Automated Layout Porting and Optimization of digital designs

Karthik Srinivasan Shekar
ka7245sh-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Academic Supervisor: Pietro Andreani

Supervisor: Babak Mohammadi

Examiner: Erik Larsson

October 4, 2021

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

The aim of the thesis is to investigate and implement an algorithm that can automate the porting of the layout from one technology node to another technology node. The layout optimization of any design block can be implemented for better performance. The primary task of the project is to develop a tool that can optimize and port the complete layout design that reduces the time and effort of manual work. The optimization algorithm is tested for each block of the design and nearly 90% of the design is DRC clean. The porting of any design blocks is also implemented and it has been tested for any DRC issues. By modifying the length and width of the design blocks using the tool, the analog layout can be generated in less time based on the design requirements. For each block of design, the pitch distance between the two poly-silicon was also modified so that we can generate the layout of the design for different process nodes.

Popular Science Summary

More revolution is happening in 21st century with technological innovation, every human being is connected one way or other in latest technology evolution. We need to learn the concept of new technology and practicing the knowledge in a meaningful manner to provide better support for the organization and uplifting the individual standard of living through technological means. According to Moore's law, the number of the transistor is doubled every 18 months. so the manufacturing of the fully custom IC is not realistic. Hence most of the IC manufacturing has to be automated to reduce the workload of the manual work hours and to release the product soon to the market. As the complexity increases the risk of manufacturing process variations is more likely to affect the system functionality.

The digital design elements such as standard cells are used to implement the fundamental logic gates such as AND, XOR, NOR, OR into larger hardware blocks that are used to build a fully custom IC. Every IC has smaller digital blocks as standard cells and it is very crucial to optimize these elements for better design metrics such as power, performance, and area. Optimizing these blocks manually is a very tedious task and it takes thousands of work hours to design.

In this thesis, an algorithm is developed to optimize the layout by the length, width, and pitch modification and design the layout based on the design requirements. Also implemented compiler to port the memory design for different technology as it saves a lot of manual work. The approach is designed in a way such that a minimal amount of technology-specific information is needed while still producing an efficient layout. Since the design requirements vary from time to time, these automation scripts are used to extract the desired layout in seconds and save a large amount of labor cost and time.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The increasing complexity of the Integrated Circuits(IC) has paved the way for more advanced developments in recent years. Since the complexity of each development is increased, Electronic Design Automation (EDA) has been subject to research in recent years. Before the development of EDA, all the tasks were implemented by manually laid out. The basic concept was to use reliable, low cost,and relatively low technology IC processes and pack a large number of projects in a wafer in a short duration.

To device, an complex digital circuits functionality in current IC, Register Transfer Level (RTL) language is commonly used. The RTL uses registers as input networks and transistors as switching devices. It provides a high-level abstraction of the hardware design and the designer uses Hardware Description Language (HDL) to code the behavioural design with low-level digital circuits components. Once the behaviour code is implemented, then it is interpreted through the synthesis process that realizes the functionality using the logic gates from the technology library.

The Standard cells are the interconnected structure of the transistor that provides the functionality of boolean logic functions such as AND, OR, inverters. The initial design of the standard cells are developed at the transistor level using the netlist. The netlist file consists of the transistor information, connection,and corresponding ports. We can use SPICE to simulate the netlist behaviour by running different analysis like transient, DC,and AC analysis. The results of the simulation give the overall idea of the power consumption, propagation delay,and leakage parameters.

A good layout design lies in the way its visual elements or polygons are organized and placed to each other. Analog layout design also impacts the manufacturability and reliability of the design. The Layout porting, modification,and migration for each digital design elements such as standard cells is a more time-consuming task when it is done manually.

For each process technology, manually redesigning is a very difficult and tedious task. In addition to this, the designer must also abide by the technology-specific design rules. The vast amount of parameters and constraints to consider forces the designer to employ automation using an optimal algorithm.

1.1 Background

As the number of transistor in a chip will double for every two years as stated by Gordon Moore, has been one of the most crucial and revolutionary law that improved the market for the semiconductor industry. The demand for better performance, power reduction and cost-effective and compact devices have been driving the semiconductor industry to change from micrometer scale to nanometer range in recent years. The size of the transistor is extremely small in 5nm range and it has been further scaled down for better design metrics, [1].

System on Chip (SoC) integrates most of the components of the electronic system. These components include a Central Processing Unit (CPU), memory blocks, input/output ports. For the higher performance of the SoCs, it is paired with dedicated and physically separate memory chips. An SoC integrates a microcontroller, microprocessor, and several processor cores such as GPU, peripheral circuits, and memory. SoCs are rising to prominence in the embedded systems market. More tightly integrated computer system designs improve performance and reduce power consumption, as well as semiconductor die area than multi-chip designs with equivalent functionality. Designers must make practical trade-offs in performance, power consumption, and die area, or PPA, [12], in virtually every SoC implementation.

It is important for design teams to choose the right embedded memories and logic library IP for their SoC designs. This high-performance functionality must work seamlessly together in a single integrated device and operate with the lowest possible power consumption to preserve battery life all at a cost that is economical enough for the consumer market. The trend of memory utilization in typical SoC shows the increasing importance of memory. Memory currently occupies a majority of SoC and the utilization is above 70%. Due to the high utilization of memory, it becomes further important to ensure its functionality and improve the Power Performance Area (PPA) of the memory layout design.

1.2 Project aim

The main objective of this thesis is to design the layout automatically by reducing the thousands of hours spent on manual design. To evaluate the performance of the design, we need to compare the area and the speed of the digital design elements with the industry-standard alternative. The main goal of this project is porting the layout to different technology and process nodes. This compiler will help the designer to reduce the time, labor cost and generate the layout based on the design requirements.

The setup for the layout automation is in place and it can generate all digital design elements with ease, but the design has its limitations that can be addressed. When designing the layout of the digital design elements, we need to consider the design parameters like leakage, area, and performance of the cell. There is always a trade-off in optimizing the design for each parameter and the designer must abide

by all the specific design rules in each technology.

The layout of the memories and standard cells are optimized based on the design requirements. The layout is modified by the length, width, and pitch of the polygons in each technology layout cells. The second task of the thesis is to generate the layout of design layout blocks from one technology node to another technology node. By generating the ported version of the layout, it will reduce the hours of manual works and minimize the turnaround. The layout is generated automatically using the GDSPY module, [16] in python and the design is 90% DRC clean. Python was chosen as the tool for developing an algorithm and automate the flow.

1.3 Thesis goals

Chapter 1. Introduction This chapter describes the main objective of the thesis and why it is important to develop an EDA tool in the current scenario.
Chapter 2. Theoretical Background: In this chapter, all the required background information required for the task is explained and the algorithm used for design is also discussed.
Chapter 3. Modification of the layout: In this chapter length, width, and pitch modification of the layout and implementation methods are explained.
Chapter 4. Porting of the layout: In this chapter porting of the standard cells and memories are implemented for different technologies and implementation methods are explained.
Chapter 5. Conclusion: Discuss the final results of the tasks and final thoughts about the thesis and future works.

Chapter 2

Theoretical Background

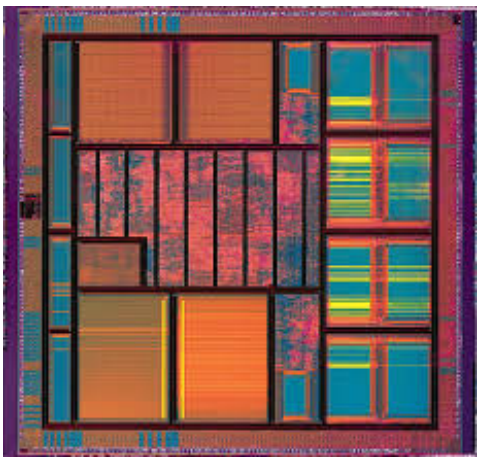

Figure 2.1: Integrated Circuits Die [23]

2.1 Introduction to VLSI

About 40 years ago, the MOS Integrated circuits technology integrated nearly less than 10,000 transistors in a chip, this paved the way for the VLSI Design of the chips with more than thousands of transistors in a single chip [29]. Subsequent advancements added more transistors as individual functions or systems were integrated over time. The first integrated circuits had only a few devices as many as ten diodes, transistors, resistors, and capacitors, making it possible to fabricate one or more logic gates on a single device known as small-scale integration (SSI), improvements led to devices with hundreds of logic gates, known as medium-scale integration (MSI). Further improvements led to large-scale integration (LSI), i.e. systems with at least a thousand logic gates as shown in table 2.1.

Current designs use extensive design automation and automated logic synthesis to layout the transistors, due to higher levels of complexity in the resulting logic functionality. Certain high-performance logic blocks like the SRAM (static

random-access memory) cells are still designed by hand to ensure the highest efficiency. Structured VLSI design is a modular methodology originated for saving microchip area. This is obtained by the arrangement of rectangular macro blocks which can be interconnected using the wiring. In the early 1980s, the methodology lost its popularity because of the advent of placement and routing tools wasting a lot of area by routing because of the progress of Moore's Law, [2].

Name	**Year**	**Transistor count**
small-scale integration(SSI)	1964	1 to 10
medium-scale integration(MSI)	1968	10 to 500
large-scale integration(LSI)	1971	500 to 20,000
very large-scale integration(VLSI)	1980	20,000 to 10,00,000
ultra-large-scale integration(ULSI)	1984	10,00,000 and more

Table 2.1: List of Semiconductor scale and Transistor count.[6]

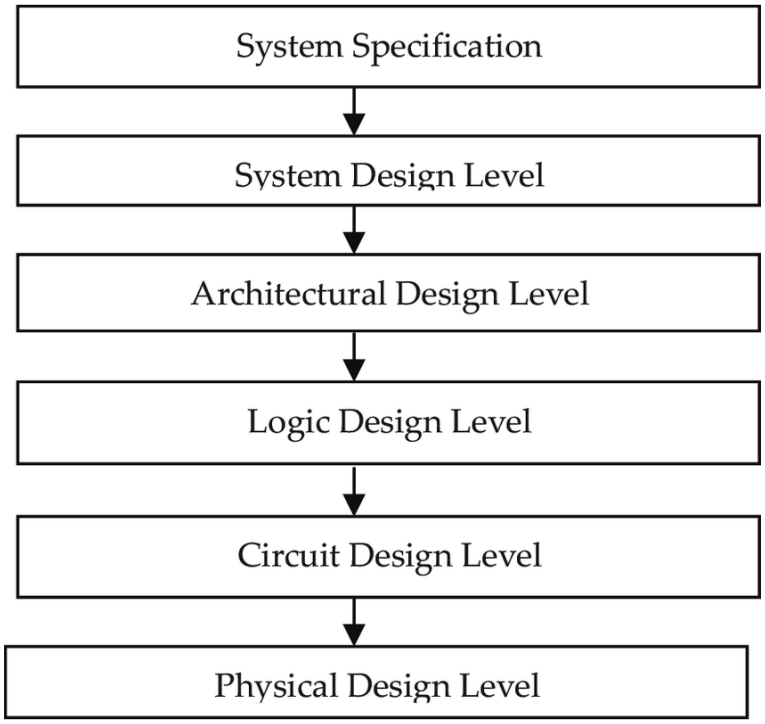

Figure 2.2: VLSI Design Implementation flow

The design flow of the VLSI chip has three stages namely behavioural, logic circuit, and layout implementation, [4] as shown in figure 2.2, At each stage of the process flow the verification is to be performed at the end of the stage. The development of the design flow is an iterative and repetitive process.

