Jakob Smonig, BSc

Automated Physical Design Optimization

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree program: Electrical Engineering and Business

submitted to

Graz University of Technology

Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Peter Söser

Institute of Electronics

Supervisor - Company

Dennis Jeurissen

NXP Semiconductors Austria

Graz, June 2019

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date

Signature

Acknowledgement

My special thanks go to Dennis Jeurissen who not only gave me the opportunity to work at NXP semiconductors during my studies, but also made this thesis as a cooperation of Graz University of Technology together with NXP possible. Furthermore, he guided me through the work and was there if support was needed.

Especially, I would like to express my gratitude to Ass.Prof. Dr.techn. Peter Söser who supervised the thesis as part of the Institute of Electronics. He gave me the freedom to develop the best outcome and helped me through the organizational procedures at the university.

Furthermore, I want to thank Dipl.-Ing. Andrea Friedrich for being there for me and supporting me during this demanding period of time.

Also, I would like to thank Javier Mauricio Velandia Torres for his support on the divide-by-two circuit and the simulation setup required for it.

Last but not least I want to thank my parents for enabling my studies at the Graz University of Technology and their backup during them.

Abbreviations

ADC	Analog to Digital Converter
ADE	Analog Design Environment
BFGS	Broyden–Fletcher–Goldfarb–Shanno
C	Capacitor
CDF	Component Description Format
CDL	Circuit Design Language
CEL	CDF Expression Language
CIW	Command Interpreter Window
CMOS	Complementary Metal-Oxide-Semiconductor
DC	Direct Current
DRC	Design Rule Check
DSPF	Detailed Standard Parasitic Format
GDS	Graphic Database System
GUI	Graphical User Interface
IC	Integrated Circuit
IP	Intellectual Property
LISP	LISt Processing
LPF	Low Pass Filter
LRC	Inductance, Resistor and Capacitor
LVS	Layout Vs Schematic
NFET	N-Channel Field Effect Transistor
PC	Phase Comparator
PCell	Parameterizable Cell
PFET	P-Channel Field Effect Transistor
PLL	Phase Locked Loop
PVS	Physical Verification System
R	Resistor
RC	Resistor and Capacitor
RF	Radio Frequency
ROD	Relative Object Design
ss	Slow-Slow
STI	Shallow Trench Isolation
tt	Typical-Typical
VCO	Voltage Controlled Oscillator

Abstract

With shrinking technology nodes and the usage of higher frequencies the influence of parasitic components on the circuit performance and thus its importance during analog circuit design continuously increases.
Additionally, with the complex and expensive small node processes in combination with tight functional constraints it is necessary to obtain the best performance possible of the used IC area. Therefore, critical circuits are required to be optimized in order to satisfy the strict constraints.

This leads to the requirement of tools and methods to cope with this challenge and the need of state of the art simulators and optimization tools.
Currently, optimizers solely for the schematic level are available. Those are applied to improve the performance of various circuits. The issue thereby is, that only the circuit level and no parasitic influences are considered. Due to the structure of the physical layers of the IC, parasitic capacitors and resistors exist which heavily influence the performance of RF circuits. Therefore, there is a need to include these influences in the optimization algorithm in order to get valid results.

To extend the optimization also to the physical design of the circuit, the existing ADE optimizer is applied. During each optimization cycle a pre-run script is executed which performs a full circuit extraction and feeds the results in the netlist used for the simulation. This provides the possibility to approximate the theoretically possible best circuit performance and to ideally utilize the required IC area.

The method is verified by optimizing a divide-by-two circuit operating in the GHz range. The results showed the limits of the schematic only optimization due the significant influences of the parasitics at high frequencies.

In this thesis, the physical design optimization was proven to yield the best output performance and its practical applicability was demonstrated. Even though, the computational resources required for the optimization increase, it is reasonable to enhance the performance of critical RF circuits.

Table of contents

Acknowledgement ..... **II**
Abbreviations ..... **III**
Abstract ..... **IV**
1 Introduction **1**
1.1 Problem Description 1
1.2 Parasitic Components 3
1.3 Aim of the Thesis 6
2 Optimization **10**
2.1 Introduction 10
2.2 Local Optimization Algorithms 12
2.2.1 Gradient Based Algorithms 13
2.2.2 Derivative Free Algorithms 15
2.3 Global Optimization 16
3 Tools and Current Way of Working **18**
3.1 Current Way of Working 18
3.2 Testbench 21
3.3 Virtuoso Analog Design Environment (ADE) 23
3.3.1 ADE Settings 24
3.3.2 ADE Optimization Flow 26
3.3.3 ADE Optimization Manipulation 28
3.4 Parameterized Cell 30
3.4.1 PCell Structure 31
3.5 Quantus QRC Extraction Solution 33
3.5.1 Extraction Types 34
3.5.2 QRC Usage 35
4 Divide-by-two Circuit **36**
4.1 Schematic 37
4.2 Function 39
4.3 Layout 41
5 Physical Design Optimization **43**
5.1 Introduction 43
5.2 Graphical User Interface 46
5.3 Created Data 56
5.4 Output 57
6 Case Study: Divide-by-two Circuit **59**

6.1 Testbench 59
6.2 Constraints 60
6.2.1 Reference Point 62
6.3 Schematic only Optimization 63
6.3.1 Extraction and Simulation 66
6.4 Initial Layout Optimization 67
6.4.1 Extraction and Simulation 69
6.5 Layout Optimization 70
7 Conclusion and Outlook 73
List of Figures 74
List of Tables 76
References 77

1 Introduction

Analog IC design is a complex task that requires extensive design expertise and holds numerous challenges. For instance, that the physical design of the circuit introduces significant performance differences in comparison to the schematic only simulation due the parasitic components. Therefore, the IC designers require tools which facilitate the circuit design, like the optimization or the extraction of the circuit.

At the beginning of this chapter, the fundamental challenge of design optimization is explained. This topic is addressed in the thesis. Secondly, insights on the different parasitic components and how they influence the circuit are provided. Finally, the aim of this thesis and the required steps are described.

1.1 Problem Description

The general analog circuit design flow shown in Figure 1 displays the complete route from the IC specifications to the finished and verified physical design which is fabricated.
At the beginning, the specifications of the IC are defined and depending on these, proper technologies are analyzed. The best fitting process technology is chosen and the different circuit topologies are compared. After the topologies for the various top-level blocks are fixed, the design of the sub cells can be performed. Therefore, the circuit must be designed, the device parameters set accordingly and verified through simulations. In order to find the best device parameters, circuit optimizers are applied.
After the schematic was successfully verified, the corresponding physical representation is generated. The layout view introduces parasitic components and thus must be extracted and simulated to analyze how the circuit performance is affected. Dependent on the results of the final verification step, a circuit and hence layout redesign is required.
Analog circuit design lacks automated tools to explore the solution space and thus a significant effort from the designer is necessary since all steps must be performed manually. [Lourenço et al. 2015]

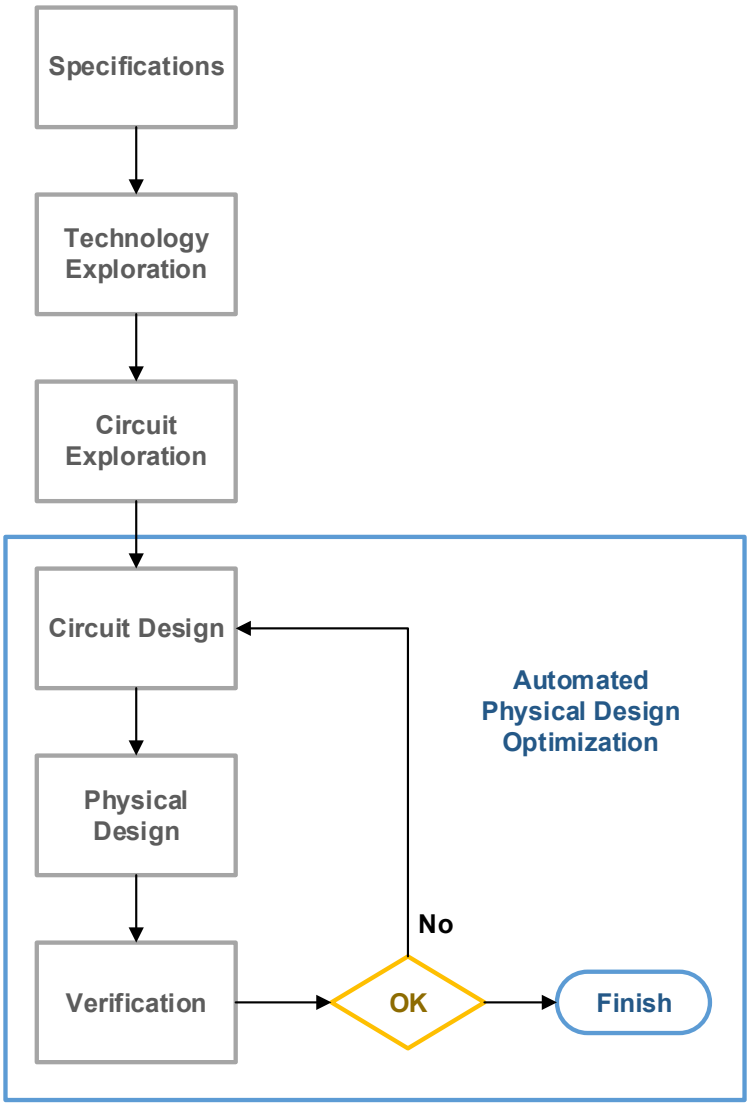

Figure 1: Analog circuit design flow

One challenge is, that the real circuit performance is only visible at the end of the completed design flow and hence redesigns create substantial additional work. It could be worthwhile to optimize the design including the layout, after having an good initial schematic optimized design. Therefore, instead of running the circuit optimization on the schematic level, excluding the parasitic influences introduced by the layout, it is beneficial to directly optimize the physical design. This assures to select the best device parameters to satisfy the circuit specification. Additionally, the verification step is already part of the device sizing and hence potential performance issues are detected earlier in the flow.

That enables more robust circuit design and avoids complex and time-consuming circuit redesigns. Therefore, the project expenses are limited and the required time to market is reduced.

1.2 Parasitic Components

If the physical design of a circuit is created and produced on silicon in the end, its performance is significantly influenced in comparison with the schematic only simulation. This is caused by the impact of all the parasitic components as resistors, capacitors or inductances which exist due the physical layer structure of the IC. It is impossible to completely avoid them, but there are measures in the layout possible to minimize their effect on the critical nets.
Therefore, for critical circuits it is of high importance that the designer is aware of this and takes these components into consideration during the design. An issue is, that the concrete parasitics are only visible after the first layout is created and if a performance issue is then found a complete redesign and new layout is necessary.

A transistor is built up of different layers of materials. Due to that, multiple parasitic capacitors, which can be seen in Figure 2, are formed. These occur since there are different voltages on two from each other insulated conductors. An example is the capacitor between gate and source C_{GS}. It is formed from the polysilicon which is set to the gate voltage and n-plus well which is on ground, or another terminal voltage. They are insulated from each other by the gate silicon oxide layer. This means that there are two conducting materials on different voltage levels separated by an insulator. This follows, that a capacity is formed which will influence the circuit behavior. Another example is the gate capacity which is the determining factor for the input capacity of the transistor. [Pandit et al. 2014]

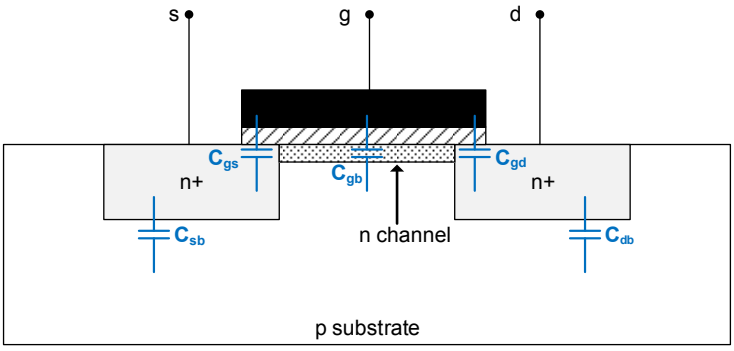

Figure 2: Parasitic NMOS capacitors

Since none of the used materials is an ideal conductor, each connection has a certain resistance which is not considered in the schematic. These can range from few milli Ohm to multiple Ohms depending on the used layer.

In general, the resistance is dependent on the material specific electrical resistivity ρ and its shape. For the layer structure of the IC the resistance can be calculated as seen in equation (1) whereby the l is its length and A the area.

$$R = \frac{\rho\, l}{A} = \frac{\rho\, l}{h\, w} \tag{1}$$

The semiconductor fabrication plants define the metal resistances as sheet resistances Rs since the height of the metal is fixed. This enables the user to simply multiply Rs with the length divided by the width of the metal connection to gain its resistance.

$$R = \frac{\rho}{h}\frac{l}{w} = R_S\frac{l}{w} \tag{2}$$

The typical sheet resistances of the different materials of a 1 µm CMOS process is shown in Table 1. If assumed that a metal wire is 8 µm long and has a width of 1 µm, its resistance is 56 mΩ. Depending of the used technology, the materials, the layer heights and thus their resistances are different. [Söser et al. 2008]

Table 1: Sheet resistances

Material	**Sheet resistances [Ω / □]**		
	Min.	**Typ.**	**Max.**
Metal	0.05	0.07	0.1
Polysilicon	15	20	30
Silicide	2	3	6
Diffusion (n+, p+)	10	25	100
N - Well	1 k	2 k	5 k

As it can be seen, the resistances of the interconnects can be important even though these are from a very large process. Therefore, it can be imagined that their influence increases with shrinking technology nodes.

For long minimal width metal connections its resistance can be significant and therefore it is necessary to verify the resistance of critical paths. Especially polysilicon which is used for the gate connection of the transistors has a high resistance. Therefore, it is recommended to change to metal1 or higher for longer connections.

Until now a simple connection of two devices on the same metal layer was analyzed. In reality, there will be connections to lower or upper level metals and to the devices in the substrate. Due the connections of different metal layers there are contacts required, whereby their resistances are a significant part of the complete wire connection.

A simplified example for the total wiring resistance in shown in Figure 3 where two devices are connected with a single first level metal stripe. The area is the height *h* times the width *w*. The total resistance of the connection consists of the two contact resistances and the sheet resistance of the metal. Since the contact resistances are a significant part it is recommended to use several contacts to minimize the total resistance.

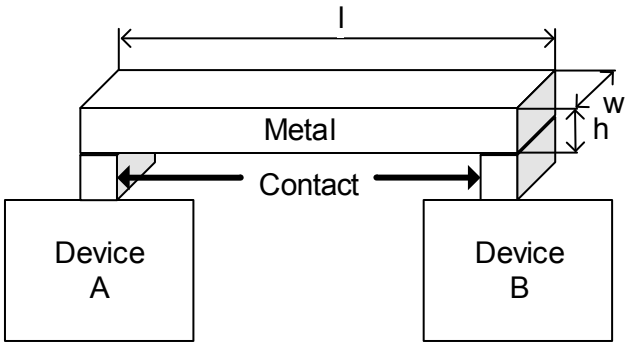

Figure 3: Example for metal resistance of wiring

For most circuits it is sufficient to extract the resistors and capacitors to verify the circuit performance. The parasitic inductances can also be extracted, but are typically not necessary since the simulation time increases significantly. However, for very high frequencies their influence on the circuit performance increases and thus it is recommended to consider them.
[Lampaert et al. 1991]

1.3 Aim of the Thesis

The goal of this master thesis is to build an automated physical design optimization flow which is generally applicable and user friendly. As it can be seen in Figure 4 it consists of the four main blocks, namely simulation, evaluation, calculation of the next point and netlisting.
The calculation of the next point for a local optimization uses a mathematical algorithm, which can be gradient or direct search based to create the new parameter set to fulfill the requirements. ADE provides two different algorithms for each type. Afterwards, the calculated parameters are used either to update the schematic or to update the PCell layout depending on the applied method.

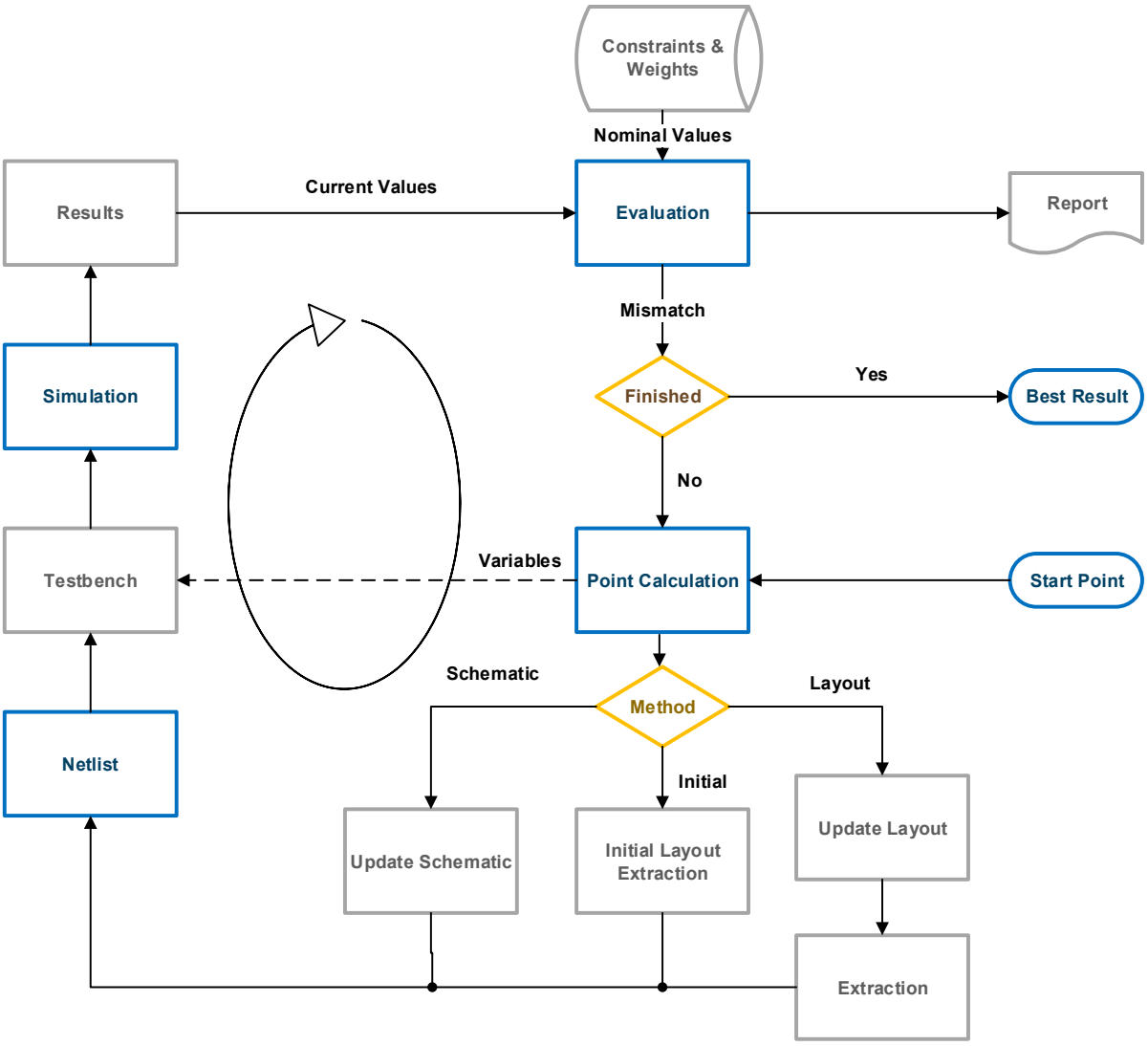

Figure 4: Optimization flow

If the schematic mode is selected only the netlist is updated which is then used for the further simulation. Since only some device parameters change and the devices as well as their connections stay the same it is possible to use variables to avoid renetlisting and to gain an additional speedup.

For the layout mode, a new layout view with the current parameter set is generated and afterwards used for the extraction of the parasitic components. The resulting extracted view is then applied for the netlisting and fed in the testbench.
This method needs more resources but has the great advantage that the parasitics are also considered. This allows to simulate the whole circuit with the calculated parameters to get precise results including all parasitic influences.

The initial mode is similar to the schematic mode, but with the difference that the parasitic capacitors of the initial layout are included in the netlist. This enables to have a first approximation of the parasitic influences in the simulations, even if only a first manually created layout exists.

The output of the netlisting phase contains all devices and their connectivity. The testbench models the environment in which the circuit is embedded and enables the verification of the design.

During the simulation phase, the existing testbench and netlist are used to perform the user-defined simulations. For the simulation various tools are available. In this thesis Mica was applied.

The output of the simulation is then used by the evaluation which compares the current obtained results with the desired nominal values from the specification. The specification is dependent on the user input regarding the output constraints and weights.
If the goal is reached the process ends and the results will be displayed. Elsewise, the next point is calculated and the whole circle starts again.

This complete automated flow must be set up and a case study on a high frequency divider is performed. This circuit was programmed as PCell to enable the physical design optimization. To increase the user friendliness a GUI to start the complete flow was created. Instead of running all tools manually one after another the user only has to define all necessary information in the GUI and afterwards the whole automated flow can be started.

At first a literature research was conducted to gain knowledge about the current state of the art and background knowledge. In addition, the current way of analog design optimization at NXP was analyzed. The next step was to set up the complete optimization flow with the current way of working for a specific design. This allowed to get specific insights of the used tools which were necessary for the automatization afterwards.

The PCell requires a Skill script which generates the cell for the given input parameter. Skill is the Virtuoso® design environment extension language which can be used to program various tools, procedures and complete PCells.
To build PCell circuits of higher complexity a library with basic components is essential. An example for such a low-level cell is a simple inverter which was built using ROD to generate the corresponding layout visible in Figure 5.

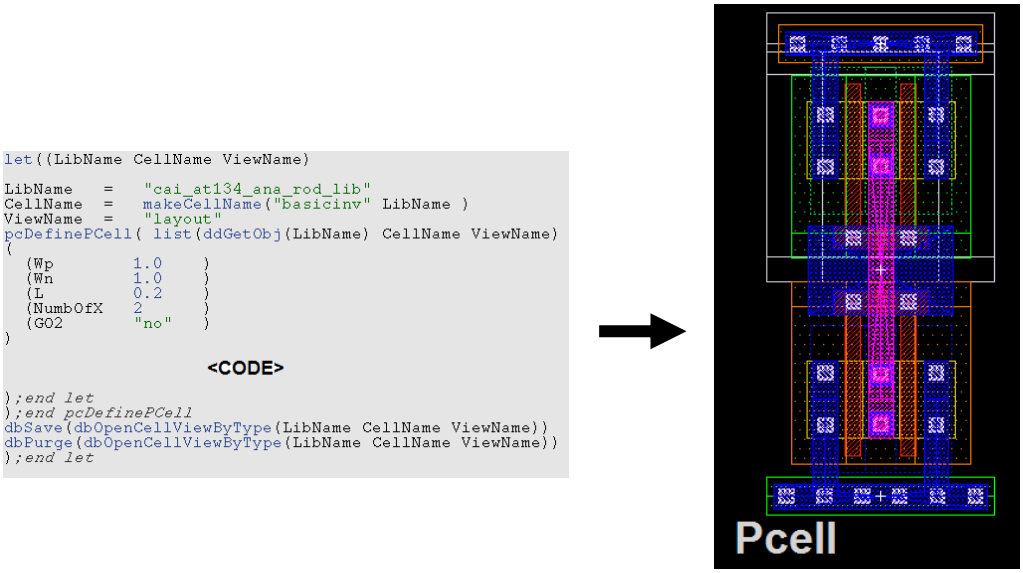

Figure 5: ROD PCell example

The next step was the automatization of the complete flow requiring a GUI for the user input. The GUI allows a broad usage range since all necessary parameters can be defined. The GUI provides individual tabs for each step of the optimizing flow. This allows its usage for various designs as shown in Figure 6.

