VLSI Implementation of a Run-time Configurable Computing Integrated Circuit - The Stallion Chip

Yingchun He

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical Engineering

Dr. Peter M. Athanas, Chair

Dr. William H. Tranter

Dr. Mark Jones

July 1998

Blacksburg, Virginia

Keywords: VLSI, Layout, Run-time Configurable Computing

VLSI Implementation of a Run-time Configurable Computing Integrated Circuit The Stallion Chip

He, Yingchun

(ABSTRACT)

Reconfigurable computing architectures are gaining popularity as a replacement for general-purpose architectures for many high performance embedded applications. These machines support parallel computation and direct the data from the producers of an intermediate result to the consumers over custom pathways. The Wormhole Run-time Reconfigurable (RTR) computing architecture is a concept developed at Virginia Tech to address the weaknesses of contemporary FPGAs for configurable computing. The Stallion chip is a full-custom configurable computing "FPGA"-like integrated circuit with a coarse grained nature. Based on the result of the first generation device, the Colt chip, the Stallion chip is a follow-up configurable computing chip. This thesis focuses on the VLSI layout implementation of the Stallion chip. Effort has been made to explain many facts and advantages of the Wormhole Configurable Computing Machine (CCM). Design techniques, strategies, circuit characterization, performance estimation, and ways to solve problems when using CAD layout design tools are illustrated.

Acknowledgments

Firstly, I would like to thank my advisor, Dr. Peter Athanas, for giving me the opportunity to work on this interesting VLSI project. Without his help and encouragement, this thesis would not have been possible. His wisdom, hard working nature, and willingness to understand and help students will remain in my mind all my life. Many thanks go to Dr. William H. Tranter and Dr. Mark Jones for taking time out of their busy schedule to be on my committee. I would also like to thank Dr. Midkiff who once served as my temporary advisor. It is Dr. Midkiff who encouraged me to make the decision to choose computer engineering as my major when I came to Virginia Tech three years ago.

No words in the world can express my gratitude to my parents and sister Yingwen. If there were a second life, I wish we could join as a family again. I am also thankful to Farooq Azam for his help and encouragement throughout my research work.

Finally, thanks are due to my research project partner, Mr. Weimin You, for his constant assistance in research and other matters. By the same time, thank Ms. Connor Forren and Mr. Bruce Watson in the writing center (Virginia Tech) for their help.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Designing the layout of an integrated circuit (IC) is the process of assigning geometric shape, size, and position to the components (transistors and connections) used for device fabrication [1]. With the rapid development of VLSI technology, it is possible to make chips smaller and smaller; however, the existing layout synthesis tools generally are still not robust enough to generate a layout which can compete with those made manually. This thesis focuses on the VLSI layout implementation of the reconfigurable computing device, the Stallion chip. The Stallion chip is the follow-up chip based on the first generation device called Colt. Colt successfully proved the concept of Wormhole Run-time reconfigurability.

1.1 Motivation

Configurable computing machines have recently attracted attention from designers seeking high computational performance [2]. CCMs achieve high performance through deep computational pipelines, concurrent execution, rapid reconfiguration, and inherent data flow. Many applications can exploit the properties of these systems, such as high-speed digital signal processing (DSP) and image processing. Field programmable gate arrays (FPGAs) are a commonly used device to implement contemporary CCMs; however, these devices are not intended for computation and have slow reconfiguration times, which limits their efficiency. Wormhole run time reconfiguration (RTR) computing machine performs partial hardware reconfiguration. In contrast to the fine-grain FPGA, the coarse-grain RTR handles the routing and programming locally rather than globally, thus giving better propagation delay and allowing faster operation. Colt is the chip which embodies the Wormhole RTR concept [3]. It employs an array of FPGAs that can efficiently implement applications, and is data flow in nature. It has fixed 16-bit word size variables along with 1-bit pathways. Stallion, the follow-up chip with Colt, remains the same word size but has additional functional options such as multi-cast in addition to broadcast and point-to-point communications. It consists of six 16-bit bi-directional data ports, two meshes of Functional Units (FU), four 16-bit multipliers producing 32-bit results each and a Smart Crossbar.

1.2 Contributions

The implementation of an integrated circuit from a structural schematic design is a task that is both time-consuming and in need of necessary knowledge about the designed circuit. Moreover, much time was spent in learning how to master the design tools. A good layout designer will not only make sure that the circuit works but also save as much area as possible. The author tried as hard as she could to follow the classical techniques to reduce the design time and enhance the chances of successful results. This thesis gives a description of those techniques that the author used during her research works, as well as her experience in solving unavoidable problems such as debugging the layout errors.

1.3 Organization

This thesis consists of five chapters. Chapter 2 introduces configurable computing and its background. Chapter 3 presents the Colt/Stallion chip architecture. Chapter 4 discusses the general rules of designing layout, which the author followed throughout her research. Chapter 5 gives out a description about the layout of the Colt/Stallion chip, pointing out the difference between the Stallion and Colt chips. The last chapter, Chapter 6 summarizes the results. Additional information about how to solve the problems in layout design is included in the appendix.

Chapter 2

Background of Configurable Computing Machines

Configurable computing machines are the result of trying to find the right balance between speed and generality. Various alternative approaches could be taken to solve signal processing and image processing problems that require substantial computation. This chapter will give a brief introduction of these methods and their relative merits. The alternative technologies examined are ASIC, DSP, FPGA, and Wormhole RTR CCM.

2.1 ASIC

ASIC stands for Application Specific Integrated Circuit. An ASIC is developed on the initiative of a user for a particular task and cannot be found as a standard circuit on the

market. Because of this focused functionality nature, ASICs generally have achieved the most efficient use of hardware resources in terms of speed and silicon area. However, an ASIC is also costly. Furthermore, it is impractical to make changes once an ASIC has been completed. An efficient ASIC for one application is generally useless for another even if the two applications have much in common.

2.2 DSP Solutions

DSP chips have been developed in recent years in an attempt to address the extensive arithmetic computational requirements of digitally based communication systems. The most basic architectural component of a DSP embeds the fast multiplier/accumulator (MAC) into the data path, rather than placing it on a co-processor of a micro-controller. DSPs use extensive pipelines, several independent on-chip memories, parallel function units and hardwired (rather than micro-programmed) pathways. The dominating architectural feature of DSPs is the so-called Harvard architecture and its variants. Harvard architectures augment classic Von Neumann [4] machines in that it has separate bus lines for data and programming instructions. The main shortcoming of the DSP is that in DSP, too much silicon area is used for overhead processing and non-computational tasks. Due to the inherent sequential nature of DSPs, concurrency is poorly exploited.

2.3 FPGA

A field programming gate array, or FPGA [5] is a highly tuned hardware circuit which can be modified at almost any point during use. FPGA chips are programmable at logical gate levels. They consist of arrays of configurable logic blocks that implement the logical functions of gates and have programmable interconnect. Logic gates are like switches with multiple inputs and a single output. They are used in digital circuits to perform all basic binary operations such as AND, NAND, OR, NOR and XOR. Sending commands to the FPGAs can alter both the logical functions of the gate performed within the logic blocks, and the connections between the blocks. FPGAs represent a new method of designing custom ICs. Unlike conventional gate arrays or ASICs, where standard gate arrays are configured during manufacture, FPGAs can be user-programmed, like a RAM or ROM, long after the integrated circuit has left the factory. They represent the logical development from Programmable Logic Devices (PLD), because they have a much greater logic capacity and offer greater flexibility.

2.4 Wormhole CCM

CCM stands for configurable computing machine. Over the past decade, much in reconfigurable architectures has been proposed and built. Examples of commercial FPGA-based reconfigurable systems are Splash-2 [6], GigaOps [7], and Wildfire [8]. For an efficient platform, the reconfiguration time should be as short as possible. However, conventional FPGAs cannot meet this requirement since they generally are scan-chain programmed and rely on

global control strategies that present a fundamental bottleneck to the potential bandwidth of configuration information flowing into them. The wormhole run-time reconfigurable devices Colt/Stallion, have been developed at Virginia Tech to address these weaknesses. Instead of using a fine-grained structure, the Colt/Stallion chips are coarse grained. Consequently, by using a distributed control and data driven partial run-time reconfigurable scheme, it is possible to create and modify custom computational pathways. In Wormhole RTR system, streams play an important role. A stream is an independent self-steering concatenation of programming information and operand data that interacts with other streams within the architecture to perform a given computational problem. Directed by the header information stored in the front of a stream, the data is transported to the desired path where related operations are executed. The top portion of the header will be stripped off from the stream and used to configure both a computational pathway through the system and the operations to be performed by the various computational elements along the path. Therefore, the size of the header diminishes as the stream propagates through the system. Steering of the stream is largely a locally controlled process because the stream flows from one unit to only a few others. Besides, since each stream has its own header information, a system with a high degree of parallelism without the communications overhead is possible [9].

Chapter 3

The Stallion Chip Design Approach

The Stallion chip is a follow-up chip based on the Colt integrated circuit chip[10]. The major architectural features of the Stallion chip remains the same as that of the Colt chip; however, in order to enhance its functionality, changes have been introduced. Knowledge of a particular chip is always necessary to a layout designer since it will help him/her in the process of designing, especially when working critical path, buffering, and other VLSI concerns. This chapter gives a brief description of the the Stallion chip.

3.1 Overview of the The Stallion Chip

The Stallion chip is different from the Colt in that it is composed of two meshes of processing elements, one crossbar, six 16-bit bi-directional data ports, and four 16x16 bits multipliers which producing a 32-bit result [11]. The Colt, on the other hand, is composed of one mesh,

one crossbar, six data ports, and one multiplier as shown in Figure 3.1. In addition to the point-to-point and broadcast communications within the chip, multi-cast communication will be feasible within the Stallion chips through the smart crossbar. Also, the Stallion chip is able to stall incoming data if it arrives too fast for the chip to handle the related operations. Under the same situation in the Colt chip, data would be dropped and lost. Figure 3.2 shows the basic architecture of the Stallion CCM.

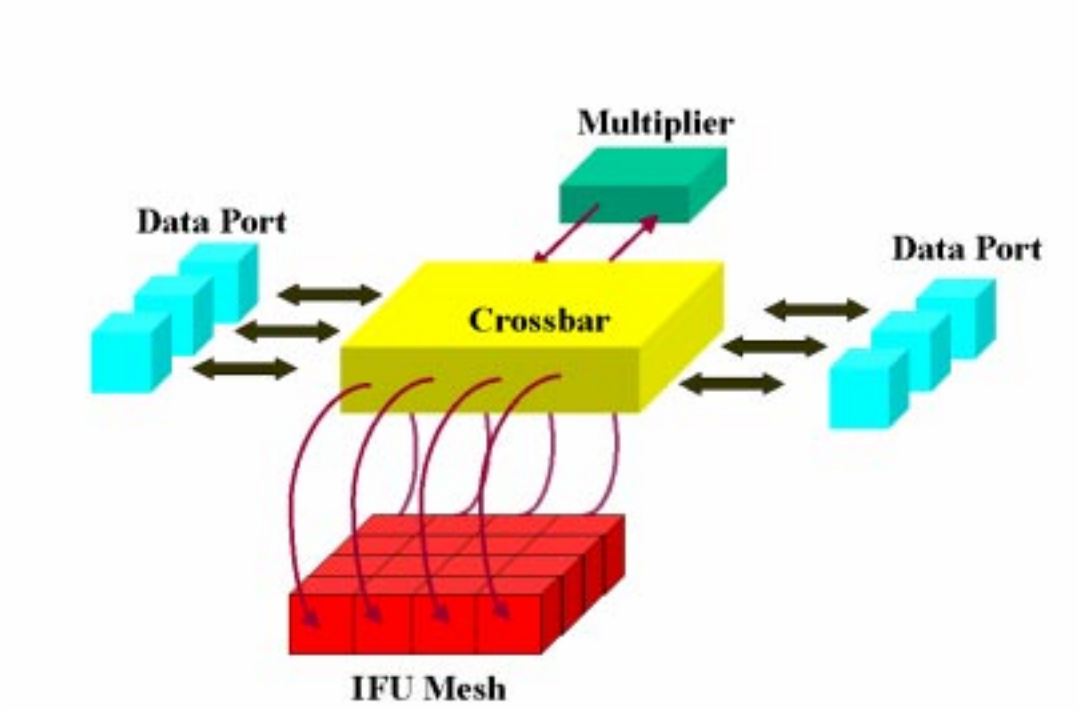

Figure 3.1: The basic architecture of the Colt chip.