Behavioral representation is the first stage of the design flow and it is important to specify the functionality of the device and how it will communicate with the other blocks. The design architecture is planned based on the ASMD chart and the flow is implemented using the hardware description languages such as Verilog or VHDL to define the behaviour of the device. After the HDL codes are successfully implemented and simulations are verified , the functional blocks from the standard cell libraries are used to synthesize the behavioural representation of the design into logic circuit implementation. After the design is verified, the gate-level netlist is generated and it will be used for the layout development of the complete design. The final stage is the physical implementation of the design. The process starts with floor planning where the pads and routing areas of the chip are specified. Once the building blocks are arranged properly at the best locations to obtain the optimized design, this process is called placement. After the placement of the blocks is completed, the routing process is performed to interconnect the building blocks of the design.

The logic gates that are used to build the complete design are commonly referred to as standard cells. These are small logic blocks that implement commonly-used functionality such as AND-gates, OR-gates, and MUXs. These cells are duplicated and frequently used across the chips. Thus the same standard cell is used for 10000 instances that would have a significant waste in an area that could have been used for more logical implementation.

2.2 Integrated Circuits

An Integrated Circuit has a set of electronic devices in a semiconductor die material made of silicon. The chip contains a large number of transistors and logic gates that integrate to form smaller circuits, have faster performance, and are less expensive than the discrete electronic components. ICs are now used in all electronic equipment and have revolutionized the world of electronics.

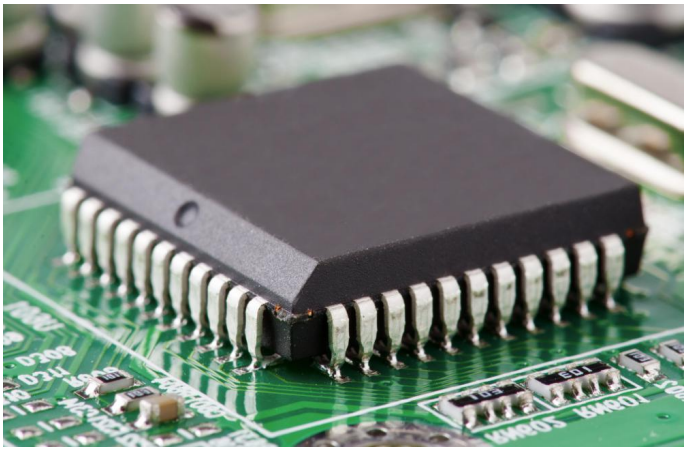

Figure 2.3: Integrated Circuit [22]

Integrated Circuits are made by technological advancement in metal oxide semiconductor device fabrication. ICs have two main advantages cost and performance. Cost is low because the chips and all the components are printed as a unit by photo-lithography process rather than constructing one transistor at a time. Performance is high because the IC's components switch quickly and consume less power because of their small size. The main disadvantage of ICs is the high cost of manual work and the time are used to design them. The initial cost is high and the ICs are commercially viable when the production of the chips in the volume is more.

Almost all modern IC chips are metal–oxide–semiconductor (MOS) integrated circuits, built from MOSFETs (metal–oxide–silicon field-effect transistors) made it possible to build high-density integrated circuits, [30]. In contrast to bipolar transistors which required several steps for the p–n junction isolation of transistors on a chip, MOSFETs required no such steps but could be easily isolated from each other. In Figure 2.4 The cross-section of the n-type Metal Oxide Semiconductor and p-type Metal Oxide Semiconductor is used to develop logic in an IC. For an analog designer, the layout of the design is seen as a topological view, where all the polygons are viewed as layers. The layout view of the transistor is shown in figure 2.5, thus the voltage is applied in the gate of the transistor, the path between the source and drain becomes conductive, similarly, for ground potential, the path will be resistive.

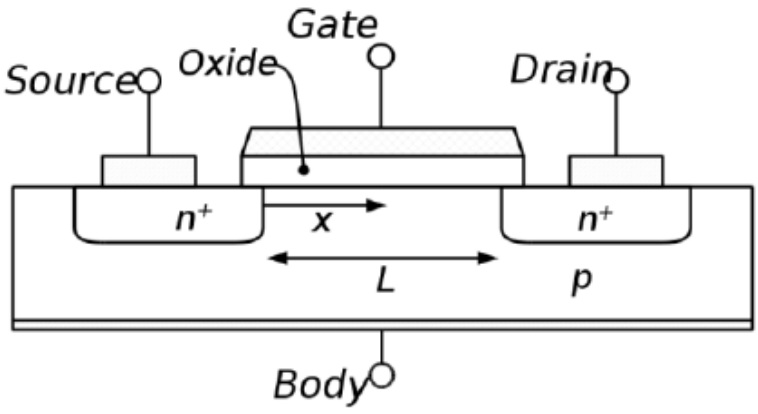

Figure 2.4: The Cross-Section view of NMOS [24]

Complementary Metal Oxide Semiconductor (CMOS) is used to design an IC, it is built using the NMOS and PMOS transistors. The source of the PMOS net is connected to the supply voltage and the source of the NMOS net is connected to the ground potential. The drain of the two transistor nets is connected to the output of the logic. Since one of the nets is completely open that gives the CMOS design for low static power consumption. This is the main reason for using CMOS design in many building blocks of the logic in Integrated Circuits.

Figure 2.6 shows the schematic view of the NAND gate, If the input A and

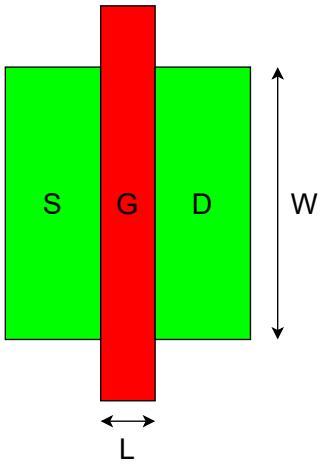

Figure 2.5: The Layout view of the transistor with Source(S), Drain(D),and Gate(G)

B is set to logic '1', the NMOS transistor will be conducting and the PMOS will be open. Thus the output voltage will be grounded due to the NMOS conducting path. If the input of A and B are set to logic '0', the PMOS transistor will conduct and the NMOS will now be open. Thus the output voltage will be the supply voltage with maximum conducting potential.

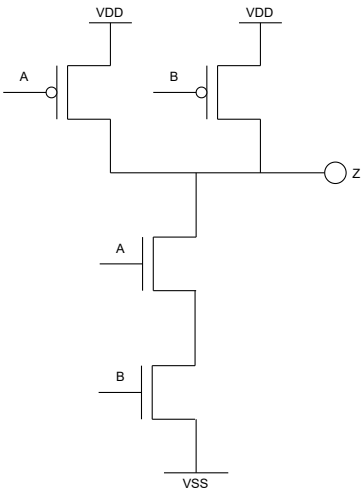

Figure 2.6: The Schematic of NAND logic gate

2.3 Read-Only Memory

Read-only Memory is a type of non-volatile memory used in all electronic devices. The Data stored in ROM cannot be modified once the memory is manufactured. These memories are used to store the software application which is rarely changed during the life of the system. It refers to memory that is hard-wired as a mask ROM integrated circuit (IC), which cannot be electronically changed after manufacture, [19]. Correction of errors, or updates to the software, require new devices to be manufactured and to replace the installed device. The Erasable programmable read-only memory(EPROM), electrically erasable programmable read-only memory(EEPROM), and flash memory can be erased and re-programmed. But usually, this can only be done at relatively slow speeds, may require special equipment to achieve, and is typically only possible a certain number of times.

ROM contains special internal electronic fuses that can be programmed for a specific interconnection pattern. The binary information stored in the chip is specified by the designer and then embedded in the unit at the time of manufacturing to form the required interconnection pattern. Once the pattern is established, it stays within the unit even when the power is turned off. So, it is a non-volatile memory as it holds the information even when the power is turned off, or you shut down your computer.

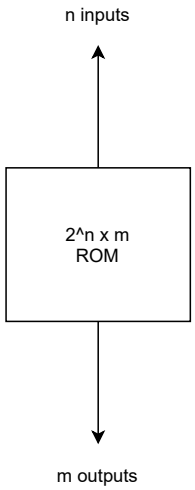

Figure 2.7: The Block Diagram ROM Cell

The information is added to a ROM in the form of bits by a process known as programming the ROM as bits are stored in the hardware configuration of the device. So, ROM is a Programmable Logic Device (PLD). As shown in figure 2.7, the block of ROM has 'n' input lines and 'm' output lines. Each bit combination of the input variables is known as an address. Each bit combination that comes out through output lines is called a word. The number of bits per word is equal to the number of output lines m. The address of a binary number refers to one of the addresses of n variables. So, the number of possible addresses with 'n' input

variables is 2n. An output word has a unique address, and as there are 2n distinct addresses in a ROM, there are 2n separate words in the ROM. The words on the output lines at a given time depend on the address value applied to the input lines.

The internal structure of the ROM consists of two basic components, decoder and OR gates. The Decoder circuit is used to encode the Binary Coded Decimal (BCD) to the decimal form. So, the input is in binary form, and the output is its decimal equivalent. All the OR gates present in the ROM will have outputs of the decoder as their output.