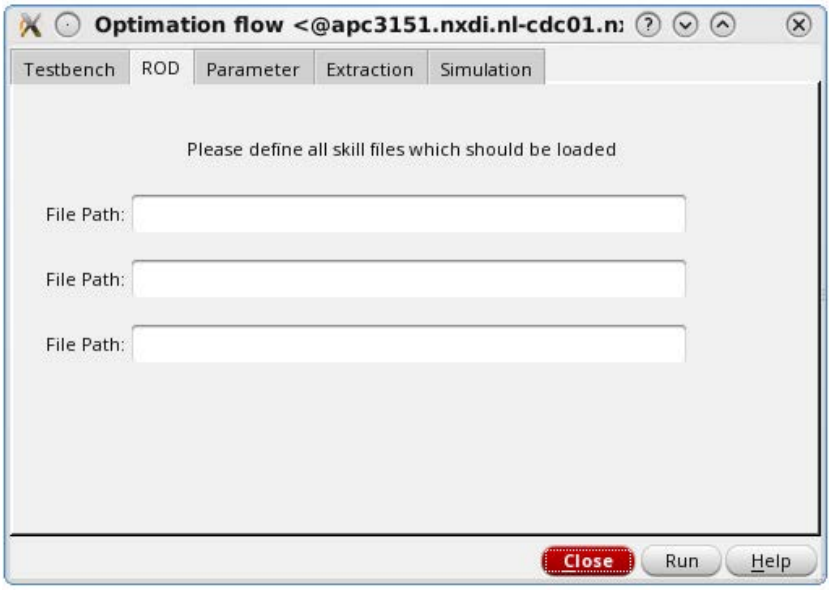

Figure 6: Virtuoso GUI example

After the completion of the whole automated flow it was combined in a cadenv package to allow simple setup and the usage in different projects. A cadenv package is an installation package which is under revision control and can be installed in the different project environments by the designers in NXP.

In addition, a documentation of the flow and its usage with examples was provided.

2 Optimization

Optimization is used in a wide range of fields to get the best result possible of certain outputs. This can mean to minimize the input while maximizing the output or to minimize the total risk related with an investment.

In this thesis different optimization algorithms are used to size circuit devices in such a way to get the best output performance possible.

2.1 Introduction

Optimization can be described as the action of making the best or most effective use of a situation or resource. [Oxforddictionaries]
In practice this is performed my minimizing a cost function in order to get the best overall result. A single objective optimization problem can be expressed in the following form, whereby f_0 is the objective respectively cost function.

$$min\, f_0(x) \qquad (3)$$
$$subject\ to\ f_i(x) < b_i \quad i = 1, \dots, m$$

The optimization variable of the problem is the vector $x=(x_1, \dots, x_n)$, f_i are the constraints functions and the constants $b_1, \dots, b_m$ are the limits of the constraints. The goal is to find the vector x^* which is the solution of the optimization problem since the objective value is the smallest of all valid vectors x. This means the value x^* needs to satisfy the following equation for any vector z where the constraints are below their boundaries.

$$f_0(z) > f_0(x^*) \quad while \quad f_i(z) < b_1, \dots, f_m(z) < b_m \qquad (4)$$

The optimization problem represents the challenge of selecting the best vector x from all available candidates so that the cost function f_0 is minimized.
[Boyd and Vandenberghe 2009]

An example for such a problem would be a portfolio optimization. There are a set of n assets available and the best way to invest in them is searched. In this case the variable x_i is the investment in the i-th asset, which means that the vector x describes the overall allocation of

all assets. The constraints f_i can be the expected profit, the total budget to invest, or the maximal investment per asset for risk diversification. The cost function would be the total risk of the investment. The goal is to find the perfect investment per asset to gain the maximal profit while minimizing the risk.

Many problems contain multi objectives which must be optimized, e.g. the power consumption and the speed of a circuit. Since the two objectives will conflict with each other, a trade-off between both must be found. Such a point would be a Pareto optimum where none of the objective functions can be improved without degrading another one. There can be multiple Pareto optima which in a set are the so called Pareto front. [Xing Tan and Mao 2005]

The objective function or cost function maps an associated cost to a function value. This function is needed to compare two points with each other and to decide which one to prefer. In general, we want to minimize the cost function and find the parameter set which fulfills this constraint. A simple cost function could be the mean squared error between the desired values *d(x)* and the function value *f(x).* [Thiel and Smith 2002]

$$\sum_{i=1}^{m} \left(d_i - f_i(x)\right)^2 \tag{5}$$

The desired values *d* are defined by the user in ADE as example that the bias current should be smaller than 2mA. The function values are gathered via simulation of the circuit with a certain parameter set. Depending on the output, the next parameter set is calculated which is used for the subsequent simulation. The goal is to find the parameter set which minimizes the cost function and therefore provides the best circuit performance.

The difference between a local and a global minimum is displayed in Figure 7 which shows a function with a local and global minimum. A function has a local minimum when its value is smaller than or equal to the value of nearby points. But the function value can be larger than a distant point. The point where the function value is smaller than, or equal all feasible points is called a global minimum. [Hassoun 1995]

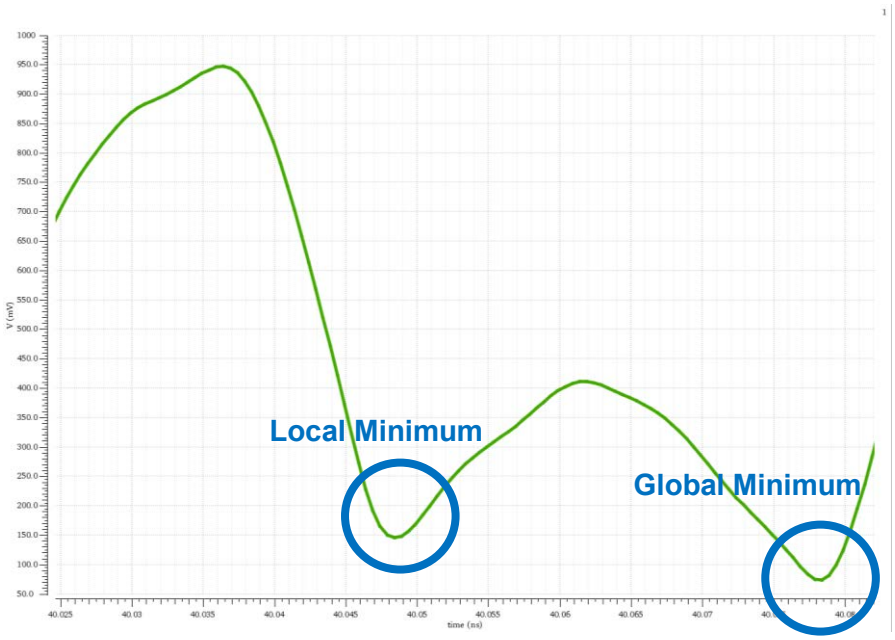

Figure 7: Local and global minimum

The global minimum is always the desired optimization result since it provides the overall best circuit performance. However, running a full global optimization including the correct parasitics for each optimization cycle can be resource demanding and take several days to finish. Therefor a local optimization is often used for a faster although not ideal result.

2.2 Local Optimization Algorithms

The big benefits of local optimization are that they are less resource intensive and faster than a global optimization and hence can manage large scale problems. A tradeoff between the invested effort and proximity of the result to the global optimum must be chosen. For critical circuits it is recommended to perform a global optimization to certainly have the best result in the end.

Local optimization has several downsides beside not searching the global minimum which must to be considered. A good start point based on the circuit knowledge and initial simulation results is crucial for a valid result. Since only the nearest minimum is searched the start point has a high influence on the output. Additionally, the selected algorithm can have a major influence on the result.

2.2.1 Gradient Based Algorithms

These local optimization algorithms need gradient information of the function and thus it must be differentiable. As a consequence, the function must be continuous for its complete domain. Whereby only continuity is not enough that the function is differentiable. A function is differentiable if the following constraint (6) is satisfied for each point x_0 of the domain definition. [Hunter 2014]

$$\lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} \tag{6}$$

This implies that for each point x the function has a distinct tangent line which is not vertical.

Gradient based algorithms calculate the gradient and change the variables in order to go in the negative direction of the gradient. This is the direction of the steepest decent and will lead to a minimum.

Conjugate - Gradient Method

The conjugate gradient method is a popular iterative algorithm used for solving large linear systems. This method relies on the simple gradient descent algorithm which calculates the gradient and follows its negative direction to find the minimum. This can be seen in equation (7) which shows the calculation of the next point x_{k+1} that is dependent of the step size α_k and the direction which is the gradient of the function. The step size can either be fixed or be determined by a line search.

$$x_{k+1} = x_k + \alpha_k \nabla f(x_k) \tag{7}$$

This method has the disadvantage to oscillate across a valley since the direction of the gradient changes if it is crossed. This leads to that the path to the minimum is a zig-zag which is not ideal, even though the minimum is found. To avoid this issue a so-called friction term p_k which is depending on the previous values is added to the calculation of the next point.

$$x_{k+1} = x_k + \alpha_k \, p_k \tag{8}$$

This ensures that the required iterations are limited and the minimum of the function is found faster.
[Ruszczynski 2006]

Broyden - Fletcher- Goldfarb - Shanno Algorithm

BFGS is an iterative quasi-Newton method which avoids computing the hessian matrix but approximates it instead. If the performance space is close to quadratic this algorithm is more efficient than the conjugate gradient method.

The basis of this algorithm is the Newton's method which uses a second order Taylor expansion to calculate the minimum as it can be seen in equation (9)

$$f(x) = f(x_n + \Delta x) \approx f(x_n) + f'(x_n)\Delta x + \frac{1}{2}f''(x_n)\Delta x^2$$

$$0 = \frac{d}{d\Delta x}\left(f(x_n) + f'(x_n)\Delta x + \frac{1}{2}f''(x_n)\Delta x^2\right) = f'(x_n) + f''(x_n)\Delta x \tag{9}$$

$$\Delta x = -\frac{f'(x_n)}{f''(x_n)}$$

The next point is calculated according the formula displayed in equation (10) whereby $f'(x_n)$ is the gradient and $f''(x_n)$ the hessian matrix of x_n.
[Snyman and Wilke 2018]

$$x_{k+1} = x_k + \Delta x = x_k - \frac{f'(x_n)}{f''(x_n)} \tag{10}$$

The issue thereby is that it requires the gradient and the inverse hessian matrix which leads to a high computation effort. This is solved by the BFGS algorithm which uses an approximated version of the hessian matrix. That avoids the full calculation of the matrix and provides a significant speedup and less resource requirements.
The hessian matrix is enhanced every step with the current information in order to have a good approximation.
[Krumke 2004]

2.2.2 Derivative Free Algorithms

Derivate free algorithms do not require a gradient and are numerical local optimization methods. Therefore, the cost function can be discontinuous and not differentiable.

Hooke and Jeeve

Instead of calculating a completely new point each time, this algorithm continues in the direction of the last result until no further improvement can be seen. This means that this algorithm is not memoryless like other algorithms e.g. the steepest decent method. Therefore, since the past information as the steps n-1 and n are used properly it may speed up the convergence. [Koziel and Yang 2011]

The algorithm consists of two phases: the exploratory and the pattern move. In the exploratory phase each variable is changed by a fixed step size in either positive or negative direction depending if the cost function improves or not. Only if it improves the variable value is set and the new point is the combination of all variable values. If no improvement of the cost function can be found the step size is reduced and the exploration starts again.
If a new point is obtained instead of starting with a new exploration around it, the pattern move phase is started. Therefore, the point is changed in the same direction and with the same step size as before as long the cost function decreases. If no further improvement can be found the exploratory phase starts again around the current point.
The algorithm continues this way until the step size is smaller than a specified boundary and then the last point is set as the output result.
[Quarteroni et al. 2007]

Brent-Powell Algorithm

This algorithm is recommended if the function is not differentiable and it is known that the starting point is already near an optimum. That is because this method searches only in a fine grid around the starting point. [Cadence 2018a]

2.3 Global Optimization

The goal of global optimization is to find a global minimum of the cost function and thus the best result possible. Therefore, the parameter set where all constraints are satisfied and the cost function is minimal must be found. There are various algorithm types available which are bases on different mathematical approaches.

Global optimization is used in a wide field like science, engineering, management and business. In all these fields there are problems where the output should be maximized while the cost should be minimized with regards of certain constraints.

Simulated Annealing

This algorithm comes from the observation in metallurgy that if certain metals are heated above their recrystallisation temperature and then naturally cooled down that the atoms do not form the strongest configuration possible. That this configuration is formed a certain cooling rate is required that the total system energy is minimized.

The basic idea of this algorithm is to select a neighboring point randomly and if the cost function value is smaller it is selected as new point. If it is larger than the current one, it is only selected if the Metropolis rule is satisfied. The rule can be seen in equation (11) whereby ΔE is the change in energy which is in this case the change of the cost function value. In analogy to metallurgy is T the temperature which means the larger the temperature the larger is the probability that a point is selected where the cost function increases. The right side of the equation is a random number between zero and one.

$$e^{-\frac{\Delta E}{T}} > R(0,1) \tag{11}$$

During the optimization, the temperature is reduced which can be made by different approaches. For example, by continuously reducing the temperature after each cycle.
[Keikha 2011]
[Allstot et al. 2003]

Evolutionary Algorithms

The biological evolution is the basic principle of these algorithms such as reproduction, mutation, recombination and selection. It is a population bases derivative free method.
As it can be seen in Figure 8 the algorithm starts with an initial population which is ideally wide spread over the domain. These are then evaluated which means that for each member of the population their cost is calculated. The results are then used for the fitness assignment where for each candidate their fitness is determined. Dependent of the fitness results the best candidates are selected which are then used to reproduce. There either two parent candidates are combined, or one candidate is mutated to create a new offspring. The new population is then again evaluated and the whole circle starts again. This iterative process continues until a good enough candidate is found. [Vikhar 2016]

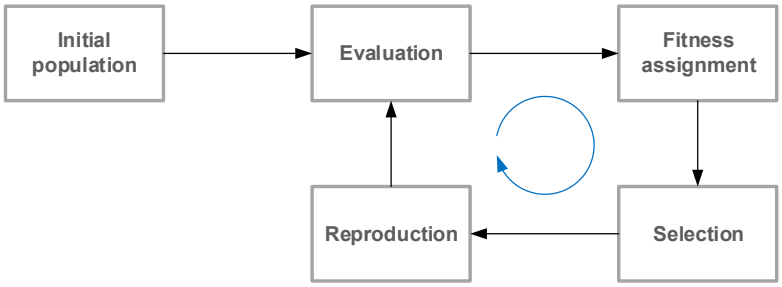

Figure 8: Evolutionary algorithms basic flow

3 Tools and Current Way of Working

In this chapter the current way of working and all the therefore needed tools are explained. As reference design for the different tools a simple inverter shown in Figure 9 is used to minimize the runtime and overall resource consumption.

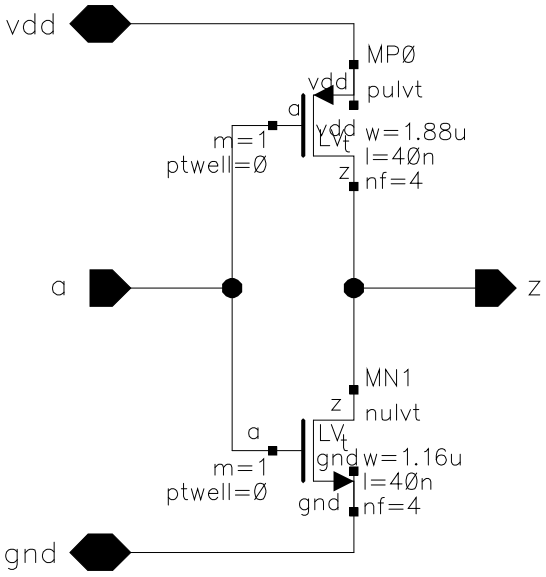

Figure 9: Inverter schematic

3.1 Current Way of Working

As already shown in Figure 1 the current design flow is that the circuit is designed, simulated and optimized at first. Afterwards the layout is built and finally an extraction is performed which is used to verify the design. Depending on the results of the simulation which includes all parasitic components a redesign is necessary or not. This can lead to time consuming updates, since the complete flow needs to be executed again.

To avoid the possible redesigns and to get a better circuit performance it would be advantageous if the parasitics could be already considered during the optimization. Until now this is done in the way shown in Figure 10. At the beginning the initial layout and schematic are created and an extraction of the layout is performed. The gained parasitic capacitors are then added in to the schematic which is then used for all further simulations and additionally is optimized. This enables that the parasitic capacitors of the initial layout are included in the simulations and therefor provide a more realistic result. However, these parasitics are only valid for the initial layout. As soon the device parameters are changed in the optimization

they are out-of-date. Therefore, they can only be used to approximate the parasitic influence. A full extraction, simulation and verification of the final design is necessary.

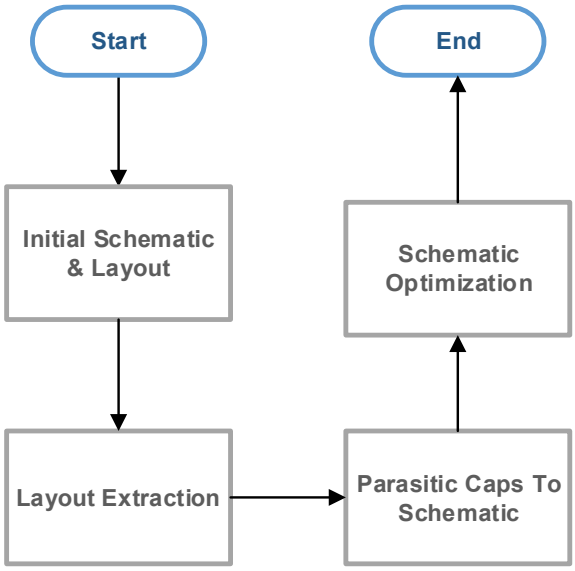

Figure 10: Initial optimization method

Figure 11 shows the initial inverter schematic which is updated with the parasitic capacitors of the initial layout. There are capacitors between all nets which influence the circuit performance even though they are in the atto Farad range or smaller. Their influence is especially important if the circuit needs to operate at a high frequency, as it can be seen later on in the example of the frequency divider in chapter 6.

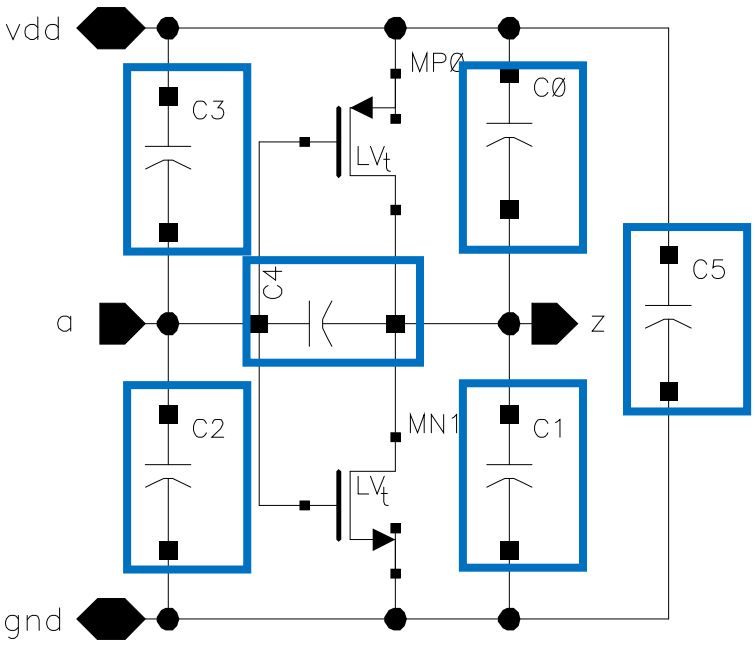

Figure 11: Initial schematic updated with parasitic capacitors

At the time being, this is a completely manual flow whereby the designer needs to perform all steps one after each other. The larger the circuit becomes the more time is required for the extraction and the placement of the capacitors in the schematic. To reduce the time consumption, the initial optimization type was added to the automated optimization flow, which performs the extraction of the initial layout and adds the capacitors to the netlist. The designer needs to keep in mind that only the parasitic capacitors are considered and none of the resistors or inductors. Therefor the performance results will change after a full extraction. Additionally, the values of the capacitors are fixed and only valid for the initial layout of the circuit. This leads to that an optimization will not find the best performance possible, since the real parasitics are changing in each cycle according the current parameters. And because the extraction is only performed once at the beginning they are not valid for the other parameter sets. This can lead to, that the optimization algorithm keeps increasing one device parameter to improve the performance, but since the parasitics are also increasing it can lead to a worse output. Therefore, the optimization will show a large parameter as the result, but ideally it would be smaller in order to limit the parasitic influence.

The usage of the initial parasitic capacitors is an easy and quick way to add a feeling of the parasitic influences to the simulations. It can only be used to approximate their impact and thus it would be advantageous to extract the parasitics in each optimization cycle. This ensures that the final result is the best one achievable and that the performance is also reached after the layout is build.

3.2 Testbench

In order to simulate a circuit, the corresponding testbench which represents the environment of the DUT is essential. As it can be seen in Figure 12, it contains the DUT cell and it models the surroundings of it. This means that it contains all the power supplies and circuits connected to the in- and outputs of the DUT which model the impedances the cell needs to load. The testbench itself has no in-, or outputs to it since it is the top level. It can use several different variables which can be changed during the simulations. For example, the frequency of the input signal could be chosen to be a variable and be swept in the simulation. Another possibility would be to create corners for the highest and the lowest input frequency to verify the function of the circuit at all corners.
[Kundert and Zinke 2004]

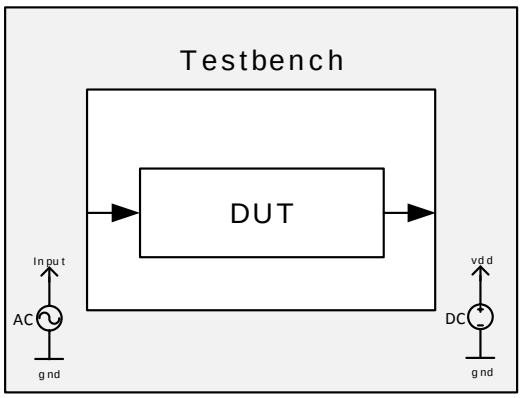

Figure 12: Testbench and DUT relation

If the in-/outputs of the cell are not properly modeled the simulation results will significantly deviate from the actual results. Therefore, it is necessary that the DUT in the testbench sees the same environment as later on the chip. For example, the input of the cell will not be an ideal square signal with instantaneous change between the two voltage levels, but instead with certain rise and fall times. In addition, the input source will have a certain impedance which will influence the DUT. Such factors must be considered during the testbench creation to get valid results.

Figure 13 displays the testbench for the inverter. In this case the DUT is the middle cell and the two others provide the in-output of the cell. The global input signal is generated by a pulse source and the supply is connected to a DC source. The two additional inverters are used to ensure that the DUT is in a realistic environment. This means that the input signal is not ideal and that the output needs to drive a load.

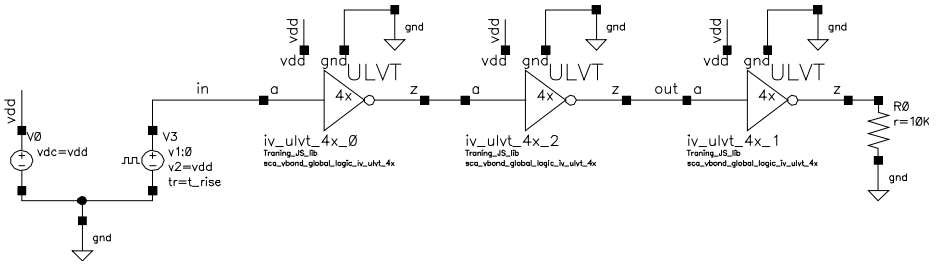

Figure 13: Inverter testbench

To increase the accuracy of the simulation it is recommended to use the extracted layout view instead of the schematic whenever possible. This will add the parasitic components to the netlist and therefore improve the accuracy. The only drawback is that the circuit must be manually extracted and the simulation time will increase, since there are more devices in the netlist. To minimize the required time and to get a first approximation of the parasitic influences a capacitor only extraction is recommended.

