Logic Perturbation Based Circuit Partitioning and Optimum FPGA Switch-Box Designs

CHEUNG Chak Chung

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

Abstract

In this thesis, a framework which integrates an efficient Graph-Based Alternative Wiring (GBAW) technique and state-of-the-art Multi-level hypergraph partitioner (hMETIS-Kway) for multi-way circuit partitioning is proposed. Moreover, the routability of Hyper-Universal Switch Box is examined by reconfiguring the switch-box architecture of well-known Field-Programmable Gate Arrays (FPGAs) router VPR.

Efficient circuit partitioning is gaining more and more importance with the increasing size of modern circuits. Conventionally, circuit partitioning is solved without altering the circuit by modeling a circuit as a hypergraph for the ease of applying graph algorithms. However, there exists rooms for further improvement on even optimum hypergraph partitioning results, if logic information can be applied for circuit perturbation. Such logic transformation based partitioning techniques are usually more complicated and harder to formulate. A multi-way partitioning framework which can couple any hypergraph partitioner is presented. This framework gives a novel logic perturbation technique for further improvement over very excellent partitioning results. This approach can integrate with any graph partitioner. Experiments on 2-, 3-, 4-, and 5-way partitionings for various circuits of different sizes from MCNC benchmarks were performed. The experimental results showed that this partitioning approach can achieve a further 15% reduction in cut size for 2-way partitioning with an area penalty of only 0.33% over existing state-of-the-art graph partitioner.

An FPGA switch box is said to be universal (hyper-universal) if detailed routing can be performed on all possible surrounding 2-pin (multi-pin) net topologies which satisfy the global routing density constraints. A switch box is optimum if it is hyper-universal and the switches inside is minimum. It has been shown that if the net topology is restricted to 2-pin nets, then a 2-D (4-way) switch box can be built to be universal with only $6W$ switches, where W is the global routing channel density. A previous work has constructed a formal mathematical model of this optimum design problem for switch boxes with arbitrary dimensions, and given a scheme to produce hyper-universal designs with less than 6.7W switches for 4-way FPGA switch boxes. The investigation of the most common 4-way switch box case is presented, and gives new theoretical results followed by extensive experimental justification. It is shown that such an optimum switch box can be built with a very low number of additional switches beyond 6W for today's practical range of low W (e.g. just $6W$ plus 1 or 2 additional switches for W up to 7). Even for arbitrary large W, the bound can be shown to be under $6.34W$. To make experimental comparison, experiments are conducted by running today's published best FPGA router VPR on large benchmarks for the popular disjoint structure and the proposed designs. The results are quite encouraging.

摘要

作者：張澤松

本論文主要對快速「基於圖像可替換線檢驗算法」和著名多層超圖分割規劃法所結合建成的結構，以及對新提出的「超全體開關盒」在著名的現場可編程門陣列佈線程序器的可佈線性作出研究。

由於有電路的數目大大增加，所以分割規劃需要更有效率的算法。首先，電路會被造型成超圖，然後利用圖算法去解決電路分割規劃問題，而且當中沒有對電路作出修改。如果把電路的邏輯資料作出電路擾亂，即使是最佳的超圖分割規劃結果仍然可獲得改進。這種基於邏輯轉換分割規劃法的技術通常是比較複雜，而且很困難用公式表示出來。本論文介紹一個可以和任何超圖分割規劃法結合的多路分割規劃法結構。這結構提供一個新穎的電路擾亂技術並從極好的分割規劃結果中獲得改進。這方法更可以和其他圖分割規劃器結成一體。二路、三路、四路和五路的超圖分割規劃實驗在不同大小的基準電路上，把實驗結果和著名的圖分割規劃法在二路分割結果上的比較，這分割規劃方法可以取得百分之十五的分割成本減少，但只是增大了百分之零點三三的面積。實驗結果證明了這多層超圖分割規劃法和圖像可替換線檢驗算法的共同合作效力。

在現場可編程門陣列中，全體「超全體」的開關盒是指能夠令所有兩端腳「多端腳」線在乎合全局佈線密度限制的情況下，都能詳細地佈線出來。一個最優良的開關盒必須是「超全體的」以及使用的切換線數目爲最少。當網絡拓樸被局限於兩端腳時，只需要用六倍全局佈線通道密度(簡稱「密度」)的切換線便可建成一個兩維「四向」的超全體的開關盒。一個早前的研究已經把數學模型去解決超全體開關盒在任何維數的最優化問題，而且，還有提出了一個少於六點七倍密度的切換線的超全體開關盒設計在四向的現場可編程門陣列的開關盒上。本論文介紹了最普遍的四向開關盒的研究和新理論性的結果及其廣泛的實驗證明。而且一個最佳的開關盒只需要很少數目的附加切換線在現今實際最少的六倍密度上(例如:在佈線路線密度等於七的時候，只需要在六倍密度上加一至兩條附加切換線)。甚至在大的佈線路線密度上，切換線的上限只低於六點三四倍的密度。爲了作出比較，選擇現今發表了最佳的現場可編程門陣列的佈線器，這個新設計及解體的開關盒分別在大型的基準電路上作出實驗和比較，而且結果令人鼓舞。

Acknowledgments

First of all, I would like to express my greatest gratefulness to my supervisor, Professor David Yu-liang Wu for his invaluable suggestions and comments on different research topics. His support and guidance in the past two years helped me to work in the right direction in academic area and social issues. I am greatly impressed by his working style which work hard and never give up. I learnt "*nothing is impossible to achieve*". I would also like to thank for his support and arrangement for the study in Chip Implementation Center (CIC), Taiwan 2000 summer. Moreover, I acquired lot of experiences and gained interests in Computer-Aided Design automation through our discussions.

I greatly appreciate the members of my dissertation committee, Professor Philip Heng-wai Leong, Professor Kin-hong Wong and Professor Hong-bing Fan for their useful feedback and suggestions. I would like to specially thank my final year supervisor, Professor Philip Heng-wai Leong for introducing me to Professor David Yu-liang Wu and teaching me in hardware design. I am greatly indebted to Dr. David Ihsin Cheng, Professor Hong-bing Fan and Professor Jiping Liu for the collaboration works with them. I would also like to thank Professor Chak-kuen Wong and Professor Evangeline Fung-yu Young for their advice and courses which enriched my background knowledge.

I would also like to thank my research partner Mr. Cliff Chin-ngai Sze for his encouragement and collaboration work on the development of GBAW. I would like to show my appreciation to Dr. Xao-long Yuan for his assistance in my research work.

Many thanks go to Professor George Karypis of University of Minnesota and the authors of METIS and Professor Jonathan Rose of University of Toronto and the authors of VPR for their prominent works on Partitioning and FPGA Routing.

Thanks also go to my marvelous friends in Room 1026 who helped me to solve many problems and made my life more enjoyable and unforgettable. Especially, I would express my appreciation to Mr. Ivan Ka-ho Leung, Mr. Hiu-yung Wong, Miss Wing-seung Yuen, Mr. Norris Monk-ping Leong and Miss Polly Po-man Wan for their frank warm-hearted help. I would like to thank for all the friends who help me and make my life more colorful.

My unplumbed gratitude goes to my parents, sister and brother-in-law for their endless love and encouragement and my aunt Kit-chi Wong for her support and understanding. Last but not least, I would express my love to my best friend, Miss Victoria Chor-kiu Lee for her sacrifices and patience. They all brought untold happiness to my life.

VITA

May 23, 1977 Born, Fujian, China

Education

B.Eng. in Computer Engineering, The Chinese University of Hong Kong

Publications

P.K. Tsang, **C.C. Cheung**, K.H. Leung, T.K. Lee and and P.H.W. Leong, "An Asynchronous Forth Microprocessor", Proceedings of the IEEE Region 10 Conference (TENCON), Vol. 2, pp. 1079-1082, 1999.

Y.L. Wu, C.N. Sze, **C.C. Cheung**, "On Improved Graph-Based Alternative Wiring Scheme for Multi-Level Logic Optimization", IEEE International Conference of Electronics Circuits and Systems (ICECS), 2000.

C.C. Cheung, Y.L. Wu, and D. Ihsin Cheng, "Further Improve Circuit Partitioning using GBAW Logic Perturbation Techniques" Proceedings of IEEE Design Automation and Test in Europe (DATE), pp. 233-239, March. 2001. Munich.

H.B. Fan, J. Liu, Y.L. Wu, and **C.C. Cheung**, "On Optimum Switch Box Designs for 2-D FPGAs" to appear in Proceedings of IEEE/ACM Design Automation Conference (DAC). June. 2001. Las Vegas.

Y.L. Yu, **C.C. Cheung**, D. Ihsin Cheng and H.B. Fan, "Further Improve Circuit Partitioning using GBAW Logic Perturbation Techniques" under review of IEEE Transactions on Very Large Scale Integration Systems (TVLSI).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Integrated Circuit (IC) technology has evolved for many times from Small Scale Integration (SSI) to Very Large Scale Integration (VLSI). ICs are widely used in computers, electronic devices and even implantation of human body. According to Moore's law, the capacity of a chip is doubled for every 18-24 months of the previous chip as shown in Figure 1.1. Due to the huge number of components of circuit design, it is not practical for designer to repeat the fabrication process. As a result, Computer-Aided Design (CAD) tools [SW96] are developed for physical design which is one of the critical step in VLSI design cycle. The goal of physical design is to fit the design into a compact area for fabrication. Physical design is a very complex process and thus it is broken into many different stages from partitioning to routing. The netlist extraction by layout synthesis, design verification and validation check are also performed on the layout during physical design stage. This thesis addresses the two important steps of physical design, partitioning and routing especially for Field-Programmable Gate Arrays (FPGAs).

In the era of deep sub-micron, many problems arise and designers have to tackle them such as the domination of interconnect delay, failure to meet timing requirement and insufficient routability on different devices. By considering

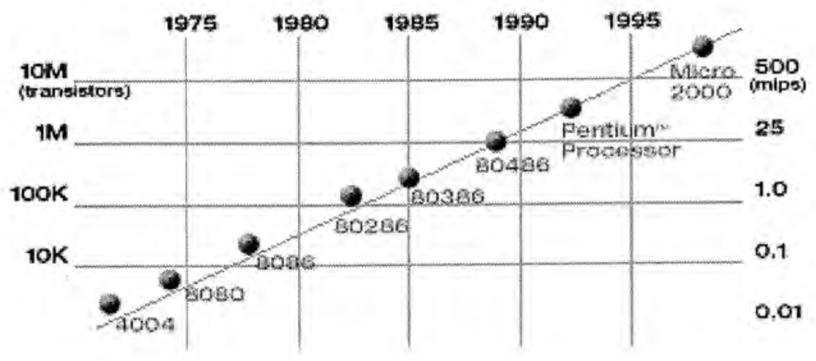


Figure 1.1: Moore's Law vs Industrial CPU size (from Intel)

more circuit information in the physical design stage, we can improve the overall performance, get a better utilization of chip area, better yield and reduce the cost. The idea behind is to apply *Alternative Wiring* techniques [EC95] in order to remove some wires along the critical path, or replace some wires in congested area with those in uncongested area. Alternative wire allows adding and removing wires inside the circuit without changing the circuit functionality. Thus, it can explore a large degree of freedom in many different physical design problems, such as logic optimization [EC95], circuit partitioning [CLMS95] and timing optimization [EEOU96].

The idea of VLSI circuit partitioning is to divide a circuit into smaller subcircuits (clusters). The objective is to minimize the interconnection between clusters with balance constraints. Due to the increase of the VLSI design complexity, circuit partitioning is getting more and more crucial. The system may consist of several hundred million transistors and most of the algorithms in physical design cycle and logic synthesis can not handle such large problem size or become ineffective. As a result, circuit partitioning becomes the first step in physical design cycle. A large system becomes smaller manageable components which can be solved in a top-down hierarchical design methodology. The feasibilities of placement, floorplanning, global routing and detailed routing also depend on the quality of the partitioning solution.

Nowadays, in deep sub-micron VLSI design, the interconnection delays dominate the gate delays. A larger die size means a longer routing delay on chip. An effective partitioning algorithm becomes the fundermental key to improve system performance. There are many different approaches to tackle this partitioning problem including move-based approaches, geometric representations, combinatorial formulations and clustering approaches [AK95b]. Recently, a new trend is to integrate existing partitioning algorithms with multi-level approach [ZSC96, KAKS97]. Most of the algorithms model the circuit as graph (hypergraph) and seek for optimal solution. However, these methods do not consider the logic function of the circuit, that is the function performed by each node.

Alternative wiring is a technique to transform a circuit into another but functionally equivalent circuit. It can be well-used for post-layout logic synthesis, logic optimization and other CAD problems. Graph Based Alternative Wiring (GBAW) is newly introduced and it is very efficient in identifying alternative wires. This motivates the integration of GBAW technique for circuit partitioning.

Nowadays, logic designers have several good logic devices to realize the digital logic functions. These devices include Small-Scale Integration (SSI) chips such as 7400 series, Application-Specific Integrated Circuits (ASICs) and Programmable Logic Devices (PLDs) such as Simple PLDs (SPLDs), Complex PLDs (CPLDs) and Field-Programmable Gate Arrays (FPGAs) [BFRV92]. With the huge logic resources, short turn-around time and user-programmability, FPGAs are easy for logic designer to modify their sophisticated design and reduce the manufacturing time. After the first emergence into market, FPGAs have been continuously growing and in the leading place amongst other logic devices. A typical FPGA composes of three major components: logic modules, routing devices and Input/Output (I/O) devices. Routing resources compose of pre-fabricated wire segments and programmable switches (switch-box).

Switch modules are the most important components of the routing resources in FPGAs. Programmable switches usually have high resistance and capacitance, and they consume large amount of area [ea88, ea89]. Thus an excessive number of switches means a slower and larger design. Higher routability means a smaller number of tracks to complete the design routing and reduces the size of FPGA. As a result, it is desirable to investigate the routability of switch-box for a better area performance design.

The rewiring technique can also be applied to layout-driven logic synthesis. GBAW technique is able to locate alternative wires such that the congested wire can be removed, thus the whole design can become routable within the smaller number of tracks.

1.2 Aims and Contribution

There are two main parts in this thesis: circuit partitioning by using logic perturbation technique and FPGA routing employing Hyper Universal Switch Box (HUSB).

Firstly, the identifying alternative wiring power of the GBAW is further improved by introducing new rewiring patterns. The comparison of logic optimizations between famous ATPG-based RAMBO and augmented GBAW is also presented. Based on new rewiring technique, logic optimizations on MCNC benchmark circuits are successfully conducted and the integration of GBAW with the well-known multi-level graph partitioner (hMETIS-kway) is also done. The results showed that it is possible to obtain an improved partitioning result from near optimal solution.

Secondly, the construction and mathematical model of Hyper Universal Switch Box (HUSB) are given. Further investigation of the quality of HUSB is done by giving new theoretical results and adopting efficient FPGA router VPR which is used to generate experimental justification. An encouraging result over the comparison between the popular disjoint structure (XC4000) and the proposed designs is obtained.

Our proposed design flow for FPGAs has two stages. In the first stage, the interal switch-box architecture is modified in order to achieve high routability. In the second stage, the rewiring engine GBAW will be integrated into the whole FPGA design flow which is not yet completed.

1.3 Thesis Overview

This thesis is organized as follows. Chapter 2 presents the background of physical design. Chapter 3 introduces the alternative wiring engine (GBAW) and shows the comparison between RAMBO and GBAW on logic optimization. Chapter 4 discusses the new approach to further improve the partitioning solution by hMETIS-kway. Chapter 5 addresses the quality of FPGA routing result by VPR with enhanced HUSB (H'USB). Finally, Chapter 6 concludes this thesis and discusses the future works.

Chapter 2

VLSI Design Cycle

A simple VLSI design cycle starts from system specification of a VLSI chip and ends as a tested and packaged chip. The design flow involves architectural, functional, logic, circuit design and the last step before fabrication is physical design which collects the net-list information of a circuit into layout. At the beginning of the VLSI design process, the user can get the specifications for the size, speed, power and the functionality of the VLSI system [She98, Mic94]. In the next step, the VLSI designers are required to design a system which meet all these constraints. Nowadays, the Hardware Design Language (HDL) such as VHDL and Verilog are commonly used in VLSI design. HDL language are used to express the circuit and timing information for simulation and verification. Since the design scale is large and it is impossible to design by hand, the Computer-Aided Design (CAD) tools become crucial and widely used for a fast and correct design. The typical design flow is shown in Figure 2.1. Logic synthesis is a step to translate the register transfer level (RTL) of a digital system to the gate net-list information. In the other words, logic synthesis is applied to the extraction from RTL language and deals with the logic optimization, testability and verification. In physical design step, there are many CAD tools developed for automatic layout, design rule check and extraction [She98]. A tested VLSI chip will be available after all these complex stages.

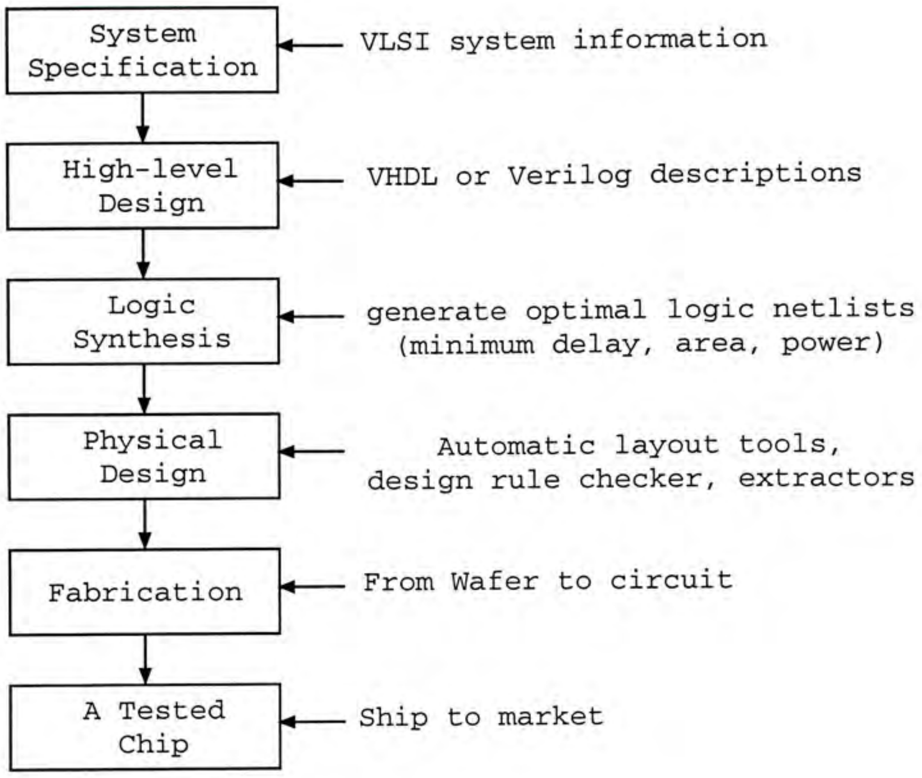


Figure 2.1: The Path to Silicon Chip

2.1 Logic Synthesis

Logic synthesis enables the mapping from RTL level to an optimized gate-level description. The generated description must fit the functional specification and user constraints. It basically begins with logic optimization and provides an efficient implementation to a high quality silicon chip. With the aid of CAD tools, the circuit designers are allowed to work at higher levels of abstraction and easily modify their design. Logic synthesis tools such as "Synopsys" not only reduce the logic but also perform timing analysis, technology mapping and ensure the testability of the generated net-list. In the process of logic synthesis, it aims at:

- minimizing the layout chip area.
- minimizing the critical path delay.
- maximizing the testability such that all test vector can be found.

2.1.1 Logic Minimization

Logic synthesis are usually divided into two-level synthesis (PLA) and Multi-level synthesis [BHSV90]. Therefore, many optimization methods are developed and can solve the problem in reasonable computation time. The algorithms include the minimization of don't cares [BBH+88], global flow [BT91], OBDD representation [Bry86] and rewiring techniques [CE93, WLF00] which change the original circuit net-list information.

2.1.2 Technology Mapping

Technology mapping [Keu87] is the process to translate the abstract description such as HDL languages into a hardware description by using a specific technology. The result is a mapped circuit. Therefore, one description language can be mapped into different architectures by varying the technology mapping library.

2.1.3 Testability

The circuit which cannot provides a complete single stuck-fault test vector is called an untestable circuit. Testability is an important criteria for circuit design, otherwise the fabricated chip cannot locate the fault wire.

2.2 Physical Design Synthesis

Physical design is also called as layout phase which is the process to determine the exact location for the logic devices and the interconnection between different devices. Physical design is a complex process and it can be broken down into several steps which are shown in Figure 2.2. Many research groups focus on each of the sub-steps in order to obtain high performance design [She98].

The cost of fabrication is very high, therefore the objective for physical design is to produce a layout with small area, few defects and high yield.

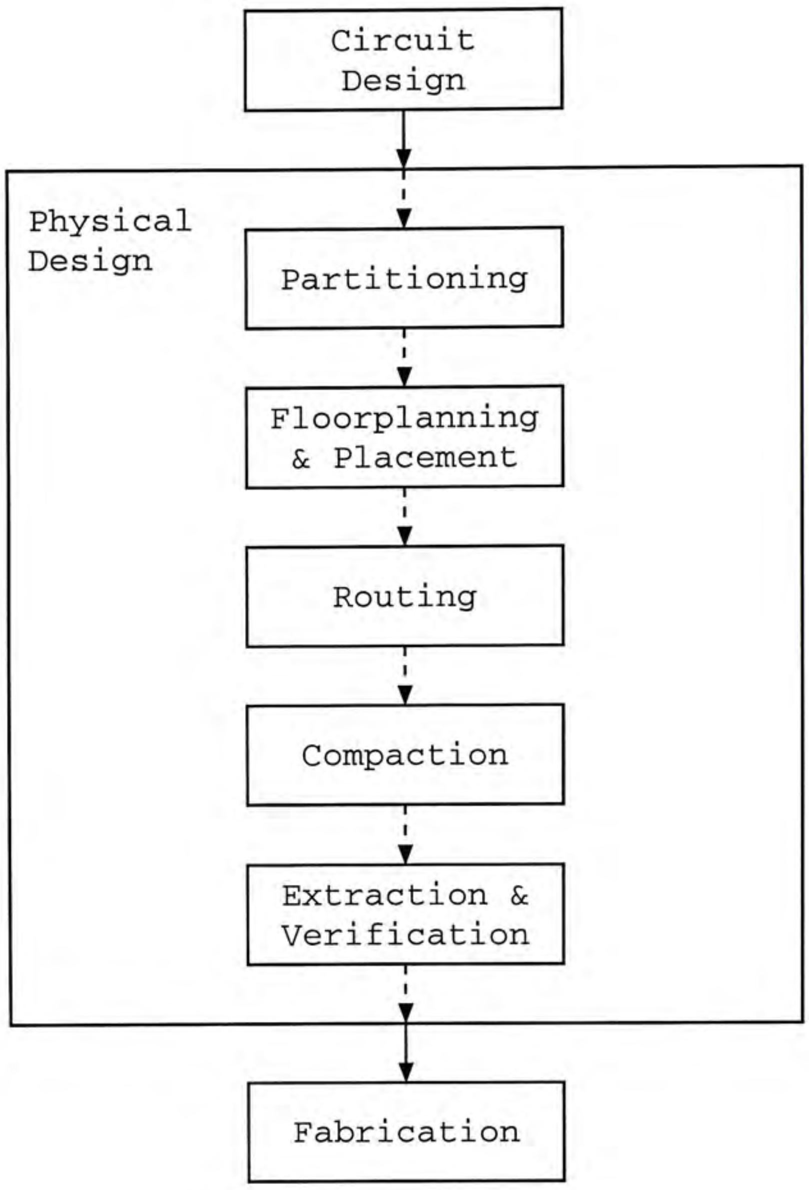


Figure 2.2: VLSI Physical Design Cycle

2.2.1 Partitioning

Partitioning is the first step in physical design owing to the increase of problem size. Many of the algorithms in the latter steps become infeasible, thus partitioning breaks the original problem into small sub-problems and solves eventually. The output of partitioning is a set of clusters which contains a set of logic gates, and the interconnections between clusters. In Figure 2.3, it shows a good and a bad bi-partitioning. The partitioning result for the good one is reduced from 9 to 1.

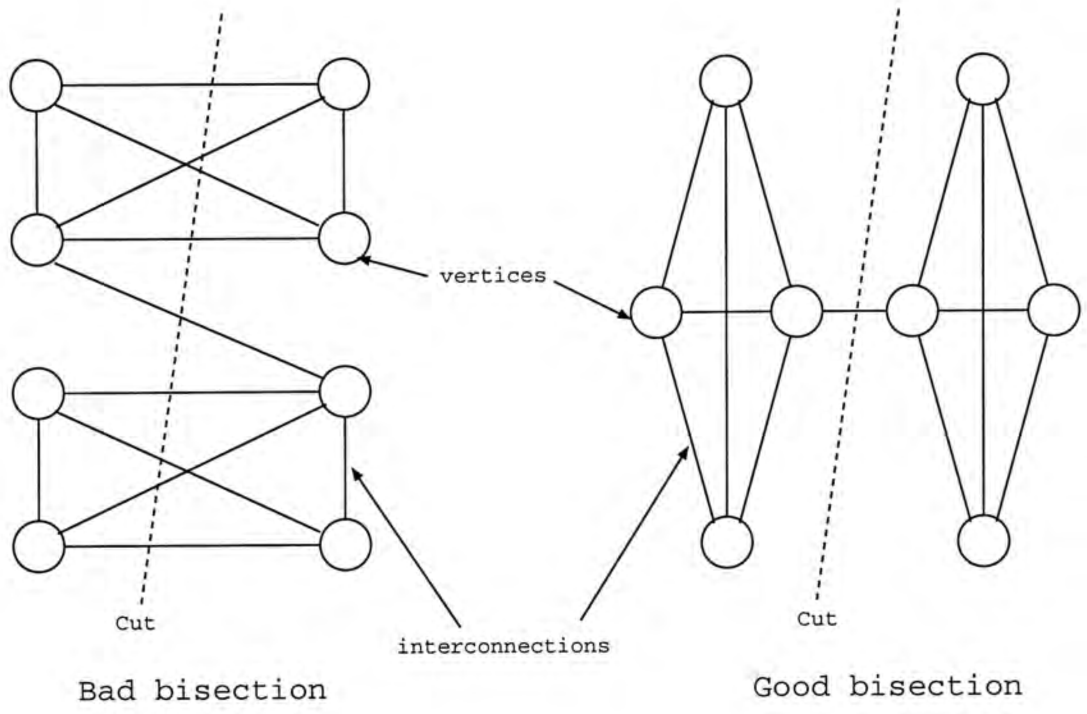


Figure 2.3: Example of Bi-partitioning

2.2.2 Floorplanning & Placement

The objective of floorplanning is to arrange the components of an IC in order to have a compact layout. The two floorplans show in Figure 2.4 are slicing and non-slicing floorplan. Slicing floorplan can be recursively cut horizontally or vertically but non-slicing cannot. In the placement stage, the block to be placed are layouted and have pin assignment. After placement, the IC will have minimum area such that the interconnections between modules can be completely routed. Since a packed placement solution does not guarantee a routable design, the placement step is usually taken again after routing.

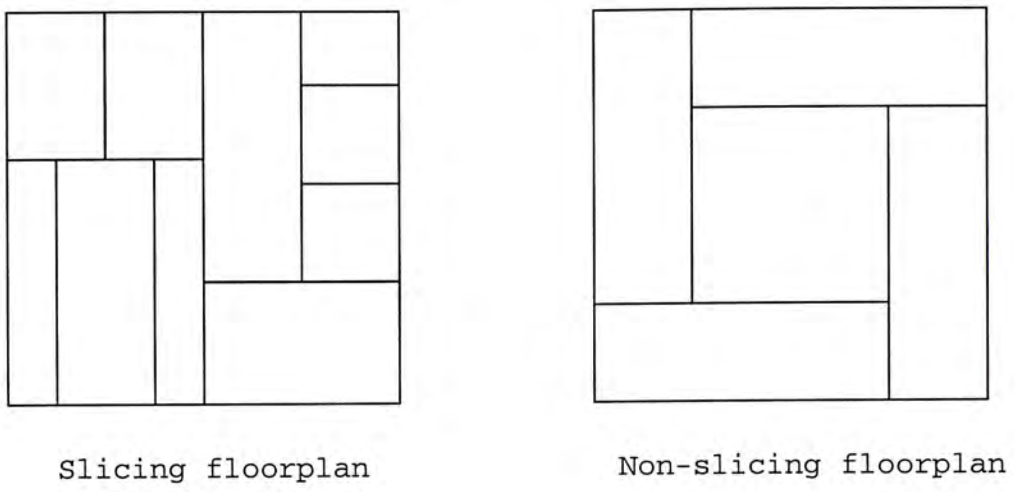


Figure 2.4: Example of Floorplanning

2.2.3 Routing

In order to connect the wires between blocks, routing step is usually done in two steps as the *global routing* and *detailed routing*. Global routing will specify the region used for detailed routing and detailed routing will determine the exact location for each connection. It aims at shorter wire length and less resources used. Figure 2.5 shows a failure channel routing example.

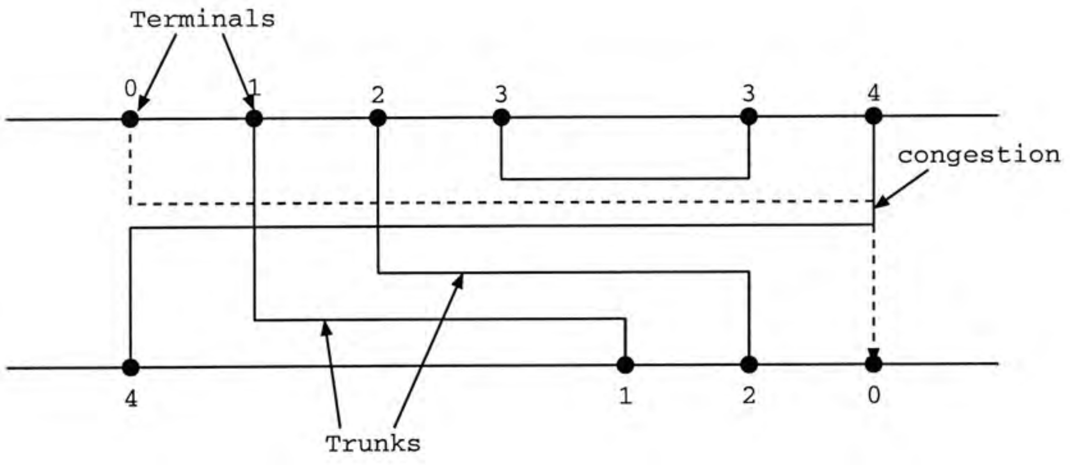


Figure 2.5: Example of Channel Routing

2.2.4 Compaction, Extraction & Verification

Compaction is a step to reduce the chip's total area and signal delay between modules of the circuit. Example of x-y coordinate compaction is shown in Figure 2.6. The circuit extraction step [BB91] consists of determining the circuit connectivity and calculating various electrical parameters such as resistances and capacitances. The last step before fabrication is to verify the design passing all the design rule.

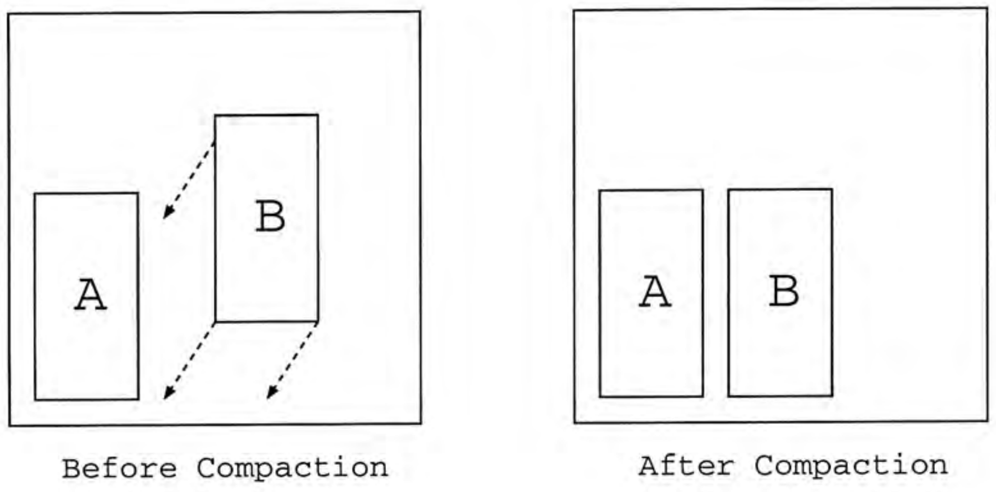


Figure 2.6: Example of Compaction

2.2.5 Physical Design of FPGAs

The time-to-market for even ASIC chip is unacceptable for many applications, therefore the short turn-around time of FPGAs takes advantage and gain more popularity. Since the most of the physical design algorithms are inapplicable in FPGAs, the CAD tools development of FPGAs are separated from others. For example, the traditional channel routing algorithm [SW96] cannot be applied to FPGAs. It mainly consists of three areas: partitioning, placement and routing.

Chapter 3

Alternative Wiring

3.1 Introduction

Alternative wiring technique has a wide range of applications such as circuit optimization [CE93, EC93, EC95, CMSC96, CvGLMS99], partitioning [CLMS95] and FPGA synthesis [CCWMS94b, CCWM97]. Alternative wiring refers to the reconstruction of circuit while keeping its functionality intact by addition and removal of wires. A wire in a circuit is redundant if its addition and removal does not change the functionality of the circuit. The concept of alternative wiring is to add a redundant wire to the circuit that in turn will make another wire become redundant and its removal would lead to some useful circuit transformation for certain objectives. Due to the wide range of application, many research groups are working on the efficiency of locating alternatives wiring. The traditional rewiring tools apply Automatic Test Pattern Generation (ATPG), Implication analysis [KSM97] and Recursive learning [KP94] techniques in order to search alternative wires. However, these kinds of techniques consume too much CPU expenditure, and another Graph-Based Alternative Wiring technique is proposed to replace the ATPG approaches.

Consider the circuit given in Figure 3.1(a). This circuit is irredundant. By adding a connection from the output of gate g_5 to the input of gate g_9 (shown as a dotted line in Figure 3.1(b)), the functionality of the circuit does not change. In other words, the added connection is redundant. However, the addition of the connection causes two originally irredundant wires becoming redundant as shown in Figure 3.1(b). After removing these two wires and associated gates that either become floating g_6 or have a single fanin g_4 and g_7, the circuit can be greatly optimized as shown in Figure 3.1(c).

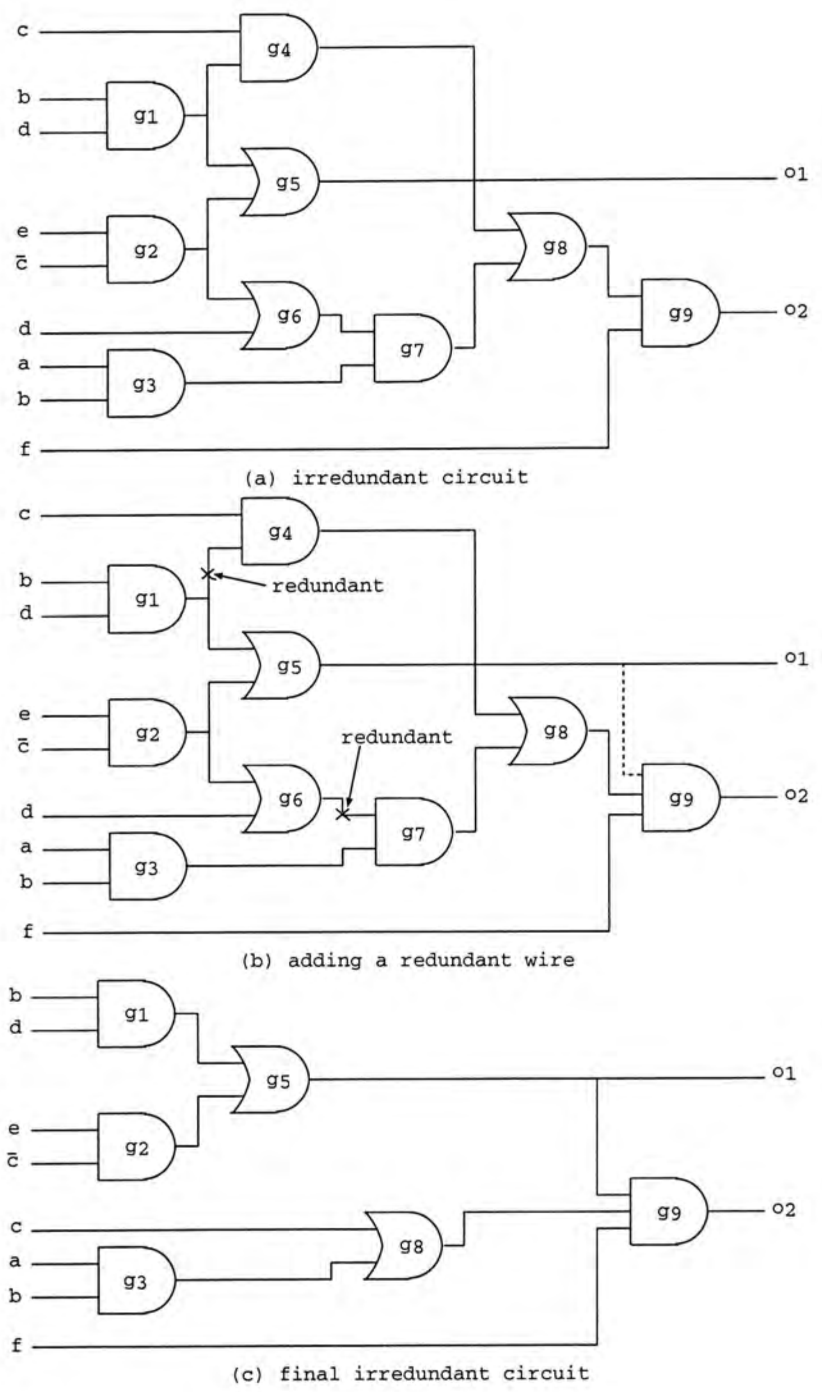


Figure 3.1: Example of alternative wiring

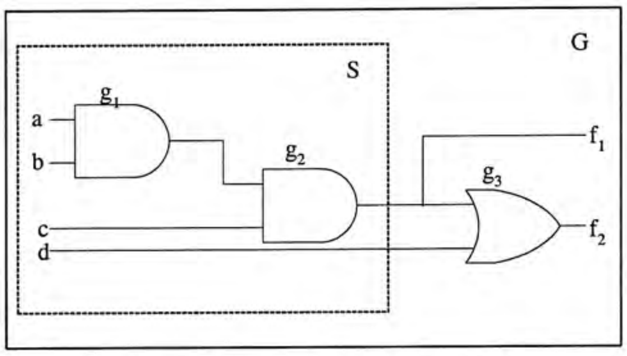
This chapter is organized as follows. The notation and definitions are introduced in Section 3.1. The application of rewiring is presented in Section 3.2. In Section 3.3, logic optimization analysis is introduced. The details of Augmented Graph-Based Alternative Wiring are stated in Section 3.4. Followed by the discussion on Logic Optimization by using GBAW and experimental results are presented in Section 3.5. Conclusion is drawn in Section 3.6.

