TOWARDS ACCURATE FPGA AREA MODELS FOR FPGA ARCHITECTURE EVALUATION

by

Farheen Fatima Khan

B.Tech, Jawaharlal Nehru Technological University, Hyderabad, India, 2002

M.Tech, Jawaharlal Nehru Technological University, Hyderabad, India, 2006

A dissertation

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2017

Author's Declaration

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A DISSERTATION.

I hereby declare that I am the sole author of this thesis. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis or dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this thesis or dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Abstract

TOWARDS ACCURATE FPGA AREA MODELS FOR FPGA ARCHITECTURE EVALUATION

Dissertation Submitted by:

Farheen Fatima Khan, Doctor of Philosophy, 2017
Electrical and Computer Engineering, Ryerson University, Canada

Advisor

Dr. Andy Ye
Electrical and Computer Engineering, Ryerson University, Canada

Field Programmable Gate Array (FPGA) devices are integrated circuit chips which can be configured by the end user. FPGA architectures have evolved into heterogeneous System-on-Chips (SoCs) devices in order to meet the diverse market demands. Integrating reconfigurable fabrics in SOCs require an accurate estimation of the layout area of the reconfigurable fabrics in order to properly accommodate early floor-planning. Hence, this work provides an evaluation on the accuracy of the minimum width transistor area models in ranking the actual layout area of FPGA architectures. Both the original VPR area model and the new COFFE area model are compared against the actual layouts with up to 3 metal layers for the various FPGA building blocks. We found that both models have significant variations with respect to the accuracy of their predictions across the building blocks. In particular, the original VPR model overestimates the layout area of larger buffers, full adders and multiplexers by as much as 38% while underestimates the layout area of smaller buffers and multiplexers by as much as 58% for an overall prediction error variation of 96%. The newer COFFE model also significantly overestimates the layout area of full adders by 13% and underestimates the layout area of multiplexers by a maximum of 60% for a prediction

error variation of 73%. Such variations are particularly significant considering sensitivity analyses are not routinely performed in FPGA architectural studies. Our results suggest that such analyses are extremely important in studies that employ the minimum width area models so the tolerance of the architectural conclusions against the prediction error variations can be quantified. This work proposes a more accurate active area model to estimate the layout area of FPGA multiplexers by considering diffusion sharing and folding. In addition, we found that comparing to the minimum width transistor area model, the traditional metal area based stick diagrams, in lieu of actual layout, can provide much more accurate layout area estimations. In particular, minimum width transistor area can underestimate the layout area of LUT multiplexers by as much as a factor of 2-3 while stick diagrams can achieve over 85% -95% percent accuracy in layout area estimation. Based on our work, we present correction factors to the commonly used FPGA building blocks, so their actual layout area can be used to achieve a highly accurate ranking of the implementation area of FPGA architectures built upon these layouts.

Acknowledgement

Foremost, I would like to thank Almighty God for everything in my life.

I would like to express my heartfelt appreciation and sincere gratitude to my advisor, Dr. Andy Ye for his invaluable guidance, support and patience throughout the course of the work. I am privileged that he has given me an opportunity to work with him and help me refine and sharpen my skills. His encouragement and assistance in revision of the papers have contributed significantly to my research work. I am not only impressed by his profound knowledge and systematic research methodology but his simplicity and gentle personality will always influence my future career as well as personal life. Further, I am also thankful to Dr. Andy Ye for providing me a Lab with all amenities to help me perform my research work for all the past years.

I also wish to thank my committee members Dr. Fei Yuan, Dr. Lev Kirischian and Dr. Vadim Geurkov for generously offering their time, helpful suggestions and feedback.

I am highly indebted to my parents for their able guidance and who have laid the foundation of my career by teaching me the morals and ethics that an ideal person needs to know. My best source of inspiration is my dad who made me feel self-confident, and my mom who is my motivation. I deeply express my gratitude for their eternal blessings. Moreover, I am highly obliged to my late father-in-law for inspiring me to always do better than before.

I want to especially thank my husband, Arshad Mohammed without his support and encouragement I could never finish the Ph.D. studies. I owe him for the sacrifices he has made to fulfill my dream. A special thanks to my children, Aleena and Hamza for being patient and understanding during the past years. Their endless love and long waits for me to play with them will always be remembered.

I am also thankful to my brother and sister and all my well-wishers who have provided me with encouraging words to accomplish my goals.

Finally, and most importantly, I would like to greatly acknowledge the financial support from my advisor, Dr. Andy Ye as well as Ryerson University.

Dedication

To my beloved parents & my husband, Arshad Mohammed.

Table of Contents

List of Tables

List of Figures

Abbreviations

FPGAs	Field Programmable Gate Arrays
SoC	System-on-Chip
ASICs	Application Specific Integrated Circuits
CAD	Computer-aided Design
VPR	Versatile Place and Route
COFFE	Circuit Optimization for FPGA Exploration
MOSIS	Metal Oxide Semiconductor Implementation Service
LUT	Look-Up-Table
IO	Input Output Block
BLE	Basic Logic Element
LC	Logic Cell
LB	Logic Block
CB	Connection Block
SB	Switch Blocks
SRAM	Static Random Access Memory
LAB	Logic Array Block
ALM	Adaptive Logic Module
CLB	Clustered Logic Blocks
D-FF	Data Flip-Flop
CMOS	Complementary Metal Oxide Semiconductor
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
IC	Integrated Circuits

List of Symbols

k	LUT size (Number of inputs per LUT)
N	Number of LUTs per cluster
I	Number of inputs per cluster
L	Wire segment length
Fc_input	Logic cluster input pin connectivity
Fc_output	Logic cluster output pin connectivity
N_{tile}	Number of tiles that are required to implement the circuit
N_i	Number of basic FPGA building blocks
E_i	Estimated layout area of each building block of the FPGA tile
E	Estimated layout area of the FPGA tile
A_i	Actual layout area of each FPGA building block
A	Actual layout area of the FPGA tile
i	Each building block
α	Scaling factor of the FPGA tile
α_i	Scaling factor for each FPGA building block
λ	Half transistor length
n	Number of folds of transistor
w_{eff}	Effective width of transistor
z	Number of inputs to decoded multiplexer
N_{inv}	Number of inverters
D_{min}	Minimum delay

F	Path effort
g	Logical effort
h	Electrical effort
C_{out}	Output capacitance of the gate
C_{in}	Input Capacitance of the gate
C_L	Load capacitance
mux	Multiplexer

Chapter 1

Introduction

1.1 Overview

FPGAs (Field Programmable Gate Arrays) are widely used digital circuits to implement several applications in digital signal processing, video/audio processing, biomedical engineering and scientific computation. They have become very popular as compared to ASICs (Application-Specific Integrated Circuits) due to their post fabrication re-programmability. Programmability of FPGAs is due to the presence of programmable logic blocks and programmable routing. The logic blocks are used to implement digital logic, while the programmable routing are used to connect the logic blocks to form larger circuits.

In the present era, the demand for high-performance devices has led to Heterogeneous Computing. FPGA devices known for post fabrication re-programmability of the hardware chip can be integrated with other devices on a single chip. However, integrating reconfigurable fabrics in SoCs (System on Chips) require an accurate estimation of the layout area of the reconfigurable fabrics in order to properly accommodate early floor-planning also an accurate estimation would help in FPGA architecture research. Hence, this research leads to heterogeneous computing with FPGAs by providing area estimations of FPGA components based on layout work rather than area based on simple equation-based models.

FPGA architectures are designed using an empirical approach with the help of CAD tools. Different architectures are evaluated by mapping benchmark circuits to the architectural models

and various quality metrics such as speed, area, and power are measured and compared. The resulting information is then used as a guide by the human architect to select the best suitable architecture for a target application. Therefore, the reliability of this data is strongly dependent on the accuracy of the models used by the different CAD tools in evaluating architectures.

Hence, this work provides an evaluation on the accuracy of the area model, the minimum width transistor area model, in ranking the actual layout area of FPGA architectures. The minimum width transistor area model is widely used area model in many previous FPGA architectural studies [1]-[10], in estimating the implementation area of proposed FPGA architectures. This model was originally introduced in the VPR tool as its area model and a modified version based on transistor sizing is used in COFFE. In this work both the original VPR area model and the new COFFE area model are compared against the actual layouts with up to 3 metal layers for the various FPGA building blocks.

As our initial work, we compare the layout areas of LUT multiplexers with three area models, VPR, COFFE and the traditional metal-area-based stick diagram layout area estimation technique. Details of stick diagram area estimation are explained in Chapter 2 and its area analysis is presented in Chapter 6 . We found that, comparing to the minimum width transistor area models, the VPR model and the COFFE model, the traditional metal area based stick diagrams can provide much more accurate layout area estimations. In particular, minimum width transistor area can underestimate the layout area of LUT multiplexers by as much as a factor of 2-3 while stick diagrams can achieve over 90 percent accuracy in layout area estimation. Then, we discuss the layout strategy for different components and present a new mirroring technique for designing LUTs, where larger size LUTs are created from smaller size LUTs using the mirroring technique. The LUT multiplexer area comparison and mirroring technique are published in [12].

After that, we present correction factors for different components. In addition, we observed that there is a wide range of inconsistency against the predicted estimations across various FPGA components. As both VPR and COFFE area models use generic area estimation equations for all components based on individual transistor count and the spacing between adjacent transistors, they consider neither circuit topology nor the actual connectivity of adjacent transistors. To determine accurate area estimations, in this work we take into account both circuit topology and connectivity of neighboring transistors including metal and diffusion sharing. Later, we propose a more accurate method to estimate active area for FPGA multiplexers extensively found in logic blocks and routing architectures. In future, we plan to have an open source library of the actual layouts of various FPGA components of different sizes. These layouts are skillfully designed best effort manual layout to achieve compact area with minimum white spaces.

1.2 Related Works

Minimum Width Transistor Area has been used in many FPGA architectural studies [1]-[10] in estimating the implementation area of proposed FPGA architectures. The model assumes a set of basic building blocks for FPGAs and takes in a set of FPGA architectural parameters as input [11]. It then estimates the total active area that is required to implement a given FPGA architecture by counting transistors. The count assumes only minimum channel length transistors and each minimum width transistor is counted as one unit of area, while the area of larger transistors are scaled as a function of their width. The minimum width transistor area model has always been considered to be a high fidelity model in ranking the implementation area of FPGA architectures [11].

In this work, we exam the fidelity of the minimum width transistor area model in ranking the implementation area of FPGA architectures. It is based on the fact that the consistency of an area model in over/under-predicting the actual layout area is extremely important in the model's ability to correctly rank architectures. In particular, when a model consistently over/under-predicts layout area, it will correctly rank architectures; even though the model gives inaccurate overall layout area. When a model inconsistently over/under-predicts layout area, on the other hand, it will lead to incorrect ranking of architectures. Consequently, in this work, we measure the consistency of the minimum transistor area model in predicting the layout area of the various fundamental FPGA building blocks.

The accuracy of the minimum transistor area model in ranking FPGA architectures is particular important in the era of FPGA-based SOC designs. With ever-increasing logic capacity, there are an increasing number of FPGA-based SOC designs [13][14]. These designs contain a mix of reconfigurable fabric and fixed logic in order to maximize performance and minimize power consumption of their target applications. Currently these FPGA-based SOC designs are mainly from traditional FPGA companies, which have the knowhow of designing efficient reconfigurable fabrics. As a result, these products are typically general-purpose in nature and are architected to target a variety of applications. But as architectural-level research progresses, it has become increasingly evident that reconfigurable fabrics can benefit a variety of applications from processors to signal processing ASICs. Consequently, there is an increasing demand for non-FPGA companies, such as CPU and ASIC manufacturers, to include reconfigurable fabrics on their traditionally fixed logic products. To decide if reconfigurable fabrics should be included in a specialized SOC design, it is extremely important to select the most efficient architecture for the

target application. It is important to understand the accuracy of the current de facto method of estimating the area of reconfigurable fabrics in correctly ranking architectural alternatives.

The minimum width transistor area model is first used in [1] to investigate the effect of logic cluster inputs on the implementation area of logic clusters. Subsequently, the model is used in several major FPGA architectural studies on logic cluster topology [2]-[6], FPGA routing area [7]-[9], effect of multi-bit routing resources on FPGA logic and routing area [10]. It also serves as one of the fundamental models used by the VPR tools [11] [15]. Few studies, however, have been conducted on the accuracy of the minimum width transistor area model. In particular, the study in [16] [17] studies the automatic generation of FPGA tiles based on FPGA architectural parameters. The automatically generated tile area is significantly larger than custom design tiles, which are used in industrial designs; and consequently no direct comparison is made between the generated tile area to the minimum width transistor area. The work in [18] used a modified version of the minimum width transistor area model based on transistor sizing of individual transistors. However, their justification is given to the modification based on layout work done for individual transistor sizing and not on the actual layout of FPGA fabric. The work in [12] examines the accuracy of the minimum transistor area model on predicting the overall layout area of FPGAs. The fidelity of the model in ranking FPGA architectures, however, has not been examined before.

This work is based on the actual layouts drawn using the publicly available SCMOS (Scalable Complementary Metal Oxide Semiconductor) deep submicron scalable layout rules in Magic [19] [20]. Once the layouts are designed, we compare physical layout area and active area to the predicted area of the minimum width transistor area models.

1.3 Motivation

The performance and logic density of FPGA architectures are strongly influenced by a set of interrelated architectural parameters. In particular, Table 1.1 shows the set of architectural parameters that define the cluster-based FPGA architecture extensively studied in academia. In particular, k is the number of inputs per look-up table (LUT). Varying k has a direct effect on the value of N, the number of LUTs per logic block, and the value of I, the number of inputs per logic block, that should be used to achieve optimum performance and logic density. Varying L, the wire segment length or multiple lengths, affects the optimum values of Fc_input, the number of input connections per logic block input pin, and the optimum value of Fc_output, the number of output connections per logic block output pin. Consequently, to identify an efficient FPGA architecture for a target benchmark set, one needs to sweep through the set of architectural parameters and empirically identify the most efficient parameter settings for each parameter. Due to time consuming nature of VLSI layout, the FPGA architectural research community has avoided measuring the actual implementation area of FPGA architectures based on actual layout area and has relied on simple equation-based minimum width transistor area model to estimate the layout area.

Table 1.1 : FPGA Architectural Parameters

Parameter	**Description**
k	LUT size (Number of inputs per LUT)
N	Number of LUTs per Cluster
I	Number of inputs per Cluster
L	Wire segment length
Fc_input	Logic cluster input pin connectivity
Fc_output	Logic cluster output pin connectivity

This simple equation-based model, however, ignores wiring area and circuit topology. By ignoring wiring area and circuit topology, the model can significantly over or under predict the layout area of FPGA building blocks. More importantly since wiring demand and circuit topology can vary significantly from one FPGA building block to another, the amount of over/under-prediction can also vary significantly from one building block to another. The variation, in turn, can significantly reduce the model's ability to accurately rank FPGA architectures.

In particular, in VPR, the implementation area of a circuit on an FPGA is calculated by the following equation:

$$E = N_{tile} * \sum_{i \in all\ building blocks} (N_i * E_i) \quad (1)$$

where N_{tile} is the number of tiles that are required to implement the circuit and N_i is the number of basic FPGA building blocks of a given type in a tile and E_i is the estimated layout area of each type of building block that the tile contains. Note that both N_{tile} and N_i values are architectural dependent, they change as the architectural parameters shown in Table 1.1 changes. The E_i values, however, are architectural independent and are only a function of the number and the sizes of the transistors in a specific building block, and this equation mirrors the actual area measurement when the full layout of an FPGA tile is available where the area can be measured by:

$$A = N_{tile} * \sum_{i \in all\ building blocks} (N_i * A_i) \quad (2)$$

where A_i is the actual layout area (including wiring area) of each FPGA building block.

It is important to note that if the estimated area E_i always consistently over/under-estimates the actual layout area A_i by a constant factor α, (i.e. $A_i = \alpha * E_i$ for all i that belongs to all building

blocks), then the actual total layout area also varies from the estimated layout area by α (i.e. A = α * E). Since the actual layout area is always a constantly scaled version of the estimated layout area regardless of the architectural parameter setting, the architectural conclusions drawn using the estimated area will always be the same as the ones drawn using the actual layout area.

If the scaling factor between A_i and E_i varies from component to component, (i.e. $A_i = \alpha_i * E_i$ for all i that belongs to all building blocks), then the total actual layout area, A, becomes:

$$A = N_{tile} * \sum_{i \in all\ building blocks} (N_i * \alpha_i * E_i) \quad (3)$$

The scaling factor between the actual layout area, A, and the estimated layout area, E, will then be dependent on the N_i values, which are architectural dependent variables. Consequently, the scaling factor between A and E will also be architecturally dependent and the architectural conclusion drawn based on the estimated layout area will deviate from the architectural conclusion drawn using the actual layout area.

In this work, we measure α_i, the ratio between the actual layout area and the estimated layout area, for each fundamental building block of the VPR FPGA architecture.

1.4 Research Objectives

Primary Objective

To study the accuracy of current area model. The current area model is based on active area which is defined by the minimum width transistor area. The active area is generally used to measure the area of any given FPGA architecture. Numerous studies [1]-[10] are based on active area. For example Ahmed and Rose [5] state that 4-input LUT size when used in any logic cluster

gives better FPGA performance and density. There is no verification on the accuracy and fidelity of the area model. These models neither consider circuit topology nor diffusion sharing. Components which have exclusively parallel and in series connected transistors can extensively employ diffusion sharing in order to minimize their layout area. This study is based on the actual layout drawn based on lambda (lambda is half the minimum transistor length) rules. Once the layouts are designed we compare physical layout area with active area. Given the importance of area model in FPGA research and an improved area model will be proposed. The specific objectives are:

- Create a more accurate method to estimate the active area of FPGA multiplexers by considering layout factors including diffusion sharing, folding and connectivity of transistors.
- Measure FPGA LUT layout area as LUT is scaled from 3-input to 6-input LUTs.
- Analyze the minimum number of metal layers needed to minimize the layout area of fundamental FPGA building blocks.
- To draw conclusion on FPGA layout area, if area is metal limited or active area limited.
- Provide correction factors for the layout area of the various FPGA components over the minimum width transistor area model.

Secondary Objective

- In future, create a library of layouts of widely used FPGA components and release it.

FPGA consists of large number of programmable logic blocks, each of which implements digital logic. These logic blocks form an integral part of any FPGA architecture. Our additional goal is to manually design layouts for these components in various transistor sizes and

configurations to meet different architectural specifications of widely used FPGA architectures. Later, we will release the designed layouts of FPGA fabric in Magic. This is an initial step to make FPGA design more accessible.

The manual layout design process is tedious and timing consuming, but will be closer to the current commercial layouts practice. We use Magic, open source IC layout software which is based on lambda rules [20]. The layouts can be scalable to any desired process, by specifying a new value to lambda. Since the layouts are scalable to any process of choice, the effort put in creating a library of generic FPGA components and making it publicly available will be a one-time effort. Our research will open the doors for small semiconductor companies to initiate the development of FPGA fabric at low cost. This will overcome the limitation of current commercial FPGA products which are highly efficient but, their designs are completely dependent on a few established FPGA vendors.

1.5 Framework

This research work focuses on island style, homogeneous FPGA architecture, where each FPGA tile is identical and consists of logic blocks, connection blocks and switch blocks. In this work, we examine the accuracy of using minimum width transistor area model, a widely used area model in many previous FPGA architectural studies, in assisting floor-planning process. Both the original VPR area model and the new COFFE area model are compared against the actual layouts area of FPGA components designed using Magic and conventional stick diagrams area estimation technique. Area comparison is done with VPR and COFFE as they are the only two currently existing FPGA area estimation tools. Here, the FPGA architecture area estimation evaluation is done for planar CMOS single die FPGA technology.

The layout tool is Magic, an open-source IC layout tool used to layout different FPGA components. Magic layout tool was deliberately chosen as it is process independent and uses scalable design rules. In contrast, using specific vendor design rules will yield a layout specific to a process and will be hard for the layout design to be ported from one process to another process. Our presented layout work is based on scalable MOSIS (Metal Oxide Semiconductor Implementation Service) deep submicron design rules. Also, note that, Magic is not binded with NDA's (Non – Disclosure Agreements), this enables our layout work to be easily published.

The layout achieved for each FPGA component is our best effort manual layout. The layout work was an exhaustive manual layout design process to achieve a compact area with minimum white spaces. We have used up to 3 metal layers to layout all the FPGA basic building blocks. Initially, only metal1 and metal 2 is used. Later, metal 3 is used to study the effect of area change with more number of metals. In this study we focus on the layout area of multiplexers, as they are one of the most widely used basic building blocks found throughout the FPGA architecture. We also measure the layout area of other basic FPGA building blocks which are architectural dependent. Layouts of larger transistor size 4x, 6x, 8x and 16x are also laid along with minimum width transistor size,1x to study the effect of prediction error of both area models VPR and COFFE.

1.6 Contributions

The main contributions of the presented work are:

- Presents a novel layout strategy using mirroring technique to create higher order multiplexers from lower order multiplexers. This ensures maximize design reuse and good quality layouts.

- Presents new active area models for LUT multiplexers and decoded multiplexers based on the mirroring strategy. True active area of a k-input LUT and decoded multiplexers can be determined depending on the number of 2:1 multiplexers that it contains. This area model is a universal area model for all SCMOS based technologies.

- Presents a mathematical equation to determine the best number of folds for transistors to achieve minimum active area.

- Provides correction factors for basic FPGA building blocks. This includes pass transistor based components and CMOS based components. The bloat factor differs from component to component across an FPGA tile and within a single component it varies with transistor size. The correction factors are presented for encoded and decoded multiplexers of varying sizes and CMOS based components in Table 6.5 and Table 6.7 respectively.

- Presents skillfully designed layouts for common FPGA structures and identifies inaccuracies in minimum width transistor area models used to make conclusion about FPGA architecture.

The outcome of our research is:

- Demonstrates that the currently used models, VPR and COFFE work well on earlier FPGA architectures (using only encoded multiplexers) but show for newer architectures (mix of encoded and decoded multiplexers) the quality of area prediction of these models decreases significantly.

- One of our study shows that a standard stick diagram based layout area model should be developed for FPGA architectural evaluations. As the minimum width transistor area

models, such a model can remain IC-process independent and can greatly enhance the accuracy of the conclusions drawn from architectural-level investigations by capturing the effect of wiring as well as folding and diffusion sharing on layout area.

- Suggest the development of more accurate area models by taking into account actual layouts including their connectivity and grouping of adjacent transistors.

Publication of this work is listed below:

Journal

J1. **F.Khan** and A.Ye, "A Study on the Accuracy of Minimum Width Transistor Area in Estimating FPGA Layout Area", *Elsevier Microprocessors and Microsystems – Embedded Hardware Design* (accepted)

J2. **F.Khan** and A.Ye, "An Evaluation on the Accuracy of the Minimum Width Transistor Area Models in Ranking the Layout Area of FPGA Architectures ", *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* (submitted)

Conferences

C1. **F.Khan** and A.Ye, "An Evaluation on the Accuracy of the Minimum Width Transistor Area Models in Ranking the Layout Area of FPGA Architectures", *26th International Conference on Field Programmable Logic and Applications (FPL)*, Lausanne, Sep. 2016, pp.1-11 (acceptance rate: 21.3%) **Received Best Paper Award**

C2. **F.Khan** and A.Ye, "An Empirical Analysis of the Fidelity of VPR Area Models", *24th IEEE International Symposium on Field-Programmable Custom Computing Machines*, pp.138, May 2016. (poster acceptance rate: 40.6%)

C3. **F.Khan** and A.Ye, "Measuring the Accuracy of Minimum Width Transistor Area in Estimating FPGA Layout Area", *23rd IEEE International Symposium on Field-Programmable Custom Computing Machines*, pp. 223-226, May 2015. (short paper acceptance rate: 38.9 %)

1.7 Dissertation Outline

This dissertation is composed of seven chapters and is organized as follows.

Chapter 1 presents the introduction by giving an overview of research, related works, motivation and its objectives and contribution of the research work.

Chapter 2 describes background information on FPGA architecture including logic block and routing architecture, minimum width transistor area model and discusses stick diagrams.

Chapter 3 presents detailed layouts of different CMOS based FPGA components along with the layout techniques used.

Chapter 4 proposes a new minimum active area model for encoded and decoded multiplexers.

Chapter 5 reveals the layouts of pass transistor based components consisting of encoded and decoded multiplexers.

Chapter 6 presents experimental results and an area analysis study is carried out by comparing the actual layout area with both area models, the VPR and the COFFE models, and also compares the layout area with stick diagram representations.

Chapter 7 concludes and indicates the future research.

Chapter 2

Background

This chapter provides the relevant background material related with our work. In the first section, we discuss the FPGA logic block architecture followed by the FPGA routing architecture. Next section presents the two minimum width transistor area models and finally the last section discusses the traditional metal area based stick diagram area estimation.

2.1 FPGA Architecture

FPGA devices are integrated circuits, which are widely used in various applications. They provide a great amount of flexibility as their hardware components can be reconfigured after manufacturing depending on the requirements of a particular application. The hardware components consist of logic blocks, and routing resources, which interconnects the logic blocks. Depending on the routing architecture, FPGA's are broadly classified as island-style, row-based or hierarchical [11]. In this research work, we focus on the island style FPGA architecture consisting of an array of tiles. It is so called because the logic blocks on either side are surrounded by routing. This type of architecture is widely used both in industry as well as in academic research. It is important to note that major commercial vendors Xilinx and Altera also use this type of architecture. Below Fig. 2.1(a) shows typical island style FPGA architecture and (b) shows its associated tile. An FPGA tile comprises of a logic block (LB), connection blocks (CB) and a switch block (SB). Kuon, Eiger and Rose [16] state that this type of architecture when laid has

benefit of a regular structure as the tile is repeated. Hence, allowing an FPGA device to be created by replicating the tile.

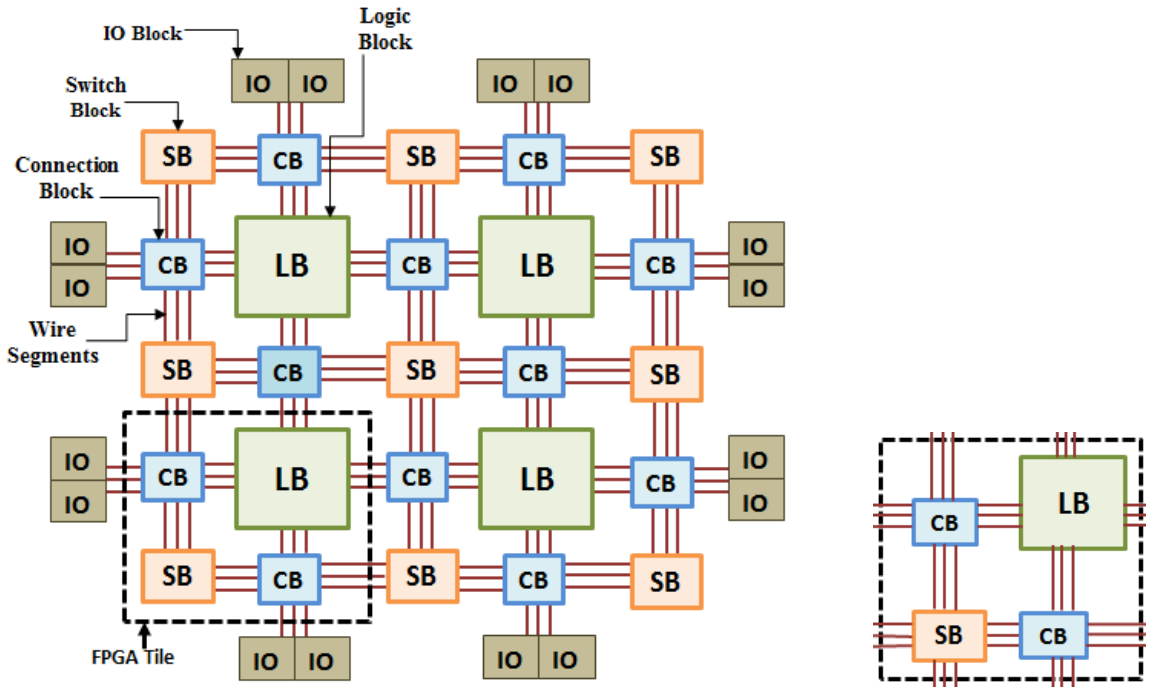


Fig. 2.1: (a) Island-style FPGA architecture (b) FPGA tile

The logic blocks are an integral part of FPGA architecture; they are used to implement digital logic. The programmable routing comprises of connection blocks and switch block. The I/O blocks are interconnected to the logic blocks via programmable routing. The entire routing architecture results in a highly flexible network.

In our work the programming technology used for the FPGAs is Static Random Access Memory (SRAM), as this is a widely used technology when compared to fuse, anti-fuse and flash. Anti-fuse and fuse can be programmed only once whereas flash can be programmed multiple times but require special IC processes; SRAM can be reprogrammed unrestricted number of times and can be implemented in conventional CMOS IC processes. As SRAMs do not need special IC

processes to manufacture, this enables FPGAs to be easily modified and updated after the development cycle. A great amount of flexibility in the routing resources and logic blocks of FPGA architecture is provided by the presence of reprogrammable SRAM switches. They are commonly used to control pass transistors, multiplexers and tri-state buffers which we will review in the later sections.

The sub-sections below provide an in-depth explanation of logic blocks and routing architecture along with circuit level details.

2.1.1 **Logic Block Architecture**

The composition of a logic block has a great impact on FPGA area and speed. The basic unit which implements logic is a look-up table (LUT) and associated with it is a data flip-flop (D-FF) and a multiplexer to select whether a combinational or sequential circuit is realized. Together these elements are generally termed as logic elements (LEs) or basic logic elements (BLEs) by Altera and logic cell (LC) by Xilinx as shown in Fig. 2.2. Previous studies [1] and [5] have shown that LUT size of 4 gives the best area and LUT size of 6 gives the best performance. Commercial FPGAs use LUT size 4 or 6.

For better performance LE/LC are grouped together and are commonly named as logic blocks (LB). With additional circuitry, Xilinx term it as *Configurable Logic Blocks (CLBs)* and Altera term it as *Logic Array Blocks (LABs).* Starting from Stratix II, Altera uses a new term as Adaptive Logic Module (ALM) to describe its logic element the details of which will be presented later in the chapter.

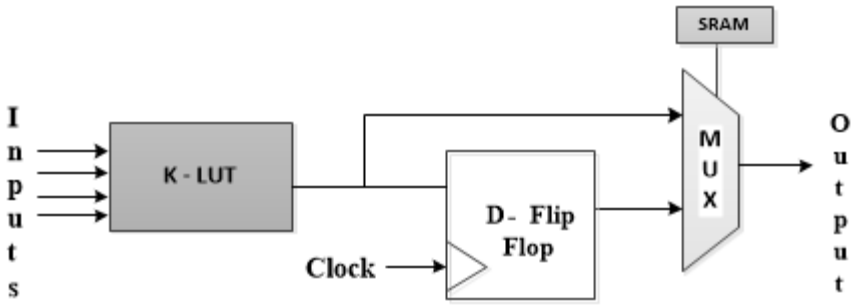


Fig. 2.2: Components of basic logic element (BLE)

Here, we present a bottom-up description of all the components of logic block architecture in a hierarchy along with functionalities and implementations. We will first give the details of look-up tables (LUTs) and its implementations. Then describe clustered logic blocks.

2.1.1.1 LUT

As discussed a LUT is the basic component which implements logic. A k-input LUT design requires 2^k SRAM cells and a 2^k input multiplexer [11]. This can realize up to 2^{2^k} boolean functions by programming the 2^k SRAM cells. As an example below, Fig. 2.3 shows the schematic of a 2-input LUT and its equivalent transistor level circuit diagram using pass transistors. As shown a 2-LUT requires 4 SRAM cells and a 4:1 multiplexer. A buffer is used to drive each input of a LUT and an inverter is used to generate the complement of the input. The multiplexer function here is realized by the pass transistors, which is one of several common implementations and may have repeaters in a deep LUT [18]. Pass-transistors are the simplest form of a transistor switch with least area utilization. The work, in [25] shows that as process technology scales down pass-transistors soon could be replaced by transmission gates. However, pass transistors remain to be used in current commercial FPGAs due to its high area efficiency.

