Technische Universität München
TUM School of Computation, Information and Technology

Hierarchical Verification and Hierarchical Synthesis of the Power-Down Mode of Analog Circuits

Maximilian Ulrich Neuner

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften **(Dr.-Ing.)**

genehmigten Dissertation.

Vorsitz:	Prof. Dr.-Ing. Thomas Eibert
Prüfer der Dissertation:	1. apl. Prof. Dr.-Ing. habil. Helmut Gräb
	2. Prof. Dr.-Ing. Steffen Paul
	3. Prof. Sachin Sapatnekar, Ph.D.

Die Dissertation wurde am 24.04.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 15.10.2023 angenommen.

Abstract

To save power, dedicated transistors can switch off the currents of analog circuits in power-down mode. Hierarchical circuits consist of several blocks. Even if the power-down mode of the individual subblocks was correctly designed, their connectivity can cause unexpected errors in the power-down mode of the overall system, e.g. currents. This work describes a new method to identify such errors and one to construct the power-down mode for hierarchical analog circuits fault-free.

Zusammenfassung

Um Energie zu sparen, stellen dedizierte Transistoren analoge Schaltungen im Power-Down-Modus stromlos. Hierarchische Schaltungen bestehen aus mehreren Blöcken. Deren Verdrahtung kann unerwartete Fehler, wie z.B. Ströme, im Power-Down-Modus des Gesamtsystems verursachen, obwohl dieser für dessen Einzelblöcke korrekt entworfen wurde. Diese Arbeit stellt eine neue Methode zur Erkennung solcher Fehler und eine zur fehlerfreien Konstruktion des Power-Down-Modus hierarchischer analoger Schaltungen vor.

Preface

During my early bachelor studies at TUM, I realized that I enjoyed the systematic, step-by-step approach software development offers to tackle and solve complex technical problems. This interest led me to my first internship at the Chair of Electronic Design Automation where I learned how to program and design graphical user interfaces for CAD tools specialized on integrated circuit design. Several more internships as well as my bachelor's thesis followed and I also completed my master studies at the same chair under the supervision of Michael Zwerger - working on a topic which should later turn out to be the foundation of my dissertation.

Many people accompanied me throughout my time at the institute; without their help and encouragement this work would most likely not exist as it is in its current form. I want to thank Prof. Schlichtmann, head of the institute, for giving me the opportunity to conduct my own research and to work in academia - I especially enjoyed teaching and passing on knowledge which I learned myself not too long ago in the same lecture rooms. I want to express my deep gratitude towards my supervisor Prof. Graeb. He always encouraged me to follow new ideas and his ability to discover new aspects and value in my work (which I did not even think of) were crucial for the successful completion of this work. Kudos as well for his patience for revising my drafts and helping me to improve my presentation skills. A special thanks also to Prof. Steffen Paul from the University of Bremen and Prof. Sachin Sapatnekar from the University of Minnesota for taking interest in my work and the time to review it.

I am still feeling honored and glad to be given the chance to conduct the lab part of the lecture "Mathematical Methods of Circuit Design" at Nanyang Technological University in Singapore which afterwards allowed me to travel the world - and in the end to meet my future wife in Malaysia.

A big thanks to my colleagues at the institute for the good working atmosphere which enabled fruitful discussions, the get-togethers during lunch time, table soccer matches during coffee breaks and other leisure time activities outside of work.

I also want to thank all the students who were actively helping me to improve the implemented software, especially Inga Abel who was always willing to explore unusual ideas which lead to unexpected or surprising results.

Finally, I want to thank my friends and family who were always supporting me in my activities, took off my mind from work and helped me in that way to refresh and collect new ideas.

Munich, December 2023

Contents

1. Introduction **3**
- 1.1. Analog Power-Down Mode Design . . . 5
- 1.2. State of the Art . . . 8
- 1.3. Problem Formulation and Contributions . . . 10
- 1.4. Thesis Structure and Prior Publications . . . 12

2. Hierarchical Circuit Representation **15**

3. Hierarchical Power-Down Verification **17**
- 3.1. Overview . . . 17
- 3.2. Voltage Propagation . . . 18
- 3.3. Short Circuit Path Computation . . . 21
- 3.4. Matching and Symmetry Assertion . . . 24
  - 3.4.1. Power-up Transformation . . . 24
  - 3.4.2. Assertion . . . 26
- 3.5. Experimental Results . . . 29
  - 3.5.1. Dual Gain Amplifier . . . 29
  - 3.5.2. High Input Impedance Differential Amplifier . . . 36

4. Hierarchical Power-Down Synthesis **43**
- 4.1. Overview . . . 43
- 4.2. Rip-Up Heuristic . . . 45
- 4.3. Gate Shut-Off . . . 45
  - 4.3.1. Constraint Optimization Problem . . . 49
  - 4.3.2. Selection of the Optimal Solution . . . 59
  - 4.3.3. Computational Complexity . . . 62
- 4.4. Experimental Results . . . 64
  - 4.4.1. Current Mirror Operational Amplifier . . . 65
  - 4.4.2. Low Resistance Operational Amplifier . . . 66
  - 4.4.3. Fully Differential Operational Amplifier . . . 67
  - 4.4.4. Dual Gain Amplifier . . . 69
  - 4.4.5. High Input Impedance Differential Amplifier . . . 73

5. Library-Free Structure Recognition **77**
- 5.1. State of the Art . . . 78
- 5.2. Overview . . . 78
- 5.3. Creation of New Library Elements . . . 80
- 5.4. Library Extension . . . 83
- 5.5. Experimental Results . . . 84
  - 5.5.1. Folded Cascode Amplifier Variants . . . 84
  - 5.5.2. Level Shifter Topologies . . . 87

6. Conclusion **91**

A. Hierarchical Structure Recognition **93**
A.1. Introduction 93
A.2. State of the Art and Contributions 93
A.3. Overview 97
A.4. Partial Circuit Flattening 98
A.5. Structure Recognition 100
A.6. Experimental Result 105

B. Hierarchical Symmetry Computation **109**
B.1. Introduction 109
B.2. State of the Art and Contributions 109
B.3. Overview 110
B.4. Signal Flow Graph Generation 111
B.5. Signal Path Analysis 116
B.6. Experimental Result 124
B.7. Discussion and Limitations 124

C. Manual Power-Down Mode Revision **129**

Bibliography **141**

1. Introduction

Analog circuits are widely considered as the interface between the real and the digital world. Most often they are the first point of interaction with time continuous electrical signals within an integrated circuit (IC).

Fig. 1.1 shows the simplified block diagram of a wireless speaker. Its antenna receives a time continuous wireless signal which is filtered and amplified by dedicated analog circuitry. The resulting signal is then binarized by an analog-to-digital converter (ADC) and processed by a digital signal processor (DSP). Finally, the signal is translated back into the analog domain by a digital-to-analog converter (DAC) and output by a speaker. The whole system is powered by a battery whose power is distributed to the system's components by a power management unit (PMU).

The runtime of the speaker is limited by the electrical energy stored in its battery. Hence, each component of the speaker should be designed as power efficient as possible in order to increase its runtime.

Fig. 1.2 shows a digital inverter and a simple analog amplifier circuit, i.e., a source circuit.

The output signal z of an inverter takes the opposite voltage level of its input signal x which corresponds to either the ground, i.e., logical zero, or the supply voltage, i.e., logical one. The inverter is embedded in sequential logic, i.e., in-between the two registers $reg1$ and $reg2$. Registers store incoming signal values from a previous logic stage and pass it onto the next one whenever the clock signal *clk* transitions from the ground to the supply voltage.

The analog source circuit amplifies a small signal input voltage at net *in* and converts it into a small signal output current at node *out*. The amplification factor, the so-called "gain", is thereby defined as the ratio of the output and input voltage in decibel.

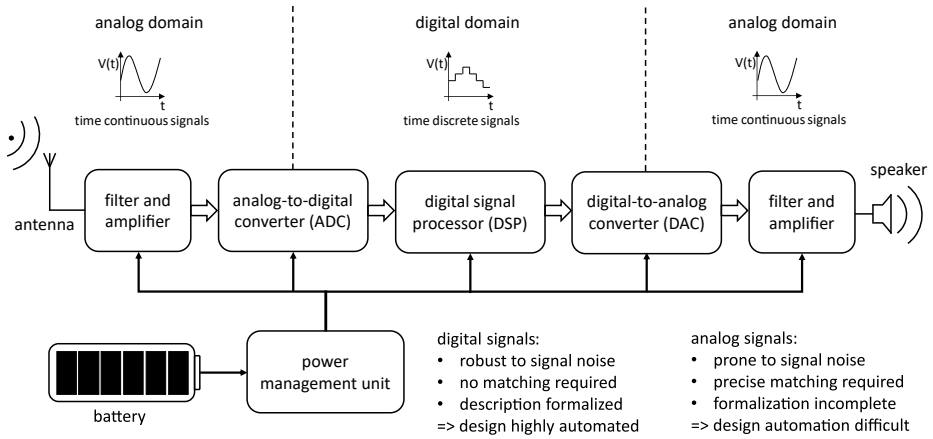
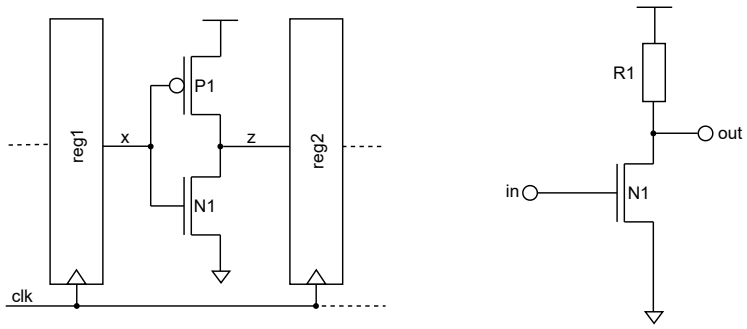


Figure 1.1.: Simplified block diagram of a wireless speaker.



(a) inverter circuit (b) source circuit

Figure 1.2.: Schematics of a simple digital (a) and a simple analog circuit (b).

The power consumption of a digital circuit has two contributors [1]:

$$P_{dig} = P_{stat} + P_{dyn}. \tag{1.1}$$

The static power dissipation

$$P_{stat} = I_{leak} \cdot V_{dd} \quad \text{with} \tag{1.2}$$

$$I_{leak} \sim \frac{W}{L} I_s e^{\frac{V_{gs}-V_{th}}{aV_T}} \tag{1.3}$$

is caused by sub-threshold leakage currents of the implemented CMOS transistors.

The dynamic power dissipation

$$P_{dyn} = \alpha C V_{dd}^2 f \tag{1.4}$$

is the power required to charge or discharge small capacitors at the affected nodes of the circuit with α describing the switching frequency in percent.

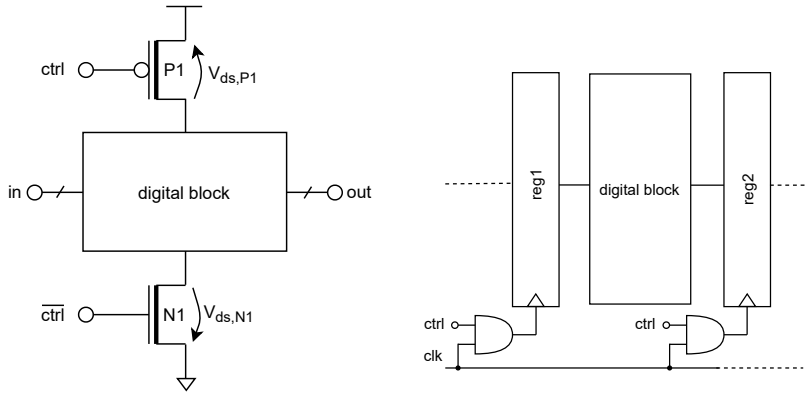
Analog circuits require a constant bias current I_B to set the operating point of the circuit and hence:

$$P_{analog} = I_B \cdot V_{dd}. \tag{1.5}$$

The static power dissipation of digital circuits can be reduced by threshold voltage scaling [2]. The threshold voltage of the transistors can be increased by applying a voltage to the substrate of the chip. This reduces the leakage current exponentially according to eq. (1.3). This technique however is not applicable for analog circuits as the transistors typically operate in saturation and not in sub-threshold region.

Another technique to reduce the static power dissipation is shown by Fig. 1.3a. Power-gating, also known as multi-threshold voltage circuits [3], uses two additional transistors to disconnect the circuit block from the ground and the supply voltage when not required for system operation. These so-called header ($P1$) and footer ($N1$) transistors have a higher threshold voltage than the transistors of the digital logic block which hence reduces the leakage current of the whole block according to eq. (1.3).

This technique can principally also be applied to analog circuits but has the following two disadvantages:



(a) power gating (b) clock gating

Figure 1.3.: Two digital power-saving techniques.

- the header and footer transistors must be sized large in order to carry the bias current of the circuit,
- the additional voltage drops between the drain and source pins of the power-gating transistors limits the available voltage swing of the circuit.

The first disadvantage is unacceptable for area-critical devices and the reduced voltage swing might cause a circuit to fail its specifications. Hence, this technique is often not suitable in practice.

Fig. 1.3b shows a technique to reduce the dynamic power dissipation of a digital circuit. Clock gating disables the *clk* signal at the registers of sequential logic circuits when not required for system operation. Hence, the dynamic power drops to zero according to eq. (1.4) as the block's frequency is zero.

This technique –as well as frequency scaling– cannot be applied to analog circuits as they are not synchronized by a clock signal.

In summary, common digital power saving techniques cannot be transferred to the analog domain. Instead, a so-called power-down mode is implemented.

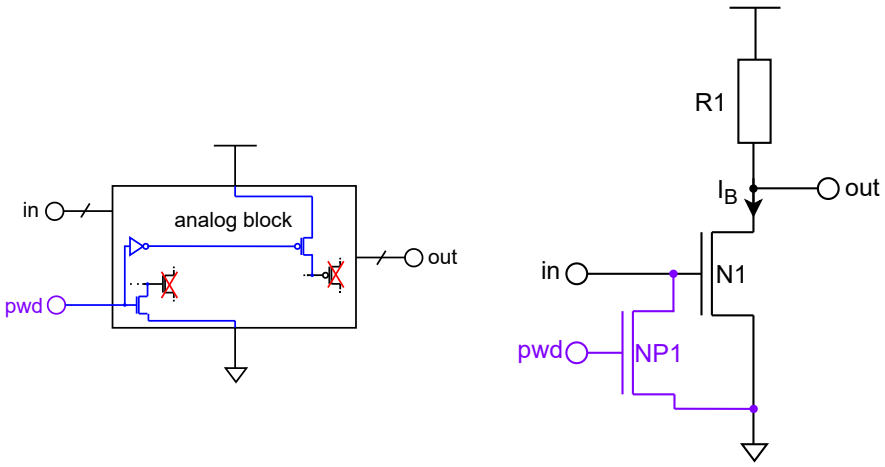
1.1. Analog Power-Down Mode Design

Fig. 1.4a illustrates the basic principle of the analog power-down mode. The power-down mode is implemented by additional transistors, highlighted in blue, which pull the internal nets of the analog block either to the ground or to the supply voltage. This forces the devices of the circuit either into the conducting or non-conducting state with the overall goal to shut off all bias currents in the circuit.

The power-down circuitry is controlled by the digital signal *pwd*. It is inactive when *pwd* is assigned to the ground voltage level *gnd*, i.e., the analog block is in normal operation.

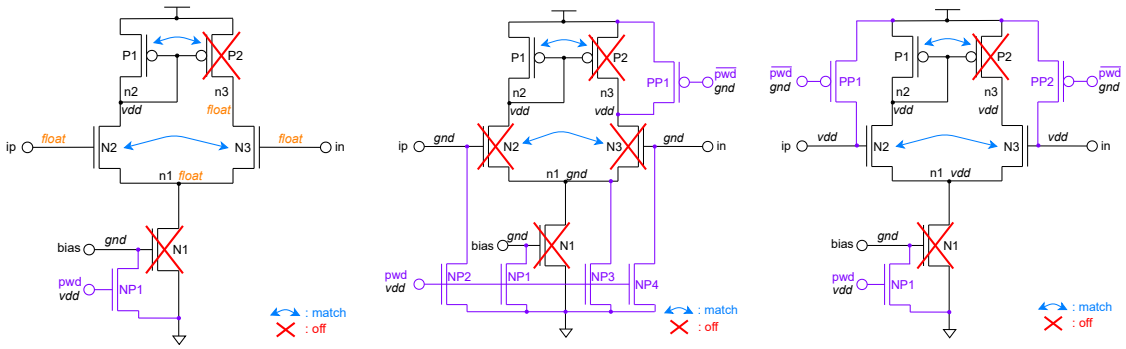
The power-down circuitry becomes active when *pwd* is set to the supply voltage level *vdd*. The analog block does not consume any power in this mode.

Fig. 1.4b shows the source circuit from Fig. 1.2 with its power-down circuitry highlighted in blue. The transistor $NP1$ pulls the gate of $N1$ to the ground voltage in power-down mode



(a) basic principle (b) source circuit

Figure 1.4.: Basic principle of the analog power-down mode (a) and the source circuit from Fig. 1.2b with its power-down circuitry (b) highlighted in blue.



(a) possible matching violations (b) matching fulfilled, area sub-optimal (b) matching fulfilled, area minimized

Figure 1.5.: Power-down mode implementation variants considering device matching.

to force $N1$ into the non-conducting state. This switches the bias current I_B of the circuit off and reduces its power-consumption to approximately zero. $NP1$ is inactive in normal operation and the bias current I_B can set the operating point of the circuit as usual.

Analog circuits require precise matching, i.e., functionally identical devices must be sized identically and should be exposed to similar operating conditions in order to work as intended [4]. This design principle also affects the analog power-down mode design.

Fig. 1.5 shows a differential input stage. It converts a differential input voltage at its input pins *in* and *ip* into a differential output current at the nodes $n2$ and $n3$. The two circuit halves processing the differential input signal, i.e., $(P1, N2)$ and $(P2, N3)$, must be built identically to show symmetrical electrical behaviour. Hence, the devices of the differential pair $dp(N2, N3)$ and of the simple current mirror load $scm(P1, P2)$ must be matched.

The publications [5; 6; 7] showed that matching deteriorates over time when the affected devices are exposed to asymmetric operating conditions, especially due to mismatched pin voltages. Thorough care must be taken to avoid voltage mismatch in power-down mode

as the affected devices could be stressed by the maximum available voltage difference in the circuit as its internal nets are either pulled to *vdd* or *gnd*. In the worst case, the circuit might fail after powering it up again due to time-dependent voltage mismatch in power-down mode. Matched devices must hence be exposed to matched voltages in power-down mode to prevent these problems.

The power-down mode of the differential input stage from Fig. 1.5a has been implemented by only one power-down transistor, namely $NP1$. The nets *in*, *ip*, $n1$ and $n3$ are high-impedant, i.e., floating, in this implementation as no connection to the supply or ground net can be established by conducting devices. The state of these nodes is hard to predict as it depends on the sub-threshold operation of $N1$, $N2$ and $N3$ which is often inaccurately captured by their corresponding device models [8; 9]. The diode-connected transistor $P1$ furthermore pulls $n2$ to the supply voltage level which can lead to asymmetric stress at the differential pair $dp(N2, N3)$ and the simple current mirror $scm(P1, P2)$ of the circuit.

The power-down circuitry from Fig. 1.5b overcomes these problems by four more power-down transistors which pull *in*, *ip* and $n1$ to the ground and $n3$ to the supply voltage level. Now, all voltages levels in the circuit are defined, i.e., not floating, and the differential pair $dp(N2, N3)$ as well as the simple current mirror $scm(P1, P3)$ are exposed to symmetrical voltages in power-down mode.

This solution however is not optimal with respect to the die area occupied by the additional power-down circuitry. The area requirements of the power-down circuitry is mainly determined by the number of inserted power-down transistors. It should be minimized whilst fulfilling the circuit's matching conditions as good as possible. The optimal implementation variant for the differential input stage is shown in Fig. 1.5c. It requires three power-down transistors in total. The transistors $PP1$ and $PP2$ turn on the differential pair $dp(N2, N3)$ in power-down mode which allows the supply voltage to propagate to $n1$ and $n3$. All matching conditions are still fulfilled whilst minimizing the area requirements of the power-down circuitry. In this example, it has been assumed that the voltages at the input and output pins of the circuit, i.e., *in*, *ip* and *out*, can be arbitrarily pulled to the supply or ground voltage level. In reality however, these voltages depend on external circuitry connected to these pins and must be considered during power-down mode design.

In summary, the design of the analog power-down mode has the following targets [10]:

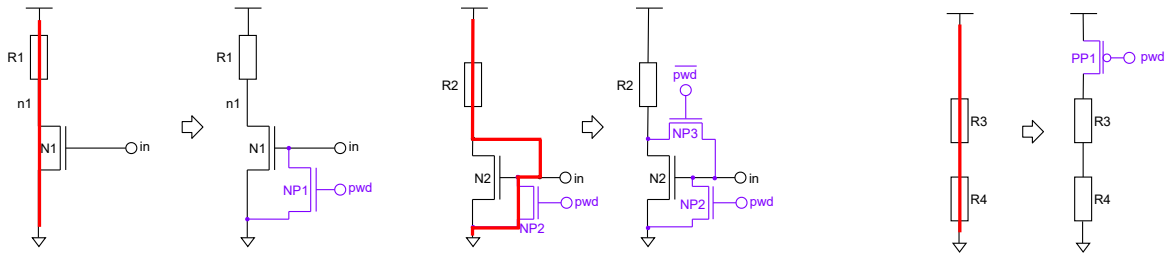
(A) All current paths must be shut off in power-down mode.
(B) Floating nodes should be avoided in power-down mode.
(C) Maximize matching to prevent time-dependent mismatch.
(D) Minimize the area requirements of the power-down circuitry.

Target (A) must be fulfilled, requirements (B), (C) and (D) should be fulfilled as good as possible.

Additionally, the normal operation of the circuit should not be affected by the inserted transistors. This must be verified by numerical simulation after the implementation of the power-down mode.

Fig. 1.6 shows the three shut-off patterns presented by [11] to implement the power-down mode of a circuit whilst following design goals (A) - (D).

The gate shut-off pattern inserts power-down transistors in parallel into the circuit to pull its internal nets either to the ground or to the supply voltage in order to turn off all current



(a) gate shut-off (b) diode rip-up (c) serial rip-up

Figure 1.6.: Shut-off patterns used for power-down mode implementation [11].

flow in the circuit. It is the main tool to implement the power-down mode of a given circuit and corresponds to the technique which has been used to switch off the source circuit and the differential stage from Figs. 1.4b and 1.5, respectively.

Some current paths in the circuit however cannot be switched off using the gate shut-off pattern. Fig. 1.6b shows a current path running over the resistor $R2$ and diode-connected transistor $N2$. The power-down transistor $NP2$ tries to switch off $N2$ by pulling its gate to the ground voltage which creates a new current path over the just inserted power-down transistor. The diode rip-up pattern inserts $NP3$ into the circuit which disables the diode configuration of $N2$ in power-down mode. Now, $N2$ can be safely switched off by $NP2$.

A serial rip-up, as shown in Fig. 1.6c, must be performed if there is no diode-connected transistor on the problematic current path. The transistor is inserted in series in-between the supply node and the first device of that current path and must be sized large enough to carry the bias current of that branch in normal operation. The serial rip-up furthermore introduces an additional voltage drop in that path which could affect the normal operation of the circuit. Hence, the serial rip-up is the last resort and a diode rip-up should be preferred whenever possible [12].

1.2. State of the Art

The manual design of the analog power-down mode is error prone, repetitive and tedious. Unintended current flows, floating nodes and mismatched voltages can be easily overseen in more complex circuits. This is especially true for subtle design errors as leakage currents through the bulk substrates of transistors [9]. Therefore, automatic tools to detect such errors are required.

The power-down verification method presented by [8] detects current paths and floating nodes in power-down mode using voltage propagation. The algorithm initializes the supply and ground nets with their corresponding voltage level; the internal nets are set to floating. Starting from the ground and supply nets, the algorithm scans the circuit for conducting devices and propagates the supply and ground voltage into the circuit. This process is repeated until no more propagation occurred. Conducting devices which connect the supply with the ground voltage are reported back to indicate that a current flow in the circuit has been found. The method supports resistors, diodes and MOS transistors and guarantees to find all potential current flow in the circuit.

The verification method presented in [10][1] models the static DC behavior of a circuit in power-down mode by a graph and uses voltage propagation to estimate its node voltages. Current flows are detected by a depth-first search from the supply to the ground nets of the circuit. The method additionally uses structure recognition[2] to identify basic analog building blocks, e.g., differential pairs and current mirrors, in the circuit which must be matched. The identified structures are then compared against the voltage estimates to detect mismatch in power-down mode.

The method presented by [18] shows how constraint programming can be used to detect floating nodes efficiently in power-down mode.

These tools are of great help for circuit designers as they either automatically verify the correct power-down mode functionality of a circuit or indicate error spots in the current power-down circuitry.

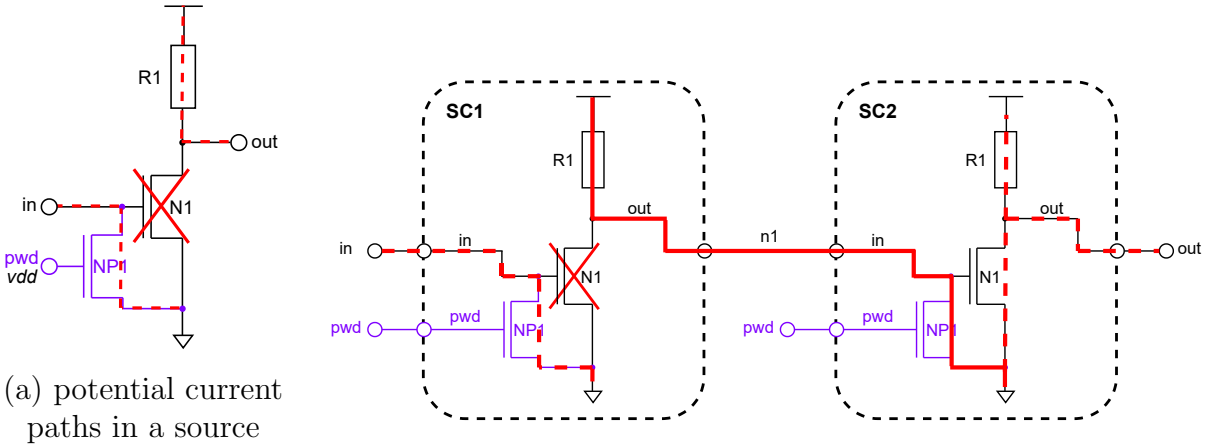
The authors of [10], however, even showed that the manual power-down mode design process can be automated which leaves designers with more time for more creative tasks. The presented method uses a set of rules to construct faultless power-down circuitry which has been derived from the verification rules presented in the same work. These rules are used to formulate two constraint programs [19] whose solutions guarantee the correct power-down mode behavior of the circuit. The constraint programs are modelled and solved using [20]. The first constraint program computes all possible net voltage combinations in power-down mode such that no current flow and no floating node can occur. Furthermore, matching constraints based on structure recognition results are formulated which must be fulfilled as good as possible. The second constraint program determines the number of required power-down transistors to implement such a net voltage combination and inserts them at the correct locations, i.e., at the nets which must be pulled to the supply or ground voltage, into the circuit. The solutions which cannot be implemented with a minimum number of power-down transistors are thereby discarded. Finally, a solution is manually selected from the remaining ones for implementation. Before formulating and solving the constraint programs, the circuit is scanned for problematic current paths which require a rip-up. The presented algorithm recursively checks for each net of the circuit if there is a conducting path from that net to the supply voltage which cannot be switched off by the gate shut-off pattern. If there is additionally another conducting path from that net to the ground voltage which cannot be switched off by the gate shut-off pattern, a problematic current path has been detected. A diode or serial rip-up is then applied to each identified problematic current path which guarantees at least one solution of the two constraint programs.

Power-down transistor sizing was investigated in [14] with the conclusion that the minimum transistor sizes are adequate for most applications. Larger transistors are only necessary for designs which require fast transition times from the normal operation to the power-down mode. The wake-up time, i.e., the transition time from the power-down to the normal operation mode, is only determined by the original circuitry and cannot be influenced by power-down transistor sizing.

The paper by [15] describes how the power-down synthesis results from [10] can be visualized by displaying the power-down transistors in the schematic editor of Cadence Virtuoso [21]

[1] Preliminary works: [9; 13; 11; 14; 15; 12]

[2] The structure recognition method by [10] is based on the "sizing rules method" which was originally published in [16]. The sizing rules method has been subsequently enhanced and extended by [4; 17; 10]



(a) potential current paths in a source circuit (SC)

(b) current path in hierarchical source circuit series (SCS)

Figure 1.7.: Potential current flows in a source circuit (a), faulty power-down mode implementation for a source circuit series (b).

which indicates that the synthesis method can be seamlessly integrated into an analog design flow.

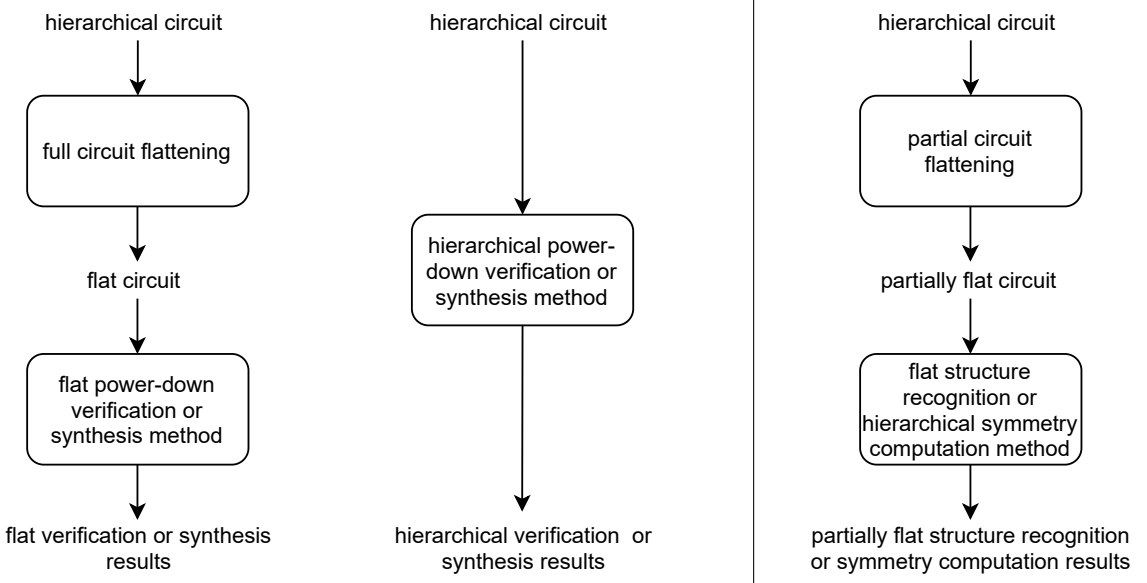
1.3. Problem Formulation and Contributions

The power-down verification and synthesis methods described by [8; 10] operate on transistor level. Complex chips however undergo a hierarchical design process: the desired functionality is first translated into a technical specification which describes the overall system behavior. That system is then successively divided into smaller and smaller parts, until single blocks can be implemented on device level [22].

A chip is thereby divided into analog and digital parts, the analog parts are sub-divided into, e.g., filters, phase locked loops (PLLs), analog to digital converters (ADCS) or digital to analog converters (DACs). These circuits are, e.g., composed of RC networks and operational amplifiers (OpAmps) which can be implemented on device level. Afterwards, the single components are assembled to form the overall system.

The power-down verification and synthesis methods by [8; 10] do not consider the interaction with and the connectivity to external circuitry which occurs in hierarchical analog designs. This can lead to design errors in the power-down mode as will be illustrated with the following example.

Fig. 1.7a shows the already known source circuit (SC) from Fig. 1.4b. The dashed red lines indicate a potential current flow in the power-down mode of the circuit which depends on the external circuitry connected to the input and output pins of the amplifier. Its input pin *in* is connected to *gnd* by the conducting power-down transistor $NP1$. A new and unintended current path would arise in the source circuit if that pin is connected to *vdd* by conducting external circuitry. The output pin *out* of the source circuit is furthermore connected to *vdd* by the resistor $R1$. Hence, an external, conducting connection to *gnd* with that pin would create another current path in power-down mode.



(a) state of the art [10] (b) hierarchical approach (c) partial circuit flattening

Figure 1.8.: Data flows of the state-of-the-art (a) and the proposed power-down verification and power-down synthesis method (b) for hierarchical analog circuits. The diagram on the right (c) shows the data flow for the new structure recognition and symmetry computation methods.

This situation is illustrated by Fig. 1.7b. Two source circuits $SC1$ and $SC2$ are connected with each other to form a source circuit series (SCS), i.e., an amplifier with two amplification stages. Now, the output pin of $SC1$ is connected to the input pin of $SC2$ which results in a current flow over the resistor $SC1/R1$ and the transistor $SC2/NP1$ in power-down mode. Such new short circuit paths, created by hierarchical circuit composition, must be detected and prevented.

The methods presented by [10] must flatten a given hierarchical design in order to be able to support it as shown in Fig. 1.8a. Circuit flattening replaces instantiated subcircuit blocks by their netlist implementation until the device level is reached. This approach does not preserve the circuit hierarchy and is hence not suitable for industrial and IP based designs. Furthermore, intermediate verification and synthesis results on the block level cannot be reused which increases the computational complexity of the verification and synthesis tasks.

This work presents a new power-down verification method and a new power-down synthesis method for hierarchical circuits which overcome these issues as shown in Fig. 1.8b. The new methods are based on [10] with the following extensions:

- all algorithms have been developed in order to preserve the circuit hierarchy. The implemented verification and synthesis methods traverse the circuit hierarchy either in a top-down or bottom-up approach or in both directions.
- intermediate results for all encountered subcircuit blocks are stored in libraries to enable their reuse and to avoid costly recomputations.

The methods presented by [10] do not consider symmetries alongside the signals paths of a given circuit. Such symmetries pose additional matching conditions on the power-down

circuitry which are now incorporated into the new power-down verification and synthesis methods.

Furthermore, the following enhancements of the analog power-down synthesis have been created:

- the rip-up procedure by [10] has been simplified by using methods from the verification process to identify short circuit paths.
- the two constraint programs by [10] have been unified which allows to compute trade-offs between the two design goals "maximize matching" and "minimize area".

The new hierarchical methods require automatic procedures to extract analog basic building blocks and signal symmetries from the circuit hierarchy in order to be able to verify and synthesize power-down circuitry considering voltage matching. Hence, a new structure recognition method has been developed based on [10] that supports hierarchical circuits following a similar approach as presented in [23]. Fig. 1.8c shows that the new approach flattens the circuit hierarchy partially, i.e., structural information connected to the pins of a subcircuit block is propagated between the hierarchy levels when required for the computations. Partial circuit flattening decouples the hierarchy levels from each other, which enables and allows the parallelization of traditional structure recognition approaches for hierarchical circuits. Furthermore, a symmetry computation method similar to [24] has been implemented and adjusted to support hierarchical circuits. The algorithm starts at the device level and propagates the collected signal flow information between the input and output pins of subcircuit blocks to the next higher hierarchy levels recursively until the top level is reached.

Last but not least, a new method to automatically generate the building block description of a given circuit has been developed. This method overcomes the problems of traditional structure recognition procedures: the scope and the complexity of the provided building block library. The library of the sizing rules method contains building blocks formed by only up to six devices [10]. New and more complicated structures, e.g., operational amplifiers, and their recognition rules must be added manually to the library which is prone to errors and can lead to wrong recognition results. The new procedure automatically partitions a given circuit into its building blocks without requiring a predefined library. Instead, three characteristics, i.e., connectivity, substrate type and tier difference of neighboring devices, are analyzed and used to determine the subblocks of the circuit. The algorithm then generates the recognition rules for the detected elements and furthermore creates new building block libraries from scratch or complements existing ones by new elements.

1.4. Thesis Structure and Prior Publications

This work is partitioned into two parts.

The first part focuses on the new power-down verification and synthesis method of hierarchical analog circuits (Chapters 3 and 4). Both methods rely on the hierarchical circuit representation described in Chapter 2.

The hierarchical structure recognition and symmetry computation method complementing the verification and synthesis procedures are described in Appendices A and B. Appendix C provides guidelines on how to manually fix detected errors in the analog power-down mode.

The second part of this work describes the new library-free structure recognition method (Chapter 5).

Each chapter provides experimental results for the respective methods.

Chapter 6 concludes this thesis.

The presented methods, algorithms and ideas have been partially published at conferences or journals. The hierarchical power-down verification method from Chapter 3 was originally published in [25], an extended version can be found in [26]. The unified constraint program and the hierarchical power-down synthesis method from Chapter 4 have been presented in [27] and [28], respectively. The library-free structure recognition method from Chapter 5 has been published at [29].

Additionally, the presented methods, other ideas and previous implementations have been partially realized in the student theses [30; 31; 32; 33; 34; 35; 36].

2. Hierarchical Circuit Representation

Hierarchical designs are formally constructed by a set of circuits

$$\mathcal{C} = \{C_1(P_1, N_1, C_{sub,1}, t_1), C_2(P_2, N_2, C_{sub,2}, t_2), ..., C_{|\mathcal{C}|}(P_{|\mathcal{C}|}, N_{|\mathcal{C}|}, C_{sub,|\mathcal{C}|}, t_{|\mathcal{C}|})\}. \tag{2.1}$$

A circuit $C_i(P_i, N_i, C_{sub,i}, t_i)$ consists of

- a set of pins $P_i = \{p_{i,1}, p_{i,2}, ..., p_{i,|P_i|}\}$,
- a set of nets $N_i = \{n_{i,1}, n_{i,2}, ..., n_{i,|N_i|}\}$,
- its subcircuits $C_{sub,i} = \{c_{sub,i,1}, c_{sub,i,2}, ..., c_{sub,i,|C_{sub,i}|}\}$,
- and is of type t_i, e.g., SCS or SC.

A subcircuit $c_{sub,i,j} \in C_{sub,i}$ instantiates another circuit from set $\mathcal{C}$. A net $n_{i,k} \in N_i$ connects one or more subcircuits of $C_{sub,i}$ with each other via their pins. The set of pins $P_i \subseteq N_i$ is a subset of the circuit's nets that can be connected to external circuitry. Each subcircuit $c_{sub,i,j}$ has its own set of subcircuits, hence forming a hierarchical circuit recursively.

A device is abstracted as a circuit which does not contain any other subcircuits and its set of pins is identical to its set of nets:

$$\text{“}C_i \in \mathcal{C} \text{ is a device”} \;\Leftrightarrow\; C_{sub,i} = \emptyset \;\wedge\; P_i = N_i. \tag{2.2}$$

The following device types $t_i \in T_d$ are thereby supported:

$$T_d = \{nmos, pmos, npn, pnp, res, cap, diode, ind\}. \tag{2.3}$$

The methods presented in Chapters 3-5 are however not limited to these device types and can be easily extended by additional device types.

Fig. 2.1 shows four different devices types of T_d and their corresponding pins. The drain, gate and source pins of the transistors are denoted as d, g, and s, respectively. The pins of a resistor or capacitor are *plus* (+) and *minus* (−).

The top-level circuit C_{top} of a hierarchical design is not part of any other circuit of set $\mathcal{C}$. The bottom level of a circuit hierarchy is reached with the device level.

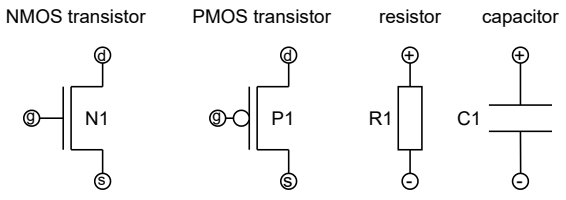


Figure 2.1.: Different device types and their corresponding pins.

circuit type t_i	circuit pins P_i	circuit nets N_i	subcircuits $C_{sub,i}$	subcircuit type $t_{sub,i,j}$
source circuit series (SCS)	$in, out,$ pwd	$in, out,$ $pwd, n1$	$SC1, SC2$	SC
source circuit (SC)	$in, out,$ pwd	$in, out,$ pwd	$N1, NP1$ $R1$	$nmos$ res
$nmos$	d, g, s	d, g, s	$\emptyset$	$-$
res	$+, -$	$+, -$	$\emptyset$	$-$

Ground and supply nets not shown.

Table 2.1.: Set of circuits $\mathcal{C}$ used to construct the source circuit series from Fig. 1.7.

Table 2.1 shows the set $\mathcal{C}$ for the source circuit series from Fig. 1.7b. The circuit consists of three pins, four nets and the two source circuits $SC1$ and $SC2$. Each of these subcircuits has three pins, three nets and three devices, i.e., two NMOS transistors and one resistor.

3. Hierarchical Power-Down Verification

Power-down mode verification checks whether the design goals (A) - (C) from Sec. 1.1 are met for a given circuit. Hence, power-down verification must provide methods

- to detect all (potential) current flows,
- to identify all floating nodes and
- to recognize all (potential) matching violations

of a circuit in power-down mode. A circuit fulfils (A) - (C) if the verification method does not detect any of the errors listed above. In the following, a new power-down verification method for hierarchical designs is presented.

3.1. Overview

Fig. 3.1 gives an overview of the new hierarchical power-down verification methodology. It is based on the methods presented in [10]. In comparison to the state-of-the-art methods, the new method does not flatten the circuit hierarchy and additionally verifies that symmetry pairs alongside symmetrical signal paths are not stressed in power-down mode. The newly introduced voltage propagation library L_V furthermore enables the reuse of intermediate results.

The method takes a top-level circuit C_{top}, the supply nets $N_{supply} = N_{vdd} \cup N_{gnd}$ and the vector $V_{P,top}$ as inputs. The vector $V_{P,top}$ defines the initial voltage levels at the pins of C_{top}. The verification report contains all detected current paths P_{sc}, floating nets N_{float} and the matching and symmetry violations E_M, E_S of the circuit in power-down mode.

In the first step, voltage propagation estimates the net voltages of the circuit in power-down mode (Sec. 3.2). This method starts at the top-level and propagates the voltages on the current level as far as possible before descending into subcircuit blocks. The algorithm traverses the circuit hierarchy in both directions as a change in the voltage level at a subcircuit pin can affect the states of the devices in the underneath or above lying hierarchy levels.

In the second step, a depth-first search algorithm detects all short circuit paths in the circuit based on the previously computed net voltage estimates (Sec. 3.3). This algorithm also traverses the circuit hierarchy in both directions as a current path can descend and ascend to lower or higher hierarchy levels through subcircuit pins.

The voltage levels at the internal nets of a short circuit path cannot be accurately predicted anymore as they would take a value in-between *gnd* and *vdd*. Hence, the previously computed voltage propagation result is not valid anymore and must be recomputed. In this case, the detected short circuit nodes N_{sc} are initialized with the voltage level *sc* which indicates that a short circuit path is running other them. The subsequent voltage propagation run might eventually reveal further potential current flows in the circuit based on this new information.

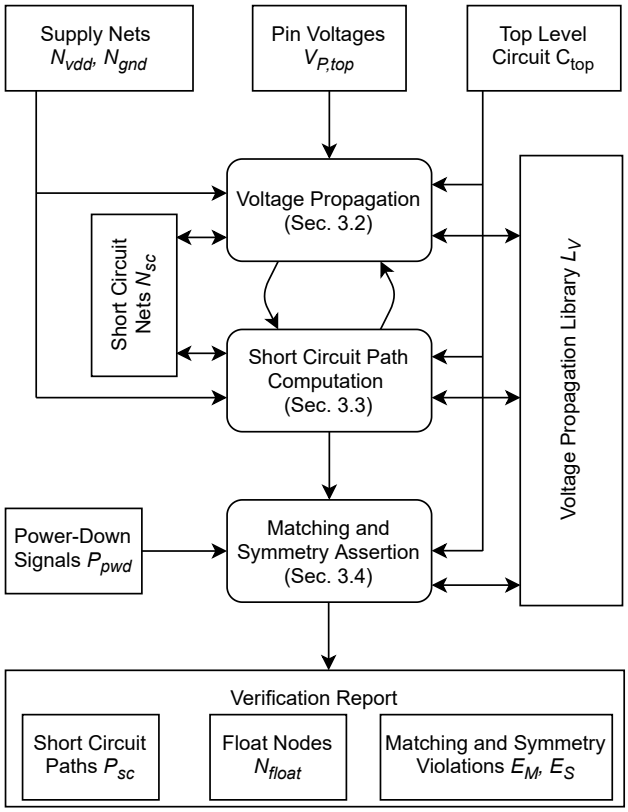


Figure 3.1.: Overview of the hierarchical power-down verification method.

Hence, voltage propagation and short circuit path computation are repeated until no more new currents paths are detected.

The last step of the verification method identifies matching and symmetry violations at analog basic building blocks and symmetry pairs (Sec. 3.4). The checks traverse the circuit hierarchy recursively from the top to its bottom using the final voltage propagation results to evaluate predefined rule sets for basic building blocks and symmetry pairs.

3.2. Voltage Propagation

The behavior of a circuit in power-down mode is determined by the static DC behavior of its devices [9]. Its internal nets are either charged or discharged to *vdd* or *gnd* by conducting devices or are high-impedant, i.e., floating $float$. Hence, the net voltages of a circuit in power-down mode can be estimated by voltage propagation.

The supply or ground voltage can propagate to another net of the circuit via conducting devices. Fig. 3.2 shows the static DC behavior of resistors, capacitors and NMOS transistors. A resistor propagates the ground and supply voltage between its plus and minus pin in both directions. A capacitor is treated as open loop in DC operation and does not propagate any voltage in power-down mode. An NMOS transistor propagates *vdd* or *gnd* between its drain and source pins when it is conducting, i.e., when its gate-source voltage is greater than its threshold voltage. In power-down mode, all nets will ideally either take the *vdd* or *gnd* voltage level. Hence, NMOS transistors are conducting if their gate pin is exposed to *vdd*. Similarly, PMOS transistors only propagate voltages between their drain and source pins if

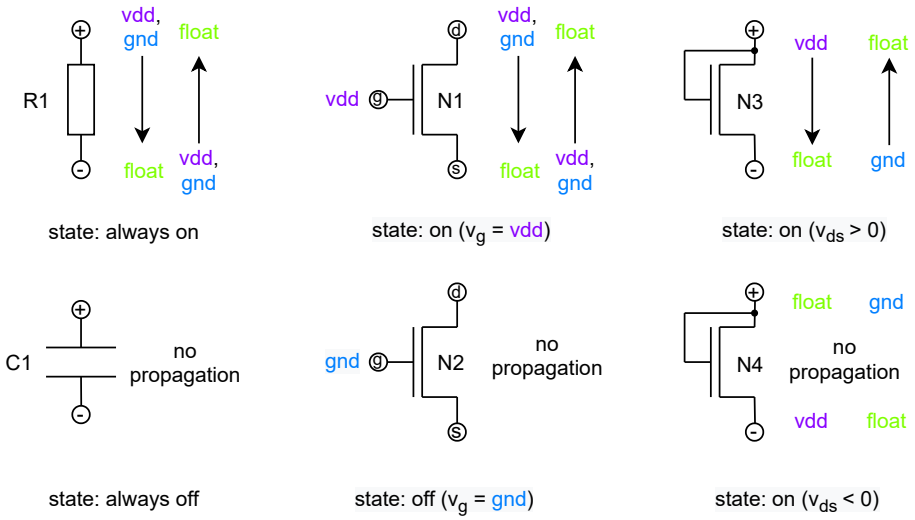


Figure 3.2.: Static DC behavior for different device types.

their gate is on *gnd*. A diode-connected transistor propagates the supply voltage *vdd* from its drain to its source pin and the ground voltage *gnd* in the opposite direction. A complete overview over all supported device types, including bipolar transistors and bulk junction diodes of transistors, can be found in [9].

Algorithm 1 uses the described static DC behavior to estimate the node voltages of a hierarchical circuit in power-down mode. The algorithm takes a circuit $C_i(P_i, N_i, C_{sub,i}, t_i)$, the voltages at its pins V_{P_i}, the supply nets $N_{supply} = N_{vdd} \cup N_{gnd}$, the short circuit nets N_{sc} and the voltage propagation results library L_V as inputs.

The library L_V allows to store and fetch intermediate voltage propagation results V_{N_i} for a specific circuit type t_i with pin voltages V_{P_i}. The vector

$$V_{N_i} = (v_{n_{i,1}}, v_{n_{i,2}}, ..., v_{n_{i,|N|}}) \tag{3.1}$$

with

$$v_{n_{i,j}} \in \{gnd, vdd, float, sc\} \tag{3.2}$$

contains the voltage estimates of each net $n_j \in N_i$ of a circuit of type t_i in power-down mode.

The set of short circuit nodes N_{sc} is empty in the initial voltage propagation run.

The algorithm iteratively propagates *gnd* and *vdd* to the internal nets of C_i via conducting devices according to Fig. 3.2. The voltages are thereby propagated as far as possible on the current hierarchy level before descending into subcircuit blocks, i.e., devices are always prioritized to subcircuit blocks during voltage propagation. This behavior is ensured by the propagation queue $Q_{prop} = (c_{sub,1}, c_{sub,2}, \ldots, c_{sub,k})$ which orders the inserted subcircuits such that devices are processed first. Subcircuits are queued up for propagation whenever a voltage level at one of its pins changed from $float$ to *vdd* or *gnd*. For devices, an additional device type specific propagation condition must be fulfilled before it is inserted into Q_{prop}: the device must be conducting and *vdd* or *gnd* at one of its pins must be connected to a floating net at one of its other pins via its conducting channel as shown in Fig. 3.2.

Algorithm 1 Voltage propagation for hierarchical circuits based on [9]

```
1: procedure VoltagePropagation(C_i(P_i, N_i, C_sub,i, t_i), V_Pi, N_supply, N_sc, L_V)
2:     Q_prop, V_Ni = initialize(V_P,i, C_i, N_supply, N_sc)
3:     for all c_sub ∈ Q_prop do
4:         V_Psub = pinVoltages(c_sub, V_Ni)
5:         if not hasResult(c_sub, V_Psub, L_V) then
6:             if isDevice(c_sub) then
7:                 propagateDevice(c_sub, V_Psub, L_V)    // According to Fig. 3.2
8:             else
9:                 voltagePropagation(c_sub, V_Psub, N_supply, N_sc, L_V)    // Descend hierarchy
10:            end if
11:        end if
12:        V_Nsub = findResult(c_sub, V_Psub, L_V)
13:        update(Q_prop, V_Ni, V_Nsub, C_i)
14:    end for
15:    storeResult(C_i, V_Pi, V_Ni, L_V)
16: end procedure
```

Algorithm 1 initializes the propagation queue Q_{prop} and the net voltages V_{N_i} in line 2 by invoking the function $initialize(V_{P,i}, C_i, N_{supply}, N_{sc})$. This function sets the pin, ground, supply and short circuit nodes from $V_{P,i}$, N_{gnd}, N_{vdd} and N_{sc} to their corresponding voltage levels. Subcircuits and devices which are connected to these nets and which are eligible for voltage propagation are inserted into Q_{prop} during that initialization step. The remaining nets of the circuit are assigned to voltage level $float$.

In the next step, the algorithm iterates over all elements $c_{sub} \in Q_{prop}$ (line 3). In each iteration, the pin voltages $V_{P_{sub}}$ of c_{sub} are determined in line 4 based on the current state of V_{N_i} by the function $pinVoltages(c_{sub}, V_{N_i})$. If another subcircuit of the same type and pin voltages as c_{sub} has already been encountered during voltage propagation, the corresponding intermediate result is reused (lines 5 and 12). Else, the voltages at c_{sub} must be propagated (lines 6 - 10). The following two cases are thereby distinguished:

- c_{sub} is a *device*: voltages are propagated according to Fig. 3.2.
- c_{sub} is a *subcircuit block*: the algorithm descends into the circuit hierarchy by calling itself recursively.

The corresponding propagation result $V_{N_{sub}}$ is stored in L_V in both cases. In line 12, that result is fetched from the library and used to update Q_{prop} and V_{N_i} of the current hierarchy level. The nets connected to the pins of c_{sub} are set to the voltage levels according to $V_{N_{sub}}$ and any subcircuits connected to these nets are inserted into Q_{prop}. The above steps are repeated until Q_{prop} is empty.

Finally, in line 15, the voltage propagation result V_{N_i} for the current hierarchy level is stored in L_V under the entry (C_i, V_{P_i}).