As shown in Figure 3.2, the input data flows into one of the six data ports from the outside, directed by the routing information brought with the data itself. After that, it passes through

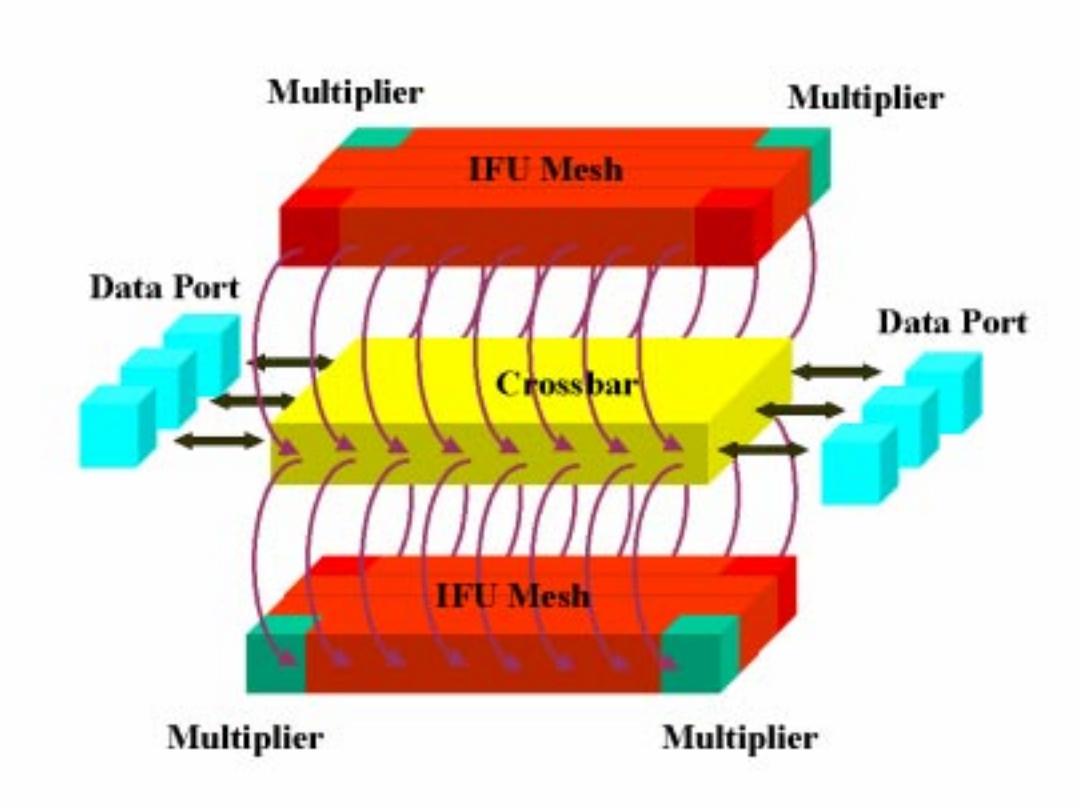

Figure 3.2: The basic architecture of the Stallion chip.

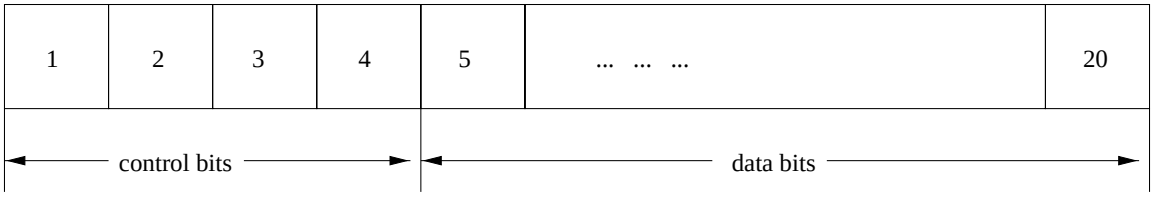

Figure 3.3: The stream data format.

a smart crossbar to the corresponding IFU, then to the respective FU. After execution, the resulting data goes back to the crossbar and then either flows through the data ports and exists or passes back to the IFU again.

There are twenty bits of data coming in parallel to the data port, four of which are controlling bits. The rest are interpreted as for data. Figure 3.3 shows the format of the data. From Figure 3.3, it is clear that computation is performed on fixed 16-bit word size variables. The 1-bit pathways are not shown.

3.2 Components and Functions of The Stallion Chip and Their Functions

3.2.1 Data Ports

As in the Colt chip, there are six data ports in the Stallion chip, each with twenty pins, of which four bits are used as controlling signals; the rest are for data. Each data port is bi-directional and can be programmed in three major modes: Raw Mode, Synchronization Mode, and Loop Mode.

- **Raw Mode:**

  Raw mode allows the data section of the stream to flow into the chip without flow control. In raw mode, data is merely accepted and passed on by the port. A flag is set

when the existing result is void because of invalid inputs.

- **Synchronization Mode:**

  Synchronization mode ensures that complete sets of valid data will enter the chip simultaneously and it is used for standard flow control. A valid data in one stream will be held until another valid data from another stream arrives.

- **Loop Mode:**

  Loop mode is an extension of synchronization mode. The only difference between loop mode and synchronization mode is that loop mode allows only one set of valid operands to exist within the chip pipeline at any given time. Until the valid result comes out no inputs will be accepted, although both of them are valid data.

3.2.2 Multiplier

As indicated by its name, the multiplier is dedicated to handle 16-bit by 16-bit operands and produce a 32-bit result with a latency of two clock cycles. Unlike the original Colt chip, the Stallion chip has four multipliers embedded in two meshes.

3.2.3 Crossbar

The meshes of the FU and four 16-bit multipliers are connected through a "Smart Crossbar." The Smart Crossbar has sixteen inputs, six of which are from the data ports, four from the bottom of the computational mesh, and the remaining two come from the 32-bit output of

the multipliers. Of the sixteen outputs, six are routed back to the data ports, eight are used by mesh, and two come to the multiplier. Unlike conventional networks [12], which apply central control, this crossbar is designed for distributed control, allowing multiple points be switched simultaneously and providing full connectivity from any input to any output. Besides, the internal cell structure of the crossbar in the Stallion chip is enhanced from that in the Colt chip.

3.2.4 Mesh

The computational mesh is the heart of the Stallion chip and can be subdivided into Interconnected Functional Units (IFUs). The IFU can be further decomposed by FUs. FU is composed of an ALU, a barrel shifter register, six control registers, registers, a FU state machine, and flags (Figure 3.4).

Figure 3.5 shows how the IFU modules fit together to form the mesh. Also the mesh is composed of two buses. The dark arrows are the bi-directional buses, which connect the IFU to its four neighbors. The dash line presents the skip buses. The skip bus is a means of sending operands over one or more IFUs in a single clock. By introducing the skip bus, it is easier to map more complicated operations onto the mesh.

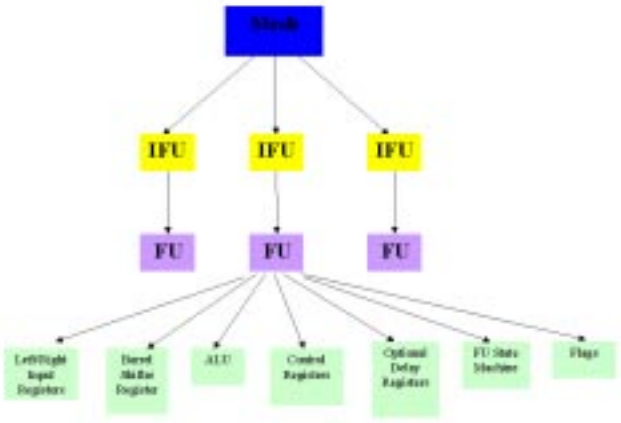

Figure 3.4: Mesh structure.

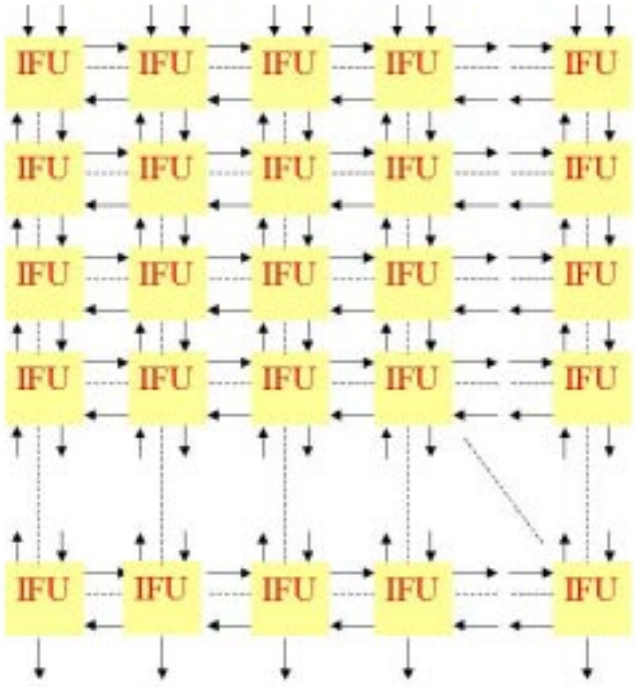

Figure 3.5: Mesh topology for the Stallion chip.

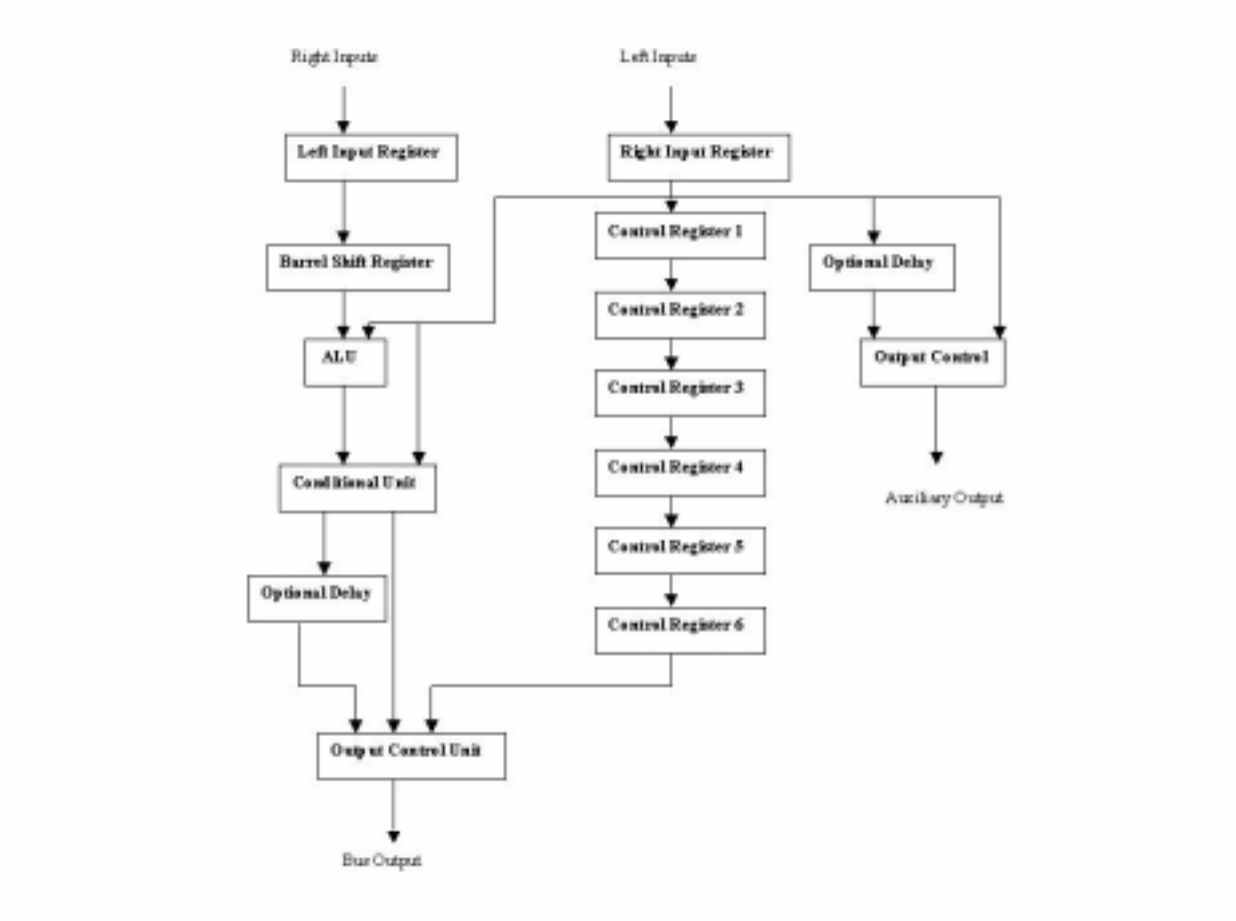

Figure 3.6: Structure of Functional Unit.

Functional Unit

The FU consists of an ALU, a barrel shift register, registers, and other control logical blocks. Their basic connections are listed in Figure 3.6. As shown in Figure 3.6, two streams of 16-bit data arrive from the data port entering the left and right registers respectively. The data through the left register goes into a barrel shifter to be shifted according to computational requirements such as multiplication, then, along with the inputs from the right input, the data is entered into ALU. Based on the lookup table, the ALU produces the operational result. The conditional unit will choose either this result or the data from the right input register as input. Either this input, with or without delay, or the input of the configurable Register 6 will be chosen to go through the output control unit depending on whether the FU is in normal mode or programming mode. On the other side, data coming from the right input register can be used as input to the ALU or to the configurable register. In addition, there is another output called Auxiliary output. This is used especially for DSP computation.

Components of Functional Unit and Their Functions

The main components of FU and their functions are listed below for reference.

Left/Right Input Register Load a new value when enabled.

Barrel Shifter Register Make it easy to implement some functions requiring shifting such as multiplication.

ALU Support Arithmetic logic operation based on a lookup table.

Configurable Control Register Store the control information.

Optional Delay Register Provide the clock latency to synchronize the path lengths of different pipeline streams when necessary.

Chapter 4

General Rules for Layout Design

With the rapid improvement of CMOS processing technology, integrated circuit area is becoming smaller and smaller for a given circuit, while the speed is becoming faster and faster. This means more transistors, more density, and more powerful chips. However, no matter how complex a circuit is, a hardware design can always be broken down into six levels: the system level, chip level, register level, gate level, circuit level, and layout level. Of the six levels, the layout level is the lowest. In the layout level, transistors are represented as geometrical figures with dimension such as length, width, and position. Chips are fabricated according to the description at the layout level.

