Connection-Switch Box Design and Optimal MST-Based Graph Algorithm on FPGA Segmentation Design

Zhou Lin

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science and Engineering

©The Chinese University of Hong Kong
June, 2004

The Chinese University of Hong Kong holds the copyright of this thesis. Any person(s) intending to use a part or the whole of the materials in this thesis in a proposed publication must seek copyright release from the Dean of the Graduate School.

香港中文大學
圖書館系統
31 MAR 2005
THE CHINESE UNIVERSITY OF HONG KONG
UNIVERSITY
LIBRARY SYSTEM

Abstract

Field-Programmable Gate Arrays (FPGAs) are a kind of Very Large Scale Integration (VLSI) circuits. They have been widely used in digital systems since their introduction in 1985. The property of instance manufacturing and low-cost prototypes allows them to be applied in various technology fields such as telecommunications, high-speed graphics and digital signal processing. In FPGAs programmable switches are used to connect all the circuit elements. Switches have high resistance and capacitance and require much chip area. Too many switches incur the penalty of circuit speed and chip area. Therefore, how to balance the routing flexibility and the performance constraints becomes a popular problem in current FPGA research.

Our first attempt is to propose the FPGA architecture applying Connection-Switch Boxes (CS-Boxes). That idea is similar to the Xilinx Virtex FPGA architecture. Our design is based on the symmetrical array FPGA architecture and combines the Connection Box and Switch Box together. The connection algorithm is designed to build the switches inside the CS-Box. To verify the efficiency of our design, we make the theoretical analysis and comparison between the new FPGA architecture and the traditional symmetrical array FPGA architecture. We take the example of $P = 5$ and $\alpha = 1.0$ (P is the number of pins on each logic block and α is the connection ratio of logic pins) and find $N_{cs} \leqslant N_x$ when $W \geqslant 3$ (N_{cs} and N_x are the switch number required by CS-Box FPGAs and symmetrical array FPGAs; W is the channel width). After that we conduct extensive experiments on MCNC benchmark circuits

and make comparison between the two architectures on channel widths, circuit delays and switch numbers. The results show that by applying the CS-Box structure the number of the switches connecting two wire segments can be reduced by up to 11.81% with small increase penalty of the channel width and the circuit delay by 0.38% and 2.34% on average respectively.

The channel segmentation design is another topic in FPGA architecture design. Chang *et al.* developed the Graph Matching-Based Algorithm to construct good segmentation designs. It tries to maximize the FPGA routability under performance constraints such as circuit speed and chip area. During the execution of the algorithm the outcome is affected by the pairing scheme, which means that the solution may be not optimal. We enhance it and present the MST-Based Graph Algorithm. It works optimally for both row-based FPGAs and symmetrical array FPGAs. The results are independent of the net merging order. We conduct experiments on ten sets of routing instances. And the results show a 4.31% reduction of net length from the Graph Matching-Based Algorithm to the MST-Based Graph Algorithm.

摘要

現場可程式化門陣列 （FPGA）是一種超大規模集成電路。自從 1985 年首次提出以來，它已經在數字系統領域得到了廣泛的應用。快速加工與低成本試製的特性使得它在遠距離通信、高速圖形學以及數字信號處理等諸多科技領域被採用。在 FPGA 中，可程式化轉換器用來連接所有的電路元件。這些轉換器有著較高的電阻與電容值，並且佔據大量芯片空間。使用過多的轉換器會對芯片的速度和所需面積產生不良影響。因此，如何在佈綫靈活性與工作約束之間達到平衡成爲了當今FPGA研究領域普遍討論的問題。

首先，我們提出了一種採用連接-轉換盒（CS-Box）的 FPGA 結構。這一想法與 Xilinx 公司的 Virtex FPGA 結構相似。我們的設計以對稱陣列 FPGA 為基本結構，將連接盒與轉換盒組合在一起。在連接-轉換盒內部，轉化器根據我們提出的連接算法進行設置。爲評估這項設計的有效性，我們在新型的 FPGA 與傳統的對稱陣列 FPGA 之間進行了理論上的分析與比較。以 $P=5$ 及 $\alpha=1.0$ 為例，我們發現當 $W \geq 3$ 時，存在 $N_{cs} \leq N_x$。隨之，我們在 MCNC 基準問題測試電路上進行了試驗，從通道寬度、電路延遲以及轉換器數目等方面對兩種結構進行比較。結果顯示，採用 CS-Box 結構後，用來連接兩段導綫的轉換器數目最多可減少 11.81%，同時伴隨著通道寬度與電路延遲的微弱增加，分別為平均 0.38%和 2.34%。

通道分段設計是 FPGA 結構設計的另一主題。Chang 等研究者設計了基於匹配理論的圖算法，用來構建較好的分段設計。這種算法試圖在電路速度與芯片面積等工作約束下，最大幅度增加 FPGA 的可佈綫性。算法的結果受到配對機制的影響，使得結果達不到最優化。我們對算法進行加強，提出基於最小生成樹的圖算法。在基於行的 FPGA 與對稱陣列的 FPGA 中，它都可以達最優化。結果不依賴網路結合順序。我們在十組佈綫實例上進行試驗。結果顯示，從基於匹配圖算法到最小生成樹圖算法，網路長度減少可達 4.31%。

Acknowledgments

First and foremost, I would like to thank my supervisor David Yu Liang Wu for his technical instruction, mentality advice and financial support. He led me into the FPGA architecture design field, an interesting research area and gave me invaluable suggestions on all my work. He does not only care my process but also gave me detailed advice on each step. So in the past two years I learned from Professor Wu how to dig deeply into my research interest as well as what attitude a real researcher should have. Besides research work Professor Wu taught me how to be a strong character in real life. He helped me out of the saddest experience in my life. After that I have the confidence to face all difficulties in the future. During my study period in Hong Kong I was given sufficient learning resources, for which I should also thank my supervisor very much.

Also I would like to thank another two professors of our department, Yiu Sang Moon and Fung Yu Young. Professor Moon is the marker of my term papers. In each semester he reviewed my term paper and attended my presentation. After my presentation he always gave me many valuable suggestions both on technique and on communication skills. My research area is near to Professor Young's. I took a course of hers and attended her study group. She helped me a lot in my academic work.

Here I should give my thanks to Cheung Chak Chung as well, who was an MPhil student of Professor Wu. With his kindly help I overcame much difficulty in research and life at the beginning of my study in Hong Kong. And

our discussion on research provided me clearer mind and helped me find a way to the solution. My thanks are given to the other students in our research group. Our discussion and cooperation is a treasure for my future work.

I would like to thank the friends in my office. They are Cheuk Man Lee, Tian Bai Ma, Tsz Yeung Wong, Ying Kin Hui and Yun Kai Liu. In daily work they helped me solve many technical problems. With their support I got to know many technique tools, which aided me to complete work more efficiently and effectively.

At last I want to thank my family. Even though I am far away from home, they gave me strong support in my work. My parents taught me how to make good balance between work and life. Their encouragement is the strongest spirit support for me.

Vita

Born in HeiLongJiang, China on August 9, 1979.

Education

Bachelor of Engineering, Harbin Institue of Technology, July 2002

M.Phil., The Chinese University of Hong Kong, August 2004 (expected)

Publications

Catherine L. Zhou, Ray C.C. Cheung and Yu-Liang Wu, " *What if Merging Connection and Switch Boxes - an Experimental Revisit on FPGA Architectures*" in Proc. Communications, Circuits and Systems, 2004 IEEE International Conference on. **Best Paper Award.**

Catherine L. Zhou and Yu-Liang Wu, " *Optimal MST-Based Graph Algorithm on FPGA Segmentation Design*" in Proc. Communications, Circuits and Systems, 2004 IEEE International Conference on.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

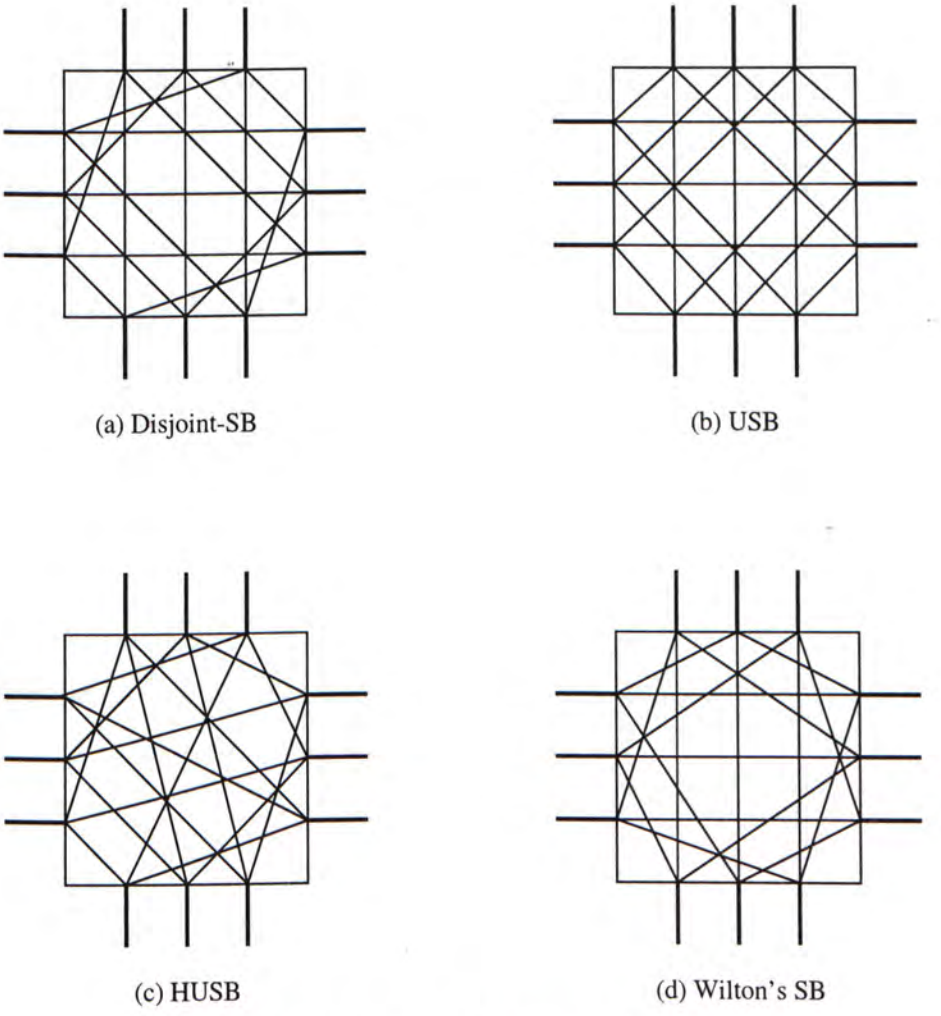
Field-Programmable Gate Array (FPGA) is a kind of Very Large Scale Integration (VLSI) circuit. It has been used for over a decade because of its wide applications in digital systems. With its reconfigurable nature users can program it by themselves in minutes. And the property of risk-free large-scale integration allows it to be applied in various technology fields such as telecommunications, high-speed graphics and digital signal processing [1]. Also in production FPGA provides instance manufacturing and very low-cost prototypes [2], especially in small and medium volumes.

Although FPGA has many features making it popular in current industry, its architecture nature incurs speed and area penalty during routing. In an FPGA programmable switches are used to connect the routing resources including logic pins, pad pins and wire segments in channels. Switches have high resistance and capacitance and they require much chip area. So too many switches lower the circuit speed and augment the chip area. On the other hand increasing the switch number aids to provide high routing flexibility. Therefore, how to balance the routing flexibility and the performance constraints has become a problem in current FPGA research.

There are many ways to improve the FPGA routability, such as designing

better physical structures of the whole chip, optimizing logic block architectures and enhancing routing algorithms. Today a lot of research works pay much attention to the FPGA routing architecture design. According to the architecture types, currently commercially available FPGAs can be classified into four classes: symmetrical array, row-based, sea-of-gates, hierarchical PLD. Figure 2.2 [2] shows their conceptual diagrams. Among them symmetrical array FPGAs are most widely applied. Based on its physical architecture researchers proposed new ideas in a variety of aspects. Some deal with the switch box's inside structure. At present there exist four types of switch boxes: Disjoint, Universal [3] [4] [5], HUSB [6] [7] and Wilton [1]. Figure 1.1 shows the four (4,3)-switch boxes. The literature [8] compares their routabilities in term of channel widths. Among them HUSB has been proven to be optimal in theory. And the experimental results also show that it outperforms the other kinds of switch boxes. Another way to improve the FPGA routability is to find a good channel segmentation design. Different segmentation designs are investigated in the literature [9] [10] [11] [12] [13] [14]. And a variety of methods were used in the past work, e.g., experimental studies, stochastic models, analytical analysis and graph-theoretic formulation. Lin *et al.* proposed the Graph Matching-Based Algorithm for FPGA segmentation design and routing. They use graph matching theory to formulate the Net Merging Problem and then extend it for general channel segmentation designs. The experimental results show that this algorithm is effective and efficient. Besides investigating the switch box structure and the channal segmentation, the Xilinx Company digs into a new research area, in which the Connection-Switch Box concept is proposed.

