FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Physical Design Implementation and Engineering Change Order Flow

Afonso Moreira

FOR JURY EVALUTATION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: José Carlos Alves

Company Supervisor: Julien Le-Coz

30/07/17

Resumo

Devido à melhoria na fabricação de semicondutores, apareceram transistores na escala do nanômetro que levaram à redução da área de um IC e ao aumento da velocidade de comutação de cada componente. Mesmo que a integração de uma quantidade tão grande de transistores num único IC tenha trazido muitas vantagens, os desafios impostos à indústria microeletrônica são incomensuráveis.

Uma das implicações do rápido crescimento na complexidade do design VLSI é que alterações no design tornaram-se inevitáveis. Quando eles ocorrem após a fabricação das mascaras usadas no processo de fotolitografia, não é viável refabricar todas as mascara, pois isso levaria a custos adicionais elevadíssimos.

Desta forma, a maioria das empresas de VLSI dependem de ECO flows para executar alterações após o tape-out. Com esse objetivo em mente, foi desenvolvido um post-mask ECO flow que implementa modificações funcionais de ECO feitas no RTL original sem que seja necessário alterar as mascaras das camadas base (camadas não-metálicas).

O passo preliminar de um ECO flow é a inserção de standard cells redundantes no design. Assim, um IP foi implementado fisicamente num ambiente industrial e, durante sua implementação, filler cells reconfiguraveis inovadoras foram colocadas na àrea do core para antecipar a possibilidade de modificações pós-tapeout.

Posteriormente, perto do prazo final do tapeout deste IP, as partes front-end e back-end do ECO flow foram implementadas. Enquanto o front-end flow gera uma nova netlist baseada na netlist original e na modificada, o back-end flow implementa fisicamente as modificações desejadas e gera um novo layout.

Para validar a correção do post-mask ECO flow implementado, verificou-se que as modificações feitas no RTL foram corretamente implementadas e que apenas as camadas de metal tiveram que ser alteradas para fazê-lo, comparando o novo layout com o original camada a camada. Consequentemente, qualquer modificação post-mask furura pode ser realizada simplesmente alterando o RTL original, executando o ECO flow e analisando os resultados.

Abstract

Due to the improvement in the manufacturing of semiconductors, nanometer scale transistors appeared, leading to the reduction of the area of an IC and an increase in the switching speed of each component. Even though the integration of such an enormous amount of transistors in a single IC brought many advantages, the challenges imposed to the microelectronic industry are incommensurable.

One of the implications of the rapid growth in VLSI design complexity is that design changes became unavoidable. If they occur after the fabrication of the photomasks used in the photolithography processing step of the semiconductor device manufacturing, it is not viable to refabricate the entire mask set as that would lead to huge added costs.

Therefore, most VLSI corporations rely heaviliy on Engineering Change Order (ECO) flows to perform modifications after the tape-out. With that objective in mind, a post-mask functional ECO flow that implements ECO modifications done on the original RTL without having to change the base layers (non-metal layers) was developed. The preliminary step of an ECO flow starts with the insertion of redundant standard cells in the design. Thus, an IP was physically implemented in an industrial environment and, during its implementation, state-of-the-art reconfigurable filler cells were placed in the core area to anticipate the possibility of post-tapeout modifications. Afterwards, near the deadline of the tapeout of this IP, both the front-end and back-end parts of the ECO flow were implemented. While the front-end flow generates a new netlist based on the original netlist and the modified one, the backend flow will physically implement the desired modifications and generate a new GDS layout.

In order to validate the correctness of the implemented post-mask ECO flow, it was verified that the modifications done on the RTL were correctly implemented and that only the metal-layers had to be changed to do so by comparing the new layout with the original one layer be layer. Consequently, any future post-mask ECO changes can be simply performed by changing the RTL, running the flow and analyzing the results, decreasing the time-to-market of the chip considerably.

Acknowledgements

First and foremost, I would like to express my sincere gratitude for my advisor in STMicroelectronics, Julien Le-Coz. Not only as a mentor and for teaching me almost everything I know about physical design in an industry environment, but also as a friend and for helping me tremendously in my adaptation to a new country. He was the one who gave me the amazing opportunity to come to France, so without him, this dissertation definitely would not be possible.

My sincere thanks also goes to my professor José Carlos Alves for its guidance and advice which helped me shape the path to take on this internship and on every aspect related with this dissertation.

Besides my advisers, I would like to thank the rest of the ST team for including me as one of their own even if my french is almost non-existent.

Most importantly, none of this could have happened without my friends and family who helped me grow and supported me in all of my important life-decisions.

Afonso Moreira

"You have power over your mind - not outside events.
Realize this, and you will find strength."

Marcus Aurelius

Contents

List of Figures

List of Tables

Abbreviations

IC	Integrated Circuit
VLSI	Very-large-scale integration
ECO	Engineering Change Order
P&R	Place & Route
CTS	Clock Tree Synthesis
EDA	Electronic Automation Design
RTL	Register Transfer Level
HDL	Hardware Description Language
LEF	Library Exchange Format
DEF	Design Exchange Format
SDC	Synopsys Design Constrains
UPF	Unified Power Format
CPF	Common Power Format
GA	Gate-Array
SoC	System-on-Chip

Chapter 1

Introduction

1.1 Context

Very-large-scale integration (VLSI) corresponds to the process of combining a huge amount of transistors into a single chip, an Integrated Circuit (IC). Due to the improvement in the manufacturing of semiconductors, nanometer scale transistors started to exist, leading to the reduction of the area of an IC and an increase in the speed of each component.

The VLSI design flow consists of multiple steps, with the physical design flow being at its lowest level. It follows after the circuit design, taking as an input the source code at the RTL level written in a Hardware Description Language (HDL). Based on the circuit representations of the components, the physical design flow aims to create a geometric representation, the integrated circuit layout. Even though the integration of such an enormous amount of transistors in a single IC brought many advantages, the challenges that imposed to the microelectronic industry are incommensurable.

Firstly, it became unfeasible to use manual operation to generate the layout of an IC. For that reason, the need of EDA (Electronic Design Automation) tools to automate this design process appeared. The huge complexity of the problems that these tool face means that no known algorithms can ensure a globally optimal solution in a time efficient manner. Different EDA tools that use different algorithms will certainly present different solutions for the same problem, and none of them will be optimal. Consequently, the market of EDA tools is highly competitive.

Simultaneously, the increase of the number of transistors also lead to an increase of power consumption. As a result, the importance of power in the design of chips started becoming more and more important, not only due to environmental concerns but also because the popularity of mobile devices increased tremendously, becoming a significant driver in the electronics industry. Thus, to address this problem, multiple low power techniques appeared and their use is imperative to reduce the power dissipation.

Another implication of the rapid growth of VLSI design complexity is that design changes became unavoidable. If they occur in the first stages of the design flow, they are not cumbersome and do not represent a big problem. However, when these changes occur towards the end of the

design cycle, where the design has converged after significant efforts and the photolithography masks might have been fabricated, it is not viable to go through the top-down design flow again as that would be too time and cost consuming.

Therefore, physical design methodologies rely heavily on ECO (Engineering Change Order) flows to solve timing closure issues, to accommodate incremental changes or even to perform modifications after the tape-out without having to re-fabricate the complete mask set, which would lead to great added costs. This method poses a big challenge to the design community. Since the ECOs are done very close to the tape-out or even after it, these are time critical missions and any inefficiency in implementation will directly impact the cost and time-to-market of the product.

1.2 Objectives

The dissertation was developed as an internship at STMicroelectronics and presents multiple goals.

First, the physical implementation of an IP in a team environment and using two P&R tools in parallel: Cadence Innovus and Synopsys ICCompilerII.

Afterwards, the comparison of the performance of these two P&R tools with timing convergence and power dissipation being the main comparison metrics.

Then, the development of the power design of the multi-power domain System-on-Chip (SoC) in which the implemented block was integrated.

Finally, the implementation of a complete post-mask functional ECO flow which should be able to perform post-tapeout modifications of the RTL without having to remanufacture the masks associated with the base-layers (non-metal layers).

1.3 STMicroelectronics

ST microelectronics was created in June 1987 following the grouping of Thomson Semi-conducteurs (France) and SGS Microelettronica (Italy). Commonly called ST, it is Europe's largest semiconductor chip maker based on revenue and 43,200 people are employed worldwide.

Since its creation, the company has greatly extended and enriched their product portfolio. Their current strategic focus is to be the leading provider of products and solutions for Smart Driving and the Internet of Things.

It is composed of 11 manufacturing sites all around the world. Crolles site, where this internship took place, is one of the company's most important production, design and process development centers, employing around 4,000 people and hosting both a 200 mm and a 300 mm fab. In this site in specific, ST manufactures integrated circuits with a capacity of 44,000 wafers per month.

1.4 RF Team

The proposed project was carried out in the MDG Central division which has the objective to deliver IPs that contain RF features such has NFC front-end circuits, RF antenna, BLE (Bluetooth Low Energy), among others. The division not only proposes new applications but also addresses requests done by other divisions, for their own products.

Inside this division, the author of this dissertation was integrated into the digital team that consists of around 10 other engineers. Part of the responsibilities of the team include the addition of digital support for the analog circuits (such as power, clock and reset management, memory access and finite state machines), verification and physical implementation, the latter being the focus of the developed work during the internship.

Concerning the objectives of the thesis, while the physical implementation of an IP and its integration in a System-on-Chip were done in a team environment along with other physical design engineers, the comparison of the performance of the two P&R tools, the insertion of ECO cells in the design and the implementation of a post-mask ECO flow were done individually by the author of this dissertation.

1.5 Structure

This document is divided into 7 chapters. In chapter 1 this dissertation is contextualized and its objectives presented. Then, a description of the company and the team in which the author of this dissertation was integrated during its development is made. Chapter 2 introduces concepts that set the theoretical baseline of this dissertation such as an overview of the physical design flow, design techniques associated with low power design and ,finally, all the different aspects associated with Engineering Change Order (ECO) . Due to the novelty of the latter in specific, a study on the existent academic literature will be made and conclusions relevant for the implementation of an ECO flow will be drawn. With the theoretical notions of this chapter in mind, chapter 3 will present a case study of the physical implementation of an IP in a team environment and the workflow developed with the objective to help the backend team drive the implementation of a gate-level netlist to the final GSD layout will be deconstructed. This IP in specific was implemented in two different tools in parallel and the performance of each will be compared in chapter 4 , with the timing convergence and dissipated power being the main comparison metrics. Afterwards, this IP was integrated in a System-on-Chip (SoC) along with other blocks. One particular aspect of the integration will be focused in chapter 5, the power design of a SoC with multiple power domains. Near the tapeout of this chip, a post-mask functional ECO flow was implemented for the developed IP to anticipate future modifications and the details of the procedure taken to do so are analyzed in chapter 6. Finally, chapter 7 presents the conclusions of the dissertation and certain optimizations that could be made in the future regarding the implemented ECO flow.

Chapter 2

Background

This chapter will present many fundamental concepts required for the developed work.

First, an overview of the physical design flow is made. This overview will set the tone for the rest of the dissertation and a deep understanding of the steps that is encompasses were fundamental for the physical implementation of an IP.

Afterwards, the design techniques associated with low power design are presented. Low power design is becoming increasingly important in VLSI and it will be specially relevant in the integration of the implemented IP in a System-on-Chip (SoC) that contains multiple power domains.

Finally, all the different aspects concerning Engineering Change Order (ECO) will be described. This section will have a big emphasis on the study of existent academic literature due to the novelty of the topic. It will also set the baseline for the culmination of this dissertation, the implementation of a post-mask ECO flow.

2.1 Physical Design Flow Overview

After the completion of the RTL and its synthesis, the next big step in the IC design flow is the physical design. It encompasses several different steps which will now be addressed.

First, the floorplan, in which the core area is defined, the Macros and pads are placed and a power grid is constructed. Then, the placement, in which the standard cells will be placed in appropriate locations depending on the chosen optimization goals. Afterwards, the clock tree synthesis. A buffer tree will be created not only due to the extremely high fanout of the clock source, but also to minimize the skew among the sequential elements. Thereafter, the different elements of the circuit will be connected using metal-layers in the routing step. Finally, in the sign-off step, the correctness of the generated layout is ensured by means of physical verification and a final timing and signal integrity analysis is done.

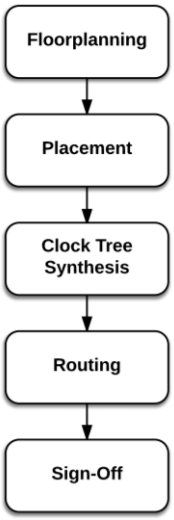


Figure 2.1: Physical design flow overview

2.1.1 Floorplanning

Floorplanning corresponds to the first step in the physical design flow. During the Floorplanning, the width and height of the core and die are initially defined. The core area is where the logic primarily sets while the area around the core is where the I/O pads are placed. There is the need for some clearance distance between the core and the I/O pads which depends on the width of VDD and ground layers.

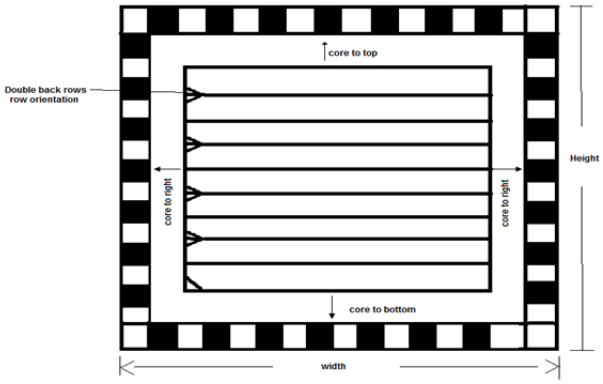


Figure 2.2: Core and die of an IC [4]

Afterwards, the Macros are placed and their arrangement is defined based upon the relationship they have with each other and their distance to the I/O pads with the objective to reduce wire length. They are normally placed manually and fixed so ensure that the following steps will not change their location.

Thereafter, the pad placement and selection takes place. There are three types of pads: power, ground, and signal. One of the objectives when placing the pads is to make sure that they have adequate power and ground connections and that the order in which they are placed reduces any

electromigration and current-switching noise related problems that might appear. While electromigration leads to premature ASIC device failure, current-switching noise may cause noise spikes on non-switching output pads [4].

After the pad placement it is necessary to do the power planning. With technology scaling, the performance of ICs continuously increases which leads to higher operating frequencies. This causes a dramatic increase in the power being delivered as internal logic switching very rapidly leads to high current demand. At the same time, technology scaling also leads to the reduction of the width of the metal layers, which leads to an increased resistance in the routing of the power and, therefore, higher IR drops.

A popular strategy to address the problem of IR drop is the creation of a mesh structure, a power grid. Instead of having only one source of power supply, we have multiple vdd and ground lines, coupled together in pairs. Any cell will use the nearest power source and ground lines in the grid. If there is a high power demand from multiple logic cells at the same time, the mesh network is able to provide the necessary current, which prevents the effects of ground bounce or vdd droop when multiple cells are transitioning simultaneously. Simultaneously, the power and ground lines should use the higher layers of metal in the design as they present the least resistance due to their greater width. Then, by using stacked vias, the lower metals layers will be used (leaving some of them for signal and clock routing) until the metal 1 VDD and GND rails in each row that will directly connect to the pins of the standard cells.

If the power supply is still unable to support the power needs of certain Macros, decoupling capacitors can be placed around them. They are charged by the main power supply that feed them when the current demand is higher. By doing so, Macros do not depend entirely on the power supply.

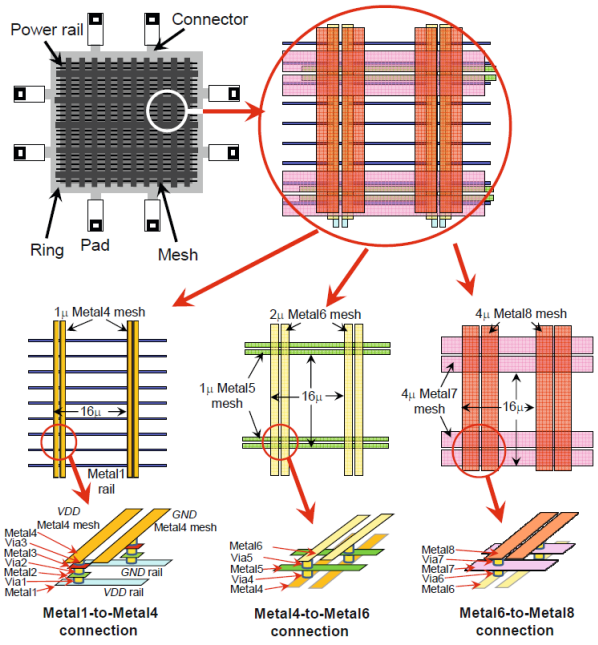


Figure 2.3: Mesh network ilustration [4]

Finally, logical cell placement blockage is implemented. It consists of a blockage in the area between the core and die (area reserved for pin placement) which ensures that there will be no logic cells accidentally placed in that region by an automatic tool. A placement blockage can also be placed around macros to reduce congestion near its pins.

2.1.2 Placement

After the Floorplanning stage is concluded, the Placement stage takes place.

Each standard cell in a library (the different libraries and other input files will be scrutinized in the next chapter) has a certain width and height associated with it as well as a delay. There can be multiple versions of a standard cell with a certain functionality. For example, a cell with a larger area but that presents less resistance and, therefore, less delay. These cells are then placed in the chip in an appropriate location by a placement algorithm based on the connection of these cells with the other cells of the circuit, the Macros and with the pads. As routing information does not exist in the placement stage, the placer uses as optimization goals estimations of routing quality metrics such as total wire length, wire congestion or maximum signal delay.

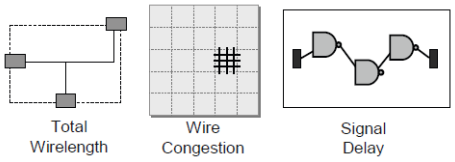


Figure 2.4: Different metrics used by placement algorithms [14]

Nowadays, most of the physical implementation design tools use numerous algorithms to automatically place the standard cells. There algorithms are continuously improved, but the main idea remains the same. These algorithms can be split into two main categories, the constructive placement and the iterative improvement [11]. When it comes to the constructive placement algorithms, a placement is built from square one using a certain method, for example, by placing a seed module in the core area and then select the other modules one by one so that the wire length of the connection between them is minimized. These methods are fast but produce layouts with poor characteristics. For that reason, they are usually used to generate the initial layout used by an iterative improvement algorithm, in which an initial placement is modified multiple times in order to reduce the estimated wire length. The drawback of the iterative improvement algorithms is that they require huge amounts of computational time.

After the execution of the placement algorithm, the placement can be optimized. Signal degradation will occur between long paths connecting the pins to the logic cells. Wire length is estimated and based on the expected degradation, buffers can be placed. At the same time, if a certain section requires higher speeds, then the standard cells of that section can be placed closer to each other to decrease the wire length delays.

At this stage, in order to check if the placement is reasonable, a timing analysis with ideal clocks can be made. Clock signal is assumed to have zero latency and skew and the timing analysis

is done using the estimated wires delays. Setup time violations can thus be detected in this stage. There is also a data slew check, which specifies the time it takes for a transition in a specific node to occur. This time should be inside a certain range. If it too fast, there will be a very high current demand which may lead to a power overshoot. However, if the transition is too slow, there will be a greater amount of time where both the NMOS and PMOS transistors are turned on, which causes short circuit currents that increase the power consumption of the circuit. This analysis leads to the swapping of certain standard cells with another version of the library in order to improve the area or reduce the timing delays.

2.1.3 Clock Tree Synthesis

The next step is the clock tree synthesis. It corresponds to the insertion of buffers or inverters (inverter being more area efficient as a a buffer is nothing more than a chain of two inverters) along the clock paths, not only because the clock net has an extremely high fanout (the clock source needs to drive multiple flip-flops) but also to ensure that the clock delay is balanced in all the clock inputs, that is, the clock signal reaches all the flip-flops of the circuit at approximately the same time (low skew) to prevent timing violations.The buffers or inverters are specially important in long wires in order to ensure the integrity of the signal, such as a small slew. However, even though minimizing the skew is one of the goals of this step, reaching a skew of exactly zero would not be desirable as that would lead to every sequential element of the design switching at exactly the same time. The result would be high current demands in the transition of the clock which would result in undesirable effects such as voltage droop (the voltage supply drops below VDD) or ground bounce (the ground voltage rises above 0V).

As clock nets are critical paths in the chip, clock net shielding might be required as it protects the clock path from cross-talk interference that might lead to glitches.

After the clock tree synthesis and the clock net shielding processes, a timing analysis with real clocks can take place. In this case, the delay caused by the wires and the buffers of the clock net are taken into account. At this stage, the tool tries to solve mostly setup violations whose condition is the following:

$$\theta + \Delta_{launch} < T + \Delta_{capture} - t_{setup} - t_{uncertainty} \tag{2.1}$$

The arrival time is the sum of the combinatorial delay θ and the delay of the clock path of the launch flop Δ_{launch}. In order for a setup violation to be prevented, the arrival time of each timing arc must be lower than the required time. The required time is the sum of the clock period T and the delay of the clock path of the capture flop $\Delta_{capture}$, subtracting the setup time of the capture flop t_{setup} and the clock uncertainty $t_{uncertainty}$. The difference between the required time and the arrival time is the *slack*.

Setup violations can be solved by decreasing the arrival time by swapping the standard cells of the problematic combinational paths by faster versions or by purposely increasing the skew between the launch flop and capture flop by placing extra buffers or inverters in the clock path of

the capture flop (which would consequently increase the required time, therefore increasing the time slack).

It should be noted that while hold violations also appear at this stage, they are usually solved later, in the sign-off stage. There are some reasons for doing so. On one hand, the steps taken to solve setup violations might simultaneously create hold violations. On the other hand, the extraction of the RC parasites during the signoff increases the delay of the combinatorial paths which might solve many of the hold violations without requiring any effort.

A popular Clock Tree Synthesis algorithm whose underlying concepts are still used in current EDA tools is the H-Tree algorithm [15]. Fig. 2.5 illustrates the implementation of an H-Tree. First, a connection from the clock driver to the center of the H structure is made, dividing the core into two sections (left and right) . Then, two shorter lines that make a right angle with the previous line are made until the center of the new sections. This process is repeated recursively, creating multiple H-shape structures, until all the clock sinks are reached. Besides process variations that inevitably occur, the clock path delay will be practically the same for every sink.

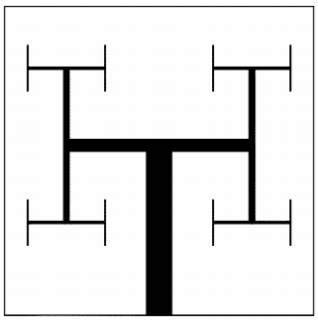


Figure 2.5: Implemented H-Tree [15]

After the Clock Tree Synthesis step finishes and all the buffers required to build the buffer-tree have been placed, there will still be empty spots left out in the core area. However, for the manufacturing of the chip it is required that the entire empty space is filled. As a result, filler cells which do not have any functionality are placed at this stage to ensure the continuity of the power lines and the N-well across the tracks. Alternatively, if certain modules require more power due to the IR-drop effect, decoupling capacitors placed like filler cells [7] can fill the space around them instead.

2.1.4 Routing

After the placement of the standard cells and the clock tree synthesis, there is the need to route the design by making all the necessary connections between the standard cells. One of the goals of the routing step is to reduce the total wire length of the design and congestion to a minimum. By doing so, the existing timing scenario can be maintained after the routing, which means that the timing convergence obtained after the CTS should be kept.

One of the most well-know and accepted routing algorithms is the Maze Lee algorithm [2] illustrated in Fig. 2.6. Initially, a routing grid is generated.The objective is to find the best connection between two end points (a source and a target, that correspond to two of the squares of the grid), with the shortest path and with as little twists and turns as possible. The first step is to label each square adjacent to the source with a 1. Then, the adjacent squares to the grid squares previously labeled with a 1 will be labeled with a 2. This process is repeated until all the grid squares from the source to the target have been labeled. Once this expansion phase is completed, by following a path from the target to the source in which the square numbers decrease sequentially to 1 there is the assurance that this path has the shortest length. However, this algorithm presents some limitations. It is slow and it requires large amounts of memory, especially for a dense layout as it has to store the routing grid information (such as the label given to each square) for every single connection.

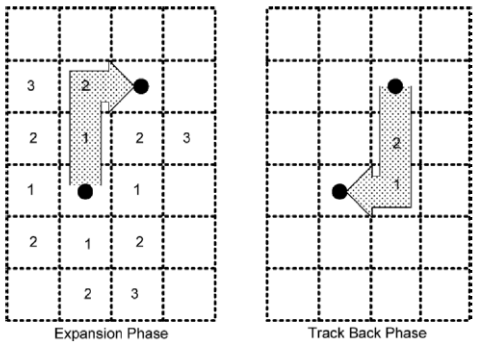


Figure 2.6: Lee's algorithm representation [4]

2.1.5 Sign-Off

After the routing of the design is concluded, multiple checks are made before the fabrication process begins.

First, an equivalence check test will certify that the gate-level netlist is functionally equivalent to the post-layout netlist produced by the P&R tool.

Afterwards, after the generation of the layout (the GDS file) by merging the DEF and LEF files exported from the P&R tool, together with device level information provided by the standard cell kit (file formats that will be explained in a later chapter of this dissertation), a dedicated physical verification tool can be used to perform multiple checks in order to gauge if the layout meets certain criteria before the tape-out.