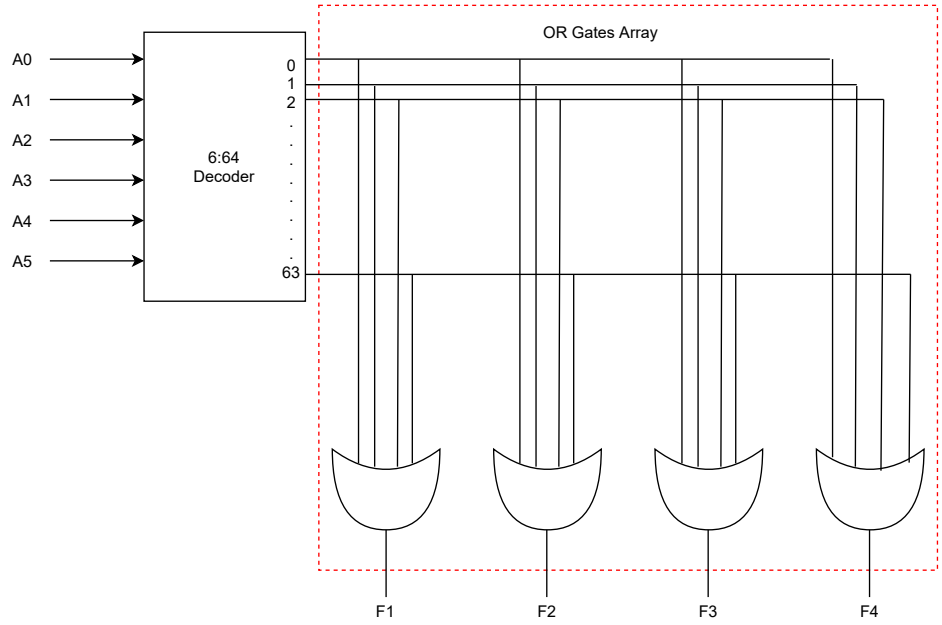

Figure 2.8: The Internal Structure Components of ROM cell

Let us take an example of 64 x 4 ROM as shown in figure 2.8. The ROM consists of 64 words of 4 bits each. Thus, there will be four output lines and each of 64 words available in the output lines s is determined from the six input lines as we need only six inputs for this ROM. For each address input, there is a unique selected word. For example, if the input address is 000000, word number 0 will be selected and applied to the output lines. If the input address is 111111, word number 63 is selected and applied to the output lines.

2.4 Static Random Access Memory and Standard Cell

Memories are used to load and store instructions and data during computations. There are three major processes involved in memory namely encoding, storage and retrieval. To form new memories, information must be changed into a usable form, which occurs through the process known as encoding. Once the information has been successfully encoded, it must be stored in memory for later use. The retrieval process allows us to bring stored memories into a usable form.

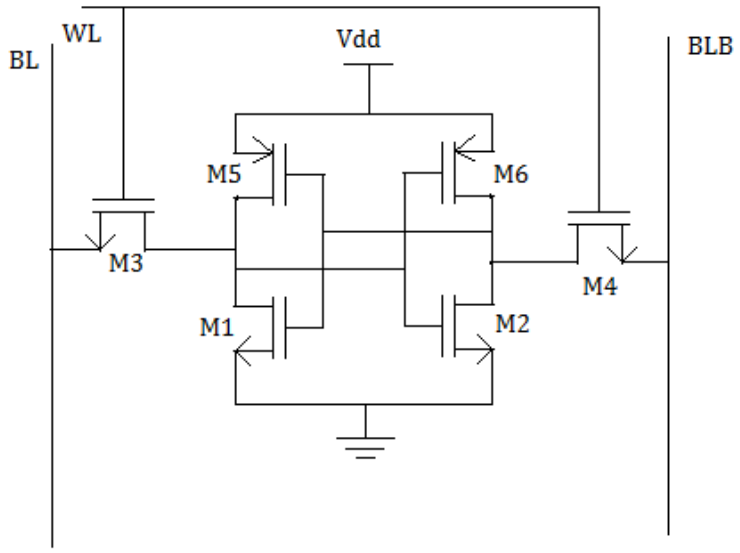

Figure 2.9: 6 Transistor SRAM bit cells

Static Random Access memory uses flip flops to store each bit of data in a memory location. These are volatile memories, where the data is lost once the power is turned off. The term static means the memory must be periodically refreshed. SRAM is faster and more expensive than DRAM, it is mainly used as cache and internal registers. SRAMs are easy to access the data from the memory location and the performance and reliability are good and power consumption is low when the memory is in an idle state. SRAM requires more transistors to design when compared to DRAM, so it occupies more area and a very expensive and has high power consumption during the read and write operation. The power consumption mainly varies on how often the SRAM, [18]. The 6 transistor bit cell version of the SRAM cell is shown in figure 2.9.

The 6 transistor SRAM bitcell has three input and output lines namely word-line(WL), bit-line(BL), and inverse bit-line(BLB). These lines are used to read and write the data into the memory of the bitcell. The inverter in between is used to store the data and retrieve the data based on the inputs of the lines. The read operation starts with the fully precharged bit-lines to full potential. The WL is then made logic high and the value in the nodes Q and inverse Q will be connected to the bit lines through the transistors M5 and M6. There will be a small change in the voltage of the bit lines, which can be detected as read value is seen in the output of the peripheral circuits. In the write operation, the input value to be written in the memory is given in bit lines and the inverse value is given to the inverse bit line. The transistors M5 and M6 prevent the value from being written to the data storage Q and inverse Q nodes. WL is then activated high and the values on both bit-lines are latched into the data storage nodes and a successful write has been performed.

The SRAM bit cell shown in figure 2.9 can store only one-bit data at a time. Therefore several bit cells are combined as a bitcell array matrix configuration to increase the storage capacity of the memory as shown in figure 2.10. The column and row decoders are used to store and retrieve the data from the memory using the BL and inverse BL lines. The read and write operation is performed on each row of the bitcell array matrix. To activate each row of the bit cell array using the WL lines, the address decoder is used. Thus the memory selects one row at a time to avoid conflicts between the memory bank. The Layout design of the SRAM memory blocks has different components like GPIO block, timing block, decoder block, MCU block, D Flip-Flop Block, and encoder block.

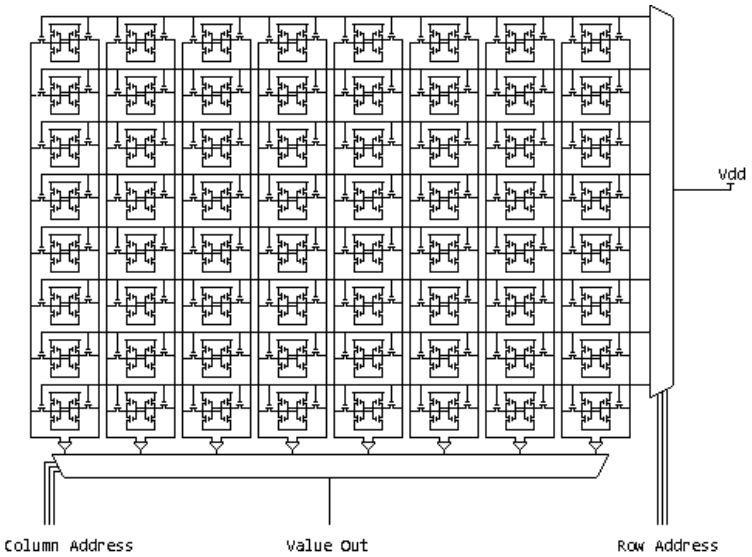

Figure 2.10: Memory array matrix architecture

A Standard cell is made up of a group of transistors and these are interconnected to form a functional building block such as AND, OR, XOR, inverters, flip-flop, etc. The cell boolean logic is called the logical view of the function and the behavior is verified based on the truth table or boolean equation. The design of the standard cell is developed at the transistor level in the form of a netlist or schematic view. The logical and the netlist view are used only to perform the simulation of the cell to check the behavior of the block using the SPICE EDA tool. The physical representation of the cell needs to be developed and it is represented as a layout view. The NAND Standard cell layout and schematic are shown in figure 2.11.

For the manufacturing process, the layout of the standard cell is an important aspect of the design flow. The layout view has base layers, which correspond to the different structures of the transistor devices, and interconnect wiring layers and via layers, which join together the terminals of the transistor formations, [3]. The interconnecting wires have a specific layer number and the corresponding via layers are used to perform the connection between the layers. The abstract view contains much less information than the layout view, it is present in the Layout Extraction Format(LEF) file. After the layout is created the design is validated using the Design Rule Check(DRC) and Layout vs Schematic(LVS) to check whether the design meets the foundry requirements. Then the Parasitic Extraction is performed to generate the parasitic netlist which contains the parasitic properties of the layout. The PEX netlist can be simulated again using the SPICE tool to check the power, timing, and noise model of the cell.

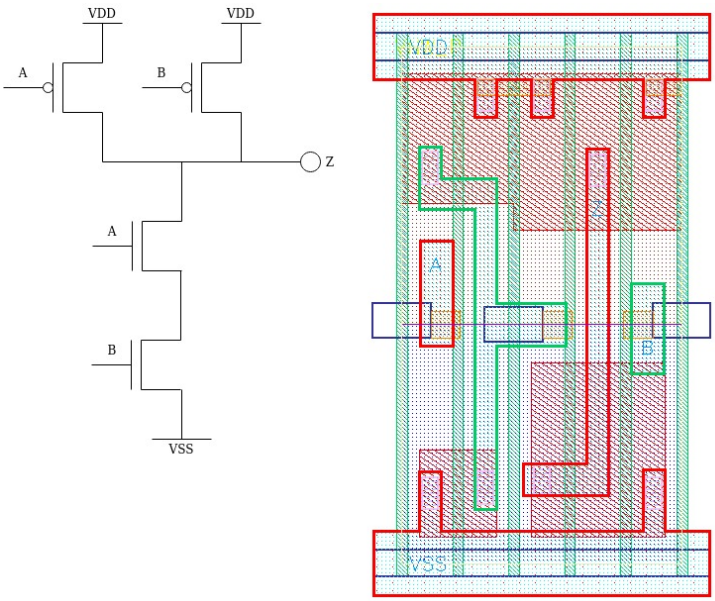

Figure 2.11: Standard Cell NAND Gate Schematic and Layout

The Standard cell library contains a collection of low-level electronic logic functions that are realized of fixed height and variable width full custom cells. The key advantage of these cells in the library is to process the complex layout with ease of automatic layout generation. The standard cell library contains the main components of the library database that have schematic, symbol, and layout view. This information is captured in LEF format, which can be used to automate the place and route EDA tools. This library contains multiple implementations of the same cell with different area and performance. This gives the designer total freedom to perform implementation trade-offs. This standard cell description is called a technology library. EDA tools use the technology libraries to automate synthesis, placement, and routing of a digital ASIC, [13]. The technology library is developed and provided by the foundry operator.

2.5 Analog Layout Design: Bottleneck in the Design Flow

The Analog Layout design can be modified automatically using different optimization algorithm. In digital design, the synthesis flow is automized using such an optimization algorithm. Similarly, the place and route are successfully employed by placing millions of transistors in a single chip. Encouraged by this success, EDA keeps a resolute focus on trying to adopt such approaches in the analog domain. The layout design is the step of the analog design flow with the least support by commercially available tools.

Due to the rejection of many existing automation approaches, analog layouts in practice are still designed by human experts in the EDA tools and largely in a manual fashion, putting up with the downside that the design productivity is significantly lower than in the digital domain. This can be addressed in two regards the effort for creating an analog layout is much higher than for a digital layout and the number of design components is usually smaller by several orders of magnitude.