All the above research works have addressed on how to improve FPGA routability and how to reduce hardware resources. Those are the two main goals in this research area today. Therefore, we put most of our effort in this field.



(a) Disjoint-SB (b) USB

(c) HUSB (d) Wilton's SB

Figure 1.1: Most Popular (4,3)-Switch Boxes at Present

1.2 Aims and Contribution

Our research work concentrates on FPGA architecture design. And our aim is to improve the FPGA routability from various aspects, e.g., decreasing the channel width, reducing the switch number and deducting the routing delay.

Firstly we address on the Connection-Switch Box (CS-Box) design and construct our own CS-Box structure. The aim of this work is to reduce the hardware resources and simultaneously keep the routability non-degraded. The

conceptual diagram of a CS-Box FPGA is illustrated in Figure 3.3. The increased flexibility inside the CS-Box improves the overall chip routability and reduces the switch number of the entire chip. We decide the connection algorithm for the switch construction inside CS-Boxes. To make a good evaluation of the new architecture, we conduct both theoretical analysis and experiments on large MCNC benchmark circuits. The experimental results show that the CS-Box FPGA outperforms the traditional symmetrical array FPGA to some degree, e.g., fewer switches, smaller channel width and less circuit delay.

Secondly our work focuses on the channel segmentation design, which also has high impact on the whole chip routability and hardware requirement. Lin *et al.* addressed the Graph Matching-Based Algorithm, which is very useful in general channel segmentation designs. The theoretical analysis and experimental results tell that the algorithm is effective and efficient. After studying the algorithm we find that it is not able to provide an optimal solution. So our work is done to enhance the Graph Matching-Based Algorithm. And the MST-Based Graph Algorithm is proposed. The discussion in theory shows that our algorithm is optimal for both row-based FPGAs and symmetrical array FPGAs. And the experimental results observe a 4.31% reduction of net length by our algorithm.

1.3 Thesis Overview

This thesis is mainly about the FPGA architecture design from different aspects. In each area we give the detailed methods, thorough theoretical discussion and experimental verification. The remainder of this thesis is organized as follows. Chapter 2 introduces the necessary background knowledge of FPGA, CAD and VPR, the router used in our experiments. Chapter 3 addresses the CS-Box structure design. In that chapter we show what the new FPGA architecture looks like and how to determine the switches inside a CS-Box. Also

we analyze the switch number required by the new FPGA and the symmetrical array FPGA. At last the experimental results are given to support our design and theoretical discussion. Chapter 4 is about our MST-Based Graph Algorithm. Based on the Graph Matching-Based Algorithm we formulate the Net Merging Problem in another way. And the theorems and corresponding proof make sure that our algorithm works optimally. During analysis the time complexity is also discussed. We generate the experimental objects with our own generation package and excute the algorithm on them. The experimental results are encouraging. Chapter 5 concludes our work.

Chapter 2

Field-Programmable Gate Array and Routing Algorithm in VPR

Field-Programmable Gate Array (FPGA) is a kind of Very Large Scale Integration (VLSI) circuits. It has been used for nearly twenty years because of its instance manufacturing and very low-cost prototypes. [2] To improve FPGAs' speed and area-efficiency researchers have done much work about FPGA architecture designs. In this chapter we will introduce the background knowledge about FPGAs. And the symmetrical array FPGA routing architecture will be described in details since our primary work concentrates on proposing new FPGA routing architectures with higher efficiency. VPR is a widely used router in current research on FPGAs. Before our applying it we analyze the routing algorithm used by this tool.

2.1 Commercially Available FPGAs

At present commercially available FPGA families include Xilinx, Actel, Altera, Plessey, Plus, Advanced Micro Devices (AMD), QuickLogic, Algotronix, Concurrent Logic and Crosspoint Solutions. Among them the first three types are used most widely.

2.2 FPGA Logic Block Architecture

Functionality is an important characterisctic of FPGA logic blocks. It refers to the number of different boolean logic functions that the block can implement. [2] The functionality of the logic block affects the amount of required routing resources in the FPGA. Since routing resources are connected by programmable switches, which take up significant chip area and have great resistance and capacitance, the functionality of the logic block has great impact on FPGAs' architecture and function.

2.2.1 Logic Block Functionality vs. FPGA Area-Efficiency

The logic block functionality is an important factor that affects FPGA area. According to [2] the total chip area needed for an FPGA consists of the logic block area plus the routing area. And 70% to 90% of the total chip area is occupied by routing. As functionality increases, the number of blocks in the circuit becomes smaller. Since each block accomplishes more functions, logic inside it is more. The total chip area will change due to the relationship of all the factors.

2.2.2 Logic Block Functionality vs. FPGA Delay-Performance

Here we apply the delay model introduced in [2]. It can be expressed in Eq. (2.1),

$$D_{TOT} = N_L \times (D_{LB} + D_R) \tag{2.1}$$

where D_{TOT} is the total delay when one logic block delay and one routing delay are incurred in each block stage. N_L is the number of logic blocks in the critical path. D_{LB} indicates the combinational delay of the logic block. D_R tells the delay between logic blocks when routing.

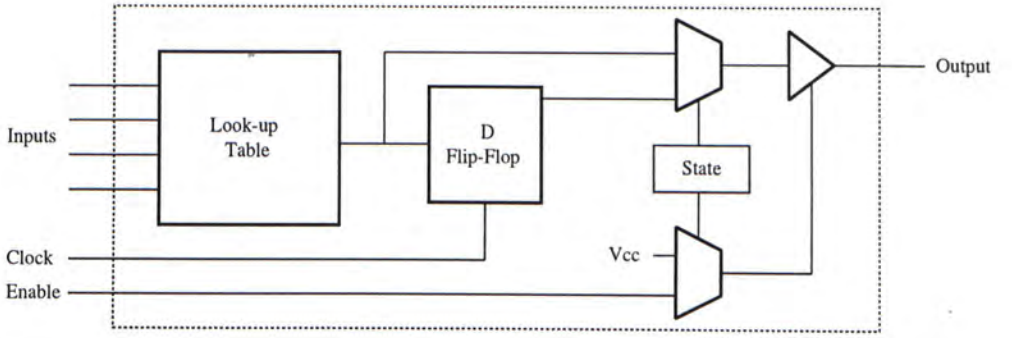


Figure 2.1: Single Output 4-LUT Logic Block, with a D Flip-Flop

Routing Architecture	*Company*
Symmetrical Array	Xilinx, QuickLogic
Row-based	Actel, Crosspoint
Sea-of-gates	Plessey, Algotronix, Concurrent
Hierarchical-PLD	Altera, Plus, AMD

Table 2.1: Commercially Available FPGAs' Routing Architecture Classification

2.2.3 Lookup Table-Based FPGAs

The logic block structure used in an FPGA strongly influences the circuit speed and chip area-efficiency. [15] Currently most FPGAs use logic blocks based on look-up tables (LUTs). A k-LUT is a LUT with k inputs, which requires 2^k SRAM cells and a 2^k-input multiplexer. And most commercial FPGAs are based on 4-LUTs. Figure 2.1 [2] shows a single output 4-LUT logic block.

2.3 FPGA Routing Architecture

The FPGA routing architecture specifies the relative width of the various wiring channels within the chip. [15] Commercial FPGAs can be classified into four groups based on the routing architectures, *symmetrical array*, *row-based*, *sea-of-gates* and *hierarchical PLD*. Figure 2.2 [2] shows the four classes

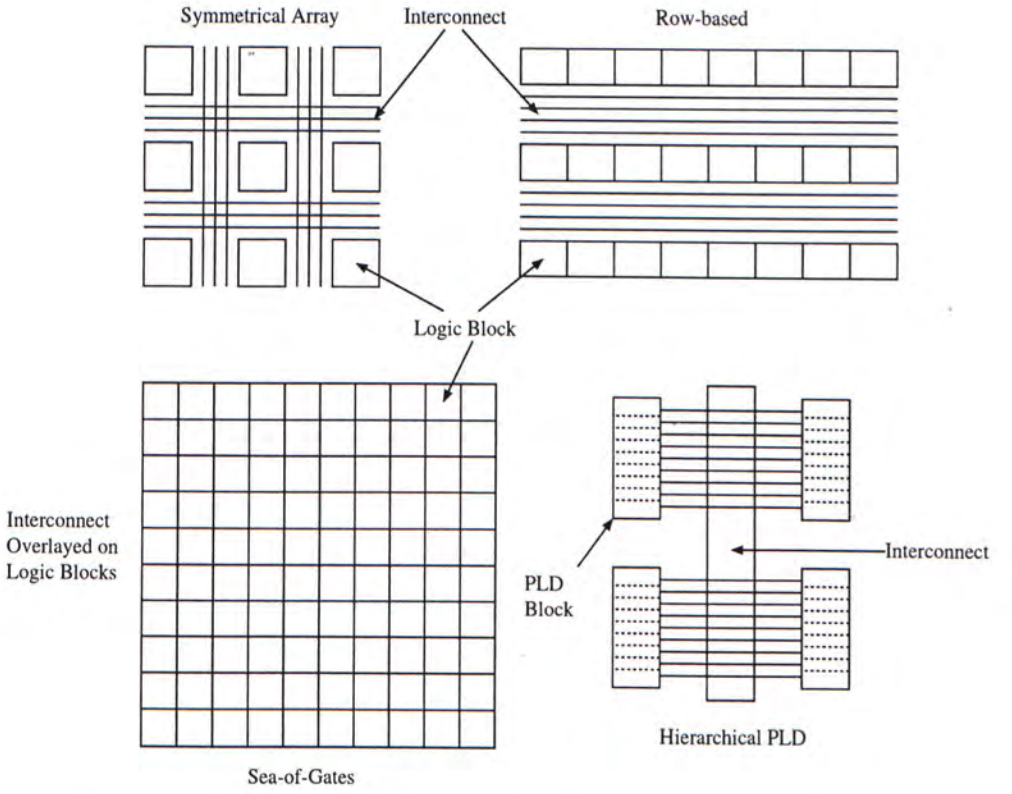


Figure 2.2: The Four Classes of FPGA Routing Architectures

of routing architectures. And Table 2.1 classifies the commercial FPGAs mentioned in Section 2.1 according to their architectures. Our research focuses on the symmetrical array FPGA architecture design. The routing architecture of that kind of FPGAs will be described in details in the following. We use the terminology defined in [2], which is also adopted throughout the thesis.

A symmetrical array FPGA consists of a two-dimensional array of *logic blocks* (Figure 3.1). The logic blocks are surrounded by *routing channels* containing a number of tracks. A *track* is a straight section of wire that spans the entire width or length of a routing channel. It can be composed of *wire segments* of various lengths. The region between two logic blocks is called a *connection box*, in which programmable *switches* connect a pin to some or all of the wire segments in the channel. A *switch box* refers to the area where

a horizontal channel and a vertical channel intersect. Programmable switches exist in the switch box and allow some of the wire segments incident to the box to be connected to others. By programming switches to be ON, users can connect short wire segments to form longer connections. Currently there are four kinds of switch boxes widely applied in FPGAs, *Disjoint*, *Universal Switch Box* (*USB*) [3] [4] [5], *Hyper-Universal Switch Box* (*HUSB*) [6] [7] and *Wilton* [1].

2.4 Design Parameters of FPGA Routing Architecture

Throughout the thesis we apply the notations defined in [16] as some of the design parameters if there is no special declaration.

W: The channel width. Equals to the number of tracks per channel.

F_s: The flexibility of the switch box. Equals to the total number of possible connections offered to each incoming wire.

F_c: The flexibility of the connection box. Equals to the number of channel wires to which each logical pin can connect.

2.5 CAD for FPGAs

According to [15] Computer-Aided Design (CAD) programs are used to implement a circuit in a modern FPGA. FPGA users describe a circuit at a higher level of abstraction. Then CAD programs convert the description into a programming file, in which each programmable switch and configuration bit has been set to the appropriate state. The converting process can be broken down into three steps, *synthesis and logic block packing*, *placement*, *Routing*. Figure 2.3 [15] helps to describe the whole procedure traceable.

