and checking the power-down voltage conditions that appear on those structures on the other hand. The problem will be described using the example circuit shown in Fig. 3.10 in the the remainder of this section.

For a given analog/mixed-signal block, it should be verified that matched devices are exposed to matched voltages in power-down mode. The circuit inside the block is connected to the outer world or other blocks on a higher hierarchy level by ports. In this section, we distinguish between ports that control the power mode, ports that do not control the power mode and supply ports. There are three supply ports in the circuit in Fig 3.10: a low supply voltage ($vdd_l$), a high supply voltage ($vdd_h$) and ground ($gnd$). This thesis follows the convention that a higher supply voltage is marked with a thicker horizontal bar than a lower one. Furthermore, the circuit in Fig 3.10 has $n_{pwd}$ pins that control the power mode and $n_{in} + n_{out}$ general input/output ports. The power-mode controlling pins and the general input/output ports are labeled with $pwd$ and $ports$, respectively.

On the right of Fig. 3.10, a simple differential amplifier is shown as example. The meaning of the used symbols is given in Fig. 3.11. The amplifier block is connected to a low voltage $vdd_1$ and a high voltage $vdd_2$. This is a typical situation for analog input/output blocks which receive signals from the digital core of the chip (low voltage domain) and operate in an analog voltage domain with higher supply voltage. The power-down signals $pwd_1$ and $pwd_2$ are passing electrostatic discharge (ESD) protection cells, level shifters and inverters to control the power-down switches $MS1$ to $MS4$. The circuit is powered down if $pwd_1$ is receiving a digital 1 signal, i.e. a voltage of $vdd_1$, and $pwd_2$ is receiving a digital 0 signal, i.e., it is pulled to $gnd$. For the other digital voltage combinations of $pwd_1$ and $pwd_2$ the amplifier is in normal operation mode. This can be summarized as follows:

- Power-down mode: $(pwd_1, pwd_2)$ is $(vdd_1, gnd)$
- Power-up mode: $(pwd_1, pwd_2)$ is one of $(gnd, vdd_1)$, $(gnd, gnd)$, $(vdd_1, vdd_1)$

The working principle of the circuit relies on the matching of the current mirrors ($M1$, $M2$) and ($M5$, $M6$) as well as the matching of the differential pair ($M3$, $M4$). The matched pairs are marked with red double arrows. Depending on the reliability requirements of the application and the aging behavior of the used technology, the matched transistor pairs have to be protected from asymmetric stress with an adequate stringency for all specified external conditions for $in1$, $in2$ and $out1$. This requirement should be verified automatically. In the following, the task is solved by combining the structure recognition approach from chapter 2 with the voltage propagation approach from section 3.2.

### 3.3.2. Overview and Definitions

The problem described in the previous section is solved by a method for automatic detection of aging critical voltage conditions in power-down mode. In this section, the overall method is described with emphasis on the general overview over the individual steps involved. The individual steps will be described in detail in later sections. The overview over the method is given by Fig. 3.12. Each block corresponds to an individual analysis step, i.e. an algorithm. The arrows indicate data being passed on. The dotted frame surrounding the diagram marks the interface of the algorithm. Arrows going into the dotted frame correspond to input data that needs to be given. In Figs. 3.14, 3.15 and 3.17, the execution of the algorithm is illustrated for the example circuit shown in Fig. 3.10. The figures are illustrating the definitions made in the following. The text in this section will follow the flow shown in Fig. 3.12 from top to bottom.

The method takes the circuit netlist $(N, D, \Gamma)$ as input. The set $N$ denotes the set of nets. The set $D$ is the set of devices. The connectivity of the circuit is represented by a function $\Gamma(p)$ which maps each pin $p$ to a net or the value *unconnected*. It is defined as

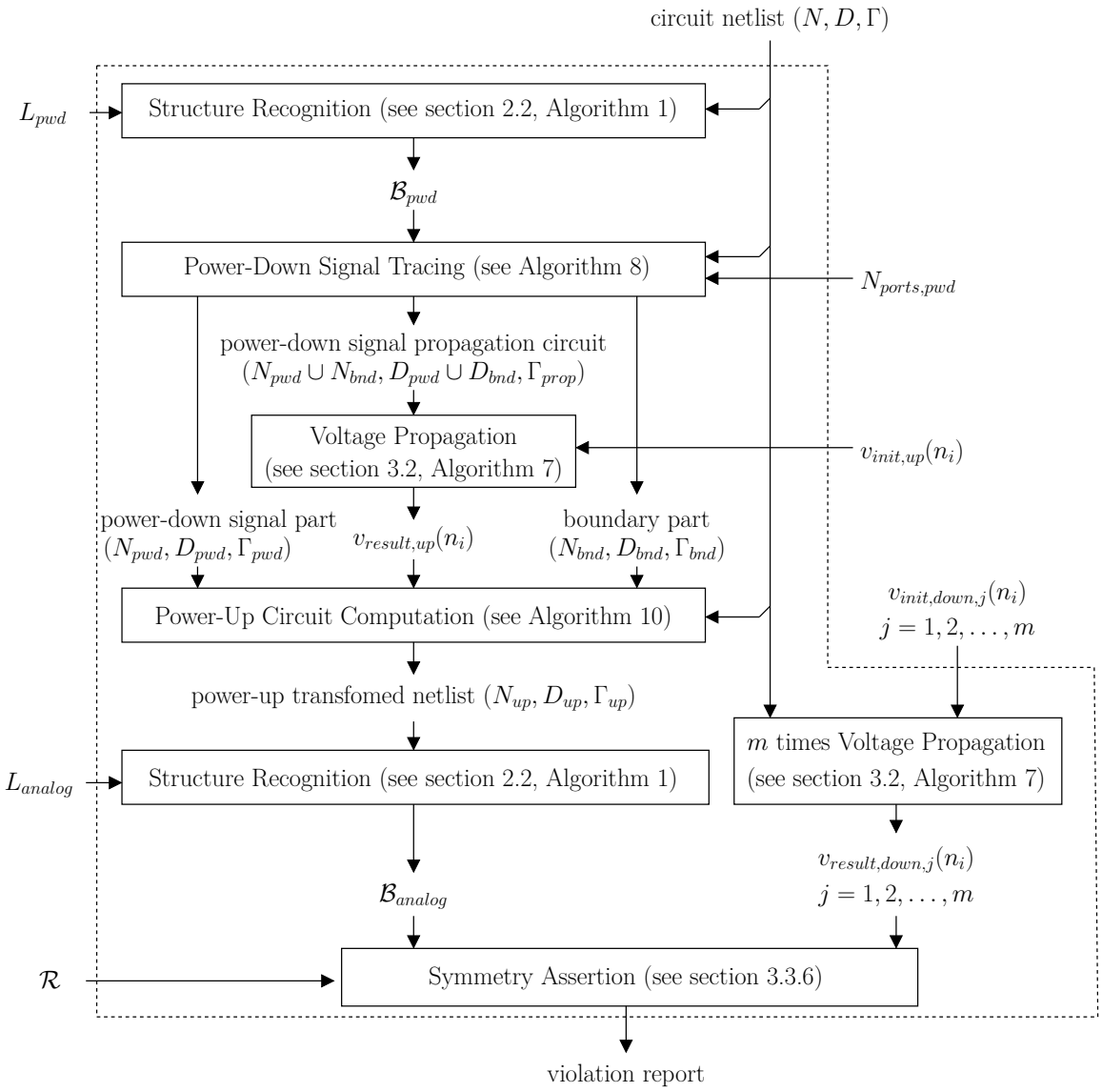



Figure 3.12.: Overview over asymmetric stress detection

follows:

$$\Gamma: \qquad P \to N \cup \{unconnected\} \qquad p \mapsto \Gamma(p) \tag{3.25}$$

with

$$\Gamma(p) = n \iff \text{“pin } p \text{ is connected to net } n\text{.”} \tag{3.26}$$

and

$$\Gamma(p) = unconnected \iff \text{“pin } p \text{ is not connected to any net.”} \tag{3.27}$$

The set $P$ is the set of device pins. The function $\Gamma(p)$ assigns exactly one net $n \in N$ to each pin $p \in P$, i.e., each pin is connected to exactly one net or it is *unconnected*. The set $P$ is the union of all pins of all devices and defined as follows:

$$P = \bigcup_{d \in D} P_d \tag{3.28}$$

The sets $P_d$ represent the pins of each device $d$. For example, the set $P_{M1}$ for a MOSFET $M1 \in D$ is given as follows:

$$P_{M1} = \{p_{M1,d},\ p_{M1,s},\ p_{M1,g},\ p_{M1,b}\}. \tag{3.29}$$

The elements of $P_{M1}$ represent the drain, source, gate and bulk pins of $M1$, respectively.

Structure recognition is executed two times: Once with the library $L_{pwd}$ as given in Fig. 2.2 of section 2.2 and once with library $L_{analog}$ as given in Fig. 2.1 in the same section. The set of building blocks recognized with $L_{pwd}$ is denoted as $\mathcal{B}_{pwd}$. Similarly, the set of building blocks recognized with $L_{analog}$ is denoted as $\mathcal{B}_{analog}$.

For analog structure recognition, the original netlist $(N, D, \Gamma)$ has to be transformed to a power-up representation $(N_{up}, D_{up}, \Gamma_{up})$ of the circuit. This is done by tracing the power-down signal from the ports that control the power mode to the power-down switches. The nets that are connected to one of the power-mode controlling ports are given in set $N_{ports,pwd}$. The signal has to be traced forward over signal processing building blocks like e.g. ESD protection structures, level shifters, inverters and nand or nor gates starting from the nets in $N_{ports,pwd}$. As first step, the signal processing structures are detected by performing structure recognition with $L_{pwd}$ on the original netlist $(N, D, \Gamma)$. A power-down signal tracing graph can be instantiated according to Fig. 3.13 using the result $\mathcal{B}_{pwd}$. Next, a power-down signal tracing algorithm (Algorithm 8) traces the power-down signal path in a depth-first-search manner (see section 3.3.3) and identifies the power-down signal part $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$ of the circuit and a boundary part $(N_{bnd}, D_{bnd}, \Gamma_{bnd})$.

*3. Power-Down Verification*

The power-down signal part contains all devices along which the power mode controlling signals are transferred forward. The boundary part corresponds to the power-down switches and the nets connected to them. It is the interface between the digital power-mode controlling signals and the actual analog circuit parts. For computing the voltage of each net in $N_{pwd}$, voltage propagation is carried out on a circuit that consists of the power-down signal part and the boundary part, i.e., the propagation is carried out on the netlist $(N_{pwd} \cup N_{bnd}, D_{pwd} \cup D_{bnd}, \Gamma_{prop})$. The input voltage vector that enables power-up mode needs to be given for this step. This input vector can be described by a function $v_{init,up}(n_i)$. Similarly, the result of this voltage propagation run can be described by a function which is denoted as $v_{result,up}(n_i)$. Based on $v_{result,up}(n_i)$ and the identified power-down and boundary circuit parts, a power-up transformed circuit $(N_{up}, D_{up}, \Gamma_{up})$ is computed (see Algorithm 10). During this step, power-down switches are replaced by short circuits or open circuits depending on their gate voltage. Then, structure recognition with $L_{analog}$ is executed on the power-up transformed netlist to identify the aging critical structures.

The voltages in power-down mode on the detected structures depend on the input voltages at the ports of the circuit. All possible voltage combinations on the ports (or all combinations of a specified subset) are checked. Therefore, $m$ independent voltage propagation runs are performed on the original netlist $(N, D, \Gamma)$ for given voltages at the input ports described by functions $v_{init,down,j}(n_i)$ with $j = 1, 2, \ldots, m$. The corresponding result voltages are denoted as $v_{result,down,j}(n_i)$. As last step, the detected critical structures in $\mathcal{B}_{analog}$ are checked for asymmetric voltages using matching rules $\mathcal{R}$ (see section 3.3.5). In the end, a verification report is generated. An example of a verification report is given in Table 3.10. In the remainder of this section, the mentioned functions to describe the input and output voltages for voltage propagation are defined.

Voltage propagation requires input voltages given in the form of the sets $N_{gnd}, N_{vdd_1}, \ldots, N_{vdd_{n_{sup}}}$ as described in section 3.2.3 (see Algorithm 4). Those initialization voltages can be expressed in terms of the functions $v_{init,up}$ and $v_{init,down,j}$ that map nets to a voltage level out of $V$ as defined in equation (3.7). Therefore, the following subsets of the set of nets can be defined:

$$N_{ports,pwd} \subseteq N \tag{3.30}$$

$$N_{ports,in/out} \subseteq N \tag{3.31}$$

$$N_{supply} \subseteq N \tag{3.32}$$

The nets that are connected to ports controlling the power-down mode are in $N_{ports,pwd}$. The nets that are connected to other input or output ports are in $N_{ports,in/out}$. The supply rail nets (including ground) are in $N_{supply}$.

The following equivalences hold:

$$v_{init,x}(n_i) = gnd \iff n_i \in N_{gnd} \tag{3.33}$$
$$v_{init,x}(n_i) = vdd_i \iff n_i \in N_{vdd_i} \tag{3.34}$$

Thereby, "$x$" is short for either "$up$" or "$down, j$".

The functions $v_{init,up}$ and $v_{init,down,j}$ are defined in the following. The $j$-th combination of power-down voltages at the ports $N_{ports,in/out}$ for which asymmetric voltage conditions are checked are given by the function $v_{init,down,j}(n_i)$. It is defined as follows:

$$v_{init,down,j}: \quad N_{ports,pwd} \cup N_{ports,in/out} \cup N_{supply} \to V \quad n_i \mapsto v_{init,down,j}(n_i) \tag{3.35}$$

The initial voltages are given for the power-mode controlling ports, the general input and output ports and the supplies.

The voltages at the power mode controlling ports that represent the power-up mode need to be given for power-up transformation. They are described by the function $v_{init,up}$ which is defined as follows:

$$v_{init,up}: \quad N_{ports,pwd} \cup N_{supply} \to V \quad n_i \mapsto v_{init,up}(n_i) \tag{3.36}$$

Only the power-mode controlling ports in $N_{ports,pwd}$ and the supplies in $N_{supply}$ have initialization voltages. No initial voltages for $N_{ports,in/out}$ need to given because those voltages are not influencing the gate voltages of the power-down switches.

Furthermore, the results of the voltage propagation runs are described by functions $v_{result,down,j}$ and $v_{result,up}$ as visualized in Fig 3.12. The result of the $j$-th voltage propagation for the input voltages $v_{init,down,j}$ is given as follows:

$$v_{result,down,j}: \quad N \to V \quad n_i \mapsto v_{result,down,j}(n_i) \tag{3.37}$$

Voltage propagation computes a voltage for all nets in the original circuit $(N, D, \Gamma)$. Consequently, $v_{result,j}$ is defined for all nets in $N$.

Analogously, the result of the voltage propagation for the input voltages $v_{init,up}$ is described by the following function:

$$v_{result,up}: \quad N_{pwd} \cup N_{bnd} \to V \quad n_i \mapsto v_{result,up}(n_i) \tag{3.38}$$

As shown by Fig. 3.12, the set of nets of the boundary part $N_{bnd}$ is computed during the power-down signal tracing step (Algorithm 8). The function $v_{result,up}(n_i)$ defines the result of voltage propagation on the computed power-down signal propagation circuit $(N_{pwd} \cup N_{bnd}, D_{pwd} \cup D_{bnd}, \Gamma_{prop})$. Consequently, the result only comprises voltage values for the nets in $N_{pwd}$ and $N_{bnd}$.

The definitions made in this section are used in the following to describe the individual steps of the algorithm in further detail.

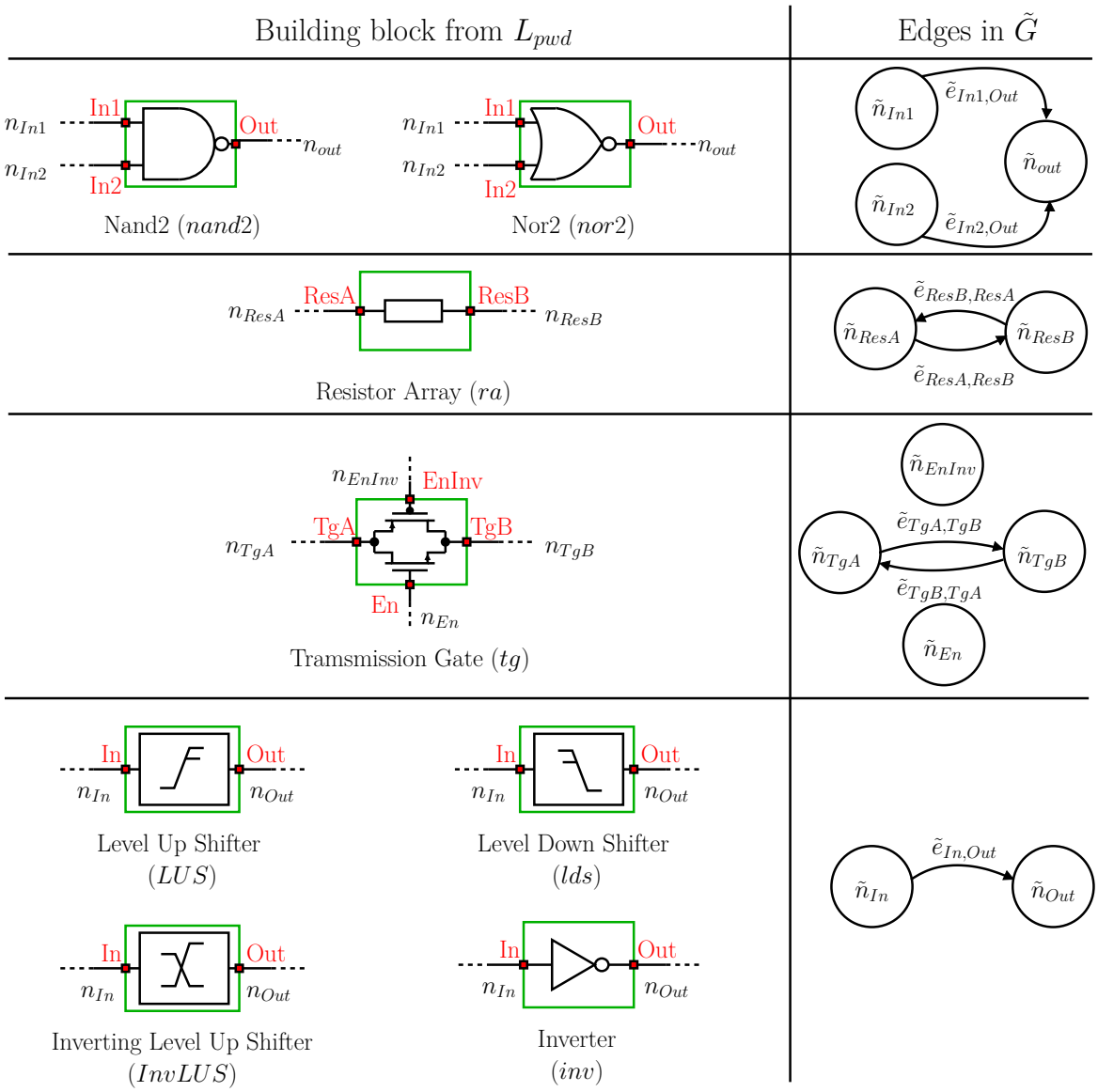



Figure 3.13.: Mapping of building blocks from $L_{pwd}$ to edges in the signal trace graph $\tilde{G} = (\tilde{N}, \tilde{E}, \Phi_{out}, \Phi_{in})$

### 3.3.3. Power-Down Signal Tracing

For tracing the power-mode controlling signals, the circuit is transformed to a graph based on the recognized building blocks $\mathcal{B}_{pwd}$. The signal tracing graph is defined as follows:

$$\tilde{G} \quad \Longleftrightarrow \quad (\tilde{N}, \tilde{E}, \Phi_{out}, \Phi_{in}) \tag{3.39}$$

Each node $\tilde{n}_i \in \tilde{N}$ of the graph corresponds to a net $n_i \in N$. Each edge $\tilde{e}_i \in \tilde{E}$ in the graph corresponds to a signal flow through a building block as shown in Fig. 3.13. The incidence functions of the graph are defined as follows:

$$\Phi_{out}(\tilde{e}_i) = \tilde{n}_i \iff \text{“}\tilde{e}_i \text{ is outgoing edge of } \tilde{n}_i\text{”} \iff \text{“}\tilde{e}_i \text{ starts at } \tilde{n}_i\text{”} \tag{3.40}$$

$$\Phi_{in}(\tilde{e}_i) = \tilde{n}_i \iff \text{“}\tilde{e}_i \text{ is incoming edge of } \tilde{n}_i\text{'}\text{”} \iff \text{“}\tilde{e}_i \text{ ends at } \tilde{n}_i\text{”} \tag{3.41}$$

Each edge and node can be marked as visited. The corresponding variables are denoted as

$$\tilde{e}_i.visited \in \{0, 1\} \tag{3.42}$$

and

$$\tilde{n}_i.visited \in \{0, 1\}. \tag{3.43}$$

A function which maps each edge $\tilde{e}_i$ to the devices from which the edge is originating can be defined as follows:

$$devices : \tilde{E} \to \mathscr{P}(D) \quad \tilde{e}_i \mapsto devices(\tilde{e}_i) = \{d_1, d_2, \ldots, d_n\} \tag{3.44}$$

Here, $\mathscr{P}(D)$ denotes the power set of $D$. The function returns all devices that belong to the building block $b_i \in \mathcal{B}_{pwd}$ which caused the creation of the edge (see Fig. 3.13). The function $devices(\tilde{e}_i)$ is used in line 23 of Algorithm 8 in order to determine the devices $D_{pwd}$ that belong to the power-down signal part of the circuit.

The power-down signal tracing is described with Algorithm 8 and illustrated in Fig. 3.14 for the example of the differential amplifier introduced in Fig 3.10. It should be noted that the off-connected transistors $M7$, $M8$, $M9$ and $M10$ are originating from ESD protection structures as defined in Fig. 3.11. Power-down signal tracing requires the power-mode controlling inputs $N_{ports,pwd}$, the structure recognition result $\mathcal{B}_{pwd}$ and the original netlist $(N, D, \Gamma)$ as input. The result of structure recognition is given in the box on the top of Fig. 3.14. The recognized building blocks are denoted as $b_1$ to $b_7$. Based on the building blocks, the graph is built. First, the graph is initialized empty (Algorithm 8, lines 1 to 3). Second, the signal trace graph is built, i.e., a node $\tilde{n}_i$ is added for each

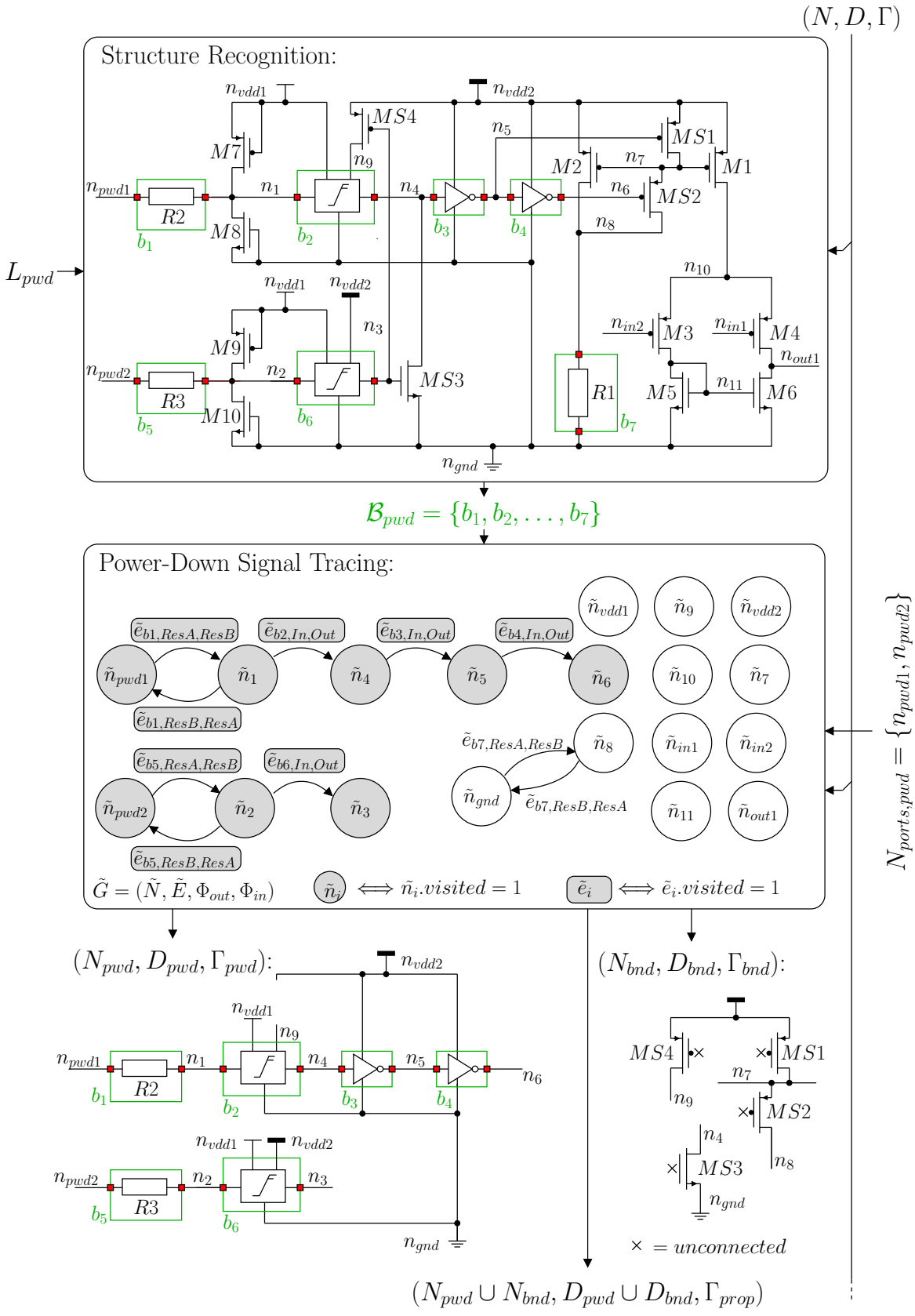



Figure 3.14.: Example (continued on page 73)

**Algorithm 8** Power-Down Signal Tracing

**Require:** $N_{ports,pwd}$, $\mathcal{B}_{pwd}$, $(N, D, \Gamma)$

// initialize signal tracing graph:
$\tilde{N} := \{\}$
$\tilde{E} := \{\}$

// build signal tracing graph:
**for all** $n_i \in N$ **do**
  Add node $\tilde{n}_i$ node to $\tilde{N}$.
**end for**
**for all** $b_i \in \mathcal{B}_{pwd}$ **do**
  Add edges to $\tilde{E}$, $\Phi_{out}$, $\Phi_{in}$ according to Fig. 3.13.
**end for**

// initialize nodes and edges as not visited:
**for all** $\tilde{n}_i \in \tilde{N}$ **do** $\tilde{n}_i.visited := 0$
**for all** $\tilde{e}_i \in \tilde{E}$ **do** $\tilde{e}_i.visited := 0$

// recursively trace signals:
**for all** $n_{port,pwd,i} \in N_{ports,pwd}$ **do**
  DepthFirstSearch($\tilde{n}_{port,pwd,i}$, $\tilde{G}$) // Algorithm 9
**end for**

// determine power-down signal part of the circuit:
$D_{pwd} := \left\{ d \in D \mid \exists_{\tilde{e}_i \in \tilde{E}} \left( \tilde{e}_i.visited = 1 \wedge \text{“}d \text{ is in set } devices(\tilde{e}_i)\text{”} \right) \right\}$ // see (3.44)
$N_{pwd} := \left\{ n_i \in N \mid \text{“a device in } D_{pwd} \text{ is connected to } n_i\text{”} \right\}$

// determine boundary part of the circuit:
$D_{bnd} := \left\{ d \in D \setminus D_{pwd} \mid \text{“}d \text{ is MOSFET”} \wedge \text{“gate of } d \text{ is connected to net in } N_{pwd}\text{”} \right\}$
$N_{bnd} := \left\{ n_i \in N \mid \text{“}n_i \text{ is connected to drain, source or bulk of a MOSFET in } D_{bnd}\text{”} \right\}$
Reduce $\Gamma$ to $\Gamma_{pwd}$, $\Gamma_{bnd}$ and $\Gamma_{prop}$. Do not change connectivity.

**return** $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$, $(N_{bnd}, D_{bnd}, \Gamma_{bnd})$, $(N_{pwd} \cup N_{bnd}, D_{pwd} \cup D_{bnd}, \Gamma_{prop})$

**Algorithm 9** DepthFirstSearch

**Require:** $\tilde{n}_{current}$, $\tilde{G}$
1: $\tilde{n}_{current}.visited := 1$
2: **for all** $\tilde{e}_{outgoing} \in \{\tilde{e} \in \tilde{E} \mid \Phi_{out}(\tilde{e}) = \tilde{n}_{current}\}$ **do**
3: $\quad \tilde{e}_{outgoing}.visited := 1$
4: $\quad n_{next} := \Phi_{in}(\tilde{e}_{outgoing})$
5: $\quad$ **if** $n_{next}.visited = 0$ **then**
6: $\quad\quad$ DepthFirstSearch($n_{next}$, $\tilde{G}$) `// recursion`
7: $\quad$ **end if**
8: **end for**

net $n_i \in N$ of the circuit and edges are added for each building block in $\mathcal{B}_{pwd}$ according to Fig. 3.13 (lines 5 to 11). The resulting signal tracing graph $\tilde{G}$ for the example circuit is shown in Fig. 3.14 in the middle box.

The library elements of $L_{pwd}$ and the corresponding signal flows defined in Fig 3.13 are typical examples for signal transferring elements on the power-down signal path. The elements in $L_{pwd}$ and the signal flows from Fig 3.13 are sufficient to analyze the industrial analog example blocks presented later in the experimental results (section 3.3.7). They should be regarded as examples to demonstrate the effectiveness of the method. In future work, the library can be extended to include new elements and/or the structure recognition algorithm can be replaced by other structure recognition methods as required.

As third step, all nodes and edges are initialized as not visited (lines 13 to 15). Fourth, the graph $\tilde{G}$ is traversed in a depth-first-search manner starting from all nets in $N_{ports,pwd}$ (lines 17 to 20). For the example in Fig. 3.14, the starting nodes are $\tilde{n}_{pwd1}$ and $\tilde{n}_{pwd2}$. Starting from those nodes, the graph is traversed by calling the recursive function DepthFirstSearch(). The function is given in Algorithm 9. It requires the currently visited node $\tilde{n}_{current}$ and the signal tracing graph $\tilde{G}$. For the top level calls of the recursion, the currently visited node is a power-mode controlling port $\tilde{n}_{ports,pwd,i}$. For the recursive calls in line 6 the next node $n_{next}$ is a node that is reachable from the current node over an outgoing edge, i.e. $n_{next} = \Phi_{in}(\tilde{e}_{outgoing})$. The graph is traversed forward along the edges in this way. The recursion only proceeds if the node on the other side was not visited yet (line 5) in order to stop the recursion for loops in the graph. Checking this condition is necessary, as the power-down signal tracing graph can contain loops even if the signal is propagating only in one direction. An example can be seen the middle box of Fig. 3.14: Loops are caused by the anti-parallel edges of $R1$ and $R2$. Each visited node and edge is marked as visited (lines 1 and 3). In Fig. 3.14, the visited nodes and edges after termination of the depth-first search are marked with gray background color for the nodes and gray boxes for edges.