4.1 Synthesis and Analysis

VLSI design is one of the most complicated processes in hardware design. Because of its complexity, the use of a Computer Aided Design (CAD) tool is unavoidable in VLSI design. There are two types of CAD tools: namely synthesis tools and analysis tools. Synthesis tools generally transfer design from a top level to lower level. *Design Analyzer* in Synopsys and *icfb* layout synthesis in Cadence are examples of synthesis tools. *Design Analyzer* transfers a circuit from behavior level to the gate level. At the same time, the *icfb* layout synthesis tool transfers a circuit from schematic cell view, which ranges from the gate level or higher, to the layout level. Analysis tools are used to test design quality. One example of an analysis tool is the *SPICE* and *Verilog* simulation tool. By using *SPICE* and *Verilog*, the chip is simulated and to make sure that it satisfies the design requirements such as logic verification, performance analysis, and test development [13]. Layout implementation uses both Synthesis and Analysis tools.

4.2 Full Custom Design Versus Half-custom Design

Layout design can be divided into full custom design and half-custom design. Full custom design is designed for a specific circuit. Half-custom design is designed for general usage and user may choose one of the options provided by the circuit or add components when necessary. Full custom design takes a long time but is more area efficient. On the other hand, half-custom design is easy to implement and more flexible. The Stallion chip is a full

custom design.

4.3 Design Strategies

A successful layout design not only satisfies the timing and area requirements but also shortens design time [14]. To achieve this goal, a designer needs to follow some commonly used rules which have already been practiced and proved by many previous designers. Although IC technology is becoming more and more complex, these basic rules generally work well because no matter how much change there is, there still exists a lot of commonality among these techniques. By following these commonalities, it is much easier to construct a successful layout. In the following sections, some of the classical techniques for reducing the complexity of IC designs will be briefly introduced.

4.3.1 Hierarchy

Hierarchy is one of the characteristics of hardware design. As mentioned at the beginning of this chapter, a design hierarchy divides the design process into six levels: the system, chip, register, gate, circuit and layout level [15]. The hierarchy in the layout implementation subdivides the layout module into sub-modules and repeats this operation until it becomes simple enough to understand the details of the design. This is much like the subroutine in computer program where large program is divided into several sections and each section is further divided into subroutines. Since there are different forms of representing the chip,

parallel hierarchy is employed in this research work. A parallel hierarchy divides the layout design into several different cell views. As with the Colt chip, the Stallion chip has three types of cell view, schematic, symbolic ,and layout. Each type of cell view is hierarchical. The schematic cell view describes the circuit structure, the symbolic cell view is for high level modeling, and the layout cell view depicts the physical implementation in silicon.

4.3.2 Regularity

Hierarchy alone will not make the design simpler if all the sub-modules are different. Regularity divides the hierarchy into a set of similar blocks, and it can exist at most of the design hierarchy levels. At the circuit level, uniform sized transistors simplify the operation, rather than having to manually optimize each device. At the logic module level, employing an identical gate makes design much easier to follow. At higher levels, using a number of identical processor structures will solve the complexity problem. In the Stallion chip design, efforts have been made to follow the regularity requirement.

4.3.3 Modularity

Modularity requires that the sub-modules have well-defined functions. If modules are well defined, the interaction with other modules will be well characterized. During the Colt/Stallion chip design process, the whole chip is subdivided into different blocks, with each block having its own well-defined functions.

4.3.4 Locality

The last strategy is the locality. Locality has two major meanings: time locality, and global locality. Time locality requires paying attention to clock generation and distribution. Critical path, if possible, should be kept within module boundaries. Global locality is used to minimize global wiring and requires floor planning. After wiring is done, modules are distributed rather than placing them first and then routing them together [16].

4.4 Characterization and Performance Estimation

Poor layout not only affects the chip area but also its performance in speed and power dissipation. Ultimately, it may affect the correct behavior of the chip. Understanding the relationship between geographical layout and system behavior will help in the design procedure. In next section, we will discuss those issues in terms of resistance estimation, capacitance estimation, switch characters, distributed RC effects, and power dissipation.

4.4.1 Resistance Estimation

The resistance of a uniform slab of conducting materials may be expressed as

$$R = \frac{\rho l}{tw} \tag{4.1}$$

where ρ, t, l and w are the resistivity, thickness, conductor length and conductor width, respectively.

The above expression may be rewritten as

$$R = R_s \frac{l}{w} \tag{4.2}$$

Where R_s is the sheet resistance having units of Ω/square. The resistance shown in Figure 4.1 is the same because the length-to-width ratios are the same although their sizes are different.

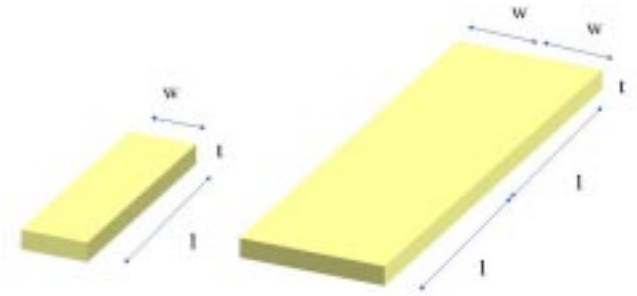

Figure 4.1: Determination of layer resistance.

For metal having a given thickness, t, the resistivity is known. But for poly and diffusion, the resistivities are significantly influenced by the concentration density of the impurities that have been introduced into the conducting regions during implantation, or the chemical changes introduced by materials such as silicon. In addition, the channel resistance increases

as temperature increases because the mobility decreases as the temperature increases. For the non rectangular shape layout, Figure 4.2 summarizes the resistance of a number of commonly encountered shapes [17].

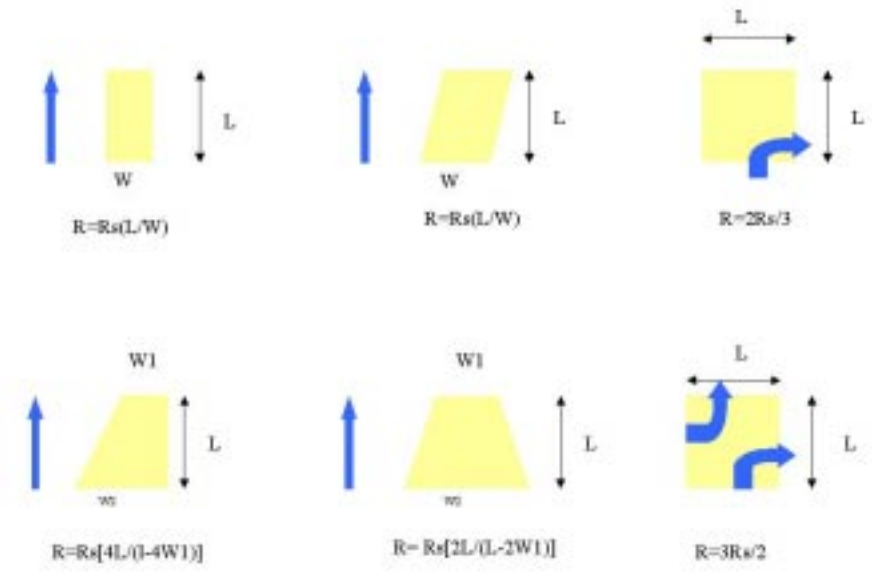

Figure 4.2: Geometry and resistance.

4.4.2 Capacitance Estimation

Capacitance plays an important role in affecting the performance of the IC chip, especially on the dynamic response. The total load capacitance on the output of a CMOS gate is the sum of gate capacitance, diffusion capacitance, and routing capacitance as shown in

Equation 4.3. Layout shape has a direct impact on the capacitance.

$$C_{load} = C_{gate} + C_{diffusion} + C_{routing} \tag{4.3}$$

- **Gate Capacitance:**

  Although more accurate modeling of the MOS transistor capacitance may be achieved by using a charge based model, for the purpose of delay calculation for digital circuits, gate capacity may be approximated by $C_g = C_{ox}A$, where A is the area of the gate and C_{ox} is the "thin-oxide" capacitance per unit area given by

$$C_{ox} = \frac{\varepsilon_0 \varepsilon_{sio_2}}{t_{ox}} \tag{4.4}$$

  with a thin-oxide $95A$ thickness for the HP05 process used in the Stallion chip, the value of C_{ox} is

$$C_{ox} = \frac{3.9 \times 8.854 \times 10^{-10}}{95 \times 10^{-8}} \approx 36.3 \times 10^{-4} \frac{pF}{\mu m^2} \tag{4.5}$$

  For example, the gate capacitance of an unit MOS transistor shown in Figure 4.3 is

$$\begin{aligned} C_g &= 4\lambda \times 5\lambda \times 36.3 \times 10^{-4} \\ &= 4 \times 0.3 \times 5 \times 0.3 \times 36.3 \times 10^{-4} \\ &= 6.5fF \end{aligned} \tag{4.6}$$

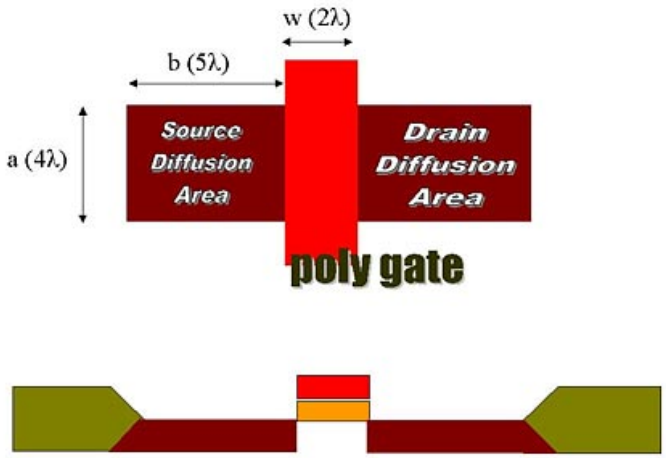

Figure 4.3: Gate and diffusion capacitance.

- **Diffusion Capacitance:**

  Diffusion capacitance or source/drain capacitance can be represented by

$$C_d = C_{ja}(ab) + C_{jp}(2a + 2b) \tag{4.7}$$

  where C_{ja}, C_{jp} represent junction capacitance per μm^2 and periphery capacitance per μm respectively, while a and b denote width and length of diffusion region μm respectively.

  From the above equation, we can tell that as the diffusion area is reduced (through scaling), the relative contribution of the peripheral capacitance becomes more important. Furthermore, since the thickness of the depletion layer depends on the voltage

across the junction, both C_{ja} and C_{jp} are functions of junction voltage, V_j. A general expression that describes the junction capacitance is

$$C_j = C_{j0}(1 - \frac{V_j}{V_{j0}})^{-m} \tag{4.8}$$

where V_j, C_{j0} and V_{j0} are junction voltage (negative for reverse biased), zero bias capacitance ($V_j = 0$) and built in junction potential respectively. m is a constant depending on the distribution of impurities near the junction and whether the junction is due to the bottom or the side of the diffusion.

Taking the unit MOS transistor as shown in Figure 4.3 for example again, the junction area is $4\lambda \times 5\lambda$, which is $1.8\mu m^2$ and the junction periphery is $4\lambda + 5\lambda$, which is $2.7\mu m$. At $V_j = 1.65$ volts (half rail) for HP05 process, $C_{ja0} = 5.69 \times 10^{-4}F$, $m_{ja} = 0.661$, $C_{jp0} = 2.00 \times 10^{-11}F$, $m_{jp} = 0.609$, and $V_j = 0.99$ volts for NMOS transistor. On the other side, $C_{ja0} = 9.19 \times 10^{-04}F$, $m_j = 0.321$, $C_{jp0} = 4.60 \times 10^{-10}F$, $m_{jp} = 0.100$, and $V_j = 0.42$ volts for PMOS transistor. Applying Equation 4.7 and Equation 4.8, it is obtained that

$$\begin{aligned} C_{nd} &= 1.8 \times 10^{-12} \times 5.69 \times 10^{-4} \times (1 + \frac{1.65}{0.98})^{-0.661}F \\ &\quad +2.7 \times 10^{-6} \times 2.00 \times 10^{-11} \times (1 + \frac{1.65}{0.98})^{-0.609}F \\ &= 1.5 \times 10^{-18}F + 2.96 \times 10^{-17}F \\ &= 3.11 \times 10^{-17}F \end{aligned}$$

$$= \quad 0.0311fF \tag{4.9}$$

$$\begin{aligned} C_{pd} &= 1.8 \times 10^{-12} \times 9.19 \times 10^{-4} \times (1 + \frac{1.65}{0.42})^{-0.321} F \\ &\quad +2.7 \times 10^{-6} \times 4.60 \times 10^{-10} \times (1 + \frac{1.65}{0.42})^{-0.100} F \\ &= 9.91 \times 10^{-16} F + 1.059 \times 10^{-15} F \\ &= 2.05 \times 10^{-15} F \\ &= 2.05fF \end{aligned} \tag{4.10}$$

- **Routing Capacitance:**

  An empirical formula based on parallel-plate theory which is computationally efficient and relatively accurate is given by

$$C = \epsilon \left[\frac{w}{h} + 1.66(\frac{w}{h})^{0.25} + 1.06(\frac{t}{h})^{0.5} \right] \tag{4.11}$$

  where w is the width of the conductor, h is the insulator thickness and t is the conductor thickness.