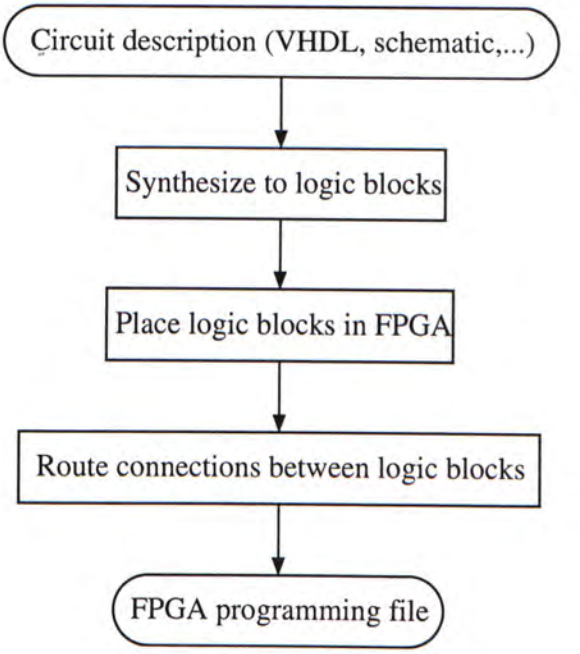


Figure 2.3: FPGA CAD Flow

2.5.1 Synthesis and Logic Block Packing

The synthesis process translates the circuit description provided by users into a netlist of basic gates. And then the netlist of basic gates is converted into a netlist of FPGA logic blocks. The goal of this procedure is to minimize the number of logic blocks and/or maximize the circuit speed. [15] The synthesis and logic block packing stage can be divided into three phases, which is illustrated in Figure 2.4 [15].

2.5.2 Placement

Placement algorithms determine which logic block within an FPGA should implement each of the logic blocks required by the circuit. [15] The placers in use today can be categorized by *min-cut* (*partitioning-based*), *analytic* and *simulated annealing based placers*. [15] The placement aim is to place connected logic block close together to minimize the required wiring, or to balance the

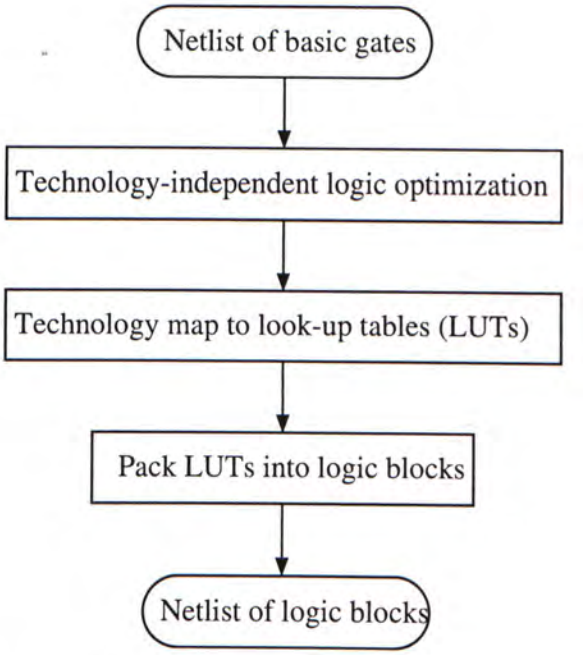


Figure 2.4: Details of Synthesis Procedure

wiring density across an FPGA, or to maximize circuit speed. [15]

2.5.3 Routing

Routing is performed after the placement of all the logic blocks have been decided in a circuit. It determines which programmable switches should be turned on to complete all connections between the logic pins required by the circuit. Logic pins and wire segments are called *routing resources*. Researchers usually use a directed graph to represent an FPGA, which is also the representation method applied in our work. In the graph all routing resources are represented by vertices. And an edge tells which two routing resources are connected. Generally there are two kinds of routers at present. One finds a path for a net in one step including which routing resources are used. The other works in two steps. Firstly, the router determines which logic block pins and channel segments are used. And secondly, the specific wire segments in each channel segments are decided to be occupied. A *channel segment* is the

length of routing channel that spans one logic block. [15]

2.5.4 Delay Modelling

Delay model is a speed measure during FPGA routing. It is used to compute the speed of a circuit and the delay of different net topologies during routing [15]. Today the most widely used delay estimate is *Elmore delay.* Okamoto *et al.* [17] introduced the Elmore delay of a source-sink path, which is shown in Eq. (2.2).

$$\sum_{i \in Source-sinkpath} R_i \cdot C(subtree_i) + T_{d,i} \tag{2.2}$$

where i denotes a wire, a buffer or a pass transistor. So R_i is the equivalent resistance of such an element (R_{wire}, R_{buf}, R_{pass}). $C(subtree_i)$ is the total capacitance of the dc-connected subtree rooted at the element i, where *dc-connected* means *directly connected by wires.* $T_{d,i}$ is the intrinsic buffer delay if i is a buffer, and 0 otherwise.

2.5.5 Timing Analysis

After the placement and routing of a circuit the timing analysis is necessary to determine the speed of the circuit. Also timing analysis is used to estimate the slack of each source-sink connection during the whole CAD flow. That helps to route some connections via fast paths to avoid slowing down the circuit. [15]

2.6 FPGA Programming Technologies

The connection between two routing resources is realized by setting the switch ON. Programming technologies are used to implement switches in FPGAs. Currently the main technologies include Static RAM (SRAM), fuse, Anti-fuse, Erasable Programmable Read-Only Memory (EPROM) and Electrically Erasable Programmable Read-Only Memory (EEPROM).

2.7 Routing Algorithm in VPR

VPR is a tool and the name means versatile packing, placement and routing for symmetrical array FPGAs. In our CS-Box structure design VPR is used to conduct the experiments on MCNC benchmark circuits. Here we investigate the routing algorithm in VPR. VPR's routing algorithm is based on the Pathfinder negotiated congestion algorithm [18] [19] and overcomes Pathfinder's shortcoming of much CPU time when processing high-fanout nets.

2.7.1 Pathfinder Negotiated Congestion Algorithm

Pathfinder routes each net in the netlist by the shortest path it can find regardless of any routing resource's overuse initially. Then it rips up and re-routes each net by the lowest cost path. The cost of using a routing resource can be expressed by Eq. (2.3).

$$(c_n + h_n) \times p_n \tag{2.3}$$

where c_n is the basic cost of using the resource n, h_n is the history cost of using the resource, and p_n is the number of signals sharing the resource at present.

After initially routing all nets, iterations are turned on until there are no shared routing resources. In each iteration every net is ripped up and re-routed by the lowest cost path. By increasing the costs of the routing resources gradually, the nets with alternative routes are forced to avoid overusing resources. And those resources are left only to the nets that need them most afterwards.

When re-routing each net, we keep a tree RT to store the partial net that has been routed. RT is initialized as the source of the processed net and is used as the expansion wavefront when connecting the unconnected sinks to the partial net. And after routing the whole net the history costs of the resources on the net are updated.

The Dijkstra's algorithm is applied in Pathfinder to find the lowest cost

```
Serial pathfinder(netlist, FPGA architecture)
1     while there are shared routing resources
2           for each net N_i
3                 RT ← source node of N_i
4                 while there are sinks of N_i which are not connected to the source
5                       P_j ← findpath(RT)
6                       add P_j to RT
7                 endwhile
8                 update cost of nodes based on congestion history
9           endfor
10    rip up nets if they have shared resources
11    endwhile


findpath(RT)
begin
      for each node in RT
            enqueue n onto Q with key 0
      endfor
      while a new sink has not been found
            dequeue node, m, with lowest key from Q
            if m was not previously dequeued
                  for each neighbor n of m
                        enqueue n on Q with cost of n plus key of m
                  endfor
      endwhile
      backtrace from the found sink till a node of RT is reached
            and return this path
end
```

Figure 2.5: Pathfinder Negotiated Congestion Algorithm

path from the expansion wavefront - the partial net that has been routed - to one sink of the net. Initially all nodes in RT are put into a queue with a key 0. Then the node with the lowest key in the queue is deleted and all its neighbors' keys are updated by plusing its key. The deleting action will stop when an unconnected sink of the net is found. The path connecting the new sink to the partial net is obtained by back tracing from the sink to a node in the expansion wavefront - RT.

The Pathfinder negotiated congestion algorithm is illustrated in Figure 2.5 [20].

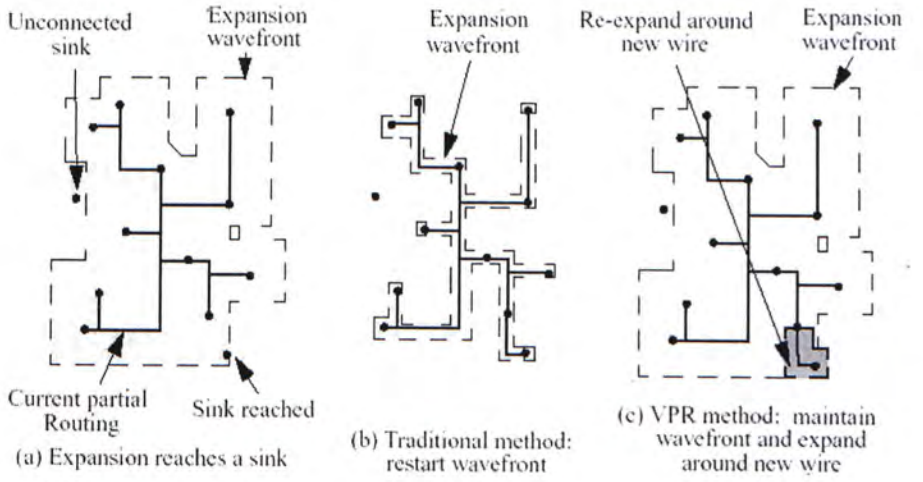


Figure 2.6: Difference between Pathfinder and VPR's Routing Algorithm

2.7.2 Routing Algorithm Used by VPR

Pathfinder's using much CPU time is its main drawback. When a new sink is found, all the nodes in *RT* are regarded as source nodes. Since Pathfinder empties the expansion wavefront after finding a new sink, for high-fanout nets it will take the router a large amount of time to expand from the partial routed net to the next sink. VPR modified Pathfinder's expansion wavefront and obtained a routing algorithm occupying less CPU time.

Instead of emptying the wavefront after finding a new sink, VPR does not only keep the current expansion wavefront but also adds the new path connecting the newly found sink to the wavefront with cost 0. Since the added part is fairly small, it will take little time to add this part to the partial net. Then when looking for the next sink, the maze router will expand around the added path first instead of starting from the scratch, which allows that the next sink can be found more quickly. The modification made on the Pathfinder algorithm can be illustrated in Figure 2.6 [21].

Chapter 3

Connection-Switch Box Design

3.1 Introduction

FPGA has been used for over a decade with its wide applications in digital systems. The conceptual diagram of the symmetrical array FPGA is illustrated in Figure 3.1 [2]. A symmetrical array FPGA consists of a two-dimensional array of logic blocks. The logic pins are connected by segments of wire. Two types of programmable switches exist in this FPGA architecture. The first type exist in the connection box (C-Box) and connect pins to wire segments or vice versa. The second type build connections between two different segments of wire. And they are in the switch box (S-Box). The architecture of an FPGA has high impact on its routability. Based on the symmetrical array FPGA architecture, many research works have addressed on the optimal switch box designs. Currently the most widely used switch box structures are *Disjoint*, *Universal Switch Box* (USB) [3] [4] [5], *Hyper-Universal Switch Box* (HUSB) [6] [7] and *Wilton* [1]. Their routabilities have been evaluated in [8]. Another advanced technology about FPGA architecture design is the Virtex architecture proposed by the Xilinx Company. The Xilinx Virtex Architecture has been applied in practice since their introduction. Figure 3.2 is the diagram showing that the Virtex architecture uses the Connection-Switch Box structure.

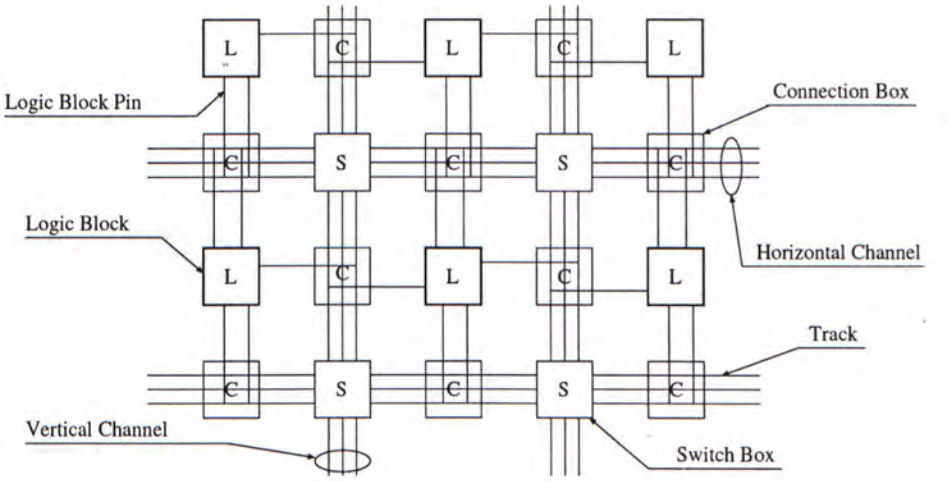


Figure 3.1: Symmetrical Array FPGA Architecture

Based on the existing technologies of FPGA architecture design, we design our own Connection-Switch Box (CS-Box) to reduce the hardware resources. The increased flexiblity inside the CS-Box improves the overall chip routability and reduces the switch number of the entire circuit. The new FPGA conceptual diagram is shown in Figure 3.3. It is based on the symmetrical array FPGA architecture. The difference is that the separate C-Box and S-Box structures are combined to form the CS-Box structure in the new FPGA. The switch building algorithm we designed tries to use as few switches as possible to accomplish routing. After theoretical analysis we conduct experiments on large MCNC benchmark circuits. The experimental results are encouraging. They show a 11.81% reduction of switch numbers from CS-Box FPGAs to symmetrical array FPGAs together with small penalty of channel widths and circuit delays. As F_{cl}, the connectivity of logic pins, in CS-Box FPGAs increases, the switch number reduction is a little degraded. While the penalty in channel widths and circuit delays is eliminated.