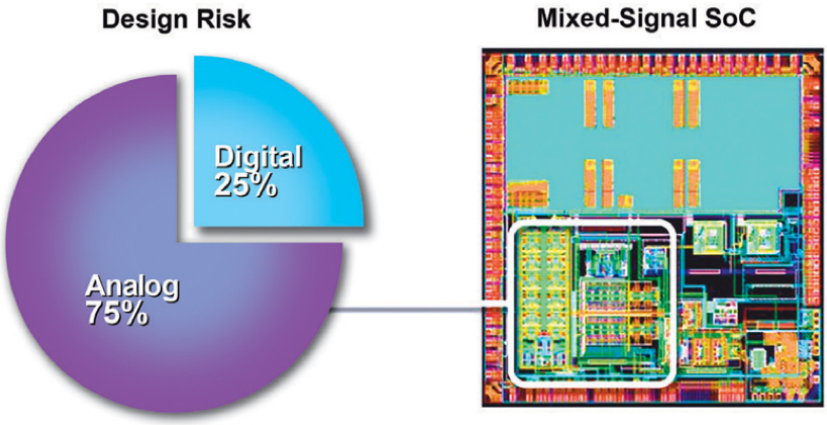

Figure 2.12: The Mixed Signal SOC [31]

The optimization algorithms are effectively employed for digital design synthesis, but could not find as much acceptance in the analog domain even though the conceded lack of automation involves a large economic growth. The overall design flow is even more worrisome when considering that time-to-market continually decreases due to shortening product life cycles. Creating a layout is a very complex problem, But in terms of this complexity, one should clearly distinguish between the discrete-valued nature of the digital domain and the continuous-valued nature of the analog domain.

2.5.1 Full Custom Analog Design

The Analog section represents the indispensable interface of an IC to its continuous-valued environment and also serves as the subsystem for powering the chip. In terms of qualitative complexity, maintaining the integrity of analog signal transmission for an optimal layout that utilizes the entire spectrum and variety of all available degrees of freedom. For the most part, this in turn opposes modification-based design automation because an abstraction of the design problem and a lack of adequate heuristics cause the resulting layout solutions to be insufficient for practical application. So, the loss of layout quality that is condoned in digital design is precisely what cannot be tolerated in the analog domain.

The need to exploit all degrees of freedom defies the use of different algorithms because these require a reduction of the degrees of freedom. For that reason, analog layout design is still done in a highly manual way and relies heavily on the knowledge, experience, skills, and inventiveness of human experts. So, the lack of automation gap is tolerated because of two reasons, the layout quality does not permit any trade-offs. Thus, opposite to the digital domain. The demand for automation is overruled by the layout quality requirements and these are intimately tied to the so-called design constraints [33].

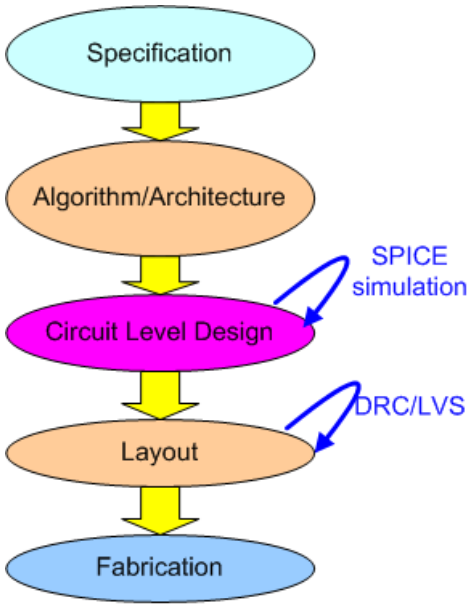

Figure 2.13: The Full Custom design flow [32]

2.5.2 Automation-based Analog Layout Design

People has proposed different approaches with varying degrees of generality for different usage scenarios. However, introducing design-dependent strategies and prior knowledge in the automation flow often benefit the flow effectiveness in the cost of jeopardizing the generality of flow. Automation-based layout design is to formulate the layout generation as a constrained optimal problem and tends to model the layout quality as the objectives.

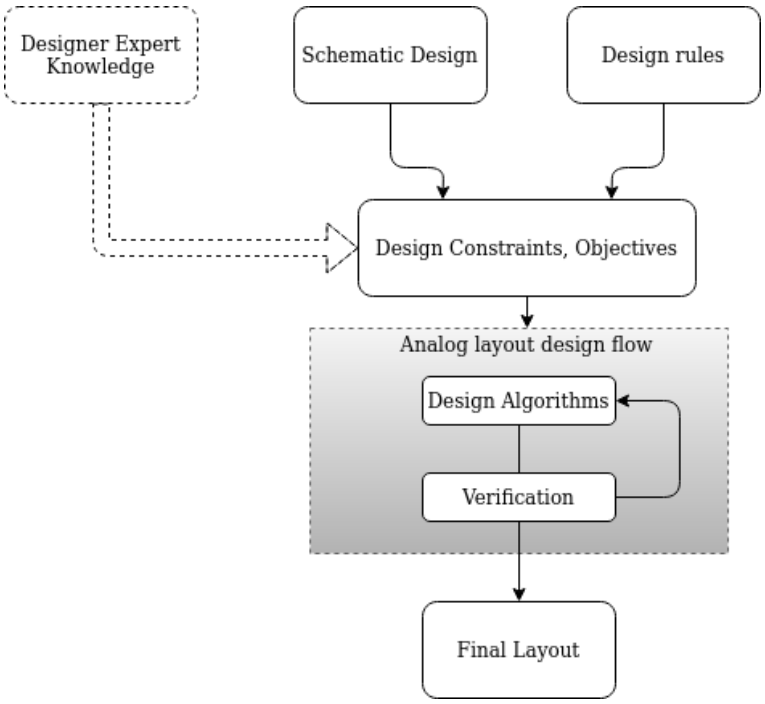

Figure 2.14: The Automation Based Design Flow Model

Like digital EDA tools, Automation-based analog layout generation often separates the process into several stages for divide-and-conquer. A common practice is to have three stages, module generation, placement, and routing. The module generator generates the layout of building blocks in a parameterized manner. The placement stage then places the layout from the module generator. In the end, the routing stage connects the nets through metal wires and vias. As shown in figure 2.14, the analog design automation flow depends on different parameters. The Schematic design and the design rule specification file is an important parameter to generate an optimal layout. Then based on the design constraints and objective from the analog layout designer, we will start the process of modifying the layout. The design process is an iterative approach, where the design algorithm is developed and then it is verified for any DRC/LVS issues. After the verification, if the generator report contains any issues, then the algorithm is modified further and verified again. This process continues till all the DRC

and LVS issues are resolved and the generator layout is DRC and LVS clean [34].

A significant challenge in automatic analog layout design is the lack of an effective method to model the layout effects on circuit performance. Figure 2.15 shows a potential flow of performance prediction-assisted automated layout generation flow. A performance modeling can give feedback to automatic layout generation and guide the tool to generate high-performance layouts.

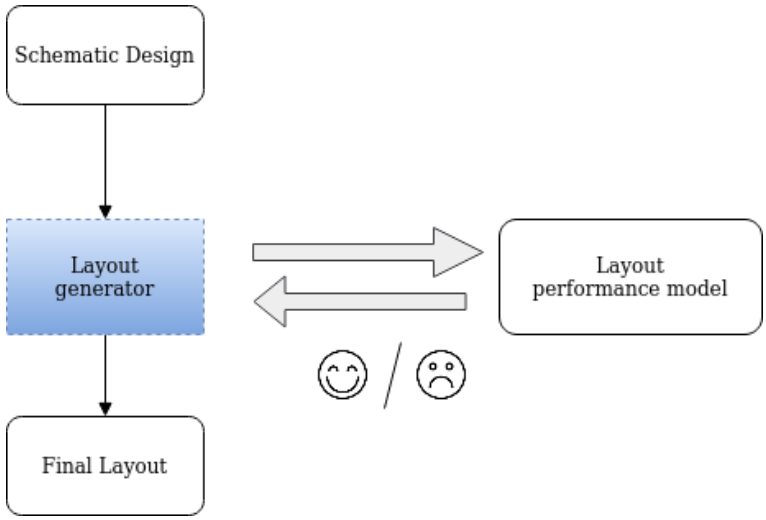

Figure 2.15: The Layout Performance Evaluation model

In the analog layout automation, several challenges are remaining unresolved. First, there lacks a principal method in formulating the design problems. The existing geometric constraints are distilled from manual layout heuristics, which are design-dependent and technology-dependent. Second, it takes efforts to label the correct constraints by hand. Although there are existing works for detecting symmetry constraints automatically, the scalability and generality of these algorithms are concerned. Layout quality is often evaluated through post-layout simulation. However, as the post-layout simulation requires fully functional layouts, the extensive efforts on fundamental building blocks such as design rule handling and module generator make individual research focus on the abstract level problem instead of real circuit performance.

Chapter 3
Layout Modification and Optimization

In this chapter, we discuss the various methods to optimize and modify the Analog layout based on the different design metrics like power, performance, and area. The automation of the modification in the layout is implemented using the Python Programming language and the module used is GDSPY, [16].

As in the EDA world, since automation of the layout modification is a bigger task, we deploy a divide and conquer strategy and try to divide the complete project into different tasks that would be easier to solve the problems. The first task of my thesis project is to modify and optimize the different digital elements like standard cells and memory blocks. we will look into the algorithm used to modify the length, width, and pitch of the polygons in the layout. The automation implementation starts with the main python script file that reads the input from the user. The user input data and the description of the input data are explained below, the user interface was developed using the Tkinter module to showcase a GUI that makes the user work on it comfortably as shown in figure 3.1.