The power-down signal part $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$ and the boundary part $(N_{bnd}, D_{bnd}, \Gamma_{bnd})$ of the circuit can be determined after depth-first search. For the example circuit, the

**Algorithm 10** Power-Up Circuit Computation

**Require:** $v_{result,up}(n_i)$, $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$, $(N_{bnd}, D_{bnd}, \Gamma_{bnd})$, $(N, D, \Gamma)$

1: `// throw away the power-down signal part:`
2: $D_{up} := D \setminus D_{pwd}$
3: $N_{up} := \{n_i \in N \mid$ "A device in $D_{up}$ is connected to $n_i\}$
4: Reduce $\Gamma$ to $\Gamma_{up}$. Do not change connectivity.
5:
6: `// Transform the transistors on the boundary (power-down switches):`
7: `// Replace switches by a short circuit or an open circuit.`
8: `// A short circuit between` $n_i$ `and` $n_j$ `creates new merged net` $n_{i,j}$`.`
9: `// An open circuit is equivalent to removal of the device.`
10: **for all** $d_{bnd} \in D_{bnd}$ **do**
11: $n_{gate} :=$ "net connected to the gate of $d_{bnd}$"
12: **if** $(techtype(d_{bnd}) = n \wedge isVdd(v_{result,up}(n_{gate})))$
$\vee(techtype(d_{bnd}) = p \wedge v_{result,up}(n_{gate}) = gnd)$ **then**
13: Replace $d_{bnd}$ by a short circuit between drain and source in $(N_{up}, D_{up}, \Gamma_{up})$.
14: **else if** $(techtype(d_{bnd}) = n \wedge v_{result,up}(n_{gate}) = gnd)$
$\vee(techtype(d_{bnd}) = p \wedge isVdd(v_{result,up}(n_{gate})))$ **then**
15: Remove $d_{bnd}$ from $(N_{up}, D_{up}, \Gamma_{up})$.
16: **end if**
17: **end for**
18:
19: `// Remove dangling nets from` $N_{up}$
20: $N_{up} := N_{up} \setminus \{n_i \in N_{up} \mid$ "No device is connected to $n_i$"$\}$
21: **return** $(N_{up}, D_{up}, \Gamma_{up})$ `// power-up representation`

resulting netlists are shown on the bottom of Fig. 3.14. The power-down signal part $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$ corresponds to all devices that were visited during depth-first search, i.e. all devices which are involved in transferring the power-down signal forward (see line 23). All nets connected to any device in $D_{pwd}$ are also part of the power-down part $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$ of the circuit (see line 24). The boundary part can be determined by finding all MOSFET transistors that are not part of the power-down part $D_{pwd}$ and which have a gate connected to any net in $N_{pwd}$ (see line 27). Those transistors translate the power-down signal to a conducting or non-conducting connection between drain and source, i.e. they configure the circuit for power-down or normal operation mode. Experimental results have shown that all power-down switches become part of $D_{bnd}$ by using this heuristic. The nets of the boundary parts are all nets connected to either bulk, drain or source of the transistors in $D_{bnd}$ (see line 28). The connectivity of the original circuit $\Gamma$ has to be reduced according to the reduced sets of nets $N_{pwd}$, $N_{bnd}$ and $N_{pwd} \cup N_{bnd}$ (line 29). The corresponding connectivities for the power-down, boundary and propagation circuit are denoted as $\Gamma_{pwd}$, $\Gamma_{bnd}$ and $\Gamma_{prop}$, respectively. The algorithm finally returns the three netlists $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$ and $(N_{bnd}, D_{bnd}, \Gamma_{bnd})$ and $(N_{pwd} \cup N_{bnd}, D_{pwd} \cup D_{bnd}, \Gamma_{prop})$ (line 31). They are used for power-up circuit computation (see Fig. 3.12).

### 3.3.4. Power-Up Circuit Computation

Power-up circuit computation is described by Algorithm 10. As stated in the algorithm and shown by the overview in Fig. 3.12, it requires the result of voltage propagation on the propagation circuit $v_{result,up}(n_i)$, the power-down circuit part $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$, the boundary part $(N_{bnd}, D_{bnd}, \Gamma_{bnd})$ and the original netlist $(N, D, \Gamma)$. The voltages of the power-down signal circuit $v_{result,up}(n_i)$ are computed by voltage propagation on the propagation circuit $(N_{pwd} \cup N_{bnd}, D_{pwd} \cup D_{bnd}, \Gamma_{prop})$ as shown in the overview in Fig. 3.12. In Fig. 3.15 the propagation circuit as well as the computed voltages are shown in the box on the top. The initialization voltages $v_{init,up}(n_i)$ that were used for the computation are shown in the table on the right. Based on the partitioning of the circuit in power-down signal part $(N_{pwd}, D_{pwd}, \Gamma_{pwd})$, boundary part $(N_{bnd}, D_{bnd}, \Gamma_{bnd})$ and rest of the circuit and the computed voltages, the power-up representation of the circuit $(N_{up}, D_{up}, \Gamma_{up})$ is computed. The resulting power-up representation of the circuit is given in Fig. 3.12 in the box at the bottom. It is computed by Algorithm 10 as follows: First, all devices belonging to the power-down signal part are removed from the original netlist (line 2). Second, only nets that are connected to any of the remaining devices are copied to the new netlist (line 3). In lines 6 to 17, the transistors in the boundary part are either replaced by a short circuit or open circuit according to the voltage at the gate.

Conducting transistors (NMOS with a gate voltage of $vdd_i$ with any $i$ or PMOS with gate $gnd$) are replaced by short circuits. Non conducting transistors (PMOS with gate voltage $vdd_i$ with any $i$ or NMOS with gate $gnd$) are replaced by open circuits. An open

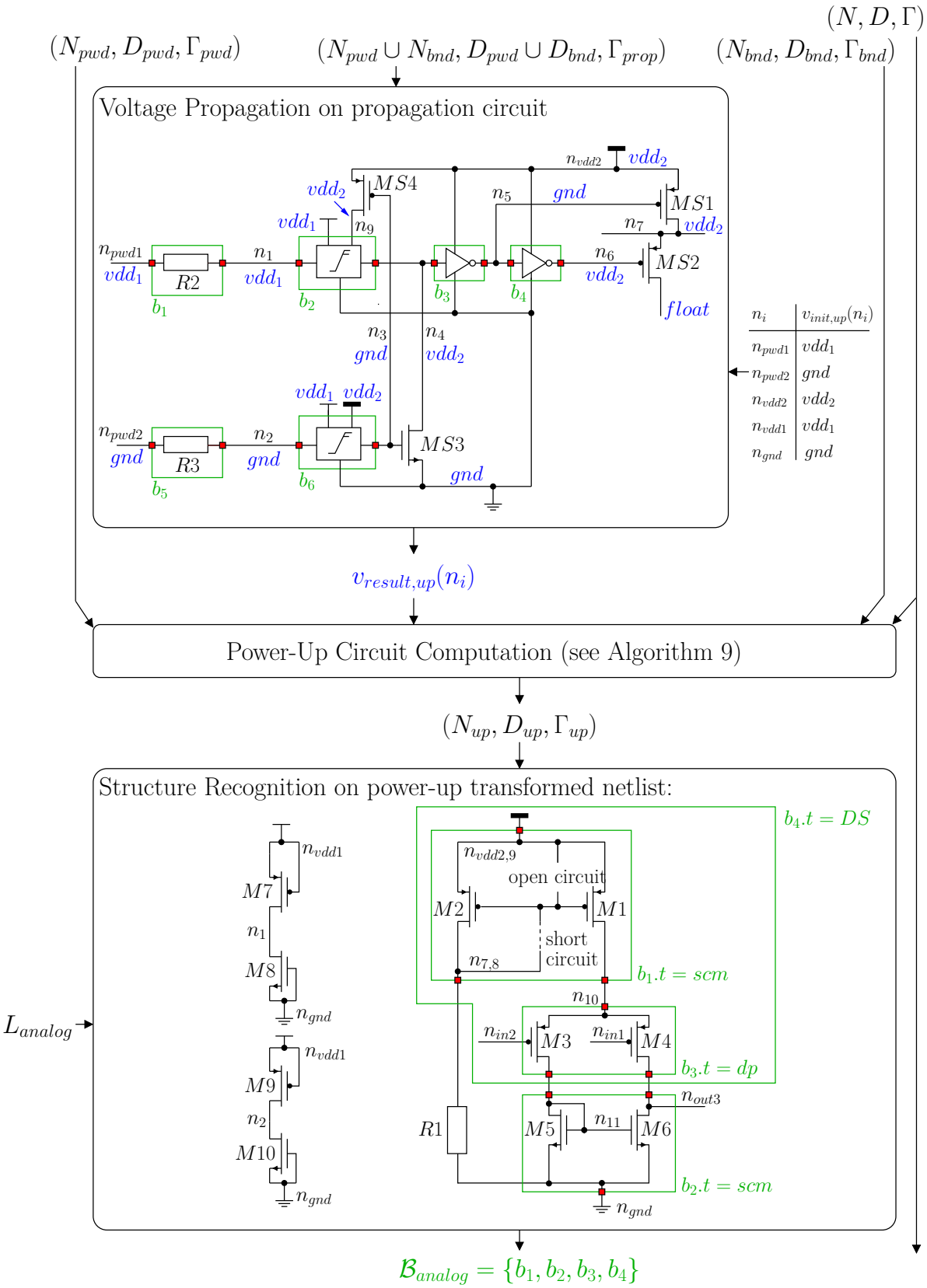



Figure 3.15.: Example (continued from page 68 / to be continued on page 83)

circuit is equivalent to deleting the device from the netlist. A short circuit is equivalent to merging two nets to a new net. The following convention is used: A net that is resulting from merging net $n_i$ and $n_j$ is denoted as net $n_{i,j}$. Merging can, for example, be implemented by connecting all pins that were connected to $n_i$ or $n_j$ to the new net $n_{i,j}$ and subsequent deleting of $n_i$ and $n_j$.

There is the possibility of dangling nets after removal of non conducting transistors, i.e., there can be nets to which no device is connected. Those nets are removed in line 20. Finally, the power-up transformed netlist $(N_{up}, D_{up}, \Gamma_{up})$ is returned in line 21. It only contains the parts of the circuit that implement the analog core functionality of the circuit. Furthermore, the connectivity of the basic building blocks is not obscured by power-down switches so that the rules of library $L_{analog}$ can be used to detect analog building blocks. Consider the current mirror in $(M1, M2)$ in the example circuit. In the original circuit as shown in the box on the top of Fig. 3.14, it would not be detected by the rule for the simple current mirror ($scm$) in $L_{analog}$ because the diode connection (connection between drain and gate) of $M1$ is broken up by power-down switch $MS2$. However, in the power-up representation as shown in the bottom box of Fig. 3.15, the simple current mirror is detected as building block $b1$.

Another example that illustrates the need for power-up circuit computation is shown in Fig. 3.16. The transformation is necesarry as switches that are conducting in power-up mode are blocking the recognition of building blocks. Fig. 3.16 shows that $MS1$ is conducting and $MS2$ is not conducting. The shown cascade current mirror ($CCM$) can only be detected as building block if power-down switch $MS1$ is replaced by a short circuit, i.e. if net $n_1$ is merged with net $n_2$. Furthermore, it is beneficial to remove the non conducting switches (in the example: $MS2$) as the removed switches can no longer be part of any falsely recognized building blocks. The quality of result can be improved in this way because the information that the transistors in $D_{bnd}$ are power-down switches is used.

The identified analog structures are potentially critical in terms of power-down voltage matching. Four analog building blocks $b_1$ to $b_4$ are detected in the example of Fig. 3.15. This information can be used for discovering unmatched power-down voltages at matched structures like, e.g., the detected differential pair ($M3$, $M4$) or the detected current mirrors ($M1$, $M2$) and ($M5$, $M6$). Finding those conditions dependent on the input voltages on the ports is subject of the next two sections. First, power-down voltage matching rules suitable for automatic checking are formulated (section 3.3.5). Then, the procedure of checking the defined rules is described (section 3.3.6).

### 3.3.5. Voltage Matching Rules

In this section, a framework for specifying voltage matching rules for power-down mode and for automatic checking of those rules is described. Specifying matching rules means specifying voltage constraints for the pins of devices or building blocks. For matched

| Strictness | Rules | Reason |
|---|---|---|
| loose | Match gate.<br>Match bulk. | Matched electric field across gate oxide, i.e., matched BTI.<br>Possible asymmetry: drain/source region |
| moderate | PMOS: gate $vdd_i$<br>NMOS: gate $gnd$<br>Match bulk. | No or reversed electric field across gate oxide, i.e., no BTI or "healing" condition.<br>Possible asymmetry: drain/source region |
| strict | Match gate.<br>Match bulk.<br>Match drain.<br>Match source. | Exactly matched electric field across gate oxide, i.e, exactly matched BTI |
| very strict | PMOS: gate $vdd_i$<br>NMOS: gate $gnd$<br>Match bulk.<br>Match drain.<br>Match source. | No or reversed electric field across gate oxide, i.e., no BTI or "healing" condition.<br>Fully symmetric also in drain/source region. |
| paranoid | All pins same voltage. | No field. No stress. |

Table 3.7.: Systematic of aging rules

devices, the framework should enable to specify that a group of device pins should have the same voltage in power-down mode. As a more stringent requirement, it should also be possible to specify a rule to force a group of pins to a given voltage, e.g. $vdd_i$ with any $i$ or $gnd$. Using those two classes of rules, matching rules of different stringency can be formulated. In Table 3.7, the rules are given in natural language together with a reason for each rule set. In the following, a framework for automatic rule checking based on the rule sets given in Table 3.7 is described. For specifying the pins to be matched or to have a given voltage, the pins need to be addressed in dependency of the building blocks defined in a library like, e.g. $L_{analog}$. A format of specifying those rules is developed in this section.

Fig. 3.16 shows an example of the building block hierarchy within a cascode current mirror ($CCM$). Following the same conventions as in the chapter on structure recognition (chapter 2), the borders of the structures are marked by green boxes and the pins are drawn as red squares. The two transistors labeled $MS1$ and $MS2$ are power-down switches connected in a typical way to switch off the current in both, the input and output branch of the current mirror. The input branch on the left side consists of two diode arrays ($da$) each consisting of two parallel MOSFETs. The output branch on the right side consists of two normal arrays ($na$) each consisting of three parallel MOSFETs.

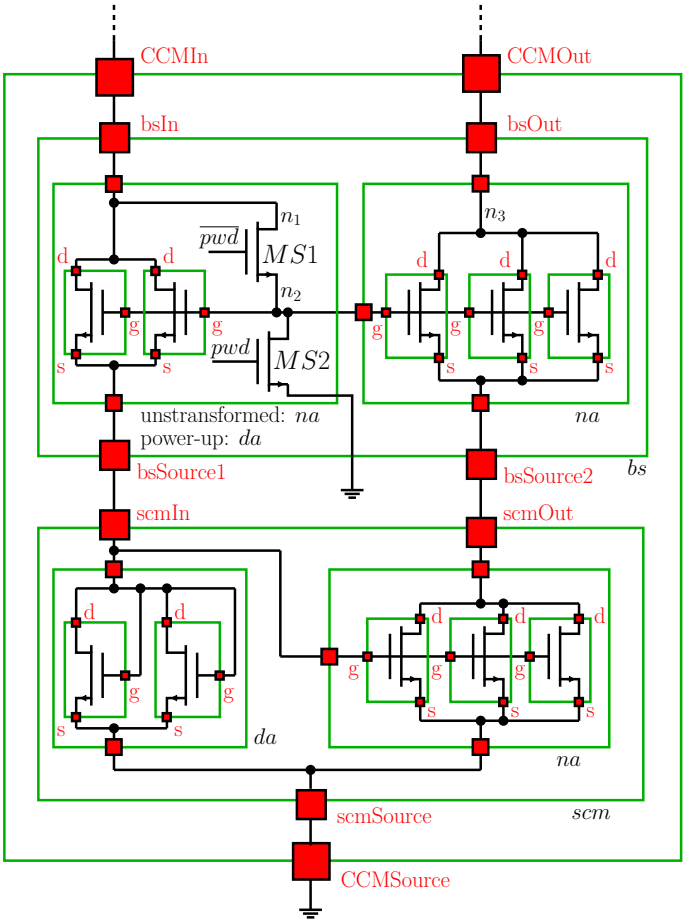



Figure 3.16.: Cascode current mirror ($CCM$) with power-down switches $MS1$ and $MS2$

Assuming that $MS1$ is a short circuit which is the case during power-up mode, the parallel transistors on the left form two diode arrays ($da$) with two children each. The upper diode array and normal array form a bias shifter ($bs$). The lower diode array and normal array form a simple current mirror ($scm$). The bias shifter on the top and the simple current mirror on the bottom finally form the cascode current mirror ($CCM$).

The power-down voltages at the pins of all transistors on the left should be matched to the voltages at the pins of all transistors on the right. As nets from the original netlist $(N, D, \Gamma)$ are merged during power-up circuit computation to new nets in the power-up transformed circuit $(N_{up}, D_{up}, \Gamma_{up})$, it is required to specify the rules in terms of device pins and not in terms of pins of building blocks of higher levels. For example, Fig. 3.16 shows that the switch $MS1$ is replaced by a short circuit, i.e., net $n_1$ is merged with net $n_2$ to a new net $n_{1,2}$. Therefore, the example rule "voltage of $CCMin$ should be matched to voltage of $CCMout$" translates to "voltage of net $n_{1,2}$ should be matched with voltage of net $n_3$" for the given example. However, as shown by the overview depicted in Fig 3.12, the voltage propagation for computation of $v_{result,j}$ is performed

on the original circuit. The result functions $v_{result,j}$ are only defined for nets from $N$ and not for the merged nets from $N_{up}$. Consequently, the rule specified above for $n_{1,2}$ and $n_3$ is not well defined, as $n_{1,2}$ can be mapped back to either $n_1$ or $n_2$. There is no bijective mapping between the original nets and the nets of the transformed circuit. In contrast, there is a bijective mapping between each device in $D_{up}$ and the corresponding device in $D$ as devices are only removed but never added during the power-up circuit computation. Consequently, rules can be formulated uniquely in terms of pins of the devices in $D_{up}$. Therefore, the individual devices of a given building block need to be addressed. Using those addresses, matching rules suitable for automatic checking can be formulated. This will be done in the following.

The library $L_{analog}$ as shown in Fig. 2.1 of section 2.2 consists of the following elements:

$$L_{analog} = \{na, da, vr1, vr2, cp, cml, dp, scm, bs, \\ IWCM, CCM, WCM, 4TCM, WSCCM\} \tag{3.45}$$

Each type $t \in L_{analog}$ is either of sort *device*, *pair* or *array* (see section 2.2). Furthermore, each type belongs to a level of the library. The level of $t$ is $level(t)$ as defined in (2.4). The level of each type in $L_{analog}$ is given in Fig. 2.1.

In order to formulate matching rules in terms of sets of *mos* pins, the set of *mos* pin names for drain ($d$), gate ($g$), source ($s$) and bulk ($b$) of a transistor is defined as follows:

$$P_{mos} = \{d, g, s, b\} \tag{3.46}$$

The pin names are annotated in red in Figs. 2.1 and 3.16.

In the following, a child address will be defined as first step. Using this child address, each child of a building block, i.e. also the devices on the lowest level, can be uniquely addressed. In order to specify voltage matching rules, individual transistor pins can be addressed using child addresses and the set $P_{mos}$.

A child address is composed of address tokens. The set of address tokens is given as follows:

$$A = \{1, 2, all\} \tag{3.47}$$

The address token *all* refers to all children of an array. For example, for the arrays *da* and *na* of $L_{analog}$, the token *all* refers to all *mos* transistors that form the *da* or *na*. The address token 1 refers to the left or upper child of a pair as drawn in Fig. 2.1. The address token 2 refers to the right or lower child of a pair. This is illustrated with the following two examples: The token 1 refers to the diode array (*da*) in a bias shifter (*bs*). The token 2 refers to the simple current mirror (*scm*) in a cascode current mirror ($CCM$).

The address tokens are combined to form a child address. The set of valid child addresses for a type $t$ is given as follows:

$$\begin{aligned} C_t &= \{\mathbf{c} \in C \mid valid_t(\mathbf{c}) = 1\} \\ \text{with } C &= \{()\} \cup A \cup A^2 \cup \cdots \cup A^{level(t)} \end{aligned} \tag{3.48}$$

The symbol () denotes an empty address. A child address $\mathbf{c}$ is a tuple of address tokens with a length between 0 and $level(t)$.

The meaning of a child address is illustrated by the example of the cascode current mirror ($CCM$) shown in Fig. 3.16. The set of valid component addresses for a $CCM$ is denoted as $C_{CCM}$. There are 11 valid component addresses for a $CCM$:

$$\begin{aligned} C_{CCM} = \{&(), \\ &(1), (2), \\ &(1,1), (1,2), (2,1), (2,2), \\ &(1,1,all), (1,2,all), (2,1,all), (2,2,all)\} \end{aligned} \tag{3.49}$$

The meaning is as follows: The empty address () refers to the $CCM$ itself. The address (1) refers to the bias shifter ($bs$) on the top. The address (2) refers to the simple current mirror ($scm$) on the bottom. The address $(1,2)$ refers to the normal array ($na$) on the top right. The address $(2,1)$ refers to the diode array ($da$) on the bottom left. The address $(1,2,all)$ refers to the three $mos$ transistors in the normal array ($na$) on the top right. The address $(2,1,all)$ refers to the two $mos$ transistors in the diode array ($da$) on the bottom left.

For each type $t$, the set $C_t$ is constrained to contain only child addresses which make sense for type $t$, by the function $valid_t(\mathbf{c})$. This is illustrated again with the example of the $CCM$. The type $CCM$ is of sort $pair$. Consequently, the first element of the tuple needs to be 1 or 2. A tuple starting with $all$ would not be valid. Both, the first and second child of a $CCM$, are also of sort $pair$. Consequently, the second element of the tuple also needs to be 1 or 2. A tuple with second element $all$ would not be valid. Furthermore, the third and last element needs to be $all$ as all the children of of the $scm$ and the $bs$ are of sort $array$. A tuple ending with 1 or 2 would not be valid.

The principle is written down formally in the following. The function $valid_t(\mathbf{c})$ returns true for all valid tuples $\mathbf{c}$ for a type $t$. It is defined recursively as follows:

$$valid_t(\mathbf{c}) := \begin{cases} 1 & \text{for} \quad \mathbf{c} \in \{()\} \\ lastValid_t(\mathbf{c}) \wedge valid_t(parent(\mathbf{c})) & \text{for} \quad \mathbf{c} \in C \setminus \{()\} \end{cases} \tag{3.50}$$

The the empty address () is always valid. The function $parent(\mathbf{c})$ returns the child address of the parent structure, i.e., the last element of $\mathbf{c}$ is removed. The function

| Building block type(s) $t$ | Matching rules $\mathcal{R}_t$ for type $t$ (all "strict") |
|---|---|
| $dp$, $scm$ | $\bigcup pin \in \{d, g, s, b\}$ $\Big\{\{(1, all, pin), (2, all, pin)\}\Big\}$ |
| $WSCCM$, $4TCM$ | $\bigcup pin \in \{d, g, s, b\}$ $\Big\{\{(1, 1, all, pin), (2, 1, all, pin)\}, \{(1, 2, all, pin), (2, 2, all, pin)\}\Big\}$ |
| $IWCM$, $CCM$ | $\bigcup pin \in \{d, g, s, b\}$ $\Big\{\{(1, 1, all, pin), (1, 2, all, pin)\}, \{(2, 1, all, pin), (2, 2, all, pin)\}\Big\}$ |
| $WCM$ | $\bigcup pin \in \{d, g, s, b\}$ $\Big\{\{(2, 1, all, pin), (2, 2, all, pin)\}\Big\}$ |

Table 3.8.: Voltage matching rules $\mathcal{R}$ for the case "strict" in Table 3.7 [Zwerger and Graeb, 2015]

$lastValid_t(\mathbf{c})$ is true if a pair is addressed by $parent(\mathbf{c})$ and the last token of $\mathbf{c}$ is 1 or 2. It is also true if an array is addressed by $parent(\mathbf{c})$ and the last token of $\mathbf{c}$ is $all$. Else it is false.

As the voltage matching rules are formulated in terms of *mos* transistor pins, we define a *mos* pin address for type $t$, by extending a valid child address of length $level(t)$ with a *mos* pin name. Therefore, the set of *mos* pin addresses for type $t$ is given as follows:

$$P_t = (C_t \cap A^{level(t)}) \times P_{mos} \tag{3.51}$$

Consequently, a *mos* pin address is a tuple of $level(t)$ address tokens and one *mos* pin name as last tuple element. Examples for pin addresses are given in Table 3.8 and in Table 3.9 where *pin* in Table 3.8 is one of $d$, $g$, $s$ or $b$.

In this thesis, three classes of matching rules are defined:

- *Matching rules* describe a set of pins that have to have an equal (and non $float$) voltage
- *Voltage rules for techtype n* describe that a specific pin should have a specific voltage for building blocks of techttype $n$
- *Voltage rules for techtype p* describe that a specific pin should have a specific voltage for building blocks of techttype $p$

The three classes of rules are defined in the following.

A matching rule $\mathcal{R}_{match,t,i}$ is a set of *mos* pin addresses, i.e., it is a subset of the power set of $P_t$:

$$\mathcal{R}_{match,t,i} \subseteq \mathscr{P}(P_t) \tag{3.52}$$

| Building block type(s) $t$ | Matching rules $\mathcal{R}_t$ for type $t$ and techtype $n$ ("strict" for $dp$ and "moderate" for current mirror types) |
|---|---|
| $dp$ | $\bigcup pin \in \{d, g, s, b\} \left\{\{(1,\ all,\ pin),\ (2,\ all,\ pin)\}\right\}$ |
| $scm$ | $\Big\{((1,\ all,\ g),\ gnd),$<br>$((2,\ all,\ g),\ gnd),$<br>$\{(1,\ all,\ b),\ (2,\ all,\ b)\}\Big\}$ |
| $WSCCM$, $4TCM$ | $\Big\{((1,\ 1,\ all,\ g),\ gnd),$<br>$((2,\ 1,\ all,\ g),\ gnd),$<br>$((1,\ 2,\ all,\ g),\ gnd),$<br>$((2,\ 2,\ all,\ g),\ gnd),$<br>$\{(1,\ 1,\ all,\ b),\ (2,\ 1,\ all,\ b)\},$<br>$\{(1,\ 2,\ all,\ b),\ (2,\ 2,\ all,\ b)\}\ \Big\}$ |
| $IWCM$, $CCM$ | $\Big\{((1,\ 1,\ all,\ g),\ gnd),$<br>$((1,\ 2,\ all,\ g),\ gnd),$<br>$((2,\ 1,\ all,\ g),\ gnd),$<br>$((2,\ 2,\ all,\ g),\ gnd),$<br>$\{(1,\ 1,\ all,\ b),\ (1,\ 2,\ all,\ b)\},$<br>$\{(2,\ 1,\ all,\ b),\ (2,\ 2,\ all,\ b)\}\ \Big\}$ |
| $WCM$ | $\Big\{((2,\ 1,\ all,\ g),\ gnd),$<br>$((2,\ 2,\ all,\ g),\ gnd),$<br>$\{(2,\ 1,\ all,\ b),\ (2,\ 2,\ all,\ b)\}\ \Big\}$ |

Table 3.9.: Mixed matching rules $\mathcal{R}$ using "strict" for $dp$ and "moderate" for all types of current mirrors (according to Table 3.7) written down for $techtype(b) = n$. One rule per line. For $techtype(b) = p$ the gate voltage $gnd$ is replaced by $vdd_i$ in each rule.

All pins in the set are required to have the same (and non $float$) voltage in power-down mode. If the voltage of one of the addressed pins is $float$, the matching rule is regarded as violated because $float$ does not guarantee a defined voltage and the actual voltages can vary quite drastically. However, for the special case that both pins are connected to the same net in the original circuit $(N, D, \Gamma)$ and the voltage of the net is $float$, the rule is not violated, as the voltages are matched due to the physical connection between the pins. Examples matching rules can be be seen in Table 3.8. The rule set in this table consists of matching rules only.

A $n/p$ voltage rule $\mathcal{R}_{voltage,n/p,t,i}$ is a tuple of a pin address and a voltage, i.e., the following relation holds:

$$\mathcal{R}_{voltage,n/p,t,i} \subseteq P_t \times \left( \{gnd\} \cup \bigcup_{i=1,2,\ldots,n} \{vdd_i\} \right) \tag{3.53}$$

A voltage rule specifies that the pin addressed with $P_t$ should have the given voltage. It applies only to building blocks of the given techtype which is either $n$ or $p$.

Examples of matching rules are given in Table 3.9. The rules for the current mirror types contain voltage rules for the gate ($g$) pins. In Table 3.9, the rules for current mirrors were written down for case of techtype $n$. For techtype $p$, the voltage $gnd$ is replaced by $vdd_i$.

The matching rules, $n$ voltage rules and $p$ voltage rules for a type $t$ are collected in the set $\mathcal{R}_t$, i.e., the set of all rules for type $t$ is defined as follows:

$$\mathcal{R}_t = \bigcup_{i=1,2,\ldots,n_v} \mathcal{R}_{match,t,i} \cup \bigcup_{i=1,2,\ldots,m_n} \mathcal{R}_{voltage,n,t,i} \cup \bigcup_{i=1,2,\ldots,m_p} \mathcal{R}_{voltage,p,t,i} \tag{3.54}$$

Here, $n_v$ is the total number of matching rules and $m_n$ and $m_p$ are the total numbers of $n$ and $p$ voltage rules, respectively. The complete set of rules is $\mathcal{R}$. It is defined as follows:

$$\mathcal{R} = \bigcup_{t \in L_{analog}} \mathcal{R}_t \tag{3.55}$$

The matching rules $\mathcal{R}$ have to be formulated using a combination matching and voltage rules that is adequate for the aging behavior of the used technology and the reliability requirements of the application.