  Poly and metal lines will actually have a higher capacitance than that predicted by the above formula because of the fringing field's effects.

4.4.3 Distributed RC Effects and Wire-length Design Guide

For long wires, especially poly, propagation delays caused by resistance capacitance (RC) in the wiring layer can dominate. To optimize speed of a long poly, one possible strategy is to segment the line into several sections and insert buffers into these sections. As circuit speed increases, even metal connections can give rise to RC-delay effects, especially in heavily loaded clock lines. In this case, a more straightforward method to solve this skew problem is to widen the clock line and distribute the clock line from the top center of the chip.

When the delay caused by wire length is small enough compared with that caused by gate delay, wires can be treated as a single electrical mode and modeled as a simple capacitive load. To satisfy the above condition, it is required that

$$\tau_w << \tau_g \tag{4.12}$$

or

$$l << \sqrt{\frac{2\tau_g}{rc}} \tag{4.13}$$

Table 4.1 [16] lists the guidelines for ignoring RC wire delays in terms of λ. This table assumes gate delay is of the order of 100ps to 500ps and the signals are lightly loaded. Heavily loaded signals such as clocks should always be checked for RC skew problems. The Stallion layout implementation follows this guideline.

Table 4.1: Guidelines for ignoring RC wire delays.

Layer	Maximum length (λ)
Metal3	10000
Metal2	8000
Metal1	5000
Silicide	600
Polysilicon	200
Diffusion	60

4.4.4 Switching Characteristics

Layout design should not only consider the effect of RC delay but also the difference between rise and fall time caused by the different characteristics of n and p transistors.

Taking a CMOS inverter as an example from analytic delay models, we have fall time

$$t_f = 3 \sim 4\frac{C_l}{\beta_n V_{dd}} \tag{4.14}$$

and rise time

$$t_r = 3 \sim 4\frac{C_l}{\beta_p V_{dd}} \tag{4.15}$$

where

$$\beta_n = \frac{\mu_n \varepsilon}{t_{ox}} \times \frac{w_n}{l_n} \tag{4.16}$$

$$\beta_p = \frac{\mu_p \varepsilon}{t_{ox}} \times \frac{w_p}{l_p} \tag{4.17}$$

For HP05 process, $\frac{\mu_n \varepsilon}{t_{ox}} = 1.7146 \times 10^{-4}$ and $\frac{\mu_p \varepsilon}{t_{ox}} = 8.8312 \times 10^{-5}$. So

$$\begin{aligned} \beta_n &= 1.71 \times 10^{-4} \times \frac{4}{2} \\ &= 3.42 \times 10^{-4} \frac{A}{V^2} \end{aligned} \tag{4.18}$$

$$\begin{aligned} \beta_p &= 8.83 \times 10^{-5} \times \frac{4}{2} \\ &= 17.66 \times 10^{-5} \frac{A}{V^2} \end{aligned} \tag{4.19}$$

For a unit inverter in the Stallion chip, assuming the routing capacitance equals to $10fF$, according to Equation 4.3, the load capacitance is,

$$\begin{aligned} C_l &= 2 \times 6.5 + 0.0311 + 2.05 + 10 \\ &= 25.08fF \end{aligned} \tag{4.20}$$

Applying Equation 4.15 and Equation 4.14, the rise and fall time are

$$\begin{aligned} t_{rise} &= 4 \times \frac{C_l}{\beta_p \times V_{dd}} \\ &= 4 \times \frac{25.08 \times 10^{-15}}{17.66 \times 10^{-5} \times 3.3} \\ &= 172.1 \times 10^{-12} s \\ &= 172.1 ps \end{aligned} \tag{4.21}$$

$$
\begin{aligned}
t_{fall} &= 4 \times \frac{C_l}{\beta_n \times V_{dd}} \\
&= 4 \times \frac{25.08 \times 10^{-15}}{3.42 \times 10^{-4} \times 3.3} \\
&= 88.9 \times 10^{-12} s \\
&= 88.9 ps
\end{aligned} \tag{4.22}
$$

Because the carrier mobility for n transistor μ_n is two times that of p transistor μ_p, $\mu_n = 2\mu_p$, for equally sized n- and p- transistors, we have $\beta_n = 2\beta_p$. Thus from the above two equations, we have $t_f = 1/2t_r$. If we wish to have the same rise and fall time from an inverter, we have to widen the p-device m times that of n-device. In general, the fall time t_f is mt_f for m n-transistors in the series. Similarly, the rise time t_r is mt_r for m p-transistors in the series. In comparison, if the m transistors are in a parallel connection, either for n-transistors or for p-transistors, the rise and fall time will be t_r/m and t_f/m respectively.

For a more graphical understanding of this, a series transistor connection is illustrated in the Figure 4.4.

4.4.5 Power Dissipation

Power dissipation refers to power that has been consumed by a device. There are two different types of power dissipation, static and dynamic. Static power dissipation is due to the leakage current or other current being drawn continually from the power source. Dynamic dissipation is due to switching transient current and charging/discharging of load capacitance.

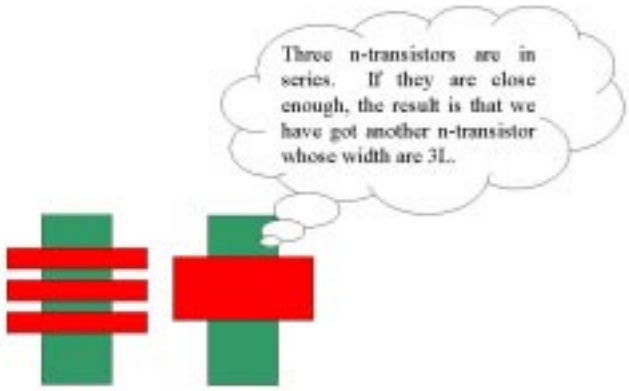

Figure 4.4: Graphical illustration of the effect of series transistors.

- **Static Power Dissipation:**

  For a complementary CMOS gate, the static dissipation is close to zero since there is no DC current path from V_{dd} to ground. The small amount is caused by sub-threshold conduction and bias leakage between diffusion regions and the substrate.

- **Dynamic Power Dissipation:**

  Dynamic dissipation is generated during transition either from high to low or from low to high. During these times, both p and n transistors are on, resulting in a short current. Dynamic dissipation is proportional to C_{load}, ${V_{dd}}^2$ and switching frequency f_p but is independent of the device parameters, and can be expressed by,

$$P_d \propto C_{load} {V_{dd}}^2 f_p \tag{4.23}$$

For the Stallion chip, the V_{dd} is $3.3V$ and the switching frequency f_p is $50MHz$. According to the Equation 4.23, the dynamic power dissipation for the Stallion chip is proportional to the load capacitance C_{load}.

- **Minimizing Power Dissipation:**

  Minimizing power dissipation may be achieved in a number of ways [18]. The elimination of DC dissipation is achieved by using a complementary gate. Minimizing devices will reduce leakage. Reducing power supply, load capacitance, and frequency will also help in reducing dynamical dissipation. In the Stallion chip design, minimum size transistors are used whenever possible to lower the dynamic power dissipation.

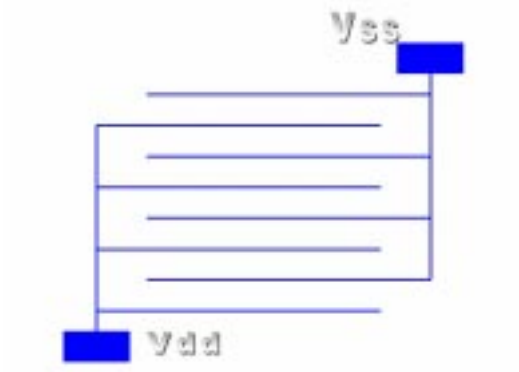

Figure 4.5: Layout patterns for V_{dd} and V_{ss}.

4.4.6 Power and Ground Routing and Contact Issues

Power distribution is becoming an increasingly critical issue in designing digital circuits. Dynamical power dissipation, which is proportional to the operation frequency, becomes

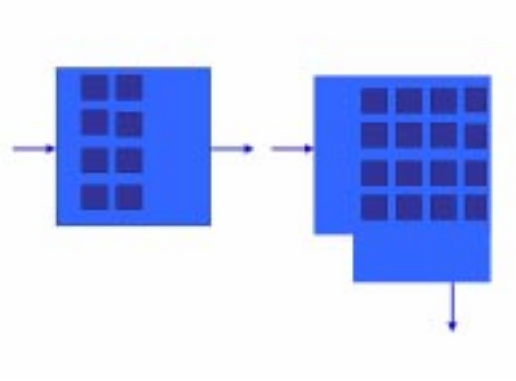

Figure 4.6: Contact structure for linear and orthogonal joints.

significant problem in high-performance chips. A simple layout of V_{dd} and V_{ss} lines is shown in Figure 4.5. The power lines should be as thick and wide as possible to minimize voltage drops and current density. In addition, the current density in a contact or in a periphery must be kept below 0.1mA/μm. If the current flow turns at a right angle or reverses, a square array of contents is generally required. If the flow is in the same direction, fewer contacts may be used. This is shown in Figure 4.6. In the layout implementation of the Stallion chip, these schemes are used.

Chapter 5

Layout Design of the Stallion Chip

Floor planning is an essential step in any VLSI layout design. Industries have databases of building blocks (e.g. ROM's, RAM's and latches) so that before the actual layout designing starts, it is possible to produce a floor-plan which is very close to that of the final product. However, this is not always true in academic environment, where such databases are generally not to be available. Luckily, Virginia Tech has Colt, which provides useful and needed database information.

The Stallion chip will be fabricated by MOSIS' scalable CMOS technology in their HP05 process. It is a triple metal, n well process with a $0.6\mu m$ drawn feature size. Different from the functional unit in Colt, which uses two-metal layers for routing, the functional unit in the Stallion chip uses one metal. The reason for using only one metal is that it allows buses to run on top of the FU. The Colt/Stallion chip is a mesh connected circuit. Each functional

unit is connected to its four neighbors. If the functional unit is made of two metals as in the Colt, it will take a huge amount of silicon area to route the buses as shown in Figure 5.1. However, if only one metal is used, there will be a great difference. As indicted in Figure 5.2, buses may run across the FU, saving silicon area and reducing the unnecessary routing delays.

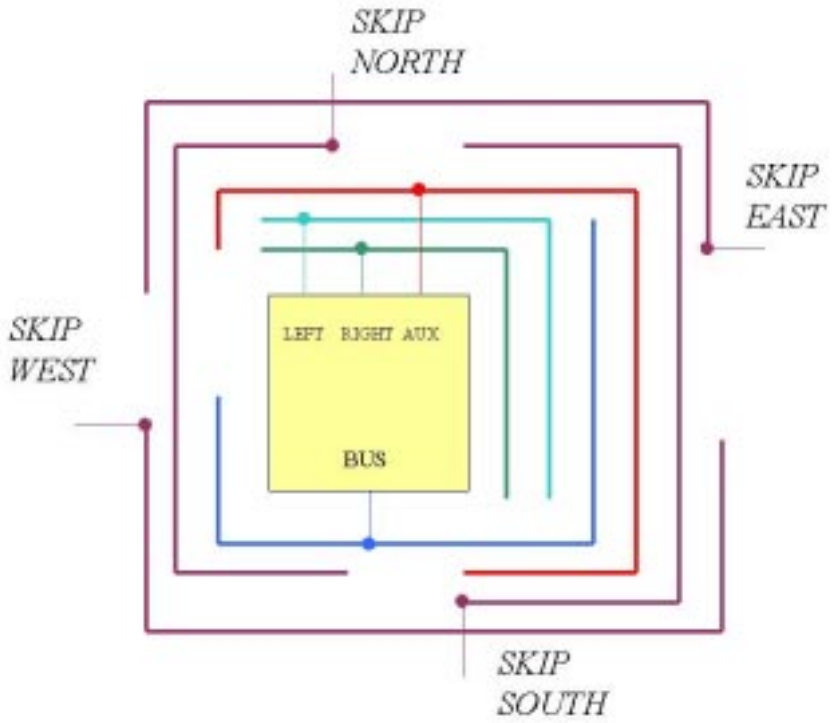

Figure 5.1: The Interconnected Functional Unit floor plan for the Colt chip.

The Functional Unit is the heart of the chip because it is where computations take place. The layout floor plan is shown in Figure 5.3. The layout implementation is shown in Figure 5.4 and Figure 5.5.