3.2 Notation and Definitions

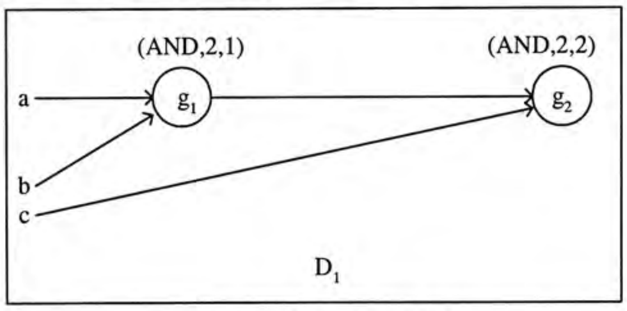
A combinational circuit can be represented by a Directed Acyclic Graph (DAG) where vertices correspond to the primary inputs (PI), primary outputs (PO) and the internal gates of the circuit. PI and PO are nodes which have only outgoing edges and incoming edges respectively. An internal node has at least two incoming edges and one outgoing edge and is associated with a Boolean function. Inverters are not considered as internal nodes, but as polarity of edges during logic domain perturbation. In a Boolean network, the in-degree of node y, denoted by $d^-(y)$, is defined as the number of edges entering y. The out-degree of node y, denoted by $d^+(y)$, is defined as the number of edges leaving y. A node y is defined by a triplet $(op, d^-(y), d^+(y))$, where op is the Boolean operator of y which can be any associative operator like AND, OR, NAND, or NOR.

A wire is replaceable if it has at least one alternative wire. A graph configuration D is used to map the logic function from a Boolean Network G. For each node n_i in sub-network S in network G, n_i is mapped to a triplet (op, i_1, i_2) in D where op denotes the operator representing the Boolean function of n_i and i_1, i_2 are non-negative integers. All edges inside S are preserved, while the edges outside S are omitted in D. In most cases, i_1 equals $d^-(n_i)$ and i_2 equals $d^+(n_i)$. The element of a triplet $(op, d^-(y), d^+(y))$ can also be don't care. For the first element, don't care means any operator. For the other elements, don't care can be any positive integers. A configuration is used to denote a minimal pattern containing both the target and its alternative wire.

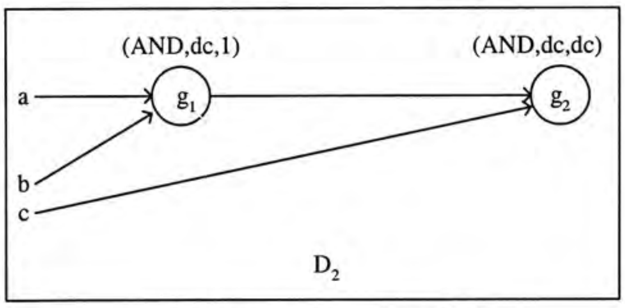
The mapping is illustrated in Figure 3.2. S is a sub-network of G. D_1 and D_2 are two mappable configurations of S. A k-local pattern denotes a minimal sub-graph with the distance between the alternative wire and its target wire being k. The distance between two wires is defined as the difference of maximum path length from any primary input to each of the wires.



(a) Boolean network *G* and its sub-networks



(b) A configuration of *S*, *D_1*



(c) Another configuration of *S*, *D_2*

Figure 3.2: Configuration of a sub-network

A wire is defined as a 2-point connection between a pair of source and sink nodes. When a larger circuit is partitioned into two sub-circuits, we define the wires crossing the partitioning cut line as cut wires. a cut net is also defined as a hyperedge connecting partitions and the cut cost as the number of partitions that the hyperedge connects.

3.3 Application of Rewiring

Alternative wiring greatly increases the flexibility of a circuit in which the wires of the circuit can be altered without changing the circuit functionality. Thus, the circuits before and after rewiring are said to be functionally equivalent. As a result, alternative wiring provides a wide range of applications in different steps of physical design automation.

3.3.1 Logic Optimization

As shown in Figure 3.1, the circuit size is reduced by adding and removing redundant wires. A smaller circuit containing the same logics always implies less resources to spend on each step of VLSI design. Therefore, logic optimization by alternative wiring is always being the frontier of all VLSI design problems [CE93].

3.3.2 Timing Optimization

The longest delay path (critical path) usually consists of more than one wire. If it is possible to break the critical path with wire which is not on the critical path. Alternative wiring technique is used to improve the performance as shown in Figure 3.3. The target wire is replaced by other alternative wires inside component 1 or 2. By several iterations which break the critical path, the longest path might be greatly reduced and high performance gain [EEOU96] can be achieved.

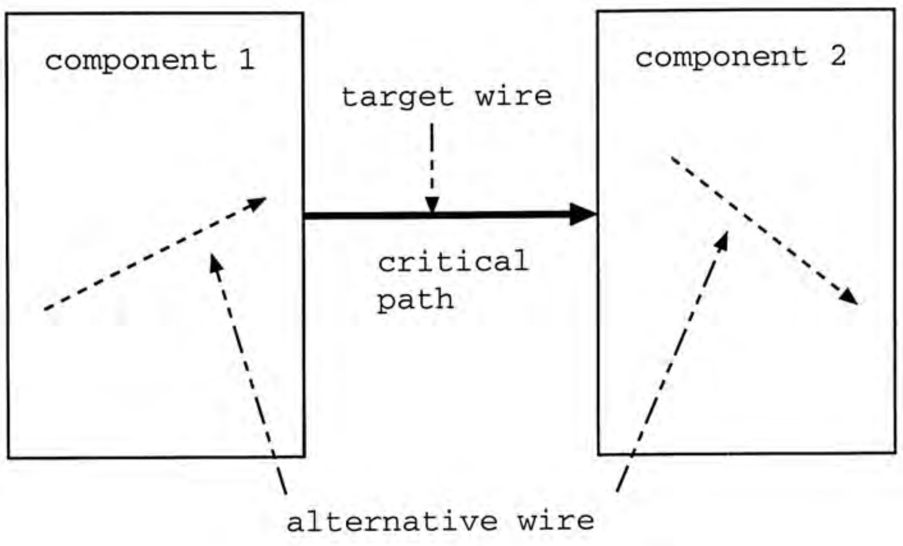


Figure 3.3: Example of Timing Optimization by alternative wiring

3.3.3 Circuit Partitioning and Routing

Figure 3.4 shows the number of wires along the cut line can be reduced by replacing with other wires not in cut set. The application of rewiring over circuit partitioning will be further investigated in the next chapter. Rewiring techniques can also be applied to resolve the unroutable FPGA design [CCWM97] as shown in Figure 3.5.

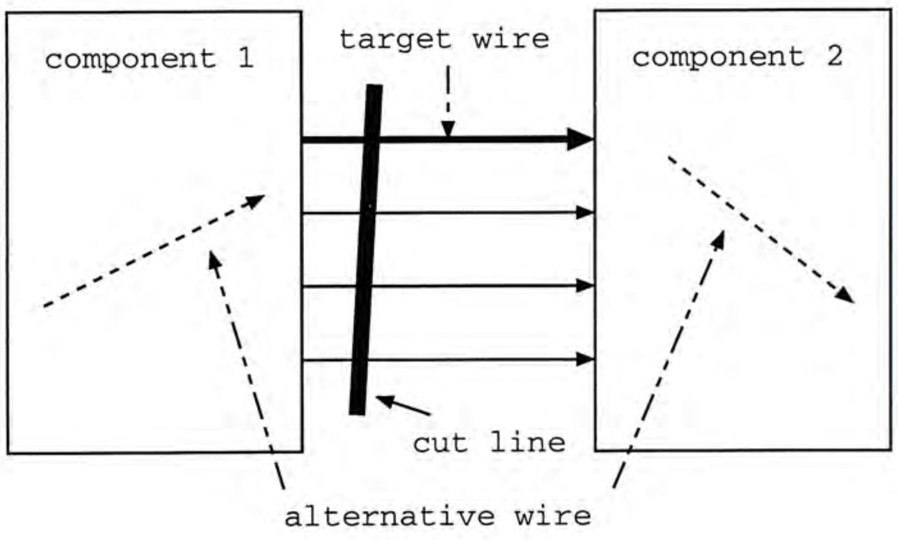


Figure 3.4: Example of Circuit Partitioning

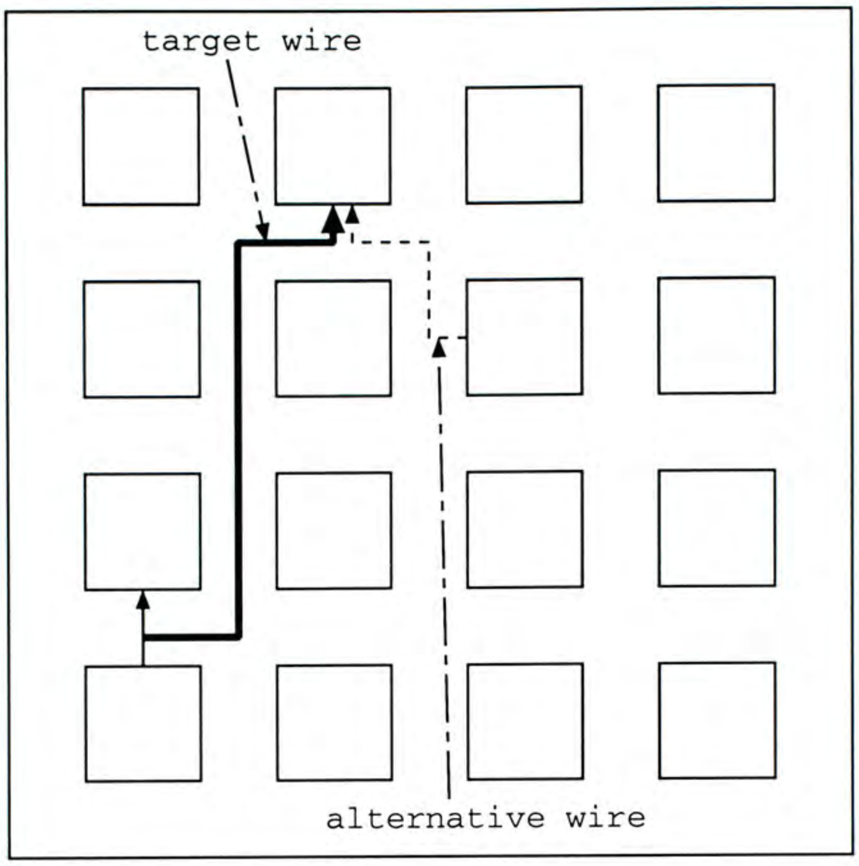


Figure 3.5: Example of Routing

3.4 Logic Optimization Analysis

3.4.1 Global Flow Optimization

Global Flow Optimization [BT91] allows the reconnection of immediate fanouts of a node to the input of other nodes. By modeling the problem as flow graph and solving the problem by maxflow-mincut algorithm, the original circuit size can be reduced. As shown in Figure 3.6, one of the node s is picked from the circuit and has the following circuit diagram. By proper transformation [BT91], a flow graph is built and is solved by maxflow-mincut method as shown in Figure 3.7. The maxflow-mincut solution is $g_1 8$, g_9 and g_8. As a result, the original six connections from s is reduced to three without changing the circuit functionality as shown in Figure 3.8. Logic optimization is done by constructing the flow graph on every node in the circuit.

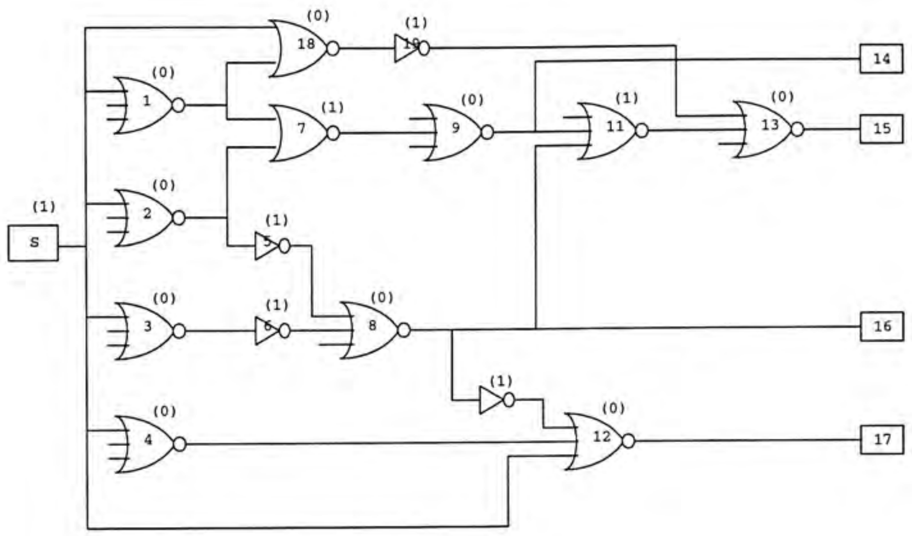


Figure 3.6: Example of Global Flow Optimization

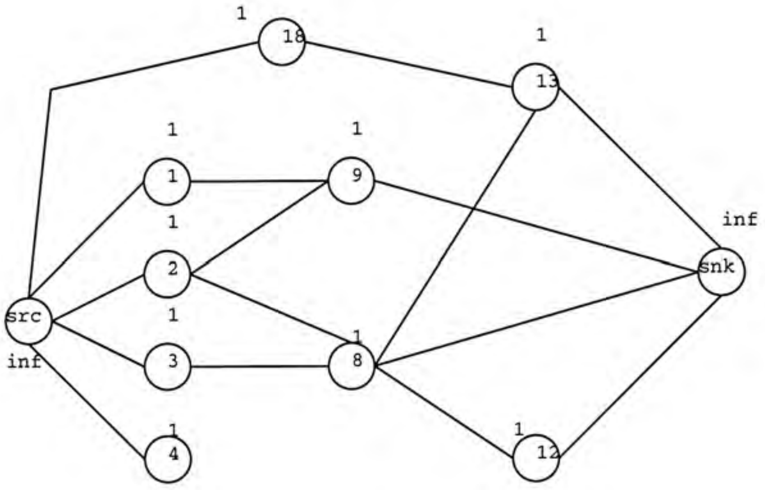


Figure 3.7: Flow graph for the circuit in 3.6

3.4.2 OBDD Representation

In [MKLC89], the Transduction (Transformation and Reduction) method is proposed for multilevel logic optimization which allows adding/deleting connections and gate merging. However, the runtime and the memory storage exponentially increase when the number of inputs increases. It is because the Transduction method uses truth tables to represent the logic function. In order to save CPU time and memory space, the same Transduction method is implemented by using Ordered Binary Decision Diagrams (OBDD) [Bry86]

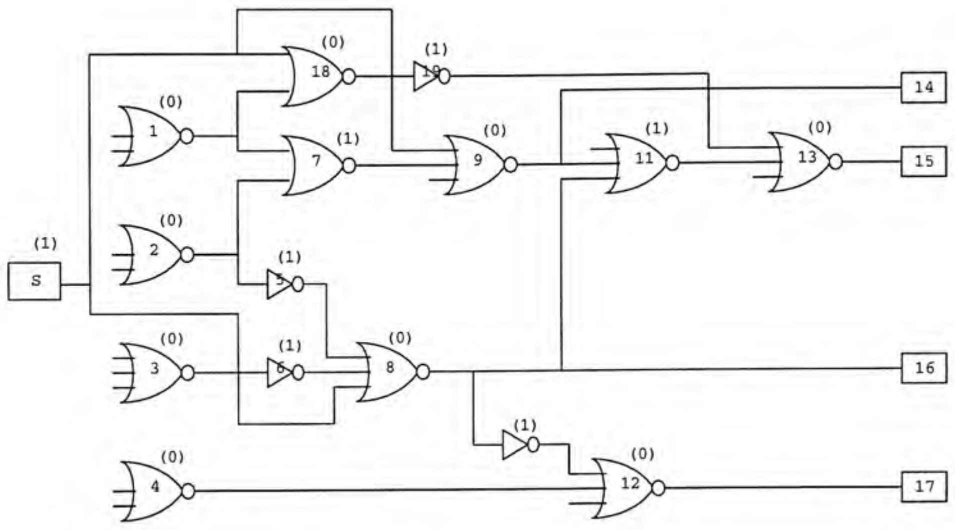


Figure 3.8: Result of fanout re-connections

as the data structures for representing the logic functions. OBDD is a compact logic representation and it is a reduced binary decision tree and requires less memory than other methods. As shown in Figure 3.9, a logic function $f = x1 \cdot x2 + x3$ is represented.

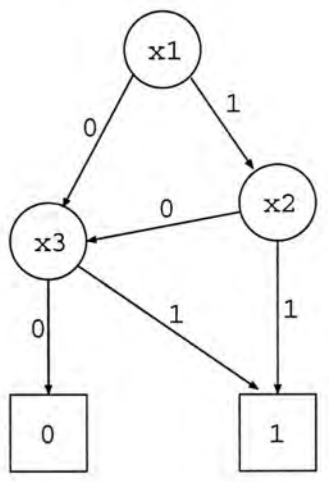


Figure 3.9: OBDD representing $f = x1 \cdot x2 + x3$

3.4.3 Automatic Test Pattern Generation (ATPG)

Nowadays, Multi-level logic synthesis becomes more practical than 2-level logic synthesis because they represent the majority of circuit designs [HS96]. Multi-level circuits refer to those circuits have arbitrary number of gates between primary input and primary output. 2-level circuits are typically represented by SOP or POS forms, i.e. AND plane plus OR plane. There are many methods which address the problem of Multi-level minimization such as considering the don't care set [BBH+88], and Automatic Test Pattern Generation (ATPG) [CE93]. ATPG techniques are built upon the fault models. There are many possible fault models and most common one is single stuck-at-0/1 fault model. If a wire connecting to AND gate detects a stuck-at-1 fault, then it can be removed directly. ATPG will generate a vector for a stuck-at-fault test in the circuit. A vector means the zero or one value to all primary inputs. For a real circuit, if there exists a fault which is untestable that means there is no suitable test vector for such fault. The wire associated with this fault is said to be redundant.

Based on this redundancy test, many ATPG based rewiring tools are proposed. There are two main classes for ATPG tools, they are add-first based and target-first based ATPG techniques. Add-first means by adding one or some redundant wires into a circuit, some other inredundant wires become redundant and thus can be removed. Target-first means first choose one wire (target wire) in the circuit. By adding a suitable wire, this target wire becomes redundant and can be removed.

By using SIS package [SSLea92] which developed by University of California, Berkeley, Redundancy Addition and Removal for Multilevel Boolean Logic Optimization (RAMBO) [EC95] is first proposed. RAMBO creates fault list for each node and removes all redundant wires. The objective is to reduce the wire count so as the area of the circuit. RAMBO shows significant improvement over combinational and sequential circuit and even better results are obtained when incorporate with MIS-II [BRSVW87]. There are many improvement works and applications built upon RAMBO such as Perturb and Simplify [CMSC96], Rewiring [CGMS96, CvGLMS99] and post-layout in FPGAs [CCWM97].

3.4.4 Graph Based Alternative Wiring (GBAW)

Graph-Based Alternative Wire (GBAW) is a newly proposed and efficient rewiring technique. It models a circuit as a directed acyclic graph (DAG) and searches alternative wires by checking graph matching between local sub-networks and the pre-specified minimal sub-graph configurations. A configuration is a minimal circuit pattern containing alternative wires within a given distance. Experiments show that the number of all such local minimal sub-graph is limited. Most of the alternative wires are located topologically "near" to their target wires. It has been shown that about 96% of the closest alternative wires [WLF00] are only 2-edge distant from their target wires. When sub-network matches a pattern, GBAW can quickly determine the target wire and the corresponding alternative wires. Obviously, if w_r is an alternative wire of w_t, then w_t is also an alternative wire of w_r. Both w_t and w_r are presented in a pattern. But in a sub-network, only one of them exists. In [WLF00], it has shown that by using GBAW as a random perturbation engine, a competitive logic optimization result is obtained when comparing to RAMBO while the runtime is much reduced.

0 local pattern

In GBAW, there are more than 40 patterns for rewiring. These 40 patterns are grouped into three main types, 0-local, 1-local and 2-local. A 0-local pattern is the edge distance between the target wire and alternative wire equals zero. On the other words, it means a node substitution for which two nodes having the same logic function can be replaced by each other. If the two nodes have the same fanins set, they are said to be logically equivalent. Figure 3.10 shows the 0-local pattern in GBAW.

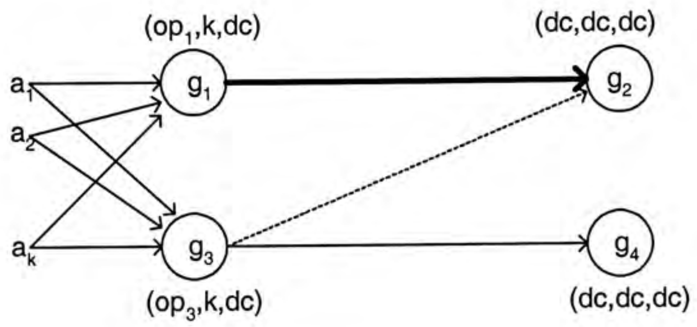


Figure 3.10: 0-local pattern in GBAW

1 local patterns

There are three basic 1-local patterns in GBAW as shown in Figure 3.11. The bolded wire represents the target wire that can be removed if the dotted wire is newly added to the circuit. For example, in case 1.1, the target wire $a \rightarrow g_1$ can be replaced by $a \rightarrow g_2$. Note that the operator of g_1 and g_2 can be changed to other logic gates. This change increases the search space of finding alternative wires in real circuit, and it reduces the runtime in identifying alternative wires.

2 local patterns

The alternative wire is 2-edge far away from the target wire is called 2-local pattern. The number of gates inside a 2-local pattern is increased as well as the number of possible 2-local pattern. Thus, it is necessary to include more 2-local patterns into GBAW. In Figure 3.12, three 2-local patterns are shown.

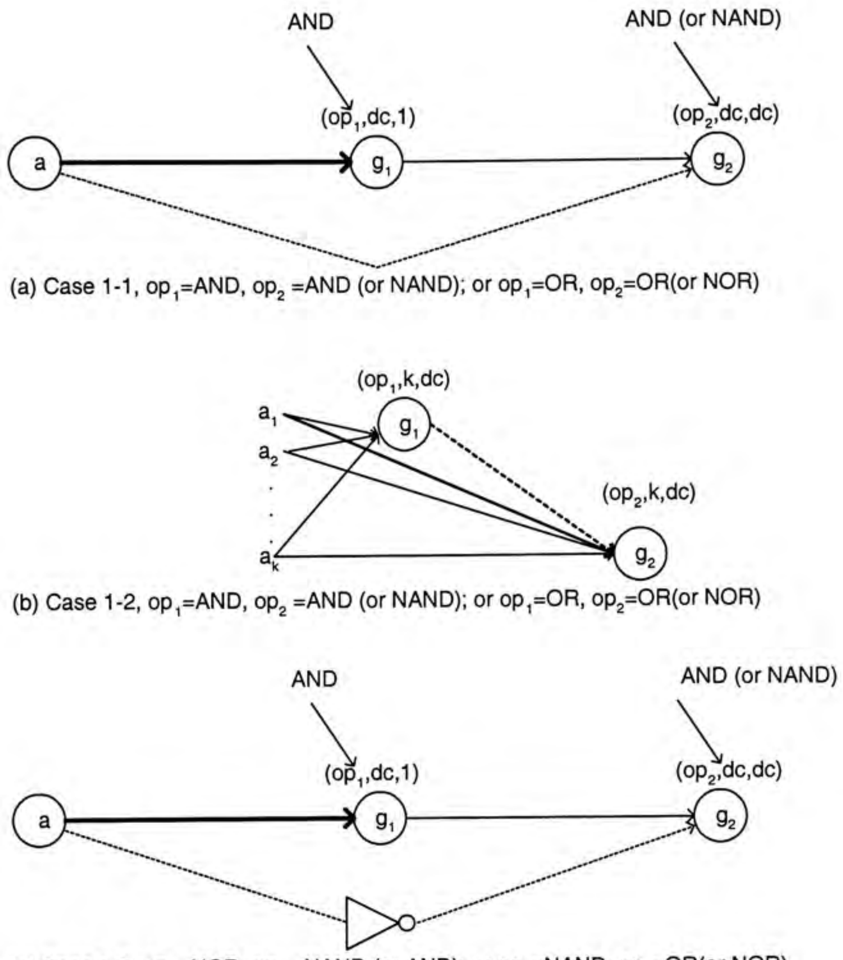

AND
AND (or NAND)
(op1,dc,1)
(op2,dc,dc)
a
g1
g2
(a) Case 1-1, op1=AND, op2 =AND (or NAND); or op1=OR, op2=OR(or NOR)
(op1,k,dc)
(op2,k,dc)
a1
a2
ak
(b) Case 1-2, op1=AND, op2 =AND (or NAND); or op1=OR, op2=OR(or NOR)
(c) Case 1-3, op1=NOR, op2 =NAND (or AND); or op1=NAND, op2=OR(or NOR)

Figure 3.11: 1-local patterns in GBAW

Proof of one 2 local pattern

In order to maximize the searching power, pattern cluster is proposed to put similar patterns together. Currently, 5 clusters are built on top of more than 30 2-local patterns and the new family members are continuously put into GBAW in order to enhance its searching ability.

In Figure 3.13 below, $a \rightarrow g_1$ is the target wire. let $g_1 = (a * x)'$ where x is the other inputs of g_1. $g_2 = (g_1 * y)'$ where y is the other inputs of g_2. $g_3 = (g_2 * z)'$ where z is the other inputs of g_3. Therefore, $g_3 = (((a*x)'y)'z)' = ((a' + x')y)'z)' = ((a'y + x'y)'z)' = ((a'y)'(x'y)'z)' = ((g_4)(x'y)z)'$. Hence, this pattern is constructed and verified. Some other 2-local patterns are shown in Appendix B.

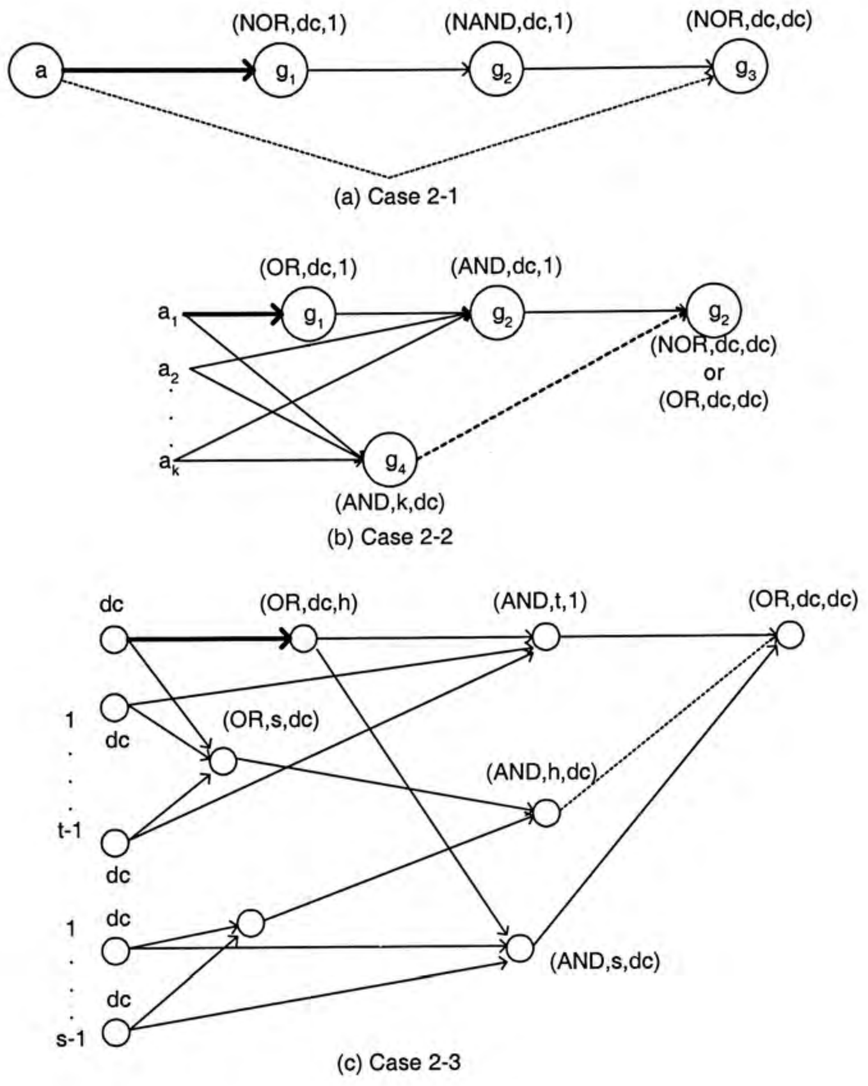


Figure 3.12: 2-local patterns in GBAW

3.5 Augmented GBAW

Owing to the high applicability of alternative wiring technique, many related algorithms were published in last decade. Some research groups prefer algorithms based on Automatic Test Pattern Generation (ATPG) to perform testing of redundancy since ATPG requires little memory to process large circuits. One of the circuit rewiring schemes based on ATPG is the well-known Redundancy-Addition and Removal for Multilevel Boolean Optimization (RAMBO) technique [CE93, EC93, EC95]. RAMBO applies ATPG-based

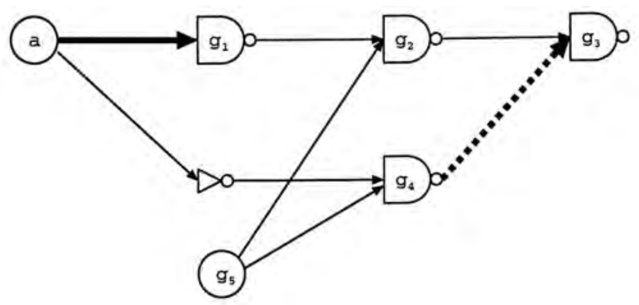


Figure 3.13: Example of one 2-local Pattern Verification

implication technique to identify redundant wires among a large number of candidate wires. It has demonstrated to be very flexible and practical in solving various CAD problems. Improvement in efficiency is achieved by eliminating unnecessary redundancy check [CvGLMS99, LWB00] on the same wire but since the number of candidate wire for testing can still be large, RAMBO may still require a long running time due to the intrinsic exponential time property of ATPG algorithms.

In [WLF00], a new graph-based technique on identifying alternative wire, called Graph-Based Alternative Wiring (GBAW), was first proposed. It first models a circuit network as a directed acyclic graph (DAG) and identifies alternative wires by performing graph pattern matching between local sub-graph of the network and the pre-specified minimal sub-graph configurations containing alternative wires within a given range limit. Experiments show that the number of all such local minimal sub-graph is limited and most of the alternative wires are located topologically "near" to their target wires. It has been shown that about 96% of the closest alternative wires are only 2-edge distant from their target wires.

GBAW produces a competitive result in finding alternative wire when comparing to RAMBO. GBAW not only performs well in searching alternative wires but also runs very fast. Experiments show that on average, the CPU running time of GBAW is just 1.4% of that of RAMBO. This significant short running time makes GBAW being a potentially considerable different technique for identifying alternative wires. The efficiency of GBAW is gained probably mainly because of its avoidance of running the could-be CPU-expensive Boolean implications. And since there exist common alternative-wire patterns that repeatedly occurs in the same circuit, it makes the one-time analysis effort of pattern based rewiring scheme practical.

It was also observed that GBAW may not find many 2-edge distant patterns, if it is limited to search the small pre-defined set of minimal configurations like the early version shown in [WLF00]. Therefore, a much extended GBAW scheme is presented, mainly 2-local pattern based, to greatly improve the effectiveness of GBAW. By including the concept of pattern cluster, the augmented GBAW is kept simple while expanding the pattern family. With the refined GBAW, an encouraging result in circuit optimization is achieved when using it as a perturbation engine. The refinement and achievement are presented in detail in the next section.

3.6 Logic Optimization by using GBAW

A much extended Graph-Based Alternative Wiring (GBAW) scheme to identify alternative wires in multi-level logic with promising results is presented. By modeling subsets of circuits as minimal graphs and applying purely graph-based local pattern search technique, more than 40 graph patterns are found and they contain alternative wires within 2-edge distance from the target wire. Applying proper grouping technique for the similar patterns, the complexity of the rewiring technique can be reduced. Experimental results on MCNC benchmarks show that the technique is much faster than the ATPG-based technique RAMBO with competitive number of alternative wires found. With this augmented pattern family of alternative wires, it is able to find 30% more alternative wires compared to RAMBO with 75 times speedup on average.

GBAW is applied in logic minimization as a perturbation engine and simplify the target circuit by *SIS* algebraic operations. Results show a further reduction of 11.1% in literal count compared to applying algebraic operations alone.

Table 3.1 shows the number of target wires which have 2-local alternative wires. The results are competitive with RAMBO (98%). For each target wire, it may have more than one alternative wire and some of them may be 2-local patterns. Since RAMBO uses ATPG techniques in locating the alternative wires, and the search space is much larger than GBAW. However, a promising result is obtained by GBAW since backward alternative wire can be done while current RAMBO does not.

Name	RAMBO (target/alter.)	GBAW (target/alter.)	Searched (%) (target/alter.)
5xp1	10/21	10/18	100/86
9sym-hdl	5/6	5/8	100/133
C1908	42/57	44/44	105/77
C2670	85/99	83/94	98/95
C3540	208/297	238/250	114/84
C432	33/44	40/40	121/91
C5315	76/113	69/73	91/65
C6288	3/17	3/18	100/106
C7552	132/219	76/82	58/37
C880	27/62	27/27	100/44
alu2	64/100	54/56	84/56
alu4	120/198	84/86	70/43
apex6	72/121	68/70	94/58
b9_n2	8/10	3/3	38/30
comp	28/44	17/17	61/39
des	671/907	795/795	118/88
duke2	35/64	29/29	83/45
misex3	50/167	41/41	82/25
rot	46/75	31/32	67/43
sao2-hdl	25/38	9/9	36/24
term1	41/77	27/28	66/36
ttt2	28/68	10/13	36/19
x3	71/82	76/76	107/93
Total	1880/2886	1839/1909	98/66

Table 3.1: 2-local pattern comparison

The improved GBAW is implemented on Sun UltraSparc 5 workstation for MCNC benchmark circuits and results are shown in Table 3.2. The smallest benchmark circuit *5xp1.blif* is shown in Appendix. The speed and the capability of locating alternative wires between RAMBO and GBAW are also compared. In the table, it is shown that GBAW are able to find 30% more alternative wires than RAMBO with only 1.38% CPU time on average.

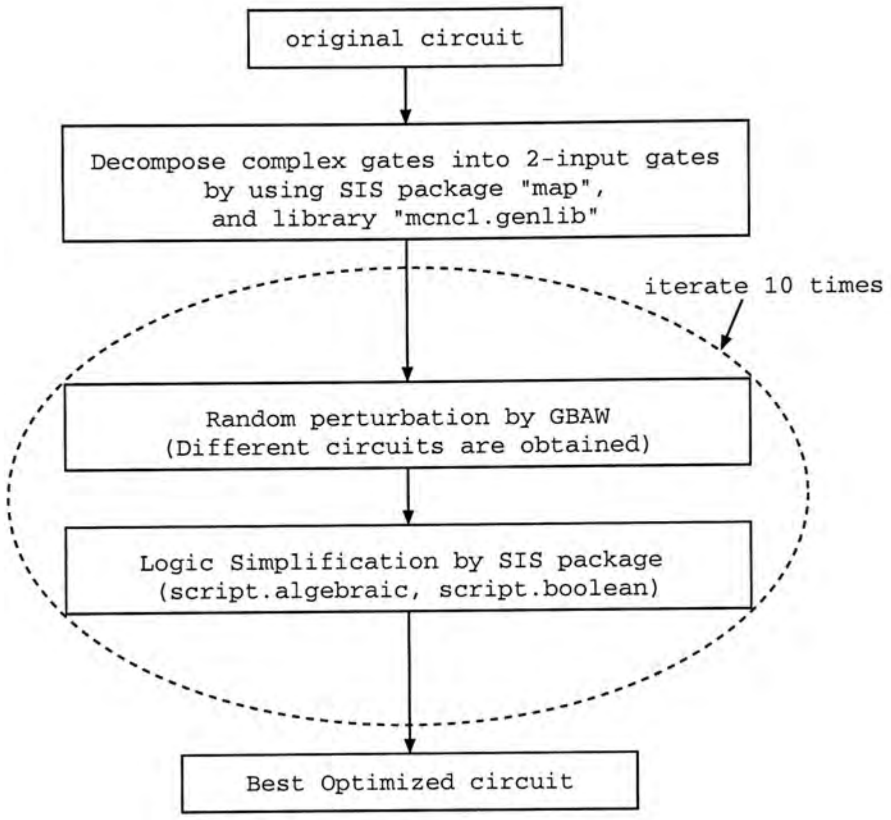

original circuit
Decompose complex gates into 2-input gates by using SIS package "map", and library "mcnc1.genlib"
iterate 10 times
Random perturbation by GBAW (Different circuits are obtained)
Logic Simplification by SIS package (script.algebraic, script.boolean)
Best Optimized circuit

Figure 3.14: The Logic Optimization Process by GBAW and SIS

To demonstrate the application of the proposed scheme, GBAW is deployed in the perturbation and simplification process and depicted in Figure 3.14. In this process, GBAW performed some random transformations on the benchmark circuits and the network structure of the circuit was changed. Then the circuits were minimized by MISII standard script *script.algebraic* provided in SIS package. Before the perturbation using GBAW, the circuits were mapped by SIS command *map* with *mcnc1.genlib* which transforms all the complex gates into gates with 2 fanins. For each circuit, the process was iterated 10 times and all the best results are shown in Table 3.3. The average reduction in circuit size is 43.0% compared to the original circuit. When compared to the minimization with algebraic script alone, a further 11.1% reduction is achieved.