The rules in Table 3.7 that are formulated in natural language can be expressed using the rules described above. As an example, the matching rules for $L_{analog}$ that correspond to the strict case in Table 3.7 are written down formally in Table 3.8. The rule set in Table 3.8 only contains matching rules as defined in equation (3.52). The rule sets can also be a combination of rules of different strictness. For example, Table 3.9 shows a rule set where the "strict" rules are applied to differential pairs ($dp$) and "moderate" rules

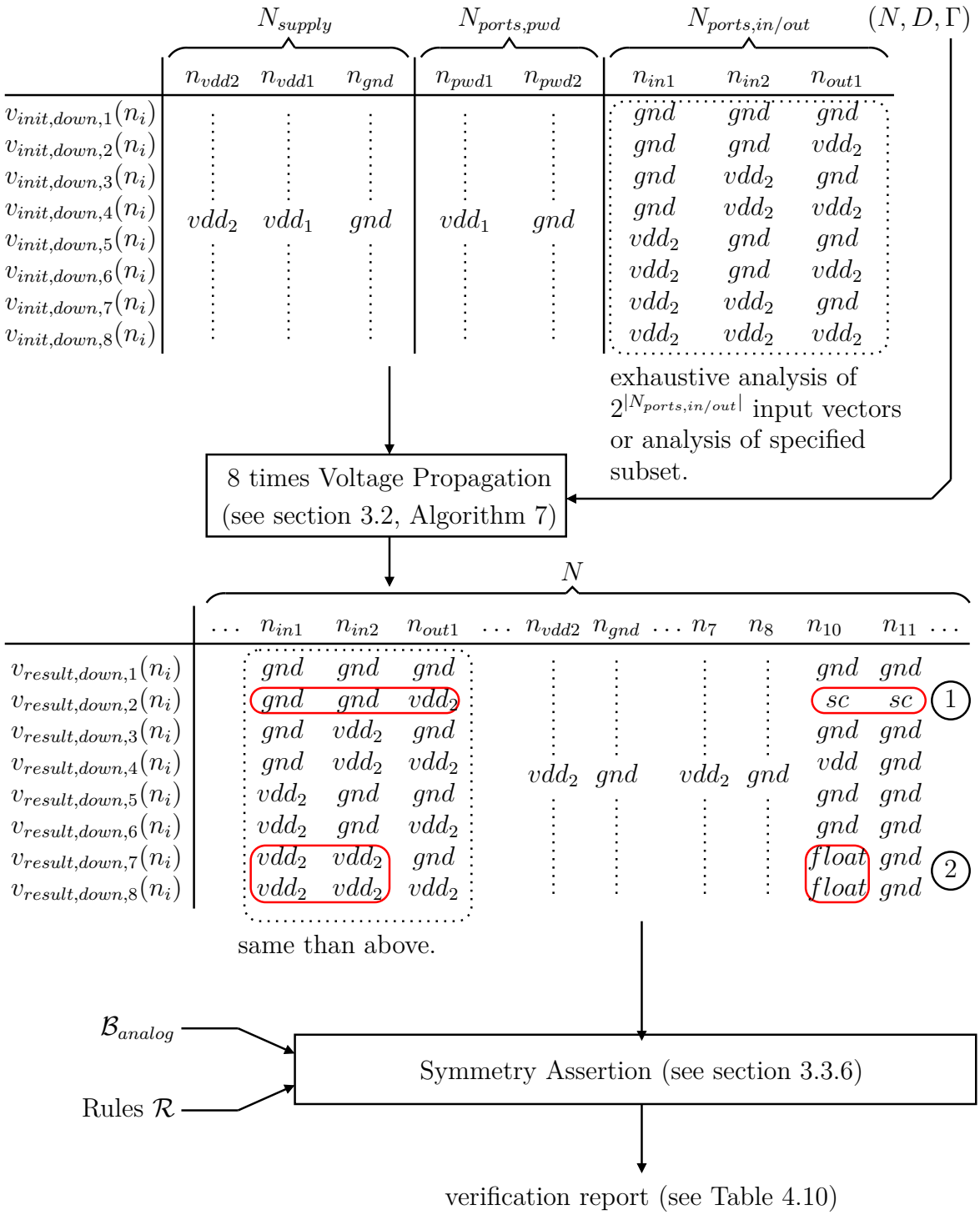
are applied to all current mirrors. The rule set in this table contains matching rules as defined in equation (3.52) as well as voltage rules as defined in equation (3.53). A rule set is required for symmetry assertion which is described in the next section.

### 3.3.6. Symmetry Assertion

Fig. 3.12 shows that symmetry assertion requires the voltage propagation results $v_{result,down,j}$ that are obtained from voltage propagation on the original netlist $(N, D, \Gamma)$ with input vector $v_{init,down,j}$, the structure recognition result $\mathcal{B}_{analog}$ and a rule set $\mathcal{R}$ as defined in the previous section. Fig 3.17 shows the input vectors $v_{init,down,j}$ as well the voltage propagation results $v_{result,down,j}$ for the example circuit. The eight input vectors with $j = 1, 2, \ldots, 8$ are shown in the table on the top of the figure. The supplies in $N_{supply}$ are kept constant at their respective values for all $j$. Similarly, the power-mode controlling ports $N_{ports,pwd}$ are kept constant at the combination that powers the circuit down. The initial values of the other three ports in $N_{ports,in/out}$ are varied. An exhaustive analysis of all $2^3 = 8$ combinations of input vectors is done in the example. If the conditions on the external ports are part of the circuit specification, only the specified subset of combinations can be checked.

Voltage propagation is run independently for each input vector. This means that this part of the analysis is open for parallel computing. The implementation used to generate the experimental results described in section 3.3.7 only uses one thread, i.e., the possibility for parallel computing is not yet exploited. This is possible future work.

An excerpt of the result of voltage propagation is given in the table in the middle of Fig 3.17. The initial values of the ports $n_{in1}$, $n_{in2}$ and $n_{out1}$ can also be found in the result. Furthermore, the supply nets $n_{vdd2}$ and $n_{gnd}$ are constant at their respective values. The original circuit shown in the box on the top of Fig 3.14 shows that net $n_7$ is pulled to $vdd_2$ by power-down switch $MS1$. This is reflected in the voltage propagation results by a constant voltage value of $vdd_2$ over all $j$ for net $n_7$. Similarly, net $n_8$ is connected to $n_{gnd}$ by the resistor $R1$. Therefore, net $n_8$ has a constant voltage value of $gnd$ for all $j$. The nets $n_{10}$ and $n_{11}$ have result voltages that vary with $j$, i.e., they vary with the values of $n_{in1}$, $n_{in2}$ and $n_{out1}$. The original circuit in Fig 3.14 shows that net $n_{11}$ is connected to $n_{gnd}$ by the diode connected transistor $M5$. Therefore, it has a voltage value of $gnd$ for all runs except the run with index $j = 2$ which corresponds to the voltages $gnd$, $gnd$, $vdd_2$ for the ports $n_{in1}$, $n_{in2}$ and $n_{out1}$. This case is marked with a circled 1. If $n_{in1}$ and $n_{in2}$ have a voltage of $gnd$, both PMOS transistors of the differential pair $(M3, M4)$ are conducting. If the output of the amplifier $n_{out1}$ is additionally pulled to $vdd_2$ by external circuitry, a definite short-circuit in power-down mode from $n_{out1}$ over $M4$, $n_{10}$, $M3$, $n_{11}$, $M5$ to $n_{gnd}$ is present in the circuit. This design error is detected. Furthermore, the results show that net $n_{10}$ is floating, if both, $M3$ and $M4$ are not conducting, i.e., if both, the voltage of $n_{in1}$ and $n_{in2}$ is $vdd_2$. This case is marked with a circled 2.




Notes:

(1) Definite short-circuit path from $n_{out1}$ over $M4$, $n_{10}$, $M3$, $n_{11}$, $M5$ to $n_{gnd}$

(2) $n_{10}$ is $float$ if $M3$ and $M4$ are not conducting, i.e., if $n_{in1}$ and $n_{in2}$ are $vdd_2$

Figure 3.17.: Example (continued from page 73)

| Error | Index $j$ of input vector $v_{init,down,j}$ |
|---|---|
| Definitive short circuit | 2 |
| Differential pair $dp$ $(M3.d, M4.d)$ | $4, 6, 8$ |
| Differential pair $dp$ $(M3.g, M4.g)$ | $3, 4, 6, 8$ |
| Simple current mirror $scm$ $(M2.d, M1.d)$ | $4, 7, 8$ |
| Simple current mirror $scm$ $(M5.d, M6.d)$ | $4, 6, 8$ |

Table 3.10.: Verification report [Zwerger and Graeb, 2015]

The symmetry rules $\mathcal{R}$ are checked for each detected building block in $\mathcal{B}_{analog}$ based on the result voltages from voltage propagation. A violation report for checking the strict rules given in Table 3.8 for the result voltages shown in Fig 3.17 is given in Table 3.10. In the first line, the detected short circuit for index $j = 2$ is reported. The next two lines report violations for the differential pair ($dp$) transistors $M3$ and $M4$: The drain voltages are not matched for $j = 4, 6, 8$ and the gate voltages are not matched for $j = 3, 4, 6, 8$. Similarly, violations for the two detected simple current mirrors ($scm$) are reported. It is noteworthy that no violation for the source pins of the differential pair $M3$ and $M4$ is reported for $j = 1$ and $j = 2$ (case marked with circled 2 in Fig 3.17), i.e. for net $n_{10}$ being $float$. This is an example for the special case mentioned in the previous section where the matching is guaranteed for a voltage of $float$ due to the physical connection between the matched devices.

A violation report similar to the one shown Table 3.10 is suggested as final output of the method. The voltage propagation results and the detected short-circuit paths can be highlighted in the schematic for further diagnosis in case of reported errors.

### 3.3.7. Experimental Results

In this section, experimental results for the detection of asymmetric voltages in power-down mode are discussed. The algorithm was tested on industrial analog circuit blocks. The algorithm is implemented in C++ using the open access library from Si2 [Si2 Consortium, 2013] to read in connectivity information of the netlist. The libraries for structure recognition $L_{analog}$ and $L_{pwd}$ as well as the the rules $\mathcal{R}$ are stored in an xml based format [World Wide Web Consortium (W3C), 2015]. The input voltage vectors $v_{init,up}$ and $v_{init,down,j}$ are read from text files. All voltage combinations for a given set $N_{ports,in/out}$ can be enumerated automatically. For all experiments, voltage propagation was run for all combinations of $vdd_i$ (with $i$ being the appropriate domain for the signal) and $gnd$, i.e. $v_{init,down,j}$ comprised $2^{|N_{ports,in/out}|}$ combinations.

In Table 3.11, the circuit size, i.e. the number of nets $|N|$ and the number of devices $|D|$, the number of swept ports $|N_{ports,in/out}|$ and the corresponding runtimes for four test circuits are given. The first line of the table shows that the runtime is below one second for the differential amplifier circuit introduced in Fig 3.10. The second line

| Circuit | $\|C\|$ | $\|N\|$ | $\|N_{ports,in/out}\|$ | runtime |
|---|---|---|---|---|
| Example Amp from Fig. 3.10[(1)] | 41 | 27 | 3 | $< 1\ s$ |
| LVDS ($0.13\mu$) [(1)] | 229 | 120 | 2 | $< 1\ s$ |
| LVDS Rx[(2)] | 620 | 265 | 7 | $45\ s$ |
| LVDS Tx[(2)] | 901 | 354 | 8 | $22\ min$ |

(1) Intel(R) Core(TM) i7-4770n CPU @ 3.4GHz, no parallelization used

(2) Intel(R) Xeon(R) CPU X5650 @ 2.67GHz, no parallelization used

Table 3.11.: Runtimes for symmetric power-down voltages verification

| | count | very strict | strict | moderate | loose |
|---|---|---|---|---|---|
| Simple Current Mirror ($scm$) | 7 | 4 drain | 4 drain | 0 | 0 |
| Wide Swing Cascode Current Mirror ($WSCCM$) | 9 | 5 drain<br>9 source | 5 drain<br>9 source | 0 | 0 |
| Differential Pair ($dp$) | 3 | 0 | 0 | 0 | 0 |

Table 3.12.: Detected analog structures and rule violations for LVDS Tx

shows the runtime for a low voltage differential signaling (LVDS) circuit implemented in a ($0.13\ \mu m$) technology. The circuit has two external ports, i.e. $|N_{ports,in/out}|$ is two. This corresponds to four combinations of $v_{init,down,j}$ to be analyzed, i.e. to four voltage propagation runs on the original circuit. For this example, the runtime was below one second. The last two lines correspond to the LVDS transmitter (LVDS Tx) and LVDS receiver (LVDS Rx) circuits that were also used to generate the experimental results for structure recognition (Table 2.1 in section 2.4) and for voltage propagation (Table 3.5 in section 3.2.7). The runtime for LVDS Tx was 45 seconds. The runtime for LVDS Rx was 22 minutes. Table 3.5 shows that a single voltage propagation run takes approximately one second for LVDS RX and approximately three seconds for LVDS Tx. Similarly, Table 2.1 shows that structure recognition takes approximately 5 seconds for LVDS RX and approximately 13 seconds for LVDS Tx. Comparing those runtimes for structure recognition and voltage propagation with the total runtimes, it can be concluded that the dominant part of the runtime is the time for the $2^{|N_{ports,in/out}|}$ voltage propagation runs, i.e. $2^7 = 128$ runs for LVDS Rx and $2^8 = 256$ runs for LVDS Tx. Reducing the runtime by exploiting the independence of these runs for parallel computing is possible future work. Furthermore, the use of specification data can reduce the number of combinations that have to be checked. In future work, it should also be investigated if bit-masking behaviour in the power-down signal part of the circuit can be exploited to reduce the number of combinations.

In Tables 3.12 and 3.13, the numbers of violated rules for LVDS Rx and LVDS Tx for rules of different levels of strictness are shown. In both tables, the column “count” notes

| | count | *dp*: very strict *scm*/*WSCCM*: moderate | *dp*: strict *scm*/*WSCCM*: moderate | all loose |
|---|---|---|---|---|
| Simple Current Mirror (*scm*) | 10 | 0 | 0 | 0 |
| Wide Swing Cascode Current Mirror (*WSCCM*) | 4 | 0 | 0 | 0 |
| Differential Pair (*dp*) | 3 | 2 gate | 0 | 0 |

Table 3.13.: Detected analog structures and rule violations for LVDS Rx

down the number of detected structures of the type given in the row. The other columns describe the tested rule set and state the number of violated rules for the structure given in the row.

For LVDS Tx, the algorithm was tested with the rule sets "very strict", "strict", "moderate" and "loose" as given in Table 3.7. Violations only occur for drain and source pins inside of current mirrors. All differential pairs in LVDS Tx fulfill the "very strict" matching and all current mirrors fulfill the "moderate" matching. This combination of matching rules corresponds to the rules given in Table 3.9.

For LVDS Rx, mixed rule sets have been tested as described by the three column headings in Table 3.13. The tested rule combinations for differential pairs/current mirrors are "very strict"/"moderate", "strict"/"moderate" and "loose"/"loose". Table 3.13 shows that two differential pairs violate the "very strict" rules. However, all differential pairs fulfill the "strict" rules. Furthermore, all current mirrors fulfill the "moderate" rules.

In order to test the effectiveness of the method to discover design errors, bugs have been deliberately introduced in the original designs. The introduced bugs are illustrated in Fig 3.18. Bug 1 on the top represents a supply change in one of the amplifiers from high voltage $vdd_2$ to the voltage $vdd_1$. This leads to a reported voltage mismatch for the gates of the the differential pair $(M3, M4)$. For another amplifier in the same design, a power-down switch pulling the gate of a differential stage to $vdd_2$ was deleted from the design. This is illustrated by Bug 2 in the middle of Fig 3.18. The error leads to a floating gate of $M4$ and therefore to a violated matching rule for the differential pair $(M3, M4)$. Inside a third amplifier, a pass gate has been modified by connecting one of the gates to signal *pwd* instead of the inverted signal $\overline{pwd}$ (Bug 3 on the bottom). As a result of this bug, a definite short-circuit coming from a higher hierarchy level of the design running over the falsely connected transistor $MS2$ and the power-down switch $MS5$ to ground was reported. The experiment demonstrates that the method succeeds in detecting the introduced errors.

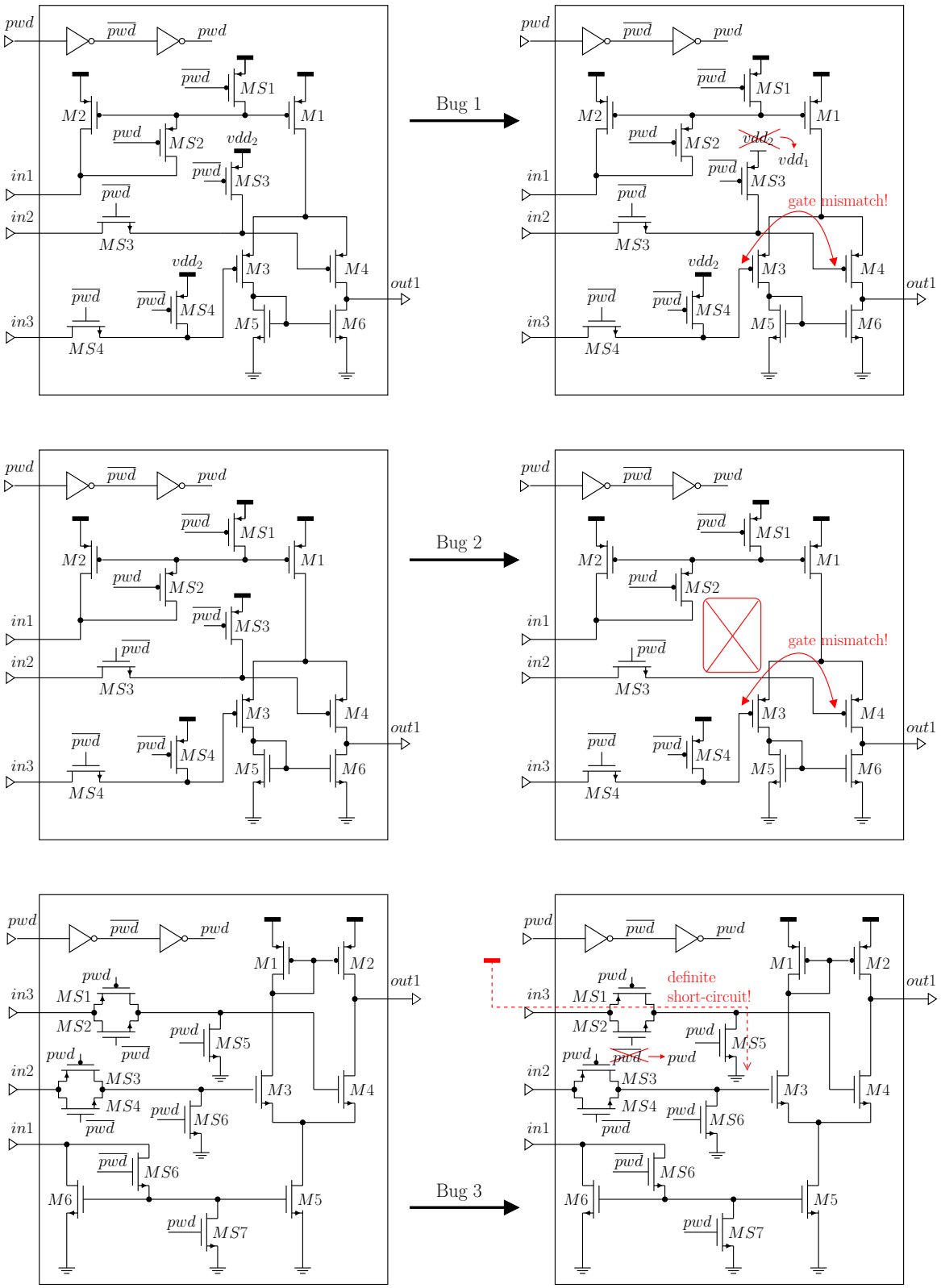



Figure 3.18.: Design bugs introduced into LVDS Tx and detected by the algorithm

# 4. Power-Down Synthesis

The verification of the correctness of the power-down circuitry was investigated in the previous chapter of this thesis. The verification problem will be reversed in this chapter, i.e., for a given circuit we will compute power-down circuitry that fulfills the requirements formulated for verification. Power-down circuitry that is "correct by construction" can be computed by following an automatic synthesis approach. Verification becomes obsolete in this way. A design flow following this idea is suggested in the next section. The power-down synthesis problem will be formulated (section 4.2) followed by a detailed description of the solution approach in section 4.3. The chapter concludes with the discussion of experimental results (section 4.4).

## 4.1. Design Flow with Automatic Synthesis

The target of this chapter is to develop an algorithm for automatic synthesis of the power-down circuitry. The availability of such a method changes the design flow of power-down circuit design. This is summarized in Fig. 4.1. The design flow suggested in Fig 3.1 (section 3.1) is shown on the left. The probability of introduced design errors can be systematically reduced by switching from manual power-down mode design to a fully-automatic or semi-automatic synthesis approach. Fig. 4.1 suggests two design flows that rely on automatic synthesis methods like the ones presented in this chapter.

The design flow shown on the upper right suggests a semi-automatic design flow which represents an evolutionary change from current state-of-the-art design flows. The power-down design is done with tool support for each sub block of the circuit individually. It is supported by the methods described in this chapter. They can be used to interactively help the designer to quickly add a correct power-down mode implementation to a given subcircuit in a systematic way. The automatically generated result can be seen as suggestion for the designer that he or she can override if required. The suggested semi-automatic approach takes away repetitive tasks from the designer while still allowing her or him to mentally track all modifications made to the design. However, verification of the full design is still required as final step as the synthesis algorithm only ensures correctness on sub-circuit level in this flow.

The design flow shown on the lower right takes the idea of power-down synthesis one step further. The power-down circuitry for the whole design is fully automatically added by an algorithm at the end of the design phase. No power-down verification step is required as the synthesized circuit is "correct-by construction". The methods presented in this

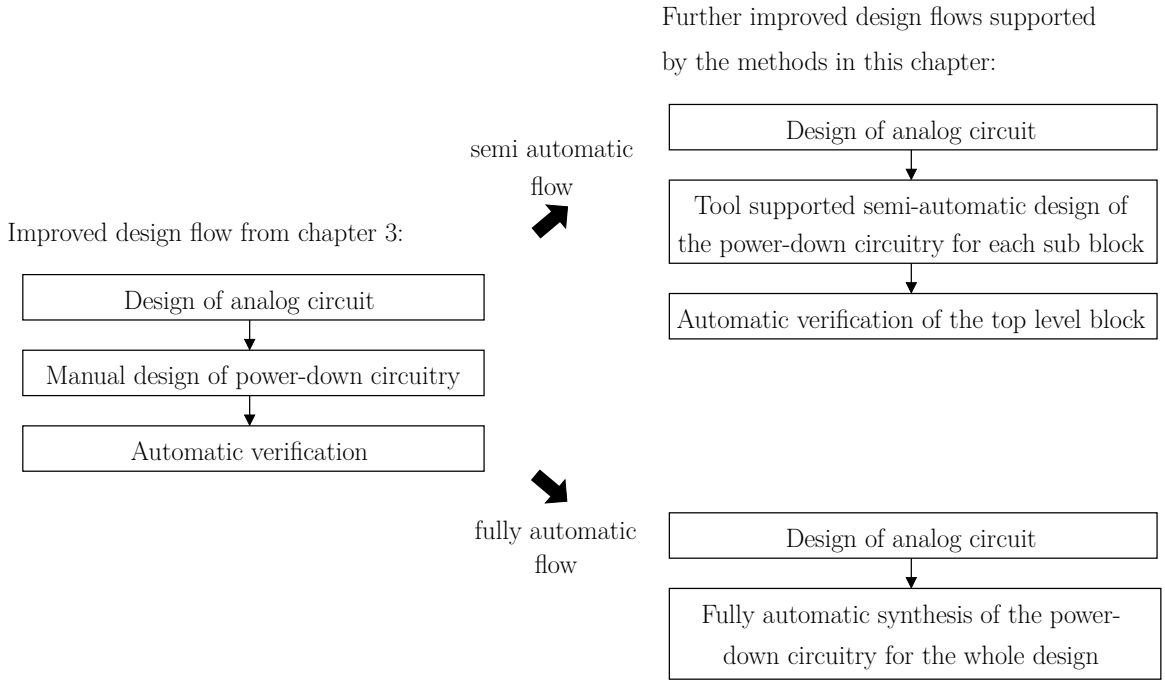



Figure 4.1.: Design flow from chapter 3 and possible further improved design flows supported by the methods in the chapter

chapter also support this kind of design flow. The experimental results in this chapter present the result of automatic power-down circuit synthesis for a complete (hierarchical) low voltage differential signalling (LVDS) design with about 180 transistors. As future work, it is suggested to test the algorithms of this chapter on even bigger designs as next step towards a design flow with fully automatic synthesis.

The flows presented in this section illustrate how the methods described in the following can support power-down mode design in practice. We will formulate the power-down synthesis problem in the next section. The chapter then continues with the description of new algorithms to solve the formulated problem.

## 4.2. The Power-Down Synthesis Problem

The power-down synthesis problem to be solved by the algorithms in this chapter can be formulated as follows:

*Power-Down Synthesis Problem [Zwerger et al., 2015a]:*

> For a given circuit, the complementary power-down circuitry should be synthesized, i.e., power-down switches with gates connected to $n_{pwd}$ or $n_{\overline{pwd}}$ should be added pursuing the objectives (A)-(E) described in the following.

The main objective is given as follows:

(A) Shut off all bias currents.

This objective has to be fulfilled for a functional power-down mode.

Furthermore, there are additional objectives that should be fulfilled as much as possible. They are given as follows:

(B) Avoid deterioration of the power-up behavior, e.g., by parasitic capacitances at critical nodes or by shift of the bias point.
(C) Avoid reliability problems due to electrical stress in power-down mode, e.g., ensure symmetric voltages at matched devices [Michael et al., 1992; Chen et al., 2001; Massier et al., 2008; Zwerger and Graeb, 2015].
(D) Avoid floating nodes in order to produce predictable behavior and facilitate verification [Zwerger and Graeb, 2013; Blieck and Janssens, 1996].
(E) Minimize the area overhead introduced by the power-down circuitry, i.e., minimize the size and the number of power-down switches.

The synthesis problem can be divided into three sub problems:

- topology synthesis
- switch sizing
- schematic back annotation

In the following, we will mainly focus on the topology synthesis following the goals (A) to (E). Topology synthesis adds power-down switches to the netlist. It follows a purely structural approach, i.e., the sizes of the switches are not determined in this step. The sizes of the switches have to be determined with a separate step. The problem of finding the sizes of the switches for a given power-down circuit topology is called "switch sizing". Section 4.4.6 shows that the sizing of the switches is mostly an easy problem, as switches of minimum width and length usually are the best choice. In order to integrate power-down synthesis into state-of-the-art design flows, the synthesized power-down circuitry needs to be written back to an existing schematic of the circuit. This step is called "schematic back annotation". For back annotation, the geometries of the graphical representation of the netlist need to be synthesized. A solution approach to this problem is briefly outlined in section 4.3.7. The next section continues with the discussion of new approaches for solving the three mentioned sub problems of power-down synthesis. The main focus is on algorithms for the topology synthesis of power-down circuitry.

## 4.3. Power-Down Synthesis Algorithm

The power-down synthesis problem is solved by the algorithms described in the following. Section 4.3.1 presents common shut-off patterns that are often found in handcrafted designs. Those patterns are also used by the algorithms suggested in this chapter. An

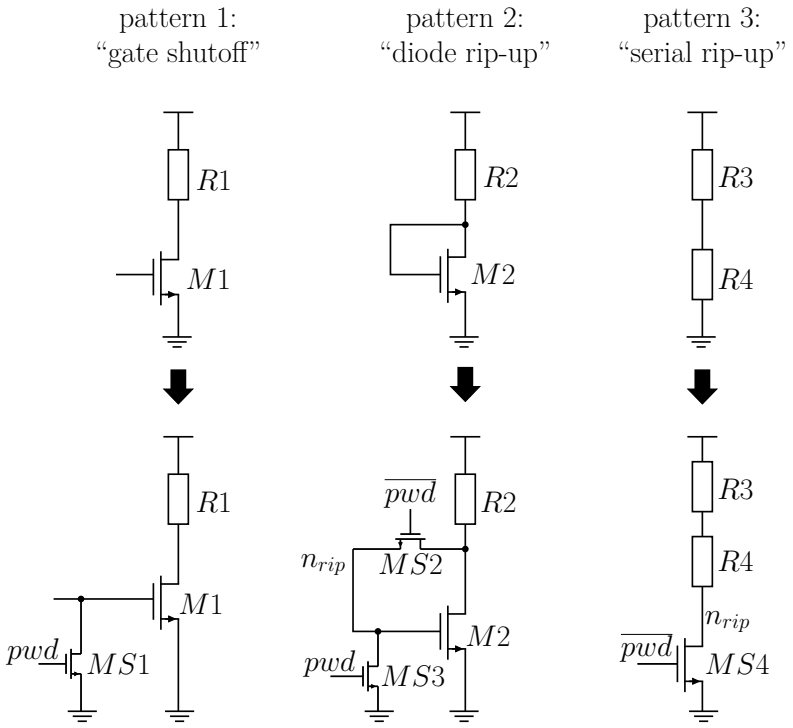



Figure 4.2.: Shutoff Strategies [Zwerger et al., 2015a]

overview over the topology synthesis algorithm is given in section 4.3.2. The basis of topology synthesis consists of three steps: A rip-up step (section 4.3.3), a gate-shutoff step (section 4.3.4) and net dependency analysis (section 4.3.5). The basic method can be extended with automatic selection of an optimal solution based on the secondary design goals (B) to (E). An example heuristic is described in section 4.3.6. The last subsection of this section briefly addresses the schematic back annotation problem (section 4.3.7).

### 4.3.1. Shutoff Strategies

The primary goal of the power-down synthesis problem is to switch off all bias currents during power-down mode (A). The additional power-down switches should not affect the power-up (normal operation) behavior of the circuit (B). Fig. 4.2 shows three basic patterns to shut down bias currents which are commonly used in handcrafted designs and applied by the algorithms described in the following.