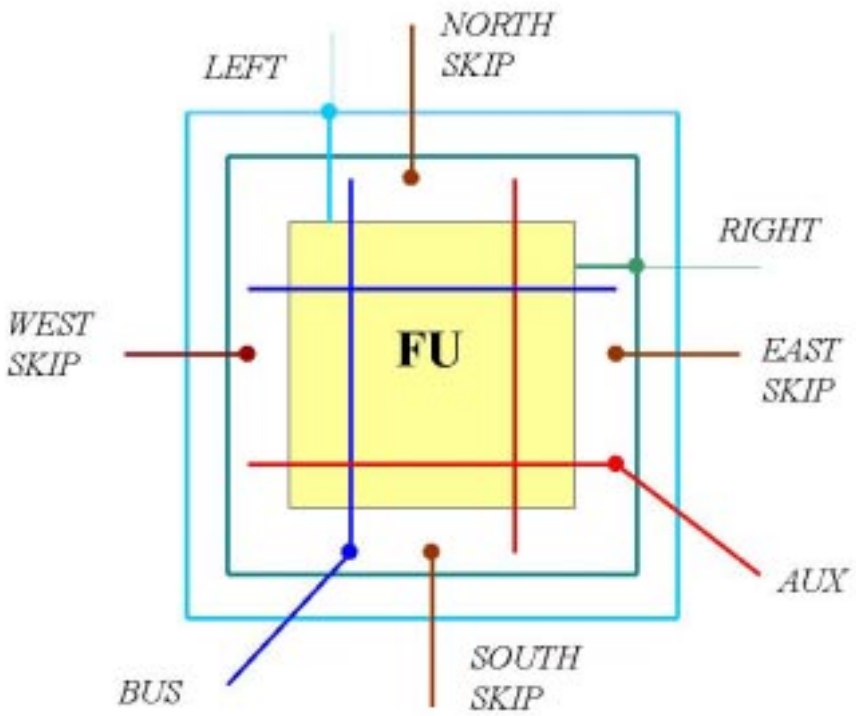

Figure 5.2: The Interconnected Functional Unit floor plan for the Stallion chip.

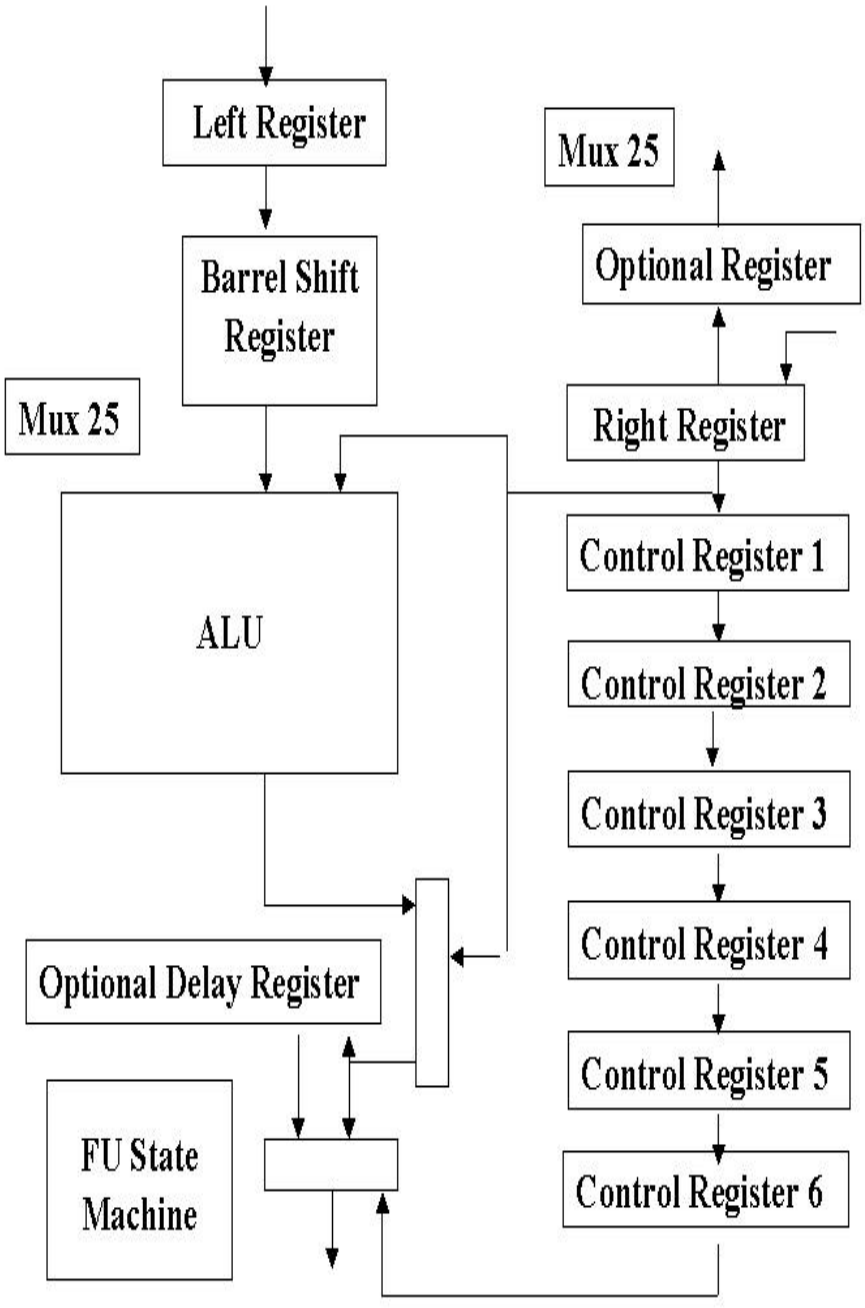

Figure 5.3: Floor plan of Functional Unit for the Stallion chip.

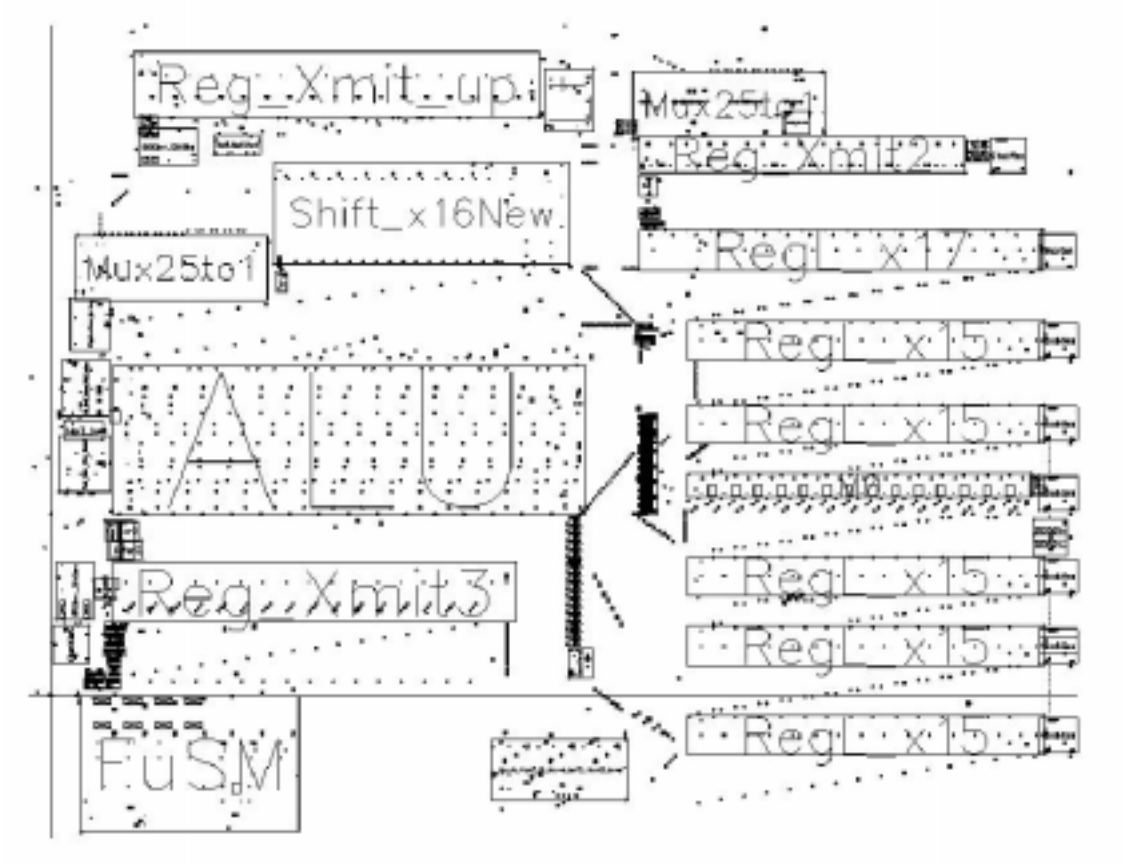

Figure 5.4: Functional Unit VLSI layout for the Stallion chip (1).

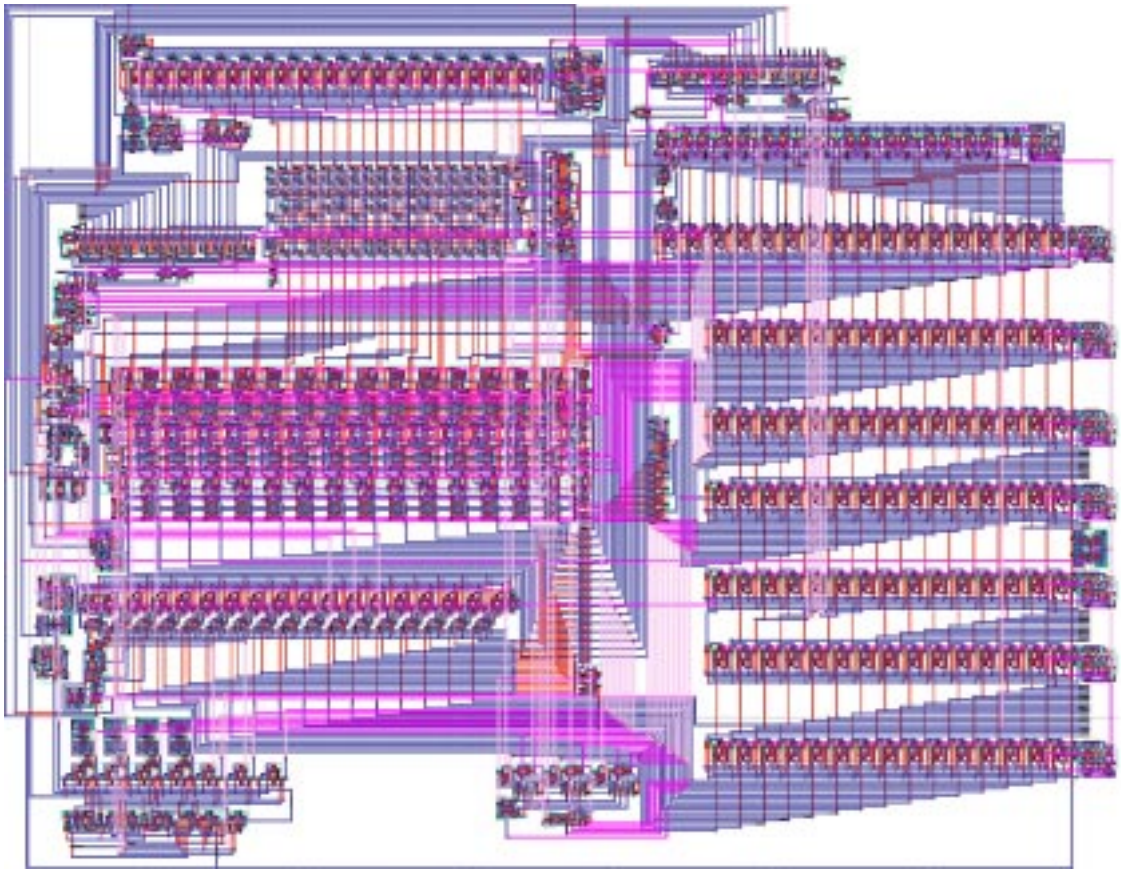

Figure 5.5: Functional Unit VLSI layout for the Stallion chip (2).

5.1 Arithmetic Logic Unit

Figure 5.6 shows the schematic cell view of the ALU which is based on a look-up table. This is a general single-bit slice of a standard Propagate (P), Generate (G), and Result (R) design. The ALU has 16-bit width. The bit vectors P, G, and R are each four bits wide and can be used to program the ALU to perform various functions, including addition, subtraction, complement, etc. For example, an addition can be specified using the pattern P = 0110, G = 1000, and R = 0110. Twelve bits are needed to specify the function that is being performed. These same twelve bits are used to control the function performed by all sixteen-bit slices across the width of the ALU. Figure 5.7 depicts the circuit used for odd bit numbers and Figure 5.8 shows the circuit used for even bit numbers. The layout of ALU is shown in Figure 5.9.

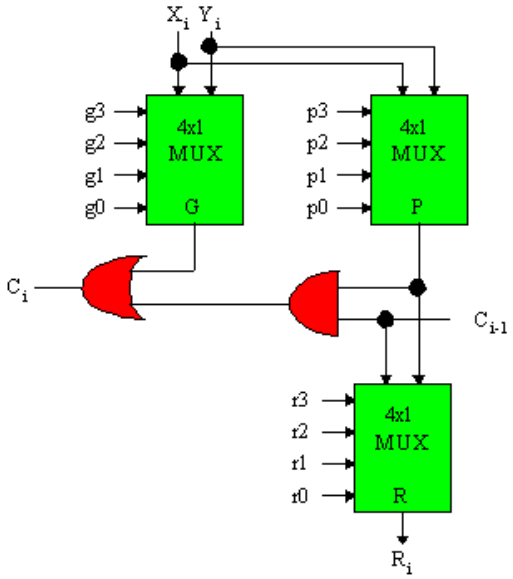

Figure 5.6: General ALU bit slice.