Name	RAMBO alt. wires	RAMBO CPU	GBAW alt. wires	GBAW CPU
5xp1	36	10.17	62	0.24
9sym-hdl	27	1.56	40	0.16
C1355	185	12.82	250	0.89
C1908	127	33.52	240	0.68
C2670	267	83.57	344	1.33
C3540	569	273.80	816	2.15
C432	129	10.26	188	0.37
C499	16	6.05	34	0.6
C5315	511	155.91	713	2.88
C6288	1352	361.18	2191	4.18
C7552	1709	143.95	617	4.2
C880	151	9.86	239	0.66
alu2	169	214.71	263	0.84
alu4	333	270.50	493	1.61
apex6	239	34.32	377	1.23
b9_n2	48	1.65	71	0.17
comp	57	9.18	58	0.21
des	1468	729.92	2204	8.7
duke2	157	46.55	281	0.63
f51m	49	6.19	65	0.25
misex	216	124.48	439	0.97
my_adder	46	1.16	0	0.23
pcler8	29	1.3	30	0.12
rot	243	48.04	406	1.1
sao2-hdl	104	16.86	153	0.39
term1	106	16.81	169	0.37
ttt2	68	9.68	133	0.34
x3	228	23.13	348	1.2
Total	8639	2657.13	11224	36.7
Normalized	1	1	1.2992	0.0138

Table 3.2: Comparison between RAMBO and GBAW

3.7 Conclusions

In this chapter, an augmented Graph-Based Alternative Wiring (GBAW) scheme is presented. Although there are more than 40 patterns included, an attractive efficiency is still maintained by using proper grouping of pattern families. GBAW has forward and backward search capability and can identify alternative wire efficiently. Experimental results showed that it is capable to find 30% more alternative wires comparing with the forward search RAMBO version. GBAW has a good coverage of alternative wires with 75 times speedup on average. When using GBAW as the perturbation engine and combining with SIS algebraic operations, there is a further reduction of 11.1% comparing with the results by algebraic operations alone.

	Circuit Size(lit.)of			Size reduction vs.	
Circuit	Original	MISII	Perturb	Original	MISII
5xp1	235	172	125	0.468	0.273
9sym-hdl	232	138	93	0.599	0.326
C1908	883	582	524	0.407	0.100
C2670	1444	800	714	0.506	0.108
C3540	2267	1398	1367	0.397	0.022
C432	392	296	229	0.416	0.226
C499	854	559	552	0.354	0.013
C5315	3282	1970	1875	0.429	0.048
C6288	5195	3393	3339	0.357	0.016
C7552	4105	2423	2285	0.443	0.057
alu2	777	488	474	0.390	0.029
alu4	1470	921	904	0.385	0.018
apex6	1417	856	839	0.408	0.020
b9_n2	208	141	130	0.375	0.078
comp	270	170	141	0.478	0.171
des	6655	3915	3766	0.434	0.038
duke2	676	430	416	0.385	0.033
f51m	244	169	146	0.402	0.136
misex3	990	644	593	0.401	0.079
my_adder	339	304	197	0.419	0.352
pcler8	174	105	96	0.448	0.086
rot	1251	797	762	0.391	0.044
sao2-hdl	439	271	221	0.497	0.185
term1	439	263	220	0.499	0.163
ttt2	376	237	203	0.460	0.143
Total	34614	21442	20211	Average reduction	
Normalized	1		0.5857	43.0%	11.1%
Normalized		1	0.9426		

Table 3.3: Results for logic optimization

It is also observed that RAMBO is still more flexible in locating the maximum number of all possible alternative wires, it might be a good approach to couple both the advantages of RAMBO and GBAW for various emphasis of CAD applications. The future direction is to extend the set of k-local pattern to cover more alternative wires for practical complex gates, extract more quantitative analysis of alternative wire pattern distributions for different kinds of circuits, which could turn to be useful for new macro library cell designs. More efficient algorithm will be developed to choose the suitable alternative wire for perturbation and to apply the GBAW technique on other CAD problems such as floorplanning and partitioning.

Chapter 4

Multi-way Partitioning using Rewiring Techniques

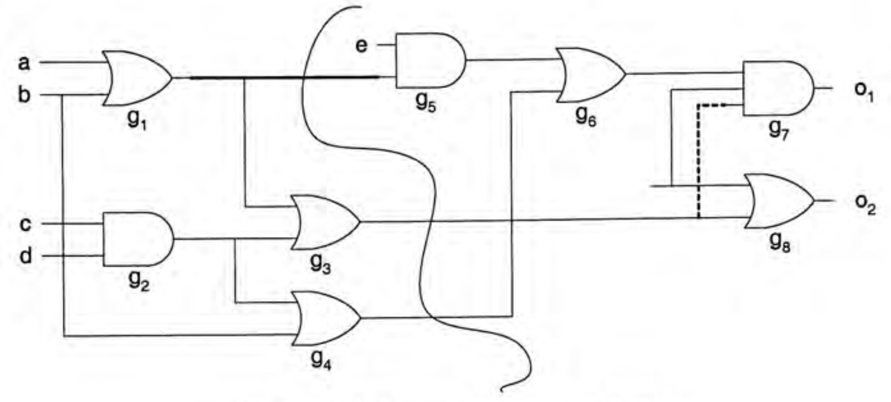
4.1 Introduction

The objective of circuit partitioning is to divide the circuit into sub-circuits such that the size of each component is reasonable and the number of interconnections between the components is minimized. As design scale expands, partitioning becomes increasingly important to circuit design automation.

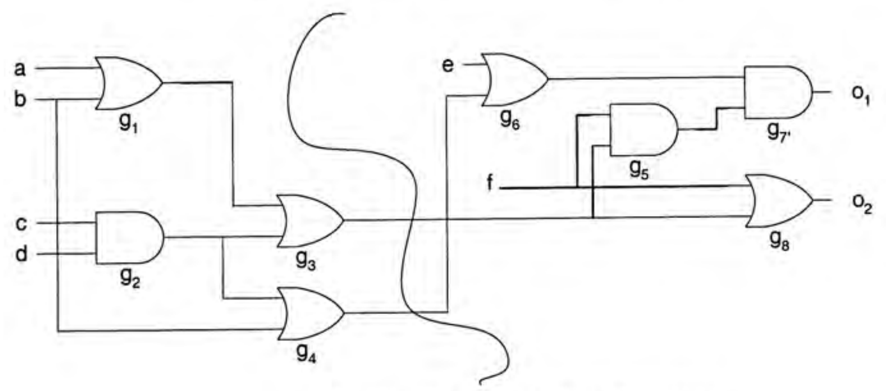
Traditionally, circuit partitioning is done by simply modeling the circuit as a graph (or hypergraph). Graph partitioning problems are known to be NP-hard [WC91]. A comprehensive survey [AK95b] has presented the recent directions of partitioning. Commonly used partitioning algorithms can be categorized into three classes. The first class strictly abide by the modeling graph, with no attempt to change the graph. High quality results have been reported by several algorithms which include iterative improvement based [WC91, FM82, DD96], clustering based [YCL92], and spectrum (eigenvector) based [HK91, ZSC96]. The second class of algorithms may modify the graph through node replications [KN91, MW96, EHS97, YW98]. Improvement is achieved by sacrificing some area due to node replications. These two classes both perform the partitioning task on the graph without considering the logic function of the circuit. The third class [CLMS95, BLSV92, CCMS93, KBZ94] couples the graph domain (nodes and their connections) and logic domain (function perform by each node). The tradeoff of improving the partitioning results is the expensive computational cost [BLSV92, CCMS93] and may only be applied to Field-Programmable Gate Array (FPGA) circuits [KBZ94].

Recently, many research works on multi-level partitioning are proposed [HL93a, HL93b, KK95c, KAKS97, AHK97b, WA98, KK99]. The general idea behind multi-level partitioning is to first cluster the whole problem by some useful algorithms to reduce the size, then apply a well-known graph domain partitioner on the coarsened graph to get a good initial solution. The graph is then unclustered and a suitable partitioning refinement algorithm is applied in order to adjust the cut edge between partitions. The quality and the runtime by multi-level partitioning are very encouraging. In particular, Karypis and Kumar [KK99] proposes a partitioner called hMETIS-Kway. It first coarsens the hypergraph, then recursively bisects the graph into k-parts, followed by uncoarsening the hypergraph with refinement algorithms. More recent research works [CKM00, CL00a, CL00b], in comparison with hMetis-Kway, showed that the solution by hMetis-Kway is of high quality that the cut size cannot be further reduced greatly.

Alternative wiring (rewiring) is the technique of adding single or multiple redundant wires or gates to a circuit such that other wires or gates become redundant and thus removable. This logic domain technique has been widely used for solving many logic level and physical level design problems [CE93, CMSC96, CLMS95, CCWMS94a, CCWM97, CMS94]. Circuit performance can be improved by removing a wire on the critical path and adding its alternative wire elsewhere. Circuit routability can also be improved by substituting an unroutable wire in congested area by a routable alternative wire in some other circuit part. The cut size of a partition can be reduced by replacing the wires crossing the cut line.



(a) Partition before applying alternative wiring



(b) Partition after applying alternative wiring

Figure 4.1: Circuit partitioning by rewiring

Figure 4.1 illustrates how rewiring can be used to further improve an already optimum partition result obtained by a typical graph domain partition algorithm. The global optimum partition result in the graph domain, with a cut size of 3, is shown in Figure 4.1(a). However, by applying the logic domain rewiring technique to replace a target wire (thick line) crossing the cut line by its alternative wire (dotted line), the cut size can be further reduced to 2 as shown in Figure 4.1(b) (without injecting area increase). From this example, rewiring can be applied to partitioning to further improve upon even optimum solution in the graph domain. Binding the logical domain rewiring technique with some graph domain partitioning tool enables a larger room for obtaining better results. The rewiring technique can be used either as a greedily guided optimization tool, or as a random perturbation tool which allows hill climbing on cut cost functions so as to pull the cost out of some local minima.

The well-known ATPG-based rewiring technique, RAMBO, is a very powerful technique for identifying alternative wires of a specified target wire for a given circuit. This technique has been used for logic perturbations and has been integrated with a graph domain partitioning algorithm to produce improved partitioning results [CLMS95]. However, as it always selects a non-negative-gain wire in replacing a target wire and it only considers simple cases of adding and replacing one single wire, it is easily trapped in local minima. Besides, the ATPG-based rewiring technique, though powerful, tends to spend much running time due to the time-consuming Boolean implication operations. Moreover, the RAMBO rewiring tool can only handle 2-input gates, therefore the benchmark circuits must be somehow pre-processed.

To investigate the possibility of perturbing the circuit without applying any Boolean operations, minimal circuit structures yielding rewiring patterns have been studied [WF99, WLF00]. Based on benchmark circuits, there is an observation that the nearest existing alternative wire is quite close to its target wire. Therefore these minimal patterns tend to be small and repeatedly appeared in a circuit. As a result, instead of applying the ATPG-based logic implications repeatedly for a same pattern, the Graph-Based Alternative Wire (GBAW) technique [WF99, WLF00] employs a more efficient graph pattern matching operation to locate alternative wires. The basic idea of GBAW is to match the sub-circuit with some "pre-specified" patterns. Rewiring by GBAW can be done without doing any logic implication or redundancy check, hence it runs very fast. Besides considering the alternative wire which is close to the target wire from those small "pre-specified" patterns, distant alternative wires can also be located by propagating the matchings in a cascading way. By coupling RAMBO and GBAW as the perturbation engine, [WYC00] proposed the bi-partitioning tool RG which also handles the 2-input gates and has a larger room for perturbation.

To expand the optimization space, the coupling notion is applied between graph and logic domain into the GBAW-Partitioner (GP). In graph domain, the well-known Fiduccia-Mattheyses (FM) partitioning algorithm [FM82] is chosen as the iterative move-based engine for its simplicity. In logic domain, an augmented GBAW is applied, which enhances the ability to locate more 2-local alternative wires, as a greedily guided perturbation engine. In the experiments, near optimum partition results were firstly obtained from the pure graph domain partitioner hMetis-Kway. Then the coupling of graph and logic domain optimization GP engine, was followed. Note that the logic perturbation process can be coupled with any powerful graph domain partitioning tool, and GBAW itself is able to handle patterns with multiple-input gates. Experiments on this partition flow for 2-, 3-, 4-, and 5-way partitionings are carried out on various MCNC benchmarks ranging from small to fairly large circuits. The results show that such a graph-logic domain coupled partitioning approach can further cut down the cut size effectively with small CPU overhead. The results show that it is a very promising direction for circuit partitioning.

This chapter is organized as follows. In Section 4.2, a brief introduction on partitioning algorithm analysis is described. In Section 4.3, the details of repartitioning by rewiring is shown. In Section 4.4, experimental results are presented. Conclusion is drawn in Section 4.5.

4.2 Circuit Partitioning Algorithm Analysis

The partitioning task is ubiquitous to many subfields of VLSI CAD. One important practical example of this problem is the placement of the electronic components onto printed circuit boards, so as to minimize the number of connections between cards. The components are modeled as the nodes of the graph, each circuit connections as edge, and the edge which interconnects between cards as a cut. Since connections between cards have high cost compared to connections within a board, the objective of partitioning in this example is to minimize the number of interconnections between cards.

Partitioning heuristics address the increasing complexity of VLSI design systems with million of transistors, have a greater impact on system performance since now designs are interconnect-dominated. In current VLSI designs, wire delays tend to dominate gate delays, thus poor partitioning results worse system performance.

It is impossible to solve the above problem by exhaustive procedure even for small problem size. For example, suppose G has n nodes of size 1 to be partitioned into k subsets and each subset has size p, where $kp = n$. There are $\binom{n}{p}$ ways of choosing the first subset, $\binom{n-p}{p}$ ways for the second and so on. Since the ordering of subsets is not important, the number of cases is

$$\frac{1}{k!}\binom{n}{p}\binom{n-p}{p}\cdots\binom{2p}{p}\binom{p}{p}$$

For example, for $n = 40$, $p = 10$ (i.e. $k = 4$), the number of cases are greater than 10^{20}. Therefore, any direct approach to finding an optimal solution will require an inordinate amount of computation, heuristic methods can produce good solutions (possibly even an optimal solution) quickly. In practice, it is more preferable to have several good solutions than one optimal one.

There are several approaches to solve the partitioning problem, such as *Move-based approaches, Geometric representations, Combinatorial formulations and Clustering approaches* [AK95b].

The first "efficient" graph bisection heuristic was introduced by Kernighan and Lin [KL70] in 1970. It was further improved the time complexity by Fiduccia and Mattheyses [FM82] in 1982. Both famous algorithms KL & FM use iterative improvement technique that are based on greedy strategy which start with some feasible initial solution and iteratively move to the best (improving) neighboring solution. The process ends when the algorithm cannot have any iterative improvement. The computational complexity of KL and FM are $O(n^3)$ and $O(n)$ respectively. KL & FM are both iterative improvement that apply move-based approaches. Other move-based approaches are simulated annealing, tabu search, genetic algorithm.

Nowadays, this two famous algorithms are still use in practical applications. However, many extensions which base on the idea of this two algorithms are evolved afterwards. Such as the Tie-breaking strategies by Krishnamurthy [Kri84], Sanchis's multi-way partitioning algorithm [San89] and Multi-level partitioning [KAKS97].

The KL & FM Algorithms are described in Section 4.2.1 and 4.2.2 respectively. Their extensions are briefly presented at the end of Section 4.4.2. Geometric representation approaches are discussed in Section 4.4.3. The Multi-level partitioning is introduced in Section 4.4.4.

4.2.1 The Kernighan-Lin (KL) Algorithm

The simplest partitioning problem is to find a minimal-cost partition of a given graph of $2n$ vertices (of equal size) into two subsets of n vertices each. KL algorithm is not only able to solve this uniform bisection problem, it also provides the basis for solving other general partitioning problems, e.g. larger problems, 2-way partitions of the vertices with unequal size. Two-way uniform partitioning is first described. Followed by the algorithm of KL and example is shown in later section.

Two-way Uniform Partitioning

The algorithm works as follow, it first starts with any initial partition X, Y of P. By interchanging the vertices between X and Y, the total cost which connecting X, Y can be reduced, where cost c_{ij} define as the cost of vertex i connect to vertex j. When the algorithm stops, it always results in a local minimum X', Y'.

Suppose $A \in X, B \in Y$ is the two initial partition, and final partition results in $B \in X = X', A \in Y = Y'$ the problem is to select the correct set A, B from X, Y without considering all the possible choices. Suppose x, y are vertices in P, an define $x \in X$, an external cost E_x and internal cost I_x, external and internal costs difference D_x and the gain (reduction in cost) are

$$\begin{aligned} E_x &= \textstyle\sum_{b \in Y} c_{xb} \\ I_x &= \textstyle\sum_{a \in X} c_{xa} \\ D_x &= E_x - I_x \\ \text{gain} &= D_x + D_y - 2c_{xy} \end{aligned}$$

First, the algorithm computes the D values for all elements of P. Second, then it choose the maximum gain from a single interchange where

$$g_i = D_{x_i} + D_{y_i} - 2c_{x_i y_i}$$

All the beginning of a pass, each vertex is *unlocked* which means it is free to be swapped. The vertex becomes *locked* after moved. KL iteratively swaps the pair of unlocked vertices with the highest gain until all vertices become locked. Therefore, g_1 is computed first, with x_i, y_i locked temporary during this pass, and called x'_1, y'_1. Then the D values for all elements of $X - x_i$ and $Y - y_i$ are recomputed. Then, a pair x'_2, y'_2 from $X - x'_1$ and $Y - y'_1$ is chosen such that $g_2 = D_{x'_2} + D_{y'_2} - 2c_{a_2' b_2'}$ is maximum where $(x'_1, y'_1$ is not considered. Process continue until all nodes have been exhausted, $(x'_3, y'_3), \ldots, (x'_n, y'_n)$ with corresponding $g_3, \ldots, g_n$.

If $A = x'_1, x'_2, \ldots, x'_k$, $B = y'_1, y'_2, \ldots, y'_k$. Note that some g_i's are negative and $\sum_1^n g_i = 0$. KL choose k which will maximize the partial sum $\sum_{i=1}^k = g_i = G$, If $G > 0$ means there is a reduction in total cost by swapping A and B from X, Y. When $G = 0$ the algorithm has arrived one of the local optimum partition. With different starting partition, the algorithm may reach different local optimum solution.

Pseudo-Code

The algorithm of KL is briefly shown below:

```
• Start with an initial bisection P = X, Y.
• Repeat
    – For i = 1 to n/2:
        * Choose a pair of free cells $a \in X, b \in Y$
          s.t. exchanging a and b gives the highest gain, gain(a,b).
        * Exchange and lock a and b.
        * Let $g_i$ = gain(a,b).
    – Find k s.t. G = $g_1 + g_2 + \cdots + g_k$ is maximized and shuffle cells up
      to this $k^{th}$ step.
• End if G = 0.
```

Illustration

In Figure 4.2 and 4.3, a partition problem is solved by KL algorithm. There are totally 2 passes, the partial sum of G is maximized during each pass and terminated when equal to zero. In the first pass, the gain of *swap*(a, d) and *swap*(b, c) equals 2, therefore *swap*(a, d) is chosen and locked them after moved. Other move will result in negative gain. Therefore, the G in the first pass is equal to 2. In the second pass, the process is terminated since the maximized partial sum is equal to zero.

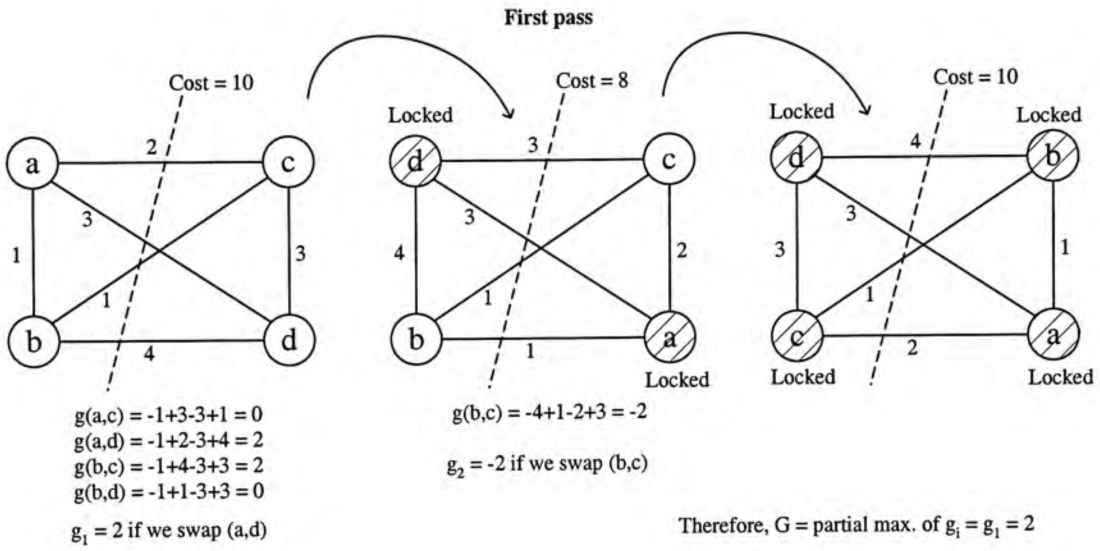


Figure 4.2: Example of KL algorithm - First pass

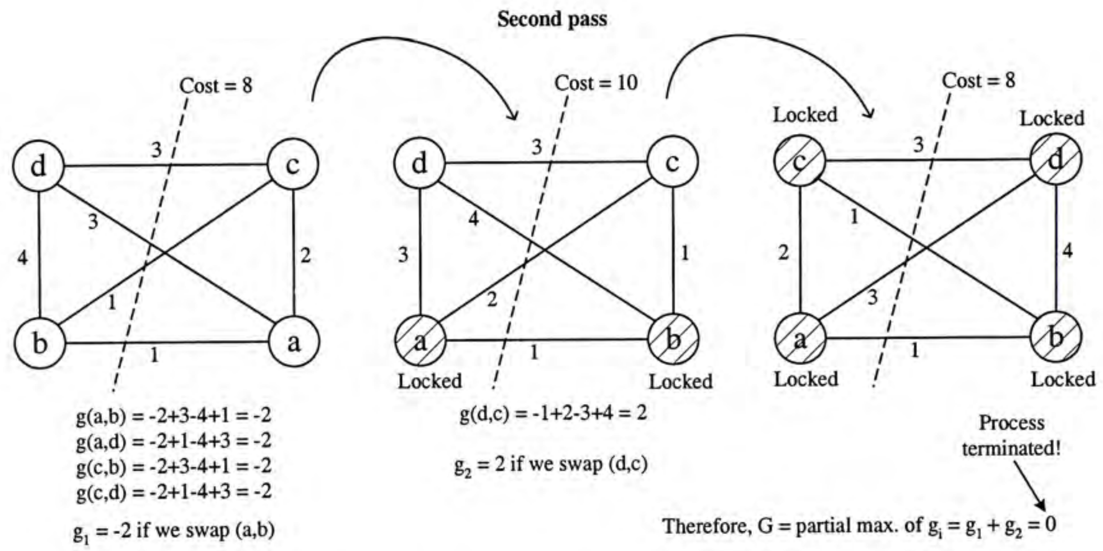


Figure 4.3: Example of KL algorithm - Second pass

4.2.2 The Fiduccia-Mattheyses (FM) Algorithm

The Fiduccia-Mattheyses algorithm is based on KL algorithm. They introduced their efficient heuristic FM by reducing the time complexity per pass to

$O(p)$ where p is the number of terminals in P. The main difference between KL and FM are: 1) move only one cell each time, 2) vertices are weighted, 3) nets can be multi-terminal, 4) maintain a *balanced* partition after each move.

The FM algorithm performs several passes, within each pass each cell moves at most once and returns the best solution observed in one pass. It terminates when there is no improvement of cost function in a pass. However, FM has reduced the time per pass to linear in the size of the vertices.

A partition $P = (X, Y)$ is balanced iff $\frac{w(X)}{w(X \cup Y)} \approx r$ for some constant $r \leq 1$ where $w(X)$ is the total size of the cells in X. To preserve balanced partition, cell b is able to move only if

$$rW - S_{max} \leq w(X) \leq rW + S_{max}$$

where size of each cell $= s(i)$, $W = w(X \cup Y) = \sum_{i=1}^{n} s(i)$ and $S_{max} = max(s(i))$ is the maximum cell size.

A special data structure is used for selecting which vertices to be moved in order to improve the running time. This structure is *Bucket list structure*, a cell with maximum gain can be found, remove a cell, insert a cell, update gain(i) for any cell i, update the MAXGAIN in linear time by using this bucket. There are two lists and one for each partition.

The algorithm starts with a balanced partition which can be obtain by first sorting the vertex weight in decreasing order and placing them in X, Y alternately. At the beginning of a pass, the gains for each cell in P are computed in linear time, and their value are inserted to the bucket list according to its gain. Once the lists are updated, the vertex to be moved is chosen by considering the maximum gain vertex x_{max} in list X, or the maximum gain vertex y_{max} in list Y. Both vertices can be moved only if they do not violated the balance condition. Notice that a move may increase the cut-cost but results in hill-climbing out of the local minimum. When no more possible move or if there are no more unlocked vertices, the pass terminates.

During an FM pass, a cell pointing to the MAXGAIN is selected from the bucket, and deleted from the linked list. The cell is locked after each move, and the gains of unlocked cells incident to the moved cell are updated in the bucket. If the bucket indexed by MAXGAIN becomes empty, the MAXGAIN will be decreased until it indexes a non-empty bucket.

Bucket list structure

FM algorithm uses the special data structure – *Bucket List* so as to improve the time-complexity. The cell to be moved is always selected by the pointer which pointed by MAXGAIN. The bucket itself will store different gain value, and a linked list will point to the specific bucket slot. This structure looks like a *comb* which can effectively improve the whole algorithm.

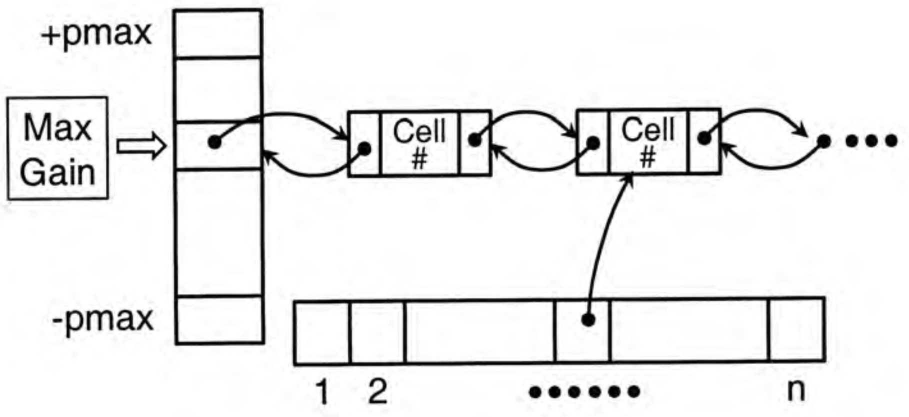


Figure 4.4: Bucket list structure

Pseudo-Code

The algorithm of FM is briefly shown below:

```
• Start with a balanced partition P = X, Y.
• Repeat
    – For i = 1 to n:
```

    * Choose a free cells $b \in X \cup Y$
      s.t. moving b to the other side gives the highest gain, gain(b), and moving b preserves *balance* in P.
    * Move and *lock* b.
    * Let g_i = gain(b).
  - Find k s.t. G = $g_1 + g_2 + \cdots + g_k$ is maximized and shuffle cells up to this k^{th} step.

- End if G = 0.

Illustration

The example of FM is shown in Figure 4.5 and Appendix C.1, C.2. The initial and final partition are shown in Figure 4.5 with the balanced constant $r = 0.5$, each cell with weight = 1, total weight of $P = 4$. Therefore the cell in the partition is valid to move only when it satisfies $1 \leq w(X) \leq 3$. The final cut-cost is changed from 10 to 6.

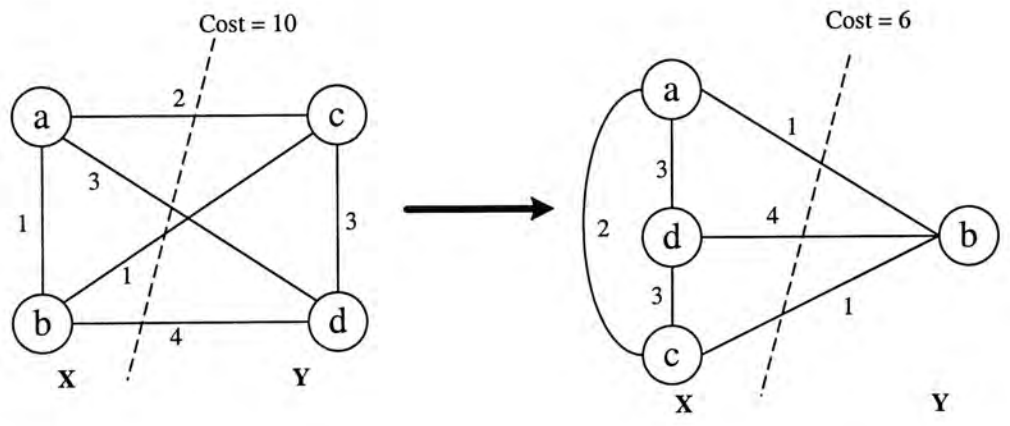


Figure 4.5: Example of FM algorithm - Initial & Final

Extension 1: Tie-Breaking Strategy

When picking the maximized partial sum G, break ties by looking ahead a certain number of steps. E.g. $g_k = g_1 = 2$, and $g_k = g_1+g_2+g_3 = 2-4+4 = 2$, the algorithm introduces a look ahead ability into the original FM algorithm. At this situation, the algorithm should able to select whether swap one pair of cell or three pairs of cell, that can introduce a better solution.

Extension 2: Multi-Way Partitioning

Partition into more than 2 clusters. Extend the idea of FM + Krishnamurthy and handle multi-way network partitioning and this technique is very useful for partitioning a network into a large number of subsets.

4.2.3 Geometric Representation Algorithm

Spectral algorithms were first proposed for placement and partitioning by Hall [Hal70]. Based on a linear ordering of the vertices using the eigenvector associated with the second smallest eigenvalue of the Laplacian of a graph, a fast bi-partitioning method were developed [HK92]. Consequently, a k-way spectral partitioning algorithm and new k-way ratio-cut cost function were presented in [CSZ94]. Subsequent methods for spectral k-way ratio-cut partitioning include **MELO**[AY95] and **DP-RP**[AK95a]. Additional approaches of spectral partitioning are presented in [Bar85].

The problem of k-way partitioning can be formulated as below. Given a graph with a set of n vertices, V. Each vertex is associated with its size attribute. A partitioning of the graph is a division of the n vertices into k disjoint, non-empty subsets $P_1, P_2, \cdots, P_k$ such that $V = P_1 \cup P_2 \cup \cdots \cup P_k$.

- The $n \times n$ adjacency matrix, A, is composed of entries a_{ij} which represent the sum of the weights of the edges between vertices i, j.
- The $n \times n$ diagonal degree matrix, D, has entries d_{ii} equal to the sum of the weights of all edges on vertex i.
- The Laplacian matrix is defined as $Q = D - A$.

- E_h is the sum of the weights of the edges which have exactly one vertex in partition h.
- $||P_h||$ is the sum of the sizes of all vertices in partition h.
- R is the $n \times k$ ratio-ed assignment matrix. It represents a solution to the partitioning problem. The entry r_ih has value $\frac{1}{\sqrt{||P_h||}}$ when vertex i is in partition h and 0 otherwise.
- M is the $n \times n$ diagonal matrix whose m_{ii} entry represents the size of vertex i.

In [Hal70], Hall shows the (1-D) weighted quadratic placement problem can be solved by the eigenvectors of the Laplacian matrix. The total weighted squared distance between n points can be express as below. That is what we want to minimize

$$z = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}(x_i - x_j)^2$$

and it can be rewritten in matrix notation (quadratic form) as

$$\text{minimize } z = x^T Q x \text{ subject to } x^T \cdot x = 1$$

The minimum eigenvalue $\lambda_1 = 0$ yields the trivial solution. The second smallest eigenvalue λ_2 is a lower bound on a non-trivial solution to the above problem. Based on this idea, Hagen and Kahng [HK92] integrate the two-way ratio-cut partitioning and spectral method. Figure 4.6 and 4.7 show an example of spectral bi-partitioning.

The k-dimensional weighted quadratic placement problem consists of finding an $n \times k$ matrix $X = [x_{ij}]$, minimizing

$$z = \frac{1}{2} \sum_{i=1}^{n} \sum_{h=k}^{n} \sum_{k=1}^{n} a_{ij}(x_{ih} - xjh)^2 = trace(X^T Q X)$$

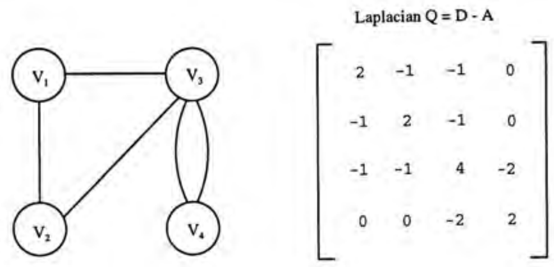


Figure 4.6: A simple graph and its Laplacian

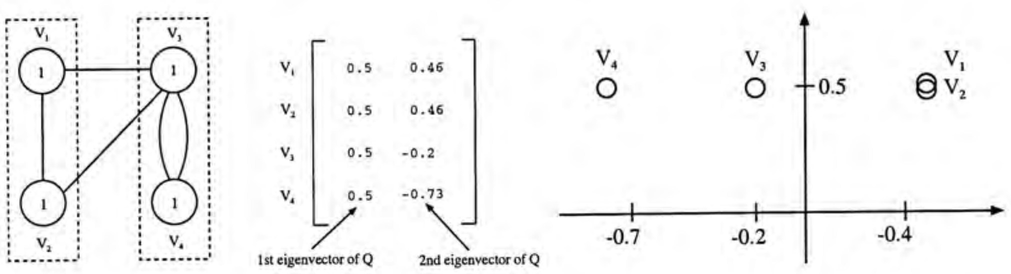


Figure 4.7: Result of spectral partitioning and linear ordering of the vertices using the second eigenvector

To avoid the trivial solution of the constraint $X^TX = I$ is used and the k eigenvectors of Q corresponding to the smallest k eigenvalues of Q and minimize z. The aim is to optimize the k-way ratio-cut cost function that is find a solution R such that minimize

$$\sum_{h=1}^{k} \frac{E_h}{||P_h||}$$

By using R and $||P_h||$ which include the actual vertex sizes rather than the number of vertices in a partition, the new spectral partitioning incorporates vertex size information. By taking vertex sizes into account, the constraint $R^TR = I$ is replaced by $R^TMR = I$. R is matrix which has a single non-zero entry in every row and for each column exactly one non-zero value among its entries. The generalized spectral partitioning formulation is defined as

$$\text{minimize } trace(X^TQX) \text{ subject to } X^TMX = I$$

where $trace(M)$ of a square matrix M is the sum of its diagonal entries. By using $M = S^TS$ and substituting $\hat{Q} = S^{-1^T}QS^{-1}$ and $\hat{X} = SX$, the original formulation is transformed to

$$\text{minimize } trace(\hat{X}^T\hat{Q}\hat{X}) \text{ subject to } \hat{X}^T\hat{X} = I$$

which is a standard quadratic assignment problem and can be solved by using the LASO library [PS79]. Below shows an example which incorporates the actual vertex sizes information into Laplacian matrix and solve the corresponding eigenvectors.

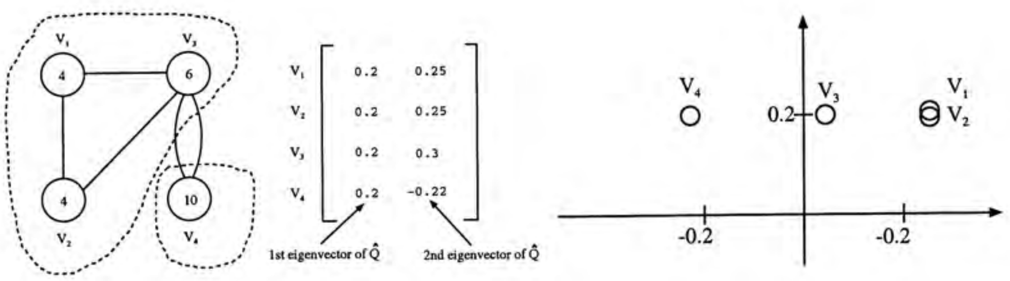


Figure 4.8: Result of spectral partitioning by using the modified Laplacian $\hat{Q}$ which incorporates the actual vertex size information

4.2.4 The Multi-level Partitioning Algorithm

Recently, many research works on Multi-level Partitioning [BS93, HL93b, KK95c, ZCS99, AHK97b, KAKS97, KK99] which first clusters the whole problem by some useful algorithm to reduce the size (i.e. in a graph by collapsing vertices and edges). Then, the reduced problem can be solved efficiently by any well-known partitioning algorithm. The last step is to do un-clustering while applying a suitable partitioning refinement algorithm in order to adjust the cut edge between partitions. Figure 4.9 shows the three phases in MP, *coarsening* phase, *initial partitioning* phase and *un-coarsening and refinement* phase.