The first pattern “gate shutoff” introduces a power-down switch $MS1$ which forces the gate of n-MOSFET $M1$ to ground, i.e., it forces $M1$ into the cutoff region. A complementary pattern can be applied to a p-MOSFET by pulling the gate to $vdd$ with a p-switch that has the inverted signal $\overline{pwd}$ connected to its gate. In both cases, the power-down switch is (nearly) not interfering with the circuit in power-up mode, as it is not conducting (B). The switch transistor can be sized small, as it only has to charge the

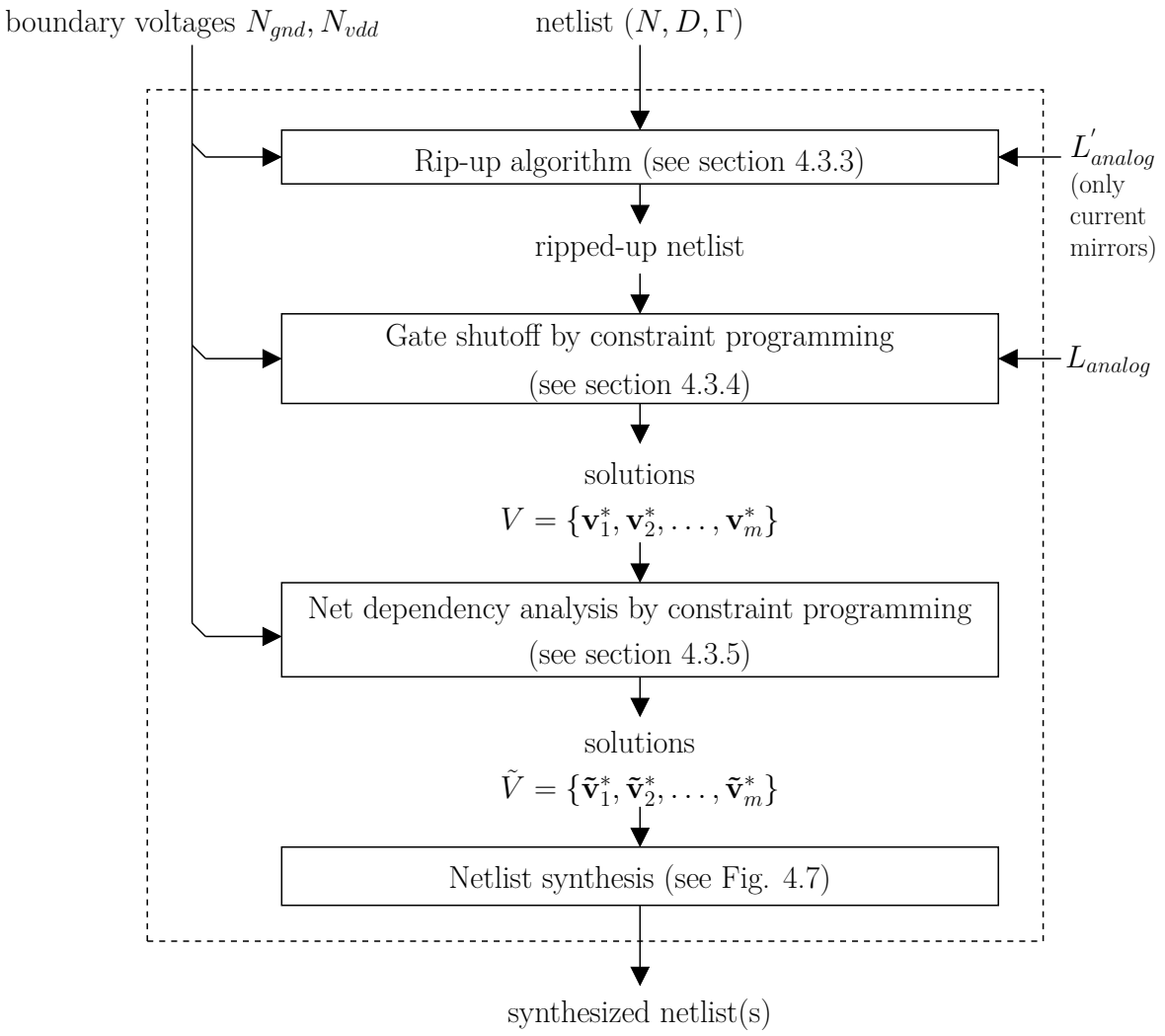



Figure 4.3.: Overview of the topology synthesis approach for power-down circuits [Zwerger et al., 2015a]

gate of $M1$ (E). It is not always possible to apply the gate shutoff pattern on every bias current path from vdd to ground. For instance, other current paths could be opened if other components are connected to the gate net of $M1$. Methods to resolve those dependencies will be presented in this chapter.

The second pattern "diode rip-up" shows an example path where gate shutoff is not possible. Adding the power-down switch $MS3$ to pull down the gate of $M2$ will open a new current path from vdd over $R2$ over the newly introduced switch $MS3$ to ground. Therefore, this current flow needs to be interrupted. This is done by adding another switch $MS2$ which will conduct in power-up mode and isolate in power-down mode. A complementary pattern can be applied for p-type diode connected transistors. The "diode rip-up" pattern has the advantage that the biasing of the original circuit on the

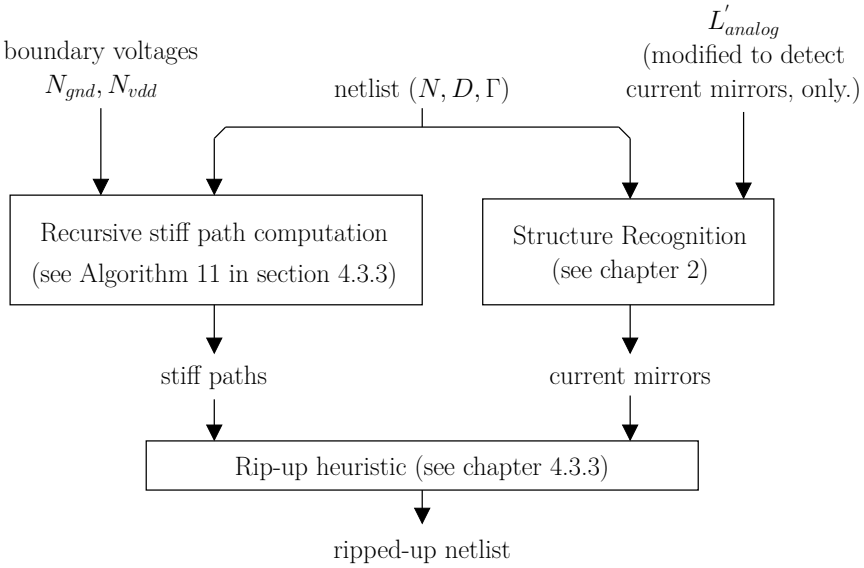



Figure 4.4.: Overview of the rip-up algorithm [Zwerger et al., 2015a]

current path is not changed. Consequently, the power-up behavior of the original circuit is hardly affected by the switches (B). Furthermore, $MS2$ and $MS3$ only have to load the gate of $M2$. No static current will be flowing over them. Hence, the switches can be sized small (E). The diode rip-up pattern can be extended for current mirrors (see Fig 4.6).

In the case that there is simply no transistor eligible for gate shutoff or diode rip-up on the path, the last resort is to rip up the current path by adding a serial switch as shown by the pattern "serial rip-up". However, this pattern has the disadvantage that the biasing of the original circuit is changed and the switch cannot be sized small as it has to carry the whole bias current. Consequently, this pattern jeopardizes goals (B) and (E) of the power-down sizing problem and, hence, should be avoided as far as possible.

In the following sections, a strategy to apply the presented basic patterns following rationale of this section is developed. First, paths where rip-up is required are identified and ripped up (section 4.3.3). Then, the remaining paths are shut off by gate shutoff (section 4.3.4). Before proceeding with the details of the method, the next section will give an overview of the method.

### 4.3.2. Overview

Fig. 4.3 gives an overview of the power-down synthesis method. The method takes the circuit netlist $(N, D, \Gamma)$ as input where $N$ is the set of nets, $D$ is the set of devices and $\Gamma$ is the connectivity as defined in (3.25). Furthermore, boundary voltage conditions described by the sets $N_{gnd}$ and $N_{vdd}$ are required. The sets contain all nets that are

known to be driven to *gnd* or *vdd* from external sources. The sets $N_{gnd}$ and $N_{vdd}$ typically contain the supply rail nets. Furthermore, they can include nets that are connected to external pins of a subcircuit. In this chapter, only one *vdd* supply voltage is taken into account. Extending the algorithms for multiple supply voltages $vdd_i$ is future work.

The method first applies a rip-up algorithm to the netlist. The rip-up algorithm follows a heuristic to rip-up current mirrors. It requires a reduced version $L'_{analog}$ of the library $L_{analog}$ (see Fig. 2.1 in chapter 2) that only contains the current mirror building blocks. The ripped-up netlist has the property that all currents can be shut off by the gate-shutoff pattern. A constraint program is formulated and solved in order to apply this pattern. The solutions of the constraint program are denoted as

$$V = \{\mathbf{v_1^*}, \mathbf{v_2^*}, \ldots, \mathbf{v_m^*}\}. \tag{4.1}$$

Each solution corresponds to a valid power-down mode implementation that takes mainly into account the goals (A), (B) and (D). A solution $\mathbf{v_i^*}$ denotes for each net which voltage it has to take in power-down mode. It has to be decided for each net and solution if it has to be pulled to the according voltage by a power-down switch or not. The number of power-down switches can be minimized (E) by analyzing the net dependencies with another constraint program (see section 4.3.5) for a given solution $\mathbf{v_i^*}$. The matching of voltages (C) is addressed in section 4.3.6 where an approach for the automatic selection of solutions is presented. The resulting solutions which take into account matching and which have a minimized number of switches are denoted as

$$\tilde{V} = \{\tilde{\mathbf{v}}_\mathbf{1}^*, \tilde{\mathbf{v}}_\mathbf{2}^*, \ldots, \tilde{\mathbf{v}}_\mathbf{m}^*\}. \tag{4.2}$$

A netlist can be derived from those solutions (see Fig. 4.7 in section 4.3.5). The output of the method are one or more netlists with added power-down switches. The individual steps are described in more detail in the following sections.

### 4.3.3. Rip-Up Algorithm

The rip-up algorithm takes the netlist $(N, D, \Gamma)$, the voltage boundary conditions $N_{gnd}$ and $N_{vdd}$ as well as a reduced library $L'_{analog}$ as input. The output is the ripped-up netlist. An overview of the rip-up method is given in Fig. 4.4.

First, all bias current paths that cannot be shut off by the gate shutoff pattern without creating new current paths are determined. Those paths are called stiff paths in the following. The computation of stiff paths is described in the following subsection. Every stiff path that connects the supply with ground needs to be ripped up. This is done by a rip-up heuristic. The rip-up heuristic requires the information which devices form a current mirror. Therefore, structure recognition (see chapter 2) is performed and the detected current mirrors are passed on to the rip-up heuristic. The rip-up heuristic is described in this section after the stiff path analysis.

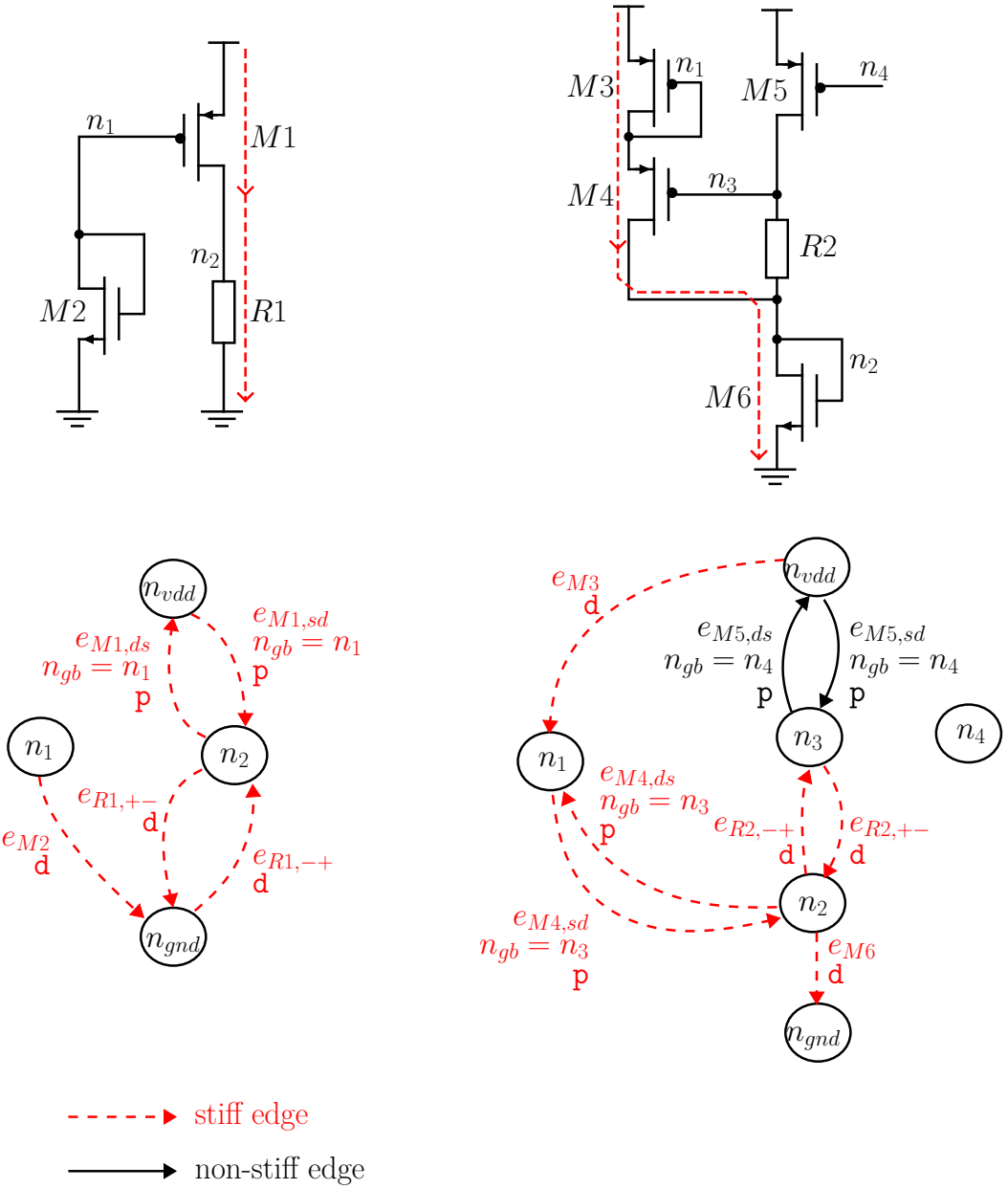



Figure 4.5.: Examples of stiff paths [Zwerger et al., 2015a]

### Stiff Path Analysis

Stiff path analysis resolves the dependencies between different bias current paths from vdd to ground. Applying gate shutoff can create new current paths (see section 4.3.1). All paths that cannot be shut off with gate shutoff without creating new current paths are called stiff paths. In general, any path between any two nets of the circuit can be classified as stiff or non-stiff. The problematic stiff paths that need to be ripped up later are those which connect the supply with ground. They can be computed recursively.

Fig. 4.5 shows two examples of stiff paths from the supply to ground. The example circuits are shown on the top. The stiff paths from the supply to ground are marked with red dashed arrows in the circuits. The corresponding graph representations of the circuits as defined in Fig. 3.3 (see section 3.2.3) are shown below the circuits.

First, we take a look at the circuit on the top left. The gate of $M1$ is "hard-wired" to ground with the diode-connected transistor $M2$. Shutting off the current through $M1$ means pulling the gate of $M1$ to vdd. However, a new current path is created if the gate of $M1$ is pulled to vdd. The new current path would be running from vdd over the new switch to $n1$ and over the diode transistor $M2$ to ground. Thus, it is not possible to shut off the current in the path over $M1$ and $R1$ with gate shutoff and without creating a new current path. It follows that the path is stiff.

Second, we take a look at the circuit on the top right. It shows a less obvious example of a stiff path. The circuit has two bias current paths. The first path runs from vdd over $M3$, $M4$ and $M6$ to ground. The second path runs from vdd over $M5$, $R2$ and $M6$ to ground.

The current in the second path can be shut off by pulling the gate of $M5$ to $vdd$. The net $n_3$ will take a voltage near the ground voltage due to $R2$ and diode transistor $M6$ if this path is shut off in this way. It follows that $M4$ is conducting.

The current in the first path ($M3$, $M4$, $M6$) cannot be shut off without creating a new current path. A new current path over the added switch is created if gate shutoff is applied to any of $M3$, $M4$ or $M6$. This can be confirmed as follows:

- Pulling $n_1$ to vdd in order to shut off $M3$ opens a path over the new switch over $M4$ and $M6$ to ground.
- Pulling $n_3$ to vdd in order to shut off $M4$ opens a path over the new switch over $R2$ and $M6$ to ground.
- Pulling $n_2$ to gnd in order to shut off $M6$ opens a path from vdd over $M3$, $M4$ and the new switch to ground.

If follows that the path is a stiff path.

The circuit is transformed into the graph defined in Fig. 3.3 (see section 3.2.3) in order to compute the stiff paths. The graphs are shown at the bottom of Fig. 4.5. The candidate paths for the paths that require rip-up run from vdd to ground. The set of all paths

from vdd to ground is $\mathcal{P}_{vdd \to gnd}$ as defined in (3.17). The candidate paths can be split into two paths: A path from vdd to a node on the path and a path from the same node to ground. The two parts of the candidate path are then element of the general set of paths $\mathscr{P}$ as defined in (3.16). The described decomposition will be used in the following in order to formulate conditions for the stiffness of paths.

A path $P$ is a tuple of edges (3.16). It can be represented as follows:

$$P = (e_1, e_2, \ldots, e_n) \quad \text{with} \quad P \in \mathscr{P} \tag{4.3}$$

Each edge $e_i$ has a property "stiff". The property is defined as follows:

$$e_i.\text{stiff} \in \{0, 1\} \tag{4.4}$$

The value $e_i.\text{stiff} = 0$ means that the edge is not stiff and $e_i.\text{stiff} = 1$ means that the edge is stiff. Stiff edges are drawn as red dashed arrows in the graphs in Fig. 4.5. The non-stiff edges are drawn as black non-dashed arrows.

A stiff path is a path that runs along stiff edges only. The following equivalence holds:

$$\text{"Path } P = (e_1, e_2, \ldots, e_n) \text{ is stiff"} \iff \bigvee_{e_i = e_1, \ldots, e_n} e_i.\text{stiff} = 1 \tag{4.5}$$

All stiff paths that connect vdd and ground require rip-up, i.e., the result of the rip-up analysis is the following set:

$$\mathcal{P}_{ripup} = \{P \in \mathcal{P}_{vdd \to gnd} \mid \text{"}P \text{ is stiff"}\} \tag{4.6}$$

It is computed in lines 12 to 16 of Algorithm 11.

The definition of the stiffness depends on the type of the edge. The type of an edge $e_i$ is $\tau(e_i)$ as defined in (3.5). The type can either be `d`, `n` or `p`. Edges of type `d` cannot be switched off by gate shutoff as they are always *on*. Consequently, they are always stiff. The following implication holds:

$$\tau(e_i) = \texttt{d} \quad \Longrightarrow \quad e_i.\text{stiff} = 1 \tag{4.7}$$

Edges of type `n` and `p` can either be *on* or $off$ if the voltages in the graph are limited to *vdd* and *gnd* (see equation (3.13) or Table 3.1). An `n` or `p` edge is stiff if it cannot be switched off by gate shutoff without creating another current path. The paths in the graph running over the switching node determine if an edge can be switched off by gate shutoff without creating another current path. The switching node of an edge $e_i$ is $s(e_i)$ as defined in (3.11). The switching node corresponds to the gate/base node of

the transistor the edge is representing. It is abbreviated as $n_{gb}$ in the following, i.e., $s(e_i) = n_{gb}$.

We will first define the stiffness property for `n` edges. An `n` edge is switched off by gate shutoff by pulling $n_{gb}$ to ground with a switch. If it is assumed that $n_{gb}$ is not located on another stiff path, the switch will create a new stiff path from the supply to ground under the following condition:

$$\text{“There is a stiff path from any node in } N_{vdd} \text{ to } n_{gb}.\text{”} \tag{4.8}$$

The edge should not be marked as stiff if $n_{gb}$ is located on another stiff path. This exception avoids that paths that depend on other stiff paths are marked as stiff. Only the “root cause path” should be marked as stiff. Therefore, the additional condition to (4.8) is that $n_{gb}$ is not located on another stiff path from supply to ground. It can be decomposed into the following two conditions:

$$\text{“There is a stiff path from any node in } N_{vdd} \text{ to } n_{gb}.\text{”} \tag{4.9}$$
$$\text{“There is a stiff path from } n_{gb} \text{ to any node in } N_{gnd}.\text{”} \tag{4.10}$$

There is a stiff path from the supply over $n_{gb}$ to ground if both conditions are fulfilled. The first condition (4.9) corresponds to the first half of the stiff path from the supply to ground. The second condition (4.10) corresponds to the second half of the stiff path from the supply to ground. Additionally, the first condition (4.9) is the same as (4.8). It follows that $n_{gb}$ is not located on another stiff path from the supply to ground if the negation of (4.10) is true. The negation of (4.10) is given as follows:

$$\text{“There is no stiff path from } n_{gb} \text{ to any node in } N_{gnd}\text{”} \tag{4.11}$$

In summary, an edge is stiff if (4.8)/(4.9) and (4.11) are fulfilled. This leads to the following definition of the stiffness property of `n` edges:

$$e_i.\text{stiff} = 1 \Longleftrightarrow \begin{cases} \text{“It exists a stiff path from any node in } N_{vdd} \text{ to } n_{gb}.\text{”} \wedge \\ \text{“There is no stiff path from } n_{gb} \text{ to any node in } N_{gnd}.\text{”} \end{cases} \tag{4.12}$$

The definition of the stiffness of `p` edges follows the same pattern. A `p` edge is switched off by gate shutoff by pulling $n_{gb}$ to vdd with a switch. The switch will create a new stiff path from the supply to ground if there is a stiff path from $n_{gb}$ to any node in $N_{gnd}$. The gate/base net $n_{gb}$ is not located on another stiff path if there is no stiff path from any node in $N_{vdd}$ to $n_{gb}$. Consequently, the definition of the stiffness property of `p` edges is given follows:

$$e_i.\text{stiff} = 1 \Longleftrightarrow \begin{cases} \text{“It exists a stiff path from } n_{gb} \text{ to any node in } N_{gnd}.\text{”} \wedge \\ \text{“There is no stiff path from any node in } N_{vdd} \text{ to } n_{gb}\text{”} \end{cases} \tag{4.13}$$

**Algorithm 11** Stiff path computation

**Require:** $N_{gnd}$, $N_{vdd}$, circuit netlist $(N, D, \Gamma)$

1: Build graph $G = (N, E, \phi^+, \phi^-, \tau)$ of circuit as described in Fig 3.3 of section 3.2.3
2: **for all** $e_i$ in $E$ **do**
3: $e_i$.stiff = *not_determined*
4: **end for**
5: **for all** $e_i$ in $E$ **do**
6: **if** $e_i$.stiff = *not_determined* **then**
7: edgeStack = $(e_i)$
8: StiffPathRecursion($e_i$, edgeStack, $G$, $N_{gnd}$, $N_{vdd}$) `// Algorithm 12`
9: **end if**
10: **end for**
11:
12: $\mathcal{P}_{ripup} = \{\}$
13: **for all** $n_{vdd} \in N_{vdd}$ **do**
14: Search all paths from $n_{vdd}$ along the forward direction of edges with $e$.stiff = 1 to any node in $N_{gnd}$. Append the resulting paths to $\mathcal{P}_{ripup}$.
15: **end for**
16: **return** $\mathcal{P}_{ripup}$

All other `n` or `p` edges that do not fulfill the conditions in (4.12) or (4.13) are not stiff. The following equivalence holds for `n` and `p` edges:

$$e_i.\text{stiff} = 0 \iff \text{“All cases not covered by (4.12) or (4.13)”} \tag{4.14}$$

The equivalences (4.12) and (4.13) hold for all stiff (red dashed) `n` and `p` edges in Fig. 4.5.

The stiffness of edges was defined in (4.12) and (4.13) using conditions that demand or exclude stiff paths between the supply, $n_{gb}$ and ground. A stiff path is defined as a path consisting only of stiff edges by equation (4.5). Equation (4.5) uses the definition of the stiffness of edges, i.e. (4.12) and (4.13). This is a circular dependency. It follows that the definitions are recursive. The circular dependency can be resolved by a recursive algorithm which is described in the following.

The algorithm to compute stiff paths is given in Algorithm 11. It takes $N_{gnd}$, $N_{vdd}$ and the circuit netlist $(N, D, \Gamma)$ as input. The algorithm executes as follows: First, the graph is built from the circuit (line 1). Second, the stiffness of all edges is initialized with the value *not_determined* (lines 2 to 4). Third, the recursive computation is started for each edge $e_i$ with undetermined stiffness (line 8). Each edge has a determined stiffness after completion of the recursive computation. Fourth, the graph can be searched for stiff paths using depth-first search starting from all nodes in $N_{vdd}$ along all stiff edges to all nodes in $N_{gnd}$ (lines 12 to 16). The fourth step is illustrated by Fig. 4.5: The stiff

**Algorithm 12** StiffPathRecursion

**Require:** $e_i$, edgeStack, $G = (N, E, \phi^+, \phi^-, \tau)$, $N_{gnd}$, $N_{vdd}$

**if** $\tau(e_i) =$ `d` **then**
  $e$.stiff = 1
**else if** $\tau(e_i) =$ `n` **or** $\tau(e_i) =$ `p` **then**
  $n_{gb} = s(e_i)$ `// gate or base node`
  **if** $n_{gb} \in N_{gnd} \cup N_{vdd}$ **then**
    $e_i$.stiff := 0 or $e_i$.stiff := 1 according to Table 4.1
  **else**
    Compute backward paths $\mathcal{P}_{bw}$ from $n_{gb}$ to any node in $N_{vdd}$.
    FowardBackwardPathsRecursion($\mathcal{P}_{bw}$, edgeStack, $G$, $N_{gnd}$, $N_{vdd}$) `// Algor.13`

    Compute forward paths $\mathcal{P}_{fw}$ from $n_{gb}$ to any node in $N_{gnd}$.
    FowardBackwardPathsRecursion($\mathcal{P}_{fw}$, edgeStack, $G$, $N_{gnd}$, $N_{vdd}$) `// Algor.13`

    $e_i$.stiff := 0 or $e_i$.stiff := 1 according to equations (4.12), (4.13) and (4.14)
  **end if**
**end if**

**Algorithm 13** FowardBackwardPathsRecursion

**Require:** paths $\mathcal{P}$, edgeStack, $G = (N, E, \phi^+, \phi^-, \tau)$, $N_{gnd}$, $N_{vdd}$

**for all** $P$ in $\mathcal{P}$ **do**
  **for all** edges $e_{next}$ of path $P$ **do**
    **if** edgeStack does not contain $e_{next}$ **and** $e_{next}$.stiff = *not_determined* **then**
      push $e_{next}$ on top of edgeStack
      StiffPathRecursion($e_{next}$, edgeStack, $G$, $N_{gnd}$, $N_{vdd}$) `// Algorithm 12`
      pop topmost element from edgeStack
    **end if**
  **end for**
**end for**

| | $n_{gb} \in N_{gnd}$ | $n_{gb} \in N_{vdd}$ |
|---|---|---|
| $\tau(e_i) = \texttt{n}$ | $e_i$.stiff := 0 | $e_i$.stiff := 1 |
| $\tau(e_i) = \texttt{p}$ | $e_i$.stiff := 1 | $e_i$.stiff := 0 |

Table 4.1.: Assignment of 0 or 1 to $e_i$.stiff for edges that are controlled by a supply node

paths marked in the circuits on the top can be found in the graphs below by tracing the red dashed edges from $n_{vdd}$ to $n_{gnd}$. Furthermore, the drawn stiff path in each example circuit is the only stiff path in each of the example circuits.

The recursive computation of the stiffness is given by Algorithm 12. It takes the current edge $e_i$, an edgeStack that traces the recursive calls, the graph $G$ and the boundary conditions $N_{gnd}$, $N_{vdd}$ as input. An edge $e_i$ is stiff if it is a diode (`d`) edge (line 2). The assignment in line 2 corresponds to (4.7).

The gate/base net $n_{gb}$ needs to be taken into account for `n` and `p` edges (line 4). The stiffness is determined according to Table 4.1 if $n_{gb}$ is a boundary net in $N_{gnd}$ or $N_{vdd}$ (line 6). Edges with $n_{gb}$ "hard wired" to a supply such that the edge is *on* are stiff. Edges with $n_{gb}$ "hard wired" to a supply such that the edge is $off$ are not stiff as they cannot be part of a short-circuit path.

The conditions (4.12) or (4.13) have to be evaluated recursively if the gate/base net $n_{gb}$ is not a boundary net in $N_{gnd}$ or $N_{vdd}$ (lines 8 to 14). All paths from any node in $N_{vdd}$ to $n_{gb}$ and all paths from $n_{gb}$ to any node in $N_{gnd}$ have to be computed for this purpose. The resulting sets of paths are called backward paths $\mathcal{P}_{bw}$ and forward paths $\mathcal{P}_{fw}$. The backward paths are defined as follows:

$$\mathcal{P}_{bw} = \{P \in \mathscr{P} \mid \text{“Path } P \text{ is starting in } N_{vdd} \text{ and ending at } n_{gb}.\text{”}\} \tag{4.15}$$

The backward paths can be computed by starting a depth-first search at node $n_{gb}$ in backward direction of the edges (line 8).

The forward paths are defined as analogously:

$$\mathcal{P}_{fw} = \{P \in \mathscr{P} \mid \text{“Path } P \text{ is starting at } n_{gb} \text{ and ending in } N_{gnd}.\text{”}\} \tag{4.16}$$

The forward paths can be computed by starting a depth-first search at node $n_{gb}$ in forward direction of the edges (line 11).

The stiffness of all paths in $\mathcal{P}_{bw}$ and $\mathcal{P}_{fw}$ has to be determined while evaluating (4.12) and (4.13) (line 14). In order to evaluate the stiffness of the paths, the stiffness of the paths' edges needs to be determined. This is done by the routine "ForwardBackwardPathsRecursion" which is called for $\mathcal{P}_{bw}$ and $\mathcal{P}_{fw}$ (lines 9 and 12). The routine "ForwardBackwardPathsRecursion" recursively determines the stiffness of all edges on all paths by calling "StiffPathRecursion" for every edge on every path. An edgeStack

tracks the visited edges during the recursion in order to avoid infinite recursive calls. The recursion is only continued for edges which were not yet visited during the current recursion and which do not have a determined stiffness yet (see condition in line 3 of Algorithm 13). The recursive call structure ("StiffPathRecursion" calls two times "ForwardBackwardPathsRecursion", 'ForwardBackwardPathsRecursion" calls "StiffPathRecursion") resolves the circular dependency between (4.5), (4.12) and (4.13).

The forward and backward paths recursion can be illustrated by the examples in Fig. 4.5. In order to determine the stiffness of $M4$ in the circuit on the right, the stiffness of the edges $e_{M4,sd}$ and $e_{M4,ds}$ has to be determined. The common gate/base net of $e_{M4,sd}/e_{M4,ds}$ is $n_{gb} = n_3$. There is one backward path $(e_{M5,sd})$ from $n_3$ towards the supply. It is computed by starting a depth-first search at $n_3$ running counter the direction of $e_{M5,sd}$ to $n_{vdd}$. The resulting set of backward paths is $\mathcal{P}_{bw} = \{(e_{M5,sd})\}$. Furthermore, there is one forward path $(e_{R2,+-}, e_{M6})$ from $n_3$ to ground. It is computed by starting a depth-first search at $n_3$ running in direction of the edges over $e_{R2,+-}$ and $e_{M6}$ to $n_{gnd}$. The resulting set of forward paths is $\mathcal{P}_{fw} = \{(e_{R2,+-}, e_{M6})\}$. The stiffness of the edges $e_{M4,sd}$ and $e_{M4,ds}$ can be determined using $\mathcal{P}_{bw}$ and $\mathcal{P}_{fw}$. Both edges are `p` edges. Consequently, equation (4.13) needs be evaluated.