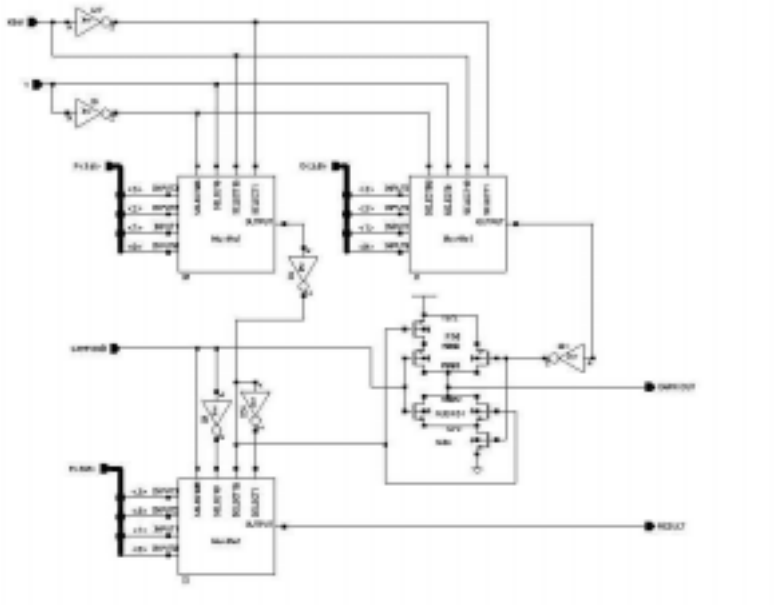

Figure 5.7: Odd ALU bit slice.

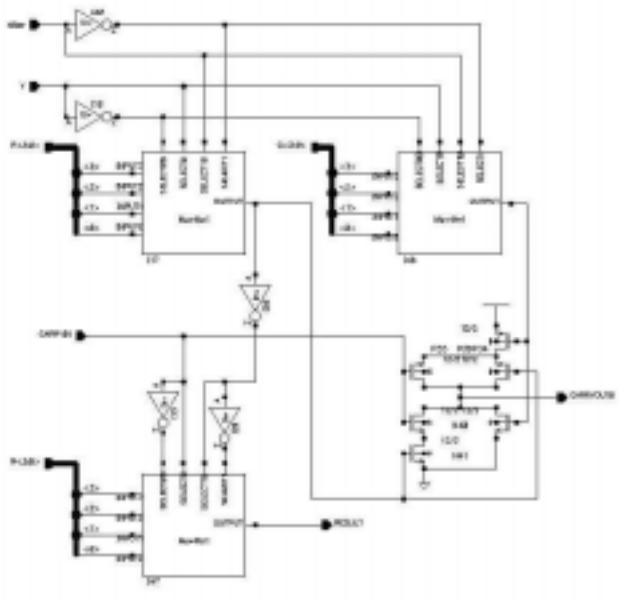

Figure 5.8: Even ALU bit slice.

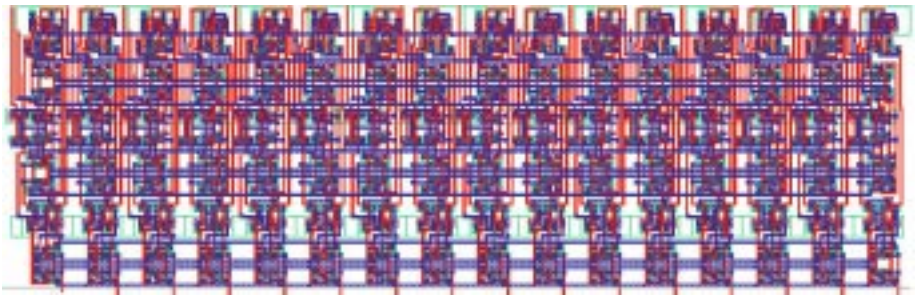

Figure 5.9: ALU layout.

5.2 Barrel Shifter Register

Different from the Barrel shifter register in Colt (Figure 5.10), the Barrel Shift in the Stallion chip has been changed as shown in Figure 5.11. Because of this change, the signal only goes through one NFET pass transistor instead of two, thus increasing the output voltage.

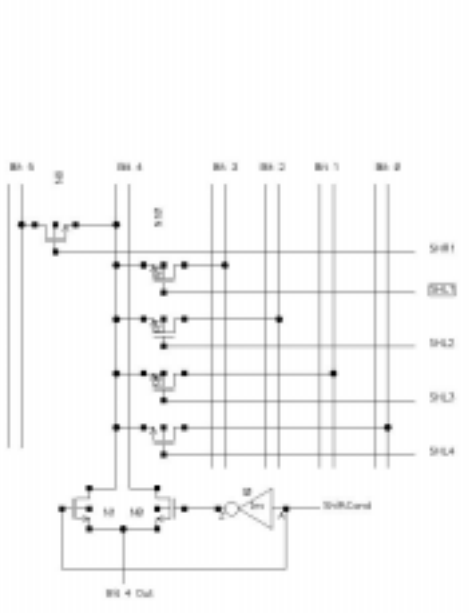

Figure 5.10: Barrel shifter register in the Colt chip.

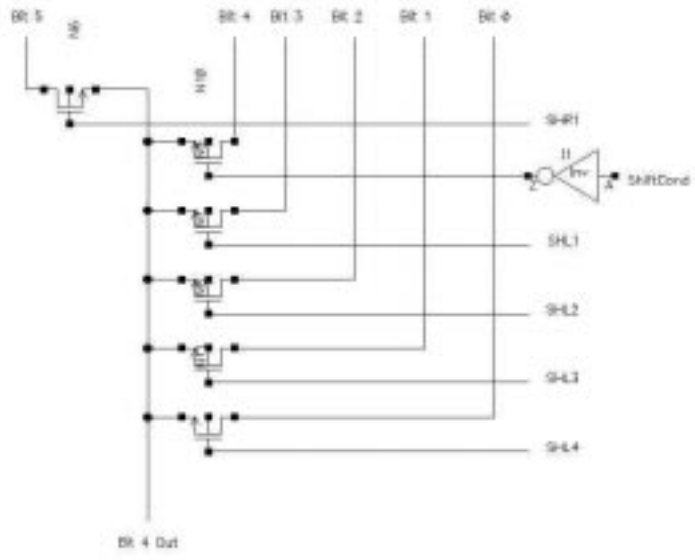

Figure 5.11: Improved barrel shifter in the Stallion chip.

The layout of Barrel Shifter is shown in Figure 5.12.

5.3 Registers/D Flip Flop

In the Colt/Stallion chip, all the registers are composed of D Flip/Flops (Figure 5.13 and Figure 5.14). A two-clock phase is used in D Flip/Flop to avoid clock skew. There are three types of registers. One is seventeen bits in width with a load enable signal. This type of register is used to store the right and left input operands and bus output. The second type of register is called Configuration Control Register. It is fifteen bits wide, with load enabling used to store configuration information as mentioned in the previous chapter. The third type of register is seventeen bits in width but without a load enable signal. This register is

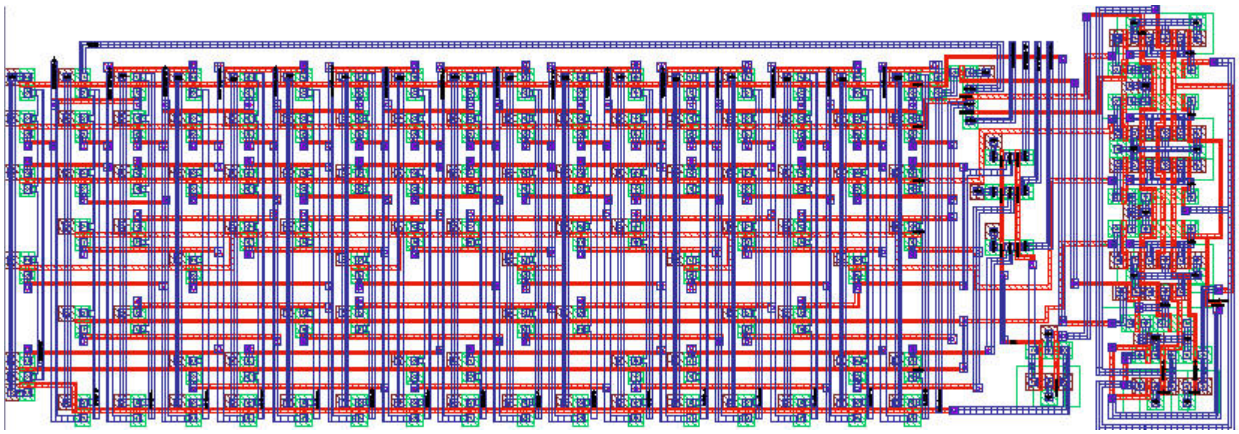

Figure 5.12: The Barrel shifter register's layout in the Stallion chip.

used to buffer the auxiliary output.

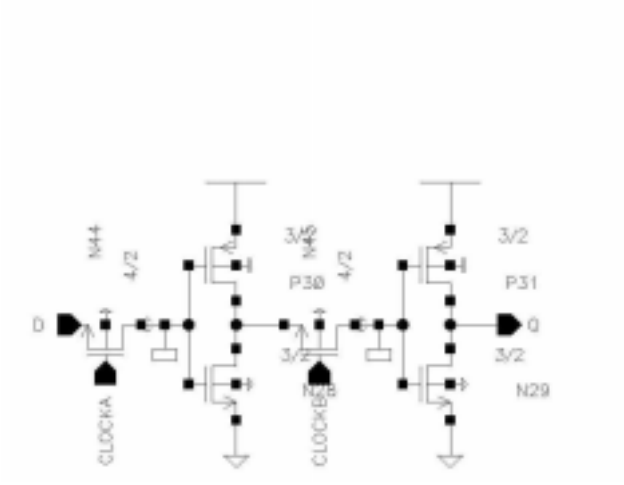

Figure 5.13: D Flip Flop - schematic.

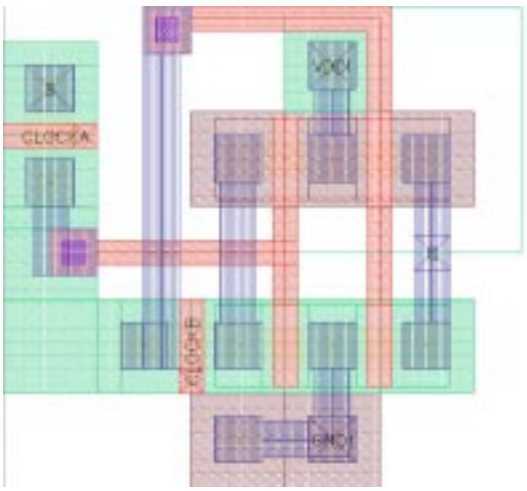

Figure 5.14: Flip Flop - layout.

5.4 Inverter

There are two different types of inverters used in the Colt/Stallion chip. The difference between them is their β size. In layout implementation, instead of increasing the size of the transistors, a large inverter has been constructed by using a donut connection. Using a donut inverter significantly decreases the output capacitance because of the smaller merged diffusion region. It avoids decreasing the circuit's operation speed, as would be the case in a linear design. Figure 5.15 shows the layout of a smaller inverter and Figure 5.16 is the layout of a donut inverter. The β of the latter is four times larger than that of the smaller inverter.

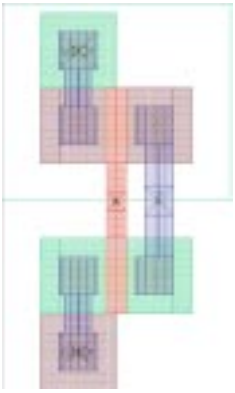

Figure 5.15: Linear inverter.

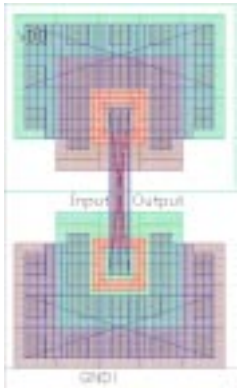

Figure 5.16: Donut inverter.

Chapter 6

Conclusion and Future Work

This thesis has presented a methodology to develop a VLSI layout for the Wormhole CCM Stallion chip. Every effort has been made to explain many facts and advantages of the Stallion chip, a Wormhole Configurable Computing Machine. Design techniques such as design strategies, circuit characterization and performance estimation, and ways to solve problems when using CAD layout design tool cadence are illustrated.

Since the Stallion chip is still under construction, much effort is needed in the future to work on its layout aspect. The layout of the Functional Unit needs optimization when it is actually connected to the Mesh. Consideration should be given as to how to embed four multipliers into the Mesh. As the simulation for the schematic design goes further, there will be unavoidable minor or even major changes in the future chip, and the existing layout will have to be modified according to those changes as well. As the project nears completion,

more attention should be paid to the floor plan, such as how to make use of the advantage of one metal routing in FU parts.

Bibliography

[1] R. Maziasz and J. Hayes. *Layout Minimization of CMOS Cells.* Kluwer Academic Publishers, 1992.

[2] J. Villasenor and W. Mangione-Smith. Configurable computing. *Scientific American*, June 1997.

[3] R. Bittner and P. Athanas. Wormhole run-time reconfiguration. In *FPGA '97: ACM/SIGDA International Symposium on Field Programmable Gate Arrays. Special Interest Group on Design Automation (SIGDA), ACM*, 1997.

[4] W. Stallings. *Computer Organization and Architecture: Designing for Performance, fourth edition.* Addison-Wesley, 1996.

[5] V. Nelson and J. Irwin. *Digital Logic Circuit Analysis and Design.* Prentice Hall, 1995.

[6] Buell D. Arnold, J. and E. Davis. Splash 2. In *Proceedings: 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 316–322, 1992.