The first multilevel work is achieved by Barnard and Simon [BS93] in 1993. They used a multilevel approach to calculate the eigenvector for the spectral partitioning algorithm. Their work has greatly inspired the research group

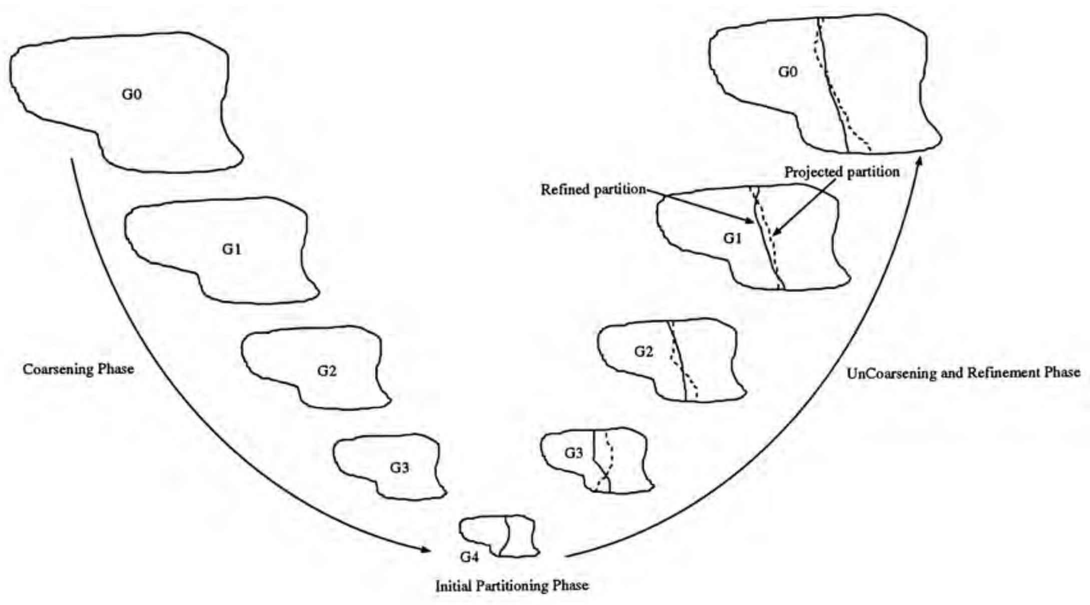


Figure 4.9: The various phases of the Multi-level graph bisection

of Hendrickson [HL93b]. Hendrickson and Leland proposed **Chaco** [HL93a] in 1993 and it works without the transferring of eigenvectors between levels by instead transferring partitions between levels. Many research works and ideas have elaborated and built based on their work. In 1995, the research group of Andrew B. Kahng has presented their work on multilevel circuit partitioning $\mathbf{ML}_C$ and $\mathbf{ML}_F$ where $\mathbf{ML}_C$ is CLIP-based [DD96] and $\mathbf{ML}_F$ is FM-based implementations. Meanwhile, the research group of Pak K. Chan has integrated multi-level scheme with their k-way hypergraph partitioning by the use of spectral methods[ZCS99, CSZ94, ZSC96] **MKP**. The algorithm also directly incorporates the vertex size information during the calculation of eigenvalues. Another famous multilevel graph partitioning tools **METIS** [KK95b, KK95a, KK95d] was proposed by the group of Vipin Kumar in 1995. Their extension works include from bisection to k-way partitioning, from graph to hypergraph [KK99], parallel multilevel graph/hypergraph partitioning **ParMETIS** [KK96] and k-way hypergraph partitioning **hMETIS** [KK99]. These multi-level tools have soon become industry standard due to their superior performance in solving the partitioning problem efficiently.

4.2.5 Hypergraph METIS - hMETIS

Karypis, Aggarwal, Kumar and Shekhar present a new hypergraph-partitioning algorithm **hMETIS** [KAKS97, KK99] that uses the multilevel paradigm. A sequence of coarser hypergraphs is constructed first and then applies initial partition (directly partition into k-parts), and this k-way partitioning is projected back to the original graph with refinement.

Coarsening Stage

Coarsening phase leads to a small hypergraph which partitions into k-parts is not significantly worse than directly applies k-way partitioning on the original graph. It also allows local refinement techniques to become more effective, and reduce the size of hyperedges. There are three schemes combined together to form the coarsening stage of **hMETIS**, *edge coarsening* (EC)[AHK97b, WA98, KAKS97], *hyperedge-coarsening* (HEC)[KAKS97], *modified hyperedge-coarsening* (MHEC).

EC scheme is a heavy-edge maximal matching of the vertices of the hypergraph. The vertices are visited in random order. For each vertex v, all unmatched vertices that belongs to the hyperedge incident to v are considered. The vertex u with highest weight is matched with v.

In HEC, an independent set of hyperedges is selected and the vertices that belong to individual hyperedges are contracted together. The hyperedges are first sorted in non-increasing hyperedge-weight order and the hyperedges of the same weight are sorted in a non-decreasing hyperedge size order.

MHEC can reduce the amount of hyperedge weight that is left exposed in successive coarse graphs. After applying HEC and the hyperedges to be contracted have been selected, the list of hyperedges is traversed again. For each hyperedge that has not yet been contracted, the vertices that do not belong to any other contracted hyperedge are matched to be contracted together.

The overview of three schemes is in Figure 4.10.

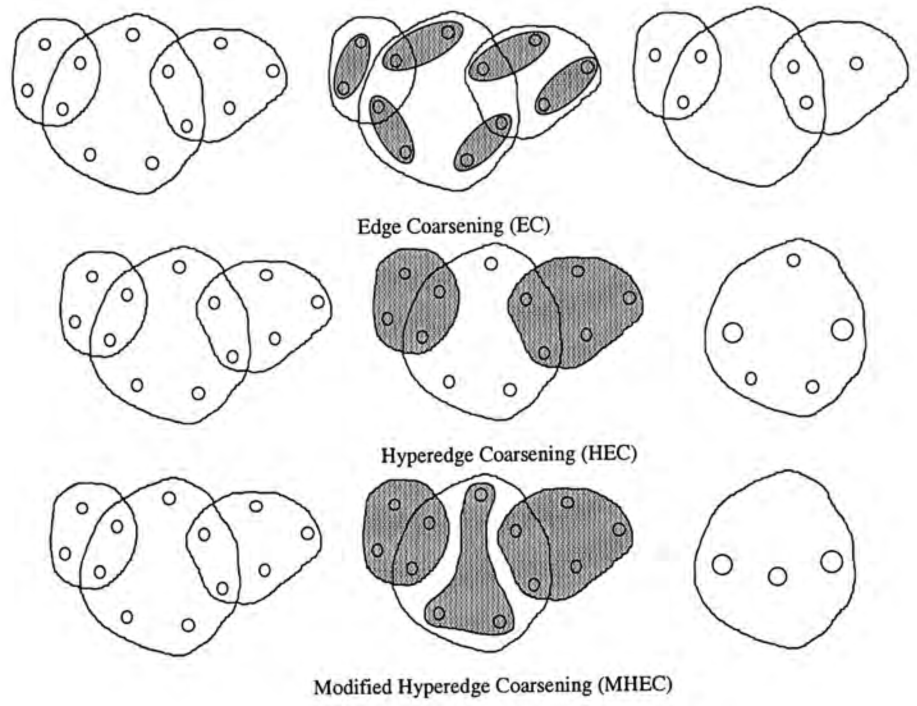

Edge Coarsening (EC)
Hyperedge Coarsening (HEC)
Modified Hyperedge Coarsening (MHEC)

Figure 4.10: Various hyperedge coarsening schemes

Initial Partitioning Stage

Two algorithms are used for computing the initial partitioning. The first one simply creates a random bisection which roughly partition into two parts that have roughly the same vertex weight. The second one starts from a randomly selected vertex and grows a region around it in a breadth-first fashion until half of the vertices are in this region. Since both algorithms are randomized, 10 initial partitionings at the coarsest graph are obtained. The k-way initial partitioning is computed by the multilevel hypergraph bisection algorithm.

Un-coarsening and Refinement Stage

During un-coarsening phase, the coarser hypergraph is projected to the next level finer hypergraph, and a partitioning refinement is used. Two different partitioning refinement algorithms are used. First one is FM which repeatedly moves vertices between partitions to improve the cut. The second algorithm is Hyperedge Refinement (HER) which moves groups of vertices between partitions so that an entire hyperedge is removed from the cut.

4.3 The GBAW Partitioning Algorithm

Assume that one pin is used in a partition for a net. The objective of a multi-way partition is essentially to minimize the number of pins required to connect all partitions. Since some of the wires may have alternative wires, by replacing some cut wires by their alternative wires that are not cut wires, cut size can be reduced. The rewiring process may lead to some new circuit graph, and in turn help escaping from local minima led by graph domain partitioning process.

A perturbation refers to the replacement of a target wire by its alternative wires. Figure 4.11 illustrates the gains regarding various perturbations in a circuit. In the figure, thick lines represent the target wires and dotted lines refer to their alternative wires. As shown in the example, there are positive, zero and negative gains.

The hMetis-Kway [KK99] partitioning tool is used to provide a fast and near optimum solution which serves as the initial partition. The well-known FM partitioning algorithm [FM82] is selected as the graph domain partitioner for its simplicity and efficiency. In fact, any other graph domain partitioner can integrate into GP. Then, the rewiring technique GBAW aiming for further improvements is applied to perform logic perturbations which include:

- substituting a wire with its alternative wire.
- adding a gate and removing some wires.
- adding one wire to remove other wires.
- adding two gates to remove other wires and so on.

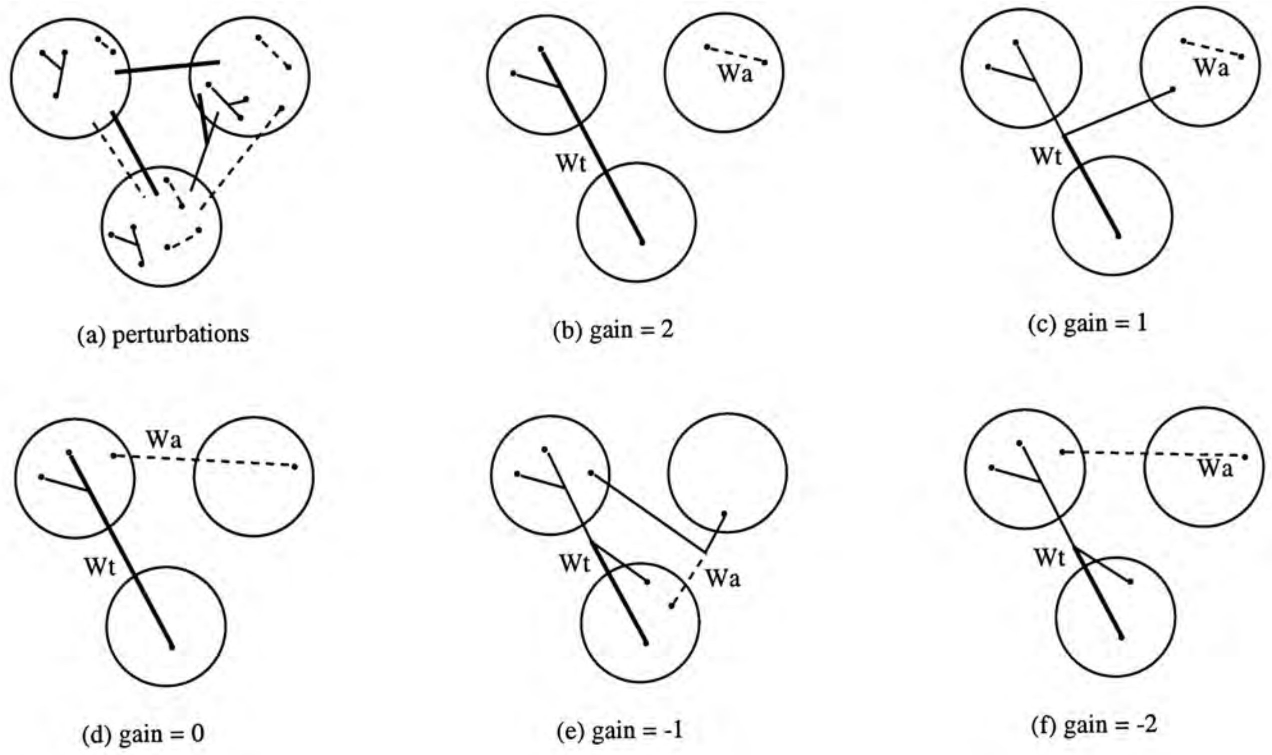


Figure 4.11: Perturbations and gains

Figure 4.13 gives the algorithm of GP. The graph domain information of the benchmark circuits are first extracted as shown in Figure 4.12.

During the perturbation process GP, only cut wires will be selected as target wires for perturbations. GP first randomly select a cut wire as the target wire. Then, GBAW is used to find the alternative wire set SWa of the target wire. Finally, among the wire set SWa, the alternative wire with the highest gain is selected for perturbation. When the SWa of the target cut wire is empty, GP may randomly select another cut wire for another trial. The number of iterations is set by m. The number of trials is limited by t times. k is the limit of perturbations. These limits serve to set some bounds for some useless runs when the total number of alternative wires of all cut wires is zero or very small. The main difference of algorithm GP and the algorithm in [CLMS95] lies in the condition of perturbations. In [CLMS95], a perturbation is performed only when the alternative wire of the selected cut wire has a non-negative gain.

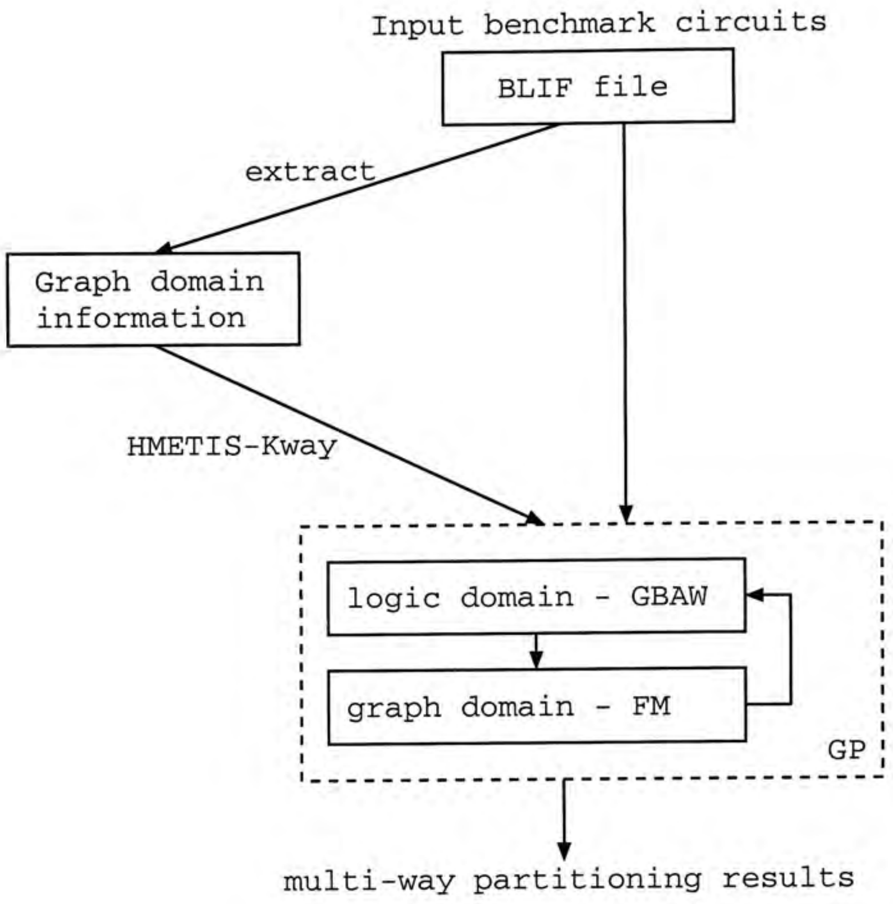


Figure 4.12: Extract the Graph domain information for hMETIS-kway

However, in the experiments, some (hill-climbing) perturbations are allowed therefore can increases the chance of obtaining better solutions. A negative-gain perturbation is also allowed to help escaping some local minimums. On the other hand, the main difference between GP and the algorithm in [WYC00] is the perturbation engine used. In [WYC00], a coupling scheme by RAMBO and GBAW (RG) is used. Since RAMBO is limited to 2-input gates circuit, RG is unable to handle multi-input gate circuits. By only integrating GBAW to GP, the partitioner can effectively locate nearly all the alternative wires of the circuit.

4.4 Experimental Results

The algorithm GBAW-Partitioner (GP) was implemented in C and the experiments were conducted on Sun Enterprise E4500 workstation with 8 GB memory in a single-processor configuration for circuits of various sizes from MCNC benchmarks. The *5xp1* benchmark circuit from MCNC is put at Appendix. The large benchmark circuits used in ISPD98 [AHK97a] are not applicable for the experiments due to the lack of logical domain information. Since the rewiring engine GBAW [WLF00] is able to locate alternative wires of multiple input gates as well as 2-input gates, thus the circuit SIS [SSLea92] simplification done by [CLMS95, WYC00] can be skipped.

Here an example of how GP works below is shown. Taking *C3540* as an example, the circuit is firstly partitioned by hMetis-Kway and has a initial hypergraph with cost 132. The algorithm of GP then further cut down the cost to 112 with the following steps.

- GP searches all the alternative wires of the wires which lie along the cut line and replace them, and the original graph is changed with gain 5.
- With logically equivalent but different graph, FM easily reduced the cost down to 122.
- Random perturbation of the graph brings another reduction by 2.
- Again, GP searches along the cut line and reduces the total cost by 5.
- By switching between graph and logic domain, the cost is reduced to 112 which is the final result.

Table 4.1 lists the statistics of alternative wires on the benchmark circuits. Column "Circuit" shows the name of the circuit. Column "alt. wire" lists the number of alternative wires found in the circuits. Column "CPU" refers to the runtime in seconds. From the experimental results listed in Table 4.1, it shows that GBAW can find 30% more alternative wires than RAMBO on average with 75 times speedup. The GBAW which include many newly found 2-local patterns is now able to locate 98% of the 2-local alternative wires found by RAMBO. Therefore, with the enhanced GBAW the unnecessary redundancy check by RAMBO can be skipped.

Circuit	RAMBO		GBAW	
	alt. wires	CPU	alt. wires	CPU
5xpl	36	10.17	62	0.24
9sym-hdl	27	1.56	40	0.16
C1355	185	12.82	250	0.89
C1908	127	33.52	240	0.68
C2670	267	83.57	344	1.33
C3540	569	273.80	816	2.15
C432	129	10.26	188	0.37
C499	16	6.05	34	0.6
C5315	511	155.91	713	2.88
C6288	1352	361.18	2191	4.18
C7552	1709	143.95	617	4.2
C880	151	9.86	239	0.66
alu2	169	214.71	263	0.84
alu4	333	270.50	493	1.61
apex6	239	34.32	377	1.23
b9_n2	48	1.65	71	0.17
comp	57	9.18	58	0.21
des	1468	729.92	2204	8.7
duke2	157	46.55	281	0.63
f51m	49	6.19	65	0.25
misex	216	124.48	439	0.97
my_adder	46	1.16	0	0.23
pcler8	29	1.3	30	0.12
rot	243	48.04	406	1.1
sao2-hdl	104	16.86	153	0.39
terml	106	16.81	169	0.37
ttt2	68	9.68	133	0.34
x3	228	23.13	348	1.2
Total	8639	2657.13	11224	36.7
Normalized	1	1	1.2992	0.0138

Table 4.1: Alternative wire statistics of RAMBO and GBAW

In the experiments, the tolerance of area imbalance of GP is set to be $\pm 20\%$ of the average area in each partitioned block. Therefore the maximal area ratios are 40%:60% and 16%:24% for 2-way and 5-way partitionings respectively. In order to explore the graph domain optimization, hMetis-Kway [KK99] was firstly run for each circuit. As a result, a nearly optimum partition solution was obtained. The next step is to select the best solution applying GP for logic perturbation to further improve the quality of the partitioning with $k = 60$ and $t = 50$. Table 4.2 to 4.5 list the experimental results for the 2- to 5-way partitionings respectively. Column "area" lists the area of the sub-circuit in terms of the number of gates. Column "#lits" lists the total number of literals of the partitioned circuits. From the results, the area penalties for 2- to 5-way are 0.33%, 0.53%, 0.61% and 0.71% respectively. Column "cut cost" lists the total number of cut pins obtained for all partitioned blocks. Column "cut wire" lists the number of cut wires of the partitioning. Column "cpu" lists the cpu time (in seconds). From the results, they showed that applying logic perturbation can further cut down the cut size of the good partitionings produced by purely graph domain based partitioner. The total number of literals is slightly increased because of the area cost of the added gates during perturbations. 14.48%, 10.18%, 9.08% and 9.24% reduction in cut size are obtained for the 2-, 3-, 4- and 5-way partitionings respectively. The last 2 columns show that the quality and CPU of GP are both much better than the results obtained by simply running FM for 250 times.

Circuit	hMetis-Kway					GP						250 FM	
	area	#lits	cut cost	cut wire	cpu	area	#lits	cut cost	cut wire	cpu	cut cost	cpu
5xp1	61:71	235	30	31	0.33	73:63	239	28	39	34	28	38
9sym-hdl	67:74	232	16	8	0.36	64:77	232	10	8	14	10	42
C1355	331:269	1055	44	35	0.65	338:266	1059	36	35	74	46	266
C1908	260:256	883	82	55	0.64	271:251	889	62	55	61	62	238
C2670	516:527	1444	42	23	0.78	517:531	1449	34	23	112	126	427
C3540	615:648	2267	132	93	1.77	614:663	2280	112	100	357	180	692
C432	119:119	392	44	27	0.46	118:130	402	36	27	33	28	78
C499	231:272	854	46	47	0.46	232:272	855	36	47	59	54	214
C5315	918:1044	3282	104	71	1.98	919:1047	3286	100	69	493	234	1232
C6288	1297:1559	5195	80	270	2.33	1309:1561	5209	78	189	960	430	1708
C7552	1281:1141	4105	18	65	1.93	1286:1142	4111	18	64	727	228	1731
C880	261:222	780	54	30	0.57	260:234	791	38	30	58	38	200
alu2	190:232	777	84	93	0.81	190:236	781	80	95	115	88	150
alu4	428:357	1470	140	106	1.16	438:360	1481	120	104	223	160	322
apex6	435:473	1417	18	11	0.79	436:473	1418	16	11	109	36	370
b9_n2	87:70	208	16	13	0.32	64:93	208	12	10	16	10	41
comp	93:91	270	6	4	0.32	93:90	269	6	3	35	6	51
des	1727:2112	6655	236	221	3.99	1565:2282	6663	146	331	752	332	2338
duke2	175:211	676	80	61	0.7	162:233	684	74	72	98	82	128
f51m	72:65	244	28	32	0.37	77:65	247	26	33	35	26	41
misex3	293:245	990	80	69	0.88	301:250	1003	76	70	149	76	198
my_adder	106:106	339	4	2	0.28	107:105	339	2	2	22	2	72
pcler8	58:72	174	8	14	0.27	58:72	174	8	14	22	8	31
rot	441:383	1251	54	38	0.80	442:384	1253	46	38	95	66	322
sao2-hdl	136:114	439	26	16	0.47	128:123	440	16	16	31	16	89
term1	124:148	439	28	14	0.53	130:148	445	24	14	62	28	88
ttt2	120:107	376	10	11	0.39	102:125	376	8	11	27	8	72
x3	471:384	1334	22	16	0.84	461:396	1336	20	16	4	40	321
too_large	2161:1913	7723	318	563	5.58	2162:1917	7728	308	544	699	1210	1995
Total Average		45506	1850				45656 +0.33%	1576 -14.48%		5476	3658	13495

Table 4.2: 2-way partitioning results of hMetis-Kway & FM & GP

4.5 Conclusions

In this chapter, a scheme coupling the graph logic domain partitioners to explore a larger optimization room of circuit partitioning is proposed. The scheme is shown to be very efficient in terms of CPU expenditure and is also

Circuit	hMetis-Kway					GP					250 FM	
	area	#lits	cut cost	cut wire	cpu	area	#lits	cut cost	cut wire	cpu	cut cost	cpu
5xp1	37:43:52	235	53	51	0.67	40:47:50	240	49	50	36	45	50
9sym-hdl	43:44:54	232	32	19	0.67	43:44:55	233	22	19	15	20	49
C1355	217:196:187	1055	84	95	1.19	217:204:189	1065	80	93	157	143	269
C1908	148:186:182	883	115	82	1.10	146:191:185	889	98	98	127	134	226
C2670	400:329:314	1444	85	59	1.34	402:337:312	1452	71	59	119	247	411
C3540	376:464:423	2267	175	172	2.88	361:506:413	2284	165	224	357	413	664
C432	67:92:79	392	56	46	0.67	72:95:83	404	54	70	67	50	91
C499	193:169:141	854	82	64	0.84	195:173:136	854	64	64	62	108	188
C5315	556:692:714	3282	117	114	3.31	521:709:739	3289	100	122	510	564	1125
C6288	810:1086:960	5195	148	429	3.98	827:1061:978	5198	165	555	704	712	1697
C7552	745:923:754	4105	94	129	3.42	750:941:739	4113	86	136	747	490	1271
C880	139:178:166	780	80	58	1.06	145:182:163	787	61	58	60	103	202
alu2	124:133:165	777	140	144	1.23	134:143:167	795	127	159	122	152	178
alu4	230:307:248	1470	214	203	2.06	218:314:268	1489	194	211	231	283	361
apex6	257:302:349	1417	75	47	1.59	243:321:358	1431	63	70	223	113	447
b9_n2	45:53:59	208	21	17	0.51	44:54:60	209	21	17	32	25	52
comp	57:62:65	270	16	8	0.59	57:62:66	271	14	7	37	20	56
des	1107:1512:1220	6655	322	342	7.06	1089:1535:1216	6656	236	628	776	657	2761
duke2	113:147:126	676	128	107	1.18	106:143:154	693	116	112	103	121	162
f51m	42:53:42	244	58	54	0.68	46:54:44	252	56	62	37	49	51
misex3	153:209:176	990	128	114	1.50	154:208:192	1006	113	162	157	139	252
my_adder	80:65:67	339	8	4	0.60	79:65:68	339	4	4	23	10	72
pcler8	41:49:40	174	17	20	0.47	42:49:39	174	15	20	25	15	37
rot	245:268:311	1251	85	68	1.44	233:272:330	1262	77	77	195	120	368
sao2-hdl	95:83:72	439	62	46	0.86	95:94:68	446	56	57	65	56	112
term1	77:88:107	439	54	38	0.88	78:91:108	443	44	38	32	54	111
ttt2	64:89:74	376	33	34	0.78	66:90:75	380	33	34	55	37	86
x3	271:262:322	1334	78	65	1.38	276:259:340	1354	67	65	106	130	419
too_large	1562:1198:1314	7723	779	976	10.46	1606:1201:1284	7740	747	1096	734	1761	2602
Total Average		45506	3339				45748 +0.53%	2999 -10.18%		5914	6761	14370

Table 4.3: 3-way partitioning results of hMetis-Kway & FM & GP

quite capable in bringing further improvements on good partition results produced by well-known partitioner hMetis-Kway. Without the integration with RAMBO, the input circuits is no longer limited to 2-input simple gate circuits. Experiments on 29 MCNC benchmark circuits for 2- to 5-way partitionings are conducted, and obtained further cutsize reductions from 14.48% to 9.24% upon the good results produced by hMetis-Kway. Moreover, the partitioning quality and CPU expenditure of GP are both better than running FM for 250 times. As GP can be integrated with any newly developed powerful graph partitioner, this partitioning scheme should be very practical and useful for many partition tasks.

Circuit	hMetis-Kway					GP						250 FM	
	area	#lits	cut cost	cut wire	cpu	area	#lits	cut cost	cut wire	cpu		cut cost	cpu
5xp1	34:27:31:40	235	66	63	0.68	38:29:29:39	239	60	78	37		58	52
9sym-hdl	36:38:31:36	232	48	26	0.80	37:40:32:32	232	29	26	16		26	52
C1355	174:158:130:138	1055	110	90	1.59	164:164:125:165	1073	96	108	166		185	305
C1908	122:125:126:143	883	137	114	1.53	126:128:121:149	890	114	126	67		184	259
C2670	262:257:262:262	1444	125	82	1.75	264:251:274:262	1452	91	114	122		274	481
C3540	324:285:356:298	2267	245	211	3.36	337:280:378:281	2278	230	286	385		458	791
C432	65:53:63:57	392	75	62	0.95	72:55:70:56	407	67	73	71		66	107
C499	122:148:116:117	854	92	77	1.26	123:150:112:120	856	81	81	131		145	218
C5315	532:511:460:459	3282	201	152	3.91	537:510:461:461	3289	192	161	523		633	1325
C6288	627:670:856:703	5195	186	582	4.52	627:678:842:716	5200	210	585	482		848	2051
C7552	537:605:647:633	4105	66	153	4.37	511:634:648:638	4114	44	153	388		732	1551
C880	108:113:136:126	780	82	58	1.33	109:123:138:123	790	69	121	124		107	230
alu2	101:128:96:97	777	178	171	1.51	99:126:103:111	794	171	193	128		190	212
alu4	162:197:204:222	1470	285	240	2.35	157:191:224:236	1493	257	308	123		352	452
apex6	233:200:261:214	1417	75	45	1.91	231:209:276:208	1433	65	77	233		139	555
b9_n2	41:48:35:33	208	36	23	0.76	46:43:36:35	209	32	23	17		31	58
comp	48:43:47:46	270	14	8	0.91	47:44:47:46	270	12	8	20		28	62
des	938:1160:947:794	6655	359	515	7.51	889:1144:953:864	6665	295	713	774		716	3322
duke2	116:95:96:79	676	159	127	1.33	113:82:115:89	698	147	137	112		158	183
f51m	37:35:36:29	244	67	67	0.75	41:39:36:28	251	65	68	40		63	55
misex3	157:138:132:111	990	188	177	1.70	160:149:143:107	1006	169	192	166		180	295
my_adder	52:54:52:54	339	12	6	0.81	52:53:52:55	339	6	6	24		23	81
pcler8	30:28:33:39	174	24	25	0.57	31:28:32:39	174	22	25	26		20	40
rot	193:191:208:232	1251	98	75	1.86	187:193:210:240	1256	93	91	205		166	457
sao2-hdl	75:60:63:52	439	93	60	1.08	69:64:73:51	446	76	60	35		74	117
term1	67:56:67:82	439	59	36	1.14	77:57:69:85	454	55	37	68		68	118
ttt2	54:66:59:48	376	51	44	0.85	55:69:60:49	382	46	46	59		48	94
x3	191:193:236:235	1334	78	52	1.86	190:196:249:244	1358	71	111	225		164	504
too_large	862:1059:964:1189	7723	1041	1345	11.58	823:1082:961:1221	7736	999	1685	756		2206	3281
Total Average		45506	4250				45784 +0.61%	3864 -9.08%		5523		8342	17308

Table 4.4: 4-way partitioning results of hMetis-Kway & FM & GP

Circuit	hMetis-Kway					GP						250 FM	
	area	#lits	cut cost	cut wire	cpu	area	#lits	cut cost	cut wire	cpu		cut cost	cpu
5xp1	26:21:23:31:31	235	78	71	0.89	23:22:32:29:31	237	73	77	37		69	60
9sym-hdl	32:30:24:24:31	232	57	31	0.96	32:27:28:25:30	233	33	31	16		35	55
C1355	142:113:100:130:115	1055	122	97	1.95	142:115:95:144:123	1074	109	110	175		222	369
C1908	91:111:93:117:104	883	153	111	1.70	84:118:84:123:112	888	119	131	72		206	313
C2670	221:221:174:207:220	1444	133	95	2.26	186:229:195:217:229	1457	106	135	129		314	578
C3540	214:258:223:307:261	2267	312	268	4.07	224:239:237:301:284	2288	265	341	206		551	990
C432	42:46:43:59:48	392	80	74	1.25	47:48:41:57:57	404	77	88	75		73	126
C499	107:104:85:103:104	854	107	95	1.45	111:101:81:108:105	857	93	102	138		173	245
C5315	371:315:370:497:409	3282	244	193	4.92	372:315:383:471:428	3289	232	268	558		706	1650
C6288	517:496:525:651:667	5195	222	601	5.21	508:498:532:659:669	5204	241	601	820		876	2340
C7552	568:504:396:533:421	4105	152	169	4.84	554:528:388:504:447	4095	135	220	787		836	1901
C880	104:95:82:111:91	780	104	84	1.58	107:91:86:107:100	787	94	157	131		155	293
alu2	71:78:79:95:99	777	209	198	1.92	67:93:88:89:99	791	192	228	130		227	245
alu4	150:127:143:183:182	1470	310	291	3.05	147:130:171:181:173	1487	279	375	129		399	554
apex6	205:205:148:157:193	1417	132	91	2.39	206:206:199:145:171	1436	125	112	249		165	656
b9_n2	25:30:28:39:35	208	36	27	0.84	24:29:33:39:39	214	36	32	36		39	65
comp	39:42:30:41:32	270	24	15	0.97	30:42:31:42:38	269	20	15	21		36	73
des	682:711:683:919:844	6655	484	382	10.03	667:699:667:954:889	6692	386	564	785		689	4246
duke2	73:70:71:94:78	676	175	148	1.69	88:77:65:89:89	696	164	168	115		183	217
f51m	27:21:24:35:30	244	81	78	0.93	32:30:21:27:32	249	77	86	39		76	59
misex3	86:103:111:133:105	990	216	215	2.23	95:96:127:107:129	1006	203	276	173		240	364
my_adder	52:41:41:39:39	339	16	8	1.12	51:41:42:39:39	339	10	8	26		30	87
pcler8	26:22:24:28:30	174	29	29	0.68	27:25:25:24:32	177	26	30	28		27	48
rot	145:147:150:209:173	1251	127	97	2.20	160:151:142:197:190	1266	121	123	217		184	543
sao2-hdl	63:51:45:43:48	439	109	74	1.46	61:60:46:44:48	448	97	93	66		93	145
term1	66:55:48:56:47	439	71	44	1.34	66:58:45:65:55	453	66	59	71		79	146
ttt2	46:55:35:45:46	376	59	58	1.15	47:53:36:51:45	383	56	72	62		65	112
x3	193:190:136:152:184	1334	122	91	2.23	193:195:170:139:199	1375	112	171	243		173	605
too_large	1003:851:661:782:777	7723	1221	1477	14.62	975:879:723:765:743	7734	1159	1806	802		2503	4478
Total Average		45506	5185				45828 +0.71%	4706 -9.24%		6309		9324	21563

Table 4.5: 5-way partitioning results of hMetis-Kway & FM & GP

```
Algorithm GP (best_partition, m, k, t){
   search_limit = 0;
   n_perturbations = 0;
   curr_partition = best_partition;
   last_partition = best_partition;
   for i=1 to m {
     while((n_perturbations < k) && (exit == false)){
       search_limit = 0;
       while(search_limit < t){
           search_limit ++;
           randomly select a cut wire $W_t$;
           use GBAW to find all alternative wires $SW_a$ for $W_t$;
           if ($SW_a$ == $\phi$){
               search_limit ++;
               continue;
           }else
               break;
       }
       if ($SW_a$ = $\phi$){
           pick alternative wire $W_1$ with the largest gain;
           replace $W_t$ with $W_1$ in curr_partition.
           curr_partition = FM(curr_partition);
           n_perturbations = n_perturbation + 1;
           if (cost(curr_partition) < cost(last_partition))
               last_partition = curr_partition;
       }
       for each wire $w${
           use GBAW to find all alternative wires $SW_a$ for $w$;
           do random perturbation on $SW_a$ in curr_partition;
       }
       curr_partition = FM(curr_partition);
       if (cost(curr_partition) < cost(last_partition))
           last_partition = curr_partition;
     }
   }
}
```

Figure 4.13: Algorithm of GBAW-Partitioner (GP)

Chapter 5

Optimum FPGA Switch-Box Designs - HUSB

5.1 Introduction

Field Programmable Gate Array (FPGA), a kind of Very Large Scale Integrated (VLSI) circuit, consists of an array of pre-fabricated functional blocks and wire segments with user-programmability of logic and routing resources. Because of their fast turn-around time and economic manufacturing cost for low volume designs, FPGAs have been used in a great amount of digital equipments. FPGA technologies are commonly classified into three major categories: (1) Look-Up-Table (LUT), SRAM based (2) multiplexer, channel organized and anti-fused, and (3) LPLD, EPROM based. In this chapter, the study of the optimum routing structure problems for the popular LUT and SRAM based two-dimensional (2-D) FPGAs is presented. The architecture of an industrial product of this type is described in [AR95, alt90, BFRV92, CWW96].

The importance of routing resource issues in FPGAs is never over-emphasized. In commercial FPGA products, the routing resource consumes most of the chip area, and is responsible to most of the circuit delay. A typical $2-D$ FPGA architecture is shown in Figure 5.1. The functional blocks (or logic cells) are marked by L, which are separated by vertical and horizontal channels. There

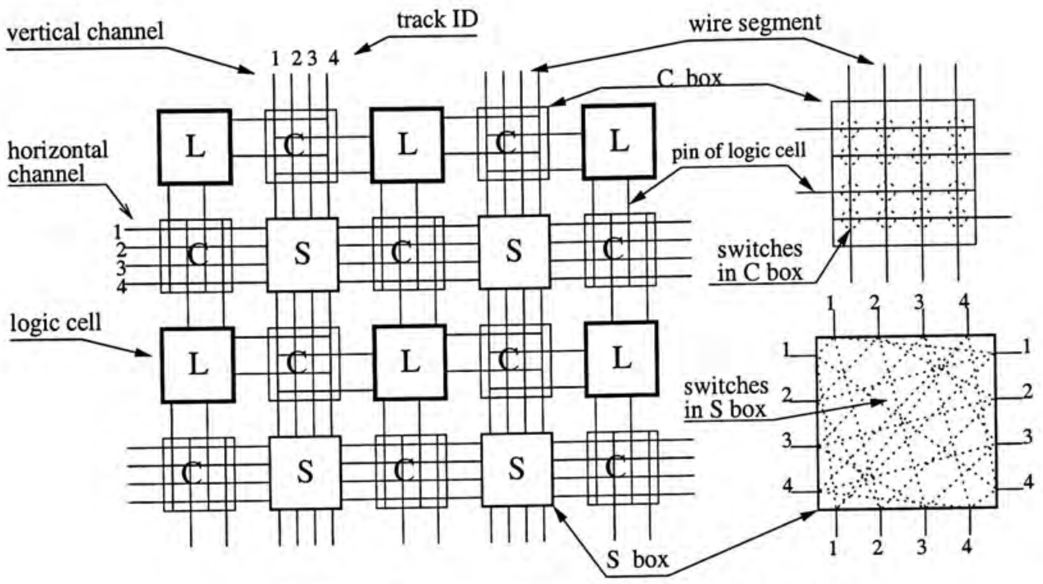


Figure 5.1: The architecture of a 2D FPGA.

are W (called channel density) prefabricated parallel wire segments running between each pair of adjacent L-cells in both vertical and horizontal channels. The wire segments in a vertical (or horizontal) channel are aligned into W vertical (or horizontal) tracks; each track within a channel is assigned an integer in $\{1, \ldots, W\}$ as its track ID. There are C-boxes in the channel between adjacent L-cells. A Switch Box (S-box), located at each intersection of a vertical and horizontal channels, contains programmable switches to connect wire segments running from its surrounding C-boxes.