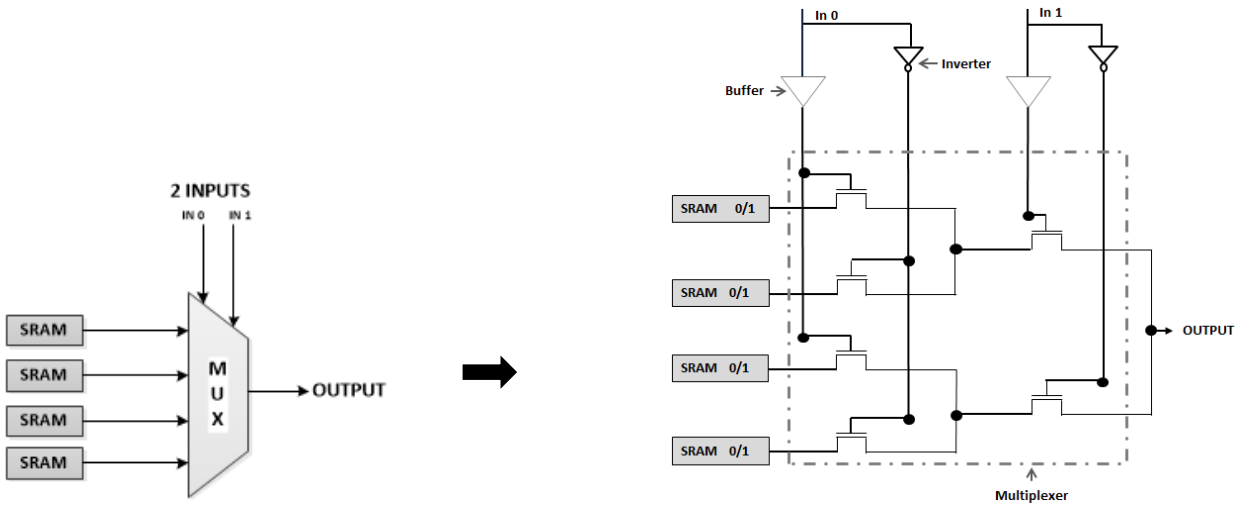


Fig. 2.3: 2-LUT schematic and circuit diagram

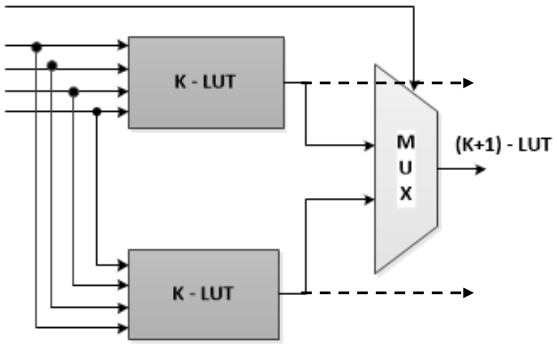


Fig. 2.4: (K+1)-LUT created with two K- LUTs

2.1.1.2 Composable LUTs

They are formed when smaller sized LUTs are fasten together to create a larger sized LUT. Fig. 2.4 shows one of the designs of (k+1)-LUT using two k-LUTs. Using composable LUTs one can realize either the functionality of two k-LUTs or a single (k+1)-LUT. Note that, as the number of inputs of a LUT is increased, more logic can be realized but on the other hand the complexity within the LUT also exponentially increases. Hence composable LUTs give a better solution. They are found since Xilinx Virtex-5 [35].

Note that a k-LUT can implement 2^{2^k} distinct functions. However, if they are n k-LUTs, one can implement $(2^{2^k})^n$ functions. So, if they are two 4-LUTs one can perform $(2^{2^4})^2 = 2^{2^5}$ functions, which is equivalent to the functions represented by a 5-LUT.

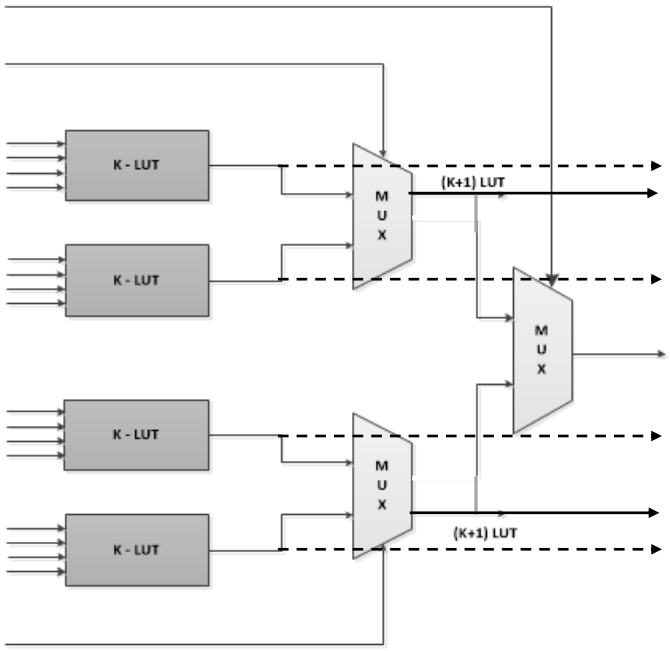


Fig. 2.5: (K+2)-LUT created from four K-LUTs

As another example, Fig. 2.5 shows a 6-LUT constructed from four 4-LUTs. This design requires 3 additional multiplexers and a total of 19 inputs. This way of implementation can either result in one of the following logic functions, four 4-LUT logic functions, two 5-LUT logic functions or one 6-LUT logic function. However, each input pin is linked with routing resources, as the number of inputs increases the area also increases making it highly area inefficient. So for implementing larger LUTs, Altera uses another variation known as Adaptive Logic Module (ALM) [22] discussed below. The main aim was to improve speed but also minimize area. It presents how efficiently a 6-LUT is partitioned into smaller LUTs [22] taking advantage of the high performance of 6-LUT an avoid area penalty form excessive inputs.

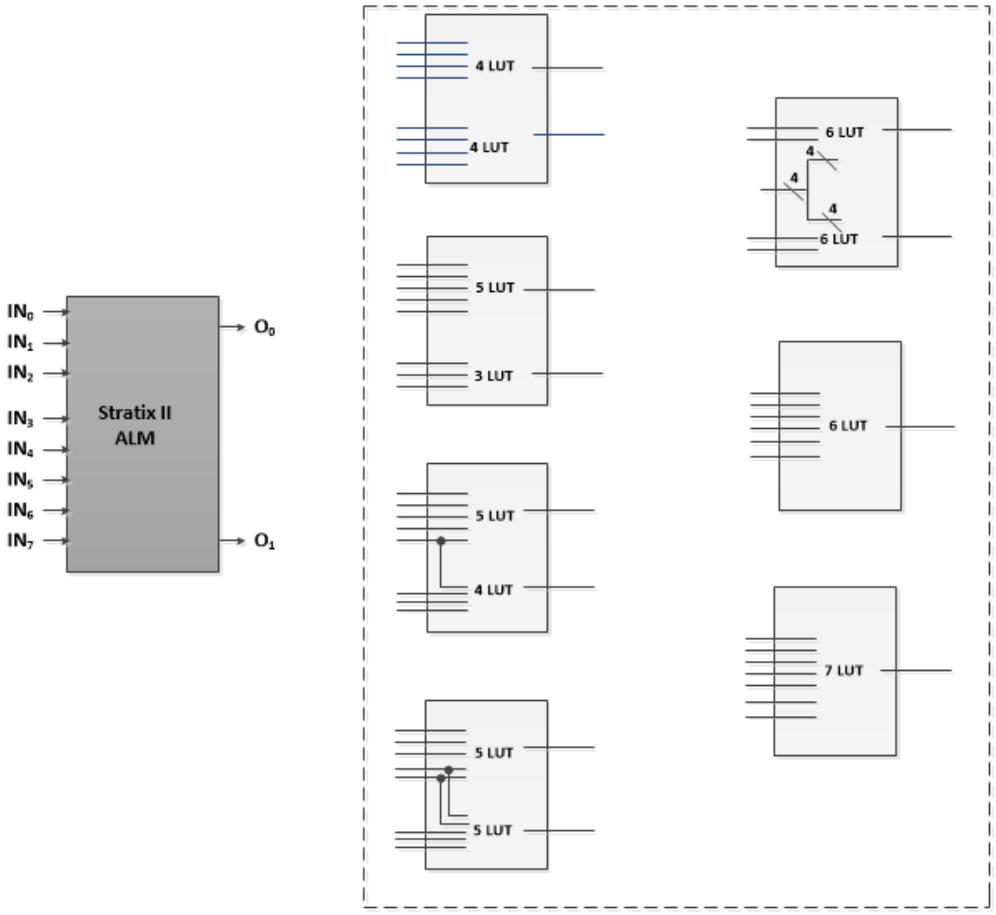


Fig. 2.6: Stratix II ALM LUT

2.1.1.3 Fracturable LUTs - Stratix II ALM

In Stratix II the fundamental component to implement logic is ALM (Adaptive Logic Module) instead of LUT. Each ALM is associated with 2 flip-flops rather than 1 flip-flop to support two logic functions. Thus, it has 8 inputs and 2 outputs. The largest LUT that a ALM can implement is a 7-LUT. As shown in Fig. 2.6, a total of 8 inputs can be configured to implement in different ways by sharing and unsharing inputs, providing greater flexibility. The ALM can implement two 4-input LUT, one 5-input and one 3-input LUT, one 5-input and one 4-input LUT, two 5-input

LUTs with two shared inputs, two 6-input LUTs with 4 shared inputs and one 7-input LUT. As reported in [3][5] a 6-LUT has better performance and a 4- LUT is more area efficient. Hence, the main aim of the ALM design is to have a balance between 6-LUT speed and 4-LUT area efficiency.

2.1.1.4 Clustered Logic Block

A logic block contains a cluster of basic logic elements (BLEs). A clustered logic block is shown in below in Fig. 2.7. This is similar to the one used in [11] and [18]. The logic cluster has N BLEs where each BLE is a k-input LUT and I is the total number of inputs to a logic cluster. For better utilization of resources, [5] found that N*k inputs is not required and k/2 *(N+1) is sufficient. In most academic research, the cluster is considered to be fully connected, that is all I inputs and N outputs are internally connected to each k-input LUT [11]. Commercial architectures can use less than fully connected logic clusters to increase area efficiency.

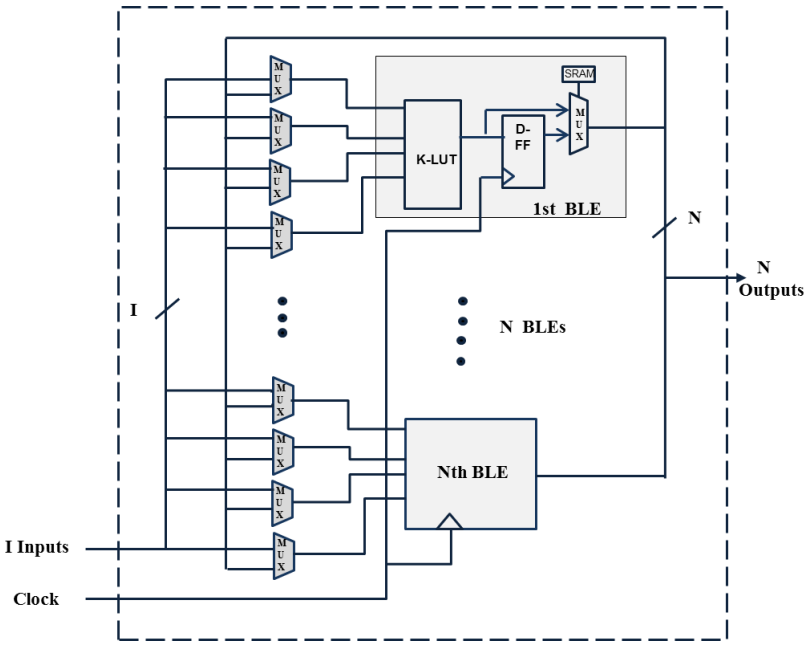


Fig. 2.7: Clustered Logic Block

It is also observed that for sufficiently large clusters, the area used by local interconnect will exceed the area saved in global interconnect. Therefore, for an optimum balance of speed and density cluster size of 3 to10 is typically used [5].

Commercial FPGA tiles such as the one shown in Fig. 2.8 and VTR tiles [15] can also include arithmetical units such as full adders. Commercial logic blocks have evolved over time in an attempt to gain more functionality and therefore are no longer pure k-LUTS. Hence, in our area comparison we also consider full adder layout area with both area models [11] and [18]. Also, the circuit topology of full adder is different from multiplexers. We would like to compare the difference in predicted area with other components.

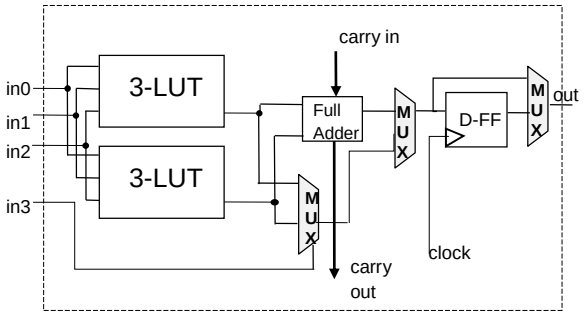


Fig. 2.8: Logic cell of commercial FPGA tile

2.1.1.5 Importance of multiplexers

In particular, the LUTs are implemented using multiplexers as discussed in Section 2.1.1.1. Multiplexers are also used to connect the LUTs together to form logic blocks as shown in Fig. 2.9 [11]. Furthermore, as shown by Fig. 2.9, multiplexers are also the key building blocks of connection blocks, which provide programmable connections between the routing tracks and logic block input pins, and the key building blocks of switch blocks, which provide inter-routing-track connections and logic blocks output to routing track connections [8]. Consequently, accurately

estimating the layout area of multiplexers is extremely important both in estimating the layout area of FPGA tiles from their architectural specifications and in choosing FPGA architectural parameter values in order to minimize FPGA layout area. As a result, in this study we focus on the layout area of multiplexers, as they are one of the most widely used basic building blocks found throughout the FPGA architecture. We also measure the layout area of other basic FPGA building blocks including buffers, SRAMs and full adders.

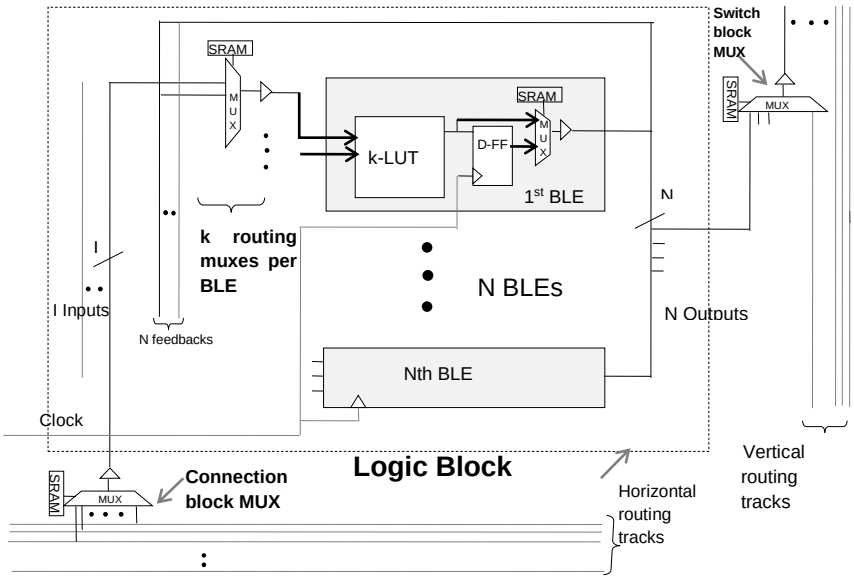


Fig. 2.9: A typical FPGA tile structure

2.1.2 **Routing Architecture**

We discuss the routing architecture for island style FPGAs. An island style FPGA has routing channels on all sides of the logic blocks. Routing within each logic block is termed as local routing and the routing which interconnects the clustered logic blocks is termed as global routing.

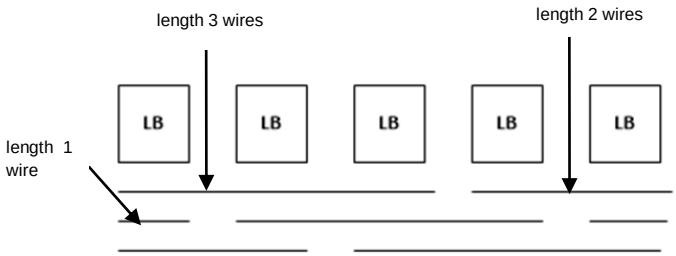


Fig. 2.10 : Wire segments distribution

In any FPGA device, logic blocks are interconnected using programmable routing to form larger circuits. Interconnections are possible with the help of routing wires, switch blocks and connection blocks. The routing wires span both horizontally and vertically across the FPGA. Logic blocks have routing channels on all four sides. The number of routing wires in each channel is denoted by W. As shown in Fig. 2.10 each channel has wire segments of different lengths in order to provide a balance of area and delay in a routing network. For example, longer wire segment span multiple logic blocks and require fewer switches, thus have less area and have higher performance. But longer wire segments are also less flexible and typically have lower utilization. As suggested in [11], FPGA architecture should have a mix of 4 and 8 length of wire segments for better area- delay product. The number of wire segments connected to each logic block input pin is Fc_input and the number of wire segments connected to each logic block output pin is Fc_output. The connection blocks connect the output of one logic block to the input of another logic block via routing multiplexers. The multiplexers used in the routing architecture are decoded type or could be hybrid and are used for faster connections. The details of decoded multiplexers are discussed in Chapter 4

In the sub-section below we present switch blocks and connection blocks.

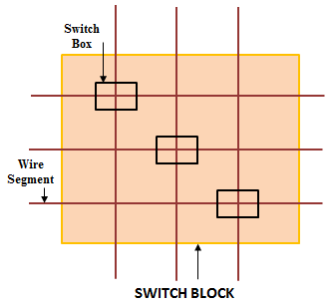


Fig. 2.11: Example of disjoint switch block

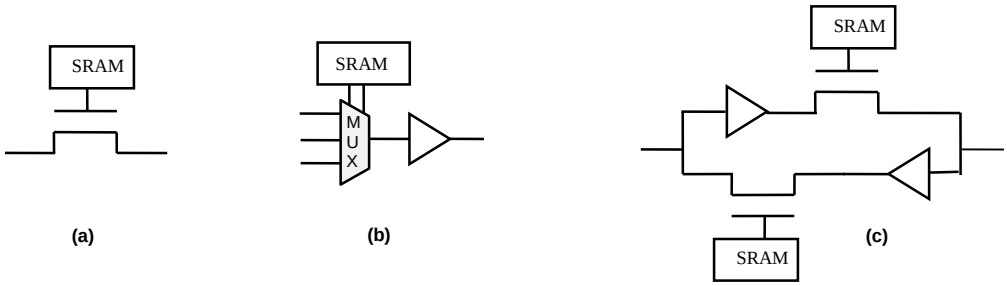


Fig. 2.12: Programmable switches (a) Unbuffered (b) Buffered uni-directional (c) Buffered bi-directional

2.1.2.1 Switch Blocks

A switch block consists of a set of switch boxes as shown in Fig. 2.11. Switch boxes consist of a set of programmable switches that connect a horizontal wire segment to a vertical wire segment. Fig. 2.12 shows programmable switches which are unbuffered, pass transistor based or buffered switches which are unidirectional or bidirectional. Commercial FPGAs use unidirectional buffered switches as they have higher performance and are more area efficient when compared to bidirectional switches [8]. Programmable switches can connect each wire in other three directions. Each wire segment spans one or more logic blocks, longer paths can be constructed by turning on the programmable switches within the switch boxes. A variety of switch block design strategies

exist and each uses a unique topology to distribute switch boxes. In academia, depending on the connection topology used they are mainly disjoint, Wilton and universal switch blocks [11][26].

2.1.2.2 Connection Blocks

Connection blocks are used to connect adjacent input and output logic block pins via routing channels.

Fig. 2.13 shows the connection of logic block input pin and the routing tracks. As shown, multiplexers are used for routing connections. The multiplexers used in routing architecture are decoded type as they are faster [22].

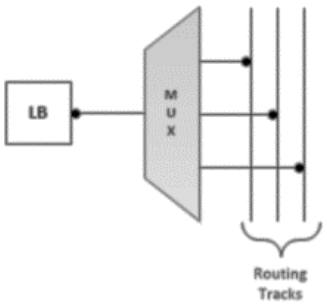


Fig. 2.13: Input connection Block

For unidirectional routing architecture the output connection block is typically merged with the switch block discussed in 2.1.2.1, Fig. 2.14 shows logic block output pin connections for bi-directional routing architectures with routing wires using shared and unshared buffer switches. Each logic block pin is connected to Fc_input number of routing wires. Buffers can also be shared; buffer sharing lowers performance but is area efficient on the other hand. No buffer sharing results in higher performance and has more area.

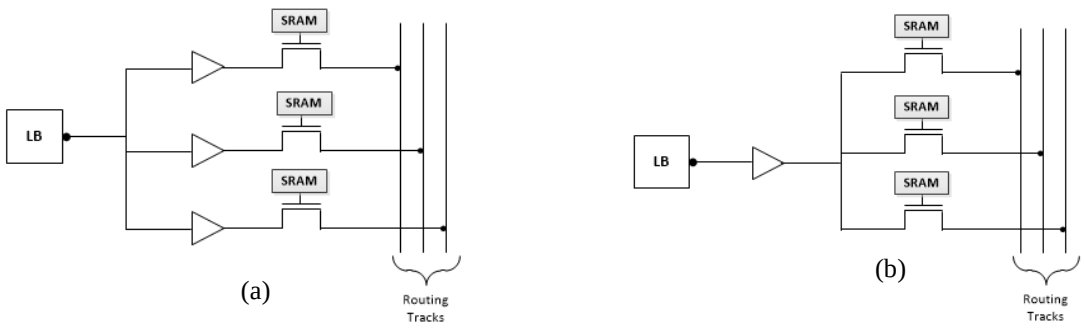


Fig. 2.14 Output Connection block (a) Unshared buffers for driving (b) Shared buffer for driving

2.2 Survey on High Level Area Estimation tools used for FPGA Based Systems

This section presents a survey of high level area estimation tools used for FPGA implementations for fast design space exploration. High Level Synthesis (HLS) tools would allow designers to enter designs at a much higher level of abstraction. These tools would take the algorithmic description of the required hardware together with certain area and performance requirements of the designer, and perform an exploration of the design space to output the hardware which meets the designer's specifications. They are developed for different input description languages. Their analysis is presented below.

FPGA-based area estimators are either used in the context of fine grained or coarse grained architectures. They either incorporate a physical model for the FPGA and estimate the area by performing actual mapping [72][75] or by using modeling equations of the FPGA functional resources [66] or by building a large database for all possible resource configurations.

Depending on the estimation methodology they can be broadly classified into two categories. Techniques which considers HW/SW Partitioning and techniques which do not consider the partition process. Area estimation for hardware/software partitioning schemes is discussed in

[66][67]. This enables maximum performance while satisfying the hardware resource constraints, designers can decide which part of the application need to be ported to hardware and which part to be executed on the processor.

Depending on the input description languages the other area estimation techniques are (C [67], SA-C [70] , System C [75], MATLAB [67][70][72][67], VHDL[66]). Most of these tools perform a transformation step to express the input description into an intermediate representation such as Control Data Flow Graph (CDFG) [66], Data Flow Graphs(DFG) [71][70], Trimran IR [67][68], VHDL AST [69] and VHDL[70] and then, the estimation process is applied on the intermediate format.

While estimating area, the routing area is considered in the estimation process for accuracy in [73] whereas in [71] it is not considered for small and medium designs as it constitutes a small fraction of the overall area and for the others the routing effect is included as a constant scale factor[75]. Also most of the above tools focus on the data path area estimation and ignore control logic; others integrate control logic and data path estimation in one tool flow [66][67][69].

Among the tools which considered HW/SW Partitioning the one presented by Nayak [68] is suitable for high level signal and image processing applications as it is developed for applications in MATLAB. It considers the routing delay and includes area for both data path and control logic.

From the techniques which do not consider partitioning the one presented by Kulkarni [70][71] in SA-C can be rated high. As the estimated error is not high and the time required for estimation is only in the order of milliseconds. Also the SA-C compiler can readily analyze the code and extract both fine grained and coarse grained parallelism. Hence, performing large number of optimizations.

The tool presented by Deng et al.[69] in MATLAB has an estimated error of only 1.87% and can be used for large designs and the time required for estimation is only in the order of microseconds as compared to minutes for a synthesis tool and therefore enables early comparisons to other approaches.

In our research work we focus on VPR[11] and COFFE [18] tool which uses minimum width transistor area model. Since minimum width transistor area is the current de facto method of estimating the area of reconfigurable fabrics, and it can be easily calculated from a set of architectural specifications. In this work, we investigate the suitability of using minimum width transistor area to directly estimate the actual implementation area of FPGA-based reconfigurable fabrics.

2.3 Minimum Width Transistor Area

Minimum Width Transistor Area has been used in several FPGA architectural studies in estimating the implementation area of proposed FPGA architectures. The minimum width transistor area model [11] defines one unit of layout area as the area required to layout the smallest transistor that contain one contact for each source and drain diffusion area as shown in Fig. 2.15. Observe that based on SCMOS deep submicron lambda based scalable rules [20][21], where lambda is half the minimum transistor length. One unit of minimum width transistor area (mwt) corresponds to 16λx13λ (208 λ^2). As the width of a transistor, x, increases, this layout area increases based on the Equation 4 [11]. As shown the model is uni-dimensional and scales linearly as a function of transistor width. The height, width and the number of metal layers that are required to achieve the predicted layout area are not considered.

$$Area_{VPR}(x) = 0.5 + 0.5x \quad (4)$$

$$Area_{COFFE\_NMOS}(x) = 0.447 + 0.128x + 0.391\sqrt{x} \quad (5)$$

$$Area_{COFFE\_CMOS}(x) = 0.518 + 0.127x + 0.428\sqrt{x} \quad (6)$$

COFFE area model discussed in [18] is based on individual transistor sizing of layouts. Its area for NMOS transistors is given by Equation 5 and for CMOS transistors is given by Equation 6. This area model has different coefficients and is based on the layout work of distinct transistors of varying sizes and produces smaller area when compared to [18]. The minimum width transistor area based on lambda based rules for NMOS only transistors corresponds to $200.93\lambda^2$ and for CMOS transistors considering N-well spacing corresponds to $223.18\lambda^2$.

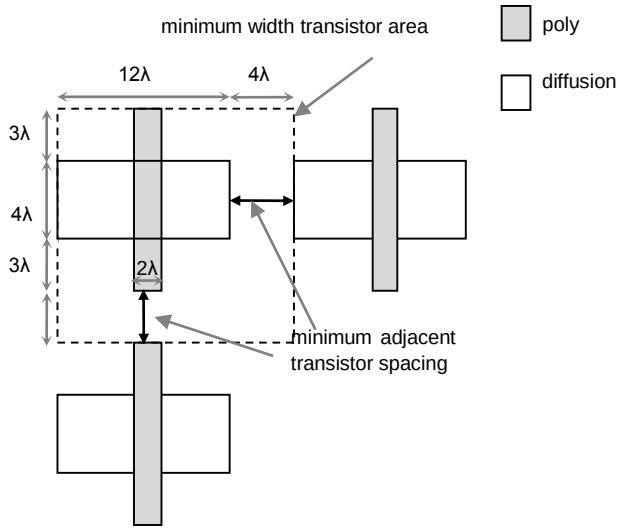


Fig. 2.15: Minimum width transistor area model

2.4 Stick Diagrams

Stick Diagrams are color coded schematic representation of a circuit at physical design level useful for planning layout and routing of the integrated circuits. The basic colors used are red for polysilicon (gate), green for n-diffusion (nMOS), brown for p-diffusion (pMOS), blue for metal 1, purple for metal 2 and similarly other metals are represented by different colors. A cross mark in

a stick diagram indicates a connection or via. Fig. 2.17 shows a detailed stick diagram representation of 2-LUT multiplexer. Stick diagram representation shows relative placement of components and all vias. It is a conventional method of estimating silicon area based on lambda rules [21]. Layout area is estimated by counting the number of tracks both horizontally and vertically. Each track pitch is defined by the width of metal wire used and the spacing between the next metal wire. If the metal width is 4λ and the minimum spacing required for the next metal wire is 4λ, then each track is 8λ. The track width varies depending on the metal used.

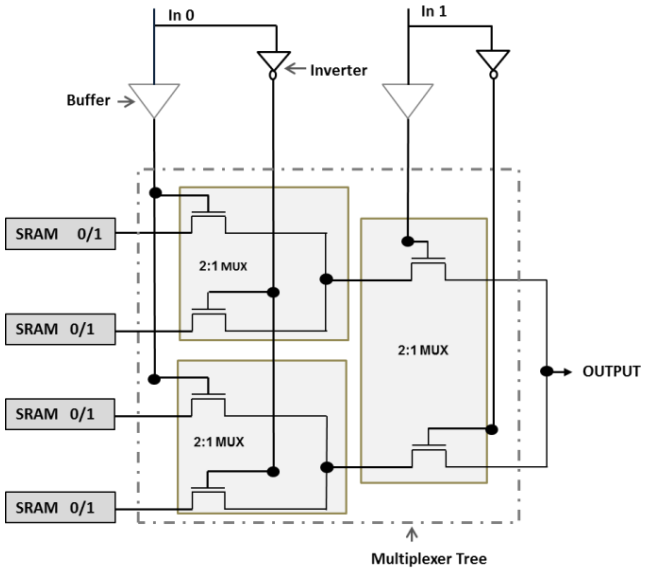


Fig. 2.16 : 2-LUT as a combination of 2:1 multiplexers

In this section we present the stick diagram representation of 2-LUT multiplexer and compare its area prediction with the minimum width transistor area model of both VPR and COFFE. The functionality and schematic of LUT is discussed in section 2.1.1.1. Here, Fig. 2.16 analyzes the LUT multiplexer tree as a combination of several 2:1 multiplexers. A 2-LUT consists of three 2:1 multiplexers as highlighted in the above figure. The multiplexer function is realized by pass

transistors, which is one of several common implementations. The 2:1 multiplexer can be laid out in different ways. Considering diffusion sharing of two transistors of 2:1 multiplexer, results in much less layout area compared with the layout of two discrete transistors. Diffusion sharing is not explicitly considered in both area models of VPR and COFFE. We consider these issues in our work and attempted to compare the stick diagram area estimation with VPR and COFFE area model. In our area estimation process, we initially use two layers of metal. Mostly, metal1 is used for internal connections and metal2 for long distance routing. We also found that using additional metal layers does not significantly reduce layout area of LUT multiplexers.

Estimating area of minimum width transistor area model to implement the pass transistor network of the 2-LUT, the minimum width transistor area model of VPR predicts $1248\lambda^2$ of layout area and COFFE predicts $1205.58\lambda^2$. This area, however, represents layout area of the layout of 6 nMOS transistors as shown in Fig. 2.16. This area estimation ignores the wiring area required to connect the transistors and also does not consider diffusion sharing.

Now measuring area from stick diagram representation, consider implementing the same schematic with the constraint that SRAM inputs comes from the left and select signals from the top, the layout area required to implement the stick diagram is shown in Fig. 2.17(a). This can be estimated using metal spacing based on the methodology outlined in [21]. As shown, the layout would occupy 2 rectangular areas of $64\lambda x24\lambda$ and $40\lambda x24\lambda$ each. Consequently, the layout area as predicted by the stick diagram is $2496\lambda^2$, which is 2x the area predicted by the minimum width transistor area.

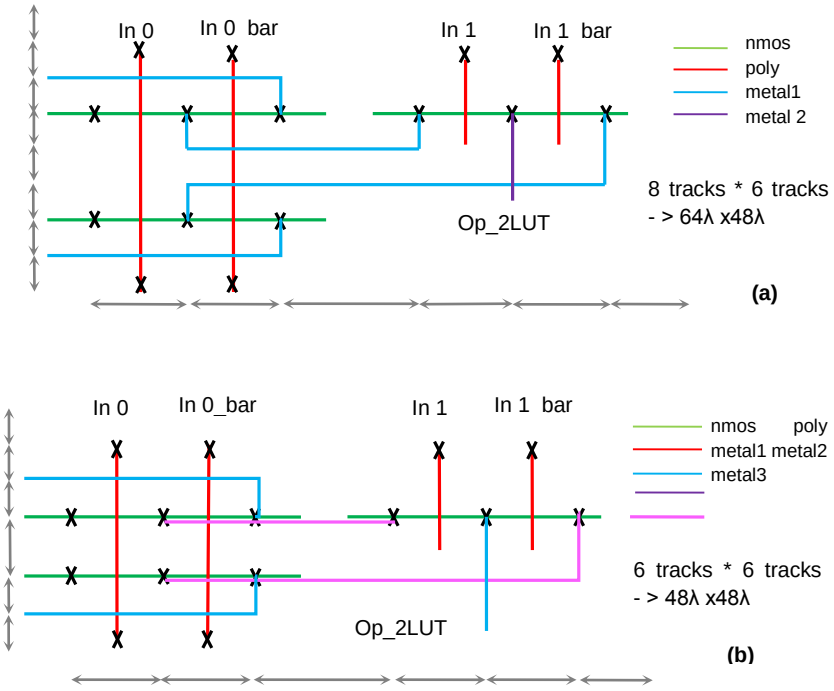


Fig. 2.17: 2 LUT multiplexer (a) Stick diagram illustration (b) Stick diagram illustration using 3 metals.