That the extracted view instead of the schematic is taken a config view can be used which defines what view to use for each instance in the hierarchy. This enables the user to choose the extracted view for specific cells and the schematic for the other cells.
Another possibility is to add the name of the extracted view to the *Switch View List* of the environment options of ADE as it can be seen in Figure 14. The crucial difference to the config view is that it gets used for each cell where the defined view *av_extracted* exists.

Figure 14: ADE environment options

3.3 Virtuoso Analog Design Environment (ADE)

The most used tool within NXP in analog circuit design regarding simulation is *Virtuoso Analog Design Environment GXL/XL.* It provides all the tools required for development, analysis and validation of a design. This includes different simulation setups, corner case evaluation and parameter sweeps. In addition, ADE provides an optimizer which can be used to improve the circuit performance. Furthermore, ADE enables the usage of different simulators and the possibility to manipulate the simulation flow with e.g. a pre-run script which is executed between the netlisting and the simulation step. [Cadence 2014]

Currently the optimizer of ADE XL is only used on the schematic level and therefore does not include any parasitic effects at all or only the ones from manually placed initial parasitic layout components. The netlist is once created from the schematic and used for all performed simulations. Device parameters which are changed during the optimization are treated as variables to avoid renetlisting in each cycle, which enables a speedup.

The ADE main window shown in Figure 15 is used to define the simulation settings, run options, parameters and the output expressions which will be optimized. An optimization can either be performed local or global. The local optimization performs less cycles since it only searches around the starting point for the minimum. This leads to the issue that only optima around the starting point will be found and not the global optimum. The advantage is that it is faster and less resource intensive. The local optimization requires a good reference point for the parameters. It is recommended to use the initial one from circuit designer.

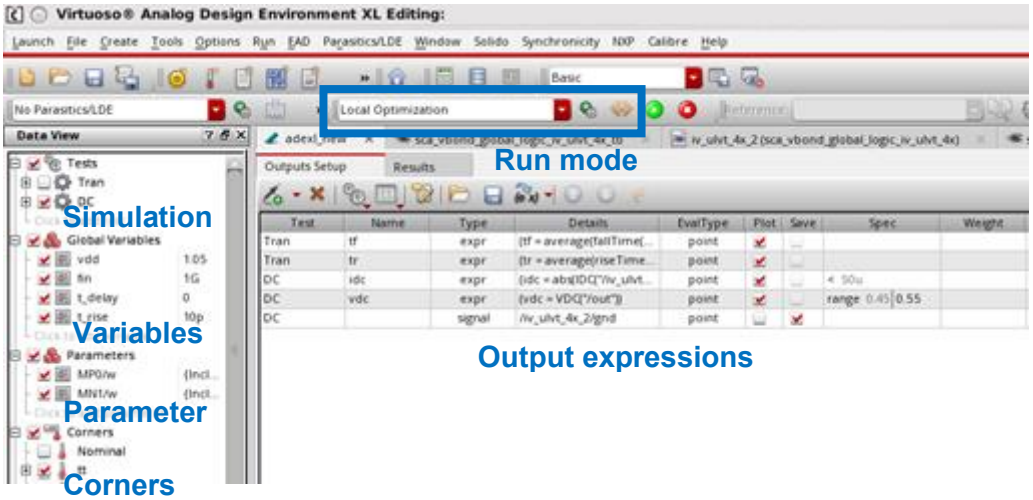

Figure 15: ADE main window

3.3.1 ADE Settings

ADE can be used for a lot of different simulations and circuit analyzations. Therefore, it provides various settings the user must define in order to get the correct output.

Run Mode

The run mode defines which kind of simulation should be performed. There is the default option *Single Run, Sweeps and Corners* which runs everything as defined by the user, but doesn't start any additional analyses.
The used modes in this thesis are *Local Optimization* and *Global Optimization* which use the simulation to gain the results of the equations which are feed in the optimizing algorithm in order to fulfill their constraints. There are four local optimization algorithms available from which the user can choose one to get the best result for the current optimization problem. [Cadence 2018c]

Parameter

There are certain device parameters of the DUT defined, which will be changed during the optimization in order to satisfy all constraints. Each parameter has a certain range in which it can be set and a step size in which it can change. That is necessary to limit the optimizer, so it does not select parameters which are too small or too large to be layouted. In addition, it limits the possible values for this parameter and therefore the overall possible parameter sets.

Output Expressions

To define what should be optimized and in which manner the output expressions are used. In general, they allow the user to define various equations which define certain cell specifications. It is possible to set and use variables in addition to the circuit data.
For an optimization it is required that at least one equation has some specifications defined which represent the desired value. This could be that the output of the equation must be smaller or larger than a fixed value or that it must be inside a specific range.
In addition, it is possible to state weights for the equations if some are more important than others. By default all are set to one, but the user can increase the weight if needed. This is beneficial if there are main constraints which must be satisfies that the circuit works properly and side constraints which should also be optimized, but which are meaningless if the circuit function fails. In this case the main constraints would get a high weight to ensure that they are always fulfilled.

Simulation Settings

These settings define the simulation type e.g. transient and all the corresponding options. In addition, it is possible to choose which simulator should be used and to set the environment options.
For the optimization flow created in this thesis the user can choose between a transient and a DC operation point simulation. Mica is used for all simulations and the environment options are set accordingly. The user can define the stop and switch view which is used by ADE to enable the usage of extracted views as described in chapter 3.2.

Variables

To enable that the user can easily change settings in the testbench it is necessary to define variables which can be quickly changed or also be modified during the simulations. It is possible to set a variable for a device parameter input, e.g. the frequency of the input source and define them in ADE as a global variable.
In the simulation settings it is possible to choose variables which should be swept in order to see their influence on the output. If an optimization is performed it is possible to treat a variable the same way as a device parameter and change it to satisfy the output constraints.

Corners

For the verification of the circuit at all process corners ADE provides the possibility to select corners for which the simulation is run. These depend on the process specific information of the semiconductor fabrication plants and define how the circuit behaves for all process corners. They are defined for typical, slow or fast NFET and PFET devices. This means that the corner *ff* means that all NFET as well the PFET devices are assumed to be fast. This could be caused that during the fabrication the gate oxide is at its lower fabrication boundary and the devices have a low threshold. [Razavi 2017]
Furthermore, the user can create additional corners for temperature, or for certain variables. This can be used to verify the functionality of the circuit for the complete required temperature range or e.g. for the input frequency range.

3.3.2 ADE Optimization Flow

By default, ADE performs the optimization as shown in Figure 16, where it starts at the calculation of the current point. Then the corresponding variables are fed in the netlist which is afterwards included in the testbench. That is used for the simulation and the generated output values are evaluated against the nominal values. If all constraints are satisfied the optimization stops and the results are displayed. Else the next point is calculated and the whole circle starts again.

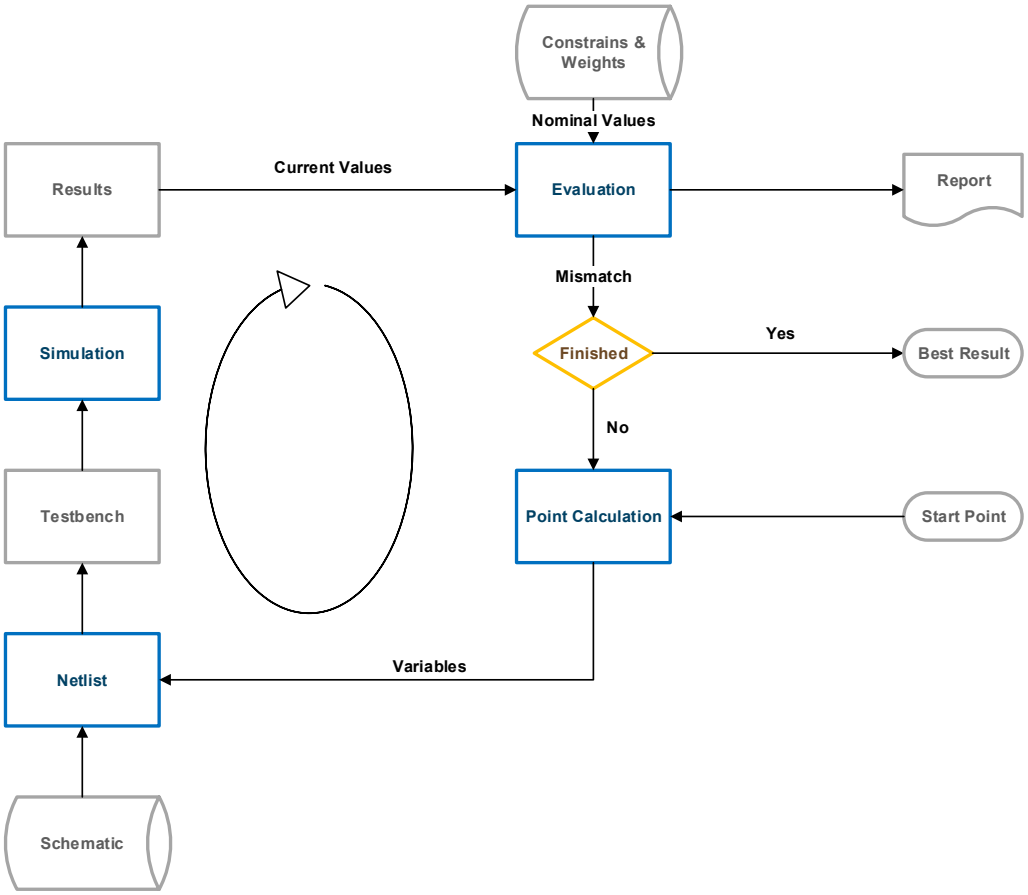

Figure 16: ADE original optimization flow

The calculation of the next parameter set is dependent on the chosen optimization algorithm. For a global optimization the parameters for a large set of simulations are created at the beginning and simulated in parallel depending on the available resources. Most of the local optimization algorithms also provide the feature to start multiple simulations at the same time instead of going through this circle sequentially. This enables that the optimization finishes faster but with the cost that more computational resources are required.

Since ADE is running through this circle several times and each time a simulation is performed it is recommended to use the conditional evaluation mode which is displayed in Figure 17. It makes sure that no unnecessary simulations are run and therefore enables a speed up. The simulations are split in two groups. The first one passes and the second one fails at the reference point. For each new point first the second group is simulated to check if an improvement was achieved. If this is the case the first group is also simulated to verify that it still passes.

If the second group fails, the current point is worse or equally than the reference point and can be discarded. Therefore, the first group is not simulated since the point will not be used. [Cadence 2018b]

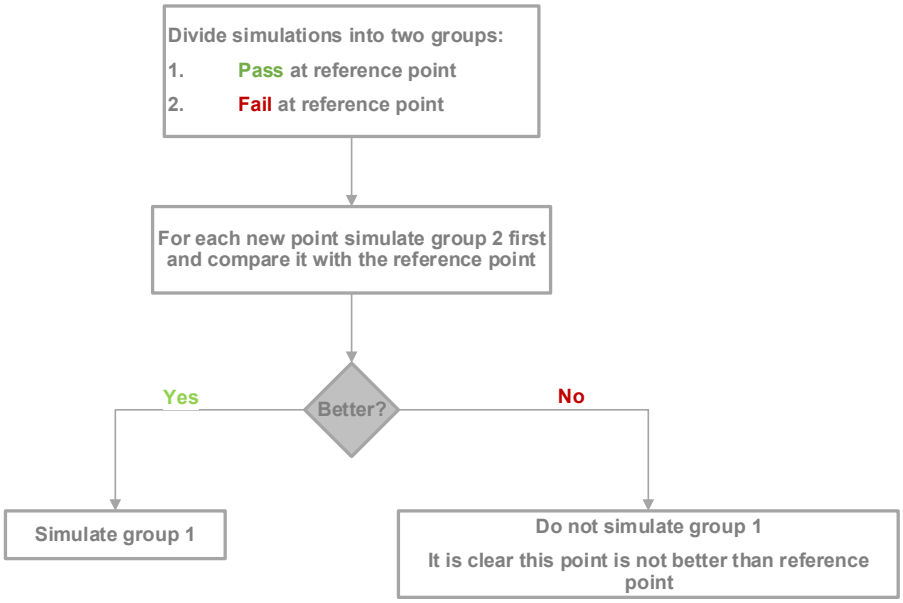

Figure 17: Conditional Evaluation Flow

3.3.3 ADE Optimization Manipulation

The standard optimizing flow always uses the same netlist and only updates the device parameter which change. This is possible since only the schematic is optimized and the parasitics are not considered. To include the parasitics and to always have the correct values for the set parameters it is necessary to update the netlist before the simulations are performed.

ADE offers the possibility to define a pre-run script which is executed before each simulation is executed. It must contain Ocean code which can perform various operations. In this case the script will be used to include the parasitic influences in the netlist. This is done according to the flow displayed in Figure 18. At first the current device parameters are collected and with them the PCell is placed. From the schematic view a CDL and from the layout view a GDS is generated. These are then used to start a LVS-QRC check which creates the required data to run QRC to generate the extracted view. Afterwards the netlist is created and used to update the existing one with the correct parasitics values. This ensures that each simulation includes the latest parasitics.

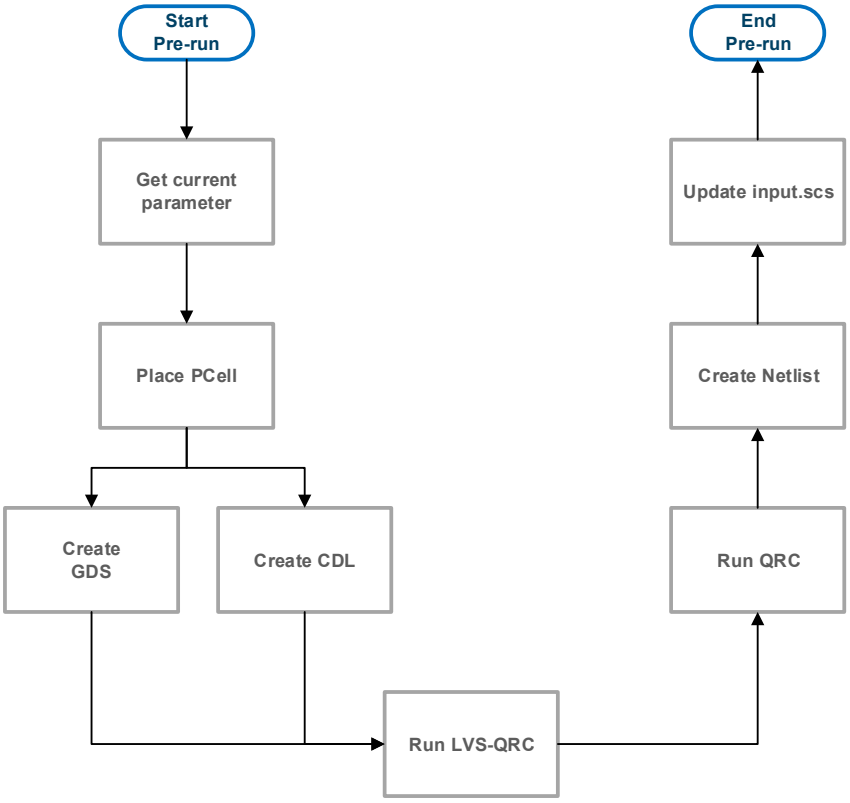

Figure 18: Pre-run script flow

For the graphical usage of ADE, the script can be defined by right clicking the test where it should be attached and selecting *Pre-Run Script* as it can be seen in Figure 19. Afterwards a GUI opens up where the location of the file can be set. Before each simulation of this test is started the specified script is executed.

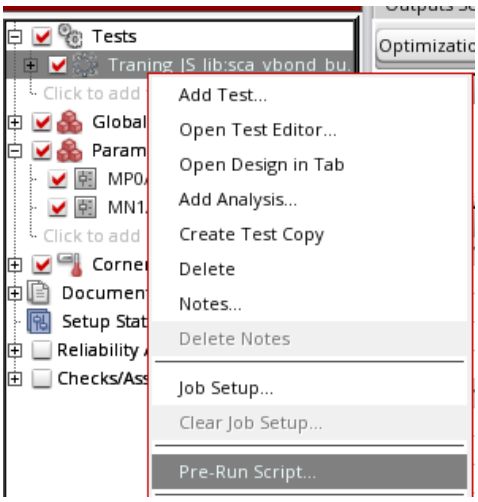

Figure 19: Defining a Pre-Run Script graphical

If the ADE session is created via a script the command *axlSetPreRunScript* can be used to define the location and *axlSetPreRunScriptEnabled* to enable the usage of the pre-run script.

3.4 Parameterized Cell

The extension language for Cadence is Skill which is based on Lisp. It provides a large set of sub routines which can be used for various purposes. Skill can be used to create custom tools which can be integrated in the Cadence programs, or to create PCells of different complexity. Furthermore, it can be used to manipulate the Open Access data structure and to add custom menus to the programs of the cadence design tool suite. [Nguyen 2008]

That a layout is automatically generated from the input parameters it is required that the circuit is a PCell programmed in Skill. Figure 20 shows a simple MOS transistor where the type and the metal connections can be specified. Therefore, the shape, used metal layers and arrangement of the transistor is dependent of the user input.

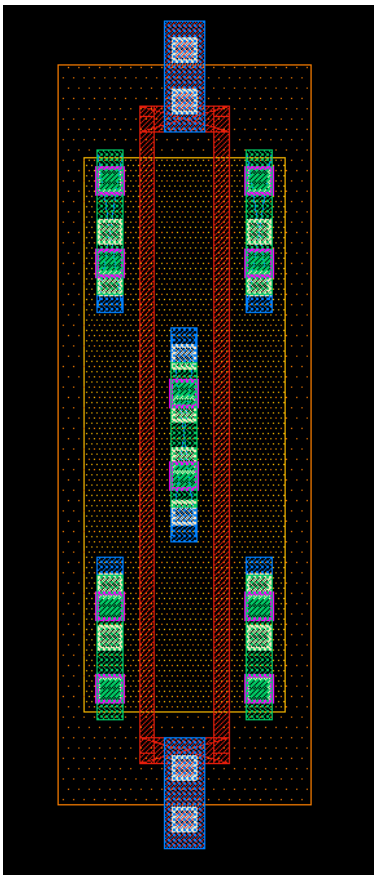

Figure 20: PCell example NMOS RF transistor

The advantage of PCells is that if the circuit parameters change there is hardly any time required to update the layout. Furthermore, the parameter and thus the layout can be changed automatically and the parasitics can be extracted for the optimization flow. In addition, if the PCell is created in a proper way it is DRC and LVS clean for its complete input

parameter range. The LVS check verifies that the same devices are placed in the schematic and the physical layout as well that they are connected correctly. DRC is used to check if the layout complies with all design rules defined by the semiconductor fabrication plant. These define the maximum and minimum ratios for the different available layers and are used to ensure that the IC can be produced. [Langner and Scheible 2017]

3.4.1 PCell Structure

PCells can be built up to the level of complete IP blocks like e.g.: a full ADC. Therefore, it is recommended to start with low level PCells, as the transistor shown above, then move on to simple logic blocks as an AND gate to then move up to circuits of higher complexities. This enables reusability of low level blocks and a structured approach.
In Figure 21 an example of a PCell hierarchy is displayed. The designer chooses the parameters of the master PCell and the lower level cells are built accordingly since the appropriate parameters are passed to each sub level cell. All lower level cells could also be used on their own as a master PCell. The bottommost level of the hierarchy are the CMOS transistors and other basic devices.

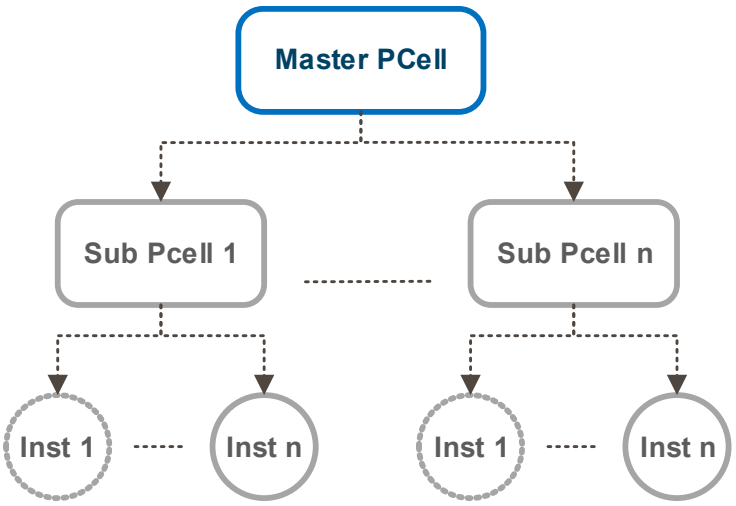

Figure 21: PCell hierarchy example

The code for a PCell follows a certain structure which is indicated in Figure 22. At The top the cell is defined which should be generated. The library, cell and view name as well type are defined and used in the *pcDefinePCell* statement to create the cell view in the database. Additionally, as another part the input parameters are set which are in this case called *P1* and *P2*. These are visible to the outside and can be chosen in order to configurate the devices as required. They can have different data types like string, integer or boolean.
The main part of the code is the function body where it is defined what is performed and in which manner. The complete generation of the cell must be programmed here. The parameters which were defined can be used inside the code to build the cell accordingly. All other local variables must be added to the let statement, which are in this case *var1* till *var3*. [Tayenjam et al. 2017]

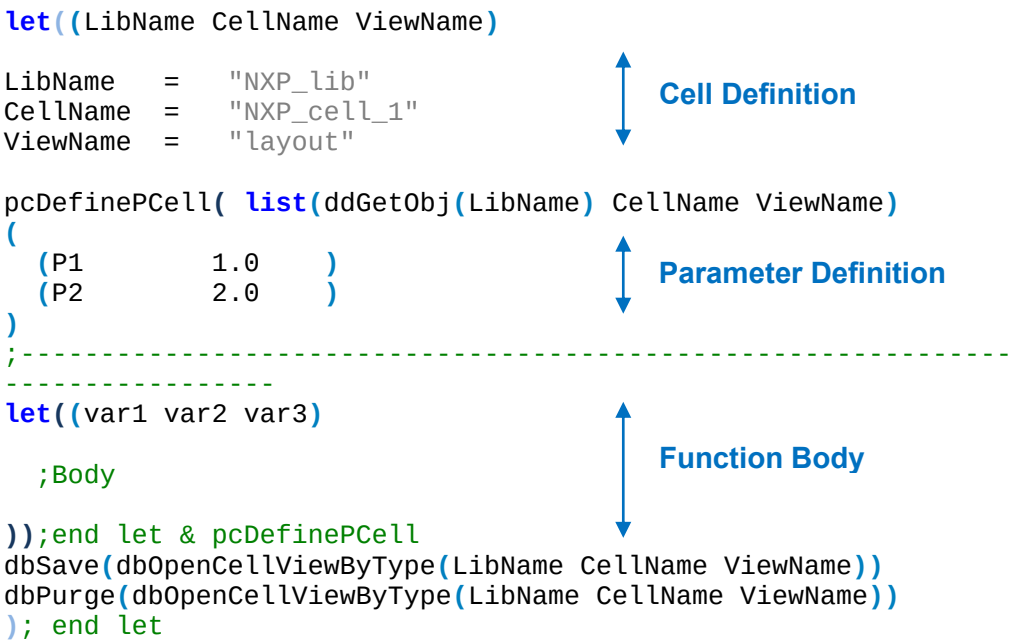

Figure 22: PCell code structure

The code must be executed once in order to generate the cell view. This is done in skill with a *load()* statement which takes the file path as its input. If the cell is then placed and the parameters are changed only the function body is executed to update the cell accordingly.

3.5 Quantus QRC Extraction Solution

In this thesis the Quantus extraction solution from cadence is applied for all extractions since it provides high accuracy and is certified for the used foundry process.