Tables 3.1 and 3.2 show the voltage propagation results for the source circuit series (SCS) of Fig. 1.7b and its two source circuits $SC1$, $SC2$ from Fig. 1.7a, respectively. The pin voltages of the source circuit series are given as $V_{P_{SCS}} = (v_{in}, v_{out}, v_{pwd}) = (float, float, vdd)$ and its supply voltages as $N_{supply} = N_{gnd} \cup N_{vdd} = \{ground, supply\}$. Algorithm 1 initializes the nets of the top-level circuit accordingly (see Table 3.1, circuit SCS, iteration one, initial

circuit	iteration	state	v_{in}	v_{out}	v_{n1}	v_{ground}	v_{supply}	v_{pwd}
SCS	①	initial	$float$	$float$	$float$	gnd	vdd	vdd
		end	gnd	vdd	vdd	gnd	vdd	vdd
	②	initial	$float$	sc	sc	gnd	vdd	vdd
		end	gnd	sc	sc	gnd	vdd	vdd

Table 3.1.: Voltage propagation results for the source circuit series (SCS) of Fig. 1.7b.

subcircuit	iteration	state	v_{in}	v_{out}	v_{ground}	v_{supply}	pwd
$SC1$	①	initial	$float$	$float$	gnd	vdd	vdd
		end	gnd	vdd	gnd	vdd	vdd
	②	initial	$float$	sc	gnd	vdd	vdd
		end	gnd	sc	gnd	vdd	vdd
$SC2$	①	initial	vdd	$float$	gnd	vdd	vdd
		end	vdd	vdd	gnd	vdd	vdd
	②	initial	sc	sc	gnd	vdd	vdd
		end	sc	sc	gnd	vdd	vdd

Table 3.2.: Voltage propagation results for the source circuits $SC1$ and $SC2$ from Fig. 1.7a.

state) and queues up the source circuits $SC1$ and $SC2$ for voltage propagation as both subcircuits are connected to the power-down signal pwd, i.e., $Q_{prop,SCS} = (SC1, SC2)$. In the next step, the algorithm descends into $SC1$ by recursively calling itself with pin voltages $V_{P,SC1} = (v_{in}, v_{out}, v_{pwd}) = (float, float, vdd)$, which are determined based on the current net voltage estimates of the source circuit series, and the supply nets N_{supply}. The algorithm initializes the net voltages of source circuit $SC1$ accordingly and sets the ground and supply node to their corresponding values (see Table 3.2, subcircuit $SC1$, iteration one, initial state). The resistor $R1$ and transistor $NP1$ of $SC1$ are hence conducting in power-down mode and propagate vdd to the output pin out and gnd to the input pin in of $SC1$, respectively. Afterwards, the algorithm ascends back to the top-level, i.e., to the source circuit series, and updates its net voltages v_{in} and v_{n1} to gnd and vdd. It then descends into source circuit $SC2$, initializes its net voltages according to $V_{P,SC2} = (v_{in}, v_{out}, v_{pwd}) = (vdd, float, vdd)$ and N_{supply}. The devices $R1$ and $N1$ of $SC2$ are inserted into the propagation queue $Q_{prop,SC2}$. Both devices are connected to the output pin of $SC2$ and are eligible for propagation. In this case, resistor $R1$ has been randomly selected first for propagation and the output pin voltage of $SC2$, i.e., v_{out}, is set to vdd. Finally, the algorithm ascends back to the schematic of the source circuit series and updates the voltage at its output pin v_{out} to vdd.

3.3. Short Circuit Path Computation

Algorithm 2 detects current flow of a circuit in power-down mode based on the net voltages computed by voltage propagation (Sec. 3.2). It performs a depth-first search from the supply to the ground nets of the circuit alongside conducting devices (Algorithm 3). Both algorithms are based on [9] and have been adapted to hierarchical designs.

Algorithm 2 Short circuit path computation based on [9]

1: **procedure** ShortCircuitPathComputation($C_{top}, N_{vdd}, N_{gnd}, V_{P_{top}}, L_V$)
2: **for all** $n \in N_{vdd}$ **do**
3: $DepthFirstSearch(n, C_{top}, N_{gnd}, V_{P_{top}}, L_V)$ // Algorithm 3
4: **end for**
5: **end procedure**

Algorithm 3 Depth-first search based on [9]

1: **procedure** DepthFirstSearch($n, C_i, N_{gnd}, V_{P_i}, L_V$)
2: **if** $n \in N_{gnd}$ **then**
3: **return** // short circuit path detected.
4: **end if**

5: $V_{N_i} = findResult(C_i, V_{P_i}, L_V)$
6: **if** $connectedToHigherLevel(n)$ **then**
7: $DepthFirstSearch(n_{super}, C_{super}, N_{gnd}, V_{P_{super}}, L_V)$ // Ascend hierarchy
8: **end if**

9: **for all** $C_{sub} \in connectedSubCircuitBlocks(n)$ **do**
10: $V_{P_{sub}} = pinVoltages(C_{sub}, V_{N_i})$
11: $DepthFirstSearch(n_{sub}, C_{sub}, N_{gnd}, V_{P_{sub}}, L_V)$ // Descend hierarchy
12: **end for**

13: **for all** $n_{con} \in connectedNets(n)$ **do**
14: $DepthFirstSearch(n_{con}, C_i, V_{P_i}, L_V)$ // Continue on this level
15: **end for**
16: **end procedure**

The depth-first search algorithm first checks whether its termination criterion is fulfilled, i.e., whether the given net n is a ground net (lines 2 - 4). In this case, a short circuit path has been found and the algorithm returns.

The search continues in all other cases.

In line 5, the algorithm loads the net voltages $V_{N,i}$ for the currently investigated circuit C_i with input voltages $V_{P,i}$ from the voltage propagation results library L_V by calling function $findResult(C_i, V_{P_i}, L_V)$. These net voltages are used to determine the conducting or non-conducting state of the devices on the current hierarchy level and the pin voltages of lower-level circuit blocks before descending the circuit hierarchy.

The algorithm thereby checks with function $connectedToHigherLevel(n)$ whether the given net n is connected to external circuitry, i.e., whether n is a pin of the current circuit, and ascends the circuit hierarchy when required by calling itself recursively (lines 6 - 8).

The depth-first search furthermore has to determine whether it has to descend into the circuit hierarchy (lines 9 - 12). Thereby, function $connectedSubCircuitBlocks(n)$ returns all subcircuits of C_i connected to net n. The depth-first search algorithm then calls itself recursively for every subcircuit block connected to n with n_{sub}, C_{sub} and $V_{P_{sub}}$ as input arguments. The net n_{sub} denotes the net, i.e., pin, of C_{sub} which is connected to net n of the current hierarchy level. The function $pinVoltages(C_{sub}, V_{N_i})$ determines the subcircuit's pin voltages $V_{P_{sub}}$ based on the current net voltages V_{N_i} of the current hierarchy level (line 10).

Lastly, the search continues on the current hierarchy level in lines 13 - 15 by calling itself recursively for all nets n_{con} connected to n by conducting devices. This set of nets is thereby computed by function $connectedNets(n)$ which determines the conducting or non-conducting state of each device connected to n based on V_{N_i}.

During the computations, the algorithm must track the hierarchy levels it has traversed in order to be able to ascend it again, as a circuit C_i does not know where in the hierarchy it has been instantiated by another circuit. This part of the algorithm is not shown, but the hierarchy path is tracked by a vector which stores tuples of the visited subcircuits and their corresponding pin voltages $(C_{super}, V_{P_{super}})$. Whenever the depth-first search algorithm descends into a subcircuit, the corresponding tuple is appended to that vector and whenever the algorithm ascends, the last visited subcircuit is restored from that vector thus allowing the algorithm to traverse the hierarchy in both directions.

A detected short-circuit path can be classified into one of the following three categories [10]:

- definite: all devices on the detected path are conducting. The path cannot be switched off by the gate shut-off pattern and a rip-up required as shown in Fig. 1.6.
- potential: at least one device on the detected path is only potentially conducting, e.g., the gate of a transistor on that path is floating. Such a path can be switched off by the gate shut-off pattern by inserting a power-down transistor which forces the corresponding device into the non-conducting state.
- induced: the detected path is potentially conducting due to another definite, potential or induced short-circuit path in power-down mode, i.e., the state of at least one of its devices is controlled by a short-circuit node. Such a path can be typically switched off by switching off the short-circuit path causing it.

The source circuit series from Fig. 1.7b contains one supply net *supply* which is connected to the source circuits $SC1$ and $SC2$. Short circuit path computation loads the voltage propagation result from the first iteration for $SC1$ from Table 3.2 and descends into it. The supply net is connected to the output pin *out* of $SC1$ via the resistor $R1$. The output pin of $SC1$ is connected to net $n1$ of the source circuit series which again is connected to the input pin of $SC2$. Hence, the algorithm consecutively first ascends back to the top-level and then descends down into $SC2$. The input pin *in* of $SC2$ is connected to the ground node via power-down transistor $NP1$. This transistor is conducting in power-down mode according to Table 3.2 and hence a short circuit path, i.e., $P1 = (SC1/R1, SC2/R2)$, has been found. The net $n1$ of the source circuit series and the nets $SC1/out$ and $SC2/in$ lie on that short circuit path and are inserted into the set of short circuit nodes N_{sc}.

Short circuit path computation also descends into $SC2$ via the supply net of the source circuit series. This net is connected to the ground node by the devices $R1$ and $N1$ inside of $SC2$ which are both conducting in power-down mode according to the voltage propagation results from Table 3.2, iteration one. Hence, a second current flow $P2 = (SC2/R1, SC2/N1)$ has been found and the output pins of $SC2$ and the source circuit series are inserted into N_{sc} as these nets lie on $P2$.

Short circuit path computation terminates afterwards and voltage propagation is repeated with $N_{sc} = \{n1, out, SC1/out, SC2/in, SC2/out\}$. The corresponding voltage propagation result is shown by Tables 3.1 and 3.2, iteration two. The current path $P2$ is now classified as induced current flow as the state of transistor $N1$ inside of $SC2$ cannot be accurately predicted anymore as the input pin of $SC2$ is a short circuit node. No new current flow is detected in the second iteration of the short circuit path computation and voltage propagation must hence not be repeated anymore.

3.4. Matching and Symmetry Assertion

Fig. 3.3 gives an overview of the matching and symmetry assertion method for hierarchical circuits in power-down mode. The method is based on [13] and has been extended to support hierarchical circuits.

It takes the top-level circuit C_{top}, the pin voltages $V_{P_{top}}$, the power-down pins P_{pwd}, the voltage propagation library L_V and the supply nets N_{supply} of the circuit as inputs.

Structure recognition (App. A) and symmetry computation (App. B) are performed to identify the basic building blocks and symmetry pairs of the given circuit. The power-down circuitry has to be removed from the circuit in a preprocessing step as it might interfere with the above-mentioned methods. Finally, the identified building blocks and symmetry pairs are asserted by evaluating specific block type and symmetry pair matching rules against voltage mismatch. The detected matching and symmetry violations, E_M and E_S, are reported back to the user.

3.4.1. Power-up Transformation

Power-up transformation removes the power-down circuitry from a given circuit. It thereby tracks the power-down signals *pwd* and $\overline{pwd}$ through electrostatic discharge (ESD) protection

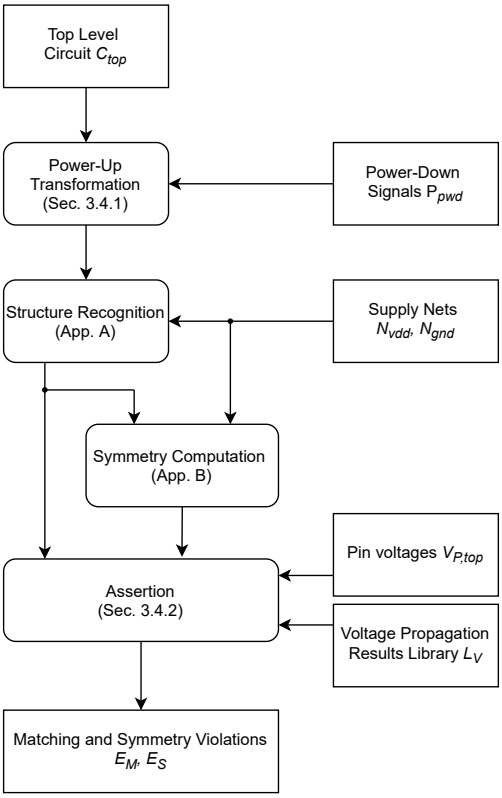
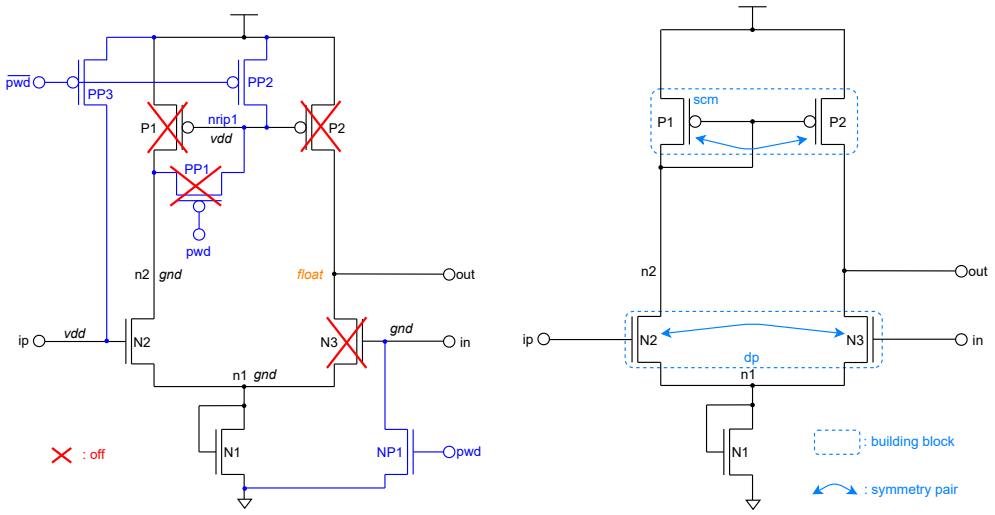


Figure 3.3.: Overview of the symmetry assertion procedure [13].

circuitry, level shifters and inverters of the integrated circuit until it reaches the gates of the power-down transistors inside the analog subcircuit blocks of a hierarchical design. The identified power-down transistors are then removed from the circuit as follows: transistors which are conducting in power-down mode pull an internal net of the circuit to *vdd* or *gnd* and have been inserted in parallel into the circuit. Hence, these transistors can simply be removed. Power-down transistors which are off in power-down mode have been inserted in series into the circuit to disrupt a current path in power-down mode. Hence, the drain and source nets connected to these transistors must be shorted after removing them. The state of the power-down transistors is determined based on the net voltage estimates stored in the voltage propagation results library L_V. Finally, the subcircuit blocks of the power-down signal path, e.g. inverters, are removed from the hierarchical design. The power-up transformation above has first been described in [13] and has been extended to support hierarchical designs.

Fig. 3.4a shows a differential stage with its power-down circuitry highlighted in blue. The voltage propagation result for this circuit is annotated with italic letters next to the net names. No short circuit path runs through the circuit.

Fig. 3.4b shows the differential stage after power-up transformation. The gate shut-off transistors $NP1$, $PP2$ and $PP3$ have been removed from the circuit. The nets $n1$ and $nrip1$ have been shorted after removal of diode rip-up transistor $PP1$. The removal of $PP1$ allows the identification of the simple current mirror $scm(P1, P2)$ by structure recognition which furthermore enables symmetry computation to detect the symmetry pairs $(N1, N2)$ and $(P1, P2)$ alongside the signal paths from the differential input (in, ip) to the single ended output *out* of the circuit.



(a) power-down mode (b) power-up mode

Figure 3.4.: Differential stage before (a) and after (b) power-down up transformation.

3.4.2. Assertion

Fig. 3.5a shows the matching rules for two analog basic building blocks, i.e., a differential pair and a simple current mirror. The voltages at the gate and drain pins of the differential pair should be pairwise matched in order to avoid asymmetric stress in power-down mode. Similarly, the voltages at the drain pins of a simple current mirror should be matched. Matching rules for other analog basic building blocks can be found in [10]. The work by [10] additionally introduced several levels of strictness for each rule set, e.g., a strict rule set for differential pairs would enforce the same voltage level at all of its pins to prevent any electrical field at its devices. In the following, the presented assertion methods are only described for matched pin pairs as stricter rule sets can be similarly evaluated.

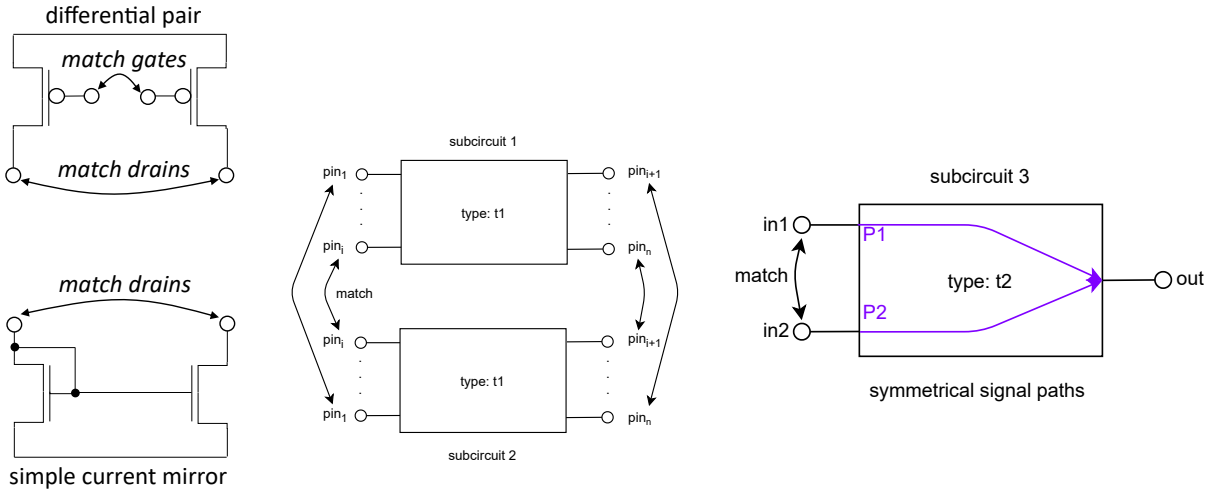
The matching rules for symmetry pairs in the signal paths of a circuit are shown in Fig. 3.5b. Same pin types should be exposed to the same voltage level to avoid asymmetric stress in power-down mode. Similarly to analog basic building blocks, strict rule sets for symmetry pairs could enforce the same voltage level at all of its pins.

Furthermore, when symmetrical signals paths inside a subcircuit block have been detected, their corresponding differential input and output pins should be matched as shown in Fig. 3.5c.

Building block and symmetry information at (sub)circuit pins are propagated in-between circuit hierarchy levels by structure recognition and symmetry computation as this information is important to reduce the solution space of the power-down synthesis constraint optimization problem (Sec. 4.3).

The evaluation of a matching rule, i.e., the comparison of the voltage levels at a pin pair, has three different outcomes:

- *fulfilled* (✓): both pins are exposed to the same defined voltage level (*vdd* or *gnd*). Any additional requirement, e.g., a specific voltage level at those pins, is fulfilled as well.
- *violated* (✗): both pins are exposed to different voltage levels and hence stressed. The voltage level *sc* is also treated as an error as there is a short circuit path in the circuit.



(a) (b) (c)

Figure 3.5.: Matching rules for: (a) analog basic building blocks [10], (b) symmetry pairs in signal paths and (c) symmetrical pin pairs of subcircuit blocks due to underlying signal path symmetries.

- *warning* (⚠): at least one of the pins is floating, i.e., it cannot be guaranteed that the matching condition is fulfilled.

Algorithm 4 shows the outline of the matching and symmetry assertion method. It takes a circuit $C_i(P_i, N_i, C_{sub,i}, t_i)$, its pin voltages V_{P_i} and the voltage propagation results library L_V as inputs. The libraries L_{struct} and L_{sym} contain all analog basic building blocks and symmetry pairs detected by structure recognition and symmetry computation (App. A and App. B). These correspond to the subcircuits and devices whose matching conditions have to be evaluated. The library L_{error} stores the detected rule violations and is filled in during the execution of the algorithm.

In lines 2 - 7, the algorithm first checks the matching rules for the identified building blocks and symmetry pairs on the current hierarchy level. It fetches the corresponding voltage propagation, structure recognition and symmetry computation results from the respective libraries by using different versions of the implemented $findResult()$ function (lines 2 - 4). The evaluation results, i.e., the sets of matching and symmetry violations E_M and E_S computed by $checkBuildingBlocks(C_B, V_{N_i})$ and $checkSymmetryPairs(C_{SP}, V_{N_i})$, are then stored under the entry (C_i, V_{P_i}) in L_{error} in lines 5 - 7.

In lines 8 - 15, the algorithm descends into the circuit hierarchy recursively and performs the symmetry checks for all subcircuit blocks c_{sub} of C_i if necessary. It first determines the corresponding pin voltages $V_{P_{sub}}$ of c_{sub} based on the net voltage estimates V_{N_i} (line 10). Function $hasResult(c_{sub}, V_{P_{sub}}, L_{error})$ checks, whether intermediate results for subcircuit c_{sub} with pin voltages $V_{P_{sub}}$ already exist in library L_{error} (line 11). If yes, that intermediate results is reused; if not, the matching and symmetry assertion algorithm calls itself recursively with the just determined pin voltage information (line 12).

Table 3.3 shows the matching and symmetry assertion results for the differential stage from Fig. 3.4a. A warning is issued for simple current mirror $scm(P1, P2)$ as its output pin, i.e., net *out*, is floating. Another warning is issued for the differential pair $dp(N1, N2)$ as one of its output pins, i.e., net *out* is floating. Furthermore, the voltages at the input pins of

Algorithm 4 Matching and symmetry assertion for hierarchical analog circuits

procedure MATCHINGASSERTION($C_i(P_i, N_i, C_{sub,i}, t_i), V_{P_i}$)

// Additional inputs:

- L_V: voltage propagation results library
- L_{struct}: structure recognition results library
- L_{sym}: symmetry computation results library
- L_{error}: matching and symmetry violations results library

// Perform checks for current hierarchy level

$V_{N_i} = findResult(C_i, V_{P_i}, L_V)$

$C_B = findResult(C_i, L_{struct})$

$C_{SP} = findResult(C_i, L_{sym})$

$E_M = checkBuildingBlocks(C_B, V_{N_i})$ // See Fig. 3.5a

$E_S = checkSymmetryPairs(C_{SP}, V_{N_i})$ // See Fig. 3.5b and Fig. 3.5c

$store(C_i, V_{P_i}, E_M, E_S, L_{error})$

// Descend into the circuit hierarchy when necessary

for all $c_{sub} \in C_{sub,i}$ **do**

  if $isSubCircuitBlock(c_{sub})$ **then**

    $V_{P_{sub}} = pinVoltages(c_{sub}, V_{N_i})$

    if not $hasResult(c_{sub}, V_{P_{sub}}, L_{error})$ **then**

      $matchingAssertion(c_{sub}, V_{P_{sub}})$

    end if

  end if

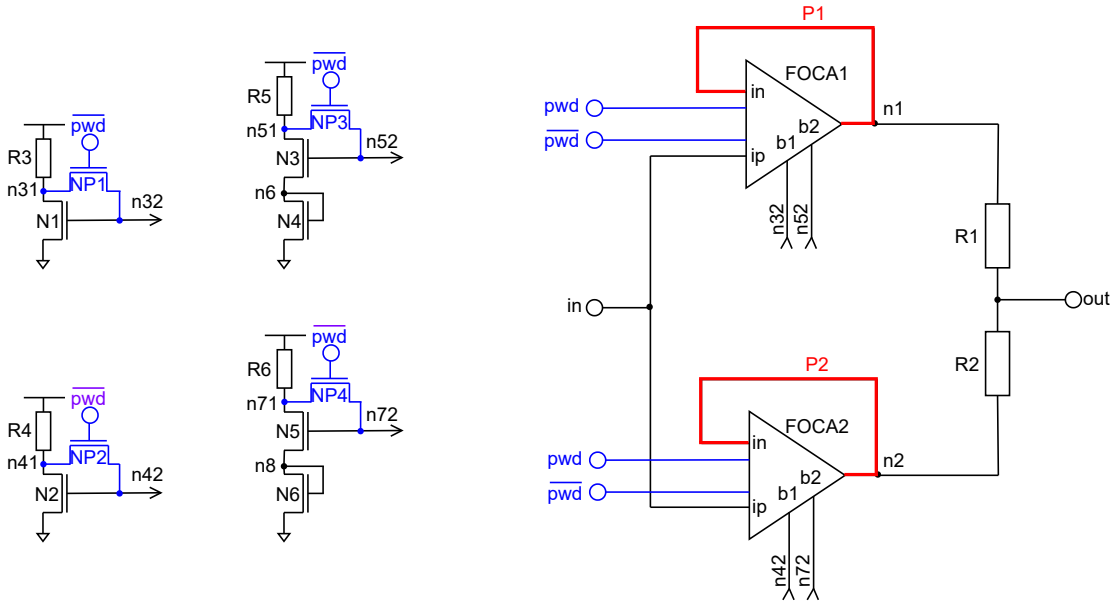
end for

end procedure

building-blocks// symmetry pairs	Evaluated pins	Evaluated nets	Result	
$scm(P1, P2)$	(in, out)	$(n2, out)$	$(gnd, float)$	⚠
$dp(N1, N2)$	$(in1, in2)$	(in, ip)	(gnd, vdd)	✗
$dp(N1, N2)$	$(out1, out2)$	$(n2, out)$	$(gnd, float)$	⚠
$(P1, P2)$	(d, d)	$(n2, out)$	$(gnd, float)$	⚠
	(g, g)	$(nrip1, nrip1)$	(vdd, vdd)	✓
	(s, s)	$(supply, supply)$	(vdd, vdd)	✓
$(N2, N3)$	(d, d)	$(n2, out)$	$(gnd, float)$	⚠
	(g, g)	(in, ip)	(gnd, vdd)	✗
	(s, s)	$(n1, n1)$	(gnd, gnd)	✓

evaluation result: fail (✗), pass (✓), warning (⚠)

Table 3.3.: Matching and symmetry assertion results for the differential stage from Fig. 3.4a.



(a) bias circuitry (b) core circuit [37]

Figure 3.6.: Bias circuitry (a) and core circuit (b) of a dual gain amplifier (DGA) with its power-down circuitry highlighted in blue. The subcircuits $FOCA1$ and $FOCA2$ are each implemented as the folded cascode amplifier from Fig. 3.7.

iteration	bias circuitry				core circuit			
	v_{nX1}	v_{nX2}	v_{n6}	v_{n8}	v_{in}	v_{out}	v_{n1}	v_{n2}
①	vdd	gnd	gnd	gnd	vdd	gnd	gnd	gnd
②	vdd	gnd	gnd	gnd	vdd	sc	sc	sc

nets $\overline{pwd}, ground$ always on gnd; nets $pwd, supply$ always on vdd;
$X = 3, 4, 5, 7$ in v_{nX1} and v_{nX2}

Table 3.4.: Voltage propagation results for the dual gain amplifier from Fig. 3.6.

$dp(N1, N2)$ differ from each other and hence a matching violation is issued. Two warnings, three fulfilled and one violated matching constraint are reported for the symmetry pairs $(P1, P2)$ and $(N2, N3)$. In summary, only 33% of all matching and symmetry constraints are fulfilled for the differential stage in power-down mode.

3.5. Experimental Results

3.5.1. Dual Gain Amplifier

Fig. 3.6 shows a dual gain amplifier (DGA). Its bias circuit consists of four current mirrors which provide the bias voltages to the core part of the circuit. The core circuit consists of two parallelly connected folded cascode amplifiers, i.e., $FOCA1$ and $FOCA2$, which doubles the output drive of the circuit [37]. The schematic of the corresponding folded cascode amplifier is shown in Fig. 3.7. The power-down circuitry of both amplifiers is highlighted in blue in both figures.

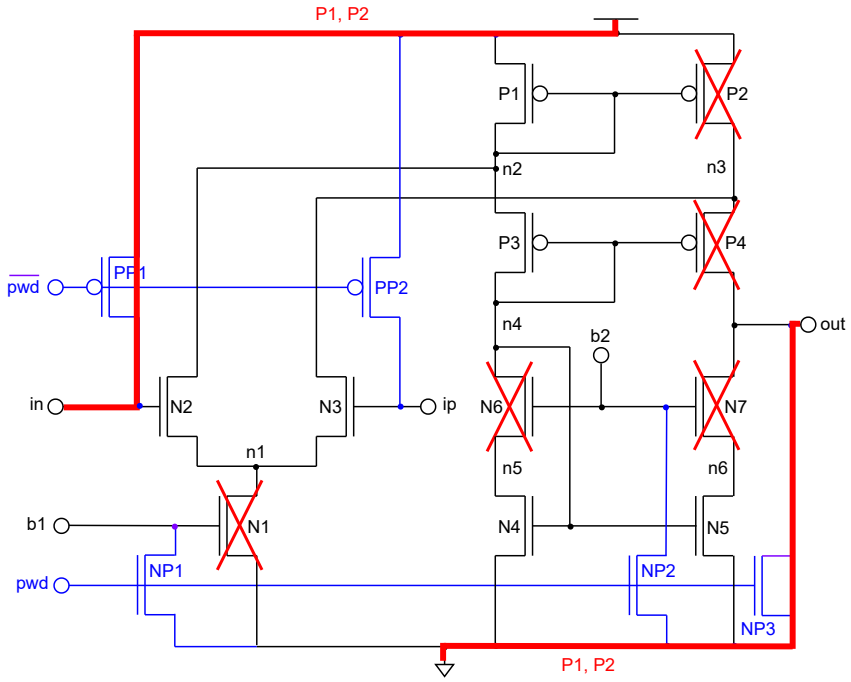


Figure 3.7.: Schematic of a folded cascode amplifier (FOCA) which implements subcircuits $FOCA1$ and $FOCA2$ of the dual gain amplifier from Fig. 3.6. Its power-down circuitry is highlighted in blue.

iteration	v_{in}	v_{ip}	v_{out}	v_{b1}	v_{b2}	v_{n1}	v_{n2}	v_{n3}	v_{n4}	v_{n5}	v_6
①	vdd	vdd	gnd	gnd	gnd	vdd	vdd	vdd	vdd	gnd	gnd
②	sc	vdd	sc	gnd	gnd	$float$	vdd	$float$	vdd	gnd	gnd

$\overline{pwd}, ground$ always on gnd; $pwd, supply$ always on vdd

Table 3.5.: Voltage propagation results for the folded cascode amplifiers from Fig. 3.7. The results are valid for $FOCA1$ and $FOCA2$ of the dual gain amplifier from Fig. 3.6 as these subcircuit are exposed to the same pin voltages.

The power-down verification method from the previous sections has been applied to the dual gain amplifier to detect any errors in its power-down circuitry. Table 3.4 shows the voltage propagation results for the bias and the core circuit of the dual gain amplifier. The pin voltages were given as

$$V_{P_{DGA}} = (v_{in}, v_{out}, v_{pwd}, v_{\overline{pwd}}) = (float, float, vdd, gnd) \tag{3.3}$$

and the supply nets as $N_{vdd} = \{supply\}$ and $N_{gnd} = \{ground\}$. Voltage propagation had to be executed twice, as current paths were detected in the circuit based on the results of the first iteration. The voltage levels of the bias circuitry do not change in the second iteration as nets $n31$, $n41$, $n51$, $n71$ are tied to vdd by the resistors $R3$-$R6$, nets $n6$, $n8$ to gnd by the diode-connected transistors $N4$, $N6$ and nets $n32$, $n42$, $n52$, $n72$ to gnd by the power-down transistors $NP1$, $NP2$ inside the subcircuits $FOCA1$ and $FOCA2$.

In the core circuit, the voltages levels of nets out, $n1$ and $n2$ change to sc in the second voltage propagation iteration as short-circuit paths run over them.

Table 3.5 shows the voltage propagation result for the folded cascode amplifiers $FOCA1$ and $FOCA2$. The results for both subcircuits are identical as they exhibit the same pin voltages. Hence, the voltage propagation results computed for one of the two folded cascode amplifiers has been reused for the other one which reduces the computational effort of voltage propagation.

Short circuit path computation detects, based on the voltage propagation results of the first iteration, four current paths in the dual gain amplifier in power-down mode:

- $P1 = (FOCA1/PP1, FOCA1/NP3)$
- $P2 = (FOCA2/PP1, FOCA2/NP3)$
- $P3 = (FOCA1/PP1, R1, R2, FOCA2/NP3)$
- $P4 = (FOCA2/PP1, R2, R1, FOCA1/NP3)$

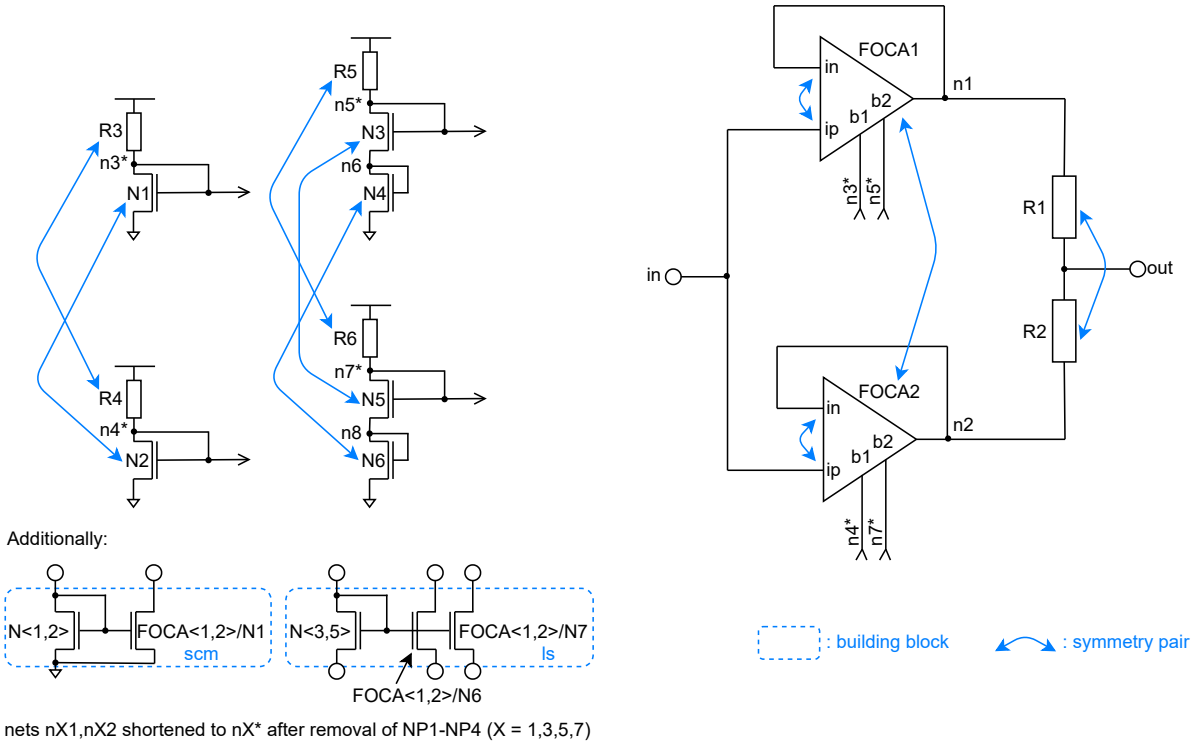
All of them follow a similar pattern: they origin at power-down transistor $PP1$ inside $FOCA1$ or $FOCA2$, switch to the top-level via their input pin in to net $n1$ or $n2$ of the dual gain amplifier, descend again into one of the folded cascode amplifiers via their output pins (eventually first running over $R1$ and $R2$ of the dual gain amplifier) and terminate at the ground node after passing $NP3$.

The internal paths of these short circuit paths are:

$$N_{sc} = \{n1, n4, out, FOCA{<}1, 2{>}/in, FOCA{<}1, 2{>}/out\}. \tag{3.4}$$

Voltage propagation has been repeated with this additional information; the corresponding results are shown in Tables 3.4 and 3.5, iteration two. No more new current paths have been detected based on these results and hence power-down verification continues with the next step, i.e., matching and symmetry assertion.

Fig. 3.8 shows the dual gain amplifier after power-up transformation and its matching and symmetry conditions identified by structure recognition (App. A) and symmetry computation (App. B). The power-down transistors $NP1$-$NP4$ in the bias part have been removed and the nets $nX1, nX2$, with $X = 3, 4, 5, 7$, have been shorted to nX^* to restore the diode-connection of transistors $N1$, $N2$, $N3$ and $N5$.



(a) bias circuitry (b) core circuit

Figure 3.8.: Symmetry and matching constraints of the dual gain amplifier from Fig. 3.6.

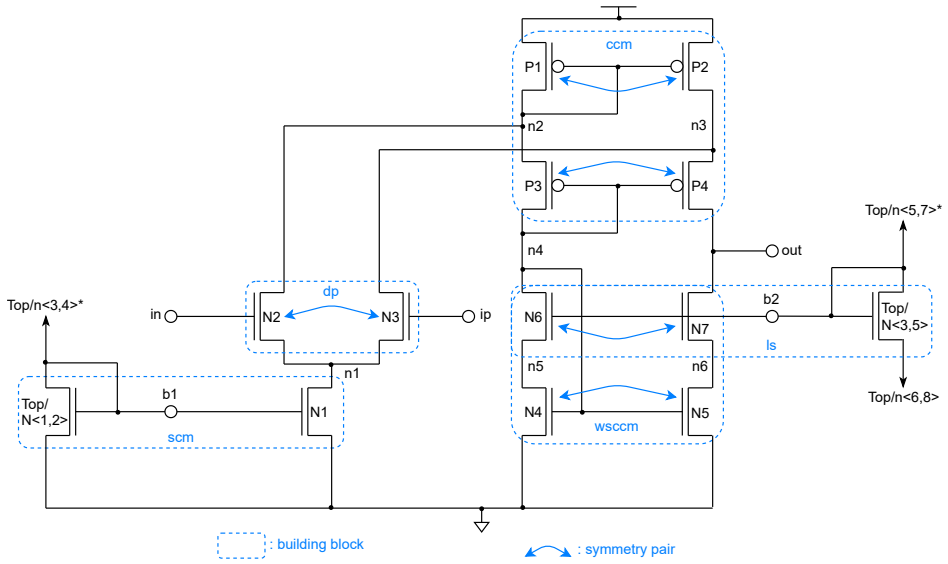


Figure 3.9.: Symmetry and matching constraints for $FOCA1$ and $FOCA2$ from Fig. 3.7.

The dual gain amplifier itself does not contain any analog basic buildings blocks. However, the transistors $N1$, $N2$, $N3$ and $N5$ of the bias circuit are connected to the transistors $N1$, $N6$, and $N7$ of $FOCA1$ and $FOCA2$ such that they form two simple current mirrors, i.e., $scm(N1, FOCA1/N1)$, $scm(N2, FOCA2/N1)$, and four level shifters, i.e., $ls(N3, FOCA1/N{<}6,7{>})$, $ls(N5, FOCA2/N{<}6,7{>})$, in total. Furthermore, the voltages at the pins of $FOCA1$, $FOCA2$ and the resistors $R1$, $R2$ must be matched in power-down mode.

Fig. 3.9 shows the folded cascode amplifier after power-up transformation and its identified building blocks and symmetry pairs. All power-down transistors can simply be removed from the circuit as none of them disables a diode-configuration or has been inserted in series into a current path of the circuit. The folded cascode amplifier consists of one differential pair $dp(N2, N3)$, one cascode current mirror $ccm(P1{-}P4)$ and one wide-swing cascode current mirror $wsccm(N4{-}N7)$. These basic building blocks also define the symmetry constraints of the folded cascode amplifier.

Furthermore, the information about the simple current mirrors and levels shifters at the bias pins $b1$ and $b2$ of $FOCA1$, $FOCA2$ due to external connectivity with the dual gain amplifier has been shared between these circuit hierarchy levels to keep their structural and symmetry information synchronized.

Table 3.6 summarizes the findings of the symmetry and matching assertion for the dual gain amplifier including its folded cascode amplifiers. Overall, 75% of the matching rules at the analog basic building blocks and about 40% of rules for the symmetry pairs of the circuit are violated.

Table 3.7 shows the matching assertion results for the dual gain amplifier in detail. Each row corresponds to one matching constraint which has to be evaluated. The first column denotes the building block which generated the matching constraint; the second column the pins of the building block whose voltage levels have to be compared; the third column the nets connected these pins and the fourth column their corresponding voltage levels computed by voltage propagation. The last column denotes the result of the assertion.

circuit	fulfilled matching constraints total	%	fulfilled symmetry constraints total	%
dual gain amplifier (DGA)	6/10	60%	16/22	72.3%
folded cascode amplifier (FOCA)	4/11	36.4%	8/15	53.3%
overall (DGA, $FOCA1$, $FOCA2$)	8/32	25.0%	32/52	61.5%

Table 3.6.: Summary of the matching and symmetry assertion results for the dual gain amplifier (DGA) and its folded cascode amplifier subcircuits $FOCA1$ and $FOCA2$ from Fig. 3.6 and 3.7.

building-blocks	Evaluated pins	Evaluated nets	Result	
$scm(N1, FOCA1/N1)$	(in, out)	$(n31, FOCA1/n1)$	$(vdd, float)$	⚠
$scm(N2, FOCA2/N1)$	(in, out)	$(n41, FOCA2/n1)$	$(vdd, float)$	⚠
$ls(N3, FOCA1/N6)$	(in, out)	$(n51, FOCA1/n4)$	(vdd, vdd)	✓
	$(s1, s2)$	$(n6, FOCA1/n5)$	(gnd, gnd)	✓
$ls(N3, FOCA1/N7)$	(in, out)	$(n51, FOCA1/out)$	(vdd, sc)	✗
	$(s1, s2)$	$(n6, FOCA1/n6)$	(gnd, gnd)	✓
$ls(N5, FOCA2/N6)$	(in, out)	$(n71, FOCA2/n4)$	(vdd, vdd)	✓
	$(s1, s2)$	$(n8, FOCA2/n5)$	(gnd, gnd)	✓
$ls(N5, FOCA2/N7)$	(in, out)	$(n71, FOCA1/out)$	(vdd, sc)	✗
	$(s1, s2)$	$(n8, FOCA1/n6)$	(gnd, gnd)	✓

evaluation result: fail (✗), pass (✓), warning (⚠)

Table 3.7.: Matching assertion results for the dual gain amplifier from Fig. 3.6.

subcircuits	Evaluated pins	Evaluated nets	Result	
$R1, R2$	$+$	$(n1, n2)$	(sc, sc)	✗
	$-$	(out, out)	(sc, sc)	✗
$R3, R4$	$+$	$(supply, supply)$	(vdd, vdd)	✓
	$-$	$(n31, n41)$	(vdd, vdd)	✓
$R5, R6$	$+$	$(supply, supply)$	(vdd, vdd)	✓
	$-$	$(n51, n71)$	(vdd, vdd)	✓
$N1, N2$	d	$(n31, n41)$	(vdd, vdd)	✓
	g	$(n32, n42)$	(gnd, gnd)	✓
	s	$(ground, ground)$	(gnd, gnd)	✓
$N3, N5$	d	$(n51, n71)$	(vdd, vdd)	✓
	g	$(n52, n72)$	(gnd, gnd)	✓
	s	$(n6, n8)$	(gnd, gnd)	✓
$N4, N6$	d	$(n6, n8)$	(gnd, gnd)	✓
	g	$(n6, n8)$	(gnd, gnd)	✓
	s	$(ground, ground)$	(gnd, gnd)	✓
$FOCA1, FOCA2$	in	$(n1, n2)$	(sc, sc)	✗
	ip	(in, in)	(vdd, vdd)	✓
	out	$(n1, n2)$	(sc, sc)	✗
	$b1$	$(n32, n42)$	(gnd, gnd)	✓
	$b2$	$(n52, n72)$	(gnd, gnd)	✓
$FOCA1$	(in, ip)	$(n1, in)$	(sc, vdd)	✗
$FOCA2$	(in, ip)	$(n2, in)$	(sc, vdd)	✗

evaluation result: fail (✗), pass (✓), warning (⚠)

Table 3.8.: Symmetry assertion results for the dual gain amplifier from Fig. 3.6.

Two warnings are issued for the dual gain amplifier as the output pins of the simple current mirrors $scm(N1, FOCA1/N1)$ and $scm(N2, FOCA2/N1)$ are floating. Furthermore, two errors are reported as the output pins of the level shifters $ls(N5, FOCA2/N6)$ and $ls(N5, FOCA2/N7)$ are on a short circuit node. Please note that the devices of these current mirrors and level shifters lie on different hierarchy levels. Matching and symmetry violations at building blocks split over several hierarchy levels are hard to spot manually as such errors are concealed in-between the different levels. The new verification methodology is able to detect such errors automatically.

Table 3.8 shows the detailed evaluation results of the symmetry rules for the dual gain amplifier. The first column lists the checked symmetry pairs, the second column their evaluated subcircuit pins, the third column the nets connected to the respective pins, the fourth column their corresponding voltage levels and the last column the evaluation result of the symmetry assertion. Six symmetry constraints of the dual gain amplifier are violated in power-down mode. Each violation is caused by one of the detected short circuit paths. The rule violations

building-blocks	Evaluated		Result	
	pins	nets		
$scm(Top/N{<}1,2{>},N1)$	(in, out)	$(Top/n{<}31,41{>},n1)$	$(vdd, float)$	⚠
$ls(Top/N{<}1,2{>},N6)$	(in, out)	$(Top/n{<}51,71{>},n4)$	(vdd, vdd)	✓
	$(s1, s2)$	$(Top/n{<}6,8{>},n5)$	(gnd, gnd)	✓
$ls(Top/N{<}1,2{>},N7)$	(in, out)	$(Top/n{<}51,71{>},out)$	(vdd, sc)	✗
	$(s1, s2)$	$(Top/n{<}6,8{>},n6)$	(gnd, gnd)	✓
$dp(N2,N3)$	$(in1, in2)$	(in, ip)	(sc, vdd)	✗
	$(out1, out2)$	$(n2, n3)$	$(vdd, float)$	⚠
$wsccm(N4-N7)$	(in, out)	$(n4, out)$	(vdd, sc)	✗
	$(inner1, inner2)$	$(n5, n6)$	(gnd, gnd)	✓
$ccm(P1-P4)$	(in, out)	$(n4, out)$	(vdd, sc)	✗
	$(inner1, inner2)$	$(n2, n3)$	$(vdd, float)$	⚠

evaluation result: fail (✗), pass (✓), warning (⚠)

Table 3.9.: Matching assertion results for $FOCA1$ and $FOCA2$ from Fig. 3.7.

at the differential inputs (in, ip) of $FOCA1$ and $FOCA2$ at the bottom of the table indicate that there are further matching and symmetry problems in the underlying folded cascode amplifiers.

These problems are revealed by Tables 3.9 and 3.10. Table 3.9 shows that only one out of six matching constraints generated by its basic analog building blocks is fulfilled. Three warnings due to floating nodes indicating possible asymmetrical stress and four errors due to short-circuit nodes in the circuit have been issued.

Table 3.10 reports four potentially violated, four violated and eight fulfilled symmetry constraints for each of the folded cascode amplifier in power-down mode.

3.5.2. High Input Impedance Differential Amplifier

Fig. 3.10a shows a high input impedance differential amplifier (HIIDA) [38]. Its subcircuits $MILLER1$, $MILLER2$ and $MILLER3$ are implemented as Miller operational transconductance amplifiers (OTAs) whose topology is shown in Fig. 3.10b. The power-down circuitry of these circuits is highlighted in blue.

Power-down verification has been executed for this hierarchical circuit with input voltages $V_{P,HIIDA} = (in, ip, out, pwd) = (float, float, float, vdd)$. The supply nets are given as $N_{vdd} = \{supply\}$ and $N_{gnd} = \{ground\}$.

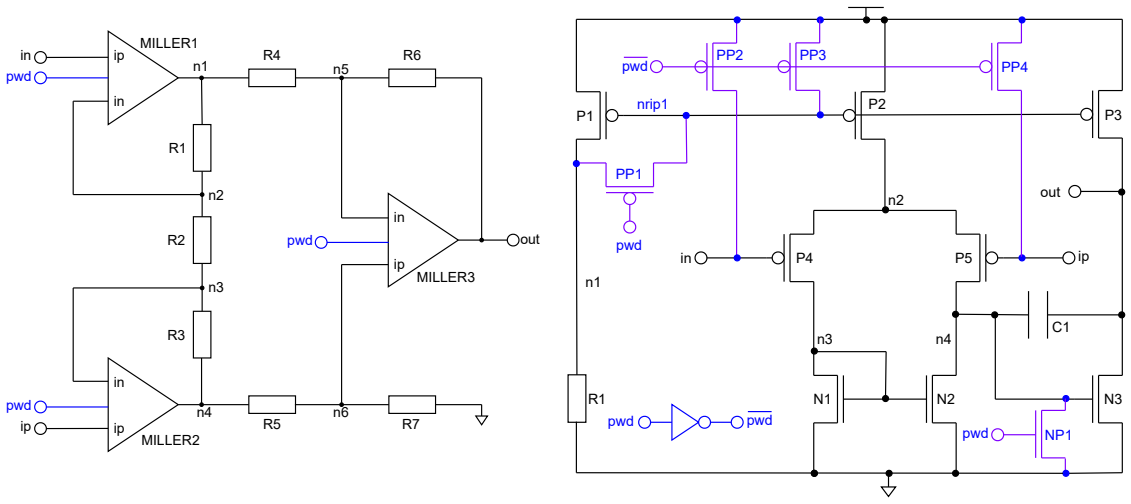
Voltage propagation and short circuit path computation identify four current paths in the circuit:

- $P_1 = (MILLER1/PP2, R2, R3, R5, R7)$
- $P_2 = (MILLER2/PP2, R3, R5, R7)$
- $P_3 = (MILLER3/PP2, R4, R1, R2, R3, R5, R7)$

subcircuits	Evaluated pins	Evaluated nets	Result	
$N2, N3$	d	$(n2, n3)$	$(vdd, float)$	⚠
	g	(in, ip)	(sc, vdd)	✗
	s	$(n1, n1)$	$(float, float)$	⚠
$N4, N5$	d	$(n5, n6)$	(gnd, gnd)	✓
	g	$(n4, n4)$	(vdd, vdd)	✓
	s	$(ground, ground)$	(gnd, gnd)	✓
$N6, N7$	d	$(n4, out)$	(vdd, sc)	✗
	g	$(vb2, vb2)$	(gnd, gnd)	✓
	s	$(n5, n6)$	(gnd, gnd)	✓
$P1, P2$	d	$(n2, n3)$	$(vdd, float)$	⚠
	g	$(n2, n2)$	(vdd, vdd)	✓
	s	$(supply, supply)$	(vdd, vdd)	✓
$P3, P4$	d	$(n4, out)$	(vdd, sc)	✗
	g	$(n4, n4)$	(vdd, vdd)	✓
	s	$(n2, n3)$	$(vdd, float)$	⚠

evaluation result: fail (✗), pass (✓), warning (⚠)

Table 3.10.: Symmetry assertion results for $FOCA1$ and $FOCA2$ from Fig. 3.7.



(a) HIIDA (b) Miller OTA

Figure 3.10.: Schematic of a high input impedance differential amplifier (HIIDA) [38]. The Miller operational transconductance amplifier (OTA) implements its subcircuits $MILLER1, MILLER2$ and $MILLER3$ (b). Their power-down circuitry is highlighted in blue.

top-level (HIIDA)		$MILLER<1,2>$		$MILLER3$	
Net	Voltage	Net	Voltage	Net	Voltage
in	vdd	in	sc	in	sc
ip	vdd	ip	vdd	ip	sc
$n1$	sc	$n1$	gnd	$n1$	gnd
$n2$	sc	$n2$	$float$	$n2$	$float$
$n3$	sc	$n3$	gnd	$n3$	gnd
$n4$	sc	$n4$	gnd	$n4$	gnd
$n5$	sc	$np1$	vdd	$np1$	vdd
$n6$	sc	out	sc	out	sc
out	sc				

For all circuits: $ground = gnd, supply = vdd$ and $pwd = vdd$.
For $MILLER1 - MILLER3$: $\overline{pwd} = gnd$.

Table 3.11.: Final voltage propagation result for the high input impedance differential amplifier (HIIDA) and the Miller operational transconductance amplifiers (OTAs) from Fig. 3.10.

- $P_4 = (MILLER3/PP4, R7)$

Hence, voltage propagation has to be repeated with the set of short circuit nets:

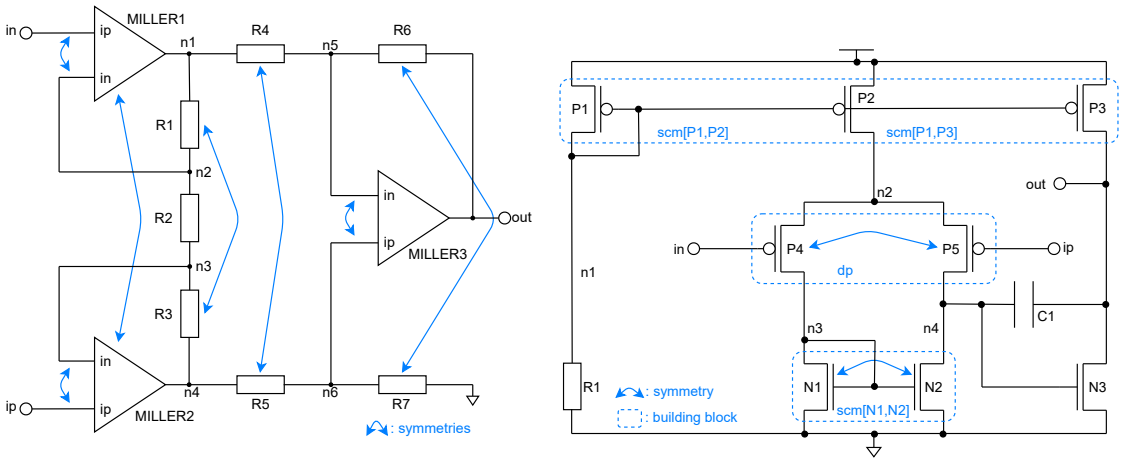
$$N_{sc} = \{MILLER1/in, MILLER2/in, MILLER3/in, MILLER3/ip, n1{-}n6\}. \quad (3.5)$$

The final voltage propagation result is displayed in Table 3.11. During voltage propagation, intermediate results for $MILLER1$ and $MILLER2$ have been reused as both Miller operational transconductance amplifiers are exposed to the same input voltage levels in power-down mode. The results reveal that there are three floating nets in the circuit: net $n2$ in each of the three Miller operational transconductance amplifiers.