This chapter is organized as follows. Section 3.2 introduces the connection algorithm. Section 3.3 analyzes the number of the switches connecting logic pins and wire segments. To achieve a fair evaluation on the proposed

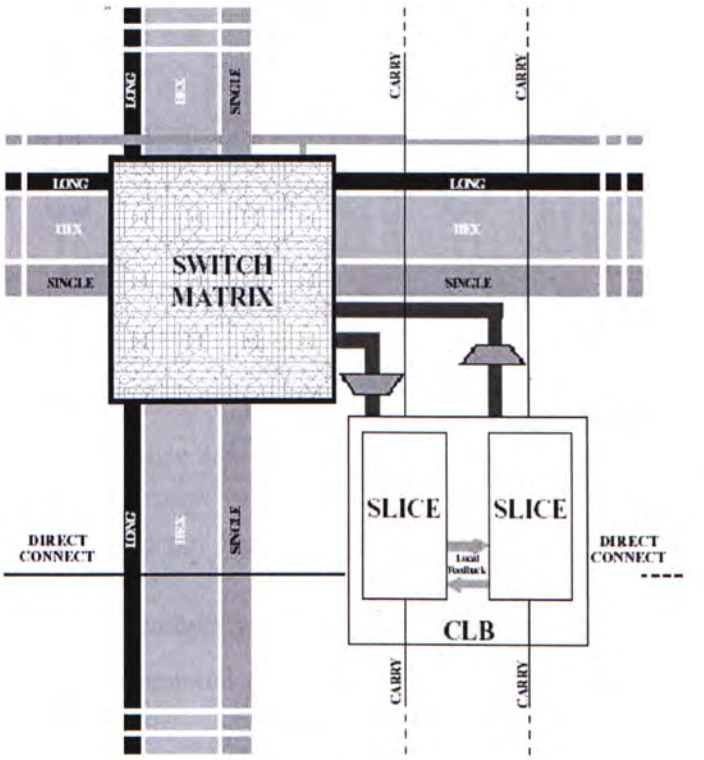


Figure 3.2: Xilinx Virtex Architecture

architecture, we compare it with the symmetrical array FPGA architecture on different objectives. Section 3.4 shows the experimental results with the comparison of channel widths, circuit delays and switch numbers. Section 3.5 draws our conclusions on this work.

3.2 Connection-Switch Box Design Algorithm

We designed two algorithms to do the connection between pins and tracks in the FPGA with CS-Boxes. One is for logic pins and the other for pad pins. Some necessary notations are defined as follows,

W : The channel width.

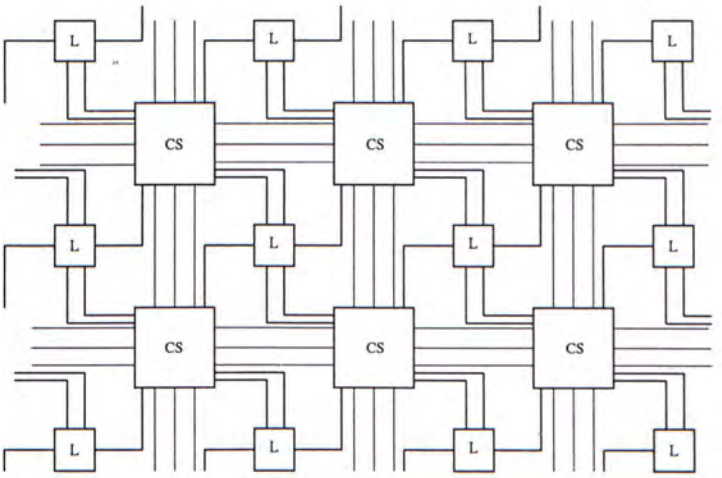


Figure 3.3: CS-Box FPGA Architecture

P : The logic (pad) pin number on each logic block (pad).

F_{cl} : The logic pin connectivity that denotes how many tracks in one channel a logic pin is connected to.

F_{cp} : The pad pin connectivity that denotes how many tracks in one channel a pad pin is connected to.

According to Figure 3.3 the CS-Box contains the two kinds of switches. And each pin connects to the tracks on the sides except the one it belongs to. The following two subsections introduce the two algorithms in details.

3.2.1 Connection between Logic Pins and Tracks

In this algorithm we regard each CS-Box as a 4-partite graph. The track(s) and the non-global logic pin(s) on one side compose of a disjoint vertex set of the graph. Here we call them track vertices (T-track) and pin vertices (P-vertex). Then the problem of building switches in the CS-Box can be formulated as settling the edges between P-vertices and T-vertices in the 4-partite graph.

According to the above notations, P is the total number of P-vertices in the 4-partite graph. W is the number of T-vertices in each disjoint set of the

graph. All the P-vertices in the graph are labeled from 1 to P. And the T-vertices in each disjoint set are labeled from 1 to W. We use v_{pi} to denote the ith P-vertex and $v_{ti,j}$ to denote the jth T-vertex of the ith disjoint vertex set. We name such a 4-partite graph as *PT-graph*. Then the switch design problem can be formulated as follows,

Problem Formulation

Let G be a PT-graph. S_i and S_j $(1 \leqslant i, j \leqslant 4$ and $i \neq j)$ are disjoint vertex sets of G. v_{pm} is a P-vertex in the graph, where $v_{pm} \in S_i$ and $1 \leqslant m \leqslant P$. A solution to this problem is that each $v_{pm} \in S_i$ should be connected to some T-vertices in S_j. The goal is that the number of the edges is as small as possible and the number of the connected T-vertices is as large as possible.

Definition 3.1 A *W-PT-graph* is a PT-graph that has W T-vertices in each disjoint vertex set.

We note that in the four disjoint vertex sets of a PT-graph, the numbers of T-vertices are the same and the numbers of P-vertices are different. Thus we construct i to i matching from the P-vertices to the T-vertices. That is, in the PT-graph the edges are built by connecting the ith P-vertex to the ith T-vertex of each set. Let $H = \{V, E\}$ be a W-PT-graph. By that method each incoming wire segment has connection with pins. And also the number of switches is not too large — in symmetrical array FPGAs the switches between pins and segments are too many if the pin connectivity is set to be 1.0. And in the following we will describe the detailed algorithm on building edges.

Algorithm Description

Case 1 $W = P$

This is the simplest case considered by the algorithm. The ith P-vertex

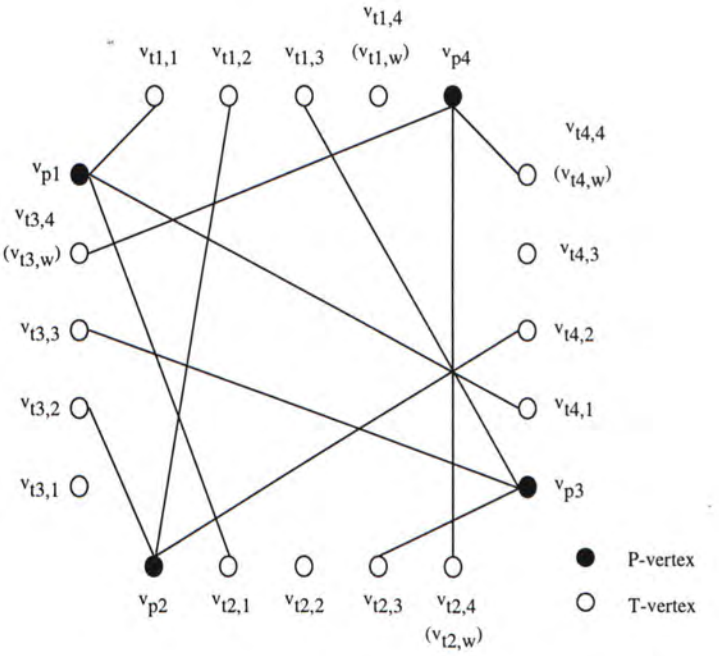


Figure 3.4: W-PT-Graph H. Case 1: $W = P$

is connected to the ith T-vertices which are not on the same side with the P-vertex. So the edge set of H is

$$E = \{(v_{pj}, v_{ti,j}) | v_{pj} \notin S_i; 1 \leqslant i \leqslant 4; 1 \leqslant j \leqslant W(P)\} \tag{3.1}$$

Figure 3.4 illustrates the 4-PT-graph H in which the edges have been decided.

Case 2 W mod $P = 0$ and $W \neq P$

Here Case 1 is used as the unit case to work out the solution. Each disjoint vertex set of H can be divided into $\frac{W}{P}$ subsets. And all the subsets can be denoted by

$$V_k^{'} = \{v_{ti,p} || p - k | \text{mod} P = 0; 1 \leqslant i \leqslant 4; 1 \leqslant p \leqslant W; 1 \leqslant k \leqslant P\} \tag{3.2}$$

The edges are set between the vertices in $V_k^{'}$ and the kth P-vertex. So

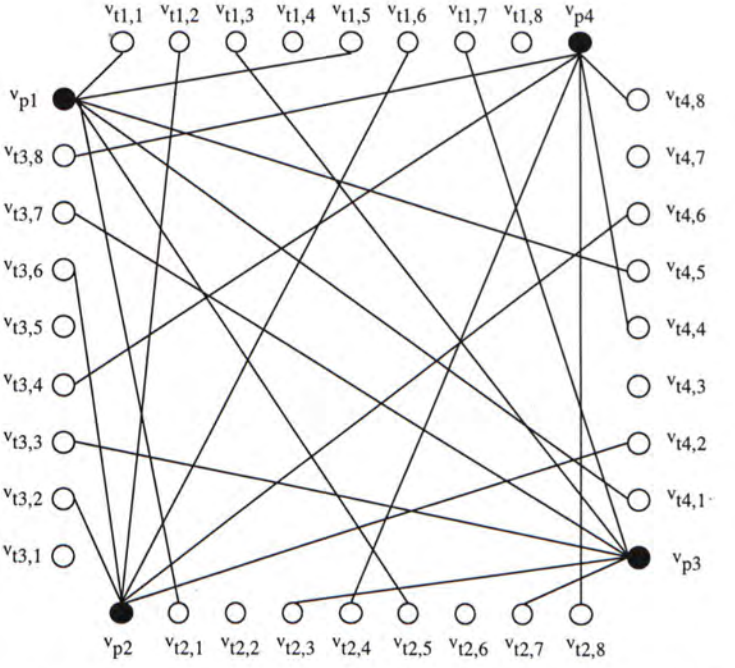


Figure 3.5: W-PT-Graph H. Case 2: $W \bmod P = 0$ and $W \neq P$

the edge set of H is

$$E = \{(v_{pj}, v_{ti,k}) | v_{pj} \notin S_i; |k-j| \bmod P = 0; 1 \leqslant k \leqslant W; 1 \leqslant j \leqslant P; \\ 1 \leqslant i \leqslant 4\} \tag{3.3}$$

Figure 3.5 illustrates the graph H in which the edges are built.