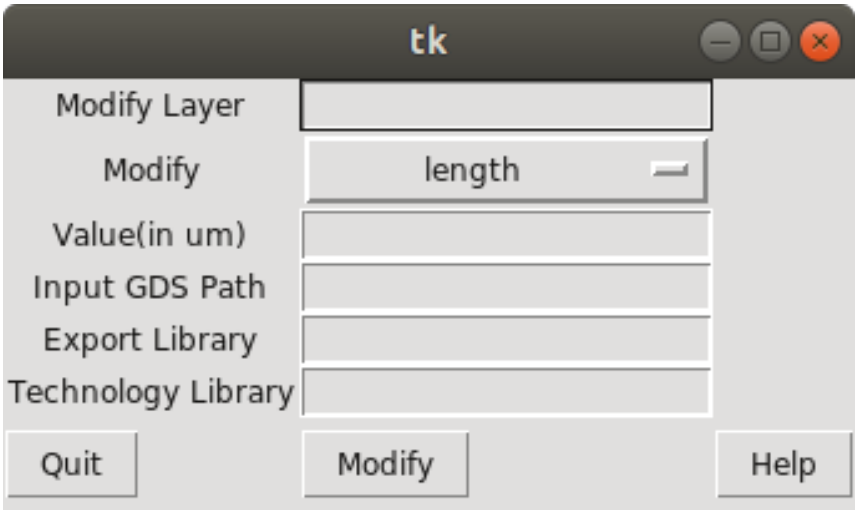

Figure 3.1: The Layout Modification input GUI

Modify_layer : Specify which polygon you want to modify in the layout.
Modify : Select from the drop-down menu what do you want to modify length/pitch/width.
Value : Specify how much you want to modify the polygon. Specify the value on the micrometer scale.
Input GDS path : Specify the input GDS file path(.gds is GDSII Stream file format) which you want to modify.
Export Library : Specify the export library name that will be loaded in the cadence library manager.
Technology Library : Specify the technology library name of the input GDS file.

After filling in all the details in the GUI shown in figure 3.1, there will be three options to proceed with. The Quit button is used to close the GUI application, the Help button is used to provide help regarding the inputs to be filled properly for the user. The Modify button is used to modify the input GDS file based on which parameter you would like to modify and the value of how much you want to modify. The input GDS is exported from the cadence tool and the modified layout is written to the output GDS path file and it is imported to the cadence tool and the design is checked for LVS and DRC clean.

3.1 Length Modification

The length of the polygons of the building blocks is increased based on the design requirements. The poly-silicon (Gate Layer) of the technology node is important to increase and decrease to improve the design metrics in the design. For a simple block of the design, increasing the length is easy to do manually, but for complex logic blocks increasing the length of the polygon is very difficult and it is time-consuming. So the automation of the task will increase productivity and get accurate results than manual work. The Input analog Layout of the Standard cell for which the modification has to be performed is shown in figure 3.2.

First, the exported input GDS file is flattened and then it is read using the GDSPY module. The GDSPY module has an inbuilt API function that can process the cells, polygons, and elements. To increase the length of the polygons we check the coordinates of the polygon boundary. If the polygon boundary x coordinate is greater than the bottom left corner x coordinate of the current polygon then moves all the coordinates of the polygons by the value that needs to be increased. First, we will append the list of polygons that needs to be increased and based on each list, the other polygons are shifted by the user-defined value. we need to create a database that can handle the layer map file of the technology nodes and it contains the layer name, layer id and datatype of the layers in the technology library to process the corresponding layers in the cell. The steps to implement the algorithm are explained below. To develop the optimized algorithm, we used the heuristic algorithm approach by implemented trial and error logic for each polygons in the layout as explained in chapter 2.

Step 1: Read all the elements, polygons from the cell and initialize them to

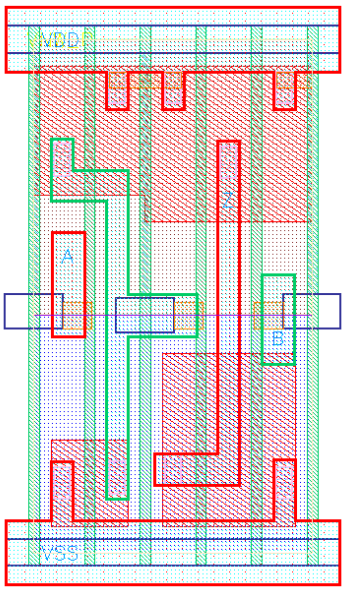

Figure 3.2: The Input Standard Cell Layout

the list. Loop through the polygon list and initialize the boundary of the polygon to be modified in the poly boundary list. And check for any duplicates in the poly boundary list and remove them.

Step 2: Loop through the poly boundary list and modify the polygons by increasing the boundary points of the polygons as the value specified by the user.

Step 3: Loop through the polygon list and update the boundary of other polygons in the update boundary list and modify the points corresponding to the poly boundary list. Here we need to check for different conditions. we need to have a separate algorithm for metal, via and contacts, and other layers in the design.

Step 4: The First condition is to check for all the layers other than via and metals, then check if the poly boundary x coordinates are less than the update boundary x coordinates then modify the update boundary list based on the user-defined modify value. Otherwise, move to the next condition.

Step 5: Next, check for the polygons which are metal, if the polygon boundary is less than the update boundary and if there is metal overlap, then the update boundary is shifted by the user-defined value. Else check for other conditions.

Step 6: Now we check for polygons that are via and similarly if the polygon boundary is less than the update boundary and it has via overlap then modify the update boundary. If all three conditions are not satisfied, then don't do any changes to the update polygon boundary. We need to check metal and via in our design because the metal and contacts have different DRC rules for different technology nodes.

Step 7: After performing the above steps for all the polygons in the cell, we create a new cell with the updated boundary polygons and add the cell to the library. Then we need to write the output GDS file with the modified layers and

import the GDS using the stream in command in cadence virtuoso. The flowchart of the algorithm is shown in figure 3.3.

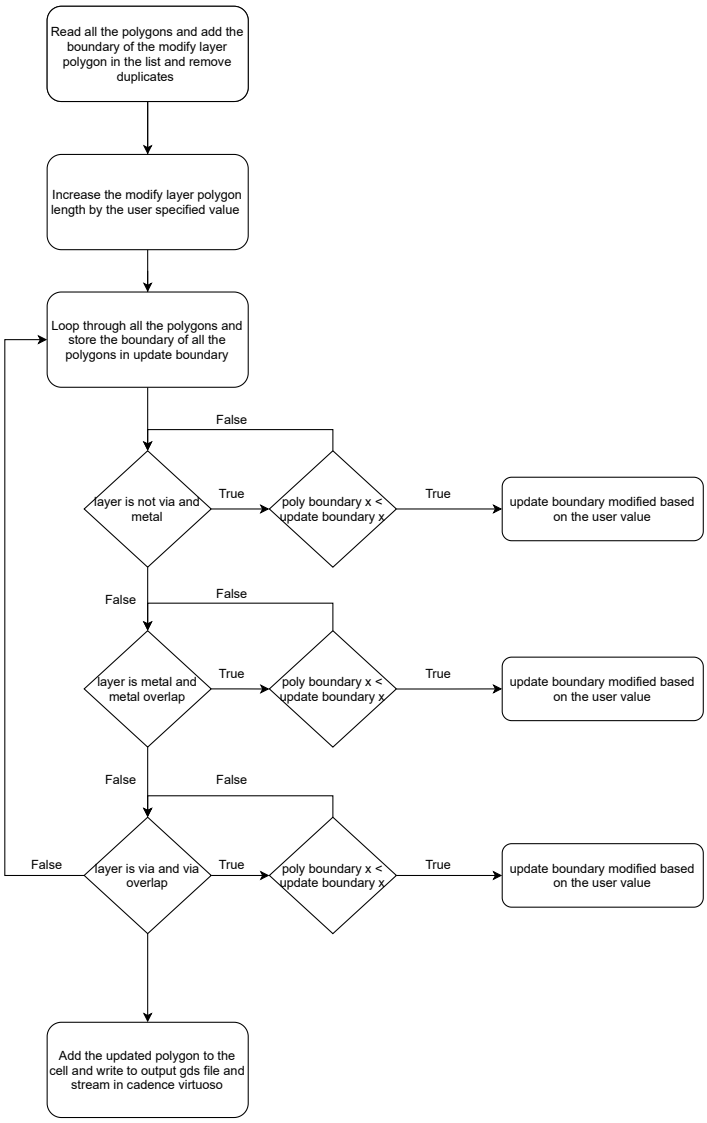

Figure 3.3: The Algorithm flow of Length Modification

The Output of the modified layout is shown in figure 3.4. The Modified layout is verified using the DRC tool to check for violations in the design. After checking for any DRC in the design for the particular technology node, then we check the LVS of the design and if both are clean.

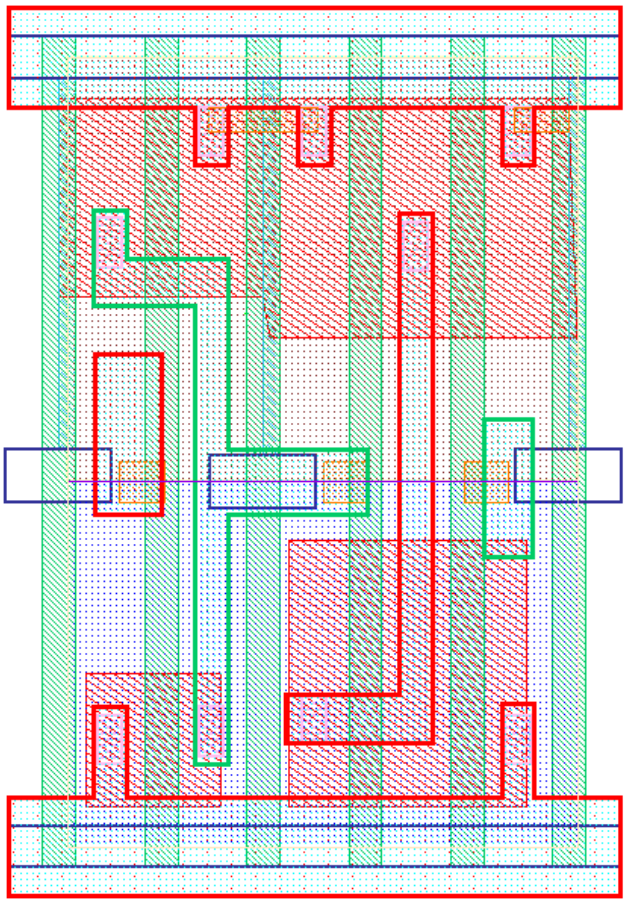

Figure 3.4: The Output Length Modified Layout

3.2 Pitch Modification

The Pitch of the polygons is increased to modify the layout for different process nodes. Pitch is the distance between the two poly-silicon polygons in the cell. For different lengths of the poly-silicon, the pitch distance between them varies. If you would like to change the pitch between the poly-silicon manually, it is a very difficult and tedious task. Thus automating it will ease the job of the designer and finish the work in minutes. The input for modifying the pitch is the same as the input used for length modification as shown in figure 3.1.

Same as the length modification, the input GDS file is flattened and then processed by the GDSPY Module in python. To increase the pitch distance between the two polygons, instead of checking the bottom left corner, we will be checking the bottom right corner and the same algorithm is implemented to increase the pitch distance. The bottom right corner is checked because we need to increase the distance between the two poly layers, so concerning the right-side x coordinates, the remaining polygons are shifted by the pitch distance. For metal and vias, the separate operation is implemented as it should be placed in the proper position to avoid any DRC violations. The pitch between two polygons is increased to change the complete design from one process node to another process node technology. At first, the Dijkstra algorithm was used to find the shortest distance between the two polysilicon and modify the pitch distance between them as discussed in chapter 2. And tired with different optimization to calculate the shortest path, but it was not feasible for the design. So I developed an algorithm that was suitable for the polygon shapes in the layout.