The voltage propagation results are then used to check the matching and symmetry conditions of the high input impedance differential amplifier in power-down mode. Its power-down circuitry first has to be removed in order to be able to identify its analog basic building blocks and symmetry pairs. The result of the power-up transformation for the high input impedance differential amplifier is shown in Fig. 3.11a, the one for the Miller operational transconductance amplifier in Fig. 3.11b.

Based on Fig. 3.11b, structure recognition identifies the differential pair $dp(P4, P5)$, the simple current mirror $scm(N1, N2)$ and the simple current mirror bank $scm(P1, P2, P3)$ in each of the three Miller operational transconductance amplifiers. The high input impedance differential amplifier does not contain any analog basic building blocks on the top-level.

Symmetry computation for the Miller operational transconductance amplifier detects the symmetry pairs $(P4, P5)$ and $(N1, N2)$. The high input impedance differential amplifier contains the symmetry pairs $(MILLER1, MILLER2)$, $(R1, R3)$, $(R4, R5)$ and $(R6, R7)$.



(a) HIIDA (b) Miller OTA

Figure 3.11.: High input impedance differential amplifier (HIIDA) [38] and its Miller operational transconductance amplifier (OTA) subcircuit blocks after power-up transformation.

Building Blocks	Evaluated Pins	Evaluated Nets	Voltages $MILLER<1,2>$		Voltages $MILLER3$	
$scm(P1, P2)$	d	$(n1, n2)$	$(gnd, float)$	✗	$(gnd, float)$	✗
$scm(P1, P3)$	d	$(n1, out)$	(gnd, sc)	✗	(gnd, sc)	✗
$scm(N1, N2)$	d	$(n3, n4)$	(gnd, gnd)	✓	(gnd, gnd)	✓
$dp(P4, P5)$	d	$(n3, n4)$	(gnd, gnd)	✓	(gnd, gnd)	✓
	g	(in, ip)	(sc, vdd)	✗	(sc, sc)	✗

evaluation result: fail (✗), pass (✓), warning (⚠)

Table 3.12.: Matching assertion results for the Miller operational transconductance amplifiers $MILLER1 - MILLER3$ from Fig. 3.10b.

Additionally, the voltages at the differential input (in, ip) of each of the three Miller operational transconductance amplifiers must be matched at the top-level. These results are shown in Fig. 3.11 and correspond to ones presented in [23].

Matching and symmetry assertion evaluates the identified structures against asymmetric stress in power-down mode. Table 3.12 shows the corresponding evaluation results for the basic building blocks of the three Miller operational transconductance amplifiers. The drain pins of the PMOS current mirrors, as well as the gate pins of the differential pair are exposed to different voltage levels in each of the three Miller operational transconductance amplifiers in power-down mode, hence failing the assertion. Only the drain pins of the NMOS simple current mirror and the differential pair are successfully evaluated. Overall, $\frac{9}{15} = 60\%$ of the building blocks matching conditions are violated.

Tables 3.13 and 3.14 show the symmetry assertion results for the high input impedance differential amplifier and its subcircuit blocks $MILLER1 - MILLER3$. All symmetry checks for the resistors of the high input impedance differential amplifier fail as the detected

subcircuits	Evaluated Pins	Evaluated Nets	Voltages	
$R1, R3$	$plus$	$(n1, n3)$	(sc, sc)	✗
	$minus$	$(n2, n4)$	(sc, sc)	✗
$R4, R5$	$plus$	$(n1, n4)$	(sc, sc)	✗
	$minus$	$(n5, n6)$	(sc, sc)	✗
$R6, R7$	$plus$	$(n5, n6)$	(sc, sc)	✗
	$minus$	$(out, ground)$	(sc, gnd)	✗
$MILLER1, MILLER2$	in	$(n2, n3)$	(sc, sc)	✗
	ip	(in, ip)	(vdd, vdd)	✓
	out	$(n1, n4)$	(sc, sc)	✗
$MILLER3$	(in, ip)	$(n5, n6)$	(sc, sc)	✗

evaluation result: fail (✗), pass (✓), warning (⚠)

Table 3.13.: Symmetry assertion result for the high input impedance differential amplifier from Fig. 3.10a.

subcircuits	Evaluated Pins	Evaluated Nets	Voltages $MILLER<1, 2>$		Voltages $MILLER3$	
$P4, P5$	d	$(n4, n3)$	(gnd, gnd)	✓	(gnd, gnd)	✓
	g	(in, ip)	(sc, vdd)	✗	(sc, sc)	✗
	s	$n2$	$float$	⚠	$float$	⚠
$N1, N2$	d	$(n3, n4)$	(gnd, gnd)	✓	(gnd, gnd)	✓
	g	$n3$	gnd	✓	gnd	✓
	s	$ground$	gnd	✓	gnd	✓

evaluation result: fail (✗), pass (✓), warning (⚠)

Table 3.14.: Symmetry assertion results for the Miller operational transconductance amplifiers from Fig. 3.10b.

short circuit paths are running over them. Furthermore, at least one pin of each Miller operational transconductance amplifier is exposed to voltage level *sc*. Overall, only one out of ten symmetry conditions is fulfilled in the top-level schematic.

For the Miller operational transconductance amplifiers, three warnings and three errors are reported: the gates of $P4$ and $P5$ are exposed to different voltage levels and their source pins are connected to a floating net.

App. C shows how this verification report can be used to revise and debug the power-down circuitry of the high input impedance differential amplifier.

4. Hierarchical Power-Down Synthesis

The previous chapter presented a method to verify the power-down mode of hierarchical analog circuits. In the following, a method to synthesize faultless power-down circuitry for such circuits is presented.

4.1. Overview

Fig. 4.1 gives an overview of the proposed hierarchical power-down synthesis method. It uses the same steps as the method presented in [12]. It, however, makes extensive reuse of intermediate results stored in several different intermediate results libraries.

The main data flow of the method is indicated by thick vertical arrows in the figure. It takes a top-level circuit C_{top}, the supply nets N_{gnd}, N_{vdd}, the power-down signals P_{pwd} and the pin voltages $V_{P_{top}}$ as input. The output is denoted as $C'_{top}(P'_{top}, N'_{top}, C'_{sub,top}, t'_{top})$ which corresponds to the top-level circuit augmented by the synthesized power-down circuitry.

The core of the method is the gate shut-off procedure (Sec. 4.3). It uses the gate shut-off pattern from Fig. 1.6a and constraint programming to insert power-down transistors in parallel into C_{top} to switch off its bias currents whilst achieving a trade-off between the two design goals "maximize matching" (C) and "minimize area" (D). Structure recognition and symmetry analysis are performed to generate the required matching and symmetry constraints for the constraint program (App. A and App. B).

The power-down synthesis method can be used to augment partially designed, yet incomplete, power-down circuitry, with its missing parts. The existing power-down circuitry, especially power-down transistors which have been inserted by the diode or serial rip-up pattern, can interfere with the structure recognition and symmetry analysis methods. E.g., a diode rip-up transistor can prevent the identification of current mirrors by structure recognition which in turn prevents symmetry analysis to accurately model the signal flow through these analog building blocks. Hence, already existing power-down circuitry is removed by the power-up transformation method from Sec. 3.4.1 before the structure recognition and symmetry computation are executed (Sec. 3.4).

The gate shut-off constraint program from Sec. 4.3 can determine a valid power-down mode implementation variant if all current paths of the circuit can be switched off by the gate shut-off pattern. Hence, C_{top} is scanned beforehand for problematic currents paths, i.e., definite short circuit paths, by voltage propagation and short-circuit path computation from Secs. 3.2 and 3.3 which are then disrupted by applying the diode or serial rip-up pattern from Fig. 1.6. The gate shut-off constraint program then guarantees the existence of at least one valid power-down mode implementation variant for the ripped-up circuit.

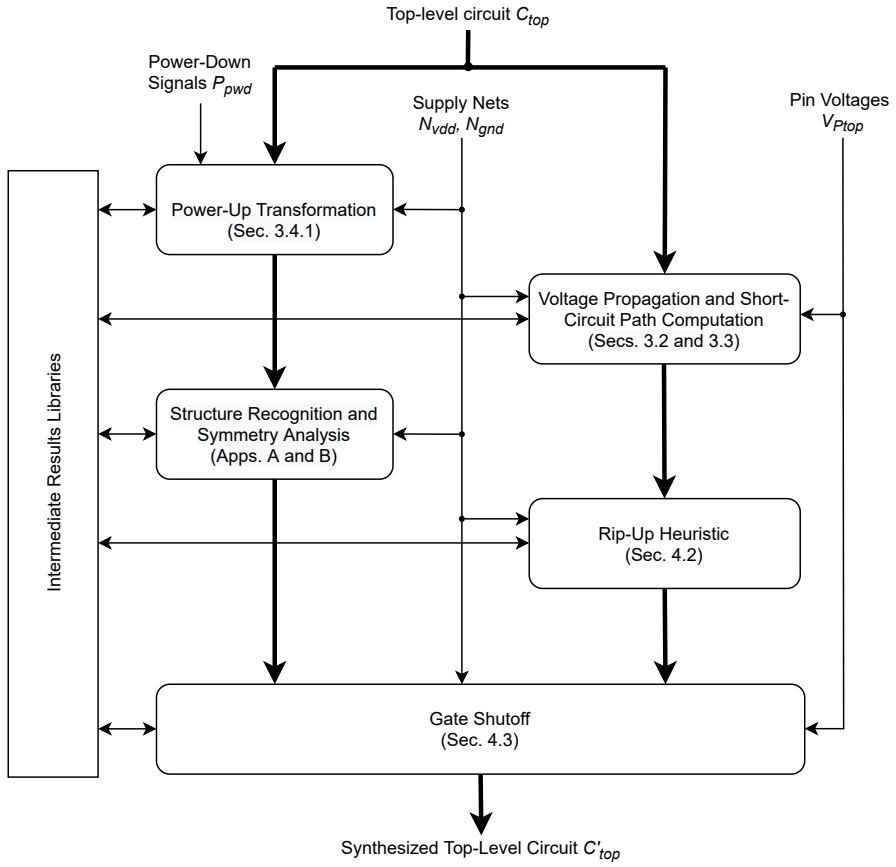


Figure 4.1.: Overview of the power-down synthesis method.

Algorithm 5 Rip-up heuristic for hierarchical analog circuits based on [12].

procedure RIPUPHEURISTIC($C_{top}(P_{top}, N_{top}, C_{sub,top}, t_{top}), \mathcal{P}_{sc}$)
  $L_{rip} = \emptyset$
  for all $P_j \in \mathcal{P}_{sc}$ **do**
    $d_{rip}, c_{rip} = computeRipupPoint(P_j)$
    if not $hasResult(L_{rip}, d_{rip}, c_{rip})$ **then**
      if $isDiodeConnected(d_{rip})$ **then**
        $c'_{rip} = applyDiodeRipup(d_{rip}, c_{rip})$
        $storeResult(L_{rip}, d_{rip}, c_{rip}, c'_{rip})$
      else
        $c'_{rip} = applySerialRipup(d_{rip}, c_{rip})$
        $storeResult(L_{rip}, d_{rip}, c_{rip}, c'_{rip})$
      end if
    end if
    $c'_{rip} = findResult(L_{rip}, d_{sub}, c_{rip})$
    $replace(C_{top}, c_{rip}, c'_{rip})$
  end for
end procedure

4.2. Rip-Up Heuristic

Algorithm 5 outlines the rip-up heuristic for hierarchical circuits based on [12]. It takes the top-level circuit C_{top} and the set of definite short circuit paths $\mathcal{P}_{sc}$ as input. The problematic current paths are thereby detected by voltage propagation and short circuit path computation from Sec. 3.2 and Sec. 3.3, respectively. The algorithm uses the library L_{rip} to store ripped-up subcircuit blocks in line 2. The method iterates over all current paths $P_j \in \mathcal{P}_{sc}$ (line 3) and determines a suitable rip-up position, i.e., a device d_{rip} on P_j inside a subcircuit block c_{rip}, by invoking function $computeRipupPoint(P_j)$ in line 4. In line 5, function $hasResult(L_{rip}, d_{rip}, c_{rip})$ checks whether intermediate results for the determined rip-up point already exist in library L_{rip}. If that is the case, the algorithm continues by reusing that result in lines 14 and 15. Otherwise, the current path P_j is ripped-up as follows: If d_{rip} is diode-connected, a diode-rip-up according to Fig. 1.6b is performed (lines 7 and 8). If a path P_j contains several diode-connected devices, the device connected to the most transistor gates is chosen for rip-up as more current paths can be switched off with a single gate shut-off transistor afterwards [12].

If d_{rip} is not diode-connected, the current path has to be switched off by a serial rip-up as shown in Fig. 1.6c (lines 10 and 11). For serial rip-up, the device which is connected to the supply net is chosen as rip-up position and a power-down transistor between the pin connected to the supply net and the supply net itself is inserted. However, other rip-up positions, e.g., at the ground node or even at an internal net of that path, could be chosen as well.

In both cases, the ripped-up circuit c'_{rip} is stored in the library L_{rip} (line 8 or 11).

Finally, in lines 14 and 15, the rip-up heuristic fetches the ripped-up circuit from L_{rip} and uses it to replace c_{rip} at the corresponding position in the top-level circuit.

Fig. 4.2 shows a differential stage series ($SDFS$). Its subcircuits $DFS1$ and $DFS2$ each implement one differential stage. Voltage propagation and short circuit path computation detected the definite short circuit path

$$P1 = (DFS2/P1, DFS2/N2, DFS2/N1) \tag{4.1}$$

in the circuit which must be ripped up in order to implement the power-down mode of the circuit. The current path $P1$ is highlighted in red in the circuit schematic.

Algorithm 5 computes transistor $P1$ of $DFS2$ as rip-up point. The transistor is diode-connected and part of the current mirror $scm(P1, P2)$. Hence, disabling the diode configuration of $P1$ will also switch off the current flow through $P2$ in power-down mode. The ripped-up version of the differential input stage is stored in $\mathcal{C}$. It is denoted as $DFS2'$ in order to be able distinguish it better from the regular, non ripped-up version. The ripped-up schematic of the differential stage series is shown in Fig. 4.3.

4.3. Gate Shut-Off

Algorithm 6 outlines the gate shut-off procedure for hierarchical analog circuits. It takes a circuit netlist C_i, the pin voltages V_{P_i} and the power-down synthesis results library L_{synth} as

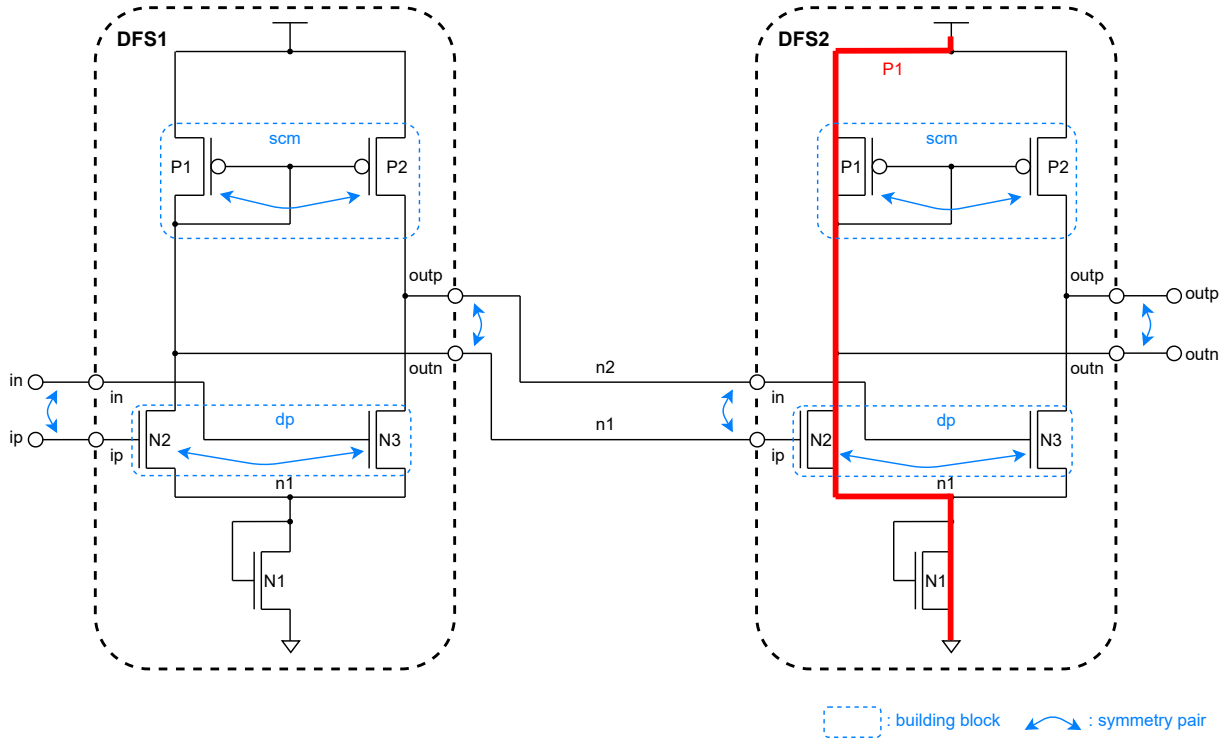


Figure 4.2.: Schematic of a differential stage series ($SDFS$). Subcircuits $DFS1$ and $DFS2$ each implement one differential stage. The definite short circuit path $P1$, detected by voltage propagation and short-circuit path computation, is highlighted in red.

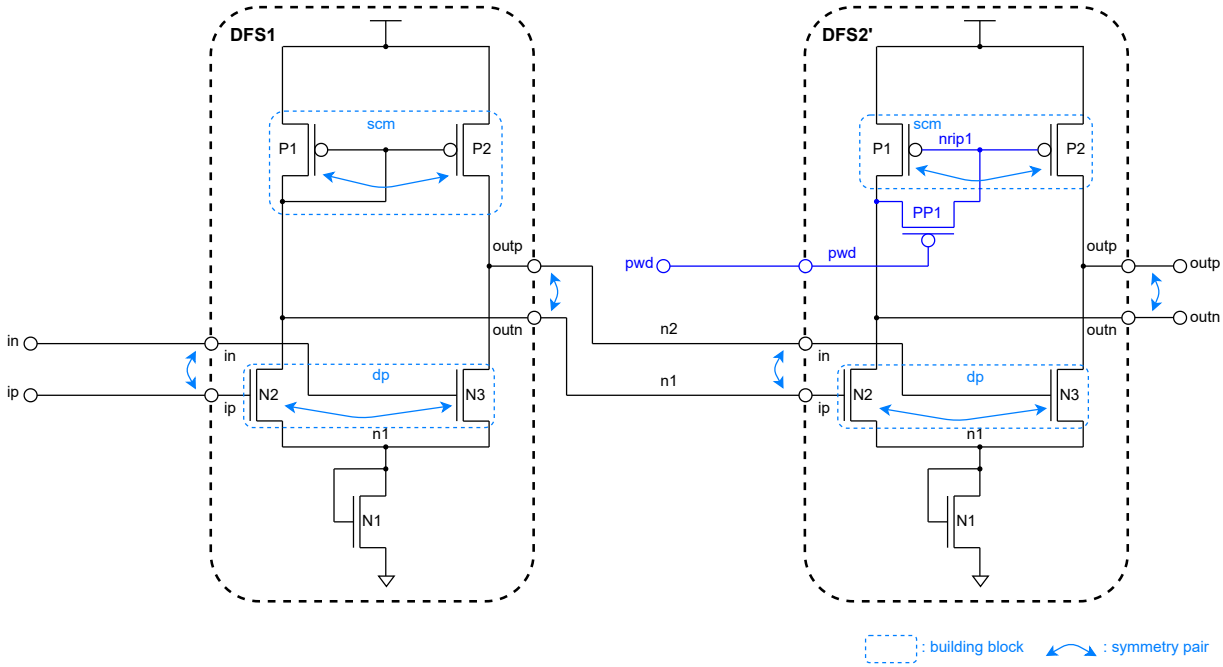


Figure 4.3.: Ripped-up schematic of the differential series from Fig. 4.2.

Algorithm 6 Gate shut-off for hierarchical circuits based on [10]

1: **procedure** GATESHUTOFF($C_i(P_i, N_i, C_{sub,i}, t_i), V_{P_i}, L_{synth}$)
    // Additional inputs L_{struct}, L_{sym}, N_{supply} not shown.
    // Output L_{synth} passed by reference and recursively filled in.

2: $V_{N_i} = voltagePropagation(C_i, V_{P_i})$ // See Algorithm 1.

    // Check if gate shut-off COP from eq. (4.15) has at least one solution.
3: **if** $currentThroughDevice(C_i, V_{N_i})$ **then**
4: **return** // Current detected, return.
5: **end if**

    // Only formulate gate shut-off COP if there is a floating node in the circuit.
6: **if** $noFloatingNode(C_i, V_{N_i})$ **then**
7: $\mathbf{V}^*_{N_i} = \{V_{N_i}\}$
8: **else**
9: $\mathbf{V}^*_{N_i} = solveGateShutoffCOP(C_i, V_{N_i})$ // See eq. (4.15).
10: **end if**
11: $storeResults(C_i, V_{P_i}, \mathbf{V}^*_{N_i}, L_{synth})$

    // Descend hierarchy recursively for each implementation variant.
12: **for all** $\mathbf{v}^*_j \in \mathbf{V}^*_N$ **do**
13: **for all** $c_{sub} \in C_{sub,i}$ **do**
14: **if** $isSubCircuitBlock(c_{sub})$ **then**
15: $V_{P_{sub}} = pinVoltages(c_{sub}, \mathbf{v}^*_j)$
16: **if not** $hasResult(c_{sub}, V_{P_{sub}}, L_{synth})$ **then**
17: $gateShutoff(c_{sub}, V_{P_{sub}}, L_{synth})$ // Recursive call.
18: **end if**
19: **end if**
20: **end for**
21: **end for**
22: **end procedure**

inputs. It furthermore takes the structure recognition results L_{struct}, the symmetry computation results L_{sym} and the supply nets N_{supply} as additional inputs. The results library L_{synth} is passed by reference and is filled in with all computed power-down mode implementation variants for the given circuit with pin voltages V_{P_i} during gate shut-off. The additional inputs are not shown in the pseudo-code for simplicity but are always accessible during the computations.

The library L_{synth} is organized as follows: An entry $(C_i, V_{P_i}, \mathbf{V}^*_{N_i})$ contains all valid power-down mode implementation variants

$$\mathbf{V}^*_{N_i} = \{\mathbf{v}^*_1, \mathbf{v}^*_2, ..., \mathbf{v}_{|\mathbf{V}^*_{N_i}|}\}, \tag{4.2}$$

$$\text{with } \mathbf{v}^*_j = (v^*_{n_1}, v^*_{n_2}, ..., v^*_{n_{|N_i|}}), \tag{4.3}$$

$$\text{and } v^*_{n_k} \in \{gnd, pullGnd, vdd, pullVdd\} \tag{4.4}$$

for the given circuit C_i with input voltages V_{P_i}. An implementation variant $\mathbf{v}^*_j \in \mathbf{V}^*_{N_i}$ assigns each net $n_k \in N_i$ to one of the following voltage levels:

- *pullGnd*, *pullVdd*: net $n_k \in N_i$ has to be pulled to *gnd* or to *vdd* by a power-down transistor.
- *gnd*, *vdd*: the supply or ground voltage inherently propagates to net $n_k \in N_i$ in power-down mode. Hence, no power-down transistor has to be inserted at such a net.

Algorithm 6 traverses the circuit hierarchy recursively in a top-down manner. As a first step, voltage propagation is performed to determine all internal nodes of the circuit to which the supply and ground voltage can inherently propagate (line 2). The algorithm then checks if there is at least one device which carries a current in power-down mode (lines 3 - 5) according to the voltage propagation result V_{N_i} by equation

$$\begin{aligned} &currentThroughDevice(C_i, V_{N_i}) \Leftrightarrow \\ &\quad \underset{c_{sub} \in C_{sub,i}}{\exists} (t_{sub} \in T_{device}) \wedge on(c_{sub}, V_{N_i}) \wedge voltageDrop(v_+(c_{sub}), v_-(c_{sub})). \end{aligned} \tag{4.5}$$

It expresses that there is a current flow in power-down mode if there is at least one device in the circuit which is conducting (on) and which additionally has a voltage potential difference between its conducting channel in power-down mode. A device is conducting in power-down mode if one of the following four conditions holds:

$$on(c_{sub}, V_{N_i}) \Leftrightarrow \begin{cases} v_g = vdd & \text{, if } t_{sub} = nmos \\ v_g = gnd & \text{, if } t_{sub} = pmos \\ v_+ = vdd \vee v_- = gnd & \text{, if } t_{sub} \in \{diode, nmos_{dio}, pmos_{dio}\} \\ true & \text{, if } t_{sub} \in \{resistor, inductor\} \\ false & \text{, else, e.g., } t_{sub} = cap. \end{cases} \tag{4.6}$$

An *nmos* or *pmos* transistor is on if its gate voltage is tied to *vdd* or *gnd*, respectively. Diodes or diode-connected transistors are conducting if their anode (v_+) is on *vdd* or if their cathode (v_-) is on *gnd*. Inductors and resistors are always conducting, capacitors are always off.

There is a voltage potential difference at the conducting device channel if:

$$voltageDrop(v_+, v_-) \Leftrightarrow \begin{cases} (v_+ = vdd \wedge v_- = gnd) \vee & \text{, if } t_{sub} \in \{resistor, inductor, \\ (v_+ = gnd \wedge v_- = vdd) & \quad nmos, pmos\} \\ v_+ = vdd \wedge v_- = gnd & \text{, if } t_{sub} \in \{diode, nmos_{dio}, pmos_{dio}\}. \end{cases} \tag{4.7}$$

Regular MOSFET transistors, resistors and inductors thereby conduct a current in both directions, diodes and diode-connected devices in only one direction, i.e., from their anode to their cathode pin.

The circuit contains a definite short-circuit path if eq. (4.5) is fulfilled and the later formulated gate shut-off constraint optimization problem (COP) will not have a solution. Hence, Algorithm 6 returns (line 4).

In the next step, the algorithm checks if there are any floating nodes in the circuit (lines 6-10):

$$noFloatingNode(C_i, V_{N_i}) \Leftrightarrow \neg \underset{n_{i,j} \in N_i}{\exists} v_{n_{i,j}} = float \tag{4.8}$$

The power-down mode of the circuit is already functional if there is no floating node in the circuit, i.e., all nodes are already tied to either *vdd* or *gnd*. There is no need to insert additional power-down transistors into the circuit by the gate shut-off COP. In fact, the voltage levels of the circuit cannot be changed by the gate shut-off pattern anymore without creating a new current flow in the circuit. The voltage propagation result V_{N_i} would in this case also correspond to the only solution of the gate shut-off COP from eq. (4.15) and hence formulating and solving the COP is skipped by assigning the voltage propagation result to $\mathbf{V_{N_i}}^*$ (line 7).

The gate shut-off COP however must be formulated and solved for the current hierarchy level C_i with the given pin voltages V_{P_i} if there are any floating nodes left in the circuit (line 9). The thereby obtained solutions $\mathbf{v}_j^* \in \mathbf{V}_{N_i}^*$ represent different valid power-down mode implementation variants for the given circuit whilst maintaining a trade-off between design goals *matching* (C) and *area* (D). The gate shut-off COP is formulated based on the devices of the current hierarchy level, its subcircuit blocks are ignored for the moment as they will be treated individually later on by recursively descending the circuit hierarchy. The obtained solutions are stored in library L_{synth}.

The algorithm iterates over all solutions $\mathbf{v}_j^* \in \mathbf{V}_{N_i}^*$ and all subcircuit blocks $c_{sub} \in C_{sub,i}$ of the circuit in the next step (lines 12-21), determines their pin voltages $V_{P_{sub}}$ based on $\mathbf{v}_j^*$ and descends into the hierarchy by calling itself recursively (lines 15 and 17). However, the algorithm first checks whether a previously computed intermediate result can be reused by determining whether another subcircuit block of the same type and pin voltages has been encountered during a previous iteration (line 16). Above steps are repeated until the device level of the circuit hierarchy has been reached.

4.3.1. Constraint Optimization Problem

The design goals of the power-down mode (A) - (D) are formalized as a constraint program in the following. The presented constraint program unifies the gate shut-off and net dependency

constraint programs from [10] which allows trade-offs between the design goals "maximize matching" and "minimize area". Thereby, Gecode [20] has been used to formulate and solve the unified gate shut-off constraint program which is the same constraint programming framework used by [12].

Constraint programming [19; 39] models a mathematical problem with a vector of variables $\mathbf{z}$, a domain D_i for each variable $z_i \in \mathbf{z}$ and a set of constraints C:

$$\mathbf{z} = (z_1\ z_2\ \ldots\ z_{n_z}),\quad z_i \in D_i,\ i = 1, 2, ..., n \tag{4.9}$$
$$C = E \cup I = \{c_1, c_2, ..., c_m\} \tag{4.10}$$

The domains D_i contain either boolean, integer or float values and are either finite or infinite. A constraint $c_j \in E$ denotes an equality constraint, i.e., $c_j(\mathbf{z}) = 0$, a constraint $c_k \in I$ an inequality constraint, i.e. $c_j(\mathbf{z}) \bigcirc 0$, with $\bigcirc \in \{>,\ <,\ \geq,\ \leq,\ !=,\ ...\}$.

The constraints $c \in C$ form a constraint satisfaction problem (CSP),

$$CSP(\mathbf{z}) \Leftrightarrow \underset{j \in E}{\forall} c_j(\mathbf{z}) = 0 \wedge \underset{k \in I}{\forall} c_k(\mathbf{z}) \bigcirc 0, \tag{4.11}$$

which must be fulfilled. A solution of eq. (4.11) is a value assignment $\mathbf{z}^*$ of $\mathbf{z}$ which fulfils the CSP.

A constraint optimization problem (COP) extends a CSP by an additional vector of target functions $\mathbf{t}(\mathbf{z})$ to

$$\max\ \ \mathbf{t}(\mathbf{z})\ \ s.t.\ \ CSP(\mathbf{z}) \Leftrightarrow 1. \tag{4.12}$$

Solving eq. (4.12) yields the best solutions of $\mathbf{t}(\mathbf{z})$ which satisfy the $CSP(\mathbf{z})$.

The gate shut-off COP is formulated based on the voltage propagation results V_{N_i} computed by Algorithm 1 for the given circuit C_i with pin voltage V_{P_i}. The following sets are thereby required as input:

$$\begin{aligned}
N_{float} &= \{n_{i,j} \mid v_{n_{i,j}} = float \wedge v_{n_{i,j}} \in V_{N_i}\} \\
N'_{vdd} &= \{n_{i,j} \mid v_{n_{i,j}} = vdd \wedge v_{n_{i,j}} \in V_{N_i}\} \\
N'_{gnd} &= \{n_{i,j} \mid v_{n_{i,j}} = gnd \wedge v_{n_{i,j}} \in V_{N_i}\} \\
C_{float} &= \{c_{sub} \in C_{sub,i} \mid t_{sub} \in T_{device} \wedge connectedToFloatingNet(c_{sub}, V_{N_i})\}
\end{aligned} \tag{4.13}$$

The set N_{float} contains all floating nets, the sets $N_{vdd'}$ and $N_{gnd'}$ all nets of the circuit to which the ground or supply voltage can inherently propagate, including the supply and ground nets and respective input voltages. The set C_{float} contains all devices of the circuit which are connected to a floating net in power-down mode by at least one of its corresponding pins:

$$connectedToFloatingNet(c_{sub}, V_{N_i}) \Leftrightarrow \underset{p_{sub,j} \in P_{sub}}{\exists} v_{p_{sub,j}} = float \wedge v_{p_{sub,j}} \in V_{N_i} \tag{4.14}$$

The supply and ground voltages inherently propagate to some extent to the internal nets of the circuit through conducting devices in power-down mode. The remaining floating nodes of the circuit must be pulled to either *vdd* or *gnd* by additional gate shut-off power-down

transistors such that all potential current flows are switched off in the circuit while taking matching and area considerations into account. Hence, the vector of net voltages and their corresponding domains from eqs. (4.3) and (4.4) have been chosen to model the power-down synthesis problem as following constraint optimization problem:

$$\begin{aligned} \max \quad & w_1 \cdot \frac{matching(\mathbf{v})}{n_{match}} - w_2 \cdot \frac{area(\mathbf{v})}{n_{float}} \text{ s.t.} \\ & noCurrentFlow(\mathbf{v}) \Leftrightarrow 1 \wedge \\ & noFloatingNet(\mathbf{v}) \Leftrightarrow 1 \wedge \\ & boundaryConditions(\mathbf{v}) \Leftrightarrow 1. \end{aligned} \tag{4.15}$$

The target function aims to find a trade-off between the design goals "maximize matching" (C) and "minimize area" (D). The design goals (C) and (D) are normalized by the number of generated matching constraints n_{match} and by the number of floating notes n_{float} in the circuit detected by voltage propagation. The weights w_1, w_2 of the target function can be chosen to either prioritize one of the two goals or to compute all Pareto-optimal solutions. The CSP part ensures that there are no current flows or floating nets in the circuit in power-down mode and sets the boundary conditions of the constraint program by initializing the supply nets and pins of the given circuit to their corresponding level.

The function

$$matching(\mathbf{v}) \Leftrightarrow buildingBlockMatching(\mathbf{v}) \; + \; symmetryMatching(\mathbf{v}) \tag{4.16}$$

consists of two parts: The first term

$$buildingBlockMatching(\mathbf{v}) \Leftrightarrow \sum_{b_j \in C_{B_{i,analog}}} \sum_{r_k \in R_{type(b_j)}} sameLevel(v_{n_1(b_j,p_1(r_k))}, v_{n_2(b_j,p_2(r_k))}) \tag{4.17}$$

generates block-type specific matching constraints for each identified analog basic building block b_j in the circuit [12]. The building block circuit $C_{B_{i,analog}}$ denotes the structure recognition result from Appendix A. It contains all analog building blocks of circuit C_i, e.g., current mirrors and differential pairs. A building block type specific rule set

$$R_{type(b_j)} = \{r_1, r_2, ..., r_{|R_{type(b_j)}|}\} \text{ with } r_k = (p_{k,1}, p_{k,2}) \text{ and } p_{k,1}, p_{k,2} \in P_{b_j} \tag{4.18}$$

contains tuples of pins which should be matched in power-down mode. E.g., the set

$$R_{dp} = \{r_1 = (in_1, in_2), r_2 = (out_1, out_2)\} \tag{4.19}$$

contains two rules which specify that the input and output pins of a differential pair *dp* should be exposed to the same voltage levels in power-down mode (see Fig. 3.5a).

The constraint

$$\begin{aligned} sameLevel(v_{n_1}, v_{n_2}) \; \Leftrightarrow \; & [gndLevel(v_{n_1}) \wedge gndLevel(v_{n_2})] \; \vee \\ & [vddLevel(v_{n_1}) \wedge vddLevel(v_{n_2})] \, . \end{aligned} \tag{4.20}$$

denotes that the voltages at the nets n_1 and n_2 should be on the same level. The symbols *gnd*, *pullGnd* and *vdd*, *pullVdd* quantitatively represent the same voltage levels in power-down mode,i.e.:

$$gndLevel(v_{n_i}) \Leftrightarrow (v_{n_i} = gnd) \vee (v_{n_i} = pullGnd) \tag{4.21}$$
$$vddLevel(v_{n_i}) \Leftrightarrow (v_{n_i} = vdd) \vee (v_{n_i} = pullVdd). \tag{4.22}$$

The functions $n_1(b_j, p_1(r_k))$ and $n_2(b_j, p_2(r_k))$ used in eq. (4.17) determine the nets of circuit C_i which are connected to the pins p_1 and p_2 of building block b_j which are specified by rule r_k.

The output pins of differential pair $dp(N2, N3)$ of the differential stage $DFS1$ from Fig. 4.3, e.g., should be matched. This corresponds to matching the voltages at nets $DFS1/outn$ and $DFS1/outp$:

$$match_{dp(DFS1/N2,DFS1/N3),out}(\mathbf{v}) \;\Leftrightarrow\; sameLevel(v_{DFS1/n2}, v_{DFS1/n3}). \tag{4.23}$$

The second term

$$symmetryMatching(\mathbf{v}) \Leftrightarrow \sum_{(c_i,c_j)\in C_{sym}} \sum_{(p_{i,k},p_{j,k})\in P_{sym}} sameLevel(v_{p_{i,k}}, v_{p_{j,k}}) \tag{4.24}$$

generates matching constraints for each symmetrical pin pair of each identified symmetry pair in the circuit including pin symmetries due to subcircuit block internal symmetrical signal paths. The input pins *in*, *ip* of $DFS2'$ from Fig. 4.3, e.g., should be matched on the top-level schematic. This corresponds to matching the voltages at nets $n1$ and $n2$ of the differential stage series:

$$sameLevel(v_{p_{DFS2',in}}, v_{p_{DFS2',ip}}) \;\Leftrightarrow\; sameLevel(v_{n1}, v_{n2}) \tag{4.25}$$

The value of eq. (4.16) corresponds to the number of fulfilled matching and symmetry constraints in power-down mode.

In addition, the optional constraint

$$symmetricTransistors(v_1, v_2) \Leftrightarrow\; (v_1 = v_2) \tag{4.26}$$

can be formulated for each matched net or pin pair in order to enforce that power-down transistors are inserted symmetrically into the circuit.

The area of the power-down circuitry is mainly determined by the number of inserted power-down transistors. Hence, the constraint

$$area(\mathbf{v}) \;\Leftrightarrow\; \sum_{n_i\in N_{float}} addTransistor(v_{n_i}), \text{ with} \tag{4.27}$$
$$addTransistor(v_{n_i}) \;\Leftrightarrow\; (v_{n_i} = pullGnd) \;\vee\; (v_{n_i} = pullVdd) \tag{4.28}$$

sums up the number of power-down transistors which have been inserted at the floating nodes of the circuit to fulfil the CSP part, i.e., eqs. (4.29), (4.31), (4.34), of the overall gate shut-off COP, eq. (4.15).

The constraint satisfaction problem

$$noCurrentFlow(\mathbf{v}) \;\Leftrightarrow \underset{c_{sub}\in C_{float}}{\forall} on(c_{sub},\mathbf{v}) \rightarrow sameLevel(v_+(c_{sub}), v_-(c_{sub})) \tag{4.29}$$

ensures that there is no current flow in the circuit in power-down mode based on the set C_{float}. This set contains all devices of the circuit through which a current flow would still be possible in power-down mode as these devices are connected to at least one floating node according to the previously voltage propagation result. All other devices have already been checked by eq. (4.5) and do not have to be considered by the COP anymore. The CSP from eq. (4.29) formulates a constraint for each floating device of the circuit which prohibits a current flow through it by either forcing it into the non-conducting state or by ensuring that there is no voltage drop between its conducting channel in power-down mode.

An NMOS transistor, e.g., is off in power-down mode iff its gate pin is exposed to the ground voltage, i.e., $v_g = gnd$. It is on iff $v_g = vdd$. In the latter case, its drain and source pin must have the same voltage level in power-down mode to ensure that there is no current flow in the circuit.

A device c_{sub} is conducting if one of the following conditions holds:

$$on(c_{sub},\mathbf{v}) = \begin{cases} vddLevel(v_g) & \text{, if } t_{sub} = nmos \\ gndLevel(v_g) & \text{, if } t_{sub} = pmos \\ vddLevel(v_+) \vee gndLevel(v_-) & \text{, if } t_{sub} \in \{diode, nmos_{dio}, pmos_{dio}\} \\ true & \text{, if } t_{sub} \in \{resistor, inductor\} \\ false & \text{, else.} \end{cases} \tag{4.30}$$

NMOS- and PMOS transistors are conducting when their gate pin is on vdd or gnd, respectively. Diodes and diode-connected transistors are only conducting in one direction, i.e., they are on if their anode voltage v_+ is on vdd or their cathode pin v_- is on gnd. Resistors and inductors are treated as always on, other device types are considered to be off. Fig. 3.2 shows the pin names and the propagation behavior of different device types.

The constraint satisfaction problem

$$\begin{aligned} &noFloatingNet(\mathbf{v}) \Leftrightarrow \\ &\quad \underset{n_i\in N_{float}}{\forall} addTransistor(v_{n_i}) \vee && (C1) \\ &\quad \underset{P_{n_j\rightarrow n_i}, n_j\in N_{vdd'}}{\exists} on(P_{n_j\rightarrow n_i},\mathbf{v}) \vee && (C2) \\ &\quad \underset{P_{n_i\rightarrow n_j}, n_j\in N_{gnd'}}{\exists} on(P_{n_i\rightarrow n_j},\mathbf{v}) \vee && (C3) \\ &\quad \underset{P_{n_j\rightarrow n_i}, n_j\in N_{float}}{\exists} \left[on(P_{n_j\rightarrow n_i},\mathbf{v}) \wedge (v_{n_j} = pullVdd)\right] \vee && (C4) \\ &\quad \underset{P_{n_i\rightarrow n_j}, n_j\in N_{float}}{\exists} \left[on(P_{n_i\rightarrow n_j},\mathbf{v}) \wedge (v_{n_j} = pullGnd)\right] && (C5) \end{aligned} \tag{4.31}$$

formulates a constraint for each floating net n_i of the circuit that ensures that either a power-down transistor is inserted to pull it to *vdd* or *gnd* (C1) or that the supply or ground voltage inherently propagates to n_i via a conducting device path $P_{n_j \to n_i}$ or $P_{n_i \to n_j}$ (C2-C4), respectively.

A path

$$\begin{aligned} &P_{n_j \to n_i} = (c_1, c_2, ..., c_k) \text{ with} \\ &n_+(c_1) = n_j,\ n_-(c_k) = n_i \text{ and} \\ &\underset{l=2,...,k-1}{\forall} n_-(c_l) = n_+(c_{l+1}) \end{aligned} \tag{4.32}$$

is a series of devices connecting net n_j with net n_i of the circuit. A path $P_{n_j \to n_i}$ allows *vdd* to propagate from n_j to n_i if it is conducting in power-down mode, i.e., iff:

$$on(P_{n_j \to n_i}, \mathbf{v}) \Leftrightarrow \underset{c_k \in P_{n_j \to n_i}}{\forall} on(c_k, \mathbf{v}) \Leftrightarrow 1. \tag{4.33}$$

Analogously, a conducting path $P_{n_i \to n_j}$ allows the ground voltage to propagate from n_j to n_i.

Condition (C2) checks if there is a conducting path from a net $n_j \in N_{vdd'}$ to floating node n_i such that the supply voltage can propagate to n_i, i.e., in forward direction. Similarly, condition (C3) checks if there is a conducting path from floating node n_i to a net $n_j \in N_{gnd'}$ such that the ground voltage can propagate to n_i, i.e., in backwards direction. Condition (C4) checks if there exists a conducting path between floating node n_j to n_i with n_j additionally being pulled to *vdd* by a power-down transistor such that *vdd* can propagate to n_i. Finally, condition (C5) checks if there exists a conducting path between floating node n_i to n_j with n_j additionally being pulled to *gnd* by a power-down transistor such that *gnd* can propagate to n_i. If there is at least one path in the circuit fulfilling one of (C2-C5) then there is no need to add a power-down transistor to n_i as *vdd* or *gnd* inherently propagates to it.

The boundary conditions model that the ground and supply voltage *vdd* and *gnd* can propagate from the nets specified by N'_{gnd} and N'_{supply} to the remaining floating nets of the circuit by initially adding a power-down transistor to these nets:

$$boundaryConditions(\mathbf{v}) \Leftrightarrow \underset{n_i \in N'_{vdd}}{\forall} (v_{n_i} = pullVdd) \wedge \underset{n_j \in N'_{gnd}}{\forall} (v_{n_j} = pullGnd) \tag{4.34}$$

After solving the COP, those transistors are removed again from the circuit.

Table 4.1 shows the voltage propagation results for the ripped-up differential stage series ($SDFS$) from Fig. 4.3 from which

$$N_{float} = \{in, ip, outp, n2\},\ N'_{vdd} = \{pwd, n1\} \text{ and } N'_{gnd} = \{outn\} \tag{4.35}$$

can be determined. There is no device which carries a current in power-down mode. However, there are still floating nodes in the circuit and hence the gate shut-off COP must be formulated and solved for it.

C_i	top-level ($SDFS$)						$DFS1$				
n_i	in	ip	$outn$	$outp$	$n1$	$n2$	in	ip	$outn$	$outp$	$n1$
v_{n_i}	$float$	$float$	gnd	$float$	vdd	$float$	$float$	$float$	vdd	$float$	gnd

C_i	$DFS2'$						global nets		
n_i	in	ip	$outn$	$outp$	$n1$	n_{rip1}	$supply$	$ground$	pwd
v_{n_i}	$float$	vdd	gnd	$float$	gnd	$float$	vdd	gnd	vdd

Table 4.1.: Voltage propagation result for the ripped-up differential stage series from Fig. 4.3.

	constraint			comment
COP	$buildingBlockMatching(\mathbf{v})$	$\Leftrightarrow$	0	no basic building block
	$symmetryMatching(\mathbf{v})$	$\Leftrightarrow$	$sameLevel(v_{in}, v_{ip}) + sameLevel(v_{n1}, v_{n2}) +$ $sameLevel(v_{n1}, v_{n2}) + sameLevel(v_{outn}, v_{outp})$	self-symmetry $DFS1$ self-symmetry $DFS2'$
	$area(\mathbf{v})$	$\Leftrightarrow$	$addTransistor(v_{in}) + addTransistor(v_{ip}) +$ $addTransistor(v_{outp}) + addTransistor(v_{n2})$	N_{float} from eq. (4.35)
CSP	$noCurrentFlow(\mathbf{v})$	$\Leftrightarrow$	1	no devices
	$noFloatingNet(\mathbf{v})$	$\Leftrightarrow$	$addTransistor(v_{in}) \wedge addTransistor(v_{ip}) \wedge$ $addTransistor(v_{outp}) \wedge addTransistor(v_{n2})$	N_{float} from eq. (4.35)
	$boundaryConditions(\mathbf{v})$	$\Leftrightarrow$	$(v_{outn} = pullGnd) \wedge (v_{n1} = pullVdd) \wedge$ $(v_{pwd} = pullVdd)$	N'_{vdd}, N'_{vdd} from eq. (4.35)

The constraints $matching(\mathbf{v}) \Leftrightarrow buildingBlockMatching(\mathbf{v}) + symmetryMatching(\mathbf{v})$ and $area(\mathbf{v})$ are equally weighted, i.e., $w_1 = w_2 = 0.5$.

Table 4.2.: Constraint optimization problem from eq. (4.15) for the differential stage series from Fig. 4.3.

solution	v^*_{in}	v^*_{ip}	v^*_{n1}	v^*_{n2}	v^*_{outn}	v^*_{outp}	v^*_{pwd}	valid
$\mathbf{v}^*_1$	$pullVdd$	$pullVdd$	vdd	$pullVdd$	gnd	$pullGnd$	vdd	✓
$\mathbf{v}^*_2$	$pullGnd$	$pullGnd$	vdd	$pullVdd$	gnd	$pullGnd$	vdd	✗

Table 4.3.: Solutions of the constraint optimization problem from Table 4.2. The power-down transistors inserted by the $boundaryConditions(\mathbf{v})$ constraint at nets $outn$, $n1$ and pwd have already been removed from this table and reverted to gnd, vdd, vdd, respectively.

$c_{sub,i}$	$DFS1$						
n_i	in	ip	$outn$	$outp$	$n1$	$supply$	$ground$
v_{n_i}	vdd	vdd	vdd	vdd	gnd	vdd	gnd

Pin voltages: $V_{P,DFS1} = (v_{in}, v_{ip}, v_{outn}, v_{outp}) = (vdd, vdd, vdd, vdd)$

Devices carrying a current by eq. (4.5): $N2, N3$

Table 4.4.: Voltage propagation results of differential stage $DFS1$ for solution $\mathbf{v}^*_2$ of the differential stage series.

Table 4.2 shows the constraint optimization problem (COP) and the constraint satisfaction problem (CSP) part for the top-level schematic of the ripped-up differential stage series. The circuit does not contain any basic analog building blocks on the top-level, hence $buildingBlockMatching(\mathbf{v})$ always evaluates to zero. Its subcircuits $DFS1$ and $DFS2'$ are self-symmetric, i.e, their differential input and output pins (in, ip and $outn$, $outp$, respectively) should be exposed to the same voltage levels in power-down mode. This is captured by constraint $symmetryMatching(\mathbf{v})$ which can have a score between zero (no matching constraints fulfilled) and four (all matching constraints fulfilled). The circuit initially has four floating nodes according to the voltage propagation results from eq. (4.35). Hence, constraint $area(\mathbf{v})$ denotes that up to four power-down transistors have to be inserted into the top-level schematic to implement its power-down mode. The constraint $noCurrentFlow(\mathbf{v})$ is always fulfilled on the top-level as it does not contain any basic devices. The four nets in, ip, $n2$ and $outp$ are floating and do not have any connections over conducting devices to one of the nets from sets N'_{vdd} or N'_{gnd} in power-down mode. Hence, one power-down transistor must be connected to each of these nets to fulfil the constraint $noFloatingNet(\mathbf{v})$. Finally, the $boundaryConditions(\mathbf{v})$ are set according to N'_{vdd} and N'_{gnd}.

Solving this COP yields the two solutions which are shown in Table 4.3. Both solutions fulfil all symmetry constraints and are implemented by four power-down transistors.

The second solution however is not valid as can be seen on the voltage propagation results for $DFS1$ from Table 4.4: the transistors $N2$ and $N3$ carry a current as their gates (in, ip) are on vdd and there is a voltage drop between their drain and source pins, i.e., between (v_{outn}, v_{n1}) and (v_{outp}, v_{n1}), respectively. Hence solution $\mathbf{v}^*_2$ is discarded by Algorithm 6.

The first solution on the other hand switches $DFS1/N2$, $DFS1/N3$ off by pulling the inputs in, ip on the top-level to gnd. The corresponding voltage propagation results are shown in Table 4.5. The circuit does not contain any device fulfilling eq. (4.5) and no floating nodes have been detected. The power-down mode of $DFS1$ is already functional and formulating and solving the gate shut-off COP is skipped as the voltage propagation

$c_{sub,i}$	$DFS1$						
n_i	in	ip	$outn$	$outp$	$n1$	$supply$	$ground$
v_{n_i}	gnd	gnd	vdd	vdd	gnd	vdd	gnd

Pin voltages: $V_{P,DFS1} = (v_{in}, v_{ip}, v_{outn}, v_{outp}) = (gnd, gnd, vdd, vdd)$

Devices carrying a current by eq. (4.5): none

Table 4.5.: Voltage propagation results for differential stage $DFS1$ based on solution $\mathbf{v}_1^*$ of the differential stage series.

$c_{sub,i}$	$DFS2$								
n_i	in	ip	$outn$	$outp$	$n1$	$nrip1$	$supply$	$ground$	pwd
v_{n_i}	vdd	vdd	gnd	gnd	gnd	$float$	vdd	gnd	vdd

Pin voltages: $V_{P,DFS2} = (v_{in}, v_{ip}, v_{outn}, v_{outp}) = (vdd, vdd, gnd, gnd)$

Devices carrying a current by eq. (4.5): none

Table 4.6.: Voltage propagation results for differential stage $DFS2$ based on solution $\mathbf{v}_1^*$ of the differential stage series.

result would correspond to the gate shut-off COP result.

Table 4.6 shows the voltage propagation results for $DFS2'$ based on solution $\mathbf{v}_1^*$ of the differential stage series. Solution $\mathbf{v}_1^*$ is still valid as there are no current carrying devices in $DFS2'$ for the provided input voltages. The sets

$$N_{float} = \{nrip1\},\ N'_{vdd} = \{in, ip, pwd, supply\},\ N'_{gnd} = \{outn, outp, n1\} \tag{4.36}$$

and

$$C_{float} = \{P1, P2, PP1\} \tag{4.37}$$

are determined based on the voltage propagation results. As there is still one floating node in the circuit, the gate shut-off COP must be formulated and solved for $DFS2'$.