Case 3 $W \bmod P \neq 0$

Let W' be an integer whose value is calculated by Eq. (3.4).

$$W' = W - (W \bmod P) \tag{3.4}$$

In this case we divide the T-vertices into two groups. The first group can be denoted by the following set (Eq. (3.5)),

$$\{v_{ti,j} | 1 \leqslant i \leqslant 4 \text{ and } 1 \leqslant j \leqslant W'\} \tag{3.5}$$

And the other T-vertices compose of the second group. Now it is clear that the first group corresponds to Case 2. The edges related to them

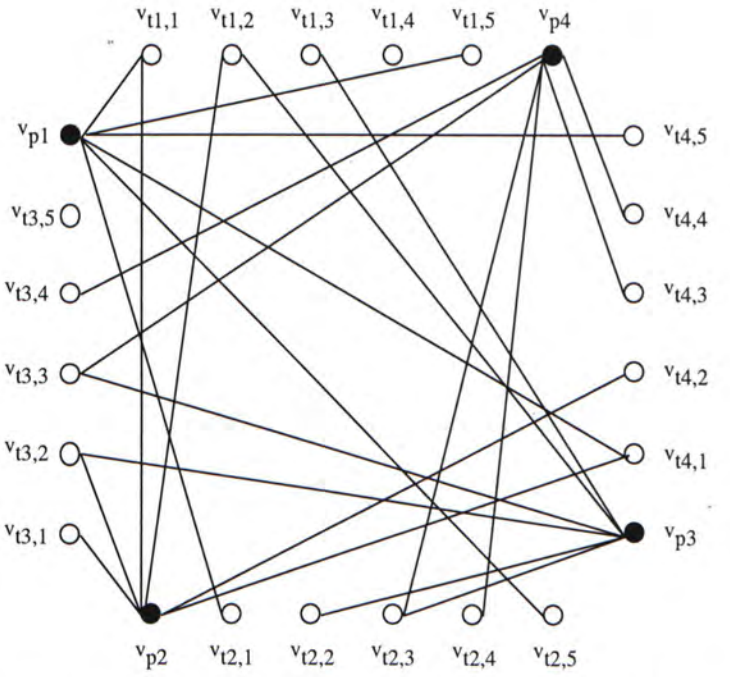


Figure 3.6: W-PT-Graph H. Case 3: $W \bmod P \neq 0$

can be determined in the way introduced in Case 2. That is,

$$E' = \{(v_{pj}, v_{ti,k}) | v_{pj} \notin S_i; |k-j| \bmod P = 0; 1 \leqslant k \leqslant W'; 1 \leqslant j \leqslant P; 1 \leqslant i \leqslant 4\} \tag{3.6}$$

For the T-vertices in the second group, we can still build the edge set E'' in the way similar to which has been introduced in Case 2 after modifying the labels of the first $P - W + W'$ T-vertices in each disjoint set. That is, for each $v_{ti,j}$ where $1 \leqslant j \leqslant P - W + W'$, relabel it as $v_{ti,j+W}$. Thus,

$$E'' = \{(v_{pj}, v_{ti,k}) | (W + k - j) \bmod P = 0; v_{pj} \notin S_i; 1 \leqslant k \leqslant P - W + W'; 1 \leqslant j \leqslant P; 1 \leqslant i \leqslant 4\} \tag{3.7}$$

Till now we can get the complete edge set E of H by

$$E = E' \cup E'' \tag{3.8}$$

Figure 3.6 illustrates the graph H with the edges settled.

We have introduced the algorithm on the connection between logic pins and tracks. According to the above design, the connectivity of each logic pin can be calculated by Eq. (3.9).

$$F_{cl} = (\lfloor \frac{W}{P} \rfloor + \beta) \times 3 \tag{3.9}$$

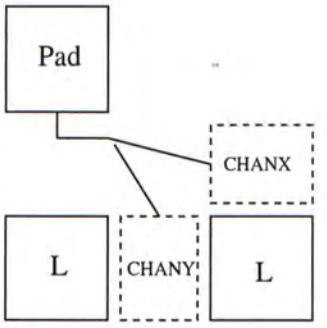
where

$$\beta = \begin{cases} 0, & \text{if } W \bmod P = 0 \\ 1, & \text{otherwise} \end{cases} \tag{3.10}$$

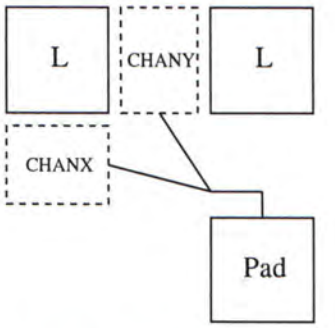
3.2.2 Connection between Pad Pins and Tracks

We proposed a separate algorithm for the connection between pad pins and tracks because of the structural difference between pads and logic blocks. Figure 3.7 shows the conceptual diagram of the connection between pad pins and tracks. Let v_p be the pad vertex (P-vertex) denoting a pad pin. Let v_{xi} (v_{yi}) be the track vertex (T-vertex) denoting a track in the x-channel (y-channel). The switches to be determined are represented by the edges connecting P-vertices and T-vertices.

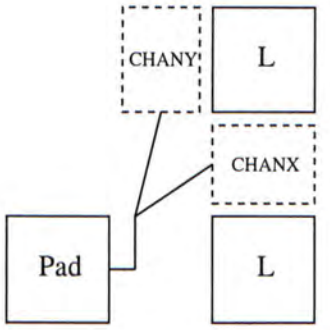
According to the definitions at the beginning of Section 3.2, F_{cp} is the connectivity of each pad vertex. In previous work we tried to use the FPGA router VPR [15] [21] to route the benchmark circuits on symmetrical array FPGAs. A fact is that some of the circuits are not able to be routed if F_{cp} equals to some number smaller than W, while all of them are able to be routed when F_{cp} equals to W. Therefore, in our design F_{cp} equals to W. The algorithm separates all the edges adjacent to one P-vertex into two groups. The edges in the first group are adjacent to the T-vertices in the x-channel; and this group is indicated by E_x. The other edges are adjacent to the T-vertices in the y-channel; and the group containing them is indicated by E_y. According



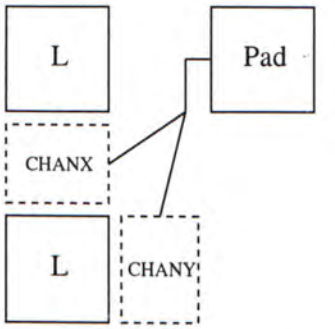
(a)Pad on The Topmost Row



(b)Pad on The Bottommost Row



(c)Pad on The Leftmost Column



(d)Pad on The Rightmost Column

Figure 3.7: Connection between Pad Pins and Tracks

to our algorithm, E_x and E_y can be represented by the following two sets,

$$E_x = \begin{cases} \{(v_p, v_{xi}) | i \bmod 2 = 0; 1 \leqslant i \leqslant W\}, & \text{if } W \bmod 2 = 0 \\ \{(v_p, v_{xi}) | i \bmod 2 = 1; 1 \leqslant i \leqslant W\}, & \text{otherwise} \end{cases} \tag{3.11}$$

$$E_y = \begin{cases} \{(v_p, v_{yi}) | i \bmod 2 = 1; 1 \leqslant i \leqslant W\}, & \text{if } W \bmod 2 = 0 \\ \{(v_p, v_{yi}) | i \bmod 2 = 0; 1 \leqslant i \leqslant W\}, & \text{otherwise} \end{cases} \tag{3.12}$$

3.3 Switch Number Comparisons

In this section we discuss the number of the switches connecting logic pins with tracks in the new FPGA architecture. Figure 3.8 shows the CS-Box with $W = 2$. The switches connecting two tracks are not shown in Figure 3.8

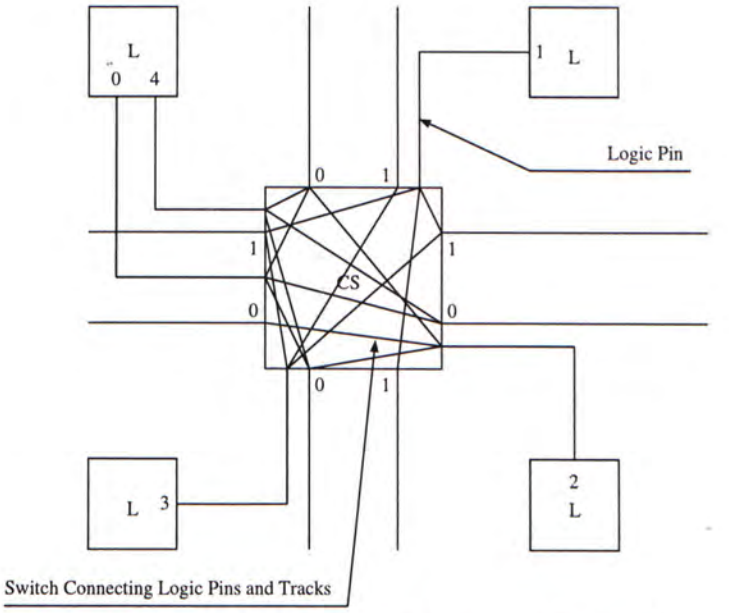


Figure 3.8: Connection-Switch Box with $W = 2$

for simplicity. The following variables are defined. W and P have the same definition as in Section 3.2.

N_{cs} : It is defined for each CS-Box and equals to the number of the switches connecting logic pins and tracks.

N_x : It is defined for each logic block in symmetrical array FPGAs and equals to the number of the switches connecting logic pins and tracks.

α : The connection of the switches connecting logic pin in the symmetrical array FPGA. It indicates how much percent of tracks the logic pin is connected to in one channel.

According to the algorithm introduced in Section 3.2,

$$N_{cs} = (\lfloor \frac{W}{P} \rfloor + \beta) \times 3 \times P \tag{3.13}$$

where β satisfies Eq. (3.10). In the symmetrical array FPGA,

$$N_x = \alpha \times W \times P \tag{3.14}$$

Let k be a non-negative integer. And W is denoted in terms of k by Ineq. (3.15).

$$kP < W \leqslant (k+1)P \tag{3.15}$$

And also

$$\lfloor \frac{W}{P} \rfloor + \beta = k+1 \tag{3.16}$$

N_{cs} and N_x can be indicated in Eq. (3.17) and Ineq. (3.18).

$$N_{cs} = (k+1) \times 3 \times P \tag{3.17}$$

$$\alpha \times k \times P^2 < N_x = \alpha \times W \times P \leqslant (k+1) \times \alpha \times P^2 \tag{3.18}$$

During the following discussion we are going to find the condition of $N_{cs} \leqslant N_x$ by Eq. (3.17) and Ineq. (3.18).

1. First, the necessary condition for $N_{cs} \leqslant N_x$ is

$$(k+1) \times 3 \times P \leqslant (k+1) \times \alpha \times P^2 \tag{3.19}$$

From Eq. (3.19) we can get

$$\alpha \geqslant \frac{3}{P} \tag{3.20}$$

2. Second, the sufficient condition for $N_{cs} \leqslant N_x$ is

$$(k+1) \times 3 \times P \leqslant \alpha \times k \times P^2 \tag{3.21}$$

From Eq. (3.21), we can get

$$k \geqslant \frac{3}{\alpha \times P - 3} \tag{3.22}$$

and

$$\alpha > \frac{3}{P} \tag{3.23}$$

3. Third, we consider that both $N_{cs} \leqslant N_x$ and $N_x < N_{cs}$ occur.

$$\alpha \times k \times P^2 < N_{cs} \leqslant (k+1) \times \alpha \times P^2 \tag{3.24}$$

(1) If $\alpha = \frac{3}{P}$, there is

$$N_{cs} = (k+1) \times P \times 3 \tag{3.25}$$

$$k \times P \times 3 < N_x \leqslant (k+1) \times P \times 3 \tag{3.26}$$

We can see that $N_{cs} = N_x$ when $N_x = (k+1) \times P \times 3$. But in most cases, $N_{cs} > N_x$.

(2) If $\alpha > \frac{3}{P}$, there is

$$k < \frac{3}{\alpha \times P - 3} \tag{3.27}$$

Take the example of $P = 5$ and $\alpha = 1.0$. Figure 3.9 illustrates the comparisn of N_{cs} and N_x. It is clear that $N_{cs} > N_x$ occurs only if $W < 3$. As P increases, N_cs becomes smaller.

From the above analysis, the FPGA architecture with CS-Boxes uses fewer switches to connect logic pins and tracks than the symmetrical array FPGA. The next section will verify this analysis with extensive experiments.