Firstly, Design Rule Check (DRC) tests are applied. They determine if all the rules set by the foundry for manufacturing are being respected. These include minimum spacing and width between metals and vias, minimum and maximum metal densities for each layer as well as the antenna rules that should be respected.

Secondly, LVS (Layout vs Schematic) tests are done to check if the generated layout corresponds to the original schematic.

Thereafter, antenna checks verify if the maximum allowed metal area to gate area is respected. This check is needed due to plasma induced gate oxide damage [15], a failure that might occur during manufacturing (more specifically, in the plasma etching process) due to charge accumulation in the metals. The discharging of this charge through gate oxide may potentially damage it.

Finally, ERC (Electrical rule check) checks are performed to access if any gate was left floating or if the wells are connected to power/ground in order to prevent latch-up problems.

After the physical verification is complete, the resistance and parasitic capacitance of every net will be extracted and a final static timing analysis is conducted. Considering most of the setup violations were solved during the timing analysis performed after the CTS step, hold violations are more common at this stage. Hold violations, which occur for the same clock edge, have the following condition:

$$\theta + \Delta_{launch} > \Delta_{capture} + t_{hold} + t_{uncertainty} \tag{2.2}$$

The arrival time is the sum of the combinatorial delay θ and the delay of the clock path of the launch flop Δ_{launch}. In order for a hold violation to be prevented, the arrival time of each timing arc must be higher than the required time. The required time is the sum of delay of the clock path of the capture flop $\Delta_{capture}$, the hold time of the capture flop t_{hold} and the clock uncertainty $t_{uncertainty}$. The difference between the arrival time and the required time is the *slack* of an hold condition.

Hold violations at this stage are usually solved by purposely increasing the combinatorial delay of the problematic path by adding buffers or by increasing the clock delay of the launch flop by adding buffers to its clock path.

Maximum capacitance checks are also performed to check if every standard cell is able to drive the capacitance of the net in its output.

2.2 Low Power Methodologies

In early CMOS technologies, power was not a major concern in design of digital integrated VLSI chips. The main priority of the designers was mostly implementing the desired functionality in the prevailing process technology. However, when CMOS technologies started to decrease in size, it became possible to fit more transistors in a die, leading to an higher power consumption (and, simultaneously, an higher heat dissipation which damages electronic components). Besides the number of transistors, lower sized technologies also imply faster switching speeds as lowering transistors' length leads to higher currents, further increasing the power consumption of the chip. Therefore, the importance of power in the design of chips started increasing in importance, not only due to environmental concerns but also because the popularity of mobile devices increased tremendously, becoming a significant driver in the electronics industry. In chips placed in portable devices power consumption is of up most importance. Batteries are currently the bottleneck of such devices as their capacity has not been improving as quickly as the performance of all the

different components of the device. As performance increases, battery lifetime decreases, which means efficient low power techniques need to be used to prevent the user from having to constantly charge the device.

The total power consumption in digital integrated circuits can be split in two major types: dynamic power and static power.

2.2.1 Dynamic Power

Dynamic power consumption is related with transitions at gate terminals and can be divided in two: switching power and short-circuit power.

Switching power is the result of the constant charge and discharge of the output capacitance of the logic gates, which includes the wire capacitance and the input capacitance of the logic gates connected to them (gate capacitance loads). Switching power can be calculated from expression 2.1, where f_{switch} is the toggling frequency (which depends on the fraction of transistors actively switching and the clock frequency), V the supply voltage and C the total capacitance that is being charged/discharged.

$$P_{switching} = f_{switch}.C.V^2 \tag{2.3}$$

Considering the quadratic nature of the supply voltage, reducing it is clearly an efficient way of decreasing the power consumption and that is exactly what is happening with the scaling down of the process technology. However, reducing it also decreases the current of the transistors and, simultaneously, the switching speed, which can be prejudicial if high performance is desirable. One way to overcome this problem is to use different supply voltages for different areas of the chip, in what is called a Multi-voltage design. By doing so, some modules can have lower supply voltages without sacrificing the voltage needs of high-speed modules. Multi-doltage designs have some implications that will be later described, like the use of level-shifters in the interface between them.

From the swiching power expression, it is also clear that reducing the clock frequency is one of the methods that can be used to reduce the switching power. The trade-off is obvious though, as performance is lost in the process.

One of the big contributors of the total switching power is the buffer tree build in the CTS step of the physical implementation. The high number of buffers/inverters lead to an high capacitance and there is an high toggling frequency as every component switches in every clock cycle. Therefore, building a clock tree that accomplishes all the skew and slew requisites with the least amount of buffer levels is one of the reasons that can highly influence the total power consumption, not only switching power but also because a lower number of components decreases static power.

A widely used technique to reduce the switching power is clock gating. In few words, this technique disconnects the clock signal from the clock pin of sequential logic, reducing the number of active components and avoiding having to charge/discharge their input capacitance.

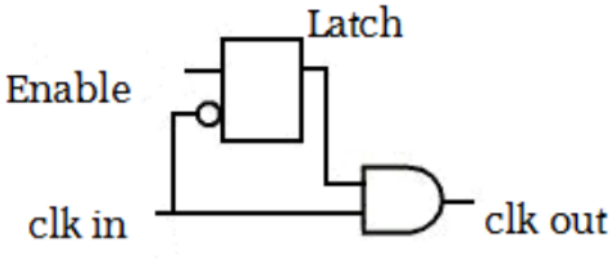


Figure 2.7: Clock gating cell

Short-circuit power, which represents less than 20% of the dynamic power consumption [16] will be described in detail later in another relevant section of this chapter.

2.2.2 Static Power

The total leakage power in CMOS circuits is determined by the contribution of leakage currents in each transistor, which has two main sources: subthreshold leakage current and gate tunneling leakage current.

$$P_{leakage} = V_{dd} . I_{leakage} \tag{2.4}$$

$$I_{leakage} = I_{subthreshold} + I_{gate} \tag{2.5}$$

Gate leakage is the current that flows through the gate oxide, due to the quantum-mechanical tunneling of electrons [17]. The full explanation for such effect is out of the scope of this thesis.

As was previously explained, supply voltage has been scaled down to keep dynamic power consumption under control, with the trade-off of lowering the drain current and, therefore, the switching speed. To maintain a high drive current capability, the threshold voltage (Vth) has to be scaled down too, as there is a dependence of the drain current and $(Vgs - Vth)^2$. However, the Vth scaling results in increasing subthreshold leakage current as can be seen by the leakage current's expression in Fig. 2.8.

ubthreshold current occurs between drain and source of PMOS and NMOS transistors when they are operating in the weak inversion region, the operating region in which the gate voltage is lower than threshold voltage.

$$I_S = I_0 W e^{\frac{V_{gs} - (V_{t0} - \eta V_{ds} - \gamma V_{bs})}{nV_T}} \left[1 - e^{\frac{-V_{ds}}{V_T}} \right]$$

Figure 2.8: Subthreshold current expression

Whereas in the past leakage power could be neglected in comparison with dynamic power, that stops being the case when transistor scaling resulted in a substantial increase of leakage currents. Fig. 2.9 compares dynamic power and leakage power consumption with the scaling down of the technology process. It should be noted that even though the reduction of the supply voltage leads to less dynamic power, the faster switching speeds due to increases in performance (possible due to the reduction in transistors length, which increases their drain current) means dynamic power is rising along with leakage power.

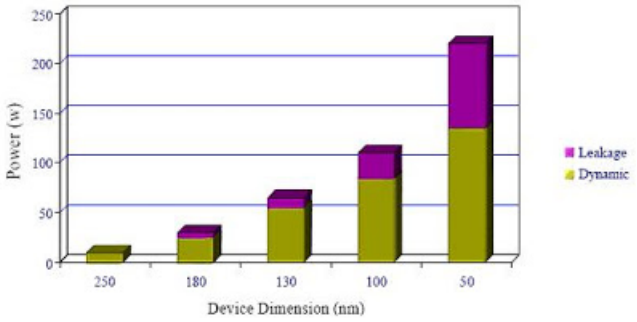


Figure 2.9: Comparison of leakage vs dynamic power with shrinking of device dimensions [31]

Consequently, taking into account static power dissipation as soon as possible in the design flow has become critical in low-power circuits. There are multiple techniques to reduce static power consumption. One of the main ones is power gating and it will be explained in detail in the next section.

2.2.3 Power Gating

The objective of power gating is to shut off the modules of the circuit which are not in use, considerably reducing static power dissipation. It has a bigger impact on the design architecture than clock gating, as power gating models have to be safely entered and exited, leading to timing delays and possible set-up violations. Simultaneously, using power gating increases the design complexity, the area of the chip and constantly turning a module on or off may lead to increased dynamic power consumption. As a result, there are clear trade-offs that should be taken into account when using this technique, making it mostly used in chips where low-power is a key feature. The different components needed to implement this technique will now be described.

2.2.3.1 Power Switches

Usually the power of shutdown modules is turned off using power switches, a CMOS transistor placed between one the power supply and the power port of the standard cells. It can be either an NMOS (footer switch, connected to ground) or a PMOS (header switch, connected to power). For larger modules, a single transistor might not have enough driving power, which means multiple transistors need to be used in parallel.

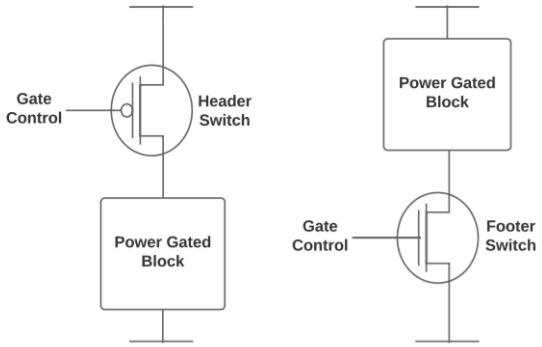


Figure 2.10: Power switch. On the left, using a PMOS as a header switch. On the right, using an NMOS as a footer switch.

The signal used to turn a module on or off is generated by the Power Management Unit (PMU), responsible for controlling all the power related components in the design (which also includes the enable signal for the isolation cell that will be explained further in this dissertation). Since this thesis is backend focused, the PMU will not be explained in great detail.

2.2.3.2 Retention Registers

When a module is turned off, their registers lose their value. If it is important to save those values so that the current state is retained on wake-up, a retention strategy needs to be used. Usually, special registers with less leakier and lower voltage flip-flops are used. Obviously though, these flip-flops need to be always powered-on even if they are placed inside a shutdown module (a concept called power island). Therefore, special care is needed when placing and routing them as it requires the routing of the supply rails of an always-on power domain into the shutdown domain. The big area overhead of these flip-flops, the possible complexity of their routability and the time delays associated with restoring the values back to the main registers at wake up need to be considered when deciding to use a retention strategy.

2.2.3.3 Isolation

When a power domain (the concept of power domain will be clarified further down in this chapter, but the name ends up being self-explanatory) is turning on or off, its output values are undefined between 0 and 1. This is a clear signal integrity issue, which might not only lead to incorrect logic values in the input of the always-on power domains but also increased power dissipation due to short-circuit currents from supply to ground. Considering the wake-up or turning-off of a module might be slow, a long transient in the input signal of the always-on power domain will lead to a long time period in which both the NMOS and PMOS transistors will be conducting, causing a direct path between supply and ground.

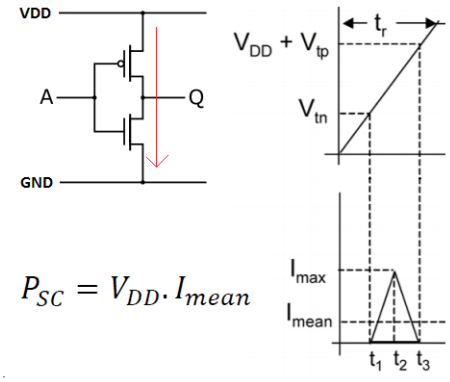


Figure 2.11: Short circuit power

Consequently, the power gated module needs to be safely entered and exited by means of isolation cells located at the interface between modules which are in shut-down and always on power domains. They have an enable signal as an input that, when active, allows the communication between the two power domains. These cells are placed in the always-on domain so that they are always powered and usually the enable signal is generated by this domain. If it is not, special care is required, as is explained in a future chapter of this dissertation. When the enable signal is off, there is no communication between the two modules and the isolation cells clamp the output node to a known voltage to ensure that the inputs of the always-on domain are not floating.

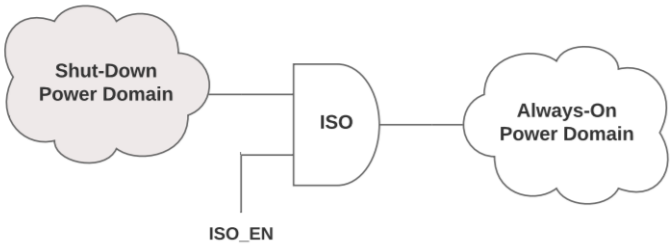


Figure 2.12: Isolation Cell. Enable signal generated by a Power Management Unit.

2.2.3.4 Level Shifters

In Multi-voltage designs, two modules cannot be directly connected as that would lead to an incorrect functionality. For example, if a module operating at $1.8V$ received a logic value of 1 from another module operating at $0.9V$ then it would not be able to correctly assess if consists of a 0 or a 1, leading to a possible error. This means there is the need to place level-shifters between two modules that operate at different voltage levels. As the name implies, the purpose of this cell is to shift voltage across different modeules, both from low to high as well as high to low.

It is not necessary that every design which has a module that might shut-off requires level-shifters as they only need to be used if two different power domains operate at 2 different voltages. But considering the power gating technique is used in low-power chips, it is also very likely that it is a Multi-voltage design.

It should be noted that besides the increased area, there is also a timing impact in the paths requiring level-shifters due to their delay.

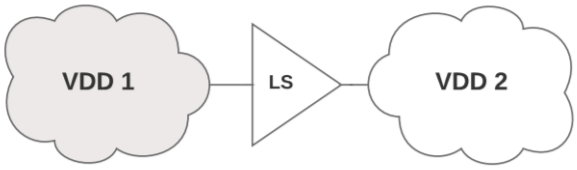


Figure 2.13: Level-Shifter

2.2.3.5 Enable Level Shifter

In power designs in which two power domains have different voltage levels and, simultaneously, one of the modules is a shut-down domain, special cells called enable level shifters can be used. These cells incorporate the functionality of a level shifter and an isolator in the same cell, leading to a smaller area as compared to a combination of two independent cells.

2.2.4 Power Intent and UPF format

Several methods for power consumption reduction, such as the power gating technique, require power information that is not supported by Hardware Description Languages (HDL). Since RTL and design constraints are not sufficient for describing power behavior, there is the need for constructs for capturing different power domains, supply rails, operating voltages, usage of special cells (isolation cells, level shifters, power switches, retention registers), among others. That is the function of the power intent of the design, which is separate from the RTL but must be coupled with it in the synthesis as it affects design behavior. This means the IC can be designed with power as a key consideration early in the flow.

Due to the need of power intent languages, the EDA industry responded with multiple vendors developing proprietary low power specification capabilities for different tools in the design and implementation flow. This solution solved the problem for each individual tool, but it is still far from ideal as the same power intent information had to be specified multiple times for all the different tools. In order to tackle this problem two global standards are currently defined for power intent specification: Unified Power Format (UPF) and Common Power Format (CPF).

The work done on the implemented block later described uses UPF, a TCL command based language that reflect the power intent of a design at a relatively high level.

2.2.5 Power Domains

The power domains are the fundamental objects of the power intent, a collection of design elements that share the same power specifications, they are powered the same way. This means that if two modules belong to the same power domain, then they turn off at the same time (in case of a shutdown power domain) and use the same operating voltages. A power domain can also be called

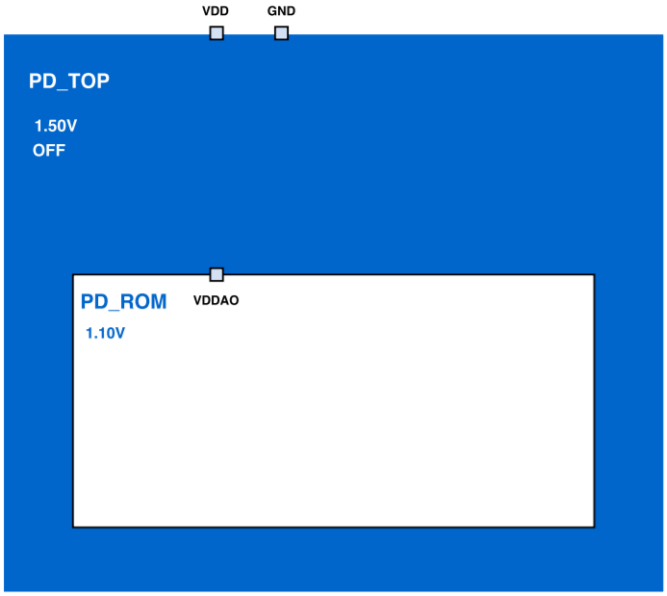


Figure 2.14: Two different power domains, a shutdown domain and an always-on domain

an always-on domain, if it never turns off. It is also important to notice that a power domain is not necessarily physical contiguous. Instances belonging to the same power domain might be located in very different locations, if they share the same power specifications.

A primary supply set (an aggregation of supply nets) and supply ports (connection points between adjacent levels of hierarchy) are thus defined for each power domain. The inclusion of modules inside a power domain is what leads to the automation opportunities that the UPF language provides, as all the elements within a power domain will be implicitly connected to its primary power set.

Besides the creation of power domain, supply sets and ports, UPF commands are also responsible for creating and connecting new objects that were not preset in the RTL HDL description of the design. This includes all the special cells associated with power consumption reduction techniques, that are created in the context of a power domain, covering isolation strategies, retention strategies, level-shifters and power switches.

2.3 ECO Problem and Solutions

Physical design methodologies rely heavily on ECO (Engineering Change Order) flows to solve timing closure issues, to accommodate incremental changes or even to perform modification after the tape-out without having to re-fabricate the complete mask set. This method poses a big challenge to the design community. Since the ECOs are done very close to the tape-out or even after it, these are time critical missions and any inefficiency in implementation will directly impact the cost and time-to-market of the product. An ECO flow can be distinguished by either pre-mask or post-mask.

A pre-mask ECO modification occurs before the fabrication of the masks which means standard cells can be freely inserted, moved or deleted. Pre-mask ECOs are usually performed close to the tapeout, to avoid having to repeat the physical implementation from the beginning due to a late design change.

After the tapeout and the masks for the different layers have been fabricated, post-mask ECOs are performed instead. The changes are realized by modifying only the photomasks of the metal layer as an all-layer ECO (which includes the base layers, sometimes also called non-metal layers) would lead to a complete re-spin of the design, defeating the purpose of doing a post-mask ECO in the first place. Nowadays, most VLSI corporations implement metal-only ECOs flows after the tape-out due to their low-cost as carrying out design changes with minimal layer modifications saves a lot of money from a fabrication point of view as each layer mask has a significant cost of its own. The base layer masks in particular represent around 70% of the total mask set cost [18], so performing metal-only ECO clearly represents a big reduction in the cost of the re-fabrication. Therefore, while the objective of pre-mask ECOs is to reduce design-time before the tapeout, the objective of post-mask ECOs is to reduce design-cost after the tapeout. This dissertation will manly focus on post-mask metal-only ECOs.

Besides the layers that are used, an ECO can also be classified in two types. Functional ECOs can be used to fix bugs, revise specifications or add a new functionality, while timing ECOs target to improve input slew, output loading and delays. Generally, a timing improvement or a functional revision are done in an isolated fashion, but it is possible to implement an ECO flow that does both simultaneously, as proposed by [9]. When it comes to the type of ECO, the study presented in this chapter will mainly focus on functional ECOs.

In order to perform ECO modifications, an ECO flow that has the objective to automatize the ECO modifications can be implemented for an integrated circuit. Manually performing an ECO modification is not only more time expensive but also much more error prone, which is not acceptable when the deadline of a project is approaching. An ECO flow for post-mask modifications can even be implemented before the tape-out of the design, already anticipating the possibility of having to perform these time critical changes in the future. This dissertation will culminate with the implementation of a post-mask functional ECO flow.

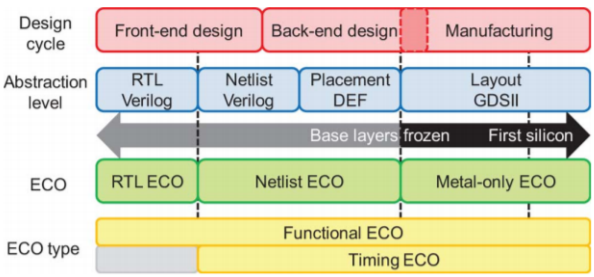


Figure 2.15: Distinction between different types of ECO both in layers and types and where they occur in the design flow. Metal-only ECOs are performed after the base-layers are frozen [13]

2.3.1 Insertion of Spare or Reconfigurable Cells

In order for post-mask ECO modifications to be performed, a critical preliminary step is the insertion of ECO cells during the physical implementation of a design. ECO cells are redundant cells which are placed in the core area even though they are not currently being used in the design. The objective of these cells is to anticipate possible modifications that might happen in the future.

ECO cells can be divided into two main groups that will be described in this section: spare cells and reconfigurable cells. Spare cells end up being equivalent to any other functional standard cell as they have a fixed functionality. On the other end, reconfigurable cells are cells that can be reconfigured into a desired functionality using only metal layers. Regardless of their type, the location of the ECO cells is of utmost importance. If it is expected that a certain module will require a high number of ECO changes (for example, a module that corresponds to a new architecture) then the vicinity of that module should be sprinkled with more ECO cells. On the other hand, modules with more stable architectures usually do not require so many ECO changes, which implies that less ECO cells should be sprinkled near them. However, in most practical cases it is difficult to predict the locations where ECO modifications are more likely. Therefore, ECO cells are usually uniformly spread over the whole design homogeneously, to ensure resources are available for ECO modifications at most locations.

2.3.1.1 Spare Cells

Spare cells, just like most standard cells, have a fixed functionality. Consequently, an appropriate selection of spare cells of different types is fundamental to ensure that the desired functionality is available when an ECO modification must be performed. Besides the spare cells that are deliberately placed, some of the standard cells might be converted into spare cells due to design changes in which they become unused.

One of the problems of using spare cells is that they contribute to the static power of the circuit as each cell dissipates leakage power resulting from the use of tie-high and tie-low cells that connect the inputs of the spare cells to power and ground nets as they cannot be left floating. Tie-high and tie-low are necessary to avoid a direct connection between the power nets and the gate-oxide present in the gates of the spare cells as gate-oxide is highly sensible to voltage surges and ESD (Electrostatic discharge) events.

At the same time, unused spare cells represent an inefficient use of area. This means that the number of unused spare cells (which unavoidably occur in the ECO design flow) should be reduced to a minimum while still ensuring that there are enough spare cells for the necessary ECO modifications.

2.3.1.2 Reconfigurable Cells

In order to tackle the problems that occur with the use of spare cells, [10] proposes the use of reconfigurable cells. When needed, these cells can be reconfigured into a desired logic function to perform the modification.

Moreover, if they end up not being used, there is the possibility to reconfigure them into decoupling capacitors, which leads to a more efficient use of the available area. Decoupling capacitors, as previously described, become more and more important with the shrinking of MOS technology as supply voltage is reduced and the resistance of the metal layers, due to a reduction in their width, is increased. Thus, IR-drop becomes more significant, the problem that decoupling capacitors address.

At the same time, if when using spare cells there are not enough logic gates of a desired type to represent the logic change, the ECO cannot be carried out using only metal layers. It has to be, then, carried out using all the layers as more cells will need to be added which results in the re-spin of the design. By using reconfigurable cells, however, there is a free selection on the logic gate type by reconfiguring the base cell with metal, which increases the likelihood of performing a successful ECO.

Another advantage of the use of reconfigurable cells is that tie-high and tie-low cells are no longer needed, leading to a further reduction of inefficient area usage and a reduction of leakage power.

In Fig. 2.16, the base cell presented [10] that will be later reconfigured is illustrated.

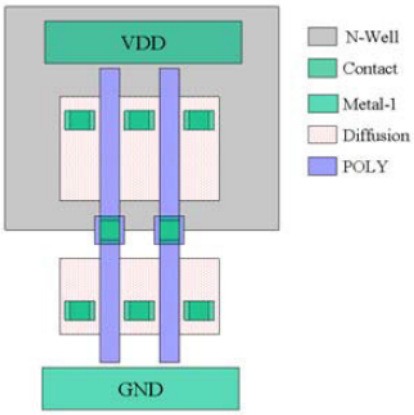


Figure 2.16: Base cell [19]

By making metal-1 connections, it is then possible to configure this base cell either into a decoupling capacitor or into different kinds of logic gates.

As a first approach, Fig. 2.17 shows how to implement a decoupling capacitor using the gate capacitance of a single NMOS transistor. The drain and source simply need to be connected to ground and the gate to vdd.

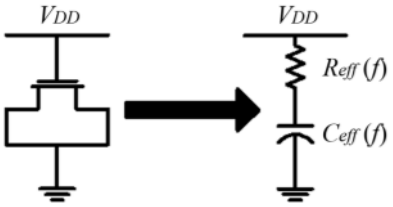


Figure 2.17: Decap implemented using a single NMOS [20]

However, when using the presented base cell, it is possible to use two NMOS and two PMOS transistors, as illustrated by Fig. 2.18. With this implementation, P1 and N1 get turned on and act as decoupling capacitors. Simultaneously, P2 and N2 appear in series with P1 and N1, resulting in reduced gate leakage due to the effect of series stacking [21]

It should be noted that unused reconfigurable cells do not necessarily need to be reconfigured into decoupling capacitors. If the decoupling capacitors already present in the design are sufficient to avoid significant IR-drop, the unused reconfigurable cells can just be left as base cells, which makes them equivalent to a filler cell that simply ensures the continuity of the power lines and the N-well across the track. The advantage of not reconfiguring the base cell into decoupling capacitors (in a situation without IR-drop issues) is that decoupling capacitors are more leaky, unnecessarily increasing the static power dissipation of the circuit.