Step 1: Read all the elements, polygons from the cell and initialize them to the list. Loop through the polygon list and initialize the boundary of the polygon to be modified in the poly boundary list. And check for any duplicates in the poly boundary list and remove them.

Step 2: Loop through the polygon list and update the boundary of other polygons in the update boundary list and modify the points corresponding to the poly boundary list. Here we need to check for different conditions. we need to have a separate algorithm for metal, via and contacts, and other layers in the design.

Step 3: Layer modify wrapper file contains all the function modules required to process the polygons in the cell. This file has all the implemented functions that need for the design of the algorithm. The first condition is to check for all the layers other than via and metals, then check if the poly boundary x coordinates are less than the update boundary x coordinates then modify the update boundary list based on the user-defined modify value. Otherwise, move to the next condition.

Step 4: Next, check for the polygons which are metal, if the polygon boundary is less than the update boundary and if there is metal overlap and check for the move center metal function, then the update boundary is shifted by the user-defined value. Else check for other conditions.

Step 5: Now we check for polygons which are via and similarly if the polygon boundary is less the update boundary and it has via overlap then modify the update boundary by half of the user predefined value. If all three conditions are

not satisfied, then don't do any changes to the update polygon boundary.

Step 6: After performing all the above steps for all the polygons in the cell, we create a new cell with the updated boundary polygons and add the cell to the library. Then we need to write the output GDS file with the modified layers and import the GDS using the stream in command in cadence virtuoso. The flowchart of the algorithm is shown in figure 3.5.

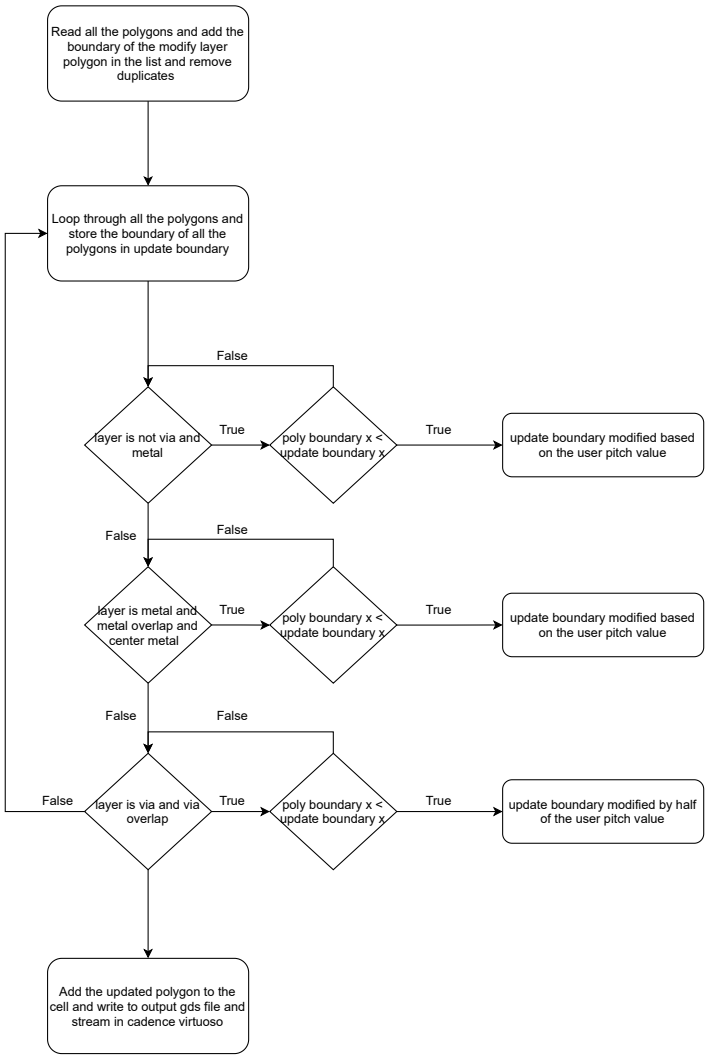

Figure 3.5: The Algorithm flow of Pitch Modification

Similar to the length modification, the Output of the modified layout is shown in figure 3.6. The Modified layout is verified using the DRC tool in the cadence to check for violations in the design. Then we check the LVS of the design and if both are clean.

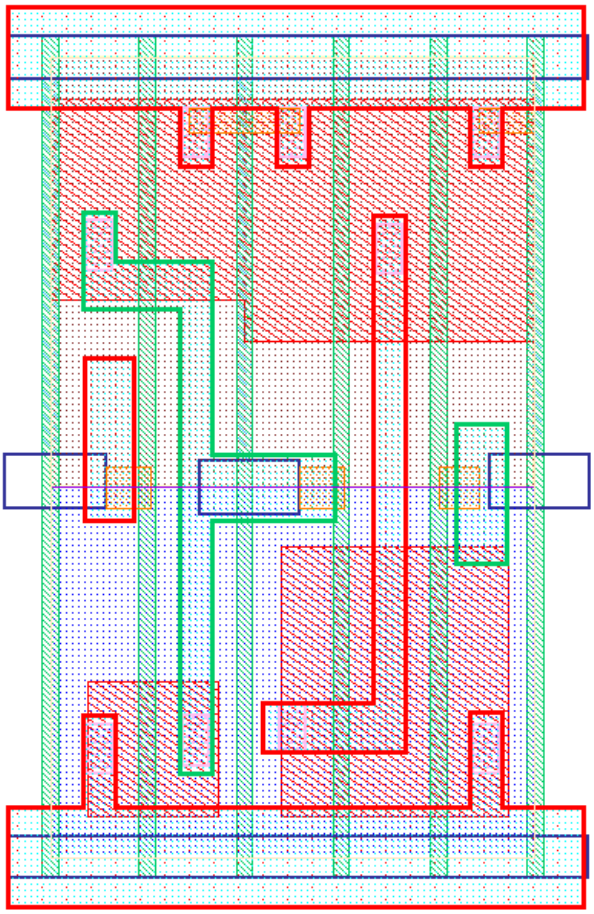

Figure 3.6: The Output of Pitch Modified Layout

3.3 Width Modification

By increasing the width of the polygons in the analog layout design, it makes the layout clean and neat. In some cases for large memory designs, each block should be interconnected in the top design so it requires an increase in the width of the polygons in some situations. The input for modifying the width is the same as the above as shown in figure 3.1.

The input GDS file is flattened to a single cell and then processed. To increase the width of the polygons we check the coordinates of the polygon boundary. If the polygon boundary y coordinate is greater than the bottom left corner y coordinate of the current polygon then moves all the coordinates of the polygons by the value that needs to be increased. First, we will append the list of polygons that needs to be increased and based on each list, the other polygons are shifted by the user-defined value.

Step 1: Read all the elements, polygons from the cell and initialize them to the list. Loop through the polygon list and initialize the boundary of the polygon to be modified in the poly boundary list. And check for any duplicates in the poly boundary list and remove them.

Step 2: Loop through the poly boundary list and modify the polygons by increasing the boundary points of the polygons as the value specified by the user.

Step 3: Loop through the polygon list and update the boundary of other polygons in the update boundary list and modify the points corresponding to the poly boundary list.

Step 4: Check for all the polygons in the cell that needs to be updated based on the poly boundary list and modify the update boundary list as per the user-defined value.

Step 5: After performing all the above steps for all the polygons in the cell, we create a new cell with the updated boundary polygons and add the cell to the library. Then we need to write the output GDS file with the modified layers and import the GDS using the stream in command in cadence virtuoso. The flowchart of the algorithm is shown in figure 3.7.

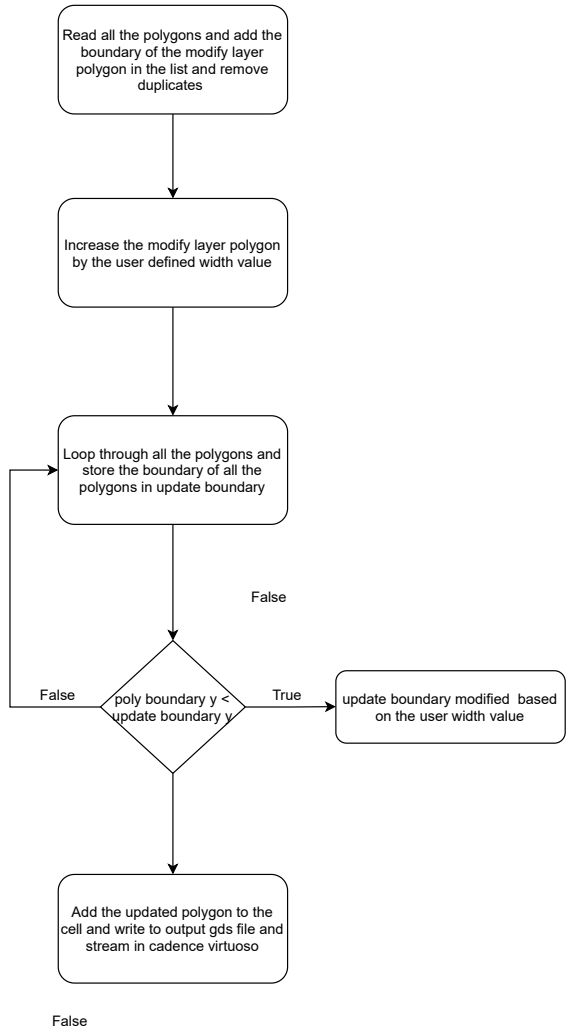

Figure 3.7: The Algorithm flow of Width Modification

The output modified layout is shown in figure 3.8. Similarly, the modified layout is verified for the DRC and LVS clean layout using the cadence tool.