The first condition of (4.13) is true if there is a stiff path from $n_3$ to ground. There is one path from $n_3$ to ground. This path is $(e_{R2,+-}, e_{M6})$. The path is a stiff path because the edges $e_{R2,+-}$ and $e_{M6}$ are `d` edges and `d` edges are always stiff according to (4.7).

The second condition of (4.13) requires that the path $(e_{M5,sd})$ is not stiff. The stiffness of path $(e_{M5,sd})$ is determined recursively by determining the stiffness of the edge $e_{M5,sd}$. The gate/base net $n_{gb}$ is $n_4$ and the edge $e_{M5,sd}$ is of type `p`. Consequently, equation (4.13) needs to be evaluated with $n_{gb} = n_4$. Nothing is connected to $n_4$. It follows that $\mathcal{P}_{bw}$ and $\mathcal{P}_{fw}$ are empty. Consequently, there is no stiff path from $n_{gb}$ to ground as required by the first condition of (4.13). It follows that the edge $e_{M5,sd}$ as well the path $(e_{M5,sd})$ are not stiff.

The recursion returns to the starting edges $e_{M4,sd}/e_{M4,ds}$. The path $(e_{M5,sd})$ has been determined as non-stiff. Hence, there is no stiff path from the supply to $n_3$ and the second condition of (4.13) is fulfilled for the edges $e_{M4,sd}$ and $e_{M4,ds}$. It follows that the edges $e_{M4,sd}$ and $e_{M4,ds}$ are stiff. The recursive procedure is started again for any edge that has still an undetermined stiffness. The graph is searched for stiff paths (lines 12 to 16 in Algorithm 11) once all edges have a determined stiffness. The resulting set of stiff paths from the supply to ground is $\mathcal{P}_{ripup}$. The paths in $\mathcal{P}_{ripup}$ are further processed by the rip-up heuristic described in the next subsection.

### Rip Up of Stiff Paths

For each stiff path in $\mathcal{P}_{ripup}$, a rip-up point has to be computed. We have seen in section 4.3.1 that diode rip up is preferred over serial rip-up due to goal (B) of the

power-down sizing problem. Another rip-up pattern is introduced in Fig. 4.6: current-mirror rip-up. In analog circuits, many bias current paths are biased by a current mirror that is fed by one input current source. The gates of all output branches of a current mirror bank are connected. Hence, the currents in all output branches can be switched off with only one gate shutoff switch if it is possible to pull the gates to ground for n-MOSFETs or to vdd for p-MOSFETs.

The input of a current mirror typically contains a diode-connected transistor which needs to be ripped up because it is part of a stiff path. The experimental results as well as the inspection of typical shutoff patterns applied by manual designs show that it is very efficient to rip up the input diode of a current mirror bank such that all output current branches can be turned off by a single gate-shutoff transistor. Consequently, the following rip-up heuristic is proposed (with decreasing order of precedence) [Zwerger et al., 2015a]:

1. Current mirror rip-up (example see Fig. 4.6)
2. Diode rip-up (see Fig. 4.4, middle)
3. Serial rip-up (see Fig. 4.4, right)

The rip-up procedure examines if a diode-connected transistor is located on the stiff path. Serial rip-up has to be applied if this is not the case. A serial rip-up heuristic can, e.g., add an n-MOS transistor between the ground net and the component connected to ground as illustrated by "case 3" in Fig 4.2. Else, a diode transistor on the stiff path has to be chosen for rip-up. The diode connected transistor can either be part of a current mirror or not. Diode rip-up as illustrated by "case 2" in Fig. 4.2 is applied in the first case. Current-mirror rip-up is applied in the second case.

The difference between current-mirror rip-up and diode rip-up is subtle: Diode rip-up only separates the gate of the diode transistor from the rest of the pins connected to the rip-up net. In contrast, current-mirror rip-up separates all current mirror gates from the other pins of the rip-up net. In order to implement current-mirror rip-up the current mirror gates need to be determined. This is done by structure recognition.

The current mirrors are detected using a modified version of the structure recognition method from chapter 2. The library $L_{analog}$ (see Fig. 2.1) is reduced to detect only current mirrors. The reduced library is called $L'_{analog}$. The modifications of structure recognition can be summarized as follows:

- The differential pair ($dp$) and the differential stage ($DS$) are left out in $L'_{analog}$. Consequently, the highest hierarchy level of $L'_{analog}$ is 3.
- The persistence of the bias shifter ($bs$) is changed from 1 to the maximum which is 3.
- All bias shifters where both, the left and the right transistor are part of another current mirror ($scm$, $4TCM$, $WSCCM$, $WCM$, $CCM$ or $IWCM$) are removed from the recognized building blocks in each iteration of the recognition process. This can be expressed as dominance relation [Massier, 2010]. Bias shifters where

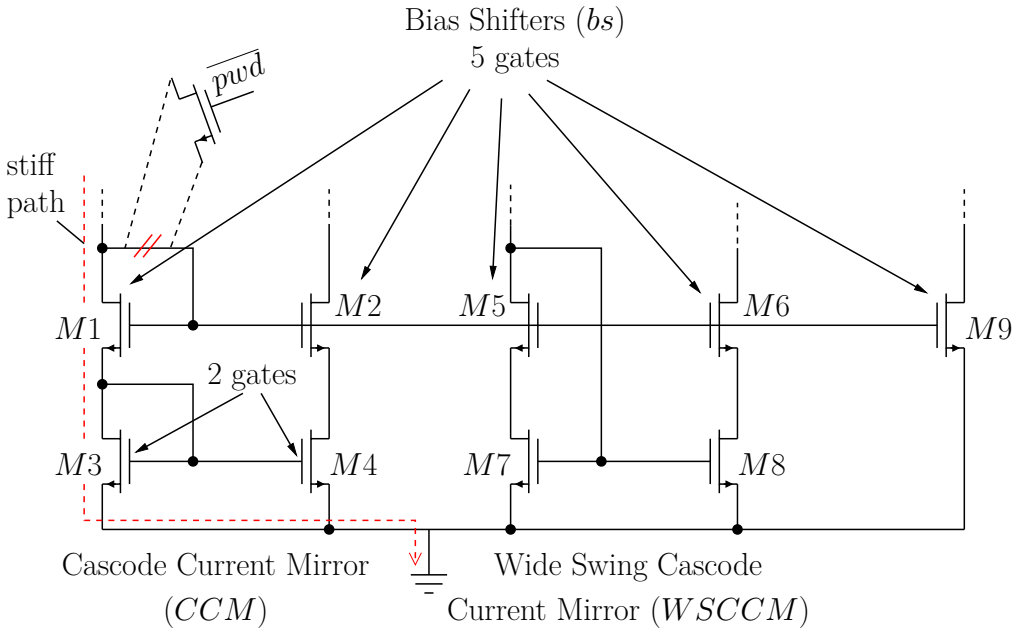



Figure 4.6.: Example of current mirror rip-up heuristic

only one transistor is part of another current mirror are kept in the recognition result. An example for this case is the bias shifter ($M1$,$M9$) as shown in Fig. 4.6.

The result of structure recognition using the modified library $L'_{analog}$ is used to determine the current mirror gates as required by the rip-up heuristic.

It is possible that a stiff path is running over more than one diode-connected transistor that is part of a current mirror. It has to be decided which transistor is chosen for rip-up in this case. The diode-connected transistor that has more current mirror gates connected to it is chosen as rip-up point. An example is given in Fig. 4.6. The rip-up path is running over $M1$ and $M3$. Both diode-connected transistors are part of a current mirror. The gate of $M1$ is connected to four other current-mirror gates whereas $M3$ is only connected to one other current-mirror gate. The transistor $M1$ is chosen for rip-up by the heuristic as more gates are connected.

The rip-up procedure described in this section is a preparation step for gate shutoff. Gate shutoff is described in the next section.

### 4.3.4. Gate Shutoff by Constraint Programming

The ripped-up circuit computed in the previous section has the property that all currents can be turned off by the gate-shutoff pattern shown in Fig. 4.2 without creating new current paths. Each net has to be assigned to a power-down voltage of *vdd* or *gnd*. A constraint satisfaction problem is formulated for this purpose [Zwerger et al., 2015a].

It constrains the voltages for each net with logic equations that are derived from the circuit and the boundary voltage conditions (supply rails and external signals). A voltage variable $v_n$ is defined for each net $n \in N$. The vector of voltage variables is denoted as $\mathbf{v}$. Each voltage variable can either take a value of $vdd$ or $gnd$ as shown by the following definition:

$$\mathbf{v} \in \{vdd, gnd\}^{|N|} \tag{4.17}$$

The constraint satisfaction problem is given as follows:

$$gateShutoffConstraints(\mathbf{v}) = 1 \tag{4.18}$$

The logic predicate $gateShutoffConstraints(\mathbf{v})$ will be defined in the following based on the voltage boundary conditions and the devices of the circuit. A solution which fulfills (4.18) is denoted as $\mathbf{v}_i^*$. The set of all solutions

$$V = \{\mathbf{v}_1^*, \mathbf{v}_2^*, \ldots, \mathbf{v}_m^*\} \tag{4.19}$$

is computed using a constraint program solver [Christian Schulte and Z.Lagerkvist, 2013].

It is guaranteed that all currents are turned off (A) for each solution. The constraint satisfaction problem (4.18) would have zero solutions if shutting all currents off with gate shutoff would not be possible. However, it is guaranteed that at least one solution exists because the constraint satisfaction problem is only formulated and solved for ripped-up netlists. The number of solutions is denoted as $m$.

The design space for power-down circuits can be explored by computing the complete set of solutions $V$. However, experimental results have shown that $V$ tends to be large. For instance, the number of solutions can easily grow up to a few hundred for circuits with less than 50 devices. A strategy for automatic selection of an optimal solution is presented in section 4.3.6 in order to reduce the number of solutions.

The constraints $gateShutoffConstraints(\mathbf{v})$ can be split into boundary voltage conditions and constraints forcing all currents in the circuit to zero:

$$gateShutoffConstraints(\mathbf{v}) \iff boundary(\mathbf{v}) \wedge noCurrent(\mathbf{v}) \tag{4.20}$$

The boundary conditions $boundary(\mathbf{v})$ are defined as follows:

$$\begin{aligned} boundary(\mathbf{v}) \iff & \bigforall_{v_n \text{ with } n \in N_{gnd}} (v_n = gnd) \\ & \wedge \bigforall_{v_n \text{ with } n \in N_{vdd}} (v_n = vdd) \end{aligned} \tag{4.21}$$

The sets $N_{gnd}$ and $N_{vdd}$ are user inputs. They contain nets which are externally connected to *gnd* and *vdd*.

The constraint $noCurrent(\mathbf{v})$ is derived from the graph model shown in Fig 3.3 (section 3.2.3). The graph consists of nodes $n \in N$ which correspond to the nets of the circuit and directed edges $e \in E$ that correspond to the devices of the circuit. The set of edges $E$ can be partitioned into three subsets for each of the three edge types `d`, `n` and `p`. Those sets are denoted as $E_d$, $E_n$ and $E_p$. They contain all edges of the corresponding type. The variable of the start node $v_{\phi^+(e)}$ and the variable of the end node $v_{\phi^-(e)}$ (see (3.3)) of an edge $e$ are abbreviated as $v_+$ and $v_-$ in the following. A predicate indicating current flow for `d` edges can be defined using the notation above:

$$current(v_+, v_-) \iff (v_+ = vdd) \wedge (v_- = gnd) \tag{4.22}$$

Based on this definition, predicates indicating current flow for `n` and `p` edges can be defined. The variable of the gate/base node $v_{s(e)}$ (see (3.11)) of `n` and `p` edges is abbreviated as $v_{gb}$ for this purpose. Current can only flow if the corresponding transistor is conducting. Consequently, a condition requiring $v_{gb} = vdd$ for `n` edges and $v_{gb} = gnd$ for `p` edges is added. This leads to the definition of the following predicates:

$$nOnAndCurrent(v_+, v_-, v_{gb}) \iff (v_{gb} = vdd) \wedge current(v_+, v_-) \tag{4.23}$$
$$pOnAndCurrent(v_+, v_-, v_{gb}) \iff (v_{gb} = gnd) \wedge current(v_+, v_-) \tag{4.24}$$

A constraint prohibiting current flow is added to $noCurrent(\mathbf{v})$ for each edge of the graph model (see Fig 3.3). This leads to the following definition:

$$\begin{aligned} noCurrent(\mathbf{v}) \iff & \bigwedge_{v_+,v_- \text{ of } e\in E_d} \overline{current(v_+, v_-)} \\ & \wedge \bigwedge_{v_+,v_-,v_{gb} \text{ of } e\in E_n} \overline{nOnAndCurrent(v_+, v_-, v_{gb})} \\ & \wedge \bigwedge_{v_+,v_-,v_{gb} \text{ of } e\in E_p} \overline{pOnAndCurrent(v_+, v_-, v_{gb})} \end{aligned} \tag{4.25}$$

All solutions for which no current is flowing (4.25) and for which the boundary conditions (4.21) are fulfilled represent a valid gate-shutoff variant of the power-down circuit. The switch positions that lead to the net voltages described by a solution $\mathbf{v}_i^*$ need to be computed as next step. This is done by the net dependency analysis described in the next section.

### 4.3.5. Net Dependency Analysis by Constraint Programming

The net dependency analysis described in this section has the goal to compute the final positions of the gate-shutoff switches for a given solution $\mathbf{v}^*$ of the gate-shutoff constraint

satisfaction problem. Net dependencies have to be taken into account while computing the switch positions. For instance, it is sufficient to add a switch to only one of two nets if they are connected by a resistor. The net dependencies can be analyzed with a constraint program. The basic principles of net dependency analysis are introduced in the next subsection.

### Basic Net Dependency Constraint Satisfaction Problem

In the following, the basic net dependency constraint satisfaction problem will be formulated. It is assumed that a solution $\mathbf{v}^*$ of the gate-shutoff constraint satisfaction problem was selected. The net dependencies are analyzed for this particular solution. It needs to be decided for each net if a gate-shutoff switch is added or not. The solution space $\{gnd, vdd\}$ of $\mathbf{v}^*$ is extended as follows:

$$\tilde{\mathbf{v}}^* \in \{Vdd, vdd, Gnd, gnd\}^{|N|} \tag{4.26}$$

The values $Vdd$ and $Gnd$ mean that a switch which pulls the net to the respective voltage is added to the net. The values $vdd$ and $gnd$ mean that the net will take the respective voltage without adding a switch due to dependencies between the nets. Fig. 4.7 illustrates how switches are added. The column "Synthesized Netlist" shows how switches are added to the circuit based on the values of $\tilde{\mathbf{v}}^*$. Furthermore, the visualization of the values $Vdd$, $vdd$, $Gnd$ and $gnd$ is introduced in the column "Visualization". The value $gnd$ is visualized by a blue shaded net. The value $vdd$ is visualized by a red shaded net. Nets that are actively pulled by a switch are marked with a red or blue arrow that is pointing up or down. The value $Gnd$ is visualized by a blue arrow pointing down. The value $Vdd$ is visualized by a red arrow pointing up. The visualization will be used for presenting the experimental results in section 4.4.

The voltage for each net is given as it is assumed that a solution $\mathbf{v}^*$ was selected. The missing part that needs to be computed is the decision if a switch is added. For this purpose, a Boolean variable $n_i$ is defined for each net $n_i$. The variable has the following meaning:

$$\begin{aligned} n_i = 1 &\iff \text{“actively pull net } n_i \text{ with a power-down switch”} \\ n_i = 0 &\iff \text{“do not add a switch to net } n_i\text{”} \end{aligned} \tag{4.27}$$

For the sake of a cleaner notation, no differentiation is made between the net $n_i$ and the corresponding Boolean variable $n_i$. The meaning can be unambiguously derived from the context. Similarly, the set of nets and the corresponding set of Boolean variables are both denoted by the symbol $N$. The Boolean variables $n_i$ are combined in a vector $\mathbf{n}$ as follows:

$$\mathbf{n} \in \{0, 1\}^{|N|} \tag{4.28}$$

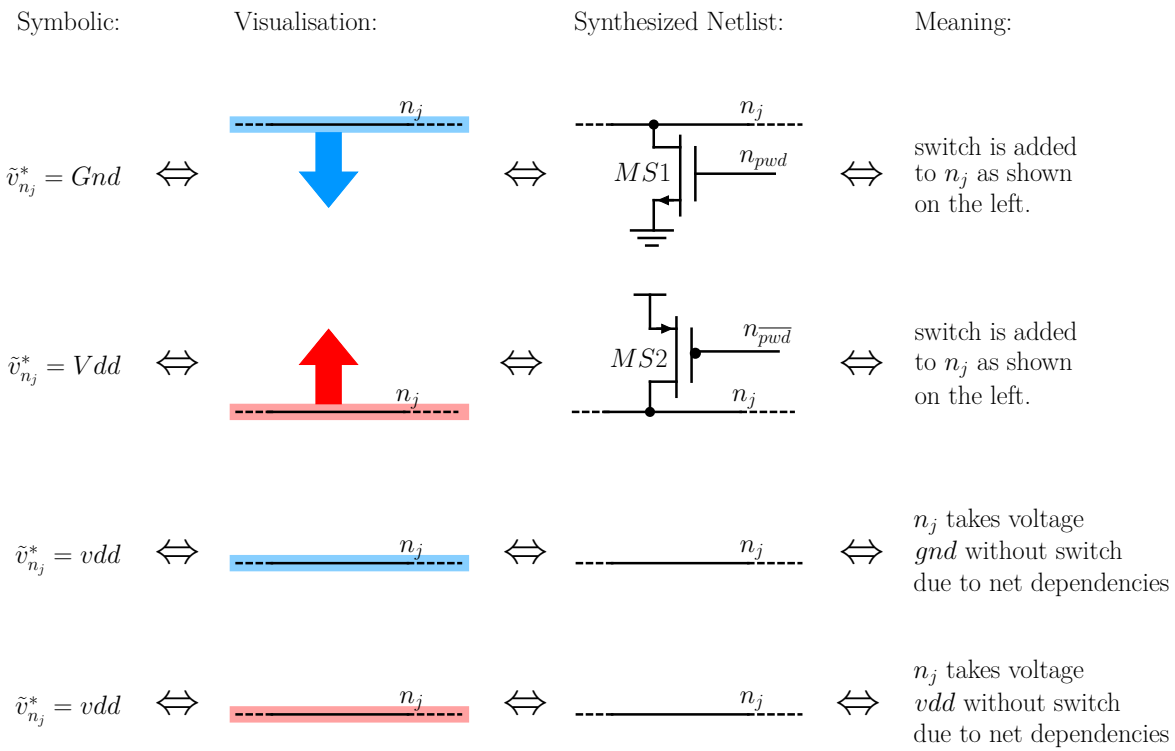



Figure 4.7.: Netlist synthesis [Zwerger et al., 2015a]

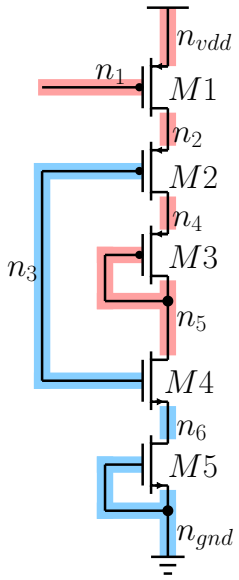
The circuit is transformed to the graph from section 3.2 (see Fig. 3.3) in order to formulate the net dependency constraint satisfaction problem. Fig. 4.8 shows an example circuit. The corresponding graph is drawn to the right of the circuit. Examples of the sets and constraints defined in the following are shown on the bottom of Fig. 4.8. The basic net dependency constraint satisfaction problem is developed in the following. The solution of this constraint satisfaction problem determines the final switch positions. First, the boundary nets $N_{vdd}$ and $N_{gnd}$ need to be taken into account. Nets that are connected by a conducting edge to any of the boundary nets (usually supply and ground) are influenced by the boundary. Consequently, a set of vdd boundary connected nets is defined as follows:

$$N_{vddBnd} = N_{vdd} \cup \{n_i \in \mathrm{N} \mid \text{“}n_i \text{ is forward reachable from any net in } N_{vdd} \text{ along edges that are } on \text{ in solution } \mathbf{v}^*\text{”}\} \tag{4.29}$$

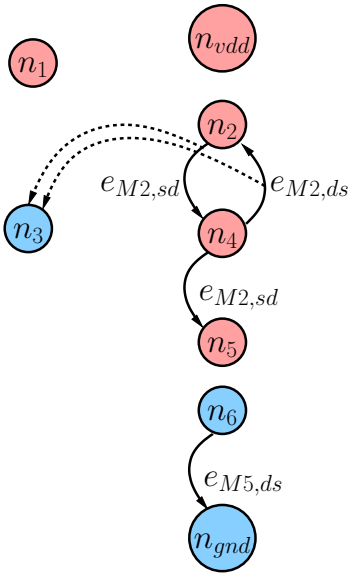
Forward reachable means that a net is reachable following the directed edges along their direction. The set of ground boundary connected nets is defined analogously as follows:

$$N_{gndBnd} = N_{gnd} \cup \{n_i \in \mathrm{N} \mid \text{“}n_i \text{ is backward reachable from any net in } N_{gnd} \text{ along edges that are } on \text{ in solution } \mathbf{v}^*\text{”}\} \tag{4.30}$$

Circuit with selected solution $\mathbf{v}^*$:




Graph (bulk omitted, $off$ edges omitted):




Examples:

$N_{vddBnd} = \{n_{vdd}\}$
$N_{gndBnd} = \{n_{gnd}, n_6\}$
$N_{bnd} = \{n_{vdd}, n_{gnd}, n_6\}$

$constr_{n_1}(\mathbf{n}) \iff \overline{n_1} \leftrightarrow 0$ $\quad R_1 = \emptyset$
$constr_{n_2}(\mathbf{n}) \iff \overline{n_2} \leftrightarrow n_4$ $\quad R_2 = \{n_4\}$
$constr_{n_3}(\mathbf{n}) \iff \overline{n_3} \leftrightarrow 0$ $\quad R_3 = \emptyset$
$constr_{n_4}(\mathbf{n}) \iff \overline{n_4} \leftrightarrow n_2$ $\quad R_4 = \{n_2\}$
$constr_{n_5}(\mathbf{n}) \iff \overline{n_5} \leftrightarrow (n_4 \vee n_2)$ $\quad R_5 = \{n_4, n_2\}$
$constr_{n_6}(\mathbf{n})$ is not needed because $n_6 \in N_{bnd}$.

Figure 4.8.: Examples of reachable nets and $constr_{n_i}(\mathbf{n})$

Backward reachable means that a net is reachable by following the edges counter their direction.

The sets $N_{gndBnd}$ and $N_{vddBnd}$ can be combined to the set of boundary connected nets:

$$N_{bnd} = N_{gndBnd} \cup N_{vddBnd} \tag{4.31}$$

The boundary connected nets can be computed using depth-first search. Examples of $N_{gndBnd}$, $N_{vddBnd}$ and $N_{bnd}$ are given in Fig. 4.8.

Nets that are connected by a conducting edge depend on each other. The direction of the edges and the voltage of the net has to be taken into account in order to determine if nets are connected. The set of reachable nets $R_i$ for net $n_i$ is introduced for this

purpose. The definition of $R_i$ depends on the voltage of $n_i$. Consequently, the sets $R_{i,vdd}$ and $R_{i,gnd}$ are defined and combined to $R_i$ in (4.34).

The set $R_{i,vdd}$ is defined as follows for nets $n_i$ with $v_i^* = vdd$:

$$R_{i,vdd} = \{n_j \in \mathrm{N} \mid \text{“}n_j \text{ is backward reachable from net } n_i \text{ along edges that are } on \text{ in solution } \mathbf{v}^*\text{”}\} \tag{4.32}$$

Backward reachable means again that “$n_j$ is reachable from $n_i$ by running along the edges counter their direction”. This is the direction in which the voltage $vdd$ would propagate from $n_j$ to $n_i$ in the voltage propagation approach from section 3.2. Furthermore, all nets that are backward reachable from a net that has voltage $vdd$ also have the voltage $vdd$ in the solution $\mathbf{v}^*$. This property can be observed in the example shown in Fig. 4.8. It is enforced by the *noCurrent* constraint defined in equation (4.25) in section 4.3.4.

The set $R_{i,gnd}$ is defined as follows for nets $n_i$ with $v_i^* = gnd$:

$$R_{i,gnd} = \{n_j \in \mathrm{N} \mid \text{“}n_j \text{ is forward reachable from net } n_{i,gnd} \text{ along edges that are } on \text{ in solution } \mathbf{v}^*\text{”}\} \tag{4.33}$$

Forward reachable means that “$n_j$ is reachable from $n_i$ running along the edges in accordance with their direction”. This is the direction in which the voltage $gnd$ would propagate from $n_j$ to $n_i$ in the voltage propagation approach from section 3.2. All nets that are forward reachable from a net that has voltage $gnd$ also have the voltage $gnd$ in the solution $\mathbf{v}^*$. This property can also be observed in the example shown in Fig. 4.8. It is enforced by the *noCurrent* constraint (4.25).

The set of reachable nets $R_i$ combines $R_{i,vdd}$ and $R_{i,gnd}$ as follows:

$$R_i = \begin{cases} R_{i,vdd} & \text{if } v_i^* = vdd \\ R_{i,gnd} & \text{if } v_i^* = gnd \end{cases} \tag{4.34}$$

The net constraint for all nets except the nets in $N_{bnd}$ can be formulated using the reachable nets $R_i$. The net constraint is defined as follows:

$$constr_{n_i}(\mathbf{n}) \iff \overline{n_i} \leftrightarrow \bigvee_{n_j \in R_i} n_j \iff n_i \leftrightarrow \bigwedge_{n_j \in R_i} \overline{n_j} \tag{4.35}$$

The first and the second version are equivalent. They can be transformed into each other by applying the law of de Morgan. The first version can be read as follows: “no switch is added to $n_i$ if a switch is added to at least one of the reachable nets”. The equivalent second version can be read as follows: “A switch is added to $n_i$ if no switch is added to any of the reachable nets”. The constraint ensures that each net $n_i$ has a defined voltage, either by directly adding a switch to $n_i$ or indirectly by a switch that is connected to a reachable net.

Equation (4.35) degrades as follows for the special case that $R_i$ is empty:

$$constr_{n_i}(\mathbf{n}) \iff \overline{n_i} \leftrightarrow 0 \iff n_i \leftrightarrow 1 \qquad (\text{for } R_i = \emptyset) \tag{4.36}$$

The constraint says that $n_i$ is 1 if $R_i$ is empty. That means that a power-down switch has to be added in any case.

The overall constrained minimization problem can be formulated as follows using the definitions made so far:

$$\begin{aligned} \min_{\mathbf{n}} \sum_{n_i \in N} n_i \quad \text{s.t.} \quad & \bigvee_{n_i \in N \setminus N_{bnd}} constr_{n_i}(\mathbf{n}) \\ & \bigvee_{n_i \in N_{bnd}} n_i = 0 \end{aligned} \tag{4.37}$$

The number of switches is minimized subject to the net constraint for each net. The boundary connected nets do not need to be pulled as this is already done by an external source.

In general, the optimization problem can have more than one solution with a minimum number of switches. The solutions are denoted as $\mathbf{n}_j^*$ with $j = 1 \ldots m$ and $m$ being the number of solutions. Any solution can be selected. The first solution that was computed by the solver [Christian Schulte and Z.Lagerkvist, 2013] was chosen during the experiments presented in section 4.4.

The selected solution $\mathbf{v}^*$ and the computed solution $\mathbf{n}^*$ can be combined to the final solution $\tilde{\mathbf{v}}^*$ as follows:

$$\begin{aligned} \tilde{v}_i^* = vdd \quad &\iff \quad v_i^* = vdd \wedge n_i^* = 0 \\ \tilde{v}_i^* = Vdd \quad &\iff \quad v_i^* = vdd \wedge n_i^* = 1 \\ \tilde{v}_i^* = gnd \quad &\iff \quad v_i^* = gnd \wedge n_i^* = 0 \\ \tilde{v}_i^* = Gnd \quad &\iff \quad v_i^* = gnd \wedge n_i^* = 1 \end{aligned} \tag{4.38}$$

The netlist synthesis for a given solution $\tilde{\mathbf{v}}^*$ is straightforward as illustrated by Fig. 4.7.

### General Net Dependency Constraint Satisfaction Problem

The basic net dependency constraint satisfaction is insufficient if there are circular dependencies in a circuit. An extension is described in the following in order to resolve circular dependencies.

The basic net constraint (4.35) assumes that a net $n_i$ is dependent on all nets that are reachable via `d`, `n` and `p` edges as specified by the definitions in (4.32) and (4.33). This

is only true under the assumption that the gate net of an `n` or `p` edge (and any net that the gate net depends on) is independent from $n_i$ (and any other net that depends on $n_i$). This assumption is wrong in case of circular dependencies in the circuit. Fig. 4.9 shows an example circuit with a circular dependency. Solving the basic constraint satisfaction problem for this circuit leads to missing switches in the synthesis result.

The circuit in Fig. 4.9 is taken from the core of the voltage controlled oscillator (shown in Fig. 4.18) for which the experimental results are discussed in section 4.4.3. The graph for the example circuit is shown on the top right of Fig. 4.9. The circular dependency can be observed by following the dotted arrows that visualize the switching function $s(e)$ as defined by equation (3.11) in section 3.2.3. The switching function $s(e)$ returns the gate/base net that controls edge $e$. The dotted arrows are pointing from the edge to the gate/base net as defined in Fig. 3.3.

It must be ensured that each edge that is used to reduce the number of switches is conducting. This can be achieved by ensuring that a switch is directly added to the gate/base of this edge or to any net that the gate/base net depends on. Consequently, the dependencies that are introduced by the gate/base nets need to be added to the constraints. This is done in the following. The additional conditions are added to the basic constraint (4.35) to form a new general constraint.