When an FPGA is used to realize a specified Boolean function, the pins used to realize the Boolean function are partitioned into groups (called nets). Then the pins in each group are connected together to form a real net by using available wire segments and switches in both C-boxes and S-boxes; different nets are disconnected. The latter process is referred to as a routing. Conventionally, the routing process is divided into two subsequent steps, global routing and detailed routing, although there is no absolute need for doing routing in these two phases. In this chapter, the term of global routing is used to specify the connection topologies for all nets. The detailed routing decides the exact assignment of wire segments and switches used to materialize the complete routing. As the connectivity within C-boxes is complete, the routability of the entire chip is fully dependent on the structure and connectivity of the S-boxes [AR95, BFRV92, CWW96, LW95, PWWY92, WC94, WMS97, WTMS94, WTMS96]. As the routing resource is relatively expensive in FPGA chips, it is clearly desirable to design switch boxes (S-boxes) with maximized routability and minimum number of programmable routing switches.

An FPGA switch box is said to be hyper-universal if detailed routing can be performed on all possible multi-pin net topologies which satisfy the global routing density constraints. The switch-box is optimum if it is hyper-universal and the switches inside is minimum.

It has been shown that if the net topology is restricted to 2-pin nets, then a 2-D (4-way) switch box can be built to be universal with only 6W switches, where W is the global routing channel density. As the routing resource is relatively expensive in FPGA chips, searching for optimum switch box designs is clearly a topic with both theoretical challenges and immense commercial silicon reduction values. In [FLW00], a formal mathematical model of this optimum switch box design problem for arbitrary dimensions has been constructed. It gave a scheme to produce hyper-universal designs with less than $6.7W$ switches for 4-way FPGA switch boxes. In this chapter, further investigation of this most common 4-way switch box case is done, and give new theoretical results followed by extensive experimental justifications. The results seem to be quite attractive and show that such an optimum switch box can be built with a very low number of additional switches beyond 6W for today's practical range of low W's. If the number of switches of an optimum 4-way switch box with a density of W will be presented as $e(4, W)$, then $e(4, W) =$ 6W for W being 2, 3, or 5; $e(4, W) =$ 6W + 1 for W = 4; $e(4, 6) <=$ 6W + 2, and $e(4, 7) <=$ 6W + 1. For arbitrary large W's, the bound can be shown to be under $6.34W$. To make an even experimental comparison, VPR [BR97] which is today's best published FPGA router is selected for large benchmarks on the popular Disjoint FPGA switch box structure and the proposed designs. The results are quite encouraging.

This chapter is organized as follows. Section 5.2 gives definitions of routing architectures. Section 5.3 introduces different FPGA routing algorithms and makes comparison. The switch-box analysis is described in Section 5.4. For completeness, some basis previous results on S-box designs, a decomposition property of global routings and terminology will also be mentioned in Section 5.5. In Section 5.6, new results for optimum or hyper-universal S-boxes are provided. Section 5.7 shows the experimental results and conclusion is drawn in Section 5.8.

5.2 Background and Definitions

5.2.1 Routing Architectures

The routing resources of FPGA consist of programmable components and wire segments. Since the number of switches in an FPGA is proportionally increase with the area and delay penalty, there is basically a trade-off between routability and the performance for routing modules in FPGA. There are two main modules, switch-module and connection-module in FPGA.

Switch-block Architecture

In [RB91], the flexibility of the switch-block and connection block are studied. The number of programmable switches between a terminal and others inside a switch-box is represented by F_s as the flexibility of a switch-box. A switch module with flexibility of 3 is shown in Figure 5.2 which means each pin inside the switch-box is connected to three other pins. A large flexibility means a large number of choices. Only switch-box allows the change of channel in the overall routing process.

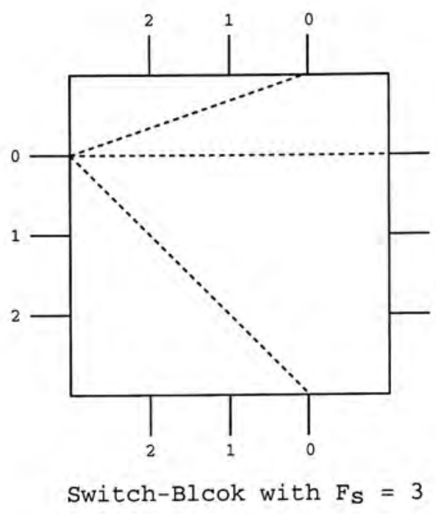


Figure 5.2: A Switch module with $F_s = 3$

Connection-Module Architecture

As shown in Figure 5.3, there are two wire segments from the logic box to the connection box. The connection box allows the connection for each edge on the wire segment. In Figure 5.4 which sourced from [RB91], the percentage of completion to successfully route the *BNRE* circuit is plotted versus the change in F_c on different F_s. It showed that for large switch-box, the increase in F_c does not greatly improve the routability.

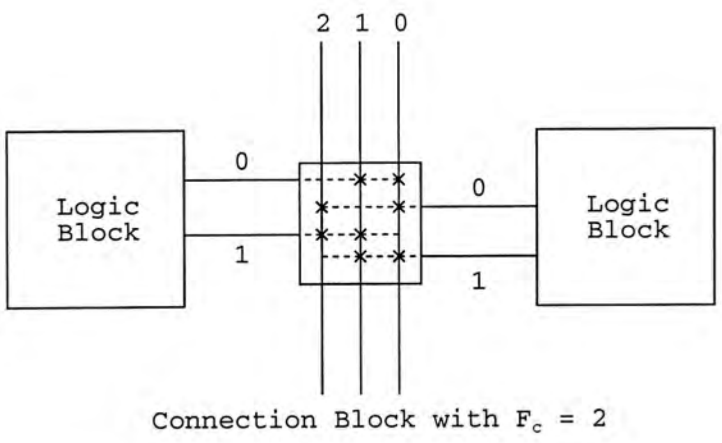


Figure 5.3: A Connection module with $F_c = 2$

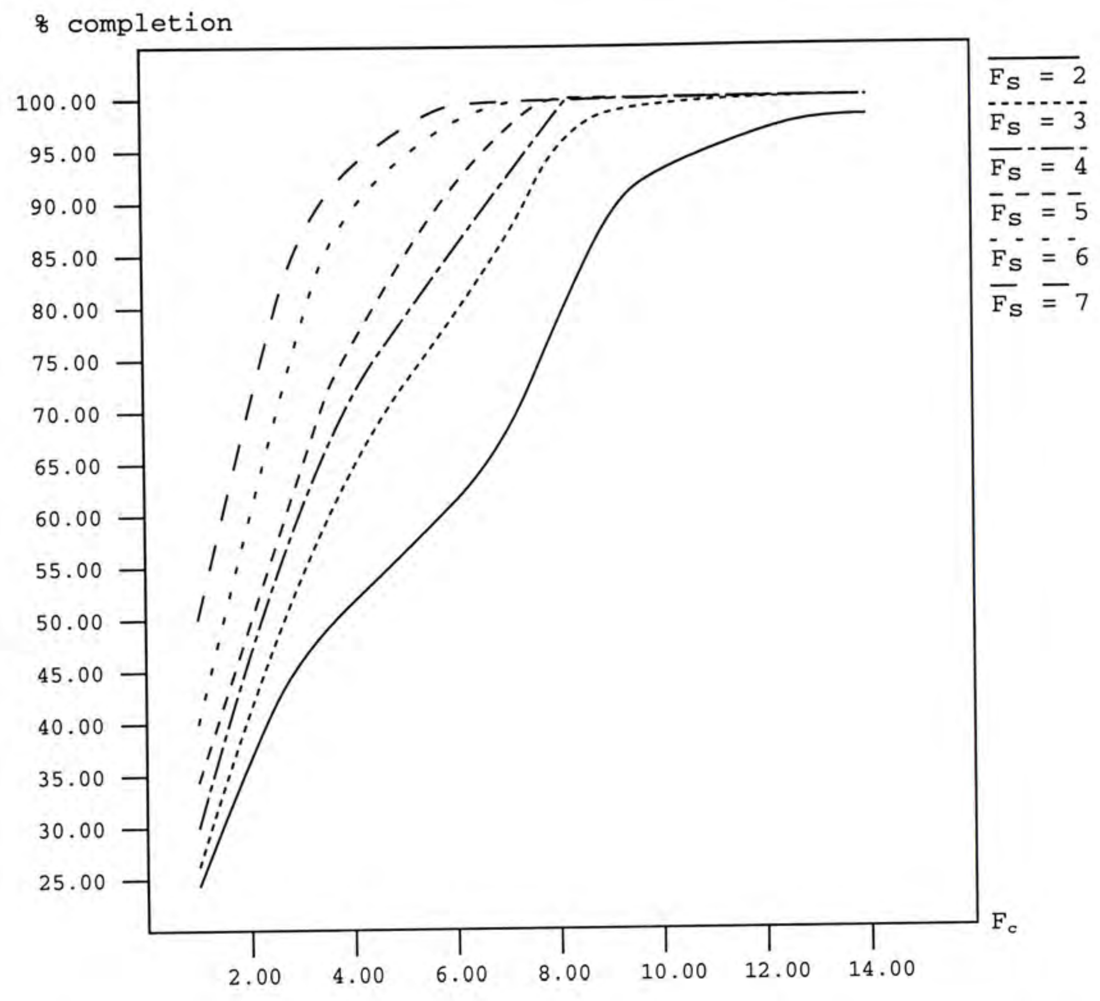


Figure 5.4: Percentage of completion versus F_c for the circuit BNRE

5.2.2 Global Routing

In the phase of global routing, the routing algorithms will generate a "loose" route for each net, that means it is assign the routing regions for each net. For example, in Figure 5.5, there are two possible global route for the connection from the middle left logic box to the right logic box. The global routing algorithm will determine the best or the shortest net without going into the detail of the routing modules.

5.2.3 Detailed Routing

In the phase of detailed routing, the algorithms will find the actual geometric and physical wire for real connection. The exact layout will be produced after

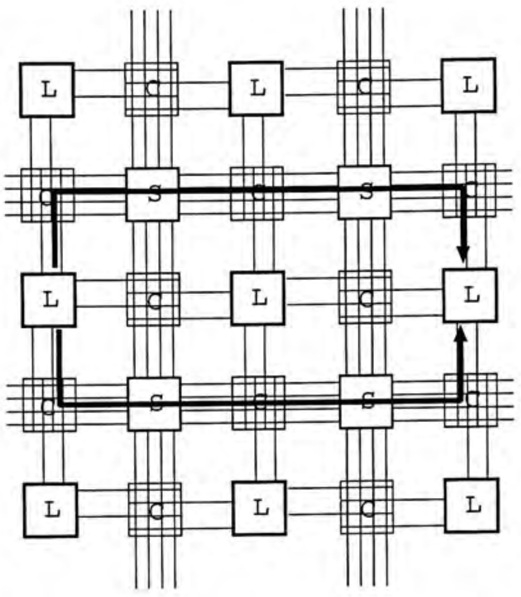


Figure 5.5: Example of Global Routing in FPGA

detailed routing. For example, in Figure 5.6, if the upper global route is selected, then the detailed routing algorithm will go further and determine the connection inside switch.

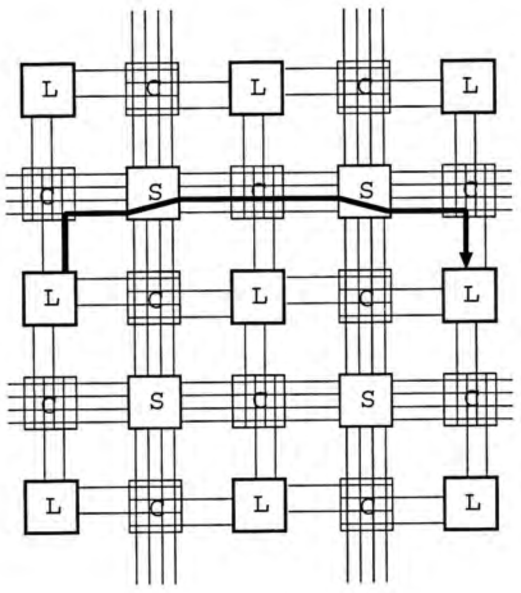


Figure 5.6: Example of Detailed Routing in FPGA

5.3 FPGA Router Comparison

5.3.1 CGE

In [BRV92], a new detailed routing algorithm which called Coarse Graph Expansion (CGE) and designed for FPGA is introduced. The algorithm supports different kinds of FPGA architecture. Since the wire segments in FPGAs are pre-placed, there exists competition between the same wiring segments when applying routing algorithm. Thus, CGE applies 2 phases to ensure the 100% routing on the industrial circuits. It can also optimize the routing delays for some critical paths.

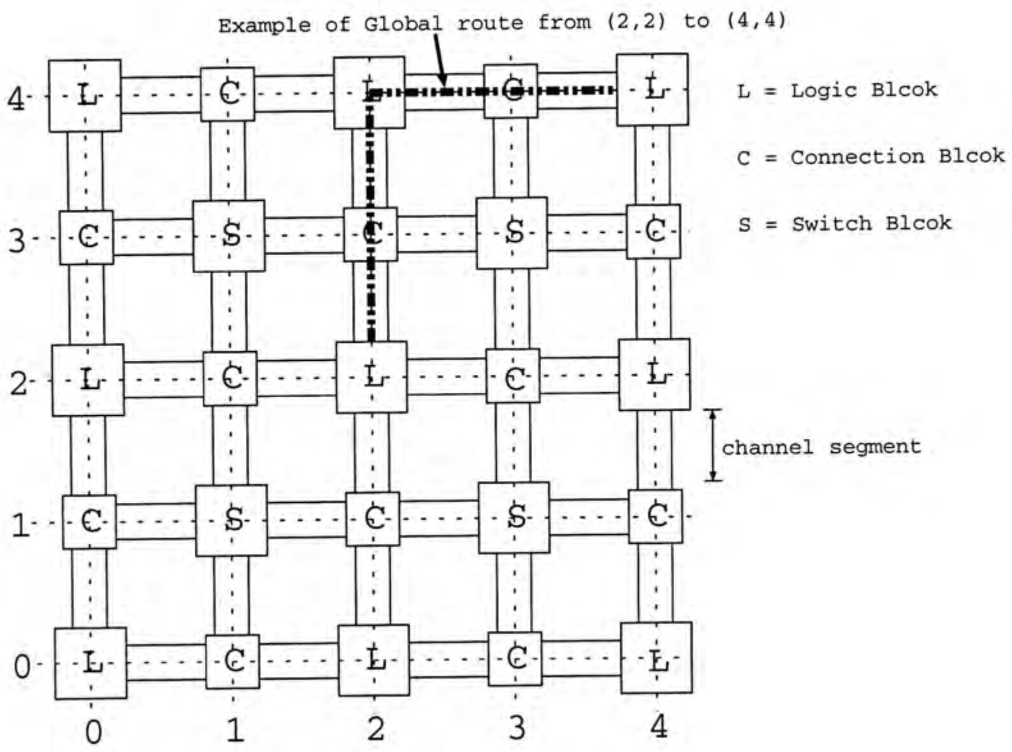


Figure 5.7: CGE: The FPGA Model with one Global Route

The input to CGE detailed routing algorithm is the LocusRoute global router [Ros90] which divides multiple-pin nets into 2-pin nets and routes them in the shortest distance paths. In the first phase of CGE, it generates a number of alternatives for the detailed routing on each global route plus pruning to reduce the number of paths. In second phase, there is a simple cost function which is the sum of the costs of its edges used to evaluate which detailed routing should be chosen.

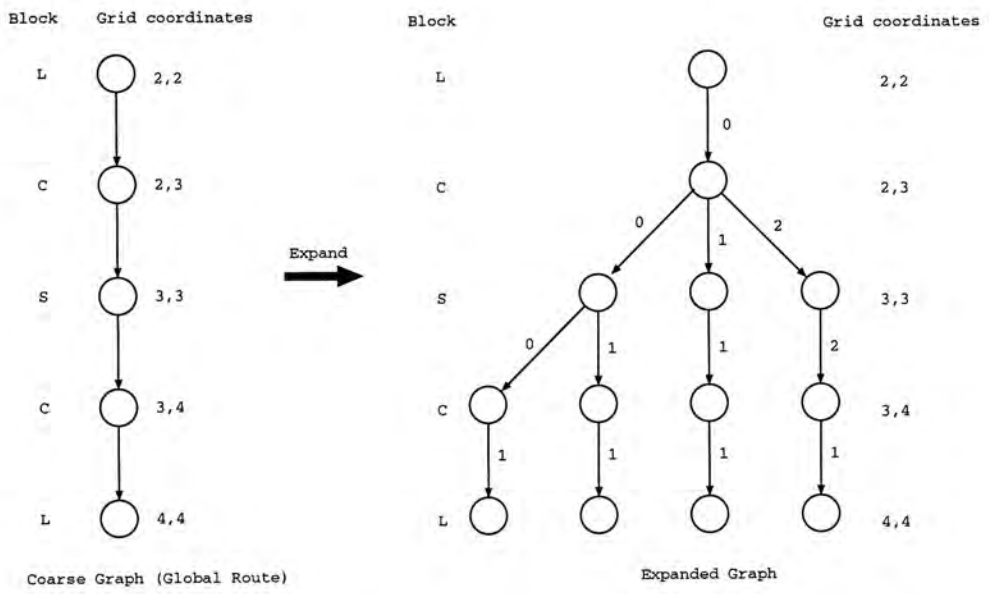


Figure 5.8: CGE: The Coarse Graph and its Expanded Graph

As shown in Figure 5.7, it shows the FPGA model with a global route from grid coordinate (2,2) to (4,4). The coarse graph and its expanded graph are also shown in Figure 5.8 and it is graph representation for the alternatives in phase 1.

5.3.2 SEGA

In [LB93], they address the allocation of wire segments which best match with the length of connection such that SEGA is able to obtain better routing result than CGE. For example, if there is a connection from (0,0) to (4,4) as shown in Figure 5.9, SEGA builds an express edge from (1,0) to (1,3) instead of three short segments as shown in Figure 5.10.

Since SEGA performs the matching between the connections and the wire segments length, it provides a high quality routing result and achieves good performance.

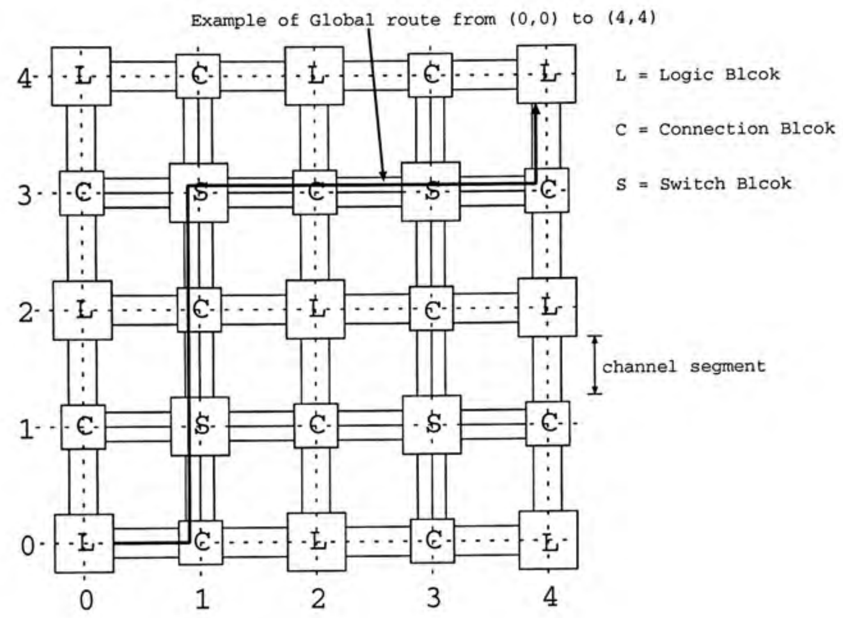


Figure 5.9: SEGA: The FPGA Model with one Global Route

5.3.3 TRACER

In [CLWL95], a RAM-based FPGA routing algorithm (TRACER) is presented. TRACER basically consists of 2 router, an expansion router plus a rip-up and rerouter. The expansion router will use up all the FPGA routing resources without considering the connections of other nets. As a result, the existing violations are solved by rip-up and rerouter.

The important issue for rerouter is to choose which net should be ripped up. The straight forward approach to select bad net might fall into local minima. Therefore TRACER does not choose the net causing a lot of violations to rip-up, but instead there is a score for each net for the selection. The score is shown below where α is set by the user:

$$score(n_i) = \alpha * \frac{actual\_length_i}{estimated\_min\_length_i} + (1-\alpha) * num\_violations_i$$

When TRACER is unable to solved all the violation among different congested nets in a certain time, the router will abort automatically. However, within feasible time, the TRACER will record the feasible solution and find a better solution. TRACER provides a better solution than CGE and SEGA.

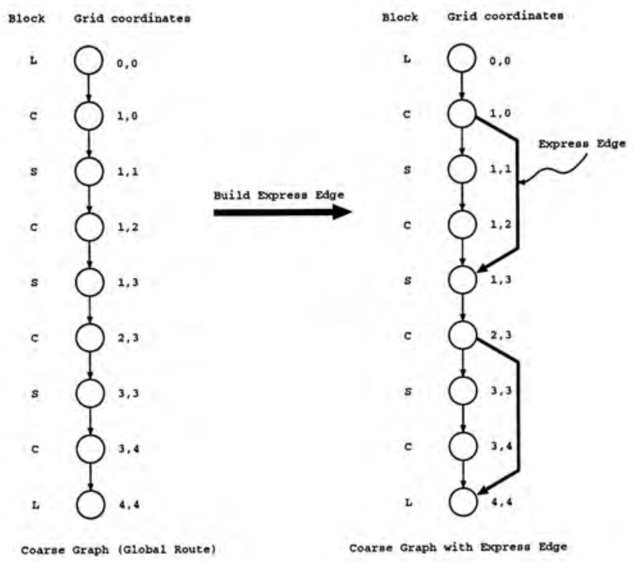


Figure 5.10: SEGA: The Coarse Graph with Express Edge

5.3.4 VPR

Versatile Place and Route (VPR) [BR97] was introduced in 1997 and is known as the best FPGA router nowadays. VPR can place and route an FPGA design into different FPGA architectures which is different from other FPGA router. The routing algorithm of VPR is based on the Pathfinder negotiated congestion algorithm [EMHB95] and it also applies two phases of routing, i.e. shortest path routing (Dijkstra's algorithm, a maze router [Lee61]), and ripping-up and re-routing. In the Table 5.1 shown below, the comparison between different routers with $F_c = W$ and $F_s = 3$ and the switch-box architectures is Disjoint type (XC4000). The inputs are the placement results by Altor [RSV85].

Global Routing	LocusRoute	LocusRoute	VPR	TRACER	VPR
Detailed Routing	CGE	SEGA	SEGA	TRACER	VPR
9symml	9	9	7	6	6
alu2	12	10	8	9	8
alu4	15	13	10	11	9
apex7	13	13	10	8	8
example2	18	17	10	10	9
k2	19	16	14	14	12
term1	10	9	8	7	7
too_large	13	11	10	9	8
vda	14	14	12	11	10
Total	123	112	89	85	77

Table 5.1: Comparison between different FPGA Routers

5.4 Switch Box Design

5.4.1 Disjoint type switch box (XC4000-type)

The most commonly used Switch-box is XC4000-type which is implemented in Xilinx XC4000 family. The architectures are illustrated in Figure 5.11 with $W = 3, 4$. Disjoint means the connection between different track is isolated as shown in Figure 5.12. There are totally 3 independent set of connections.

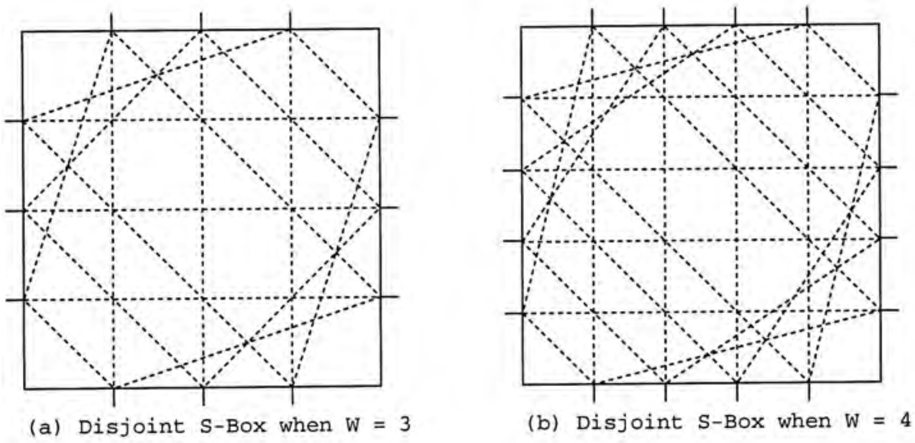


Figure 5.11: XC4000-type Switch-Box Architectures for W = 3 and 4

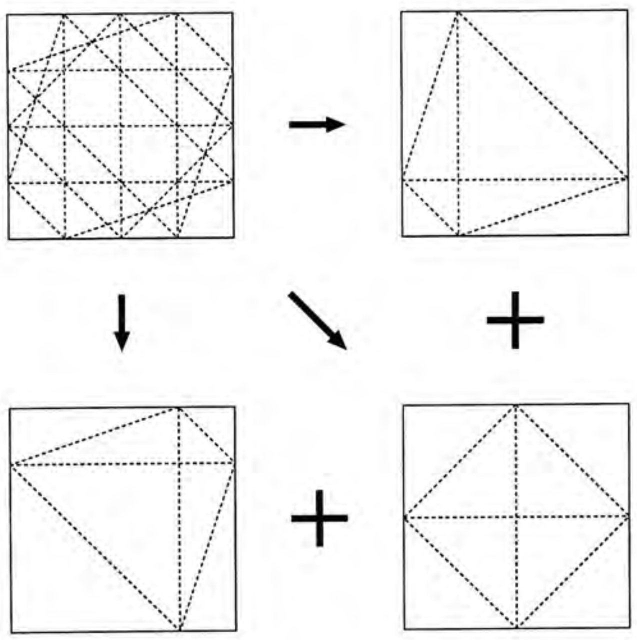


Figure 5.12: Disjoint relationship between different tracks for W = 3

5.4.2 Anti-symmetric switch box

In Figure 5.13, it shows two different size of anti-symmetric switch with $F_s = 3$ which is implemented in [RB91].

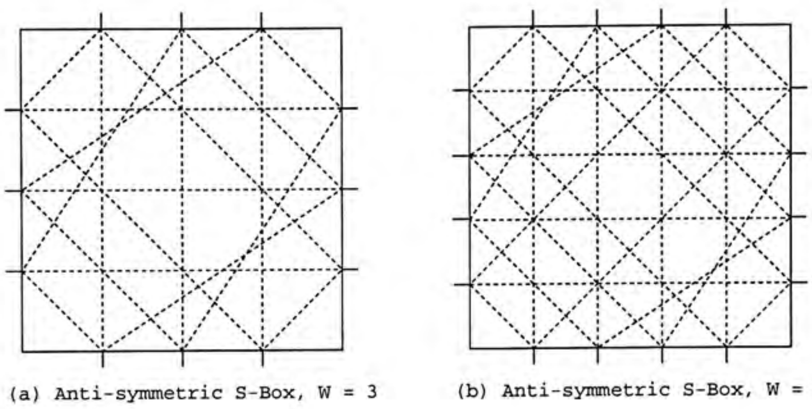

(a) Anti-symmetric S-Box, W = 3
(b) Anti-symmetric S-Box, W = 4

Figure 5.13: Anti-symmetric Switch-Box Architectures for W = 3 and 4

5.4.3 Universal Switch box

In [CWW96] a so called Universal Switch Box (USB) structure, which is a 4-way S-box of density W with $6W$ switches, has been proposed. However, this model only accommodates 2-pin nets, therefore designing a general S-box for all kinds of nets is important and necessary. In Figure 5.14, it shows two different USB architectures.

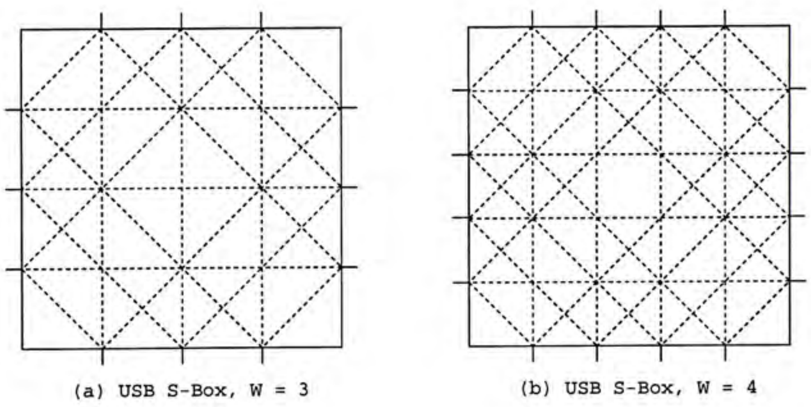


Figure 5.14: Universal Switch-Box Architectures for W = 3 and 4

5.4.4 Switch box Analysis

For 2-D type FPGA, there are six types of connection as depicted in Figure 5.15. If there are two type 1, two type 2, one type 3 and one type 5 connections, the possible routing results are shown in Figure 5.16, 5.17, 5.18 for Disjoint, Anti-symmetric and Universal switch-box respectively.

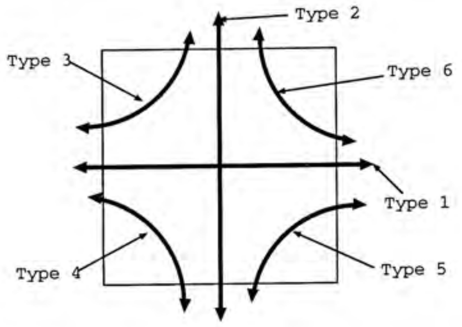


Figure 5.15: Six different types of connections

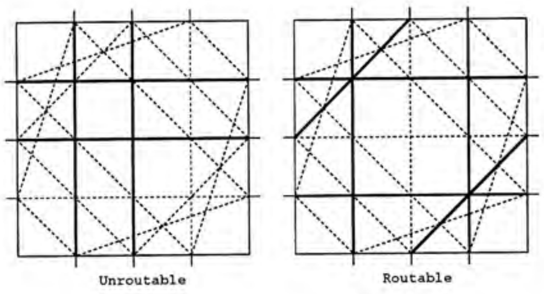


Figure 5.16: Example of routing over Disjoint Switch-box

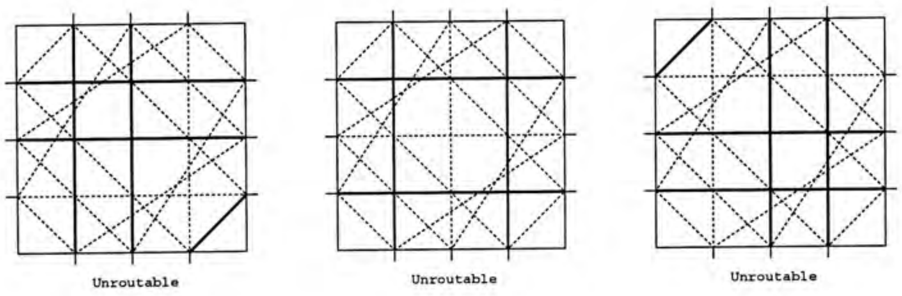


Figure 5.17: Example of routing over Anti-symmetric Switch-box

From the above figures, the required connection is unroutable in anti-symmetric switch-box but is is routable in both Disjoint and Universal switch-box as shown in Figure 5.18. Another example is depicted in Figure 5.19, it shows the routability of USB is higher than Disjoint type because Disjoint S-box failed to route one type 1, one type 3 and one type 6. From [CWW96] the results are put in Table 5.2, USB has higher flexibility than anti-symmetric and disjoint type S-box. A larger routing capacity always means a better routing solution, therefore it is necessary to improve the routability.

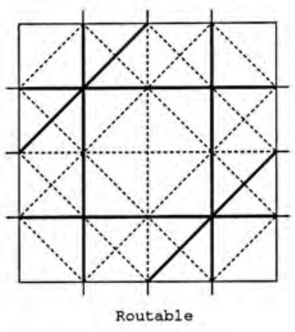


Figure 5.18: Example of routing over Universal Switch-box

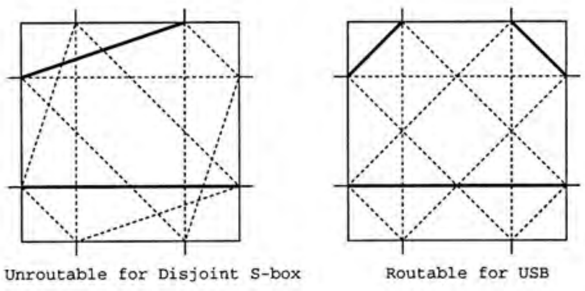


Figure 5.19: Another comparison between Disjoint and Universal Switch-box

Circuit	F_c	USB	Disjoint	Anti-symmetric
BUSC	9	10	11	10
DMA	10	11	14	11
DFSM	10	11	15	11
BNRE	12	12	14	14
Z03	14	14	15	14
Total	-	58	69	60

Table 5.2: Detailed routing result by CGE over different switch-box

5.5 Terminology

The terminology and symbols of graphs are referred to [BM76]. Let $G = (V(G), E(G))$ be a simple graph with vertex set $V(G)$ and edge set $E(G)$. $|V(G)|$ (or $|G|$) and $|E(G)|$ are denoted as the number of vertices and edges in G, respectively. Let $S \subset V(G)$. $G[S]$ denotes the induced subgraph of G by S. $v_{i_1}v_{i_2}\ldots v_{i_l}$ is used to denote the path with consecutive vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_l}$.

Let W and k be positive integers with $k \geq 2$. In [FLW00], a (k, W)-**global routing** is represented as a collection $GR = \{N_i | i = 1, \ldots, l\}$ of non-empty subsets of $\{1, \ldots, k\}$ such that each element of $\{1, 2, \ldots, k\}$ belongs to exactly W subsets of GR. W is called the **density** of the global routing, and a (k, W)-global routing is also called a k-**way global routing with density** W. Each N_i in GR is referred to as a **net** of the global routing. Note that a global routing GR is a multiple set; two equal sets in GR represent two different nets in the global routing. Note also that a net of cardinality n corresponds to an n-pin net. For simplicity, 1-pin nets are allowed to ensure that each element of $\{1, 2, \ldots, k\}$ appears exactly W subsets of GR.

Let GR_1 and GR_2 be global routings and m a positive integer. The union of GR_1 and GR_2 as multiple set is denoted by $GR_1 + GR_2$, and mGR_1 is the union of m GR_1's.

Given a global routing a local and mathematical view, further view is taken to the track with ID j on the i-th side of an S-box as a vertex $v_{i,j}$ and a switch connecting the track with ID j on the i-th side and the track with ID m on the h-th side is an edge $v_{i,j}v_{m,h}$. Therefore, any k-way S-box of density W can be represented as a k-partite graph G on $\cup_{i=1}^{k} V_i$, where $V_i = \{v_{i,j} | j = 1, \ldots, W\}$ and each V_i is an independent set in G for $i = 1, \ldots, k$. The graph G is called as a (k, W)-**design**. In particular, a 4-way S-box of density W is a $(4, W)$-design.

Let G be a (k, W)-design on $(V_1, \ldots, V_k)$. A **detailed** (k, W)-**routing** (or shortly detailed routing) of a (k, W)-global routing $\{N_i | i = 1, \ldots, l\}$ in G is a set of mutually vertex disjoint subgraphs $\{T(N_i)|i = 1 \ldots, l\}$ of G satisfying: (1) $T(N_i)$ is a tree of $|N_i|$ vertices, and (2) $|V_j \cap V(T(N_i))| = 1$ if $j \in N_i$, for $i = 1, \ldots, l$. $T(N_i)$ is called a **detailed routing** of N_i.

Hyper-universal design

A **hyper-universal** (k, W)**-design** on $(V_1, \ldots, V_k)$ is a (k, W)-design on $(V_1, \ldots, V_k)$ such that it contains a detailed routing for each (k, W)-global routing. For example, the complete k-partite graph on $(V_1, \ldots, V_k)$ (in which, there is an edge joining each pair of vertices $v_{i,j}$ and v_{i_1,j_1} with $i_1 \neq i$) is a hyper-universal (k, W)-design. A hyper-universal (k, W)-design represents a k-way S-box of density W (also called a (k, W) **S-box**) which can accommodate any (k, W)-global routings. A (2,4) and a (3,4) design are depicted in Figure 5.20.