To verify the design including all parasitic components it is necessary to perform an extraction of the layout. The output can be an extracted view as shown in Figure 23, or alternatively the parasitics are written to a DSPF file which can be included in the simulation. Both outputs include all devices of the design and in addition all extracted parasitic components. For this example, an RC extraction of the inverter layout was performed. The extracted view shows the poly and metal layers and for each found device their symbol with the according parameters.

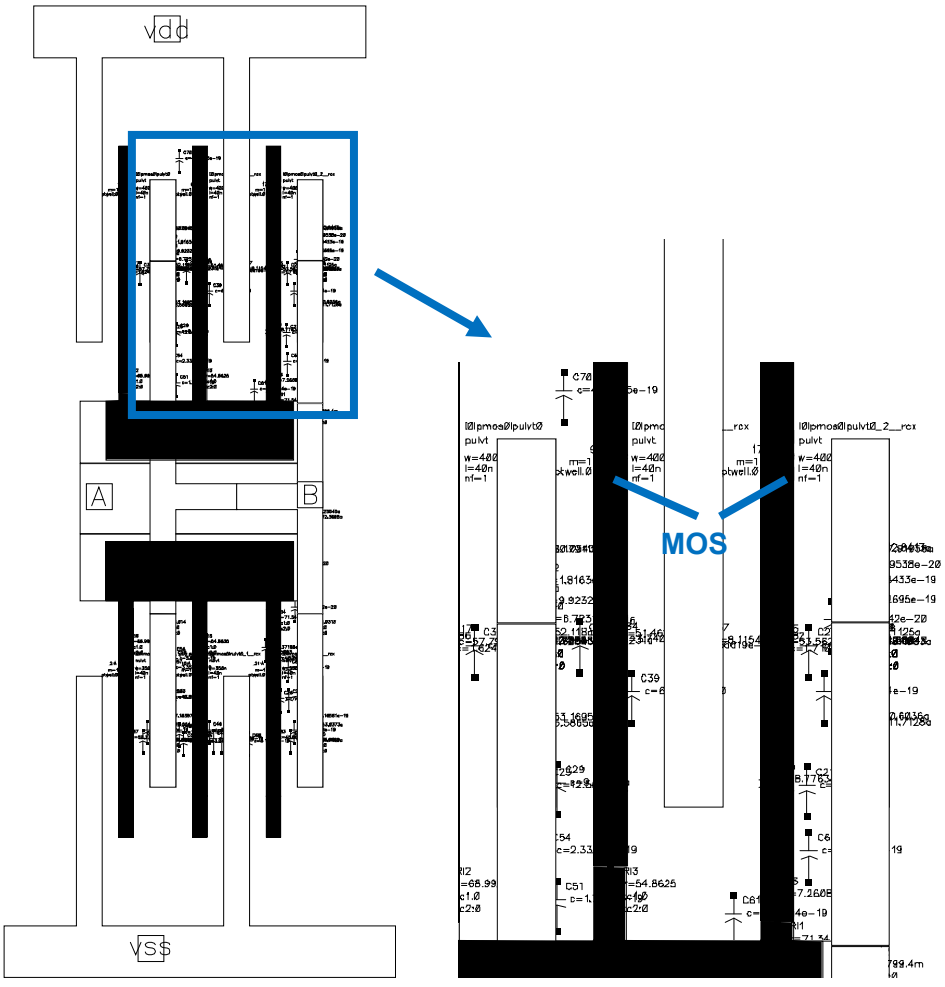

Figure 23: Extracted view of the inverter

3.5.1 Extraction Types

Quantus QRC provides the extraction of resistors, capacitors and inductances. Whereby resistors or capacitors can be extracted alone (R/C only), together (RC) or joint with the inductances (LRC).

The capacitors are calculated with a field solver which simulates the electrostatic field among the wires. This is required since the capacitance is the coefficient relating the electric potential and the electric charge. In order to limit the computational resources and simulation time, state of the art optimizers use on one side technology pre-characterization which is performed once per technology. It provides look-up tables for various test structures which were simulated with 2- or 3-D field solvers to obtain accurate results. On the other side a pattern matching approach is applied that chops the signal path into small pieces. The resulting pieces are then matched with the lookup tables to get their capacitance. Therefore, the challenge is to create a lookup table which on one hand includes enough test structures to be accurate and on the other hand not too much since the complexity of the pattern-matching procedure increases significantly with the number of test structures.
[Yu and Wang 2014]

For long wires in the nanometer regime it is necessary to also extract inductances in order to get a realistic simulation result. Therefore, the inductance calculation technique partial element equivalent circuit is applied. This method solves the issue that in modern interconnect structures are no dc paths which form a well-defined loop. It is assumed that the induced current returns at infinity which avoids the necessity of a return loop.
Therefore, the partial inductance of each line element can be calculated with the aid of field solvers and then be combined in the partial inductance matrix L. This matrix then is combined with the RC matrix and is simulated to determine the current loop.
[Wong et al. 2005]

RC or C extractions are performed after the layout is finished to verify the performance of the design. LRC extractions are usually only performed for critical and high frequency designs since they increase the extraction and simulation time significantly.

3.5.2 QRC Usage

That QRC can be started a certain data set is required which can be created via LVS-QRC. This means that the user needs to run this check beforehand in order that the data is created. Therefore, a normal LVS check can be started but with the option *Create Quantus QRC Input Data* enabled in the *Output* settings as it can be seen in Figure 24. If you use the default directory for the *QRCDataDir* QRC automatically finds it and uses the data from there.

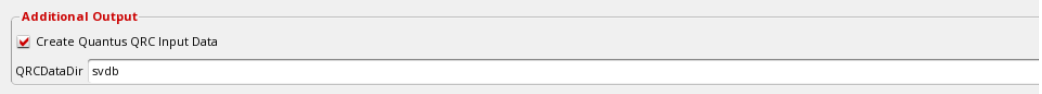

Figure 24: LVS-QRC option

QRC can be started as shown in Figure 25. This will open up the GUI where the cell and technology information are defined. These are needed to set up the main GUI.
If QRC is started from the layout view which should be extracted the default options can be used.

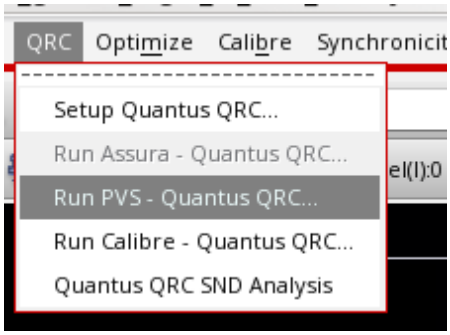

Figure 25: Start menu of QRC

Afterwards the QRC main GUI opens up, where all the different settings can be defined and the extraction can be started.
The GUI allows to set various settings regarding all parts of the extraction flow like extraction options or filtering options. The important settings are also available in the automated optimization flow where they can be chosen by the user as it can be seen in chapter 5.2.

4 Divide-by-two Circuit

The divide-by-two circuit is deployed in the PLL and is necessary to lower the VCO frequency so that it can be used by other circuits, such as the programmable divider. Hence, it is a critical circuit which needs to operate accurately at high frequencies.

Figure 26 shows the structure of a PLL which applies a divide-by-two circuit to pre-scale the VCO output in order that the programmable frequency divider can function properly. It must be considered that the reference frequency must also be halved to get the correct output frequency.
Especially if the VCO output frequency is near the maximum speed of the technology it is required to pre-scale its output. Therefore divide-by-two circuits are used since they can operate at the higher frequencies than dividers with other division factors.
[Razavi 1998]

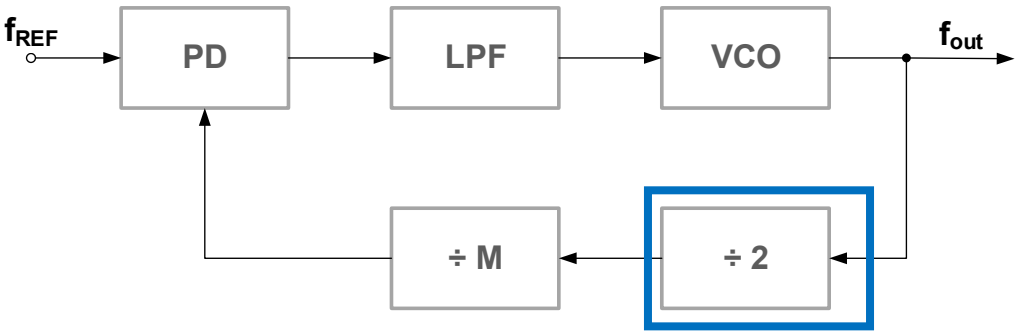

Figure 26: PLL structure

The PLL synchronizes its output signal f_{out} with the input signal f_{ref}. Therefore, the input of the VCO is adjusted so that the signals are equal in frequency and phase. This is accomplished since the phase detector generates a signal proportional to the phase error and the low pass filter removes its AC component. This signal is then fed in the VCO which changes its output frequency accordingly. By adding a programmable divider in the loop and the usage of an accurate crystal oscillator as reference the output can be chosen as a specified higher frequency. [Chen 2003]

4.1 Schematic

The divide-by-two circuit is build according to the Razavi topology which consists of two identical D-Latches in a master-slave configuration. The input signal which should be divided by two is required to be mirror-inverted. This means if one is high the other one must be low and vice versa.

The two latches are periodically and complementary switching between the sense and the latch mode. In the sense mode the input signal is taken and set on the output. In the latch mode the current output is kept. The latches change their modes with the level of the input (v*_vi). While one latch is in write mode the other one is in latch mode and vice versa. Since it takes two cycles to pass the input data from one latch to the other a division by two is achieved.

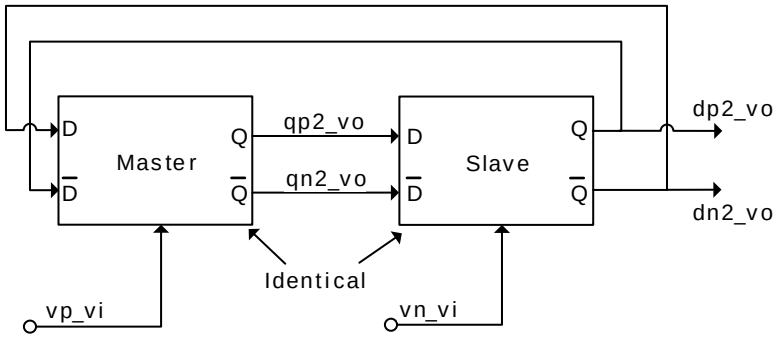

Figure 27: Divide-by-two circuit structure

The symbol of the divider is used to instantiate it into other circuits. It shows all the input/output pins and the parameters of the cell. On the top and bottom are the supplies of the circuit, on the left the two mirror-inverted inputs and on the right the four outputs. There are four outputs since the two signals from master to slave are also connected to the outside.

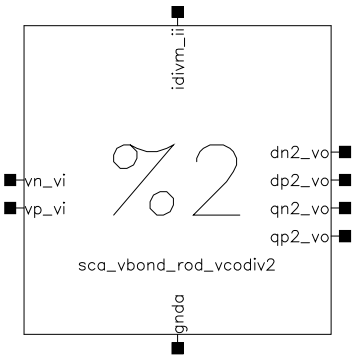

Figure 28: Divide-by-two circuit PCell symbol

The complete schematic is displayed in Figure 29 where the two identical master and slave latches are evident. This structure is chosen to avoid the usage of PMOS devices in the critical signal path since they would limit the maximal speed of the circuit significantly.

Each latch consists of four NMOS and two PMOS devices. The output of the master is the input of the slave. The output of the slave is connected to the input of the master but with the positive to the negative and vice versa. Hence there is a negative feedback loop which causes the output to toggle.
[Joshi et al. 2012]

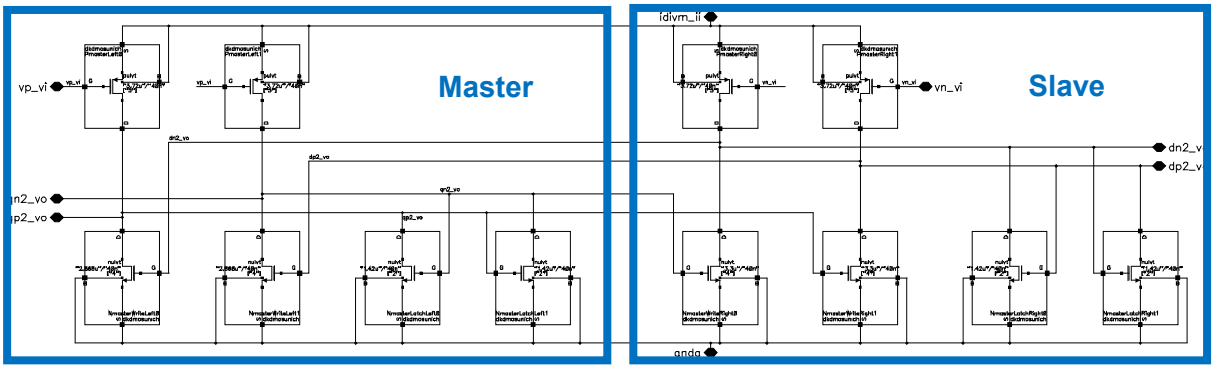

Figure 29: Divide-by-two circuit PCell schematic

4.2 Function

To explore the function in more detail the used latch of the slave is examined which can be seen in Figure 30. The master latch has the same structure but with slightly different parameters. The functionality is the same and thus only the slave is analyzed.
Each latch consists out of two write and two sense NMOS as well of two PMOS transistors. The slave latch has at its clock input *vn_vi* connected and at the two signal inputs *qn2_vo* and *qp2_vo*. Its output is connected to the pins *dn2_vo* and *dp2_vo.*

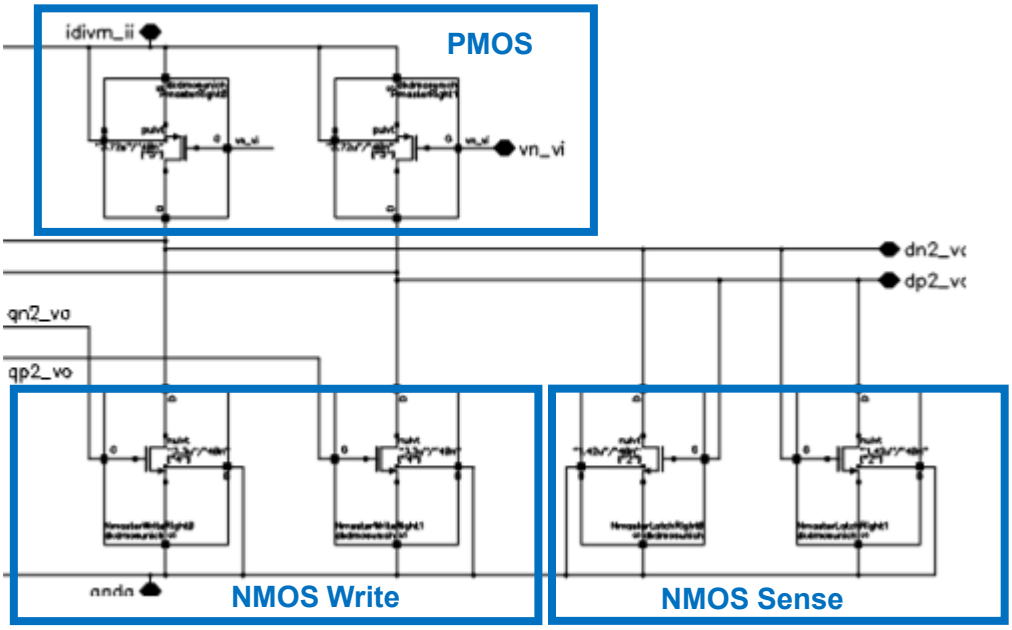

Figure 30: Divide-by-two circuit schematic right half

The PMOS are pull-up devices which are enabled if their input at the gate is at low level. If they are off, the circuit is set in the sensing mode where its output follows the input signals and otherwise it is in the latch mode.
The sense devices are used to scan the input in the sense mode and change the output voltage according the signal input. The write devices form a regenerative loop which discharge the output in the latch mode.

If the clock input changes from high to low the PMOS transistors drive a current on the cross coupled NMOS latch pair and output voltage rises asymmetrically according to the voltages at the gates of the NMOS sense devices.
If the clock input changes from low to high the PMOS transistors are closed and the NMOS write transistors discharge the output voltages.

This latch topology does not disable its output devices when changing from sense to latch mode. Even though this would cause a timing problem in general digital circuits it does not harm the divider function. That can be explained by the following two facts. First, since NMOS transistors are applied as the signal input devices they can only change their state when their input goes from low to high. Second, the latch can only change its output from low to high if it is in latch mode since elsewise the PMOS pull-up transistors are disabled. Hence, if it is in sense mode it cannot overwrite the value in the connected latch which is latch mode.

The waveforms of the divide-by-two circuit are displayed in Figure 31. On top is the clock input signal and its inverse signal which should be divided. Below are the output signals of the slave and the master illustrated once as the two complementary signals and once as the contained information. Clearly, the input frequency is divided by the factor of two also if the output signal is slightly degraded. The cause therefore is, that the PMOS devices provide a path from supply to ground if they are on and hence degrade the logic level. In addition, this results in a static power consumption.
Further, both outputs of the latch are low if it is in sense mode. One is pulled down by an input device and the other one maintains the state from the previous cycle.
[Razavi et al. 1995]

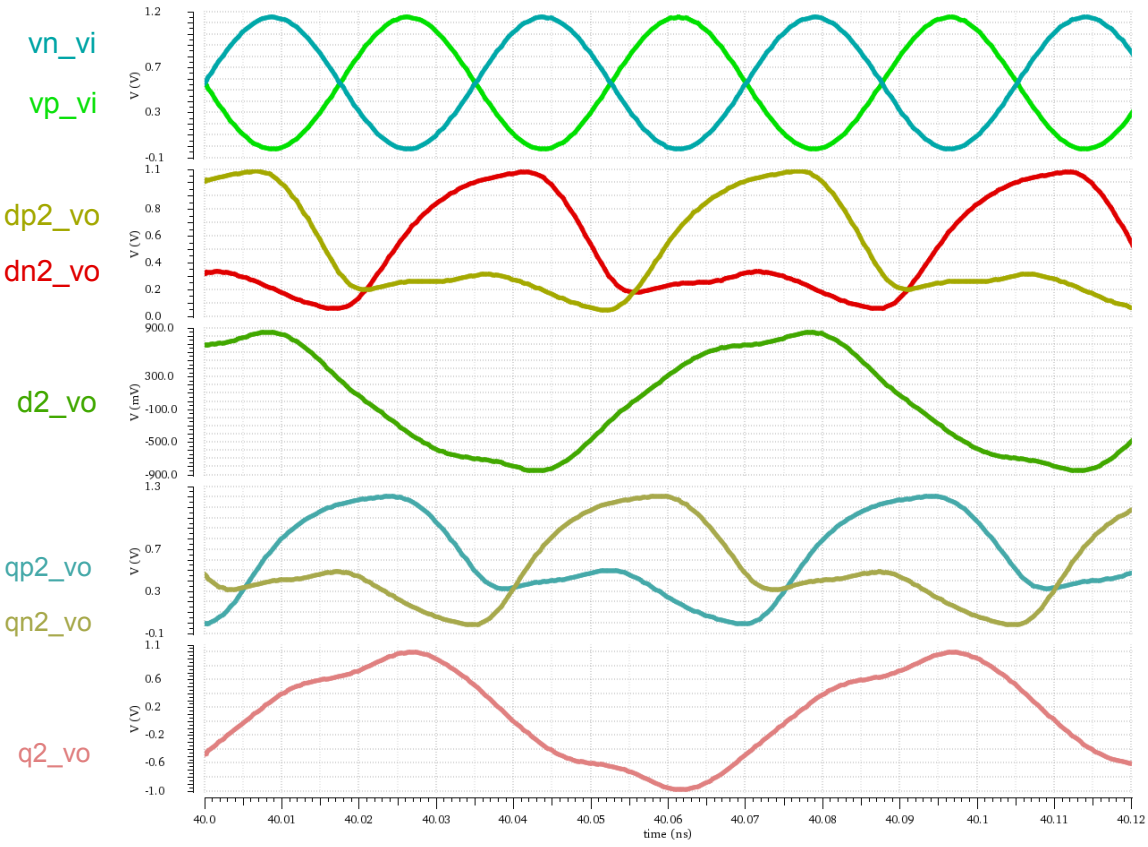

Figure 31: Divide-by-two circuit input vs output signals

4.3 Layout

The layout is built highly symmetrical to enable good matching between the devices to achieve the best performance. Since it is a PCell it can be easily built up for different parameters.

On top the PMOS transistors are located which are all together in one N-well which is connected with a guarding around the devices as it can be seen in Figure 32. The distance from the end of the devices to the guard ring is calculated in order to be able to place the contacts with equal distance.

Below are the NMOS transistors of the master latch on the left and the ones of the slave on the right side. They are separated by a PP guard ring to minimize their influence on each other. The write/sense parts are placed in a common centroid structure to ensure good matching.

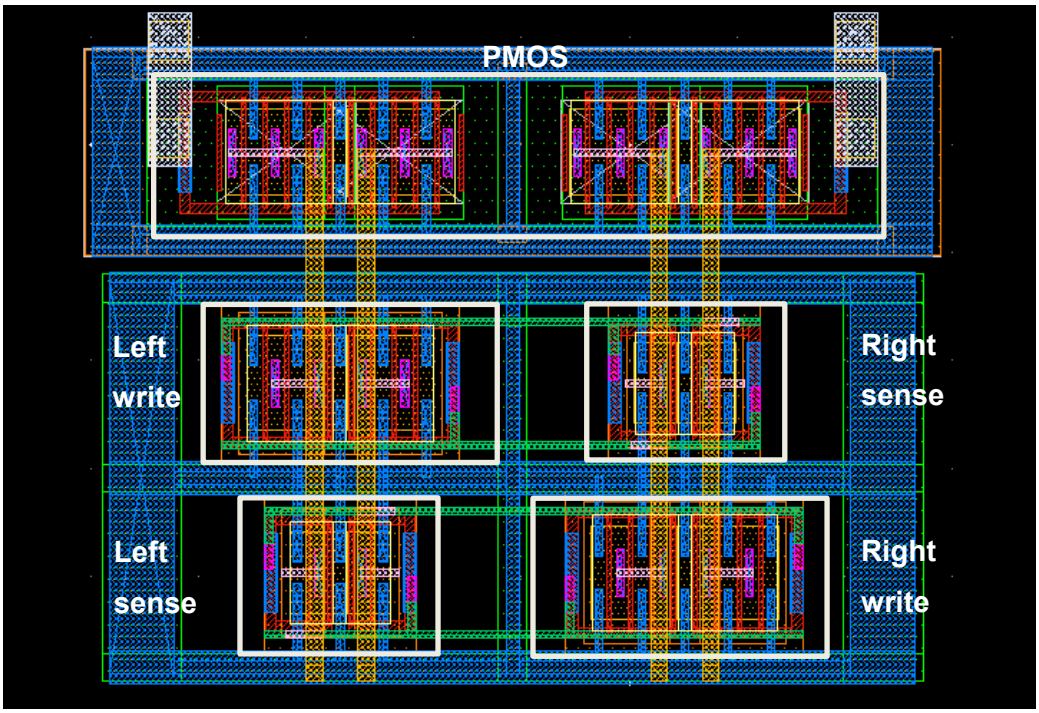

Figure 32: Divide-by-two circuit PCell layout

The PCell is LVS and DRC error clean and can be extracted. If the input parameters change the layout is automatically rebuilt according them.

The PCell takes the transistor characteristics as its input parameters as it is visible in Figure 33. The width and the number of fingers for the different transistors of the latches can be defined. The width of the NMOS transistors can be chosen differently for the left and right side. The option *Customize Layout* can be used to adjust the layout such as the number of substrate contacts. If the cell is simulated on schematic level and its parameters should be passed to the lower level cells the option *Use pPar for W,L, and fold* must be used. *pPar* allows to set a parameter value to one of the parent cell. This ensures that the parameters from the top cell are correctly passed to the lower level cells during the netlisting step.