Table 4.7 shows the gate shut-off COP for $DFS2'$ and $\mathbf{v}_1^*$. The building block and symmetry matching constraints are determined by differential pair $dp(N2, N3)$ and simple current mirror $scm(P1, P2)$. Their values are solely determined by the voltage propagation results as none of the generated matching constraints depend on floating node $nrip1$. The value of the area constraint however is determined by $nrip1$. It can be minimized if it could be avoided to add a power-down transistor to it. The three transistors $DFS2'/P1$, $DFS2'/P2$ and $DFS2'/PP1$ are all connected to $nrip1$. Hence, a constraint is formulated for each of these transistors to forbid current flow through them. A power-down transistor must be added to $nrip1$ or transistor $DFS2'/PP1$ must be switched on in power-down mode to allow gnd to propagate to $nrip1$ to ensure that there is no floating node in the circuit anymore. This transistor however is always off in power-down mode as the boundary conditions set the power-down signal pwd to vdd. Hence, a power-down transistor to pull $nrip1$ to vdd must be inserted into the circuit. The remaining boundary conditions are set according to eq. (4.36).

	constraint			comment
COP	$buildingBlockMatching(\mathbf{v})$	$\Leftrightarrow$	$sameLevel(v_{outn}, v_{outp}) + sameLevel(v_{in}, v_{ip}) + sameLevel(v_{outn}, v_{outp})$	$dp(N2, N3)$ $scm(P1, P2)$
	$symmetryMatching(\mathbf{v})$	$\Leftrightarrow$	$sameLevel(v_{outn}, v_{outp}) + sameLevel(v_{in}, v_{ip}) + sameLevel(v_{outn}, v_{outp})$	$dp(N2, N3)$ $scm(P1, P2)$
	$area(\mathbf{v})$	$\Leftrightarrow$	$addTransistor(v_{nrip1})$	N_{float} from eq. (4.36)
CSP	$noCurrentFlow(\mathbf{v})$	$\Leftrightarrow$	$[gndLevel(v_{nrip1}) \rightarrow sameLevel(v_{outn}, v_{supply})] \wedge$ $[gndLevel(v_{nrip1}) \rightarrow sameLevel(v_{outp}, v_{supply})] \wedge$ $[gndLevel(v_{pwd}) \rightarrow sameLevel(v_{outn}, v_{nrip1})] \Leftrightarrow 1$	transistor $DFS2'/P1$ transistor $DFS2'/P2$ transistor $DFS2'/PP1$
	$noFloatingNet(\mathbf{v})$	$\Leftrightarrow$	$addTransistor(v_{nrip1}) \vee gndLevel(v_{pwd}) \Leftrightarrow 1$	N_{float} from eq. (4.36)
	$boundaryConditions(\mathbf{v})$	$\Leftrightarrow$	$(v_{outn} = pullGnd) \wedge (v_{outp} = pullGnd) \wedge$ $(v_{n1} = pullGnd) \wedge (v_{ground} = pullGnd) \wedge$ $(v_{in} = pullVdd) \wedge (v_{ip} = pullVdd) \wedge$ $(v_{pwd} = pullVdd) \wedge (v_{supply} = pullVdd) \Leftrightarrow 1$	N'_{gnd} from eq. (4.36) N'_{vdd} from eq. (4.36)

The constraints $matching(\mathbf{v}) \Leftrightarrow buildingBlockMatching(\mathbf{v}) + symmetryMatching(\mathbf{v})$ and $area(\mathbf{v})$ are equally weighted, i.e., $w_1 = w_2 = 0.5$.

Table 4.7.: Constraint optimization problem from eq. (4.15) for differential stage $DFS2'$ and solution $\mathbf{v}_1^*$ of the differential stage series from Fig. 4.3.

solution	v_{in}^*	v_{ip}^*	v_{outn}^*	v_{outp}^*	v_{n1}^*	v_{nrip2}^*	valid	matching	area
$\mathbf{v}_1^*$	vdd	vdd	gnd	gnd	gnd	$pullVdd$	✓	6	1

Nets *supply*, *pwd* on *vdd*; net *ground* on *gnd*

Target function value: $0.5 \cdot \frac{6}{6} - 0.5 \cdot \frac{1}{1} = 0.0$

Table 4.8.: Solutions of the constraint optimization problem from Table 4.7. The power-down transistors inserted by the boundary conditions have already been reverted to *vdd* or *gnd*, respectively.

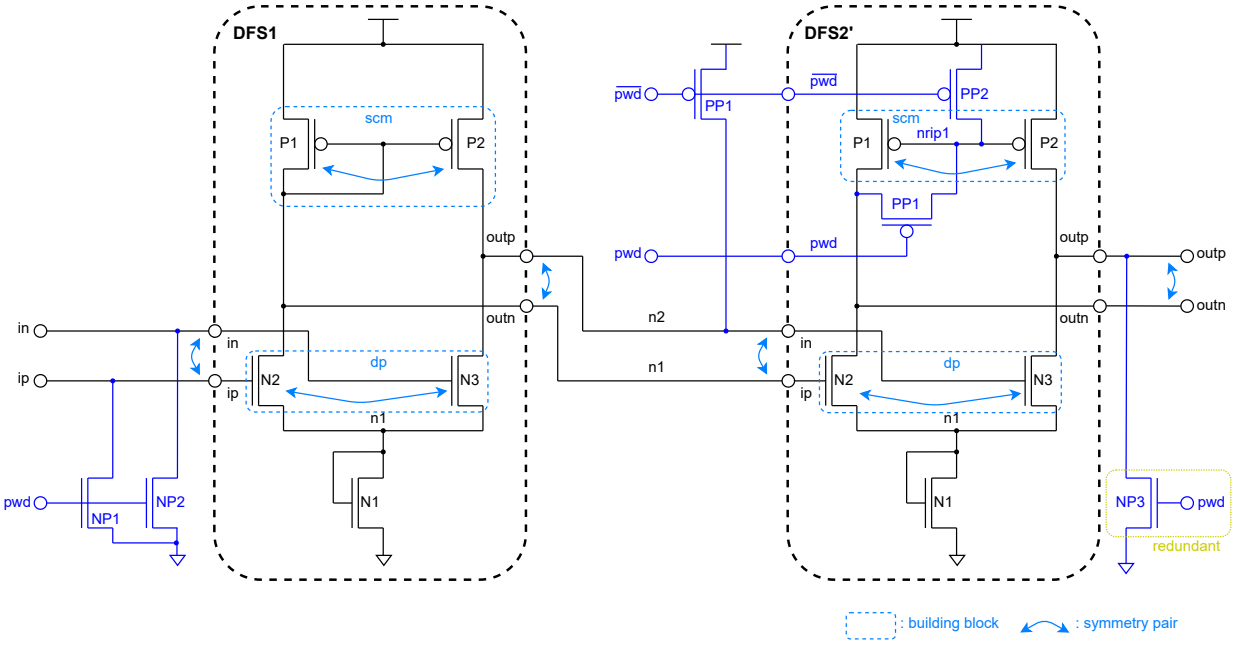


Figure 4.4.: Final synthesized power-down circuitry for the differential stage series from Fig. 4.3.

Solving the COP from Table 4.7 yields one solution which is shown in Table 4.8. One power-down transistor to pull $nrip1$ to vdd is inserted to switch off $DFS2'/P1$ and $DFS2'/P2$ in power-down mode. The solution fulfils all six matching constraints. Hence, the overall value of the target function is $0.5 \cdot \frac{6}{6} - 0.5 \cdot \frac{1}{1} = 0.0$.

The power-down mode of the differential stage series has been successfully synthesized as its subcircuits $DSF1$ and $DFS2'$ each have one valid power-down mode implementation variant for the input voltages provided by top-level solution $\mathbf{v}_1^*$. The final synthesized power-down circuitry of the differential stage series is shown in Fig. 4.4. It requires five power-down transistors in total to implement it. The power-down transistor $NP3$ is however redundant as *gnd* would inherently propagate to the output pin *outp* of the differential stage series via transistors $DFS2'/N1$ and $DFS2'/N3$. This redundant power-down transistor has been inserted into the circuit by the gate shut-off COP because the underlying circuit topologies of subcircuits $DFS1$ and $DFS2'$ have not been considered while formulating the gate shut-off COP for the differential stage series. More specifically, it has not been considered that *gnd* can propagate to *outp* if $DFS2'/N3$ is switched on in power-down mode.

Such redundant power-down transistors can be automatically detected by a depth-first search. It must be searched for conducting cycles starting from the ground or the supply node. If such a cycle exists and a power-down transistor lies on it, that power-down transistor is redundant and can safely be removed from the circuit.

4.3.2. Selection of the Optimal Solution

The individually computed power-down implementation variants for each subcircuit of a hierarchical design have to be combined to an overall solution. This can lead to a high number of implementation variants as all possible solutions have to be enumerated. E.g., consider a hierarchical circuit with two levels and six different subcircuit blocks. The gate

$\mathbf{v}_i^*$	v_{in}	v_{ip}	v_{n1}	v_{n2}	v_{outn}	v_{outp}	sub-circuits
$\mathbf{v}_1^*$	$pullGnd$	$pullGnd$	vdd	$pullGnd$	gnd	$pullGnd$	$DFS1$ $\mathbf{v}_1^*$, $DFS2'$ $\mathbf{v}_1^*$
$\mathbf{v}_2^*$	$pullGnd$	$pullGnd$	vdd	$pullGnd$	gnd	$pullVdd$	$DFS1$ $\mathbf{v}_1^*$, $DFS2'$ $\mathbf{v}_2^*$
$\mathbf{v}_3^*$	$pullGnd$	$pullGnd$	vdd	$pullVdd$	gnd	gnd	$DFS1$ $\mathbf{v}_2^*$, $DFS2'$ $\mathbf{v}_3^*$
$\mathbf{v}_4^*$	$pullVdd$	$pullGnd$	vdd	gnd	gnd	$pullGnd$	$DFS1$ $\mathbf{v}_3^*$, $DFS2'$ $\mathbf{v}_1^*$
$\mathbf{v}_5^*$	$pullVdd$	$pullGnd$	vdd	gnd	gnd	$pullVdd$	$DFS1$ $\mathbf{v}_3^*$, $DFS2'$ $\mathbf{v}_2^*$

Table 4.9.: Valid solutions of the gate shut-off COP for the differential stage series from Fig. 4.3 if the symmetry and matching information is not propagated between the bottom and the top-level of the circuit hierarchy. The solutions for differential stages $DFS1$ and $DFS2'$ are shown in Tables 4.10 and 4.11, respectively.

shut-off COP computes four solutions for the top-level and five solutions for each of its subcircuit blocks. In the worst-case, $4 \cdot 5^6 = 62500$ solutions for the hierarchical design need to be examined. For designs with even more subcircuit blocks and hierarchy levels, the number of solutions quickly becomes unmanageably large.

Typically, only a small subset of hierarchical solutions is optimal for the given design. In the following, an efficient strategy to find the optimal solutions for hierarchical circuits is presented. Therefore, two scores based on the optimization targets "maximize matching" and "minimize area" are assigned to each solution $\mathbf{v}_j^*$ of each circuit C_i. The flat score

$$flatScore(C_i, \mathbf{v}_j^*) = w_1 \cdot \frac{matching(\mathbf{v}_j^*)}{n_{match}} - w_2 \cdot \frac{area(\mathbf{v}_j^*)}{n_{area}} \tag{4.38}$$

corresponds to the value of the target function of the gate shut-off COP from eq. (4.15) for solution $\mathbf{v}_j^*$ and considers only the current hierarchy level C_i. The target function of the gate shut-off COP is maximized by constraint programming. Hence, solutions with a high flat score should be preferred over solutions with a low flat score as more matching constraints are fulfilled in power-down mode while requiring less chip area. This is especially true for transistor level circuits as they do not contain any subcircuit blocks, i.e., their best solutions have the highest flat score.

The hierarchical score of a solution includes the subcircuit blocks of C_i as follows:

$$hierScore(C_i, \mathbf{v}_j^*) = flatScore(C_i, \mathbf{v}_j^*) + \sum_{c_{sub} \in C_{sub,i}} \max_{v^*_{c_{sub,i,j}} \in \mathcal{V}^*_{c_{sub,i}}} hierScore(c_{sub}, \mathbf{v}^*_{c_{sub,i,j}}). \tag{4.39}$$

It is recursively computed: only the best solutions for each subcircuit block of C_i, i.e., only sub-solutions with the highest hierarchical score for that subcircuit, are selected. These scores are then added to the flat score of the current hierarchy level. In this way, only optimal sub-solutions are propagated from the bottom to the top-level of the circuit hierarchy.

Table 4.9 shows the solutions of the gate shut-off COP for the differential stage series from Fig. 4.3. In this experiment, the information that the nets (in, ip), $(n1, n2)$ and $(outn, outp)$ of the differential stage series should be matched in power-down mode has not been propagated from its subcircuits $DFS1$ and $DFS2'$ to the top-level schematic. The gate shut-off COP has five valid solutions in this case, the other eleven computed solutions have been discarded as they would have caused a short-circuit path in the circuit.

$\mathbf{v}_i^*$	v_{in}	v_{ip}	v_{n1}	v_{outn}	v_{outp}	v_{supply}	v_{ground}
$\mathbf{v}_1^*$	gnd	gnd	gnd	vdd	gnd	vdd	gnd
$\mathbf{v}_2^*$	gnd	gnd	gnd	vdd	vdd	vdd	gnd
$\mathbf{v}_3^*$	vdd	gnd	gnd	vdd	gnd	vdd	gnd

Table 4.10.: Power-down mode implementation variants for differential stage $DFS1$ required to implement the solutions of the differential stage series shown in Table 4.9.

$\mathbf{v}_i^*$	v_{in}	v_{ip}	v_{n1}	v_{nrip1}	v_{outn}	v_{outp}	v_{pwd}	v_{supply}	v_{ground}
$\mathbf{v}_1^*$	gnd	vdd	gnd	$pullVdd$	gnd	gnd	vdd	vdd	gnd
$\mathbf{v}_2^*$	gnd	vdd	gnd	$pullVdd$	gnd	vdd	vdd	vdd	gnd
$\mathbf{v}_3^*$	vdd	vdd	gnd	$pullVdd$	gnd	gnd	vdd	vdd	gnd

Table 4.11.: Power-down mode implementation variants for differential stage $DFS2'$ required to implement the solutions of the differential stage series shown in Table 4.9.

Tables 4.10 and 4.11 show the solutions required to implement the different power-down mode implementation variants of the differential stage series from Table 4.9. The solutions for $DFS1$ have thereby been computed by voltage propagation as there is no floating node in the circuit after setting the pin voltages of $DFS1$ according to $\mathbf{v}_1^*$ - $\mathbf{v}_5^*$ of the differential stage series. For $DFS2'$, the gate shut-off COP must be solved as $nrip1$ is always floating. Solutions $\mathbf{v}_1^*$ and $\mathbf{v}_3^*$ of $DFS1$ and solutions $\mathbf{v}_1^*$ and $\mathbf{v}_2^*$ of $DFS2'$ have thereby been reused for different solutions of the differential stage series which saves computational effort.

Table 4.12 shows the flat and hierarchical scores of the COP solutions of the differential stage series from Table 4.9 including the scores for its subcircuits $DFS1$ and $DFS2'$. Area and matching are equally weighted, i.e., $w_1 = w_2 = 1$. The flat scores on the top-level circuit are all negative as there is no matching or symmetry information available and three or four power-down transistors are required to implement the respective solution. The underlying sub-circuits $DFS1$ and $DFS2'$ have mostly positive scores, with the exceptions of $DFS1$ $\mathbf{v}_3^*$ and $DFS2'$ $\mathbf{v}_2^*$. These solutions do not fulfil any matching constraint and require up to one power-down transistor for their implementation. The best solution for $DFS1$ is $DFS1$ $\mathbf{v}_2^*$ as it fulfils all six matching constraints in power-down mode and no power-down transistor is inserted into the circuit. Its flat score is

$$flatScore(DFS1, \mathbf{v}_2^*) = hierScore(DFS1, \mathbf{v}_2^*) = 6.0/6.0 = 1.0 \tag{4.40}$$

which also corresponds to its hierarchical score as it does not contain any subcircuit blocks. The best solution for $DFS2'$ is $DFS2'$ $\mathbf{v}_3^*$ with a flat and hierarchical score of 0.0 as it also fulfils all matching constraints and is implemented by one power-down transistor. The solution $SDFS$ $\mathbf{v}_3^*$ of the differential stage series instantiates the two optimal subcircuit solutions $DFS1$ $\mathbf{v}_2^*$ and $DFS2'$ $\mathbf{v}_3^*$ which yields the highest hierarchical score of the five top-level solutions:

$$hierScore(SDFS, \mathbf{v}_3^*) = -0.75 + 1.0 + 0.0 = 0.25 \tag{4.41}$$

This solution is the optimal power-down mode implementation variant for the whole hierarchical circuit. It corresponds to the solution shown in Fig. 4.4 which has been directly

solution	matching	area	subcircuits	scores	
	n_{ok}/n_{match}	n_{pwd}/n_{float}		flat	hier
$SDFS\ \mathbf{v}_1^*$	-	4/4	$DFS1\ \mathbf{v}_1^*$, $DFS2'\ \mathbf{v}_1^*$	-1.0	-1.0
$SDFS\ \mathbf{v}_2^*$	-	4/4	$DFS1\ \mathbf{v}_1^*$, $DFS2'\ \mathbf{v}_2^*$	-1.0	-1.66
$SDFS\ \mathbf{v}_3^*$	-	3/4	$DFS1\ \mathbf{v}_2^*$, $DFS2'\ \mathbf{v}_3^*$	-0.75	0.25
$SDFS\ \mathbf{v}_4^*$	-	3/4	$DFS1\ \mathbf{v}_3^*$, $DFS2'\ \mathbf{v}_1^*$	-0.75	-1.08
$SDFS\ \mathbf{v}_5^*$	-	3/4	$DFS1\ \mathbf{v}_3^*$, $DFS2'\ \mathbf{v}_2^*$	-0.75	-1.75
$DFS1\ \mathbf{v}_1^*$	2/6	-	-	0.33	0.33
$DFS1\ \mathbf{v}_2^*$	6/6	-	-	1.0	1.0
$DFS1\ \mathbf{v}_3^*$	0/6	-	-	0.0	0.0
$DFS2'\ \mathbf{v}_1^*$	4/6	1/1	-	0.66	-0.33
$DFS2'\ \mathbf{v}_2^*$	0/6	1/1	-	-1.0	-1.0
$DFS2'\ \mathbf{v}_3^*$	6/6	1/1	-	0.0	0.0

Area and matching equally weighted, i.e., $w_1 = w_2 = 1$.

Table 4.12.: Flat and hierarchical score for the power-down mode implementation variants of the differential stage series from Table 4.9.

computed by the COP as the matching and symmetry information of the underlying hierarchy levels has been propagated to the top-level schematic. This example emphasizes the importance of sharing matching conditions between hierarchy levels as this additional information will reduce the solution space of the top-level COP significantly which avoids the exploration of suboptimal and invalid implementation variants and hence reduces the computational effort of the power-down synthesis method.

4.3.3. Computational Complexity

The hierarchical synthesis approach splits the power-down synthesis problem for a hierarchical design into smaller sub-problems. This reduces the overall complexity of the power-down synthesis problem compared to the approach using circuit flattening as will be shown in the following.

In general, the cardinality $|\mathcal{L}|$ of the COP's solution space is bound by $4^{|N|}$ as the domain of each net of the circuit contains four different voltage levels. The size of the solution space reduces to $4^{|N|-|N_{supply}|-|P_{top,known}|}$ as the voltage levels of the supply nets N_{supply} are known and the user may specify voltages at a subset of the top-level pins $P_{top,known} \subseteq P_{top}$ of the circuit.

The supply nets and the pins of the subcircuits are shorted with the supply nets and internal

nets of the above hierarchy level during circuit flattening, which yields:

$$\begin{aligned} |N_{flat}| &= \underbrace{|N_{top} \cup N_{supply}|}_{\text{top-level and supply nets}} + \underbrace{\sum_{t_i \in T} y_i \cdot (|N_i| - |N_{supply,i}| - |P_i|)}_{\text{subcircuits, supply nets and pins shorted}} \\ |\mathcal{L}_{flat}| &= 4^{|N_{flat}| - |N_{supply}| - |P_{top,known}|} \\ &= 4^{|N_{top} \cup N_{supply}| + \sum_{t_i \in T} y_i \cdot \left(|N_i| - |N_{supply,i}| - |P_i|\right) - |N_{supply}| - |P_{top,known}|} \\ &= 4^{|N_{top} \setminus N_{supply}| - |P_{top,known}| + \sum_{t_i \in T} y_i \cdot \left(|N_i| - |N_{supply,i}| - |P_i|\right)} \end{aligned} \tag{4.42}$$

with y_i as the total number of occurrences of a specific subcircuit type $t_i \in T$ in the hierarchical circuit. The supply nets are first made globally available on the top-level and are then shorted with their corresponding occurrences in the subcircuit blocks. The solution space of the COP of the flat power-down synthesis approach can be reduced to

$$|\mathcal{L}'_{flat}| = 4^{|N_{float,flat}|} \tag{4.43}$$

if it is formulated based on the voltage propagation result for the flattened circuit.

The hierarchical approach presented by this work preserves the circuit's hierarchy and reuses intermediate results whenever possible. Starting with the top-level, it formulates and solves the COP from eq. (4.15) for each hierarchy level and subcircuit individually, which partitions the overall synthesis problem into smaller sub-tasks:

$$|\mathcal{L}_{hier}| = \underbrace{4^{|N_{top}| - |N_{supply,top}| - |P_{top,known}|}}_{\text{top-level COP}} + \underbrace{\sum_{t_i \in T} x_i \cdot 4^{\left(|N_i| - |N_{supply,i}| - |P_i|\right)}}_{\text{individual subcircuit COPs}} \tag{4.44}$$

with x_i as the number of different pin voltage combinations for which the COP has to be solved for a subcircuit of type $t_i \in T$. The supply nets are in this case individually treated for each hierarchy level and subcircuit.

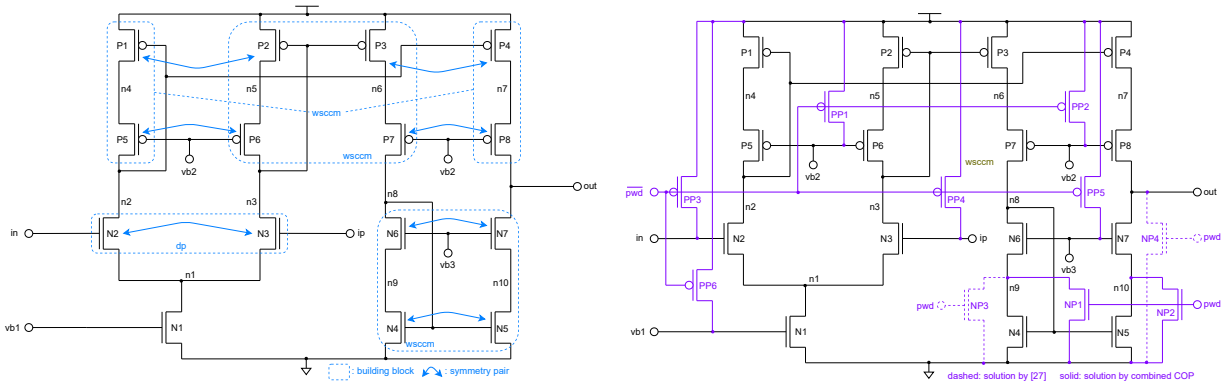
As the COP is formulated based on the floating nodes determined by voltage propagation, eq. (4.44) can be simplified to:

$$|\mathcal{L}'_{hier}| = \underbrace{4^{|N_{float,top}|}}_{\text{top-level COP}} + \underbrace{\sum_{t_i \in T} x_i \cdot 4^{\left(|N_{float,i}|\right)}}_{\text{individual subcircuit COPs}} \tag{4.45}$$

with

$$\begin{aligned} 0 \leq |N_{float,top}| &\leq |N_{top}| - |N_{supply,top}| - |P_{top,known}| \\ 0 \leq |N_{float,i}| &\leq |N_i| - |N_{supply,i}| - |P_i|. \end{aligned} \tag{4.46}$$

The comparison of eqs. (4.42) and (4.44) shows that $|\mathcal{L}_{hier}|$ is significantly smaller than $|\mathcal{L}_{flat}|$ when the gate shutoff COPs of the flat and hierarchical approach are formulated without considering voltage propagation results. The cardinality of the solution space of the new hierarchical approach is typically still smaller compared to the one of the flat approach when



(a) Building blocks and symmetry pairs. (b) Synthesized power-down circuitry.

Figure 4.5.: Analysis and synthesis results for a current mirror operational amplifier [40].

considering voltage propagation results. The flat approach sees all floating nodes at once while the hierarchical approach sees them distributed amongst several hierarchy levels for which it formulates individual COPs. However, it is important to propagate matching and symmetry conditions between the hierarchy levels, as the solution space on the higher levels is less constrained than on device level which could lead to the exploration of an unnecessary large number of sub-optimal solutions.

The differential stage series from Fig. 4.3 has 12 nets in total after circuit flattening. The net voltages of the ground and supply net and the voltage of the power-down signal *pwd* are thereby already known. The size of the solution space of the flat COP is hence bound by $|\mathcal{L}_{flat}| = 4^9 = 262144$. Voltage propagation would detect five floating nodes in the flat circuit, i.e., *in*, *ip*, *n*2, $DFS2'/nrip1$ and *outp*. Hence the solution space of the flat COP could be reduced to $|\mathcal{L}'_{flat}| = 4^5 = 1024$.

The top-level of the differential stage series has four floating nets according to the voltage propagation results from Table 4.1. The gate shutoff COP solution space of the differential stage series is hence bound by $|\mathcal{L}'_{SDFS}| = 4^4 = 256$. The corresponding COP has only one valid solution, hence the gate shut-off COP has to be formulated and solved for the differential stages $DFS1$ and $DFS2'$ only once. $DFS1$ has no floating nets according to Table 4.5 and hence $|\mathcal{L}'_{DFS1}| = 4^0 = 1$. $DFS2'$ has one floating net according to Table 4.6 and hence $|\mathcal{L}'_{DFS2'}| = 4^1 = 4$. The overall size of the solution space for the hierarchical circuit is then $|\mathcal{L}'_{hier}| = |\mathcal{L}'_{SDFS}| + |\mathcal{L}'_{DFS1}| + |\mathcal{L}'_{DFS2'}| = 4^4 + 1 + 4 = 261$, which is significantly smaller compared to the flat approach.

4.4. Experimental Results

In the following, experimental results for five different circuits are presented. The first three results demonstrate the advantages of a unified constraint programming approach compared to the two-stage approach presented by [10]. Afterwards, synthesis results for two hierarchical circuits are presented.

solution	v_{in}	v_{ip}	v_{out}	v_{vb1}	v_{vb2}	v_{vb3}	v_{n1}	v_{n2}
new	$pullVdd$	$pullVdd$	gnd	$pullVdd$	$pullVdd$	$pullVdd$	gnd	gnd
[10]	$pullVdd$	$pullVdd$	$pullGnd$	$pullVdd$	$pullVdd$	$pullVdd$	gnd	gnd

solution	v_{n3}	v_{n4}	v_{n5}	v_{n6}	v_{n7}	v_{n8}	v_{n9}	v_{n10}
new	gnd	vdd	vdd	vdd	vdd	gnd	$pullGnd$	$pullGnd$
by [10]	gnd	vdd	vdd	vdd	vdd	gnd	$pullGnd$	gnd

Table 4.13.: Power-down synthesis results of the new method from Chap. 4 and the approach by [10] for the current mirror operational amplifier.

building block	matched Nets	eq. (4.20)		eq. (4.26)	
		[10]	new	[10]	new
$dp(N2, N3)$	(in, ip)	✓	✓	✓	✓
	$(n2, n3)$	✓	✓	✓	✓
$wsccm(P1, P4, P5, P8)$	$(n4, n7)$	✓	✓	✓	✓
	$(n2, out)$	✓	✓	✗	✓
$wsccm(P2, P3, P6, P7)$	$(n5, n6)$	✓	✓	✓	✓
	$(n3, n8)$	✓	✓	✓	✓
$wsccm(N4, N5, N6, N7)$	$(n8, out)$	✓	✓	✗	✓
	$(n9, n10)$	✓	✓	✗	✓

Table 4.14.: Comparison of the building block matching results for the current mirror operational amplifier for the new synthesis method from Chap. 4 and the approach by [10].

4.4.1. Current Mirror Operational Amplifier

Fig. 4.5 shows the schematic of a current mirror operational amplifier [40]. The left side of the figure shows the building blocks and symmetry pairs of the circuit, the right side shows it augmented by additional power-down circuitry computed by the COP from Sec. 4.3 and [10].

Structure recognition identifies one differential pair, $dp(N2, N3)$, and three wide-swing cascode current mirrors, $wsccm(P1, P4, P5, P8), wsccm(P2, P3, P6, P7), wsccm(N4{-}N7)$ in the circuit. Furthermore, symmetry analysis computes seven symmetrical device pairs in the circuit. The devices are labelled such that $(Ni, Ni{+}1)$ with $i = 2, 4, 6$ and $(Pj, Pj{+}1)$ with $j = 1, 3, 5, 7$ form a symmetry pair.

The new power-down synthesis method and the approach by [10] each compute one implementation variant for the given circuit as shown in Table 4.13. The differences between the two solutions are highlighted in red. The two solutions fulfil all generated basic building block matching constraints according to eq. (4.20) as shown in Table 4.14, i.e., matched pin pairs are exposed to the same voltage level in power-down mode. However, the new approach additionally ensures that the power-down transistors are inserted symmetrically at wide-swing cascode current mirrors $wsccm(P1, P4, P5, P8)$ and $wsccm(N4, N5, N6, N7)$ by

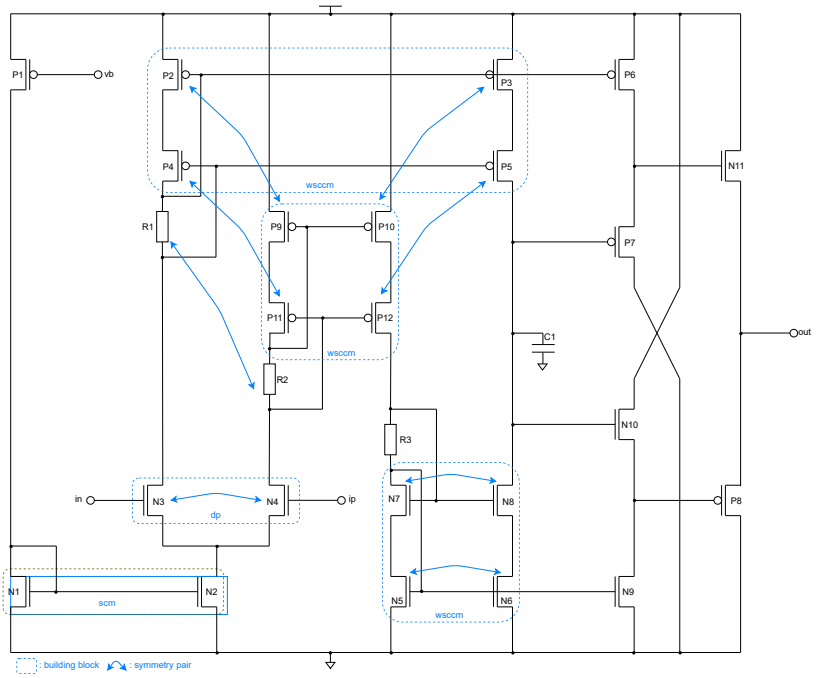


Figure 4.6.: Basic building blocks and symmetry pairs of a low resistance operational amplifier [41].

fulfilling eq. (4.26). The old approach by [10] does not consider symmetrical placement constraints of power-down transistors at matched structures and hence computes four different implementation variants for the shown voltage values in its second constraint optimization problem. From those four solutions, one is arbitrarily chosen, leading to the shown asymmetric transistor placement. Hence, the constraints generated by eq. (4.26) help to reduce the number of power-down mode implementation variants further.

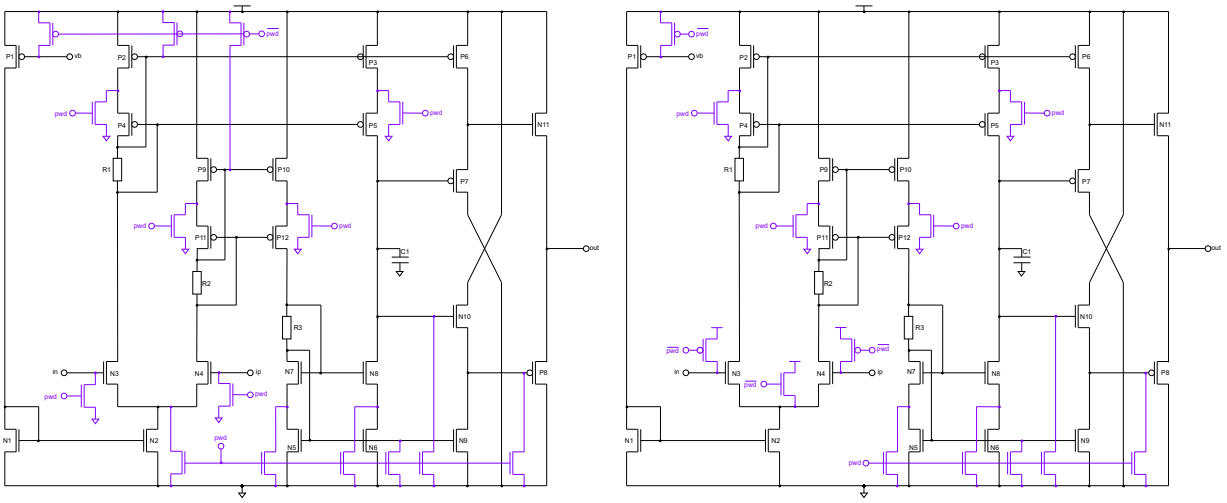
This effect can also be observed in the evaluation of the symmetry pair matching constraints. In general, the approach by [10] does not consider symmetry pair constraints. The computed solution by [10] still fulfils all generated constraints according to eq. (4.24) which usually cannot be guaranteed. However, the power-down transistors are not placed symmetrically at $(N6, N7)$ and $(P7, P8)$ compared to the solution of the new approach. Both solutions are implemented by seven power-down transistors.

4.4.2. Low Resistance Operational Amplifier

Fig. 4.6 shows the schematic of a low resistance operational amplifier [41]. Structure recognition (App. A) identifies one simple current mirror, one differential pair and three wide-swing cascode current mirrors in the circuit. Furthermore, seven symmetry pairs are detected by symmetry analysis (App. B).

Fig. 4.7 shows two power-down mode implementation variants computed by the new COP. For the first solution, the weights w_1 and w_2 have been chosen such that matching is prioritized over area. It fulfils about 80% of the generated matching and 100% of the generated symmetry constraints and can be implemented by 15 power-down transistors.

The second solution minimizes the area requirements of the power-down circuitry. It is implemented by 13 transistors, two transistors less compared to the first solution at the cost



(a) Maximize matching. (b) Minimize area.

Figure 4.7.: Power-down synthesis results for the low resistance operational amplifier from Fig. 4.6.

of sacrificing about 10% of matching, i.e., the constraint at the drain pins of the simple current mirror $scm(N1, N2)$ is not fulfilled any more.

The solution computed by [10] corresponds to the one shown in Fig. 4.7 as it always maximizes matching before determining the placement of the power-down transistors. Hence, the approach by [10] cannot compute the second solution which might be more relevant for area critical designs.

4.4.3. Fully Differential Operational Amplifier

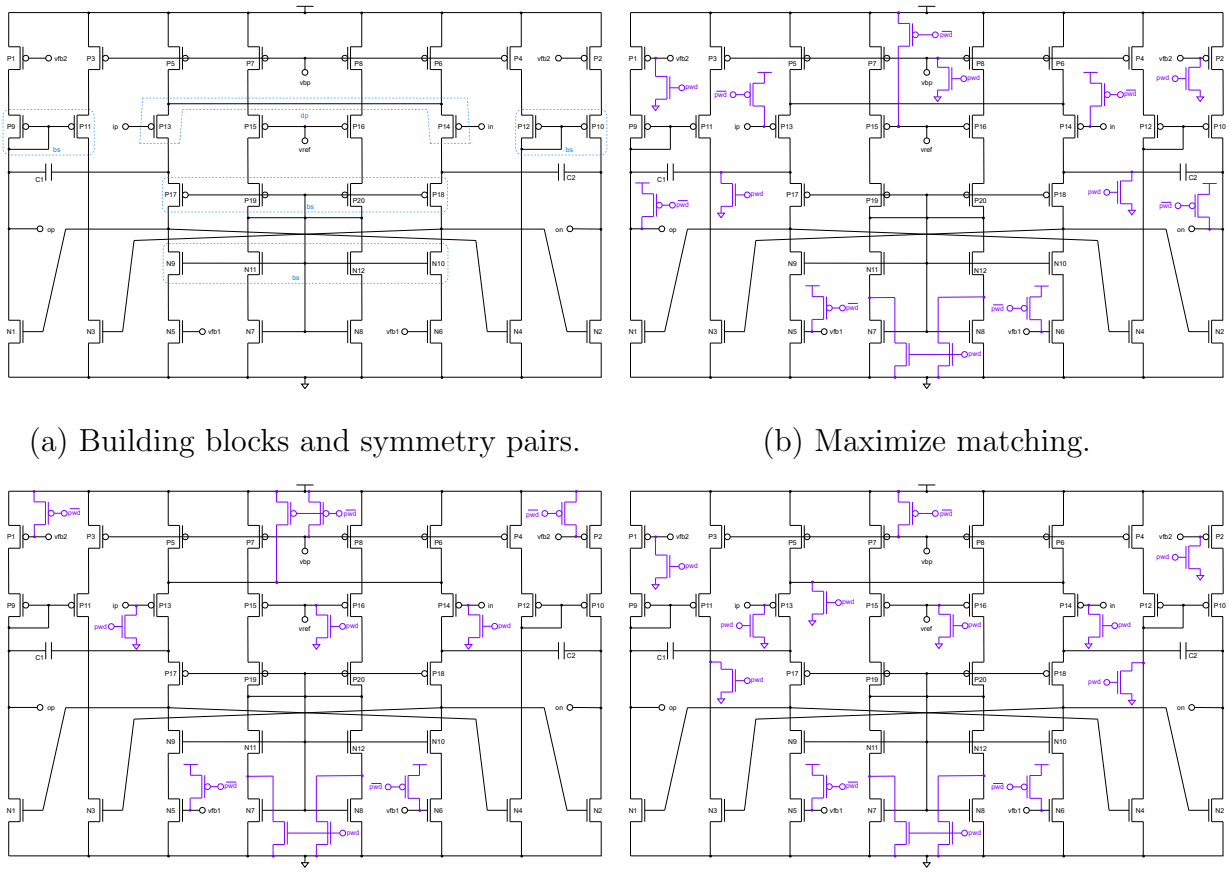
Fig. 4.8 shows several analysis and synthesis results for a fully differential operational amplifier [42]. It contains one differential pair, two bias shifters and two bias shifter banks.

The devices of the circuit are labelled such that $(Ni, Ni+1)$ with $i = 1, 3, ..., 11$, $(Pj, Pj+1)$ with $j = 1, 3, ..., 19$ and $(C1, C2)$ form a symmetry pair.

Power-down synthesis has been repeated three times for this circuit with different weights. The first solution (Fig. 4.8a) maximizes matching and fulfils all generated matching conditions. It requires twelve power-down transistors for its implementation which have been placed symmetrically into the operational amplifier.

The second solution (Fig. 4.8b) minimizes the area of the power-down circuitry and requires nine transistors. These transistors are also inserted symmetrically into the circuit, however, only about 45% percent of the matching conditions are fulfilled.

The last solution (Fig. 4.8c) represents a trade-off between the two design goals “maximize matching” and “minimize area”. With two more additional power-down transistors, i.e., 11 in total, about 82% of the generated matching conditions can be fulfilled, an increase by 37% compared to the minimum area solution.



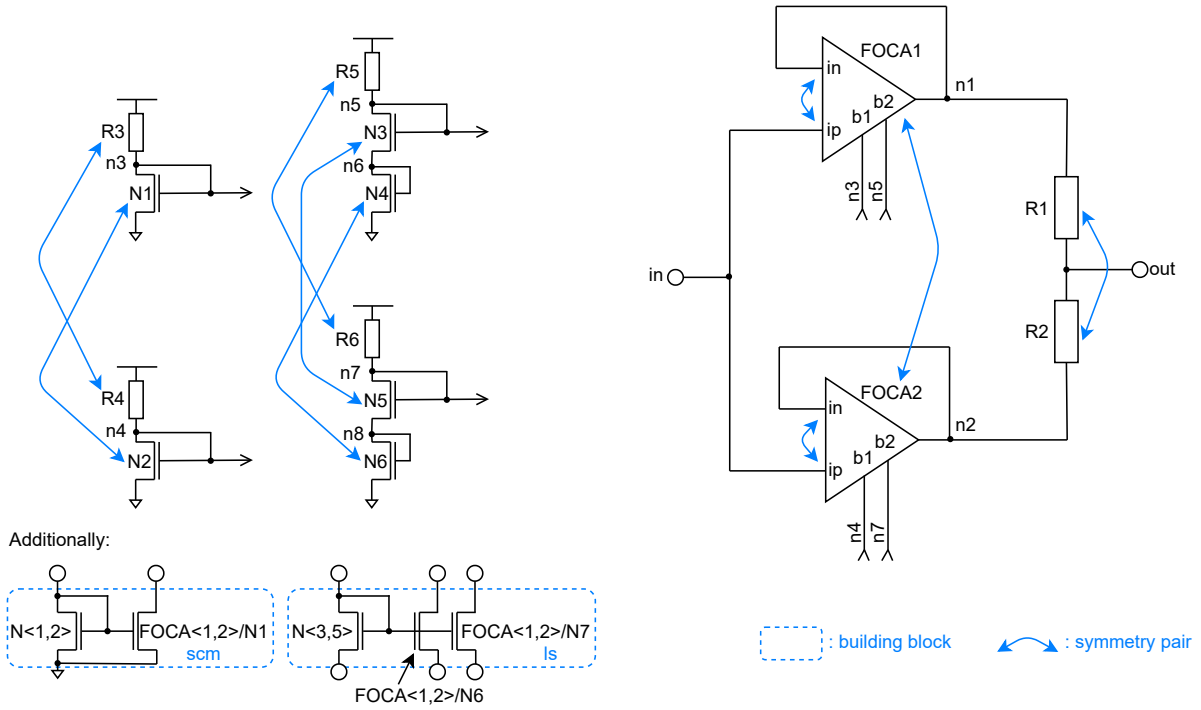
(a) Building blocks and symmetry pairs.

(b) Maximize matching.

(c) Minimize area.

(d) Trade-off.

Figure 4.8.: Analysis and synthesis results for a fully differential operational amplifier [42].



(a) bias circuitry (b) core circuit

Figure 4.9.: Symmetry and matching constraints of the dual gain amplifier from Fig. 3.6.

The second and third solution can only be computed by the unified constraint optimization problem formulated by eq. (4.15), hence, providing a designer more freedom in the selection of suitable power-down circuitry for the target application compared to [10].

4.4.4. Dual Gain Amplifier

In the following, the power-down circuitry for the dual gain amplifier from Fig. 4.9 with its two folded cascode amplifiers $FOCA1$ and $FOCA2$ from Fig. 4.10 is synthesized. The dual gain amplifier has four problematic current paths in its bias circuitry:

- $P1 = (R3, N1)$
- $P2 = (R4, N2)$
- $P3 = (R5, N3, N4)$
- $P4 = (R6, N5, N6)$

These have been detected by voltage propagation and short circuit path computation from Secs. 3.2 and 3.3. All paths run over at least one diode-connected transistor. Hence, a diode rip-up is applied to each of them. The transistors $NP1$, $NP2$, $NP3$ and $NP4$ are inserted by the rip-up heuristic into the dual gain amplifier to disable the diode-configurations of $N1$, $N2$, $N3$ and $N5$ in power-down mode, respectively. The heuristic thereby chooses $N3$ and $N5$ over $N4$ and $N6$ as rip-up points as each of these transistors is part of a level shifter. Hence, disabling the diode-configurations of $N3$ and $N5$ in power-down mode switches off

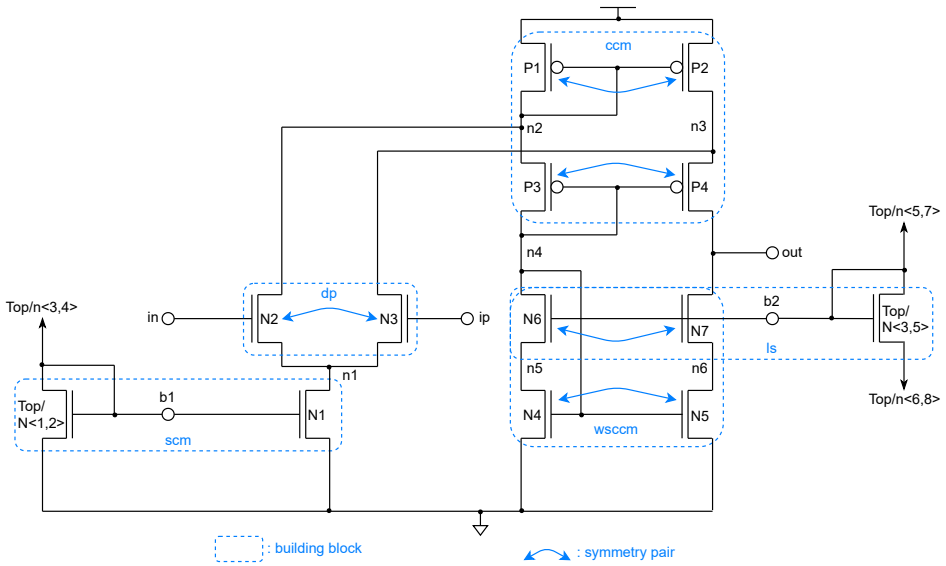


Figure 4.10.: Symmetry and matching constraints for $FOCA1$ and $FOCA2$ from Fig. 4.9.

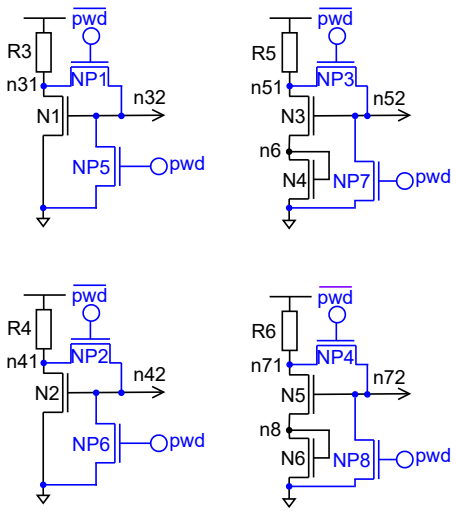


Figure 4.11.: Bias circuitry of the dual gain amplifier from Fig. 4.9a after rip-up and gate shut-off.

	core circuit				ripped-up bias circuit			
$\mathbf{v}_i^*$	v_{in}	v_{out}	v_{n1}	v_{n2}	v_{nX1}	v_{nX2}	v_{n6}	v_{n8}
$\mathbf{v_1}^*$	$pullGnd$	$pullGnd$	gnd	gnd	vdd	$pullGnd$	gnd	gnd
$\mathbf{v_2}^*$	$pullVdd$	$pullVdd$	vdd	vdd	vdd	$pullGnd$	gnd	gnd

Table 4.15.: Solutions of the gate shut-off COP for the dual gain amplifier for its core circuit and its ripped-up bias circuits from Figs. 3.8b and 4.11.

$\mathbf{v}_i^*$	v_{in}	v_{ip}	v_{out}	v_{vb1}	v_{vb2}	v_{n1}	v_{n2}	v_{n3}	v_{n4}	v_{n5}	v_{n6}
$\mathbf{v_1}^*$	gnd	gnd	gnd	gnd	gnd	$pullVdd$	vdd	$pullVdd$	vdd	gnd	gnd
$\mathbf{v_2}^*$	vdd	vdd	vdd	gnd	gnd	vdd	vdd	vdd	vdd	gnd	gnd

Table 4.16.: Synthesized implementation variants for the folded cascode amplifiers $FOCA1$ and $FOCA2$ from Fig. 3.9.

more current paths in the dual gain amplifier. Gate shut-off afterwards inserts power-down transistors $NP5$ - $NP8$ in each solution of the gate shut-off COP of the dual gain amplifier into the circuit such that $N1$, $N2$, $N3$ and $N5$ are forced into the non-conducting state in power-down mode. The resulting power-down circuitry for the bias circuit of the dual gain amplifier is shown in Fig. 4.11.

The two folded cascode amplifiers $FOCA1$ and $FOCA2$ from Fig. 4.10 remain unchanged after rip-up as no problematic current path runs through them.

Tables 4.15 and 4.16 show the power-down mode implementation variants computed by the gate shut-off COP for the ripped-up dual gain amplifier and its folded cascode amplifiers $FOCA1$ and $FOCA2$. The symmetry and matching constraints have thereby been generated according to Figs. 4.9 and 4.10. The shown solutions fulfil all generated matching constraints and can be implemented with a minimum number of power-down transistors on their respective hierarchy level.

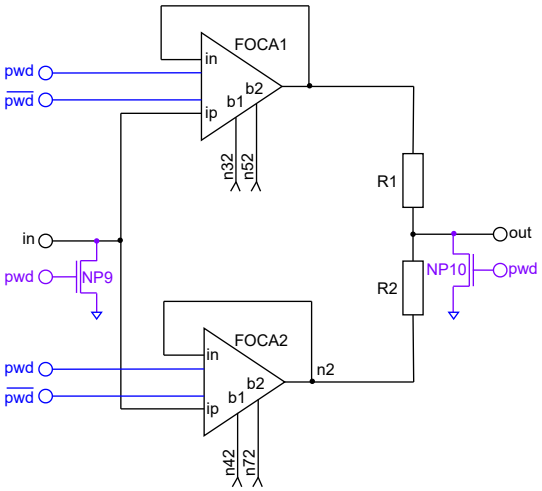
Fig. 4.12 shows the schematics corresponding to the solutions $\mathbf{v}_1^*$ and $\mathbf{v}_2^*$ of the dual gain amplifier (DGA) and the folded cascode amplifier (FOCA), respectively. Their corresponding flat and hierarchical scores are annotated in the table at the bottom of the figure.

Solutions $\mathbf{v}_1^*$ and $\mathbf{v}_2^*$ of the dual gain amplifier fulfil all matching constraints and can be implemented with four gate shut-off transistors, yielding a flat score of $\frac{32}{32} - \frac{6}{8} = 0.25$ under the assumption that the design goals "maximize matching" and "minimize area" are equally weighted, i.e., $w_1 = w_2 = 1$.

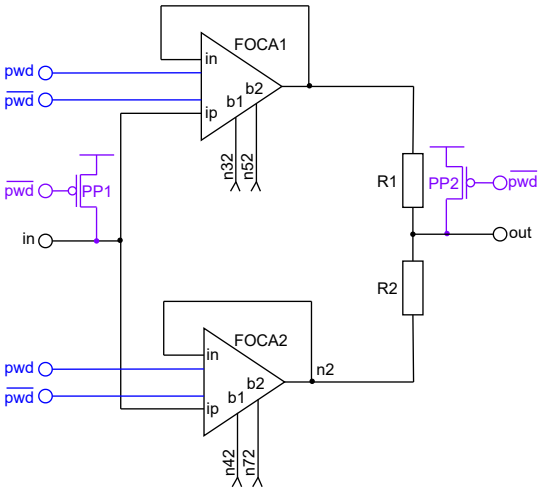
Both computed solutions for the folded cascode amplifier also fulfil all matching constraints. However, two power-down transistors are required to implement $\mathbf{v}_1^*$ and none for $\mathbf{v}_2^*$. Hence, the flat score of $\mathbf{v}_1^*$ is significantly lower than the one for $\mathbf{v}_2^*$, i.e., 0.0 compared to 1.0.

The dual gain amplifier solution $\mathbf{v}_1^*$ instantiates the folded cascode amplifier solution $\mathbf{v}_1^*$ two times, resulting in an overall hierarchical score of $0.25 + 2 \cdot 0.0 = 0.25$. The second solution of the dual gain amplifier implements the power-down circuitry of $FOCA1$ and $FOCA2$ with their second solution $\mathbf{v}_2^*$, yielding a hierarchical score of 2.25.

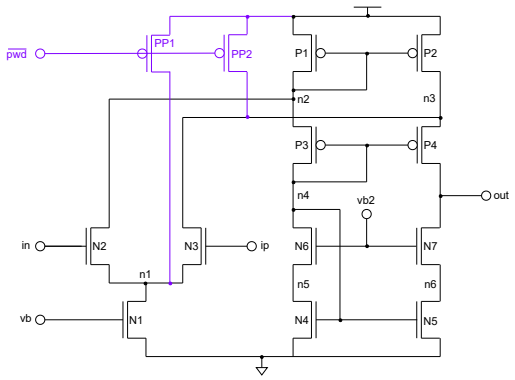
Hence, $\mathbf{v}_2^*$ is the optimal solution for the overall circuit as it can be implemented with fewer power-down transistors compared to solution $\mathbf{v}_1^*$.