[7] California Giga Operations Corp., Berkeley. *Spectrum Reconfigurable Computing Platform User's Manual*, 1995.

[8] Inc Annapolis Micro Systems. A family of reconfigurable computing engines. http://www.annapmicro.com/wfhtm.html.

[9] R. Bittner. *Wormhole Run-Time Reconfiguration: Conceptualization and VLSI Design of a High Performance Computing System.* PhD thesis, Virginia Polytechnic Institute and State University, 1997.

[10] M. Musgrove. VLSI implementation of a run-time reconfigurable custom computing integrated circuit. Master's thesis, Virginia Polytechnic Institute and State University, 1996.

[11] T. Yang. A stream-based in-line allocable multiplier for configurable computing. Master's thesis, Virginia Polytechnic Institute and State University, 1997.

[12] M. Mano. *Computer Systems Architecture, 3rd edition.* Prentice Hall, 1993.

[13] M. Cherbaka. Verification and configuration of a run-time reconfigurable custom computing integrated circuit for dsp applications. Master's thesis, Virginia Polytechnic Institute and State University, 1996.

[14] M. Annaratone. *Digital CMOS Circuit Design.* Kluwer Academic Publishers, 1986.

[15] J. Armstrong and F. Gray. *Structured Logic Design with VHDL.* PTR Prentice Hall, 1993.

[16] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design.* Addison-Wesley, 1994.

[17] P. Athanas. *EE5545 class notes*, 1996.

[18] A. Bellaouar and M. Elmasry. *Low-power Digital VLSI Design: Circuits and Systems.* Kluwer Academic Publishers, 1995.

[19] Cadence Design System Inc. *Cell Design Tutorial Version 4.3*, March 1994.

Appendix A

Using Cadence Tools

During the design process, the author gained some useful experience while using the Cadence tools to debug the LVS unmatched problems and to achieve file after the icfb crashed. The following section describes the solutions to the problems stated above. Here the LVS stands for Layout versus Schematic. It is an important process in the layout design after the layout has passed design rule check (DRC) and has been extracted. These solutions are explained by debugging errors in the layout of the a simple circuit which is named ClockGen_bad. For further information about how to create layout cell view, process DRC, and perform extraction, see Appendix A of [10]. In addition, Cadence's Manual [19] and SourceLink maintained by the Cadence Inc on the Internet provide very detailed tutorials and instructions on how to use the Cadence tools.

A.1 How to debug merged and rewired nets and instances in LVS

Rewired and merged errors are the most common errors in the layout design. When the si.out or si.log reports rewired devices or merged nets there is always an error. For example, running LVS on ClockGen_bad.ext cell view generates the following output:

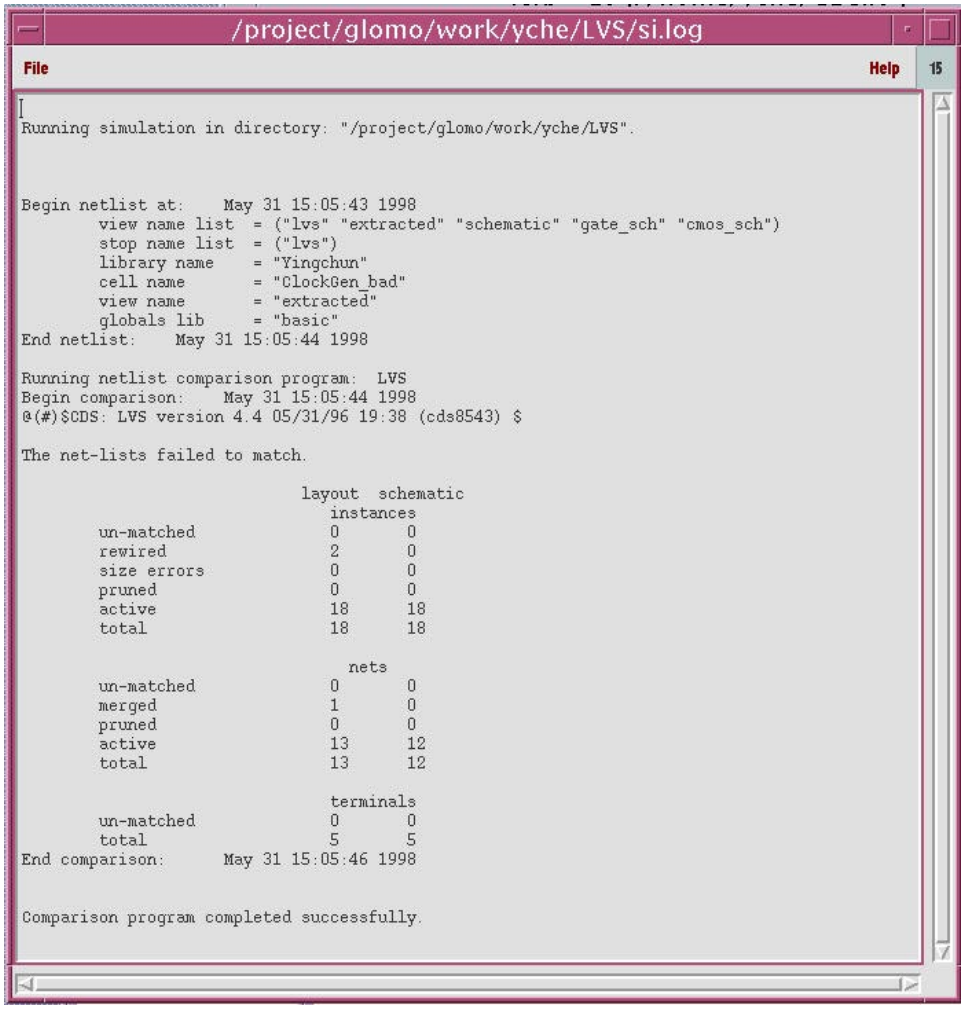

Figure A.1: Si.log.

Although there are zero unmatched instances and nets in the layout and schematic cell views, there are two rewired instances and one merged net in the layout. Rewired instances and merged nets are always errors. The layout does not match the schematic until there are no rewired instances or merged nets. To debug the errors, look at the merged nets first.

A.1.1 Merged Nets

Merged nets in the extracted view usually indicate an open connection. To find the merged nets:

1. Bring up the Verify->LVS->Error Display form from the window under the extracted view.

2. Set only "Merged nets" to be displayed.

3. Select "Display Errors." This highlights all merged nets in the extracted view.

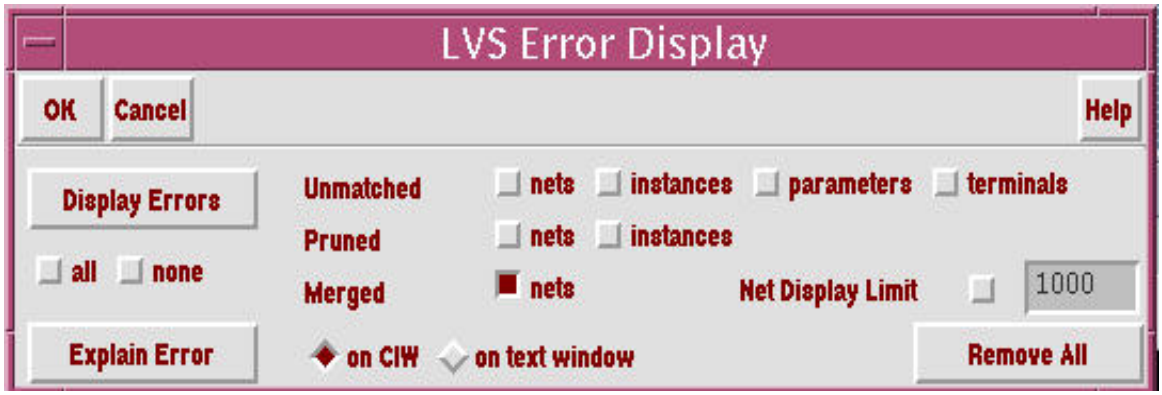

Figure A.2: LVS error display.

4. Select "Explain Error." This will prompt "Point at the object to explain: "

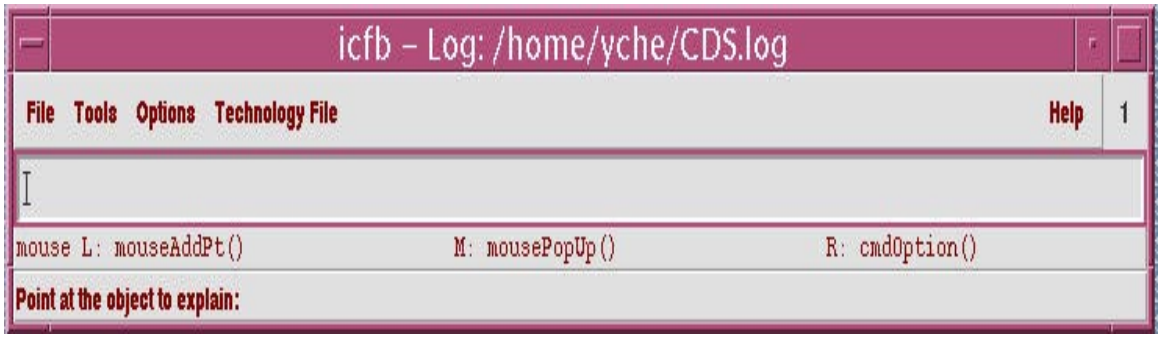

Figure A.3: CIW window.

5. Click on the highlighted net in the extracted view window. This will display a message as shown in A.4.

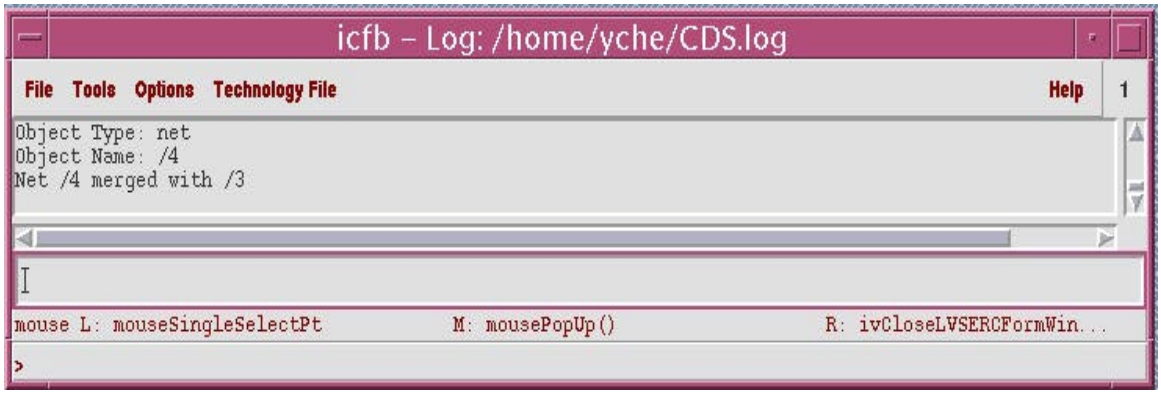

Figure A.4: Explain result.

The message above means that the net named "/4" has been probed. At the same time, the iLVS thinks that this net should be connected to the net named "/3" and it has matched devices based on merging those two nets. To find net /3

(a) Bring up the Verify->Probe form.

(b) For "Probing Method" turn on "single probe."

(c) For "Probing Scope" turn on "unmatched."

(d) For "Probe Type" turn on "net only."

(e) Select "Add Device or Net" and type "3" in the CIW, including the quotes.

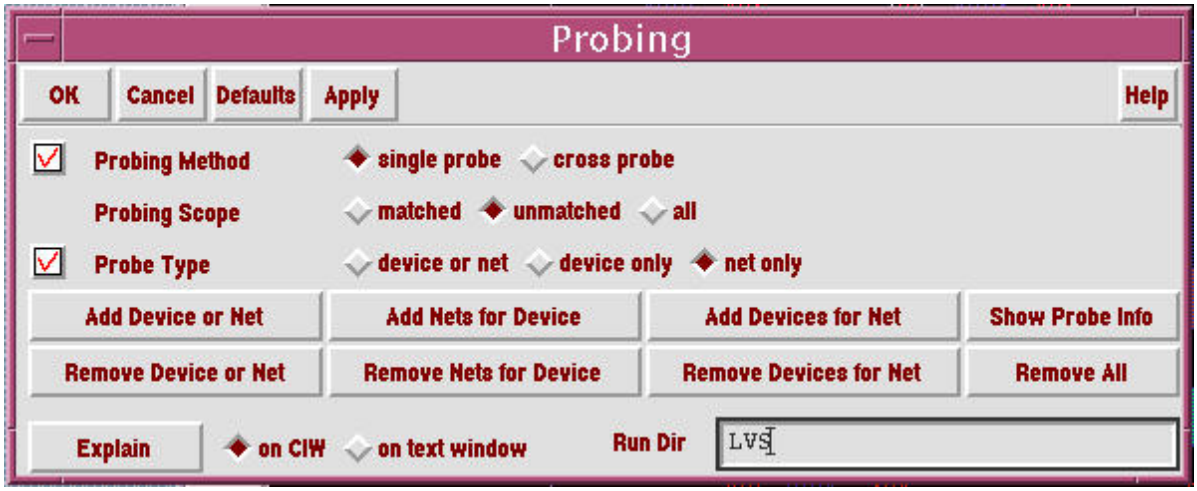

Figure A.5: Probing menu.