Further, one can reduce this layout area by increasing the number of metal layers employed. For example, using three layers of metal, the layout area can be reduced to $2304\lambda^2$, as shown by Fig. 2.17(b). The area, however, still represent 1.84x minimum width transistor area and adding additional metal layers does not further reduce layout area.

Consequently, the VPR and COFFE models exclusively used in FPGA architectural studies provides inconsistent results in comparison to the stick diagram model which is used in more general VLSI layouts. Therefore, there is a need to verify the accuracy of each of the three models based on the real layout results.

2.5 Summary

In this chapter, we presented the type of FPGA architecture used in our study. We also discussed the logic block architecture and presented a detail discussion on the implementation of LUTs and logic blocks clusters. Routing architecture and its components where described. We then described the VPR and COFFE theoretical area models used to estimate area. The VPR area model uses a single equation to estimate area for both NMOS and CMOS components. In contrast, the COFFE uses two separate equations to estimate area. The typical method of estimating area using stick diagrams is also discussed and compared with VPR and COFFE indicating their inconsistency in estimating area. The layout of various FPGA components examined in this work are presented in Chapter 3 and Chapter 5 . The detailed area comparisons are presented in Chapter 6 .

Chapter 3

CMOS based FPGA Components

In this chapter we present different CMOS based FPGA components. A detail study is done at logic level, transistor level design as well as layout design. Here, layout is manually done using open source VLSI layout tool, Magic [20] and layout verification of components is done using IRSIM tool [33]. The sections below discuss Inverters, Buffers, Tri-state Buffers and SRAM cells. They are widely used CMOS based FPGA components found in logic blocks and routing architectures. Full adder circuit is also considered as it is currently used in commercial FPGA logic block architectures.

3.1 Inverters

Inverter is the basic unit of digital electronics. They are used in constructing buffers, tri-state buffers, D flip-flops and SRAM cells. It is a key element of complementary logic enabled by (Complementary Metal Oxide Semiconductor) CMOS technology. Devices using CMOS technology are extensively known for low static power consumption and high noise immunity.

Inverters are made of both p-type and n-type Metal Oxide Semiconductor Field Effect Transistors (MOSFETs). An inverter function can be explained as follows, for low input voltages it outputs high voltage and for high input voltages it outputs low voltage.

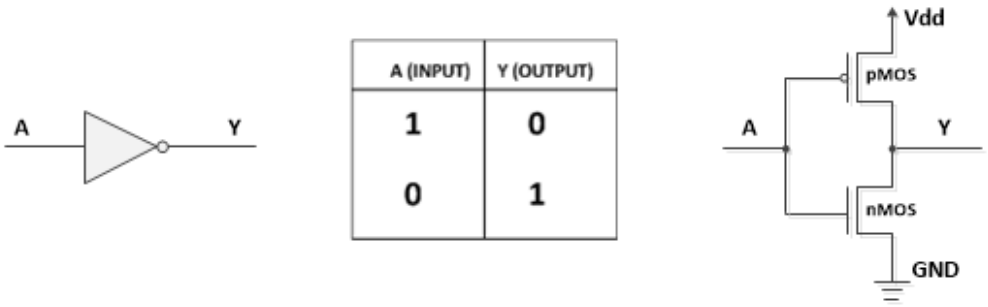


Fig. 3.1: Inverter symbol, function, circuit diagram.

When designing CMOS circuits, the drive strengths of the transistors need to be balanced since the mobility of electrons is approximately twice that of holes [28][21][28]. A pMOS transistor as same size of nMOS transistor has higher resistance. To achieve symmetrical characteristics, equal rise and fall delays, the pMOS transistor should be *n* times wider than the nMOS transistor. In Fig. 3.2, Cadence is used to simulate Inverter transfer characteristics for varying sizes of pMOS transistor and the value of *n* is determined. We notice that V_{in}=Vdd/2, equal rise and fall is achieved when n falls between 2 and 4. Depending on the process used, the size of pMOS transistor is chosen. In our experimentation we consider the size of pMOS to be twice of nMOS transistor for equal rise and fall delays.

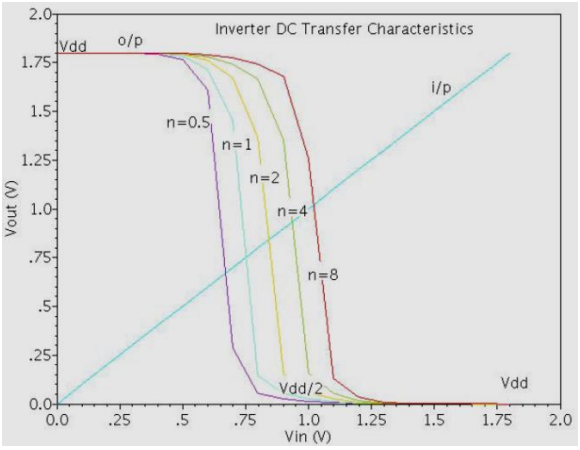


Fig. 3.2: Inverter DC characteristics for different sizing of pMOS transistor.

Here, a unit inverter is one whose nMOS transistor is of unit size and pMOS transistor is twice of unit size. The size of a transistor indicates the width of p-type or n-type channel. For example, an inverter of drive strength 2x is one whose nMOS transistor is two times the width of unit size transistor and pMOS transistor is four times that of unit size transistor. As the size of transistor increases the layout of transistor also increases, but this can be made compact by folding the transistor into multiple fingers.

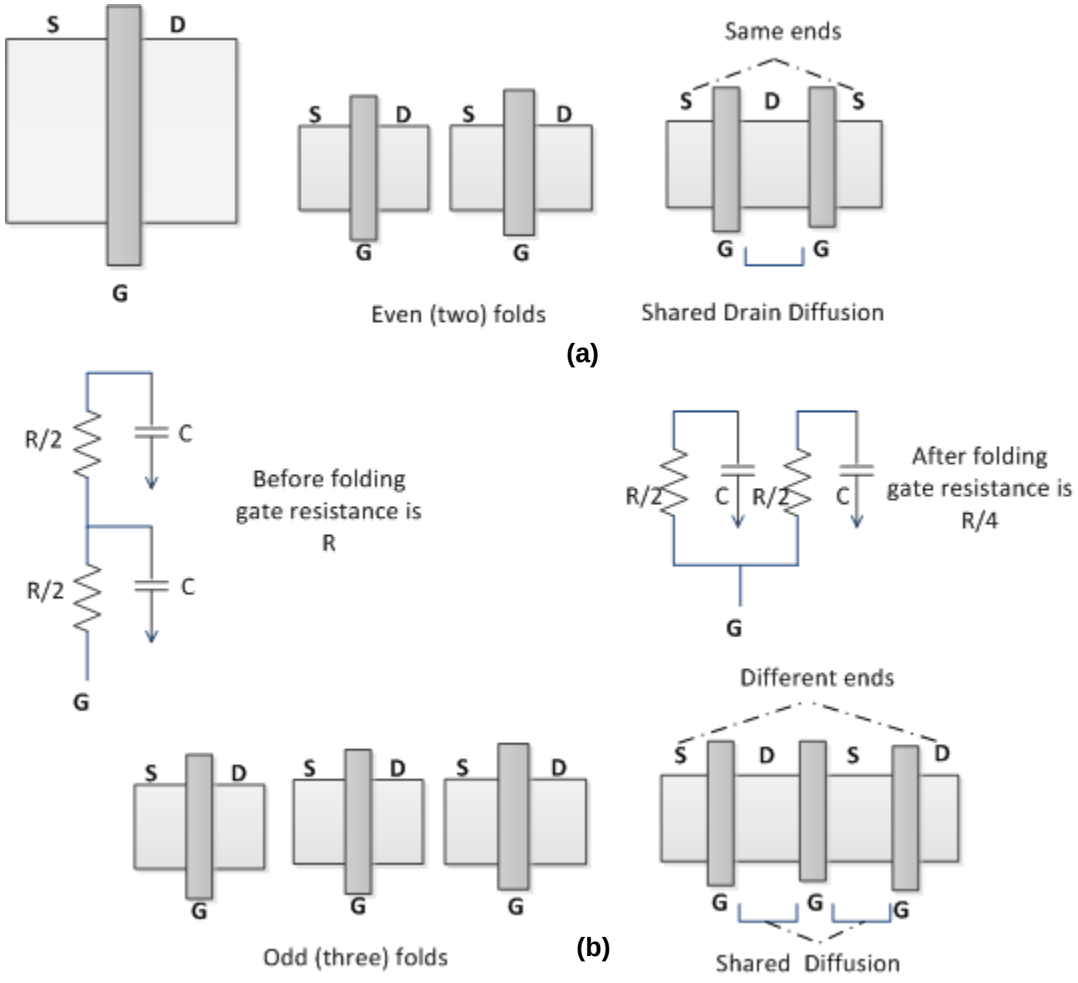


Fig. 3.3: Folding of transistors

Folding or also known as fingering is a technique in which a wider transistor is equally divided into smaller portions and connected in parallel and placed such that the diffusion regions are

shared. This essentially decreases the height of the wide transistor to be closer to square layouts rather have layouts with higher aspect ratio. Folding results in reduced diffusion capacitance, improved delay and reduced transistor active area. Due to folding, the gate (polysilicon) resistance is also decreased. This also enables the transistor on and off faster [29]. The gate capacitance (gate to source) at the input does not affect the delay because it is neither charged nor discharged during an output transition [21]. As an example of folding technique, Fig. 3.3(a) above shows when transistors are split into even (two) numbers, the ends are same (S, S). Here, the gate resistance is decreased from R to R/4, enabling it to drive faster. As shown in Fig. 3.3 (b), when transistors are split into odd portions the two ends are different (S, D), this leads to more connectivity and more drain capacitance [29] [30]. Therefore, in this case even number of folds is preferred.

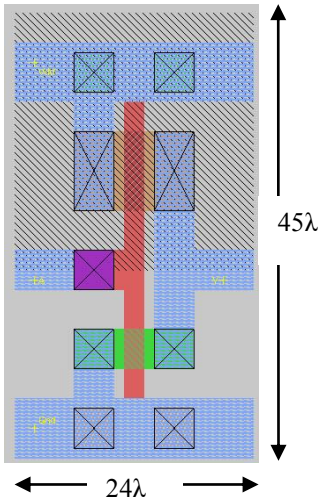


Fig. 3.4 : Inverter Layout (a) unit inverter

In this work we consider layout verification with:

- Different number of even folds.
- Diffusion sharing between adjacent inverter in multistage designs to reduce area.

- To the resultant layout additional 4λ is added to the width and height for minimum adjacent spacing between components.

In particular, Fig. 3.4 shows the layout of unit inverter laid using Magic, an open source layout tool. This results in layout area of 24λ x 45λ which is 1080 λ^2. This layout area includes both the power and the ground, represented by metal1, 6λ wide and also includes n-well. The nwell surrounds pdiffusion by minimum 6λ in all directions. This layout area does not include additional 4λ adjacent spacing between other components. However, Fig. 3.5 presents the layout by including minimum adjacent spacing and ignores power and ground. The reason behind this is power and ground can be shared by multiple components and by including this would result in an overestimation of area. Also, both VPR and COFFE area model do not consider power and ground in their area model but considers minimum adjacent spacing. Hence, we have included minimum adjacent spacing when estimating area. Fig. 3.5 results in layout area of 28λ x 37λ which is 1036 λ^2.

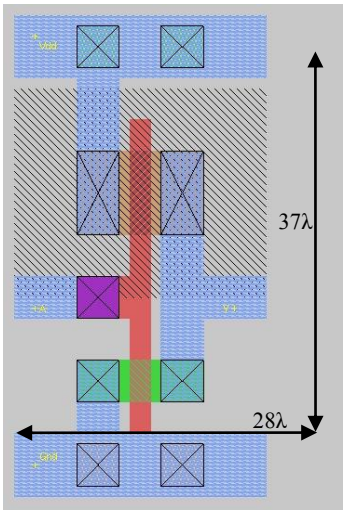


Fig. 3.5: Unit-Inverter layout considering minimum adjacent spacing

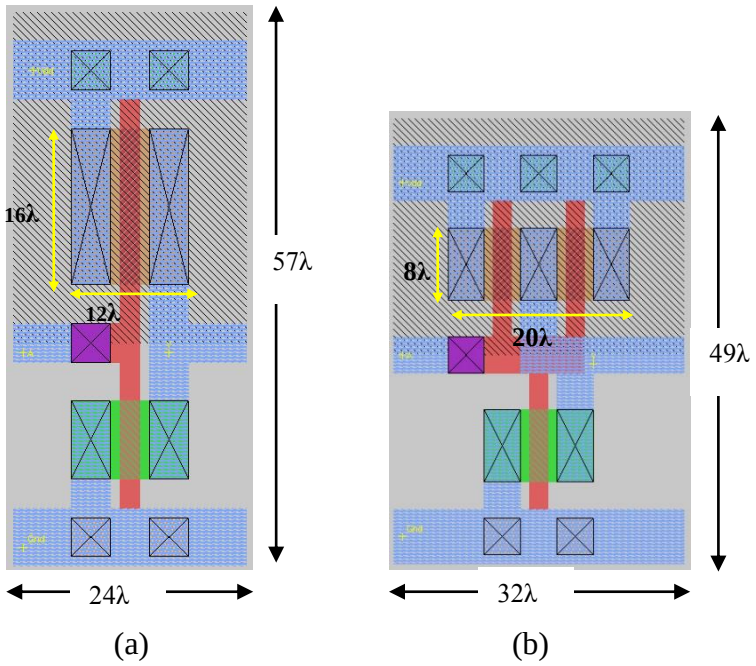


Fig. 3.6: Inverter Layout (a) 2x inverter unfolded, 1368λ^2 (b) 2x inverters folded, 1568λ^2

The layout of 2x inverter is shown in Fig. 3.6. As discussed, a 2x inverter is one whose nMOS transistor is two times of unit size transistor and thereby the pMOS transistor would be four times the unit size transistor, pMOS is twice the width of nMOS. Fig. 3.6(a) shows layout of 2x inverter without folding this resulted in total layout area of 1368λ^2 and Fig. 3.6(b) shows layout of 2x inverter, the pMOS transistor is with two folds this resulted in total layout area of 1568λ^2. Compare 2x inverter layout area without folding and with folding technique. Note, folding technique reduces the pMOS active area from 192λ to 160λ but overall layout area increases by 200λ^2. Thus, folding could be used for very wide transistors to result in compact layout. For area comparison with VPR and COFFE, we use unfolded layout area as this resulted in reduced area. This layout area considering minimum adjacent spacing and ignoring power and ground is shown in Fig. 3.7. This corresponds to a layout area of 1372λ^2. Layout verification is done using IRSIM tool [33]. IRSIM

is a switch level simulator used as an interface with Magic. Fig. 3.7(b) shows the simulation of inverter layout. When input, A is 0 or low the output Y is 1 or high, which describes the functionality of inverter.

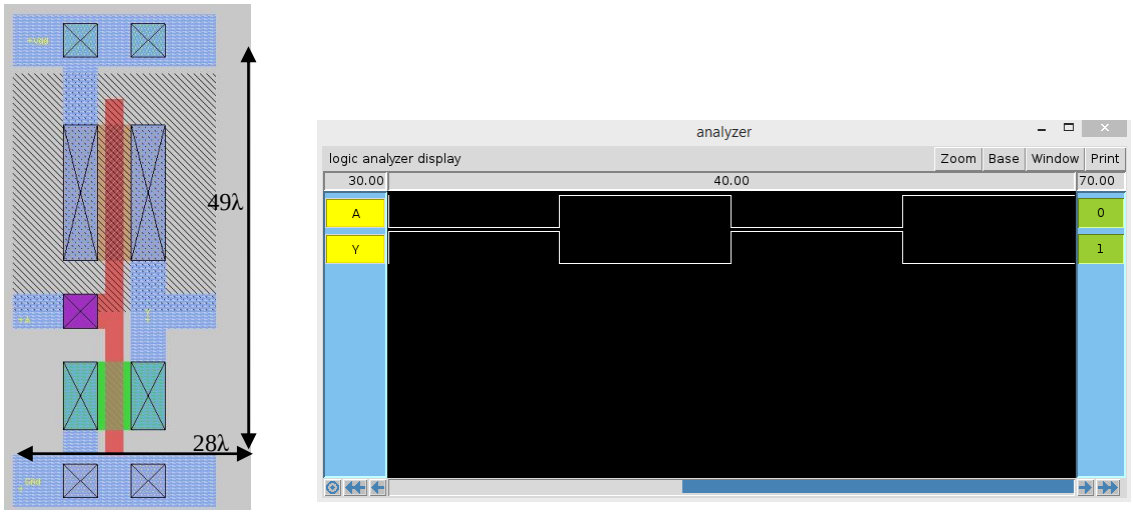


Fig. 3.7: 2x inverter (a) layout considering minimum adjacent component spacing and ignoring power and ground (b) IRSIM Simulation for Inverter

3.2 Buffers

In FPGA design, buffers are widely used in logic block architecture and routing architecture. Buffers are formed by a chain of inverters of gradually increasing sizes and are used to drive loads with larger capacitances, this improves performance. Transistor sizing of buffers is done to minimize area-delay product of an FPGA [18], larger transistors have more current handling capability. Fig. 3.8 shows a buffer formed by a chain of inverters. Buffers can either be non-inverting or inverting. The number of stages, inverters N in the chain determines the delay characteristics. Delay is defined by two factors, a constant component called as *parasitic delay*, P and a variable component proportional to the load present at the output called *effort delay or stage*

effort, F, given by equation 7. Parasitic delay depends on internal capacitance of the gate and does not change with transistor size. The stage effort depends on two parts, *logic effort,* g and *electrical effort or fanout,* h. Logic effort is a measure of how much input capacitance a gate requires to match a unit inverter and electrical effort is the ratio of output capacitance of the load to the input capacitance.

$$D_{\min} = NF^{1/N} + P \text{ [21]} \quad (7)$$

$F = \prod g_i h_i$ where F is path effort, g is logical effort and h is electrical effort C_{out}/C_{in} (8)

The logical effort and parasitic delay for an inverter is 1[21].

Stage effort at each stage is $F^{1/N}$ which mainly depends on the output load C_L to be driven and input load. In Fig. 3.8, stage effort is represented by the variable k.

The minimum number of stages N can be calculated as $\log_4 F$ ($4 = F^{1/N}$).

Theoretical calculations suggest stage effort of 4 is a good choice for minimum delay and with 15% additional delay it could vary between 2.7 to 6 [21].

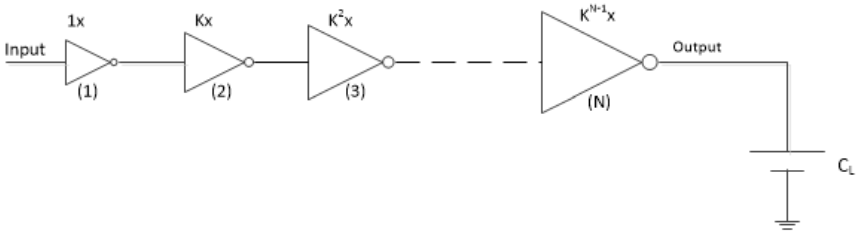


Fig. 3.8: Buffer formed by a chain of inverters

In our layout work we use the same sizes of buffers as suggested by [11]. In logic block architecture and connection blocks a buffer of two stages is used with drive strength of four times the minimum. FPGA architectures most commonly used typically buffer sizes of 4x and 16x [11]. Fig. 3.9 shows a 4x buffer which maintains a stage ratio of four as discussed above. To drive smaller loads as in LUT a two stage buffer is used as shown in Fig. 3.10 with drive strength of two times the minimum.

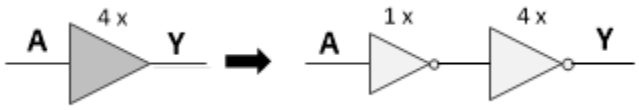


Fig. 3.9: 4x buffer to drive large loads

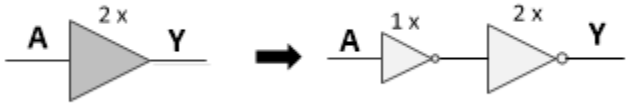
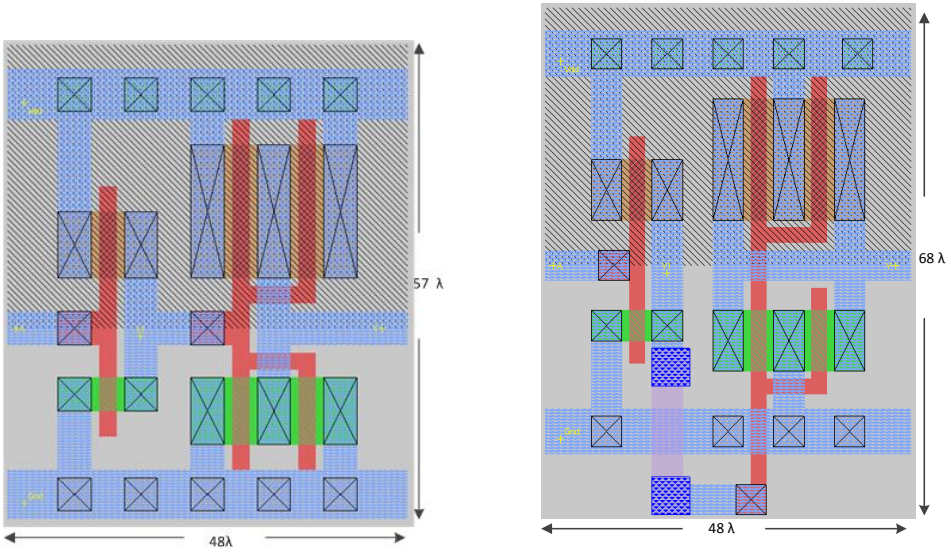


Fig. 3.10: 2x buffer to drive small loads

3.2.1 **4x Buffer**

Here we discuss different design options for 4x buffer. Fig. 3.11 shows two layouts of 4x buffer, where the second stage nMOS and pMOS transistors have two folds. Both layouts produce similar output as shown in Fig. 3.12. The output capacitance in Fig. 3.11(a) is less when compared to Fig. 3.11(b), as it is folded such that the output is connected to only one contact, resulting in better performance. Also, the layout area of Fig. 3.11(a) is $528\lambda^2$ less when compare to Fig. 3.11(b).

Hence, the layout in Fig. 3.11(a) is considered to be a better design when compared to Fig. 3.11(b) with less output capacitance and reduced area. Fig. 3.12 shows the simulation output of 4x buffer. Note, 4x buffer is a non-inverting buffer, the output Y is same as input A.



(a) Area is 2736 λ^2 (b) Area is 3264 λ^2

Fig. 3.11: Layout of 4x Buffer, second stage transistors are with two folds (a) Good design (b) Bad design resulting in slower gate as output is connected to two contacts.

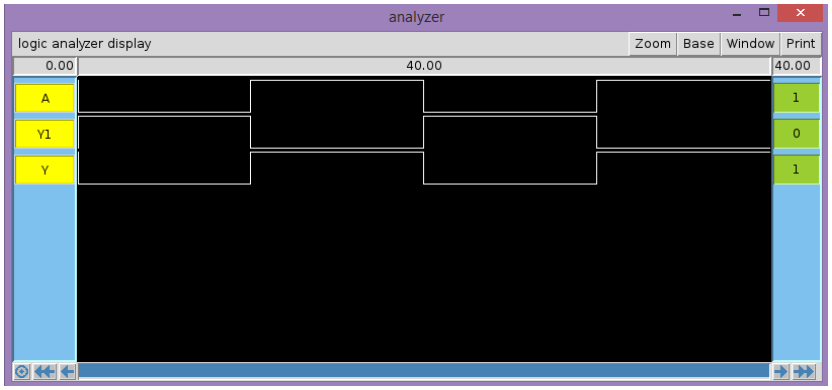
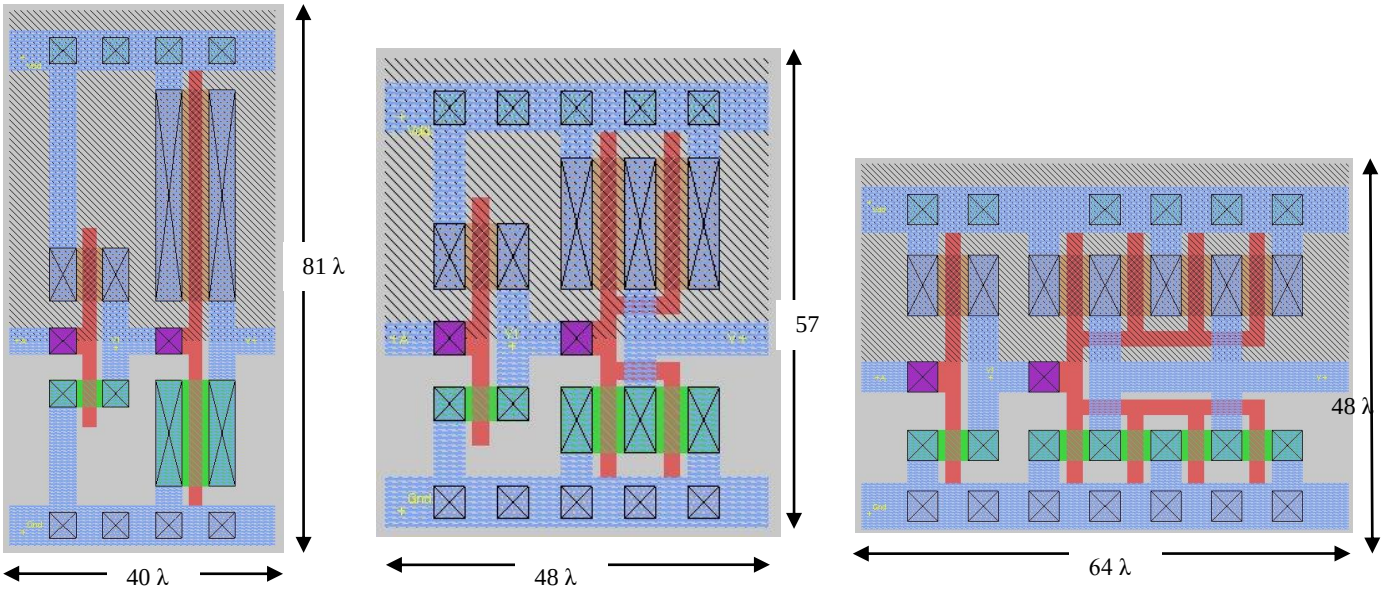


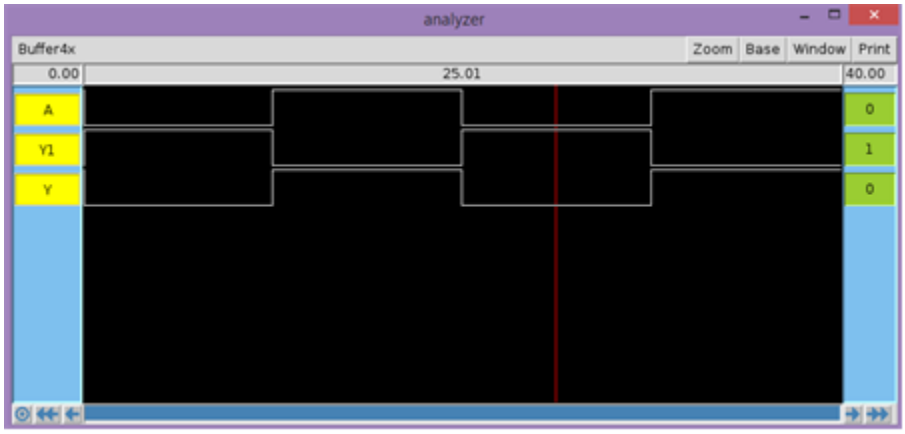
Fig. 3.12: IRSIM Simulation for 4x Buffer

3.2.1.1 Effect of folding on layout area

In this sub section we present layouts of 4x buffer with no folds and with two folds and four folds and compare their areas without diffusion sharing. Fig. 3.13 shows the different layouts of 4x buffers and its simulation. Fig. 3.13(a) shows 4x buffer without folding which resulted in layout area of $3240\lambda^2$. Fig. 3.13(b) shows the layout of 4x buffer in which the second stage nMOS and pMOS transistors have two folds this resulted in layout area of $2736\lambda^2$ and Fig. 3.13(c) shows the 4x buffer with four folds this resulted in area of $3072\ \lambda^2$. Comparing layout areas of 4x buffer with and without folding. Note that, layout area in which the second stage nMOS and pMOS transistor have two folds resulted in minimum area. Observe that, the number of folds of a transistor may either decrease or increase the layout area.



(a) Area is $3240\lambda^2$ (b) Area is $2736\ \lambda^2$ (c) Area is $3072\ \lambda^2$



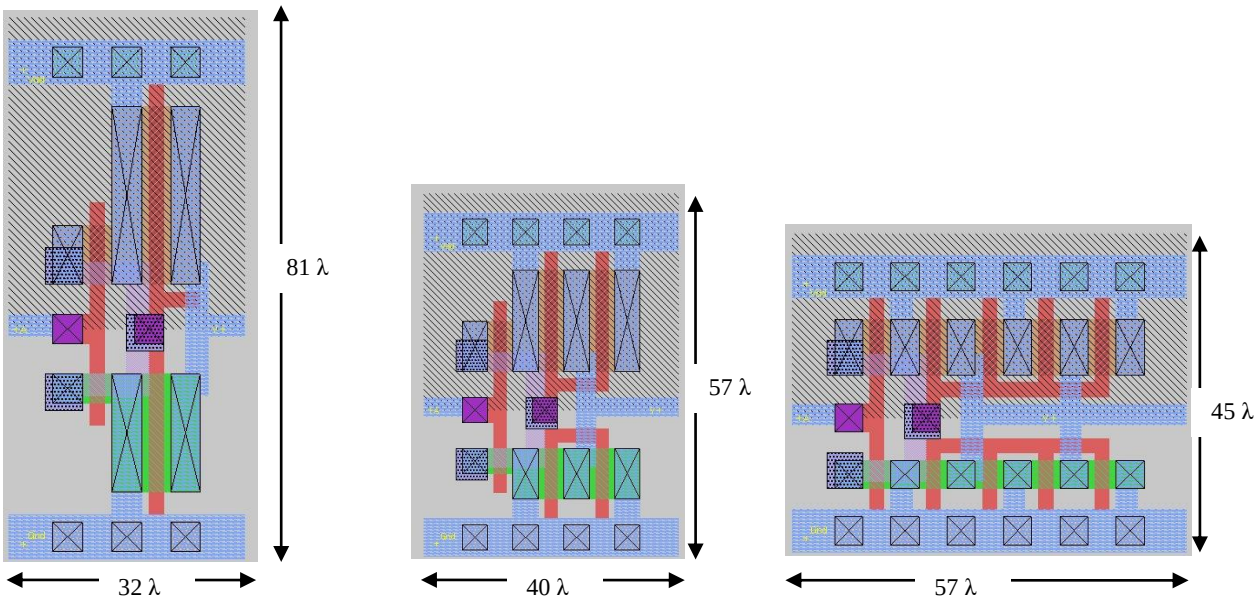
(d)

Fig. 3.13: 4x Buffer layout (a) no folds (b) two folds (c) four folds (d) IRSIM simulation

3.2.1.2 Effect of diffusion sharing and number of folds on layout area

In this subsection we observe the effect of diffusion sharing and folding on layout area. Fig. 3.14 shows the layout of 4x buffer with diffusion sharing with no folds, two folds and three folds. Note that, diffusion sharing is done between the first stage of inverter and second stage of inverter, among sources of nMOS and pMOS transistors. Comparing layouts of Fig. 3.14 and Fig. 3.13, note that diffusion sharing reduces the width of layout area for all the three layouts by 8λ. Fig. 3.14(b) gives the minimum layout area for 4x buffer with diffusion sharing and two folds.

For area comparison with VPR and COFFE we consider 4x buffer layout area with diffusion sharing and two folds as this resulted in minimum area. Fig. 3.15 shows the layout area considering minimum adjacent component spacing which results in area $2112\lambda^2$.