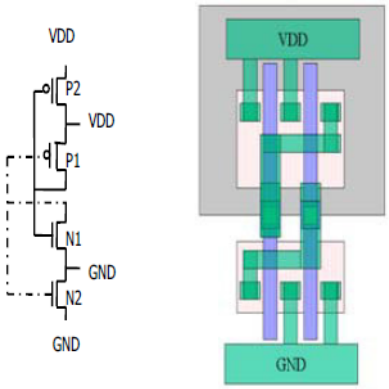


Figure 2.18: Reconfigured decoupling capacitor [19]

When ECO modifications are necessary, the base cell can also be easily reconfigured into an inverter, a NAND gate or a NOR gate by making the metal-1 connections illustrated in Fig. 2.19.

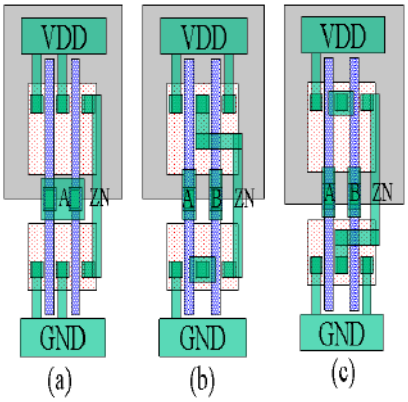


Figure 2.19: Reconfigured Logic Cells. (a) Inverter; (b) NAND; (c) NOR. [19]

2.3.1.3 Metal-Configurable Gate-Array Cells

The drawback of using the previously described reconfigurable cells is when cells with an higher complexity or different driving capabilities are required, as there is the need to connect multiple reconfigurable base cells. The internal connection among the reconfigurable cells leads to extra delay that induces timing degradation. One possible solution would be to insert both reconfigurable cells (for the more simple logic gates) and spare cells (for the most complex ones). Another solution for this problem that uses a different type of reconfigurable cells, metal-monfigurable Gate-Array (GA) cells is proposed by [22], which sometimes can also be called GA reconfigurable cells. Fig. 2.20 illustrates the concept of these cells. They consist of an array of tiles in which each tile represents a basic reconfigurable cell, similar to the one presented previously. If a more complex logic gate is desired, multiple adjacent tiles can be used. Fig. 2.21 compares the GA reconfigurable cells approach with the previously described single reconfigurable cell approach in the implementation of an AND gate. The single reconfigurable cell approach will need additional wiring for internal connections, which results in timing degradation.

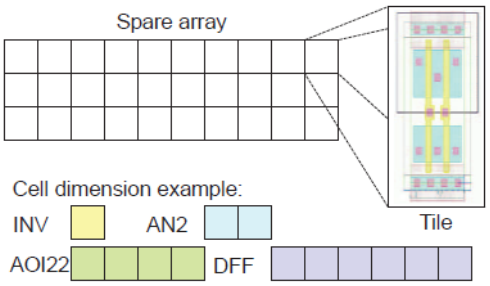


Figure 2.20: Reconfigured gate-array reconfigurable cells [22]

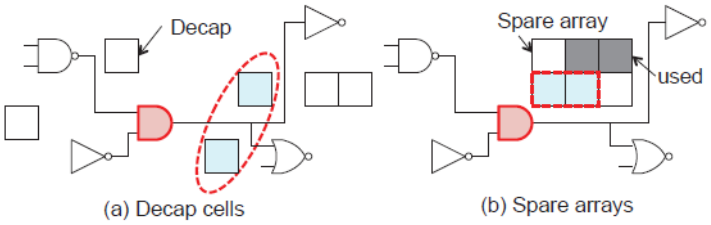


Figure 2.21: Comparison reconfigurable cells and gate-array spare cells in the implementation of an AND gate [22]

Because of the flexibility of metal-configurable GA cells, only 0.5% to 1% of the chip area is occupied by spare arrays (this ratio is very low compared with 2% to 5% for standard spare cells)[23].

However, it is also true that for the same driving strength, a functional cell made from reconfiguring the GA reconfigurable cells is slower and less area-efficient than a standard spare cell [23].

2.3.2 Performing metal-only ECO functional modifications

The different steps that should be carried out in order to implement a metal-only ECO are illustrated in Fig. 2.22 and they will be explained during this section. Firstly, assuming an already implemented netlist which contains either spare cells or reconfigurable cells, the logic difference between this netlist and the RTL or gate-level netlist with the desired modifications is extracted. Secondly, an ECO synthesizer will apply the desired changes in the implemented netlist without modifying the base layers by using the available ECO cells in the design. Finally, after ensuring that the new synthesized netlist is functionally equivalent to the ECO RTL, the changes applied by the synthesizer can be routed using metal layers.

It is also important to note that this explanation (as well as the implemented metal-only ECO flow discussed further in this dissertation) will focus on combinatorial and not sequential modifications. Sequential changes are more involved and usually require rerunning the clock tree synthesis, which means they are usually not performed in metal-only layers. Some other considerations to have in mind when performing sequential ECO modifications will be analyzed further in this dissertation as it will be more relevant then.

2.3.3 Logic Difference Extraction

After the insertion of ECO cells, it is necessary to extract the logic difference between the original placed design and the new design that needs to be implemented, the ECO RTL. This difference corresponds to the ECO list, which consists on a list of functional changes that should be made. Usually, a short ECO list leads to a high probability of a feasible ECO [13].

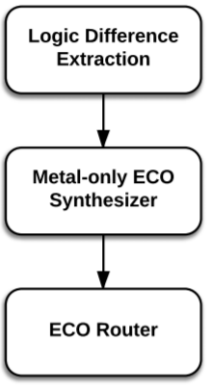


Figure 2.22: Different steps in an ECO flow [12]

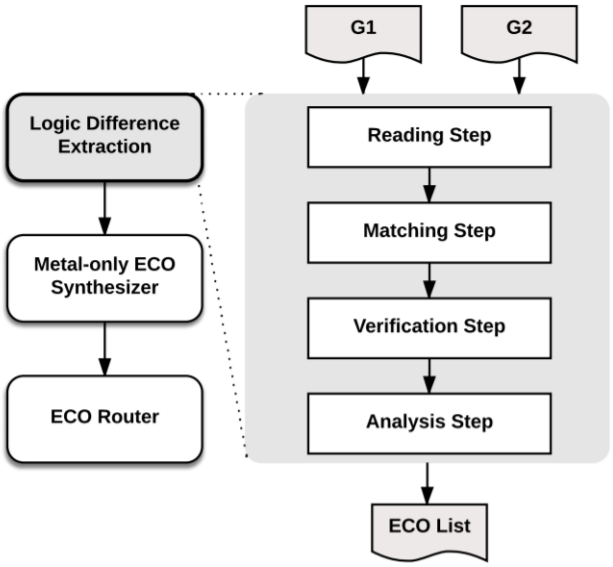


Figure 2.23: Different steps in the logic difference extraction

The logic difference between the netlists is determined by an equivalency checking tool, which uses formal mathematical techniques to prove that two designs are functional identical. It should be noted that equivalence checking tools have more uses outside of ECO modifications, specially in formal verification. Formal verification is used to prove that two representations of the same design exhibit exactly the same behavior. For example, after the synthesis the gate-level netlist can be compared with the RTL netlist to ensure that they are functional identical. If they are not, then mistakes occurred in the synthesis of the design. Another example would be after the routing, by comparing the pre-routed and the post-routed netlists.

2.3.3.1 Reading Step

During the reading step, both the original placed design (which will be called golden design from now on in this chapter) and the modified design are automatically segmented into logic cones and compare points. This segmentation of the design allows the analysis of smaller and more manageable sections.

Compare points are the design nodes at which the functionality is compared. They can be primary outputs, inputs of registers or the inputs of black boxes. Black boxes usually represent Macros whose function is unknown, such as a memory or an analog IP. Logic cones are groups of combinational logic that drive the compare points. They contain an input-border that consists of primary inputs or outputs of registers and black boxes and an output-border that consists of the compare point that the logic cone drives.

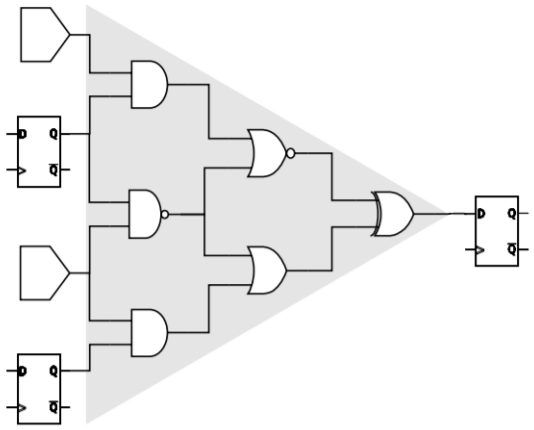


Figure 2.24: Logic cone and its input and output borders

2.3.3.2 Matching Step

After breaking the golden design and modified design into logic cones and compare points, the tool attempts to map the compare points of the golden design to the correspondent compare points of the modified design.

There are various techniques to map the compare points, and multiple of them can be used in succession until all the compare points are mapped. The matching techniques can be divided into two main categories, name-based and function-based.

In most tools, name-based matching methods are used first. Three different types of name-based matching can be used. Exact-name matching corresponds to mapping the compare points that have exactly the same name. However, it is common for synthesis tools to change the names of the instances, specially when an hierarchical design is flattened and instances belonging to a certain module are placed in the top-level. This leads to the use of a name-filtering method that excludes certain special characters or lower/upper case differences from the comparison. For example, if in the modified design there is an instance with the name TOP/sub1/ABC[10] it could

be matched with an instance from the golden design with the name TOP/Sub1_Abc10. If both of these methods fails, a final name-based method can be used that compares the names of the nets attached to a compare point. If two compare points with different names are driven by nets with the same name, then they are considered the same.

The vast majority of the compare points of the golden design and reference design are matched by name-based techniques. If not, non name-based (also called function-based) techniques can be used. The main function-based matching technique corresponds to topological equivalence. If the logic cones driving two unmatched compare points are topologically equivalent (if they have the same structure), then those two compare points are considered matched.

The final method is signature analysis, an iterative analysis of the compare points' functional signatures by applying vectors derived from random pattern generation to the inputs of the logic cones. However, signature analysis matching has a considerable run-time, which makes it a not very common technique.

Every compare point needs to be matched between the two designs before the verification step. If the methods described above used by did not manage to do it, then it is the job of the engineer to determine manually the matching between the last remaining unmatched compare points.

2.3.3.3 Verification Step

After the compare points have been matched, the next step is to verify if the functionality of the logic cone that drives each matching compare point is the same. Many algorithms are available to compare and prove the equivalence of logic cones. The most popular way to verify the equivalence of two logic cones is through the use of a SAT engine, whose mechanisms will be understood in a later section.

Once the verification step is complete, an equivalence checking tool generates a list of any compare point that is not equivalent, which means that for the same input-boundary of a logic cone, the output-boundary (compare point) is not the same. Based on the equivalence of the logic cones, the mapped compared points can be either equivalent or non-equivalent.

2.3.3.4 Analysis Step

Finally, in the last step of the logic difference extraction, the logic cones driving the non-equivalent points are analyzed. Equivalence checking tools provide various isolation capabilities to help isolate the difference between two different logic cones, which is needed as logic cones can be too big to find the difference manually.

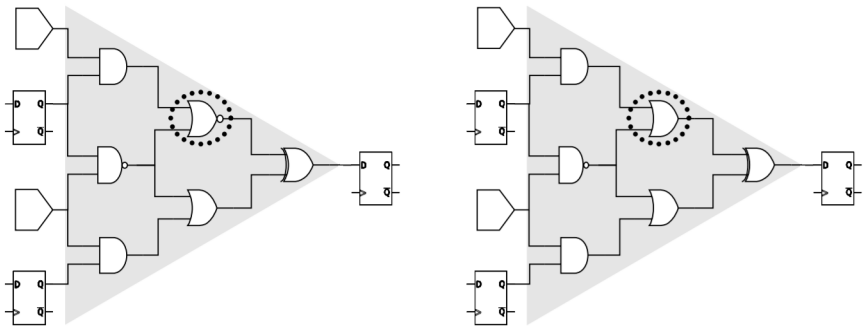


Figure 2.25: Two non-equivalent logic cones as their compare point shows different logic values. The difference that led to the non-equivalence must be detected

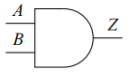
The extracted differences between the logic cones corresponds to the ECO list, the changes that need to be applied to the implemented netlist in order to accumulate the new functionality.

Besides being used to extract the ECO list, an equivalence checker tool can also be used to ensure that the new netlist generated by the ECO synthesizer is functionally equivalent to the RTL with the ECO modifications, in which case non-equivalent compare point are not expected.

2.3.3.5 Academic research on logic difference extraction

Most of the academic research on logic difference extraction tackles the problem of minimizing the size of the ECO list (sometimes also called ECO patch) as a smaller ECO list increases the likelihood of a feasible ECO due to a smaller number of modifications. The state-of-the-art work on this topic uses techniques based on SAT (Boolean satisfiability problem) engines [8][25].

While all the mathematical intricacies of the Boolean satisfiability problem are out of the scope of this dissertation, it can be summarized as the problem of finding an assignment of variables $x_1, x_2, ..., x_n$ such that a given boolean function $F(x_1, x_2, ..., x_n)$ evaluates to true [26]. If there is such an assignment, then F is said to be satisfiable, if there is not, F is unsatisfiable. For example, Fig.2.26 shows the boolean function F_{AND}. The inputs of this boolean function (A,B and Z) that correspond to the correct functionality of an *AND* gate satisfy F_{AND}.



A	B	Z	F_{AND}
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$$F_{\text{AND}} = (A + \overline{Z}) \cdot (B + \overline{Z}) \cdot (\overline{A} + \overline{B} + Z)$$

Figure 2.26: Boolean function F_{AND} [26]

The problem of finding non-equivalences between two designs can thus be reduced to the problem of initially translating both of the circuits in boolean functions and then using a SAT

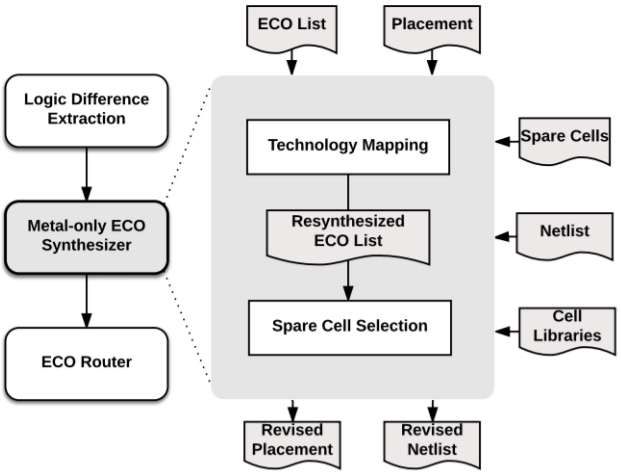


Figure 2.27: Overview of ECOS (based on [13])

engine to seek situations in which a boolean function of one of the circuits is satisfiable but not on the other.

Both [8] and [25] use this SAT technique. While [8] initially assumes the ECO list is the entire original netlist and iteratively removes the equivalent sections by subtracting them, [25] does the opposite. Initially the ECO list is empty and the non-equivalencies are repeatedly added until the ECO list is complete. [24] did a review on these two approaches and concluded that for a high number of modifications, the procedure taken by [25] produces a smaller ECO list than the one taken by [8].

2.3.4 Metal-Only ECO Synthesis

An ECO synthesizer that completes the changes described in the ECO list accurately and efficiently is required. It uses the physical information of the spare cells (both the location and cell type) to rewire the circuit with the minimum cost.

In order to deeply understand the problem in question and a possible solution, a proposed metal-only ECO synthesizer named ECOS will be analyzed [13]. Of course there are multiple methods to apply the desired ECO modifications, and,as such, each tool might use a different ECO synthesizer. However, the formulated problem and the necessary features remain essentially the same, so analyzing this metal-only ECO synthesizer in specific will provide better intuition. Fig. 2.27 shows an overview of the ECO synthesizer that will be analyzed.

2.3.4.1 Problem Formulation

The metal-only ECO synthesis problem can be formulated as follows:

“ **The Minimum-Cost ECO Synthesis Problem:** Given the netlist and placement of a design, the cell library, a set of spare cells and an ECO list (a list of functional changes), complete the

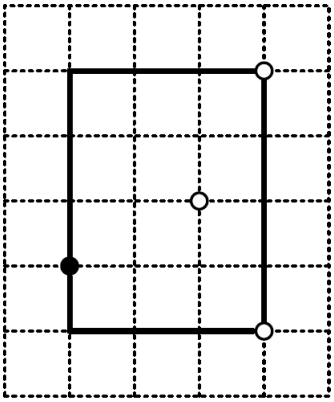


Figure 2.28: Bounding box of a net that connects four ports [4]

ECO list using the available spare cells, create the revised netlist with the minimum cost (without sacrificing timing and routability) and generate the revised set of spare cells [13]."

There are multiple ways to define the cost of a metal-only ECO modification, the loss of routability and timing. In this case, the cost will be modeled by the summation of the half-perimeter wirelength (HPWL) of all the nets of the revised design. HPWL is an wire estimation method widely used in different steps of the Physical Design Flow (such as placement and CTS) to estimate the total wire length before the routing step, allowing to evaluate the timing and routability of the design in earlier stages of the flow.

It uses a wiring bounding box for each net. This bounding box represents the smallest rectangle that encloses all ports of the net. The half-perimeter wire length estimation of a net is one-half of its wiring bounding box [4]. In fact, this estimation is a lower-bound on the real wire length as will be seen in the following examples.

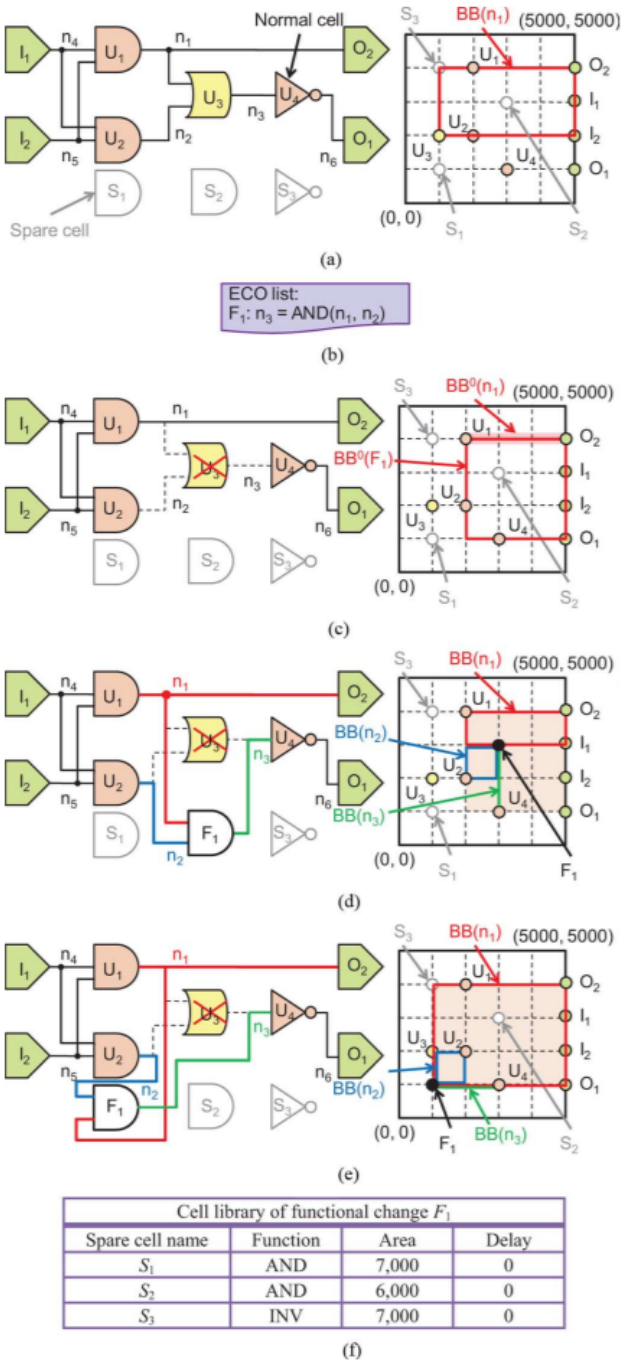
Taking Fig. 2.28 as an example, if we wish to estimate the wire length of a net that connects the 4 nodes, we start by drawing a box that encloses all of them, the bounding box of the net. If we consider that the side of each square of the grid has 1 λ of length, then the perimeter of the bounding box is 14 λ, which means the HPWL estimation of this net is 7 λ. By manually connecting the 4 nodes, it is easy to see that the real cost of this net is in fact 8 λ. Thus, the HPWL metric was a fair estimation of the lower bound of the total wire length.

The design example illustrated in Fig. 2.29(a) will be used to present the metal-only synthesizer ECOS. This design contains two inputs, two outputs, four used logic cells, six nets and three unused spare cells (two AND and one INV). On the right of Fig. 2.29(a), the placement of each of these cells is illustrated. To simplify, each cell has an area of 0, which means both the input and output pins of each is located in a single point. The bounding box of net 1 is also showed.

The summation of the half-perimeter wire length of this design is as follows:

$$\sum_{i=1..6} HPWL(n_i) = 6000 + 1000 + 3000 + 5000 + 2000 = 22000 \tag{2.6}$$

Fig. 2.29(b) shows the ECO list that contains the functional change F_1 to be made. In this



Cell library of functional change F_1			
Spare cell name	Function	Area	Delay
S_1	AND	7,000	0
S_2	AND	6,000	0
S_3	INV	7,000	0

Figure 2.29: (a) A design example for an ECO modification. On the left, a gate-level representation of the initial design. On the right, the placement of each cell, including the spare cells. The bounding-box of net n_1 is highlighted (b) The ECO list with the functional change to be made (c) Cell U_3 will not be used anymore which means it can be disconnected. (d) The revised netlist and placement after cell U_3 is replaced with the spare cell S_2. The bounding box of each cell associated with the functional change is highlighted. Since the cell U_3 is not used anymore, it is freed up and can serve as a spare cell for future modifications (e) Same functional change if spare cell S_1 had been chosen instead (f) The spare cell library for F_1 [13]

case, F_1 represents $n_3 = AND(n_1, n_2)$ which means the OR logic cell U_3 should be replaced by an AND gate.

2.3.4.2 Possible Hand-Made ECO and its limitations

This example is relatively simple and an hand-made ECO could easily be made by disconnecting U_3's inputs and outputs as showed in Fig. 2.29(c). Afterwards, one of the AND spare cells (S_1 or S_2) has to be chosen: considering the closer proximity of S_2 to U_1,U_2,U_4 and to the output O_1 this spare cell is the obvious choice between the two. The cost of the revised design of Fig. 2.29(d) is given by

$$\sum_{i=1..6} HPWL(n_i) = 4000 + 2000 + 2000 + 5000 + 5000 + 2000 = 20000 \tag{2.7}$$

To ensure that the choice was correct, the cost of the revised design by choosing S_1 instead would have been:

$$\sum_{i=1..6} HPWL(n_i) = 7000 + 2000 + 2000 + 5000 + 5000 + 2000 = 23000 \tag{2.8}$$

However, there are multiple reasons why an hand-made ECO is not feasible in most practical situations and why an automatic ECO synthesizer is necessary. Firstly, because an ECO list might be large, making the hand-made ECO too time consuming and error prone. Secondly, because the number of spare cells in a design is limited. If there are multiple functional changes (FCs), several FCs may prefer the same spare cell, leading to competition problems that need to be dealt with. Finally, because the available spare cells might not directly match the desired functionality. For example, if the ECO list in Fig. 2.29(b) used an NAND instead of an AND, then there would not be a direct match, both S_2 and S_3 would have to be used. In some situations it might considerably increase the complexity of choosing the preferred spare cells based on proximity.

2.3.4.3 Terminology

Before going in depth in the explanation of ECOS, some terminology needs to be clarified (As a side note, a functional-change will be often referred to as FC from now on).

Definition 2.3.1. $\mathbf{HPWL^0(F_j)}$ - Pre-ECO HPLW : *HPLW* associated with the bounding box covering the FC's F_j related nets after unused connections are removed and before the FC is applied.

In the design example, the unused connections correspond to the connections between nets n_1, n_2 and n_3 that were connected to U_3. Therefore, $HPWL^0(F_1) = 6000$.

Definition 2.3.2. $\mathbf{HPWL^1(F_j, S_k)}$ - Lower bound of assigning a spare cell S_k to a functional change F_j : HPLW associated with bounding box covering S_k (spare cell that was chosen to perform the modification) and the pins of Fj's related nets.

In the design example, $HPWL^1(F_1, S_2) = 6000$.

With this terminology out of the way, the metal-only synthesizer ECOS can be presented, still using Fig. 2.29 as a design example. As shown in the ECOS overview in Fig. 2.27 (shown in the beginning of this section), it is composed of two steps: Technology Mapping and Spare Cell Selection.