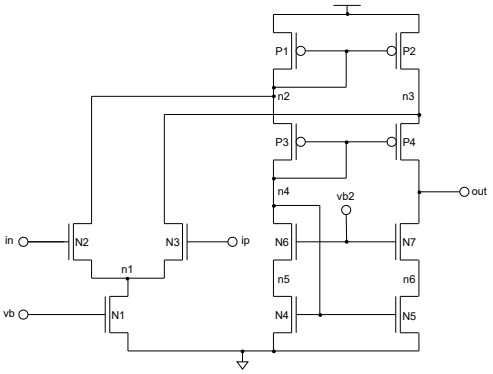
DGA solution $\mathbf{v}_1^*$



DGA solution $\mathbf{v}_2^*$



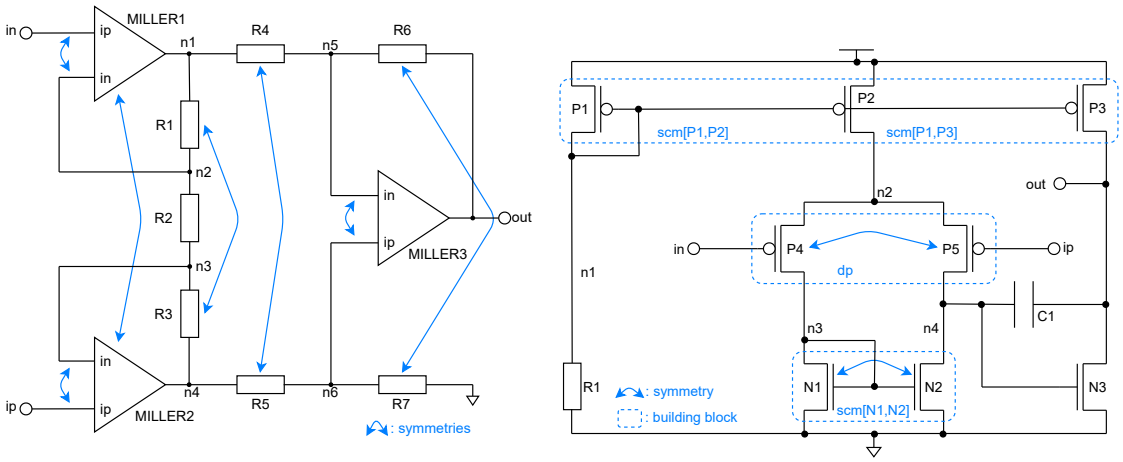
FOCA solution $\mathbf{v}_1^*$



FOCA solution $\mathbf{v}_2^*$

solution	matching	area	subcircuits	scores	
	n_{ok}/n_{match}	n_{pwd}/n_{float}		flat	hier
DGA $\mathbf{v}_1^*$	32/32	6/8	2x FOCA $\mathbf{v}_1^*$	0.25	0.25
DGA $\mathbf{v}_2^*$	32/32	6/8	2x FOCA $\mathbf{v}_2^*$	0.25	2.25
FOCA $\mathbf{v}_1^*$	26/26	2/2	-	0.0	0.0
FOCA $\mathbf{v}_2^*$	26/26	-	-	1.00	1.00

Figure 4.12.: Valid power-down mode implementation variants for the dual gain amplifier (DGA) and folded cacscode amplifier (FOCA) from Tables 4.15 and 4.16, and their corresponding flat and hierarchical scores.



(a) HIIDA (b) Miller OTA

Figure 4.13.: Basic building blocks and symmetry pairs of a high input impedance differential amplifier (HIIDA) [38] and its Miller operational transconductance amplifier (OTA) subcircuit blocks.

4.4.5. High Input Impedance Differential Amplifier

In Sec. 3.5, the power-down mode of the high input impedance differential amplifier (HIIDA) from Fig. 3.10 has been verified. In the following, fault-free power-down circuitry is synthesized for the original circuit from Fig. 4.13. The supply nets are given as $N_{supply} = N_{vdd} \cup N_{gnd} = \{supply, ground\}$ and the input and output pins are left floating.

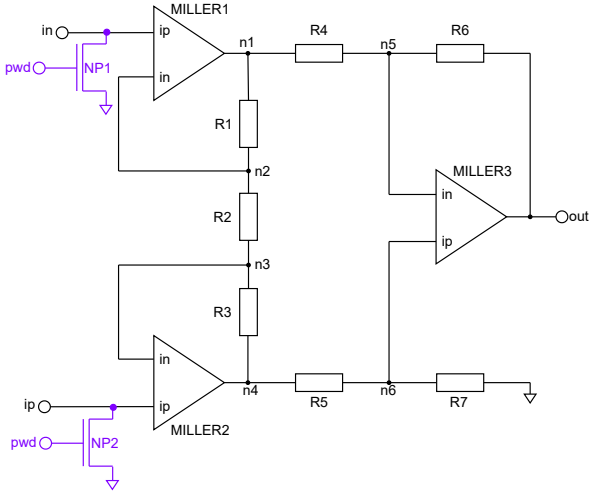
In the first step, three definite short circuit paths,

$$P_{<1,2,3>} = (MILLER<1,2,3>/P1, MILLER<1,2,3>/R1), \tag{4.47}$$

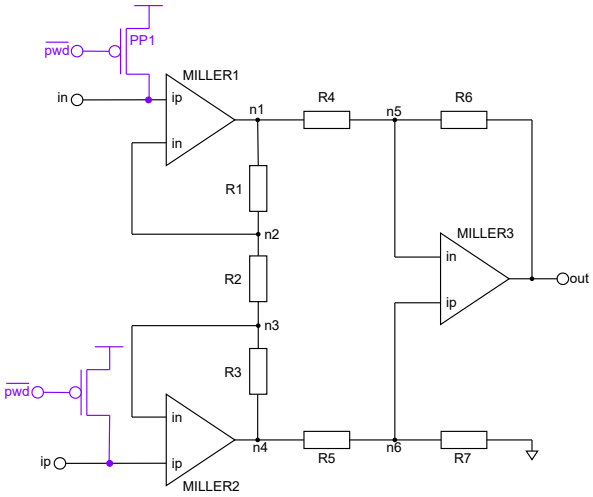
are identified. The diode-rip up pattern is applied to these paths as $P1$ is diode-connected. The rip-up is performed only once for the Miller operational transconductance amplifier (OTA) and the modified circuit is used to replace the remaining two Miller OTAs in the original circuit.

The gate shut-off procedure computes one optimal solution $\mathbf{v}_1^*$ for the HIIDA, using the matching and symmetry information from Fig. 3.11. Furthermore, a second sub-optimal solution $\mathbf{v}_2^*$ is provided to emphasize on the importance of propagating structural and symmetry information between circuit hierarchy levels: the symmetry conditions at the differential inputs of the Miller OTAs have not been made available on the top-level for that solution. The synthesized power-down circuitries for each solution are highlighted in blue in the schematics, their corresponding flat and hierarchical scores according to eqs. (4.39) and (4.38) are displayed in the table underneath them.

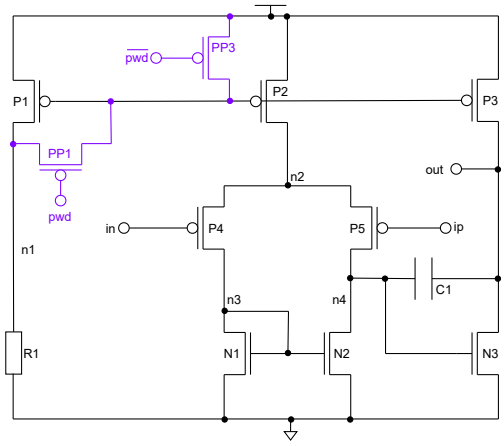
The first solution is fully symmetric, i.e., all symmetry constraints are fulfilled. The symmetry conditions at the differential inputs of $MILLER1$, $MILLER2$ and $MILLER3$ ensure that *in* and *ip* of the high input impedance differential amplifier are pulled to *gnd*, yielding only one solution for the top-level. The pins of the three Miller OTAs are all exposed to *gnd*. Hence, gate shut-off has to be performed only once for the Miller OTA which results in Miller



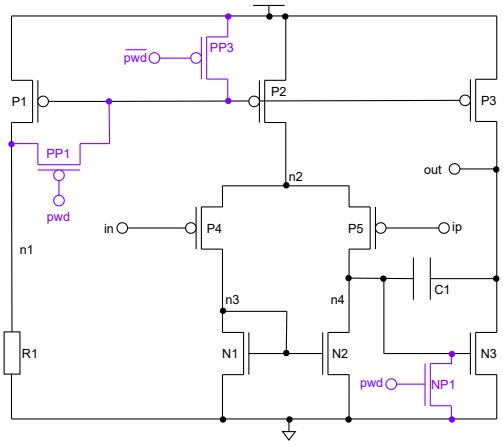
HIIDA Solution $\mathbf{v}_1^*$



HIIDA Solution $\mathbf{v}_2^*$



Miller OTA Solution $\mathbf{v}_1^*$



Miller OTA Solution $\mathbf{v}_2^*$

solution	matching	area	subcircuits	scores	
	n_{ok}/n_{match}	n_{pwd}/n_{float}		flat	hier
HIIDA $\mathbf{v}_1^*$	12/12	2/2	3x Miller OTA $\mathbf{v}_1^*$	0.0	0.0
HIIDA $\mathbf{v}_2^*$	10/12	2/2	1x Miller OTA $\mathbf{v}_1^*$, 2x Miller OTA $\mathbf{v}_2^*$	-0.167	-0.475
Miller OTA $\mathbf{v}_1^*$	13/13	1/1	-	0.0	0.0
Miller OTA $\mathbf{v}_2^*$	11/13	2/2	-	-0.154	-0.154

Figure 4.14.: Synthesis results for the high input impedance differential amplifier (HIIDA) and its Miller operational transconductance amplifiers (OTAs).

OTA solution $\mathbf{v}_1^*$. The computed solution fulfils all matching and symmetry constraints and its power-down circuitry is implemented by only one gate shut-off transistor which shuts off the PMOS current mirror bank $scm(P1, P2, P3)$ in power-down mode. Its corresponding flat score is $\frac{13}{13} - \frac{1}{1} = 0.0$ and the hierarchical score of solution $\mathbf{v}_1^*$ of the high input impedance differential amplifier evaluates to $0.0 + 3 \cdot 0.0 = 0.0$.

In a second experiment, the information that the differential inputs of the Miller OTAs should be matched was not propagated to the top-level. In this case, a second solution $\mathbf{v}_2^*$ is computed for the high input impedance differential amplifier which pulls its input pins *in* and *ip* to *vdd*. Now, gate shut-off has to be performed twice for the Miller OTAs: the first time with input voltages $V_{P,MILLER3} = (v_{in}, v_{ip}, v_{out}) = (gnd, gnd, gnd)$ and the second time with $V_{P,MILLER<1,2>} = (v_{in}, v_{ip}, v_{out}) = (gnd, vdd, gnd)$. The solution for $V_{P,MILLER3}$ corresponds to Miller OTA solution $\mathbf{v}_1^*$, the one for $V_{P,MILLER<1,2>}$ is shown as Miller OTA solution $\mathbf{v}_2^*$ on the bottom right of Fig. 4.14. Here, the problem of not propagating matching and symmetry conditions throughout the hierarchy levels becomes obvious: the differential pair $dp(P4, P5)$ is stressed and two gate shut-off transistors are required two implement this solution. Its flat score is -0.154 and the hierarchical score of high input impedance differential amplifier solution $\mathbf{v}_2^*$ computes to -0.475. This solution is discarded in favor of $\mathbf{v}_1^*$ during the later selection process, but this solution would have never been computed in the first place if the structural and symmetry information would have been made available between the two hierarchy levels.

Hence, this second experiment demonstrates another advantage of propagating structural information between different hierarchy levels for power-down synthesis: sub-optimal solutions are not further explored which reduces the computational complexity of the implemented algorithms.

5. Library-Free Structure Recognition

The power-down mode verification and synthesis methods presented in Chaps. 3 and 4 make use of the structure recognition method described in App. A. Structure recognition identifies analog basic building blocks which require matching, e.g., differential pairs and current mirrors. Power-down verification and power-down synthesis evaluate the identified matching conditions to detect faulty or to construct faultless power-down circuitry.

The structure recognition method described in App. A uses a predefined building block library which is similar to the one of the enhanced sizing rules method presented in [10]. Fig. 5.1 shows a small excerpt of the building block library presented in App. A.

The array library L_{array} contains parallelly connected devices of the same type which show the same electrical behavior as a device of their combined device dimensions. The pair library L_{pair} is organized hierarchically in ranks. Rank one contains analog building blocks which are formed by two arrays, e.g., a simple current mirror and a bias shifter. The higher ranks contain building blocks which are formed by two lower ranked elements, e.g., a cascode current mirror consists of a bias shifter and simple current mirror.

Apart from power-down verification and synthesis, the structure recognition results of this library can also be used to generate sizing constraints for numerical circuit optimization tools [4] and placement constraints for layout generators [17], which improve the quality of the given designs significantly.

In [10], a building block library dedicated to track the power-down signals through inverters, ESD protection circuitry and simple level shifters was presented. The results are used to partition the circuit into its digital power-down circuitry and its original analog circuit topology. This method has been adapted to hierarchical designs as described in Sec. 3.4.1.

These examples illustrate the importance of structure recognition in analog circuit design. Structure recognition has one main limitation: the size and the scope of the provided building block library. The afore mentioned libraries contain up to 24 elements of up to six devices. Additionally, the libraries were hand-crafted: the building blocks, their recognition rules and their organization in ranks had to be manually defined. This process is tedious and error prone as forgetting to specify only one recognition rule can lead to wrong structure recognition results.

The reliable identification of larger circuit topologies, e.g., operational amplifiers or level converters, can play an important role for circuit verification and circuit synthesis. Electrostatic discharge (ESD) events can cause permanent damage at interface circuits between different voltage domains [43]. Hence, detecting circuit parts susceptible to ESD damage in a chip and automated verification checks of its implemented ESD protection circuitry can speed up the overall design process of that chip.

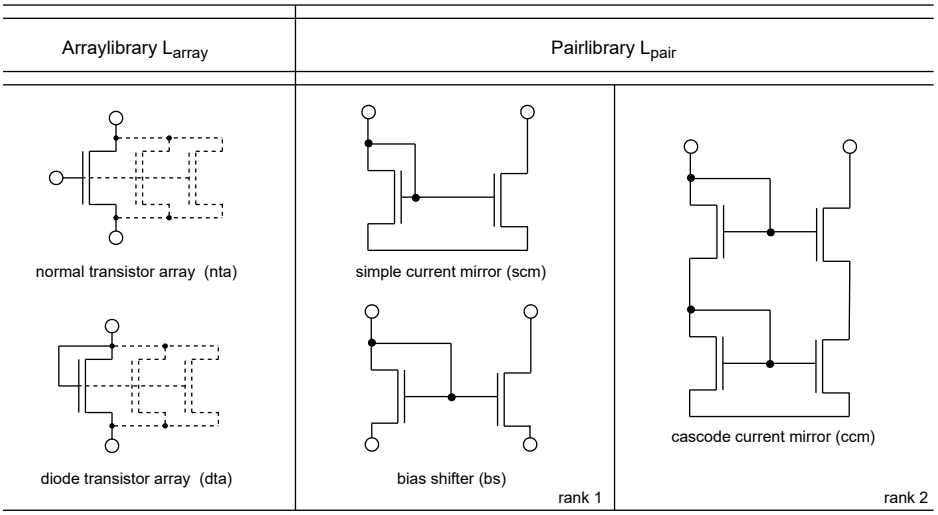


Figure 5.1.: Excerpt of the basic building block library from App. A.

5.1. State of the Art

To overcome the limitation of structure recognition, a given building block library could be extended by adding new building blocks manually to it, until a specific topology, e.g., a Miller OTA, can be identified. Instead, automatic approaches could be used.

In [44], new topological features of an analog circuit are extracted using unsupervised learning techniques and generates a hierarchical building block representation of the circuit. It still relies on predefined rule-sets to be able to identify new structures in a circuit.

[45] uses graph convolutional neural networks to cluster the devices of an analog circuit into its functional blocks, e.g., circuit biases, loads and input pairs, based on geometrical information of the circuit's devices in the schematic. It does not generate a hierarchical building block description of the circuit which could be used for structure recognition.

[46] uses a graph convolutional network to extract and classify subblocks according to their type and functionality, e.g., low noise amplifiers (LNAs), operational transconductance amplifiers (OTAs), mixer or oscillators, in a hierarchical design. Analog basic building blocks inside these subblocks are detected by traditional graph isomorphism approaches. The gathered information is used to annotate netlists for circuit optimization and layout generation.

This work presents a new, deterministic method which automatically decomposes a given circuit topology into its building blocks. The generated building block description can be used by structure recognition to identify the analyzed topology in large netlists.

5.2. Overview

Algorithm 7 outlines the main procedure of the method. It takes a circuit $C(P, N, C_{sub,}, t)$, an array library L_{array} and a pair library L_{pair} as input. It outputs the library L_C which contains all elements required to identify circuit C by structure recognition. The array library must contain the array definitions for all supported device types and configurations in the circuit, e.g., the array library from Fig. A.3. The provided pair library L_{pair} can either

Algorithm 7 Automatic generation of the building block description of a circuit

```
1: procedure LIBRARYGENERATION(C(P, N, C_sub,, t), L_array, L_pair)
2:     L_C = L_array ∪ L_pair
3:     repeat
4:         C_B = structureRecognition(L_C, C)    // See Alg. 14
5:         T = findTopLevelStructures(C_B)
6:         E_new = createNewLibraryElements(T)    // See Alg. 8
7:         extendLibrary(L_C, E_new)    // See Alg. 9
8:     until |T| = 1
9:     return L_C
10: end procedure
```

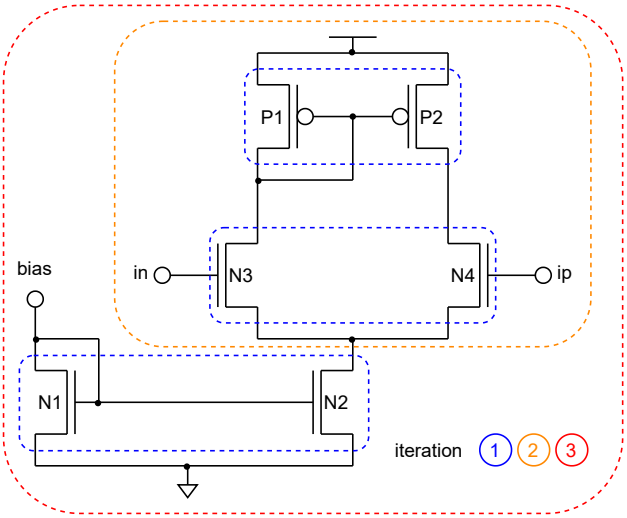


Figure 5.2.: Differential input stage and its generated building blocks.

be empty or contain already predefined ranks and building blocks. In the first case, the pair library of L_C is generated from scratch, in the second case, it is augmented by the missing ranks and library elements required to identify circuit C.

In the first step, L_C is initialized by creating a copy of L_{array} and L_{pair} (line 2). Then, structure recognition is performed on the given circuit to identify the building blocks specified by L_C (line 4). The top-level structures T, i.e., the building blocks which are not part of another higher ranked block, are extracted from the recognition result C_B in line 5. Algorithm 8 then groups the top-level building blocks into a set of new library elements E_{new} (line 6). It analyzes three neighboring characteristics of possible new building block pairs for that purpose (Sec. 5.3). In line 7, Algorithm 9 extends library L_C by the new elements and additional ranks whenever required (Sec. 5.4). The above steps are repeated until L_C contains the full building block description of the given circuit, i.e., until the set of top-level structures T contains only one element (lines 3 - 8).

Fig. 5.2 shows a differential input stage of an operational amplifier. The colored boxes indicate how the devices and building blocks are automatically grouped into new library elements by Algorithms 7 - 9 until the full differential input stage can be identified by structure recognition. Fig. 5.3 shows the corresponding computed building block library L_C.

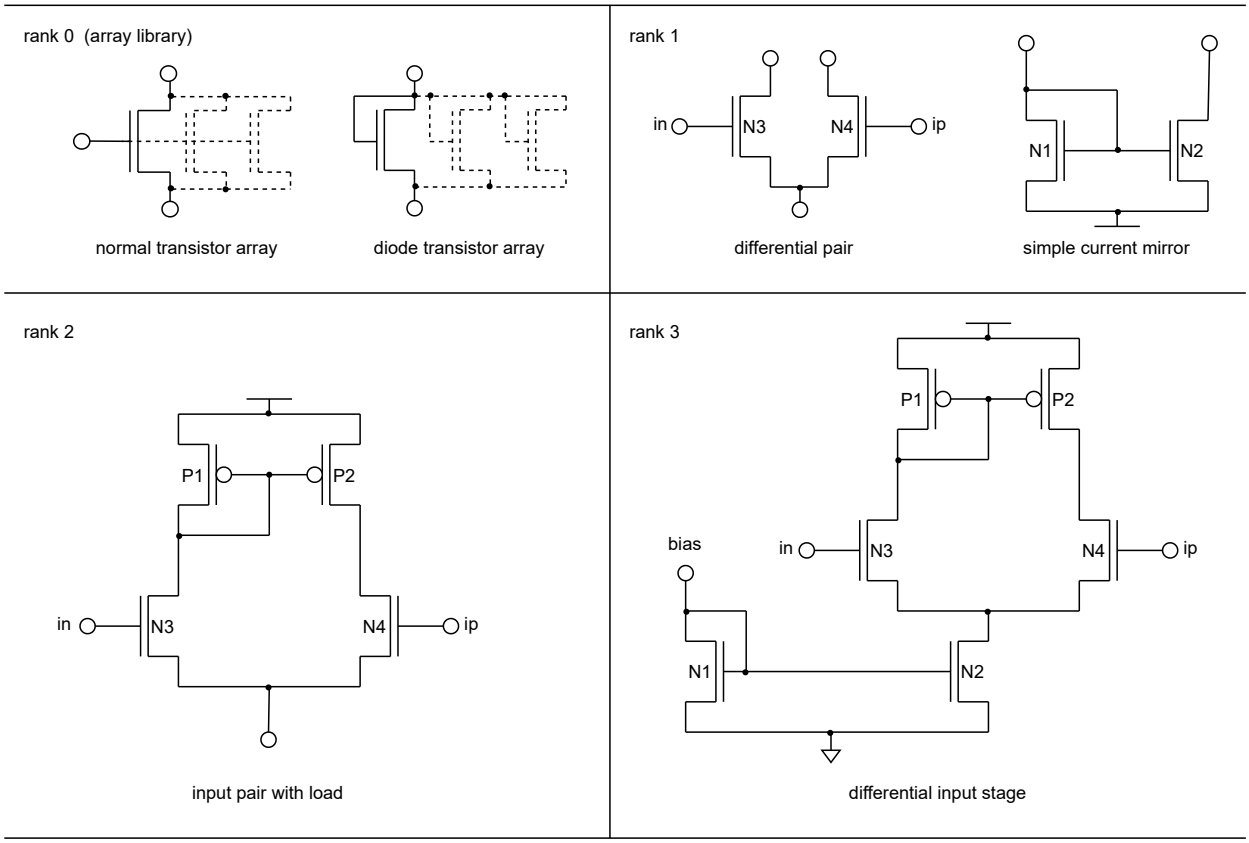


Figure 5.3.: Generated building block library for the differential input stage from Fig. 5.2.

The initial pair library provided to Algorithm 7 was empty, i.e., the pair library to identify the differential input stage has been built from scratch. In the first iteration, the transistors $(N1, N2)$, $(P1, P2)$ and $(N3, N4)$ are grouped into new library elements by Algorithm 8. These elements correspond to the traditional analog basic building blocks "simple current mirror" and "differential pair" and have been labeled accordingly by hand.

In the second iteration, the algorithm groups $[(P1, P2), (N3, N4)]$ together which combines the transconductance with the load of the input stage. Finally, the bias circuitry $(N1, N2)$ is connected to this structure to form the given differential input stage.

5.3. Creation of New Library Elements

Algorithm 8 groups the identified top-level building blocks T into new library elements. Structure recognition should be able to identify the new elements as unambiguous as possible. The algorithm assigns a "group strength" to each possible new building block pair. The group strength is computed based on the *connectivity, substrate type* and *rank difference* of two building blocks according to eq. (5.1). A high group strength indicates an easier identification of the considered building block pair by structure recognition.

Algorithm 8 operates as follows: in line 2, it initializes the set of new library elements E_{new} empty. Then it determines the set of tentative new building blocks B by forming the Cartesian Product of the top-level structures T in line 3. Pairs which are not connected with each other by at least one net are automatically discarded. The remaining pairs are stored

Algorithm 8 Heuristic to discover new building block pairs

```
1: procedure createNewLibraryElements(T)
2:     E_new = ∅
3:     B = (T × T)    // Tentative building blocks.
4:     B_w = sortByGroupStrengths(B)
5:     for all (b_i, b_j) ∈ B_w with decreasing group strengths do
6:         e_new = generateRecognitionRules(b_i, b_j)
7:         E_new = E_new ∪ e_new
8:         eliminateOverlappingBlocks(B_w, b_i, b_j)
9:     end for
10:    removeDuplicateElements(E_new)
11:    return E_new
12: end procedure
```

in the set B_w together with their group strength in decreasing order (line 4). The group strengths of each tentative building block are thereby determined by eq. (5.1). The algorithm iterates over all elements $(b_i, b_j) \in B_w$ with decreasing group strengths (line 5), generates the required recognition rules to identify (b_i, b_j) and stores the new library element e_{new} in the set E_{new} (lines 6 and 7). Afterwards, all tentative building blocks which are overlapping with (b_i, b_j) are eliminated from B_w in line 8. These steps are iteratively repeated until B_w is empty. Afterwards, duplicate elements are removed from E_{new} in line 10. Duplicate elements are created when B_w contains several copies of non-overlapping, but identically connected tentative building blocks. Finally, the algorithm returns the set of new libary elements E_{new} (line 11).

The group strength of a tentative building block pair is calculated as follows:

$$g(b_i, b_j) = 2 \cdot con(b_i, b_j) + sst(b_i, b_j) - \Delta r(b_i, b_j), \tag{5.1}$$

where $con(b_i, b_j)$, $sst(b_i, b_j)$ and $\Delta r(b_i, b_j)$ denote the connectivity, substrate type weight, and the rank level difference between two blocks (b_i, b_j), respectively.

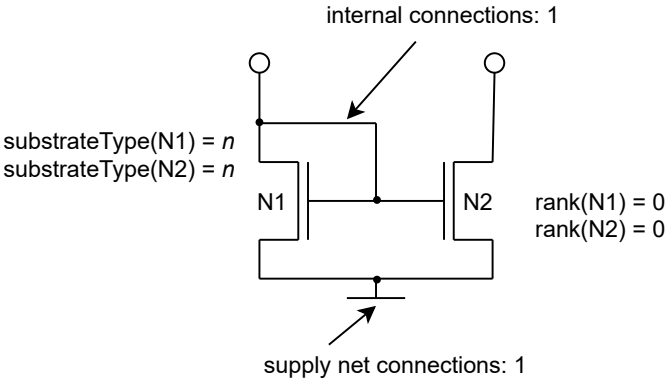
The connectivity weight is defined as:

$$con(b_i, b_j) = 3 \cdot internal(b_i, b_j) + 2 \cdot supply(b_i, b_j). \tag{5.2}$$

The function $supply(b_i, b_j)$ computes the number of connections between b_i and b_j via the supply nets of the circuit and $internal(b_i, b_j)$ denotes the number of connections via internal nets between these two blocks. Typically, many different devices or building blocks are connected with each other by supply nets. Hence, internal connections between two blocks are valued higher compared to connections over supply nets, i.e., the number of building block internal connections is multiplied by factor 3, the number of connections over supply nets by factor 2.

The substrate type weight is defined as:

$$sst(b_i, b_j) = \begin{cases} 2 & \text{, same substrate type } (n \text{ or } p) \\ 1 & \text{, mixed types } (n \text{ and } p) \\ 0 & \text{, else (one block has } u\text{-type)} \end{cases} . \tag{5.3}$$



(a) characteristics

$$con(N1, N2) = 3 + 2 = 5$$
$$sst(N1, N2) = 2$$
$$\Delta r(N1, N2) = |0 - 0| = 0$$
$$g(N1, N2) = 2 \cdot 5 + 2 - 0 = 12$$

(b) group strength

Figure 5.4.: Characteristics and group strength of a simple current mirror.

Analog basic building blocks are typically formed by subblocks of the same substrate type, e.g., differential pairs and current mirrors are solely formed by either NMOS- or PMOS-transistors. These blocks are then combined to input and output stages of operational amplifiers by connecting basic building blocks with different substrate types, e.g, an NMOS differential pair to a PMOS simple current mirror as shown in Fig. 5.2. Tentative building blocks with subblocks of the same substrate type get a weight of two, mixed substrate types of one and all others of zero. These scores ensure that new library elements with the same substrate type are preferably combined to a new building block on lower ranks.

The rank difference between two blocks is defined as:

$$\Delta r(b_i, b_j) = |rank(b_i) - rank(b_j)| \,. \tag{5.4}$$

The function $rank(b_i)$ returns the rank on which the building block can be located in the generated library L_C. Algorithm 7 generates a building block library organized in ranks. Tentative building blocks with a low absolute rank difference between its subblocks should be preferred over blocks with a higher rank difference in order to reduce the number of ranks of the generated library.

Please note: the weights of the connectivity and the substrate type scores of eqs. (5.2) and (5.3) have been heuristically chosen. Furthermore, the connectivity weight is multiplied by two in eq. (5.1) as it is the main characteristic for an unambiguous identification: tightly connected blocks are easier to identify by structure recognition.

Fig. 5.4 shows how the group strength of the simple current mirror $(N1, N2)$ is computed. The two devices share an internal and a supply net connection, have the same substrate type and are located on the same rank. Hence, the group strength of $(N1, N2)$ corresponds to:

$$g(N1, N2) = 2 \cdot (2 + 3) + 2 - |0 - 0| = 12. \tag{5.5}$$

Table 5.1 shows all tentative building blocks and their assigned group strengths of the differential input stage from Fig. 5.2 for each iteration of Algorithm 7. In the first iteration, seven, in the second two and in the last iteration, only one tentative building block is generated. The group strength of each block is computed by Algorithm 8 according to eq. (5.1).

$(b_i, b_j) \in B'_w$	$g(b_i, b_j)$	used
$(N1, N2)$	12	yes
$(P1, P2)$	12	yes
$(N2, N3)$	8	no
$(N2, N4)$	8	no
$(N3, N4)$	8	yes
$(N3, P1)$	7	no
$(N3, P2)$	7	no
$(N4, P2)$	7	no

Iteration 1

$(b_i, b_j) \in B'_w$	$g(b_i, b_j)$	used
$[(N3, N4), (P1, P2)]$	13	yes
$[(N1, N2), (N3, N4)]$	7	no

Iteration 2

$(b_i, b_j) \in B'_w$	$g(b_i, b_j)$	used
$\{[(N3, N4), (P1, P2)], (N1, N2)\}$	6	yes

Iteration 3

Table 5.1.: Weights of the tentative buildings formed in the first iteration of Algorithm 7 for the circuit of Fig. 5.2.

Algorithm 9 Insertion of new library elements

```
1: procedure EXTENDLIBRARY(L_C, E_new)
2:     for all e_new = (b_i, b_j) ∈ E_new do
3:         rank = maxRank(b_i, b_j) + 1
4:         if rank > maxRank(L_C) then
5:             addNewRank(L_C, rank)
6:         end if
7:         addNewElement(e_new, rank)
8:     end for
9: end procedure
```

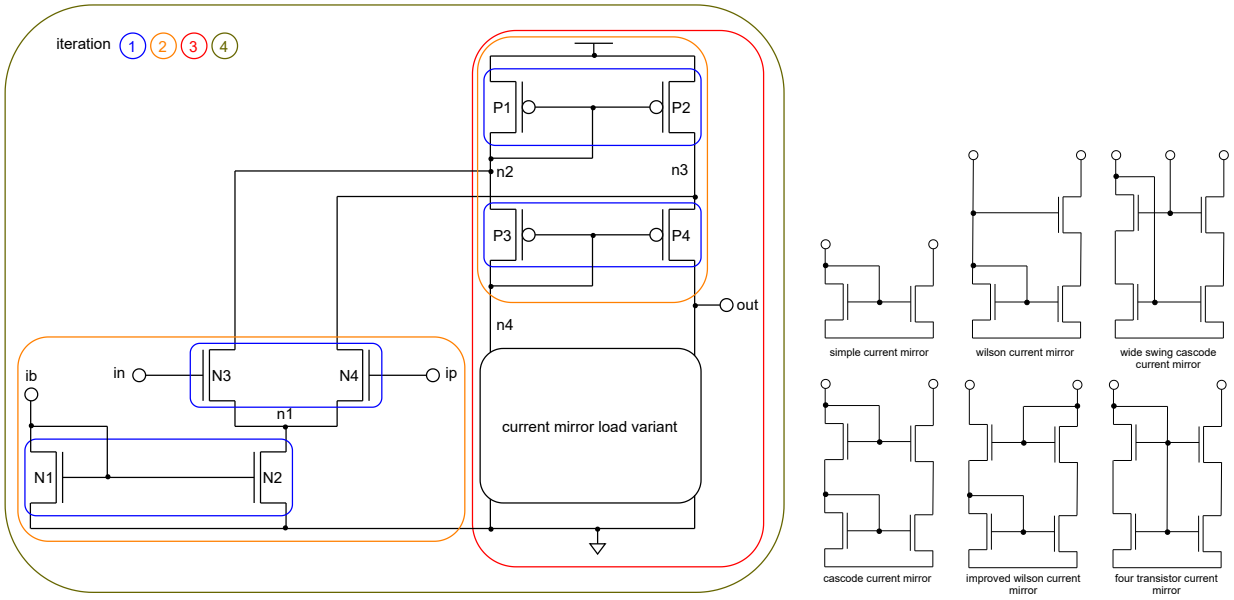
Algorithm 8 iterates over the tentative building blocks according to decreasing group strengths. In the first iteration, the recognition rules for $(N1, N2)$ are generated and the blocks $(N2, N3)$, $(N2, N4)$ are removed from B_w as they overlap with $(N1, N2)$ while having a lower group strength. Similarly, all other blocks except $(N3, N4)$ are discarded as they overlap with $(P1, P2)$.

In the second iteration, the recognition rules for $[(N3, N4), (P1, P2)]$ are generated as that block has a higher group strength than $[(N1, N2), (N3, N4)]$.

Finally, $\{[(N3, N4), (P1, P2)], (N1, N2)\}$ are combined to the new library element which corresponds to the building block description of the differential input stage.

5.4. Library Extension

Algorithm 9 iteratively inserts the newly created elements $e_{new} = (b_i, b_j) \in E_{new}$ into the building block library L_C. It first determines the rank of the new building block by incrementing the maximum rank of (b_i, b_j) by one (line 3). The algorithm afterwards creates a new library rank for (b_i, b_j) if required (line 5) and inserts it into its corresponding rank (line 7).



(a) folded cascode amplifier (b) load variants

Figure 5.5.: Schematic of a folded cascode amplifier with different current mirror loads.

Algorithm 7 creates two new library elements for the differential input stage from Fig. 5.2 in its first iteration. As the initial pair library was empty, a new rank is initialized and the two elements are inserted into it. In the second and third iteration, one new library element each has been inserted into a new rank of L_C by Algorithm 9. The final building block library consists of three ranks with six library elements in total as shown in Fig. 5.3.

5.5. Experimental Results

This section presents experimental results for the library-free structure recognition. First, a folded cascode amplifier (FOCA) with different current mirror loads is investigated. It will be shown that the analog basic building block library presented by [4] can be created without prior knowledge of typical analog circuit elements, e.g., current mirrors or differential pairs.

Afterwards, a new library dedicated to identify level shifters is presented. Level shifters are interface circuits which convert signals between different voltage domains of a chip. These circuits are susceptible to ESD damage and need suitable protection [43].

The generated library elements are manually labeled according to their functionality in the circuit afterwards, indicating that the implemented algorithms are able to capture the functional elements of the circuit based on the three simple neighboring characteristics presented in Sec. 5.2.

5.5.1. Folded Cascode Amplifier Variants

Fig. 5.5a shows the schematic of a folded cascode amplifier. Its load is implemented by one of the current mirror variants shown in Fig. 5.5b.

In a first experiment, library free structure recognition is performed on the folded cascode amplifier with a cascode current mirror load. The initially provided pair library is empty, i.e., the pair library required to identify this folded cascode amplifier variant is built from scratch. The array library contains the array definitions for resistor, capacitor, normal transistor and diode transistor arrays. The result of the method corresponds to the basic version of the folded cascode amplifier building block library which is then subsequently extended by the library elements required to identify the other variants with various different current mirror loads.

Algorithms 7-9 group the devices of the amplifier in the first iteration to the following building block pairs:

- $(N1, N2), (P1, P2)$ and the devices in the load form a simple current mirror,
- $(P3, P4)$ build a bias shifter,
- $(N3, N4)$ are grouped to a differential pair.

These blocks are highlighted in blue in Fig. 5.5.

In the second iteration, highlighted in orange, $[(N1, N2), (N3, N4)]$ are assembled to a differential input stage and $[(P1, P2), (P3, P4)]$ to a cascode current mirror.

The algorithm combines that cascode current mirror with the current mirror load in the third iteration to the output stage of the amplifier as shown in red. The tentative pair formed by the differential input stage $ds(N1{-}N4)$ and the PMOS cascode current mirror $ccm(P1{-}P4)$ as well as the tentative pair by $ccm(P1{-}P4)$ and the current mirror load variant have the same group strength. Hence, a tie breaker rule has to be applied to decide which of the two pairs should be kept. In this case the latter pair remains in the computation process as both of its subblocks are connected to the output pin *out* of the amplifier which indicates that both blocks belong to the output stage of the amplifier.

In the fourth and last iteration of the algorithm, the input and the output stage are combined to form the folded cascode amplifier.

The generated building blocks are shown in the building library from Fig. 5.6 in the colors of the iteration they have been created. The remaining elements of that library are required to identify the other five folded cascode amplifier variants from Fig. 5.6. These elements are generated by executing the presented algorithms for each variant once more with the basic version of the folded cascode amplifier library as input. In this way, the missing elements are successively added to the library as shown in Fig. 5.6. The final library consists of five ranks and 29 building blocks.

The generated library contains furthermore all analog basic building blocks of the library provided by [4]. Hence, the new method was able to "learn" the most common analog basic building blocks by analyzing the given amplifiers using the three simple neighboring characteristics *connectivity, substrate type* and *rank difference*.

Finally, it has been shown that the automatic library generation overcomes the main limitation of structure recognition: the scope, complexity and completeness of the provided building block library. The library from Fig. 5.6 is able to identify six folded cascode amplifier variants and can easily be extended by other circuit topologies.

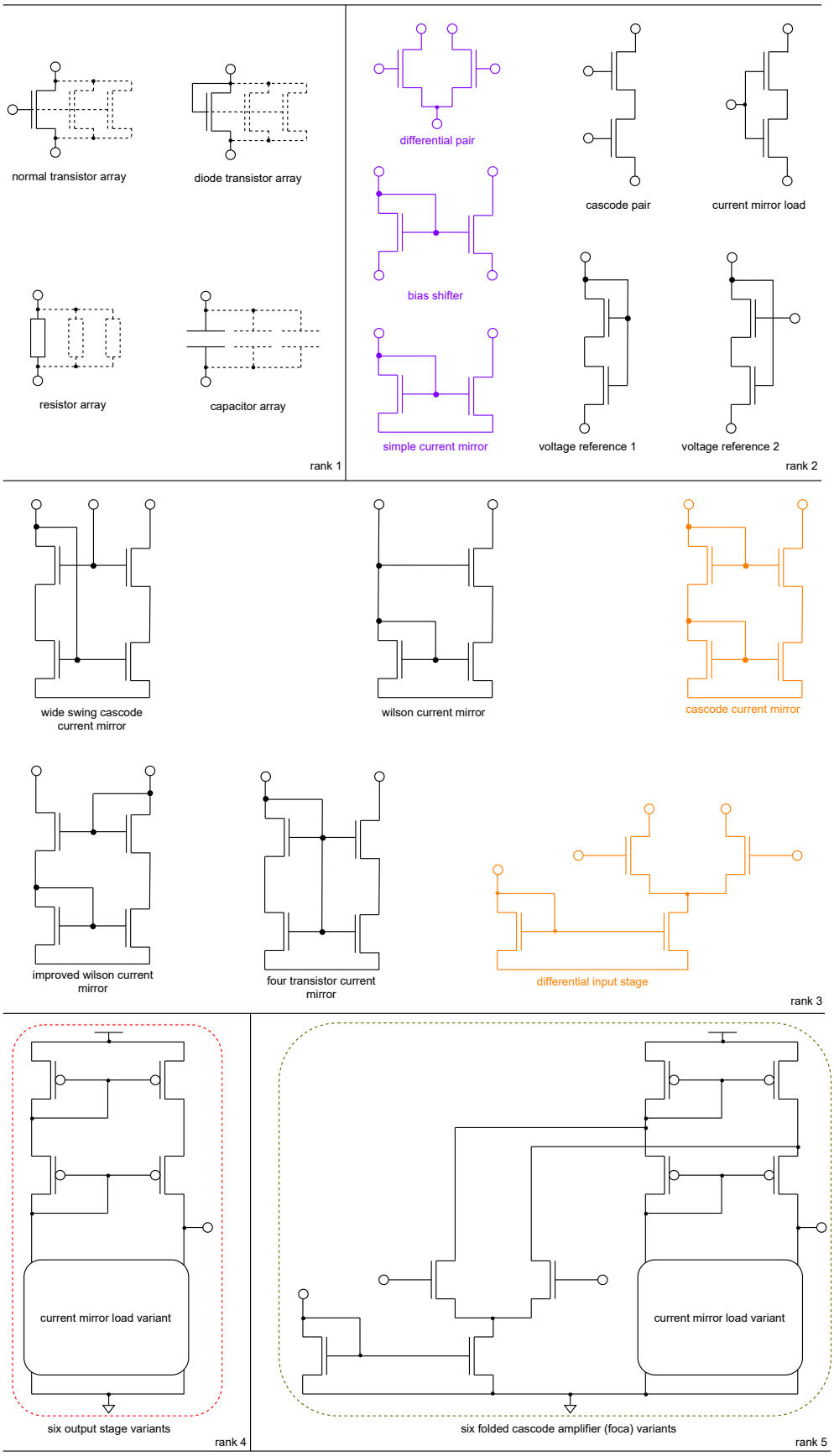


Figure 5.6.: Generated building block library for the six folded cascode amplifier variants from Fig. 5.6.

5.5.2. Level Shifter Topologies

Level shifter circuits are interface circuits between different voltage domains of a chip: they convert signals from lower voltage to higher voltage domains, e.g., from the digital to the analog domain of an integrated circuit and vice versa. Such interface circuits are susceptible to ESD events and must be protected by suitable protection circuitry [43].

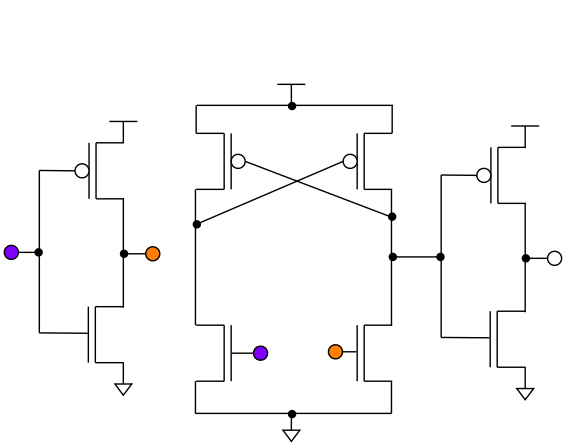
Structure recognition can help to verify the functionality of an ESD protection network by its capability to reliably identify interface circuits in large designs. However, structure recognition requires a building block library containing all used level shifter topologies in a given design. Library-free structure recognition can automatically generate this library.

Figs. 5.7 shows five different level shifter topologies. The basic elements of the differential cascade voltage switch [47] and the pass gate level converter [48] have been used to design more sophisticated topologies, e.g., the pass gate level converter with keeper device [48], the single supply level converter [49] and the even more elaborate single supply true voltage level shifter [50]. These topologies either improve the speed, reduce the power consumption or lead to less routing congestion compared to the conventional differential cascade voltage switch and pass gate level converter topologies.

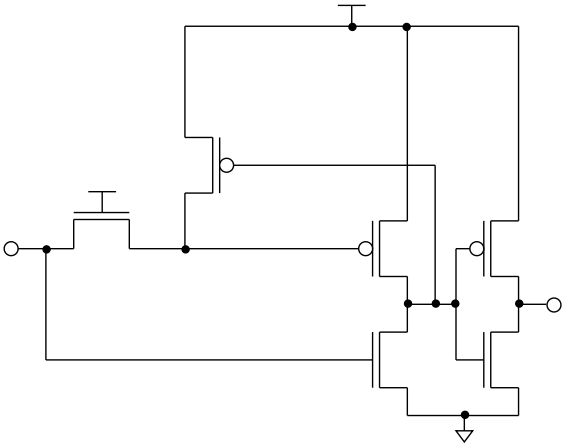
In the following, the method presented in Sec. 5.2 is used to generate a building block library for structure recognition which is dedicated to level shifter topologies. Starting with a library containing only array definitions, the algorithm first generates the building block description of the differential cascade voltage switch. In a second run, the elements required to identify the pass gate level converter are added to the library. The library is then subsequently extended to include the building block descriptions of the pass gate level converter with keeper device, the single supply level converter and finally by the single supply true voltage level shifter.

The final generated building block library is shown in Fig. 5.8. It consists of six ranks and 24 elements. The building blocks marked with bold letters correspond to one of the circuit topologies shown in Fig. 5.7. The remaining library elements have been labelled according to their functionality in the corresponding circuit. E.g., rank four contains the input stage of the single supply true voltage level converter, rank two a NOR-Gate and the input and output stages of the analyzed level shifter topologies which are further split up into smaller input and output circuitry on rank one. Hence, all of the generated library elements fulfil a specific functionality in the corresponding circuit, i.e., the given circuit topologies have been decomposed into meaningful building blocks by the method presented in Sec. 5.2.

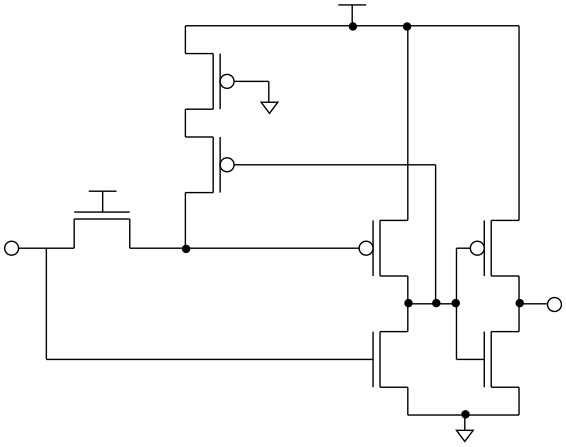
The resulting library can easily be extended by other level shifter topologies when required.



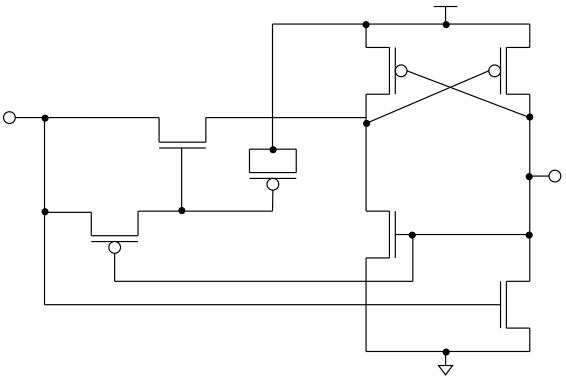
(a) differential cascade voltage switch (dcvs)



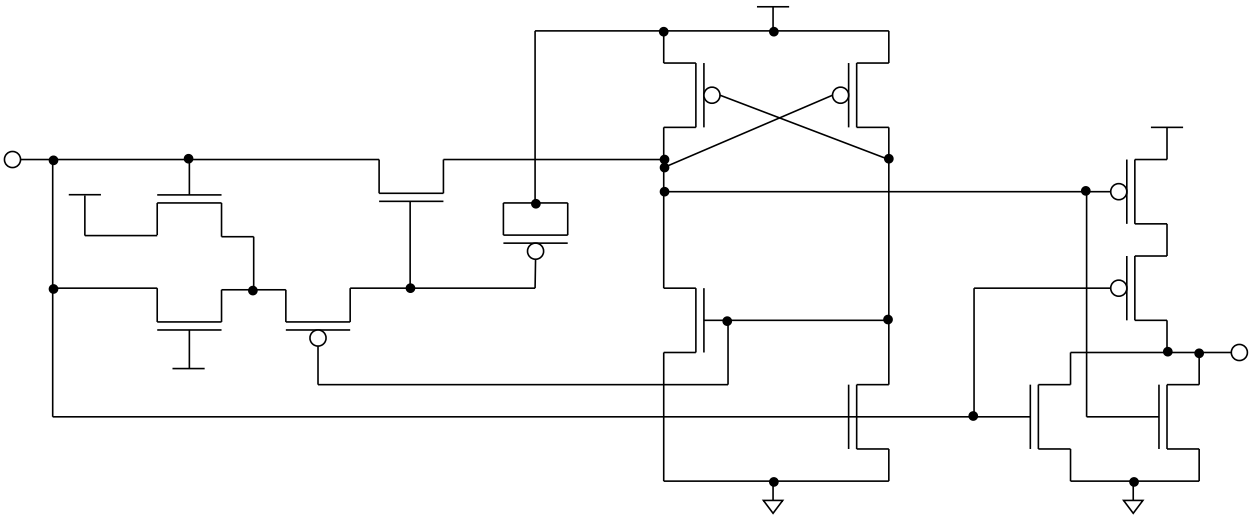
(b) pass gate level converter (passGateLC)



(c) passGateLC with keeper device (passGateLCKeeper)



(d) single supply level converter (singleSupplyLC)



(e) single supply true voltage level converter (singleSupplyTVVS)

Figure 5.7.: Five different level shifter topologies [47; 48; 49; 50].

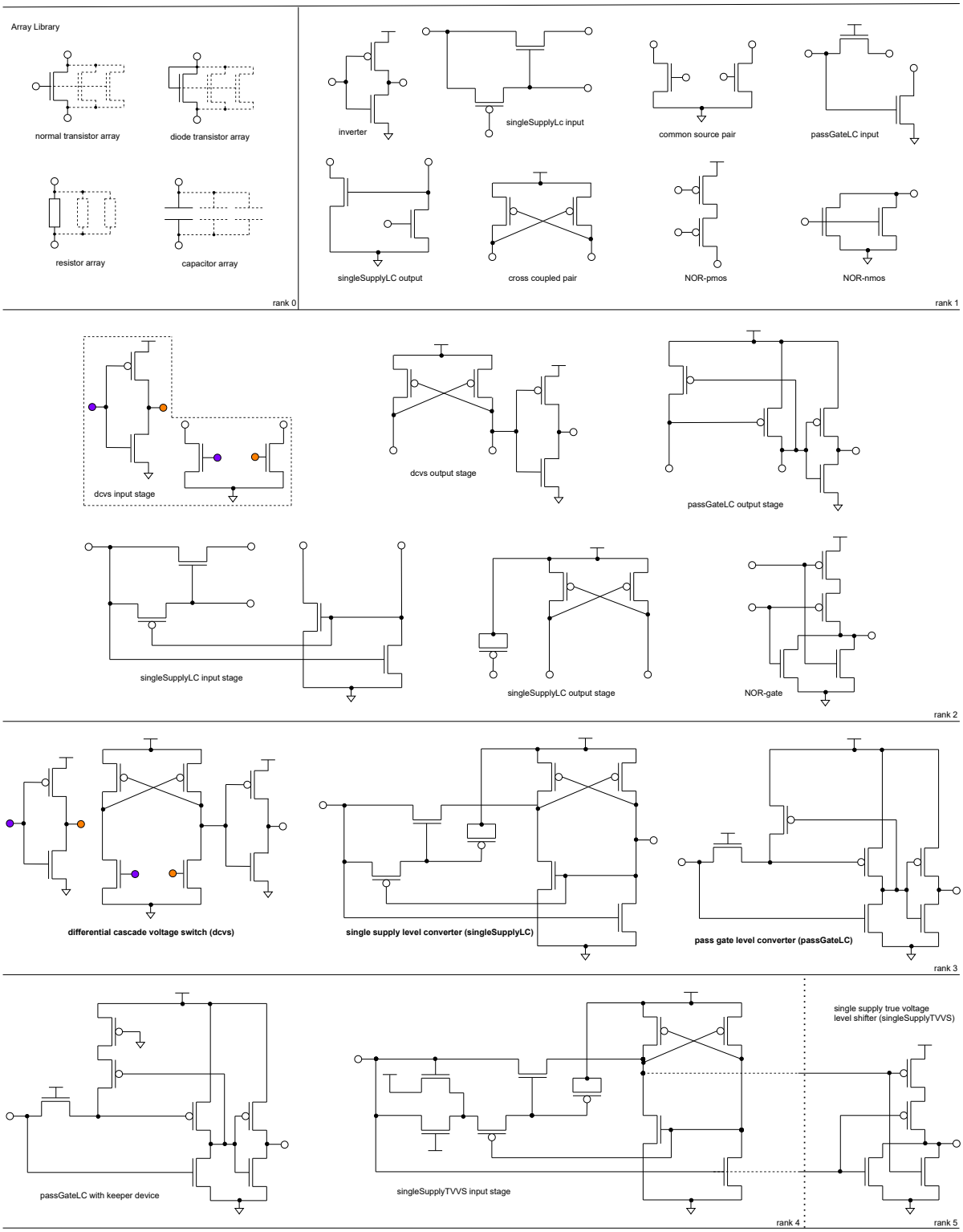


Figure 5.8.: Level converter library generated by Algorithms 7 - 9.