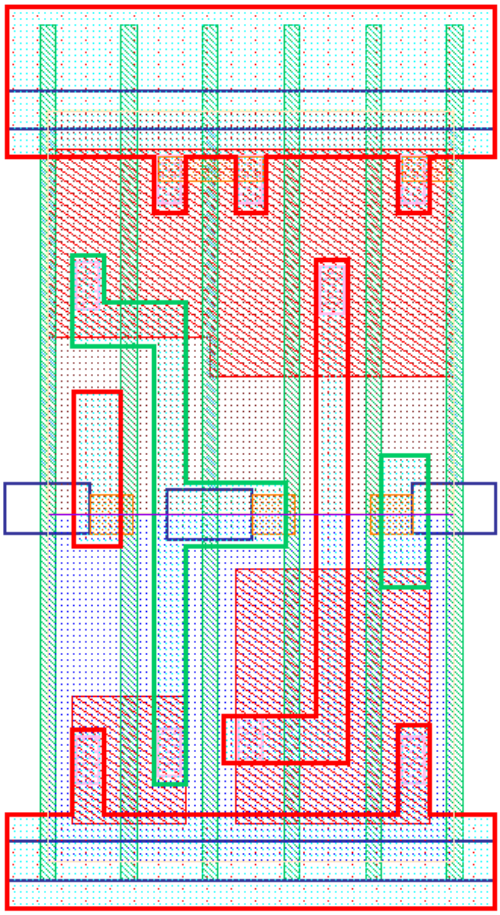

Figure 3.8: The Output of Width Modified Layout

Chapter 4
Layout Porting

In this chapter, we will look into the concept of porting the layout from one technology node to another and how it is implemented efficiently. Porting the analog layout of any design building block is time-consuming and needs a lot of manual work. Thus automating the task will ease the job of the designer and it can finish in minutes. The Automation of the layout porting is implemented using Python programming and the pitch, length, and width modification of the layout algorithm is also used in this design as we discussed in the previous chapter.

Similar to the previous chapter, the task is divided into sub-tasks and each part is implemented separately and tested using the python test cases. We will look into the different files that are required to implement this task. The main.py file is the main python file that receives the input from the user. Based on the user design requirements, the other files are called in the main file and then processed. There are different technology nodes in the ASIC development. The input data from the user is received from the Tkinter GUI as shown in figure 4.1 and process the data in the main python file. The input data and the description of the input are explained below.

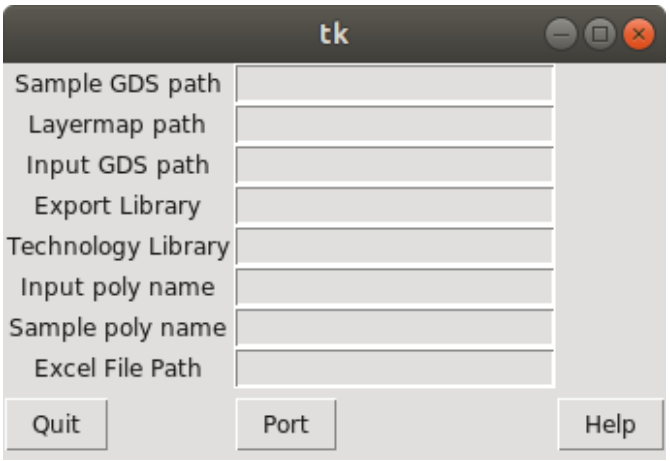

Figure 4.1: The Input Data GUI Application

Sample GDS path : Specify the sample GDS file to the technology node that you want to port the layout.
Layermap path : Specify the path of the layermap file of the sample GDS.
Input GDS path : Specify the path of the input GDS file(.gds is GDSII Stream file format) which you want to port.
Export Library : Specify the export library name that will be imported in cadence library manager.
Technology Library : Specify the technology library name of the input GDS file.
Input poly name : Specify the poly-silicon(gate) layer name of the input GDS.
Sample poly name : Specify the poly-silicon(gate) layer name of the sample GDS.
Excel file path : Specify the path of the layer name mapping excel file that contains all the polygon layer name pairs of the input and the output layout.

After filling in all the detail in the GUI as shown in figure 4.1, the Help button in the GUI shows all the input data information and what files need to be included. All the input in the GUI is important and it needs to fill properly without any mistake and then we need to click the Port button to start the porting process. Once the porting is completed, the output GDS file is imported to the cadence virtuoso layout view and verified for any DRC and LVS issues. The design flow of the task is explained in figure 4.2.

The first task of the design is to read all the input data from the user using the GUI. Then the output technology node layermap file is read and all the layer names, id, and datatype are processed and different function are created to get the layer name, get the layer id and datatype based on the other parameters. The third task is to create a database for both the input and sample layout, it contains all the length, width, and pitch information for all the layers in the technology. After the database creation, the input layer name mapping excel file is read and created pairs based on the layers in different technologies. This file contains all the layer names corresponding to different technologies. After processing all the inputs, the length, width, and pitch of the layout is modified for each layer in the input layout. And finally, the design is ported to the new technology node and the generated layout is imported to the cadence virtuoso layout view.

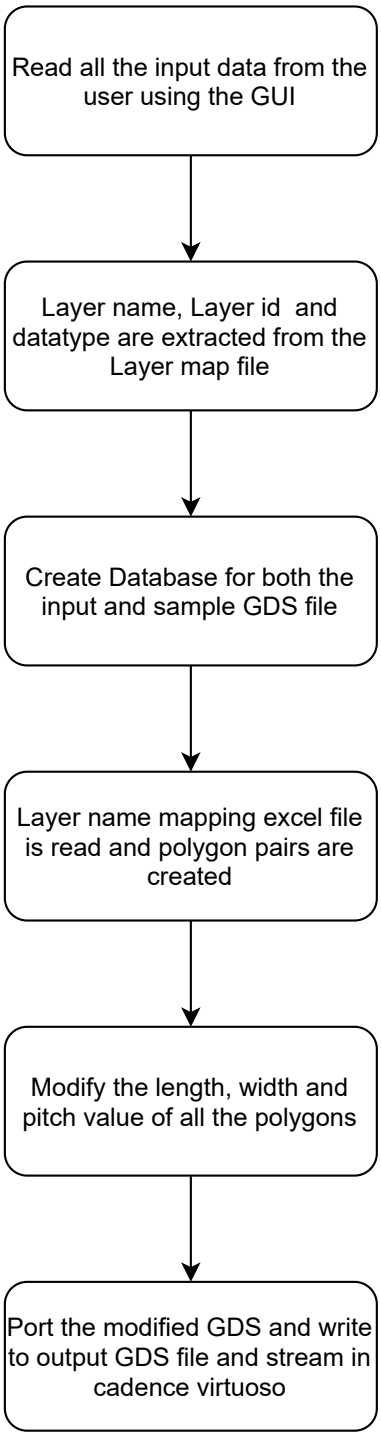

Figure 4.2: The Design Flow of Layout Porting

4.1 Database Creation

The porting of the analog layout is implemented in a generic manner, by storing all the data in the database and processing all the implementation steps by accessing the information from the database. Porting of the design contains all the polygons length, width and pitch distance between the polysilicon, which are changed to generate the layout in different technology node as each technology nodes has its own rules and specification. Hence based on the sample layout of the new technology node the database is created and it contains all the length, width, and pitch of the polygons in the layout. Similarly for the input layout of the previous technology is also created and the information is accessed and processed. The sample layout is shown in figure 4.3 and the input layout is shown in figure 4.4.

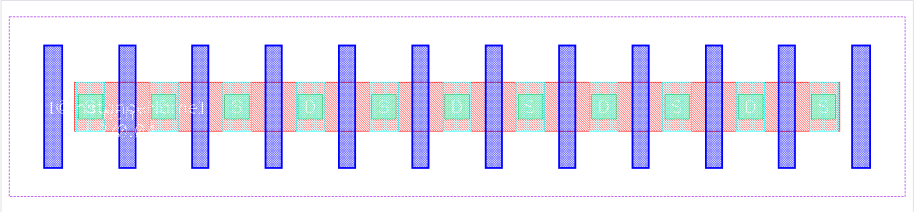

Figure 4.3: The Sample layout of Porting Technology

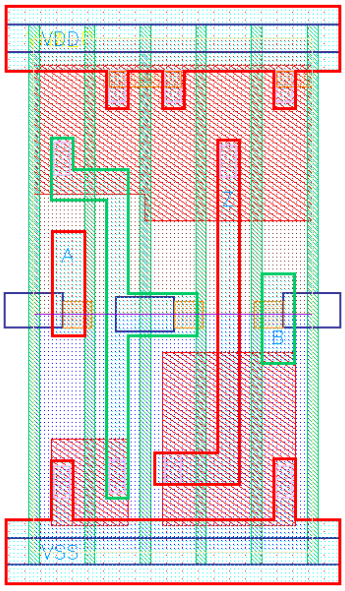

Figure 4.4: The Input layout to be Ported

Step 1: First the sample GDS file is read using the GDSPY python module and the cell and their corresponding polygons are extracted from the library to process them.

Step 2: The function created in the wrapper file is used to get the layer name based on the id and datatype and stored in the dictionary along with the length, width, and pitch value for all the polygons.

Step 3: Similarly for the input GDS file, all the layer name, pitch, length, and width value is extracted and stored in the separate dictionary.

Step 4: After storing all the values in the dictionary, each pair of layer names are accessed from the database and the value that needs to be changed by length and width is also updated in the dictionary. Then the pitch value is also calculated and return as the output.

The output of the database created is shown in table 4.1.

Input Polygons	Sample Polygons	Length	Width
Poly-Silicon	Poly-Silicon	1	-35
Oxide	Oxide	83	-2
Via	Via	2.5	0
Metal	Metal	0.5	-2.5
N-Substrate	N-Substrate	84	4
Boundary	Boundary	85	-25

Table 4.1: Length and Width value to be modified (Size are normalized).

Input Polygons	Length	Width	Sample Polygons	Length	Width	Length to be modified	Width to be modified
Poly-Silicon	1	50	Poly-Silicon	2	15	1	-35
Oxide	7	8	Oxide	90	6	83	-2
Via	1.5	3	Via	4	3	2.5	0
Metal	3	8	Metal	3.5	5.5	0.5	-2.5
N-Substrate	26	22	N-Substrate	110	26	84	4
Boundary	25	45	Boundary	110	20	85	-25

Figure 4.5: The Output of the Database Creation (Size are normalized)

4.2 Porting the Layout

The porting of the digital design elements are implemented to reduce the time to market for different technology node and deliver the chips faster based on the design requirements. Semiconductor companies must act faster to create best-in-class integrated circuits. To get greater performance, analog designers are trained to take advantage of changes in semiconductor technology with innovative schematic designs. To expand product offerings and control profit margins, companies need to assess how their schematic circuits will perform in diverse technologies and cost-

effective fabrication facilities.

After the database creation and initialization, all the pairs of the technology are created, the length, width, and pitch of all the polygon layers are modified using the same algorithm as explained in the previous chapter. After modifying the length and width of the layout, the modified layout gds file is updated and the output of the design is shown in figure 4.6. After the modification of the length and width of all the polygons in the input layout, then the pitch of the design is modified based on the pitch value from the database dictionary, and the modified GDS file is generated with the updated length, width, and pitch value. The layout of the modified GDS file in the cadence tool is shown in figure 4.7.