2.3.4.4 Technology Mapping Step

As spare cells are limited both in type and quantity, an automatic method to choose spare cells of the proper type and simultaneously taking into account the distance of all the cells involved in the modification is needed. Not only should it be able to deal with simple situations ,such as the previously described hand-made ECO (where S_2 was chosen over S_1 because of a lower cost of 20000 compared to 23000), but also when the desired ECO functionality is mismatched with the available spare cells.

ECOS was build on top of a logic synthesis environment called ABC, guiding it with spare cells types and proximity based on the physical information of the design. To do so, each functional-change (FC) F_j will have its own customized spare cell list based on these parameters which will indicate what are the preferred spare cells to minimize the cost of the ECO modifications. The physical proximity specification will be modeled by the cell area of each S_k which corresponds to the cost of assigning that S_k to the FC F_j (which is equivalent to $HPWL^1(F_j, S_k)$), ensuring that guided ABC will give appropriate preferences and lead to a resynthesized ECO list with good proximity. Taking the design example again, the area assigned for each spare cell for F_1 is the following:

$$area(F_1, S_1) = HPWL^1(F_1, S_1) = 7000 \tag{2.9}$$

$$area(F_1, S_2) = HPWL^1(F_1, S_2) = 6000 \tag{2.10}$$

$$area(F_1, S_3) = HPWL^1(F_1, S_3) = 7000 \tag{2.11}$$

Based on these results, Fig. 2.29(f) presents the spare cell list for this FC. In this case, the delay of each cell S_k was set to zero, to simplify the problem. If the desired FC is not directly mapped by one of the spare cells available in the library, guided ABC can use two different methods to increase the number of cell types.

Firstly, it supports constant insertion. For example, if an FC requires an INV but the only available spare cell is a NAND, a logic value of '1' can be set in one of the inputs of the NAND. Using the other input of the NAND, there is now a spare cell which is functionally equivalent to an inverter. This means a cell of type INV would be available in the spare cell list of this FC even if an INV spare cell did not truly exist.

Secondly, if the functionality of the ECO list mismatches the available spare cells, a FC might be converted into multiple spare cells. For example, if a NAND is required but only an AND and an INV are present, then guided ABC will recognize that both the AND and the INV can be used together. If two spare cells are required for the desired functionality, then a single FC will be converted into two functional-changes, each with its own customized spare cell list that indicate the spare cells areas. When this situation occurs, there are internal connections between the FCs and the spare cells lists of these FCs will depend on each other. The implications of this detail will be explained in the second step of ECOS, the spare cell selection step.

2.3.4.5 Spare Cell Selection

After the technology mapping step creates a spare cell list for each functional change, there is another problem to handle: the competition among different FCs for the same spare cell. Therefore, the decision of the selection of a spare cell for an FC F_j is delayed until this competition problem is solved.This competition problem can be modeled by the stable marriage problem:

" **The Stable Marriage Problem:** Given a set of men and women, each man has ranked the women in order of preference and each woman has done likewise. The objective is to marry them off in pairs such that there are not a man and a woman who are not married to each other but both would prefer each other to their actual mates. If there are no such pairs, all the marriages are stable."

If the preference lists for each FC are complete and have no ties, Gale-Shapley algorithm [33], listed in Fig. 2.30, showed that a stable marriage exists for any ranking.

The key idea of ECOS is to model functional changes as men, and spare cells as women. Every woman (spare cell) is then ranked in a man's preference list.The preference that a FC gives to each spare cell reflects the added cost that would result from choosing it. The higher the preference, the lower the added cost. The added cost represents the difference between the cell area (provided by the spare cells list of the technology mapping step, which ends up being the real cost of the choice) and $HPWL^0(F_j)$, whose meaning was explained above.

$$pref(F_j,S_k) = \Delta cost(F_j,S_k) = area(F_j,S_k) - HPWL^0(F_j) \tag{2.12}$$

As an example, F_1 has the following preferences values. Considering S_3 does not have the correct type to perform the desired functionality, its added cost will be infinity to ensure that S_3 will never be the choice.

$$pref(F_1,S_1) = \Delta cost(F_1,S_1) = area(F_1,S_1) - HPWL^0(F_1) = 1000 \tag{2.13}$$

$$pref(F_1,S_2) = \Delta cost(F_1,S_2) = area(F_1,S_2) - HPWL^0(F_1) = 0 \tag{2.14}$$

$$pref(F_1, S_3) = \Delta cost(F_1, S_3) = \infty \tag{2.15}$$

With this framework in mind, ECOS performs spare cell selection based on Gale-Shapley's stable matching algorithm, listed in Fig. 2.30. Gale-Shapley's algorithm is male-optimal, so every functional change tends to find its best selection of spare cells.

```
StableMatching(M, W)
// M: the set F of FCs; W: the set S of spare cells
1.  Initialize all m ∈ M and w ∈ W as free
2.  while ∃ free man m who hasn't proposed to all women do
3.     w = the highest ranked women in m's preference list
4.     if w is free then
5.        (m, w) become engaged
6.     else // some pair (m', w) is currently engaged
7.        if w prefers m to m' then
8.           (m, w) become engaged
9.           m' becomes free
10.    Update preference
```

Figure 2.30: Modified stable matching algorithm [13]

It should be noted that while the Gale-Shappley's algorithm presented is relatively straight forward, the line related with the update of the preferences (line 10) and its intricacies are out of the scope of this dissertation. It is related with the conversion of an FC into multiple FCs, leading to internal connections between them and a dependence of their spare cells lists (that indicate the area associated with each spare cell). Assigning a spare cell to one of those FCs will automatically change the internal connections to the other FC and, as a result, change its preferences by adding an induced cost component.

2.3.4.6 Academic research on ECO synthesis

After deeply understanding the problem ECO synthesizers try to solve, the concepts associated with the cost of an ECO modification and the required features of a solution by analyzing the synthesizer ECOS, other aspects related with the synthesis of functional ECO present in the literature can be more easily grasped.

When it comes to academic research on synthesis using standard spare cells,[27] does a study regarding the constant insertion technique of the technology mapping step in which the insertion of constants on the spare cells' inputs increases the number of possible solutions. It proves that the inclusion of this technique on the synthesis reduces the area required to find a feasible mapping solution by 80% as it becomes less likely that a cell with the desired functionality is far from the place where the ECO modification is needed.

Also regarding the technology mapping step of the synthesis, [28] proposes a technique that takes into account the quantities and location of the spare cells based on simulated annealing that

does incremental spare-cell assignment in order to explore the search space (the set of possible solutions) probabilistically. However, this simulated annealing technique is slow and , consequently, not suitable in larger designs.

[29] tackles this drawback by proposing a resource and constrain aware technology mapping engine which uses boolean matching algorithms to speed up the process of finding a near-optimal solution. First, it represents the circuit described by the ECO list into an AIG (And-inverter graph) representation. Afterwards, this AIG graph is divided in multiple parts with specific cuts and the truth table of each part is computed. Finally, each of these parts is mapped with real spare cells by using boolean mapping algorithms that use NPN-equivalence classes to speed up the mapping process. Equivalence of two functions defined under the NPN-classification states equivalency of two Boolean functions under input Negation, input Permutation or output Negation. This means that it is possible to achieve identical values for both truth table outputs by permutation or negation of the function inputs and/or negation of the function output, then the functions are equivalent.

When it comes to ECO synthesizers that use reconfigurable cells instead of spare cells, although they are becoming more and more popular in the industry, not much academic research targets this ECO technique. Nonetheless, regarding the optimization problems of an ECO synthesizer, while the problems associated with the quantity of available cells remains, the problems associated with the desired functionality are fundamentally different. Regarding Gate-Array reconfigurable cells in specific, even though they can be reconfigured into the desired functionality, a new issue related with fragmentation occurs. Fragmentation is related with the need to use tiles from separate GA reconfigurable cells to perform the desired functionality. It is clearly troublesome due to the fact that the internal connections between the GA cells lead to timing degradation and congestion which reduces the routability of the design. Therefore, to model this issue in the optimization problem given to an ECO synthesizer, [23] proposes a new metric, "aliveness". It models the capability of a GA cell into being reconfigurale into the desired type which will depend on its number of free tiles.

To objectively define this new metric, [23] assumes there are m different types of functional cells in which a GA cell can be reconfigured in and that each of those types occupies s_i tiles of a gate-array cell (with i conditioned by $1 \leq i \leq m$). Then, for a gate-array with k free tiles and assuming it implements z_i cells of size s_i, the following inequality can be defined:

$$s_1 z_1 + s_2 z_2 + ... + s_m z_m \leq k \tag{2.16}$$

Considering the equation $s_1 z_1 + s_2 z_2 + ... + s_m z_m = k$ represents a plane in the m-dimensional space, one can assume m equals to three for a graphical representation. The result is the plane illustrated by Fig. 2,31., which makes a tetrahedron with the three axis. The points inside this tetrahedron are the possible solutions of the inequality. For a more intuitive example, if we consider three different types of functional cells ($m = 3$) and that the first type occupies two cells ($s_1 = 2$), then, for a GA with four free tiles ($k = 4$), if they are reconfigured in two cells of the first type ($z_1 = 2$), both z_2 and z_3 must necessarily be zero as there are no free tiles left. This situation

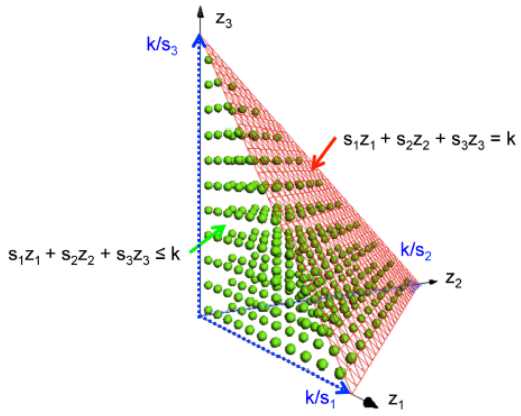


Figure 2.31: The aliveness of a gate-array cell can be defined as the volume inside the tetrahedron if there are three possible types of functional cells it can be reconfigured in [23]

in particular corresponds to the interception of the plane with the z_1 axis, a point that respects the inequality. Consequently, since it is desired that a GA reconfigurable cell can be reconfigured into as many functions as possible, "aliveness" can be defined as the number of integer points inside the tetrahedron or, seen from a different perspective, its volume.

2.3.5 ECO Routing

Finally, after achieving an equivalence check between the revised netlist and the output of the ECO synthesizer (the process being exactly the same showed for the logic difference extraction, but now no differences are expected) the rewiring of the inputs and outputs of the selected spare cells is conducted using an ECO router. The already existing routing patterns from the original design make the ECO routing complicated as it will be difficult to pass the design rules with such an enormous amount of obstacles. Besides, while a default router is allowed to rip up and reroute the existing nets in order to provide more flexibility, reduce congestion and, therefore, reduce the likelihood of DRC violations, an ECO router is more greedy and limits the number of reroutes of existing nets to a minimum in order to reduce the cost of the ECO modifications.

Another characteristic of ECO routing, is that it is possible to perform it using only some of metal layers. By doing so, only the masks of the metal layers being used need to be remanufactured, greatly reducing the cost of the ECO.

Fig. 2.32 illustrates a situation in which ECO routing using a limited number of layers occurs. Initially, the cell on the far left was connected to the cell on the far right. However, an ECO modification required disconnecting the cell on the far left and connecting a spare cell instead, using only the lower metal layers. The interesting situation comes in how the cell on the left gets disconnected: while the lower metals of the net can simply be removed as a new mask will be made for them anyway, the upper metal layers of the net need to remain (even if they are not being used), to ensure that the masks of these metal prevail intact.

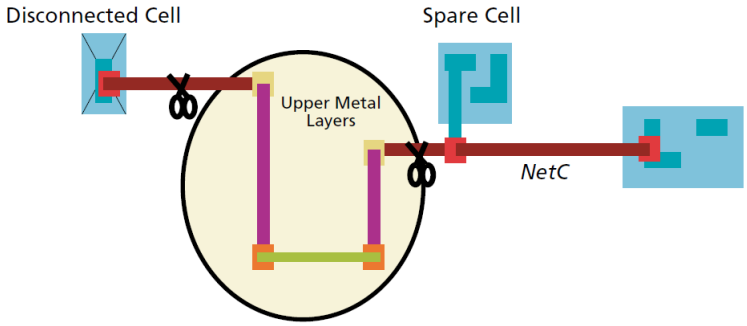


Figure 2.32: ECO Routing using a limited number of metal layers [32]

2.3.5.1 Academic research on ECO routing

Regarding the literature on ECO routing, [30] introduces two interesting techniques not addressed in the rest of the literature.

The first technique concerns the minimization of the number of rewiring layers which, as stated before, has an impact on the mask's re-spin cost. The objective is to introduce a routing-resource-aware metric in the spare cell selection so that the number of rewiring layers can be taken into account earlier in the ECO flow. In a sense, even though this technique has routing implications, it ends up affecting the ECO synthesis by adding a new optimization variable. Fig. 2.32 illustrates this concept. In this situation, both the spare cells have the same functionality and one of them needs to substitute the original cell. By choosing the spare cell on the bottom, both the metal layers will need to be rewired, meaning both the masks need to be re-spinned. By choosing the one on the top, however, the routing resources associated with the metal layer 2 can be reused, avoiding the need to change this mask. Consequently, the ECO synthesizer should take this concept into consideration when making the choice between the two.

The second technique proposed by [30] concerns the use of redundant wires, not only for reducing the number of metal-layers required by the ECO, but also to reducing the number of rewires of existing nets to alter the timing convergence of the taped-out design as little as possible. There are two types of redundant wires being considered. On one hand, the wires that became unused. If a specific net is not needed anymore it can be used for another ECO modification instead of simply deleting it. On the other hand, the dummy wires used to achieve layout uniformity. By using these wires for routing, obstacles can be crossed over with higher metal layers without having to actually re-spin their metal masks. Fig.33 illustrates this idea, in which a dummy wire in metal-2 is used to cross over an obstacle.

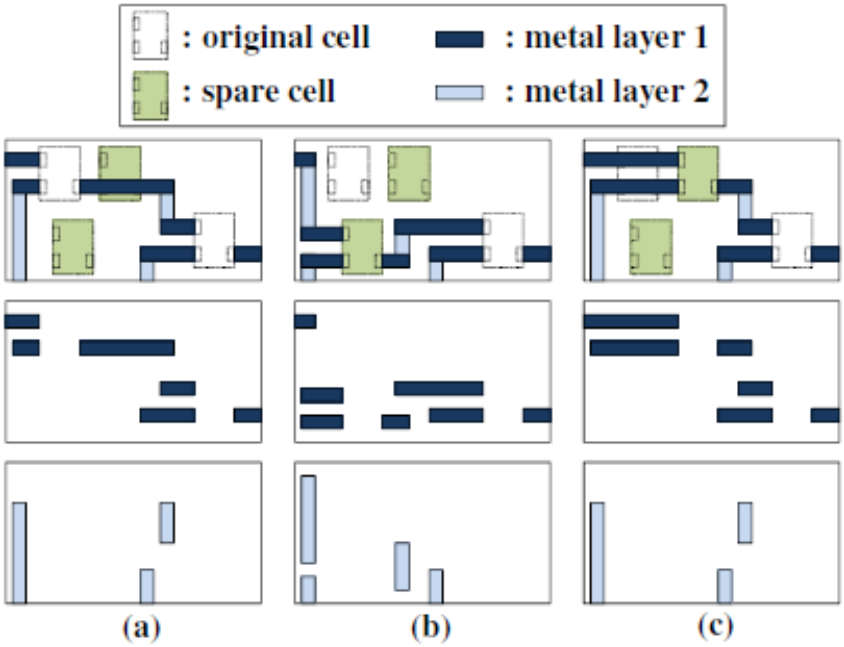


Figure 2.33: (a) A timing-violated path. (b) Spare cell rewiring by changing the horizontal and the vertical metal layers. (c) Spare cell rewiring by changing only the horizontal metal layer, and thus the mask for the vertical metal layer can be reused to save the cost [30]

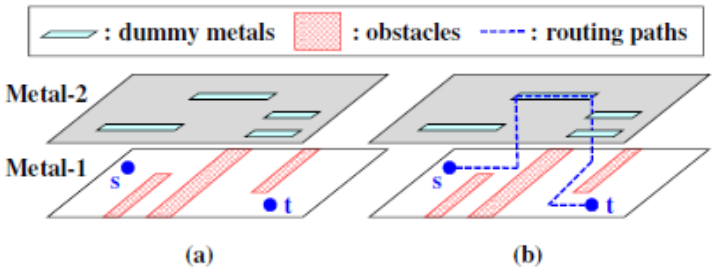


Figure 2.34: (a) Routing configuration before rewiring. (b) Rewiring a net by reusing a dummy metal [30]

Chapter 3

Physical Implementation of an IP

The implemented circuit called TOP_ANA consists of an IP block , later integrated into a SoC, for NFC contactless transactions using smart cards.

This block contains an analog front-end (AFE) Macro, a RAM Macro and digital logic responsible for management, control and memory access. The digital logic uses a process technology of 40*nm* and it contains more than 70,000 instances. For routing, it uses a total of six metal layers. While the metal layers 1 to 5 are intended for most of the interconnections and are made of cooper, the uppermost layer (AP) is intended for the the power grid only and fabricated in aluminum due to the lower resistance.

3.1 Physical Implementation using ODIF KIT

In order to implement this block, Open Digital Implementation Flow (ODIF), a kit developed by STMicroelectronics designed to assist engineers, was used. It enables the execution of the flow by providing an abstraction layer on top of Synopsys ICCompiler II or Cadence Innovus tools that consists of a design working area for running specific tasks.

This kit fits into the design flow between the Hand-off and the Sign-off, driving the implementation of a gate-level netlist to a the layout, a GDS design file which contains the representation of the geometric shapes, text labels, and other information about the layout.

The complex functionality of this kit can be briefly described by its two main directories: the DESIGN directory and the FLOW directory.

3.1.1 DESIGN Directory

In the DESIGN directory, information used as an input to the physical flow in order to configure the P&R tool is defined: the synthesized gate-level netlist; the design's power intent; the standard cells libraries used by the design; the technology files defining the rules of the selected process; the definition of the different scenarios; the design constrains; among many other files related with the setup of the P&R tool being used.

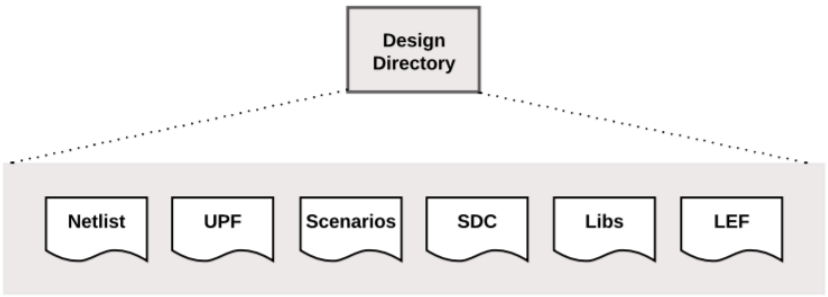


Figure 3.1: Files present in the design directory

A big percentage of the effort in the physical implementation of an IP goes into the setup of all of these input files. It is of up most importance to ensure that the constrains are in agreement with the gate-level netlist and that all the requirements set by the digital designer are present. This means, for example, having correct pin lists, correct clock and generated clock definitions, identification of multicycle and asynchronous paths, among many other consideration that will be presented. Constant communication with the RTL team is needed, specially when the RTL is in an early Alpha stage where every modification needs to be updated in the constrains. The setup files are therefore a "work in progress" throughout the different stages of the development of an IP.

Besides the synthesised gate-level netlist, the power intent UPF file (as seen in chapter 2) and the definition of the different scenarios (that corresponds to a specific mode of operation and PVT corner) some of the most important setup files are related with the design constrains and the libraries (both the standard cell libraries and the technology libraries). These different input files will now be presented in detail.

3.1.1.1 Design Constrains

The constrains represent design restrictions applied in various steps of the design flow of a chip (such as logic synthesis, P&R and static timing analysis) that determine what the tools can or cannot do, how they should behave. This means that if there are multiple ways to solve a specific problem, the design constrains will limit the number of solutions. If the tools do not find any solution for a problem it might mean that the constrains are too restrictive.

In order to allow multiple tools to understand the same constrains files (avoiding the need to re-write them for each tool) a standard format called Synopsys Design Constraints (SDC) is used. It is a TCL command based language in which most of the commands follow the same structure. They require a design object as a command argument (the object that will be restricted) as well as parameters that affect the restriction. For example, in a clock definition:

```
create_clock [get_ports $clkA$] -period 10 -waveform {0 5}
```

In this straight forward example, a clock object is being created with a clock source *clkA* (the design object), a period of $10ns$ and a duty cycle of 50%. These parameters will serve as

restrictions when performing timing analysis on all the nets associated with the clock port *clkA*. For instance, when checking if any setup violations occurred.

The constrains can be divided in multiple types such as timing constrains, boundary conditions and timing exceptions. In the case of the DESIGN directory, each of these constrain types corresponds to a different file, which will now be briefly described.

Timing Constrains

These constraints are related to timing specifications of the design. Some of the main commands to define timing specifications are:

- **create_clock** - Clock definition that creates a clock object and defines its waveform in the current design, as explained in the example above.
- **create_generated_clock** - Defines a new clock signal by dividing or multiplying another clock in the design, and binds this new clock it with a source pin (usually a Q pin of a flip-flop, as that is normally the output of a clock divider)
- **set_input_delay** - Defines the arrival time relative to a clock edge on the input ports of the design. This input path delay models the delay from an external flip-flop to at input port of the module and will be used for setup and hold analysis on paths that transverse the interface with the exterior. However, considering this flip-flop comes from an unknown exterior, it corresponds to an imaginary flip-flop connected to a virtual clock (a clock that is defined like the others but does not have a real clock source). Because of that, a virtual clock with an user specified delay must be created that connects to that imaginary flop.
- **set_output_delay** - Similar to the input delay, but specifies the data required time on output ports instead.

Boundary Conditions

These constrains define the system interface between the block being implemented and the exterior, they setup the environment of the design under analysis. Some of the main commands to define boundary conditions are:

- **set_driving_cell** - Adds a default driving cell, which means all the design elements connected to the input pins of the design must respect the constrains defined in the Liberty File (explained later in this section) such as the maximum fanout allowed by the driving cell.
- **set_load** - Adds a default load cell on every output, which means all the design elements connected to the output must have enough driving power to respect the load cell's constrains, such as its maximum allowed transition time.

Timing Exceptions

The exceptions, in contrast with the previously defined constraints, do not represent design restrictions but specific situations in which the constrains do not apply.

- **set_false_path** - There can be timing arcs in the design where changes in the launch flop will not lead to changes in the capture flop as there is no input vector that can excite that timing arc. This means these timing arcs should not be subjugated to timing analysis. Fig. 3.1. shows an example of one of these case, in which no change in the flip-flop D_1 will impact the value captured by D_4.

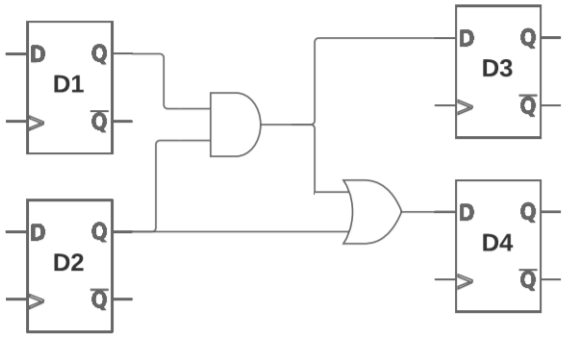


Figure 3.2: Example of a false path from flip-flop D_1 to flip-flop D_4

Another common example is in the reset signal de-assertion to modules whose clock is gated. Since there is no active clock in those modules, there is no chance of metastability happening, which means the timing analysis should be turned off for those timing arcs.

- **set_multicycle_path** - In some systems there can be some expected combinatorial paths whose propagation delays are longer than the period of the clock. In that case, it might not be requirement to capture the source signal transition within one clock cycle. These paths would lead to setup violations in timing analysis, so they need to be defined as multicycle paths.

- **set_case_analysis** - Sets constant values to a list of pins whose values are valid only during timing analysis. The objective is to disable certain timing arcs during TA such as the ones related with scan chains by setting the scan enable signal to zero.

- **set_clock_groups** - Defining clock groups is useful in the exceptions constrains to specify clocks that are asynchronous between each other. This situation is common when a signal crosses from one clock domain to another (using a two flop synchronizer to synchronize this signal with the new clock domain), where launch and capture flop of this timing arc are asynchronous between each other. In the following command, the group that consists of

clkA and clkA_div8 (which are synchronous between each other) is defined as asynchronous to clkB.

```
set_clock_groups -asynchronous -group {clkA clkA_div8} {clkB}
```

3.1.1.2 Libraries

Different libraries are necessary for the physical implementation of an IP. Libraries are collections of different parameters, physical information and abstract views associated with the standard cells and Macros that are being used in the design. Two of the most important library file types which had to be consulted during the implementation of the IP will now be described: the Liberty Files (Lib) and Library Exchange Format (LEF).

Liberty Files (Lib)

The Liberty Files (Lib) specify timing, power and capacitance parameters associated with the standard cells as well as the functionality associated with their output pins. Usually a Lib file is specified for each scenario (that corresponds to specific mode of operation and PVT corner) as they are necessary for Multi-Mode Multi-Coner (MMMC) timing analysis.

In a sense, the Liberty File for each library in the design can also be considered a design constrain as the parameters of the standard cells end up being restrictions that need to respected For example, it specifies the maximum transition time allowed in the input of each stan and its maximum fanout.