The reachable nets in $R_i$ for net $n_i$ as defined in (4.34) are distinguished by an index $k = 1, \ldots, m_R$ for the purpose of formulating the general constraint. The set $R_i$ can be represented as follows by using index $k$ as running index for its members:

$$R_i = \{n_{(i,1)}, n_{(i,2)}, \ldots, n_{(i,m_R)}\} = \{\ldots, n_{(i,k)}, \ \ldots\} \tag{4.39}$$

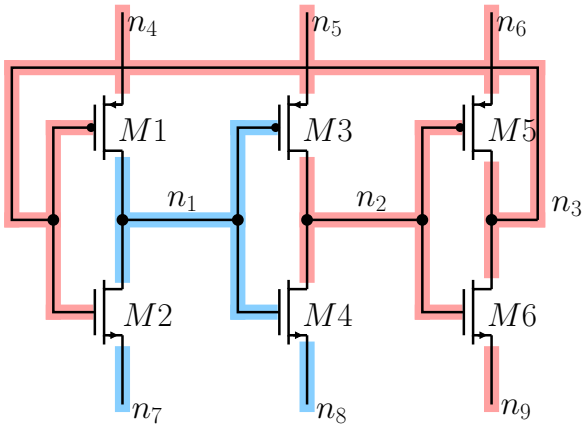
A set of gate/base nets of the `n` and `p` edges on the path from $n_i$ to net $n_{(i,k)}$ can be computed for each reachable net $n_{(i,k)}$. The gate/base nets can be computed by storing the gate/base nets while traversing the graph forward or backward as required by the definitions (4.32) and (4.33). The resulting set of gate/base nets of the edges on the path from $n_i$ to $n_{(i,k)}$ is denoted as $G_{(i,k)}$.

The members of set $G_{(i,k)}$ can be distinguished by using the index $l = 1, \ldots, m_G$. $G_{(i,k)}$ can be represented as follows with index $l$:

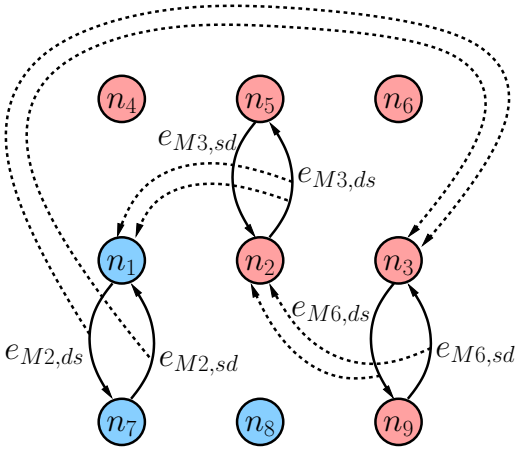
$$G_{(i,k)} = \{n_{(i,k,1)}, n_{(i,k,2)}, \ldots, n_{(i,k,m_G)}\} = \{\ldots, n_{(i,k,l)}, \ \ldots\} \tag{4.40}$$

The recursive general constraint for net $n_i$ can be formulated using $R_i$ and the corresponding sets $G_{(i,k)}$ for each net $n_{(i,k)} \in R_i$. The recursive general constraint is defined by the following equations (4.41), (4.42) and (4.43). An example of the definition is shown in the lower half of Fig. 4.9. It illustrates how $constr_{n_1}(\mathbf{n})$ is built up recursively for the given example circuit. The sets $R_i$ and $G_{(i,k)}$ are noted down for each evaluation of (4.43). The net $n_1$ was arbitrarily chosen. An analogous constraint can be written down for the nets $n_7$, $n_5$ $n_2$, $n_3$ and $n_9$.

Circuit with selected solution $\mathbf{v}^*$:




Graph (bulk omitted, $off$ edges omitted):




Example:

$$constr_{n_1}(\mathbf{n}) \iff \overline{n_1} \leftrightarrow n_7 \wedge recursion_{n_1,n_3}(\mathbf{n}) \qquad R_1 = \{n_7\},\ G_{(1,1)} = \{n_3\}$$

$$recursion_{n_1,n_3}(\mathbf{n}) \iff \overline{n_3} \leftrightarrow n_9 \wedge recursion_{n_1,n_2}(\mathbf{n}) \qquad R_3 = \{n_9\},\ G_{(3,1)} = \{n_2\}$$

$$recursion_{n_1,n_2}(\mathbf{n}) \iff \overline{n_2} \leftrightarrow n_5 \wedge recursion_{n_1,n_1}(\mathbf{n}) \qquad R_2 = \{n_5\},\ G_{(2,1)} = \{n_1\}$$

$$recursion_{n_1,n_1}(\mathbf{n}) \iff 0$$

Figure 4.9.: Example of the recursive general constraint

The recursive general constraint for a net $n_i$ is defined as follows:

$$constr_{n_i}(\mathbf{n}) \iff \overline{n_i} \leftrightarrow \bigvee_{\substack{n_{(i,k)} \in R_i \\ k=1,\ldots,m_R}} \left( n_{(i,k)} \wedge \underbrace{\left( \bigwedge_{\substack{n_{(i,k,l)} \in G_{(i,k)} \\ l=1,\ldots,m_G}} recursion_{n_i,n_{(i,k,l)}}(\mathbf{n}) \right)}_{\text{additional condition}} \right) \quad (4.41)$$

with

$$recursion_{n_i,n_r}(\mathbf{n}) = \begin{cases} 1 & \text{if } n_r \in N_{bnd} \\ 0 & \text{if “net } n_i \text{ is the same net as net } n_r\text{”} \\ constr_{n_i,n_r}(\mathbf{n}) & \text{else.} \end{cases} \quad (4.42)$$

and with

$$constr_{n_i,n_r}(\mathbf{n}) \iff \overline{n_r} \leftrightarrow \bigvee_{\substack{n_{(r,k)} \in R_r \\ k=1,\ldots,m_R}} \left( n_{(r,k)} \wedge \left( \bigwedge_{\substack{n_{(r,k,l)} \in G_{(r,k)} \\ l=1,\ldots,m_G}} recursion_{n_i,n_{(r,k,l)}}(\mathbf{n}) \right) \right) \quad (4.43)$$

where $n_r$ is the gate/base net $n_{(r,k,l)}$ of the previous recursion or the gate/base net $n_{(i,k,l)}$ of the first recursion.

Equations (4.41) to (4.43) are equivalent to (4.35) if the term that is marked as “additional condition” is assumed to be 1. This assumption is true if there are no circular dependencies in the circuit. The experimental results have shown that this is the case for all circuits discussed in section 4.4 except the voltage controlled oscillator. Omitting the recursive part can lead to missing switches in the final solution. Consequently, it is suggested to use the general constraint satisfaction problem for net dependency analysis.

### 4.3.6. Automatic Selection of an Optimal Solution

Complete synthesis results can be computed by using the rip-up method, solving the gate-shutoff CSP for the ripped-up netlist and solving the net dependency CSP for one solution of the gate-shutoff CSP. The three methods can be used to explore the design space for power-down circuits. Experimental results have shown that solving the net dependency CSP for each solution of gate-shutoff CSP leads to many solutions even for smaller circuits. For example, the gate shutoff CSP of the amplifier shown in Fig. 4.16 has 784 solutions [Zwerger et al., 2015a] with multiple net dependency CSP solutions. An automatic procedure to select solutions based on matching conditions and (C) minimum number of switches (E) is described in this section. For this purpose, the gate-shutoff constraint satisfaction problem (4.18) is first extended to an optimization problem which

maximizes the matching. The solutions that lead to a minimum number of switches after solving the net dependency CSP can be chosen in a second step.

The objective function for the optimization problem is derived from the building blocks $b_i \in B_{analog}$ that are recognized with the structure recognition method described in chapter 2 and the matching rules in Table 3.8 in section 3.3.5. The set of matching rules for type $t$ is denoted as $\mathcal{R}_t$. Each matching rule $R \in \mathcal{R}_t$ addresses a pair of pins that needs to have a matched voltage (see Table 3.8). The maximization problem is given as follows:

$$\max_{\mathbf{v}} \sum_{b_i \in B_{analog}} \sum_{R \in \mathcal{R}_{type(b_i)}} match_{b_i,R}(\mathbf{v}) \quad \text{s.t.} \quad gateShutoffConstraints(\mathbf{v}) = 1 \tag{4.44}$$

The constraints $gateShutoffConstraints(\mathbf{v})$ are defined in (4.20) in section 4.3.4. The voltage matching to be maximized is expressed by functions $match_{b_i,R}(\mathbf{v})$. They are defined as follows:

$$match_{b_i,R}(\mathbf{v}): \quad \{vdd, gnd\}^{|N|} \rightarrow \mathbb{N}, \quad \mathbf{v} \mapsto match_{b_i,R}(\mathbf{v}) \tag{4.45}$$

A function $match_{b_i,R}(\mathbf{v})$ is defined for each detected building block $b_i \in B_{analog}$ and for each matching rule $R \in \mathcal{R}_{type(b_i)}$ defined for the type of the building block (Table 3.8). A heuristic that matches first differential pairs ($dp$) and then current mirrors ($scm$, $WSCCM$, $4TCM$, $IWCM$, $CCM$ or $WCM$) is proposed. The rationale behind this heuristic is that the matching of differential pairs is usually more critical than the matching of current mirrors in typical analog circuits. Consequently, the differential pairs need to have a higher weight in the objective function than all current mirrors together. The weight for matching of differential pairs is set to $n_{cm} + 1$ with $n_{cm}$ being the number of detected current mirrors. This leads to the following definition of $match_{b_i,R}(\mathbf{v})$ for building blocks of type $dp$:

$$\begin{aligned} match_{b_i,R}(\mathbf{v}) &:= n_{cm} + 1 \quad \text{if} \quad \text{“matching rule } R \in \mathcal{R}_{dp} \text{ is fulfilled for } b_i\text{”} \\ match_{b_i,R}(\mathbf{v}) &:= 0 \quad \text{if} \quad \text{“matching rule } R \in \mathcal{R}_{dp} \text{ is violated for } b_i\text{”} \end{aligned} \tag{4.46}$$

The weight for current mirrors is set to one. The function $match_{b_i,R}(\mathbf{v})$ is defined as follows for current mirrors:

$$\begin{aligned} match_{b_i,R}(\mathbf{v}) &:= 1 \quad \text{if} \quad \text{“matching rule } R \in \mathcal{R}_t \text{ with type } t \text{ being any current mirror is fulfilled for } b_i\text{”} \\ match_{b_i,R}(\mathbf{v}) &:= 0 \quad \text{if} \quad \text{“matching rule } R \in \mathcal{R}_t \text{ with type } t \text{ being any current mirror is violated for } b_i\text{”} \end{aligned} \tag{4.47}$$

The heuristic given by (4.46) and (4.47) was used for generating the experimental results presented in section 4.4. The number of solutions $|V|$ (see Fig 4.3) is reduced drastically

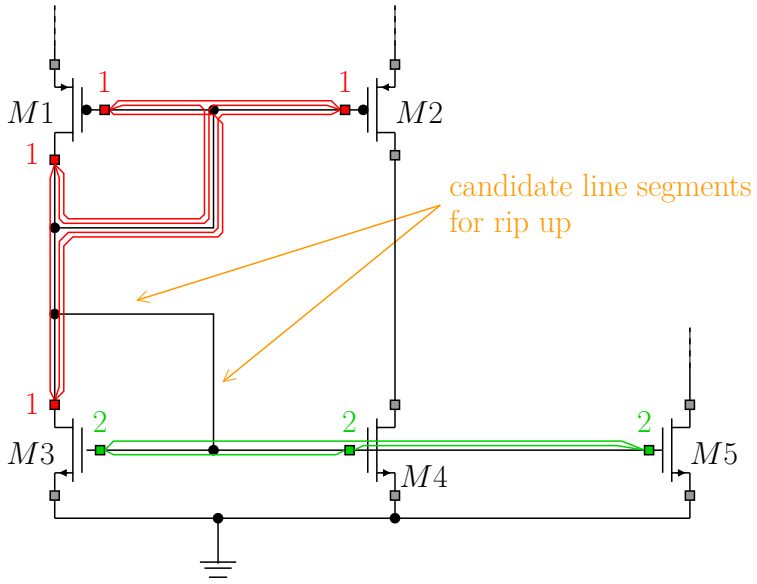



Figure 4.10.: Example for current mirror rip-up back annotation

by using the heuristic. The net dependency analysis described in the previous section is carried out for all remaining solutions. The result is the set $\tilde{V}$ (see Fig 4.3). The number of switches is known after net dependency analysis. The solutions $\tilde{\mathbf{v}}^* \in \tilde{V}$ that have the minimum number of switches are further processed by the netlist synthesis.

### 4.3.7. Schematic Back Annotation

The methods described so far synthesize the power-down circuitry for a given circuit. In state-of-the-art design flows the circuits are given as schematics. Schematics do not only contain the structural information given by a netlist. They additionally contain a graphical representation of the circuit. The synthesis results should be annotated back into an existing schematic in order to integrate the synthesis method into state-of-the-art design flows.

An incremental approach that adds the synthesized switches to an existing schematic is suggested [Neuner, 2015]. Synthesized schematics that were generated by this approach are shown in Figs 4.13, 4.15 and 4.20 (section 4.4).

The steps carried out by the schematic synthesis method can be summarized as follows:

1. Identify and remove a line segment that is eligible for rip-up as illustrated by Fig 4.10 (description see below).
2. Place copies of predefined templates of the switches (see Fig 4.11) into the schematic. The rip-up templates are placed at the positions of the line segments that were iden-

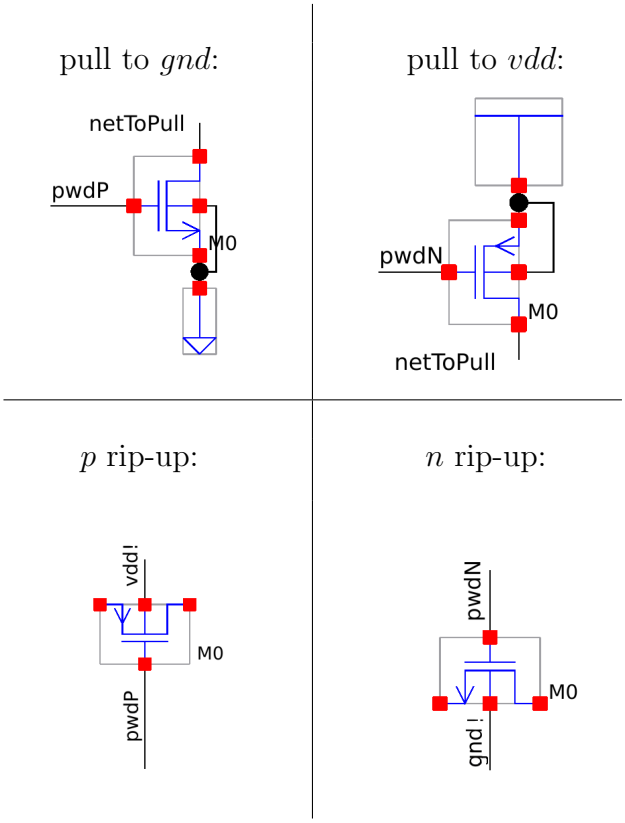



Figure 4.11.: Open Access template cell views used for schematic back annotation

tified and removed in the first step. The gate-shutoff switches are placed next to existing line segments and connected by a small additional line segment.
3. Place the switches next to the existing schematic elements and connect them with a "connection by name" scheme if placing is not possible.

The identification of line segments eligible for rip-up is based on the principle of identifying paths between all pins that are connected to the same net. The principle is illustrated in Fig 4.10. The pins marked with a red 1 (group 1) need to stay connected to the original net. The pins marked with a green 2 (group 2) need to be connected to a newly synthesized net. The partitioning of the pins of the rip-up net into group 1 and group 2 is determined by the rip-up heuristic. Fig 4.10 shows the partitioning that is the result of current-mirror rip up of the diode transistor $M3$. All paths between all pins in group 1 and all paths between the pins in group 2 are drawn. The paths can be computed by performing a depth-first search on the line segments. Line segments eligible for rip-up can be identified based on the computed paths. A line segment is eligible for rip-up if none of the paths is passing it. There is no guarantee that such a line segment exists for every schematic. However, the back annotation can resort to a "connection by name" scheme in this case.

## 4.4. Experimental Results

The experimental results for the power-down synthesis method are discussed in this section. Results for a symmetrical BiCMOS amplifier are presented as an example of a circuit that contains bipolar transistors (section 4.4.1). Sections 4.4.2 and 4.4.3 show results for two CMOS amplifiers. The first amplifier is a fully differential amplifier with current mode feedback. The second amplifier is a folded cascode amplifier with biasing part. Results for a voltage controlled oscillator are shown in section 4.4.4. The oscillator has circular dependencies that need to be taken into account during net dependency analysis as described in section 4.3.5. Section 4.4.5 presents results for a low voltage differential signalling (LVDS) driver. Experimental results for switch sizing including simulation results for the synthesized power-down circuitry are discussed in section 4.4.6. The experiment for the LVDS driver is an example of synthesizing the power-down circuitry for a hierarchical design.

The synthesis solutions will be annotated using the notation in Fig 4.7. All experimental results were generated using the rip-up procedure from section 4.3.3, the gate-shutoff CSP from section 4.3.4, the general net dependency CSP from section 4.3.5 and the heuristic for automatic selection of results from section 4.3.6. The algorithms were implemented in C++. The software tool has a command line interface as well as graphical user interface. The input schematics for schematic back annotation were generated using the Cadence Virtuoso Design Environment. They are stored in an Open Access [Si2 Consortium, 2013] database. The synthesis tool directly reads the Open Access

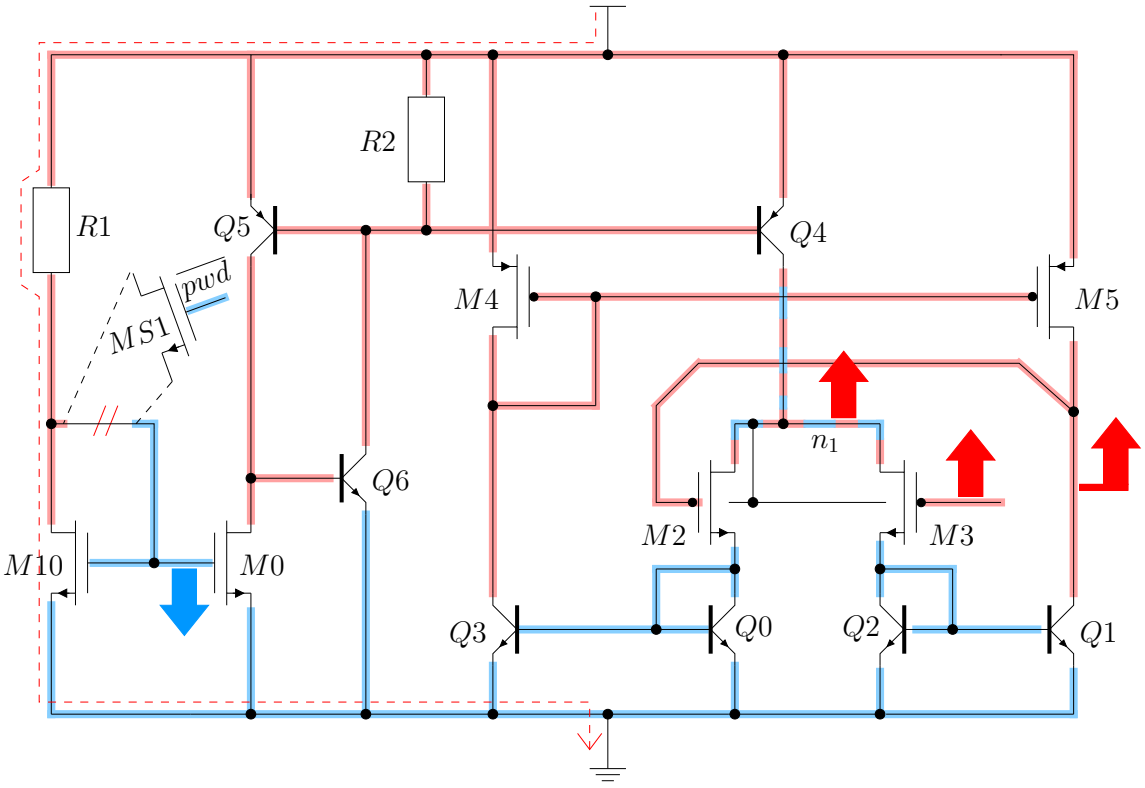



Figure 4.12.: Illustration of the two solutions for the symmetrical BiCMOS OTA. Net $n_1$ can either be pulled up or down (notation see Fig 4.7).

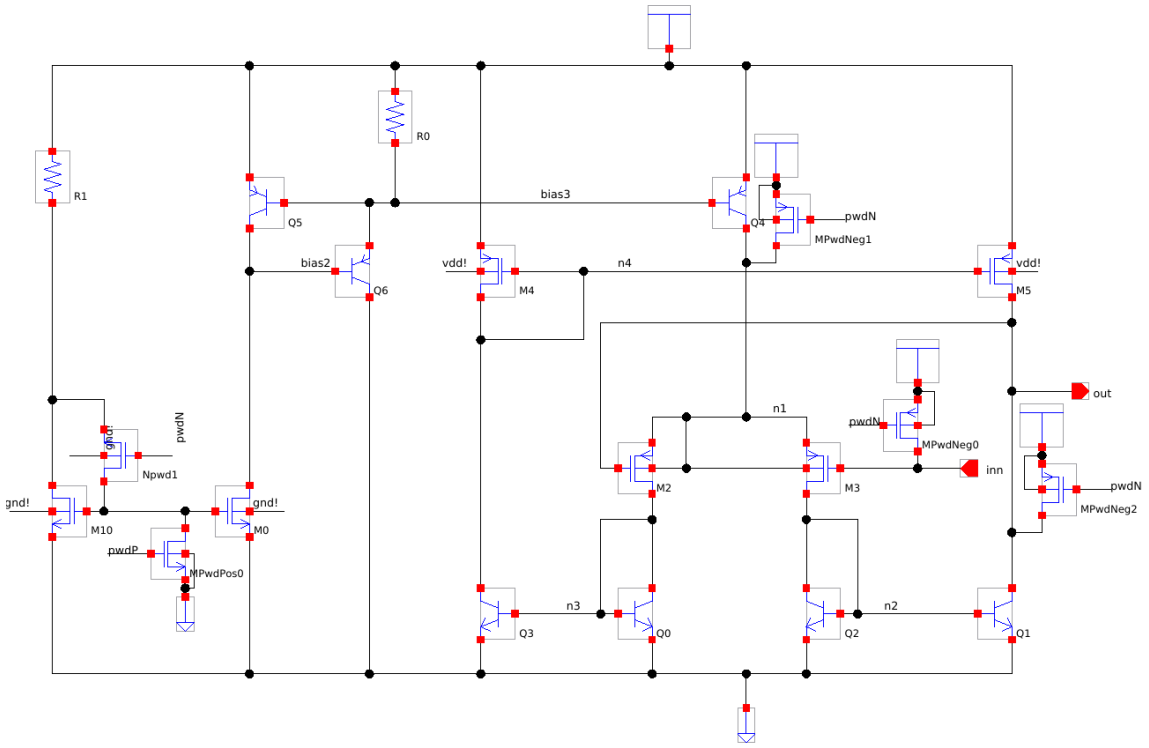



Figure 4.13.: Schematic with synthesized switches for the symmetrical BiCMOS OTA

database or it reads SPICE netlists that were exported with a netlister. All experiments were done on a standard workstation with an Intel(R) Core(TM) i7-4770 CPU running at 3.40 $GHz$. The possibilities of parallel computing were not exploited by the implementation.

The synthesized circuits were checked for correctness using the voltage propagation method from section 3.2. It was verified that all synthesized power-down circuits do not have floating nodes or short circuits.

The power-down switches were annotated back into an existing schematic for all circuits except the hierarchical LVDS circuit. The method from section 4.3.7 was used for this purpose. The resulting schematics are shown in the following subsections. They were written back to the Open Access database to be accessible by other design tools, e.g. the Cadence Virtuoso Design Environment.

### 4.4.1. Symmetrical BiCMOS Amplifier

Fig. 4.12 shows the BiCMOS amplifier that is discussed in this section. It is configured in voltage follower configuration, i.e., the inverting input and the output are connected. One stiff path from vdd over $R1$ and $M10$ to ground was detected. The path is visualized by the red dashed line. The computed rip-up point is indicated by two red lines "slicing up" the line segment that is replaced by the switch $MS1$. The gate-shutoff CSP for the BiCMOS amplifier has five solutions [Zwerger et al., 2015a]. Two solutions with maximum matching and minimum number of switches remain after applying the heuristic for automatic selection of solutions (see section 4.3.6). One pull-down switch and three pull-up switches are added to the circuit in both solutions. This is indicated by the corresponding blue and red arrows. The solutions differ for net $n_1$. It can either be pulled up or down. This is indicated by the alternating coloring of the net. Net $n_1$ can be considered to be a floating net because $Q4$, $M2$ and $M3$ are not conducting. The switch for this net can also be omitted to save a switch if goal (D) of the power-down synthesis problem is followed in a loose way. The computation took approximately one second.

Fig. 4.13 shows the schematic with back annotated switches. It shows the line segment where the rip-up switch was inserted. It also shows the other four gate-shutoff switches placed next to the line segments.

### 4.4.2. Fully Differential Amplifier with Current Mode Feedback

Fig. 4.14 shows the result for the fully differential amplifier with current mode feedback. The circuit has one stiff path from vdd over $R1$ over $M26$ to ground. The stiff path is marked with a red dashed line. The computed rip-up point is the diode connection of $M26$. A rip-up switch $MS1$ is added.

The gate-shutoff CSP (see section 4.3.4) has 260 solutions. One solution with maximized matching and a minimized number of switches remains after applying the heuristic for automatic selection of solutions (section 4.3.6). The solution is annotated in Fig. 4.14. Seven pull-up gate-shutoff transistors (red arrows) and three pull-down gate-shutoff transistors (blue arrows) are added. The boundary conditions for the synthesis only contained the vdd supply net and the ground net. No boundary conditions for $in+$, $in-$, $out+$, $out-$ and $cm$ were set. Those nets can be added to the boundary conditions to compute a solution adhering to those requirements if the specification of the external circuitry connected to those pins requires different voltages. For instance, the nets $in+$ and $in-$ were added to $N_{gnd}$ in another experiment. The computation then yields another solution with $in+$ and $in-$ having the voltage $gnd$. The synthesis for the fully differential amplifier with current mode feedback takes about one second computation time.

The schematic with back annotated switches of the solution in Fig. 4.14 is shown in Fig. 4.15. It shows the ripped-up stiff path and the placed gate-shutoff switches.

### 4.4.3. Folded Cascode Amplifier with Biasing Circuitry

Fig. 4.16 shows the synthesis result for the folded cascode amplifier with biasing circuitry. One stiff path from vdd over $M15$, $M17$ and $M19$ to ground was computed. It is marked with a red dashed line. Two diode connected transistors that are part of a current mirror ($M17$ and $M19$) can be found on this path. The rip-up heuristic chooses the diode transistor with more connected current mirror gates for rip up (see section 4.3.3). $M17$ forms current mirrors with $M24$, $M25$, and $M26$. Hence, it is connected to three other current mirror gates. $M19$ forms a current mirror with $M20$, $M21$, $M22$ and $M23$. Hence, it is connected to four other current mirror gates. Consequently, $M19$ is connected to more gates and its diode connection is chosen for rip-up.

The gate-shutoff CSP has 784 solutions [Zwerger et al., 2015a]. Eight solutions with a gate-shutoff switch count of 10 remain after automatic selection of solutions as described in section 4.3.6. The remaining eight solutions differ in the voltages of the nets $n_1$, $n_2$, $n_3$ and the input nets $in+$ and $in-$. This is illustrated in Fig. 4.16 by the alternating coloring. The annotated example solution adds nine pull-up gate-shutoff transistors (red arrows) and one pull-down gate-shutoff transistor (blue arrow) to the circuit.

Specifying boundary conditions for $in+$, $in-$ and $out$ can further reduce the number of solutions if they are given. Matching requires $n_1$, $n_2$ and $n_3$ to be $Vdd$ as in the annotated solution. A closer inspection of the result for this circuit yields that ($M9$, $M10$, $M11$, $M12$) was recognized as wide swing cascode current mirror. Consequently, the matching between $n_1$ and $n_2$ is enforced for all solutions. However, the matching between ($n_1$, $n_2$) and $n_3$ as well as the matching between $n_4$ and $n_3$ is not enforced. Structure recognition with library $L_{analog}$ does not generate all required matching constraints.