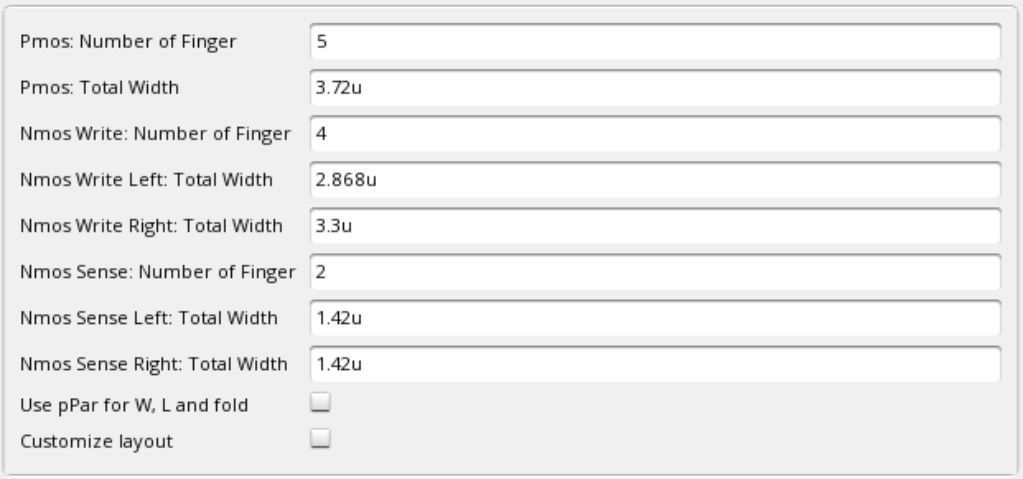

Figure 33: Divide-by-two circuit PCell parameter

That the layout can be built successfully some restrictions are necessary which are shown in Table 2. These define the maximum input values and verify that the different input parameters align with each other. This is required so that the layout can be built symmetrically.

Table 2: Divide-by-two circuit parameter restrictions

Constraint
PMOS NF > 3
PMOS NF > NMOS Write NF
NMOS Write NF >= NMOS Sense NF
PMOS NF < 25
Max Width < 100u
Max Contacts < 11

5 Physical Design Optimization

Until now the circuit optimization was only performed on schematic level where no parasitic devices are included. However, the circuit performance is significantly influenced by the parasitic components introduced via the physical design. Therefore, it would be beneficial to consider the parasitic influences during the optimization. This enables to avoid a redesign which could be required after the extraction and validation of the circuit.

In this chapter the physical optimization flow and the therefore created GUI is explained. Further, the data created during the execution and the optimization output are described.

5.1 Introduction

The user-friendly GUI allows the designer to effortless setup an optimization of any circuit whereby the steps displayed in Figure 34 will be executed.
At the beginning, it is necessary to collect all required data and options from the user via the GUI explained in chapter 5.2. Afterwards, the ADE view is created and configurated according the collected user input. In addition, the directories and files for the parasitics extraction etc. must be created. That is performed by several skill, or ocean procedures.
After everything is set up an initial optimization cycle is performed to verify that the settings are correct and the results are valid. In the next step the ADE optimization session will be started. Depending on the circuit complexity, optimization type and the specification boundaries the session runs a few minutes to hours. Once the optimization has finished the output is collected, verified and displayed to the user.

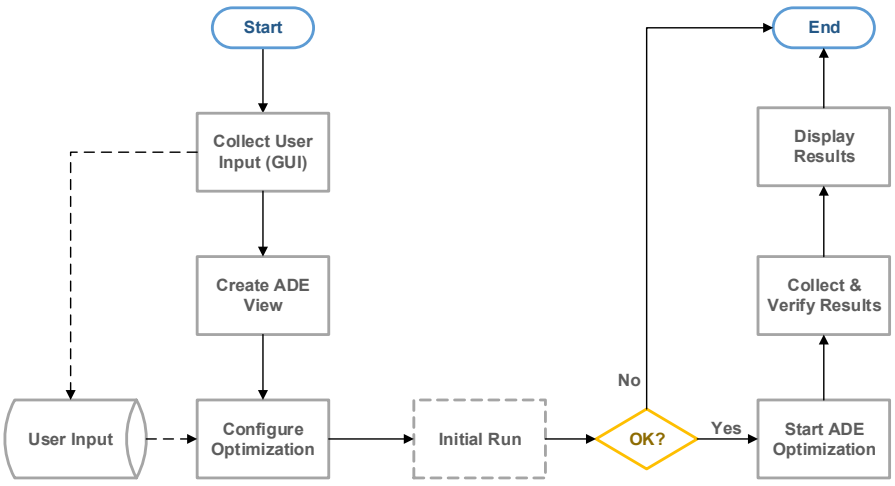

Figure 34: Automatization flow

The main GUI can be opened via the NXP menu in the CIW. Therefore, the needed code must be loaded which is done via a cadenv package.
The creation and configuration of the ADE session, as well the initial optimization run is done within one ocean procedure which is located in a sperate ocean script. This procedure gets called as soon as the user clicks on the *run* button in order to start the optimization flow.

The default optimization flow of ADE is displayed in Figure 35 where it is visible that there is an optimization cycle which is performed until the constraints are satisfied. It consists of the calculation of the current parameters and variables which are fed in the netlist and testbench. These are simulated and the results are evaluated against the constraints. If all are satisfied the optimal result has been found and it is reported. Elsewise the cycle continues and the next parameter point is calculated.

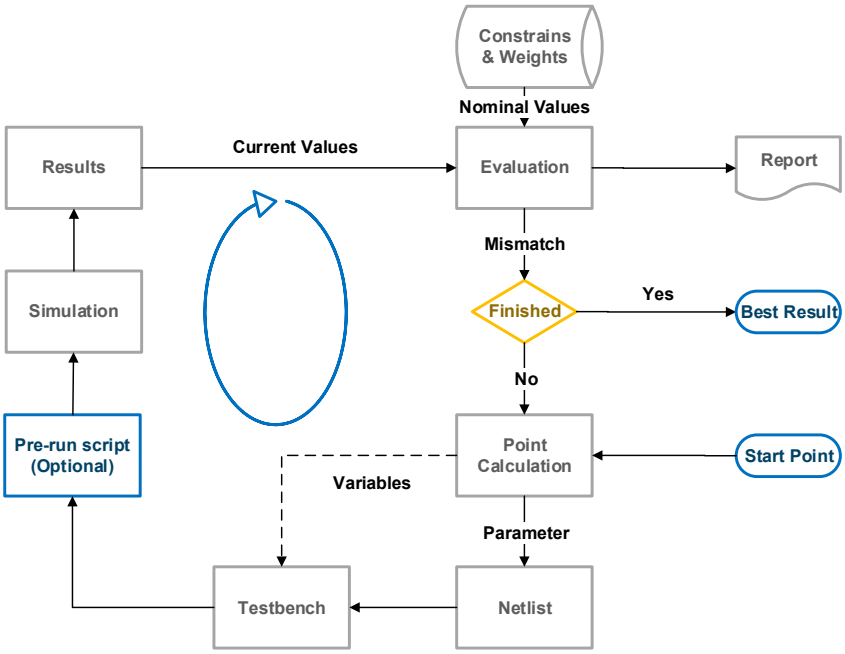

Figure 35: Default ADE optimization flow

In order to include the parasitic influences into the optimization it is necessary to add an extra step between the netlist/testbench updates and the simulation. This is the extraction of the parasitic components of the circuit with the current parameters, which are then fed back to the netlist to include them in the simulation. The method to enable this extra step is the usage of a pre-run script. ADE provides the possibility to define an ocean pre-run script which is executed between the testbench update and the simulation as shown in Figure 35.

The flow of the pre-run script is shown in Figure 36. The first step is to collect the current parameters and next to instantiate the DUT PCell with them. This is used for the GDS and CDL creation afterwards. They are required for the LVS-QRC check in the following step. Then it is possible to run a QRC extraction in order to obtain the parasitics. These are used to update the input.scs file which is applied for the simulation.

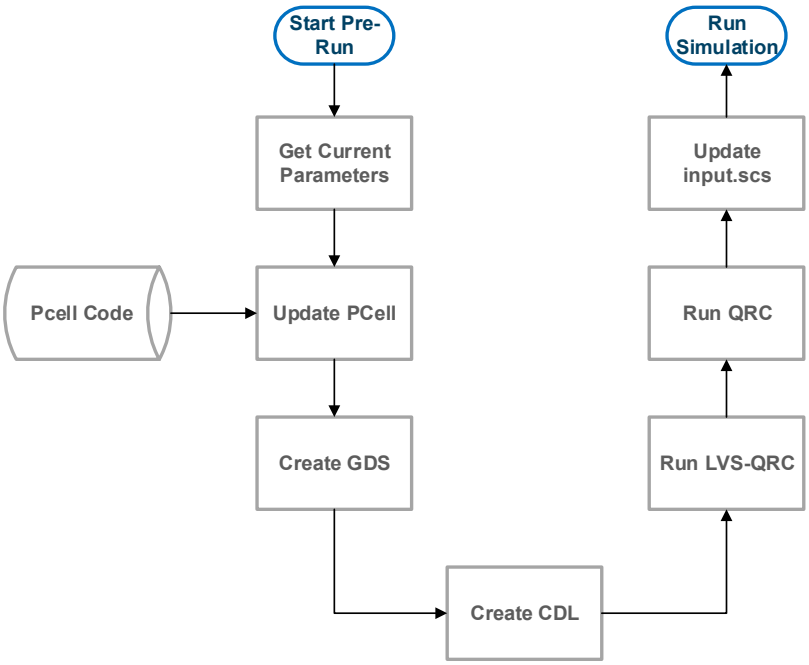

Figure 36: Pre-run flow

In order to keep the setup the same as it would be done directly in ADE for only schematic level, it is necessary to probe certain sub level voltages. This is required if voltages on nets of sub blocks are used in the equation. Through the extraction the hierarchy and net names are lost and therefore the simulator would not find the chosen net. To avoid this issue, it is required to bring the nets to the top level with a net name the simulator recognizes.

This is implemented by changing the sublevel nets in the equations to specific automated generated names during the ADE setup. These names are then used to create the according pin in the flattened layout and schematic in the pre-run script. That way the same names are used in the equations and the netlist of the extracted view. This provides that the simulator finds all nets and can calculate all equations.

For the latest version the usage of sub level currents in the equations is not supported since it would be required to add additional terminals to the views.

5.2 Graphical User Interface

The main GUI is required so that all the data and options for the optimization etc. can be specified by the user. Therefore, the GUI consists of several tabs were each one is dedicated to one part of the optimization flow. In addition, it allows to save the current state and to load a previously saved state. This enables the user to save all options and use them again at another time.
Further, the GUI provides the possibility to load options from an existing ADE view. Since there were already some simulations performed beforehand to prove the feasibility of the circuit, all the equations etc. are already defined in an ADE view. Therefore, it is possible to simply load these into the main GUI via the *Load ADE* button.
The help button can be used to quickly open the user guide of the automated optimization GUI. Additionally, each major part of the GUI has a question mark button which opens a popup containing a brief description of the according options.

Figure 37 displays the first tab *Testbench*, which is used to define the testbench, the DUT and the equations which are optimized. The user can use the drop-down list which contains all available libraries to define the required one. That updates the cell list with the cells which are part of this library. If the cell is selected the view list is updated as well. As soon library, cell and view are chosen the DUT can be defined as one of the sub cells of the testbench.
The equations are essential for the optimization since they define what should be optimized and in which manner. To simply define them the possibility to use the cadence calculator is available. Each equation needs a name and a target which must be defined after the calculator was closed. The name is used for the display of the simulation results and should be set to describe the equation (e.g.: risetimeVout). The target sets the boundaries which are used during the optimization. It can define if the result of the equation should be smaller or larger than a certain value or be in a range between two specified values. Additionally, a weight can be stated if the equations have different priorities in which they should be satisfied. If none is specified the default value of one is used.

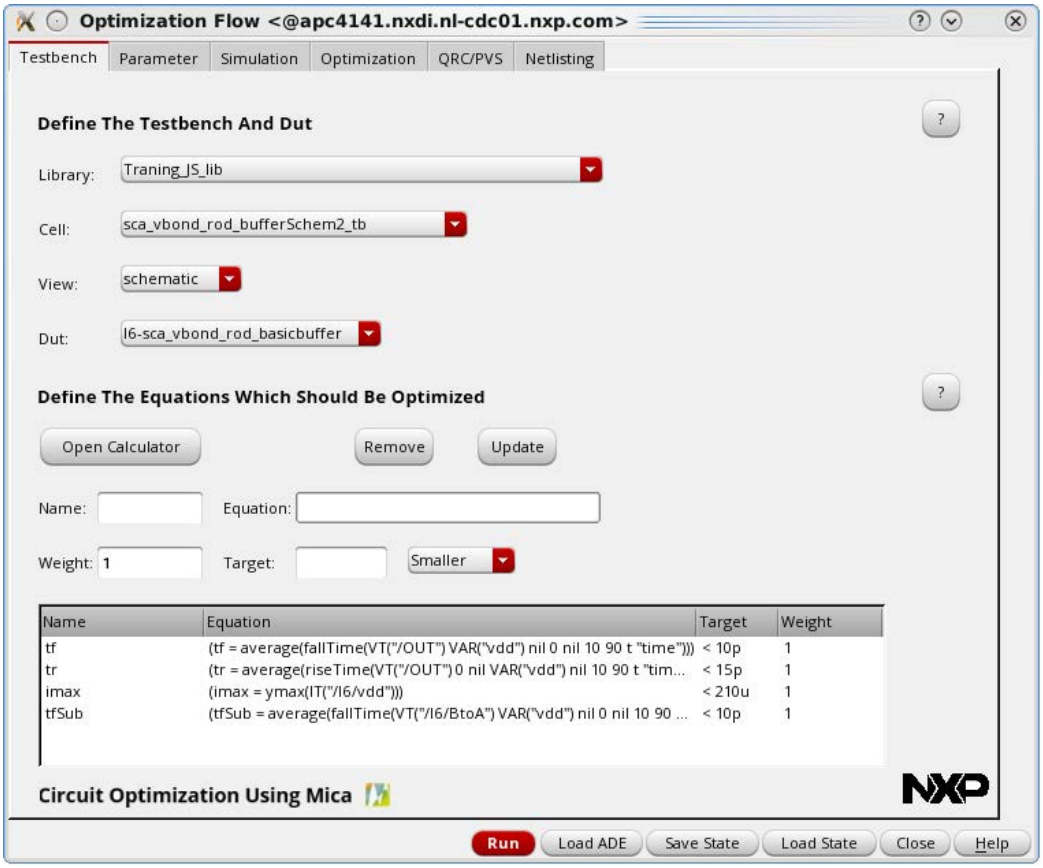

Figure 37: Optimization GUI Testbench Tab

If the *Open Calculator* button is pressed, the built in cadence calculator opens which can be seen in Figure 38. This allows the designer to easily define all equations in the well know manner.

If a signal type is selected the testbench schematic is opened so that the user can define the voltage, or current which should be probed. The equations can be defined in the normal manner and be sent to the stack afterwards.

After all equations were defined the user needs to simply close the calculator. This triggers that the schematic is closed and the equations are loaded into the main GUI. There the last step is that the name, range and weight of the equation must be defined.

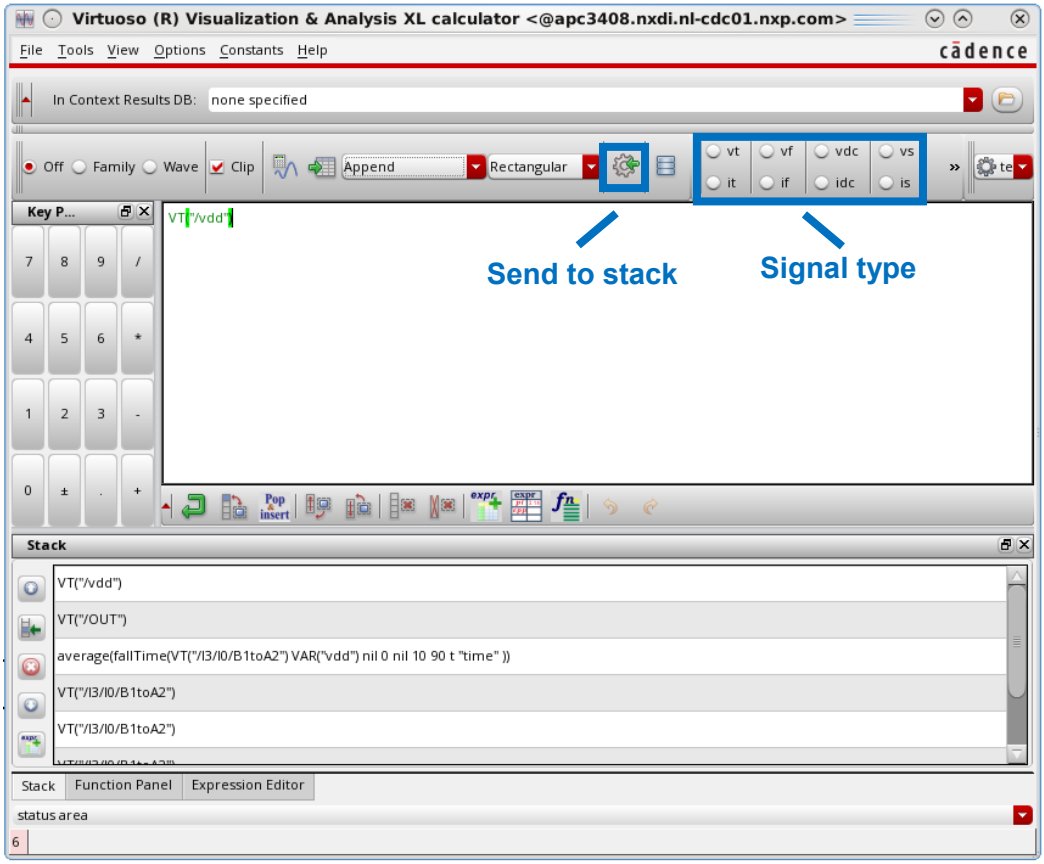

Figure 38: Optimization GUI Calculator Usage

The *Parameter* tab displayed in Figure 39 is used to set the device parameters which are modified during the optimization in order to satisfy the targets of the equations. Therefore, the name, range, step size and start value of the device parameter must be defined. To easily do so, the hierarchy tree is used to collect and define the parameters. It can be opened by clicking on the *Collect Parameter* button.

Defined parameters can be changed by clicking on them and either pressing the *Remove* button or, by updating the values and pressing *Update*.

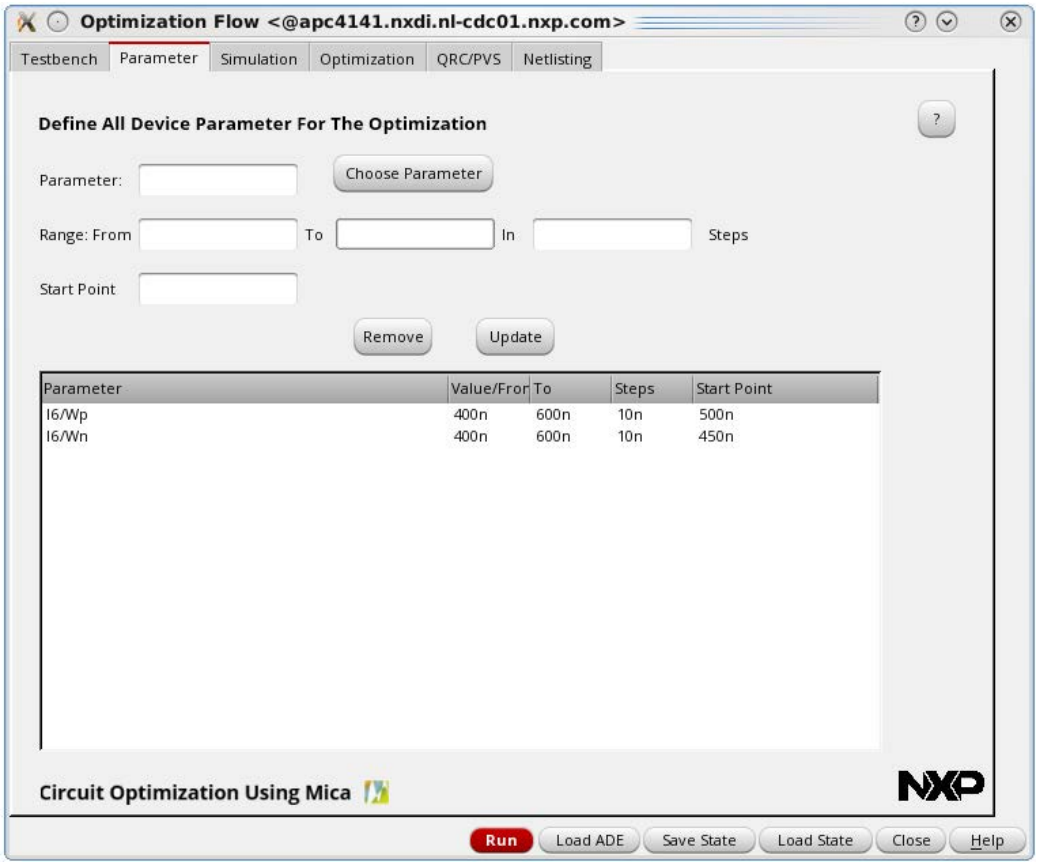

Figure 39: Optimization GUI parameter tab

Figure 40 shows the *Choose Parameter Form*, which consists of a field with the hierarchy tree on the left and the parameters of the selected cell on the right side. The tree is built of the testbench until a PCell is found. The parameters of this PCell can be used for the definition. For the selected parameter the range, step size and starting value must be defined at the bottom of the GUI. The *Submit to Main Gui* button is used to send the definition to the main GUI and the field in the parameter tab is updated.

If a parameter should be matched with the ones from other devices this is possible by enabling *Match with others*. First a parameter must be selected and all the information defined. Afterwards the *Match with others* field has to be enabled and the other cell views can be selected while holding the CRTL key. Lastly, the *Submit to Main Gui* button must be used to send the matched parameters to the main GUI.
If device parameters are matched ADE always changes them in the same way. This would be required if different transistors of the circuit must match in the physical design.

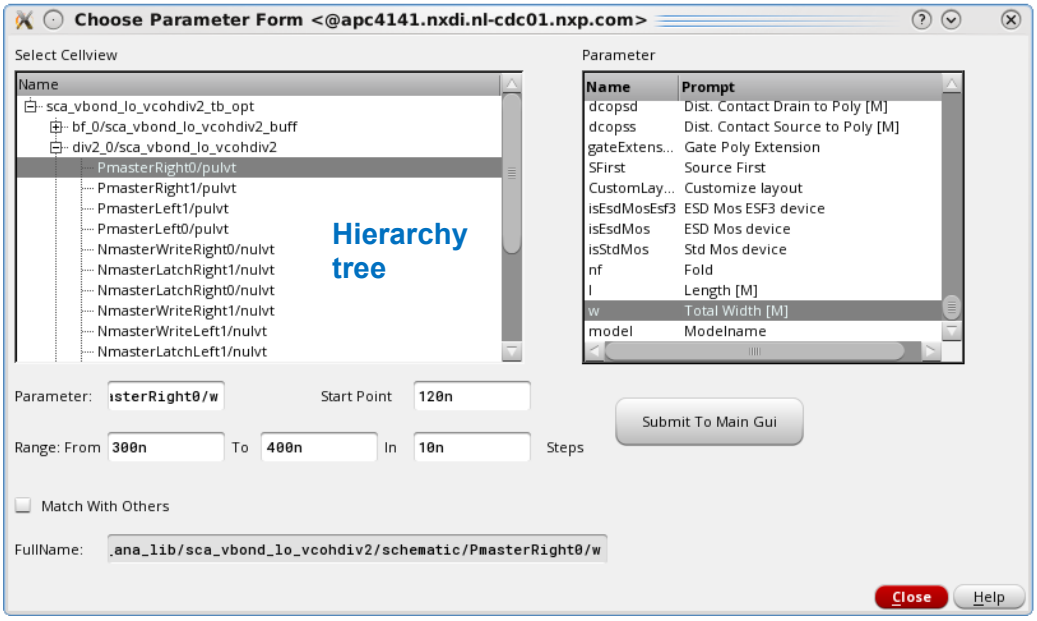

Figure 40: Choose parameter GUI

All options regarding the simulation are defined in the *Simulation* tab presented in Figure 41. The analysis as well the regarding options are selected at the top and the global variables are defined at the bottom.