The maximum transition time of an input pin of a cell defines the longest time required for its driving pin to change its logic value. In case a tool detects the maximum transition time constraint is being violated, it might put a buffer in the output of the driving pin to ensure a faster transition. Obviously, this value will depend of the operating frequency of the cell, which means a max transition value will be defined for each scenario (mode of operation and corner). It should be noted that the value assigned to the maximum transition variable is not only related with logic correctness (a well defined logic value of '0' or '1') but also related with low power considerations, as a long transition leads to high short-circuit power dissipation, as already explained in chapter 2.

Another constrain associated with the Liberty Files is the maximum fanout which measures the loading-driving capabilities of a cell, affecting the number of inputs of gates that can be safely connected to its output. Again, a buffer might be added in the output pin of the cell to increase its driving power in case this constraint is being violated.

Library Exchange Format (LEF)

There are two different types of LEF files that can be distinguished, the technology LEF and the standard cells LEF.

On one hand, the technology LEF includes technology information provided by the foundry related with layers and vias definitions and many different rules. These include minimum spacing and width between metals and vias, minimum and maximum metal densities for each layer as well as the antenna rules that should be respected.

On the other hand, the standard cells LEF include information provenient from the standard cell provider. Information such as the size of each cell, an abstract geometric view (using metal polygons only) of the layout of the pins , and many other characteristics associated with their pins. The information regarding the pins include their dimension and location inside the cell, their direction (input, output or inout) as well as their function (such as signal, power or ground).

The standard cells LEF corresponds to the basic information needed for the purpose of the P&R tool being utilized instead of using a more heavy file like the GDS (which includes base layer information such as the poly and implant regions as well as the metal layer information not related with the pins) that are unnecessary for the P&R. When generating the GDS of the layout, however, the LEF files provided by the P&R tool need to be merged together with the device level information so that a complete view of each cell exists.

3.1.2 FLOW Directory

The FLOW directory is divided in multiple files (as illustrated by Fig. 3.3) that will now be analyzed.

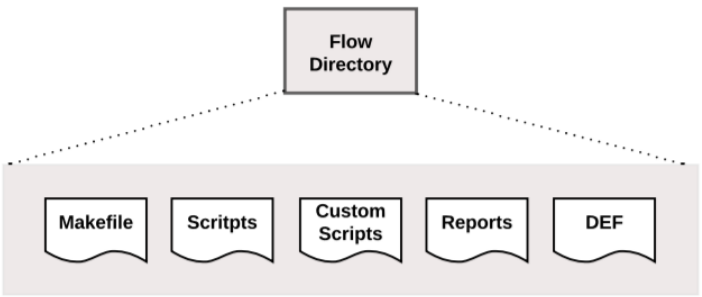


Figure 3.3: Files present in the flow directory

It contains all the different TCL scripts with sequences of commands for the different steps of the physical flow (design import, floorplanning, placement, cts, routing, signoff, as well as different optimization steps performed between each step) that take as an input the configurations made in the DESIGN directory.

Besides the default scripts present in this directory that can be reused and tweaked as desired for multiple projects, there's also the possibility to replace them by completely custom made scripts that might use the low-level commands provided by the tools as well as other higher-level commands provided by the kit.

The Makefile present in this directory works as a task sequencer, executing each step of the flow sequentially. In fact, the concept of a flow in general (being it a backend flow, a frontend flow

or even an ECO flow as will be seen in a further chapter) is heavily related with the dependencies of a Makefile. For example, if a clock tree synthesis task is launched but the Makefile checks that the floorplan task was the most recently executed task (by comparing the dates on a generated tag) then the results of the placement task are outdated and it will need to be automatically executed before the CTS can be performed.

The resulting database of each of the steps is saved, along with different logs that report the results of it. One important file in particular that is generated after each step of the flow is performed and which will be referenced regularly in this dissertation is the DEF (Design Exchange Format). It is a specification that consists of an ASCII representation of the physical layout at any point during the layout process, a physical view that can be visualized using a P&R tool. By comparing the physical view regarding the DEFs saved immediately before and after a certain step, one can visualize exactly what happened during that step, which is invaluable for debugging when the information provided by the logs and the tools' reports is not sufficient to do so.

The DEF ends up being an abstracted view of the GDS, containing only the rows definitions (which determine the valid locations for the placement of the standard cells), the location and orientation of each instance and the metal connections between them. The DEF and the LEF (which contains the information regarding the size of the cell and location and layout of the pins) provide a complete description of required information by a P&R tool to perform its job. The DEF can then be merged with the LEF and the device level information regarding the base layers (non-metal layers) as well as the metal information inside each standard cell (not related with the pins) to generate the final GDS file, the file containing all the information about the physical layout of a circuit which is ultimately sent to the foundry.

3.1.3 Floorplan Script

In the case of the implementation of this block, a custom made TCL script for the execution of the floorplan step in ICCompilerII was developed. It makes sense to have a custom script of the floorplan stage in particular, as it varies considerably from project to project. The number and size of the Macros, the number and location of the pins and the desired core shape and size are some of the reasons why the floorplan step is highly project dependent.

First, the core shape and size were defined, taking into account the size and shape of the AFE and RAM Macros. Another important parameter to define is the site type, which in this case was set to "CORE9T". It corresponds to height of each standard cell (which is also the height of each row) given in number of tracks, in this case 9 tracks (the other usual site in the industry being 12 tracks). The height of each track is technology information provided by the foundry and corresponds to the minimum spacing between a metal1 wire and a metal1 via. The number of tracks is also related to routing resources, as each wire runs in a track.

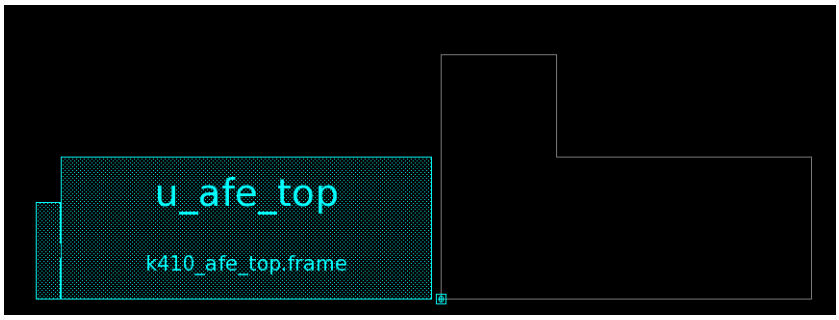


Figure 3.4: Core Shape

Then, the Macros were placed in their desired locations manually and oriented in such a way to allow their pins to be in contact with either the area reserved for the standard cells or with the boundaries of the core, in case of the AFE Macro pins which will be directly connected to the rest of the SoC once the block is integrated. After their placement, they were set with the status "Fixed" to ensure the P&R tool will not change their location in the following steps. As can be seen by the following picture, the Macros tightly fit in, as a result of the correct sizing of the core area in the previous step.

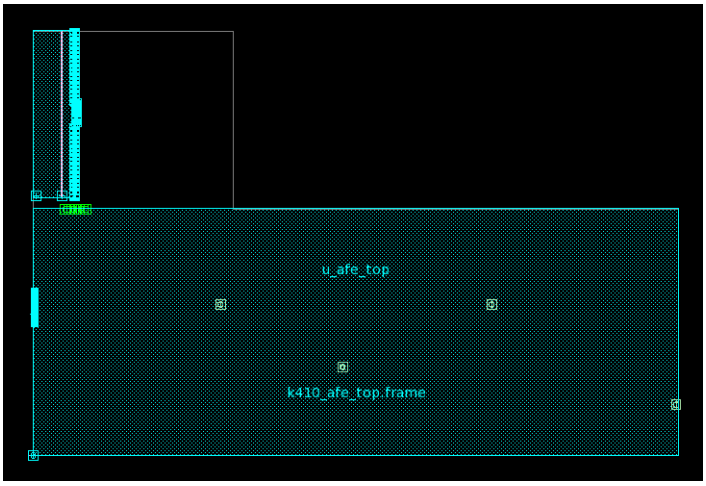


Figure 3.5: Macro placement

Afterwards, a placement halo was added around the AFE Macro to block the placement of standard cells in its interface. The objective is to reduce the congestion in the proximity of the pins of this Macro and ,for that reason, improve the routability of the design.

In the next step, the digital area pins are placed in the desired edge, honoring the pin spacing dimensions and with the desired metal layer.

After the pin placement, endcap cells (also known as boundary cells) are placed in the end of each row. They are used to isolate the different blocks in a SoC by protecting each block from external signals, and, most importantly, they break the continuity of the N-Well, preventing N-Well spacing violations issues in the DRC cleanup of the SoC once the block is integrated.

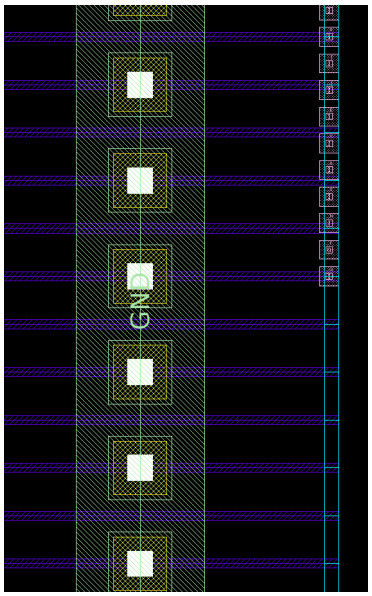


Figure 3.6: Close up of the mesh network, endcaps and pins

Finally, the power-ground distribution network was designed by adapting scripts from previous projects. First, a mesh network consisting of a set of stripes laid on in pairs of VDD-GND using the uppermost layers of the design was constructed. Higher metal layers are preferred for the construction of the mesh network since their increased width minimizes resistance and IR drop effects. Then, metal 1 rails where this mesh network meets the standard cells were created and connected to the mesh using stacked vias.

A floorplan DEF (Design Exchange Format) is thus generated by ICCompilerII. The floorplan DEF can then be used as an input to the next step, the placement. This means that if it is desired to run the physical flow on a different tool (in this case, Cadence Innovus) there is not the need to write a new floorplan TCL script, the already generated floorplan DEF representation can be simply loaded.

It should be noted that the floorplan displayed on this dissertation was created for an early stage of the circuit design. This means that the floorplan had to be severely tweaked as changes both in the shape and size of the AFE Macro and in the number of pins of the block occurred. Increasing the number of standard cells, which naturally happens as new functionalities are added in the RTL, means that the area definied for the standerd cells needs to be increased in order to maintain the same design density. The final floorplan design of this block is sensible information so it will not be illustrated.

However, by creating a script instead of simply using the graphical user interface to generate the floorplan, most of the changes that occurred throughout the development of the IP could be easily made by simply changing some variables of the script instead of having to start the floorplan step by scratch.

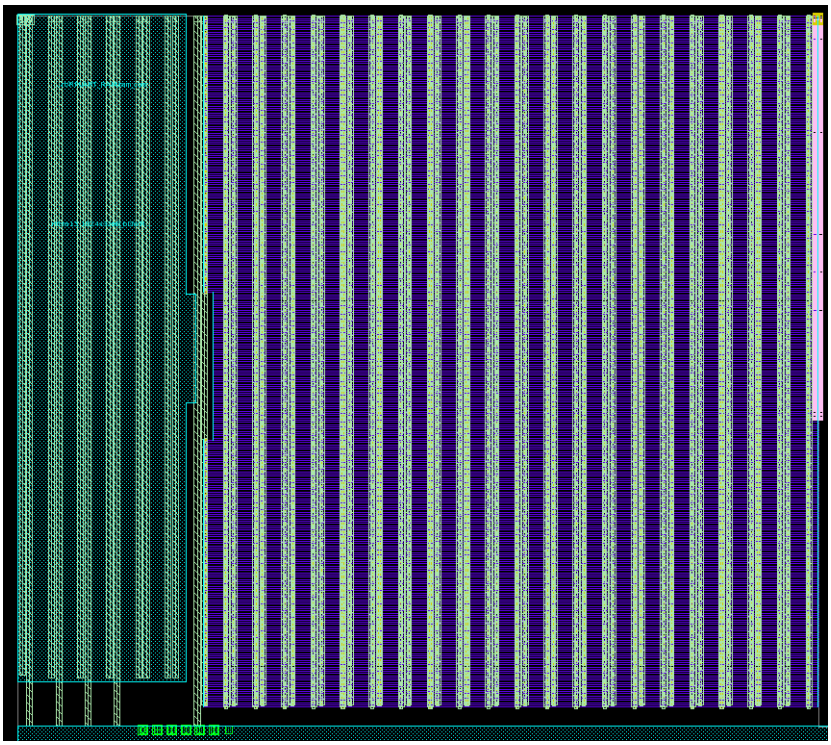


Figure 3.7: Standard-cell area after the Floorplan Step

3.1.4 Running the rest of the flow

After the floorplan was created, the remaining steps of the physical flow (up until the routing step) were performed one by one using the scripts provided by the ODIF kit.

After each step, there is the need to analyze the logs generated by the kit and correct any possible errors. Considering most of the features of the kit had been used in previous projects and were well tested already, most of the errors are generally related with the inputs of the flow that were defined in the DESIGN directory. Besides simple mistakes like incorrect path names or library definitions, the vast majority of the errors that had been corrected were related with the design constrains.

For example, the timing analysis after a certain step could lead to violations that were not specified in the timing exceptions. Setup errors in timing arcs in which the clocks are asynchronous (not defined as such using set_clock_groups) or related with a reset in paths not defined as false paths were common.

At the same time, changes in the RTL not yet updated in the constrains are very frequent. For instance, a new generated clock that was not defined in the timing constrains could lead to the clock pin of some flip-flops being connected to a net that is not of the type "clock" ,which means they would not be considered by the clock tree synthesis and timing analysis.

Therefore, the physical implementation of an IP is far from being a linear procedure. A design might be routed already, but if a small change occurs in the RTL, then the physical design flow needs to be repeated from the very beginning. It might seem inefficient or perhaps even strange not

to simply wait until the work of the frontend team is done until the physical implementation starts so that only one run of the physical design flow has to be made. However, it is intended to tapeout the circuit shortly after the frontend team has finished, specially considering time-to-market is the driving business requirement for the semiconductor business nowadays, making this parallelism between the two teams important. Every time the RTL suffers a small tweak the physical design will generally not require big modifications, ensuring that RTL changes near the tape-out can be easily implemented.

For an early RTL stage, the block was implemented in parallel in two different P&R tools, Cadence Innovus and Synopsys ICCII up until the routing step (which means the sign-off of the block was not finalized yet). In the next chapter, a comparison on the performance of the two tools with a special emphasis on the timing convergence and the power is made.

Chapter 4

Comparison of Synopsys ICCompilerII and Cadence Innovus

Many of the problems that physical design EDA tools need to face cannot be solved by polynomial-time algorithms. Thus, for problems of this nature, no known algorithms can ensure a globally optimal solution in a time efficient manner. As an illustrative example, finding an optimal placement of only 20 cells in a single-row so that wire-length is minimized and assuming evaluating the wire-length of a possibility takes 1 microsecond would take more than 77 thousand years [14]. This means different EDA tools that use different algorithms will certainly present different solutions for the same problem, and none of them will be optimal.

In this section, a comparison on the timing convergence and power dissipation presented by Cadence Innovus and Synopsys ICCompilerII on the implementation of the IP described previously will be made. Unfortunately, the license agreement provided by Cadence and Synopsys to STMicroelectronics explicitly prohibits the publication of the comparison of benchmark results. For this reason, instead of using the real names of the tools throughout the report, one of the tools will be named tool A and another tool will be named B. A comparison of tool A and B will be made, but the correspondence of tool A and B with either Innovus or ICCompilerII will remain inside STMicroelectronics.

This comparison was done while the RTL was still in an early design stage, with the intent of choosing one of the tools over the other for the rest of the development of the IP in case one them showcased significantly better quality of results. But of course, there are many other factors to keep in mind when it comes to making that decision, such as the level of expertise of the backend engineers in both tools or the number of available licenses.

4.1 Timing Convergence - With Post-Placement and Post-CTS Optimizations

When it comes to the metrics that will be compared, a summary on the maximum clock latency (relevant for setup analysis), minimum clock latency (relevant for hold analysis) and maximum

skew will be presented for each clock. A single mode of operation was used and in two different corners, the fastest and the slowest. Table 4.1 shows the results obtained for these metrics using the timing reports provided by the tools. Tool A presents a lower maximum clock latency for every clock of the IP and a lower minimum clock latency. When it comes to the skew, it depends from clock to clock.

Clock	Tool A	Tool B	Corner	Number of Flops
Clock: clk_peri				988
Maximum launch latency	1,2288	1,3735	Worst	
Minimum capture latency	0,3443	0,4105	Best	
Maximum skew	0,3931	0,2699	Worst	
Clock: clk_sys_apb				403
Maximum launch latency	1,0118	1,0058	Worst	
Minimum capture latency	0,2834	0,2837	Best	
Maximum skew	0,2993	0,1891	Worst	
Clock: clk_rf_13M56				426
Maximum launch latency	1,5425	1,659	Worst	
Minimum capture latency	0,1197	0,1867	Best	
Maximum skew	0,3357	1,0695	Worst	
Clock: clk_sys_ahb				24
Maximum launch latency	1,0095	1,2253	Worst	
Minimum capture latency	0,3032	0,3545	Best	
Maximum skew	0,3989	0,1736	Worst	
Clock: clk_848k				13
Maximum launch latency	1,211	1,5804	Worst	
Minimum capture latency	0,4431	0,4997	Best	
Maximum skew	0,0471	0,1248	Worst	

Table 4.1: Summary of latencies and skew per clock

Afterwards, a more detailed comparison of the critical paths (the path with the worst slack) of both tools for the clk_peri clock net was made. This clock was chosen as it is the one with the most clock sinks, which means achieving timing convergence will be a bigger challenge for the tools. For the original frequency of clk_peri (which is confidential), both tools had no trouble with timing convergence, which means all the paths had positive slack. Therefore, in order to truly test the full capabilities of the tools, clk_peri's frequency was incrementally increased by changing the constrain files until one of the tools was not able to converge. Naturally, every time the frequency of the clock was increased in the constrains, there was the need to re-run the physical implementation flow from the very beginning in both tools.

When the frequency was increased to $250MHz$, which is considerably higher than the original, tool B was not able to converge. The next two tables show the timing report for the critical path of both tools. In the first one, the critical path of tool A is analyzed in both tools. The two managed to have positive slack. However, in the critical path of tool B, while tool A was able to achieve a positive slack the same did not happen with tool B. Tool A, besides presenting a lower arrival time (due to the standard cells in the combinatorial path being closer to each other and/or the use of faster cells), presents higher clock network delay between the launch and capture flip-flops (skew) which increases the required time. Even though the CTS step tries to minimize the skew, it can

actually be beneficial in certain timing paths to prevent set up violations. To do this, the tool can place additional buffers in the clock path of the capture flop and purposely increasing the clock skew for this particular path.

Timing Analysis	Tool A	Tool B
Required Time		
Clock Period (ns)	2.0000	2.0000
Clock Network Delay (ns)	1.2551	1.1163
Setup (ns)	-0.2551	-0.2354
Uncertainty (ns)	-0.5500	-0.5500
CPRR Adjustment (ns)	0.0000	0.0000
Total (ns)	2.4528	2.3309
Arrival Time (ns)	2.3249	2.0815
Slack (ns)	0.1279	0.2494

Table 4.2: Critical path of tool A analyzed on both tools

Timing Analysis	Tool A	Tool B
Required Time		
Clock Period (ns)	4.0000	4.0000
Clock Network Delay (ns)	1.7429	1.1121
Setup (ns)	-0.2194	-0.244
Uncertainty (ns)	-0.5500	-0.5500
CPRR Adjustment (ns)	0.0431	0.0516
Total (ns)	5.0166	4.3683
Arrival Time (ns)	4.5161	4.8152
Slack (ns)	0.5006	-0.4469

Table 4.3: Critical path of tool B analyzed on both tools

Something to note on the previous results is that one of the main reasons for the timing convergence of the design, even for such an high clock frequency, is the post-placement and post-CTS optimizations. During these optimizations steps, the tools might reduce the arrival time considerably by swapping the original standard cells of the combinatorial paths by faster but also bigger versions of the cells that are available in the library. Obviously, even if timing convergence was obtained, there was the trade off of an higher design density.

4.2 Timing Convergence - Without Post-Placement and Post-CTS Optimizations

Even though the optimizations described in the last chapter are clearly important, it was desired to have a more fair comparison of the Placement and Clock Tree Synthesis algorithms of both tools. For that reason, several TCL scripts responsible for the physical design flow were modified so that the optimizations responsible for the swapping of the original standard cells present on the netlist were turned off. At the same time, the timing reports were analyzed right after the clock tree

synthesis step, which means the routing algorithm did not impact these results, unlike the previous comparison.

Even with the optimizations turned off, neither tool had trouble converging with the original clk_peri's frequency. A similar approach to the previous section was used where the clock frequency was incrementally increased. As expected, the minimum clock frequency in which one of the tools failed to achieve timing convergence was considerably lower with the optimizations turned off, $44MHz$ instead of the previous $250MHz$. The next table shows a report on the summary of multiple timing metrics while the one after it shows a detailed comparison of the critical path of both tools (which unlike the previous results, the critical path ended up being the same for both tools).

Clock	Tool A	Tool B	Corner
clk_peri			
Maximum launch latency (ns)	1,2634	1,2164	Worst
Minimum capture latency (ns)	0,3357	0,3931	Best
Maximum skew (ns)	0,3377	0,2713	Worst
clk_sys_apb			
Maximum launch latency (ns)	1,0285	0,9971	Worst
Minimum capture latency (ns)	0,2822	0,2928	Best
Maximum skew (ns)	0,2592	0,1939	Worst
clk_rf_13M56			
Maximum launch latency (ns)	1,3654	1,778	Worst
Minimum capture latency (ns)	0,0507	0,2322	Best
Maximum skew (ns)	0,345	0,9966	Worst
clk_sys_ahb			
Maximum launch latency (ns)	0,9472	1,2164	Worst
Minimum capture latency (ns)	0,2468	0,3769	Best
Maximum skew (ns)	0,1366	0,1742	Worst
clk_848k			
Maximum launch latency (ns)	1,1418	1,7539	Worst
Minimum capture latency (ns)	0,3791	0,5275	Best
Maximum skew (ns)	0,0274	0,2082	Worst

Table 4.4: Comparison of critical path timings for 44.44MHz frequency clock

Timing Analysis	Tool A	Tool B
Required Time		
Clock Period (ns)	22,5	22,5
Clock Network Delay (ns)	1,025	1,0434
Setup (ns)	-0,1788	-0,2377
Uncertainty (ns)	-0,55	-0,55
CPRR Adjustment (ns)	0,1482	0,1452
Total (ns)	22,9444	22,9008
Arrival Time (ns)	19,8451	23,7484
Slack (ns)	3,0993	-0,8476

Table 4.5: Comparison of critical path timings for 44.44MHz frequency clock

As can be seen from the results presented in this table, the reason why tool A managed to achieve a positive slack while tool B did not, ended up being a shorter arrival time.

By taking a closer look between the start-point and the end-point of this critical path, while both tools presented exactly the same type (leading to same size) of standard cells (which is expected, considering post-placement and post-CTS optimizations were turned off) tool B placed extra buffers in the combinatorial path, leading to a bigger combinatorial delay and, as a result, to setup violations. The buffers were placed after cells with high fanouts, in order to reduce it. As an example, this critical path contains in tool A one cell with a fanout of 40. In tool B, however, because of the insertion of a buffer, the fanout of that cell gets reduced to 20 (which means the buffer also has a fanout of 20 as it gets connected to the rest of the standard cells).

Nonetheless, the design constrains given to both tools are equal and neither of them had violations concerning maximum fanout or maximum slew. This could imply that tool B, by being more pessimistic regarding these constraints, ended up placing extra buffers in the combinatorial path that lead to the setup violation. It could also imply that a better placement algorithm by tool A lead to the standard cells of this critical path being closer to each other and, as a result, the extra buffers were not required to respect the design constraints. Some extra observations regarding these results will be made further in this chapter.

4.3 Power

Afterwards, a comparison on the power consumption of the implemented design using both tools was made, with all the optimizations turned on.

In order to generate the power reports, it was necessary to ensure that both P&R tools are using the same toggle rate. Toggle tate is the rate at which an instance switches compared to the clock it belong to and it is expressed as a percentage. In this case, it was set as 10% for both tools, ensuring the same switching activity between them which is one of the parameters of the switching, as seen in Chapter 2.

Obviously, if the objective was to do an accurate power analysis of the design simply assuming a toggle rate of 10% would not be sufficient and realistic switching activity figures obtained from gate-level simulations would be needed. However, in this case the goal was not to do an accurate power analysis but to compare the power performance of the tools, which means ensuring both of them are under the same conditions is sufficient. At the same time, this comparison was done in an early RTL stage, rendering an accurate power analysis useless as a lot of logic was still going to be added.

The following tables show the power reports provided by the tools. Internal power corresponds to the power dissipated inside a cell for the charging and discharging of its internal capacitances and also due to short circuit currents while the switching power is related with the charging and discharging of the load in the output of each cell.