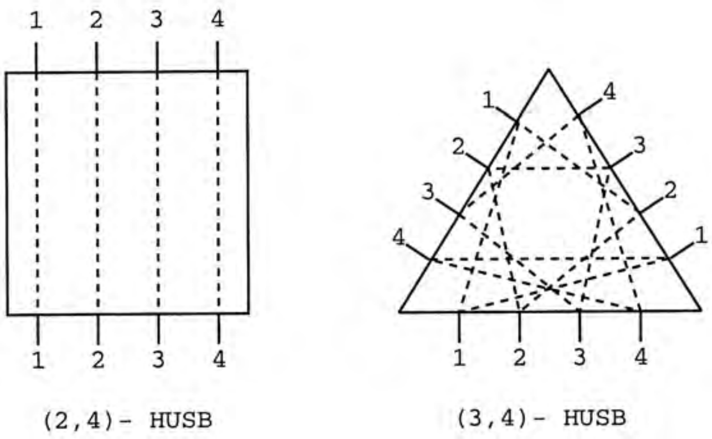


Figure 5.20: (2,4) and (3,4) design

Optimum design

An **optimum** (k, W)**-design** is a hyper-universal (k, W)-design with the minimum number of edges. Clearly, the number of edges in an optimum (k, W)-design is uniquely determined by k and W, which is denoted by $e(k, W)$.

A global routing is called **primitive** if it does not contain two unequal nets of size 1. If a (k, W)-global routing GR is not primitive, then the unequal nets of size 1 are combined into nets of size 2 to obtain a primitive (k, W)-global routing GR'. Any detailed global routing of GR' will induce a detailed global routing of GR by simply deleting the edges of those one edge trees representing the nets of size two in GR' which are obtained by combining the unequal nets of size 1 in GR. Therefore, to verify that a (k, W)-design is hyper-universal, it is only necessary to show that each primitive global routing is detailed routable.

This approach depends on a very nice decomposition property of global routings. Let GR be a (k, W)-global routing and GR' be a sub-collection of GR. If GR' is a (k, n)-global routing with $n < W$, GR' is called a **sub-global routing** of GR. GR is said to be **minimal** if it does not contain subglobal routings. The following result was proved in [FHL].

Lemma 1 For any integer k with $k \geq 2$, there exists an integer $f(k)$ such that any k-way global routing GR could be decomposed into minimal k-way subglobal routings with densities at most $f(k)$. Moreover, $f(k) = k - 1$ for $k = 2, 3, 4$.

In [FLW00], a general reduction technique is developed for designing (k, W) S-boxes.

I. Find $f(k)$ and all k-way minimal global routings. The existence of $f(k)$ and the finiteness of the number of minimal k-way global routings are guaranteed by Lemma 1.

II. Determine p and r such that $W = pq + r$, p and r are as small as possible so that any (k, W)-global routing is a union of q subglobal routings of density p and a subglobal routing of density r. (Note that as k is fixed and each W corresponds to a unique r, it may has more than one r as W varies but there are finitely many such r's for all W).

III. Design a hyper-universal (k, p) S-box S_1 and a (k, r) S-box S_2 with the number of switches as small as possible. Then a disjoint union of q copies of S_1 and an S_2 is a hyper-universal (k, W) S-box.

IV. Design an efficient detailed routing algorithm to detailed-route any k-way global routing in the S-boxes designed in III.

An efficient algorithm for IV is given. Now the 4-way S-box designs will be focused.

For Step 1, Lemma 1 shows that $f(4) = 3$. For Step 2, [FLW00] showed that any $(4, W)$-global routing can be decomposed as a union of $(4, 6)$-global routings and at most 1 $(4, r)$-global routing for some $r = 1, 2, 3, 4, 5$ and 7.

The following are all primitive minimal $(4, W)$-global routings.

For Step III, [FLW00] gave a hyper-universal $(4, W)$-design $F(W)$ with less than $6.7W$ switches.

The goal of this chapter is to further investigate Step III for obtaining better $(4, i)$-designs for $i = 3, 4$ and 6 and a better $(4, W)$-design than the $(4, W)$-design $F(W)$ constructed in [FLW00].

The following result was proved in [FLW00] which will be used in this chapter.

Lemma 2 Let $G = ((V_1, V_2, V_3, V_4), E)$ be a hyper-universal $(4, W)$-design. Then $|E| \geq 6W$. If G is restricted on any two parts, it gives a hyper-universal $(2, W)$-design. If G is restricted on any three parts, it gives a hyper-universal $(3, W)$-design. The optimum $(2, W)$-design is a perfect matching, and an optimum $(3, W)$-design is a Hamilton cycle which includes each vertex in the graph and forms a circle. Moreover, the optimum $(3, 4)$-design must be a Hamiltonian cycle.

$$GR_1^1 = \{\{1,2,3,4\}\}$$

$$\begin{aligned}
GR_{2,1}^1 &= \{\{1,2\},\{3,4\}\}\\
GR_{2,2}^1 &= \{\{1,3\},\{2,4\}\}\\
GR_{2,3}^1 &= \{\{1,4\},\{2,3\}\}
\end{aligned}$$

$$\begin{aligned}
GR_{3,1}^1 &= \{\{1\},\{2,3,4\}\}\\
GR_{3,2}^1 &= \{\{2\},\{1,3,4\}\}\\
GR_{3,3}^1 &= \{\{3\},\{1,2,4\}\}\\
GR_{3,4}^1 &= \{\{4\},\{1,2,3\}\}
\end{aligned}$$

$$\begin{aligned}
GR_{1,1}^2 &= \{\{1,2,3\},\{1,2,4\},\{3,4\}\}\\
GR_{1,2}^2 &= \{\{1,2,3\},\{2,3,4\},\{1,4\}\}\\
GR_{1,3}^2 &= \{\{1,2,4\},\{2,3,4\},\{1,3\}\}\\
GR_{1,4}^2 &= \{\{1,3,4\},\{2,3,4\},\{1,2\}\}\\
GR_{1,5}^2 &= \{\{1,2,3\},\{1,3,4\},\{2,4\}\}\\
GR_{1,6}^2 &= \{\{1,2,4\},\{1,3,4\},\{2,3\}\}
\end{aligned}$$

$$\begin{aligned}
GR_{2,1}^2 &= \{\{1,2,3\},\{1,4\},\{2\},\{3,4\}\}\\
GR_{2,2}^2 &= \{\{1,2,4\},\{1,3\},\{2\},\{3,4\}\}\\
GR_{2,3}^2 &= \{\{2,3,4\},\{1,4\},\{2\},\{1,3\}\}\\
GR_{2,4}^2 &= \{\{1,2,3\},\{1,4\},\{3\},\{2,4\}\}\\
GR_{2,5}^2 &= \{\{1,4,3\},\{1,2\},\{3\},\{2,4\}\}\\
GR_{2,6}^2 &= \{\{2,4,3\},\{1,2\},\{3\},\{1,4\}\}\\
GR_{2,7}^2 &= \{\{2,4,3\},\{1,2\},\{4\},\{1,3\}\}\\
GR_{2,8}^2 &= \{\{1,4,3\},\{1,2\},\{4\},\{2,3\}\}\\
GR_{2,9}^2 &= \{\{1,2,4\},\{1,3\},\{4\},\{2,3\}\}\\
GR_{2,10}^2 &= \{\{1,2,4\},\{4,3\},\{1\},\{2,3\}\}\\
GR_{2,11}^2 &= \{\{1,4,3\},\{4,2\},\{1\},\{2,3\}\}\\
GR_{2,12}^2 &= \{\{2,1,3\},\{4,2\},\{1\},\{4,3\}\}
\end{aligned}$$

$$\begin{aligned}
GR_{3,1}^2 &= \{\{1,2\},\{3,1\},\{2,3\},\{4\},\{4\}\}\\
GR_{3,2}^2 &= \{\{1,2\},\{4,1\},\{2,4\},\{3\},\{3\}\}\\
GR_{3,3}^2 &= \{\{1,3\},\{4,1\},\{3,4\},\{2\},\{2\}\}\\
GR_{3,4}^2 &= \{\{3,2\},\{4,3\},\{2,4\},\{1\},\{1\}\}
\end{aligned}$$

$$GR_1^3 = \{\{1,2,3\},\{1,2,4\},\{3,4,1\},\{2,3,4\}\}$$

$$\begin{aligned}
GR_{2,1}^3 &= \{\{1,2,3\},\{1,4\},\{2,4\},\{3,4\},\{1,2,3\}\}\\
GR_{2,2}^3 &= \{\{2,3,4\},\{1,2\},\{1,3\},\{1,4\},\{2,3,4\}\}\\
GR_{2,3}^3 &= \{\{3,4,1\},\{2,1\},\{2,3\},\{2,4\},\{3,4,1\}\}\\
GR_{2,4}^3 &= \{\{4,1,2\},\{3,1\},\{3,2\},\{3,4\},\{4,1,2\}\}
\end{aligned}$$

Table 5.3: Primitive minimal 4-way global routings

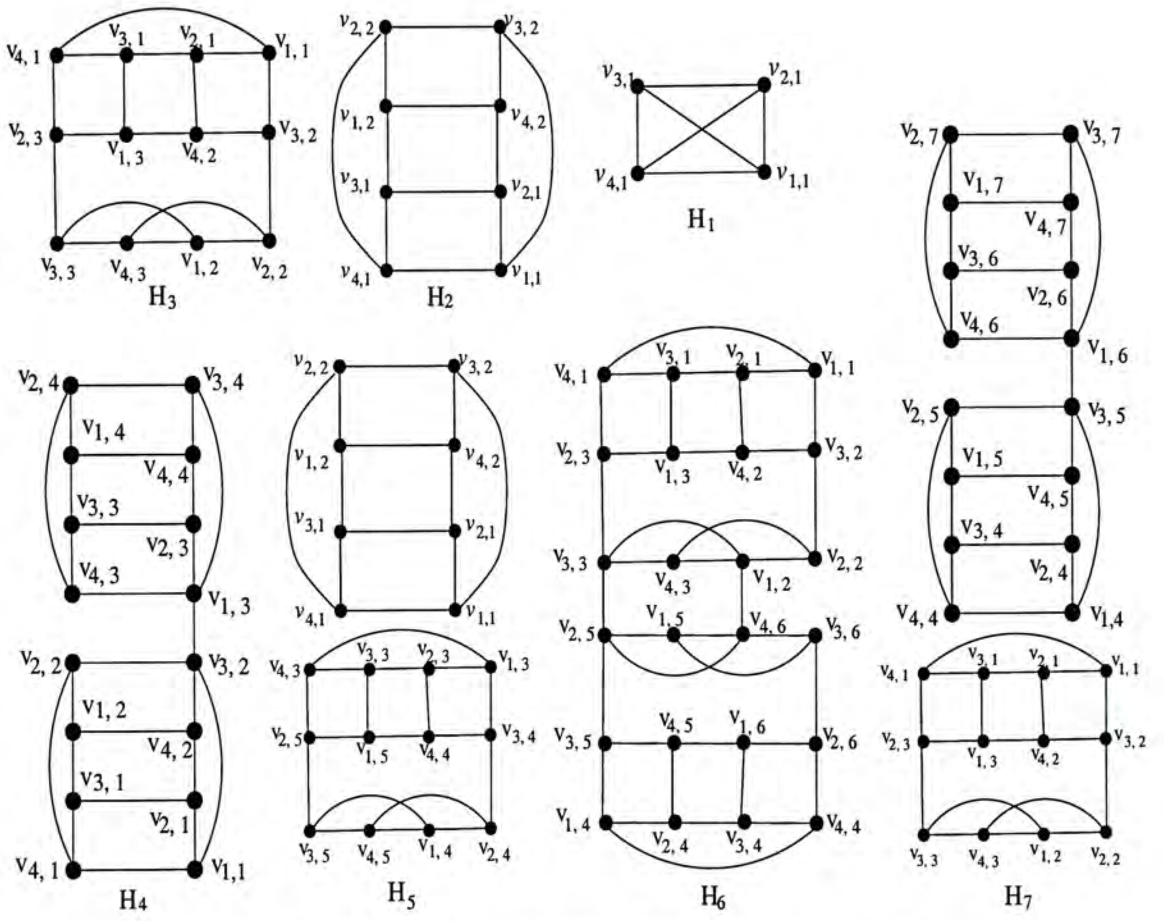


Figure 5.21: A family of hyper-universal 4-way designs.

5.6 Hyper-universal $(4, W)$-design analysis

To design a hyper-universal $(4, W)$-design, Step III of the reduction design technique mentioned in Section 2 is processed. Figure 5.21 provides a family of $(4, i)$-designs H_i for $i = 1, 2, 3, 4, 5, 6, 7$ which are necessary for designing a $(4, W)$-design for all W according to this technique. Figure 5.22 also shows the structure of (4,7)-design H_7 on the S-box. In [FLW00], it showed that H_i is an optimum $(4, i)$-design for $i = 1, 2$. The following lemmas claim that H_i's are hyper-universal or optimum $(4, i)$-designs for $i = 3, 4, 5, 6, 7$. The prove of these lemmas is discussed in this later section.

Lemma 3 H_i is an optimum $(4, i)$-design for $i = 3, 4, 5$.

Lemma 4 H_i is a hyper-universal $(4, i)$-design for $i = 6, 7$.

Now define $G(W)$ as the following graph:

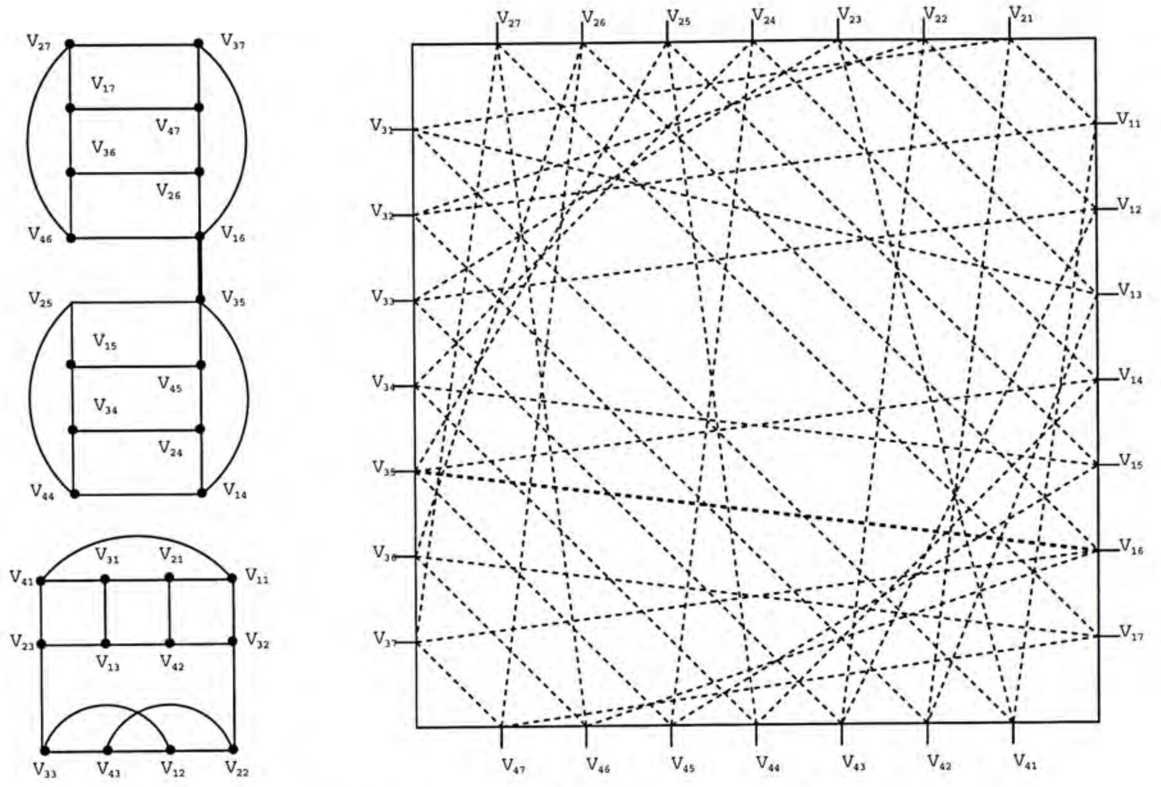

V27
V37
V17
V47
V36
V26
V46
V16
V25
V35
V15
V45
V34
V24
V44
V14
V31
V21
V41
V11
V23
V32
V13
V42
V33
V43
V12
V22
V27 V26 V25 V24 V23 V22 V21
V31 V32 V33 V34 V35 V36 V37
V11 V12 V13 V14 V15 V16 V17
V47 V46 V45 V44 V43 V42 V41

Figure 5.22: Structure of H7

$$G(W) = \begin{cases} \text{disjoint union of } h\ H_6\text{'s} & \text{if } W = 6h, \\ \text{disjoint union of } (h-1)\ H_6\text{'s and a } H_7 & \text{if } W = 6h+1, \\ \text{disjoint union of } h\ H_6\text{'s and a } H_2 & \text{if } W = 6h+2, \\ \text{disjoint union of } h\ H_6\text{'s and a } H_3 & \text{if } W = 6h+3, \\ \text{disjoint union of } h\ H_6\text{'s and a } H_4 & \text{if } W = 6h+4, \\ \text{disjoint union of } h\ H_6\text{'s and a } H_5 & \text{if } W = 6h+5. \end{cases}$$

By the definition of $G_i, i = 1, \ldots, 7$, it is easy to see that the number of edges of $F(W)$ for $W > 1$ is

$$|G(W)| = \begin{cases} \frac{19}{3}W & \text{if } W = 0(mod\ 6), \\ \frac{19}{3}W - \frac{4}{3} & \text{if } W = 1(\text{mod } 6), \\ \frac{19}{3}W - \frac{2}{3} & \text{if } W = 2(\text{mod } 6), \\ \frac{19}{3}W - 1 & \text{if } W = 3(\text{mod } 6), \\ \frac{19}{3}W - \frac{1}{3} & \text{if } W = 4(\text{mod } 6), \\ \frac{19}{3}W - \frac{5}{3} & \text{if } W = 5(\text{mod } 6). \end{cases}$$

Theorem 5 For $W > 1$, $G(W)$ is a hyper-universal $(4, W)$-design.

Proof.

If $W = 6h + 1$, then $h \geq 1$ and any $(4, 6h + 1)$-global routing GR can be decomposed into a union of $(h - 1)$ $(4, 6)$-global routings and a $(4, 7)$-global routing. (sometimes, GR can be decomposed into h $(4, 6)$-global routings and a $(4, 1)$-design. But this does not always happen.) Now it is easy to see that $G(6h + 1)$ contains a detailed routing of GR as $G(W)$ is a disjoint union of $(h - 1)$ H_6's and an H_7, and H_6 and H_7 are hyper-universal $(4, 6)$-design and $(4, 7)$-design, respectively by Lemma 4.

Let $W = 6h + i$ where $i \neq 1$. Since the densities of minimal 4-way global routings are $1, 2$ or 3, any $(4, 6h+i)$-global routing GR can be decomposed into h $(4, 6)$-global routings and a $(4, i)$-global routing for $2 \leq i \leq 5$. Since $G(6h+i)$ is a disjoint union of h H_6's and an H_i, and H_6 and H_i are hyper-universal $(4, 6)$-design and $(4, i)$-design, respectively by Lemma 3, then $G(6h+i)$ contains a detailed routing of GR. ∎

Next the proof of the lemmas 3 and 4 are divided into subsections. For simplicity, each vertex of H_i is labeled by only the side label, where the corresponding track belongs to.

5.6.1 H_3 is an optimum $(4, 3)$-design

Note that H_3 has 18 edges which is the lower bound. Therefore, it is sufficient to show that H_3 is hyper-universal. It is suffice to show that H_3 contains all possible $(4, 3)$-global routings obtained by combining the primitive minimal global routings in Table 5.3. All these primitive are depicted in Appendix E. Notice that the permutation $\sigma = (1, 4)(2, 3)$ is an automorphism of H_3, thus it is only necessary to detailed-route GR in H_3 where GR is a union of those global routings in Table 5.3 which are not isomorphic to each other under σ. These global routings are shown in Table 5.4.

$$GR_1^1 = \{\{1,2,3,4\}\}$$

$$GR_{2,1}^1 = \{\{1,2\},\{3,4\}\}$$
$$GR_{2,2}^1 = \{\{1,3\},\{2,4\}\}$$
$$GR_{2,3}^1 = \{\{1,4\},\{2,3\}\}$$

$$GR_{3,1}^1 = \{\{1\},\{2,3,4\}\}$$
$$GR_{3,2}^1 = \{\{2\},\{1,3,4\}\}$$

$$GR_{1,1}^2 = \{\{1,2,3\},\{1,2,4\},\{3,4\}\}$$
$$GR_{1,2}^2 = \{\{1,2,3\},\{2,3,4\},\{1,4\}\}$$
$$GR_{1,3}^2 = \{\{1,2,4\},\{2,3,4\},\{1,3\}\}$$
$$GR_{1,6}^2 = \{\{1,2,4\},\{1,3,4\},\{2,3\}\}$$

$$GR_{2,1}^2 = \{\{1,2,3\},\{1,4\},\{2\},\{3,4\}\}$$
$$GR_{2,2}^2 = \{\{1,2,4\},\{1,3\},\{2\},\{3,4\}\}$$
$$GR_{2,3}^2 = \{\{2,3,4\},\{1,4\},\{2\},\{1,3\}\}$$
$$GR_{2,7}^2 = \{\{2,4,3\},\{1,2\},\{4\},\{1,3\}\}$$
$$GR_{2,8}^2 = \{\{1,4,3\},\{1,2\},\{4\},\{2,3\}\}$$
$$GR_{2,9}^2 = \{\{1,2,4\},\{1,3\},\{4\},\{2,3\}\}$$

$$GR_{3,1}^2 = \{\{1,2\},\{3,1\},\{2,3\},\{4\},\{4\}\}$$
$$GR_{3,2}^2 = \{\{1,2\},\{4,1\},\{2,4\},\{3\},\{3\}\}$$

$$GR_1^3 = \{\{1,2,3\},\{1,2,4\},\{3,4,1\},\{2,3,4\}\}$$

$$GR_{2,1}^3 = \{\{1,2,3\},\{1,4\},\{2,4\},\{3,4\},\{1,2,3\}\}$$
$$GR_{2,3}^3 = \{\{3,4,1\},\{2,1\},\{2,3\},\{2,4\},\{3,4,1\}\}$$

Table 5.4: Primitive minimal 4-way global routings which are not σ isomorphic

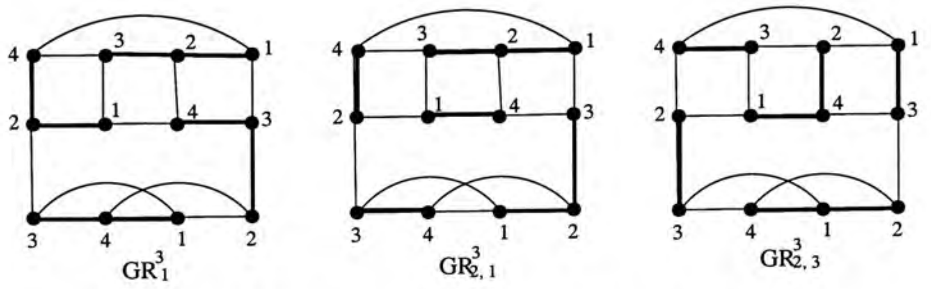


Figure 5.23: Detailed Routing of GR (Case 1)

Let GR be a $(4,3)$-global routing which is a union of global routings from Table 5.4.

Case 1. $GR \in \{GR^3_1, GR^3_{2,1}, GR^3_{2,3}\}$.

A detailed routing of GR is given by Figure 5.23.

Case 2. GR consists of only density 1 global routings.

Subcase 2.1. GR does not contain $GR^1_{2,3}$.

In this case, the graph H_3 can be partitioned into three subgraphs G_1, G_2 and G_3: G_1 consists of the lower level, G_2 consists of the left of the top and second levels and G_3 consists of the right of top and second level. Note that each subgraph can detailed-route any of $\{GR^1_1, GR^1_{2,1}, GR^1_{2,2}, GR^1_{3,1}, GR^1_{3,2}\}$. Therefore, it is sufficient to detailed-route GR in H_3.

Subcase 2.2. $GR = 3GR^1_{2,3}$s.

A detailed routing is given in Figure 5.24(a).

Subcase 2.3. GR contains only one $GR^1_{2,3}$.

If GR contains $GR^1_{2,3} \cup GR^1_{2,2}$, then it is possible to detailed-route $GR^1_{2,3} \cup GR^1_{2,2}$ as shown in Figure 5.24(b), and if GR contains $GR^1_{2,3} \cup GR^1_{3,2}$, and the detailed-route $GR^1_{2,3} \cup GR^1_{3,2}$ are showed in Figure 5.24(c). Note that the unused part in H_3 can detailed-route any of $\{GR^1_1, GR^1_{2,1}, GR^1_{2,2}, GR^1_{3,1}, GR^1_{3,2}\}$. This proves that GR is routable in H_3.

For those GR's which do not contain $GR^1_{2,2}$ and $GR^1_{3,2}$, detailed routings in H_3 is shown in Figure 5.24(d),(e),(f).

Subcase 2.4. GR contains two $GR^1_{2,3}$'s.

If detailed-route of $2GR^1_{2,3}$ is performed as shown in Figure 5.24(g), then

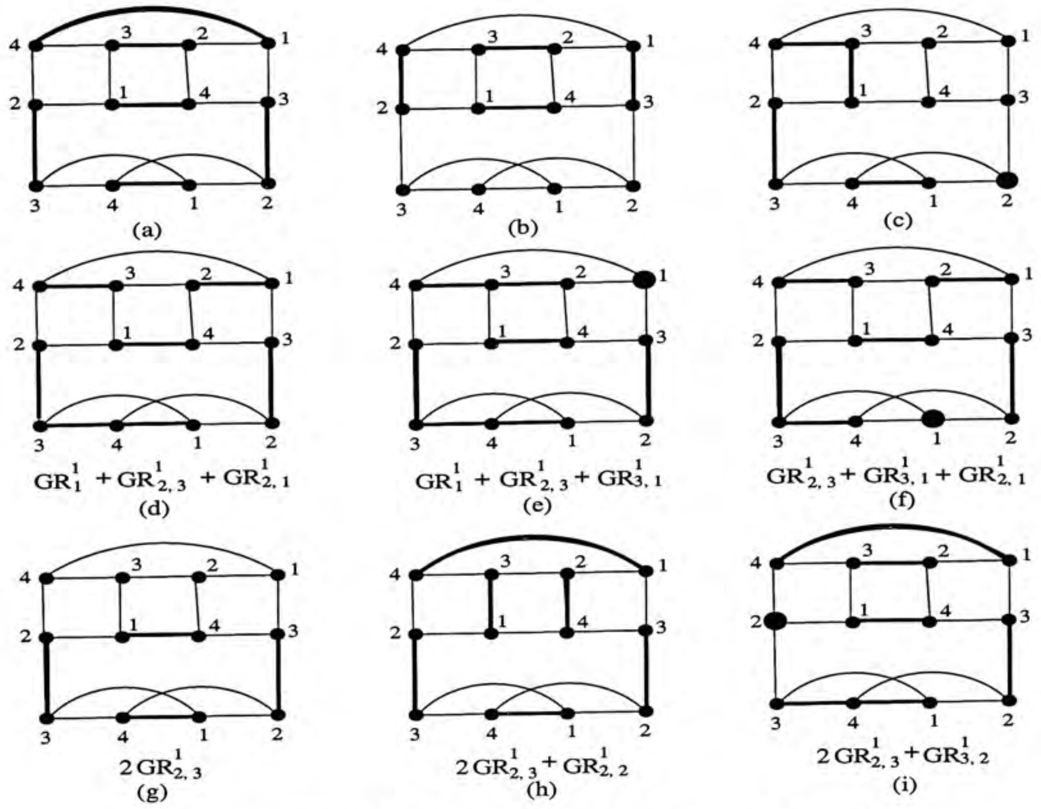


Figure 5.24: Detailed Routing for Table 3.2

the unused top level can detailed-route any of $\{GR_1^1, GR_{2,1}^1, GR_{3,1}^1\}$. For $GR = 2GR_{2,3}^1 + GR_{2,2}^1$ or $GR = 2GR_{2,3}^1 + GR_{3,2}^1$, a detailed routing is given by Figure 5.24(h),(i).

Case 3. GR contains a minimal global routing of density 2.

Figure 5.25 shows all detailed routings of density 2 global routings in H_3. Note that the lower level of H_2 is not used in all the routings in Figure 5.25 and the lower level can be used to detailed-route $GR_1^1, GR_{2,1}^1, GR_{2,2}^1, GR_{3,1}^1$ and $GR_{3,2}^1$. Therefore, detailed-route GR can be done in H_3 if GR is a union of a density 2 global routing and a density 1 global routing from $\{GR_1^1, GR_{2,1}^1, GR_{2,2}^1, GR_{3,1}^1, GR_{3,2}^1\}$.

A detailed routing of GR in H_3 is given in Figure 5.26 when GR is a union of density 2 global routing and $GR_{2,3}^1$.

This completes the verification that it is detailed routable in H_3 for all $(4,3)$-global routings, and hence H_3 is hyper-universal.

Note that only when $GR \in \{2GR_{2,3}^1 + GR_{2,2}^1, 2GR_{2,3}^1 + GR_{3,2}^1, GR_{2,3}^1\}$, it is needed for the top level edge $\{1,4\}$ in the detailed routing. This information

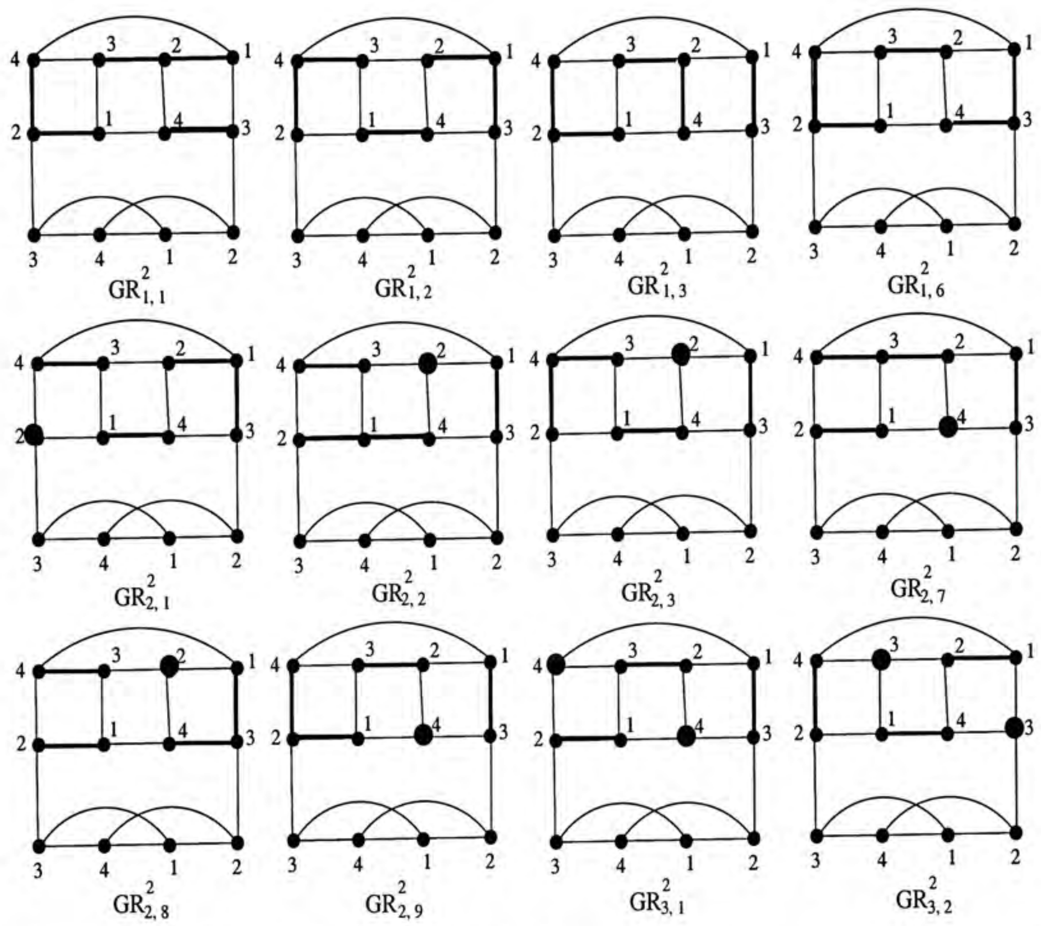


Figure 5.25: Detailed Routing of density 2 Global Routing in H_3

will be used later.

5.6.2 H_4 is an optimum $(4,4)$-design

The first step is to show that H_4 is hyper-universal. Let GR be any $(4,4)$-global routing which is a union of global routings from Table 5.3. If GR is a union of a minimal $(4,3)$-global routing and a $(4,1)$-global routing from Table 5.3, the five minimal $(4,3)$-global routings can be detailed-routed as in Figure 5.27(a). Note that the unused part in H_4 is a cycle $1,2,4,3$ which can be used to detailed-route all the $(4,1)$-global routings except $GR^1_{2,3}$ from Table 5.3. If GR contains a $GR^1_{2,3}$, a detailed routing of GR in H_4 is given in Figure 5.27(b).

Now assume that GR is a union of two $(4,2)$-global routings, then GR is routable in H_4 as H_4 contains two disjoint H_2's.

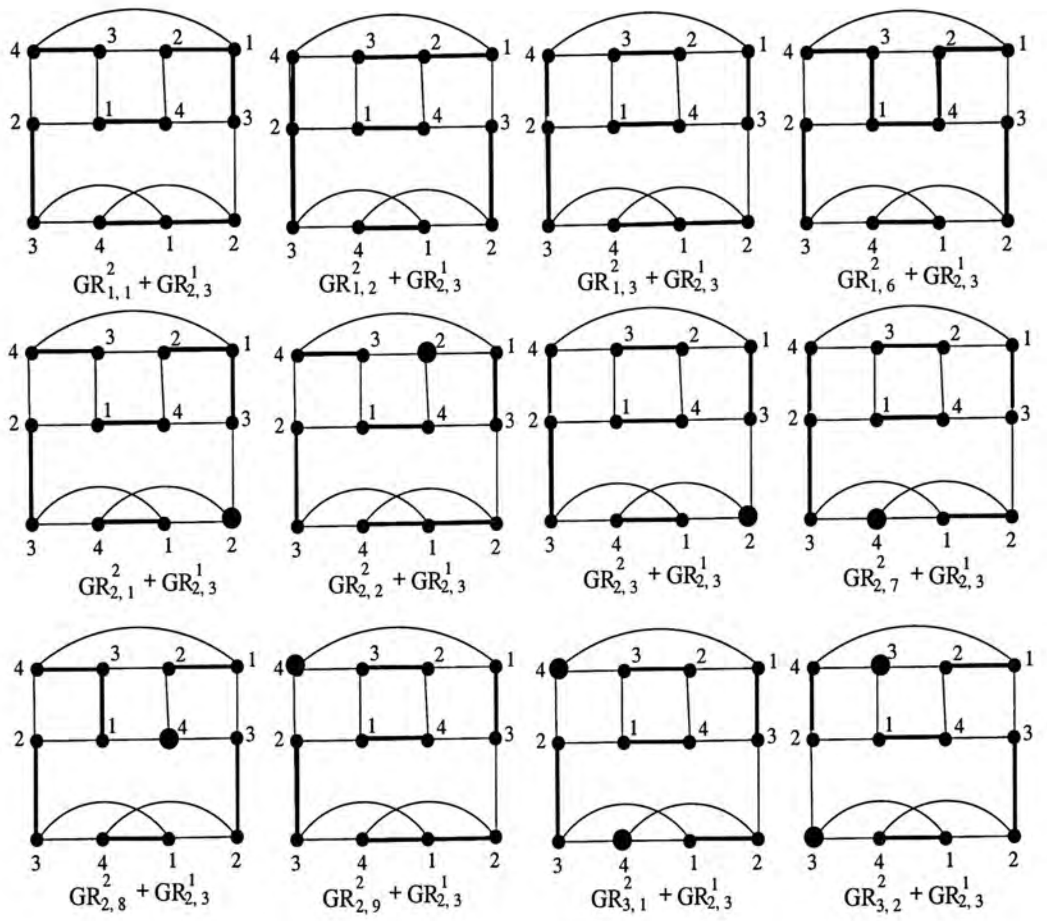

$GR^2_{1,1} + GR^1_{2,3}$
$GR^2_{1,2} + GR^1_{2,3}$
$GR^2_{1,3} + GR^1_{2,3}$
$GR^2_{1,6} + GR^1_{2,3}$
$GR^2_{2,1} + GR^1_{2,3}$
$GR^2_{2,2} + GR^1_{2,3}$
$GR^2_{2,3} + GR^1_{2,3}$
$GR^2_{2,7} + GR^1_{2,3}$
$GR^2_{2,8} + GR^1_{2,3}$
$GR^2_{2,9} + GR^1_{2,3}$
$GR^2_{3,1} + GR^1_{2,3}$
$GR^2_{3,2} + GR^1_{2,3}$

Figure 5.26: Detailed Routing of GR in H_3

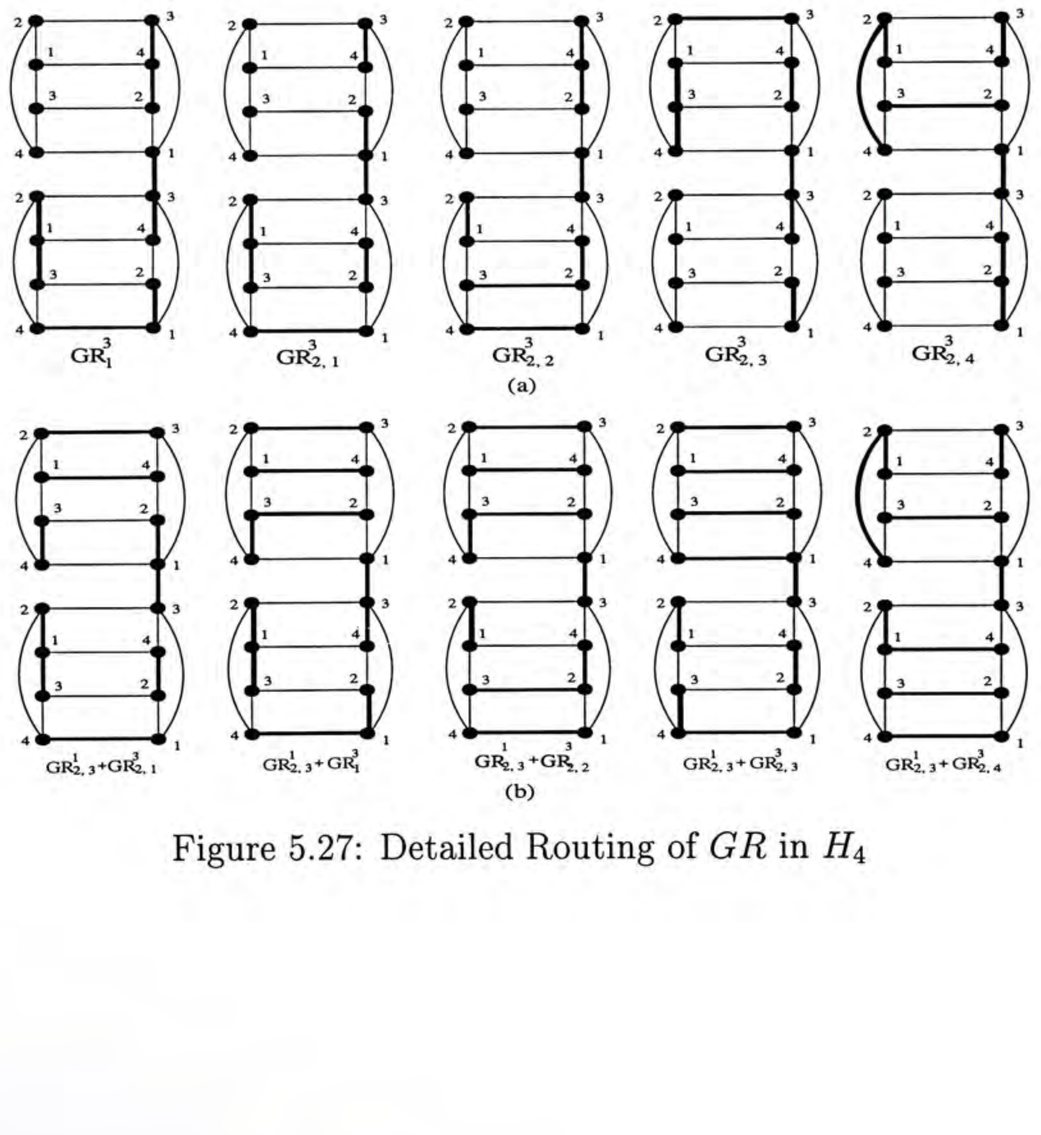


Figure 5.27: Detailed Routing of GR in H_4

This proves H_4 is hyper-universal. The fact that H_4 is an optimum design follows from the following result which also indicates that the lower bound $6W$ given in Lemma 2 is not achievable in general.