3.4 Experimental Results

After the theoretical analysis and comparison, we observe the loss and gain of the routability, performance and size on the two types of FPGAs on the platform of a 1500 MHz Intel Pentium 4 PC with 512M RAM. The currently best known router VPR [15] [21] is used to test the routability and efficiency of the CS-Box structure. The iteration number for routing is 100. And the routing algorithm is the Breadth-First Search Algorithm. The Wilton switch box

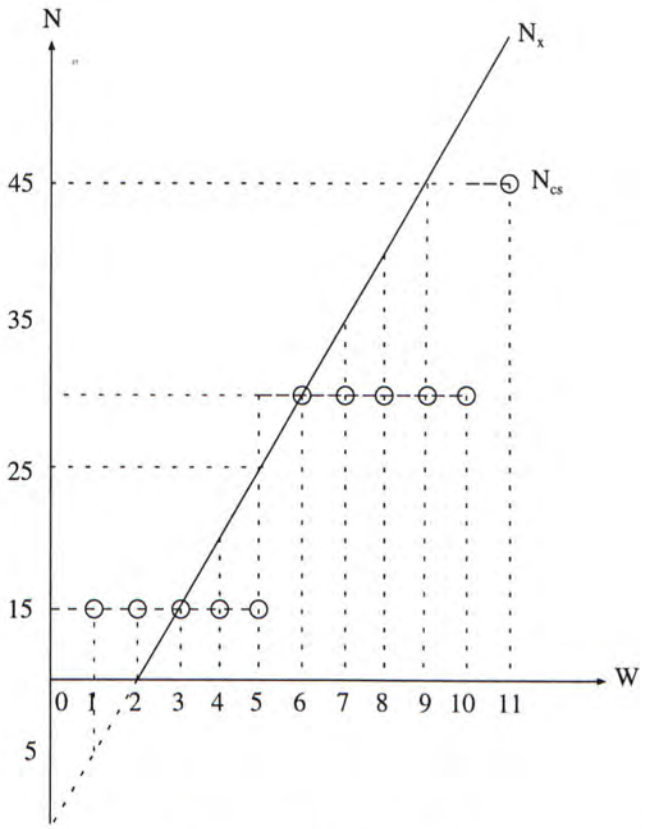


Figure 3.9: Comparison Diagram of N_{cs} and N_x when $\alpha = 1.0$ and $P = 5$

is applied to make the connection between tracks, i.e., S-Box in symmetrical array FPGAs. We try different values of F_{cl} to route the benchmark circuits. F'_{cl} is defined as *basic value* and computed by Eq. (3.28).

$$F'_{cl} = (\lfloor \frac{W}{P} \rfloor + \beta) \times 3 \tag{3.28}$$

where, β satisfies Eq. (3.10). Then Eq. (3.29) is used to obtain the different F_{cl} values, where, $x = 0, 1, 2, \ldots$.

$$F_{cl} = F'_{cl} + x \times 3 \tag{3.29}$$

Table 3.1 shows the channel width requirements of different benchmark circuits. The circuit delay information is presented in Table 3.2. In Table 3.3 we show the number of the switches required to connect logic pins and tracks

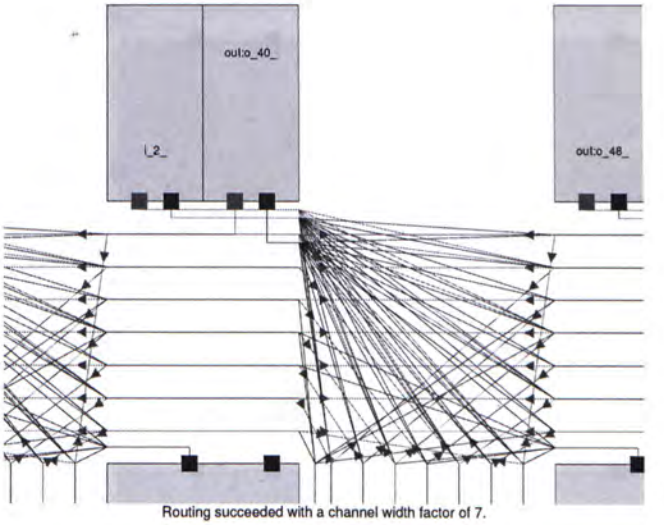


Figure 3.10: CS-Box Structure: Pad Pin Connection

in each benchmark circuit. From the tables, applying CS-Boxes can reduce switch numbers with small penalty of circuit delays and channel widths when $F_{cl} = F'_{cl}$ is used in the new FPGA. As F_{cl} of the CS-Box FPGA increases, the reduction of switch numbers becomes less while the penalty on circuit delays and channel widths is eliminated. Figure 3.10 and 3.11 show the switch connections on the proposed design. The entire routing of the e64 benchmark circuit is shown in Figure 3.12.

We have conducted experiments on the proposed CS-Box FPGA and the traditional symmetrical array FPGA. From Table 3.3 the channel width has an average reduction of 6.99% when $F_{cl} = F'_{cl} + 1$. At the same time the circuit delay and the switch number are reduced by 2.10% and 4.91% on average. From these experimental results we observe that the FPGA routability has been improved by applying the CS-Box structure.

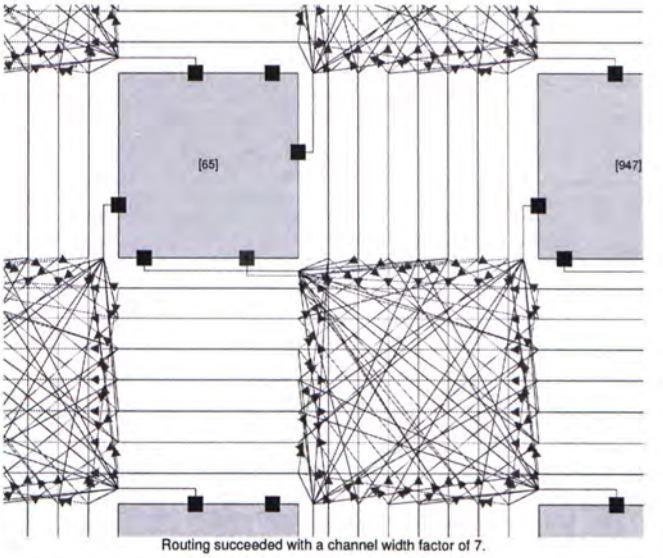


Figure 3.11: CS-Box Structure: Logic Pin Connection

3.5 Summary

We develop a new FPGA architecture, in which the disjoint C-Box and S-Box structures are combined to the CS-Box structure. There are two types of switches in the CS-Box: one of the switch boxes introduced at the beginning of Chapter 3 is used to make connection between tracks, and the second type are the switches between pins and tracks. We introduce the connection algorithm to build the second type of switches. In the theoretical analysis of the switch number, we take the example of $P = 5$ and $\alpha = 1.0$ and find $N_{cs} \leqslant N_x$ when $W \geqslant 3$. The experimental results show a 11.81% reduction on average in the number of switches inside the C-Box from the symmetrical array FPGA to the CS-Box FPGA, accompanied by small penalty of channel widths and circuit delays when $F_{cl} = F'_{cl}$ in CS-Box FPGAs. The larger F_{cl} in the new FPGA, the smaller the reduction of switch numbers. But the penalty of the channel widths and circuit delays is eliminated.

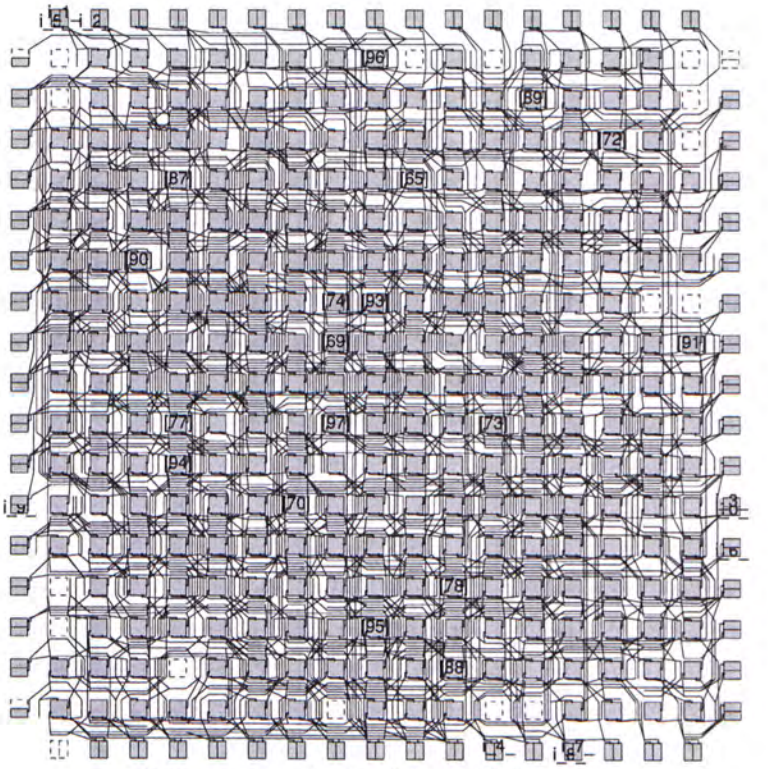


Figure 3.12: Routing Result of *e64* by Using CS-Boxes, $W = 7$

FPGA Architecture	Symmetric-Array			with CS-Box Structure	
F_{cl}	W	$0.9W$	$0.8W$	F'_{cl}	$F'_{cl}+1$
Circuit Name	-	-	-	-	-
alu4	10	10	10	10	9
apex2	11	11	11	11	11
apex4	13	12	13	13	13
b9	4	4	4	4	3
bigkey	6	6	6	6	6
dalu	6	6	6	6	6
des	8	7	7	6	5
diffeq	7	8	8	8	7
dsip	7	7	7	7	6
e64	7	7	8	7	7
elliptic	10	11	10	11	10
ex1010	10	10	10	10	10
ex5p	13	13	13	13	13
misex3	10	10	10	11	10
my_adder	4	4	4	4	3
s1423	5	5	6	6	4
s298	8	8	8	8	7
s38417	8	8	7	7	7
s38584.1	7	7	7	7	7
seq	11	11	11	11	11
tseng	7	7	7	6	6
unreg	4	4	4	5	3
Total	176	176	177	177	164
Comparison	*			+0.57%	-6.82%
		*		+0.57%	-6.82%
			*	0	-7.34%
Average Comparison Result				+0.38%	-6.99%
"*" indicates the result in the column is used to make comparison					

Table 3.1: Channel Width Requirements

FPGA Architecture	Symmetric-Array			with CS-Box Structure	
F_{cl}	W	$0.9W$	$0.8W$	F'_{cl}	$F'_{cl}+1$
Circuit Name	-	-	-	-	-
alu4	1.85386	1.56852	1.90095	1.66109	1.66125
apex2	1.607	1.63341	1.3474	1.73219	1.76429
apex4	1.42238	1.43766	1.5797	1.31174	1.26781
b9	0.36672	0.341982	0.365442	0.293081	0.446448
bigkey	1.62792	1.98333	2.40621	1.48558	1.49052
dalu	1.33564	1.22028	1.18171	1.43429	1.10201
des	1.69223	2.04452	1.54254	2.39677	2.09814
diffeq	1.39117	0.977372	1.11989	1.14553	1.07257
dsip	1.20737	1.56772	1.52951	1.78454	1.72232
e64	0.493154	0.43602	0.424841	0.416376	0.431577
elliptic	2.86504	1.96451	2.64456	2.01054	2.52907
ex1010	3.8845	3.35407	3.66273	3.84098	3.12935
ex5p	1.37349	1.22278	1.26679	1.34682	1.32818
misex3	1.71805	1.44106	1.53773	1.68674	1.96567
my_adder	0.534498	0.530612	0.547252	0.596041	0.449367
s1423	0.73654	0.675588	0.651449	0.664029	0.592272
s298	2.70654	2.2548	2.34736	2.68695	2.29187
s38417	1.72278	1.83241	1.79846	2.36803	1.84059
s38584.1	1.72601	2.7448	2.61386	1.98874	2.59543
seq	1.48821	1.82375	1.90447	1.78949	1.54419
tseng	1.5054	1.25099	1.17455	1.18151	1.05066
unreg	1.99059	0.187206	0.1983	1.81475	1.52287
Total	33.457561	32.49339	33.745704	34.002532	32.525871
Comparison	*	*	*	+1.63% +4.64% +0.76%	-2.78% +0.09% -3.61%
Average Comparison Result				+2.34%	-2.10%
"*" indicates the result in the column is used to make comparison					

Table 3.2: Circuit Delay Comparisons ($e^{-7}ns$)

FPGA Architecture	Symmetric-Array			with CS-Box Structure	
F_{cl}	W	$0.9W$	$0.8W$	F'_{cl}	$F'_{cl}+1$
Circuit Name	-	-	-	-	-
alu4	80000	72000	64000	48000	72000
apex2	106480	968000	87120	87120	87120
apex4	84240	71280	66096	58320	58320
b9	2420	2420	1936	1815	3630
bigkey	87480	75816	72900	87480	87480
dalu	34680	30056	28900	34680	34680
des	158760	123039	119070	119070	119070
diffeq	53235	54756	47151	45630	68445
dsip	102060	90396	87480	87480	131220
e64	10115	8959	8959	8670	8670
elliptic	186050	186050	148840	167445	167445
ex1010	231200	208080	184960	138720	138720
ex5p	70785	65340	55539	49005	49005
misex3	72200	64980	57760	64980	64980
my_adder	980	980	784	735	1470
s1423	5625	5625	5625	6750	6750
s298	77440	69696	60016	58080	87120
s38417	262440	236196	196830	196830	196830
s38584.1	229635	203391	196830	196830	196830
seq	97020	88220	79380	79380	79380
tseng	38115	33759	32670	32670	32670
unreg	980	980	784	735	1470
Total	1991940	1788799	1603630	1570425	1693305
Comparison with *	*			-21.16%	-14.99%
		*		-12.21%	-5.34%
			*	-2.07%	+5.59%
Average Comparison Result				-11.81%	-4.91%
"*" indicates the result in the column is used to make comparison					