The analysis type can be chosen to be a transient or DC operating point simulation. For the transient simulation the start, stop, initial- and max-step value must be defined. Further, the maximal number of jobs and parallel processes can be set, if the simulations should be performed in parallel. *Max Jobs* defines the maximum jobs started in the ADE session and *Nr Parallel Procs* the number of parallel processors per job.

The global variables of the simulation must be defined at the bottom fields, by specifying name and value and clicking on *Submit*. If a variable should be swept the *Sweep* field must be enabled and the range as well as the start value must be chosen. This way it is possible to change variables during the optimization to satisfy the constraints.

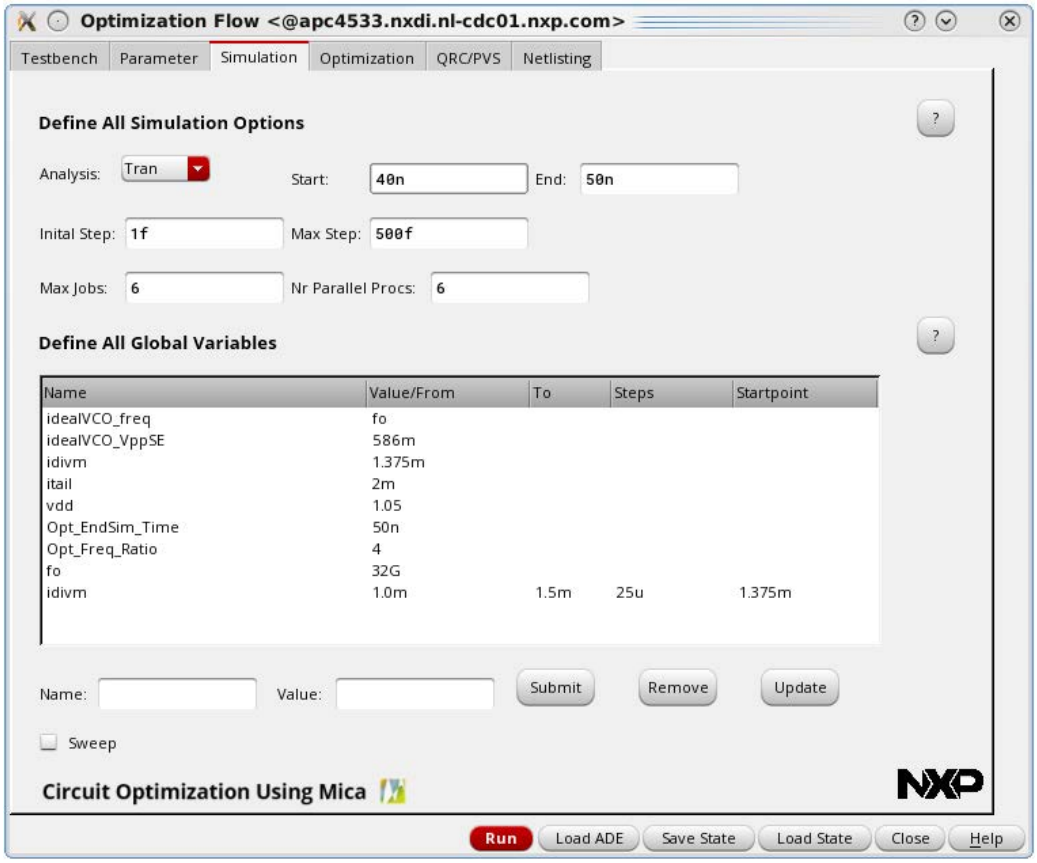

Figure 41: Optimization GUI simulation tab

To define settings related to the optimization the *Optimization* tab shown in Figure 42 is used. The optimization *Type* can be set to either *Schematic*, *Initial* or *Layout*. Dependent of the selection is a full, none or only an initial extraction performed.

For the *Schematic* only type just the schematic is considered like in the normal ADE optimization. If initial is chosen, the initial layout is extracted and the parasitic capacitors are added to the netlist which is simulated. This allows to include an approximated influence of the parasitics since they heavily impact the circuit performance. However, these parasitics are kept constant if even the devices and therefor the real parasitics change. The advantage of this type is that the layout doesn't need to be a PCell and that it is less time consuming than the layout type.

For the layout type the PCell is placed in each optimization cycle with the current parameter set, then extracted and included into the netlist which is simulated. This ensures that always the correct parasitics are used. Therefore, is it required that the DUT is a PCell. Even though it is more resource intensive it leads to the best results since no additional extraction and verification is necessary afterwards.

The field *Run in Background* defines if everything is executed in the current CIW, or in a new virtuoso which is started in the background. Therefore, the CIW is either blocked, or everything is performed in the new background CIW which requires additional resources.

The optimization mode can be chosen between a local and a global optimization. For the local optimization one algorithm out of the four available ones must be selected.

In addition, the evaluation mode must be chosen. The conditional mode avoids running unnecessary simulations and therefore provides a speed up. The full mode performs each simulation and hence requires more resources.

The optimization must have at least one stop criterion, whereby three are available. The first one is reached if all specifications are met. The second one allows to define a time and the third one a simulation point limit after which the optimization is stopped.

The corner model file contains all available process corners and is by default set to the one of the installed cadenv package. The *Choose Corners* button can be used to define the corners which should be used for the simulation. It opens the form shown in Figure 43.

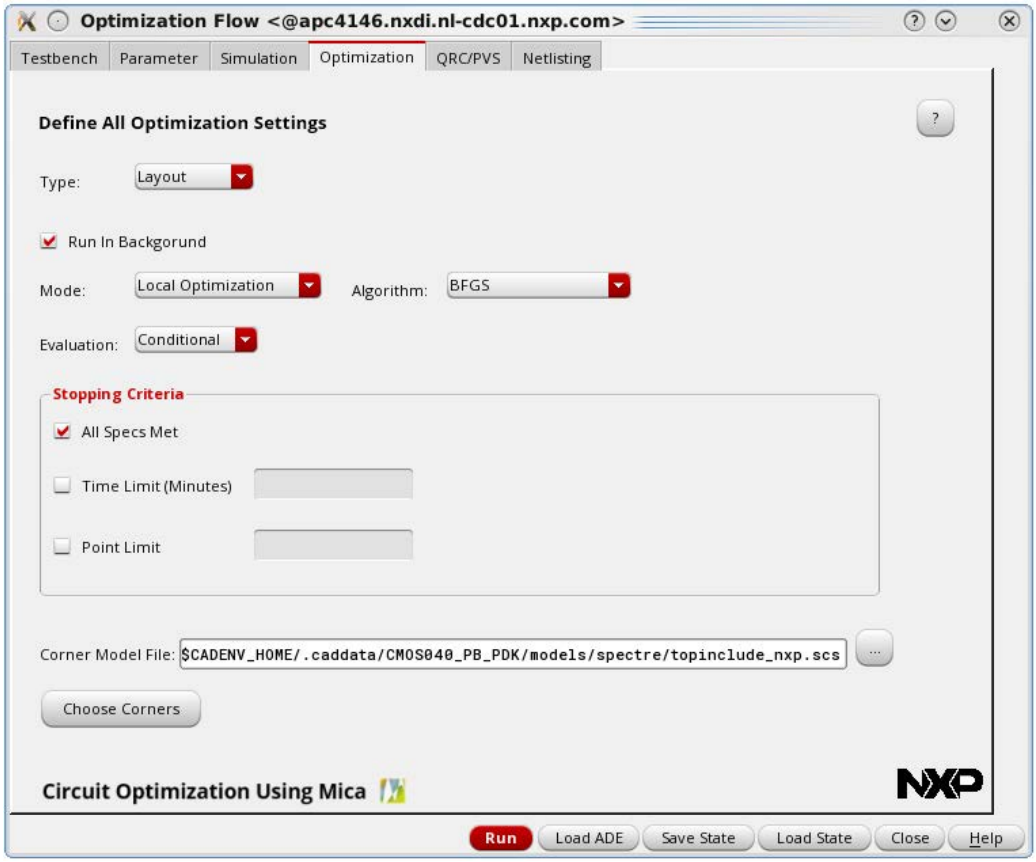

Figure 42: Optimization GUI optimization tab

The *Choose Corners Form* shows all available corners on the left and the selected ones on the right side. To select a corner from the available ones it is necessary to highlight it by clicking on it and use the “>” button. To deselect a corner the same can be performed using the “<” button.

There is the possibility to add variables to certain corners. Therefore the “+” button can be used since it makes the fields on the bottom visible. Then the user can specify the variable and its value(s). To add it to one or more corners the user needs to highlight them in the right field and press the *Add* button. They will be added to the corners as it can be seen for the variable *fo* in Figure 43. That enables the designer to specify custom corners like minimal and maximal temperature, supply voltage etc.

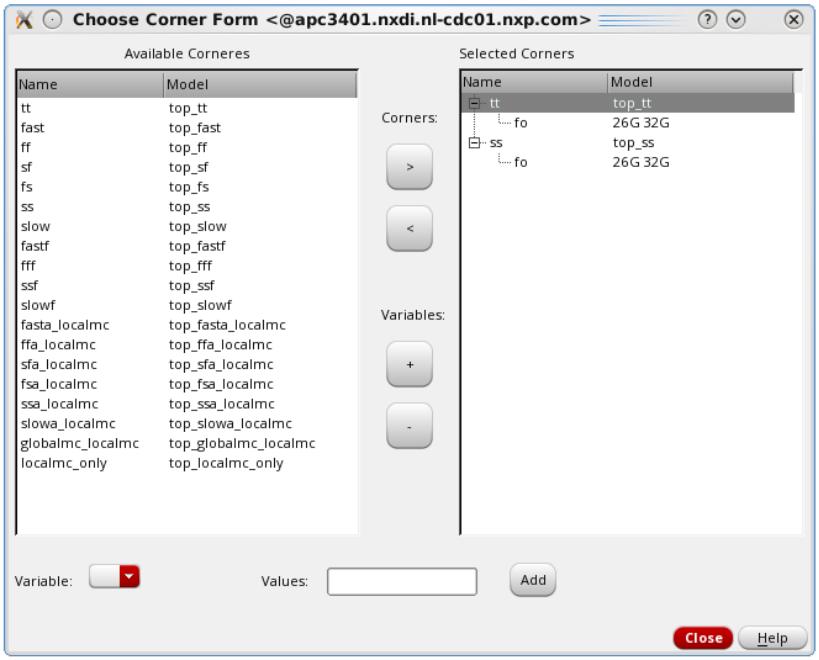

Figure 43: Choose Corner Form

If the *Layout* or *Initial* optimization type from Figure 42 is selected the QRC/PVS and Netlisting tab can be used to define the according settings.

All options regarding the QRC extraction and the for it needed PVS LVS check are defined in the QRC/PVS tab which is visible in Figure 44.

The extraction type determines if only the resistances, or capacitors or both should be extracted from the layout. Depending on the selection different extraction and filtering options for QRC can be set. The rule set specifies the extraction rules which should be used.

The Tec*hnology Library File* defines the used process technology and is used from the installed cadenv package by default. It is required from QRC for the technology information which is used for the extraction of the parasitic components.
In the *LVS QRC Rulefile* the PVS LVS settings including the QRC files creation are set. By default it is used from the cadenv package. The performed LVS check depends on this file since it specifies which rules to use.

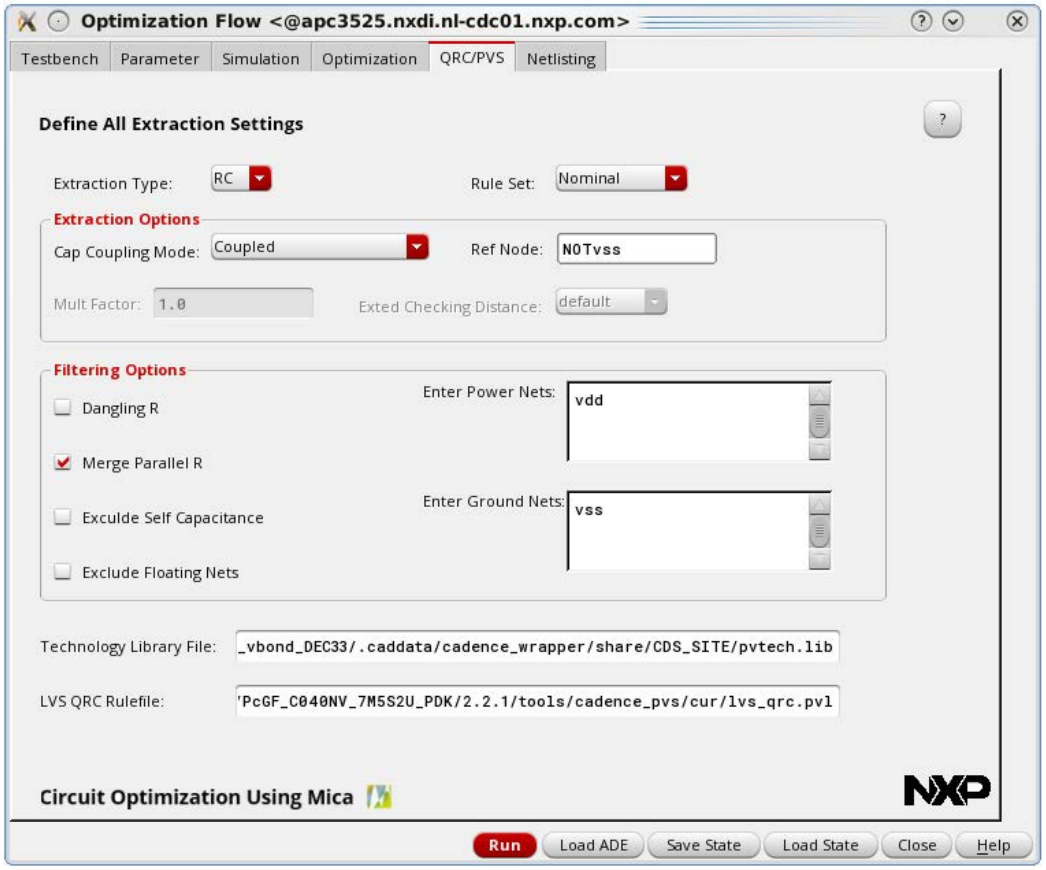

Figure 44: Optimization GUI QRC/PVS tab

The netlister is applied to create the netlist of the exacted layout view, which is then used for the simulation. Therefore, the settings can be defined in the *Netlisting* tab displayed in Figure 45. The selected options are directly fed in the *si* netlister tool.

The *Switch* and *Stop Views* are used to set the environment options of the ADE setup. This means there it can be defined if specific views of the cells placed in the testbench should be used for the netlisting and hence for the simulation. Therefore, the usage of the extracted view can be chosen by adding av_extracted at the beginning of the switch views.

The temporary cell is needed to create the extracted view and the for it needed schematic and layout view. In this cell a schematic and layout view with the current device parameters is created for each optimization cycle. These are then used for the generation of the extracted view which is then netlisted with the settings defined above.

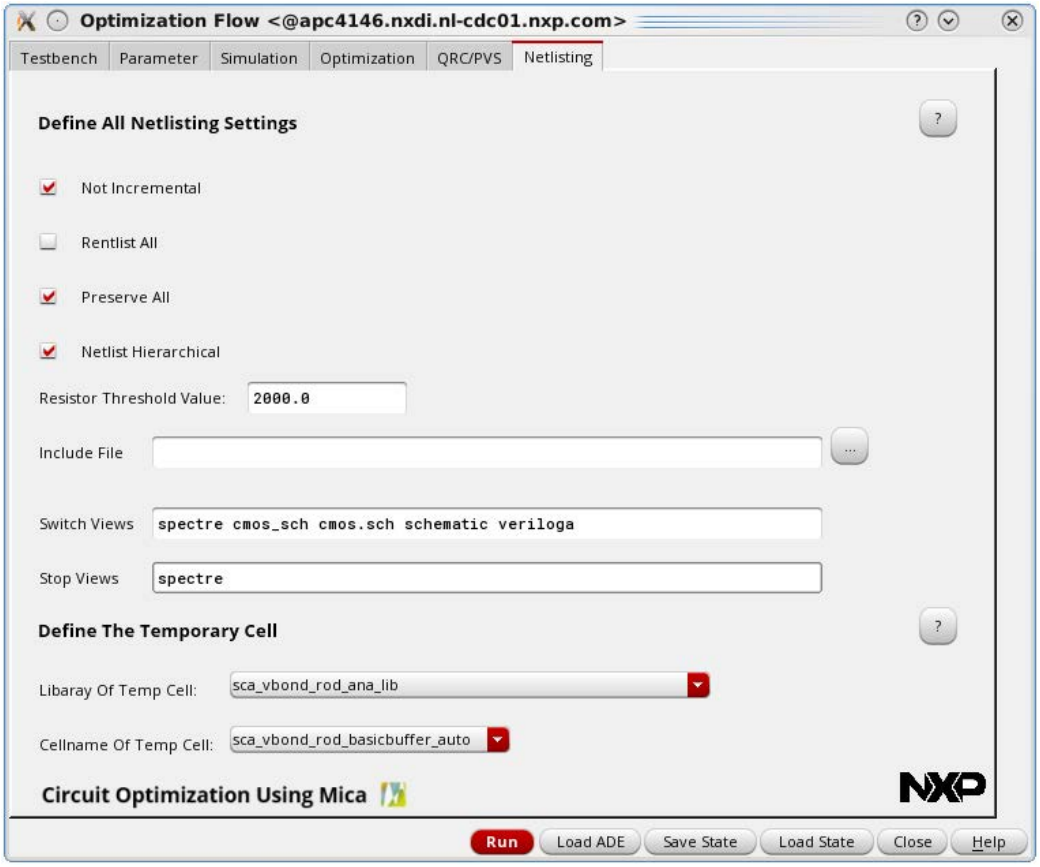

Figure 45: Optimization GUI optimization tab

After the options in all tabs are specified the optimization can be started via the *Run* button on the bottom of the GUI. Afterwards the results are displayed as described in chapter 5.4. If something is not set up correctly and the optimization cannot be performed an error message will open.

5.3 Created Data

The run directory of the automated optimization is located at the following location.

$WORK/rundirAutomatedOptimization/<runName>

The directory structure of the run directory is displayed in Figure 46 where it is visible that each run has a separate sub run directory. Its name includes the start date and time as well as the optimization type. Inside are the log and temporal files created. Depending on the run options there are more or less files and folders created. The results are stored in the file *optResultsTop3.txt* and the log of the background virtuoso is in *backgroundVirt.log*.

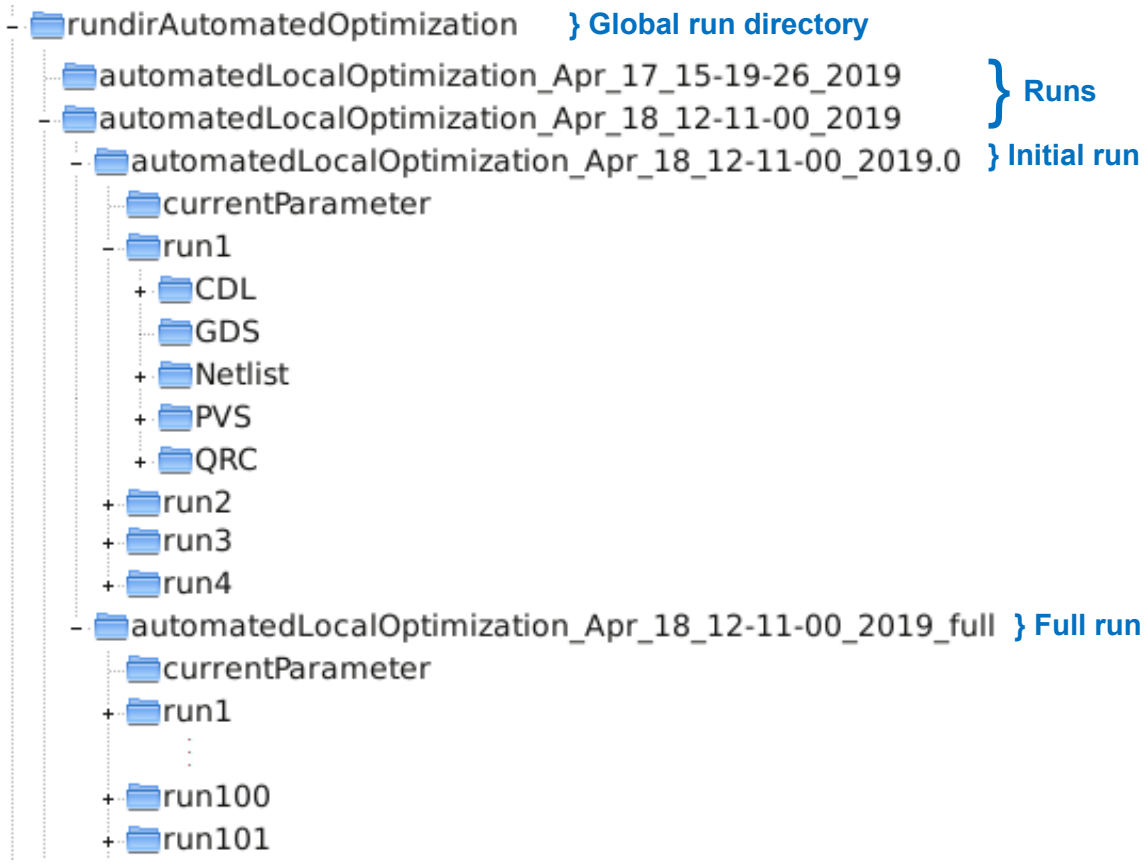

Figure 46: Optimization run directory content

The run directory contains two sub directories with the same name but with either *.0* or *_full* added. The first one is for the initial simulation and only if this one succeeds the full simulation will be started in the second directory. The initial simulation performs only one optimization cycle and stops afterwards. It is used to verify that everything was set up correctly.

If a schematic only optimization was selected there are no further sub directories than the specific run directory. For an initial or layout optimization there is a run directory for each performed simulation, whereby the number at the end is continuously increasing. That is required due to the usage of the pre-run script since it needs to execute several steps which need their own data and run directories. For example, the Netlisting step creates the netlist, its run data and log file in the folder *Netlist*.

The folder *currentParameter* contains text files with the current parameter sets, which are needed since only the changes of parameters from one cycle to another cycle are reported. Therefore, in order to place the PCell it is required to know the previous parameters as well.

5.4 Output

After the optimization finished successfully the results are displayed in the *Optimization Results* form shown in Figure 47. The three best results with the according parameters are presented in a descending order.
For each result the name, value, status and set target is shown. This allows a quick overview if the specifications where satisfied or not.