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage
Sequential (mW)	0,3096	0,01784	4,64E-05	0,3274	41,11%
Combinational (mW)	0,04493	0,1057	3,72E-05	0,1507	18,92%
Clock Network (mW)	0,02359	0,1706	1,26E-06	0,1942	24,38%
Other (mW)	0,08118	0,04266	2,40E-04	0,1242	15,59%
Total (mW)	0,4593	0,3368	0,000325	0,7965	100%

Table 4.6: Power Report of Tool A

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	Percentage
Sequential (mW)	0,4173	0,00813	4,49E-05	0,4255	38,80%
Combinational (mW)	0,0524	0,2744	3,56E-05	0,3268	29,80%
Clock Network (mW)	0,004954	0,2062	5,60E-06	0,2558	23,40%
Other (mW)	0,1024	0,02937	2,19E-04	0,0875	8,00%
Total (mW)	0,5771	0,5181	0,000305	1,0956	100%

Table 4.7: Power Report of Tool B

It can be seen from these results that on both tools leakage power seems rather low, specially compared with the expected leakage power seems in chapter 2. However, it should be noted that the testing conditions are not realistic. A 10% toggling rate for every single cell ends up being unrealistically high, specially considering most modules have their clock gated and that devices spend the vast majority of their time in standby.

In the end, the implemented design using tool A dissipates $0.7965mW$ while the one using tool B dissipates $1.0956mW$.

By observing the total power used by the clock network, it is possible to make a comparison with the previous timing results. Tool A presented less clock latency, which implies that less CTS levels and buffers were used in the clock network, resulting in less dissipated power. Simultaneously, having a simpler clock tree network with less levels presents other advantages, such as less routing congestion.

Regarding the differences in combinatorial power consumption, it is likely that for achieving timing convergence tool B had to use bigger standard cells to reduce the arrival time, leading to higher currents and more dissipated power.

4.4 Observations

As observed in the previous sections, tool A presents better quality of results, both in timing convergence and in power dissipation.

Of course there are other metrics, such as runtime, that are also relevant when if comes to the performance of the tools, but making a fair comparison might not be feasible. In the case of runtime, the resources provided for the execution of each job (such as number of CPUs or amount of RAM memory allocated) is highly variable and not completely under the control of the physical design engineer.

Regarding the timing convergence, there is an important observation to be made. Considering the timing analysis was perform for an early design stage and, consequently, performed inside

the environment of the P&R tools, there are some implications. Each P&R tool, besides using different algorithms for each of the steps of the physical design flow, also uses different timing analysis simulation engines. As a result, some aspects such as the estimation of the RC extraction or the pessimism might defer from tool to tool. Obviously though, the quality of the timing analysis engine itself is an important comparison metric as it has direct implications on the physical implementation. In the example used above, regarding the insertion of buffers by tool B in the critical path to reduce the fanout of high-fanout cells, it seems the timing analysis engine of tool A did not see the need for the insertion of those buffers and, therefore, no setup violations occurred.

Nonetheless, in order to get a truly accurate comparison of the timing convergence of each tool, the sign-off had to be finalized and a more precise sign-off timing analysis tool (the same tool for both implementations) had to be used. It could be noticed that tool A, by being less pessimistic in its methods to reach timing convergence (using the same example again, by not inserting buffers after high fanout cells in the critical path), could lead to other timing violations only detected during the sign-off (after a complete RC extraction and, consequently, taking into account the real delay of each interconnection).

Chapter 5

Integration of the implemented block in a SoC

The previously implemented block TOP_ANA was then inserted into a System-on-Chip (SoC). In fact, the other components of the SoC correspond to a test chip which has the objective of emulating the radiation emitted by the rest of the circuitry of the contact-less card to test TOP_ANA, as some of the blocks of the final chip will only be developed after TOP_ANA is fully validated.

Even though it is expensive to develop and fabricate a test chip in silicone, there are some reasons to do so. On one hand, the integration of the implemented block TOP_ANA will be validated. This means that when the rest of the components of the final product are developed it will be quick to integrate TOP_ANA as it had been done before. On the other hand, TOP_ANA will be tested under similar conditions to the ones of the final product. Validating TOP_ANA with real conditions instead of having to resort to simulations greatly reduces the validation time (it could be the difference between one month and one year). A rigorous validation is specially important in the case of this block as the analog macro that belongs to it contains experimental capacitors.

Concerning the integration of this block in a SoC, multiple steps are needed. Considering the implemented block TOP_ANA will be seen as a Macro (a black-box) by the rest of the SoC, there is the need of making a Lib and a LEF (library formats explained in chapter 3) representations of the block. By doing so, this block can be instantiated in RTL, just like every module of the design. Simultaneously, the P&R tool can abstract itself from what is going on inside this block and simply focus on the characteristics provided by both the Lib and the LEF.

Another critical step is the design of the power intent of the SoC, which is considerably more complex than the power intent of the block as it contains two power domains, both a shut-down and an always-on domain with different voltage levels. The design of the power intent of the SoC will be the focus of this chapter.

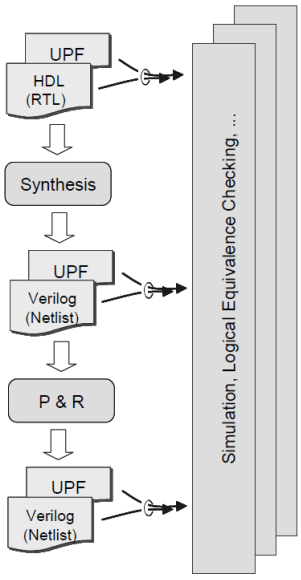


Figure 5.1: Role of the power intent in the design flow [35]

5.1 Role of the Power Intent in the Design Flow

The power intent of the design (the UPF file) is part of the design source. Both of them are coupled and serve as an input to Design Compiler (the synthesis tool used in this project) to produce the gate-level netlist.

At the same time, the UPF file is an important input to a P&R tool. It provides information on the desired location of its components, which is relevant in the placement step. Simply specifying the power domain of a component is not enough information since, as explained in chapter 2, a power domain is not necessarily physical contiguous, even if that might not be a desirable property, as will be seen later in this section. Two instances with the same power specifications might be placed in completely different locations.

Finally, the UPF file is also used for power aware simulations. Without it, simulations would simply assume a constant voltage which is not the case in multi-voltage designs or if some kind of power gating technique is being used.

5.2 Power Intent of the System-on-Chip

5.2.1 Power Domains

When it comes to the power design of the block, there are two different power domains, pd_top and pd_rom, illustrated in Fig. 5.2. Even though pd_top has multiple supply ports, only the supply ports VDD and GND are relevant for the explanation. VDD has 4 possible values, $1.5V$ for an high performance operation mode, $1.1V$ for a normal performance mode, 0.9 for a low performance operation mode and finally OFF, in the case of shut-down. It is important to note that although this power domain can be turned off, it does not use any power switch to do so. To

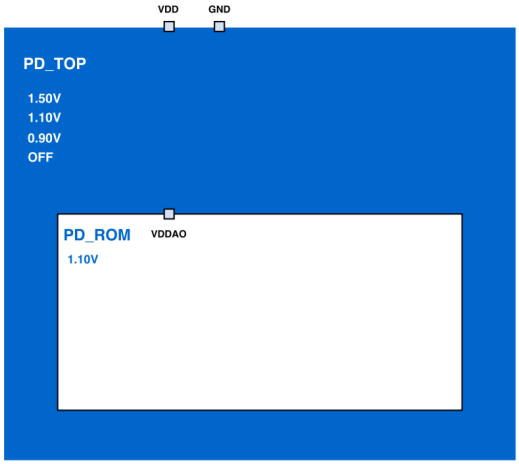


Figure 5.2: Power domains of the power design

explain it briefly as the functionality and design of the block embedded in the test chip is not the focus of this thesis, the contacless NFC card does not have its own power supply, it is battery-less. Instead, the field effect of the NFC card reader (sensed by the analog Macro placed in TOP_ANA) is the one that supplies the power. As a result, if the card is not being read it is simply turned off. At the same time, the power provided by the field is highly dependent on the distance from the card, which is why there are different performance operation modes.

Besides the already described pd_top power domain, there is also the pd_rom power domain which has the supply port $VDDAO$, a $1.1V$ supply that is always on. In this case, it corresponds to a power domain that is not powered by the field effect of the card reader but by an external power supply. This always-on power domain is concerned with blocks of the test chip and not with the IP block that is being tested.

5.2.2 Isolation and Level-Shifter Strategies

Considering there are two power domains with different voltage levels and that one of the power domains is a shut-down domain, both a level shifter and an isolation strategy had to be devised in the communication from the shut-down domain to the always-on domain and are shown in Fig. 5.3 (in the case of the communication from the always-on domain and shut-down domain there is no need for isolation as the always-one domain never turns off). In this design, instead of using independent level shifters and isolators, enable level shifters placed in the interface of the two power domains were used in order to save area. Regarding retention strategies, no retention registers were used in the shut-down domain. That is due to the fact that it is intended to reset every value on wake-up, which would make the retention registers useless.

The most interesting detail of this power design is that the enable signal for the isolation of the power domains (ISO_EN) is, as a frontend design decision, generated inside pd_top, the shut-down domain (more specifically, inside the block TOP_ANA which belongs to this power domain). This decision has some implications.

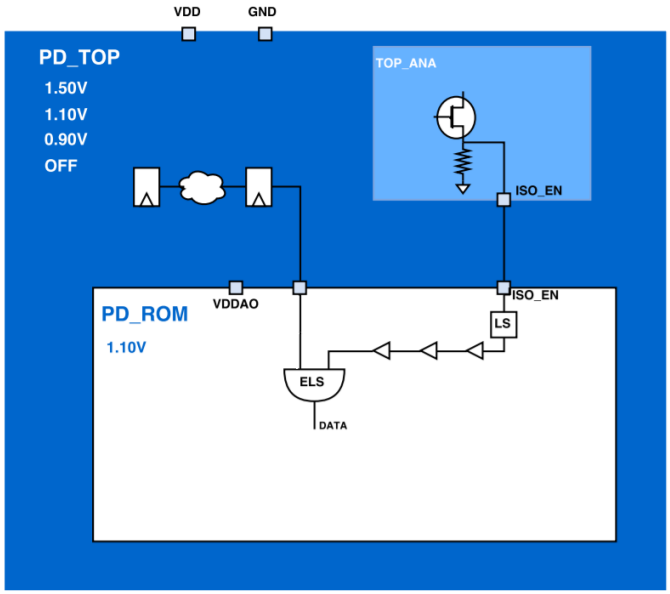


Figure 5.3: Isolation and level-shifter strategies

One clear problem is that it takes some time until it goes to zero (just like every other signal in this domain) which would defeat the purpose of the isolation, as signals with values below $1.1V$ would be connected to the the always-on domain before the isolation between the domains is activated, leading to the problems previously described in chapter 2, such as short-circuit currents. To solve this problem, a pull-down resistor that brings the enable signal to zero quicker was used inside the block that generates it.

Another problem is that there is the need to bufferize this signal, specially considering that (similarly to a clock signal) there is a huge fan-out associated with it as it is connected to the enable pin of every enable level shifter. However, if this signal is bufferized in pd_top then there would be signal integrity problems during the shut-down as the power supply voltage of the buffers would decrease. To work around this problem, the ISO_EN signal is first transferred to the always-on domain (with the need to shift its voltage, just like every other signal that crosses domains) and only then is it bufferized, by always-on buffers. To ensure that the P&R tool does not bufferize this signal on the shut-down domain, a "don't touch" flag must be raised on this net. Another solution to this problem would have been to bufferize the ISO_EN signal on the shut-down domain but using always-on buffers (which would therefore be in their own power island) . Nonetheless, this would require routing the power rails of the VDDAO power supply into the shut-down domain which would have been complex and would have decreased the routability of the design.

Finally, it was important to set this signal as an exception for the isolation strategy (otherwise, there would be the need to have another isolation enable to control the ISO_EN signal itself) and , as a result, to choose an independent level-shifter cell from the library.

To validate the level-shifter and isolation strategies devised, a visual representation of the strategies was observed using the GUI of Design Compiler after the synthesis of the block.

Chapter 6

Implementation of a metal-only ECO Flow

Nowadays, most VLSI corporations implement metal-only functional ECOs after the tapeout as carrying out design changes using only metal-layer changes saves a lot of money from a fabrication point of view considering the base-layer masks represent the majority of the cost of the mask set. For that reason, a metal-only ECO flow was developed for the implemented block, TOP_ANA.

When it comes to the implementation of an ECO flow, an important preliminary step is the insertion of ECO cells in the design during the block implementation. In this case, both reconfigurable cells and spare flip-flops were placed in the core area.

Afterwards, near the deadline of the tape-out of the SoC that integrates TOP_ANA, a metal-only ECO flow for functional ECOs was implemented for the same block. There was no need to perform any post-mask ECO modifications as the tapeout occurred briefly before the date this dissertation was written. Therefore, in order to validate the correctness of the flow, ECO modifications were performed in the RTL of the block that were not going to be made in silicon. However, with the implementation of an automatic metal-only ECO flow, any future post-mask ECO changes can be simply performed by changing the RTL, running the flow and analyzing the results, decreasing the time-to-market of the chip considerably.

6.1 Insertion of ECO Cells in the Design

During the physical implementation of the block, one important step was the insertion of cells for the possibility of future ECO modifications. In a sense, the insertion of ECO cells in the core area ends up being the very first step of an ECO flow.

In this design two different strategies were used: the insertion of Gate-Array reconfigurable filler cells and the insertion of spare flip-flops.

As was discussed in chapter 2, reconfigurable cells lead to less leakage power dissipation and are a more efficient use of area than spare cells. However, for cells of higher complexity there is the need to connect multiple reconfigurable base cells and the internal connections among them lead

to timing degradation. For that reason, due to the complexity of a flip-flop compared to most of the combinatorial cells, all of the inserted spare cells consisted of flip-flops, anticipating possible modifications in the sequential logic. A simple example is the addition of a state in a finite-state machine.

At the same time, sequential changes are more involved, and a different approach needs to be taken when performing them. Firstly, because the clock tree synthesis might need to be rerunned, which makes sequential changes rarer in post-mask ECO. Secondly, because in order for the sequential changes to be detected by an equivalence checking tool (that performs the logic difference extraction of the design, as seen in chapter 2), a spare flip-flop needs to be connected manually in the implemented netlist so that it can be seen as a compare point by the tool. It is for the second reason that inserting spare flip-flops also makes sense: a spare flip-flop module can be found on the netlist and this connection can be made manually, while a reconfigurable cell has no inputs, outputs or functionality.

6.1.1 Insertion of Spare Flip-Flops

Since spare flip-flops were placed along with all the other standard cells of the design, the placement script of the FLOW directory had to be adapted.

First, the number of spare flip-flops to place had to be defined. Considering the unpredictability of future modifications in the design, this number will always be an approximation based on what is expected. As was stated in the literature review, a rule of thumb is to have 2% to 5% of the design area occupied by spare cells [23]. However, in this design only spare flip-flops are being used, which makes that percentage unnecessarily high. The percentage was then defined to be only 3% of the number of sequential elements in the design. Afterwards, a spare module had to be created and placed, using the desired flip-flop cell type.

6.1.2 Insertion of Reconfigurable Cells

When it comes to the insertion of reconfigurable cells in the design, an interesting aspect is that they belong to a Gate-Array (GA) reconfigurable filler cells library and are placed as if they were filler cells, simply filling a percentage of the empty core area after all the other cells of the design were inserted.

More precisely, they were placed between the post-CTS optimizations step (in which standard cells can be swapped and buffers inserted to fix timing violations) and the routing step. They consist of an array of tiles in which each tile represents a basic reconfigurable cell, as was explained in chapter 2. The number of tiles for each Gate-Array filler cell is not fixed as it will depend on the size of the empty gap that it will be filled. If the gap the GA Filler cell needs to fill is considerably small it might even consist of a single tile.

One of the advantages of inserting reconfigurable cells as if they were filler cells is that they do not represent an inefficient use of area at all even if left unused, a big benefit when compared

to the spare cells. After all, the space they are filling would be left empty regardless and a normal filler cell which only extends the power rails and the well would take that spot.

Moreover, another benefit of this approach is that it does not interfere with the rest of steps of the flow, allowing the clock tree synthesis algorithm to place buffers in optimal locations and allowing more free empty space for the optimization steps to upsize or downsize standard cells at will. Simultaneously, the empty spots left out after the different steps of the flow tend to be somewhat uniform, allowing a homogeneous distribution of the reconfigurable cells throughout the core area.

Besides, it is natural to place the reconfigurable cells just like filler cells as the base cells without any functionality (before any reconfiguration occurs) end up behaving just like a filler cell, by simply allowing the continuity of the power rails and the well. The only difference between them and fillers ends up being the unconnected polysilicon gate and the active regions in case a functionality is required.

In order to validate the insertion of the ECO cells in the design, the implemented block was visualized using the GUI of Cadence Innovus, as illustrated in Fig. 6.1 (it can also be seen that some changes in the foorplan had already occurred at this stage compared to the one generated with the original script). In yellow, the spare flip flops are selected while the reconfigurable ECO cells are in red. The distribution of these cells is homogeneous and if a future ECO modification is required, it is very likely that there is either a reconfigurabe cell or a flip-flop in its proximity.

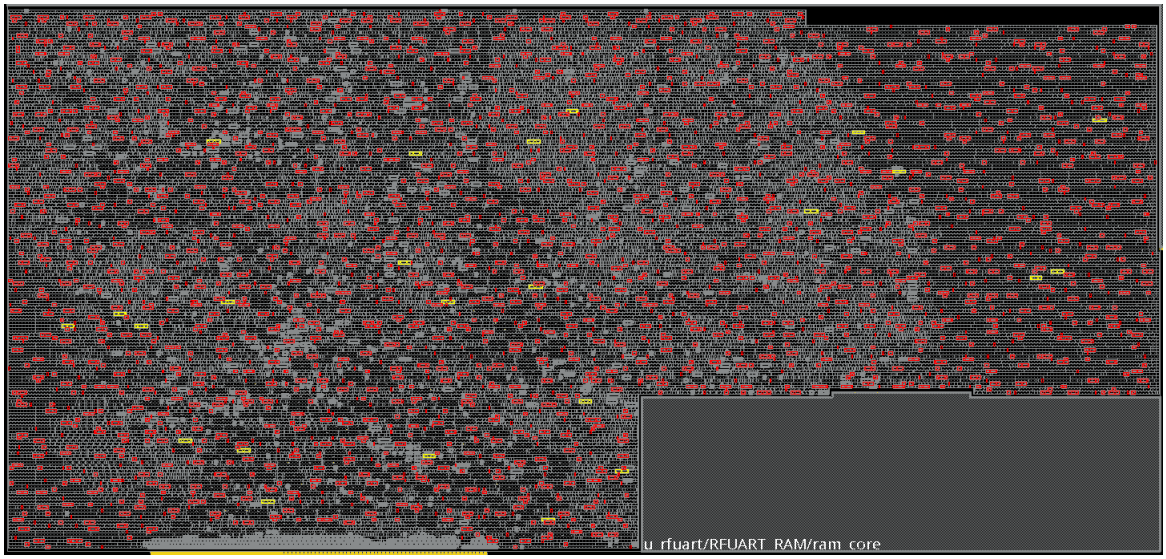


Figure 6.1: Result of the insertion of spare cells and reconfigurable cells in the design

Regarding the possibility of reconfiguring the unused reconfigurable GA fillers into decoupling capacitors (as seen in chapter 2) to decrease the IR-drop effect, the decoupling capacitors already present in the design were enough to reduce it to appropriate levels. In that case, it is simply better to leave the GA fillers as filler cells due to the increased leakage of the decoupling capacitors.

As a side note, some terminology needs to be clarified regarding these cells. GA fillers and reconfigurable cells will be considered equivalent along this dissertation from now on, while their

reconfigured versions are called reconfigured cells. At the same time, ECO cells are related with cells used to performs ECO modifications in general: both the GA fillers and spare cells (either a flip-flop or a freed combinatorial cell).

6.2 ECO Flow Overview

The overview of the implemented metal-only ECO flow for functional modifications is illustrated in Fig. 6.2. The objective of the flow it to perform ECO modifications done on the original RTL of the already taped-out chip and generate a new GDS file which should have equivalent base layers with the original design.

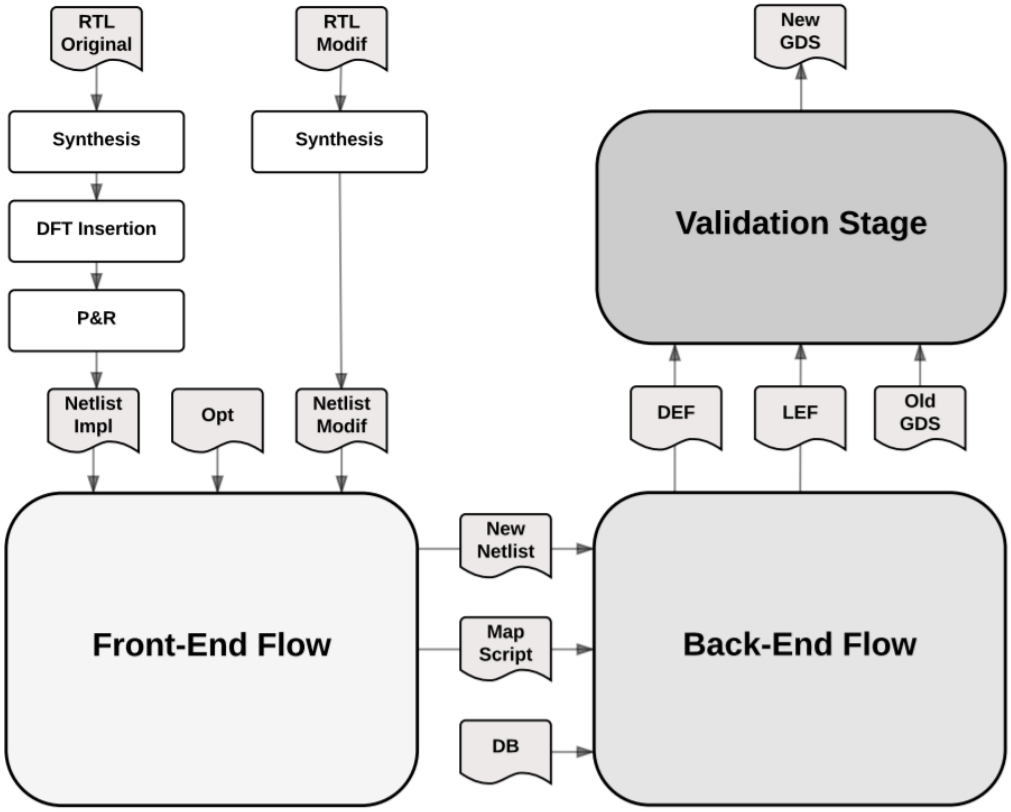


Figure 6.2: Overview of the implemented ECO flow. Legend: RTL Original - RTL of implemented design; RTL Modif - RLT of modified design; Netlist Impl - gate-level netlist of implemented design; Netlist Modif - gate-level netlist of modified design; New Netlist - new gate-level netlist of implemented design which includes the ECO modifications; Opt- Input files for patch optimization, explained further in this thesis; DB - Latest database of the P&R tool; DEF - Design Exchange Format file provided by the P&R tool, contains the information regarding the placement and routing of the design; LEF - Library Exchange Format file provided by the P&R tool, contains the informations regarding the size and metal polygons of each standard cell; Old GDS - GDS of the already tapedout chip; New GDS - GDS with the metal-only ECO modifications

The implemented ECO flow requires as an input the already implemented netlist , the gate-level netlist of the modified RTL, the latest P&R database of the implemented design, the GDS

associated with the taped-out design, as well as other optimization files that will be analyzed later in this chapter. While it would be possible to directly use as an input the modified RTL, it would increase the complexity of the front-end flow considerably (due to an higher number of differences between the designs) and the flow would become more error prone. At the same time, the digital area of the implemented block TOP_ANA is relatively small, which means synthesizing the modified RTL is not too time expensive.

The flow encompasses three major steps, which will be explained in detail in the next sections: The front-end flow, the back-end flow and the validation stage. The front-end flow, by taking the implemented netlist, the modified netlist and some optimization files, has the objective to generate the new netlist with the ECO modifications and a mapping script used to guide the back-end flow. Then, the back-end flow will open the latest P&R database of the taped-out design and implement the required modifications using only metal-layers by sourcing the script provided by the front-end. Finally, by taking as an input the DEF (which contains the placement and routing information of the implemented design, as seen in and LEF files provided by the back-end flow as well as device level information present in the GDS files of the standard cell kit, the validation stage will generate the new GDS file and ensure that only the base-layers of the old GDS were modified by comparing them layer by layer.

6.3 Front-End ECO Flow

In order to generate the new netlist and a mapping script for the back-end flow based on the implemented netlist and the modified gate-level netlist, the front-end flow requires three major steps, illustrated in the diagram of Fig. 6.3.

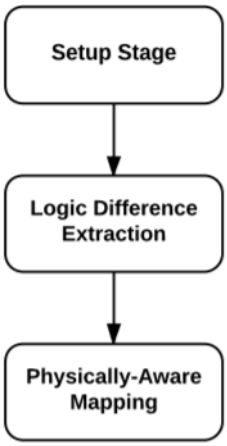


Figure 6.3: Stages of the front-end flow

Firstly, the setup stage, which has the objective to read the designs and the Libs and set up mapping constrains. Afterwards, the logic difference extraction step, which has the goal of finding the non-equivalent points between the implemented netlist and the modified netlist and generate a

patch file. Finally, the physically-aware mapping step, which has the objective of optimizing the patch that will be applied to the netlist and generate the mapping script provided to the backend-flow.

The tool used to perform the front-end flow is Conformal ECO Designer, by Cadence. It provides equivalence-checking, functional ECO analysis and physically-aware optimization capabilities that guide the front-end flow through the required steps to generate the new netlist and the mapping script. A script was created with the required Conformal ECO commands for each stage of the flow and is sourced by the tool to perform the desired ECO modifications.