(a) Area is $2592\lambda^2$ (b) Area is $2280\lambda^2$ (c) Area is $2565\lambda^2$

Fig. 3.14: Layout of 4x buffer with diffusion sharing (a) no folds (b) two folds (c) four folds

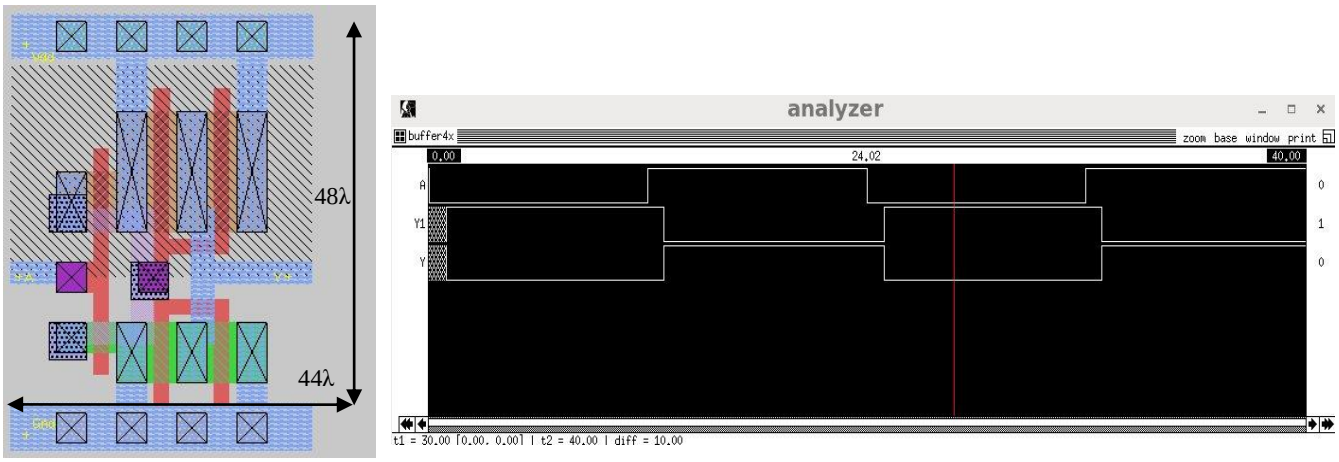


Fig. 3.15: 4x buffer (a) layout with minimum adjacent spacing (b) simulation

3.2.2 **Multistage Buffer**

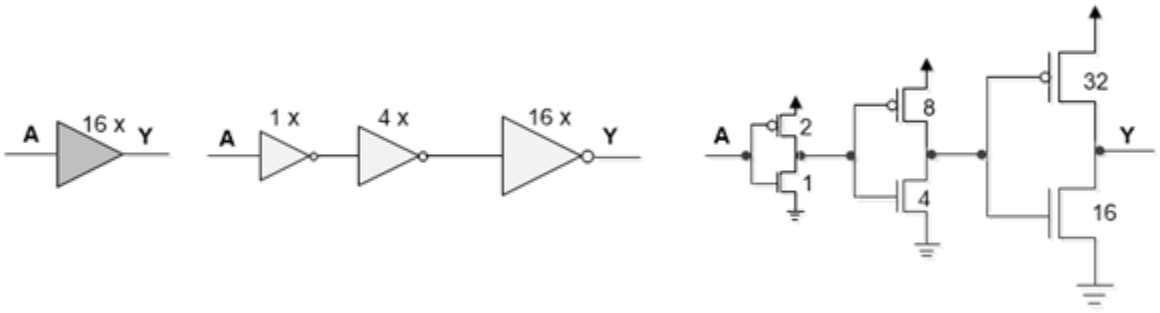
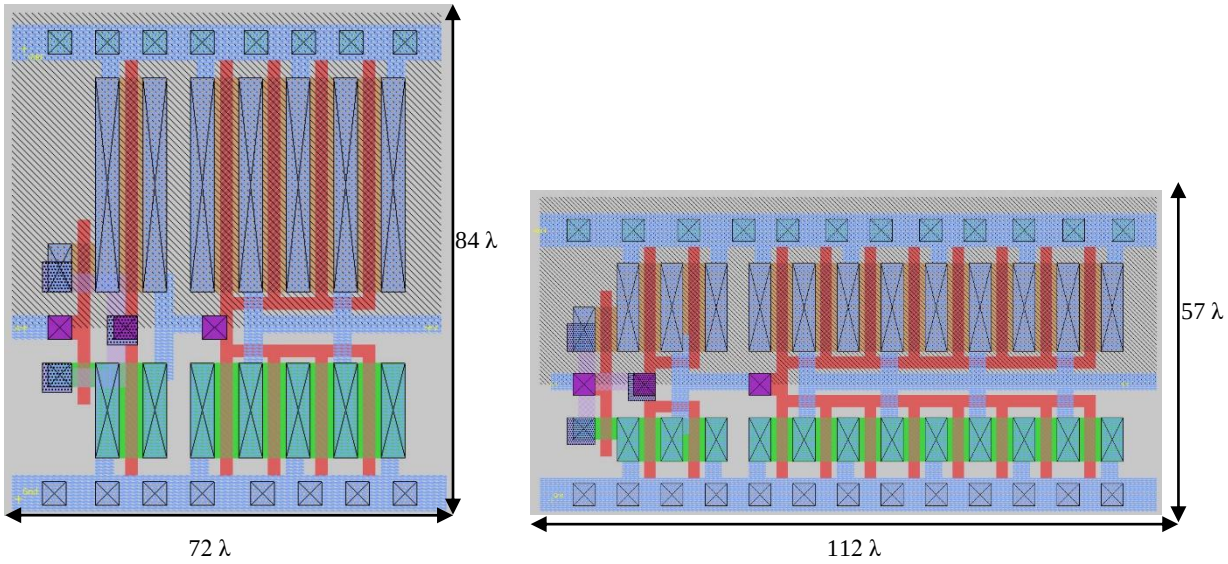


Fig. 3.16: Multistage Buffer 16x of stage ratio four

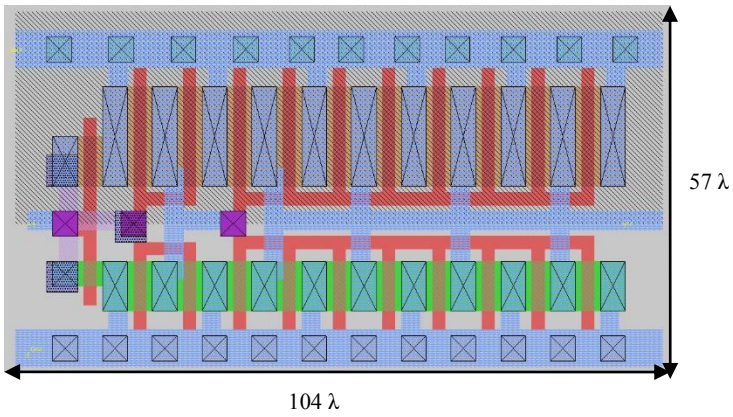
A 16x multi stage buffer, formed by a chain of inverters is shown in Fig. 3.16. The layout of multistage buffer considers diffusion sharing and optimal number of folds to achieve minimum area. In the previous section it is seen that layouts with diffusion sharing results in reduced area. Here we present two different cases of diffusion sharing. One with diffusion sharing between 1x and 4x inverter and the second case with diffusion sharing done between 1x and 4x and also between 4x and 16x.

Fig. 3.17 shows two different layouts for first case. Note that, diffusion sharing is done between the 1x inverter and 4x inverter, among sources of both nMOS and pMOS transistors, resulting in a compact layout. This is achieved by flipping the 1x inverter so that both source of 1x and 4x can share diffusion. Fig. 3.17(a) shows layout with diffusion sharing, where second stage transistor has no folds and Fig. 3.17(b) shows layout with diffusion sharing, where second stage transistor has two folds. Fig. 3.17(a) with no folds resulted in minimum area. On the other side when diffusion sharing is done at both stages, transistor with two folds resulted in minimum area.



(a) Area is 6048 λ^2 (b) Area is 6384 λ^2

Fig. 3.17 : 16x multistage buffer with diffusion sharing (a) between first and second stage inverter with no folds (b) between first and second stage inverter with two folds



Area is 5928 λ^2

Fig. 3.18: 16x multistage buffer with diffusion sharing at all stages

Fig. 3.18 shows the layout with diffusion sharing for second case, where diffusion sharing is done between 1x and 4x inverter and 4x and 16x inverter. Note, diffusion sharing is not possible

between 4x and 16x inverter of Fig. 3.17(a), as the end of 4x inverter is drain terminal and the start of 16x inverter is source terminal.

For comparison with VPR and COFFE, Fig. 3.19 shows the resultant layout area for multistage buffer with adjacent component spacing and ignoring power and ground. This resulted in area of $5292\lambda^2$.

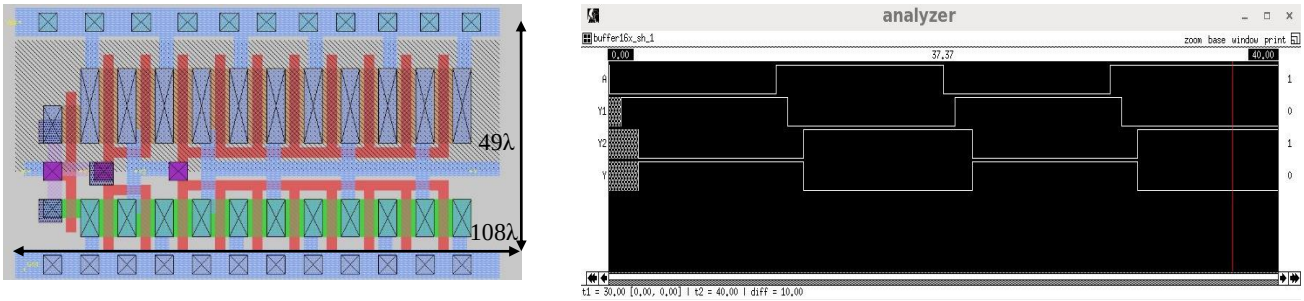


Fig. 3.19: Multistage buffer (a) layout showing minimum adjacent component spacing (b) simulation

3.2.3 **Tristate Buffers**

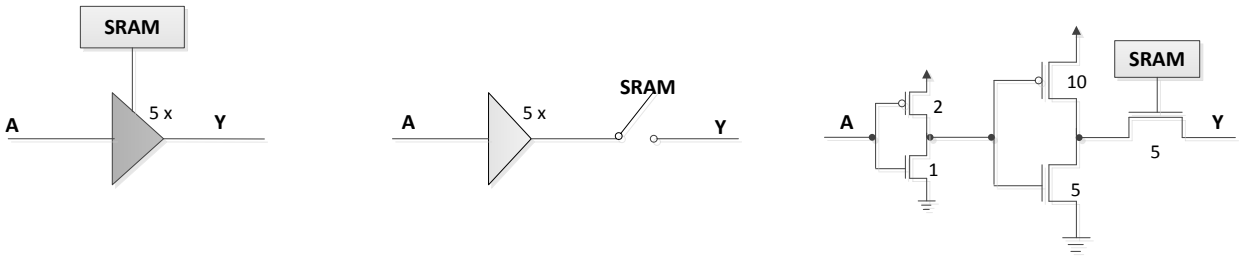


Fig. 3.20: Tristate Buffer 5x minimum size

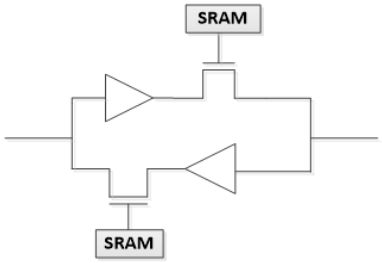


Fig. 3.21: Bi-directional paths using tri-state buffers

Tristate buffers are also used to drive large loads; they function similar to buffer but they have an additional input to control as a switch. Fig. 3.20 shows the schematic and circuit diagram of tristate buffer. A tristate buffer is a buffer with an additional pass transistor. Switching is enabled by pass transistor with SRAM cell. Note, that programmable interconnect in an FPGA is usually based on tristate routing switches [34]. They are used to connect logic block output pin with routing tracks. Tristate buffers and pass transistors are an example of such type of switches. It is experimentally suggested that best area-delay is achieved when all tristate buffers are five times the minimum size [11]. Tristate buffers are also used to create bi-directional paths in routing tracks when connected back to back as shown in Fig. 3.21. Buffer sharing is also possible in connection blocks as discussed in section 2.1.2.2. Since tristate buffers are not used in modern uni-directional routing architecture their layout is not investigated in this work.

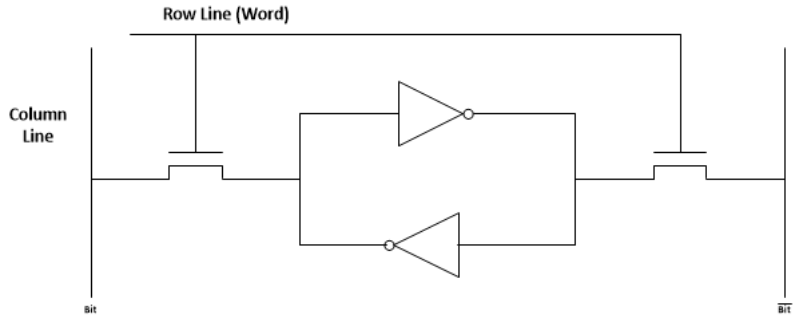
3.3 SRAM

Static Random Access Memory (SRAM) cells are used as memory cells which enable data to be read or written and hold data as long as power is applied. SRAM is important for the reprogramability of FPGAs. They are preferably used when compared to other memory elements

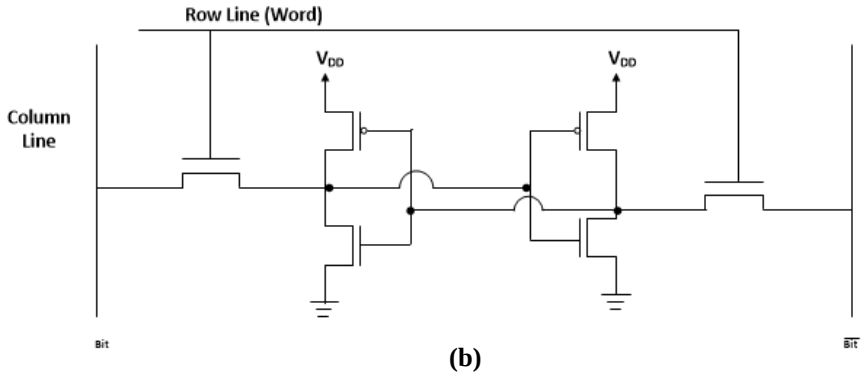
in FPGA architecture as they are easily compatible with CMOS processes and more dense when compared with other memory devices like flip flops.

Fig. 3.22 shows the schematic and circuit diagram of SRAM cell. As shown an SRAM cell is commonly constructed from 6 transistors containing two inverters connected in a feedback loop which holds the state and two access transistors to read or write the state. The bit lines are used to transfer data for both read and write operations. Access to the cell is possible via the word line.

Fig. 3.23 shows the layout of 6 transistor SRAM cell. Fig. 3.23 (a) shows the layout in which all polysilicon (gate) are vertical which is usually preferred for deep submicron rules, this gives layout area of $2173\lambda^2$ on the other hand Fig. 3.23(b) uses both horizontal and vertical polysilicon which results in area of $1881\lambda^2$. On the other hand, VPR estimates area as $1248\lambda^2$ and COFEE estimates area as $1339.08\lambda^2$.

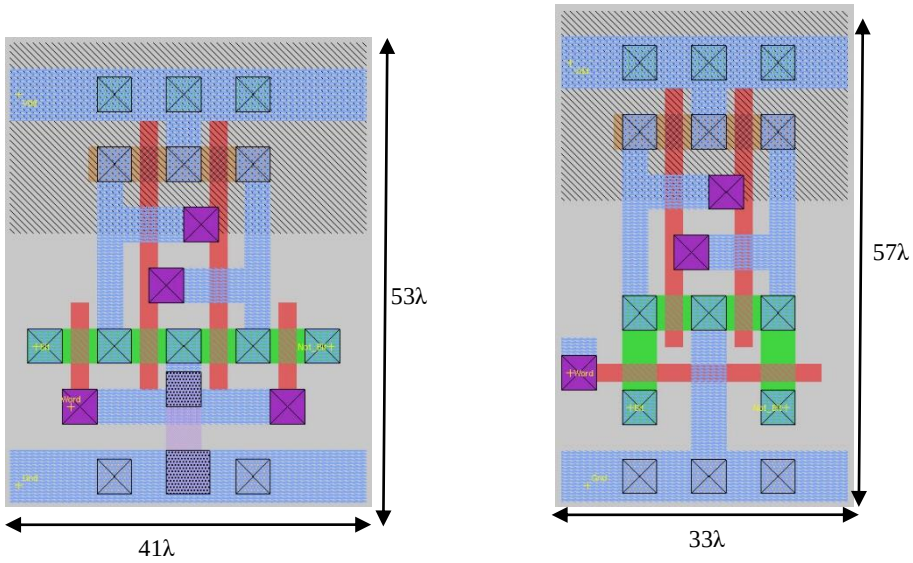


(a)



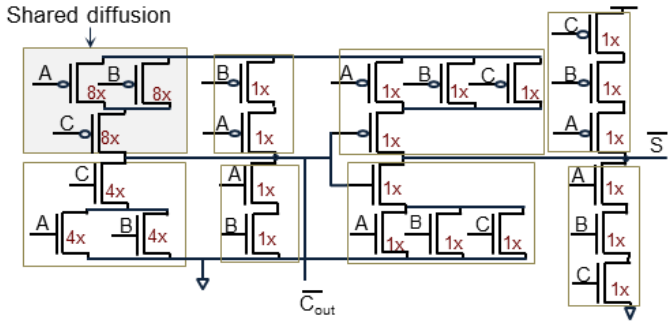
(b)

Fig. 3.22: SRAM (a) schematic (b) circuit diagram

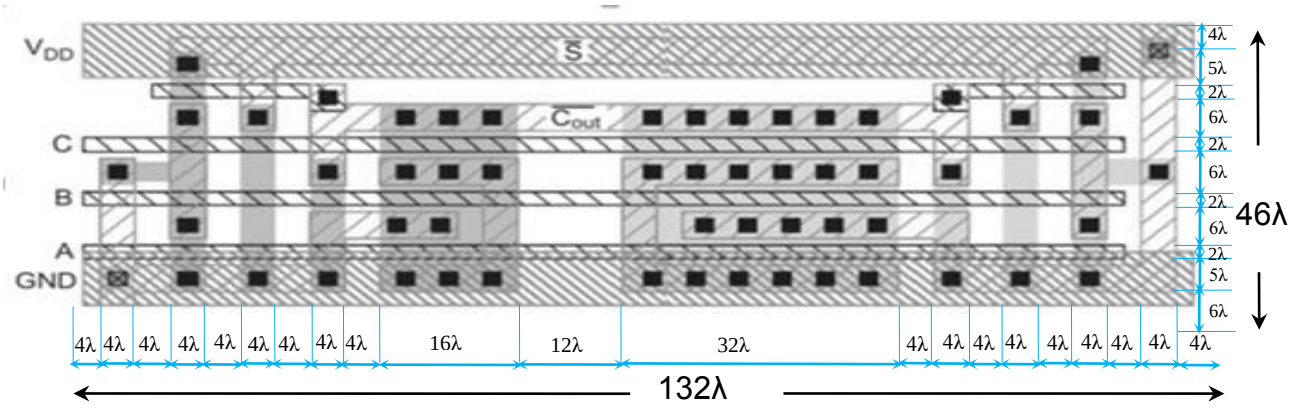


(a) Area is $2173\lambda^2$ (b) Area is $1881\lambda^2$

Fig. 3.23: 6T SRAM Layout



(a)



(b)

Fig. 3.24 : Full adder (a) schematic (b) layout [21]

3.4 Full adder

Full adder circuits are integrated in FPGA logic block architecture to increase performance. However, FPGA tile architecture of VPR and COFFE do not include full adders in their logic blocks, but commercial FPGAs include specific logic components such as full adders in their logic blocks to increase performance. Full adder is reviewed for area analysis. Fig. 3.24 shows the schematic and layout of full adder as described in [21]. Full Adder circuit has a different circuit topology when compared with other components. It is important to note that unlike the recursively Y-connected topology of multiplexers, the full adder circuit as shown in Fig. 3.24(a), contains exclusively parallel and in-series connected transistors. These transistors can extensively employ diffusion sharing in order to minimize their layout area. Fig. 3.24(b) shows the layout of full adder. The layout area estimated using deep submicron rules is $6072\lambda^2$ whereas the area estimated by VPR is given by $8112\lambda^2$ and area predicted by COFFE is given by $6888.96\lambda^2$.

3.5 Summary

In this chapter, we discussed the different CMOS based FPGA components commonly used in a FPGA tile. It presents a compact layout and layout area for buffers and full adders.

Detail area analysis and comparison for CMOS based FPGA components are presented in Chapter 6 .

Chapter 4

Pass Transistor based FPGA Components

Part I –Active Area Modeling

This chapter presents an in depth discussion on pass transistor based components. Here, we review pass transistor based multiplexers. FPGA architecture consists of large number of multiplexers in logic blocks as well as programmable routing resources. The multiplexers are either encoded or decoded type. Encoded multiplexers are used to build LUTs. The multiplexer tree of LUT is fully encoded and is realized by pass transistors using nMOS transistors. The decoded multiplexers are mainly used as routing multiplexers, as they offer better area-delay product in routing switches than encoded multiplexers[22][24]. Decoded multiplexers are also constructed from pass transistors. This chapter describes the design of multiplexers to achieve minimum active area.

A detail discussion is presented below starting from 2:1 active area minimization of multiplexers followed by a discussion on the active area minimization for larger encoded and decoded multiplexers.

4.1 Multiplexer

A multiplexer is a data selector, with 2^n inputs has n select lines. Fig. 4.1 shows the symbol of 4:1 multiplexer. Fig. 4.2(a) shows a 4:1 encoded multiplexer implemented as a binary tree using pass transistors and Fig. 4.2(b) shows 4:1 decoded multiplexer implemented by pass transistors.

Alternatively, both encoded and decoded multiplexers can be implemented using transmission gates instead of pass transistors [25].

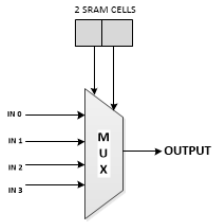


Fig. 4.1: 4:1Multiplexer symbol

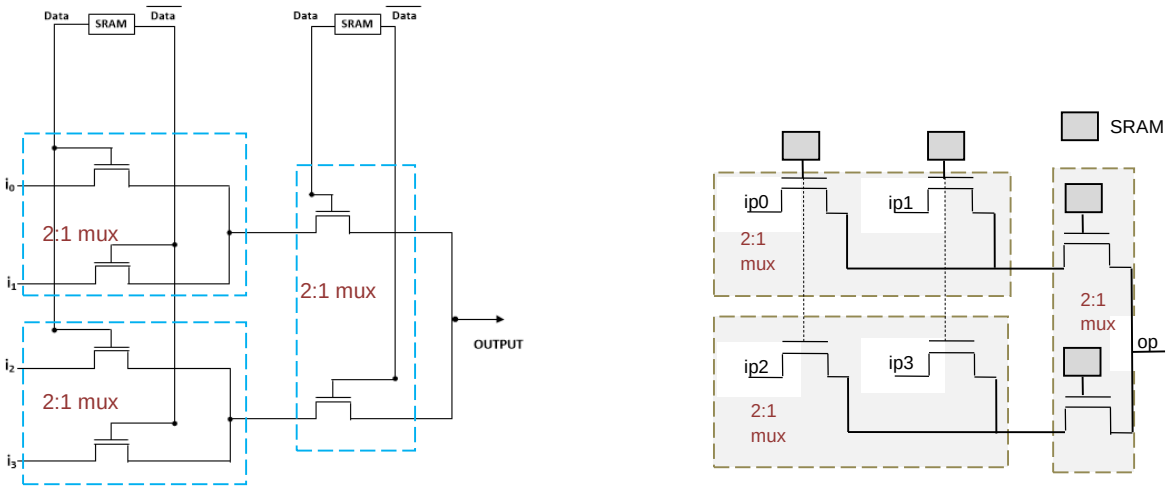


Fig. 4.2 : 4:1 Multiplexer (a) Encoded (b) Decoded

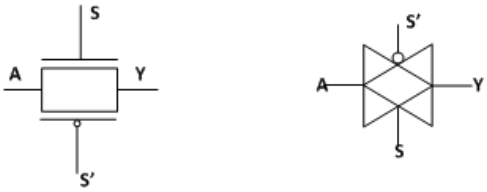


Fig. 4.3: Transmission gate

Transmission gates are made of both nMOS and pMOS transistors connected in parallel. Fig. 4.3 shows the symbol of transmission gate, the drain and source terminals of each FET transistor

are connected together. Note that, the functionality of the transmission gate is similar to tristate buffers, the gate terminals of FET transistors, S and S' in transmission gate acts as control signals.

In our study, we only consider pass transistor based implementation of multiplexers as they are currently the most commonly used multiplexer structures in commercial FPGAs [25], and they consume significantly less layout area than transmission gate based multiplexers.

4.2 2:1 Multiplexer

Observe that, as shown in Fig. 4.2, both 4:1 encoded and decoded multiplexers are a combination of three 2:1 multiplexers. Larger multiplexers are constructed from small multiplexers. For example, an 8:1 multiplexer is built from two 4:1 multiplexers. Consequently, larger multiplexers are also a combination of several 2:1 multiplexers. Hence, the 2:1 multiplexer forms the primary building block for both encoded and decoded multiplexers of all sizes. In this section, a detailed analysis on the layout strategy of 2:1 multiplexer is presented.

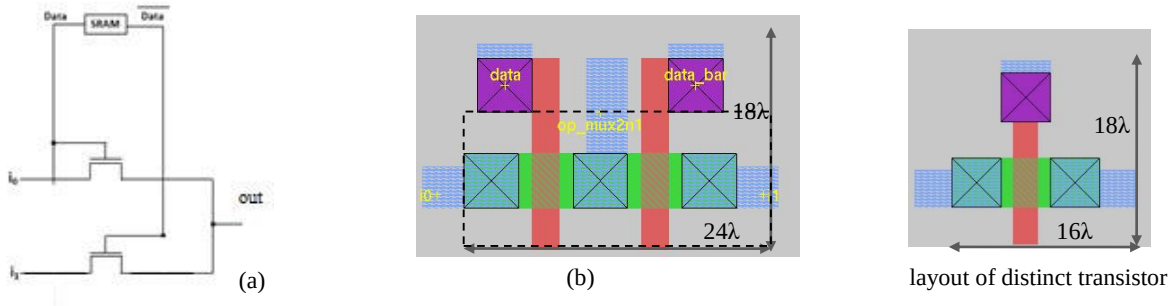
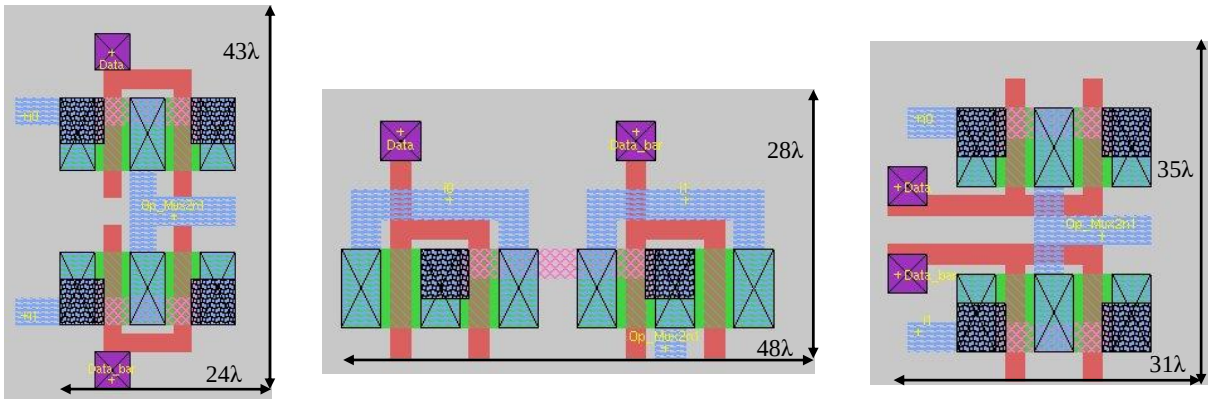


Fig. 4.4: 2:1 multiplexer (a) schematic (b) layout with effective width 1x, area is $432\lambda^2$

The 2:1 multiplexer is built from two transistors as shown in Fig. 4.4(a). Given any transistor size, the layout area of 2:1 multiplexer changes as a function of diffusion sharing and transistor folding. Here we present different layouts of 2:1 multiplexers to motivate the need to

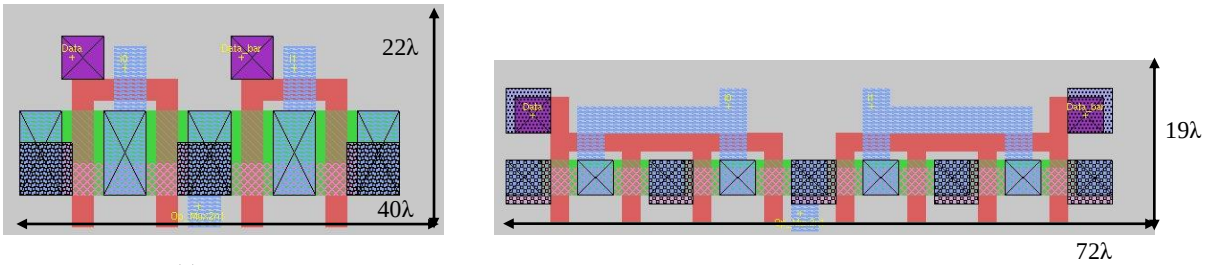
find a systematic method to find the minimum layout area of 2:1 multiplexers based on diffusion sharing and transistor folding. Fig. 4.4(b) shows the layout of 2:1 multiplexer of effective transistor width 1, with diffusion sharing of two transistors instead of two distinct transistors. The layout area of 2:1 multiplexer results in $432\lambda^2$, which is smaller than the layout area $576\lambda^2$, of two distinct transistors.



(a) (b) (c)

Fig. 4.5: Different layouts of 2:1 multiplexers of size 4x with 2 folds and without diffusion sharing(a) area is 24λ x 43λ ($1032\lambda^2$) (b) area is 48λ x 28λ ($1344\lambda^2$) (c) area is 31λ x35λ ($1085\lambda^2$)

As the transistor size increases, one not only has to consider the issue of diffusion sharing but also folding. For example, when the transistor size is increased to 4 times the minimum width, there are three fundamental layout strategies: folding without diffusion sharing, diffusion sharing with folding and diffusion sharing without folding. Each strategy affects the final layout area. In particular, Fig. 4.5 shows the effect of folding but without diffusion sharing. Three different layouts are shown, where each transistor has two folds and each transistor is placed differently. Fig. 4.5(a) resulted in minimum layout area of $1032\lambda^2$. Fig. 4.6(a) shows the layout with 2 fold and diffusion sharing; this resulted in area of $880\lambda^2$. Fig. 4.6(b) shows the layout with 4 fold and diffusion sharing; this result in area of $1368\lambda^2$.