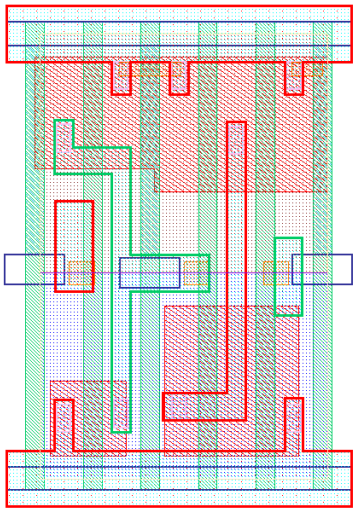

Figure 4.6: The Length and Width Modified Layout

After modification of the length, width, and pitch value based on the dictionary that contains all the layer information. The porting of the design will be started and the steps involved in the process are explained below.

Step 1: Read all the elements, polygons from the cell and initialize them to the list. Loop through the polygon list and initialize the boundary of the polygon to be modified in the poly boundary list.

Step 2: Loop through the polygon list and another loop for accessing the pairs list of the polygons in different technology.

Step 3: Check for polygon matching from the polygon list and the pairs list and get the layer id and datatype from the function in the wrapper that returns the layer id and datatype based on the layer name and assign the new id to the polygons.

Step 4: After performing all the above steps for all the polygons in the cell, we create a new cell with the updated polygons and add the cell to the library. Then we need to write the output GDS file with the modified layers and import the

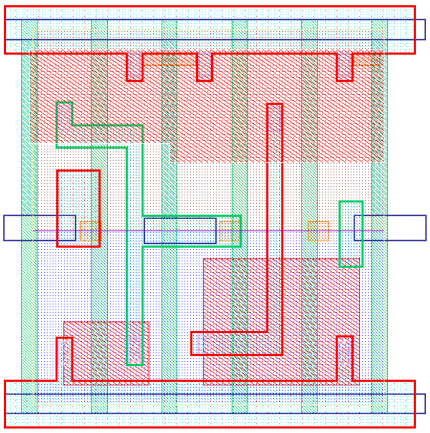

Figure 4.7: The Modified Layout after Database creation

GDS using the stream in command in cadence virtuoso layout view. The output of the generated GDS is shown in figure 4.8.

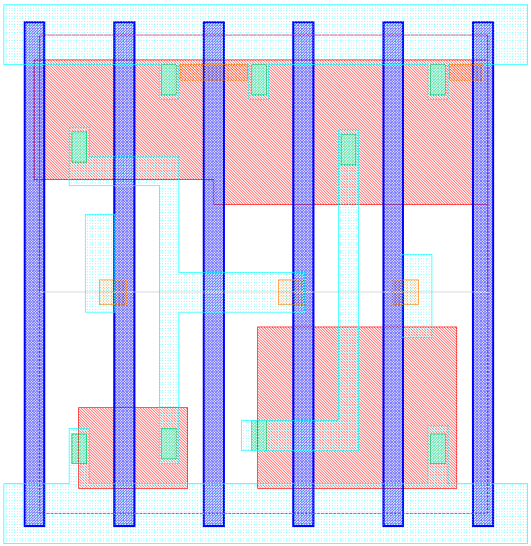

Figure 4.8: The Final Output of the Ported Layout

Chapter 5
Conclusion

The Layout Optimization and Porting of the different design layout blocks have been implemented in this project. The Layout modification is an important aspect of the Analog designer. For each design requirement, the layout is modified to meet the design specification. The main goal of the thesis is to devise a compiler that can modify the length, width, and pitch of each polygon in the cell of the analog layout. The automation algorithm is developed to optimize the layout of the digital design elements such as different design elements based on the design metrics like power, performance, and area of the digital design. These design metrics are an important aspect of any ASIC design and it is a primary aspect of the semiconductor companies to meet these design aspects in developing a product and meet time to market.

In Chapter 3, we were discussing the layout modification by length, width, and pitch of the polygons. The algorithm for modifying the length, width, and pitch of the polygons in the design was explained and the generated output layout is also imported in cadence virtuoso layout view and checked for any DRC in the design. This automation of the layout reduces the time required by manual work and the output layout is generated in minutes. This will ease the work of the analog designer and improve the time to market the developed memory chips. The Length modification of the layout is checked for DRC and it's nearly 80% DRC clean. The Pitch modification of the layout is also checked for DRC and it's 90% DRC clean. The Width modification of the layout is 100% DRC clean. By modifying the length, width, and pitch of the layout blocks, we can reduce the leakage by nearly 5% and there was an increase in the area of 2%.

In Chapter 4, the porting of the layout was discussed and the implementation flow was explained. The approach to convert an analog layout of the digital design layout blocks from technology node A to technology node B. It is separated into technology conversion to transfer design with the corresponding polygons and an optimization step for sizing all the polygons and paths based on the technology rule specification. The Porting of the analog layout was successfully implemented based on the output technology node specification. To port the design to different technology takes a lot of time to modify each polygon in the layout if it is done manually, automating this process will generate a ported version of the design in less time.

The interactive design optimization and the porting tool show significant potential for optimization algorithms to be used in the early conceptual stages of the design process to help designers explore solutions and trade-offs. Understanding the concept of the designer is an extremely important part of the design process, and it is one that tools typically ignore because of the understood nature of creativity and subjective judgement. The interactive tool assists the designer with design generation and evaluation, rather than attempting to automate these processes completely. This compiler tool is an interactive environment that processes the design building blocks based on the design requirements and can modify the layout in seconds and reduce a lot of manual work.

5.1 Future Works

This project can be expanded to a larger extent. Thus, some of the most important points for making this tool competitive in the industrial domain have been identified and are listed below

- The tool designed will modify and port the analog layout in the proper design flow. But the algorithm devised will work for most of the polygon shapes, but for some shapes, the design will give some DRC. So the algorithm can be optimized further to generate the layout with 100% DRC clean.
- The Input GDS file is flattened and then modified for any optimization, so it needs to be changed to maintain the hierarchy of each block in the cell and to maintain the hierarchy after modification.
- The Modified layout generated does not contains the labels, the algorithm can be improved by transferring the labels from the input layout on the proper polygons.

References

[1] Simple Exact Algorithm for Transistor Sizing of Low-Power High-Speed Arithmetic Circuits Tooraj Nikoubin, Poona Bahrebar, Sara Pouri, Keivan Navi, and Vaez Iravani.

[2] VLSI Routing in Multiple Layers using Grid based Routing Algorithms, May 2014 International Journal of Computer Applications 93(16).

[3] Standard Cell Routing via Boolean Satisfiability, 1Nikolai Ryzhenko, 2Steven Burns, Strategic CAD Labs, Intel Corporation, 1Moscow, Russia.

[4] S. K. Karandikar and S. S. Sapatnekar, "Fast comparisons of circuit implementations," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 12, pp. 1329–1339, 2005.

[5] C. H. Chang, J. Gu, and M. Zhang, "A review of 0.18-μm full adder performances for tree structured arithmetic circuits," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, no. 6, pp. 686–694, 2005.

[6] Production Layout Optimization for Small and Medium Scale Industry Yosra Ojaghia, Alireza Khademia, Noordin Mohd Yusofa,*, Nafiseh Ghorbani Renania, Syed Ahmad Helmi bin Syed Hassana

[7] An interactive optimization tool for architectural floorplan layout design. `https://www.cmu.edu/me/ddl/publications/2001-Michalek-MSThesis.pdf`

[8] Analog IP Porting by Topology Conversion and Optimization Udo Sobe, Achim Graupner, Enno Böhme, Andreas Ripp and Michael Pronath ZMD AG, MunEDA GmbH

[9] H.Graeb, S.Zizala, J.Eckmueller, and K.Antreich, "The sizing rules method for analog integrated circuit design," in IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 343–349, 2001.

[10] G.Stehr, M.Pronath, F.Schenkel, H.Graeb, and K.Antreich, "Initial sizing of analog integrated circuits by centering within topology-given implicit specifications," in IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2003.

[11] Heuristic Algorithms Maad M. MijwelUniversity of Baghdad May 2015

[12] Power, Area, and Performance Optimization of Standard Cell Memory Arrays Through Controlled Placement May 2016 Adam Teman, Bar Ilan University, Davide Rossi, Pascal Meinerzhagen, Luca Benini University of Bologna

[13] Performance driven FPGA design with an ASIC perspective Andreas Ehliar https://www.diva-portal.org/smash/get/diva2:158202/FULLTEXT04.pdf

[14] Path Planning Algorithm Using the Particle Swarm Optimization and the Improved Dijkstra Algorithm Hwan Il Kang, Byunghee Lee, Kabil Kim

[15] Introduction to VLSI systems January 1980, Carver Mead, California Institute of Technology, Lynn Conway, University of Michigan

[16] https://gdspy.readthedocs.io/en/stable/

[17] https://www.elprocus.com/how-integrated-circuits-work-physically/

[18] Review on Performance of Static Random Access Memory (SRAM) February 2015, Santhiya. V, Mathan Natarajamoorthy

[19] Read-only memory (ROM), January 2003, Michael J. Flynn Stanford University

[20] https://www.redblobgames.com/pathfinding/a-star/implementation.html

[21] https://en.wikipedia.org/wiki/Simulated_annealing#:~:text=Simulated%20annealing%20(SA)%20is%20a,space%20for%20an%20optimization%20problem.

[22] https://www.andersdx.com/ic-integrated-circuit/

[23] https://en.wikipedia.org/wiki/Very_Large_Scale_Integration

[24] Evaluation of SPICE P-N Junction's, BJT's and MOSFET's Model Parameters, October 2013, Ankara, Turkey, Mohammad Maadi.

[25] A Review of Optimization Techniques in Artificial Networks, September 2016 International Journal of Advanced Research 4(9):1668-1686, Omid Ghasemalizadeh, Meysam Khaleghian, Saied Taheri

[26] https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

[27] https://www.101computing.net/a-star-search-algorithm/

[28] https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/

[29] Introductory Chapter: VLSI, By Kim Ho Yeap and Humaira Nisar, Submitted: September 29th 2016 Reviewed: April 12th 2017 Published: February 28th 2018, DOI: 10.5772/intechopen.69188

[30] https://en.wikipedia.org/wiki/Integrated_circuit

[31] https://semiengineering.com/mixed-messages-for-mixed-signal/

[32] https://asic-soc.blogspot.com/2013/06/full-custom-ic-design.html

[33] Layout Automation in Analog IC Design with Formalized and Nonformalized Expert Knowledge, Daniel Marolt, Tag der mündlichen Prüfung: 04.06.2018, Institut für Halbleitertechnik der Universität Stuttgart 2018 https://core.ac.uk/download/pdf/186747795.pdf

[34] Challenges and opportunities toward fully automated analog layout design,Hao Chen, Mingjie Liu, Xiyuan Tang, Keren Zhu, Nan Sun, and David Z. Pan, 2020 http://www.jos.ac.cn/fileBDTXB/journal/article/jos/2020/11/PDF/20070021.pdf