Each of the stages of the front-end flow will now be uncovered and a detailed description will be made on the different steps that they consist of. In order to better understand each stage, an example of an ECO modification done on the original RTL will be used. The number of modifications was rather low so that running the different steps of the flow and debugging could be quicker.

6.3.1 Setup

The diagram in Fig. 6.4 shows the steps taken in the setup stage. Initially, Conformal is placed in Setup mode. At this stage, both the liberty files (Libs, which besides all the timing and power properties of each standard cell also contain the functionality associated with the output pin of each cell) and the two netlists are read by the tool in preparation for the logic difference extraction, as information about the functionality of the designs is everything Conformal will need to do so. By reading the libs, which contain the functionality of every standard cell, and the netlists, the tool will be able to segment both the designs into compare points and logic cones. As seen in chapter 2, this is a required preparatory step to start an equivalence checking between the two designs.

While compare points correspond to the design nodes at which the functionality is compared, the logic cones are the combinatorial logic that drive the compare points. Fig. 6.8, which will also serve as an example to a further stage of the flow, shows an example of a compare point and a logic cone for both the original design and modified design.

Then, both of the designs are flatten so that a flatten comparison can be done between them (which is the default and recommended comparison mode by the tool manual). In the explanation of the comparison between compare points further in this chapter the reason for doing so will be clarified.

In the final step of the setup stage, some constraints associated with the mapping of the compare points need to be defined. First, since one of the netlists was already implemented, scan chains are present in the netlist. Scan chains correspond to chains of flip-flops connected as shift registers used for DFT (Design for Testability) by allowing the observability of the state of internal registers of a design. Clearly though, these scan chains correspond to compare points and logic cones present in the implemented netlist that will not be matched by the modified netlist. To disable them during the equivalence checking as they are not related with the functionality of the design, the command *add pin constrain 0* needs to be used (similarly to what is done in the exception constrains with *set_case_analysis*, as seen in chapter 3).

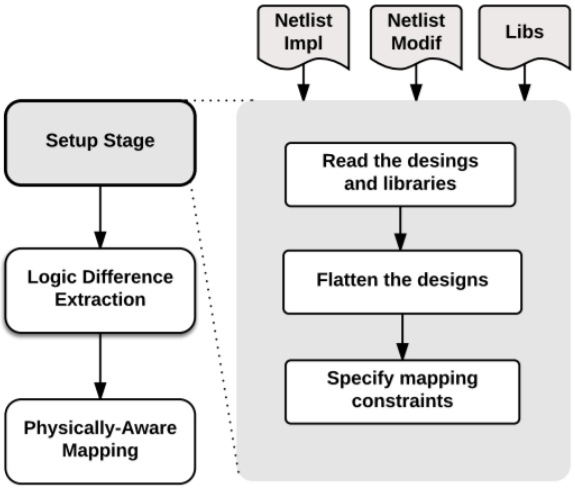


Figure 6.4: Steps of the setup stage

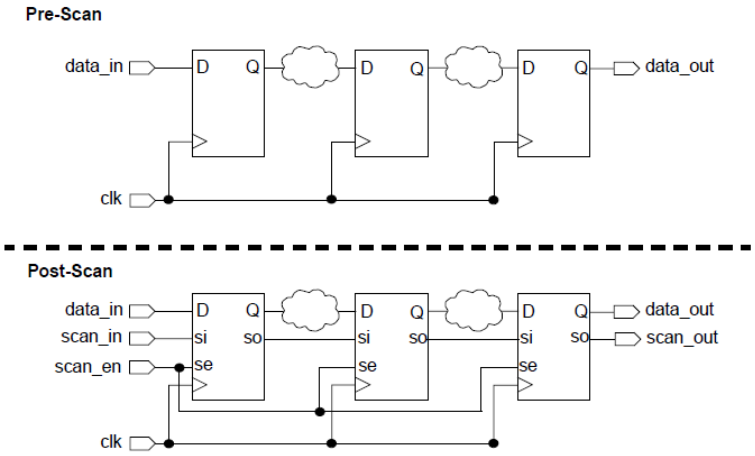


Figure 6.5: Comparison of a design with and without scan-chains inserted [34]

Yet another constrain to keep in mind is related with clock gating. Even though the modified netlist is not at the RTL level but at the gate-level (meaning the clock gating cells with latches to prevent meta-stability shown in chapter 2 were already inserted) it does not necessarily have the same number of latches as the implemented design, i.e., the same number of compare points. The reason is a concept called clock gating de-cloning/cloning [34]. During synthesis optimizations, if multiple clock-gated cells share the same enable signal then these cells can be merged into a single one to save area and power, a technique called clock gating de-cloning. However, synthesis tools do not have access to information related with the distance between each clock-gating element and the leaf cells. Thus, during the implementation, the P&R tool might detect that clock-gating related timing violations occur due to these physical distances. To solve these violations, it might do the opposite of what the synthesis tool did: increase the number of clock-gating cells to reduce the distance between them and the leaf cells (clock-gating cloning), as illustrated by Fig. 6.6.

For that reason, an option that remodels the clock-gating logic of the clock pin of the flip-flops before the logic difference extraction step needs to be turned on so that this technique is taken into account.

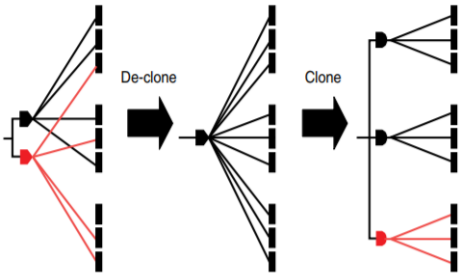


Figure 6.6: Clock-gating cloning and de-cloning [34]

6.3.2 Logic Difference Extraction

After this initialization stage, the logic difference extraction between the two designs can finally start by placing Conformal in the Equivalance Checking mode and by following a procedure similar to the steps already described in chapter 2.

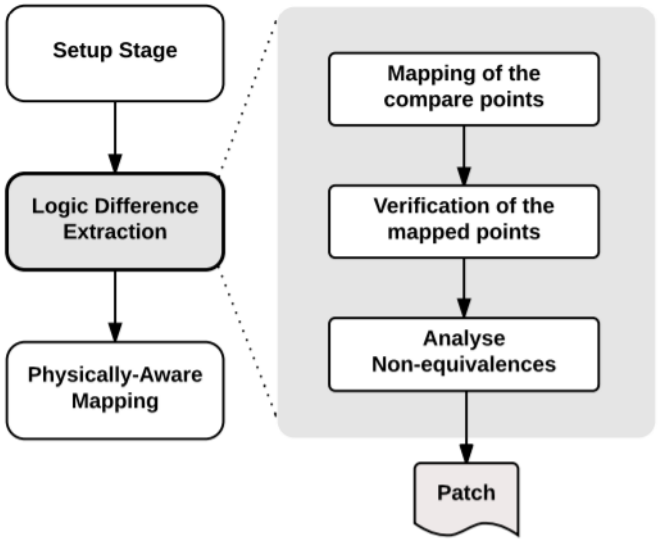


Figure 6.7: Steps of the logic difference extraction stage

First, the tool attempts to map the compare points of the implemented design to the correspondent compare points of the modified design. Initially, it uses name-based mapping techniques as they are more time efficient. Then, it uses mapping algorithms based on functionality of the logic cones to map the remaining ones. For every compare point in each design, Conformal reports three possible results: mapped, unreachable or not-mapped. Fig. 6.8 shows an example (which will be analyzed throughout the logic difference extraction stage) of two compare points that were

mapped between the original design and the modified design. Even though the names of the compare points are not exactly the same, the tool can still map them by filtering certain characters. Unreachable points, on the other hand, correspond to points without any logic cone driving them. Consequently, these compare points (which are usually related with the DFT constrains added in setup) do not influence the functionality of the circuit and are not relevant for the logic difference extraction.

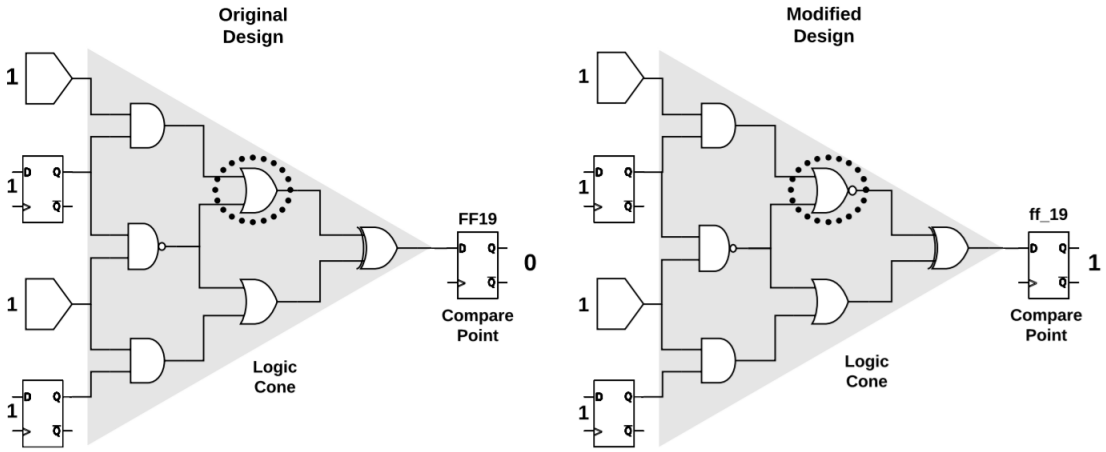


Figure 6.8: Two non-equivalent logic cones as their compare point shows different logic values. The difference that led to the non-equivalence must be detected

Not-mapped points, however, are problematic and all of them need to be solved before the equivalence checking can proceed. The two mapping constraints taken in the setup stage had the intent to reduce the possibility of not-mapped compare points. A not-mapped compare point might appear if during the synthesis sequentially-equivalent flip-flops are merged to save power, in which case the logs of the synthesis tools need to be checked. This situation will be more likely if two different synthesis tools were used for the implemented netlist and the modified netlist, which was not the case in this ECO modification example. Another possible reason for a not-mapped compare point is a sequential ECO modification in which a spare flip-flop was not manually connected in the implemented netlist, as that would clearly lead to a new added compare point that is not present in both designs.

Afterwards, there is a verification stage, in which the mapped compare points are compared with each other by checking the logic equivalence of the logic cones that drive them. Again, Conformal might report three different possibilities: equivalent, not-equivalent and aborted. By taking Fig. 6.8 as an example again, it can be seen that for the same inputs of the logic cone (the input-boundary) the value presented by the compare point is not the same in the original design and the modified design. Therefore, these two mapped compare points are marked as non-equivalent. As the number of ECO modifications in the ECO example being considered was low, only 13 points points were reported as non-equivalent and none were aborted. Aborted points are usually related with logic cones that are more complex, i.e, have more logic. There is a

certain timing threshold given to the tool to determine the equivalence between logic cones. If that timing threshold is surpassed, then no conclusion is reached and that point is given as "aborted". Besides some commands that have the intent of solving these aborted points, another possibility is performing an hierarchical comparison instead of the default and recommended flatten compare (in which both the designs are flatten before the comparison starts). It is intuitive to understand why doing an hierarchical comparison would help: the hierarchical boundaries would slice the logic cones, increasing their number while reducing the complexity of each.

However, while this divide and conquer strategy could help with the aborted points, boundary optimizations that occur during the synthesis of the design could lead to false non-equivalent points that would not occur if a flatten comparison was made instead. As an example, a possible boundary optimization that synthesis tools can do to optimize area is called inversion push. This technique occurs when an inversion is pushed across a register boundary, as illustrated by Fig. 6.9.

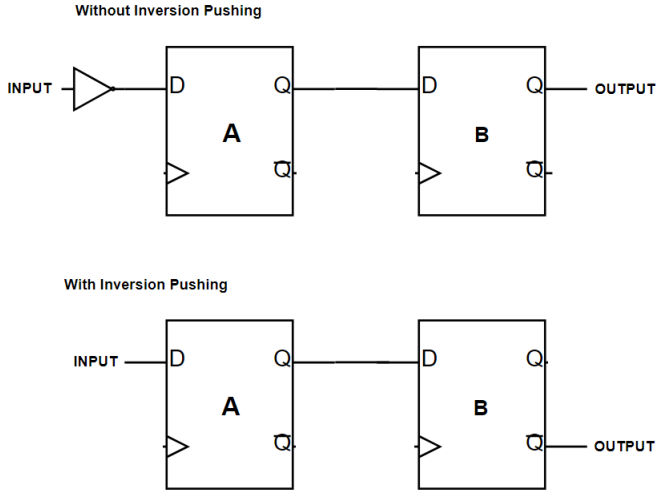


Figure 6.9: Inversion-pushing optimization technique used by synthesis tools

By pushing the inversion to the negative Q of the flip-flop B, there is no need to have the inverter connected to the input of the flip-flop A and, consequently, some area could be saved. Obviously, if this optimization had been done during the synthesis of the modified design but not during the synthesis of the implemented design, the compare points associated with the D pin of flip-flop A would be marked as non-equivalent. To take the inversion pushing technique into account during the comparison of the mapped compare points, the flatten compare has an option called phase mapping. With this option, the tool becomes aware of this technique and marks inverted-equivalent points as equivalent in case it detects inversion pushing was done. Inverted-equivalent compare points are points that have an inverted phase between them for the same inner-boundary conditions (the inputs of the logic cone, as seen in chapter 2).

Finally, in the last step of the logic difference extraction, the logic cones driving the non-equivalent points are analyzed and, based on the found differences, a patch is generated which contains the verilog module with the logic change and the port names corresponding to the nets

of the implemented design. By observing Fig. 6.8 yet again, it can be seen that the difference detected after analyzing the non-equivalent logic cones is a NOR gate in the modified design that corresponded to an OR gate in the original one. This verilog patch ends up being equivalent to the ECO list presented in chapter 2 as it defines the logic differences between the implemented and modified designs without taking into account the available ECO cells and their location, as will be seen in the example of the following section. It will serve as an input to the next step of the Front-End ECO flow: the physically aware ECO cell mapping of this patch.

6.3.3 Physically Aware Mapping

To better understand the physically aware mapping step, one of the performed modifications in specific will be used as an example, which will also be relevant in the explanation of the Back-End flow further in this chapter. The next verilog lines show the gate-level representation of the RTL change in the implemented netlist, the modified netlist and the new netlist, after the ECO modification is applied in the implemented netlist. It corresponded to a simple change of an *assign* statement, in which an OR was changed for a NOR, equivalent to the example in Fig. 6.8 used to explain the logic difference extraction.

```
#Original implemented netlist(G1)
HD40WT_OR2X5 U19(.A(n1),.B(n2),.Z(n3));

#Modified gate-level netlist(G2)
HD40WT_NOR2X7 U22(.A(n1),.B(n2),.Z(n3));

#Netlist after the ECO modification(G3)
HD40WT_ECO_NOR2X4 ECO_1(.A(n1),.B(n2),.Z(n3));
```

In relation to the standard cell libraries, there are some aspects that need to be clarified. As was seen in the insertion of ECO cells in the design, Gate-Array reconfigurable filler cells were added to the implemented design and they belong to the library HD40WT_ECO. However, this library also contains the reconfigured versions of the GA filler cells, meaning they have the same base layers but different functionality and dimensions. In fact, reconfiguring a GA filler is nothing more than swapping it with another cell from this library. This concept will become more clear in the explanation of the Back-End flow. Another detail is the "X" followed by a number in the name of the cell type (for example, X5 in the case of the OR2 cell in the original implemented netlist). This value represents the driving capability of the standard cell, which will obviously also affect the W/L ratio as versions of functional equivalent cells with higher W/L ratios will also have more driving strength. Finally, the 40 in "HD40WT" implies that a 40*nm* technology is being used.

When it comes to the physically aware mapping stage, it encompasses two steps, as shown by Fig. 6.10. This figure also includes the required input files of this stage, the patch generated by the logic different extraction and files for the optimization of the patch which will be analyzed later. For this example, patch.v corresponds to:

```
#Patch.v
HD40WT_NOR2X7 U22(.A(n1),.B(n2),.Z(n3));
```

It simply corresponds to the instance present in the modified gate-level netlist, a difference that was extracted in the previous step. However, only the connections and the functional behavior of the output pin present in the Lib of the specified cell are relevant. Not only does it not belong to the correct library of the reconfigured cells (HD40WT_ECO) but also because the driving strength does not take any placement information of the implemented design into account.

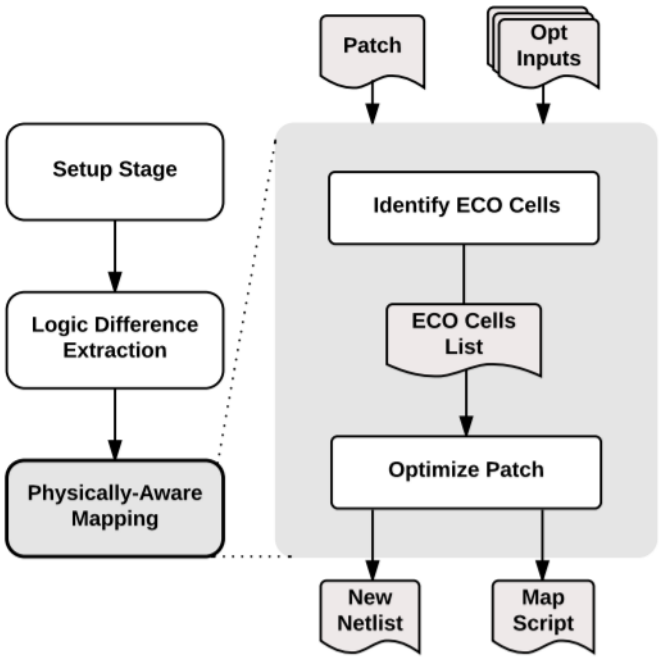


Figure 6.10: Steps of the physically-aware mapping stage

The initial step of this stage corresponds to identifying the ECO cells present in the design, using the *add_spare* command. To identify the ECO cells, both the DEF (already described in chapter 3) containing the instances and their placement information as well as a string pattern necessary to identity the ECO cells instances are required. For example, in this design in specific the pattern corresponds to the prefix ECO_*. Besides the ECO cells that were previously placed, it is also necessary to add to the ECO cells list the instances which were normal standard cells but became free as a result of the modification. After all, considering this ECO flow is intended for post-mask modifications and that a placed cell can not be simply deleted as that would certainly affect the base layers, it might as well be used as a spare cell for future ECOs. In the example being analyzed, the instance U19 present in the original netlist was therefore promoted to a spare cell.

After the available ECO cells are identified, the patch optimization step can begin (using Conformal's command with the name *optimize patch*). The function of this command ends up being the similar to the one of an ECO synthesizer, by completing the changes described in the patch by

using the physical information of the ECO cells and while trying to reduce the loss of routability and timing. The inner workings and algorithms used by this command are not know, but based on the previous study done on chapter 3 and adapting it to this particular design, a good way to formulate the optimization problem that the tool is trying to solve and wrap everything together would be the following:

Patch Optimization: Given an implemented design, an ECO patch, the placement of GA fillers and freed spare cells as well as the corresponding cell libraries, select cells from the ECO cells list to perform the functional changes described in the patch without violating the design constrains and with the minimum cost by estimating the loss of routability and timing degradation.

Based on the formulation of the optimization goal the tool tries to achieve, the other inputs to this step (illustrated in Fig. 6.11) can be more easily understood. As previously described, the patch corresponds to the functional modifications that need to be performed; The Libs allow to reconfigure the free GA fillers into the correct type by comparing the possibilities with the desired functionality of the patch; The DEF (which contains the information regarding the placement and routing of the design) is necessary for the information regarding the location, orientation and quantities of ECO cells so that decisions based on the physical distance can be made to allow location-aware ECO cell mapping; The LEF files (which contain the size of each cell and the metal polygons of the pins), complement the location-aware mapping by allowing to understand if a specific GA filler is big enough to be reconfigured into a cell with the desired functionality and driving strength, as well as providing the exact location of the pins inside each cell; The design constrains (as well as the standard cell constraints present in the Libs) allow the tool to choose a cell of the appropriate driving strength so that constrains are not violated and timing degradation can be reduced to a minimum; Finally, the revised ECO list provided by the previous step provides information regarding available ECO cells, including the recently freed spare-cells.

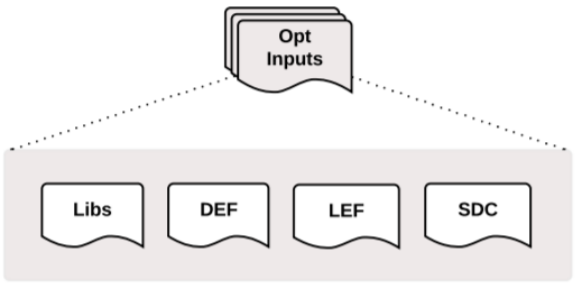


Figure 6.11: Inputs required by the optimize patch step besides the netlist and the patch

With these input files, a technology mapped optimized patch is generated by the tool and applied to the implemented netlist, the result being a new netlist with the ECO modifications. As seen from the verilog lines above, the ECO modification in the example appears in the new netlist as HD40WT_ECO_NOR2X4. As expected, it belongs to the same library as the ECO GA filler

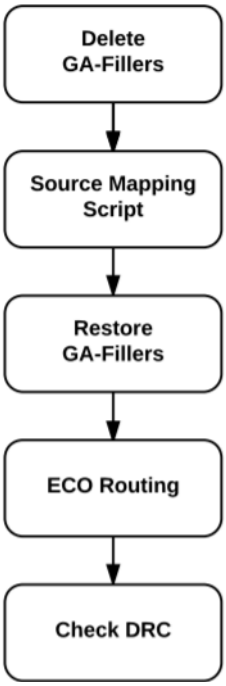


Figure 6.12: Stages of the backend flow

cells and it corresponds to a reconfiguration of one of them to a NOR with a driving strength of X4. Besides synthesizing a new netlist, this patch optimization step also generates a mapping script that will be used to guide a P&R tool to map the newly added ECO logic to specific ECO cells resources in the layout and to create a revised list of the available ECO cells. This file is a critical aspect of the Back-End flow and will be explained in detail in the next section.

6.4 Back-End ECO Flow

After the front-end flow has generated the new netlist and the mapping script, the back-end flow can initiate. The P&R tool used for this stage was Cadence Innovus and a script was created to implement all the desired changes (inside which the mapped script is sourced). In fact, considering it is desired to run the commands on the latest database of the implemented design, an ST script intended for post-signoff optimizations was adapted to accommodate the back-end ECO flow for this particular block. After this script is sourced by the tool, a new database is saved and the DEF and LEF provided by it will be used by the validations stage to generate the new GDS file.

The back-end flow encompasses the steps illustrated in the diagram of Fig. 6.12. While the first three steps are related with the reconfiguration of the mapped GA fillers, the last two are related with ECO routing and the required DRC checks.

6.4.1 Reconfiguration of the mapped GA Filler Cells

In order to understand the process taken to reconfigure the chosen GA filler cells into the correct standard cell type, the mapping script provided by the optimize patch step in the Front-End flow needs to be deconstructed and understood. Considering the second step of the Back-End flow corresponds to sourcing this script, the first step of the flow will be skipped for now. The diagram showcased in Fig. 6.13 illustrates the procedure followed by the script to perform the physically-aware ECO modifications. Some of TCL commands of the script associated with the ECO modification example used in the last section will be used to clarify these steps. For a more graphical representation, Fig. 6.14 illustrates this example.

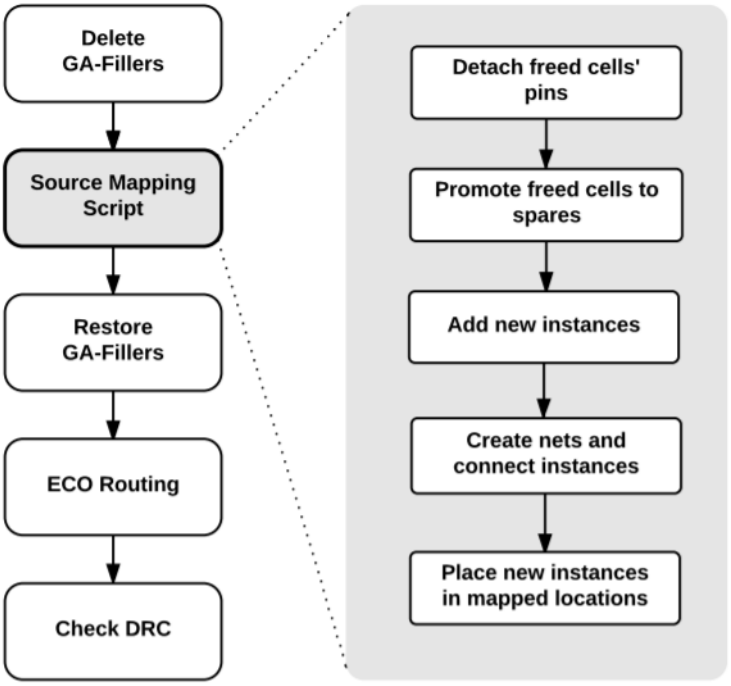


Figure 6.13: Steps followed by the mapping script

First, the script simply disconnects the pins of every freed cell, the cells that became unused after the modification. Their inputs pins also need to be connected to either GND or VDD as they can not be left floating.

```
#Disconnecting the pins of the freed U19 instance
detachTerm {U19} {Z} {A} {B}
attachTerm {U19} {A} {B} 1'b0
```

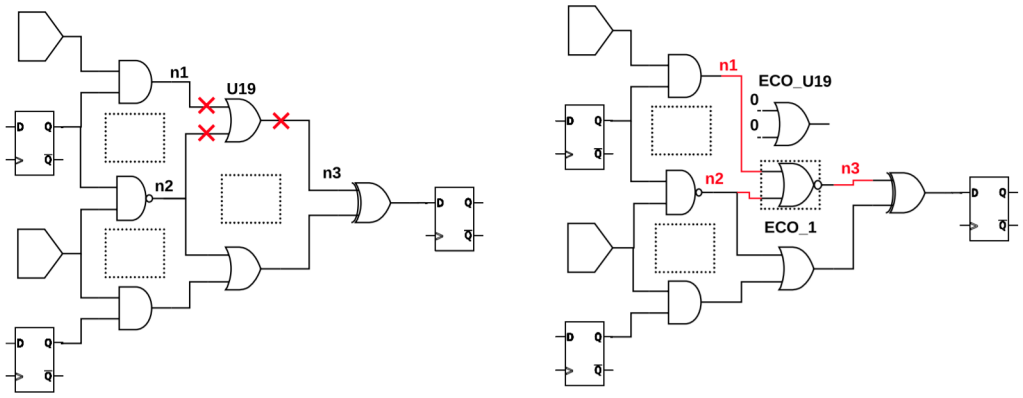


Figure 6.14: Actions performed by the mapping script. The dotted squares represent the spots where a GA filler is present. On the left, nets $n1$,$n2$ and $n3$ are disconnected from U19 and the inputs of this cell are connected to GND. On the right, the reconfigured NOR gate was placed in the spot chosen by the physically-aware mapping step and logically connected to the nets $n1$,$n2$ and $n3$.