6. Find out the highlighted net "/3" in the extracted cell view and then open the symbolic cell view, and connect it to the net "/4."

   If the layout is correct and these two nets should not be connected, check the schematic for correctness.

A.1.2 Rewire

If one of the terminals is connected to a net other than the net to which it is connected, then the extracted view will match a device in the schematic, and the iLVS will "rewire" the bad terminal. Typically, rewired terminals are caused by a cross-wiring of nets in the layout.

Rewired devices are reported in si.out and si.log. To display the rewired terminals

1. Bring up the Verify->LVS->Error Display form from the window under the extracted view.
2. Set only "unmatched terminals" to be displayed.
3. Select "Display Errors." This highlights all unmatched terminals in the extracted view.

When highlighting unmatched terminals, it is difficult to see what has been highlighted because the terminals are very small. To find information about which terminal has just been displayed

1. Bring up the Verify->Probe form from the window under the extracted view.
2. Select "Show Probe Info." This will prompt: "Click at the desired source window location."
3. Click on the extracted view window. This will display a window that has a list of all the probes as shown in A.6.

Figure A.6: Window showing probe results.

The unmatched terminals are listed in the file "termbad.out" under /LVS/layout/ directory. The listing of A.6 shows that nets /9 and /10 have been cross-connected.

The error can be found by cross-probing the nets and the devices. These are the steps to follow:

1. Bring up the Verify->Probe form from the window under the extracted view.
2. For "Probing Scope" turn on "unmatched."
3. For "Probing Method" turn on "cross probe."
4. For "Probe Type" turn on "net only."
5. Select "Add Device or Net" and type "9" in the CIW, including the quotes.

Net /9 will be highlighted in both the extracted view and the schematic view. This lets us see the net which the iLVS thinks has been cross-connected.

To find the devices that have rewired terminals, the device can be cross-probed by name:

1. Change "Probe Type" to "device only."

2. Select "Add Device or Net" and type "+17" in the CIW, including the quotes.

3. Select "Add Device or Net" and type "+15" in the CIW, including the quotes.

After the nets having been probed in the extracted view and in the schematic, trace through the layout to see where the iLVS thinks the device should be connected. By doing that, the crossed connection can be found. Remember that the iLVS always treats the schematic as the master view and rewires the extracted devices to make them match the schematic. Sometimes the schematic has an error that causes the mismatch, so if the layout seems to be correct, check the schematic for errors.

A.1.3 Turning Off Re-Wiring

The merge and rewire capability can be disabled in the Verify->LVS form by turning off "Apply re-wiring" under "LVS Options."

A.2 How to find net correspondence between extracted and schematic views of the same design without using graphical cross-probing

Select "Create textual cross reference file" under LVS options in LVS form and then run LVS. By selecting this option an "xref.out" file will be generated in the LVS run directory. In this file there are two columns of node numbers and instance numbers. The first column is the node numbers and instance numbers in the extracted view, and the second column is the node numbers and instance numbers in the schematic view. By using this cross-reference table along with the "netlist" files in the layout and schematic directories under the LVS run directory, the net names in extracted view are mapped to those in the schematic view.

In our previous example, we get the following xref.out file:

Column 1	Column 2
N0	N7
N1	N3
N2	N4
N3	N2
N5	N5
N6	N14
N7	N11
N8	N18
N9	N9
N10	N0
N11	N1
N12	N10
I0	I0
I1	I1
I2	I14
I3	I2
I4	I15
I5	I3
I6	I11
I7	I10
I8	I17
I9	I7
I10	I5
I11	I12
I12	I13
I13	I4
I14	I6
I15	I8
I16	I9
I17	I16

where column 1 represents the node numbers (N#) and instance numbers (I#) from schematic view and column 2 denotes the node numbers (N#) and instance numbers (I#) from extracted view.

The following schematic netlists can be opened by bringing up the LVS->Show Run Information form and selecting "Schematic Netlist:"

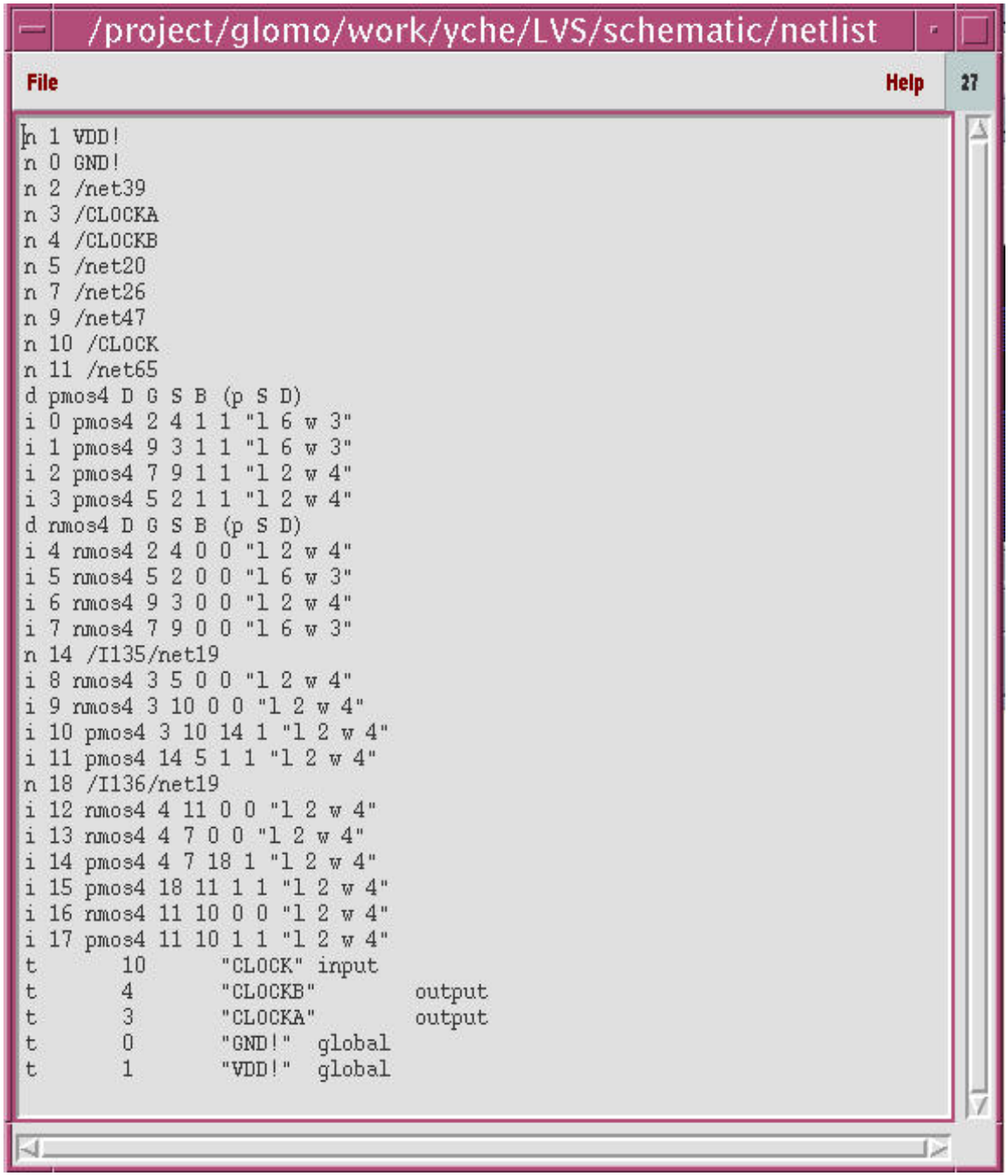

Figure A.7: Schematic netlists.

Similarly, open the layout netlist file by bringing up the LVS->Show Run Information form and selecting "Layout Netlist." At this point, the layout netlist shows:

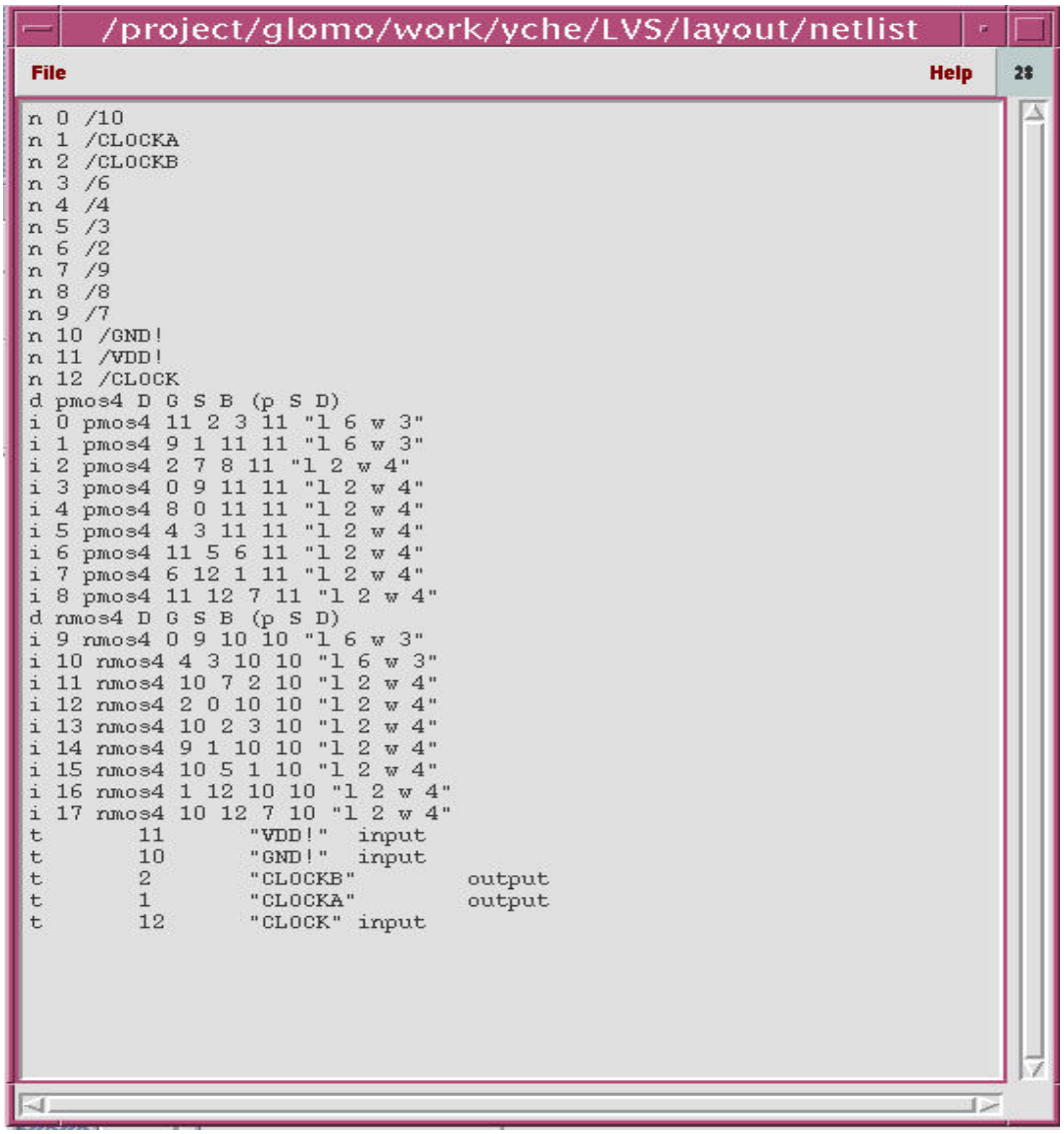

Figure A.8: Layout netlists.

So from the above information, the schematic netlist matches the layout netlist as indicated by the following table:

/net26	/10
/CLOCKA	/CLOCKA
/CLOCKB	/CLOCKB
/net39	/6
etc.	etc.

A.3 What to do if LVS is not generating matching statistics but says "End Comparison"

Under this situation, the LVS probably core dumped. This can be due to the extremely long simLibPath in the si.env file in the LVS run directory. In that case it will create an overflow condition for the buffer that holds this path string and cause the LVS to crash. The "End Comparison" message is actually printed out by the si. As a result, when the LVS crashes, the si takes over without warning that the LVS has crashed, and prints the "End Comparison" message.

The solution is to reduce the path length in the si.env file in the LVS run directory after the netlists are generated by typing:

"LVS <layout><schematic>" in UNIX to run the comparison program, where <layout> is the path to the layout netlist and <schematic> is the path to the schematic netlist.

A.4 What to do when icfb crashes because of segmentation errors

When modifying or debugging the design, no matter whether in schematic, layout, or extracted cell view, it is possible that the icfb may crash. Once the icfb crashes, go to the home directory as indicted by Unix prompt and open the panic.log file to find out the temporary file name and its path. Go to the related directory (if it crashes under the schematic cell view, it is under /schematic; if it crashes under layout cell view, it is under /layout) and copy the file sch.cd- (or layout.cd-) onto sch.cdb (or layout.cdb). Then reopen icfb again and continue working.

Vita

He, Yingchun was born in January 1965 in Shanghai, China. She received her Electronic Engineering Bachelor Degree in July 1987, from the Department of Electronic Engineering Beijing Tsinghua University. From July 1987 to May 1995, she worked in Shanghai Institute of Laser Technology and Institute of Fiber Optics, Shanghai University. In August 1995, she enrolled in the graduate program at Virginia Tech to pursue her Masters Degree in Electrical Engineering.