6. Conclusion

The design and verification of analog circuits is still mainly done manually in industrial practice. Designs are hierarchically partitioned into smaller parts until individual subblocks, e.g. operational amplifiers, are assigned to an analog circuit designer. The sized analog blocks are then combined to form the overall system. Extensive evaluation and tweaking of the subblocks is thereby necessary to make the overall system functional.

This problem aggravates if analog blocks are equipped with additional power-down circuitry: the correct power-down mode functionality of the overall system cannot be guaranteed even though it has been correctly implemented for each individual subblocks of the system.

This work presented two new methods which automatically verify or synthesize the power-down mode of hierarchical circuits. The methods exploit the hierarchical structure of a given design by reusing intermediate results whenever possible. Experimental results have been presented to demonstrate the proposed methods.

The presented power-down verification and synthesis methods use structure recognition and symmetry computation procedures to extract matching constraints of a given circuit. These traditional circuit analysis methods have been adapted to hierarchical analog designs.

Structure recognition is mainly limited by the provided basic building block library. A new approach has been presented which automatically generates the building block description of a given circuit which can be used by structure recognition. The approach partitions the given circuit into its building blocks by analyzing three neighboring characteristics of its devices. It has been shown that the new approach captures the same analog basic building blocks as previous predefined libraries by analyzing operational amplifiers. It also successfully partitioned level shifter circuits into their input and output circuitry which enables the reliable identification of these ESD susceptible structures during ESD verification.

A. Hierarchical Structure Recognition

A.1. Introduction

Analog circuits are composed of so-called "basic building blocks", e.g., current mirrors or differential pairs. These blocks implement a specific functionality in the circuit, e.g., the differential pair $dp(N2, N3)$ of the folded cascode amplifier from Fig. A.2 converts a differential input voltage at its input pins in, ip into a differential output current at its nets $n2$, $n3$. This differential current is then amplified and mirrored to the output of the amplifier by the cascode current mirror $ccm(P1 - P4)$ and the wide swing cascode current mirror $wsccm(N4 - N7)$. The shown folded cascode amplifier implements the subcircuits $FOCA1$ and $FOCA2$ of the core circuitry of the dual gain amplifier from Fig. A.1b. $FOCA1$ and $FOCA2$ are biased by the circuit shown in Fig. A.1a. The transistors $N1$, $N2$ each form a simple current mirror $scm(N<1,2>, FOCA<1,2>/N1)$ with transistor $N1$ of the folded cascode amplifier to bias the differential input stage of $FOCA1$ and $FOCA2$. Similarly, $N3$ and $N5$ form level shifters $ls(N<3,5>, FOCA<1,2>/N<6,7>)$ with transistors $N6$ and $N7$ of $FOCA1$ and $FOCA2$ to set the bias voltage of their wide swing cascode current mirrors.

Analog basic building blocks require specific operating conditions to fulfill their functionality, e.g., the transistors of a simple current mirror of differential pair must operate in saturation region to mitigate the influence of channel modulation effects [4].

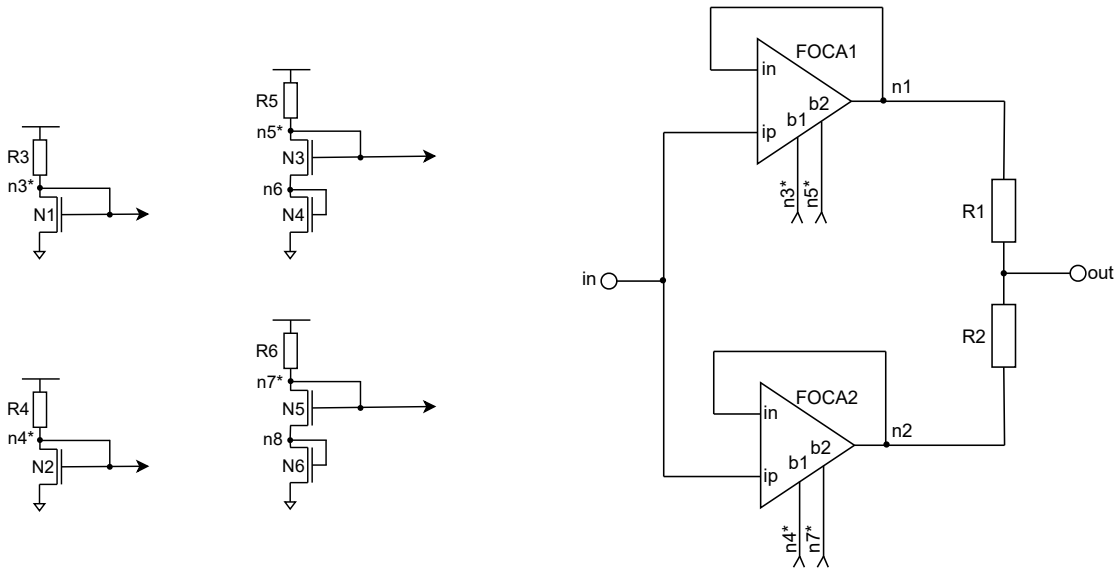
Apart from circuit sizing, this structural information can also be used for circuit verification and synthesis tasks. The power-down verification and synthesis method from Chaps. 3 and 4 consider matching conditions at analog basic building blocks during their execution.

An experienced designer can easily identify analog basic building blocks from a circuit schematic by inspection and use it for sizing or verification tasks. For computers, specialized software is required to extract this information from a schematic, generally known as *structure recognition*.

A.2. State of the Art and Contributions

The hierarchical structure recognition method implemented in this work is based on the "sizing rules method" which was originally published in [16]. The presented algorithms have been further refined in [4; 51; 10]. All versions require a predefined building block library L_{analog} as input. This library is shown in Fig. A.3.

The library is partitioned into an array library L_{array} and a pair library L_{pair}. Each element of L_{array} and L_{pair} defines the recognition rules to identify the corresponding structure in a circuit.



(a) bias circuitry (b) core circuit

Figure A.1.: Dual gain amplifier from Fig. 3.6 without power-down circuitry.

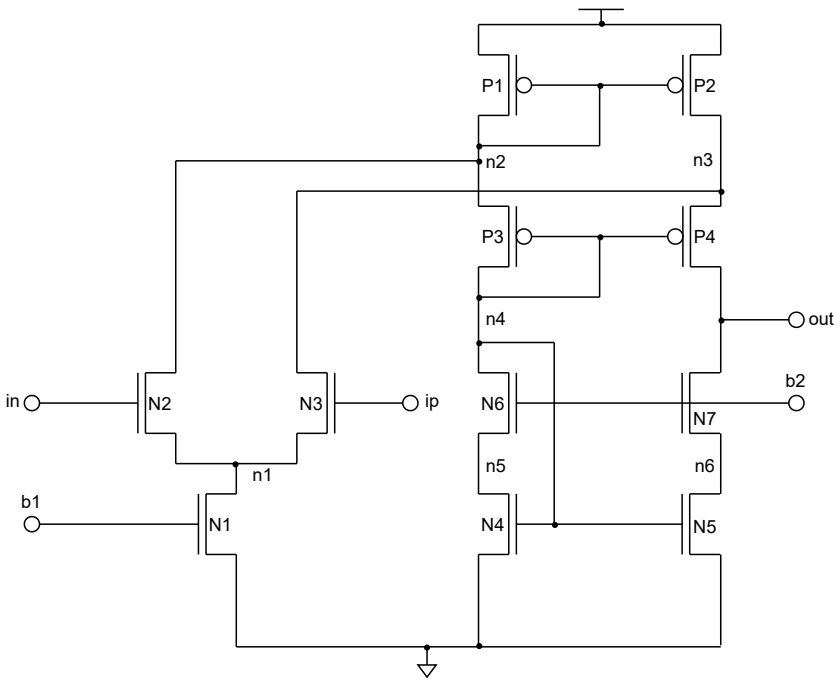


Figure A.2.: Schematic of a folded cascode amplifier which implements subcircuits $FOCA1$ and $FOCA2$ of the dual gain amplifier from Fig. A.1.

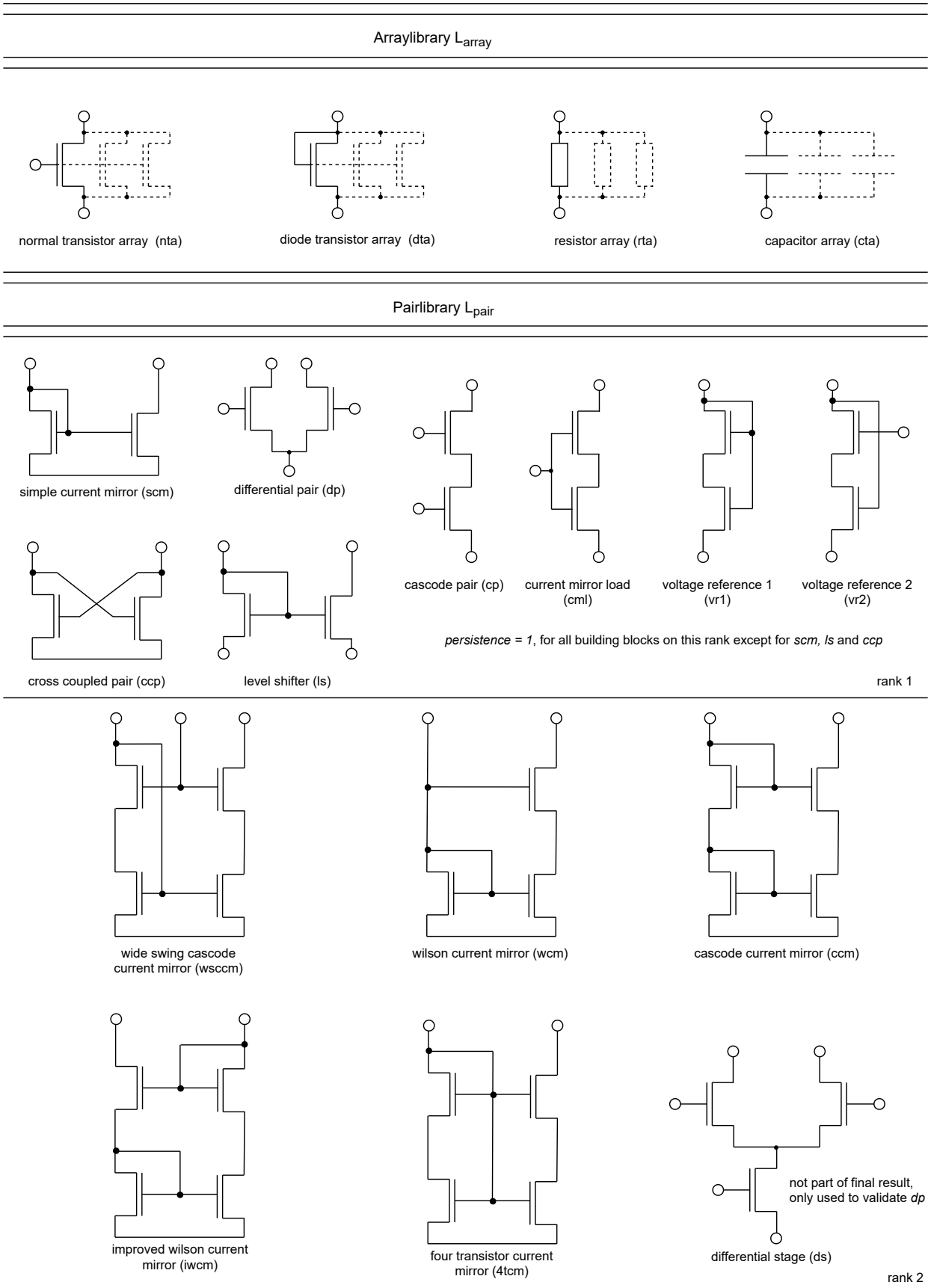


Figure A.3.: Analog basic building block library $L_{analog} = L_{array} \cup L_{pair}$ based on [4; 51; 10].

A device array consists of one or more parallelly connected devices which exhibit the same electrical behavior as a device of their combined sizes. A normal transistor array (*nta*), e.g., consists of a set of NMOS- or PMOS-transistors whose drain, gate and source pins are connected in parallel by three different nets. Combining the devices of a circuit to arrays speeds up the recognition process of analog building blocks as fewer devices need to be considered in the later iterations of the algorithm [17]. The recognition rules and an algorithm to efficiently identify arrays in a circuit is presented in [51], Sec. 3.2.2. This algorithm has been re-implemented in this work without any changes.

The pair library L_{pair} is organized in ranks: a building block *pair* of rank i consists of two building blocks from the lower ranks $i-1, ..., 0$ with the array library being considered as rank zero. The pair library can have an arbitrary number of ranks. A level shifter of rank one, e.g., consists of a normal and a diode transistor array, a Wilson current mirror of a simple current mirror and a normal transistor array.

The sizing rules method iteratively scans the circuit for the provided library elements by starting at the lowest and ending at the highest rank. Ambiguities during the recognition process are resolved by dominance graphs in [4; 51]. Ambiguities occur when an identified array or pair could be assigned to two different higher ranked library elements, e.g., a normal transistor array could be assigned to a differential pair and a simple current mirror at the same time. The relations defined by the dominance graph decide in such cases which of the identified pairs should be kept in the recognition process. The other pairs are then discarded. In above example, the simple current mirror would be kept and the differential pair discarded as the devices of a simple current mirror are tightly connected and hence unambiguously identifiable.

This mechanism has been simplified by [10] by assigning an expiration date, a so-called "persistence", to each element of the pair library L_{pair}. The persistence defines for how many ranks an identified pair should be kept in the iterative recognition process. A pair is discarded if the currently processed rank minus the rank of the pair exceeds its persistence. E.g., a cascode pair of rank one has a persistence of one. All cascode pairs which have not become part of a wide swing cascode current mirror are discarded after all elements of rank two have been identified in the circuit.

The sizing rules method by [10] supports hierarchical circuits by circuit flattening. The circuit hierarchy is not maintained rendering it unsuitable for industrial and IP based designs. Intermediate results cannot be reused to speed up the structure recognition algorithm.

Another approach to support hierarchical designs based on the sizing rules method is presented in [23]. It traverses the circuit hierarchy recursively, flattens it partially which makes required information for structure recognition available in-between the different hierarchy levels of the circuit. This method however does still not reuse intermediate results.

The work by [52] simplifies structure recognition by encoding the elements of the analog basic building block library based on their connectivity. The circuit is then scanned for pairs with the same encoding as the library elements. Hierarchical circuits are flattened at lower hierarchy levels; the subcircuits at higher levels are separated from each other and structure recognition is performed independently for each them. However, it is not stated how this partitioning is done.

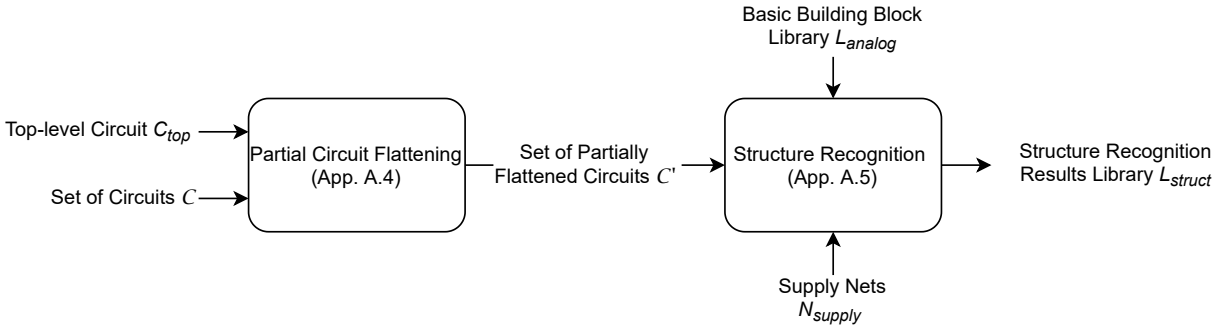


Figure A.4.: Overview of the hierarchical structure recognition method.

In this work, structure recognition has been modified to support hierarchical circuits by partial circuit flattening (Fig. 1.8c). Similar to [23], this approach first makes connectivity information required for structure recognition available in-between the different circuit hierarchy levels. This step decouples the hierarchy levels from each other which would, in comparison to [23], allow the parallelization of the recognition process. However, this parallelization has not been implemented yet. The decoupling of the hierarchy levels furthermore splits up the whole recognition process into smaller sub-problems and has to be executed only once for each circuit type in the partially flattened design which hence reduces the computational complexity of the overall problem.

The algorithms and procedures of the sizing rules method have been further enhanced as follows: the state-of-the-art method [10] merges the recognition results of each rank into one data structure. It will be shown that organizing the results in ranks will further simplify and decrease the runtime of the pair recognition and ambiguity resolution procedures.

A.3. Overview

Fig. A.4 gives an overview of the new hierarchical structure recognition method. It takes a top-level circuit C_{top} and the set $\mathcal{C}$ which contains all circuit definitions instantiated by subcircuits of C_{top} as input.

The circuit is *partially* flattened (App. A.4) by recursively propagating connectivity information between the hierarchy levels similarly to [23]. Structure recognition however is not performed yet. The resulting set of partially flattened circuits $\mathcal{C}'$ is input to the actual structure recognition method (App. A.5).

The hierarchy levels are now decoupled from each other. Hence, structure recognition only has to iterate over each element of $\mathcal{C}'$ instead of recursively descending into the hierarchy. This enables the parallelization of the method for hierarchical designs. However, parallel execution was not implemented in this work. Each circuit type is inherently processed only once, the circuit hierarchy is maintained and the recognition results can be easily back annotated to the original circuit.

The output of the algorithm is the structure recognition results library L_{struct} which contains the identified building blocks of each circuit type t'_i.

Algorithm 10 Partial circuit flattening.

```
1: procedure partialCircuitFlattening(C_top(P_top, N_top, C_sub,top, t_top), C)
2:     C' = C
3:     C'_top = C_top(P_top, N_top, C_sub,top, t_top)
4:     recursivePartialCircuitFlattening(C'_top)    // Modifies C' inherently
5:     return C'
6: end procedure
```

Algorithm 11 Recursive partial circuit flattening.

```
1: procedure recursivePartialCircuitFlattening(C_i(P_i, N_i, C_sub,i, t_i))
2:     for all c_sub ∈ C_sub,i do
3:         if isSubCircuitBlock(c_sub) then
4:             partialCircuitFlattening(c_sub)    // Recursive call
5:             propagateConnectivity(C_i, c_sub)    // See Algorithm 12
6:         end if
7:     end for
8: end procedure
```

A.4. Partial Circuit Flattening

The devices forming a basic building block can be distributed between several hierarchy levels for hierarchical designs. E.g., the transistors $N1$ and $N2$ of the dual gain amplifier from Fig. A.1 each form a simple current mirror with the transistor $N1$ of the folded cascode amplifiers $FOCA1$ and $FOCA2$ from Fig. A.2.

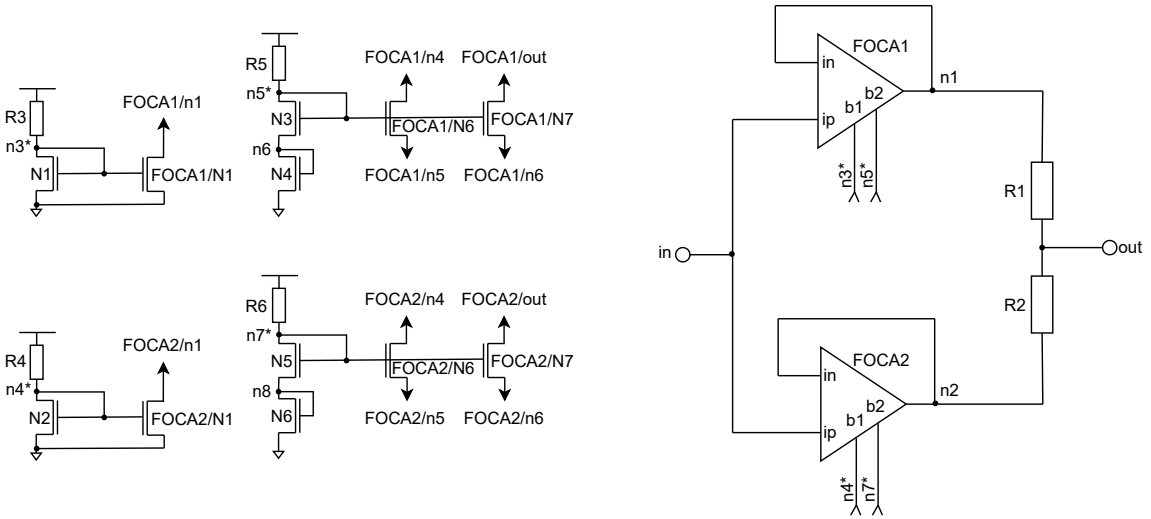
This connectivity information must be propagated between the two hierarchy levels such that structure recognition is able to identify these two simple current mirrors.

Algorithms 10 to 12 flatten a given circuit hierarchy partially by augmenting each hierarchy level with the devices connected to its respective external pins. Algorithm 10 initializes C'_{top} and $\mathcal{C}'$ as copies of C_{top} and $\mathcal{C}$ in lines 2 and 3. Afterwards, it calls Algorithm 11 in line 4 to flatten the circuit hierarchy partially in a recursive manner.

Algorithm 11 first descends to the bottom level of the hierarchy by calling itself recursively for each subcircuit of C_i in lines 2 and 4. There, in line 5, it calls Algorithm 12 which makes the connectivity information between the circuit C_{high} and its subcircuit C_{low} available as follows: it determines a set of devices $C_{con,high} \subseteq C_{sub,high}$ which are externally connected to the pins P_{low} of the given subcircuit (line 2). Furthermore, the algorithm determines a set

Algorithm 12 Sharing connectivity information between two hierarchy levels.

```
1: procedure propagateConnectivity(C_high, C_low)
       // C_high = C_high(P_high, N_high, C_sub,high, t_high)
       // C_low = C_low(P_low, N_low, C_sub,low, t_low)
2:     C_con,high = findHigherConnectedDevices(P_low, C_high)    // C_con,high ⊆ C_sub,high
3:     C_con,low = findLowerConnectedDevices(P_low, C_low)    // C_con,low ⊆ C_sub,low
4:     C_sub,low = C_sub,low ∪ C_con,high
5:     C_sub,high = C_sub,high ∪ C_con,low
6: end procedure
```



(a) bias circuitry (b) core circuit

Figure A.5.: Partially flattened dual gain amplifier from Fig. A.1.

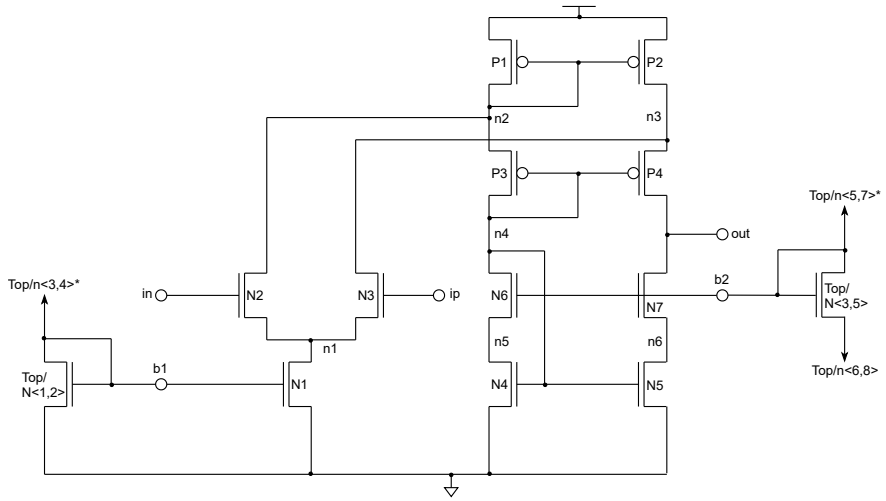


Figure A.6.: Partially flattened folded cascode amplifier from Fig. A.2.

of devices $C_{con,low} \subseteq C_{sub,low}$ which are internally connected to the pins P_{low} (line 3). Then, in lines 4 and 5, the higher levels devices $C_{con,high}$ are inserted into the set of subcircuits of C_{low} and the lower-level devices $C_{con,low}$ into $C_{sub,high}$. The connectivity information is propagated in both directions in order to keep the affected hierarchy levels synchronized.

After processing the device level, Algorithm 12 ascends the hierarchy again, repeating the steps described above for every subcircuit in the currently investigated hierarchy level. The algorithms directly operate on the circuits contained by $\mathcal{C}'$. Hence, all results are automatically available in $\mathcal{C}'$ which is the return value of Algorithm 10 in line 5.

Figs. A.5 and A.6 show the dual gain amplifier and the folded cascode amplifier from Figs. A.5 and A.6 after partial circuit flattening. The transistors $N1$ and $N2$ of the dual gain amplifier are connected to transistor $N1$ inside the folded cascode amplifiers $FOCA1$ and $FOCA2$, respectively. Furthermore, transistors $N3$ and $N5$ on the top-level are connected to $N6$ and $N7$ of $FOCA1$ and $FOCA2$, respectively. Hence, this connectivity information has been

Algorithm 13 Structure recognition for hierarchical analog circuits.

```
1: procedure HIERARCHICALSTRUCTURERECOGNITION(C', L_analog, N_supply)
2:     L_struct = ∅
3:     for all C'_i(P'_i, N'_i, C'_sub,i, t'_i) ∈ C' do    // Prallel execution possible.
4:         C_B,t'_i = structureRecognition(C'_i, L_analog, N_supply)    // See Algorithm 14
5:         L_struct = L_struct ∪ (t'_i, C_B,t'_i)
6:     end for
7:     return L_struct
8: end procedure
```

made available to both hierarchy levels. Please note: the resistors of the dual gain amplifier which are also connected to pins $b1$ and $b2$ of the folded cascode amplifiers have not been inserted into the corresponding lower-level schematics as analog basic building blocks are formed by MOSFET or bipolar transistors only. Hence, this connectivity information is not relevant for structure recognition and does not need to be propagated in-between hierarchy levels.

A.5. Structure Recognition

Algorithm 13 outlines the new hierarchical structure recognition method. It takes the set of partially flattened circuits $\mathcal{C}'$ from the preprocessing step, the basic building block library L_{analog} from Fig. A.3 and the supply nets of the circuits N_{supply} as inputs.

In line 2, it first initializes the results library L_{struct} empty. Then, in line 3, the algorithm simply iterates over all elements of the set $\mathcal{C}'$ and calls Algorithm 14, the main structure recognition procedure which is dedicated to *flat* circuits (line 4). This is only possible because the hierarchy levels have been decoupled from each other by partial circuit flattening. The flat algorithm ignores all subcircuit blocks of C'_i as all required structural information of other hierarchy levels connected to C'_i has been made available to it during partial circuit flattening. The results of the flat structure recognition method are inserted into L_{struct} in line 5.

The final output of Algorithm 13 is the structure recognition results library L_{struct} which contains all identified building blocks of each circuit of type t'_i (line 7).

Algorithm 14 shows the enhanced structure recognition method based on [10]. It takes a circuit C_i, the array and pair libraries, L_{array} and L_{pair}, and the supply nets N_{supply} as inputs.

The structure recognition results for C_i are organized in ranks. The set $C_{B,i}$ contains one building block circuit C_{B,i,r_j} for each rank of the provided building block library L_{analog}. A building block circuit C_{B,i,r_j} contains all identified building blocks of rank j of circuit C_i plus their connectivity. This representation reduces the complexity of the pair recognition algorithm by [17] and simplifies the arbitration mechanisms introduced by [10] which will be explained later in this section.

The structure recognition algorithm first initializes the set of building block circuits $C_{B,i}$ empty (line 2). Then, the arrays of circuit are identified in lines 3 - 7, using the array recognition from [17]. The resulting array circuit C_{B,i,r_0} is stored in $C_{B,i}$.

Algorithm 14 Enhanced analog structure recognition algorithm based on [10].

1: **procedure** STRUCTURERECOGNITION($C_i, L_{array}, L_{pair}, N_{supply}$)
2: $C_{B,i} = \emptyset$ // Set of building block circuits, organized in ranks.

// Recognition of arrays
3: $C_{B,i,r_0} = \emptyset$ // Array circuit initialized empty.
4: **for all** $array \in L_{array}$ **do**
5: $C_{B,i,r_0} = C_{B,i,r_0} \cup arrayRecognition(array, C_i)$ // Similar to [17], Sec. 3.2.1.
6: **end for**
7: $C_{B,i} = C_{B,i} \cup C_{B,i,r_0}$

// Recognition of pairs
8: **for all** $r_j \in L_{pair}$ **do** // Iterate over all ranks.
9: $C_{B,i,r_j} = \emptyset$
10: **for all** $pair \in r_j$ **do** // Iterate over all pairs of the current rank.
11: $C_{B,i,r_j} = C_{B,i,r_j} \cup pairRecognition(pair, C_{B,i})$ // See Alg. 15 and 16.
12: **end for**
13: $C_{B,i} = C_{B,i} \cup C_{B,i,r_j}$

// Arbitration simplified compared to [10]
14: $selfArbitration(C_{B,i})$ // Algorithm 18
15: $dominanceRelations(C_{B,i})$ // Algorithm 19
16: **end for**

17: **return** $C_{B,i}$
18: **end procedure**

Algorithm 15 Pair recognition by [17].

1: **procedure** PAIRRECOGNITION($p, C_i, \mathcal{B}$)
2:
3: $BB = \emptyset,\ TBB = \emptyset$
4: **for all** $n \in N_i$ **do**
5: $TBB_1 = findBlocks(p_{c,1}, n, \mathcal{B})$
6: $TBB_2 = findBlocks(p_{c,2}, n, \mathcal{B})$
7: $TBB = TBB \cup (TBB_1 \times TBB_2)$
8: **end for**
9: **for all** $tbb \in TBB$ **do**
10: **if** $rulesFulfilled(tbb, p)$ **then**
11: $BB = BB \cup \{tbb\}$
12: **end if**
13: **end for**
14: **return** BB
15: **end procedure**

Algorithm 16 Enhanced pair recognition.

procedure PAIRRECOGNITION(p, C_B)
$C_{B,r_{c,1}}, C_{B,r_{c,2}} = find(C_B, p)$
$BB = \emptyset,\ TBB = \emptyset$
for all $n \in N_{r_{c,1}}$ **do**
$TBB_1 = findBlocks(p_{c,1}, n, C_{B,r_{c,1}})$
$TBB_2 = findBlocks(p_{c,2}, n, C_{B,r_{c,2}})$
$TBB = TBB \cup (TBB_1 \times TBB_2)$
end for
for all $tbb \in TBB$ **do**
if $rulesFulfilled(tbb, p)$ **then**
$BB = BB \cup \{tbb\}$
end if
end for
return BB
end procedure

In the next step, the algorithm iterates over each rank r_j of the pair library L_{pair} in ascending order (line 8), initializing a new building block circuit C_{B,i,r_j} for each rank (line 9). The algorithm then iterates over each pair of that rank, identifies and inserts the detected pairs into C_{B,i,r_j} which in turn is inserted into $C_{B,i}$ after all library elements of that rank have been identified (lines 10, 11 and 13). Ambiguities encountered during the recognition process are resolved afterwards in lines 14 and 15 by applying a self-arbitration mechanism and dominance relations for the recognition results of the current rank. Finally, in line 17, Algorithm 14 returns the building block circuit $C_{B,i}$ which contains all analog structures identified in the given circuit C_i.

Pairs are identified by a set of recognition rules [17]. These can be summarized as follows:

- *block type rules:* each subblock of a pair must have a specific building block type, e.g., the type of the subblocks of a cascode current mirror must be "level shifter" and "simple current mirror".
- *substrate type rules:* each subblock must have a specific substrate type, e.g., a differential pair always consists of two MOSFETs with the same substrate type, i.e., two NMOS or two PMOS transistors. Analog building blocks are typically formed by two subblocks of the same substrate type.
- *connection rules:* define which pins of two subblocks have and do not have to be connected with each other via a net. E.g., the gate pin of a normal transistor array has to be connected to the drain pin of a diode transistor array to form a simple current mirror. Furthermore, their sources have to be connected and any other connections between the two arrays are not allowed.
- *supply net rules:* define which pins of the two subblocks have and do not have to be connected to a supply or ground net of the circuit. E.g., a differential is not allowed to be connected to the ground or supply net of a circuit. Additionally, rules for specific supply rails, e.g., the positive supply or ground rail can be specified.

An algorithm to efficiently identify pairs in a circuit using the first two rules is published in [17], Sec. 3.1.1. This algorithm has been further refined in this work as explained in the

Algorithm 17 Self-arbitration by [10].

```
1: procedure selfArbitration(B)
2:     k = maxRank(B)
3:     repeat
4:         removed = 0
5:         for all b ∈ B do
6:             if not hasParent(b)
7:                and expired(b, k) then
8:                 C_B = C_B \ {b}
9:                 removed = 1
10:            end if
11:        end for
12:    until removed = 0
13: end procedure
```

Algorithm 18 Enhanced self-arbitration.

```
1: procedure pairRecognition(C_B)
2:     k = maxRank(C_B)
3:     for all i = k − 1 to i = 1 do
4:         for all b ∈ C_{B,r_i} do
5:             if not hasParent(b)
6:                and expired(b, k) then
7:                 C_B = C_B \ {b}
8:             end if
9:         end for
10:    end for
11: end procedure
```

following.

Algorithms 15 and 16 compare the two versions of the pair recognition algorithm. Both algorithms iterate over a set of nets (line 4), try to find all occurrences of the arrays or pairs $p_{c,1}$ and $p_{c,2}$ which form the currently investigated pair p at these nets (lines 5 and 6), build the Cartesian product TBB of the found structures (line 7) and check whether all other recognition rules are fulfilled for the created tentative building block tuples (lines 9 - 13). Finally, in line 14, the algorithms return all building block pairs BB fulfilling all recognition rules provided by p.

The differences between the two versions lie in the set of nets the two algorithms are iterating over and in the sets in which they are searching for the subblocks $p_{c,1}$ and $p_{c,2}$. The previous version iterates over *all* nets $n \in N_i$ of the given circuit C_i (line 4, Algorithm 15). The enhanced version only iterates over the nets $n \in N_{r_{c,1}}$ of the building block circuit $C_{B,r_{c,1}}$ (line 4, Algorithm 16) which is determined in line 2 of the enhanced pair recognition algorithm. $C_{B,r_{c,1}}$ thereby corresponds to the building block circuit of rank $r_{c,1}$ on which pairs of type $p_{c,1}$ can be found. A net is only inserted into a building block circuit if a valid pair connected to it has been identified during structure recognition. Hence, the number of nets for a building block circuit is less than or equal to the overall number of nets in the circuit, i.e., $|N_{B,r_j}| \leq |N_i|$. This effect becomes more significant for higher ranked building block circuits. The overall runtime of the pair recognition algorithm improves as it has to iterate over a set of nets which is continuously getting smaller for higher ranks.

The old version furthermore merged all identified building blocks into the data structure $\mathcal{B}$, i.e., a net could be connected to several building blocks of different ranks at the same time. This leads to a higher computational complexity of the function $findBlocks(p_{c,1/2}, n, \mathcal{B})$ which tries to find building block $p_{c,1/2}$ at net n. The new data structure separates the recognition results for each rank, i.e., a net of a building block circuit of rank r_k is only connected to building blocks of the same rank, hence reducing the lookup time of the function $findBlocks(p_{c,1/2}, n, C_{B,r_{c,1}})$.

Algorithms 17 and 18 outline the old [10] and new version of the self-arbitration process using the persistence mechanism. Self-arbitration is used to resolve ambiguities early in the

Algorithm 19 Removal of dominated building blocks.

```
1: procedure DOMINANCERELATIONS(C_B)
2:     D = findDominanceRelations(maxRank(C_B))
3:     for all d ∈ D do
4:         B_M = findDominatingBuildingBlocks(C_B, d)
5:         B_S = findDominatedBuildingBlocks(C_B, d_i)
6:         for all (b_m, b_s) ∈ (B_M × B_S) do
7:             if overlap(b_m, b_s) then
8:                 remove(C_i, b_s)
9:             end if
10:        end for
11:    end for
12: end procedure
```

recognition process and is called whenever the identification of each library element of the given rank has been completed. The persistence of a building block defines for how many iterations after its identification it will remain in the structure recognition process without becoming part of a higher ranked building block. E.g., a differential pair has a persistence of one. Hence, an identified differential pair must become part of a differential stage on the next rank. Otherwise it will be discarded.

The implementation of [10] uses the set $\mathcal{B}$ which contains all arrays and pairs of the circuit. The algorithm iterates over each building block $b \in \mathcal{B}$ and checks whether it is has to be discarded (lines 5 and 7). A block is discarded from $\mathcal{B}$ in line 8 if it has not become part of a higher ranked building block and if its persistence expired [10]:

$$expired(b, r_k) = \begin{cases} 1 & \text{, if } r_k > r_b + persistence(b) \\ 0 & \text{, else.} \end{cases} \tag{A.1}$$

The rank r_k thereby corresponds to the highest rank of an identified structure of $\mathcal{B}$ and is determined by function $maxRank(\mathcal{B})$ in line 2. Removing an element from $\mathcal{B}$ might affect the result of eq. (A.1) for other already processed building blocks, i.e., previously valid building blocks might have become invalid after a higher ranked structure has been removed from $\mathcal{B}$. Hence, the self-arbitration mechanism is repeated until no more elements can be removed from $\mathcal{B}$ (lines 3 and 12).

In the new implementation, self-arbitration has to iterate only once over all identified building blocks, starting from the pairs stored in the highest ranked building block circuit down to the array circuit (line 3). Hence, the runtime of the arbitration procedure has been improved compared to [10].

Algorithm 19 shows a second mechanism to resolve ambiguities during the recognition process by dominance relations. A dominance relation specifies one building block type which dominates another set of building block types. A dominated building block is removed from C_B if it overlaps with another building block dominating it (lines 7 and 8). For each rank, Algorithm 19 fetches a predefined set of dominance relations and enforces them on the specified building block types (lines 2 and 3 - 11).

The dominance relations presented by [10] are limited to cross coupled pairs. In this version, the definition of dominance relations has been generalized, providing a fine-grained control

mechanism to resolve building block ambiguities. Furthermore, the application of dominance relations has been simplified compared to the implementation of [10] as the requested building block types can be directly looked up in the corresponding building block circuits.

A.6. Experimental Result

Dual Gain Amplifier

Bias and Core Circuitry

Fig. A.7 shows the structure recognition results for the bias and circuitry of the dual gain amplifier (DGA) from Fig. A.5. The upper left shows the array circuit C_{B,DGA_{bias},r_0} of the bias circuitry. It contains four resistor, six diode and six normal transistor arrays.

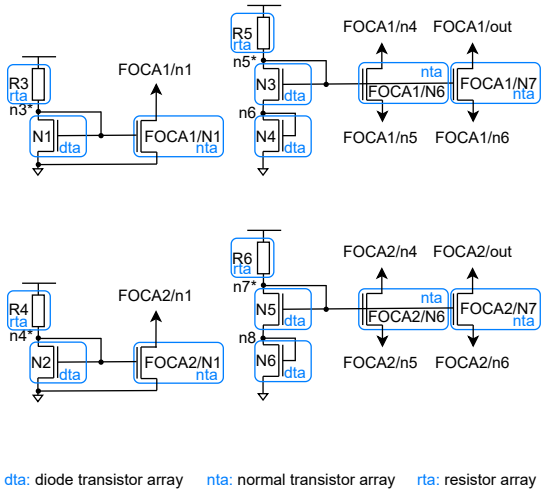
The upper right shows the array circuit C_{B,DGA_{core},r_0} of the core circuit. Two resistor arrays have been identified. The folded cascode amplifier subcircuits $FOCA1$ and $FOCA2$ have been drawn in dashed lines as structure recognition ignores subcircuit blocks. Net in is also dashed as it is not connected to one of the resistor arrays and hence not part of C_{B,DGA_{core},r_0}.

The lower part of Fig. A.7 shows the building block circuit C_{B,DGA_{bias},r_1} which contains all elements of rank one of the bias circuitry. It consists of two simple current mirrors and four level shifters. The core part does not contain any elements from rank one is hence empty.

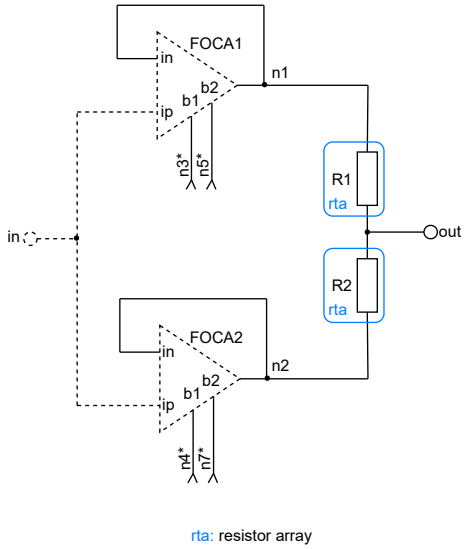
The building block circuit of the complete amplifier $C_{B,DGA,r_1} = C_{B,DGA_{bias},r_1} \cup C_{B,DGA_{core},r_1}$ has four fewer nets than the combined array circuits $C_{B,DGA,r_0} = C_{B,DGA_{bias},r_0} \cup C_{B,DGA_{core},r_0}$. Pair recognition is called for each net and each library element of the current rank. Rank two of L_{analog} contains six elements. Hence, $4 \cdot 6 = 24$ executions of the pair recognition algorithm are saved with the new data structure compared to [17; 10] as the circuit on rank one contains four fewer nets than the original circuit.

Folded Cascode Amplifier

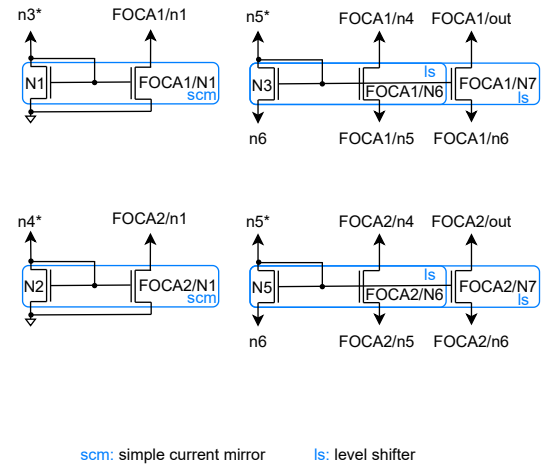
Figs. A.8 and A.9 show the structure recognition results for the folded cascode amplifier (FOCA) from Fig. A.6. The array circuit $C_{B,FOCA,r_0}$ contains eleven normal and two diode transistor arrays. On rank one, three simple current mirrors, three level shifters, one differential pair, one voltage reference two and three cascode pairs are identified. On the next rank, one cascode current mirror, one improved Wilson current mirror and a differential stage are recognized. Please note: the differential stage is only used to validate the previously identified differential pair and is discarded afterwards again as it does not generate any additional constraints for the power-down verification or power-down synthesis procedures from Chaps. 3 and 4. The cascode pairs $cp(N1, N2)$ and $cp(N1, N3)$ overlap with the differential pair $dp(N2, N3)$ and did not become part of a higher ranked structure. Hence, self-arbitration removes them from $C_{B,FOCA,r_1}$ which is indicated by dashed blue lines in the figure. The final recognition results of the folded cascode amplifier are summarized on the bottom of Fig. A.9.



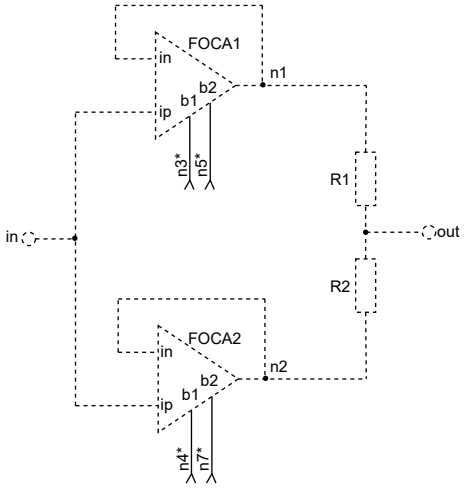
Bias circuitry of the dual gain amplifier: building block circuit C_{B,DGA_{bias},r_0}



Core circuit of the dual gain amplifier: building block circuit C_{B,DGA_{core},r_0}

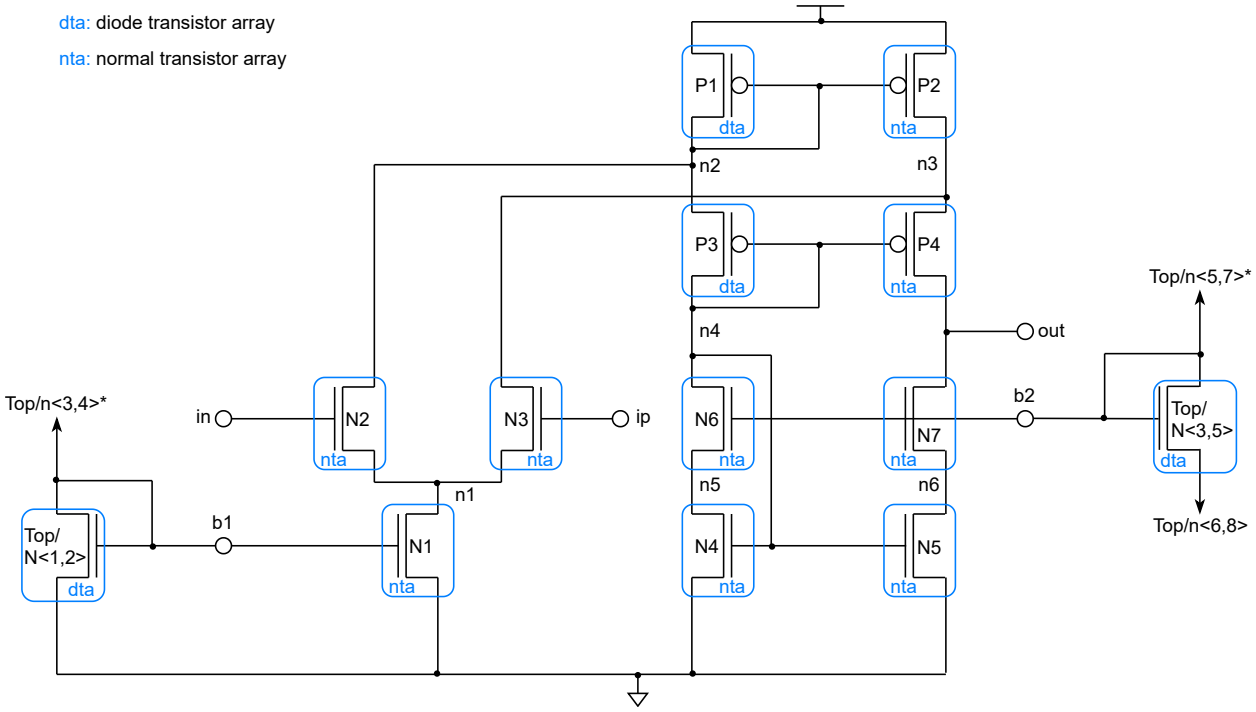


Bias circuitry of the dual gain amplifier: building block circuit C_{B,DGA_{bias},r_1}

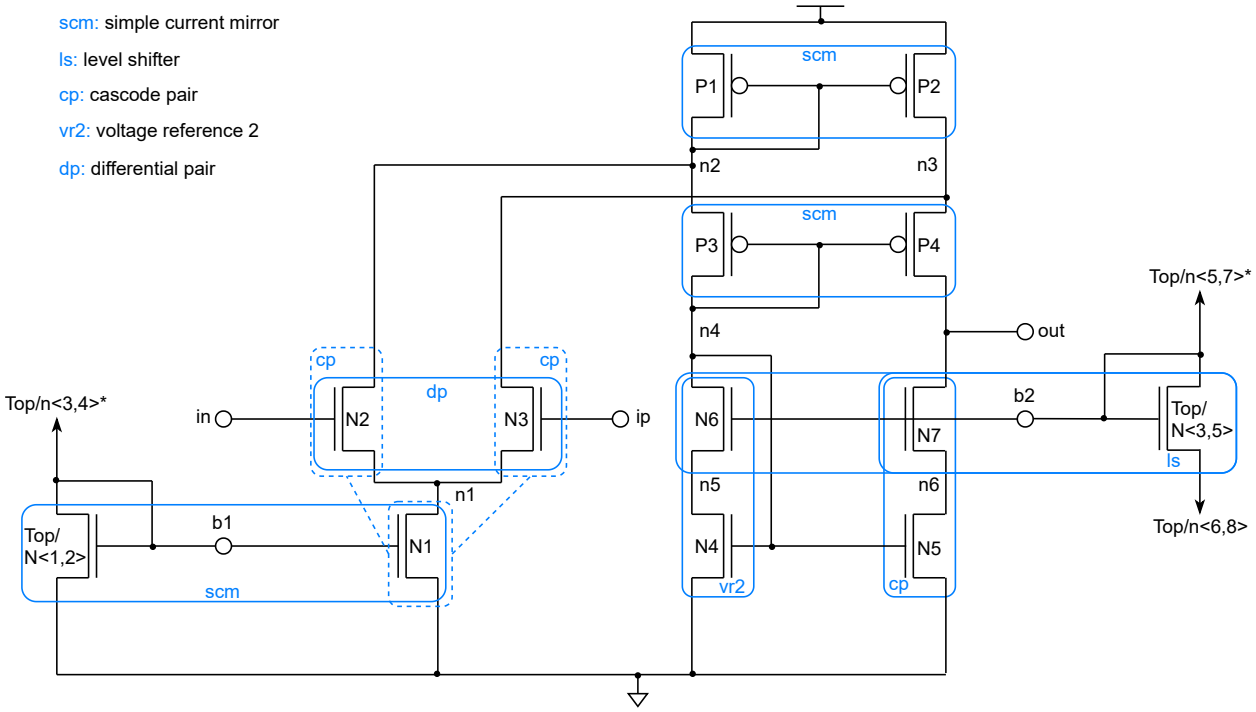


Core circuit of the dual gain amplifier: building block circuit C_{B,DGA_{core},r_1}

Figure A.7.: Structure recognition results for the bias circuitry of the dual gain amplifier (DGA) from Fig. A.5a.