(a) (b)

Fig. 4.6 : Layout of 2:1 multiplexer of size 4x with folding and diffusion sharing, (a) 2 folds- area $880\lambda^2$ (b) 4 folds- area $1368\lambda^2$

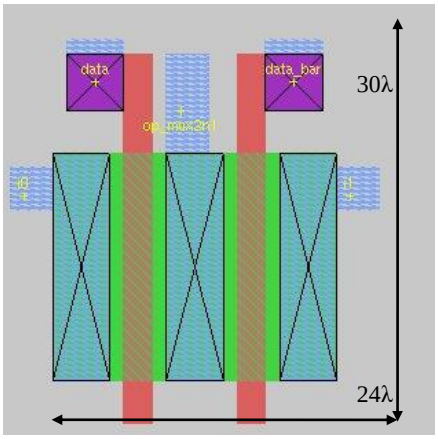
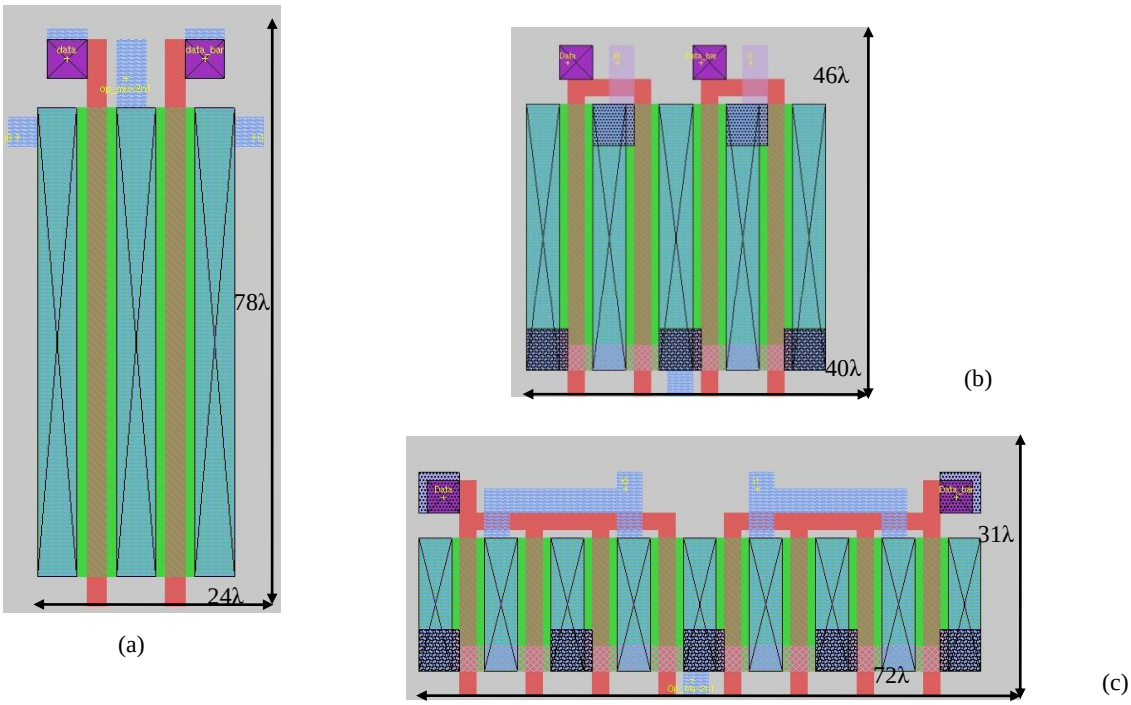


Fig. 4.7: Layout of 2:1 multiplexer of size 4x with diffusion widening and diffusion sharing, area is 24λ x 30λ ($720\lambda^2$)

Fig. 4.7 shows the layout of 2:1 multiplexer of transistor size 4x with only diffusion widening and diffusion sharing; this resulted in area of $720\lambda^2$. The layouts above show that diffusion sharing results in less area as compared with layouts without diffusion sharing since diffusion sharing minimizes active area also reduces the overall wiring requirement of a layout. Also, note that for transistor size 4, diffusion widening without transistor folding gives minimum layout area as shown in Fig. 4.7.

Fig. 4.8, however, shows the diffusion sharing layouts when the transistor size is increased to 16x of minimum width. As shown, in this case, the layout area with two folds results in minimum layout area. Consequently, the minimum layout area for a given transistor size is a function of the number of folds that a layout employs.



(a) (b) (c)

Fig. 4.8 : Layout of 2:1 multiplexer of transistor size 16x (a) no folds, 24x78=1872λ^2 (b) two folds, 40x46= 1840 λ^2 (c)four folds, 72x31=2232 λ^2

4.3 Active area modeling of 2:1 multiplexer

Both the VPR and the COFFE area models are based on the premise that, given unlimited number of metal layers, the actual layout area will eventually approach active area [11]. Neither area models, however, explicitly consider diffusion sharing and transistor folding. We discuss the grouping of neighboring transistors for utilizing maximum diffusion sharing along with transistor

sizing to generate area efficient layouts of 2:1 multiplexers. The effective width of the transistor is represented by w_{eff}.

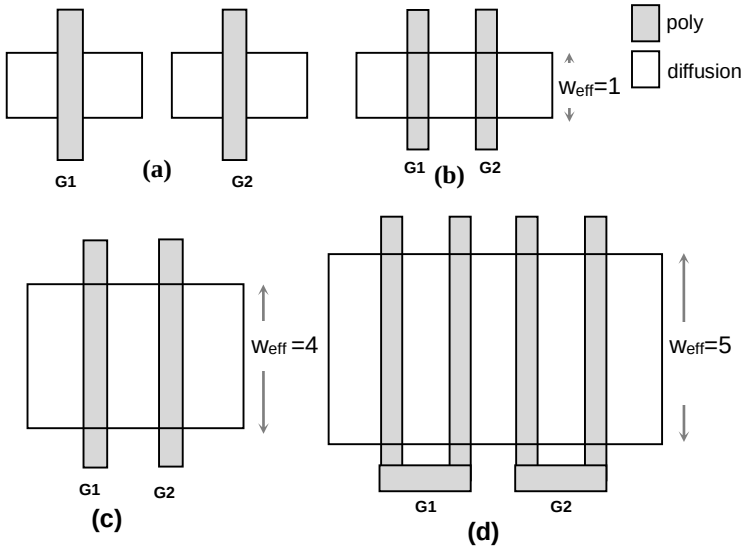


Fig. 4.9: (a) Two pass transistors with minimum width, w_{eff}=1 (b) 2:1 multiplexer from two transistors (w_{eff}=1) with shared drain diffusion (c) 2:1 multiplexer of transistors with width 4 (w_{eff}=4) from diffusion widening (d) 2:1 multiplexer of transistors with width 10 (w_{eff}=10) with two folds.

The 2:1 multiplexer can be laid out in different ways. Fig. 4.9(a) shows two discrete transistors of minimum drive strength. If transistors have small drive strengths, a compact layout can be created by simply sharing the drain diffusion of two transistors as shown in Fig. 4.9(b). Transistors with average drive strengths can share diffusion regions after diffusion widening as shown in Fig. 4.9(c) and transistors with high drive strengths need both fingering and diffusion widening before they are connected as shown in Fig. 4.9(d) to achieve less area [15]. Note that, in a practical layout of 2:1 multiplexer when diffusion regions of two transistors are shared it results in much less layout

area compared with the layout of two separate transistors as shown in Fig. 4.9(a). This is not considered in both area models of VPR and COFFE.

We consider these issues in our work and attempted to compare the layout area with the previous two area models for an FPGA fabric. Layouts for different sizes of multiplexers are presented in Chapter 5 .

4.3.1 **Diffusion sharing without transistor folding**

Active area calculation with minimum transistor spacing of 2:1 multiplexer is discussed by considering diffusion sharing and topology. Previous models [11] and [18] ignored diffusion sharing, and active area calculation was based on two discrete transistors. Fig. 4.10(a) depicts an area model for 2:1 multiplexer in terms of lambda without transistor folding while using diffusion sharing to reduce layout area and diffusion widening to increase drive strength. According to deep submicron SCMOS scalable rules the minimum width of a transistor with one contact is 4λ. Then the width of a transistor with w_{eff} times the minimum width becomes $4w_{eff}\lambda$. Considering diffusion sharing, the active area would be less than two discrete transistors. In particular, when w_{eff} is equal to one, two discrete transistors require the active area of 2 mwt while with diffusion sharing, as shown in Fig. 4.10(a), becomes active area 1.5 mwt.

Consequently, active area equation of 2:1 multiplexer for transistors with w_{eff} wide transistors in terms of lambda is given by

$$Active\_Area_{2:1mux} = (9 + 4w_{eff})(5+1)4\lambda^2 \quad (9)$$

$$= (9 + 4w_{eff})24\lambda^2 \quad (10)$$

In terms of mwt, the equation becomes

$$= \frac{(9+4w_{eff})24\lambda^2}{208\lambda^2} mwt \tag{11}$$

$$= \frac{(9+4w_{eff})3}{26} mwt \tag{12}$$

The equation above gives the true active area of 2:1 multiplexer for transistors that employ diffusion sharing but do not consider folding.

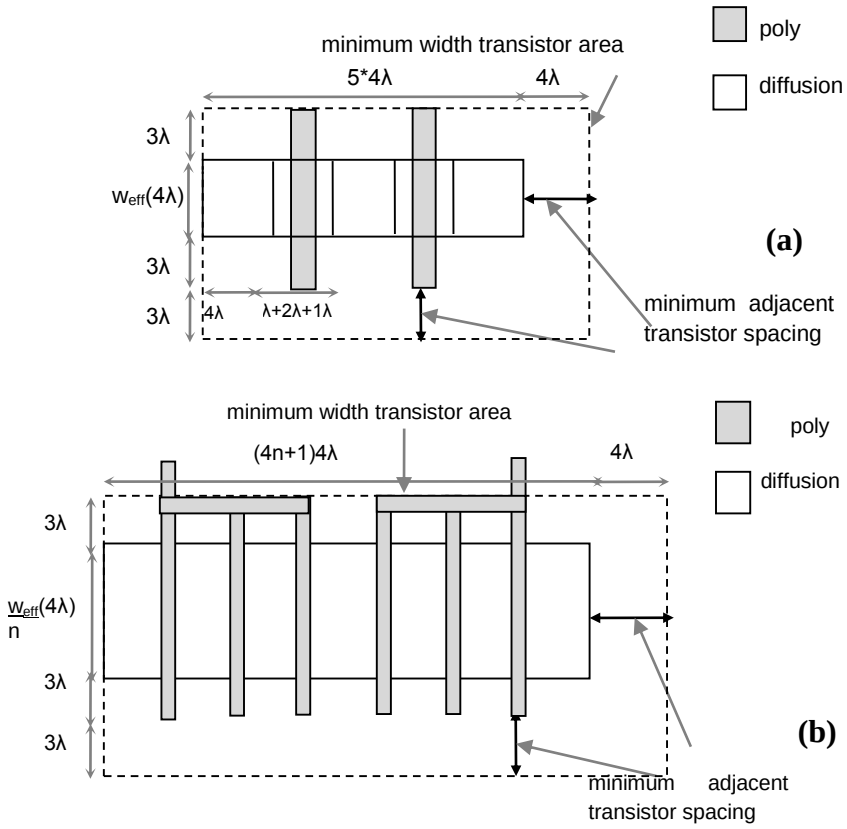


Fig. 4.10: Multiplexer with diffusion sharing (a) transistors with small drive strengths (b) transistors with large drive strengths.

4.3.2 **Diffusion sharing with transistor folding**

When the width of the transistors increases, it becomes essential to fold the transistors to achieve a compact layout for both area savings and performance [22]. Fig. 4.10(b) shows the area model of 2:1 multiplexer formed by diffusion sharing and folding. When a transistor of width w_{eff} is folded n times, the width of the diffusion region becomes w_{eff}/n. The resultant width in terms of lambda is $4w_{eff}/n\lambda$.

Using Fig. 4.10(b), one can generalize the active area equation of 2:1 multiplexer for transistors with n folds in terms of lambda as:

$$Active\_Area_{2:1mux} = (9 + 4\frac{w_{eff}}{n})(4n + 2)4\lambda^2 \tag{13}$$

$$= \frac{(9n + 4w_{eff})(2n + 1)8}{n}\lambda^2 \tag{14}$$

In terms of mwt, the equation becomes

$$Active\_Area_{2:1mux} = \frac{(9n + 4w_{eff})(2n + 1)8\lambda^2}{n \times 208\lambda^2}\ mwt \tag{15}$$

$$= \frac{(9n + 4w_{eff})(2n + 1)}{26n}\ mwt \tag{16}$$

Note, if n=1, the above equation is the same as Equation 7.

4.3.2.1 Number of folds

In this section we present the number of folds for any transistor size of a 2:1 multiplexer to achieve minimum area. Table 4.1 below shows active area comparison without folding (1 fold) and with 2 folds and 3 folds for different transistor sizes.

Table 4.1: Effect of area on folding

Transistor size	**without folding**	**with 2 folds**	**with 3 folds**
4x	600 λ^2	680 λ^2	802 λ^2
6x	792 λ^2	840 λ^2	952 λ^2
10x	1176 λ^2	1160 λ^2	1250 λ^2
12x	1368 λ^2	1320 λ^2	1400 λ^2

Observe that, when transistor effective width, w_{eff} is 4 and 6 folding is not required and when transistor effective width, w_{eff} is 10 and 12 the number of folds required are 2. We can conclude that for small transistor sizes; there is no need of folding and only diffusion widening is sufficient with diffusion sharing and for large transistors; folding is essential for significant area savings.

To determine the best number of folds in order to achieve the minimum active area, we can differentiate Equation 16 with respect to n and set the derivative to zero as shown in Equation 17 and 18. Finally, Equation 19 shows the number of folds, n, that is needed to achieve minimum active area as a function of the effective width, $w_{eff.}$

$$\frac{\partial Active\_Area_{2:1mux}}{\partial n} = \left(\frac{18}{26} - \frac{4w_{eff}\, n^{-2}}{26} \right) mwt = 0 \tag{17}$$

$$n^2 = \frac{4w_{eff}}{18} \tag{18}$$

$$n = \left\lceil \frac{\sqrt{2w_{eff}}}{3} \right\rceil = \left\lceil 0.471\sqrt{w_{eff}} \right\rceil, n \geq 1$$

n *is the number of folds of transistor with drive strength* w_{eff} (19)

4.4 Active area estimation for larger multiplexers

In section 4.1, we have observed that the primary building block of both encoded and decoded multiplexer is 2:1 multiplexer. Thus, based on the number of 2:1 multiplexers contained in each encoded and decoded multiplexer, true active area can be calculated as shown below.

4.4.1 **Encoded Multiplexers**

The exact active area of a k-input LUT can be calculated based on the number of 2:1 multiplexers that it contains. Thus the exact active area of a k-LUT is given by

$$Active\_Area_{k-LUT} = (2^k - 1)Active\_Area_{2:1mux} \tag{20}$$

4.4.2 **Decoded Multiplexers**

Consequently, the exact active area of our decoded multiplexer layouts can be calculated using the following equation,

$$Active\_Area_{z:1\ dmux} = (\frac{z}{2} + 1)Active\_Area_{2:1mux} \tag{21}$$

where z is the number of inputs to the multiplexer

4.5 Summary

In this chapter, we presented the importance of folding and diffusion sharing with change in transistor size and effect on area. We observed that smaller transistors do not need folding only diffusion widening is sufficient with diffusion sharing. However, folding is essential for larger transistor sizes for significant area savings.

This chapter also presented accurate active area models for both encoded and decoded multiplexers and also described a mathematical equation to determine the best number of folds for given transistor size.

Chapter 5

Pass Transistor based FPGA Components

Part II – Layout of Multiplexers

This chapter discusses the layout details of different sizes of encoded multiplexers used in LUTs and decoded multiplexers used in the routing architecture. Different layouts in our study are presented which led to achieve a compact layout. We have used up to 3 metal layers to layout all the FPGA basic building blocks including multiplexers. Initially, only metal1 and metal2 is used. Mostly, metal 1 is used for internal connections and metal 2 for long distance routing. Later, metal 3 is used to study the effect of area change with more layer of metal.

5.1 Encoded Multiplexers

LUTs are the basic components of FPGA which implements logic. LUTs are typically implemented by encoded multiplexers. In Chapter 2 we have discussed their functionality and schematic. Here, we explain the design of LUTs and also describe the different layouts for different sizes of LUTs. Higher order LUT's are created from lower order LUT's using mirroring technique.

One of our main focuses of research is a compact LUT layout design. Therefore, we exhibit a detailed layout of 4 LUT, as it is a widely suggested size for best area [5][11]. However, commercial FPGAs use LUT size as 6 which gives minimum area and delay product and therefore, we have also laid 5 LUT's and 6 LUT's of varying sizes.

5.1.1 **2-LUT**

A k-input LUT design requires 2^k SRAM cells and a 2^k input multiplexer [11]. The SRAM cells are the inputs to the multiplexer tree. Here, the multiplexer tree is constructed by pass transistors. The 2-LUT design requires 4 SRAM cells and a 4:1 multiplexer. Fig. 5.1 shows the schematic and circuit diagram of 2 LUT. Fig. 5.2, shows the layout of 2-LUT multiplexer tree, 4:1 multiplexer with minimum transistor width 1x. It comprises of three 2:1 multiplexers.

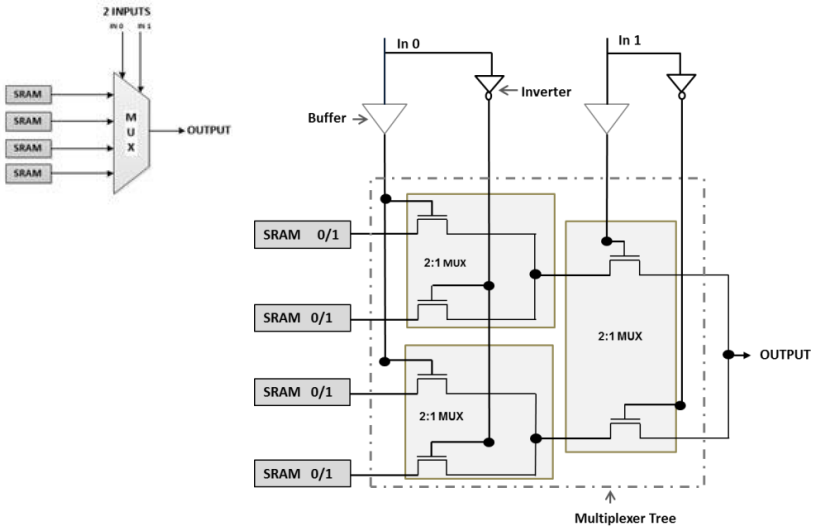


Fig. 5.1: 2-LUT schematic and circuit

We have laid the 4:1 multiplexers in two different ways and compared their area with and without internal connections. Fig. 5.2 shows two different layout designs without internal connections. Fig. 5.2(a) shows the layout when the output multiplexer is placed horizontal to one of the input multiplexers. This has an area of 49x43 λ^2 with white spaces only at the bottom and could be used for other connections. Fig. 5.2(b) shows the layout when output multiplexer is placed vertical and has an area 45x43λ^2. This is little less when compared to Fig. 5.2(a). The final layout

with internal connections is shown in Fig. 5.3 and similar area difference is noticed as in Fig. 5.3, the area with vertical 2:1 multiplexer at the output has lesser area.

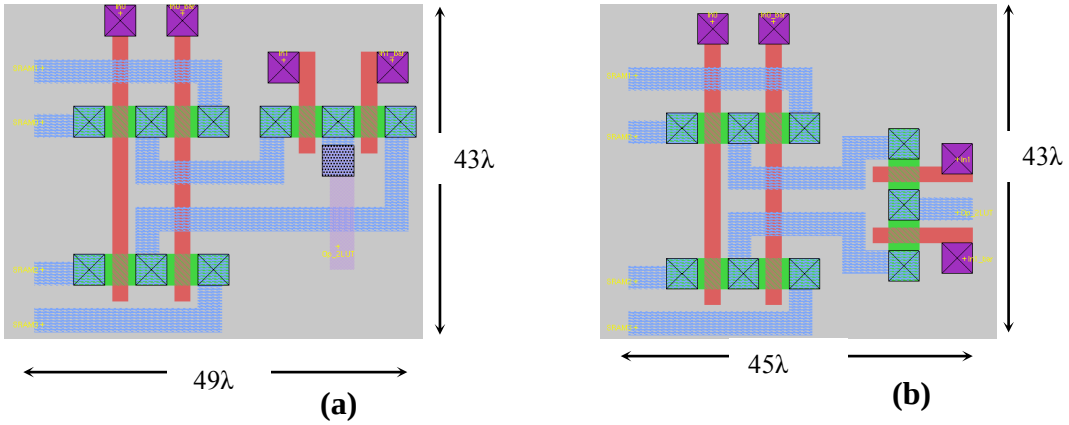


Fig. 5.2 : Two different orientations of 2 LUT layout (a) Area is 49x43 λ^2 (b) Area is 45x43 λ^2

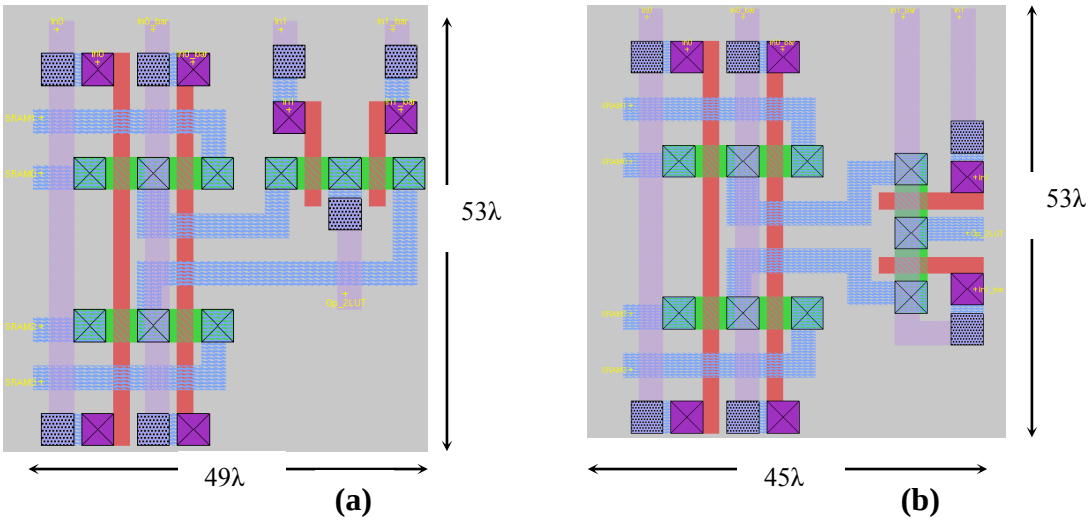


Fig. 5.3: 2-LUT layouts with internal connections (a) 49x53 λ^2 (b) 45x53 λ^2

Finally, we decide to use the first layout orientation with 8% more area. This is because all multiplexers have the same orientation, having their polysilicons vertical. Also, fixed orientation is a requirement for nanometer processes to have their low voltage transistors as vertical. Fig. 5.4(a) shows the 2 LUT layout with stacked via and transistor width 1x which resulted in an area

of 2640λ^2 and Fig. 5.4(b) shows the 2 LUT layout with stacked via and transistor width 6x which resulted in an area of 4845λ^2.

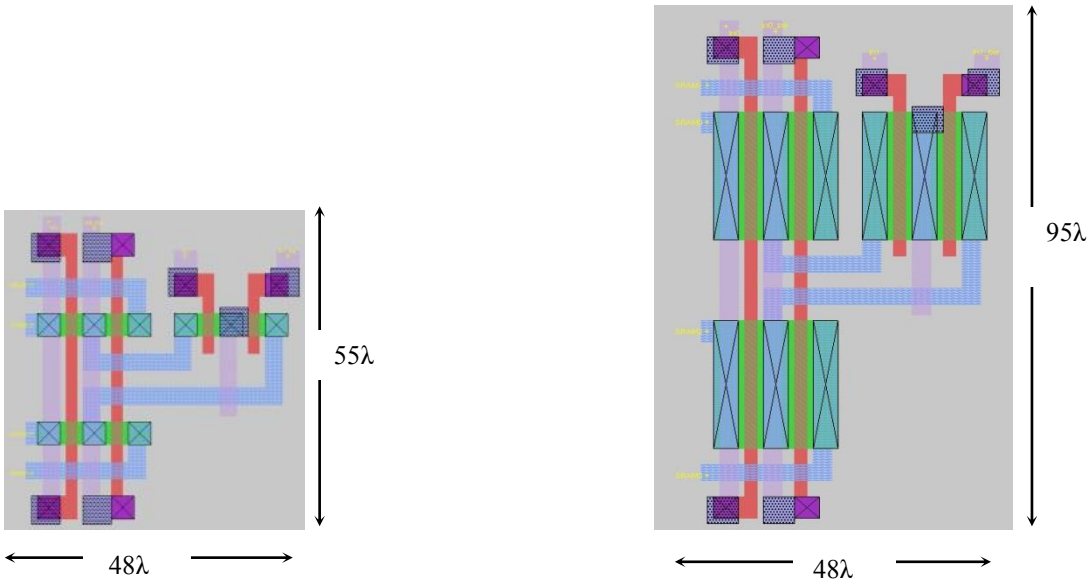


Fig. 5.4: 2-LUT layout (a) effective transistor width 1x and resulted in area of 2640λ^2 ,(b)effective transistor width 6x and resulted in area of 4560λ^2.

5.1.2 **3-LUT**

A 3-LUT design requires 8 SRAM cells and an 8:1 multiplexer. In the layout process we have constructed the 3-LUT multiplexer tree, 8:1 multiplexer from the already laid out 2- LUT. We have tried two different approaches using mirroring technique. One by taking the mirror image of 2 LUTs side by side which resulted in area of 49x96λ^2 and another by taking the mirror image one below the other which resulted in area of 102x43λ^2. Both methods use an additional 2:1 multiplexer for the creation of 3-LUT multiplexer. Observe that, when 2 LUTS are placed side by side as shown in Fig. 5.6 it results in less white space and less area with an area difference of 318λ^2. Fig.

5.7 shows the final 3-LUT multiplexer tree with transistor size 1x and all internal connections with an area of 6958λ^2. Fig. 5.8(a) shows its associated stick diagram using 2 metals which resulted in area 7680λ^2 and Fig. 5.8(b) shows the stick diagram using 3 metals which resulted in an area of 4608 λ^2. Fig. 5.9(a) shows the layout of 3-LUT multiplexer with stacked via and transistor size 1x which resulted in an area of 6958λ^2 and similarly Fig. 5.9(b) shows the layout of 3-LUT multiplexer with transistor size 6x which resulted in an area of 10878 λ^2.

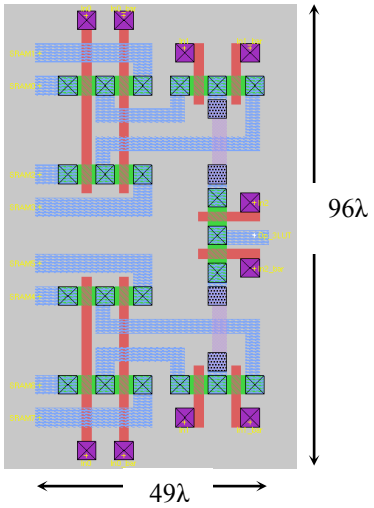


Fig. 5.5: 3- LUT layout when 2- LUTs are placed one below the other. Area is 49x96 =4704λ^2

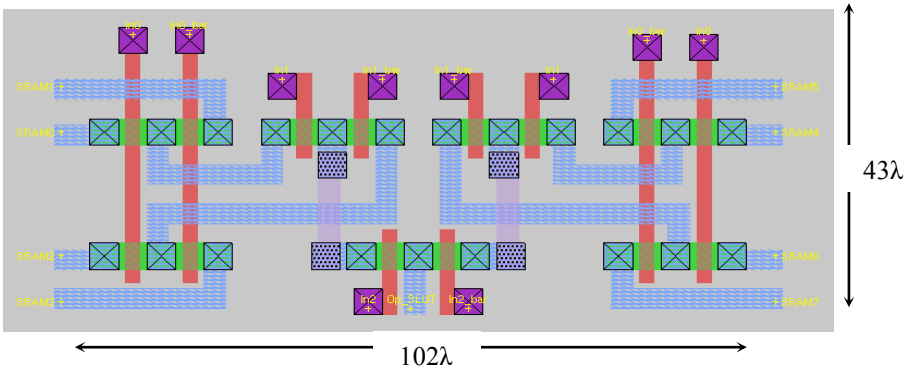


Fig. 5.6: 3-LUT layout when 2- LUTs are placed side by side. Area is 102x43= 4386 λ^2

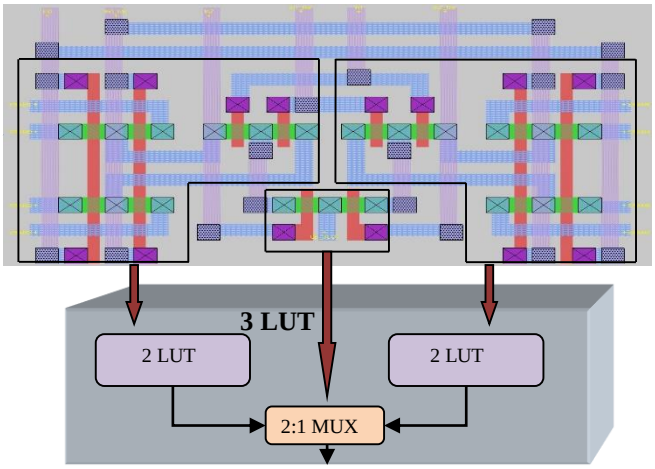
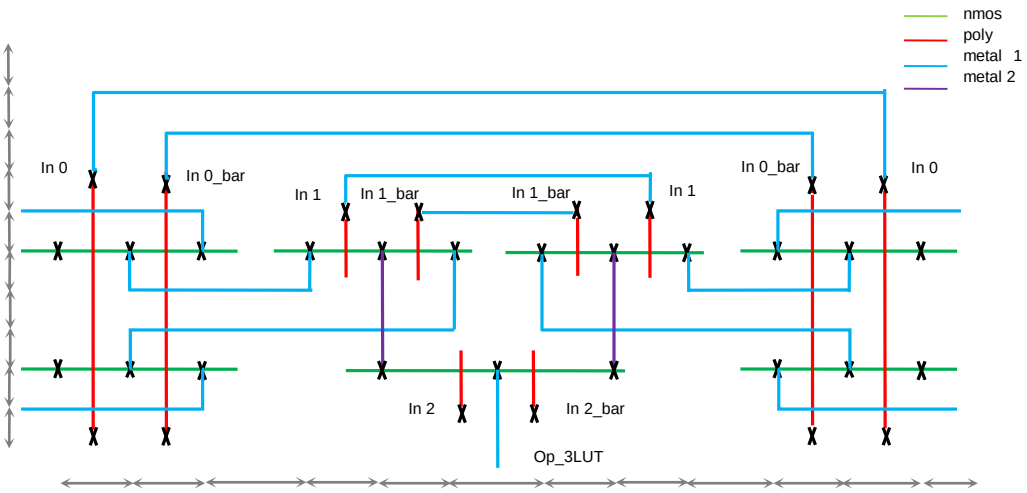


Fig. 5.7 3- LUT layout using 2 metals with internal connections. Area is 98x71=6985 λ^2



(a)

10 tracks * 12 tracks - >80λ x96λ

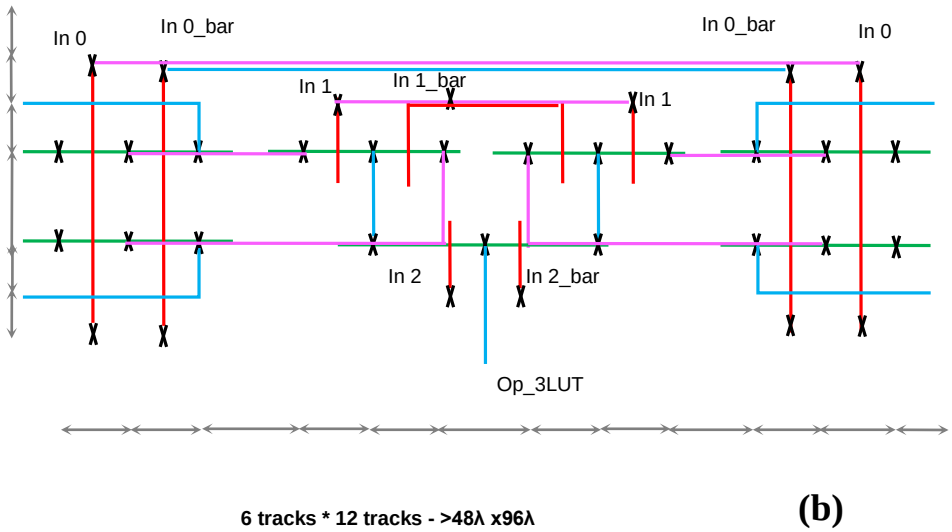


Fig. 5.8: 3-LUT multiplexer stick diagram illustration (a) 2 metal, (b) 3 metal

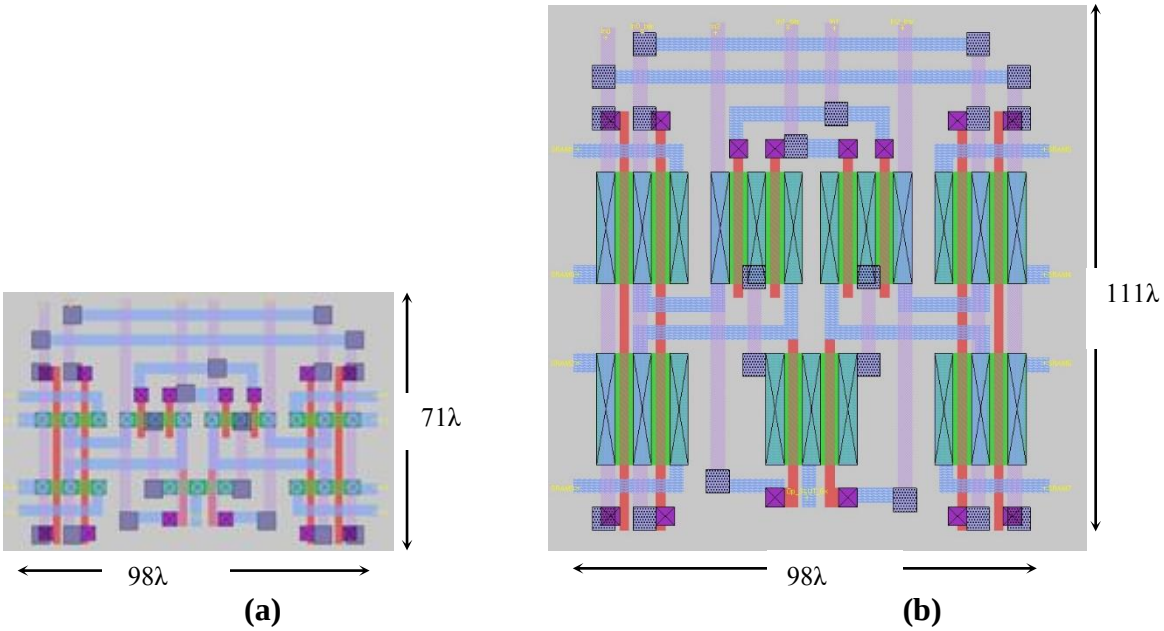


Fig. 5.9: 3-LUT layout (a) with effective transistor width 1x. Area is 98x71=6958 λ^2

and (b) with effective transistor width 6x. Area is 98x111=10878 λ^2

5.1.3 **4-LUT**

A detailed study on the layout of 4 LUTs is carried out. Here, we have tried different options to accomplish a simple floor plan but yet a compact layout with minimum area. A 4-LUT design requires 16 SRAM cells and a 16:1 multiplexer. The layout of 16:1 multiplxer tree is designed from two 3 LUTs and one 2:1 multiplexer. Below is an in-depth discussion of different options.