Afterwards, the cells that became unused need to be promoted to spare cells so that they can be used for other ECO modifications. To do so, they need to be renamed in order for the pattern used by the "Identify ECO Cells" in the Front-End flow can detect them the next time the ECO flow is run.

```
#Promoting U19 to a spare cell
ecoSwapSpareCell {ECO_U19} {U19}
```

The next step is to add the new instances in the design (the reconfigured version of the GA fillers) and to make the required connections due to the ECO modification. At this stage, these connections are purely logical, only after the routing step will they actually become physical. It should also be noted that in this particular modification the nets to whom the added cell connected to were already present in the design. If not, new nets required by the ECO modification are created by the script.

```
#Add reconfigured cell to the design
addInst -cell HD40WT_ECO_NOR2X4 -inst {ECO_1}

#Make the necessary connections
attachTerm {ECO_1} {A} {n1}
attachTerm {ECO_1} {B} {n2}
attachTerm {ECO_1} {Z} {n3}
```

Finally, the mapping script places the newly added reconfigured cell in the desired location and with the desired orientation.

```
#Placement of the reconfigured cell
placeInstance {ECO_1} 1137.240 41.580 R0
```

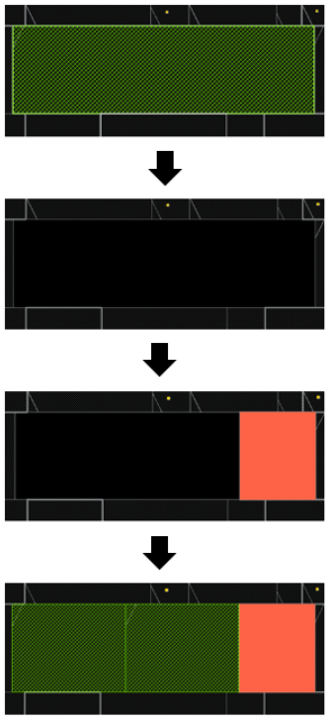


Figure 6.15: Strategy used with the GA fillers. In green, the GA filler cells. In orange, the reconfigured cell.

The location determined by the "Physically Aware Mapping" step, besides being on a spot that minimizes the cost of the ECO modification, should also be on top of a GA Filler cell big enough to accommodate the new instance, which was ensured by the optimize patch command by comparing the dimensions of each with the information present on the LEF file. However, as was explained in the last section, even though both the GA filler cells and the reconfigure cells belong to the same library (HD40WT_ECO) and have equivalent base layers (non-metal layers), they are still independent cells with their own properties. Obviously, the reconfigured cell can not be simply placed in a location which already has another standard cell, even if that cell is a GA filler. For that reason, the first and third steps of the Back-End flow are required. Initially, every single ECO GA filler cell is deleted from the design. Then, the mapping script that was analyzed is sourced and the new reconfigured cells are placed in the core area, occupying some of the empty spots left out from the deletion of the GA fillers. Finally, the GA fillers are added back in the design, filling the rest of the empty space. The four pictures shown in Fig. 6.15, obtained using the GUI of the P&R tool for each of these stages illustrates the procedure that was taken.

At first sight, it might seem strange to simply delete and place cells in a taped-out design. After all, the masks are already fabricated and a full re-spin of the chip is undesirable. However, it is only relevant to look at the initial and final databases of the implemented design. Even if some placement and deletion orders were given to the tool, the truth is that the only difference is the reconfiguration of a portion of the GA fillers, which means the base layers should remain the same.

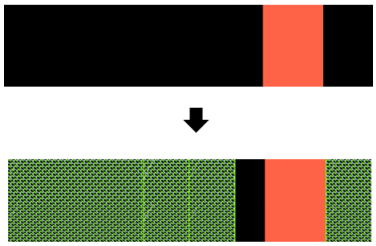


Figure 6.16: Gap found after GA filler cells re-insertion

Another aspect to take into account is that GA fillers, just like every other standard cell, have a limited number of different versions. As a result, they are restricted in widths variety (in the case of the HD40WT_ECO library used in this design, only 3) and, therefore, confined to fill gaps of such widths. Consequently, there needs to be a relationship between the widths of the reconfigured cells available in the HD40WT_ECO library and the width of the available GA fillers:

Condition: The width of the empty space left out by a reconfigured cell after its placement, must necessarily be a multiple of of one of the available GA fillers widths or be the multiple of at least one of the combinations of their sum.

By looking at the library (and even by simply observing Fig. 6.15 and Fig. 6.16) one can notice that there are three different available GA filler widths which are not multiples of each other. It is perfectly understandable, as having them as multiples of each other would be redundant and greatly diminish the number of possible combinations. Of course this aspect has an impact on the potential of the ECO modifications. On one hand, by limiting the functionality of the reconfigured cells that fit a certain gap. Even if the gap is big enough, it still needs to respect the condition defined above. On the other hand, by limiting the possible driving strengths that can be chosen for a certain gap, as not all of them will respect the condition.

Fig. 6.16 illustrates an example where this condition was violated. It occurred during the implementation of the flow, a situation in which a gap appeared in the core area after the re-insertion of the GA fillers. While the empty space to the right of the inserted cell respected the condition, the empty space on the left did not: the resultant gap after the re-insertion is too small to be filled by any GA filler, it could only be filled by a normal filler cell as they have smaller versions. By taking a closer look at the problem, it was found that the inserted cell did not belong to the HD40WT_ECO library and, as a result, did not respect the condition defined above. This bug occurred due to an incorrect pattern choice in the "Identify ECO Cells" step, leading to the insertion of cells from other libraries too. For that reason, it is important to check the existence of any empty gaps left out in the design area after the re-insertion of the fillers (by using the command *checkFiller*. In any case, even if by chance the empty space had been filled, this situation would still be problematic due to the modification of the base layers as a non-reconfigured cell was placed in the core area.

6.4.2 ECO Routing and DRC verification

After the reconfiguration of the chosen GA Fillers, the logic connections that were performed (or ,in some cases, deleted) by sourcing the mapping script need to be routed. To do so, an ECO router needs to be used. As was seen in chapter 2, the already existing routing patterns from the original design make the ECO routing complex.

While a default router is allowed to rip up and reroute the existing nets in order to provide more flexibility, reduce congestion and, therefore, reduce the likelihood of DRC violations, an ECO router is more greedy and limits the number of reroutes of existing nets to a minimum in order to reduce the cost of the ECO modification (in this case, by minimizing the impact on the timing convergence of an already taped-out design). For that reason, not only is it less likely that the routing can be completed successfully, but even if it does, DRC violations might have been committed in the process. Consequently, it is of upmost importance to perform a DRC verification right after this step is concluded, inside the P&R environment. Of course this DRC test does not substitute the DRC performed by a sign-off tool. Less checks are performed since the P&R tool uses the abstracted view LEF for the cells, which does not contain the base layers. In the ECO example being considered, all the metal layers were used in the ECO routing (up until metal 5) and the DRC check was clean.

6.4.2.1 Tests on the feasibility of limiting the allowed metal-layers

After completing the routing of these ECO modifications, checking the DRC and validating the flow by verifying the base-layers were unchanged (which will be explained in the next section) some tests were done on the routing of the same ECO modifications. The objective was to gauge the feasibility of performing the ECO routing using only some of the metal layers, which would reduce the number of metal masks and, therefore, greatly reduce the cost of a possible ECO.

As a sanity check, the number of layers was initially restricted to metal 1 only. In these circumstances, the routing simply could not be completed, which is completely understandable as the power rails that supply the VDD and GND pins of the standard cells are routed in metal1, the lowest level of the power mesh grid. Therefore, it is not possible to transverse from one row to another without using an higher metal layer.

Then, the routing was tested using firstly only the metal layers 1 to 2 and then 1 to 3. In both situations the routing was completed successfully, but DRC violations occurred during the DRC verification. While in the first case (using the metal1 and metal2 layers) there was a considerable number of violations, in the second one only a relatively small number of checks were reported (around 10). If these modifications were going to be implemented in silicon, it would have made sense to individually analyze these few DRC checks (which were mostly related with minimum spacing violations) and either tweak the ECO routing settings or try to manually solve them as that could lead to considerable savings in the re-fabrication of the masks.

Considering these modifications were just simulations, the same test was applied for metal layers 1 to 4 and both the routing and the DRC verification were successful. Therefore, limiting

the number of allowed metal layers for the ECO routing would deny the need to re-manufacture the metal 5 mask.

However, these results should be taken with a grain of salt. In these simulations the number of ECO modification was rather low (only 13 non-equivalences) and, therefore, easier to route. At the same time, it is important to remember that this block (TOP_ANA) was integrated in a SoC along with other blocks. If a new set of metal-masks is going to be fabricated for some metal-only ECO modifications in TOP_ANA, it is very likely that other blocks of the SoC will take advantage of the re-fabrication of the masks to perform some ECO modifications of their own. By bundling together the desired ECO modifications, a single set of metal-mask masks can be re-fabricated instead of multiple. Consequently, it is very likely that some of these modification will require routing using metal 5 regardless, which means it would be useless (and even detrimental) to limit the routing resources in TOP_ANA.

6.5 Validation Stage

In order to validate the flow by generating the new GDS, performing a final DRC check and finally comparing the old GDS with the new to ensure the base layers remained unchanged, a physical verifcation tool called Calibre (from Mentor Graphics) was used. Besides the GDS of the already taped-out design, the inputs to the validation step are the DEF and LEF from the P&R tool that together with the device level information (present in the GDS files of the standard cell kit) will be used to generate the new GDS of the design. The steps taken by this stage are illustrated in the diagram of Fig. 6.17.

6.5.1 Generating the new GDS file and performing a final DRC

As was previously stated, in P&R environments standard cells and Macros are represented as LEF cells which are an abstracted view of the real cells that contain information regarding the size and location and layout of the pins (the only information required by the P&R tool). All the LEF cells present in the design must be merged with the GDS device level information concerning the base layers and the metal-layers inside the standard cell besides the pins. Afterwards, this complete GDS view of each standard cell is merged with the placement and routing information present in the DEF and a full GDS description of the design is generated.

After the generation of the GDS file, a final DRC test was done on the same phyisical verification tool, also taking into account violations related with the base layers. In the ECO example, since the DRC verification was clean (reinforcing the DRC test done on the P&R tool in the Back-End flow), a comparison between the old GDS and the newly created one must then be made.

6.5.2 Performing the XOR comparison between the two GDS

After the a successful DRC verification, it is fundamental to compare the differences in each layer between the two GDS files to ensure the base layers remain unchanged.

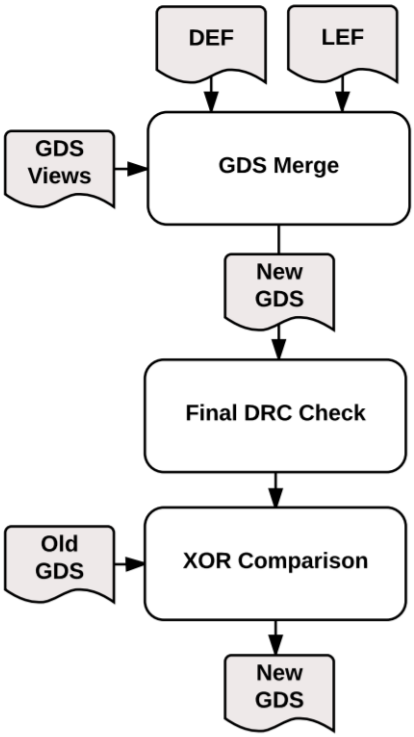
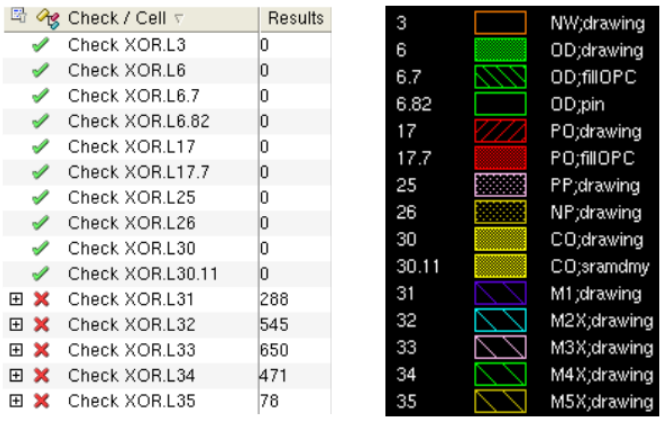


Figure 6.17: Steps followed by the validation stage

To do so, the layout XOR feature (fastXOR) provided by Calibre was used. It takes as an input the old GDS and the new GDS and overlaps them, layer by layer. Afterwards, it performs a comparison of each layer in order to detect any differences. Fig. 6.17, extracted from Calibre's GUI, shows the differences found by the XOR for each base and metal layers in the design (layers related with vias and pins are not shown for a simpler representation).



Check / Cell	Results
Check XOR.L3	0
Check XOR.L6	0
Check XOR.L6.7	0
Check XOR.L6.82	0
Check XOR.L17	0
Check XOR.L17.7	0
Check XOR.L25	0
Check XOR.L26	0
Check XOR.L30	0
Check XOR.L30.11	0
Check XOR.L31	288
Check XOR.L32	545
Check XOR.L33	650
Check XOR.L34	471
Check XOR.L35	78

Figure 6.18: XOR results between the old GDS and the new GDS. On the left, the differences found for each layer. On the right, the association between the labels and the respective metal layers.

Expectedly, while all the base layers remained the same, differences were found in every

metal-layer as no restrictions were given to the ECO routing. To better appreciate the reconfiguration procedure, Fig. 6.19 shows the differences in the base-layers and metal 1 layers of the old and new GDS views. While the base layers are identical, it is possible to observe the reconfiguration of the GA filler into a functional cell using metal1 connections.

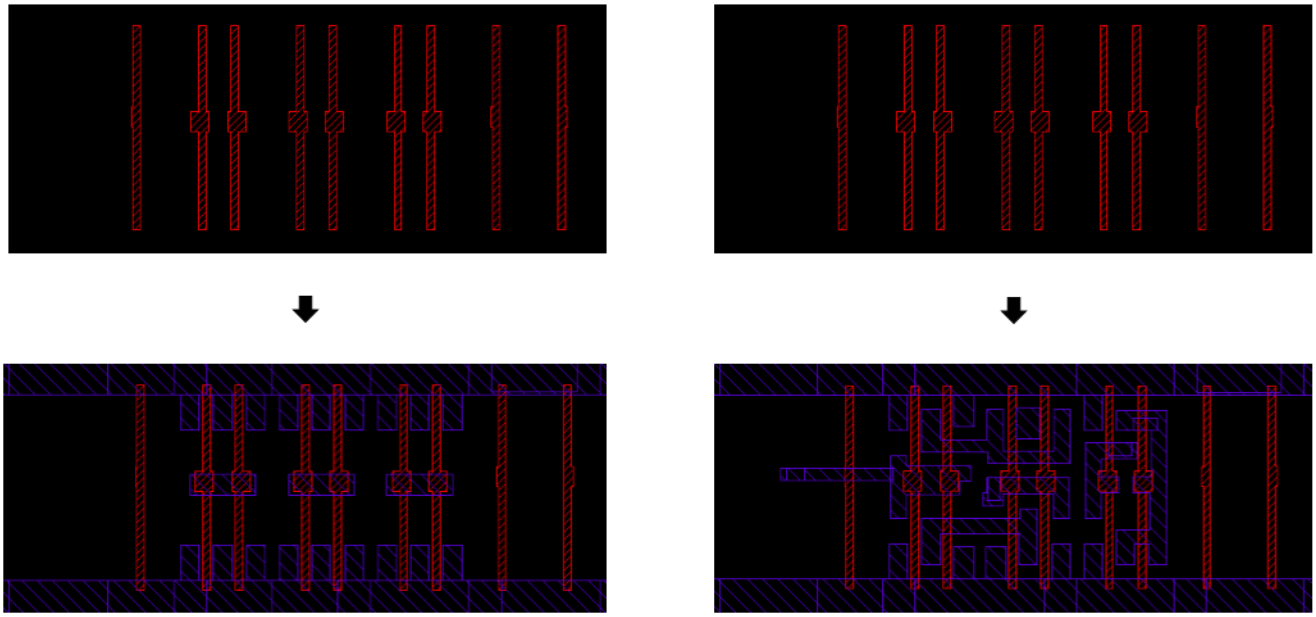


Figure 6.19: Comparison of the old GDS with the new GDS

Chapter 7

Conclusions and further work

Considering the initial objectives of this dissertation, it can be said that they were fulfilled. An IP was implemented in a team environment and two P&R tools were compared, with some relevant conclusions regarding the quality of the results presented by both concerning timing convergence and consumed power. Then, the implemented IP was integrated into a SoC and low power techniques were used, which are imperative nowadays to reduce the power consumption of integrated circuits. Finally, a post-mask function ECO flow was implemented and validated, meaning any future ECO modifications in this block can be simply performed by changing the RTL, running the flow and analyzing the results, decreasing the time-to-market of the chip considerably. Nonetheless, some further work can be made to improve some of the developed work.

When in it comes to the comparison of the two P&R tools, it would have been interesting to understand how to have more control on the resources provided for the execution of each job (such as number of CPUs and amount of RAM allocated) so that a fair comparison on runtime could be made. Even though one of the tools had better quality of results both in timing convergence and power, runtime is also a critical aspect, specially considering the importance of time-to-market nowadays.

Also concerning the the two tools, in order to get a truly accurate comparison of the timing convergence, the sign-off had to be finalized and a more precise sign-off timing analysis tool (the same tool for both the implementations) had to be used, as already explained in chapter 4.

Regarding the implementation of the ECO flow in specific, since it was developed near the tapeout of the SoC (which coincided with the date this dissertation was written) there was not enough time to further test and optimize it. One interesting aspect to analyze would have been the limits of the ECO cells insertion strategy that was used by making an high amount of RTL modifications and understanding at what point the number of available ECO cells simply is not enough to implement them. By doing this study, the feasibility of a desired post-mask ECO could be accessed in advance and no time had to be wasted performing it.

On the same note, a study could also have been made concerning the type of ECO cells that were used. Different approaches such as a mixture of both combinational spare cells and gate-array reconfigurable filler cells with different percentages could be used to reach an optimal insertion

strategy. In this case, the results would be purely theoretical and relevant for futher projects considering this design was already finalized and that changing the ECO insertion strategy is obviously not possible after the tapeout.

However, both of these studies would require an high amount of ECO simulations and each one of them would require multiple modifications, leading to an unreasonable amount of time to reach a definite conclusion, specially considering the number of available licenses for each EDA tools is an important resource in the industry

References

[1] Shantanu Dutt and Wenyong Deng. VLSI circuit partitioning by cluster-removal using iterative improvement techniques. In *Proceedings of the 1996 IEEE/ACM International Conference on Computer-aided Design*, ICCAD '96, pages 194–200. IEEE Computer Society.

[2] F. Rubin. The lee path connection algorithm. 23(9):907–914.

[3] F. O. Hadlock. A shortest path algorithm for grid graphs. 7(4):323–334.

[4] Khosrow Golshan. *Physical Design Essentials: An ASIC Design Implementation Perspective*. Springer, 2007 edition edition.

[5] A. Jayalakshmi. Functional ECO automation challenges and solutions. In *2nd Asia Symposium on Quality Electronic Design (ASQED)*, pages 126–129.

[6] Eric Cheung, Xi Chen, Furshing Tsai, Yu-Chin Hsu, and Harry Hsieh. Bridging RTL and gate: correlating different levels of abstraction for design debugging. In *2007 IEEE International High Level Design Validation and Test Workshop*, pages 73–80.

[7] Haihua Su, Sachin S. Sapatnekar, and Sani R. Nassif. An algorithm for optimal decoupling capacitor sizing and placement for standard cell layouts. In *Proceedings of the 2002 International Symposium on Physical Design*, ISPD '02, pages 68–73. ACM.

[8] Smita Krishnaswamy, Haoxing Ren, Nilesh Modi, and Ruchir Puri. DeltaSyn: An efficient logic difference optimizer for ECO synthesis. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, ICCAD '09, pages 789–796. ACM.

[9] H. Y. Chang, I. H. R. Jiang, and Y. W. Chang. Simultaneous functional and timing ECO. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 140–145.

[10] Hsien-Te Chen, Chieh-Chun Chang, and TingTing Hwang. New spare cell design for IR drop minimization in engineering change order. In *2009 46th ACM/IEEE Design Automation Conference*, pages 402–407.

[11] K. Shahookar and P. Mazumder. VLSI cell placement techniques. 23(2):143–220.

[12] C. Y. Tan and I. H. R. Jiang. Recent research development in metal-only ECO. In *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4.

[13] I. H. R. Jiang and H. Y. Chang. ECOS: Stable matching based metal-only ECO synthesis. 20(3):485–497.

[14] Andrew B. Kahng, Jens Lienig, Igor L. Markov, and Jin Hu. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. Springer Science & Business Media. Google-Books-ID: DWUGHyFVpboC.

[15] E. G. Friedman. Clock distribution networks in synchronous digital integrated circuits. 89(5):665–692.

[16] Paulo Francisco Butzen and Renato Perez Ribas. *Leakage Current in Sub-Micrometer CMOS Gates*.

[17] Márcio Cherem Schneider and Carlos Galup-Montoro. *CMOS Analog Design Using All-Region MOSFET Modeling*. Cambridge University Press, 1 edition edition.

[18] Artur Balasinski. Optimization of sub-100-nm designs for mask cost reduction. 3(2):322.

[19] Hsien-Te Chen, Chieh-Chun Chang, and TingTing Hwang. New spare cell design for IR drop minimization in engineering change order. In *2009 46th ACM/IEEE Design Automation Conference*, pages 402–407.

[20] X. Meng, R. Saleh, and K. Arabi. Layout of decoupling capacitors in IP blocks for 90-nm CMOS. 16(11):1581–1588.

[21] Neil Weste and David Harris. *CMOS VLSI Design: A Circuits and Systems Perspective*. Pearson, 4 edition edition.

[22] H. Y. Chang, I. H. R. Jiang, and Y. W. Chang. Timing ECO optimization using metal-configurable gate-array spare cells. In *DAC Design Automation Conference 2012*, pages 802–807.

[23] H. Y. Chang, I. H. R. Jiang, and Y. W. Chang. ECO optimization using metal-configurable gate-array spare cells. 32(11):1722–1733.

[24] C. Y. Tan and I. H. R. Jiang. Recent research development in metal-only ECO. In *2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 1–4.

[25] B. H. Wu, C. J. Yang, C. Y. Huang, and J. H. R. Jiang. A robust functional ECO engine by SAT proof minimization and interpolation techniques. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 729–734.

[26] A. C. Ling, S. D. Brown, S. Safarpour, and J. Zhu. Toward automated ECOs in FPGAs. 30(1):18–30.

[27] Y. M. Kuo, Y. T. Chang, S. C. Chang, and M. Marek-Sadowska. Spare cells with constant insertion for engineering change. 28(3):456–460.

[28] N. A. Modi and M. Marek-Sadowska. ECO-map: Technology remapping for post-mask ECO using simulated annealing. In *2008 IEEE International Conference on Computer Design*, pages 652–657.

[29] S. L. Huang, C. A. Wu, K. F. Tang, C. H. Hsu, and C. Y. Huang. A robust ECO engine by resource-constraint-aware technology mapping and incremental routing optimization. In *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, pages 382–387.

[30] S. Y. Fang, T. F. Chien, and Y. W. Chang. Redundant-wires-aware ECO timing and mask cost optimization. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 381–386.

[31] D. Blaauw, Anirudh Devgan, and Farid Najm. Leakage power: trends, analysis and avoidance. In *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.*, volume 1, pages T–2 Vol. 1.

[32] Cadence. *Encounter Conformal ECO Designer*. Cadence, 2016 edition.

[33] D. Gale and L. S. Shapley. College admissions and the stability of marriage. 69(1):9–15.

[34] Ken Cheung. Practical guide to low-power design – user experience with CPF | EDA blog.

[35] IEEE standard for design and verification of low-power integrated circuits. pages 1–736.