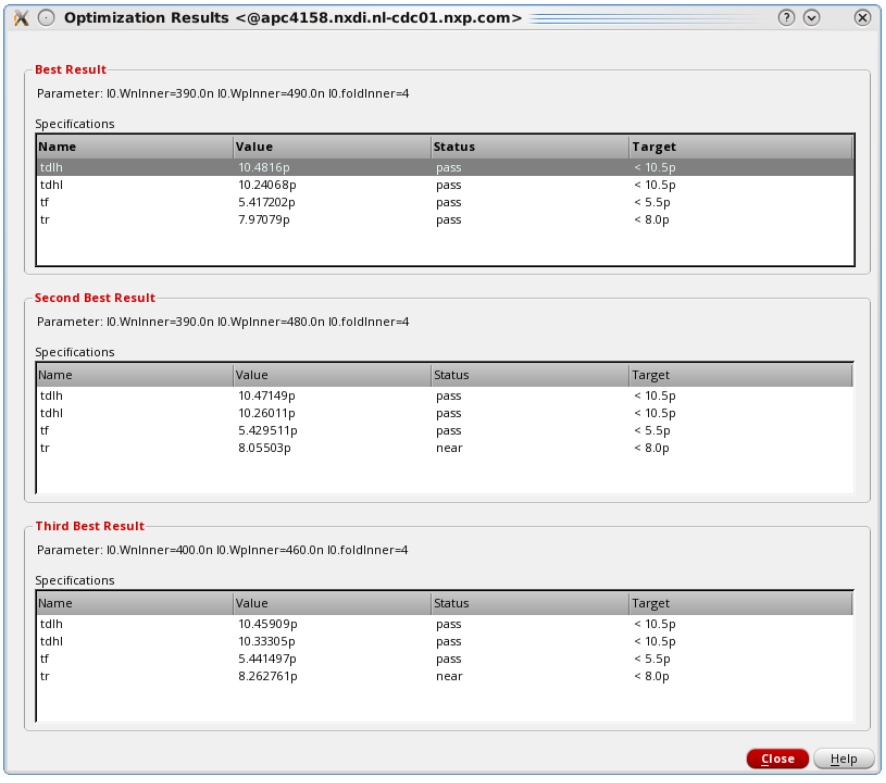

Figure 47: Optimization results GUI

In the testbench cell the view adexl_auto is created which was used for the optimization. After the run has finished it can be opened to get the detailed results and to plot the signals. Therefore, the latest history results must be loaded in ADE as shown in Figure 48.

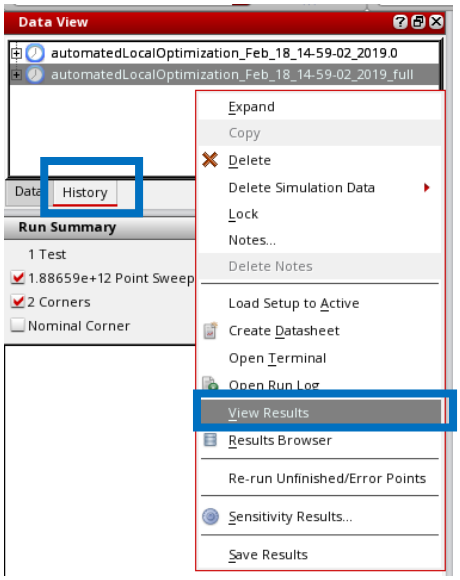

Figure 48: Loading of optimization results

After the latest history was loaded it is shown in the ADE results windows as displayed in Figure 49. The results are placed in a decreasing order with the best one on top.

For further information it is possible to right click the results and open the used netlist, or the created log file.

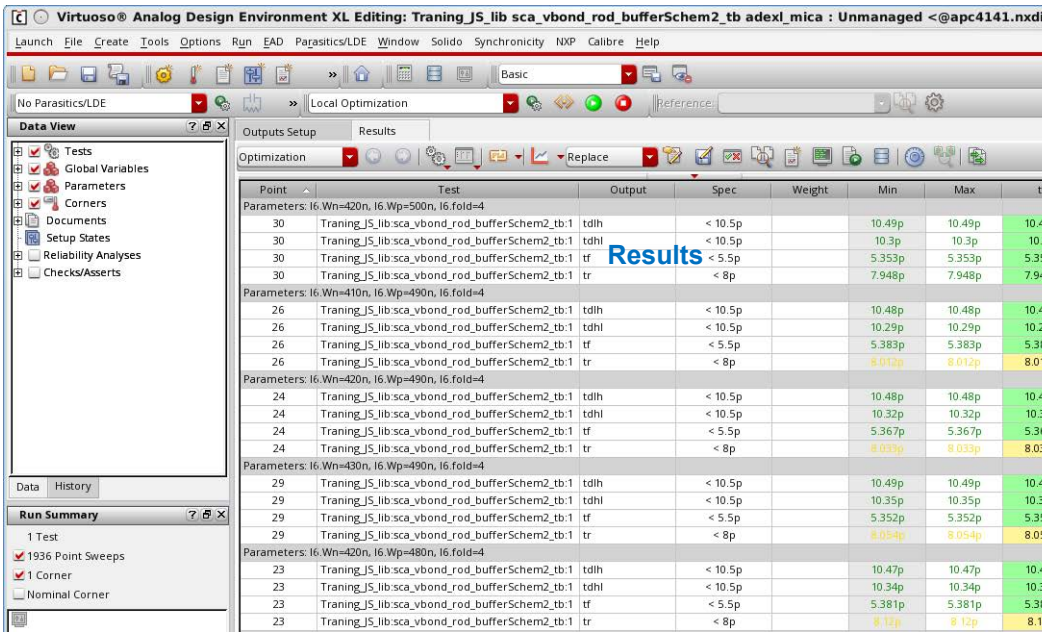

Figure 49: ADE results window

6 Case Study: Divide-by-two Circuit

To verify the optimization flow and the thereout gained improvements a case study on the divide-by-two circuit is conducted. Therefore, the three available optimization modes schematic, initial and layout are performed. Their output results are analyzed and compared with each other to illustrate the pros and cons of each method.

6.1 Testbench

The testbench of the divide-by-two circuit consists of the simplified VCO, the divider itself with its biasing and the buffer etc. with its supply as visible in Figure 50. The VCO and the bias are simplified to reduce the simulation time and computational resources.

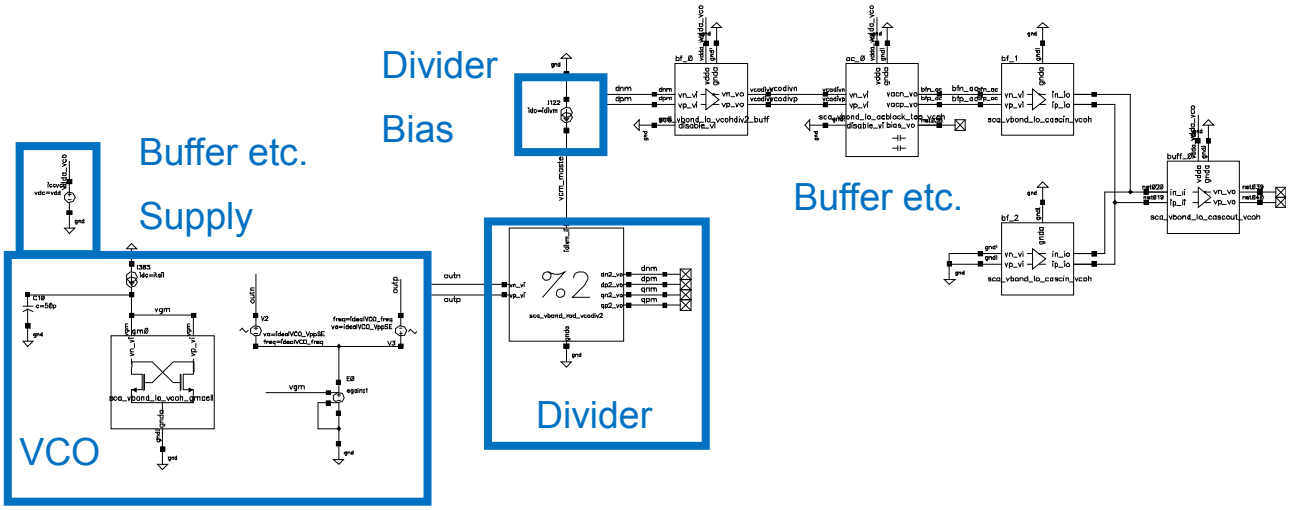

Figure 50: Divide-by-two circuit testbench

The divider is the DUT which parameters are changed to satisfy the performance constraints. The VCO generates its input which is a mirror-inverted 32GHz sinusoidal signal. The output signal of the divider is connected to the buffer stage which is its load. The input frequency must be divided by a factor of two so that the circuit performs its desired operation. If that is not ensured for all corners a redesign is necessary.

6.2 Constraints

The device parameters which are adjusted by the optimization algorithm must stay inside of certain boundaries so that the physical design of the circuit is feasible. Therefore, the parameter constraints in Table 3 are defined. The first is the bias current of the divider which can be swept between 1.0 mA and 2.0 mA. This parameter is special insofar that it can be also found in the output constraints below. This is due the fact, that it must be selected that a correct circuit behavior is given but should also be minimized to lower the power consumption.

The other parameters are from the PCell of the circuit and define the widths of the transistors. Their boundaries ensure that the devices have a good matching and can be built in a symmetrical manner.

Additionally, the start point of each parameter is set which is required for the optimization algorithms as reference point. As mentioned in chapter 2 it is recommended to choose the start point as good as possible in terms of the circuit performance. Therefore, the circuit knowledge and the results of previous simulations must be applied.

Table 3: Divide-by-two circuit parameter constraints

Parameter	Constraint	Start Point
idivm	1m < x < 2.0m	1.375 mA
WN_writeL	2.8µ < x < 4.8µ	3.5 µm
WN_writeR	2.8µ < x < 4.8µ	3.5 µm
WN_senseL	1.4µ < x < 2.4µ	1.5 µm
WN_senseR	1.4µ < x < 2.4µ	1.5 µm
WP	3.5µ < x < 5.5µ	4.0 µm

The actual desired results of the optimization are defined with the output constraints. These set certain boundaries for the equations defined by the designer.

The *FreqPeak* equations on the top must be exactly one which defines that the circuit is in its operational range and the basic function is given. Therefore, they are defined to be in the range of 0.99999 and 1.00001.

VCM_masterdiv and *Icc_masterdiv* define the supply voltage and current respectively which are used to limit the power consumption of the circuit.

The equations for *dp* and *qp* at the bottom are used to ensure that the signals are not distorted in order that the differential information can be calculated. Only the positive output of *d* and *q* is considered, since the negative is affected by the same influences and in the same manner as the positive.

Table 4: Divide-by-two circuit output constraints

Name	Constraint	Weight
dm_FreqPeak	0.99999 < x < 1.00001	100
qm_FreqPeak	0.99999 < x < 1.00001	100
vcodiv_FreqPeak	0.99999 < x < 1.00001	100
vcodiv_MagPeak	0.55 < x < 0.7 V	30
VCM_masterdiv	< 1.2 V	50
Icc_masterdiv	< 1.5m	30
dp_Vpeakmax	0.9 < x < 1.0 V	50
dp_Vpeakmin	< 50 mA	80
qp_Vpeakmax	0.9 < x < 1.0 V	50
qp_Vpeakmin	< 50 mA	80

In Addition, each output has a weight assigned which defines how big its influence on the combined result is. That is necessary since if the first three *FreqPeak* equations are not satisfied the circuit is not dividing by two but by another factor. If that is the case all other constraints are meaningless since the main function is not given. Therefore, these equations have the highest weight what ensures that they are always satisfied. All other outputs have lower weights defined according to their importance.

All simulations are performed for the ss as well the tt process corners to ensure that the divider functions properly even for the slowest MOS devices possible.

The challenge of the constraint definition is on one side to define enough equations to assure the proper circuit function but on the other side not too much to limit the simulation time and the required computational resources.
The same is true for the parameter constraints since they have a substantial influence on the optimization algorithm. It is recommended to analyze the circuit beforehand and only sweep reasonable parameters. Further, parameters which severely change the output (e.g.: multiples of a transistor) should be avoided since most local optimizers will struggle to find the best result.

6.2.1 Reference Point

The reference point for the simulations is chosen according to the parameters displayed in Table 5. The device parameters are known to provide a reasonable performance and therefore can be used as a starting point for the optimization.

Table 5: Divide-by-two circuit reference point parameters

Parameter	Value
idivm	1.375 mA
WN_writeL	3.5 µm
WN_writeR	3.5 µm
WN_senseL	1.5 µm
WN_senseR	1.5 µm
WP	4.0 µm

The output results for the starting point are visible in Table 6. The circuit performs the division by two as the *FreqPeak* equations are exactly one. However, there are further improvements necessary since three constraints are not satisfied. The total performance error is 11.71% and should be minimized ideally to zero percent.

Table 6: Divide-by-two circuit reference point output results

Name	Status	Results ss	Results tt
dm_FreqPeak	Pass	1	1
qm_FreqPeak	Pass	1	1
vcodiv_FreqPeak	Pass	1	1
vcodiv_MagPeak	Near	710.8 mV	745.9 mV
VCM_masterdiv	Pass	1.125 V	1.05 V
Icc_masterdiv	Pass	1.375 mA	1. 375 mA
dp_Vpeakmax	Pass	973.8 mV	940.2 mV
dp_Vpeakmin	Fail	76.33 mV	60.41 mV
qp_Vpeakmax	Fail	1.17 V	1.104 V
qp_Vpeakmin	Pass	-38.69 mV	-37.13 mV
Error [%]		**7.98 %**	**3.73 %**

6.3 Schematic only Optimization

During the schematic optimization the widths of the transistors are modified in order to satisfy the output constraints. However, a closer look at the netlist reveals that multiple parameters of the standard MOS transistors are changed. This is due their dependency of the width and/or fold of the transistor. An example would be the two parameters *as* and *ad* which are the area of the source and drain. They are dependent on the width and fold of the transistor. The same is true for *ps* and *pd* which are the perimeter of drain and source. *nrd* and *nrs* is set as the source/drain resistance square which defines the resistance of the drain/source area and they are also dependent of width and fold.

sa, *sb* and *sd* are STI parameters which define different STI spacings which can be seen in Figure 51. *sa* is the space from the poly of the left side to the end of the active area and *sb* is the one on the right side. *sd* is the spacing between the poly stripes. The sub source and drain areas *as* and *ds 1-2* add up to the overall areas mentioned above.

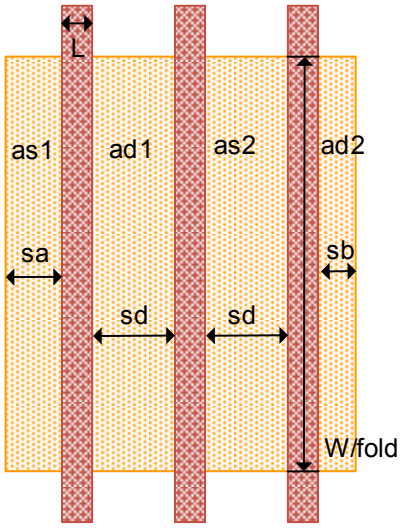

Figure 51: STI parameter

The areas of source and drain are not only dependent to the width and fold of the transistor but also to the STI parameters. Equation (12) shows the detailed formula for the two areas and their dependences of other parameters. There are three cases which depend on the fold of the transistor. The first case is if the fold is one, where the area is the product of the width and *sa*. Dependent if the fold is even or odd two different cases apply. The formulas show that for an even fold there is one more source as drain connection and the same number for an odd fold.

$$ad = \begin{cases} \frac{fold}{2} sd \frac{W}{fold} & fold\ even\ and\ < 1 \\ \frac{W}{fold}\left(\left\lfloor \frac{fold}{2} \right\rfloor sd + sa\right) & fold\ \ odd\ and\ < 1 \\ W\ sa & fold = 1 \end{cases} \qquad as = \begin{cases} \frac{W}{fold}\left(2\ sa + \left(\frac{fold}{2} - 1\right) sd\right) & fold\ even\ and\ < 1 \\ \frac{W}{fold}\left(\left\lfloor \frac{fold}{2} \right\rfloor sd + sa\right) & fold\ \ odd\ and\ < 1 \\ W\ sa & fold = 1 \end{cases} \tag{12}$$

The parameters from above are needed for the transistor model since they define the exact geometry of the device. For example, the area or perimeter of the source is crucial for the capacitance to nearby nodes.

A global optimization was performed which required two hours of simulation time while applying five processes in parallel. This shows that this method is relatively fast and feasible for initial simulations to gather more insights of the circuit. The total amount of 706 points were simulated and which lead to the device parameters displayed in Table 7. It is visible that the layout size increased significantly since all parameters other than *WN_writeL* increased.

These results where to be expected since if a transistor is increased it is possible to drive more current and can work at higher frequencies. However, with the transistor size also its parasitic components increase what could cause a severe issue since they will worsen the performance. Since they are not considered during the optimization the extraction performed in 6.3.1 is necessary to verify the physical design.

Table 7: Divide-by-two circuit schematic optimization parameter results

Parameter	Value	Change
idivm	1.4 mA	1.82%
WN_writeL	2.82 µm	-19.43%
WN_writeR	4.69 µm	34.00%
WN_senseL	2.27 µm	51.33%
WN_senseR	1.72 µm	14.67%
WP	5.4 µm	35.00%

If the circuit is built with the device parameters from above and supplied with a bias current of 1.4 mA the performance results shown in Table 8 are achieved. As mentioned this the ideal schematic only result.

It can be seen that all output constraints are satisfied and thus the circuit works properly. Therefore, the total performance error is zero percent.

Table 8: Divide-by-two circuit schematic optimization output results

Name	Status	Results ss	Results tt
dm_FreqPeak	Pass	1	1
qm_FreqPeak	Pass	1	1
vcodiv_FreqPeak	Pass	1	1
vcodiv_MagPeak	Pass	629.6 mV	678.8 mV
VCM_masterdiv	Pass	1.06 V	983.1 mV
Icc_masterdiv	Pass	1.4 mA	1.4 mA
dp_Vpeakmax	Pass	933 mV	901.1 mV
dp_Vpeakmin	Pass	-3.533 mV	5.695 mV
qp_Vpeakmax	Pass	998.7 mV	952.8 mV
qp_Vpeakmin	Pass	-31.9 mV	-38.64 mV
Error [%]		**0.00 %**	**0.00 %**

Figure 52 shows the graphical verification of the output trough the waveform results. On top the input signal is visible which should be divided by the factor of two.

On the bottom the outputs *d* and *q* are displayed which have exactly the double periodic time and thus half of the frequency. The outputs are no ideal sinusoidal signals but sufficient for the connected buffer and regenerative circuits.

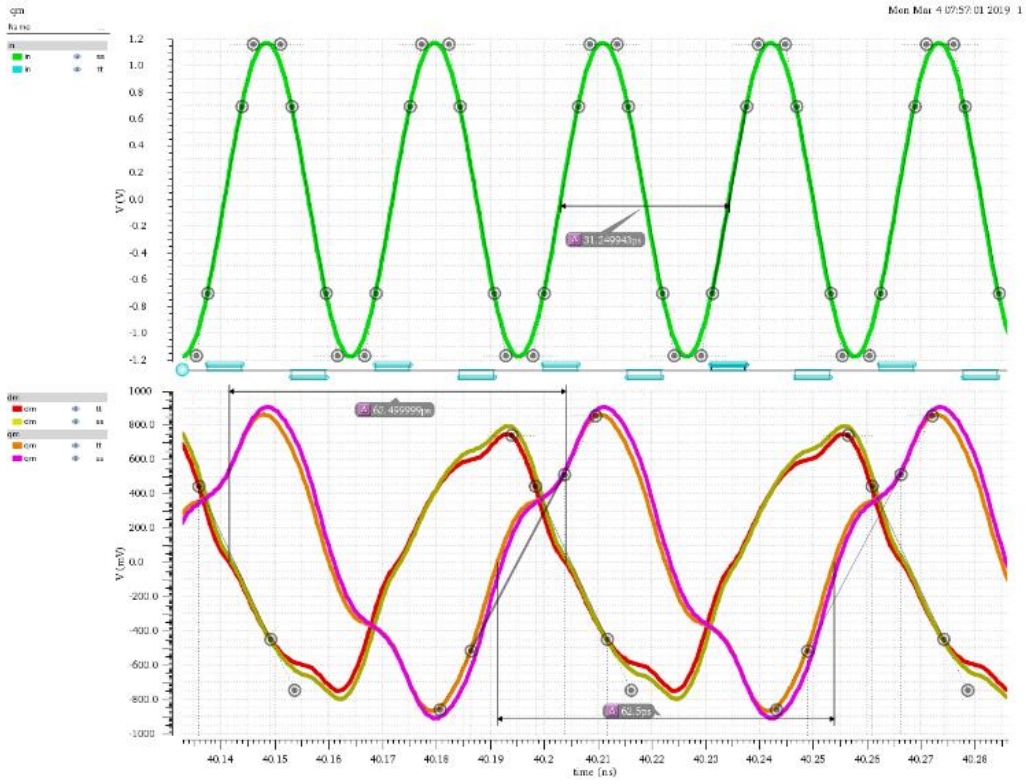

Figure 52: Function verification through waveform results - schematic

As expected from the resulted parameters, in Figure 53 it is visible that the layout of the divide-by-two circuit clearly increased compared with the starting point. The required area of the divider increased by +10.17 % compared with the starting point.
A good matching of the transistors as well their symmetrical placement can be achieved.

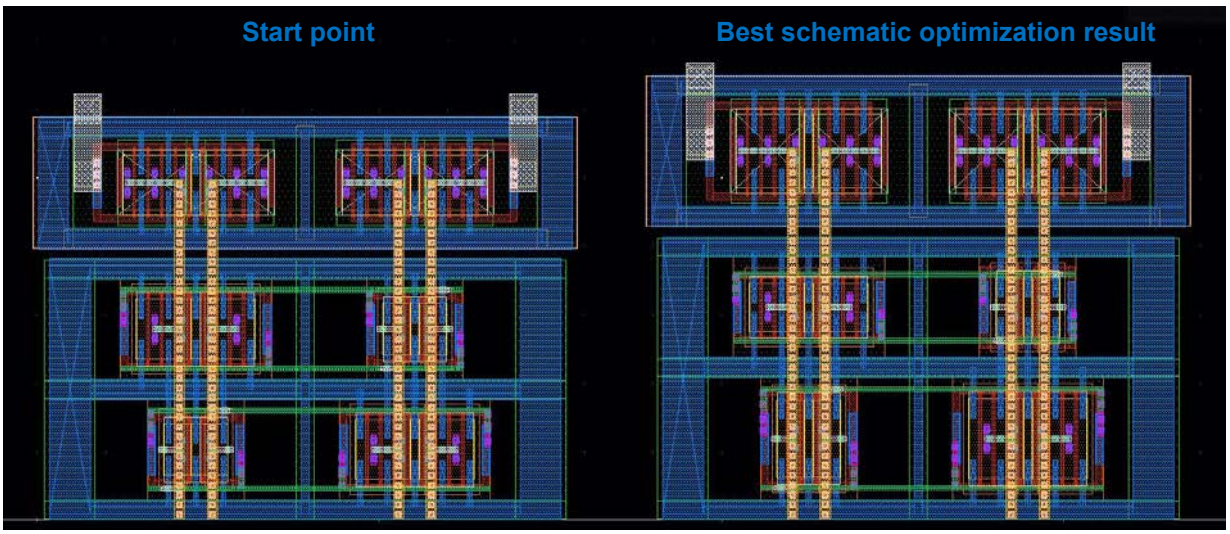

Figure 53: Layout of the best schematic optimization result vs start point

6.3.1 Extraction and Simulation

To verify the results of the schematic only optimization it is required to perform an extraction of the physical design with the resulted device parameters. Therefore, the layout view is created and the parasitic resistors and capacitors are extracted. Afterwards a simulation is set up which uses the extracted view instead of the schematic of the DUT.
The resulting output is displayed in Table 9 where it is visible that the circuit function is not provided at the slow process corner. In general, the circuit performance is not sufficient since nearly all constraints are failing. Therefore, a redesign is required in order that the circuit works correctly for both corners.
This would need a new analysis of the circuit to generate a new parameter set. That is then used to create a new layout view which is then again extracted and simulated. Since the creation of the layout is usually done manually this provokes a major expenditure of time.