Option 1: In this layout a 4 LUT is designed from two 3 LUTs and a 2:1 multiplexer as illustrated in Fig. 5.10. The 2:1 output multiplexer is placed inside, near the output of the 3 LUTs. This layout results in more white spaces which are unutilized. Therefore, in our next layouts we try to minimize the white spaces. The total area of 4 LUT with internal routing for this layout is 141 x 113 =15933 λ^2.

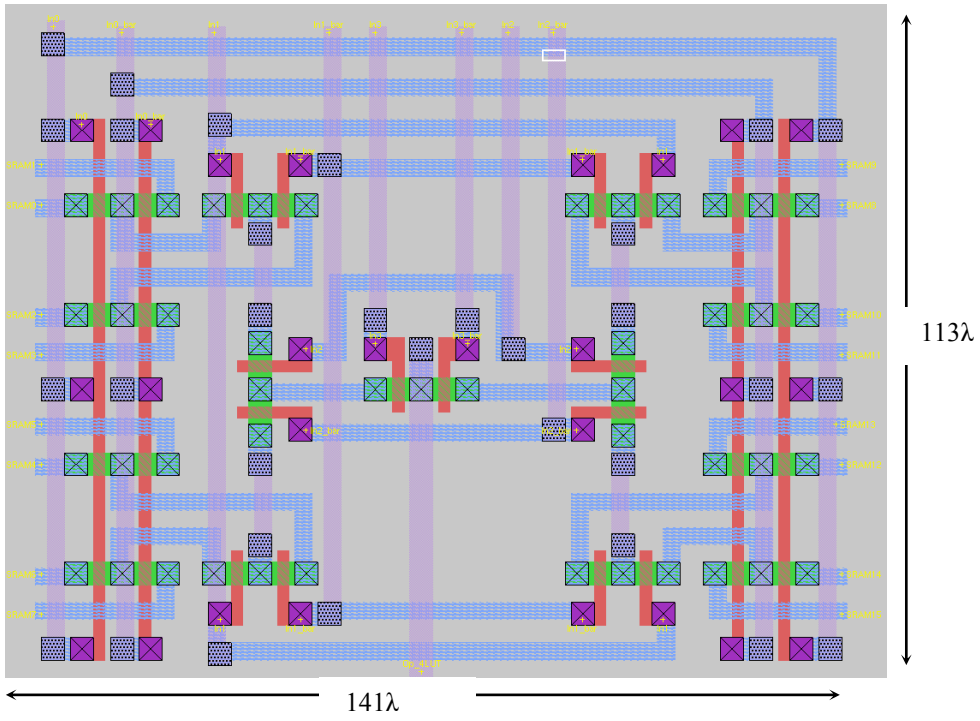


Fig. 5.10: 4 LUT layout, when output 2:1 multiplexer is placed in between the 3 LUTs.

Option 2: This layout in Fig. 5.11 is the modification of the first layout, option 1. Here, we try to minimize the white spaces by placing the output multiplexer outside. We observed that the white spaces could be reduced when the two 3 LUTs are placed side by side and the output multiplexer just below the 3-LUTs. However, this resulted in higher area, 2.5 % more when compared to the option 1. It was inspected that the extra area resulted from routing. The total area for this layout with all internal routing is 122 x 134= 16348 λ^2.

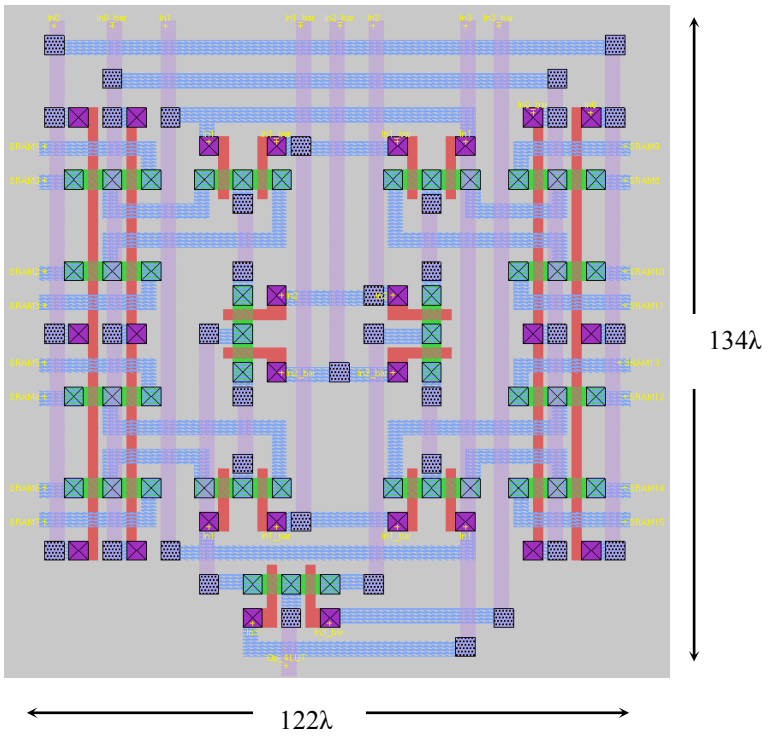


Fig. 5.11: 4 LUT layout when output 2:1 multiplexer is placed below the 3 LUTs.

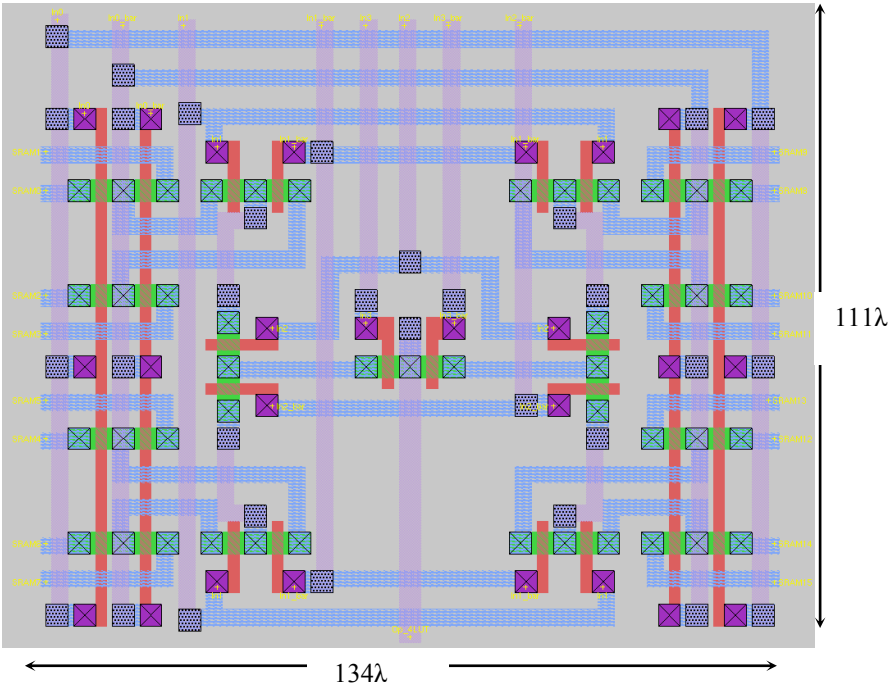


Fig. 5.12: 4-LUT layout, when output 2:1 multiplexer is placed in between the modified 3-LUTs.

Option 3: This layout in Fig. 5.12 is the improved version of option1; keeping the output multiplexer inside and minimizing the internal white spaces. Here, the 3 LUT layout was also slightly modified to further minimize the white spaces. The area for this layout is 134 x 111= 14874 λ^2, resulting in 6.6 % (1059 λ^2) area saving when compared to option1.

Option 4: This layout is our final 4 LUT layout with minimum internal white spaces and the most compact layout. The 3 LUTs layout used here is presented in Fig. 5.13. The 2:1 multiplexer at the output is placed below the 3 LUTs. This layout resulted in the best area when compared with all the three layout options. The total area with all internal routing is 104 x 126 =13104 λ^2. An area saving of 17% is achieved when compared to option 1 layout and an area saving of 13% is achieved when compared to option 3. Fig. 5.13 shows the corresponding compact 4-LUT layout with minimum internal white spaces.

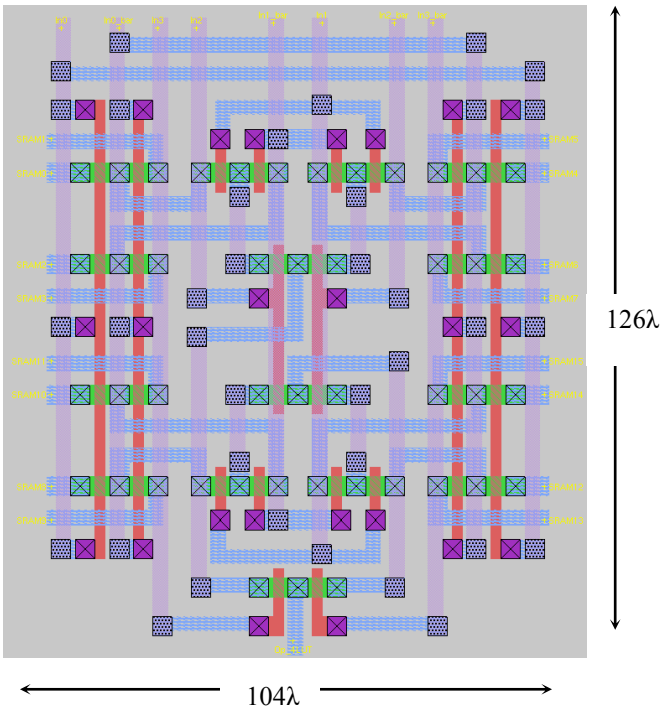
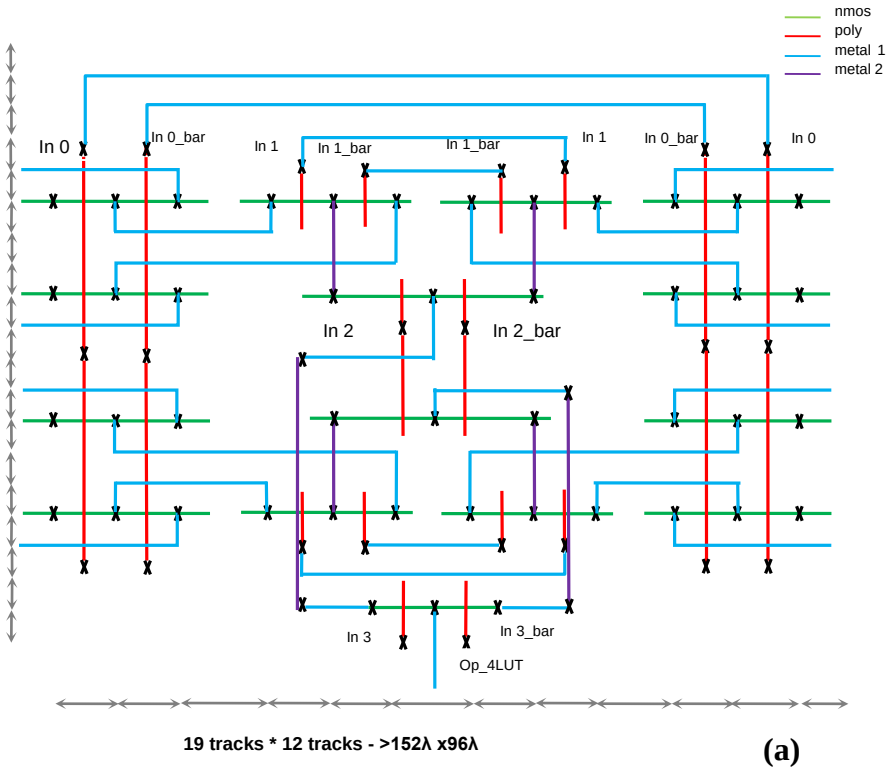


Fig. 5.13: Compact 4 LUT layouts with minimum internal white spaces



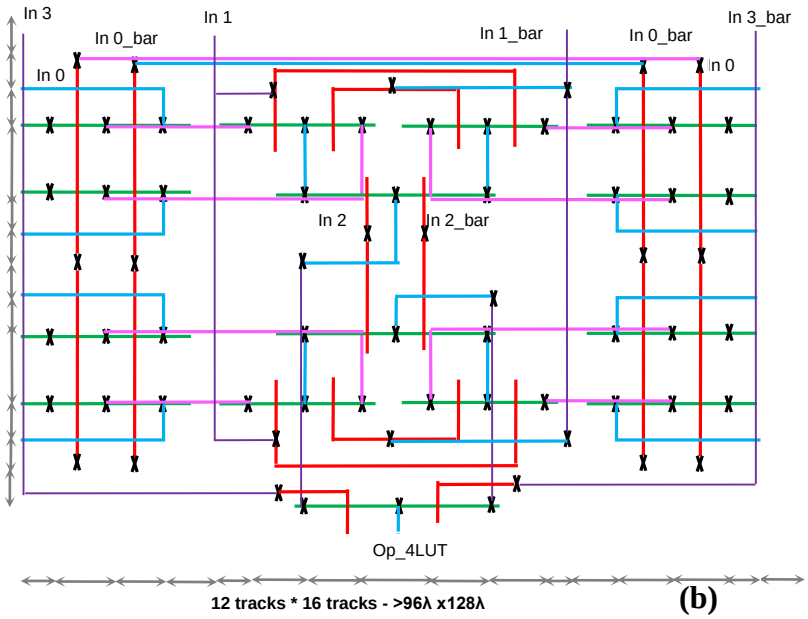


Fig. 5.14: 4-LUT multiplexer, stick diagram illustration (a) 2 metal (b) 3 metal

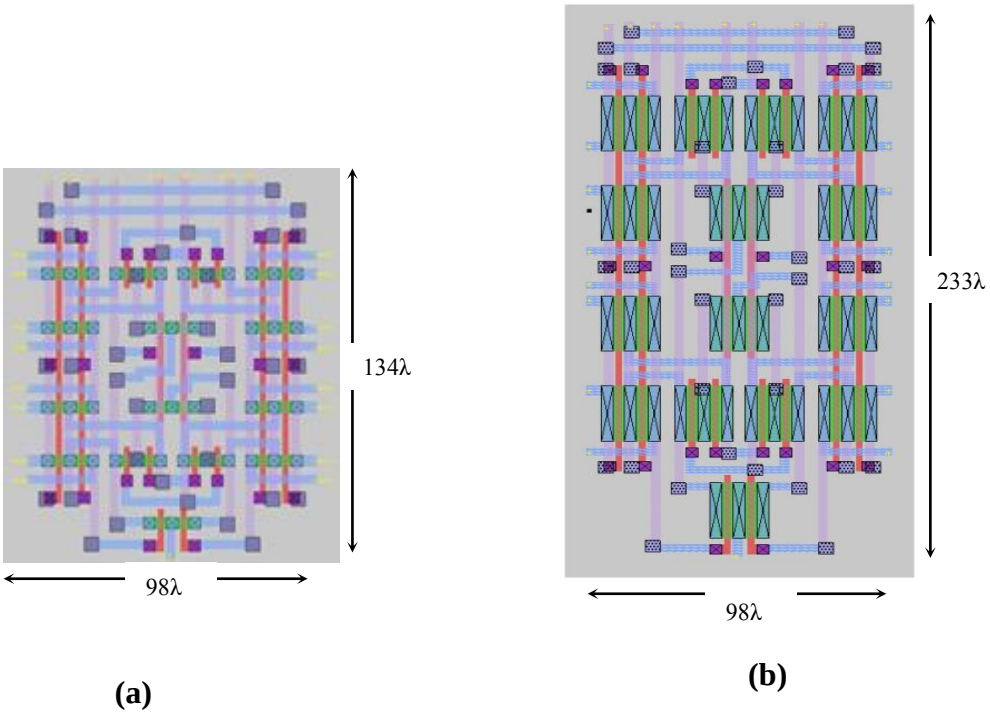


Fig. 5.15: 4-LUT layout (a) with transistor size 1x, Area is 98x134=13132 λ^2

and (b)with transistor size 6x, Area is 98x233=22834 λ^2

Stick diagram representation of 4-LUT is shown in Fig. 5.14. Fig. 5.14(a) shows the 4-LUT stick diagram using 2 metals which resulted in an area of $14592\lambda^2$ and similarly Fig. 5.14(b) shows the 4-LUT stick diagram using 3 metals which resulted in an area of $12288\lambda^2$. The final 4-LUT layout with stacked via is shown in Fig. 5.15.

5.1.4 **5-LUT and 6-LUT**

Similarly, higher order LUTs can be created using mirroring technique. Fig. 5.16 shows 5-LUT layout using 2 metals with an area of 33580 λ^2. Fig. 5.17 shows 6-LUT layout using 2 metals with an area of 76659 λ^2.

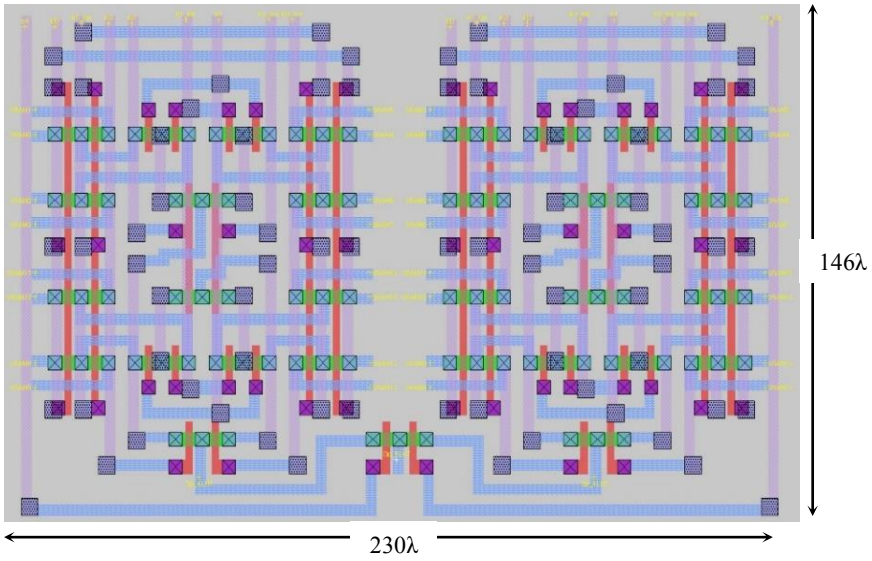


Fig. 5.16: 5-LUT layout. Area is 230x146=33580 λ^2

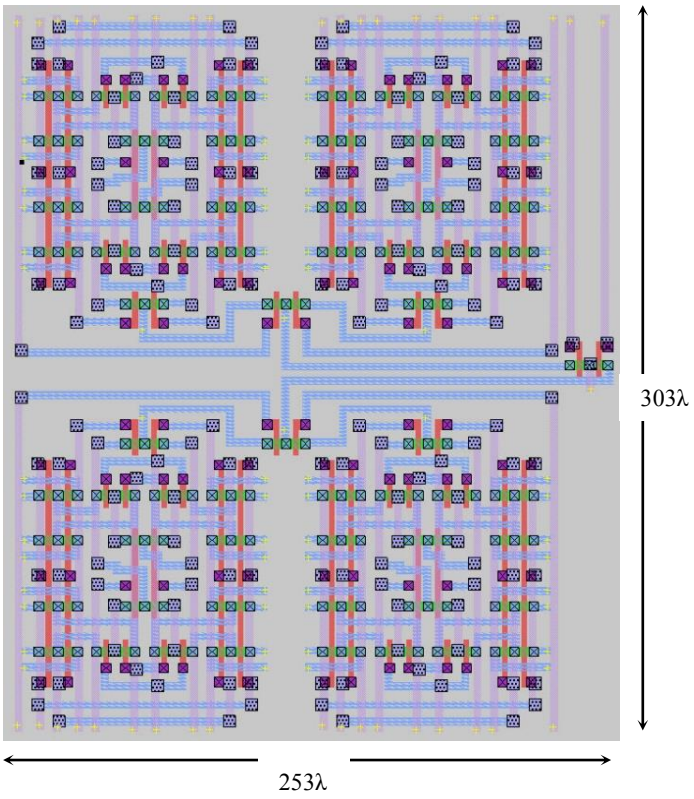


Fig. 5.17 : 6-LUT Layout. Area is 253x303=76659 λ^2

5.2 Decoded Multiplexer

The decoded multiplexers are commonly used to implement the local routing networks and connection blocks [8][22]. They offer better area delay product in routing switch blocks when compared to encoded multiplexers. Fig. 5.18 shows the schematic of an 8-input decoded multiplexer, with a two-level multiplexer topology used in [18][24]. Observe that the decoded multiplexer is also constructed from pass transistors and the 2:1 multiplexer forms the primary building block. The 8-input decoded multiplexer contains five 2:1 multiplexers as own in schematic. Fig. 5.20(b) shows the layout of 8-input decoded multiplexer.

4-input decoded multiplexer is discussed in Chapter 4 Fig. 4.2(b) shows the 4-input decoded multiplexer schematic. It contains three 2:1 multiplexers and its layout is similar to the 4-input encoded multiplexer.

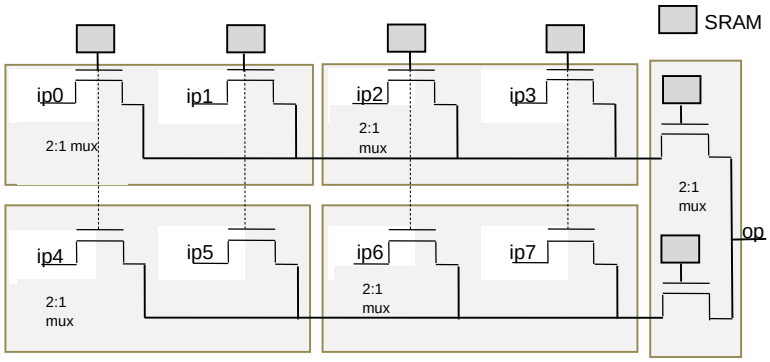


Fig. 5.18: 8: 1 Decoded Multiplexer

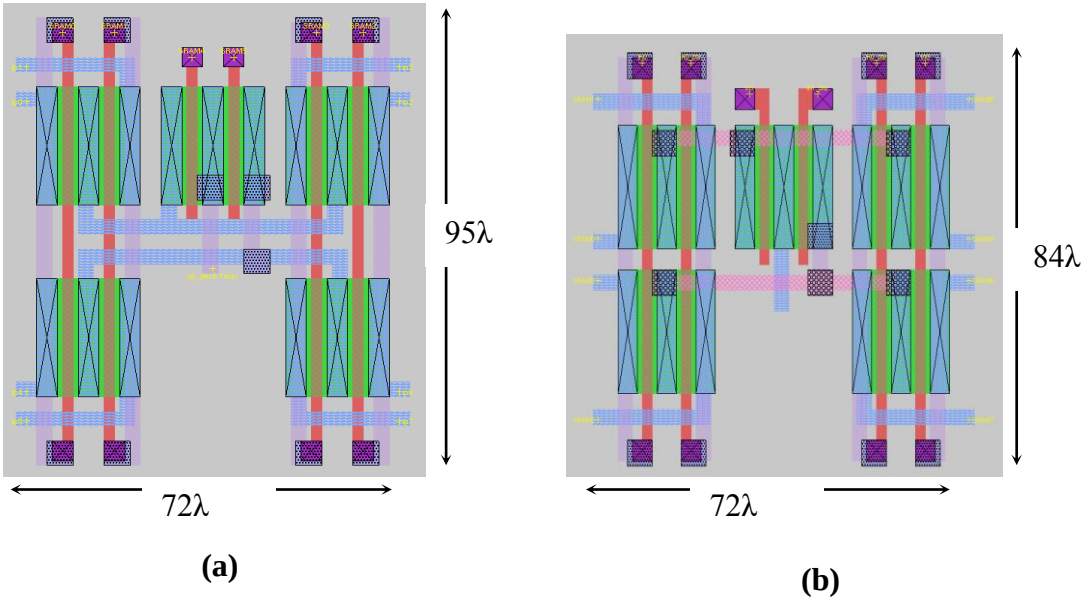


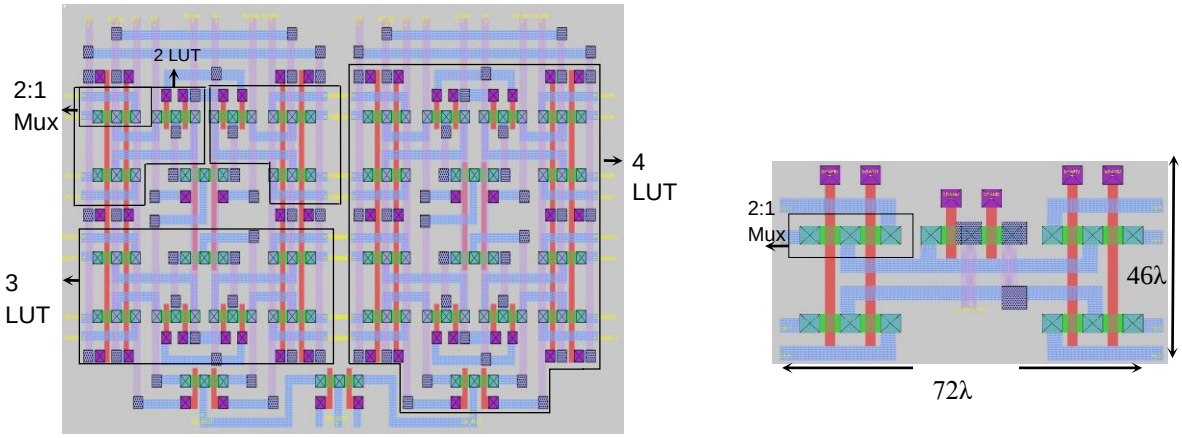
Fig. 5.19: 8:1 decoded multiplexer, 6x transistor size (a) 2metals, Area is 72x95= 6840λ^2 (b) 3metals, area is 72x84=6048λ^2

Fig. 5.19(a) shows a compact 8:1 decoded multiplexer of transistor size 6x using 2 metals that resulted in $6840\lambda^2$ and Fig. 5.19(b) shows a compact 8:1 decoded multiplexer of transistor size 6x using 3 metals that resulted in $6048\lambda^2$.

5.3 Layout strategy for encoded and decoded multiplexers

In this work, we initially laid out 4:1, 8:1, and 16:1 encoded multiplexers using both 2 layers and 3 layers of metals and 1x, 4x, 6x, 8x and 16x minimum width transistors. These multiplexers correspond to 2-input, 3-input, and 4-input LUTs, respectively. Higher order LUTs 5 and 6 have also been laid. A general strategy of mirroring is used to reuse the lower order multiplexers to create higher order multiplexers. In particular, as shown by, a higher order LUT can be created by two lower order LUTs and an additional 2:1 multiplexer.

To ensure good layout quality and maximize design reuse, we first create a dense 2:1 multiplexer for a given number of metal layers and transistor size. A 2-LUT is then created by mirroring one 2:1 multiplexer into two 2:1 multiplexers and by adding an additional 2:1 multiplexer to select the outputs of the two mirrored multiplexers. Similarly a 3-LUT is created by mirroring one 2-LUT into two 2-LUTs and adding an additional 2:1 multiplexer to select the outputs of the two mirrored 2-LUTs. The same process is repeated for creating other higher order LUTs. Given this mirroring strategy, the exact active area of a k-input LUT can be calculated based on the number of 2:1 multiplexers that it contains.



(a) **(b)**

Fig. 5.20: Layout using 2 metals (a)5-LUT layout (b) 8:1 decoded multiplexer

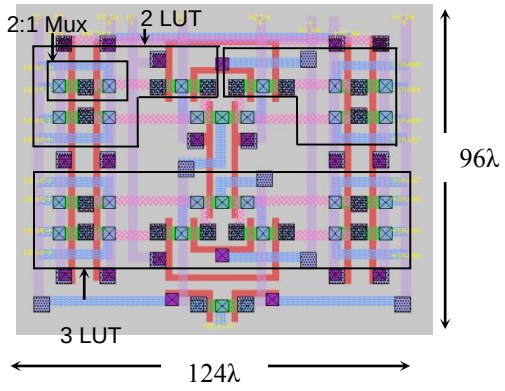
We use similar layout strategy to create layouts for 4:1, 8:1 and 16:1 decoded multiplexers. These multiplexers are also based on the same layout of 2:1 multiplexers used in the layouts of the encoded multiplexers. In particular, our decoded multiplexers are based on the two-level multiplexer topology used in [18],[22], and [24] as shown in Fig. 5.18. As with encoded multiplexers, six versions of layouts are created for each decoded multiplexer by varying the number of metal layers from 2 to 3 and using 1x, 6x and 16x minimum width transistors. The two-metal layout of the 8:1 multiplexer is shown in Fig. 5.20(b). As shown the decoded multiplexer is also constructed out of a series of 2:1 multiplexer layouts.

It is important to note that since the interconnect area of a layout depends on the transistor drive strength, w_{eff}, the number of metal layers used and the circuit topology of a component, wiring contributes to a significant amount of the total layout area of multiplexers. This is due to the large number of input signals and the recursively Y-connected topology of the multiplexers, which limit the extent of diffusion sharing. Consequently, the Y-topology of multiplexers requires

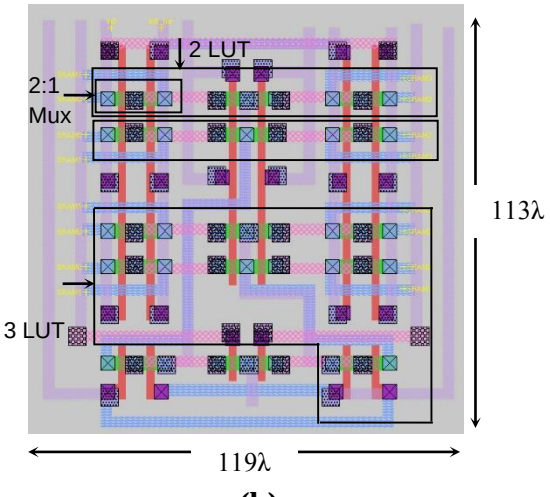
more layout area per unit of active area than the layout of the series and parallel topologies found in other FPGA components such as buffers and full adders.

5.3.1 **Comparing mirroring strategy with row and column strategy suggested by VPR**

Comparing the above mirrored layout strategy to the strategy suggested by the VPR area model [11], where transistors are uniformly distributed in a row and column format. In particular, a 4-LUT requires 30 transistors and hence, can be arranged in six rows and five columns. After diffusion sharing and interconnect of transistors we found that the mirroring strategy works well for layouts containing small transistors where wiring area dominates the total layout area. In particular, as shown by Fig. 5.21(a) and (b), for transistor size 1x our layout strategy using mirroring technique is 13% smaller than the row and column strategy for laying out 4-LUTs where the row and column strategy evenly distributes the 15 2:1 multiplexers over 5 rows and 3 columns.