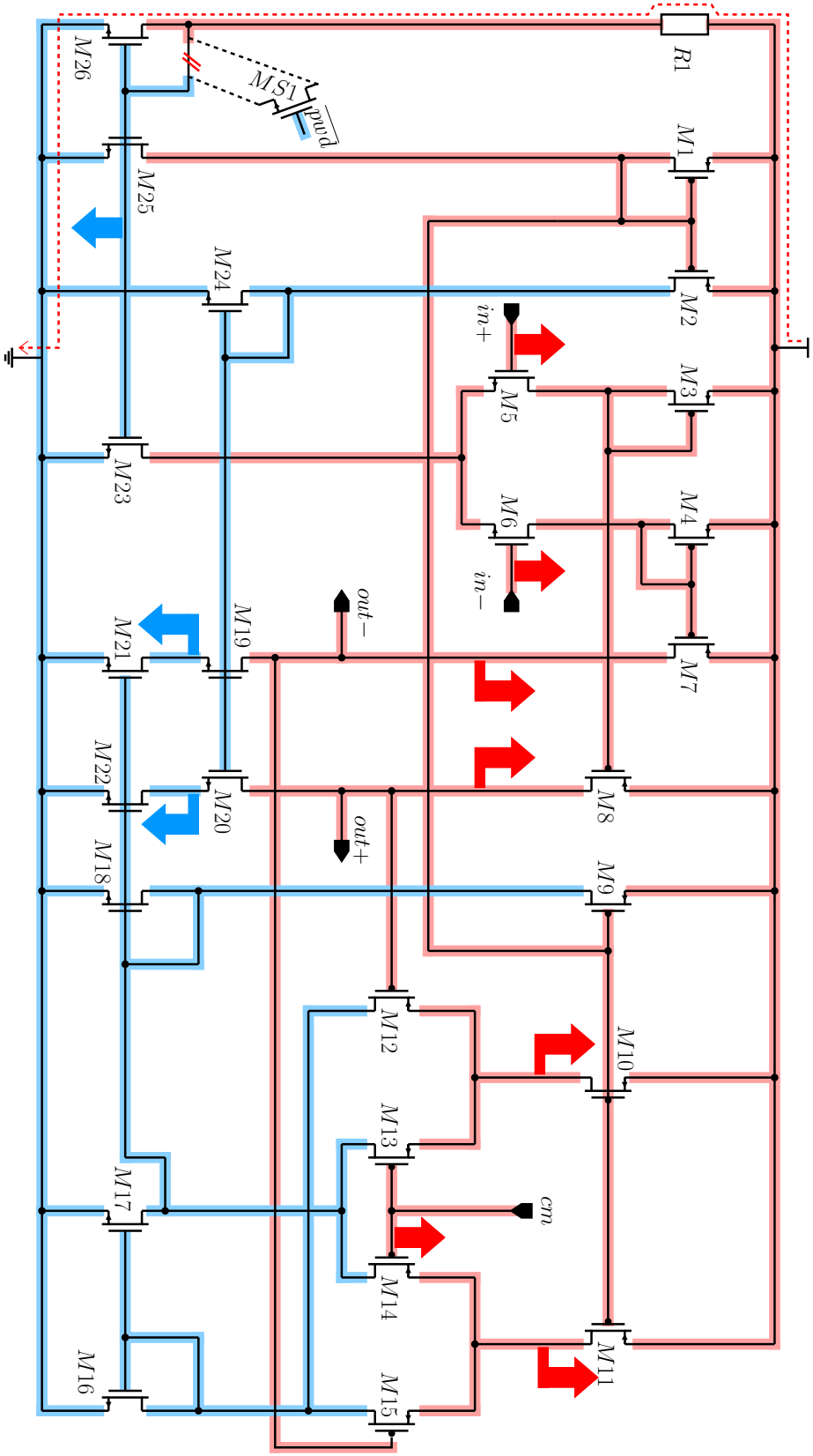



Figure 4.14.: Synthesis result for the fully differential amplifier with common mode Feedback (notation see Fig 4.7)

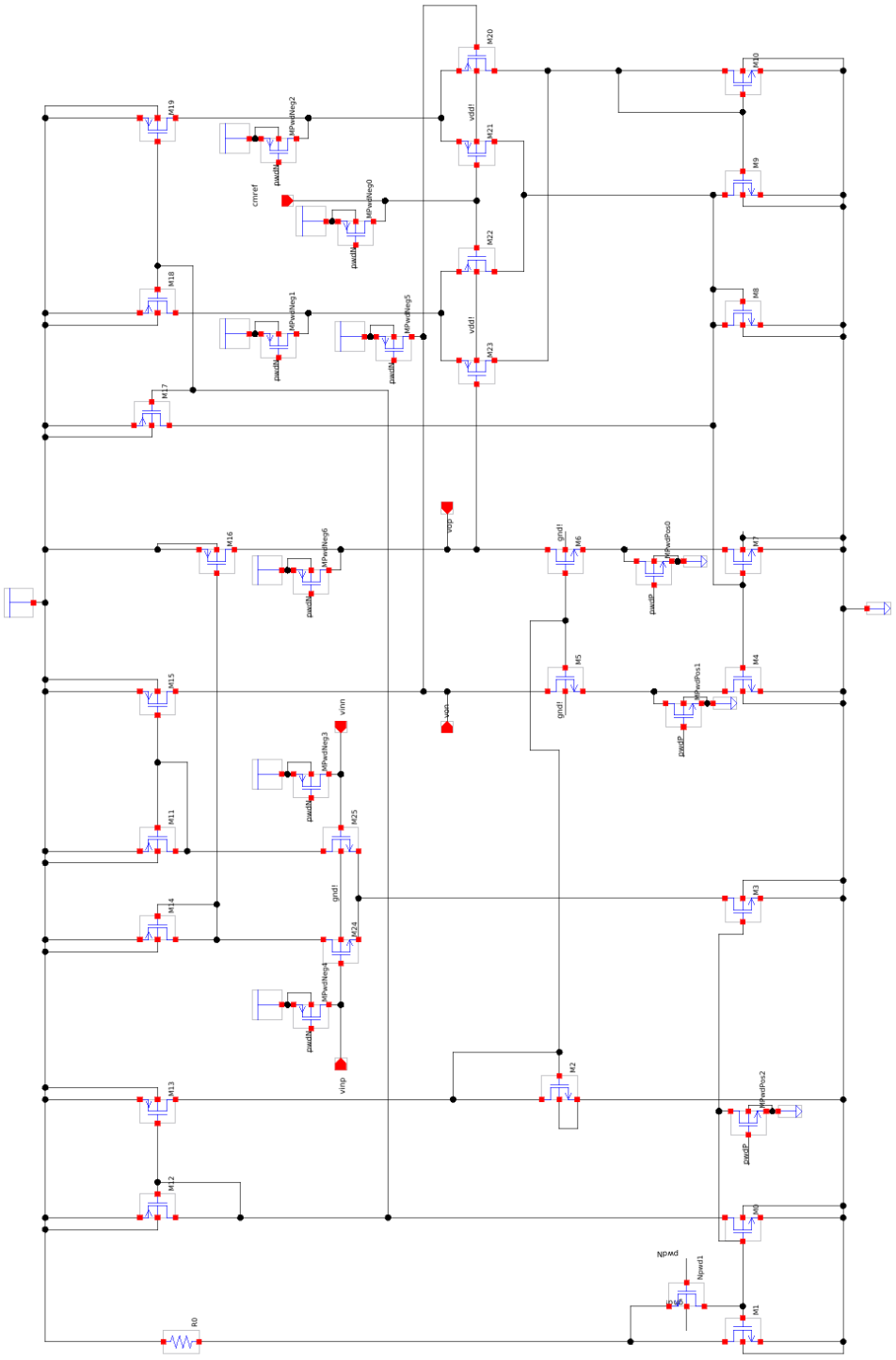


Figure 4.15.: Schematic with synthesized switches for the fully differential amplifier

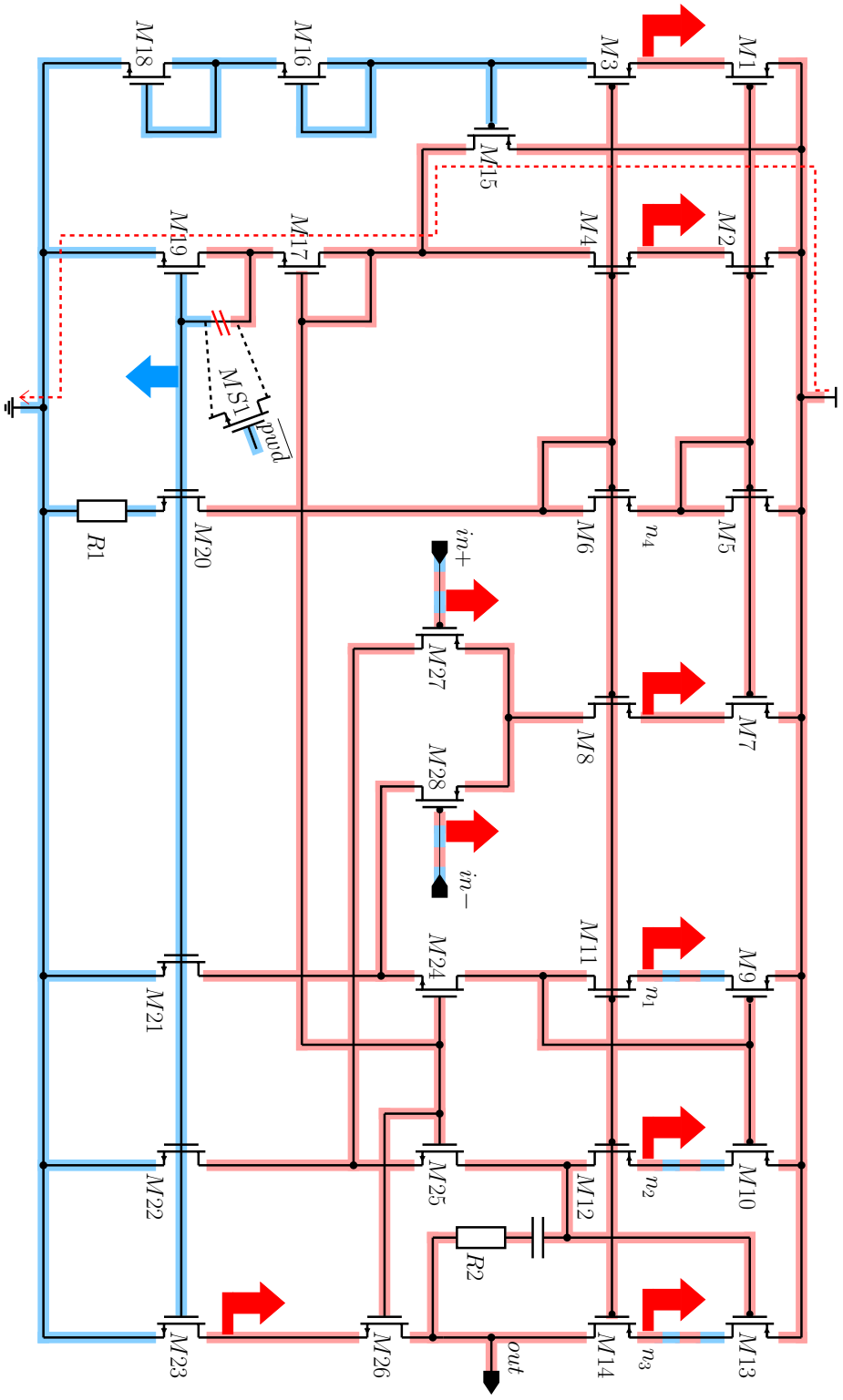



Figure 4.16.: Synthesis result for the folded cascode amplifier with biasing circuitry (notation see Fig 4.7)

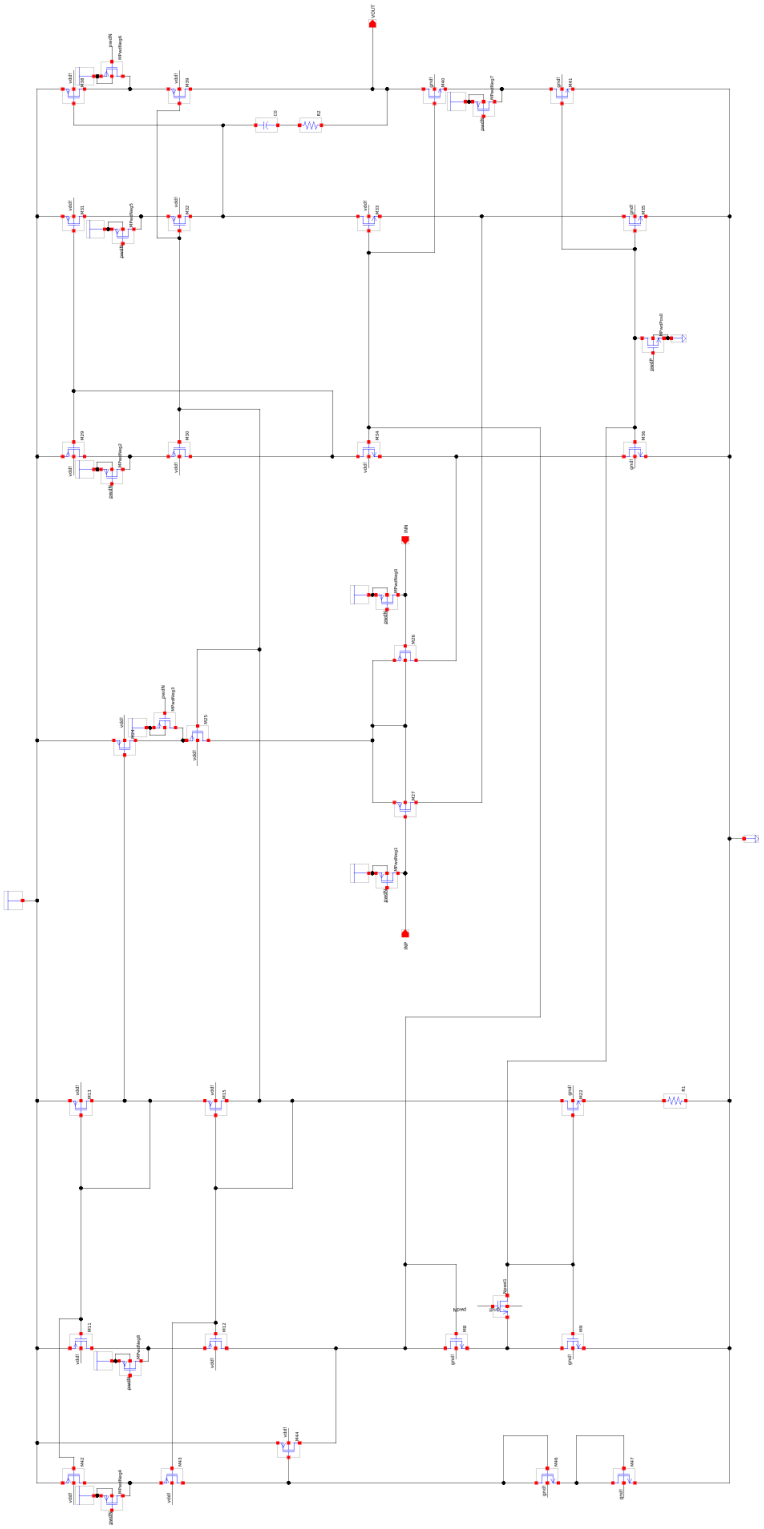


Figure 4.17.: Schematic with synthesized switches for the folded cascode amplifier

The missing constraint between $n_4$ and $n_3$ is generated if the structure recognition is computed with a modified library where the bias shifter ($bs$) has a maximum persistence and a dominance relation for current mirrors is applied (as described for $L_{analog'}$ in section 4.3.3) This leads to half the number of final solutions (four solutions). However, the matching between ($n_1$, $n_2$) and $n_3$ is still not enforced for those solutions. It is suggested that the gate-net connections of the current mirrors are taken into account to establish "transitive matching" in future work.

The computation of the synthesis result shown in Fig. 4.16 took approximately one second. Fig. 4.17 shows the schematic with back annotated switches of the solution shown in Fig. 4.16. It shows the placements of the gate-shutoff transistors and the rip-up transistor.

### 4.4.4. Voltage Controlled Ring Oscillator

The voltage controlled ring oscillator used for the experiment in this section is shown in Fig. 4.18. It can be turned off by pulling the input pin *cntrl* to vdd. Consequently, no stiff path is present in this circuit and no rip-up is required. The gate shutoff CSP has 64 solutions [Zwerger et al., 2015a]. Six solutions with maximum matching and minimum power-down switch count remain after applying the heuristic for automatic selection of solutions (section 4.3.6). The solutions are shown in Fig. 4.19. The experiments show that using the general net dependency CSP (see section 4.3.5) leads to a correct power-down mode implementation despite the circular dependency caused by the feedback of the ring oscillator.

The computation for the voltage controlled oscillator took about one second. A schematic with the back annotated switches of solution 1 from Fig. 4.19 is shown in Fig. 4.20. Fig. 4.20 shows the placement of the switches into the existing schematic.

### 4.4.5. Low Voltage Differential Signaling Driver

The experiments described in sections 4.4.1 to 4.4.4 demonstrate how the methods described in this chapter can be used in a semi-automatic design flow to implement power-down circuits on the lowest hierarchy level of a design. An experiment that demonstrates the power-down synthesis for a hierarchical design is discussed in this section. The power-down circuitry was synthesized for the low voltage differential signaling (LVDS) driver that is shown in Fig. 4.21. The circuit is the same as the circuit denoted as "LVDS ($0.13\mu$)" in Table 3.11. The power-down circuitry was manually removed and added again by automatic synthesis. The LVDS circuit without power-down circuitry comprises 183 components and 102 nets (see Fig. 4.21). It has an input pin *data* that receives a digital data bit. The bit is transmitted on a wire pair connected to *out*+ and

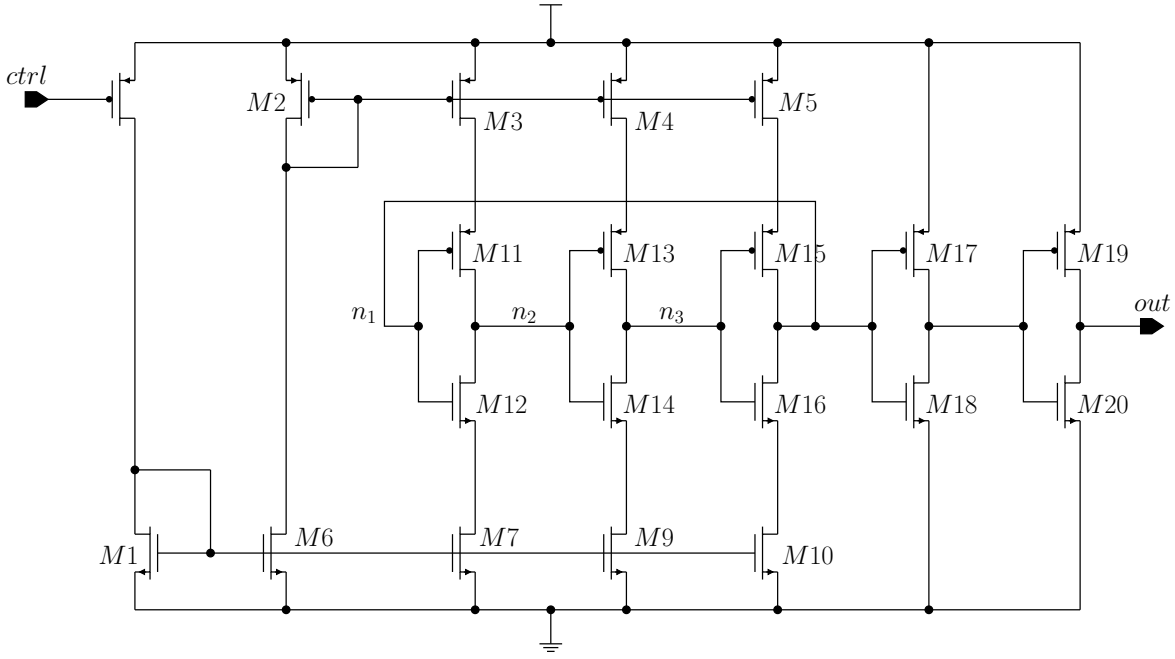



Figure 4.18.: Schematic of the voltage controlled ring oscillator (synthesis result see Fig. 4.19)

*out*−. Furthermore, the circuit receives a reference voltage on pin $v_{ref}$. The voltage is used to regulate the common mode voltage of the wire pair.

The LVDS circuit was read from a spice netlist. One rip-up path was computed and ripped up. The synthesized solutions were written out as a set of spice netlist files. The number of solutions after applying the heuristic for automatic selection of solutions (section 4.3.6) was 24. The number of gate-shutoff switches was 20 for all solutions. It could be verified that none of the 24 synthesized circuits had any short circuit paths or floating nodes by using the voltage propagation approach from section 3.2. The total computation time for computing the 24 solutions was 12 seconds. The right side of Fig. 4.21 gives an overview of the time spent for the individual steps of the synthesis.

It can be seen that the algorithms effectively and efficiently synthesize the power-down circuitry for the LVDS design. The author did not have access to bigger (industrial) circuit examples at the time doing the experiments. Investigating the runtime for bigger circuits remains for future research.

### 4.4.6. Switch Sizing

The power-down synthesis method described so far does not take into account switch sizing. However, the transistor sizes, i.e. the channel widths and lengths, have to be determined to complete the netlist. Sizing has to follow the goals (A) to (E) formulated

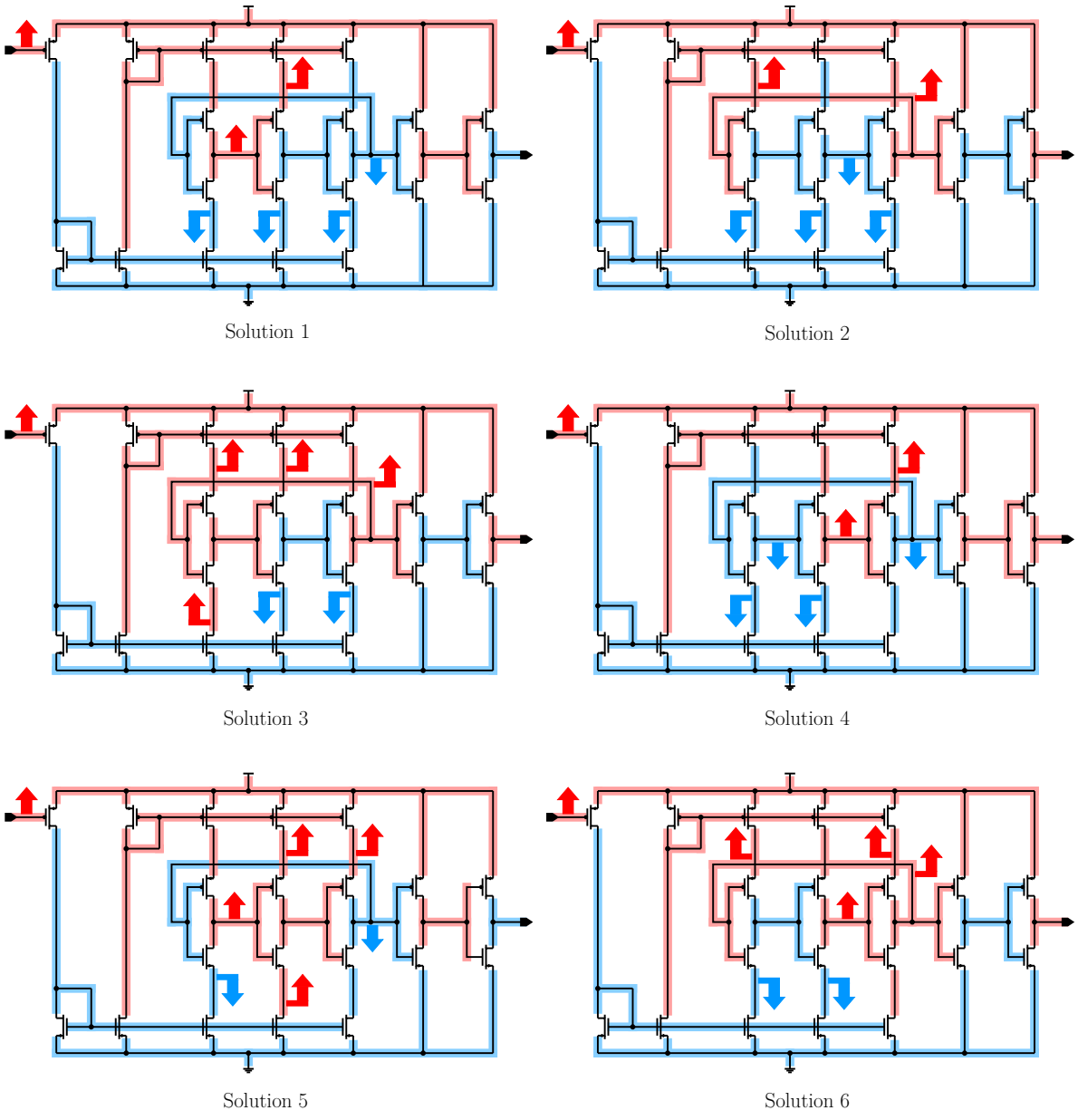



Figure 4.19.: Synthesis result for the voltage controlled ring oscillator (notation see Fig 4.7).

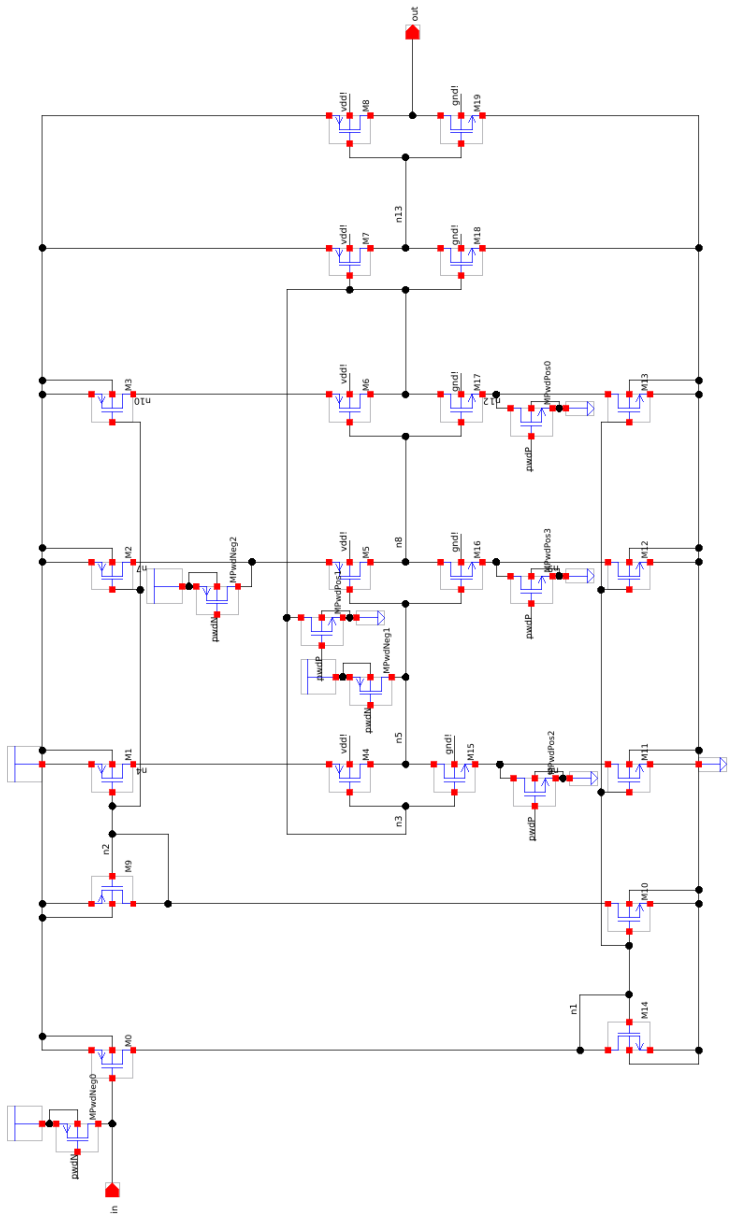
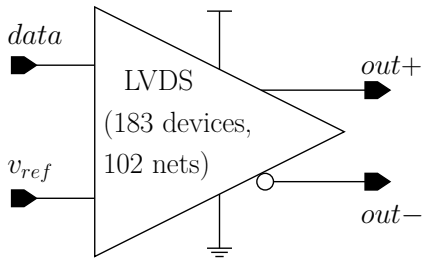



Figure 4.20.: Schematic with synthesized switches of solution 1 for the voltage controlled oscillator




| | |
|---|---|
| Rip-Up: | 1 $s$ |
| Gate shutoff: | 8 $s$ |
| Net dependency analysis: | 1 $s$ |
| Overhead (e.g. read/write files): | 2 $s$ |
| Total runtime: | 12 $s$ |

Figure 4.21.: Symbol of the LVDS driver and runtimes for power-down synthesis

for the power-down synthesis problem in section 4.2. The heuristics for the structural synthesis of the power-down circuitry have been carefully chosen to fulfill those goals as much as possible. Serial rip-up is avoided as much as possible (see section 4.3.1). The results presented in this section show that serial-up is avoided by the rip-up heuristic very effectively. Consequently, the goal (B) ("no change in power-up behavior") is fulfilled very well for all circuits. This is confirmed by case studies that investigate the quantitative behavior of the synthesized power-down networks [Shrivastava, 2016]. The case studies were done with the transistor models of the Austria Microsystems 350 $nm$ technology. The minimum transistor size for this technology is $W = 400\ nm$ and $L = 350\ nm$. The examined circuits were a Miller amplifier, the folded cascode amplifier shown in Fig. 4.16, the amplifier with common mode feedback shown in Fig. 4.14, and the voltage controlled oscillator shown in Fig. 4.18. The power-up performances of the circuits were determined by numerical circuit simulation. Additionally, the power consumption in power-down mode was determined. Furthermore, the transient behavior of the transitions from power-up to power-down mode and vice versa were examined.

The simulation results can be summarized as follows [Zwerger et al., 2016; Shrivastava, 2016]:

- The power-up behavior is influenced the least if the minimum size is chosen for all power-down switches.
- The power-up performance deviation caused by the power-down circuitry was mostly between $-0.1$ % and $+0.1$ % if the minimum size was chosen for all power-down switches. Two exceptions were observed: The phase margin of the Miller amplifier and the folded cascode amplifier changed by $-3.18$ % and $-1.25$ %, respectively.
- The DC as well as the transient power-consumption reached a value near zero for all synthesized power-down networks.
- The highest encountered settling time for the transition from power-down to power-up at minimum switch size was less than 1 $\mu s$. The down to up settling time is insensitive to the size of the switches. It is mainly determined by the circuit itself.
- The highest encountered settling time for the transition from power-up to power-down at minimum switch size was less than 1.5 $\mu s$. The up to down settling time is influenced by the size of the switches. However, the up to down time is

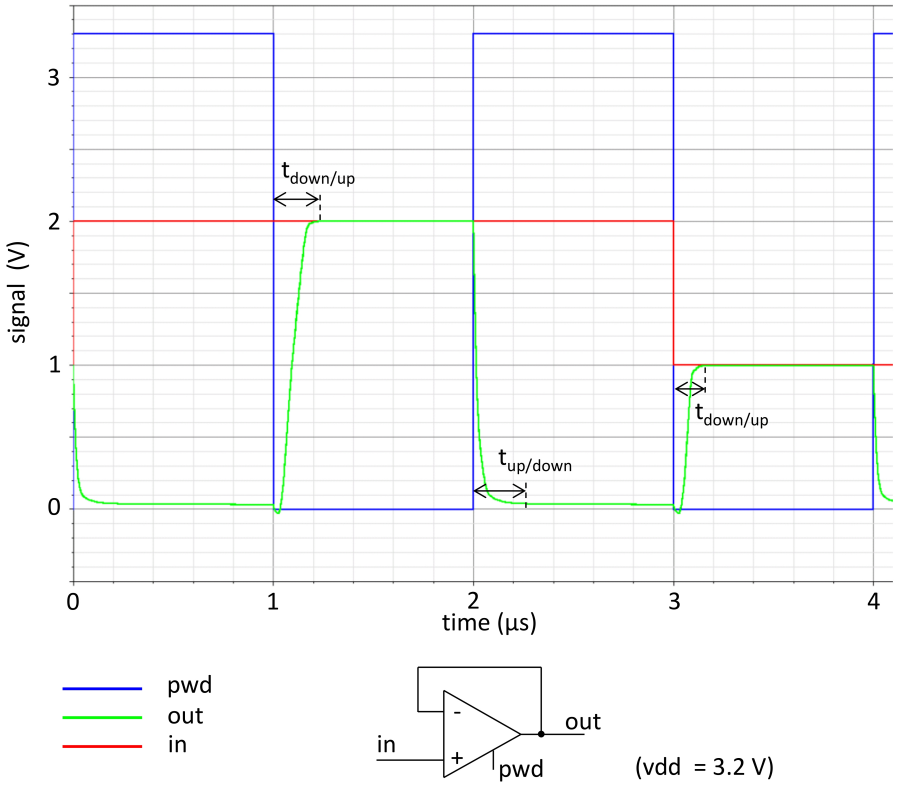



Figure 4.22.: Example simulation result for the folded cascode amplifier with synthesized power-down circuitry [Zwerger et al., 2016; Shrivastava, 2016]

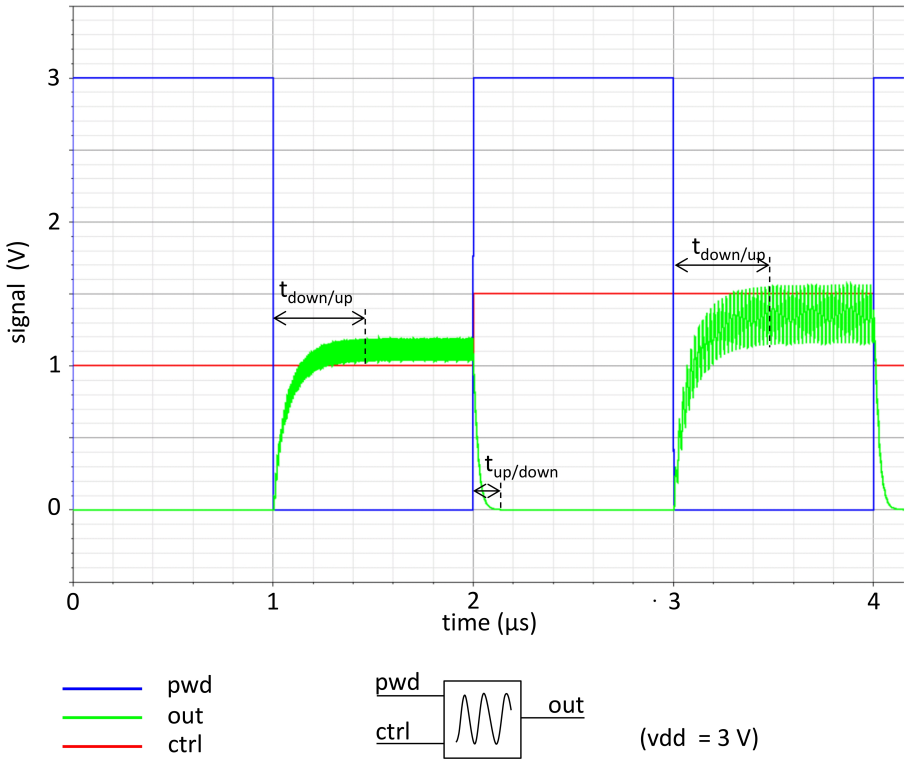



Figure 4.23.: Example simulation result for the voltage controlled oscillator with synthesized power-down circuitry [Zwerger et al., 2016; Shrivastava, 2016]

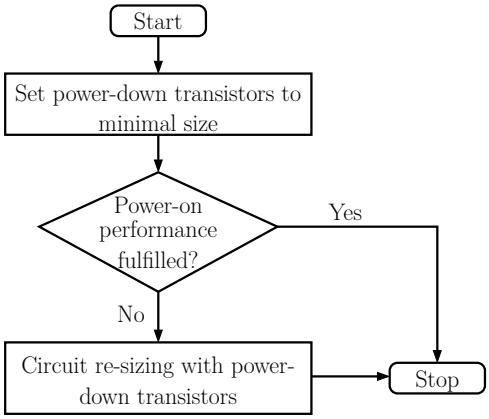



Figure 4.24.: Proposed flow for switch sizing

less important than the down to up time because it determines the latency for activating the system.