Table 3.3: Switch Number Requirements

Chapter 4

Optimal MST-Based Graph Algorithm on FPGA Segmentation Design

4.1 Introduction

Figure 4.1 and 4.2 show two types of FPGA architectures that are most widely used today, the row-based FPGA and the symmetrical array FPGA. In FPGAs signals are transmitted between logic blocks through wire segments, between which switches exist. Users can have two short segments connected by programming the switch between them to be ON. A switch set to be ON can keep or change the direction in which a signal propagates. Also switches can help choose relatively short path(s) from the source to the sink(s). But the problem is that switches have high resistance and capacitance. Using a large amount of such components may incur great delay penalty [10] [22] [23]. To get a good balance between the routability and the switch number researchers have attempted to design various segmentation architectures for FPGAs in [9] [10] [11] [12] [13] [14] [24] [25]. Among those literatures Chang *et al.* [24] [25] proposed the Graph Matching-Based Algorithm for FPGA segmentation design and routing. The algorithm considers the similarity of input routing instances

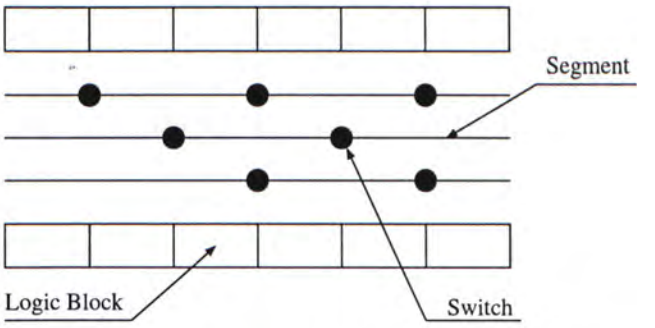


Figure 4.1: Row-Based FPGA Architecture

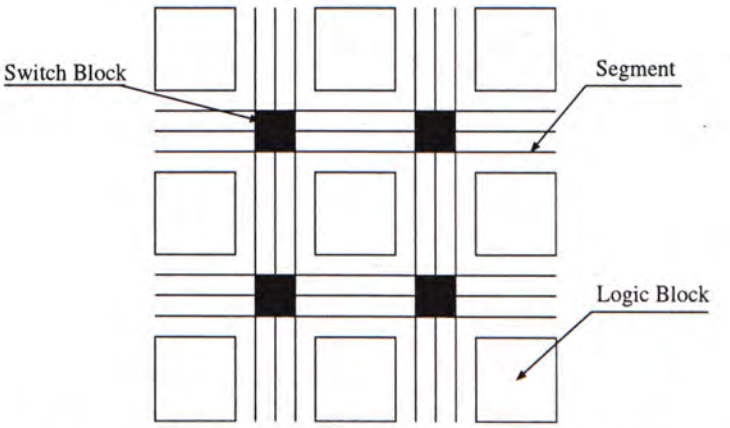


Figure 4.2: Symmetrical Array FPGA Architecture

and formulates the Net Matching Problem to construct the segmentation architecture. It can provide segmentation outperforming those used in commercially available FPGAs [25]. But the algorithm considers two routing instances each time and merges all instances in a tree-like bottom-up manner. Therefore, the result is dependant on the paring scheme. And the experiments show an average variation of 5% by using two paring scheme [10].

We improve the Graph Matching-Based Algorithm and propose the Minimum Spanning Tree (MST)-Based Graph Algorithm. The MST-Based Graph Algorithm considers all input instances at once. The merging result is independent of the paring scheme. It is proved to be optimal for row-based FPGAs in $O(m^3n^3)$, where m is the number of routing instances and each routing instance contains at most n nets. Then it is extended for symmetrical array FPGAs

and also proven to be optimal with the time complexity of $O(m^3p^3 + m^2p^2q^2)$, where p is the maximum net number of all routing instances and q denotes the maximum subnet number of all nets. After theorectical analysis we perform the algorithm on a variety of objects to verify our design. The experimental results show a 4.31% reduction of net length from the Matching-based algorithm to the MST-based algorithm.

The remainder of this chapter is organized as following. Section 4.2 introduces the MST-Based Graph Algorithm. In that section the algorithm is proved to be optimal for both row-based FPGAs and symmetrical array FPGAs. And the time complexity is also given. To support our theoretical design and analysis we execute the MST-based algorithm on ten sets of routing instances. And the results are shown in Section 4.3. Section 4.4 concludes our work.

4.2 MST-Based Graph Algorithm on FPGA Channel Segmentation Design

In this section we will introduce the MST-Based Graph Algorithm on FPGA channel segmentation design. The channel segmentation design problem is to determine a channel segmentation architecture to achieve "best" routability [10]. "Best" routability means that the segmentation architecture can accomodate as many routing instances as possible [10]. Chang *et al.* solved this problem in the Matching-based algorithm [10] [24] [25]. They considered the similarity of input routing instances and formulated the Net Matching Problem to construct the segmentation architecture. The net matching procedure is effective and efficient. But it cannot get optimal results when there are more than two routing instances because of its dependance on the pairing scheme. The MST-Based Graph Algorithm processes all routing instances at once. No

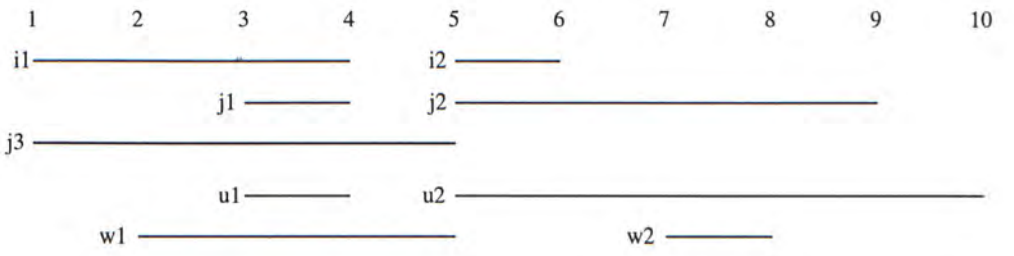


Figure 4.3: Row-Based FPGA Channel with Four Routing Instances

instance merging order affects the results.

In our FPGA channel segmentation design we use a weighted undirected m-partite graph to formulate the Net Merging Problem, where m is the number of routing instances. In the graph the algorithm finds a set of edges whose total weight is maximized. These edges compose of a forest satisfying the following condition,

No two vertices in a tree belong to the same disjoint vertex set.

Each edge tells that the two nets represented by the two vertices can be merged. And the edge weight is the overlapping length of the two nets. Then all the selected edges indicate which pairs of nets can be merged. The larger the total weight of selected edges, the smaller the total length of nets after merging. Firstly the MST-based algorithm is developed for row-based FPGAs. It runs in $O(m^3n^3)$, where m is the number of routing instances and each routing instance contains at most n nets. Then the extended algorithm is developed for symmetrical array FPGAs with the time complexity of $O(m^3p^3 + m^2p^2q^2)$, where p is the maximum net number of all routing instances and q denotes the maximum subnet number of all nets. It has been proven that the MST-Based Graph Algorithm can optimally solve the Net Merging Problem.

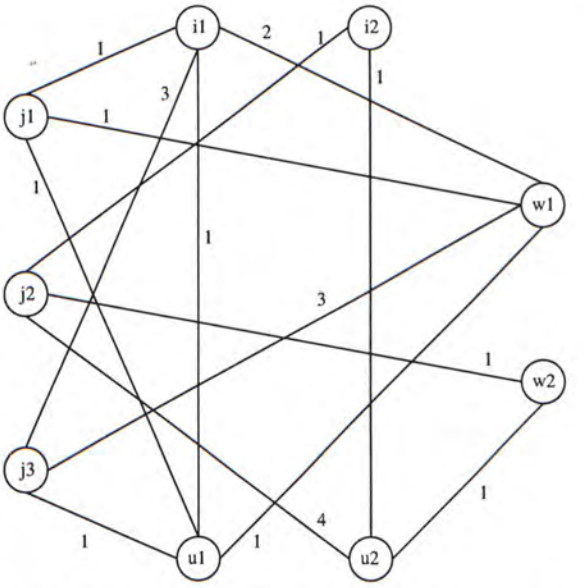


Figure 4.4: Weighted Undirected 4-Partite Graph Representation of Figure 4.3

4.2.1 Net Merging Problem of Row-Based FPGAs

In the MST-Based Graph Algorithm each net is regarded as a *vertex*. The nets in the same routing instance compose of one *disjoint vertex set*. A weighted *edge* exists between two vertices if the two nets represented by them overlap each other and belong to different routing instances. The *weight* is the overlapping length. Figure 4.3 shows a row-based FPGA channel, in which there are four routing instances $\{R_i, R_j, R_u, R_w\}$. And the routing instances have different numbers of nets.

$$R_i = \{i_1, i_2\}$$
$$R_j = \{j_1, j_2, j_3\}$$
$$R_u = \{u_1, u_2\}$$
$$R_w = \{w_1, w_2\}$$

Figure 4.4 is the weighted undirected 4-partite graph to formulate the Net Merging Problem of this example. After constructing the graph the Kruskal's

Algorithm [26] is applied to select a set of edges whose total weight is maximized. The selected edges compose of a forest observing the following condition,

No two vertices in a tree belong to the same disjoint vertex set.

The selection procedure ensures that no two nets in the same routing instance are merged. According to the algorithm the larger the total weight of selected edges, the smaller the total length of nets after merging. That will be proved in the remainder of this section.

Suppose that there are m routing instances and each routing instance contains at most n nets. The number of the nets in the ith routing instance is denoted by n_i. We let R_i and N_j indicate the ith routing instance and the jth net respectively. In the graph we use e_i to represent an edge telling that two nets can be merged. $len()$ is the function computing net length. We have R as the final routing instance after performing the MST-Based Graph Algorithm. Then the following theorems exist.

Theorem 4.1

$$len(R) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} len(R_i, N_j) - \sum_{i=1}^{k} len(e_i) \tag{4.1}$$

where k means that there are k pairs of nets which can be merged.

Proof:

1. When $k = 1$, the theorem is trivially true.

2. Suppose that the theorem is true when there are $k - 1$ pairs of nets to be merged. That is,

$$len(R) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} len(R_i, N_j) - \sum_{i=1}^{k-1} len(e_i) \tag{4.2}$$

3. When there are k pairs of nets that can be merged, we consider the first $k-1$ edges firstly. The current reduced net length should be

$$\sum_{i=1}^{k-1} len(e_i) \tag{4.3}$$

After the kth edge is selected, a new pair of nets is merged and the net length is reduced further by e_k. So the total reduced net length is

$$len(e_k) + \sum_{i=1}^{k-1} len(e_i) = \sum_{i=1}^{k} len(e_i) \tag{4.4}$$

Therefore,

$$len(R) = \sum_{i=1}^{m} \sum_{j=1}^{n_i} len(R_i, N_j) - \sum_{i=1}^{k} len(e_i) \tag{4.5}$$

■

Theorem 4.2 The Net Merging Problem of row-based FPGAs can be solved in $O(m^3n^3)$ by the MST-Based Graph Algorithm.

Theorem 4.3 The MST-Based Graph Algorithm can optimally solve the Net Merging Problem of row-based FPGAs.

Proof: Suppose that the MST-Based Graph Algorithm is not optimal. There is at least one non-selected edge e_1 whose selection will provide a legal solution and $len(R)$ will become smaller.

1. Suppose that the section of e_1 will not cause that other selected edge(s) has (have) to be deleted from the forest.

   Since our algorithm processes edges in the non-increasing order of weights, the weight of e_1 should be equal to or less than the smallest weight of the selected edges. Because the addition of e_1 will not make any other selected edge deleted, according to our algorithm, choosing it will not form circle(s) as well as no two nets in the same routing instance will be merged. So when executing the algorithm, e_1 should be selected, which conflicts with our hypothesis.