(a)



(b)

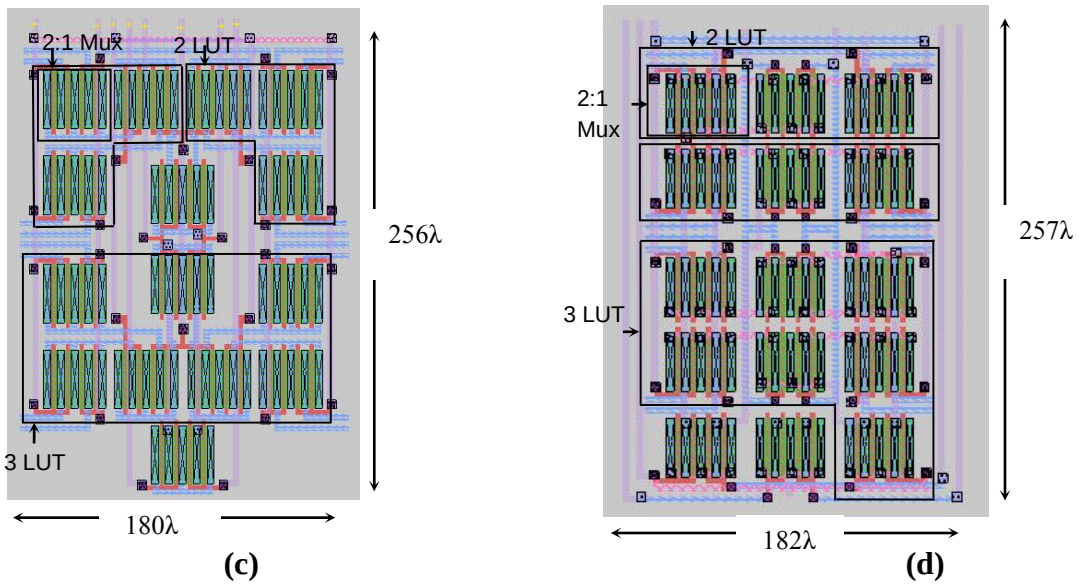


Fig. 5.21 : 4-LUT layout using 3 metals (a) mirroring technique with 1x transistors (b) row and column technique with 1x transistors (c) mirroring technique with 16x transistors (d) row and column technique with 16x transistors

As transistor size increases, however, the active area starts to dominate the total layout area and the area advantage of the mirroring strategy correspondingly reduces. As shown by Fig. 5.21(c) and (d), the mirroring strategy is only 1.5% smaller than the row and column strategy where the transistor size is increased to 16x minimum width.

Consequently, for all our experimental analysis we have considered the layout area resulting from mirroring strategy as it results in minimum layout area.

5.4 Summary

In this chapter, we described the layouts of encoded and decoded multiplexers for different multiplexer inputs. It reveals the exhaustive layout work done to achieve compact area. This chapter also presented a novel mirroring technique for multiplexers where higher order

multiplexers can be created from lower order multiplexers. The mirroring technique discussed resulted in minimum area when compared with the layout strategy of VPR.

Chapter 6

Experimental Analysis and Results

This chapter presents the experimental results. We first compare the actual SCMOS deep submicron layout area for LUT multiplexers for minimum size transistor, 1x with stick diagram area prediction which is IC process independent and also with minimum width transistor area models VPR [11] and COFFE [18]. We have put our best effort to make the layouts as compact as possible and compare the area results.

We second compare the theoretical minimum layout area (the active area) against the VPR and the COFFE area predictions. We then justify the number of metal layers that are used in our work and compare the full layout area with the predicted layout area of VPR and COFFE. In particular, we measure the area that a model over/under-estimates as a percentage of the actual layout area (i.e. $(E_i - A_i)/A_i$ or $1/\alpha_i - 1$), where A_i is the actual layout area (including wiring area) of the FPGA building block, E_i is the estimated layout area and α_i is the ratio between the actual layout area and the estimated layout area. Results for both the estimated area from the VPR area model and the COFFE area model are presented. Both the encoded and decoded multiplexer layout areas are then presented as a function of the multiplexer size and transistor size.

We later compare LUT multiplexers with routing multiplexers. Detail analysis and comparison is provided. In particular, an example of 16:1 multiplexer is used to discuss the difference when it is used as a LUT multiplexer and when it is used as a routing multiplexer.

Finally, the models' ability to predict the layout area of multiplexers is compared to the models' ability to predict the layout area of CMOS-based FPGA building blocks.

Table 6.1: Stick Diagram and Layout comparison

	Area in λ^2								
LUT mux size	**Layers**	**Stick Diagram**	**Layout Area**	**Difference**	**% Difference**	**VPR Area [11]**	**Layout area / VPR area**	**COFFE Model [18]**	**Layout area / COFFE area**
2 LUT mux	2 metal	3072	2640	432	16	(6x208) 1248	2.25	1205.6	2.33
	3 metal	2304	2208	96	4		1.77		1.83
3 LUT mux	2 metal	7680	6958	722	10	(14x208) 2912	2.40	2813	2.48
	3 metal	4608	4704	-96	-2		1.62		1.67
4 LUT mux	2 metal	14592	13132	1460	11	(30x208) 6240	2.10	6027.9	2.18
	3 metal	12288	11904	384	3		1.91		1.97
5 LUT mux	2 metal	37632	33580	4052	12	(62x208) 12896	2.60	12457.6	2.70
	3 metal	29952	29252	700	2		2.27		2.35
6 LUT mux	2 metal	85312	76659	8653	11	(126x208) 26208	2.93	25317.18	3.03
	3 metal	67392	63036	4356	7		2.41		2.49

6.1 Stick Diagram Comparison

Here we compare the layout area for different LUT multiplexers for minimum size transistor, 1x, with stick diagram prediction and also compares the layout area to the VPR area model [11] and the COFFE area model [18]. Note that, early FPGA architectural studies used LUT multiplexers constructed out of 1x transistors [11]. Table 6.1 compares stick diagram area with actual SCMOS deep submicron layout area for transistor size 1x. As shown in column 6, the stick diagram model accurately estimates the layout area to within 85% -95% of the actual layout area. Note that the

inaccuracy is mainly due to the fact that in stick diagram track separation is uniformly considered to be 8λ but in the actual layout this may vary from 6λ to 8λ. As shown in column 8 and 10, both area models underestimate the actual layout area by a factor of 2-3. The reason for the underestimation is that both of these models only consider transistor spacing and size but do not consider the actual connectivity between the transistors. From our analysis we observe that stick diagram area is much closer to the actual layout area. Stick Diagrams can achieve over 90 percent accuracy in layout area estimation while remaining IC-process independent. This work is presented in [12].

6.2 Active Area Comparison

Both the VPR and the COFFE area models are based on the premise that, given unlimited number of metal layers, the actual layout area will eventually approach active area [11]. Neither area models, however, explicitly consider diffusion sharing and transistor folding.

The true active area used by a 2:1 multiplexer can be calculated using Equation 14, which taking into account of diffusion sharing. The area then can be minimized for a given transistor size, w_{eff}, based on Equation 17, which calculates the best number of folds, n, for a given value of w_{eff}.

Table 6.2 shows the minimum active area calculations for the transistor sizes of 1x minimum width to 34x minimum width for the 2:1 multiplexer. Also shown are the estimated VPR area and COFFE area in column 3 and 4 respectively and column 5 and 6 shows the percentage of over/under-estimations for each model as a percentage of the actual active area. The percentage values are also plotted in Fig. 6.1(a).

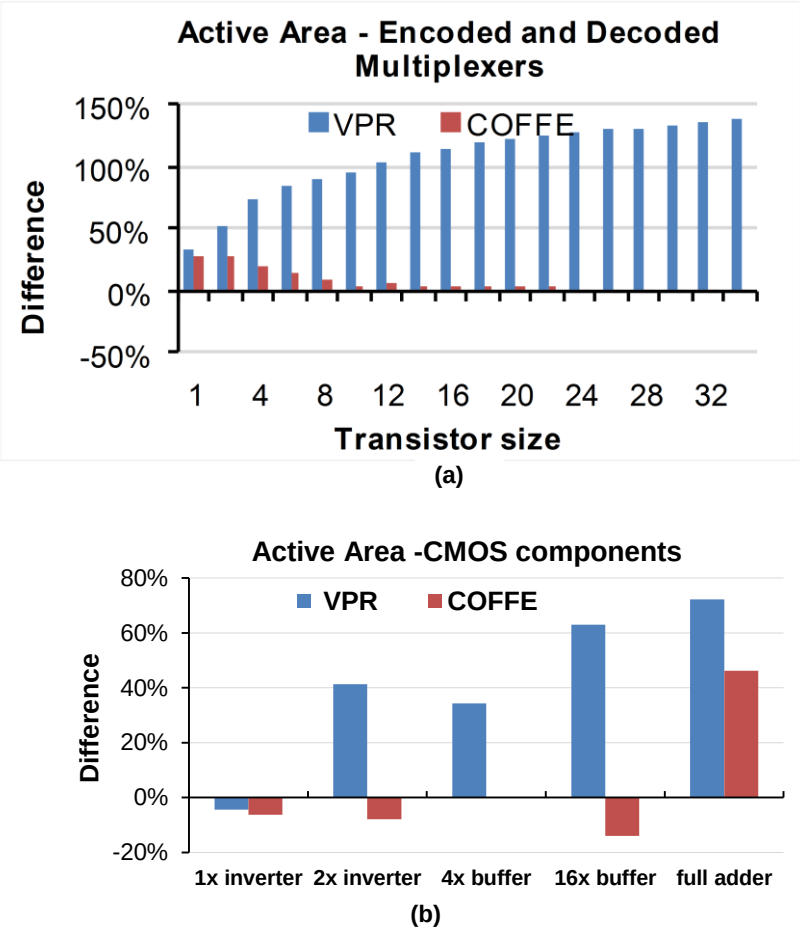


Fig. 6.1 Active area comparison (a) Encoded and Decoded Multiplexers (b) CMOS based components

As shown, for both the encoded and decoded multiplexers, VPR overestimates active area by 33% to 139% for transistor sizes of 1x-34x. The COFFE model, on the other hand, performs much better and overestimates for small transistor sizes of 1x-6x from 14% to 29% and is very close to the active area for larger transistor sizes which ranges from overestimation of 9% (8x) to underestimation of 3% (34x).

Note that, for our layouts shown in Fig. 5.20(a) and Fig. 5.20(b), the 2:1 multiplexer is repeatedly used to construct larger multiplexers. As a result, as shown by Equation 15 and 16, the

active area of a larger encoded or decoded multiplexer can be calculated as the product of the number of 2:1 multiplexers that the multiplexer contains and the active area of one 2:1 multiplexer. Since both the VPR and COFFE models can be similarly decomposed into the product of the number of 2:1 multiplexers that a multiplexer contains and the active area of one 2:1 multiplexer, the over/under-estimation values shown Fig. 6.1(a) also represent the over/under-estimation values of all multiplexers investigated in this work.

Table 6.2: Active Area of Multiplexers

Multiplexer	**Area in λ^2**				
Transistor size	**Active Area**	VPR Area	COFFE Area	**VPR%** difference	**COFFE%** difference
1	312	416	402	33	29
2	408	624	522	53	28
4	600	1040	724	73	21
6	792	1456	904	84	14
8	984	1872	1072	90	9
10	1176	2288	1233	95	5
12	1320	2704	1388	105	5
14	1480	3120	1540	111	4
16	1640	3536	1689	116	3
18	1800	3952	1835	120	2
20	1960	4368	1978	123	1
22	2120	4784	2120	126	0
24	2280	5200	2261	128	-1
26	2440	5616	2400	130	-2
28	2600	6032	2538	132	-2
30	2744	6448	2674	135	-3
32	2893	6864	2810	137	-3
34	3043	7280	2944	139	-3

Table 6.3 shows the measured active area for buffer sizes of 1x to 16x as well as the full adder presented in Chapter 3 . The table also shows the VPR and the COFFE estimated area for these

components in column 3 and 4 respectively. The percentages of over/under-estimation are then shown in column 5 and 6, respectively, and are plotted in Fig. 6.1(b).

Table 6.3: Active Area CMOS based Components

	Area in λ^2				
Component	**Active Area**	VPR Area	COFFE Area	**VPR %** difference	**COFFE %** difference
1x inverter	544	520	510	-4	-6
2x inverter	736	1040	678	41	-8
4x buffer	1472	1976	1471	34	0.1
16x buffer	4416	7176	3813	63	-14
Full adder	4712	8112	6889	72	46

As shown, VPR underestimates for 1x inverter by 4% but overestimates for larger inverters and buffers and full adder by a maximum of 72%. In contrast, COFFE underestimates for inverters and buffers by a maximum of 18% and overestimates for full adder by 46%.

Overall the VPR model ranges from underestimating active area by 4% to overestimating active area by 139%. The COFFE model has a smaller range of error. In particular, it has a range of underestimation of active area by 14% to overestimation of active area by 46% for CMOS components and underestimation by 3% to overestimation by 29% for multiplexers.

It is also important to note that active area calculations based on Equation 15 and 16 and minimized by Equation 14 are equal to the actual active area measured from our layouts. Consequently, they are much more accurate for calculating the active area of multiplexers than both the VPR and COFFE models due to their additional consideration for both diffusion sharing and transistor folding. Furthermore, since the layout for many basic CMOS components such as buffers and full adders are widely available [21], an active area model based on the direct

measurement of the actual layout area is feasible for CMOS components and is also more accurate than both the VPR and COFFE models.

6.3 Selection of the number of metal layers for layout

We use the publicly available Intel 45nm process metal stack as our guide to decide the number of metal layers to use in the layout of the fundamental FPGA building blocks on typical commercial IC processes. In particular, Table IV shows the metal stack for the Intel 45nm process [21]. As shown, only the bottom 3 metal layers are minimum width. The higher level layers have significantly increased minimum metal width and consequently are less suitable for short distance connections. Consequently, in this work we use up to 3 metal layers for intra-building-block routing.

Table 6.4 : INTEL 45nm Metal Stack [19]

Layer	thickness(nm)	width(nm)	pitch(nm)
M9	7 µm	17.5 µm	30.5 µm
M8	720	400	810
M7	504	280	560
M6	324	180	360
M5	252	140	280
M4	216	120	240
M3	144	80	160
M2	144	80	160
M1	144	80	160

Using only the minimum width metals allows us to create a set of highly flexible layouts for FPGA building blocks that can be used across a wide range of IC processes with a minimum amount modification. Furthermore, we observe only 5%-10% area reduction when metals are

increased from 2 to 3 layers. Since the M9 layer is reserved for distributing power to different power-gated domains across the die [21] on the Intel process, this leaves us with 2 metal layers for power and clock distribution and 3 metal layers for implementing global and local routing network on this 9 metal layer process.

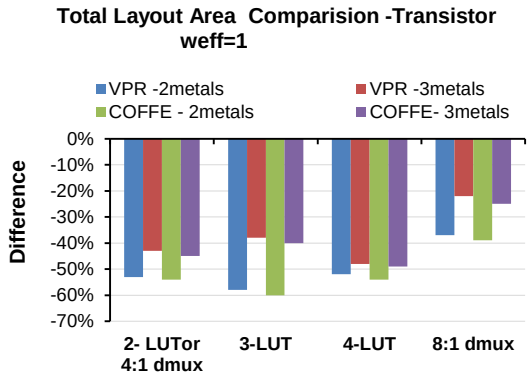
6.4 Multiplexers based on 1x transistors

The layout area of the multiplexers investigated in this work is shown in Table 6.5. Also shown are the VPR estimated area and COFFE estimated area for each component in column 5 and 6 respectively. Finally, the area that the VPR and COFFE model over/under-estimates as a percentage of the actual layout area is shown in column 7 and 8 respectively.

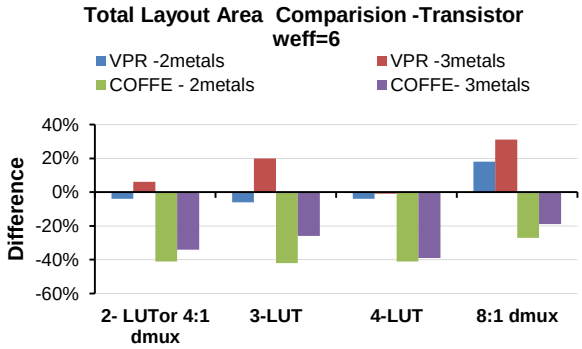
Since early FPGA architectures exclusively use encoded multiplexers that are constructed out of 1x transistors [11], we first investigate the layout area of these multiplexers. The percentages of over/under-estimation are plotted in Fig. 6.2(a). The figure shows that both VPR and COFFE underestimate the total layout area of the encoded and decoded multiplexers.

The figure also shows that for the same type of multiplexers, there is little variation in percentage of over/under-estimation for the VPR model. In particular, the percentage of underestimation only varies by 6% (from 52% to 58%) when VPR is used to estimate the layout area of encoded multiplexers with 2 layers of metal. The variation grows to 10% (from 38% to 48%) for 3 layers of metal. When both encoded and decoded multiplexers are considered, however, the variation in the percentage of underestimation grows. Specifically, the variation grows from 6% to 21% and 11% to 26% for 2 and 3 layers of metals respectively. Similarly, the COFFE model underestimates the layout area by 54% to 60% (a variation of 6%) for 2 layers of metal and by

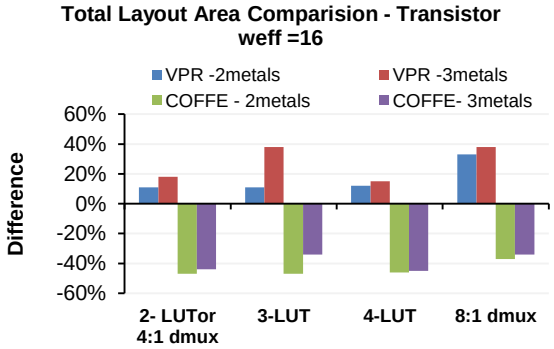
40% to 49% (a variation of 9%) for 3 layers of metal for encoded multiplexers. The variation grows from 6% to 21% and 9% to 24% for 2 and 3 layers, respectively, when both types of multiplexers are considered.



(a)



(b)



(c)

Fig. 6.2: Layout area comparison (a) transistor w_{eff}=1 (b) transistor w_{eff}=6 (c) transistor w_{eff}=16

The result shows that both the VPR and COFFE model are likely to produce very accurate ranking of earlier FPGA architectures where only encoded multiplexers are used throughout the architectures and where the buffers and pass transistors do not consist of a significant amount of total layout area. For newer architectures which employ a mix of encoded and decoded

multiplexers [22], however, the accuracy of both models in correctly ranking the layout area of FPGA architectures decreases due to the increased variation in prediction errors.

The result also shows that the COFFE area model has similar accuracy for estimating the layout area of 1x multiplexers as the VPR area model. Similar variations also exist between the prediction error of the COFFE model and the VPR model (21% for 2 layers and 26% vs. 24% for 3 layers). This is due to the fact that the COFFE model has a similar amount of error in estimating the active area of 1x multiplexers as the VPR model (29% vs. 33%) as shown in Fig. 6.1(a).

6.5 Effect of Transistor Size on the Consistency of Prediction Errors

Modern FPGAs are increasingly incorporating multiplexers that are constructed out of larger transistors [18]. To investigate the effect of transistor sizing on the consistency of the prediction errors, we laid out multiplexers using transistors that are 6x of minimum width and 16x minimum width. Fig. 6.2(b) and Fig. 6.2(c) plots the over/under-estimations as a percentage of the actual layout area for multiplexers.

Along with Fig. 6.2(a), the figures show that the prediction error of the VPR model varies significantly with changing transistor sizes. In particular, the variation in prediction error varies from an underestimation of 48% (16:1 encoded multiplexer, 1x) to an overestimation by 38% (16:1 encoded multiplexer, 16x; and 8:1 decoded multiplexer, 16x) for an overall variation of 86% for 3 metal layers and layouts with 2 metal layers have a similar variation of 91%.

Table 6.5 : Total Layout Area – Encoded and Decoded Multiplexers

	Area in λ^2							
Component	**Transistor size**	**Reference**	**Metal Layers**	**Layout Area (λ^2)**	**VPR Area**	**COFFE Area**	**VPR %** difference	**COFFE %** difference
2 LUT (4:1 encoded mux)	1x	**Fig. 5.4(a)**	2	2640	1248	1206	-53	-54
		A1	3	2208	1248	1206	-43	-45
	6x	**Fig. 5.4(b)**	2	4560	4368	2708	-4	-41
		A7	3	4128	4368	2708	6	-34
	16x	**B3**	2	9523	10608	5066	11	-47
		A10	3	8989	10608	5066	18	-44
3 LUT (8:1 encoded mux)	1x	**Fig. 5.9(a)**	2	6958	2912	2813	-58	-60
		A2	3	4704	2912	2813	-38	-40
	6x	**Fig. 5.9(b)**	2	10878	10192	6319	-6	-42
		A8	3	8526	10192	6319	20	-26
	16x	**B4**	2	22320	24752	11820	11	-47
		A9	3	17978	24752	11820	38	-34
4LUT (16:1 encoded mux)	1x	**Fig. 5.15 (a)**	2	13132	6240	6028	-52	-54
		A3	3	11904	6240	6028	-48	-49
	6x	**Fig. 5.15 (b)**	2	22834	21840	13541	-4	-41
		A9	3	22158	21840	13541	-1	-39
	16x	**B5**	2	47160	53040	25328	12	-46
		A12	3	46080	53040	25328	15	-45
4:1 dmux	1x	**Fig. 5.4(a)**	2	2640	1248	1206	-53	-54
		A1	3	2208	1248	1206	-43	-45
	6x	**Fig. 5.4(b)**	2	4560	4368	2708	-4	-41
		A7	3	4128	4368	2708	6	-34
	16x	**B3**	2	9523	10608	5066	11	-47
		A10	3	8989	10608	5066	18	-44
8:1 dmux	1x	**Fig. 5.20(b)**	2	3312	2080	2009	-37	-39
		A6	3	2664	2080	2009	-22	-25
	6x	**Fig. 5.19(a)**	2	6840	7280	4514	6	-34
		Fig. 5.19(b)	3	6048	7280	4514	20	-25
	16x	**B6**	2	13320	17680	8443	33	-37
		A13	3	12810	17680	8443	38	-34

The COFFE model has a reduced variation. This is mainly due to the more accurate active area estimation at larger transistor sizes as shown in Fig. 6.2(a). The variation, however, still is significant and ranges from an underestimation of 56% (8:1 encoded multiplexer, 1x) to an

underestimation of 25% (8:1 decoded multiplexer, 6x) for an overall variation of 24% for 3 metal layers and layouts with 2 metal layers has a variation of 26%. These variations are a direct result of the models not considering the effect of transistor sizing on diffusion sharing and transistor folding. Furthermore, the models do not consider the varying proportion of wiring area and active area as the transistor size increases.

6.6 Comparison of LUT multiplexer with Routing multiplexer

The inconsistency in the VPR and COFFE models in estimating layout area can create biases in FPGA architectural studies. For example, a 16:1 multiplexer can be used to construct both a 4-LUT and a 16:1 routing multiplexer. In constructing the 4-LUT, the 16:1 multiplexer requires an additional 16 bits of SRAM as shown in Fig. 6.3(a). In constructing the 16:1 routing multiplexer, on the other hand, only 4 bits of SRAM is required as shown in Figure 14(b). Since 6-transistor SRAM cells are typically used in FPGAs ([11] and [18]) both the VPR and COFFE area models more accurately predict the true layout area of SRAM cells than the true layout area of multiplexers, both models produce a more accurate layout area prediction for 4-LUTs because of larger number of SRAM cells than 16:1 routing multiplexers. In particular, when the 16:1 multiplexer constructed using 1x transistors is used to construct a 4-LUT, both the VPR and COFFE area models are relatively accurate and the true layout area is only 17% and 12% bigger than the estimated area of VPR (as shown in row 5 column 11 of Table 6.6) and COFFE (as shown in row 5 column 13 of Table 6.6) models respectively.

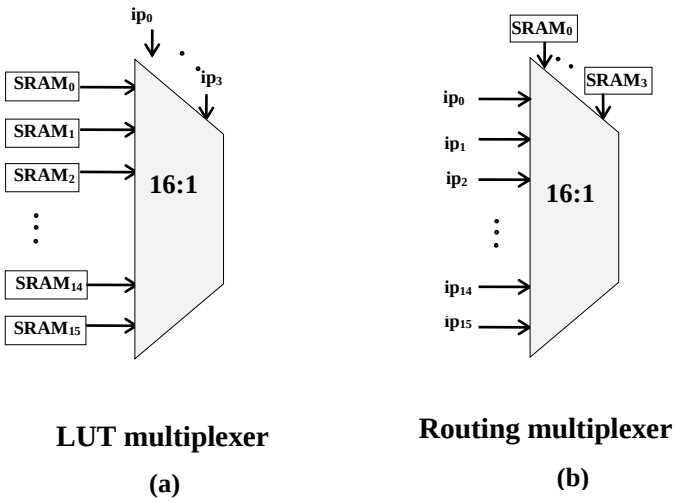


Fig. 6.3 : Multiplexer (a) LUT (b) Routing

When used to construct a routing multiplexer, on the other hand, the actual layout area is 48% to 46% larger than the estimated area of VPR (as shown in row 5 column 10 of Table 6.6) and COFFE (as shown in row 5 column 12 of Table 6.6) respectively. Such a large variation in estimation error would lead to architectural studies that unfairly favor the use of LUTs over routing multiplexers. In particular, studies such as [1], where the utilization of LUTs within basic logic clusters are sacrificed in order to increase overall area efficiency by increasing the efficiency of local and global routing resources, should take into account these variations in LUT and routing multiplexer area to more accurately characterize the optimal trade-offs between LUT utilization and routing resource utilization.

Table 6.6 also shows LUT area versus routing multiplexer area in row 3 to 7 for multiplexers of sizes 4:1 to 64:1 with 1x transistors. As shown, for small multiplexer sizes, the variation between the area estimation made by the VPR and COFFE area models and the true layout area are relatively small for both LUTs and routing multiplexers. In particular, for 4:1 routing multiplexers, the true layout area is 21% (row 3 column 10) and 17% (row 3 column 12) larger

than the area prediction of the VPR and COFFE area models respectively for routing multiplexers. The 2-LUT constructed using the same 4:1 multiplexer, on the other hand, is 10% (row 3 column 11) to 5% (in row 3 column 13) larger than the area model predictions. As the multiplexer size increases, the variation increases as well. In particular, for 64:1 multiplexers, the true layout area is 108% (row 7 column 10) and 110% (row 7 column 12) larger than the area model predictions for routing multiplexers, while the 6-LUT constructed using the same multiplexer is only 50% (row 7 column 11) and 24% (row 7 column 13) larger than the area model predictions.

The table also shows the area comparison between true layout area and area model predictions for larger transistor sizes (from row 9 to row 27). As shown, large prediction variations persist at larger transistor sizes. In particular, for the 64:1 multiplexers and 16x transistor size, true layout area is 117% (row 27 column 12) larger than the COFFE area model prediction for routing multiplexers while the true layout area is only 65% (row 27 column 13) larger than the COFFE area model prediction for LUT area. It is also important to note that in architectures that contain a mix of transistor sizes and multiplexer sizes, the prediction variations can be even larger. For example, the true layout area is 20% (row 13 column 10) smaller than VPR area model prediction for 8:1 routing multiplexers constructed out of 16x transistors. If the same architecture also contains 64:1 routing multiplexers of 1x transistors, the true layout area of these multiplexers is 108% (row 7 column 10) larger than the VPR area model. Similarly, the COFFE area prediction varies from 130% smaller than the true layout area (64:1 routing multiplexer with 8x transistors) to 4% smaller than the true layout area (3-LUT with 1x transistors). These results show that correction factors produced by this work are important in increasing the accuracy of future FPGA architectural studies in selecting the correct mix of FPGA resources for implementing efficient FPGA fabrics on SOCs.

Table 6.6: LUT Area vs Routing Multiplexer Area

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	Multiplexer	Transistor size	Layout Area			VPR Area		COFFE Model		Layout area / VPR Area		Layout area / COFFE Area	
2			Mux area	Routing mux with SRAM	LUT mux with SRAM	Routing mux with SRAM	LUT mux with SRAM	Routing mux with SRAM	LUT mux with SRAM	Routing mux with SRAM	LUT mux with SRAM	Routing mux with SRAM	LUT mux with SRAM
3	**4 to 1**	**1x**	2208	4548	6888	3744	6240	3876.80	6548	1.21	1.10	1.17	1.05
4	**8 to 1**	**1x**	4704	8214	14064	6656	12896	6819.80	13497.8	1.23	1.09	1.20	1.04
5	**16 to 1**	**1x**	11904	16584	30624	11232	26208	11370.30	27397.5	1.48	1.17	1.46	1.12
6	**32 to 1**	**1x**	29252	35102	66692	19136	52832	19135.60	55196.8	1.83	1.26	1.83	1.21
7	**64 to 1**	**1x**	63036	70056	137916	33696	106080	33330.78	110795.58	2.08	1.30	2.10	1.24
8													
9	**4 to 1**	**4x**	3360	5700	8040	5616	8112	4843.97	7515.17	1.01	0.99	1.18	1.07
10		**8x**	4896	7236	9576	8112	10608	5887.19	8558.39	0.89	0.90	1.23	1.12
11		**16x**	8989	11329	13669	13104	15600	7736.83	10408.03	0.86	0.88	1.46	1.31
12													
13	**8 to 1**	**4x**	7056	10566	16416	11024	17264	9076.59	15754.59	0.96	0.95	1.16	1.04
14		**8x**	10192	13702	19552	16848	23088	11510.78	18188.78	0.81	0.85	1.19	1.07
15		**16x**	19402	22912	28762	28496	34736	15826.61	22504.61	0.80	0.83	1.45	1.28
16													
17	**16 to 1**	**4x**	19220	23900	37940	20592	35568	16206.24	32233.44	1.16	1.07	1.47	1.18
18		**8x**	29264	33944	47984	33072	48048	21422.35	37449.55	1.03	1.00	1.58	1.28
19		**16x**	52208	56888	70928	58032	73008	30670.56	46697.76	0.98	0.97	1.85	1.52
20													
21	**32 to 1**	**4x**	49700	55550	87140	38480	72176	29129.94	65191.14	1.44	1.21	1.91	1.34
22		**8x**	76964	82814	114404	64272	97968	39909.90	75971.10	1.29	1.17	2.08	1.51
23		**16x**	117298	123148	154738	115856	149552	59022.86	95084.06	1.06	1.03	2.09	1.63
24													
25	**64 to 1**	**4x**	107532	114552	182412	73008	145392	53641.73	131106.53	1.57	1.25	2.14	1.39
26		**8x**	166860	173880	241740	125424	197808	75549.39	153014.19	1.39	1.22	2.30	1.58
27		**16x**	241500	248520	316380	230256	302640	114391.87	191856.67	1.08	1.05	2.17	1.65

6.7 FPGA CMOS components

Layout area comparison of other FPGA-related CMOS components is done against the VPR and the COFFE models as shown in Table 6.7. The same data is plotted in Fig. 6.4. We measured the layout area of 1x, 2x, 4x, 16x buffers and the full adder. Note that both models underestimate area for 1x, 2x, and 4x buffers. VPR overestimates the area of 16x buffers and full adder while

COFFE underestimate area for 16x buffer and overestimates for the full adder. Overall the estimation error ranges from an underestimation of 50% to overestimation of 36% for the VPR model for an overall variation of 84%. For the COFFE model, the variation ranges from an underestimation of 51% to overestimation of 13% for an overall variation of 64%.

Both models show a large variation in prediction error for CMOS components due to the large variation in circuit topology and wiring demand across CMOS circuits.

Table 6.7 : Total Layout area –CMOS based FPGA Components

	Area in λ^2					
Component	**Reference**	**Full Layout Area**	VPR Area	COFFE Area	**VPR %** difference	**COFFE %** difference
1x inverter	**Fig. 3.5**	1036	520	509.6	-50	-51
2x inverter	**Fig. 3.7**	1372	1040	678.08	-24	-51
4x buffer	**Fig. 3.15**	2112	1976	1470.56	-6	-30
16x buffer	**Fig. 3.19**	5292	7176	3812.64	36	-28
Full adder	**Fig. 3.24**	6072	8112	6888.96	34	13

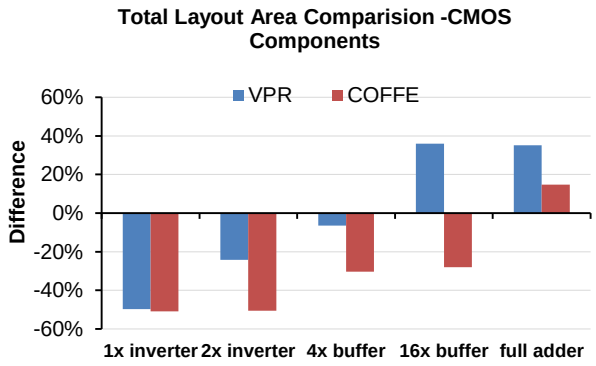


Fig. 6.4: Layout area comparison for CMOS components

6.8 Summary

This chapter has described the experimental analysis performed to determine the accuracy of minimum width transistor area models widely used in FPGA architectural studies. We investigate the suitability of using minimum width transistor area to directly estimate the actual implementation area of FPGA-based reconfigurable fabrics.