Figs. 4.22 and 4.23 show examples of transient simulation waveforms. The settling times of the transitions from power-up to power-down and vice versa are marked with $t_{up/down}$ and $t_{down/up}$.

The waveform for the folded cascode amplifier is shown in Fig 4.22. The folded cascode amplifier was simulated in voltage follower configuration. Hence, the green output signal follows the red input signal in power-up mode. The output is pulled to ground in power-down mode.

The waveform for the voltage controlled oscillator is shown in Fig 4.23. The oscillation frequency of the green signal at the output pin *out* changes in power-up mode according the red control voltage *ctrl*. The output is pulled to ground in power-down mode.

The case studies confirm that the goals (A) ("all currents off") and (B) ("no change in power-up behavior") of the power-down sizing problem are fulfilled very well for the used technology and examined example circuits. Furthermore, the case studies show that choosing the minimum length and width for the switches leads to satisfying results in most of the examined cases. The case study showed one case in which a power-up performance specification was not met anymore after adding the synthesized power-down circuitry. Adding an automatic power-up performance optimization step led to a circuit that fulfilled the performance again. Therefore, a sizing flow like shown in Fig. 4.24 is suggested. First, the minimum switch sizes of the given technology are chosen for the power-down switches. The case studies suggest that in most cases this will lead to a circuit that still fulfills the power-up specifications. A subsequent optimization step (e.g. [Gielen et al., 1990; Medeiro et al., 1994; Antreich et al., 2000]) with power-down switches in place can be added for the cases where the specification is not met. In future work, this flow has to be confirmed for other technologies and more circuits.

# 5. Conclusion

Modern systems on chip (SoCs) integrate many different functionalities on a single IC. Most functions are implemented with digital circuits. However, powerful analog and mixed-signal circuits are needed for interaction of the systems with the outer environment. While design automation for digital is highly automated, analog design still requires a high portion of manual design effort. In the introduction of this thesis, it was shown that verification of the analog/mixed-signal systems is a major bottleneck in the chip development process.

Modern systems on chip comprise many different circuit blocks on a single chip. Only a subset of the blocks is active at the same time for typical use cases. Hence, blocks that are currently not required are powered down. The interaction of sub systems in power-down mode and subsystems in operation mode introduces possibilities for subtle design errors and, hence, new challenges for chip verification. This is also true for analog circuit blocks. In this thesis, it was pointed out that floating nodes make numerical simulation an unreliable choice for verification. Instead, structural approaches are proposed.

In order to contribute to closing the analog verification bottleneck in chip design, this thesis proposes new methods for analog power-down mode verification and synthesis. The main contributions of this thesis can be summarized as follows:

- A new structural voltage propagation approach (section 3.2) is proposed in order to avoid the limitations of numerical circuit simulation of power-down mode. The method is capable of detecting floating nodes and subtle leakage current paths in power-down mode. It introduces a new structural graph model of the static behavior of the circuit which forms the basis for power-down synthesis.
- The first method for analog power-down reliability verification was introduced (section 3.3). It detects subtle reliability problems by combining structure recognition and voltages propagation with new voltage matching rules suitable for automatic verification.
- The first method for automatic analog power-down synthesis was developed (chapter 4). It comprises the steps of circuit rip-up, gate-shutoff computation, net dependency analysis, switch sizing and schematic backannotation.

Experimental results for analog circuits of different sizes and types are presented for all of the methods. Additionally, the efficiency and efficacy of the sizing rules approach for structure recognition [Massier et al., 2008] was demonstrated for the first time for industrial analog circuits with sizes up to 20.000 devices.

The methods for power-down circuit verification and synthesis enable systematic power-down circuit design and help to reduce the manual efforts required for it. Furthermore, they help to reduce the risk of subtle design errors that are only discovered late in the development process, e.g., after manufacturing of the first silicon samples. On the one hand, verification methods can be used to augment state-of-the-art industrial design flows by adding a tool supported automatic sign-off step after the manual design. On the other hand, synthesis goes one step beyond verification. It exploits the systematic of common power-down mode designs in order to develop algorithms that automatically compute power-down circuits that are correct by construction. The algorithms can be used for extensive design-space exploration of power-down mode design. New heuristics for efficiently and effectively choosing appropriate solutions out of the given design space have been presented. The presented synthesis algorithm applies a heuristic of rip-up patterns that are commonly found in manual designs and takes into account voltage matching rules. It allows a fully-automatic as well as a semi-automatic design flow. Experimental results demonstrate the efficiency and efficacy of the method for different classes of analog circuits. Solutions for switch sizing and schematic backannotations were investigated for the given example circuits. Fully automatic synthesis was demonstrated for a hierarchical LVDS driver.

The work in this thesis shows that the systematics in analog power-down circuits can be exploited for automatic verification and synthesis of power-down circuits. Compared to the automatic synthesis of the analog "power-up mode functionality" the author believes that power-down circuit synthesis is much more open for design automation. The work of this thesis shows that analog power-down circuits actually behave very digital. Hence, the design process can rely on very rough models that capture only the qualitative behavior. By introducing a static graph model, this thesis demonstrated that this fact can successfully be exploited to build versatile power-down verification and synthesis algorithms.

# List of Figures

# List of Tables

# Bibliography


Adler, D. (1991). Switch-level simulation using dynamic graph algorithms. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 10(3):346–355.

Ahmadyan, S. N., Kumar, J. A., and Vasudevan, S. (2013). Runtime verification of nonlinear analog circuits using incremental time-augmented rrt algorithm. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 21–26.

Antao, B. A. A. and Brodersen, A. J. (1995). Archgen: Automated synthesis of analog systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 3(2):231–244.

Antreich, K., Eckmueller, J., Graeb, H., Pronath, M., Schenkel, F., Schwencker, R., and Zizala, S. (2000). Wicked: analog circuit synthesis incorporating mismatch. In *Custom Integrated Circuits Conference, 2000. CICC. Proceedings of the IEEE 2000*, pages 511–514.

Asai, H. and Suzuki, S. (1992). A new piecewise-linear simulation algorithm using switch-level pattern dynamics. In *Circuits and Systems, 1992. ISCAS '92. Proceedings., 1992 IEEE International Symposium on*, volume 2, pages 899–902 vol.2.

Balasubramanian, S. (2016). Mixed-Signal – Successful Techo-on-Tours, Huge Focus at DAC 2013. *Cadence Mixed-Signal Design Blog*. http://community.cadence.com/cadence_blogs_8/b/ms/archive/2013/05/31/mixed-signal-huge-focus-for-cadence-dac-2013, accessed March 8th 2016.

Blaauw, D. T., Saab, D. G., Banerjee, S. P., and Abraham, J. A. (1991). Functional abstraction of logic gates for switch-level simulation. In *Design Automation. EDAC., Proceedings of the European Conference on*, pages 329–333.

Blieck, S. and Janssens, E. (1996). Software Check for Power-Down Mode of Analog Circuits. In *European Solid-State Circuits Conf. (ESSCIRC)*, pages 404–407.

Bonfini, G., Chiavacci, M., Mariani, R., and Pescari, E. (2006). A mixed-signal verification kit for verification of analogue-digital circuits. In *Design, Automation and Test in Europe, 2006. DATE '06. Proceedings*, volume 2, pages 1–6.

Boyle, G. R., Pederson, D. O., Cohn, B. M., and Solomon, J. E. (1974). Macromodeling of integrated circuit operational amplifiers. *IEEE Journal of Solid-State Circuits*, 9(6):353–364.

Bryant, R. E. (1984). A Switch-Level Model and Simulator for MOS Digital Systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, C-33(2):160–177.

Bryant, R. E. (1987). A Survey of Switch-Level Algorithms. *Design Test of Computers, IEEE*, 4(4):26–40.

Bryant, R. E. (1988). Data parallel switch-level simulation. In *Computer-Aided Design, 1988. ICCAD-88. Digest of Technical Papers., IEEE International Conference on*, pages 354–357.

Cadence Design Systems (2013). Virtuoso® RelXpert Reliability Simulator User Guide. *Cadence® MMSIM Product Documentation*.

Cadence Design Systems (2015a). *Spectre Accelerated Parallel Simulator (Software)*. www.cadence.com.

Cadence Design Systems (2015b). *Spectre Circuit Simulator (Software)*. www.cadence.com.

Chen, Y. et al. (2001). Stress-Induced MOSFET Mismatch for Analog Circuits. In *Integrated Reliability Workshop Final Rep., IEEE Int.*, pages 41–43.

Chouard, F. (2012). *Device Aging in Analog Circuits for Nanoelectronic CMOS Technologies*. PhD thesis, Technische Universität München.

Chouard, F. et al. (2011). An Analog Perspective on Device Reliability in 32nm High-k Metal Gate Technology. In *Design and Diagnostics of Electron. Circuits Syst. (DDECS)*, pages 65–70.

Christian Schulte, G. T. and Z.Lagerkvist, M. (2013). *Modeling and Programing with Gecode*.

Crols, J., Donnay, S., Steyaert, M., and Gielen, G. (1995). A high-level design and optimization tool for analog rf receiver front-ends. In *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on*, pages 550–553.

Dahlgren, P. (1995). A multiple-dominance switch-level model for simulation of short

faults. In *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on*, pages 674–680.

Darfeuille, S. (2012). PERL scripts-based netlist analysis tool for the detection of ESD "big buffer" configurations. In *Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2012 International Conference on*, pages 269–272.

Dastidar, T. R. and Chakrabarti, P. P. (2005). A verification system for transient response of analog circuits using model checking. In *VLSI Design, 2005. 18th International Conference on*, pages 195–200.

Degrauwe, M. G. R., Nys, O., Dijkstra, E., Rijmenants, J., Bitz, S., Goffart, B. L. A. G., Vittoz, E. A., Cserveny, S., Meixenberger, C., van der Stappen, G., and Oguey, H. J. (1987). Idac: an interactive design tool for analog cmos circuits. *IEEE Journal of Solid-State Circuits*, 22(6):1106–1116.

Ebeling, C. (1988). Geminiii: a second generation layout validation program. In *Computer-Aided Design, 1988. ICCAD-88. Digest of Technical Papers., IEEE International Conference on*, pages 322–325.

Eick, M. (2013). *Structure and Signal Path Analysis for Analog and Digital Circuits*. PhD thesis, Technische Universität München.

Ferent, C. and Doboli, A. (2013). Symbolic matching and constraint generation for systematic comparison of analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(4):616–629.

Ferent, C., Montano, S., and Doboli, A. (2012). A prototype framework for conceptual design of novel analog circuits. In *Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), 2012 International Conference on*, pages 13–16.

Fernández, F. V., Rodríguez-Vázquez, A., and Huertas, J. L. (1991). Interactive ac modeling and characterization of analog circuits via symbolic analysis. *Analog Integrated Circuits and Signal Processing*, 1(3):183–208.

Ferraria, T. F. P. (2013). Verification of Analog Circuits in Power-Down Mode. Master's thesis, Universidade Nova de Lisboa.

Georgakos, G. (2014). Circuit Analyzer Systems and Methods. US 2014/0019928 A1.

Ghosh, A. and Vemuri, R. (1999). Formal verification of synthesized analog designs. In *Computer Design, 1999. (ICCD '99) International Conference on*, pages 40–45.

Gielen, G. G. E., Walscharts, H. C. C., and Sansen, W. M. C. (1989). Isaac: a symbolic simulator for analog integrated circuits. *IEEE Journal of Solid-State Circuits*, 24(6):1587–1597.

Gielen, G. G. E., Walscharts, H. C. C., and Sansen, W. M. C. (1990). Analog circuit design optimization based on symbolic simulation and simulated annealing. *IEEE Journal of Solid-State Circuits*, 25(3):707–713.

Grabowski, D., Grimm, C., and Barke, E. (2006). Semi-symbolic modeling and simulation of circuits and systems. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4 pp.–986.

Hajj, I. N. (1990). An algebra for switch-level simulation. In *Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*, pages 488–491.

Hanna, K. (2000). Reasoning about analog-level implementations of digital systems. *Formal Methods in System Design*, 16(2):127–158.

Harjani, R., Rutenbar, R. A., and Carley, L. R. (1989). Oasys: a framework for analog circuit synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(12):1247–1266.

Harvey, J. P., Elmasry, M. I., and Leung, B. (1992). Staic: an interactive framework for synthesizing cmos and bicmos analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(11):1402–1417.

Hassoun, M. M. and Ackerman, J. E. (1990). Symbolic simulation of large-scale circuits in both frequency and time domains. In *Circuits and Systems, 1990., Proceedings of the 33rd Midwest Symposium on*, pages 707–710 vol.2.

Hastings, A. (2006). *The Art of Analog Layout*. Pearson Prentice Hall, second edition.

Hershenson, M., Boyd, S. P., and Lee, T. H. (2001). Optimal design of a cmos op-amp via geometric programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(1):1–21.

Ho, C.-W., Ruehli, A., and Brennan, P. (1975). The modified nodal approach to network analysis. *IEEE Transactions on Circuits and Systems*, 22(6):504–509.

Hogan, M. (2014). Reduce Verification Complexity in Low/Multi-Power Designs. *Mentor Graphics White Paper, online: http://www.mentor.com, no publication date, accessed in September 2014*.

Huang, L. P. and Bryant, R. E. (1993). Intractability in linear switch-level simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(6):829–836.

ITRS (2007). The International Technology Roadmap for Semiconductors – Design. http://www.itrs.net.

ITRS (2011). The International Technology Roadmap for Semiconductors – Design. http://www.itrs.net.

Jiao, F., Montano, S., Ferent, C., Doboli, A., and Doboli, S. (2015). Analog circuit design knowledge mining: Discovering topological similarities and uncovering design reasoning strategies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1045–1058.

Jones, L. G. and Blaauw, D. T. (1994). A cache-based method for accelerating switch-level simulation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(2):211–218.

Juneja, D. (2015). Power Off or Power Down? http://www.analog.com/library/analogdialogue/archives/49-03/power_up_down.html, accessed March 9th 2016.

Keating, M., Flynn, D., Aitken, R., Gibbons, A., and Shi, K. (2007). *Low Power Methodology Manual*. Springer.

Ker, M. D., Lin, C. Y., and Hsiao, Y. W. (2011). Overview on esd protection designs of low-parasitic capacitance for rf ics in cmos technologies. *IEEE Transactions on Device and Materials Reliability*, 11(2):207–218.

Kobylarz, T. J. (1969). Simulation of linear and nonlinear circuits for symbolic analysis and transient response. *IEEE Transactions on Education*, 12(4):299–305.

Koh, H. Y., Sequin, C. H., and Gray, P. R. (1990). Optimal design of a cmos op-amp via geometric programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(2).

Kravitz, S. A., Bryant, R. E., and Rutenbar, R. A. (1991). Massively parallel switch-

level simulation: a feasibility study. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(7):871–894.

Kruiskamp, D. L. W. (1995). Darwin: Cmos opamp synthesis by means of a genetic algorithm. In *Design Automation, 1995. DAC '95. 32nd Conference on*, pages 433–438.

Lee, P. (2012). *Introduction To Low-Power Design In VLSIs (Chapter 1.2)*. lulu.com. ISBN-10: 1105346560 / ISBN-13: 978-1105346569.

Leitao, P. (2012a). Backannotations for the Verification of Power-Down-Mode (Technical Documentation). Technical report. Technische Univeristät München, Institute for Electronic Design Automation.

Leitao, P. (2012b). Backannotations for the Verification of Power-Down-Mode (User Documentation). Technical report. Technische Univeristät München, Institute for Electronic Design Automation.

Lescot, J. et al. (2012). Static Low Power Verification at Transistor Level for SoC Design. In *ISLPED*.

Li, B. and Gu, D. (1992). Sscnap: a program for symbolic analysis of switched capacitor circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(3):334–340.

Liu, X.-X., Tan, S. X. D., Hao, Z., and Shi, G. (2012). Time-domain performance bound analysis of analog circuits considering process variations. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pages 535–540.

Lourenço, N. and Horta, N. (2012). Genom-pof: Multi-objective evolutionary synthesis of analog ics with corners validation. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, pages 1119–1126, New York, NY, USA. ACM.

Lu, Z. and Bell, D. A. (2010). Hierarchical verification of chip-level esd design rules. In *Electrical Overstress/ Electrostatic Discharge Symposium (EOS/ESD), 2010 32nd*, pages 1–6.

Maler, O. and Nickovic, D. (2004). *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems: Joint International Conferences on Formal Modeling and Analysis of Timed Systmes, FORMATS 2004, and Formal Techniques in Real-Time and Fault -Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24,*

*2004. Proceedings*, chapter Monitoring Temporal Properties of Continuous Signals, pages 152–166. Springer Berlin Heidelberg, Berlin, Heidelberg.

Maricau, E. and Gielen, G. (2011). Transistor Aging-Induced Degradation of Analog Circuits: Impact Analysis and Design Guidelines. In *European Solid-State Circuits Conf. (ESSCIRC)*, pages 243–246.

Massier, T. (2010). *On the Structural Analysis of CMOS and Bipolar Analog Integrated Circuits*. PhD thesis, Technische Universität München.

Massier, T. et al. (2008). The Sizing Rules Method for CMOS and Bipolar Analog Integrated Circuit Synthesis. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 27(12):2209–2222.

Maulik, P. C., Carley, L. R., and Rutenbar, R. A. (1995). Integer programming based topology selection of cell-level analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(4):401–412.

Medea+/Catrene Office (2009). EUROPEAN Design Automation Roadmap, 6th Edtion. http://www.catrene.org/web/communication/publ_eda.php.

Medeiro, F., Fernandez, F. V., Dominguez-Castro, R., and Rodriguez-Vazquez, A. (1994). A statistical optimization-based approach for automated sizing of analog cells. In *Computer-Aided Design, 1994., IEEE/ACM International Conference on*, pages 594–597.

Medhat, D. (2009). Easier cross-domain signal protection for mixed-signal SoCs. *EE Times, online: http://eetimes.com*.

Medhat, D. (2012). Power-Aware Verification in Low Power ICs. *Chip Design Magazine, online: http://chipdesignmag.com*.

Meissner, M. and Hedrich, L. (2015). Feats: Framework for explorative analog topology synthesis. *TCAD*.

Mentor Graphics (2015a). *Calibre® PERC™ Software*. www.mentor.com.

Mentor Graphics (2015b). *Eldo Classic (Circuit Simulator Software)*. www.mentor.com.

Mentor Graphics (2015c). *Eldo Premier (Circuit Simulator Software)*. www.mentor.com.

Mentor Graphics (2016). *Calibre® PERC™ Datasheet, online: www.mentor.com, accessed March 2016*.

Meyer, W. and Camposano, R. (1993). Fast, accurate, integrated gate and switch-level fault simulation. In *European Test Conference, 1993. Proceedings of ETC 93., Third*, pages 194–199.

Meyer-Jungclaussen, M. (2014). Entwicklung eines OpenAccess Schematic-Viewers mit Qt. Master's internship report, Technische Universität München.

Meyer zu Bexten, V., Tristl, M., Jerke, G., Marquardt, H., and Medhat, D. (2015). Physical verification flow for hierarchical analog ic design constraints. In *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pages 447–453.

Michael, C. et al. (1992). Mismatch Drift: A Reliability Issue for Analog MOS Circuits. In *Reliability Physics Symposium 1992. 30th Annual Proceedings., International.*

Milošević, M. (2009). *Hierarchical Pattern Matching in VLSI.* PhD thesis, Technische Universität München.

Mlayeh, H. (2012). Dokumentation zur XML-Strukturbibliothek für xcat . Technical report. Master's internship report, Technische Universität München.

Nagel, L. (1975). *SPICE2: A computer program to simulate semiconductor circuits.* Ph. D. dissertation, Univ. of California, Berkeley.

Neuner, M. (2013). Power-Down-Synthese unter Verwendung von Constraint-Programmierung. Bachelor's thesis, Technische Universität München.

Neuner, M. (2015). Power-Down-Schaltungs- und Schaltbildsynthese. Master's thesis, Technische Universität München.

Ochotta, E. S., Rutenbar, R. A., and Carley, L. R. (1996). Synthesis of high-performance analog circuits in astrx/oblx. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(3):273–294.

Ohlrich, M., Ebeling, C., Ginting, E., and Sather, L. (1993). Subgemini: Identifying subcircuits using a fast subgraph isomorphism algorithm. In *Design Automation, 1993. 30th Conference on*, pages 31–37.

Pelz, G. and Roettcher, U. (1994). Pattern matching and refinement hybrid approach to circuit comparison. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(2):264–276.

Phelps, R., Krasnicki, M., Rutenbar, R. A., Carley, L. R., and Hellums, J. R. (2000). Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search.

*IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(6):703–717.

Plas, G. v. d., Debyser, G., Leyn, F., Lampaert, K., Vandenbussche, J., Gielen, G. G. E., Sansen, W., Veselinovic, P., and Leenarts, D. (2001). Amgie-a synthesis environment for cmos analog integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 20(9):1037–1058.

Pompl, T., Schlunder, C., Hommel, M., Nielen, H., and Schneider, J. (2006). Practical aspects of reliability analysis for ic designs. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 193–198.

Ramachandran, V. (1986). Algorithmic aspects of mos vlsi switch-level simulation with race detection. *IEEE Transactions on Computers*, C-35(5):462–475.

Rewienski, M. (2011). A Perspective on Fast-SPICE Simulation Technology. In *Simulation and Verification of Electronic and Biological Systems*, pages 23–42. Springer Netherlands.

Rubanov, N. (2006). A high-performance subcircuit recognition method based on the nonlinear graph optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(11):2353–2363.

Rutenbar, R. A., Gielen, G. G. E., and Antao, B. A. (2002). *ComputerAided Design of Analog and MixedSignal Integrated Circuits*. Wiley-IEEE Press.

Schillmaier, F. (2014). Power-Down Synthese unter Verwendung konstruktiver Regeln. Bachelor's thesis, Technische Universität München.

Schlunder, C. et al. (2003). On the Degradation of p-MOSFETs in Analog and RF Circuits under Inhomogeneous Negative Bias Temperature Stress. In *12th IEEE Int. Workshop on Cellular Nanoscale Networks and their Applicat.*, pages 5–10.

Schlünder, C. (2010). Design for Reliability (DfR). In *GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf*.

Sedgewick, R. (1988). *Algorithms*. Addison-Wesley Publishing Company.

Shi, C. J. R. and Tan, X.-D. (2000). Canonical symbolic analysis of large analog circuits with determinant decision diagrams. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(1):1–18.

Shrivastava, G. (2016). Sizing of Power-Down Switches for Analog Circuits. Master's thesis, Technische Universität München.

Si2 Consortium (2013). *Open Access Database*. online: http://www.si2.org/, accessed in April 2013.

Starzyk, J. and Konczykowska, A. (1986). Flowgraph analysis of large electronic networks. *IEEE Transactions on Circuits and Systems*, 33(3):302–315.

Steinhorst, S. and Hedrich, L. (2010). Improving verification coverage of analog circuit blocks by state space-guided transient simulation. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 645–648.

Steinhorst, S. and Hedrich, L. (2012a). Analog assertion-based verification on partial state space representations using asl. In *Specification and Design Languages (FDL), 2012 Forum on*, pages 98–104.

Steinhorst, S. and Hedrich, L. (2012b). Equivalence checking of nonlinear analog circuits for hierarchical ams system verification. In *VLSI and System-on-Chip, 2012 (VLSI-SoC), IEEE/IFIP 20th International Conference on*, pages 135–140.

Steinhorst, S. and Hedrich, L. (2012c). Trajectory-directed discrete state space modeling for formal verification of nonlinear analog circuits. In *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference on*, pages 202–209.

Svensson, C. M. and Tjarnstrom, R. (1988). Switch-level simulation and the pass transistor exor gate. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 7(9):994–997.

Synopsys (2015). *CustomSim Circuit Check (CCK) (Software)*. www.synopsys.com.

Synopsys (2015a). *CustomSim (Circuit Simulator Software)*. www.synopsys.com.

Synopsys (2015b). *Hspice (Circuit Simulator Software)*. www.synopsys.com.

Vandris, E. and Sobelman, G. (1990). Fast switch-level fault simulation using functional fault modeling. In *Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*, pages 74–77.

Vlachas, P.-R. (2014). Verification of the Power-Down Mode of Analog Circuits using Constraint Programming. Bachelor's thesis, Technische Universität München.

Wambacq, P., Fernandez, F. V., Gielen, G., Sansen, W., and Rodriguez-Vazquez, A.

(1995). Efficient symbolic computation of approximated small-signal characteristics of analog integrated circuits. *IEEE Journal of Solid-State Circuits*, 30(3):327–330.

Wambacq, P., Gielen, G. G. E., Kinget, P. R., and Sansen, W. (1999). High-frequency distortion analysis of analog integrated circuits. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(3):335–345.

Wang, Q. (2015). Mixed-Signal SOC Verification – Using Analog Behavioural Models. *Analog Dialogue 49-03*. http://www.edn.com.

Wei, Y. and Doboli, A. (2008). Structural macromodeling of analog circuits through model decoupling and transformation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(4):712–725.

World Wide Web Consortium (W3C) (2015). *Extensible Markup Language (XML)*. online: https://www.w3.org/, accessed in November 2015.

Yang, J., Tan, S. X. D., Qi, Z., and Gawecki, M. (2005). Hierarchical symbolic piecewise-linear circuit analysis. In *Behavioral Modeling and Simulation Workshop, 2005. BMAS 2005. Proceedings of the 2005 IEEE International*, pages 140–145.

Yilmaz, E. and Green, M. M. (1999). Applying globally convergent techniques to conventional dc operating point analyses. In *Simulation Symposium, 1999. Proceedings. 32nd Annual*, pages 153–158.

Yin, L., Deng, Y., and Li, P. (2013). Simulation-assisted formal verification of nonlinear mixed-signal circuits with bayesian inference guidance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(7):977–990.

Yu, Q. and Sechen, C. (1995). Efficient approximation of symbolic network functions using matroid intersection algorithms. In *Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on*, volume 3, pages 2088–2091 vol.3.

Yu, T., Tan, S. X. D., Cai, Y., and Tang, P. (2014). Time-domain performance bound analysis for analog and interconnect circuits considering process variations. In *Design Automation Conference (ASP-DAC), 2014 19th Asia and South Pacific*, pages 455–460.

Yuan, J. and Svensson, C. (1988). Cmos circuit speed optimization based on switch level simulation. In *Circuits and Systems, 1988., IEEE International Symposium on*, pages 2109–2112 vol.3.

Zaki, M. H., Tahar, S., and Bois, G. (2006). Formal verification of analog and mixed

signal designs: Survey and comparison. In *Circuits and Systems, 2006 IEEE North-East Workshop on*, pages 281–284.

Zhang, H. and Shi, G. (2011). Symbolic behavioral modeling for slew and settling analysis of operational amplifiers. In *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, pages 1–4.

Zwerger, M. and Graeb, H. (2012a). Short-Circuit-Path and Floating-Node Verification of Analog Circuits in Power-Down Mode. In *Int. Conf. Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*.

Zwerger, M. and Graeb, H. (2012b). Verifikation des Power-Down-Modus von analogen Schaltungen. In *GMM/GI/ITG-Fachtagung Zuverlässigkeit und Entwurf*.

Zwerger, M. and Graeb, H. (2013). Verification of the Power-Down Mode of Analog Circuits by Structural Voltage Propagation. *Analog Integrated Circuits and Signal Processing*.

Zwerger, M. and Graeb, H. (2015). Detection of Asymmetric Aging-Critical Voltage Conditions in Analog Power-Down Mode. In *DATE*.

Zwerger, M., Neuner, M., and Graeb, H. (2015a). Power-Down Circuit Synthesis for Analog/Mixed-Signal. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

Zwerger, M., Shrivastava, G., and Graeb, H. (2016). Power-Down Synthesis for Analog Circuits including Switch Sizing. In *Symbolic Methods and Applications in Circuit Design (SMACD)*.

Zwerger, M., Vlachas, P.-R., and Graeb, H. (2015b). A Fast Analytical Approach for Static Power-Down Mode Analysis. In *IEEE International Conference on Electronics, Circuits, & Systems (ICECS)*.