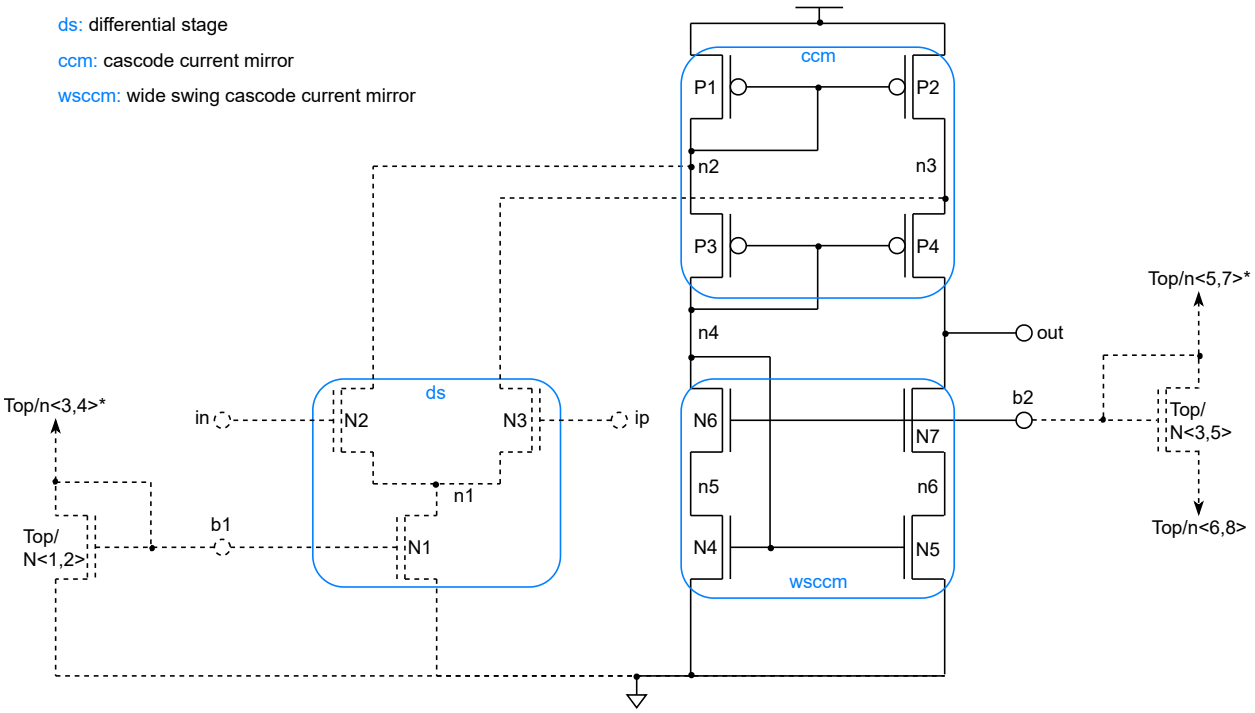


Folded cascode amplifier: building block circuit $C_{B,FOCA,r_0}$

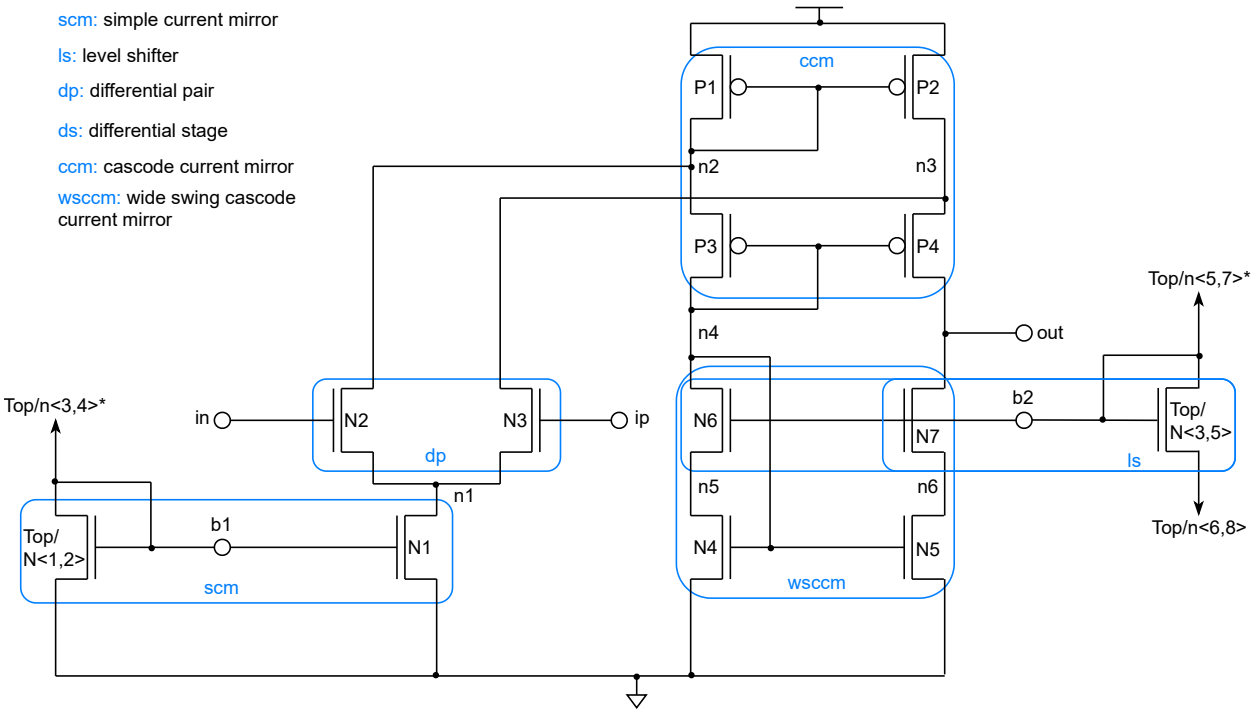


Folded cascode amplifier: building block circuit $C_{B,FOCA,r_1}$

Figure A.8.: Structure recognition results of rank zero and rank one for the folded cascode amplifier (FOCA) from Fig. A.6.



Folded cascode amplifier: building block circuit $C_{B,FOCA,r_2}$



Folded cascode amplifier: final results $C_{B,FOCA}$

Figure A.9.: Structure recognition results of rank two and final results for the folded cascode amplifier (FOCA) from Fig. A.6.

B. Hierarchical Symmetry Computation

B.1. Introduction

Symmetry is a widely used design principle in analog circuit design: fully differential amplifiers are built by two identical circuit halves, their layout is symmetrical which improves the accuracy, power-supply and common mode rejection ratio of the amplifier and makes it less susceptible to manufacturing variations, cross talk, DC coupling and substrate noise [53; 54; 55; 17].

The symmetry information inside a circuit can be further used to constrain the solution space of numerical circuit optimization tools, e.g., Wicked [56], which improves their runtime and the quality of the computed results [57]. Symmetries must also be considered in the circuit's layout to minimize the influence of parasitic mismatch on its performances [58; 59].

B.2. State of the Art and Contributions

The automatic identification of symmetries in netlists and layouts is an ongoing research topic. Many methods thereby follow a graph-based approach. The method by [58] additionally uses a sensitivity analysis, [53] grows two symmetric connection trees around a center element, [60; 61] uses weighted bipartite graph matching, [62] adapted the Gemini II algorithm [64] and [54; 63] analyze signal paths for symmetry computation in their respective graph models. All of these methods require a fully symmetric netlist as inputs, the netlists for [58; 60; 61] must be sized. The methods [53; 60; 61; 62; 54; 63] claim to be able to handle near symmetries by netlist transformations. However, it is often not explained in detail how this can be achieved.

The authors of [55; 65] extract symmetries from a circuit layout using a graph model.

The methods presented by [24; 59; 57] introduced a signal flow graph library which models the signal flow of the analog basic building blocks from [4] qualitatively. The signal flow graph of the given circuit is then obtained by merging the graph models of its basic building blocks which are identified by structure recognition, e.g., [51]. The methods by [24; 59] recursively track the signal flow from the inputs to the outputs of the circuit whilst identifying symmetry pairs in the traversed edges. The method by [57] formulates and solves a constraint optimization problem based on this signal flow graph to identify symmetries in the core, i.e., signal processing, and bias part of the circuit. The method is able to identify multiple symmetry axes in the design.

The method by [66] implements similar ideas to [24; 59], but additionally uses machine learning algorithms to identify approximate graph symmetries based on the graph edit distance metric.

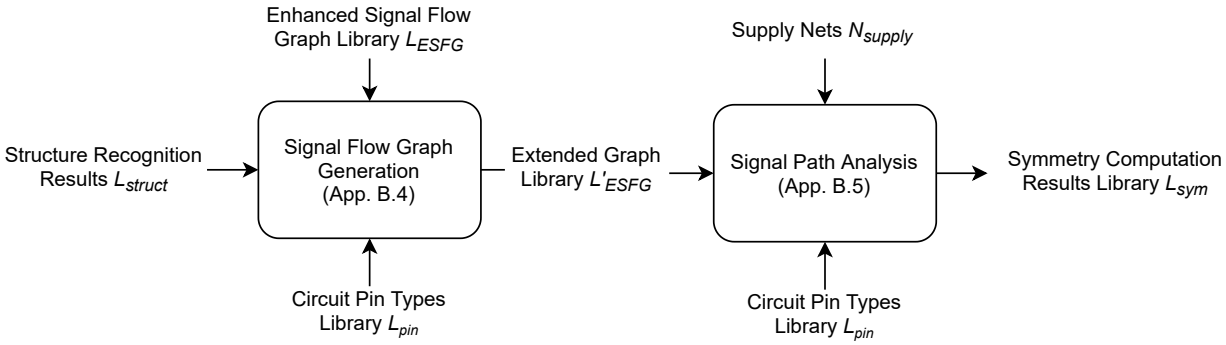


Figure B.1.: Overview of the hierarchical symmetry computation method.

[52] identifies analog basic building blocks by vector encodings. The same publication presents a method which identifies symmetries in the circuit by tracking devices with identical encodings alongside signal paths similarly to [24; 59].

[67] performs a Kolmogorov-Smirnov test to measure the similarity of the eigenvalue distribution of two graphs to extract symmetries from the circuit.

Most of the mentioned methods operate on device level. The methods [66; 52; 67] claim to support hierarchical designs; they however state that they have to flatten hierarchical designs in a preprocessing step.

[23] presents a hierarchical symmetry computation approach based on [57]. It handles circuit hierarchies by traversing them recursively, alternately executing structure recognition, signal flow graph generation and symmetry computation. This method does not reuse intermediate results and cannot be parallelized due to its recursive nature.

This work presents a new hierarchical symmetry computation method. It uses the hierarchical structure recognition results from App. A. In contrast to [23], the circuit's hierarchy levels are already decoupled due to partial circuit flattening in the structure recognition process. This enables the parallelization of symmetry computation which has, however, not been implemented yet. Furthermore, intermediate results are reused whenever possible.

For symmetry computation itself, an algorithm similar to [24; 59] has been implemented. However, the following adjustments have been made: the algorithm traces the signal paths not only in forward, but also in backward direction allowing to identify symmetries in the bias and feedback paths of the given circuit. Additional edge attributes are introduced which define more accurate matching conditions. Approximate matching conditions for single nodes allow a greater flexibility during symmetry computation.

B.3. Overview

Fig. B.1 gives an overview of the implemented hierarchical symmetry computation method. It takes a structure recognition results library L_{struct}, the enhanced signal flow graph library L_{ESFG}, the supply nets N_{supply} and the pin type library L_{pin} as inputs. In the first step, it extends the signal flow graph library L_{ESFG} to L'_{ESFG} based on the structure recognition results L_{struct} of the given circuit. L'_{ESFG} then contains one signal flow graph for each circuit type used to form the hierarchical design.

An enhanced signal flow graph (ESFG) models how the input signals of a circuit traverse its building blocks to its outputs [24; 59].

The symmetry computation is performed by an algorithm similar to [24; 59] for each element of L'_{ESFG}. That algorithm recursively tracks symmetrical edges alongside the signal paths of the given graph. The (differential) input, (differential) output and bias pins of a circuit C_i, specified by the library L_{pin}, serve as starting or end point for the computations.

B.4. Signal Flow Graph Generation

Analog basic building blocks process a given (differential) input signal in various different ways, e.g., a simple current mirror amplifies and copies a reference current from its input to its output pin; a differential pair converts a differential input voltage into a differential output current.

This behavior is qualitatively modelled in [59] by a so called "enhanced signal flow graph" $G(N, E)$. The set N contains the nodes of the graph which correspond to a subset of the nets in the circuit. The set E contains directed edges $e_j = (n_+, n_-)$ which model a signal flow between its start and end its nodes, n_+ and n_-, respectively. Each edge has an attribute

$$att(e_j) = (p_+, p_-, bb, sst) \tag{B.1}$$

which contains information about the building block bb from which the edge has been generated. The substrate type of bb is denoted as sst and the pins of bb at which the edge originates and terminates are written as p_+ and p_-, respectively. The substrate type is either of type n or p, i.e., the building block is entirely formed by NMOS or PMOS transistors, or of type u, which is used for passive devices or a mixture of NMOS and PMOS transistors.

The attributes of two edges e_k, e_l are identical if the following equation holds [17]:

$$\begin{aligned} att(e_k) == att(e_l) \;\Leftrightarrow\; & p_+(e_k) == p_+(e_l) \wedge && (\text{"same start pin"}) \\ & p_-(e_k) == p_-(e_l) \wedge && (\text{"same end pin"}) \\ & type(bb(e_k)) == type(bb(e_l)) \wedge && (\text{"same building block type"}) \\ & sst(bb(e_k)) == sst(bb(e_l)) && (\text{"same substrate type"}) \end{aligned} \tag{B.2}$$

This condition is crucial to identify symmetries in a signal flow graph as will be shown in Sec. B.5.

The graph model $G(N, E)$ from [51] for flat circuits has been extended for hierarchical circuits to $G_i(P_i, N_i, E_i, t_i)$. The symbol t_i denotes a circuit type from set $\mathcal{C}$ (see eq. (2.1)) and the set of pins $P_i \subseteq N_i$ represents possible connections to external signal flow graphs.

Fig. B.2 shows the signal flow graph library for the analog basic building blocks from Fig. A.3. The signal flow of a cross coupled pair, e.g., is modelled by two antiparallel edges between its input and its output pin.

The shown library is almost identical to the one presented in [17]. However, the following adjustments have been made: only elements without a persistence in the structure recognition library are included; signal flows between the inner pins of more sophisticated current

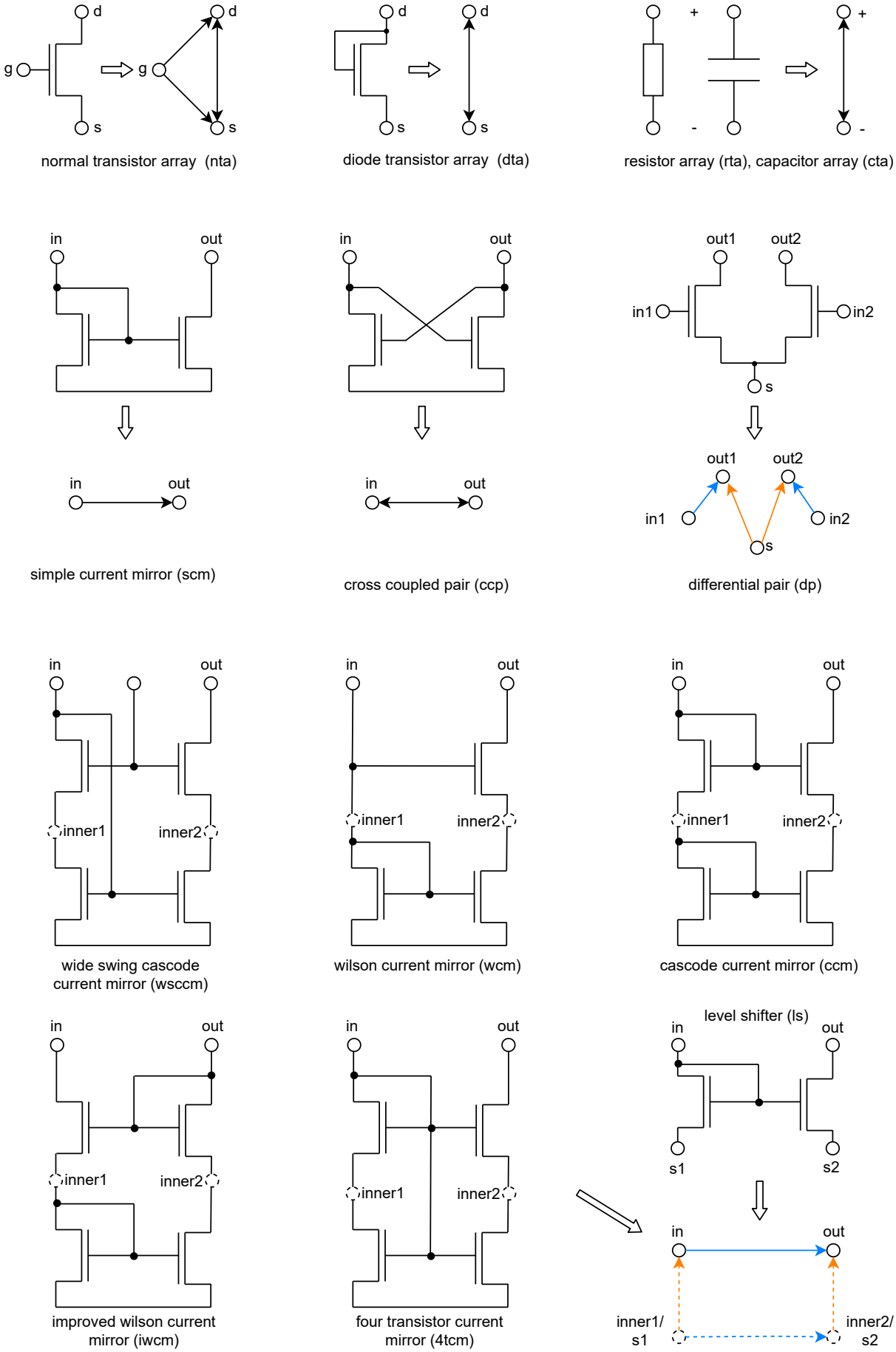
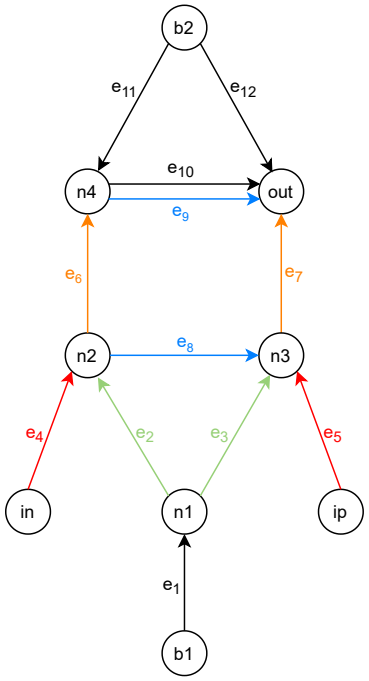


Figure B.2.: Signals flow graph library L_{ESFG} for the analog basic building blocks of Fig. A.3 according to [17].



(a) signal flow graph $G_{D,FOCA}$

edge	attributes			
	p_+	p_-	bb	sst
e_1	in	out	$scm(TOP/N{<}1,2{>},N1]$	n
e_2	s	$out1$	$dp(N2,N3)$	n
e_3	s	$out2$	$dp(N2,N3)$	n
e_4	$in1$	$out1$	$dp(N2,N3)$	n
e_5	$in2$	$out2$	$dp(N2,N3)$	n
e_6	$inner1$	$inner2$	$ccm(P1-P4)$	p
e_7	in	out	$ccm(P1-P4)$	p
e_8	$inner1$	in	$ccm(P1-P4)$	p
e_9	$inner2$	out	$ccm(P1-P4)$	p
e_{10}	in	out	$wsccm(N4-N7)$	n
e_{11}	in	out	$ls(TOP/N{<}3,5{>},N6)$	n
e_{12}	in	out	$ls(TOP/N{<}3,5{>},N7)$	n

(b) edge attributes

Figure B.3.: Detailed signal flow graph and corresponding edge attributes of the folded cascode amplifier (FOCA) from Fig. A.6. The structure recognition results used for signal flow graph generation are shown in Fig. A.9.

mirror structures are considered in order to support folded cascode structures. The dashed edges of the current mirrors from rank two of L_{ESFG} (see Fig. B.2) are only kept if at least one non-optional edge of another building block starts or ends at the inner pins of these current mirrors. Additionally, identically colored edges of building blocks of the same type are considered symmetrical even if they originate and end at different pins of such blocks. E.g., a differential pair converts a differential input voltage at the pins $(in1, in2)$ into a differential output current at the pins $(out1, out2)$. Hence, the edges $e_1 = (in1, out1)$ and $e_2 = (in2, out2)$ are symmetric and drawn with the same color.

The signal flow graph of a circuit is generated by merging the signal flow graphs of each of its top-level building blocks.

Fig. B.3 shows the signal flow graph and the corresponding edge attributes for the folded cascode amplifier (FOCA) from Fig. A.6. The circuit contains one simple current mirror, one differential pair, two level shifters, one cascode and one wide-swing cascode current mirror as top-level building blocks (see Fig. A.9). The signal flow through the wide-swing cascode current mirror $wsccm(N4 - N7)$ is modelled only by edge e_{10} between the nodes $n4$ and out of the graph as no other non-optional edge starts or ends at its inner pins. The optional edges from the two level shifters which would have been inserted at nodes $n5$ and $n6$ have also been removed from the final graph due to the same reason.

The signal flow at the inner pins of the cascode current mirror has to be captured in the graph as these pins are connected to the output pins of the differential pair via edges.

When encountering a subcircuit block during graph generation, its underlying detailed signal

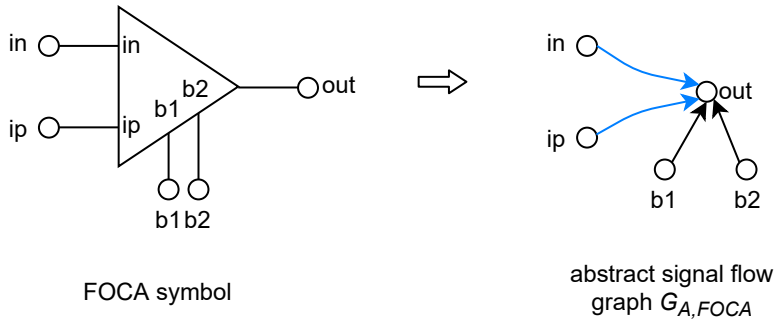


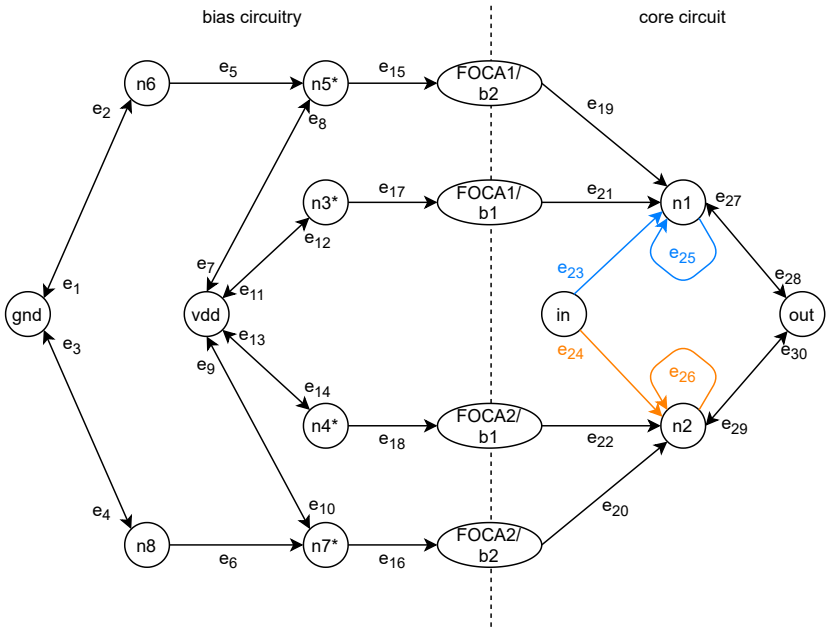
Figure B.4.: Abstract signal flow graph $G_{A,FOCA}$ of the folded cascode amplifier from Fig. A.6.

flow graph has to be abstracted such that its internal behavior is simplified yet qualitatively captured on the hierarchy level containing the subcircuit block. Therefore, an edge between each input and each output pin of the subcircuit block is generated. Bias pins are thereby treated as inputs, as small changes in the bias current or bias voltage signal are only reflected at the outputs of the circuit. Hence, each subcircuit of $\mathcal{C}$ is associated to an *abstract* and a *detailed* graph model $G_{A,i}$ and $G_{D,i}$, respectively. The set of pins P and the set of nets N are identical for all abstract signal flow graphs $G_{A,i}$. Furthermore, the abstract graph model for analog basic building blocks corresponds to its detailed graph model.

Fig. B.4 shows the abstract signal flow graph $G_{A,FOCA}$ of the folded cascode amplifier from Fig. B.3. The folded cascode amplifier has one differential input, *in* and *ip*, two bias pins, $b1$ and $b2$, and one output pin *out*. Hence, the abstract graph model contains four edges in total. Each of these edges is pointing from an input or a bias pin to the output. The edges from the differential input pins are drawn in blue as they are considered to be symmetric.

The abstract graph model of the folded cascode amplifier is then used to construct the detailed signal flow graph of the dual gain amplifier from Fig. A.5. The corresponding signal flow graph is shown in Fig. B.5. It consists of 16 nodes and 30 edges in total. The core circuit instantiates the abstract folded cascode amplifier graph model $G_{A,FOCA}$ two times and contains two edges each for the resistor arrays $rta(R1)$ and $rta(R2)$. The bias circuitry consists of four transistor arrays which corresponds to eight edges in the graph, two simple current mirrors, i.e., two edges, and two level shifters each of them generating two edges. The attributes of the edges are shown at the bottom of Fig. B.5.

Algorithm 20 shows how the signal flow graph for hierarchical circuits is generated. It takes the structure recognition results library L_{struct} and the signal flow graph library L_{ESFG} from Fig. B.2 as input. L_{ESFG} is iteratively extended by the abstract and detailed graph models for all building block circuits $C_{B,i}$ of L_{struct}. The algorithm first initializes L'_{ESFG} as a copy of the signal flow graph library L_{ESFG} from Fig. B.2 (line 2). Then, it iterates over the elements $C_{B,i}$ of L_{struct} and initializes an empty detailed graph model $G_{D,i}$ for each of it (lines 3 and 4). This graph model is filled in by merging the abstract graph model $G_{A,bb}$ of each of its top-level building blocks into it (lines 6 - 9). A top-level building block is either a subcircuit block of $C_{B,i}$ or an analog basic building block which is not part of a higher ranked building block from the structure recognition library (Fig. A.3). If an abstract graph model for a subcircuit block is not present in L'_{ESFG} yet, it is automatically created by function $findOrCreateAbstractGraphModel(bb, L'_{ESFG}, L_{pin})$ as described in the previous paragraphs and stored in L'_{ESFG} (line 7). After all top-level blocks of $C_{B,i}$ have



(a) detailed signal flow graph

edge	bias circuitry attribute				edge	core circuit attribute			
	p_+	p_-	bb	sst		p_+	p_-	bb	sst
e_1	d	s	$dta(N4)$	n	e_{19}	$b2$	out	$FOCA1$	u
e_2	s	d	$dta(N4)$	n	e_{20}	$b2$	out	$FOCA2$	u
e_3	d	s	$dta(N6)$	n	e_{21}	$b1$	out	$FOCA1$	u
e_4	s	d	$dta(N6)$	n	e_{22}	$b1$	out	$FOCA2$	u
e_5	$s1$	in	$ls(N3, FOCA1/N6)$	n	e_{23}	ip	out	$FOCA1$	u
e_6	$s1$	in	$ls(N5, FOCA2/N6)$	n	e_{24}	ip	out	$FOCA2$	u
e_7	$+$	$-$	$rta(R5)$	u	e_{25}	in	out	$FOCA1$	u
e_8	$-$	$+$	$rta(R5)$	u	e_{26}	in	out	$FOCA2$	u
e_9	$+$	$-$	$rta(R6)$	u	e_{27}	$+$	$-$	$rta(R1)$	u
e_{10}	$-$	$+$	$rta(R6)$	u	e_{28}	$-$	$+$	$rta(R1)$	u
e_{11}	$+$	$-$	$rta(R3)$	u	e_{29}	$+$	$-$	$rta(R2)$	u
e_{12}	$-$	$+$	$rta(R3)$	u	e_{30}	$-$	$+$	$rta(R2)$	u
e_{13}	$+$	$-$	$rta(R4)$	u					
e_{14}	$-$	$+$	$rta(R4)$	u					
e_{15}	in	out	$ls(N3, FOCA1/N<6, 7>)$	n					
e_{16}	in	out	$ls(N5, FOCA2/N<6, 7>)$	n					
e_{17}	in	out	$scm(N1, FOCA1/N1)$	n					
e_{18}	in	out	$scm(N2, FOCA2/N1)$	n					

(b) attributes

Figure B.5.: Signal flow graph and edge attributes of the dual gain amplifier from Fig. A.5. The structure recognition results for signal flow graph generation of the bias circuitry are shown in Fig. A.7. The abstract signal flow graph for $FOCA1$ and $FOCA2$ is shown in Fig. B.4.

Algorithm 20 Hierarchical signal flow graph generation.

```
1: procedure BUILDSIGNALFLOWGRAPH(L_struct, L_ESFG, L_pin)
2:     L'_ESFG = L_ESFG
3:     for all C_B,i ∈ L_struct do
4:         G_D,i = ∅
5:         T_i = findTopLevelStructures(C_B,i)
6:         for all bb ∈ T_i do
7:             G_A,bb = findOrCreateAbstractGraphModel(bb, L'_ESFG, L_pin)
8:             G_D,i = G_D,i ∪ G_A,bb
9:         end for
10:        L'_ESFG = L'_ESFG ∪ G_D,i
11:    end for
12:    return L'_ESFG
13: end procedure
```

Algorithm 21 Symmetry computation for analog circuits.

```
1: procedure SYMMETRYCOMPUTATION(L_ESFG, L_pin, N_supply)
2:     L_sym = ∅
3:     for all G_D,i ∈ L_G do
4:         P_diff,in,i = findDifferentialInputs(L_pin, G_D,i)
5:         P_single,in,i = findSingleInputs(L_pin, G_D,i)
6:         P_bias,i = findBiasPins(L_pin, G_D,i)
7:         for all (p_j, p_k) ∈ P_diff,in,i :
8:             diffNodeSymmetries(p_j, p_k, G_D,i, L_sym, P_bias,i, N_supply)    // See Alg. 22
9:         for all p_l ∈ P_single,in,i :
10:            singleNodeSymmetries(p_l, G_D,i, L_sym, P_bias,i, N_supply)    // See Alg. 23
11:    end for
12:    return L_sym
13: end procedure
```

been processed, the detailed graph model G_{D,t_i} is also stored in L'_{ESFG} (line 10). Finally, Algorithm 20 returns the library L'_{ESFG} in line 12 all which contains all generated signal flow graphs.

The signal flow graph generation for a hierarchy level is decoupled from all other hierarchy levels in the design as the partial circuit flattening method from Sec. A.4 made the required hierarchical information to model the signal flow available on each hierarchy level. Hence, signal flow graph generation can be parallelized.

B.5. Signal Path Analysis

Algorithms 21 - 26 show the implemented symmetry computation method for analog circuits. As shown in the previous section, partial circuit flattening allows to decouple the circuit hierarchy levels for signal flow graph generation. The same is true for symmetry computation: Algorithm 21 simply iterates over every element of a given enhanced signal flow graph library L_{ESFG} in line 3 and executes Algorithms 22 and 26 in lines 7 and 10 which perform the actual computations for the given graph $G_{D,i}$ without requiring to traverse the circuit hierarchy.

The library L_{pin} contains the following information for each circuit type t_i:

- a set of differential input pins $P_{diff,in,i} \subseteq (P_i \times P_i)$,
- a set of single input pins $P_{single,in,i} \subseteq P_i$,
- and a set of bias pins $P_{bias,i} \subseteq P_i$.

The single and differential input pins carry the (differential) input signals processed by the circuit. The bias pins provide voltages or currents which set the operating point of the circuit and do not process any signals in general.

Algorithm 21 uses the sets $P_{diff,in,i}$ and $P_{single,in,i}$, which are determined by functions $findDifferentialInputs(L_{pin}, G_{D,i})$ and $findSingleInputs(L_{pin}, G_{D,i})$ in lines 4 and 5, as starting points of Algorithms 22 and 26. These track the (differential) input signals through the graph alongside symmetrical edges recursively. The recursive search stops when a bias pin or a supply net of the circuit has been reached. The output pins of the circuit have been explicitly excluded as termination criterion as there might be feedback paths in the circuit which have to be traced back.

Algorithms 22 and 26 partition a given detailed signal flow graph G_D into the following five sets:

- N_s: this set contains all nodes of the graph which are not considered symmetric to any other node of the graph. However, incoming and outgoing edges of such nodes can be pairwise symmetrical.
- E_s: this set contains all edges which do not yield any symmetry condition. They simply transfer a signal from one node to another.
- N_{diff}: this set contains all symmetrical node pairs of the graph. Two nodes are symmetric if it is possible to pairwise match their incoming and outgoing edges.
- E_{diff}: this set contains all symmetrical edge pairs in the graph. Two edges are symmetric if their attributes match, i.e., eq. (B.2) holds, and if they origin or end at the same node or a symmetrical node pair.
- E_{cl}: this set stores all conversion links of a graph, i.e., edges which convert a differential signal into a single ended signal. Conversion links are self-symmetric, i.e., their corresponding devices must be sized identically and layed out symmetrically.

These sets are stored in the library L_{sym} which is the output of Algorithm 21.

Fig. B.6 shows several different scenarios of node and edge connectivity which can be encountered in a signal flow graph during symmetry computation. Their corresponding partitioning is annotated under each of them.

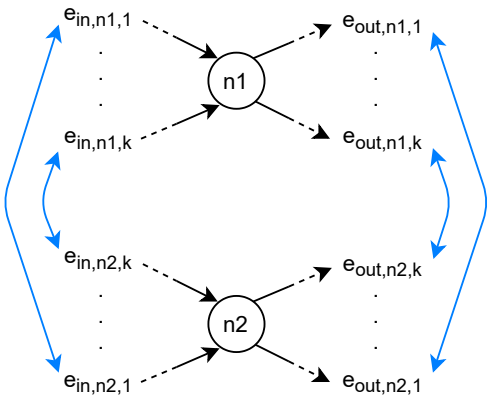
The nodes $n1$ and $n2$ are symmetric as it is possible to pairwise match their incoming and outgoing edges.

Node $n5$ is a merge point in the graph, i.e., two symmetrical edges end in $n5$ which corresponds to merging a differential signal into a single ended one.

The branch node $n6$ splits a single ended signal into a differential one by two symmetrical edges.

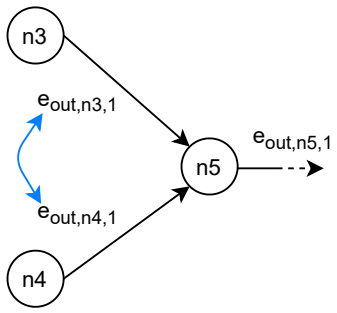
A special case of signal conversion occurs when a differential signal is merged from one node into the other by one self-symmetric edge. Such edges are called “conversion links” from now on. In the example, edge $e_{out,n9,1}$ converts the differential signal at nets $(n9, n10)$ into a single ended one. The merge point is $n10$ as the conversion link ends there.

symmetric node pair



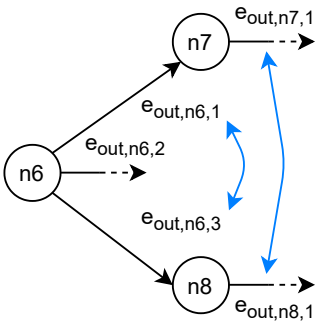
$N_s = \emptyset$
$E_s = \emptyset$
$N_{diff} = \{(n1, n2)\}$
$E_{diff} = \{(e_{in,n1,i}, e_{in,n2,i}) \mid i = 1, 2, ..., k\} \cup$
$\{(e_{out,n1,i}, e_{out,n2,i}) \mid i = 1, 2, ..., k\}$
$E_{cl} = \emptyset$

merge point



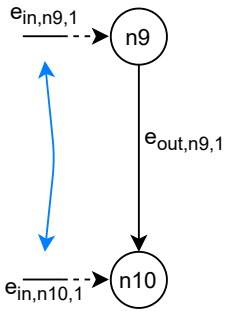
$N_s = \{n5\}$
$E_s = \{e_{out,n5,1}\}$
$N_{diff} = \{(n3, n4)\}$
$E_{diff} = \{(e_{out,n3,1}, e_{out,n4,1})\}$
$E_{cl} = \emptyset$

branch point



$N_s = \{n6\}$
$E_s = \{e_{out,n6,1}\}$
$N_{diff} = \{(n7, n8)\}$
$E_{diff} = \{(e_{out,n6,1}, e_{out,n6,3}),$
$(e_{out,n7,1}, e_{out,n8,1})\}$
$E_{cl} = \emptyset$

conversion link



$N_s = \{n10\}$
$E_s = \emptyset$
$N_{diff} = \{(n9, n10)\}$
$E_{diff} = \{(e_{in,n9,1}, e_{in,n10,1})\}$
$E_{cl} = \{e_{out,n9,1}\}$

Figure B.6.: Different node and edge types encountered during symmetry computation. Symmetry conditions are indicated by blue arrows.

Algorithm 22 Symmetry computation for differential nodes.

```
1: procedure DIFFNODESYMMMETRIES(n_i, n_j, G_D)
     // Additional inputs L_sym, P_bias and N_supply not shown.
2:   N_m = detectConversionLinks(n_i, n_j, G_D)    // Alg. 24.
3:   if N_m ≠ ∅ then
4:     N_s, N_diff = matchEdges(E_in,n_i, E_in,n_j, G_D)    // Alg. 25.
5:     for all (n_k, n_l) ∈ N_diff : diffNodeSymmetries(n_k, n_l, G_D)    // Recursion.
6:     for all n_p ∈ N_m ∪ N_s : singleNodeSymmetries(n_p, G_D)    // Alg. 23.
7:   else
8:     N_s, N_diff = edgeSymmetries(n_i, n_j, G_D)    // Alg. 25.
9:     for all n_p ∈ N_s : singleNodeSymmetries(n_p, G_D)    // Alg. 23.
10:    for all (n_k, n_l) ∈ N_diff : diffNodeSymmetries(n_k, n_l, G_D)    // Recursion.
11:  end if
12: end procedure
```

Algorithm 23 Recursive symmetry computation for single nodes.

```
1: procedure SINGLENODESYMMETRIES(n_i, G_D)
     // Additional inputs L_sym, P_bias and N_supply not shown.
2:   if isBiasPin(n_i) or isSupplyNet(n_i) then
3:     return
4:   end if
5:   N_s, N_diff = edgeSymmetries(n_i, G_D)    // Alg. 26.
6:   for all (n_k, n_l) ∈ N_diff : diffNodeSymmetries(n_k, n_l, G_D)    // Alg. 22.
7:   for all n_j ∈ N_s : singleNodeSymmetries(n_j, G_D)    // Recursion.
8: end procedure
```

Algorithms 22 and 26 require a known symmetrical node pair (n_i, n_j) or an unsymmetrical single node n_s as starting point for symmetry computation. From there, the signal paths of the graph are tracked recursively alongside incoming and outgoing symmetrical edge pairs. Typically used starting points are the (differential) input pins of the given circuit which are provided by the library L_{pin}. However, any other internal known symmetrical node pair or unsymmetrical node of the circuit would be suitable. The pin library L_{pin} must thereby be specified by the designer.

Algorithm 22 discovers symmetries at a differential node pair (n_i, n_j) as follows: It first detects all conversion links between n_i and n_j by calling Algorithm 24 in line 2. The detected conversion links are thereby removed from G_D. Furthermore, this method returns the merge point of the detected conversion link and stores it in the set N_m. If N_m is not empty, i.e., there is a conversion link between n_i and n_j, symmetries in the merged single ended signal are computed by calling Algorithm 23 (line 6). Before that however, symmetries in the incoming edges of (n_i, n_j) have to be detected by calling Algorithm 25 in line 4.

If there is no conversion link between (n_i, n_j), then all edge symmetries in the incoming and outgoing edges of the two nodes have to be computed (lines 8 - 10). The sets N_s and N_{diff}, computed by Algorithm 25 in line 8, contain the next node (pair) which has to be investigated by either calling Algorithm 23 or itself recursively.

Algorithm 23 determines further symmetries at a single node n_i of graph G_D. It operates as follows: it calls Algorithm 26 which partitions the incoming and outgoing edges into the

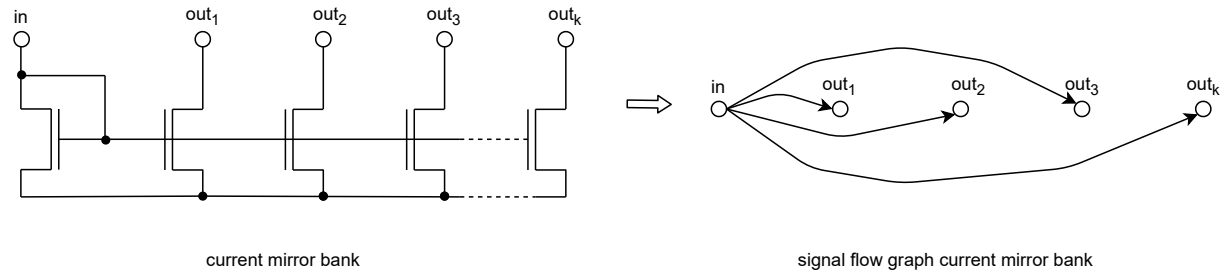


Figure B.7.: A current mirror bank and its corresponding signal flow graph. A unique edge mapping does not exist.

Algorithm 24 Detection of conversion links.

```
1: procedure DETECTCONVERSIONLINKS(n_i, n_j, G_D)
     // Results N_m, E_cl, E_diff are stored in L_sym.
2:   N_m = ∅
3:   E_cl = findConnectingEdges(n_i, n_j)
4:   G_D = G_D \ E_cl
5:   for all e_k ∈ E_cl do
6:     if getEndNode(e_k) == n_i and E_out,n_i == ∅ :  N_m = N_m ∪ n_i
7:     if getEndNode(e_k) == n_j and E_out,n_j == ∅ :  N_m = N_m ∪ n_j
8:   end for
9:   if both n_i, n_j ∈ N_m then    // Antiparallel edges.
10:    N_m = ∅
11:  end if
12:  return N_m
13: end procedure
```

sets E_s and E_{diff} (line 5). Furthermore, that algorithm determines the next single nodes N_s and node pairs N_{diff} to investigate based on these two sets. Algorithm 23 iterates over N_s and N_{diff} either calling itself recursively or calling Algorithm 22. The algorithm stops immediately when n_i is a bias pin or a supply net of the circuit as these net types in general do not carry any signal information.

Additionally, bias pins are often input to a reference current of a current mirror bank which copies and distributes those currents to different branches of the circuit. The attributes of the generated current mirror bank edges have all identical attributes which could lead to falsely identified symmetry pairs in the circuit. This is another reason why bias pins are not taken as start point for symmetry computation. Instead, the edges in the signal flow graph are traced back to the bias and supply pins of the circuit to determine symmetries in the bias part of the circuit.

Algorithm 24 shows how conversion links between two nodes n_i and n_j are detected in a graph G_D. In line 2, the algorithm initializes the set of merge points N_m empty. Then it tries to find edges connecting n_i with n_j by calling function $findConnectingEdges(n_i, n_j)$ in line 3, i.e., edges which have n_i and n_j as start and end node or vice versa, respectively. These edges are removed from the graph G_D and determined self-symmetric (line 4).

Afterwards, the algorithm computes the merge point of the two nodes for each detected conversion link E_{cl} in lines 5 - 8: a signal merges from node n_i into node n_j if there is an

Algorithm 25 Computation of edge symmetries for differential nodes.

```
 1: procedure EDGESYMMETRIES(n_i, n_j, G_D)
 2:     N_{s,in}, N_{diff,in} = matchEdges(E_{in,n_i}, E_{in,n_j}, G_D)
 3:     N_{s,out}, N_{diff,out} = matchEdges(E_{out,n_i}, E_{out,n_j}, G_D)
 4:     return N_{s,in} ∪ N_{s,out}, N_{diff,in} ∪ N_{diff,out}
 5: end procedure

 6: procedure MATCHEDGES(E_{n_i}, E_{n_j}, G_D)
      // Results N_s, N_diff, E_diff are stored in L_sym.
 7:     N_s, N_diff, E_diff = ∅
 8:     for all e_k ∈ E_{n_i} do
 9:         e_l = findMatchingEdge(E_{n_j}, e_k)
10:         E_diff = E_diff ∪ (e_k, e_l)
11:         n_p = getOtherEnd(e_k, n_i), n_q = getOtherEnd(e_l, n_j)
12:         if n_p == n_q then    // Merge point detected.
13:             N_s = N_s ∪ {n_p}
14:         else   // New node symmetry detected.
15:             N_diff = N_diff ∪ (n_p, n_q)
16:         end if
17:     end for
18:     G_D = G_D \ E_diff
19:     return N_s, N_diff
20: end procedure
```

edge connecting n_i with n_j (line 6). Furthermore, the set of outgoing edges E_{out,n_i} must be empty after removing E_{cl} from the graph. Otherwise, the signal continues to be processed at n_i. The same conditions hold for an edge merging a signal from n_j into n_i (line 7).

A special case occurs when both n_i and n_j have been inserted into N_m: this can happen if the two nodes are connected by anti-parallel edges (line 9). Then, no signal merging in the conventional sense took place and N_m is emptied again in line 10.

Finally, the algorithm returns the detected merge points N_m (line 12).

Algorithm 25 identifies new merge points N_s, symmetrical node pairs N_{diff} and symmetrical edge pairs E_{diff} based on the incoming (line 2), i.e., $E_{n_i,in}$ and $E_{n_j,in}$, and the outgoing (line 3), i.e., $E_{n_i,out}$ and $E_{n_j,out}$, edges of a given node pair (n_i, n_j). In this stage of the algorithm, the conversion links between n_i and n_j have already been removed from the graph G_D. Hence, it must be possible to match all remaining edges pairwise. The algorithm thereby uses the function $matchEdges(E_{n_i}, E_{n_j}, G_D)$ shown at the bottom of Algorithm 25. That function initializes the set of merge points N_s, symmetrical nodes N_{diff} and symmetrical edges E_{diff} empty (line 7). Then in lines 8 and 9, it iterates over all edges of $e_k \in E_{n_i}$ and tries to find an edge $e_l \in E_{n_j}$ which has the same attributes as n_j according to (B.2). The detected symmetrical edge pair (e_k, e_l) is stored in E_{diff} in line 10 and the next nodes n_p and n_q which have to be investigated are determined by the function:

$$getOtherEnd(e, n) = \begin{cases} n_+ & \text{, if } n == n_-, \\ n_- & \text{, if } n == n_+ \end{cases} \tag{B.3}$$

Algorithm 26 Computation of edge symmetries for single nodes.

```
 1: procedure EDGESYMMETRIES(n_i, G_D)
 2:     N_{s,in}, N_{diff,in} = matchEdges(E_{in,n_i}, G_D)
 3:     N_{s,out}, N_{diff,out} = matchEdges(E_{out,n_i}, G_D)
 4:     return N_{s,in} ∪ N_{s,out}, N_{diff,in} ∪ N_{diff,out}
 5: end procedure

 6: procedure MATCHEDGES(E_{n_i}, G_D)
      // Results N_s, N_diff, E_s, E_diff are stored in L_sym.
 7:     N_s, N_diff, E_s, E_diff = ∅
 8:     for all e_j ∈ E_{n_i} do
 9:         if hasMatchingEdge(E_{n_i}, e_j) then    // Branch point detected.
10:             e_k = findMatchingEdge(E_{n_i}, e_j)
11:             E_diff = E_diff ∪ (e_j, e_k)
12:             (n_p, n_q) = getOtherEnds(e_j, e_k)
13:             N_diff = N_diff ∪ (n_p, n_q)
14:         else   // Simple signal transfer.
15:             N_s = N_s ∪ getOtherEnd(e_j, n_i)
16:             E_s = E_s ∪ e_j
17:         end if
18:     end for
19:     G_D = G_D \ (E_s ∪ E_diff)
20:     return N_s, N_diff
21: end procedure
```

in line 11. This function determines whether n corresponds to the start or the end node of e and returns its corresponding other node.

In the next step, the algorithm determines whether (e_k, e_l) merges a differential signal into a single ended one by evaluating whether n_p and n_q correspond to the same node (line 12). If a merge point has been detected, n_p is stored in N_s (line 13); otherwise, (n_p, n_q) is inserted into N_{diff} as a new node symmetry has been detected (line 15). Finally, in lines 18 and 19, the identified edge symmetries E_{diff} are removed from G_D and the algorithm returns N_s and N_{diff}.

Algorithm 26 computes the symmetries in the incoming and outgoing edges in lines 2 and 3 for a single node n_i, similarly to Algorithm 25 for differential node pairs. The function $matchEdges(E_{n_i}, G_D)$ shown at the bottom of the algorithm first initializes the sets N_s, N_{diff}, E_s and E_{diff} empty (line 7). Then, it iterates over all edges $e_j \in E_{n_i}$ and tries to find another edge $e_k \in E_{n_i}$ which has the same attributes as e_j by calling function $hasMatchingEdge(E_{n_i}, e_j)$ (lines 8 and 9). If that is the case, the corresponding edge pair is inserted into E_{diff} and the new differential node pair (n_p, n_q) into N_{diff} (lines 10 - 13). If no matching edge is found for e_j, then the signal simply flows through e_j to its other end node (lines 15-16). That node and e_j are inserted into N_s and E_s, respectively.

Finally, the algorithm removes the edges in E_s and E_{diff} from G_D and returns the node sets N_s and N_{diff} in lines 19 and 20.

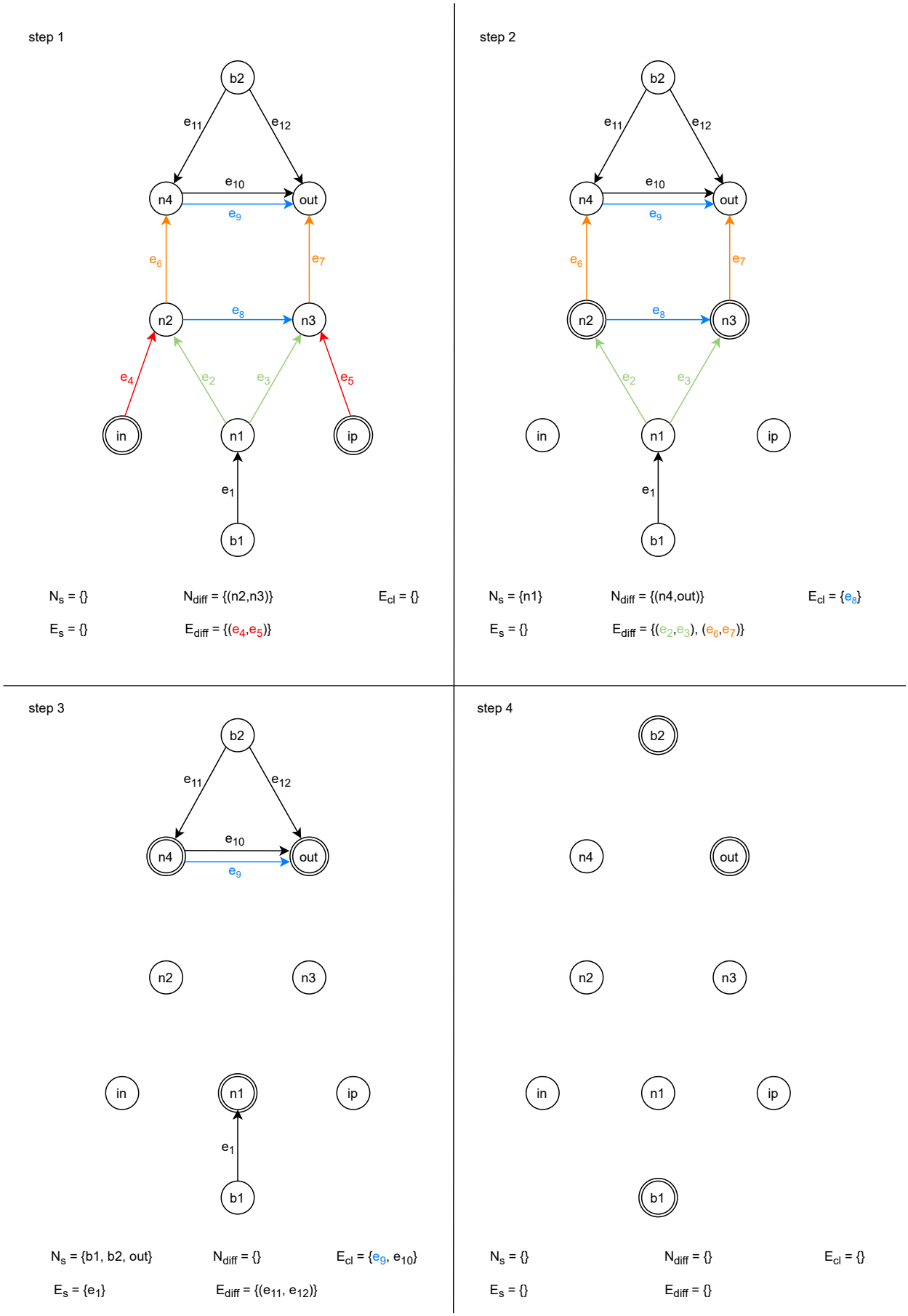


Figure B.8.: Symmetry computation on the graph model of the folded cascode amplifier from Fig. B.3. Each iteration shows intermediate steps of Algorithms 21 - 26.

B.6. Experimental Result

The implemented algorithms are illustrated at the example of the signal flow graph of the folded cascode amplifier from Fig. B.3 in the following. The starting point is the differential input pin pair (in, ip), the bias pins of the circuit are given as $b1$ and $b2$, the output pin as *out*. The nodes which are currently processed in each step are marked by double circles. Underneath each step, the computed intermediate results are shown.

In the first step, Algorithm 22 detects the differential edge and differential node symmetries (e_4, e_5) and $(n2, n3)$ in the graph, respectively. The edges e_4 and e_5 are symmetric because their attributes fulfil eq. B.2: they originate and end at the input and output pins of the differential pair $dp(N2, N3)$. The two edges are afterwards removed from the graph and Algorithm 22 calls itself recursively for $(n2, n3)$. Here, the algorithm discovers the conversion link e_8, the edge symmetries (e_2, e_3), (e_6, e_7), the merge point $n1$ and the symmetrical node pair $(n4, out)$. Node $n3$ is not treated as merge point as there is still another outgoing edge at node $n2$ after the removal of e_8 from the graph. Algorithm 22 calls itself recursively again for $(n4, out)$ where it detects the conversion links e_9 and e_{10}, the edge symmetry (e_{11}, e_{12}) and node *out* as merge point of the differential input signal (e_6, e_7). Furthermore, edge e_1 has been classified by Algorithm 23 as an edge which does not yield any symmetry conditions and $b2$ as branch point of e_{11} and e_{12}. In the last step, symmetry computation terminates as all edges of the graph have been processed.