Theorem 6 There is no 3-regular hyper-universal $(4,4)$-design. An optimum $(4,4)$-design must have at least 25 edges.

Proof: Suppose that there is a 3-regular hyper-universal $(4,4)$-design G. Then G is a 4-partite graph on (V_1, V_2, V_3, V_4); there is a 4-matching between each pair of V_i, V_j for $i \neq j$ and the induced subgraph of G on each set $V_i \cup V_j \cup V_m$ $(i \neq j \neq m)$ is a cycle (see Lemma2). Based on these facts, all such 3-regular graphs can be constructed.

It is easy to observe (by relabeling if necessary) that any graph described in the above paragraph contains the graph G_1 as shown in the Figure 5.28(a). Begin from G_1, selection of a matching between V_1 and V_3 is done so that the induced subgraph on $V_1 \cup V_2 \cup V_3$ is a cycle. There are 6 different choices. For each of these choices, it is necessary to select a matching between V_1 and V_4 to obtain G so that the induced subgraphs of G on $V_1 \cup V_2 \cup V_4$ and on $V_1 \cup V_3 \cup V_4$ are Hamiltonian cycles. Total 23 such graphs are constructed. For each of these graph, a $(4,4)$-global routing is found and it is not detailed routable in the graph (see Figure 5.28). As a result, it showed that there is no 3-regular hyper-universal $(4,4)$-design.

5.6.3 H_i is a hyper-universal $(4,i)$-design for $i = 5, 6, 7$

Note that any $(4,5)$-global routing is a union of a $(4,3)$-global routing and a $(4,2)$-global routing, and H_5 contains an H_3 and a disjoint H_2, therefore, H_5 is hyper-universal. Also note that H_5 has 30 edges which is the lower bound of an hyper-universal $(4,5)$-design. Therefore, H_5 is an optimum design.

H_7 is a hyper-universal $(4,7)$-design is similarly proved.

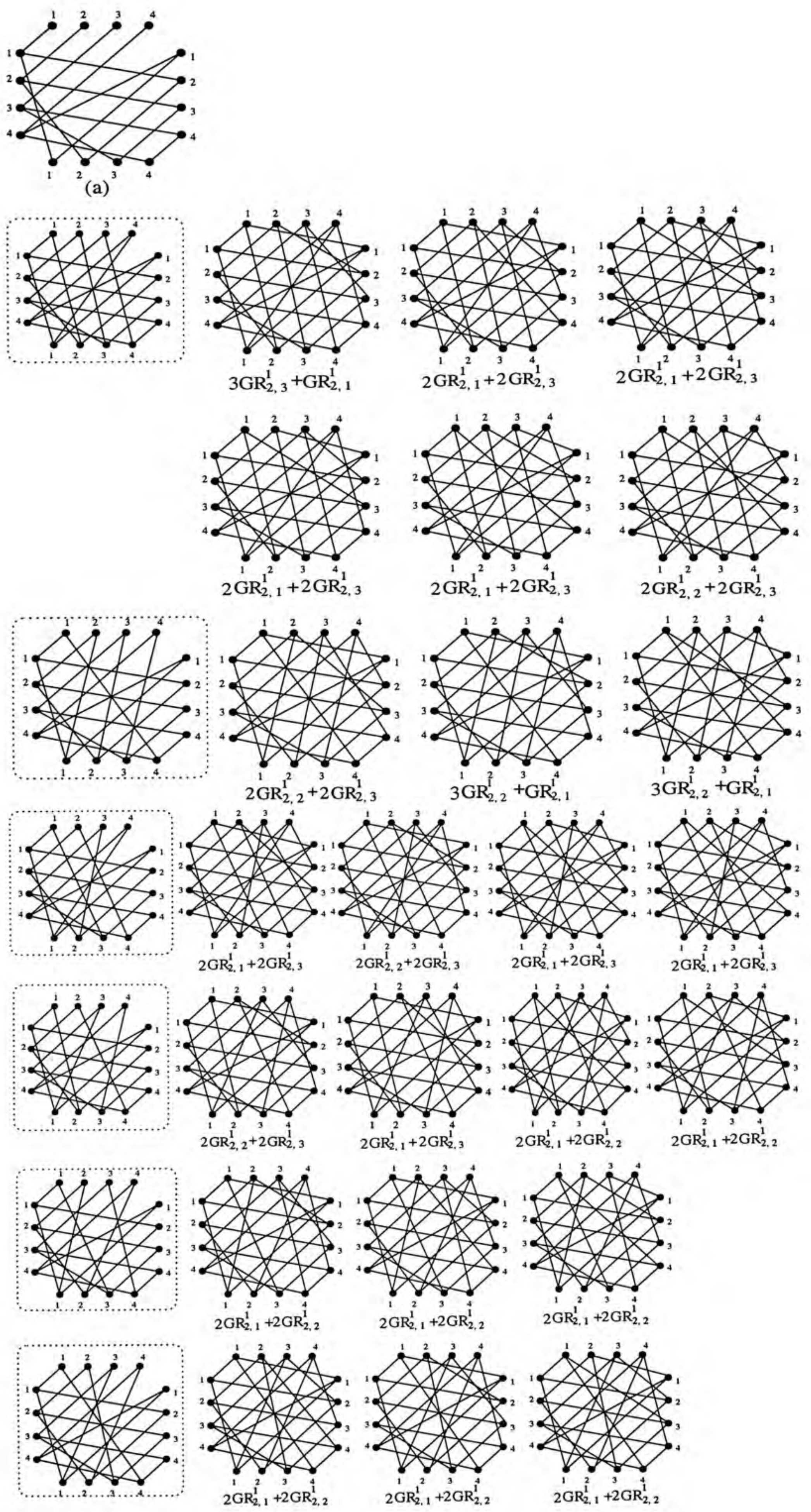

(a)
$3GR^1_{2,3}+GR^1_{2,1}$
$2GR^1_{2,1}+2GR^1_{2,3}$
$2GR^1_{2,1}+2GR^1_{2,3}$
$2GR^1_{2,1}+2GR^1_{2,3}$
$2GR^1_{2,1}+2GR^1_{2,3}$
$2GR^1_{2,2}+2GR^1_{2,3}$
$2GR^1_{2,2}+2GR^1_{2,3}$
$3GR^1_{2,2}+GR^1_{2,1}$
$3GR^1_{2,2}+GR^1_{2,1}$
$2GR^1_{2,1}+2GR^1_{2,3}$
$2GR^1_{2,2}+2GR^1_{2,3}$
$2GR^1_{2,1}+2GR^1_{2,3}$
$2GR^1_{2,1}+2GR^1_{2,3}$
$2GR^1_{2,2}+2GR^1_{2,3}$
$2GR^1_{2,1}+2GR^1_{2,3}$
$2GR^1_{2,1}+2GR^1_{2,2}$
$2GR^1_{2,1}+2GR^1_{2,2}$
$2GR^1_{2,1}+2GR^1_{2,2}$
$2GR^1_{2,1}+2GR^1_{2,2}$
$2GR^1_{2,1}+2GR^1_{2,2}$
$2GR^1_{2,1}+2GR^1_{2,2}$
$2GR^1_{2,1}+2GR^1_{2,2}$
$2GR^1_{2,1}+2GR^1_{2,2}$

Figure 5.28: (4,4)-Global Routing

The first step to show H_6 is a hyper-universal S-box is to prove H_6 containing a detailed routing for every $(4,6)$-global routing. Then, note that the permutation $\sigma = (1,4)(2,3)$ is an automorphism of H_6. Therefore, it is necessary to check those global routings which are union of global routings from Table 5.4. Let GR be such a global routing.

Case 1 $GR = GR_1 + GR_2$, where both GR_1 and GR_2 are $(4,3)$-global routings.

It is easy to see that H_6 contains a detailed routing of GR as H_6 contains two disjoint H_3's.

Case 2 $GR = GR_1 + GR_2 + GR_3$, where each GR_i $(i = 1,2,3)$ is a primitive minimal global routings from Table 5.4.

Let $K(i, i+1)$ be the subgraph of H_6 which consists of the levels i and $i+1$. There are three disjoint subgraphs $K(1,2)$, $K(3,4)$ and $K(5,6)$ of H_6. In Figure 5.25, it shows that $K(1,2)$ and $K(5,6)$ can detailed-route any minimal $(4,2)$-global routings in Table 5.4. All detailed routing of minimal global routing in $K(3,4)$ are routable. Therefore, there is a detailed routing for GR in H_6, and hence H_6 is hyper-universal.

5.7 Experimental Results

From the combinatorial analysis shown above, it seems a bit surprising to see that an optimum (hyper-universal) S-box can actually be built using only very few more switches beyond the widely believed lower bound of $6W$. It is also interesting to see that there exist HUSBs with switch density of only $6W$ for some W's and the construction of optimum HUSBs seem to hardly possess regular scalability which can be observed in the construction of some 4-way USB family. Besides the theoretical analysis, in order to get some experimental justification, the experiments adopted the currently known best FPGA router VPR [BR97], which is available on the Web. The logic block structure for the VPR runs is set to consist of one 4-input LUT and one flip-flop. The input or output pin of the logic block is able to connect to any track in the adjacent channels, $F_c = W$. Inside the switch box, each input wire segment can connect to three other output wire segments of other channels, $F_s = 3$. In appendix D, the HUSBs H_i for $i = 1, \ldots, 9$ are depicted.

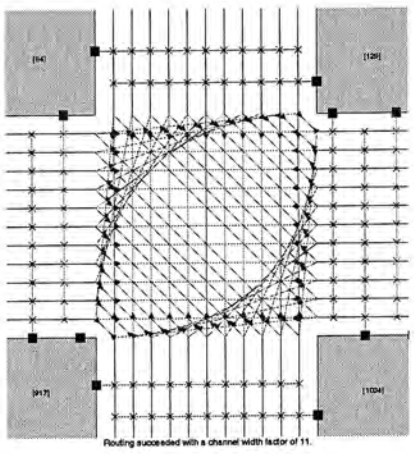


Figure 5.29: The structure of Disjoint S-Box, W=11

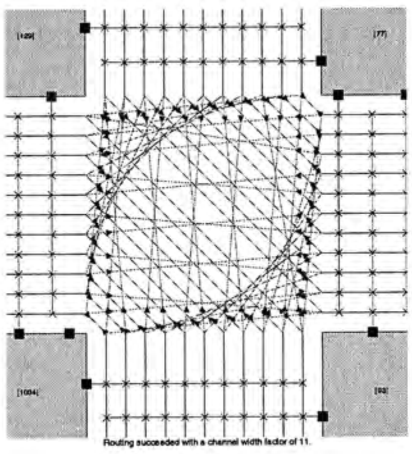


Figure 5.30: The structure of H'USB(4,11) (H'USB(4,7) and H'USB(4,4))

In order to have an even comparison (partially is also due to the limitation of VPR router limiting F_s to 3) with the well-known Disjoint structure, the experiments deliberately eliminate the "additional" switches of the HUSBs to make the H'USBs have density of $6W$, which is the same as Disjoint S-boxes. Figure 5.29- 5.36 show some S-box structures and routing results of the experiments.

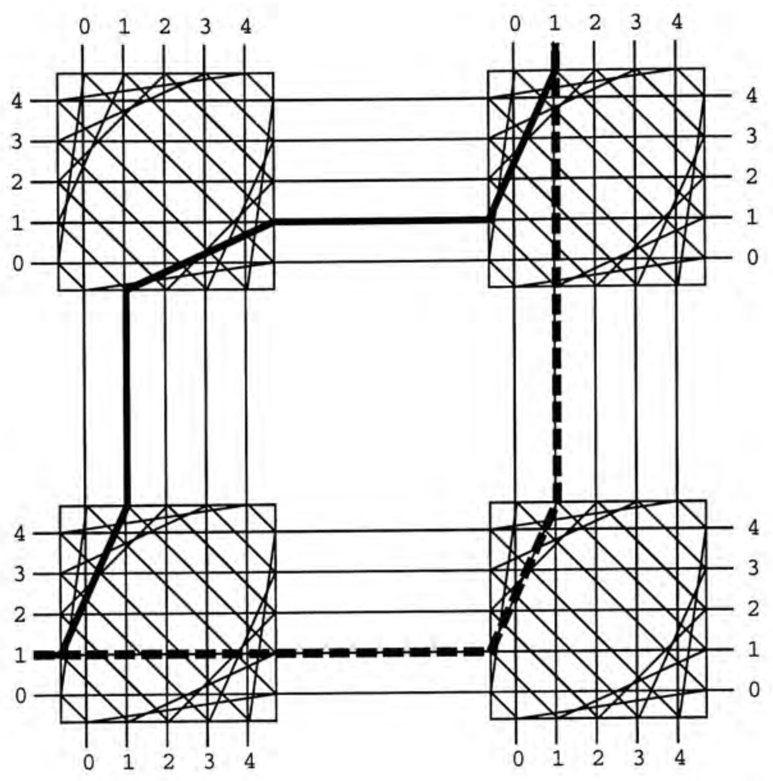


Figure 5.31: Detailed Routing by using Disjoint S-Box

As shown in Figure 5.31 and Figure 5.32, there is more degree of freedom for HUSB switch-box if detailed routing is carried out from the lower left switch-box to upper right one. It is because HUSB switch-box allows the router to choose track 1 or 2 instead of Disjoint type which only allows routing on track 1. In Figure 5.33 and Figure 5.34, it shows Disjoint S-box is insufficient to route GR_1^3 while H_3 provides higher routability.

In Table 5.5, it showed the comparing results of the number of tracks required to route some larger MCNC benchmark circuits [Yan91] by Disjoint and the H'USB FPGAs. Overall, the H'USB FPGAs use about 10% less tracks than the Disjoint FPGAs. (Beware that since the VPR is a simulated annealing based non-deterministic router, the results which produced for Disjoint FPGAs could be a bit different to their other reported results.)

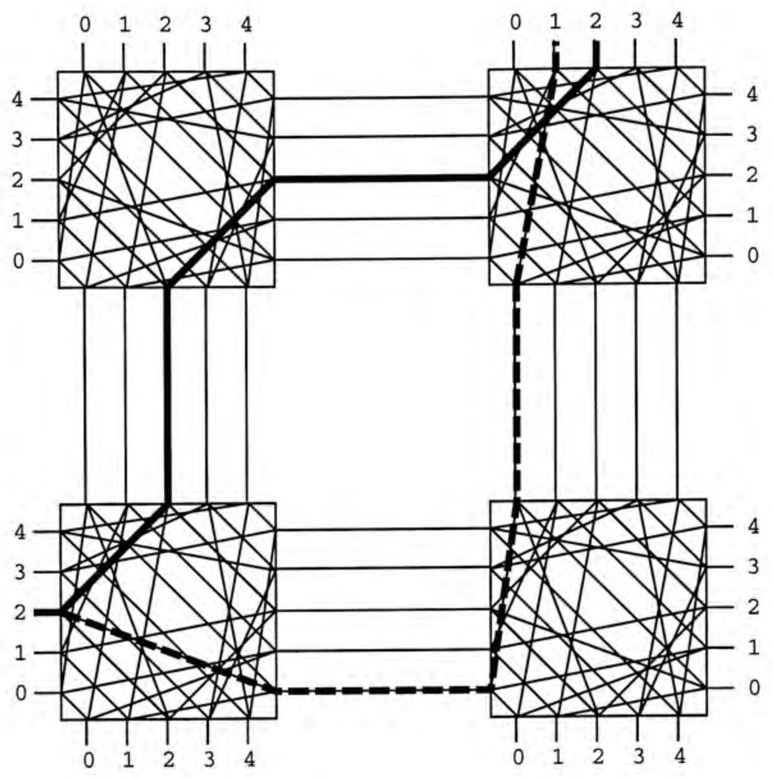


Figure 5.32: Detailed Routing by using H'USB S-Box

5.8 Conclusions

In this chapter, the reduction design technique developed in [FLW00] in 4-way S-box designs is applied. The new $(4, W)$ S-box has at most $6.\overline{3}W$ switches compared with the previous one which has about $6.\overline{6}W$ switches. Note that, according to the new design, a complete database for detailed routings in H_i for $i = 2, 3, 4, 5, 6$ and 7 is built and hence have an efficient detailed routing algorithm to detailed-route any $(4, W)$-global routing in the S-box $G(W)$.

One example is provided to show that $6W$ is not a lower bound of the number of switches in an optimum $(4, W)$ S-box for some W while which is widely believed to be. This suggests that the newly design $G(W)$ is very close to an optimum. Currently, research works on applying this combinatorial analysis models for other FPGA routing architecture designs are conducting.

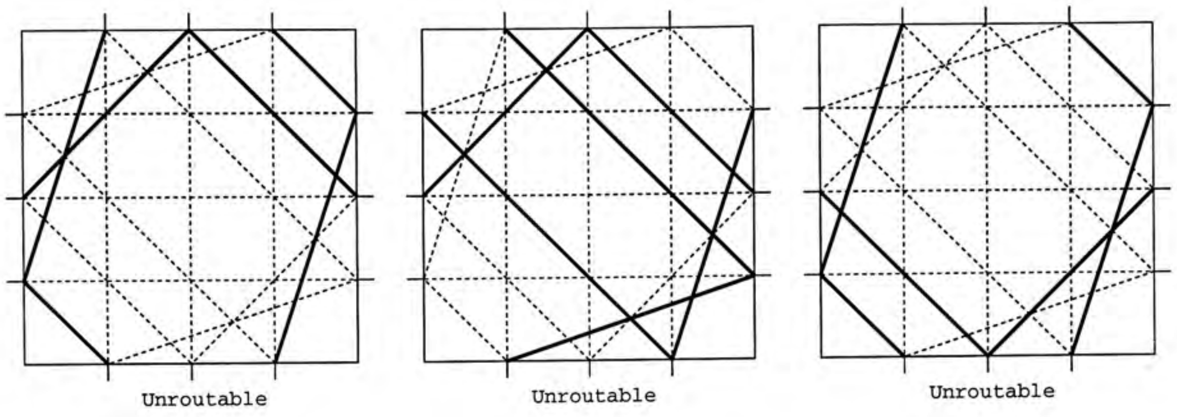


Figure 5.33: Detailed Routing by using Disjoint S-Box on GR_1^3

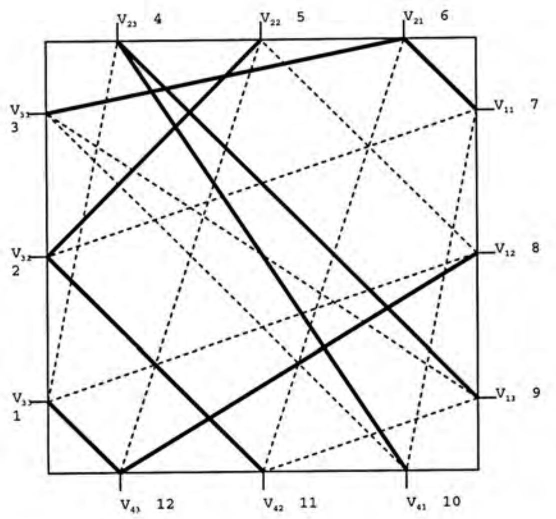


Figure 5.34: Detailed Routing by using Hyper-Universal S-Box on GR_1^3

	Disjoint	H'USB
alu4	12	10
apex2	12	11
apex4	15	13
bigkey	8	7
des	9	8
diffeq	9	8
dsip	7	7
elliptic	11	11
ex5p	15	13
misex3	13	12
seq	12	12
spla	16	14
tseng	8	7
e64	9	8
Total	156	141 (-9.62%)

Table 5.5: Channel widths required for different benchmark circuits $F_C = W$, $F_S = 3$

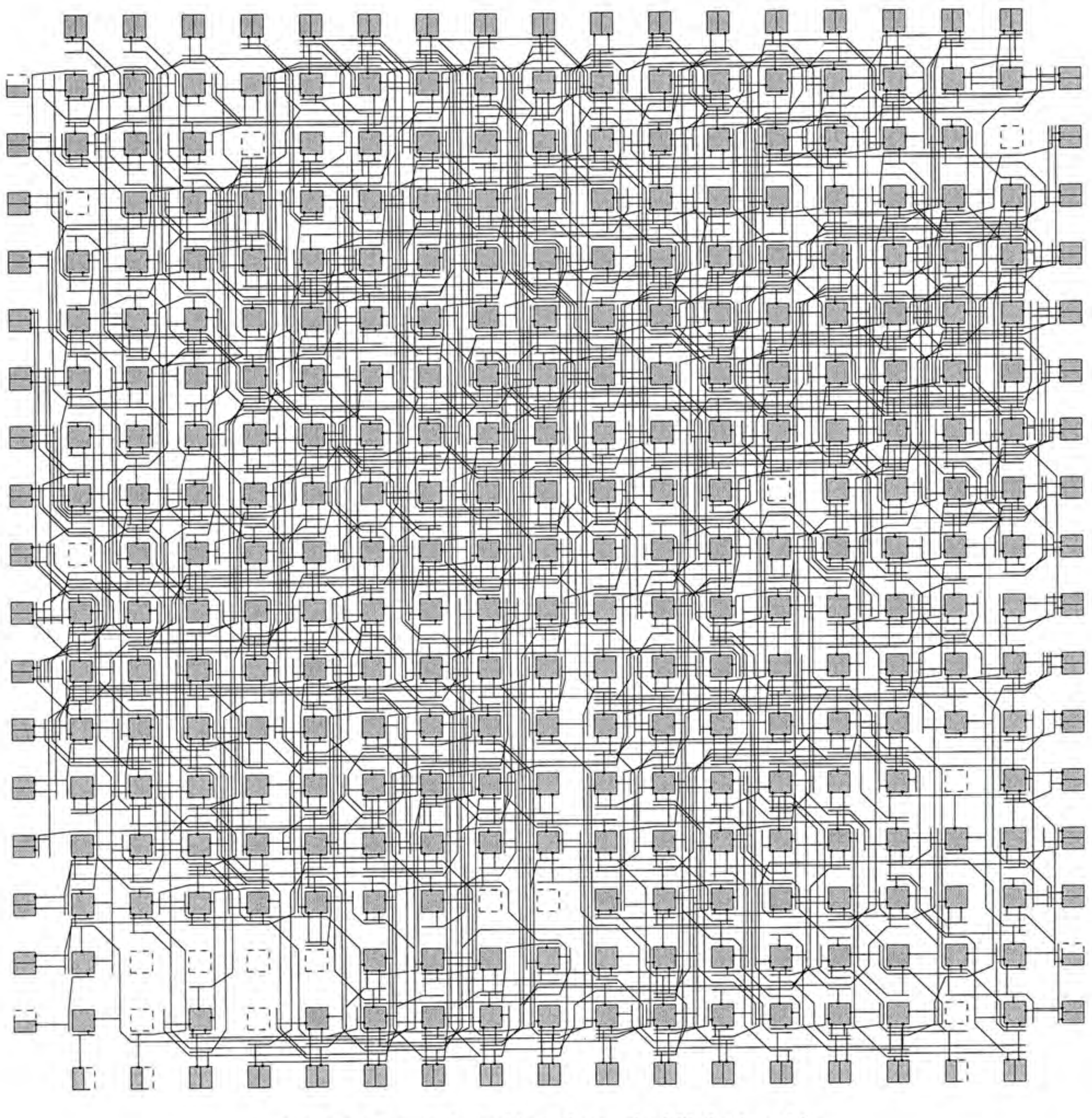


Figure 5.35: Routing result of **e64** by using Disjoint S-Box, W=9

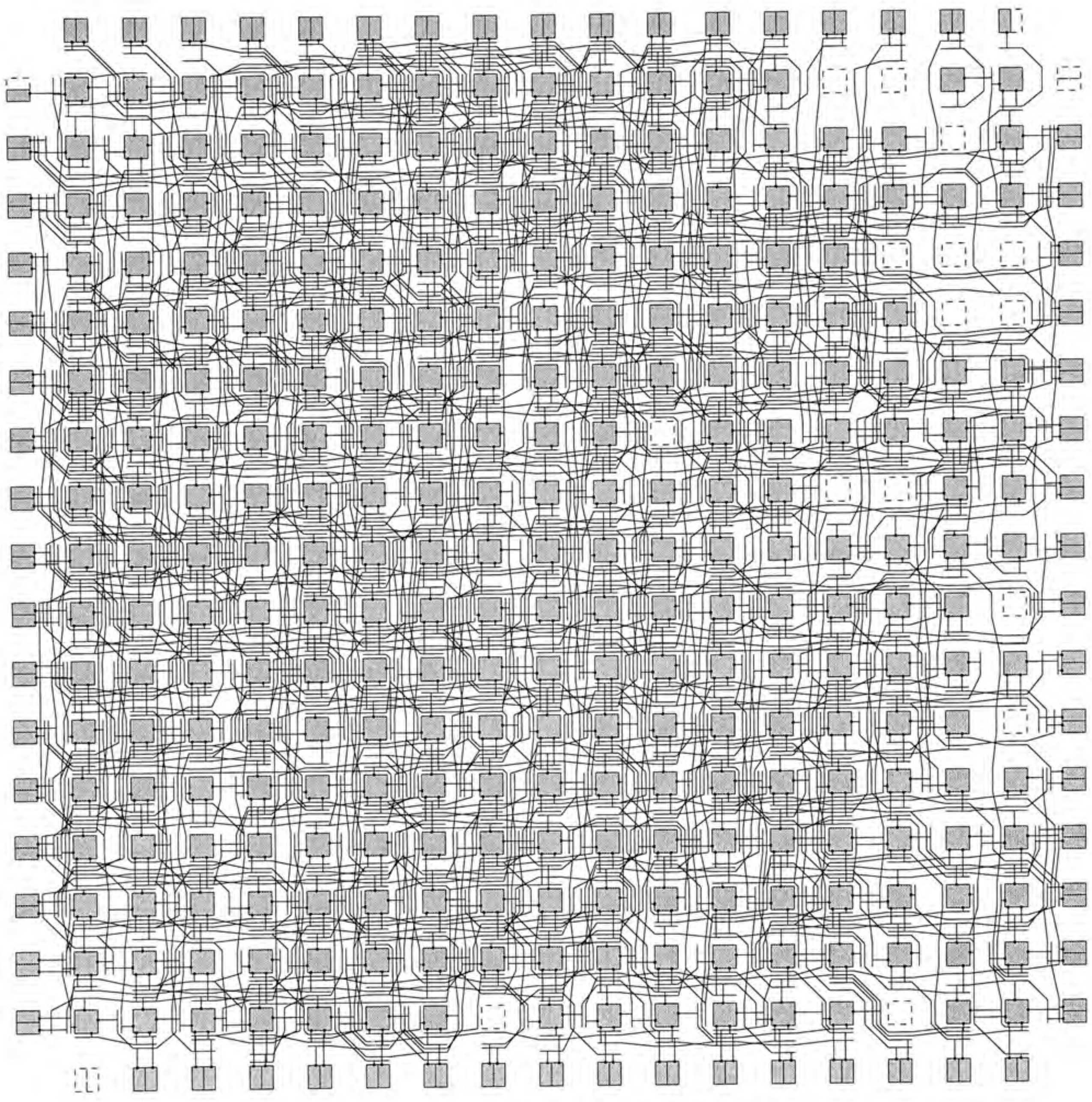


Figure 5.36: Routing result of **e64** by using H'USB S-Box, W=8

Chapter 6

Conclusions

6.1 Thesis Summary

In this thesis, the physical design flow was first discussed, followed by introducing the newly proposed alternative wiring tool Graph-Based Alternative Wiring (GBAW). Logic optimization and the circuit analysis on performance by using GBAW were examined. In Chapter 3, an intensive comparison with famous rewiring tool RAMBO was conducted. Experimental results showed that GBAW is able to identify alternative wires efficiently and obtained a very good logic optimization solution.

In Chapter 4, the GBAW-Partitioner (GP) was proposed with graph domain and logic domain. In the login domain, the logic gate information was used by the rewiring technique GBAW in order to alter the circuit so as to escape from local minima in graph domain. In the graph domain, the FM algorithm can be replaced by any other partitioner. Experimental results showed that the near optimal partitioning result by state-of-the-art Multi-level partitioner hMETIS-Kway can be further improved. Rewiring techniques can be applied in many different CAD and VLSI problems such as designs with timing constraints and layout-driven constraints.

In Chapter 5, the architectures of different switch-box architectures and routing algorithms were presented. The routability of Hyper-Universal Switch

Box (HUSB) was studied and was proved its optimality. Extensive experimental results were done by using the Disjoint S-Box and H'USB on the VPR FPGA router.

6.2 Future work

6.2.1 Alternative Wiring

The alternative wires are usually closed to the wire to be removed. More patterns should be located and put into GBAW in order to enhance the searching power. However, the performance of GBAW should be put into consideration. A good data structure and organization of pattern members should be built in order to reduce runtime of GBAW. As a result, careful experiments should be conducted and ensure the integrity and performance of GBAW.

6.2.2 Partitioning Quality

Partitioning is always the first step in physical design and most partitioning algorithms can be transformed or mapped with multi-level approaches. The bottleneck of the best partitioning result is the lack of using logic domain information. Thus, it is possible to integrate rewiring technique with other partitioning algorithms. Improved results are expected.

6.2.3 Routing Devices Studies

Higher routability means that the same circuit design can be put into the same FPGA with less physical wire segments. It also helps to reduce the heat dissipation and total path delay. Nowadays, XC4000 architecture FPGA are going to be replaced by newly released Virtex, Virtex-E and Virtex-II FPGAs. The next research step in FPGA routing should be the investigation on the methods which improve the routing of new FPGAs.

Bibliography

[AHK97a] Charles J. Alpert, Jen-Hsin Huang, and Andrew B. Kahng. The ISPD98 circuit benchmark suite. In *Proceedings International Symposium on Physical Design*, pages 80–85, 1997.

[AHK97b] Charles J. Alpert, Jen-Hsin Huang, and Andrew B. Kahng. Multilevel circuit partitioning. In *Proceedings 34th ACM/IEEE Design Automation Conference*, 1997.

[AK95a] Charles J. Alpert and Andrew B. Kahng. Multiway partitioning via geometric embeddings, orderings, and dynamic programming. *IEEE Transactions on computer-Aided Design*, 14(11):1342–1358, Nov 1995.

[AK95b] Charles J. Alpert and Andrew B. Kahng. Recent directions in netlist partitioning. *Integration, the VLSI Journal*, 19(1-2):1–81, 1995.

[alt90] *The Maximalist Handbook.* Altera Corp., 1990.

[AR95] M.J. Alexander and Gabriel Robins. New performance FPGA routing algorithms. *Proceedings of Design Automation Conference*, pages 562–567, 1995.

[AY95] Charles J. Alpert and S. Z. Yao. Spectral partitioning: The more eigenvectors, the better. In *Proceedings 32nd ACM/IEEE Design Automation Conference*, pages 195–200, 1995.

[Bar85] E. R. Barnes. An algorithm for partitioning the nodes of a graph. *SIAM Journal on Algorithm and Discrete Method*, 3:541–550, Dec 1985.

[BB91] K. P. Belkhale and P. Banerjee. Parallel algorithms for VLSI circuit extraction. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(5):604–618, May 1991.

[BBH$^+$88] K. A. Bartlett, R. K. Brayton, G. D. Hachtel, R. M. Jacoby, C. R. Morrison, R. L. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang. Multilevel logic minimization using implicit don't cares. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 7(6):723–740, June 1988.

[BFRV92] S. Brown, R.J. Francise, J. Rose, and Z.G. Vranesic. *Field-Programmable Gate Arrays.* Kluwer-Academic Publisher, Boston MA, 1992.

[BHSV90] R. K. Brayton, G. D. Hachtel, and A. Sangiovanni-Vincentelli. Multilevel logic synthesis. *Proceedings of the IEEE*, 78(2):264–300, Feb 1990.

[BLSV92] M. Beardslee, B. Lin, and A. Sangiovanni-Vincentelli. Communication based logic partitioning. In *EURO-DAC '92. European*, pages 32–37, 1992.

[BM76] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications.* London: Macmillan Press, 1976.

[BR97] Vaughn Betz and Jonathan Rose. A new packing, placement and routing tool for FPGA research. In *Seventh International Workshop on Field-Programmable Logic and Applications (Available for download from http://www.eecg.toronto.edu/~jayar/software.html)*, pages 213–222, 1997.

[BRSVW87] R. K. Brayton, R. L. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang. MIS: A multiple-level logic optimization system. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(6):1062–1081, Nov 1987.

[BRV92] S. Brown, J. Rose, and Z. G. Vranesic. A detailed router for field-programmable gate arrays. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11:620–628, 1992.

[Bry86] R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computer*, pages 677–691, Aug 1986.

[BS93] S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. In *Proceedings 6th SIAM Conf. Parallel Processing for Scientific Computing*, pages 771–718, 1993.

[BT91] C. L. Berman and L. H. Trevillyan. Global flow optimization in automatic logic design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(5):557–564, May 1991.

[CCMS93] D. I. Cheng, S. C. Chang, and M. Marek-Sadowska. Paritioning combinational circuits in graph and logic domains. In *Proceedings SASIMI-93*, pages 404–412, 1993.

[CCWM97] Shih-Chieh Chang, Kwang-Ting Cheng, Nam-Sung Woo, and Marek-Sadowska M. Postlayout logic restructuring using alternative wires. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(6):587–596, June 1997.

[CCWMS94a] S. C. Chang, K. T. Cheng, N. S. Woo, and M. Marek-Sadowska. Layout driven logic synthesis for FPGA. In *Proceedings 31th ACM/IEEE Design Automation Conference*, pages 308–313, June 1994.

[CCWMS94b] Shih Chieh Chang, Kwang Ting Cheng, Nam Sung Woo, and M. Marek-Sadowska. Layout driven logic synthesis for FPGAs. In *Proceedings 30th Design Automation Conference*, pages 308–313, June 1994.

[CE93] Kwang Ting Cheng and Luis A. Entrena. Multi-level logic optimization by redundancy addition and removal. In *Proceedings EDAC-93*, pages 373–377, Feb 1993.

[CGMS96] S. C. Chang, L. P. V. Ginneken, and M. Marek-Sadowska. Fast boolean optimization by rewiring. In *Proceedings of the International Conference on Computer-Aided Design*, pages 262–269, 1996.

[CKM00] Andrew E. Caldwell, Andrew B. Kahng, and Igor L. Markov. Improved algorithms for hypergraph bipartitioning. In *Proceedings Asia South Pacific Design Automation Conference 2000*, 2000.

[CL00a] Jason Cong and Sung Kyu Lim. Edge separability based circuit clustering with application to circuit partitioning. In *Proceedings Asia South Pacific Design Automation Conference 2000*, 2000.

[CL00b] Jason Cong and Sung Kyu Lim. Performance driven multiway partitioning. In *Proceedings Asia South Pacific Design Automation Conference 2000*, 2000.

[CLMS95] D. I. Cheng, C. C. Lin, and M. Marek-Sadowska. Circuit partitioning with logic perturbation. In *Proceedings of the International Conference on Computer-Aided Design*, pages 650–655, 1995.

[CLWL95] Ching-Dong Chen, Yuh-Sheng Lee, A.C.-H. Wu, and Youn-Long Lin. TRACER-fpga: a router for RAM-based FPGA's. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 14(3):371–374, 1995.

[CMS94] S. C. Chang and M. Marek-Sadowska. Perturb and simplify: Multi-level boolean network optimizer. In *Proceedings of the International Conference on Computer-Aided Design*, pages 2–4, Nov 1994.

[CMSC96] S. C. Chang, M. Marek-Sadowska, and K. T. Cheng. Perturb and simplify: Multilevel boolean network optimizer. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1494–1504, Dec 1996.

[CSZ94] P. K. Chan, Martine D. F. Schlag, and J. Y. Zien. Spectral k-way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1088–1096, 1994.