We first compared the actual SCMOS deep submicron layout area for LUT multiplexers for minimum size transistor, 1x with conventional stick diagram area prediction and also with minimum width transistor area models of VPR and COFFE. We second compared the theoretical minimum layout area (the active area) against the VPR and the COFFE area predictions. We then compared LUT multiplexers with routing multiplexers. Finally, we compared the models ability to predict multiplexer layout area with the predicted layout area od CMOS based components.

Chapter 7

Conclusion and Future Research

This chapter summarizes the conclusion of our study and the contribution of this research and also presents the future work to be addressed.

7.1 Summary

Based on our layout work, for commonly used FPGA components we conclude that the minimum width transistor area models [11] and [18] do not give accurate area estimations nor does it scale with the same factors for all components. They are inaccurate especially in layouts where wiring area dominates. COFFE underestimates for buffers and encoded and decoded multiplexers with any number of inputs but overestimates for full adders. However, VPR underestimates for encoded multiplexers and small size buffers and overestimates for decoded multiplexers with large transistor sizes, large size buffers and full adders. This variation is due to the fact that different components have different circuit topologies. Components which have exclusively parallel and in series connected transistors can extensively employ diffusion sharing in order to minimize their layout area. Minimum width transistor area models from [11] and [18] also do not consider the connectivity and grouping of adjacent transistors. More accurate area models for FPGA components can be developed based on actual layouts by carefully taking into account the actual connectivity and grouping of adjacent transistors.

7.2 Future Research

- Create an open source version of the layouts of the actual FPGA building blocks, so their actual layout area can be used to achieve a highly accurate ranking of the implementation area of FPGA architectures built upon these layouts.

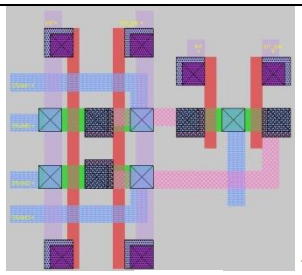
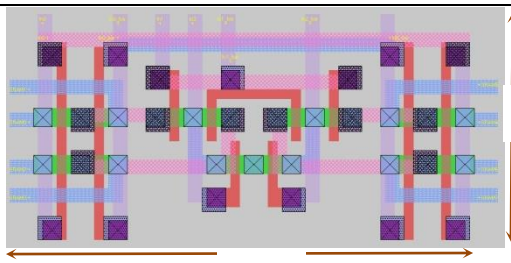
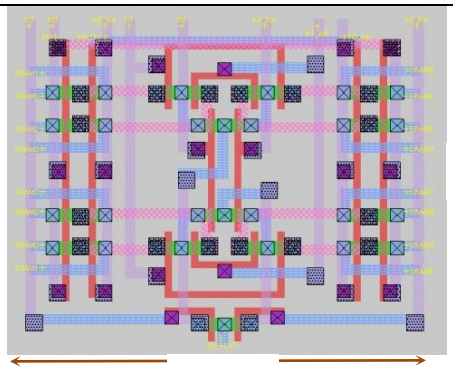
- Study the effect of placement of 2:1 multiplexer in LUTs and decoded multiplexers to further minimize area.

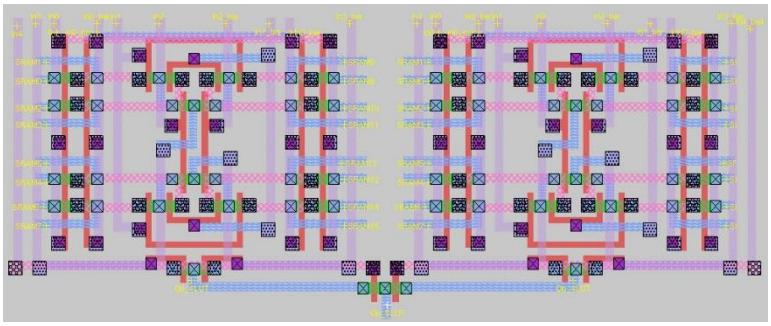
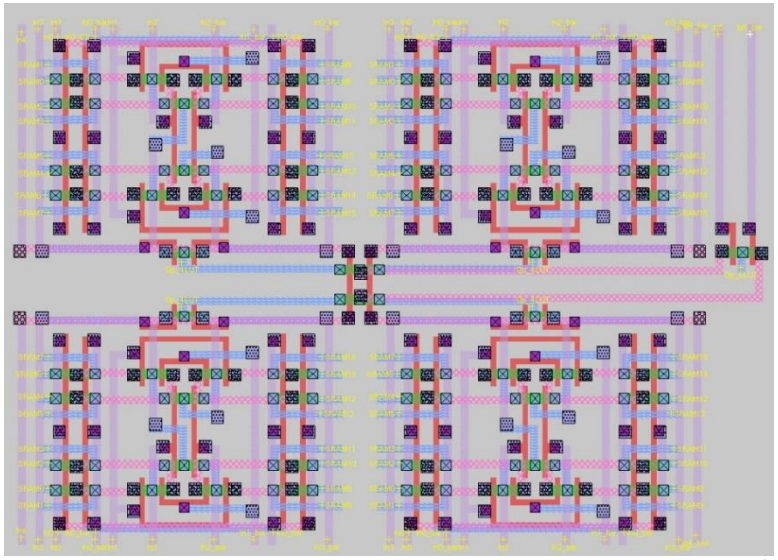
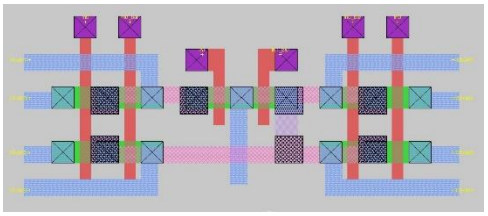
- We presented new active area models for both encoded and decoded multiplexers. Further, development of more area models for other logic block components like buffers and 1-bit full adder could be formulated depending on circuit topology and connectivity of transistors and also components from routing architecture like switch blocks and connection blocks of the FPGA architecture could be examined.

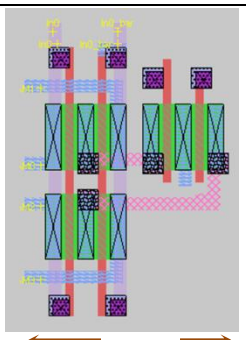
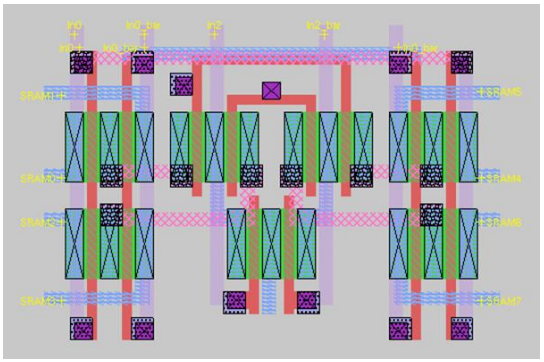
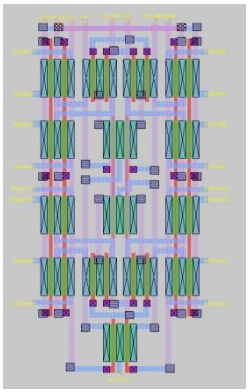
- Our current work mainly focused on layouts of FPGA logic block architecture components and routing architecture for carefully selected representative multiplexer sizes and buffers. One can also explore collecting layout data on other components such as block RAMs and multipliers for the future. Later integrate all the components of logic block and routing architecture to form an FPGA tile.

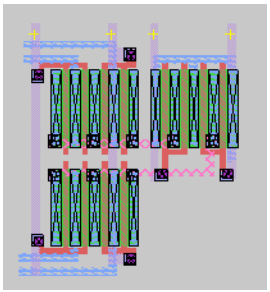
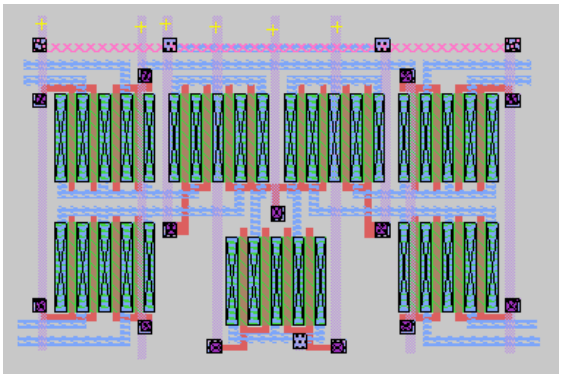
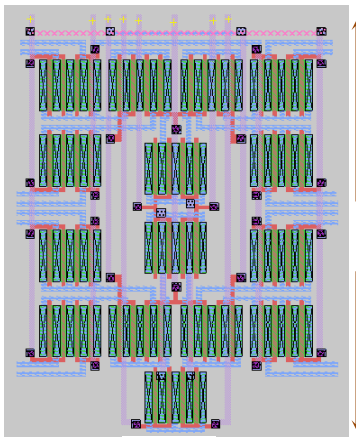
Appendix A – Layouts using 3 metal

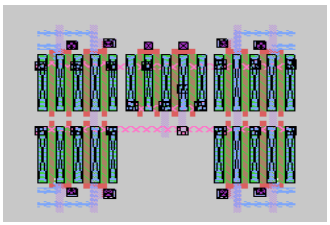
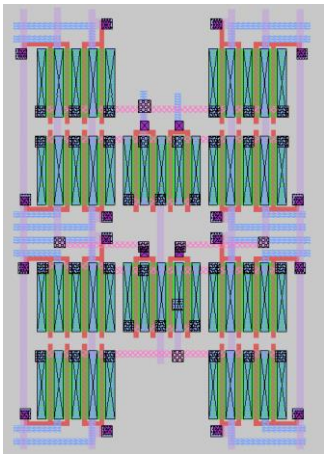
Area is measured by considering adjacent minimum inter component spacing.

No.	Component	Transistor size	Layouts
A1	2-LUT 4:1 Encoded and Decoded Multiplexer	1x	 46λ 48λ 2208λ^2
A2	3-LUT 8:1 Encoded Multiplexer	1x	 48λ 98λ 4704λ^2
A3	4-LUT 16:1 Encoded Multiplexer	1x	 96λ 124λ 11904λ^2

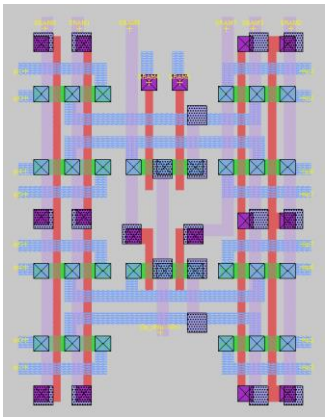
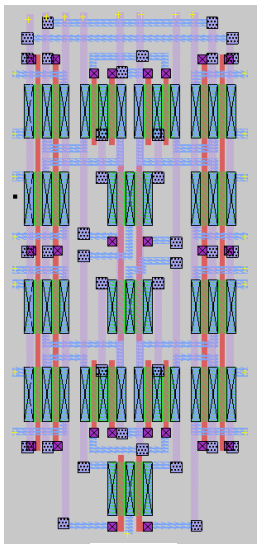
A4	5-LUT 32:1 Encoded Multiplexer	1x	 103λ 284 29252λ^2
A5	6-LUT 64:1 Encoded Multiplexer	1x	 204λ 309λ 63036λ^2
A6	8:1 Decoded Multiplexer	1x	 37λ 72λ 2664λ^2

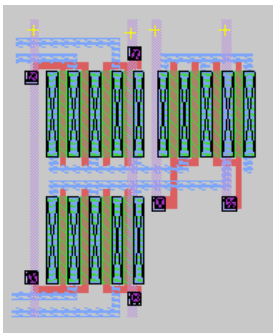
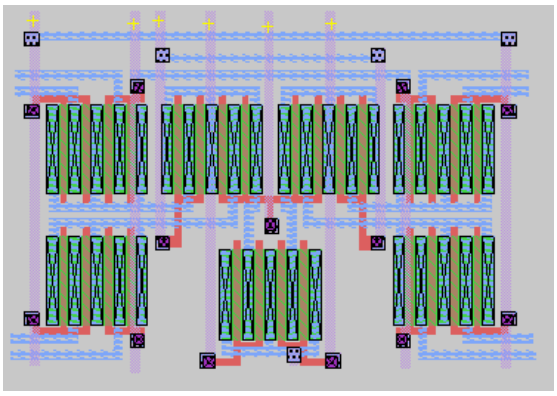
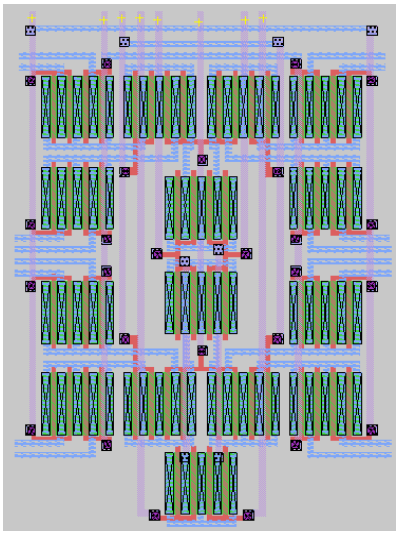
A7	2-LUT 4:1 Encoded and Decoded Multiplexer	6x	 86λ 48λ 4128λ^2
A8	3-LUT 8:1 Encoded Multiplexer	6x	 87λ 98λ 8526λ^2
A9	4-LUT 16:1 Encoded Multiplexer	6x	 226λ 98λ 22148λ^2

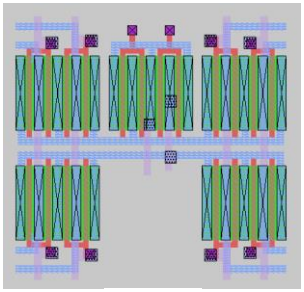
A10	2-LUT 4:1 Encoded and Decoded Multiplexer	16x	 101λ 89λ 8989λ^2
A11	3-LUT 8:1 Encoded Multiplexer	16x	 101λ 178λ 17978λ^2
A12	4-LUT 16:1 Encoded Multiplexer	16x	 256λ 180λ 46080λ^2

A13	8:1 Decoded Multiplexer	16x	 105λ 122λ $12180\lambda^2$
A14	16:1 Decoded Multiplexer	16x	 201λ 137λ $27537\lambda^2$

Appendix B – Layouts using 2 metal

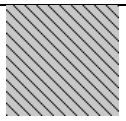
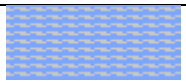
No.	Component	Transistor size	Layouts
B1	16:1 Decoded Multiplexer	1x	 111λ 74λ $7474\lambda^2$
B2	4-LUT 16:1 Encoded Multiplexer	6x	 233λ 98λ $22834\lambda^2$

B3	2-LUT 4:1 Encoded and Decoded Multiplexer	16x	 107λ 89λ 9523λ^2
B4	3-LUT 16:1 Encoded Multiplexer	16x	 124λ 180λ 22320λ^2
B5	4-LUT 16:1 Encoded Multiplexer	16x	 262λ 180λ 47160λ^2

B6	8:1 Decoded Multiplexer	16x	 111λ 120λ 13320λ^2

Appendix C – Deep Submicron SCMOS Magic rules

C.1. Common Layers

Name of layer	What layer represents	Color/Type
ndiff	ndiffusion	
pdiff	pdiffusion	
nwell	well	
pwell	well	
poly	polysilicon	
pc	contact from metal1 to polysilicon	
ndc	contact from ndiffusion to metal1	
pdc	contact from pdiffusion to metal1	
m1	metal1	
m2	metal2	
m3	metal3	
m2c	contact (connects metal 2 to metal 1)	

m3c	contact (connects metal 3 to metal 2)	

C.2. Commonly used Design Rules

Structure	**Minimum Value (λ)**
pdiff width	4
ndiff width	4
ndiff-pdiff spacing	12
nwell width	12
nwell- pdiff overhang	6
pwell width	12
pwell- ndiff overhang	6
pdc/ndc width	4
pdc/ndc – pdc/ndc spacing	3
poly width	2
poly-poly spacing	3
pc width	4
pc-pc spacing	4
poly-gate overhang	3
poly-diff spacing	1
m1 width	3
m1-m1 spacing	3
m2 width	3
m2-m2 spacing	4
m2c width	5
m2c-m2c spacing	4
m3 width	3

m3-m3 spacing	4
m3c width	5
m3c-m3c spacing	4

References

[1] V. Betz and J. Rose, "How Much Logic Should Go in an FPGA Logic Block?," *IEEE Design & Test Magazine, vol. 15, no. 1*, pp. 10-15, Jan-Mar. 1998.

[2] A. Marquardt, V. Betz and J. Rose, " Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," *in Proc. 1999 ACM/SIGDA Inter. Symp.FPGAs*, pp. 37-46, 2009.

[3] A. Marquardt, V. Betz and J. Rose, "Speed and area tradeoffs in cluster-based FPGA architectures," *IEEE Trans. VLSI Sys., vol. 8, no. 1*, pp. 84-93, Feb. 2000.

[4] G. Lemieux and D. Lewis, "Using Sparse Crossbars within LUT Clusters," *in Proc. 2001 ACM/SIGDA Inter. Symp. FPGAs*, pp. 59-68, Feb. 2001.

[5] E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density," *IEEE Trans. VLSI Sys., vol. 12, no. 3*, pp. 288-298, Mar. 2004.

[6] G. Zgheib, L.Yang, Z. Huang, D. Novo, H. P-Afshar, H. Yang, and P. Ienne. "Revisiting and-inverter cones". *in Proc. 2014 ACM/SIGDA Inter. Symp. FPGAs*, pp.45-54, Feb. 2014.

[7] V. Betz and J. Rose, "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density," *in Proc. 1999 ACM/SIGDA Inter. Symp. FPGAs*, pp. 59-68, Feb. 1999.

[8] G. Lemieux et al., "Directional and Single-Driver Wires in FPGA Interconnect," *in Proc. 2004 IEEE Inter. Conf. FPT*, pp. 41–48, Dec 2004.

[9] A. M. Smith, G. A. Constantinides, and P. Y. K. Cheung, "Area estimation and optimization of FPGA routing fabrics,"*in Proc. 2009 Inter. Conf. FPL*, pp. 256-261, Sept 2009.

[10] P. Chen and A. Ye, "The effect of multi-bit correlation on the design of field-programmable gate array routing resources," *IEEE Trans. VLSI. Syst., vol. 19, no. 2,* pp. 283-294, Feb. 2011.

[11] V. Betz, J. Rose, and A. Marquardt, Architecture and CAD for Deep-Submicron FPGAs, Boston: Kluwer Academic Publishers, Feb. 1999.

[12] F. Khan and A. Ye, "Measuring the Accuracy of Minimum Width Transistor Area in Estimating FPGA Layout Area", *in Proc. FCCM 2015*, pp. 223-226.

[13] Zynq-7000 All Programmable SOC Overview, Xilinx Inc, Sanclose, CA,2014.

[14] Meeting the Performmance and Power Imperative of Zettabyte Era with Genaeration 10, Altera Corp.,Sanclose, CA,2014.

[15] J.Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose,and V. Betz, "VTR 7.0: Next Generation Architecture and CAD Systemfor FPGAs," *ACM TRETS, vol. 7, no. 2*, June 2014, pp. 6:1–6:30

[16] I. Kuon, A. Egier, and J. Rose. "Design, layout and verification of an FPGA using automated tools," *in Proc. 2005 ACM/SIGDA Inter. Symp. FPGAs*, pp. 215-226, Feb. 2005.

[17] K. Padalia, R. Fung, M. Bourgeault, A. Egier, and J. Rose. "Automatic transistor and physical design of FPGA tiles from an architectural specification," *in Proc. 2003 ACM/SIGDA Inter. Symp. FPGAs*, pp 164-172, Feb. 2003.

[18] C. Chiasson and V. Betz. "COFFE: Fully-Automated Transistor Sizing for FPGAs", *in Proc. 2013 IEEE Inter. Conf. FPT,* pp. 34-41, Dec. 2013.

[19] MOSIS Scalable CMOS, MOSIS Integrated Circuit Fabrication Service, Marina del Rey, CA, 2009.

[20] Magic VLSI Layout Tool, http://opencircuitdesign.com/, 2015

[21] N. H. E. Weste and D. Harris, CMOS VLSI Design Circuits and Systems Perspective, Pearson Addison-Wesley, 2005.

[22] D. Lewis et al., "The Stratix II™ Logic and Routing Architecture," in *Proc. 2005 ACM/SIGDA Inter. Symp. FPGAs*, pp. 14-20, Feb. 2005.

[23] A. Gupta, J. P. Hayes, "Optimal 2-D cell layout with integrated transistor folding," *in Proc. 1998 IEEE/ACM Inter. Conf., Comp.-Aided Des.*, pp. 128-135, 8-12, Nov. 1998.

[24] C. Chen et al., "Efficient FPGAs using Nanoelectromechanical Relays," *in Proc. 2010 ACM/SIGDA Inter. Symp. FPGAs*, pp. 273-282, Feb. 2010.

[25] C. Chiasson and V. Betz, "Should FPGAS abandon the pass-gate?," *Field Programmable Logic and Applications (FPL), 2013 23rd International Conference on* , pp.1-8, 2-4 Sept. 2013.

[26] I. Kuon, R. Tessier, and J. Rose. 2008. *FPGA Architecture: Survey and Challenges*. Found. Trends *EDA*, pp.135-253, Feb 2008.

[27] A. H. Lam. *An Analytical Model of Logic Resource Utilization for FPGA Architecture Development*. Master's thesis, University of British Columbia, 2010

[28] J. Rabaey, *Digital Integrated Circuits: A Design Perspective.* Upper Saddle River, NJ: Prentice Hall, 1996.

[29] L. E. Han, V. B. Perez, M. L. Cayanes and M. G. Salaber, *CMOS Transistor Layout KungFu*, http://www.eda-utilities.com/, 2005.

[30] C. Saint, J. Saint, *IC Mask Design: Essential Layout Techniques*, McGraw-Hill Publishers, May 2002.

[31] D. Clein, *CMOS IC LAYOUT: Concepts, Methodologies, and Tools*, Newnes Publishers, Dec, 1999.

[32] R. J. Baker, *CMOS Circuit Design, Layout, and Simulation 2nd ed.*, Wiley-IEEE Press, 2005.

[33] IRSIM- Open Circuit Design, http://opencircuitdesign.com/irsim/

[34] G. Lemieux and D. Lewis. 2002. "Circuit design of routing switches". *in Proc. 2002 ACM/SIGDA 10th Inter. Symp. FPGAs*. pp 19-28, 2002.

[35] R. Zurawski, Embedded Systems Design and Verification, CRC Press, Jun 25, 2009.

[36] I. Kuon and J.Rose. "Measuring the Gap between FPGAs and ASICs". *In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume 26, Number 2, pp. 203-215, February, 2007.

[37] H. Wong, V. Betz, and J.Rose. 2011. "Comparing FPGA vs. custom CMOS and the impact on processor microarchitecture", in *Proc. of the 19th ACM/SIGDA Inter.Symp. Field programmable gate arrays (FPGA).*, pp 5-14, 2011

[38] A. H. Pereira, V. Betz "CAD and Routing Architecture for Interposer-based multi-FPGA systems", in *Proc. of the 22nd ACM/SIGDA Inter. Symp. Field programmable gate arrays (FPGA)*, pp. 75- 84, 2014.

[39] VPR and T-VPack User's Manual, http://www.eecg.toronto.edu/vpr/VPR_5.pdf

[40] COFFE User's Manual,

http://www.eecg.utoronto.ca/~charlesc/COFFE_User_Manual.pdf

[41] Ognjen Šćekić, *FPGA Comparative Analysis*, 2005, EFT, University of Belgrade

[42] P. Chow, Soon Ong Seo, J. Rose, K. Chung, G. Paez-Monzon and I. Rahardja, "The design of an SRAM-based field-programmable gate array. I. Architecture," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 2, pp. 191-197, June 1999.

[43] P. Chow, Soon Ong Seo, J. Rose, K. Chung, G. Paez-Monzon and I. Rahardja, "The design of a SRAM-based field-programmable gate array-Part II: Circuit design and layout," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 3, pp. 321-330, Sept. 1999.

[44] J.Luu, C. McCullough, S. Wang, S. Huda, Y. Bo, C. Chiasson, K. Kent, J. Anderson, J. Rose, and V. Betz ,"On Hard Adders and Carry Chains in FPGAs", *in Proc. FCCM 2014*, pp. 52-59.

[45] S. A. Chin, J. Luu, S. Huda, J. H. Anderson, "Hybrid LUT/Multiplexer FPGA Logic Architectures", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 24, no. 4,* 2016, pp.1280 - 1292.

[46] J.Luu, Ph.D Thesis, University of Toronto.

[47] J. Cong, Y.Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", *in IEEE Transactions on Computer-Aided Design, Volume 13, Issue 1,* January, 1994, pp. 1-12.

[48] V.Betz and J.Rose. "FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density". i*n Proc. of the International Symposium on Field-Programmable Gate Arrays*, February, 1999, pp. 59-68.

[49] M. Purnaprajna, P Ienn. "A Case for Heterogeneous Technology-Mapping: Soft versus Hard Multiplexers". 2013 *IEEE 21st Annual International Symposium on Field-Programmable Custom Computing Machines*, Seattle, WA, 2013, pp. 53-56.

[50] Guy Lemieux and David Lewis. Design of Interconnection Networks for Programmable Logic. Kluwer Academic Publishers, 2004.

[51] G. Zgheib, M. Lortkipanidze, M. Owaida, D. Novo, and P. Ienne. 2016. "FPRESSO: Enabling Express Transistor-Level Exploration of FPGA Architectures". *in Proc. of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (FPGA '16). ACM, New York, pp. 80-89.

[52] H. Parandeh-Afshar, H. Benbihi, D.Novo, and P. Ienne. 2012. "Rethinking FPGAs: elude the flexibility excess of LUTs with and-inverter cones". *In Proc. of the ACM/SIGDA international symposium on Field Programmable Gate Arrays (FPGA '12)*. ACM, New York, NY, USA, pp.119-128.

[53] Spartan-6 FPGA User Guides, Spartan-6 FPGA Configurable Logic Block User Guide, http://www.xilinx.com/support/documentation/user_guides/ug384.pdf

[54] 7 Series FPGAs Configurable Logic Block (v1.6), Xilinx, 2014.

[55] 7 Series FPGAs Overview (v1.15), Xilinx, 2014.

[56] Virtex-4 Family Overview (v3.1), Xilinx, 2010.

[57] Virtex-5 Family Overview (v5.0), Xilinx, 2009.

[58] Virtex-6 Family Overview (v2.4), Xilinx, 2012.

[59] Stratix Device Handbook, Volume 1, Altera Corporation, 2005.

[60] Stratix II Device Handbook, Volume 1, Altera Corporation, 2007.

[61] Stratix III Device Handbook, Volume 1, Altera Corporation, 2010.

[62] Stratix IV Device Handbook, Volume 1, Altera Corporation, 2012.

[63] Stratix V Device Overview, Altera Corporation, 2014.

[64] A. Gupta and J. P. Hayes, "A hierarchical technique for minimum-width layout of two-dimensional CMOS cells," *Proceedings Tenth International Conference on VLSI Design*, Hyderabad, 1997, pp. 15-20.

[65] D. Hill and N. S. Woo, "The benefits of flexibility in lookup table-based FPGAs," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 2, Feb 1993, pp. 349-353.

[66] M. B. Abdelhalim and S. E. D. Habib, "Fast FPGA-based area and latency estimation for a novel hardware/software partitioning scheme," 2008 *Canadian Conference on Electrical and Computer Engineering*, Niagara Falls, ON, 2008, pp. 775-780.

[67] Siew-Kei Lam, Wen Li; Srikanthan, T. “High level area estimation of custom instructions for FPGA-based reconfigurable processors,” *6th International Conference on Information, Communications & Signal Processing,* 10-13 Dec. 2007, pp.1-5.

[68] A. Nayak, M. Haldar, A. Choudhary and P. Banerjee, 2002, “Accurate area and delay estimators for FPGAs,” *Proceedings Design, Automation and Test in Europe Conference and Exhibition,* Paris, 2002, pp. 862-869.

[69] L. Deng, K. Sobti and C. Chakrabarti. 2008. “Accurate models for estimating area and power of FPGA implementations,” *IEEE International Conference on Acoustics, Speech and Signal Processing*, Las Vegas, NV, 2008, pp. 1417-1420.

[70] D. Kulkarni, Walid A. Najjar, R. Rinker, and Fadi J. Kurdahi., “Fast area estimation to support compiler optimizations in FPGA-based reconfigurable systems*,” 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2002. Proceedings*, pp. 239- 247.

[71] D. Kulkarni, Walid A. Najjar, R. Rinker, and Fadi J. Kurdahi. 2006. “Compile-time area estimation for LUT-based FPGAs*”. ACM Trans. Des. Autom. Electron. Syst. 11*, 1 (January 2006), pp. 104-122.

[72] M. Xu and F. Kurdahi. “Area and Timing Estimation for Lookup Table Based FPGAs”. In *Proceedings of the 1996 European conference on Design and Test* (EDTC '96). IEEE Computer Society, Washington, DC, USA, pp. 151-157.

[73] M. Xu and F. Kurdahi, “ChipEst-FPGA: a tool for chip level area and timing estimation of lookup table based FPGAs for high level applications,” *Proceedings of the ASP-DAC '97. Asia and South Pacific,* 28-31 Jan 1997, pp.435-440.

[74] C. Brandolese, W. Fornaciari and F. Salice, “An area estimation methodology for FPGA based designs at systemc-level,” 2004, *Proceedings. 41st Design Automation Conference*, San Diego, CA, USA, 2004, pp. 129-132.

[75] X. Wei, J. Chen, Q. Zhou, Y. Cai, J. Bian, X. Hong, "MacroMap: A technology mapping algorithm for heterogeneous FPGAs with effective area estimation," *Field Programmable Logic and Applications*, 8-10 Sept. 2008, pp.559-562.

[76] V. Betz and J. Rose, "Circuit design, transistor sizing and wire layout of FPGA interconnect," In *Proceedings of the IEEE Custom Integrated Circuits, 1999.*, San Diego, CA, 1999, pp. 171-174.

[77] K. Roy, "Optimum Gate Ordering of CMOS Logic Gates using Euler Path Approach: Some Insights and Explanations", *Journal of Computing and Information Technology* – CIT15,2007, pp. 85-92.

[78] B.Bas, VLSI Lecture Notes, University of California, Davis.

[79] K. Huang, Y. Hu, X. Li, B. Liu, H. Liu and J. Gong, 2012. "Off-path leakage power aware routing for SRAM-based FPGAs," *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Dresden, 2012, pp. 87-92.

[80] C. Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson and B. Hutchings, 2011 "RapidSmith: Do-It-Yourself CAD Tools for Xilinx FPGAs," *21st International Conference on Field Programmable Logic and Applications*, Chania, 2011, pp. 349-355.

[81] http://rapidsmith.sourceforge.net

[82] A. Singh and M. Marek-Sadowska. 2002. "Efficient circuit clustering for area and power reduction in FPGAs". *In Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays (FPGA '02)*. ACM, New York, NY, USA, 59-66.

[83] A. Lam, S. J. E. Wilton, P. Leong and W. Luk, "An analytical model describing the relationships between logic architecture and FPGA density," 2008 *International Conference on Field Programmable Logic and Applications*, Heidelberg, 2008, pp. 221-226.

[84] A. A. Aggarwal and D. M. Lewis, "Routing architectures for hierarchical field programmable gate arrays," *Proceedings 1994 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, Cambridge, MA, 1994, pp. 475-478.

[85] Haixia Gao, Yintang Yang, Xiaohua Ma and Gang Dong, "Analysis of the effect of LUT size on FPGA area and delay using theoretical derivations," *Sixth international symposium on quality electronic design (isqed'05)*, 2005, pp. 370-374.

[86] S. Yan, D. Li, L. Wang, Y. Xiao and M. Tang, "A novel methodology of layout design by applying euler path", 2010, *10th IEEE International Conference on Solid-State and Integrated Circuit Technology*, Shanghai, 2010, pp. 818-820.

[87] F.Khan and A.Ye, "An Empirical Analysis of the Fidelity of VPR Area Models", 2016, *24th IEEE International Symposium on Field-Programmable Custom Computing Machines*, Washington, DC, May 2016, pp.138.

[88] F. Khan and A. Ye, "An evaluation on the accuracy of the minimum width transistor area models in ranking the layout area of FPGA architectures," 2016, *26th International Conference on Field Programmable Logic and Applications (FPL)*, Lausanne, 2016, pp. 1-11.