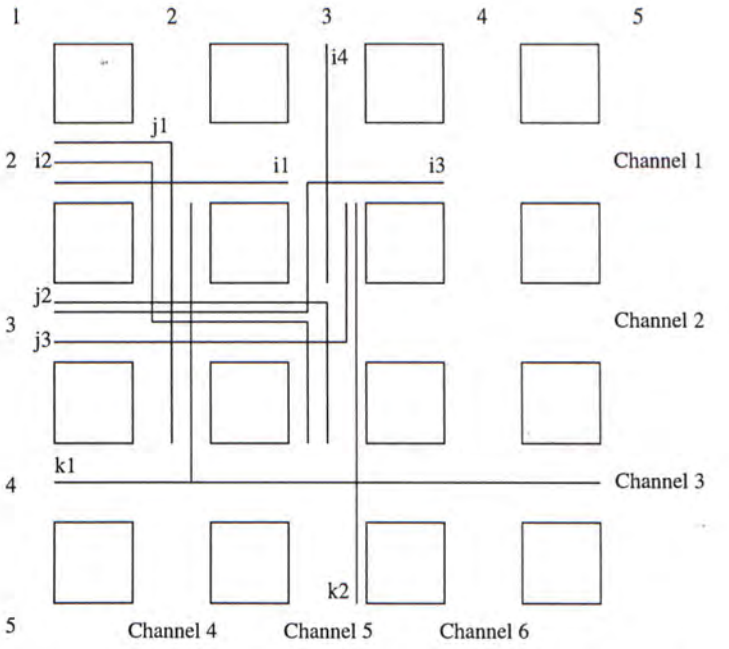


Figure 4.5: Symmetrical Array FPGA with Three Routing Instances

2. Suppose one or more selected edges must be deleted from the forest after the addition of e_1.

   Here the total weight of the deleted edge(s) should be less than that of e_1. That is, the weight of each deleted edge must be less than that of e_1. According to our algorithm the edges are processed in weights' non-increasing order. e_1 should be chosen before considering those edges, which conflicts with our hypothesis.

Based on the above analysis, there will not be such edges as e_1. And the MST-Based Graph Algorithm is optimal.

■

4.2.2 Extended Net Merging Problem of Symmetrical Array FPGAs

The Net Merging Problem of symmetrical array FPGAs is solved in a similar way to that of row-based FPGAs. And the formulation procedure is the

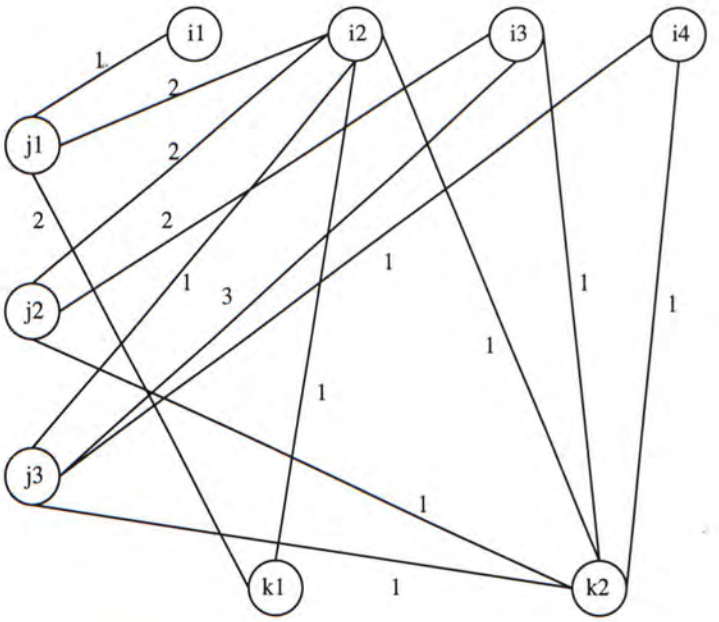


Figure 4.6: Weighted Undirected 3-Partite Graph Representation of Figure 4.5

same as the one introduced in [25]. m still represents the number of routing instances. p is the maximum number of nets among all routing instances. q denotes the maximum number of subnets among all nets. Without additional definitions, all the other notations follow the meanings declared in Section 4.2.1. In the weighted undirected m-partite graph each vertex represents a net. An edge indicates that two nets overlap each other in at least one channel. And the weight is the total length of the overlapping part(s). After constructing the graph the Kruskal's Algorithm [26] is applied to determine which pairs of nets can be merged. Figure 4.5 shows a symmetrical array FPGA example with three routing instances. And Figure 4.6 is its graph representation.

Theorem 4.4 The Net Merging Problem of symmetrical array FPGAs can be solved in $O(m^3p^3 + m^2p^2q^2)$.

Theorem 4.5 The MST-Based Graph Algorithm can optimally solve the Net Merging Problem of symmetrical array FPGAs.

Proof: Here we keep using a weighted m-partite graph to formulate the Net

Merging Problem. The algorithm solves the problem optimally only if the total weight of selected edges is maximized. The selection scheme applied here is able to find a set of edges satisfying the problem specifications with the total weight maximized, which has been proved in Theorem 4.3. Therefore, the algorithm is optimal to solve the Net Merging Problem of symmetrical array FPGAs as well.

4.3 Experimental Results

To support our algorithm design we execute it on different objects on the platform of a 1500 MHz Intel Pentium 4 PC with 512M RAM. We randomly generated ten sets of routing instances from the instance-generating package of our own in C language. Each of the sets contains fifty routing instances ($m = 50$). And each instance contains at most fifty nets ($n = 50, p = 50$). All channel lengths are set to be one hundred. Table 1 shows the comparison results of the Matching-Based Graph Algorithm and the MST-Based Graph Algorithm. We can see that our algorithm gains a 4.31% reduction of net length compared with the results of the Matching-Based Graph Algorithm.

4.4 Summary

We propose the MST-Based Graph Algorithm on FPGA segmentation design. Based on the Graph Matching-Based Algorithm, it is improved to optimally solve the Net Merging Problem of row-based FPGAs in $O(m^3n^3)$ and symmetrical array FPGAs in $O(m^3n^3 + m^2p^2q^2)$. To verify our design we conduct experiments on ten random sets of routing instances generated by our own instance-generated package in C language. By comparing the results a 4.31% net length reduction from the Matching-Based Graph Algorithm to the MST-Based Graph Algorithm is observed. Our work is expected to contribute to

-	*Net Length*		
-	*Before Merging*	*After Merging*	
Set Number	-	*Matching*	*MST*
1	32243	2044	1901
2	30457	2066	2056
3	31796	2086	1997
4	32134	2041	1958
5	28398	1934	1824
6	30819	1967	1863
7	27677	2018	1940
8	28476	2006	1895
9	30727	2126	2042
10	32968	1968	1907
Total Length	305767	20256	19383
Comparison Result	-	0.00%	-4.31%

Table 4.1: Net Length Comparison

the future research on FPGA channel segmentation design.

Chapter 5

Conclusions

FPGA architecture design is a main topic for improving the FPGA function today. There are a variety of methods to improve the FPGA working ability. Since FPGA came into industry nearly twenty years ago, many research works have addressed this area. After studying the literatures on this topic, we develope our work in two different fields.

Based on the Xilinx Virtex Architecture, we develope a CS-Box FPGA architecture, in which the connection box and switch box are combined to the CS-Box structure. Inside the CS-Box we introduce the connection algorithm to build the switches. Since our main aim is to reduce the hardware requirement, we discuss the switch numbers required by the new FPGA and the symmetrical array FPGA. In the theoretical analysis we take the example of $P = 5$ and $\alpha = 1.0$ and find $N_{cs} \leqslant N_x$ when $W \geqslant 3$. The experimental results show a 11.81% reduction on average in the number of switches connecting two wire segments, accompanied by the small penalty of channel widths and circuit delays when $F_{cl} = F'_{cl}$ in CS-Box FPGAs. The larger F_{cl} in the new FPGA, the smaller the reduction of switch numbers. But the penalty of channel widths and circuit delays is eliminated.

The channel segmentation design is another topic in FPGA architecture design. We studied a lot of research works. The authors proposed a variety of methods to solve the segmentation construction problem. Lin *et al.*'s Graph

Matching-Based Algorithm addressed a new direction in this area. After investigating the related works we find that their algorithm cannot provide optimal solutions. Therefore, we design the MST-Based Graph Algorithm. Basically we enhance the formulation method of the Net Merging Problem. In theory it is proved to be optimal for both row-based FPGAs and symmetrical array FPGAs. The time complexities are $O(m^3n^3)$ and $O(m^3p^3+m^2p^2q^2)$. To verify our design we conduct experiments on ten random sets of routing instances. By comparing the results a 4.31% reduction in net length by our algorithm is observed.

Our research work deals with different topics in FPGA architecture design. And they are expected to be improved further. We hope that our work can contribute to the future research on the FPGA study field.

Bibliography

[1] S. Wilton, *Architecture and Algorithms for Field-Programmable Gate Arrays with Embedded Memory*, PhD dissertation, University of Toronto, 1997.

[2] S. D. Brown, R. J. Francis, and Z. G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, Boston, 1992.

[3] Y.-W. Chang, D. Wong, and C. Wong, Universal Switch Modules for FPGA Design, ACM Transactions on Design Automation of Electronic Systems **1**, 80 (1996).

[4] H. Fan, J. Liu, and Y.-L. Wu, Combinatorial Routing Analysis and Design of Universal Switch Blocks, in *Proc. Conference on Asia South Pacific Design Automation*, pages 641–644, Yokohama, Japan, 2001, IEEE.

[5] M. Shyu, G.-M. Wu, Y.-D. Chang, and Y.-W. Chang, Generic Universal Switch Blocks, IEEE Transactions on Computers **49**, 348 (2000).

[6] H. Fan, J. Liu, and Y.-L. Wu, General Models and a Reduction Design Technique for FPGA Switch Box Designs, IEEE Transactions on Computers **52**, 21 (2003).

[7] H. Fan, J. Liu, and W.-L. Wu, On Optimum Switch Box Designs for 2-D FPGAs, in *Proc. IEEE/ACM Design Automation Conference*, pages 203–208, Las Vegas, Nevada, 2001.

[8] H. Fan, J. Liu, and Y.-L. Wu, On Optimal Hyper Universal and Rearrangeable Switch Box Designs, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **22**, 1637 (2004).

[9] V. Betz and J. Rose, FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density, in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 59–68, Monterey, CA, 1999.

[10] Y.-W. Chang, J.-M. Lin, and M. Wong, Matching-Based Algorithm for FPGA Channel Segmentation Design, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **20**, 784 (2001).

[11] W.-K. Mak and D. Wong, Channel Segmentation Design for Symmetric FPGAs, in *IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 496–501, Austin, TX, 1997.

[12] M. Pedram, B. S. Nobandegani, and B. T. Preas, Design and Analysis of Segmented Routing Channels for Row-Based FPGAs, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **13**, 1470 (1994).

[13] K. Roy and M. Mehendale, Optimization of Channel Segmentation for Channelled Architecture FPGAs, in *IEEE Custom Integrated Circuits Conference*, pages 4.4.1–4.4.4, Boston, 1992.

[14] K. Zhu and D. Wong, On Channel Segmentation Design for Row-Based FPGAs, in *IEEE/ACM International Conference on Computer-Aided Design*, pages 26–29, California, US, 1992.

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

[16] J. Rose and S. Brown, Flexibility of Interconnection Structures for Field-Programmable Gate Arrays, IEEE Journal of Solid-State Circuits **26**, 277 (1991).

[17] T. Okamoto and J. Cong, Buffered Steiner Tree Construction with Wire Sizing for Interconnect Layout Optimization, in *IEEE/ACM International Conference on Computer-Aided Design*, pages 44–49, San Jose, CA, USA, 1997.

[18] C. Ebeling, L. McMurchie, S. Hauck, and S. Burns, Placement and Routing Tools for the Triptych FPGA, IEEE Transactions on VLSI Systems **3**, 473 (1995).

[19] V. Betz and J. Rose, Directional Bias and Non-Uniformity in FPGA Global Routing Architectures, in *International Conference on Computer Aided Design*, pages 652–659, San Jose, California, US, 1996.

[20] P. Chan and M. Schlag, Acceleration of an FPGA Router, in *IEEE Symposium on FPGA-Based Custom Computing Machines*, pages 175–181, Napa Valley, CA, 1997.

[21] V. Betz and J. Rose, VPR: A New Packing, Placement and Routing Tool for FPGA Research, in *Proc. 7th International Workshop on Field-Programmable Logic and Applications*, pages 213–222, London, UK, 1997.

[22] B. Fallah and J. Rose, Timing-Driven Routing Segment Assignment in FPGAs, in *Canadian Conference on VLSI*, pages 124–130, Halifax, NS, Canada, 1992.

[23] M. Khellah, S. Brown, and Z. Vranesic, Modelling Routing Delays in SRAM-based FPGAs, in *Canadian Conference on VLSI*, pages 13–18, Banff, Alberta, Canada, 1993.

[24] Y.-W. Chang, J.-M. Lin, and D. Wong, Graph Matching-Based Algorithms for FPGA Segmentation Design, in *IEEE/ACM International Conference on Computer-Aided Design*, pages 34–39, San Jose, California, 1998.

[25] J.-M. Lin, S.-R. Pan, and Y.-W. Chang, Graph Matching-Based Algorithms for Array-Based FPGA Segmentation Design and Routing, in *Design Automation Conference, Asia and South Pacific*, pages 851–854, Kitakyushu, Japan, 2003.

[26] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, Mass., 1990.

CUHK Libraries
004144564