Table 9: Divide-by-two circuit extracted output results

Name	Status	Results ss	Results tt
dm_FreqPeak	Fail	0	1
qm_FreqPeak	Fail	2	1
vcodiv_FreqPeak	Fail	0	1
vcodiv_MagPeak	Fail	45.79 mV	457.4 mV
VCM_masterdiv	Pass	1.1 V	1.014 V
Icc_masterdiv	Pass	1.4 mA	1.4 mA
dp_Vpeakmax	Fail	583.9 mV	753.5 mV
dp_Vpeakmin	Fail	75.63 mV	33.59 mV
qp_Vpeakmax	Fail	788.3 mV	760.1 mV
qp_Vpeakmin	Fail	476.4 mV	216.2 mV
Error [%]		**167.89%**	**44.82 %**

The results above show the significant influence of the parasitic components and the limits of the schematic only optimization. Therefore, it is necessary to include the parasitic influences or at least an approximation of them in the simulations.

6.4 Initial Layout Optimization

To avoid the redesign issue elaborated in the previous chapter and to limit the required resources at the same time an initial layout optimization can be used.

Therefore, the capacitors of the initial layout are extracted and added to the netlist. This gives the advantage that the layout is not required to be a PCell and it is generally applicable.

Figure 53 displays the master latch of the divider which is updated with the capacitors of the initial layout. It is visible that several capacitors are added to the schematic and even though each on its own has a small capacitance their general influence on the circuit is significant.

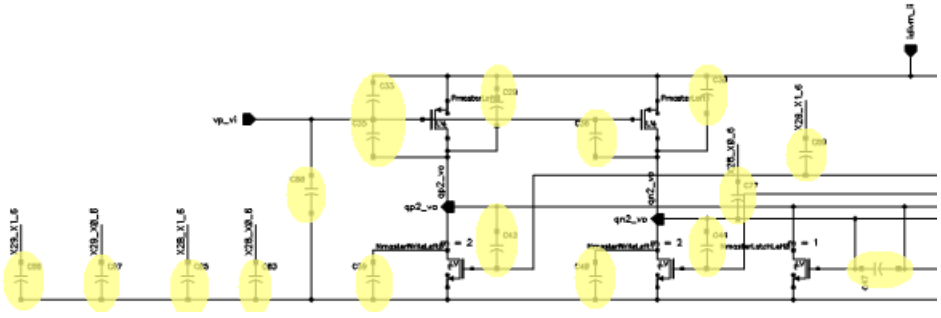

Figure 54: Divide-by-two circuit master latch with initial capacitors

This enables to include an approximated influence of the parasitic components into the simulations without drastically increasing the netlist and thus simulation complexity.
However, exclusively the parasitic capacitors are considered and they are only valid for the initial layout. That has the consequence, that the physical design needs a complete extraction and validation after the initial optimization finished and the layout was built.
A global optimization which took around eighteen hours while applying ten processes in parallel was performed. In total 5815 simulations where carried out to gain the parameter results displayed in Table 10. In contrast to the schematic only optimization not all parameters increased, but some decreased in size due the approximated influence of the parasitic components.

Table 10: Divide-by-two circuit initial layout optimization parameter results

Parameter	Value	Change
idivm	1.68 mA	22.18%
WN_writeL	2.8 µm	-20.00%
WN_writeR	3.36 µm	-4.00%
WN_senseL	1.74 µm	16.00%
WN_senseR	1.8 µm	20.00%
WP	3.85 µm	-3.75%

The results of the simulation are visible in Table 11 where it is apparent that not all constraints could be satisfied. A total error of 1.98 % is remaining. All constraints are nearly reached, except the bias current since it has a high influence on the circuit but a low weight.

Table 11: Divide-by-two circuit initial layout optimization output results

Name	Status	Results ss	Results tt
dm_FreqPeak	Pass	1	1
qm_FreqPeak	Pass	1	1
vcodiv_FreqPeak	Pass	1	1
vcodiv_MagPeak	Near	658.1 mV	708.5 mV
VCM_masterdiv	Near	1.215 V	1.13 V
Icc_masterdiv	Fail	1.68 mA	1.68 mA
dp_Vpeakmax	Near	930 mV	897.1 mV
dp_Vpeakmin	Pass	32.26 mV	24.3 mV
qp_Vpeakmax	Pass	990.4 mV	956 mV
qp_Vpeakmin	Pass	45.63 mV	9.415 mV
Error [%]		**0.99 %**	**0.99 %**

6.4.1 Extraction and Simulation

In order to verify the obtained results an RC extraction of the divide-by-two circuit with the parameters of the initial optimization is performed. The resulting output is displayed in Table 12 where it can be seen that still nearly all constraints are satisfied.

Table 12: Divide-by-two circuit initial extraction results

Name	Status	Results ss	Results tt
dm_FreqPeak	Pass	1	1
qm_FreqPeak	Pass	1	1
vcodiv_FreqPeak	Pass	1	1
vcodiv_MagPeak	Near	655.1 mV	708 mV
VCM_masterdiv	Near	1.235 V	1.15 V
Icc_masterdiv	Fail	1.68 mA	1.68 mA
dp_Vpeakmax	Near	923.9 mV	892.2 mV
dp_Vpeakmin	Pass	29.92 mV	22.28 mV
qp_Vpeakmax	Pass	986.8 mV	953.8 mV
qp_Vpeakmin	Fail	73.76 mV	31.01 mV
Error [%]		**1.11 %**	**1.05 %**

In comparison with the schematic only optimization, the influence of the full extraction is far less significant. This is because the parasitic capacitors are already part of the netlist and provide a good approximation. Especially, since the circuit architecture is fixed and only the widths of the transistors are modified the initial optimization is particularly feasible.

The initial optimization type offers a good tradeoff between simulation resources and the output accuracy. Therefore, its usage is recommended for cells with moderate significance to gain improved parameters for the physical design.
Additionally, it can be used to generate a good starting point for a local layout optimization of critical circuits. This avoids the necessity of a global layout optimization which is time and resource demanding.

6.5 Layout Optimization

In order to completely avoid possible redesigns and to get the highest accuracy possible a local optimization of the physical design of the layout view is performed. Therefore, the PCell is each cycle placed with the current parameters and extracted. The resulting netlist is used for the simulations and thus includes the parasitic influences.
A total of 242 points were completely simulated in a duration of approximately twenty-four hours. Figure 13 displays the resulting device parameters of the divide-by-two circuit. It can be seen, that in comparison with the initial optimization no parameter decreased, but all increased or stayed the same. This can be explained, since if the transistor is to small it cannot drive the connected load which is increased since all parasitic capacitors and resistors are considered during the simulation.

Table 13: Divide-by-two circuit layout optimization parameter results

Parameter	Value	Change
idivm	1.68 mA	22.18%
WN_writeL	3.5 µm	0.00%
WN_writeR	4.092 µm	16.91%
WN_senseL	2.14 µm	42.67%
WN_senseR	1.95 µm	30.00%
WP	4.21 µm	5.25%

In Figure 14 the simulation results of the divide-by-two circuit applying the parameters from above are visible. The total error is reduced down to 1.37%. All constraints except the bias current and *dp_Vpeakmax* are satisfied. The maximum peak of the *dp* signal is very close at the constraint boundary so it can be accepted. Only the biasing is twelve percent above the selected limit. But as it is a secondary constraint and has the lowest weight of all constraints the result is valid.

Since the RC parasitics are already considered during the optimization process no further extraction and validation of the physical design for the selected constraints is necessary. It would only lead to the same results. The acquired parameter can be used for further simulations and corner analyses which were not possible to be part of the optimization since of their complexity.

Table 14: Divide-by-two circuit layout optimization output results

Name	Status	Results ss	Results tt
dm_FreqPeak	Pass	1	1
qm_FreqPeak	Pass	1	1
vcodiv_FreqPeak	Pass	1	1
vcodiv_MagPeak	Pass	626.8 mV	673.2 mV
VCM_masterdiv	Pass	1.191 V	1.107 V
Icc_masterdiv	Fail	1.68 mA	1.68 mA
dp_Vpeakmax	Near	900.4 mV	865.3 mV
dp_Vpeakmin	Pass	44.27 mV	33.09 mV
qp_Vpeakmax	Pass	953.1 mV	916.8 mV
qp_Vpeakmin	Pass	26.07 mV	-513 µV
Error [%]		**0.54%**	**0.83 %**

In Figure 55 the waveform results of the input and output of the divide-by-two circuit applying the results of the layout optimization are visible. It is apparent that the input signal is divided by the factor of two and that the output signal is of good grade. This means the signal is not distorted and can be used by the connected circuits.

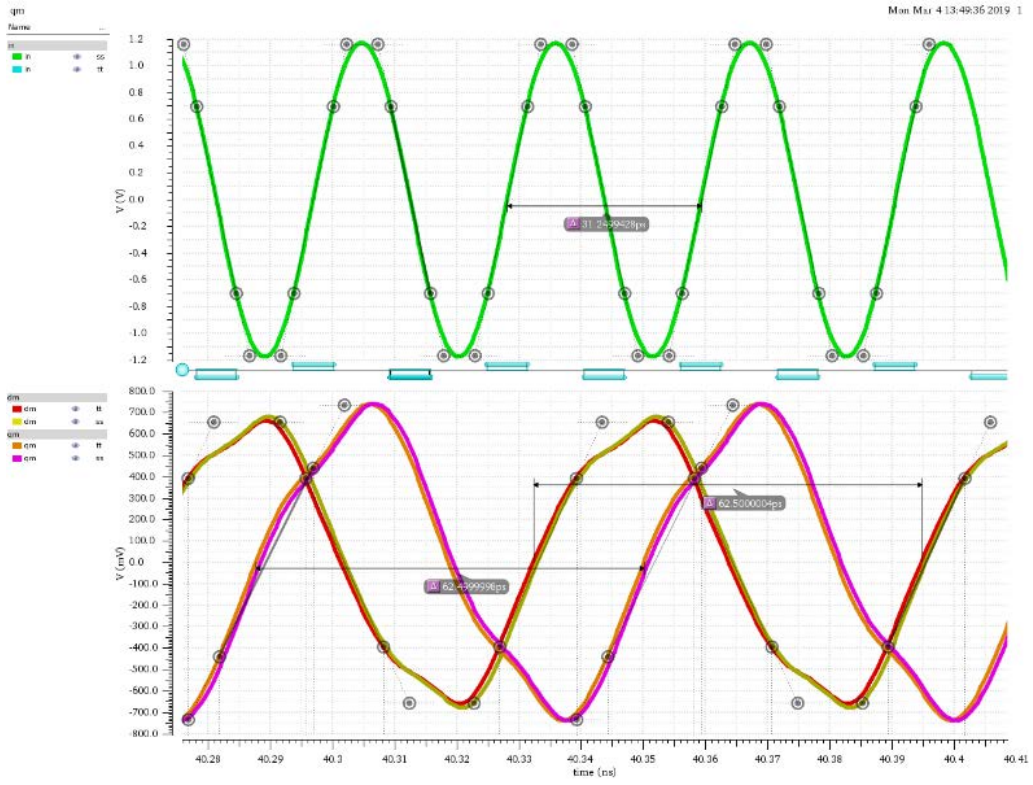

Figure 55: Function verification through waveform results - layout

The change in the required area of the physical design is displayed in Figure 56. An area increase of 3.43 % is necessary which is applicable since the total size of the circuit is rather small.

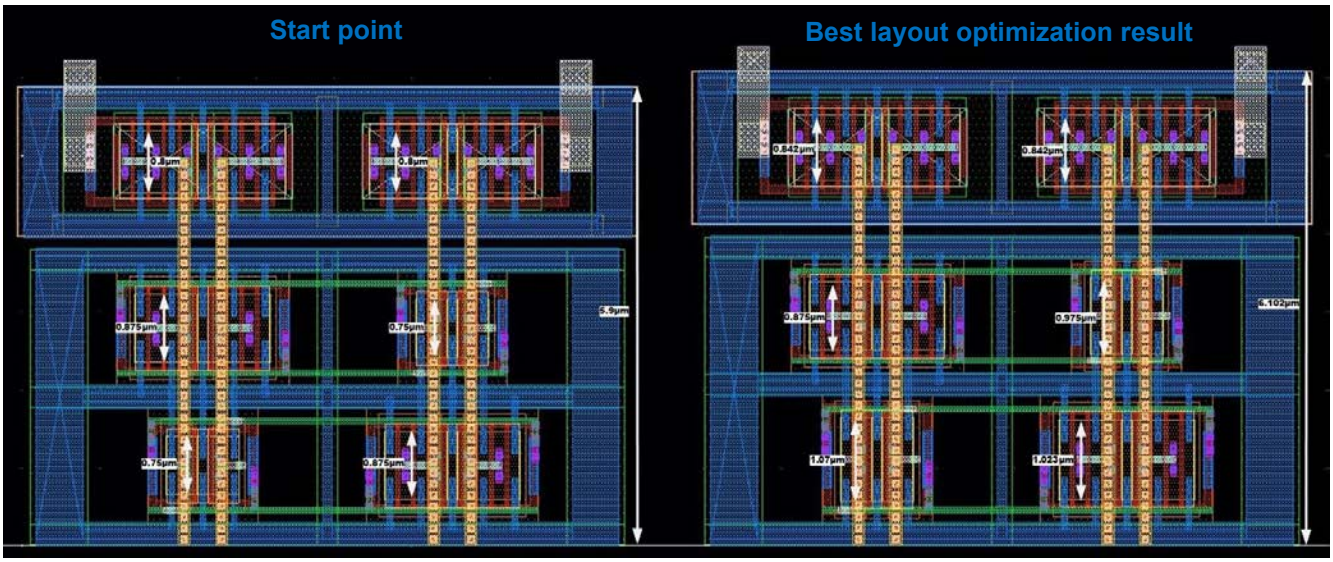

Figure 56: Layout of the best layout optimization result vs start point

The optimization of the physical design which includes the parasitic influences in the optimization is proven to provide the best results. Especially for small but critical circuits it is recommended to apply this optimization type since the accuracy is unmatched by the others and it limits the required circuit validation.

However, it requires high computational resources and simulation time. A global simulation will take several days so it must be verified if it is reasonable. If a local optimization can be applied the investment pays off since the circuit can be increased significantly and the simulation time is limited.

7 Conclusion and Outlook

Physical design always introduces parasitic components which significantly influence the circuit performance. There can be measures taken in order to reduce them, but they can never be eliminated because of the layer structure of the IC. Therefore, it is essential to consider them during the circuit design.
Instead of spending unnecessarily extended design effort on the schematic level it is beneficial to include the physical design, as it contains all parasitic components. This however, requires at least an initial layout but provides more accurate simulation results.
The ideal flow would replace manual creation of the layout by the application of ROD to generate a PCell. That can be used for a physical design optimization which includes all parasitic effects and provides the best output performance possible. If the DUT is not yet programmed in Skill or is not that critical, an initial simulation is recommended. It approximates the parasitic influences by including the parasitic capacitors of the initial layout into the simulation.
To enable a general employment of the automated physical optimization flow a cadenv package is created. This installation package allows the designers to easily set up the optimization flow for different project environments.

Additional features could be implemented in the future, like the possibility to probe sublevel currents to enable their usage in the output equations which are optimized. Further, the optimization flow could be directly integrated in ADE or a similar simulation environment. This would enable the employment of the simulation setups created by the designers. Without the need to start any other tool it would be possible to configure and run a layout optimization.
Because of the potential of the physical design optimization, especially for critical RF circuits and the user friendliness of the GUI, it is to be assumed that it will be applied in the future for critical circuit designs.

List of Figures

Figure 1: Analog circuit design flow 2
Figure 2: Parasitic NMOS capacitors 3
Figure 3: Example for metal resistance of wiring 5
Figure 4: Optimization flow 6
Figure 5: ROD PCell example 8
Figure 6: Virtuoso GUI example 9
Figure 7: Local and global minimum 12
Figure 8: Evolutionary algorithms basic flow 17
Figure 9: Inverter schematic 18
Figure 10: Initial optimization method 19
Figure 11: Initial schematic updated with parasitic capacitors 19
Figure 12: Testbench and DUT relation 21
Figure 13: Inverter testbench 22
Figure 14: ADE environment options 22
Figure 15: ADE main window 23
Figure 16: ADE original optimization flow 26
Figure 17: Conditional Evaluation Flow 27
Figure 18: Pre-run script flow 28
Figure 19: Defining a Pre-Run Script graphical 29
Figure 20: PCell example NMOS RF transistor 30
Figure 21: PCell hierarchy example 31
Figure 22: PCell code structure 32
Figure 23: Extracted view of the inverter 33
Figure 24: LVS-QRC option 35
Figure 25: Start menu of QRC 35
Figure 26: PLL structure 36
Figure 27: Divide-by-two circuit structure 37
Figure 28: Divide-by-two circuit PCell symbol 37
Figure 29: Divide-by-two circuit PCell schematic 38
Figure 30: Divide-by-two circuit schematic right half 39
Figure 31: Divide-by-two circuit input vs output signals 40
Figure 32: Divide-by-two circuit PCell layout 41
Figure 33: Divide-by-two circuit PCell parameter 42
Figure 34: Automatization flow 43
Figure 35: Default ADE optimization flow 44

Figure 36: Pre-run flow 45
Figure 37: Optimization GUI Testbench Tab 47
Figure 38: Optimization GUI Calculator Usage 48
Figure 39: Optimization GUI parameter tab 49
Figure 40: Choose parameter GUI 50
Figure 41: Optimization GUI simulation tab 51
Figure 42: Optimization GUI optimization tab 52
Figure 43: Choose Corner Form 53
Figure 44: Optimization GUI QRC/PVS tab 54
Figure 45: Optimization GUI optimization tab 55
Figure 46: Optimization run directory content 56
Figure 47: Optimization results GUI 57
Figure 48: Loading of optimization results 58
Figure 49: ADE results window 58
Figure 50: Divide-by-two circuit testbench 59
Figure 51: STI parameter 63
Figure 52: Function verification through waveform results - schematic 65
Figure 53: Layout of the best schematic optimization result vs start point 66
Figure 54: Divide-by-two circuit master latch with initial capacitors 67
Figure 55: Function verification through waveform results - layout 71
Figure 56: Layout of the best layout optimization result vs start point 72

List of Tables

Table 1: Sheet resistances ____ 4
Table 2: Divide-by-two circuit parameter restrictions ____ 42
Table 3: Divide-by-two circuit parameter constraints ____ 60
Table 4: Divide-by-two circuit output constraints ____ 61
Table 5: Divide-by-two circuit reference point parameters ____ 62
Table 6: Divide-by-two circuit reference point output results ____ 62
Table 7: Divide-by-two circuit schematic optimization parameter results ____ 64
Table 8: Divide-by-two circuit schematic optimization output results ____ 65
Table 9: Divide-by-two circuit extracted output results ____ 67
Table 10: Divide-by-two circuit initial layout optimization parameter results ____ 68
Table 11: Divide-by-two circuit initial layout optimization output results ____ 68
Table 12: Divide-by-two circuit initial extraction results ____ 69
Table 13: Divide-by-two circuit layout optimization parameter results ____ 70
Table 14: Divide-by-two circuit layout optimization output results ____ 71

References

ALLSTOT, D.J., CHOI, K., AND PARK, J. 2003. *Parasitic-Aware Optimization of CMOS RF Circuits*. Kluwer Academic Publishers.

BOYD, S. AND VANDENBERGHE, L. 2009. *Convex Optimization*. Cambridge university press.

CADENCE, D.S. 2014. Virtuoso Analog Design Environment GXL. https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/custom-ic-analog-rf-design/virtuoso-analog-design-environment-gxl-ds.pdf.

CADENCE, D.S. 2018a. Running a Local Optimization. *Virtuoso ADE Assembler User Guide ICADV12.3*. https://support.cadence.com/apex/techpubDocViewerPage?xmlName=assembler.xml&title=Virtuoso%20ADE%20Assembler%20User%20Guide%20--%2020%20-%20Running%20a%20Local%20Optimization&hash=1068533&c_version=ICADV12.3&path=assembler/assemblerICADV12.3/asmOptimize.html#1068533.

CADENCE, D.S. 2018b. Virtuoso ADE Assembler User Guide. https://support.cadence.com/apex/techpubDocViewerPage?xmlName=assembler.xml&title=Virtuoso%20ADE%20Assembler%20User%20Guide%20--%2020%20-%20Running%20a%20Local%20Optimization&hash=1068533&c_version=ICADV12.3&path=assembler/assemblerICADV12.3/asmOptimize.html#1068533.

CADENCE, D.S. 2018c. Virtuoso Analog Design Environment XL User Guide. https://support1.cadence.com/tech-pubs/Docs/adexl/adexlICADV12.3/adexl.pdf.

CHEN, W.-K. 2003. *Analog Circuits and Devices*. CRC Press LLC.

HASSOUN, M. 1995. *Fundamentals of Artificial Neural Networks*. MIT Press, London.

HUNTER, J.K. 2014. *An Introduction to Real Analysis*. University of California at Davis.

JOSHI, H., RANJAN, S.M., AND NATH, V. 2012. Design of High Speed Flip-Flop Based Frequency Divider for GHz PLL System: Theory and Design Techniques in 250nm CMOS Technology. *International Journal of Electronics and Computer Science Engineering*.

KEIKHA, M.M. 2011. Improved Simulated Annealing using Momentum Terms. IEEE.

KOZIEL, S. AND YANG, X.-S. 2011. *Computational Optimization, Methods and Algorithms*. Springer, Berlin Heidelberg.

KRUMKE, S.O. 2004. *Nonlinear Optimization*. Technical University of Kaiserslautern.

KUNDERT, K. AND ZINKE, O. 2004. *The Designer's Guide to Verilog-AMS*. Springer US.

LAMPAERT, K., GIELEN, G., AND SANSEN, W. 1991. *Analog Layout Generation for Performance and Manufacturability*. Springer Science+Business Media LLC.

LANGNER, K. AND SCHEIBLE, J. 2017. Formal Verification of a Transistor PCell. IEEE.

LOURENÇO, R., LOURENÇO, N., AND HORTA, N. 2015. *AIDA-CMK: Multi-Algorithm Optimization Kernel Applied to Analog IC Sizing*. Springer.

NGUYEN, Q. 2008. *CAD Scripting Languages*. RAMACAD INC.

OXFORDDICTIONARIES. Definition of optimization in English. https://en.oxforddictionaries.com/definition/optimization.

PANDIT, S., MANDAL, C., AND PATRA, A. 2014. *Nano-scale CMOS Analog Circuits: Models and CAD Techniques for High-Level Design*. CRC Press.

QUARTERONI, A., SACCO, R., AND SALERI, F. 2007. *Numerical Mathematics*. Springer.

RAZAVI, B. 1998. *RF Microelectronics*. Prentice Hall.

RAZAVI, B. 2017. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill Education.

RAZAVI, B., LEE, K.F., AND YAN, R.H. 1995. Design of high-speed, low-power frequency dividers and phase-locked loops in deep submicron CMOS. IEEE.

RUSZCZYNSKI, A. 2006. *Nonlinear Optimization*. Princeton University Press.

SNYMAN, J.A. AND WILKE, D.N. 2018. *Practical Mathematical Optimization*. Springer.

SÖSER, P., WINKLER, G., PRIBYL, W., KRASSER, E., AND HARTL, H. 2008. *Elektronische Schaltungstechnik*. Pearson Studium.

TAYENJAM, S., VANUKURU, V.N.R., AND S., K. 2017. A PCell Design Methodology for Automatic Layout Generation of Spiral Inductor using SKILL Script. IEEE.

THIEL, D.V. AND SMITH, S. 2002. *Switched Parasitic Antennas for Cellular Communications*. Artech House.

VIKHAR, P.A. 2016. Evolutionary Algorithms: A Critical Review and its Future Prospects. IEEE.

WONG, B.P., MITTAL, A., STARR, G., AND CAO, Y. 2005. *Nano-CMOS circuit and physical design*. John Wiley & Sons, Inc.

XING TAN, G. AND MAO, Z.-Y. 2005. Study on Pareto front of multi-objective optimization using immune algorithm. IEEE.

YU, W. AND WANG, X. 2014. *Advanced Field-Solver Techniques for RC Extraction of Integrated Circuits*. Springer.