Fig. B.9 shows the final symmetry computation results for the dual gain amplifier from Fig. A.5, Fig. B.10 the results for its folded cascode amplifier subcircuits $FOCA1$ and $FOCA2$. The results for the dual gain amplifier have been computed with the input pin *in* and the signal flow graph from Fig. B.5 as inputs.

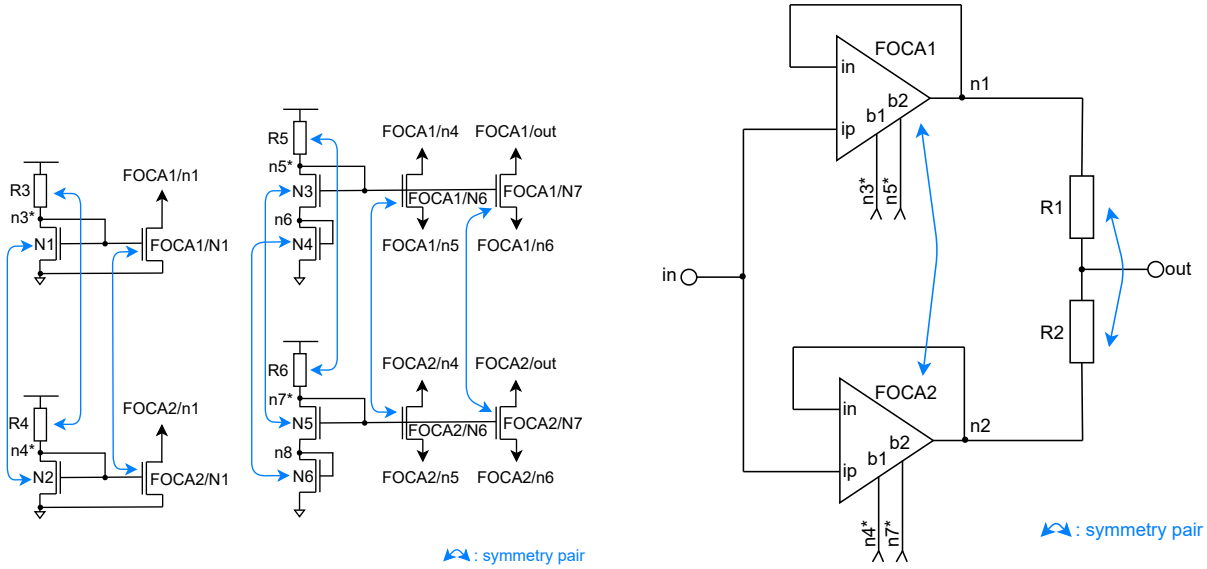
A method to annotate the identified conversion links and symmetrical edge pairs back to device level is described in [17], Sec. 5.3.3. A similar method has been implemented for this work which is additionally supporting the back-annotation to subcircuit block level.

Overall, nine symmetrical device pairs and one symmetrical subcircuit block pair have been identified on the top-level of the dual gain amplifier. Six more device symmetries have been found in each of the folded cascode amplifiers $FOCA1$ and $FOCA2$.

B.7. Discussion and Limitations

The implemented method is greedy, i.e., it accepts any edge pair as symmetric as long as their attributes fulfil eq. (B.2) and if they origin or end at a symmetrical node pair. Hence, wrong results can be produced when ambiguities are encountered during symmetry computation as shown on the left of Fig. B.11: the four outgoing edges of $n1$ all have identical attributes. However, only nodes $(n2, n3)$ and $(n4, n5)$ are symmetrical in the circuit. Hence, the greedy approach can result in falsely identified symmetry pairs as shown in the center of the figure. A mechanism which resolves such ambiguities has been presented by [24]. This mechanism has not been implemented yet.

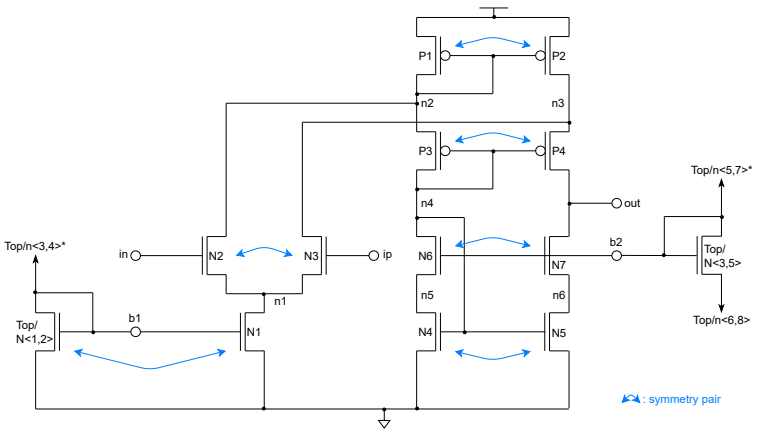
Conversion links typically consist of only one edge in the graph. However, there might be scenarios where a signal is merged into another one by a series of edges as shown on the right side of Fig. B.11. The edges of such conversion paths must be either all self-symmetric



bias circuitry			core circuit		
edges		subcircuits	edges		subcircuits
E_{diff}	(e_1, e_3) (e_2, e_4)	$(N4, N6)$	E_{diff}	(e_{19}, e_{20})	$(FOCA1, FOCA2)$
	(e_5, e_6)	$(N3, N5)$ $(FOCA1/N6, FOCA2/N6)$		(e_{21}, e_{22})	$(FOCA1, FOCA2)$
	(e_7, e_9) (e_8, e_{10})	$(R5, R6)$		(e_{23}, e_{24})	$(FOCA1, FOCA2)$
	(e_{11}, e_{13}) (e_{12}, e_{14})	$(R3, R4)$		(e_{25}, e_{26})	$(FOCA1, FOCA2)$
	(e_{15}, e_{16})	$(N3, N5)$ $(FOCA1/N6, FOCA2/N6)$ $(FOCA1/N7, FOCA2/N7)$		(e_{27}, e_{29}) (e_{28}, e_{30})	$(R1, R2)$
	(e_{17}, e_{18})	$(N1, N2)$ $(FOCA1/N1, FOCA2/N1)$			

(a) bias circuitry (b) core circuit

Figure B.9.: Symmetry computation results for the bias circuitry (a) and the core circuit (b) of the dual gain amplifier from Fig. A.5.



edges		subcircuits
E_{diff}	(e_2, e_3) (e_4, e_5)	$(N2, N3)$
	(e_6, e_7)	$(P1, P2)$ $(P3, P4)$
	(e_{11}, e_{12})	$(N6, N7)$
E_{cl}	e_8, e_9	$(P1, P2)$, $(P3, P4)$
	e_{10}	$(N4, N5)$ $(N6, N7)$
E_s	e_1	–

Figure B.10.: Symmetry computation results for the folded cascode amplifier from Fig. A.6.

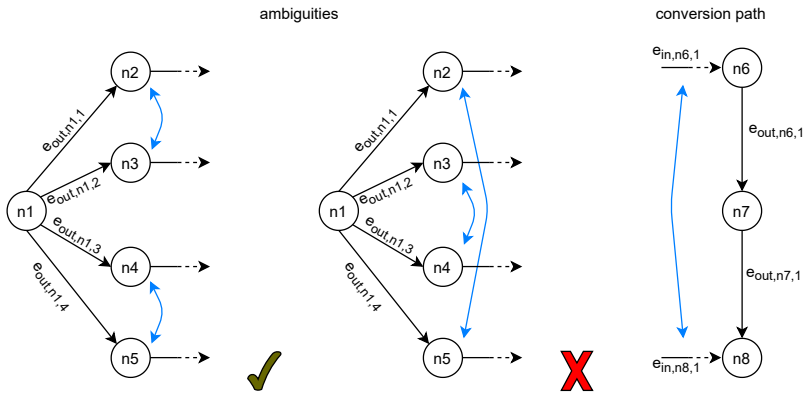


Figure B.11.: Ambiguities and conversion paths in a signal flow graph.

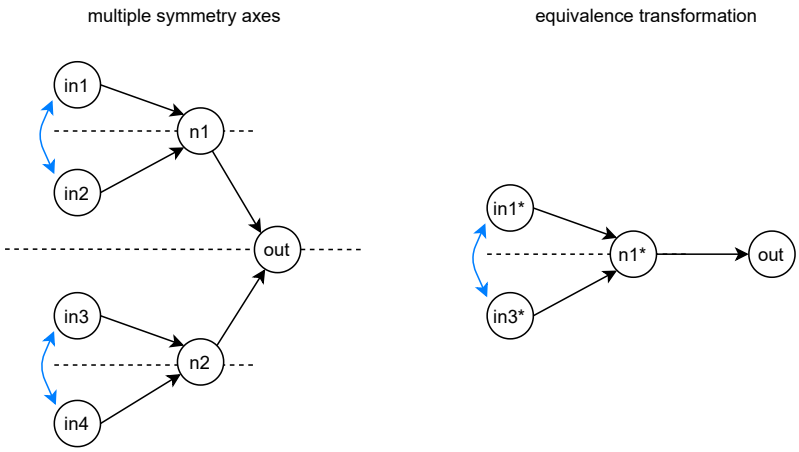


Figure B.12.: Multiple symmetry axes in a graph.

or point symmetric around the center of such a path. Conversion paths can be detected by a depth-first search algorithm [68]. However, this mechanism has not been implemented yet.

Finally, multiple symmetry axes in a circuit as shown in Fig. B.12 cannot be detected yet. The presented algorithms can identify the node symmetries $(in1, in2)$, $(in3, in4)$ and $(n1, n2)$, but not that $[(in1, in2), (in3, in4)]$ are also symmetric.

The work by [17], Sec. 5.1.5, overcomes this problem by a so-called "symmetry equivalence transformation". It combines detected node and edge symmetries into graph components which each represent one of the detected symmetries. In the example on the right, nodes $in1*$, $in3*$ and $n1*$ are the symmetry equivalents for $(in1, in2)$, $(in3, in4)$ and $(n1, n2)$. The symmetry computation algorithm is then rerun for the resulting equivalence graph which reveals further symmetries and symmetry axes in the circuit. These two steps are repeated until no more new symmetries are discovered in the circuit. However, also this method has not been implemented yet.

C. Manual Power-Down Mode Revision

The verification results computed by the power-down verification method from Chap. 3 can be used to revise and resolve the detected errors in the power-down circuitry of a given circuit. In the following, three heuristics are presented to analyze the provided error report.

The design errors are handled according to their severeness, i.e., short circuit paths are handled before dealing with matching and symmetry violations and floating nets. Finally, redundant power-down transistors are identified and removed from the circuit.

After taking one or more of the following measures, it must be verified that no new design errors are introduced by inserting or replacing a power-down transistor by re-running the verification method.

Step 1: Short Circuit Paths

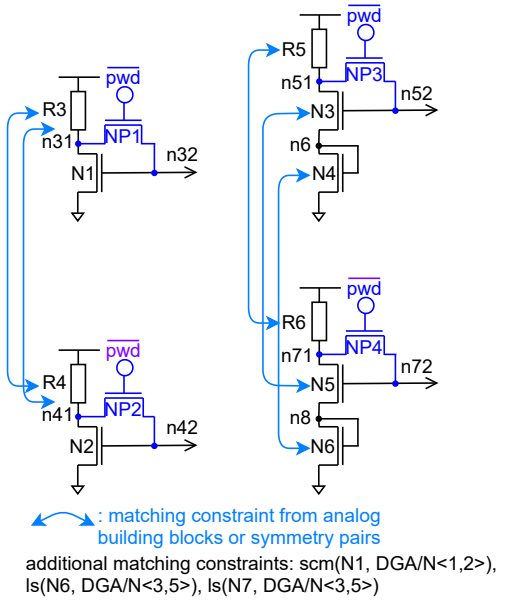
Short circuit paths can be classified into three different types: $definite$, *potential* and *induced* [9].

All devices of a *definite* short circuit path are conducting in power-down mode, i.e., the gates of its transistors are connected to a net with voltage level *gnd* for PMOS and *vdd* for NMOS transistors, respectively. A definite short circuit path can be shut off as follows: if it runs over one or more power-down transistors, removing at least one of those transistors will switch it off. The net which has been previously pulled to *vdd* or *gnd* by the removed power-down transistor takes the opposite voltage level automatically. If it does not run over a power-down transistor, a net on that path has to be split up according to the rip-up patterns from Fig. 1.6.

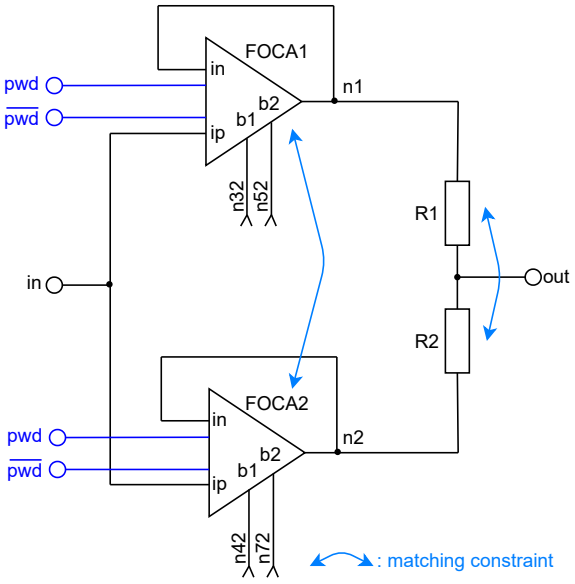
A current path is a *potential* short circuit path if the gate of at least one of its transistors is connected to a floating net. The state of such a transistor is not predictable and could cause a current flow in the circuit. By inserting an additional power-down transistor to force at least one of those transistors into the non-conducting state will switch the potential current flow off.

Induced short circuit paths are caused by other potential or definite short circuit paths which run over at least one of the gates of the induced path. Therefore, all potential and definite short circuit paths should be fixed first. Afterwards, the induced current flows either disappeared or turned into definite or potential ones. In the latter case, additional measures have to be taken as described above.

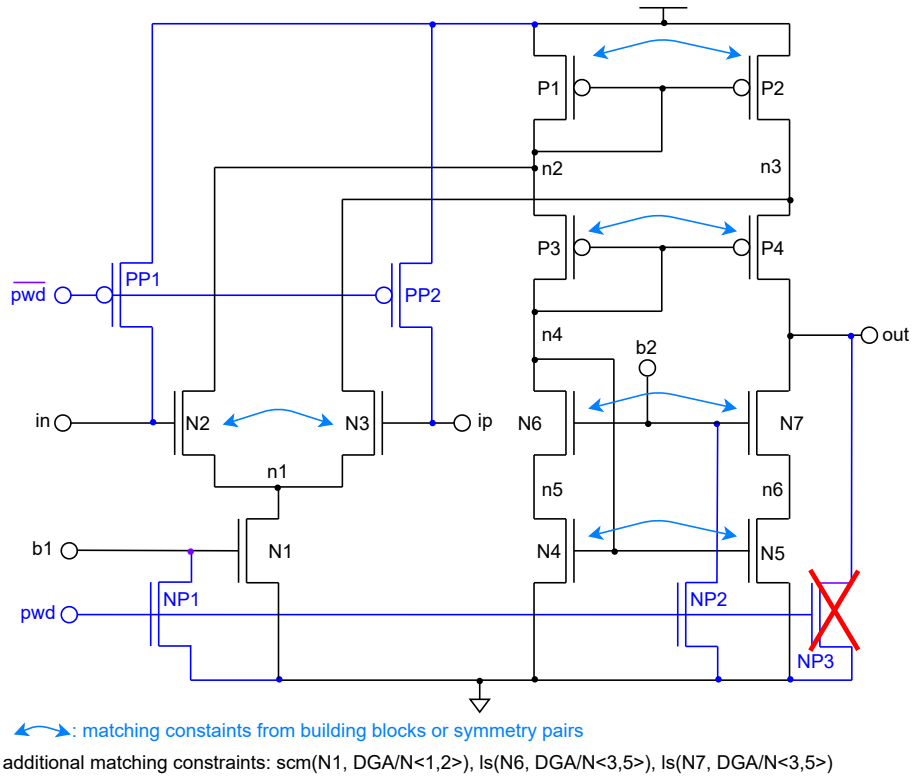
The detected current paths in the dual gain amplifier from Fig. 3.6 are definite and run over the power-down transistors $PP1$ and $NP3$ inside the folded cascode amplifiers $FOCA1$ and $FOCA2$. Hence, removing one of these transistors from the circuit will shut them off. Removing $PP1$ however would allow the ground voltage to propagate to the input pin *in* of the folded cascode amplifiers which would stress their differential pair $dp(N2, N3)$. Hence,



Step 1: no changes



Step 1: no changes



Step 1: $NP3$ removed

Figure C.1.: Measures taken to fix the short circuit paths of the dual gain amplifier and its folded cascode amplifier subcircuits from Figs. 3.6 and 3.7.

$NP3$ must be removed from the circuit. The corresponding modified power-down circuitry is shown in Fig. C.1.

Step 2: Matching and Symmetry Violations

A matching or a symmetry violation is reported whenever the voltage levels at the pins of a matched or symmetrical subcircuit pair differ from each other.

If one of the pins is connected to a floating net and the other to a net with voltage level *vdd* or *gnd*, then a power-down transistor can be added to pull the floating net to the corresponding voltage level.

If both nets have different defined voltage levels, i.e., *vdd* and *gnd*, it can be checked whether one of those nets is pulled to *vdd* or *gnd* by a power-down transistor. That transistor can be replaced by another power-down transistor which pulls the net to the opposite voltage level.

If both nets have different defined voltage levels and both are not connected to a power-down transistor, then it is not possible to resolve the symmetry violation by the above stated measures. In those cases, it must be decided by the designer whether the affected subcircuits are critical for the operation of the circuit and whether the error can be waived or not. If the error cannot be waived, additional measures, e.g. isolating the affected subcircuits in power-down mode by pass gates, must be considered.

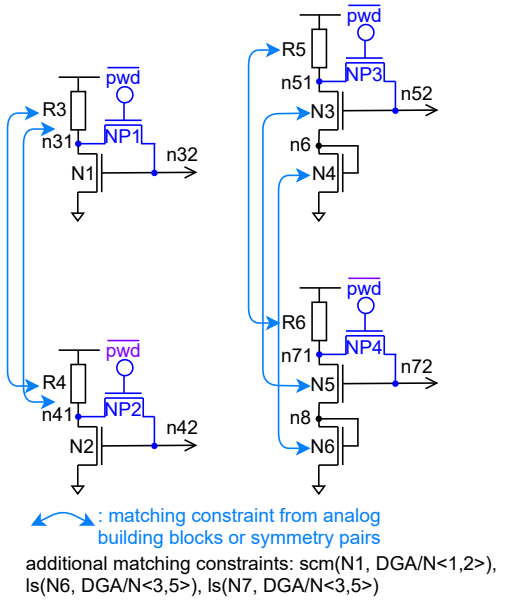
By removing $NP3$ from the power-down circuitry of the folded cascode amplifiers in the previous step, all matching and symmetry violations reported in Sec. 3.5.1 have been resolved. The supply voltage propagates over the top level to the output pins of the folded cascode amplifiers which then fulfill the matching constraints at their cascode and wide-swing cascode current mirrors. Furthermore, $N2$ and $N3$ are conducting. The supply voltage can now propagate to $n1$ and $n3$ which sets the previously floating nets to a defined voltage level. The matching constraints at the simple current mirrors and level shifters which are each formed by one device of the bias circuitry of the dual gain amplifier and by one device of its folded cascode amplifiers are also fulfilled. The resistors $R3$ and $R4$ pull the nets $n31$ and $n41$, i.e., the input pins of the simple current mirrors, to *vdd* which corresponds to the voltage level at their output pins, i.e., *vdd* at net $n1$ of $FOCA1$ and $FOCA2$. It can be shown that the constraints at the level shifters hold analogously.

Hence, no measures have to be taken for the two circuits as shown in Fig. C.2. A subsequent verification run does not report any short circuit paths, floating nodes or matching violations in power-down mode.

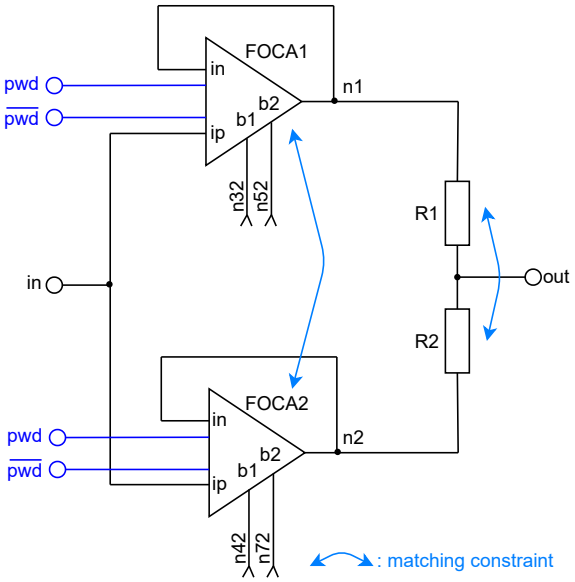
Step 3: Floating Nets and Redundant Power-Down Transistors

After steps one and two have been successfully executed, the remaining floating nets of the circuit can either be pulled to *vdd* or *gnd* by power-down transistors or they can be left floating to reduce the area occupied by the power-down circuitry. This must be carefully decided by the designer.

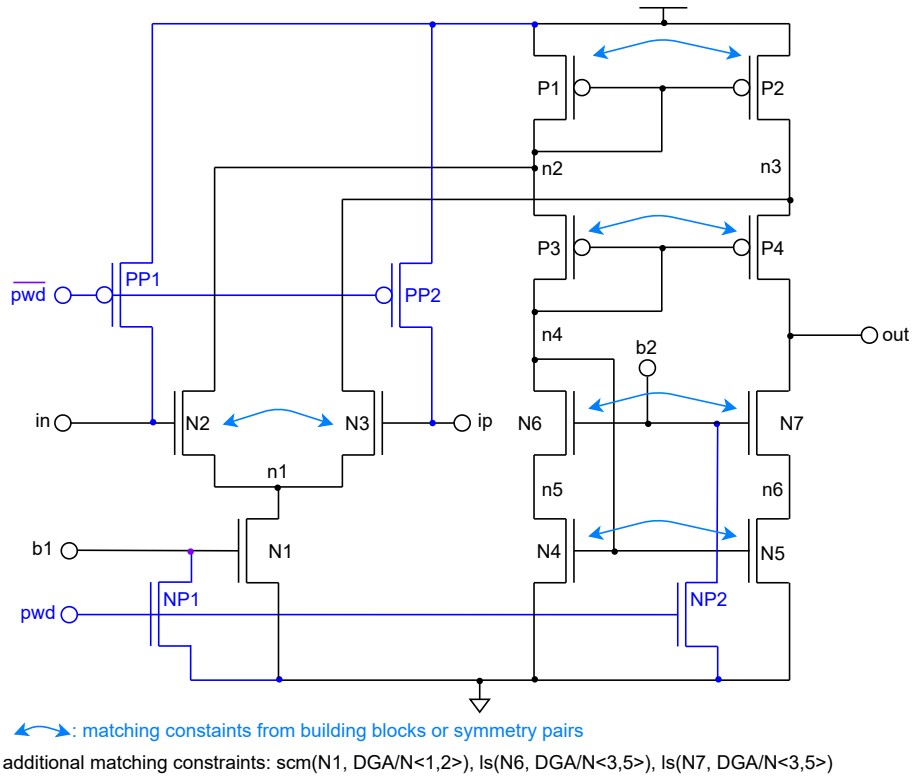
Furthermore, power-down transistors which can be removed from the power-down circuitry whilst maintaining its functionality can be identified as follows: a power-down transistor is



Step 2: no changes

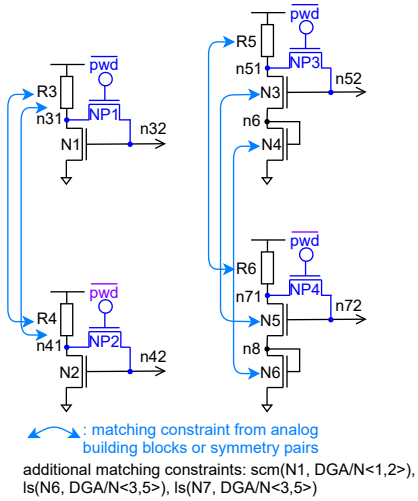


Step 2: no changes

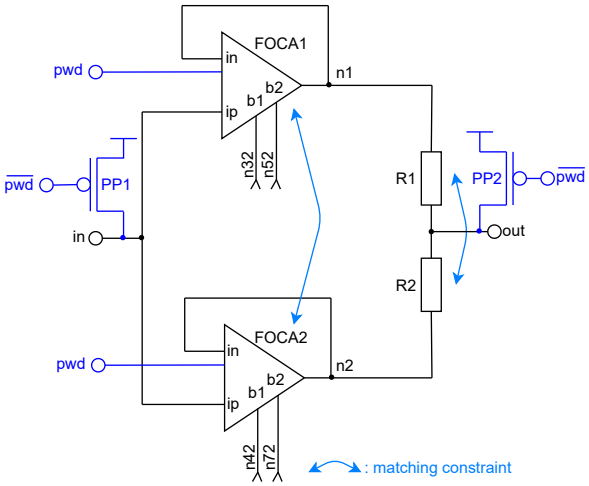


Step 2: no changes

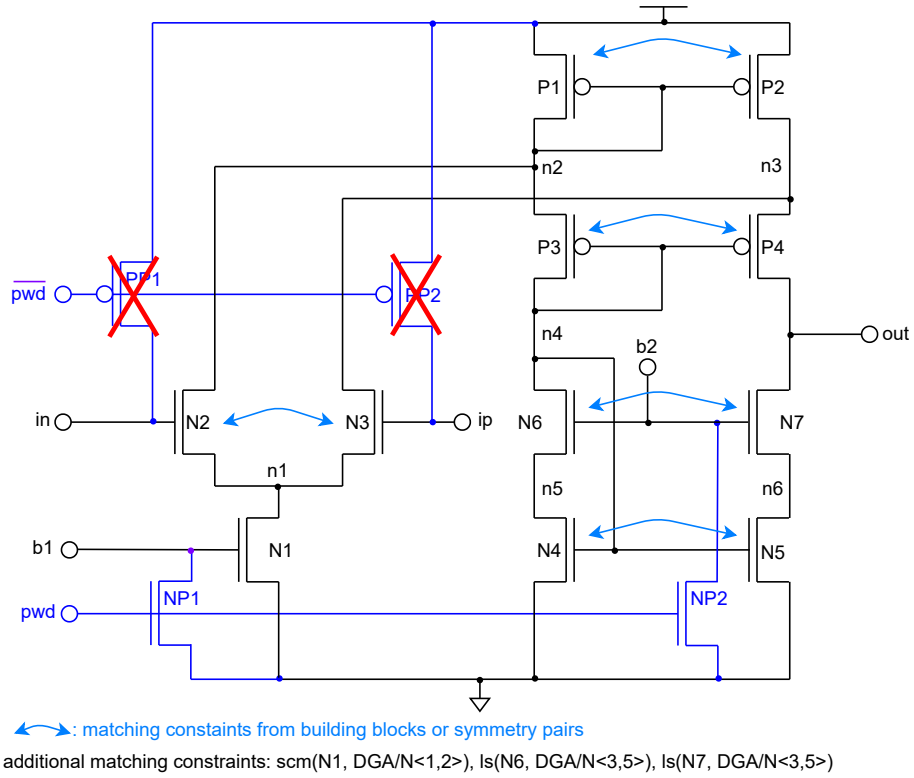
Figure C.2.: Measures taken to resolve matching and symmetry violations in the dual gain amplifier and its folded cascode amplifier subcircuits from Figs. 3.6 and 3.7.



Step 3: no changes



Step 3: insertion of $PP1$ and $PP2$



Step 3: removal of $PP1$ and $PP2$

Figure C.3.: Measures taken to reduce the area of the power-down circuitry of the dual gain amplifier and its folded cascode amplifier subcircuits from Figs. 3.6 and 3.7.

redundant at net n of the circuit, if there exists another conducting path from n to the supply or ground net of the circuit which does not run over the examined power-down transistor, i.e., *gnd* or *vdd* would inherently propagate to n.

In the dual gain amplifier from Fig. C.2, the input pins *ip* of the folded cascode amplifiers $FOCA1$ and $FOCA2$ are connected by net *in*. Hence, pulling this net to *vdd* by a power-down transistor allows to remove $PP2$ from the folded cascode amplifiers. Furthermore, the pins *in* and *out* of $FOCA1$ and $FOCA2$ are connected with each other over the resistors $R1$ and $R2$. Hence, inserting a power-down transistor at the output pin of the dual gain amplifier allows the supply voltage to propagate to the afore-mentioned pins of the folded cascode amplifiers. Hence, $PP1$ can be removed again from $FOCA1$ and $FOCA2$. Fig. C.3 illustrates the above taken measures. Overall, two power-down transistors can be removed from the hierarchical design whilst still maintaining its functionality.

List of Figures

1.1. Simplified block diagram of a wireless speaker. 3
1.2. Schematics of a simple digital (a) and a simple analog circuit (b). 4
1.3. Two digital power-saving techniques. 5
1.4. Basic principle of the analog power-down mode (a) and the source circuit from Fig. 1.2b with its power-down circuitry (b) highlighted in blue. 6
1.5. Power-down mode implementation variants considering device matching. . . 6
1.6. Shut-off patterns used for power-down mode implementation [11]. 8
1.7. Potential current flows in a source circuit (a), faulty power-down mode implementation for a source circuit series (b). 10
1.8. Data flows of the state-of-the-art (a) and the proposed power-down verification and power-down synthesis method (b) for hierarchical analog circuits. The diagram on the right (c) shows the data flow for the new structure recognition and symmetry computation methods. 11

2.1. Different device types and their corresponding pins. 15

3.1. Overview of the hierarchical power-down verification method. 18
3.2. Static DC behavior for different device types. 19
3.3. Overview of the symmetry assertion procedure [13]. 25
3.4. Differential stage before (a) and after (b) power-down up transformation. . . 26
3.5. Matching rules for: (a) analog basic building blocks [10], (b) symmetry pairs in signal paths and (c) symmetrical pin pairs of subcircuit blocks due to underlying signal path symmetries. 27
3.6. Bias circuitry (a) and core circuit (b) of a dual gain amplifier (DGA) with its power-down circuitry highlighted in blue. The subcircuits $FOCA1$ and $FOCA2$ are each implemented as the folded cascode amplifier from Fig. 3.7. 29
3.7. Schematic of a folded cascode amplifier (FOCA) which implements subcircuits $FOCA1$ and $FOCA2$ of the dual gain amplifier from Fig. 3.6. Its power-down circuitry is highlighted in blue. 30
3.8. Symmetry and matching constraints of the dual gain amplifier from Fig. 3.6. 32
3.9. Symmetry and matching constraints for $FOCA1$ and $FOCA2$ from Fig. 3.7. 33
3.10. Schematic of a high input impedance differential amplifier (HIIDA) [38]. The Miller operational transconductance amplifier (OTA) implements its subcircuits $MILLER1, MILLER2$ and $MILLER3$ (b). Their power-down circuitry is highlighted in blue. 37
3.11. High input impedance differential amplifier (HIIDA) [38] and its Miller operational transconductance amplifier (OTA) subcircuit blocks after power-up transformation. 39

4.1. Overview of the power-down synthesis method. 44

4.2. Schematic of a differential stage series ($SDFS$). Subcircuits $DFS1$ and $DFS2$ each implement one differential stage. The definite short circuit path $P1$, detected by voltage propagation and short-circuit path computation, is highlighted in red. . . . 46
4.3. Ripped-up schematic of the differential series from Fig. 4.2. . . . 46
4.4. Final synthesized power-down circuitry for the differential stage series from Fig. 4.3. . . . 59
4.5. Analysis and synthesis results for a current mirror operational amplifier [40]. 64
4.6. Basic building blocks and symmetry pairs of a low resistance operational amplifier [41]. . . . 66
4.7. Power-down synthesis results for the low resistance operational amplifier from Fig. 4.6. . . . 67
4.8. Analysis and synthesis results for a fully differential operational amplifier [42]. 68
4.9. Symmetry and matching constraints of the dual gain amplifier from Fig. 3.6. 69
4.10. Symmetry and matching constraints for $FOCA1$ and $FOCA2$ from Fig. 4.9. 70
4.11. Bias circuitry of the dual gain amplifier from Fig. 4.9a after rip-up and gate shut-off. . . . 70
4.12. Valid power-down mode implementation variants for the dual gain amplifier (DGA) and folded cacscode amplifier (FOCA) from Tables 4.15 and 4.16, and their corresponding flat and hierarchical scores. . . . 72
4.13. Basic building blocks and symmetry pairs of a high input impedance differential amplifier (HIIDA) [38] and its Miller operational transconductance amplifier (OTA) subcircuit blocks. . . . 73
4.14. Synthesis results for the high input impedance differential amplifier (HIIDA) and its Miller operational transconductance amplifiers (OTAs). . . . 74

5.1. Excerpt of the basic building block library from App. A. . . . 78
5.2. Differential input stage and its generated building blocks. . . . 79
5.3. Generated building block library for the differential input stage from Fig. 5.2. 80
5.4. Characteristics and group strength of a simple current mirror. . . . 82
5.5. Schematic of a folded cascode amplifier with different current mirror loads. . 84
5.6. Generated building block library for the six folded cascode amplifier variants from Fig. 5.6. . . . 86
5.7. Five different level shifter topologies [47; 48; 49; 50]. . . . 88
5.8. Level converter library generated by Algorithms 7 - 9. . . . 89

A.1. Dual gain amplifier from Fig. 3.6 without power-down circuitry. . . . 94
A.2. Schematic of a folded cascode amplifier which implements subcircuits $FOCA1$ and $FOCA2$ of the dual gain amplifier from Fig. A.1. . . . 94
A.3. Analog basic building block library $L_{analog} = L_{array} \cup L_{pair}$ based on [4; 51; 10]. 95
A.4. Overview of the hierarchical structure recognition method. . . . 97
A.5. Partially flattened dual gain amplifier from Fig. A.1. . . . 99
A.6. Partially flattened folded cascode amplifier from Fig. A.2. . . . 99
A.7. Structure recognition results for the bias circuitry of the dual gain amplifier (DGA) from Fig. A.5a. . . . 106
A.8. Structure recognition results of rank zero and rank one for the folded cascode amplifier (FOCA) from Fig. A.6. . . . 107
A.9. Structure recognition results of rank two and final results for the folded cascode amplifier (FOCA) from Fig. A.6. . . . 108

B.1. Overview of the hierarchical symmetry computation method. 110
B.2. Signals flow graph library L_{ESFG} for the analog basic building blocks of Fig. A.3 according to [17]. 112
B.3. Detailed signal flow graph and corresponding edge attributes of the folded cascode amplifier (FOCA) from Fig. A.6. The structure recognition results used for signal flow graph generation are shown in Fig. A.9. 113
B.4. Abstract signal flow graph $G_{A,FOCA}$ of the folded cascode amplifier from Fig. A.6. 114
B.5. Signal flow graph and edge attributes of the dual gain amplifier from Fig. A.5. The structure recognition results for signal flow graph generation of the bias circuitry are shown in Fig. A.7. The abstract signal flow graph for $FOCA1$ and $FOCA2$ is shown in Fig. B.4. 115
B.6. Different node and edge types encountered during symmetry computation. Symmetry conditions are indicated by blue arrows. 118
B.7. A current mirror bank and its corresponding signal flow graph. A unique edge mapping does not exist. 120
B.8. Symmetry computation on the graph model of the folded cascode amplifier from Fig. B.3. Each iteration shows intermediate steps of Algorithms 21 - 26. 123
B.9. Symmetry computation results for the bias circuitry (a) and the core circuit (b) of the dual gain amplifier from Fig. A.5. 125
B.10. Symmetry computation results for the folded cascode amplifier from Fig. A.6. 126
B.11. Ambiguities and conversion paths in a signal flow graph. 126
B.12. Multiple symmetry axes in a graph. 127

C.1. Measures taken to fix the short circuit paths of the dual gain amplifier and its folded cascode amplifier subcircuits from Figs. 3.6 and 3.7. 130
C.2. Measures taken to resolve matching and symmetry violations in the dual gain amplifier and its folded cascode amplifier subcircuits from Figs. 3.6 and 3.7. . 132
C.3. Measures taken to reduce the area of the power-down circuitry of the dual gain amplifier and its folded cascode amplifier subcircuits from Figs. 3.6 and 3.7. . 133

List of Tables

2.1. Set of circuits $\mathcal{C}$ used to construct the source circuit series from Fig. 1.7. . . 16

3.1. Voltage propagation results for the source circuit series (SCS) of Fig. 1.7b. . 21
3.2. Voltage propagation results for the source circuits $SC1$ and $SC2$ from Fig. 1.7a. 21
3.3. Matching and symmetry assertion results for the differential stage from Fig. 3.4a. 28
3.4. Voltage propagation results for the dual gain amplifier from Fig. 3.6. 29
3.5. Voltage propagation results for the folded cascode amplifiers from Fig. 3.7. The results are valid for $FOCA1$ and $FOCA2$ of the dual gain amplifier from Fig. 3.6 as these subcircuit are exposed to the same pin voltages. 30
3.6. Summary of the matching and symmetry assertion results for the dual gain amplifier (DGA) and its folded cascode amplifier subcircuits $FOCA1$ and $FOCA2$ from Fig. 3.6 and 3.7. 34
3.7. Matching assertion results for the dual gain amplifier from Fig. 3.6. 34
3.8. Symmetry assertion results for the dual gain amplifier from Fig. 3.6. 35
3.9. Matching assertion results for $FOCA1$ and $FOCA2$ from Fig. 3.7. 36
3.10. Symmetry assertion results for $FOCA1$ and $FOCA2$ from Fig. 3.7. 37
3.11. Final voltage propagation result for the high input impedance differential amplifier (HIIDA) and the Miller operational transconductance amplifiers (OTAs) from Fig. 3.10. 38
3.12. Matching assertion results for the Miller operational transconductance amplifiers $MILLER1 - MILLER3$ from Fig. 3.10b. 39
3.13. Symmetry assertion result for the high input impedance differential amplifier from Fig. 3.10a. 40
3.14. Symmetry assertion results for the Miller operational transconductance amplifiers from Fig. 3.10b. 40

4.1. Voltage propagation result for the ripped-up differential stage series from Fig. 4.3. 55
4.2. Constraint optimization problem from eq. (4.15) for the differential stage series from Fig. 4.3. 55
4.3. Solutions of the constraint optimization problem from Table 4.2. The power-down transistors inserted by the $boundaryConditions(\mathbf{v})$ constraint at nets $outn$, $n1$ and pwd have already been removed from this table and reverted to gnd, vdd, vdd, respectively. 56
4.4. Voltage propagation results of differential stage $DFS1$ for solution $\mathbf{v}_2^*$ of the differential stage series. 56
4.5. Voltage propagation results for differential stage $DFS1$ based on solution $\mathbf{v}_1^*$ of the differential stage series. 57
4.6. Voltage propagation results for differential stage $DFS2$ based on solution $\mathbf{v}_1^*$ of the differential stage series. 57
4.7. Constraint optimization problem from eq. (4.15) for differential stage $DFS2'$ and solution $\mathbf{v}_1^*$ of the differential stage series from Fig. 4.3. 58

4.8. Solutions of the constraint optimization problem from Table 4.7. The power-down transistors inserted by the boundary conditions have already been reverted to *vdd* or *gnd*, respectively. 58
4.9. Valid solutions of the gate shut-off COP for the differential stage series from Fig. 4.3 if the symmetry and matching information is not propagated between the bottom and the top-level of the circuit hierarchy. The solutions for differential stages $DFS1$ and $DFS2'$ are shown in Tables 4.10 and 4.11, respectively. 60
4.10. Power-down mode implementation variants for differential stage $DFS1$ required to implement the solutions of the differential stage series shown in Table 4.9. 61
4.11. Power-down mode implementation variants for differential stage $DFS2'$ required to implement the solutions of the differential stage series shown in Table 4.9. 61
4.12. Flat and hierarchical score for the power-down mode implementation variants of the differential stage series from Table 4.9. 62
4.13. Power-down synthesis results of the new method from Chap. 4 and the approach by [10] for the current mirror operational amplifier. 65
4.14. Comparison of the building block matching results for the current mirror operational amplifier for the new synthesis method from Chap. 4 and the approach by [10]. 65
4.15. Solutions of the gate shut-off COP for the dual gain amplifier for its core circuit and its ripped-up bias circuits from Figs. 3.8b and 4.11. 71
4.16. Synthesized implementation variants for the folded cascode amplifiers $FOCA1$ and $FOCA2$ from Fig. 3.9. 71

5.1. Weights of the tentative buildings formed in the first iteration of Algorithm 7 for the circuit of Fig. 5.2. 83

Bibliography

[1] M. Horowitz, T. Indermaur, and R. Gonzalez, "Low-power digital design," in *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, 1994, pp. 8–11.

[2] R. Gonzalez, B. Gordon, and M. Horowitz, "Supply and threshold voltage scaling for low power cmos," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 8, pp. 1210–1216, 1997.

[3] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai, "A 0.9-v, 150-mhz, 10-mw, 4 mm/sup 2/, 2-d discrete cosine transform core processor with variable threshold-voltage (vt) scheme," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1770–1779, 1996.

[4] T. Massier, H. Graeb, and U. Schlichtmann, "The sizing rules method for cmos and bipolar analog integrated circuit synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 12, pp. 2209–2222, 2008.

[5] E. Maricau and G. Gielen, "Transistor aging-induced degradation of analog circuits: Impact analysis and design guidelines," in *2011 Proceedings of the ESSCIRC (ESSCIRC)*, Sep. 2011, pp. 243–246.

[6] C. Michael, H. Wang, C. S. Teng, J. Shibley, L. Lewicki, C.-M. Shyu, and R. Lahri, "Mismatch drift: a reliability issue for analog MOS circuits," in *30th Annual Proceedings Reliability Physics 1992*, March 1992, pp. 81–84.

[7] Y. Chen, J. Zhou, S. Tedja, Frank Hui, and A. S. Oates, "Stress-induced mosfet mismatch for analog circuits," in *2001 IEEE International Integrated Reliability Workshop. Final Report (Cat. No.01TH8580)*, 2001, pp. 41–43.

[8] S. Blieck and E. Janssens, "Software check for power-down mode of analog circuits," in *ESSCIRC '96: Proceedings of the 22nd European Solid-State Circuits Conference*, 1996, pp. 404–407.

[9] M. Zwerger and H. Graeb, "Short-circuit-path and floating-node verification of analog circuits in power-down mode," in *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2012, pp. 241–244.

[10] M. Zwerger, "Verification and synthesis of analog power-down circuits," Ph.D. dissertation, Technical University of Munich, 2017.

[11] M. Zwerger, M. Neuner, and H. Graeb, "Power-down circuit synthesis for analog/mixed-signal," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 656–663.

[12] ——, "Analog power-down synthesis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 12, pp. 1954–1967, 2017.

[13] M. Zwerger and H. Graeb, "Detection of asymmetric aging-critical voltage conditions in analog power-down mode," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2015, pp. 1269–1272.

[14] M. Zwerger, G. Shrivastava, and H. Graeb, "Power-down synthesis for analog circuits including switch sizing," in *2016 13th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2016, pp. 1–4.

[15] M. Neuner, M. Zwerger, and H. Graeb, "Power-down schematic synthesis for analog/mixed-signal circuits," in *ANALOG 2016; 15. ITG/GMM-Symposium*, 2016, pp. 1–6.

[16] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich, "The sizing rules method for analog integrated circuit design," in *IEEE/ACM International Conference on Computer Aided Design. ICCAD 2001. IEEE/ACM Digest of Technical Papers (Cat. No.01CH37281)*, 2001, pp. 343–349.

[17] M. Eick, "Structure and signal path analysis for analog and digital circuits," Ph.D. dissertation, Technical University of Munich, 2013.

[18] M. Zwerger, P. Vlachas, and H. Graeb, "A fast analytical approach for static power-down mode analysis," in *2015 IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, 2015, pp. 1–4.

[19] R. Barták, "Constraint programming: In pursuit of the holy grail," in *Week of Doctoral Students (WDS99)*, June 1999, pp. 1–4.

[20] G. T. C. Schulte and M. Z. Lagerkvist, *Modeling and Programming with Gecode*, 6th ed., www.gecode.org, June 2020.

[21] Cadence Design Systems, Inc, https://www.cadence.com/, 2020.

[22] J. DeLisle, "What is analog ic design?" All About Circuits, Tech. Rep., Dec. 2020, online; accessed 18.03.202021. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/what-is-analog-ic-design/

[23] D. Sridharan, "Automatic structural analysis for hierarchical designs," Master's thesis, Technical University of Munich, 2013.

[24] K. Lu, "Symmetry Recognition for Automatic Sizing of Analog Integrated Circuits," Master's thesis, Technische Universität München, Sep. 2009.

[25] M. Neuner and H. Graeb, "Power-down mode verification for hierarchical analog circuits," in *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2019, pp. 125–128.

[26] ——, "Verification and revision of the power-down mode for hierarchical analog circuits," *Integration*, vol. 73, pp. 1–9, 2020.

[27] ——, "Synergetic algorithm for power-down synthesis," in *2020 European Conference on Circuit Theory and Design (ECCTD)*, 2020, pp. 1–5.

[28] ——, "Hierarchical analog power-down synthesis," in *2020 IEEE International Conference on Electronics Circuits and Systems (ICECS)*, 2020.

[29] M. Neuner, I. Abel, and H. Graeb, "Library-free Structure Recognition for Analog Circuits," in *Design, Automation and Test in Europe (DATE)*, Feb. 2021.

[30] I. Abel, "Eine Datenstruktur zur Identifikation von Funktionsblöcken in analogen Schaltungen," Bachelor Thesis, Technical University of Munich, Chair of Electronic Design Automation, Tech. Rep., 2016.

[31] ——, "Automatische Regelgenerierung zur Erkennung von Schaltungsstrukturen," Research Internship, Technical University of Munich, Chair of Electronic Design Automation, Tech. Rep., Mar. 2018.

[32] E. Haardt, "Power-Down-Verifikation hierarchischer analoger Schaltungen," Jan. 2019.

[33] ——, "Sizing Constraints for Rail-to-Rail Amplifiers," Master's thesis, Oct. 2020.

[34] A. Dangi, "Framework for Symmetry Computation in Analog Circuits," DAAD WISE internship report, Technical University of Munich, Chair of Electronic Design Automation, Tech. Rep., 2017.

[35] S. Muralitharan, "Automatic Symmetry Constraint Generation and their Application to Wicked," DAAD WISE internship report, Technical University of Munich, Chair of Electronic Design Automation, Tech. Rep., 2018.

[36] E. Shi, "ESD Verification at Circuit Level based on Structural Analysis," TUM PREP internship report, Technical University of Munich, Chair of Electronic Design Automation, Tech. Rep., 2018.

[37] Analog Devices, "Paralleling amplifiers increases output drive," http://www.surfline.com/surf-news/maldives-surf-access-controversy-update_75296/, 2020, online; accessed 23.11.2020.

[38] G. B. Clayton, *Operational Amplifiers*, 5th ed. Newnes, Aug 2003.

[39] T. W. Francesca Rossi, Peter van Beek, *Handbook of Constraint Programming*, 2nd ed. Elsevier, 2006, vol. Volume 2.

[40] K. M. T. C. Carusone, D. Johns, *Analog Integrated Circuit Design.* New Jersey, USA: John Wiley & Sons, Inc., 2011.

[41] P. E. Allen and D. R. Holberg, *CMOS Analog Circuit Design*, 3rd ed. Oxford University Press, 2012.

[42] Y. Wenger and B. Meinerzhagen, "Implementation of a fully-differential operational amplifier with wide-input range, high-impedance common-mode feedback," in *PRIME*, July 2019, pp. 1–4.

[43] H. Hung, M. Ker, S. Chen, and C. Chuang, "Abnormal esd damages occur in interface circuits between different power domains in nd-mode mm esd stress," in *IPFA*, Jul 2006, pp. 163–166.

[44] H. Li, F. Jiao, and A. Doboli, "Analog circuit topological feature extraction with unsupervised learning of new sub-structures," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 1509–1512.

[45] K. Settaluri and E. Fallon, "Fully automated analog sub-circuit clustering with graph convolutional neural networks," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 1714–1715.

[46] K. Kunal, T. Dhar, M. Madhusudan, J. Poojary, A. Sharma, W. Xu, S. M. Burns, J. Hu, R. Harjani, and S. S. Sapatnekar, "Gana: Graph convolutional network based automated netlist annotation for analog circuits," in *2020 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2020, pp. 55–60.

[47] M. Lanuzza, P. Corsonello, and S. Perri, "Low-power level shifter for multi-supply voltage designs," *TCSII*, vol. 59, no. 12, pp. 922–926, Dec 2012.

[48] S. H. Kulkarni and D. Sylvester, "High performance level conversion for dual v/sub dd/ design," *TVLSI*, vol. 12, no. 9, pp. 926–936, Aug 2004.

[49] Q. A. Khan, S. K. Wadhwa, and K. Misri, "A single supply level shifter for multi-voltage systems," in *VLSID*, Jan 2006, pp. 4 pp.–.

[50] R. Garg, G. Mallarapu, and S. P. Khatri, "A single-supply true voltage level shifter," in *DATE*, Apr 2008, pp. 979–984.

[51] M. Eick and H. Graeb, "A versatile structural analysis method for analog, digital and mixed-signal circuits," in *2012 International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2012, pp. 1–4.

[52] H. Liu, S. Li, Z. Wu, and L. Li, "Vector coding method for symmetry detection in the analog circuits," in *2020 IEEE 14th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 2020, pp. 82–86.

[53] M. E. Kole, J. Smit, and O. E. Herrmann, "Modeling symmetry in analog electronic

circuits," in *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*, vol. 1, 1994, pp. 315–318 vol.1.

[54] Qingsheng Hao, Song Chen, Xianlong Hong, Yi Su, Sheqin Dong, and Zhiyi Qu, "Constraints generation for analog circuits layout," in *2004 International Conference on Communications, Circuits and Systems (IEEE Cat. No.04EX914)*, vol. 2, 2004, pp. 1334–1338 Vol.2.

[55] Sambuddha Bhattacharya, N. Jangkrajarng, Roy Hartono, and C. . R. Shi, "Hierarchical extraction and verification of symmetry constraints for analog layout automation," in *ASP-DAC 2004: Asia and South Pacific Design Automation Conference 2004 (IEEE Cat. No.04EX753)*, 2004, pp. 400–405.

[56] MunEDA GmbH, "WiCkeD," http://www.muneda.com/, 2020.

[57] M. Eick and H. E. Graeb, "Mars: Matching-driven analog sizing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 31, no. 8, pp. 1145–1158, 2012.

[58] E. Charbon, E. Malavasi, and A. Sangiovanni-Vincentelli, "Generalized constraint generation for analog circuit design," in *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, 1993, pp. 408–414.

[59] M. Eick, M. Strasser, K. Lu, U. Schlichtmann, and H. E. Graeb, "Comprehensive generation of hierarchical placement rules for analog integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 2, pp. 180–193, 2011.

[60] B. G. Arsintescu, "A method for analog circuits visualization," in *Proceedings International Conference on Computer Design. VLSI in Computers and Processors*, 1996, pp. 454–459.

[61] B. G. Arsintescu and S. A. Spanoche, "Global stacking for analog circuits," in *Proceedings EURO-DAC '96. European Design Automation Conference with EURO-VHDL '96 and Exhibition*, 1996, pp. 392–397.

[62] S. Y. Su Yi, S. D. Sheqin Dong, Q. H. Qingsheng Hao, X. H. Xiangqing He, and X. H. Xianlong Hong, "Automated analog circuits symmetrical layout constraint extraction by partition," in *ASIC, 2003. Proceedings. 5th International Conference on*, vol. 1, 2003, pp. 166–169 Vol.1.

[63] Zhe Zhou, Sheqin Dong, Xianlong Hong, Qingsheng Hao, and Song Chen, "Analog constraints extraction based on the signal flow analysis," in *2005 6th International Conference on ASIC*, vol. 2, 2005, pp. 825–828.

[64] C. Ebeling, "Geminiii: a second generation layout validation program," in *[1988] IEEE International Conference on Computer-Aided Design (ICCAD-89) Digest of Technical Papers*, 1988, pp. 322–325.

[65] S. Bhattacharya, N. Jangkrajarng, and C. . R. Shi, "Multilevel symmetry-constraint generation for retargeting large analog layouts," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 945–960, 2006.

[66] K. Kunal, J. Poojary, T. Dhar, M. Madhusudan, R. Harjani, and S. S. Sapatnekar, "A general approach for identifying hierarchical symmetry constraints for analog circuit layout," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–8.

[67] M. Liu, W. Li, K. Zhu, B. Xu, Y. Lin, L. Shen, X. Tang, N. Sun, and D. Z. Pan, "S3det: Detecting system symmetry constraints for analog circuits with graph similarity," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020, pp. 193–198.

[68] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Addison-Wesley Professional, 2011.