[CvGLMS99] Shih-Chieh Chang, van Ginneken, L.P.P.P., and M. Marek-Sadowska. Circuit optimization by rewiring. *IEEE Transactions on Computer*, 48(9):962–969, September 1999.

[CWW96] Y. W. Chang, D. F. Wong, and C. K. Wong. Universal switch modules for fpga design. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 1(1):80–101, Jan 1996.

[DD96] S. Dutt and W. Deng. VLSI circuit partitioning by cluster-removal using iterative improvement techniques. In *Proceedings of the International Conference on Computer-Aided Design*, pages 194–200, 1996.

[ea88] H. Hsieh et al. A 9000-gate user-programmable gate array. In *Proceedings of CICC*, pages 15.3.1–15.3.7, 1988.

[ea89] A. El Gamal et al. An architecture for electrially configurable gate arrays. *IEEE Transactions on Solid-State Circuits*, 24(2):394–398, 1989.

[EC93] Luis A. Entrena and Kwang Ting Cheng. Sequential logic optimization by redundancy addition and removal. In *Proceedings of the International Conference on Computer-Aided Design*, pages 310–315, Nov 1993.

[EC95] Luis A. Entrena and Kwang Ting Cheng. Combinational and sequential logic optimization by redundancy addition and removal. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 14(7):909–916, July 1995.

[EEOU96] L.A. Entrena, J.A. Espejo, E. Olias, and J. Uceda. Timing optimization by an improved redundancy addition and removal technique. In *Design Automation Conference, 1996, with EURO-VHDL '96 and Exhibition, Proceedings EURO-DAC '96*, pages 342–347, 1996.

[EHS97] Morgan Enos, Scott Hauck, and Majid Sarrafzadeh. Replication for logic bipartitioning. In *Proceedings of the International Conference on Computer-Aided Design*, 1997.

[EMHB95] C. Ebeling, L. McMurchie, S. A. Hauck, and S. Burns. Placement and routing tools for the triptych FPGA. *IEEE Transactions on Very Large Scale Integration*, pages 473–482, 1995.

[FHL] H.B. Fan, P. Haxell, and J. Liu. The global routng – a combinatorial design problem (submitted).

[FLW00] H. Fan, J. Liu, and Y. L. Wu. General models for optimum arbitrary-dimension FPGA switch box designs. In *Proceedings of the International Conference on Computer-Aided Design*, pages 93–98, 2000.

[FM82] C. M. Fiduccia and R. M. Mattheyses. A linear time heuristic for improving network partitions. In *Proceedings 19th ACM/IEEE Design Automation Conference*, pages 175–181, 1982.

[Hal70] K. M. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17:219–229, 1970.

[HK91] L. Hagen and A. B. Kahng. Fast spectral methods for ratio cut partitioning and clustering. In *Proceedings of the International Conference on Computer-Aided Design*, pages 10–13, 1991.

[HK92] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 11(9):1074–1085, Sept 1992.

[HL93a] B. Hendrickson and R. Leland. The chaco user's guide. *Sandia Nat. Labs., Albuquerque, NM, Technical Report SAND93-2339*, 1993.

[HL93b] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. *Sandia Nat. Labs., Albuquerque, NM, Technical Report SAND93-1301*, 1993.

[HS96] Gary D. Hachtel and Fabio Somenzi. *Logic Synthesis and Verification Algorithms.* Kluwer-Academic Publisher, Boston MA, 1996.

[KAKS97] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proceedings 34th ACM/IEEE Design Automation Conference*, pages 526–529, 1997.

[KBZ94] R. Kuznar, F. Brglez, and B. Zajc. Multi-way netlist partitioning into heterogeneous FPGAs and minimization of total device cost and interconnect. In *Proceedings 31th ACM/IEEE Design Automation Conference*, pages 238–243, 1994.

[Keu87] K. Keutzer. DAGON: Technology binding and local optimization by dag matching. In *Proceedings 24th Design Automation Conference*, pages 341–347, 1987.

[KK95a] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. *Technical Report 95-037, Department of Computer Science, University of Minnesota*, 1995.

[KK95b] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *Technical Report 95-035, Department of Computer Science, University of Minnesota*, 1995.

[KK95c] G. Karypis and V. Kumar. Multilevel graph partitioning schemes. In *Proceedings of the 1995 International Symposium on Physical Design*, pages 113–122, 1995.

[KK95d] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Technical Report 95-064, Department of Computer Science, University of Minnesota*, 1995.

[KK96] G. Karypis and V. Kumar. Parallel multilevel graph partitioning. In *Proceedings of International Parallel Processing Symposium*, 1996.

[KK99] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proceedings 36th ACM/IEEE Design Automation Conference*, 1999.

[KL70] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49:291–307, 1970.

[KN91] C. Kring and A. R. Newton. A cell-replicating approach to mincut-based circuit partitioning. In *Proceedings of the International Conference on Computer-Aided Design*, pages 2–5, 1991.

[KP94] W. Kunz and D. K. Pradham. Recursive learning: An attractive alternative to the decision tree for test generation for digital circuit. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1143–1158, Sept 1994.

[Kri84] B. Krishnamurthy. An improved min-cut algorithm for partitioning VLSI networks. *IEEE Transactions on Computers*, 33(5):438–446, 1984.

[KSM97] W. Kunz, D. Stoffel, and P. R. Menon. Logic optimization and equivalence checking by implication analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(3):266–281, March 1997.

[LB93] G. Lemieux and S. Brown. A detailed routing algorithm for allocating wire segments in field-programmable gate arrays. In *Proceedings ACM/SIGDA Physical Design Workshop, Lake Arrowhead, CA*, pages 215–226, 1993.

[Lee61] C. Y. Lee. An algorithm for path connections and its applications. *IRE Transactions on Electronic Computer*, EC10:346–365, 1961.

[LW95] Y.S. Lee and Allen C.H. Wu. A performance and routability driven router for FPGAs considering path delays. *Proceedings 32th ACM/IEEE Design Automation Conference*, pages 557–561, 1995.

[LWB00] Wangning Long, Yu-Liang Wu, and Jinian Bian. IBAW: An implication-tree based alternative-wiring logic transformation algorithm. In *Proceedings Asia South Pacific Design Automation Conference 2000*, 2000.

[Mic94] Giovanni De Micheli. *Synthesis and Optimization of Digital Circuits.* McGraw-Hill, New York, 1994.

[MKLC89] S. Muroga, Y. Kambayashi, H. C. Lai, and J. N. Culliney. The transduction method-design of logic networks based on permissible functions. *IEEE Transactions on Computer*, 38(10):1404–1424, Oct 1989.

[MW96] Wai-Kei Mak and D. F. Wong. Minimum replication min-cut partitioning. In *Proceedings of the International Conference on Computer-Aided Design*, 1996.

[PS79] B. N. Parlett and D. S. Scott. The lanczos algorithm with selective orthogonalization. *Mathematics and Computations*, 33(11):217–238, 1979.

[PWWY92] J.F Pan, Y.L. Wu, C.K. Wong, and G. Yan. On the optimal four-way switch box routing structures of FPGA greedy routing architectures. *Integration, the VLSI Journal*, 25:137–159, 1992.

[RB91] J. Rose and S. Brown. Flexibility of interconnection structures for field-programmable gate arrays. *IEEE Transactions on Solid-State Circuits*, 26(3):277–262, 1991.

[Ros90] J. S. Rose. Parallel global routing for standard cells. *IRE Transactions on Computer-Aided Design*, pages 1085–1095, Oct 1990.

[RSV85] J. S. Rose, W. M. Snelgrove, and Z. G. Vranesic. ALTOR: An automatic standard cell layout program. In *Canadian Conference on Very Large Scale Integration*, pages 169–173, 1985.

[San89] L. A. Sanchis. Multi-way network partitioning. *IEEE Transactions on Computers*, 38(1):62–81, 1989.

[She98] Naveed Sherwani. *Algorithms for VLSI Physical Design Automation.* Kluwer-Academic Publisher, Boston MA, 3 edition, 1998.

[SSLea92] E. M. Sentovich, K. J. Singh, L. Lavagno, and et. al. SIS: A system for sequential circuit synthesis. *ERL Memorandum No. UCB/ERL*, M92/41, 1992.

[SW96] M. Sarrafzadeh and C. K. Wong. *An Introduction to VLSI Physical Design.* McGRAW-HILL International Editions, Electrical Engineering Series, 1 edition, 1996.

[WA98] Sverre Wichlund and Einar J. Aas. On multilevel circuit partitioning. In *Proceedings of the International Conference on Computer-Aided Design*, pages 505–511, 1998.

[WC91] Y. C. Wei and C. K. Cheng. Ratio cut partitioning for hierarchical designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(7):911–921, July 1991.

[WC94] Y.L. Wu and D. Chang. On NP-Completeness of 2-d FPGA routing architectures and a novel solution. In *Proceedings of International Conference on Computer-Aided-Design*, pages 362–366, 1994.

[WF99] Y. L. Wu and Hongbing Fan. On local configuration analysis of alternative wires in boolean networks. *ITC-CSCC'99*, pages 868–871, 1999.

[WLF00] Yu-Liang Wu, Wangning Long, and Hongbing Fan. A fast graph-based alternative wiring scheme for boolean networks. In *Proceedings IEEE/ACM International VLSI Design, 2000*, 2000.

[WMS97] Y.L. Wu and M. Marek-Sadowska. Routing for array type FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 16(5):506–518, 1997.

[WTMS94] Y.L. Wu, S. Tsukiyama, and M. Marek-Sadowska. On computational complexity of a detailed routing problem in two-dimensional FPGAs. In *Proceedings 4th Great Lakes Symposium VLSI*, March 1994.

[WTMS96] Y.L. Wu, S. Tsukiyama, and M. Marek-Sadowska. Graph based analysis of 2-d FPGA routing. *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, 15(1):33–44, 1996.

[WYC00] Y. L. Wu, X. L. Yuan, and D. I. Cheng. Circuit partitioning with coupled logic restructuring techniques. In *Proceedings Asia South Pacific Design Automation Conference 2000*, 2000.

[Yan91] S. Yang. Logic synthesis and optimization benchmarks version 3.0. *Technical Report, Microelectronics Centre of North Carolina*, 1991.

[YCL92] C. W. Yeh, C. K. Cheng, and T. T. Y. Lin. A probabilistic multicommodity-flow solution to circuit clustering problems. In *Proceedings of the International Conference on Computer-Aided Design*, pages 428–431, 1992.

[YW98] Hannah Honghua Yang and D. F. Wong. Optimal min-area min-cut replication in partitioned circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(11):1175–1183, November 1998.

[ZCS99] J. Y. Zien, Pak K. Chan, and Martine D. F. Schlag. Multilevel spectral hypergraph partitioning with arbitrary vertex sizes. *IEEE Transactions on Computer-Aided Design*, 18(9):1389–1399, Sept 1999.

[ZSC96] J. Y. Zien, Martine D. F. Schlag, and P. K. Chan. Multi-level spectral hypergraph partitioning with arbitrary vertex sizes. In *Proceedings of the International Conference on Computer-Aided Design*, 1996.

Appendix A

5xp1 - Berkeley Logic Interchange Format (BLIF)

```
.model top
.inputs i_5_ i_6_ i_3_ i_4_ i_1_ i_2_ i_0_
.outputs o_1_ o_2_ o_0_ o_9_ o_7_ o_8_ o_5_ o_6_ o_3_ o_4_
.default_input_arrival 0.00 0.00
.default_output_required 0.00 0.00
.default_input_drive 0.30 0.30
.default_output_load 1.00
.default_max_input_load 999.00
.names i_6_ [565]
0 1
.names i_0_ [570]
0 1
.names i_2_ [571]
0 1
.names [565] [570] [571] i_5_ [1079]
0— 1
-0– 1
–0- 1
—0 1
.names i_3_ o_8_
0 1
.names [565] [570] o_8_ i_5_ [1044]
0— 1
-0– 1
–0- 1
—0 1
.names i_1_ [576]
0 1
.names [570] [576] [571] [578]
0– 1
-0- 1
–0 1
.names i_4_ [563]
0 1
.names [563] i_5_ [579]
0- 1
-0 1
```

```
.names [578] [579] [1163]
1- 1
-1 1
.names [1079] [1044] [1163] [581]
0– 1
-0- 1
–0 1
.names i_5_ [564]
0 1
.names [564] i_4_ [586]
0- 1
-0 1
.names i_0_ i_2_ i_1_ [592]
0– 1
-0- 1
–0 1
.names [586] [592] [1250]
1- 1
-1 1
.names i_6_ i_0_ [564] i_2_ [1246]
0— 1
-0– 1
–0- 1
—0 1
.names i_6_ i_0_ [564] i_1_ [1248]
0— 1
-0– 1
–0- 1
—0 1
.names [1250] [1246] [1248] [598]
0– 1
-0- 1
–0 1
.names [570] o_8_ [576] [583]
0– 1
-0- 1
–0 1
.names [579] [583] n_n101
00 1
.names [565] [570] [576] i_5_ [1110]
0— 1
-0– 1
–0- 1
—0 1
.names i_0_ i_3_ i_1_ [588]
0– 1
-0- 1
–0 1
.names [586] [588] [1183]
1- 1
-1 1
.names [1110] [1183] [589]
0- 1
-0 1
.names [581] [598] n_n101 [589] n_n47
0000 1
.names [563] [565] i_5_ [567]
0– 1
-0- 1
```

```
–0 1
.names i_4_ [564] i_6_ [1234]
0– 1
-0- 1
–0 1
.names n_n47 [567] [1234] o_1_
0– 1
-0- 1
–0 1
.names i_6_ i_0_ i_4_ i_2_ [1197]
0— 1
-0– 1
–0- 1
—0 1
.names [565] [570] [571] i_4_ [1094]
0— 1
-0– 1
–0- 1
—0 1
.names i_6_ i_0_ i_4_ i_1_ [1195]
0— 1
-0– 1
–0- 1
—0 1
.names [1197] [1094] [1195] [608]
0– 1
-0- 1
–0 1
.names i_0_ i_4_ i_5_ [620]
0– 1
-0- 1
–0 1
.names [565] [620] n_n95
00 1
.names [563] i_6_ [621]
0- 1
-0 1
.names [578] [621] n_n96
00 1
.names [583] [621] n_n97
00 1
.names n_n95 n_n96 n_n97 n_n57
000 1
.names [565] [570] o_8_ i_4_ [1097]
0— 1
-0– 1
–0- 1
—0 1
.names i_0_ i_1_ [571] n_n36
000 1
.names n_n36 i_4_ i_3_ i_5_ [1227]
0— 1
-0– 1
–0- 1
—0 1
.names i_6_ i_1_ i_4_ i_5_ [1229]
0— 1
-0– 1
–0- 1
```

```
—0 1
.names n_n57 [1097] [1227] [1229] [624]
0— 1
-0– 1
–0- 1
—0 1
.names [608] [624] n_n53
00 1
.names i_2_ i_3_ i_1_ [627]
0– 1
-0- 1
–0 1
.names [567] [627] n_n117
00 1
.names [564] [565] [628]
0- 1
-0 1
.names [628] i_2_ i_3_ [563] n_n118
0000 1
.names i_6_ i_1_ i_5_ [563] n_n119
0000 1
.names n_n117 n_n118 n_n119 n_n56
000 1
.names [564] i_6_ [635]
0- 1
-0 1
.names [635] i_2_ i_4_ i_1_ n_n123
0000 1
.names [565] i_0_ i_5_ i_4_ n_n124
0000 1
.names i_0_ [563] i_5_ [641]
0– 1
-0- 1
–0 1
.names i_6_ [641] n_n146
00 1
.names n_n123 n_n124 n_n146 n_n54
000 1
.names i_6_ i_0_ i_5_ [563] n_n120
0000 1
.names [565] [563] [646]
0- 1
-0 1
.names [588] [646] n_n121
00 1
.names [592] [646] n_n122
00 1
.names n_n120 n_n121 n_n122 n_n55
000 1
.names n_n53 n_n56 n_n54 n_n55 o_2_
0— 1
-0– 1
–0- 1
—0 1
.names [586] [1203]
0 1
.names [565] [576] i_4_ [564] n_n130
0000 1
.names [565] [641] n_n131
```

```
00 1
.names i_6_ i_0_ i_1_ [563] n_n132
0000 1
.names [1203] n_n130 n_n131 n_n132 [308]
0000 1
.names i_4_ o_8_ [318]
00 1
.names i_2_ [318] i_5_ i_6_ [1100]
0— 1
-0– 1
–0- 1
—0 1
.names [308] [1100] [1094] [1097] o_0_
0— 1
-0– 1
–0- 1
—0 1
.names [627] [1276]
0 1
.names [1276] i_4_ i_5_ [1225]
0– 1
-0- 1
–0 1
.names i_4_ i_5_ i_6_ [1223]
0– 1
-0- 1
–0 1
.names [1225] [620] [1223] o_9_
0– 1
-0- 1
–0 1
.names o_8_ i_2_ [1038]
0- 1
-0 1
.names [571] i_3_ [1065]
0- 1
-0 1
.names [1038] [1065] o_7_
0- 1
-0 1
.names i_0_ [576] [571] [667]
0– 1
-0- 1
–0 1
.names i_0_ o_8_ [571] [669]
0– 1
-0- 1
–0 1
.names [669] [1067]
0 1
.names [570] [571] i_1_ [671]
0– 1
-0- 1
–0 1
.names o_8_ [671] n_n139
00 1
.names [1067] n_n139 [262]
00 1
.names [570] o_8_ i_2_ [1104]
```

```
0– 1
-0- 1
–0 1
.names i_0_ i_3_ i_2_ [1106]
0– 1
-0- 1
–0 1
.names [667] [262] [1104] [1106] o_5_
0— 1
-0– 1
–0- 1
—0 1
.names i_2_ i_1_ [1212]
0- 1
-0 1
.names o_8_ i_1_ [1208]
0- 1
-0 1
.names [571] [576] i_3_ [1216]
0– 1
-0- 1
–0 1
.names [1212] [1208] [1216] o_6_
0– 1
-0- 1
–0 1
.names i_5_ [667] [1115]
1- 1
-1 1
.names i_6_ [570] [564] i_1_ [1113]
0— 1
-0– 1
–0- 1
—0 1
.names [1115] [1110] [1113] [682]
0– 1
-0- 1
–0 1
.names [628] [669] n_n113
00 1
.names i_2_ i_1_ [683]
0- 1
-0 1
.names i_0_ [683] i_5_ o_8_ n_n114
0000 1
.names [635] i_0_ o_8_ [571] n_n115
0000 1
.names [682] n_n113 n_n114 n_n115 n_n60
0000 1
.names [564] [588] n_n111
00 1
.names [564] [592] n_n112
00 1
.names n_n111 n_n112 [194]
00 1
.names [1044] [194] [698]
0- 1
-0 1
.names [564] [583] n_n89
```

```
00 1
.names i_6_ [570] i_5_ i_1_ n_n90
0000 1
.names [1079] [1080]
0 1
.names [698] n_n89 n_n90 [1080] [172]
0000 1
.names i_6_ i_0_ i_5_ i_2_ [1258]
0— 1
-0– 1
–0- 1
—0 1
.names [565] [570] [564] [576] n_n87
0000 1
.names [564] [578] n_n88
00 1
.names n_n87 n_n88 [176]
00 1
.names n_n60 [172] [1258] [176] o_3_
0— 1
-0– 1
–0- 1
—0 1
.names [571] [576] i_6_ [1218]
0– 1
-0- 1
–0 1
.names i_6_ [671] n_n84
00 1
.names i_6_ i_0_ i_3_ [576] n_n85
0000 1
.names i_6_ [570] i_1_ [571] n_n106
0000 1
.names n_n84 n_n85 n_n106 n_n70
000 1
.names [565] [588] n_n107
00 1
.names [565] [592] n_n108
00 1
.names [565] [627] n_n109
00 1
.names n_n107 n_n108 n_n109 n_n69
000 1
.names [565] [583] n_n110
00 1
.names i_6_ i_2_ i_3_ [576] n_n142
0000 1
.names i_6_ [571] i_1_ o_8_ n_n143
0000 1
.names n_n110 n_n142 n_n143 n_n68
000 1
.names [1218] n_n70 n_n69 n_n68 o_4_
0— 1
-0– 1
–0- 1
—0 1
.end
```

Appendix B

Proof of some 2-local patterns

The construction of new minimal patterns is a joint work with another M. Phil. student, Cliff Chin-Ngai Sze, under the supervision of Professor Yu-Liang Wu, in the Department of Computer Science and Engineering. In this thesis, only partial patterns are shown for illustration.

Pattern 1

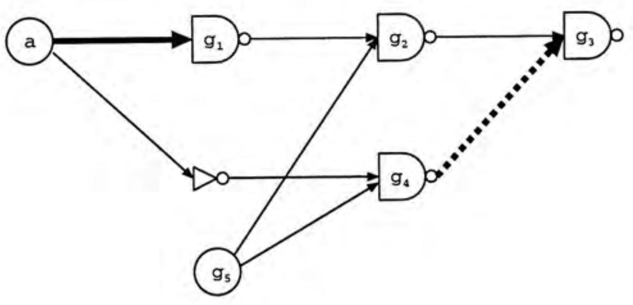


Figure B.1: New 2-local pattern 1

In Figure B.1, $a \rightarrow g_1$ is the target wire. let $g_1 = (a*x)'$ where x is the other inputs of g_1. $g_2 = (g_1*y)'$ where y is the other inputs of g_2. $g_3 = (g_2*z)'$ where z is the other inputs of g_3. Therefore, $g_3 = (((a*x)'y)'z)' = ((a'+x')y)'z)' = ((a'y+x'y)'z)' = ((a'y)'(x'y)'z)' = ((g_4)(x'y)z)'$

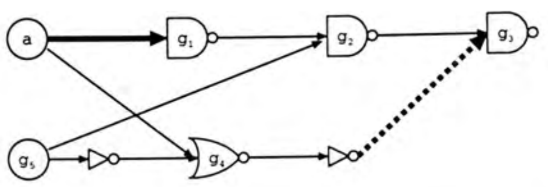


Figure B.2: New 2-local pattern 2

Pattern 2

In Figure B.2, $g_3 = (((a * x)'y)'z)' = ((a' + x')y)'z)' = ((a'y + x'y)'z)' = ((a'y)'(x'y)'z)' = ((a + y')(x'y)z)' = ((g_4)'(x'y)z)'$

Pattern 3

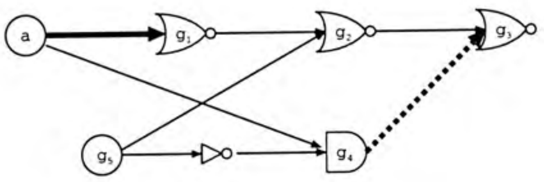


Figure B.3: New 2-local pattern 3

In Figure B.3, $g_3 = (((a+x)'+y)'+z)' = ((a+x)y'+z)' = (ay'+xy'+z)' = ((g_4) + xy' + z)'$

Pattern 4

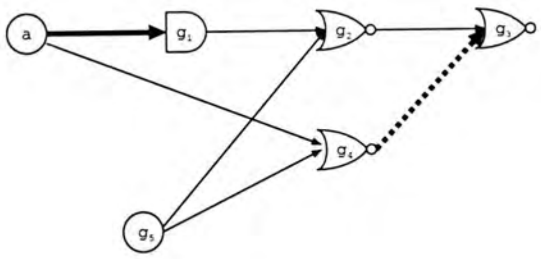


Figure B.4: New 2-local pattern 4

In Figure B.4, $g_3 = (((ax) + y)' + z)' = ((ax)'y' + z)' = ((a' + x')y' + z)' = (a'y' + x'y' + z)' = ((a + y)' + (x + y)' + z)' = ((g_4) + (x + y)' + z)'$

Appendix C

Illustrations of FM algorithm

In Figure C.1 and Figure C.2, it shows 2 pass of FM algorithm on 4 vertices with 6 weighted edges. The bucket structure is used to store the intermediate value for FM algorithm. The final cost is reduced to 6.

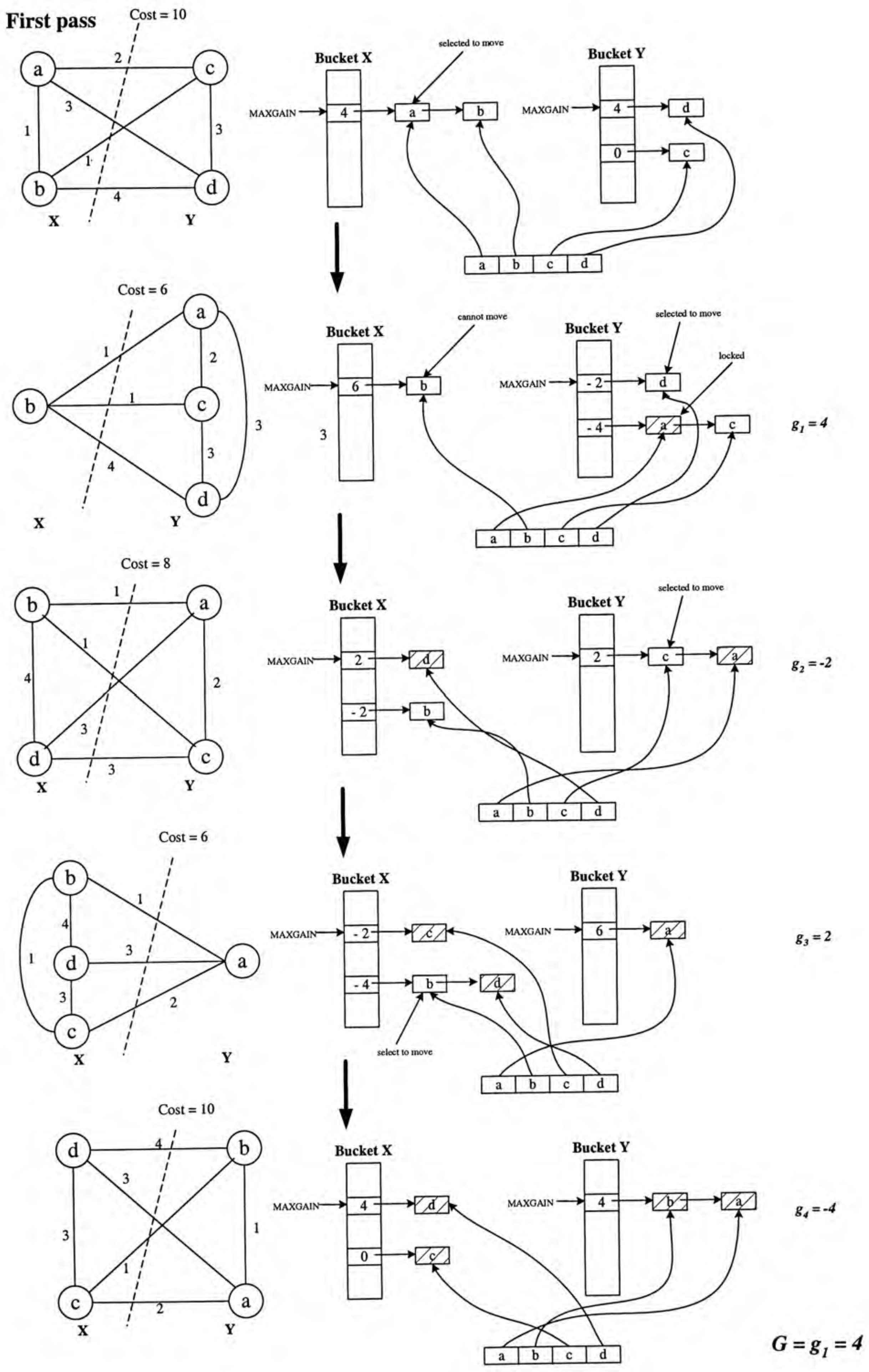


Figure C.1: Example of FM algorithm - First pass

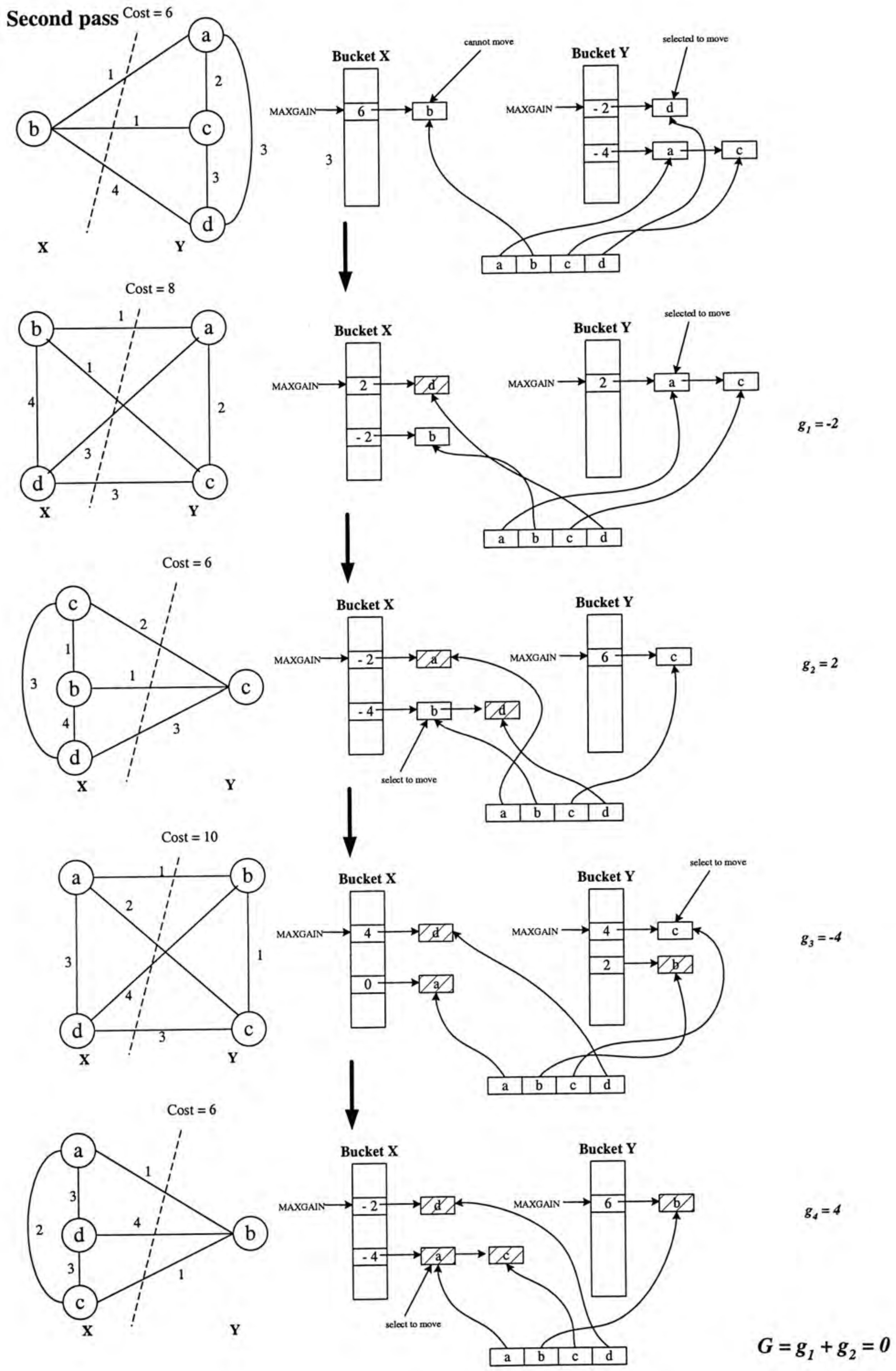


Figure C.2: Example of FM algorithm - Second pass

Appendix D

HUSB Structures

In this chapter, the Hyper-Univeral Switch Box (HUSB) H_i for i from 1 to 9 is shown. For i = 8, the HUSB is actually built by H_7 + H_1, and H_7 + H_2 build H_9.

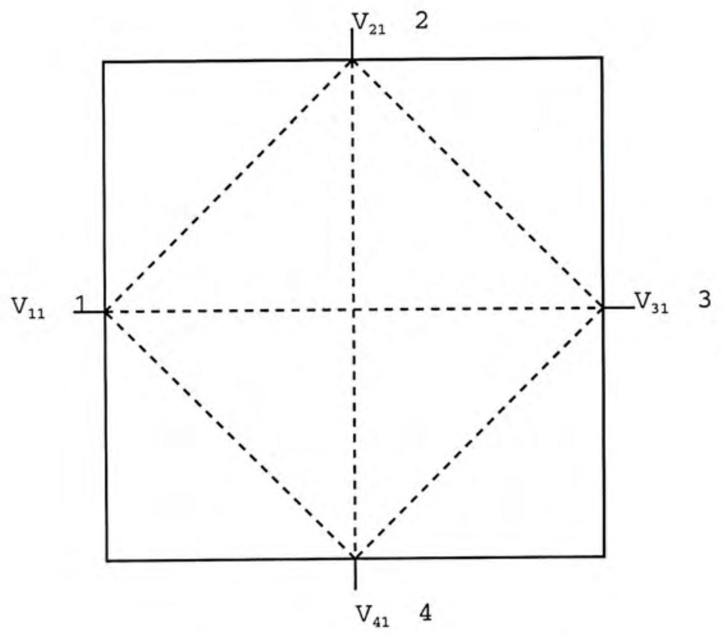


Figure D.1: H1 S-Box

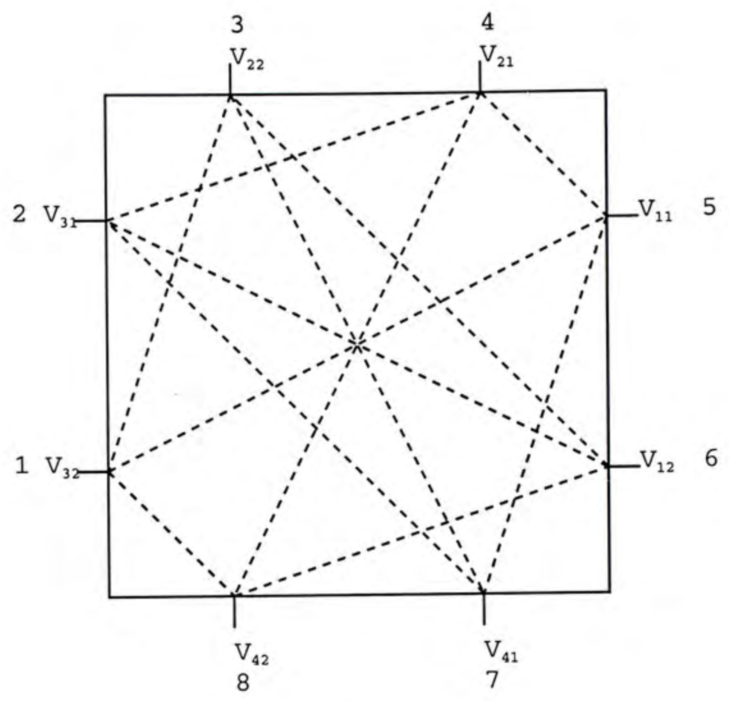


Figure D.2: H2 S-Box

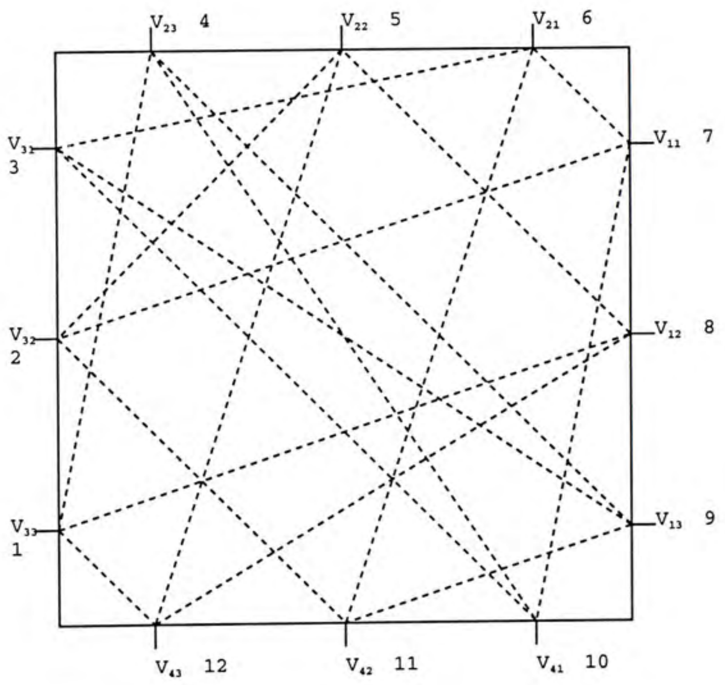


Figure D.3: H3 S-Box

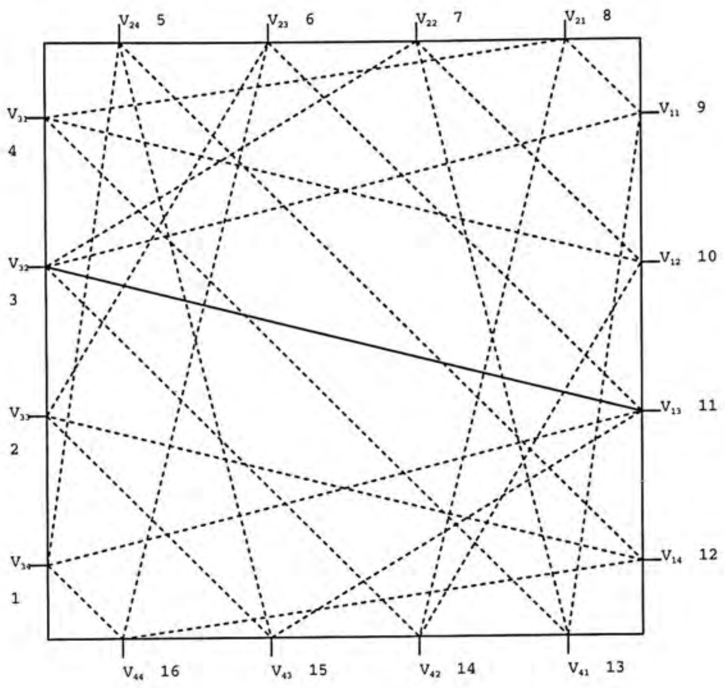


Figure D.4: H4 S-Box

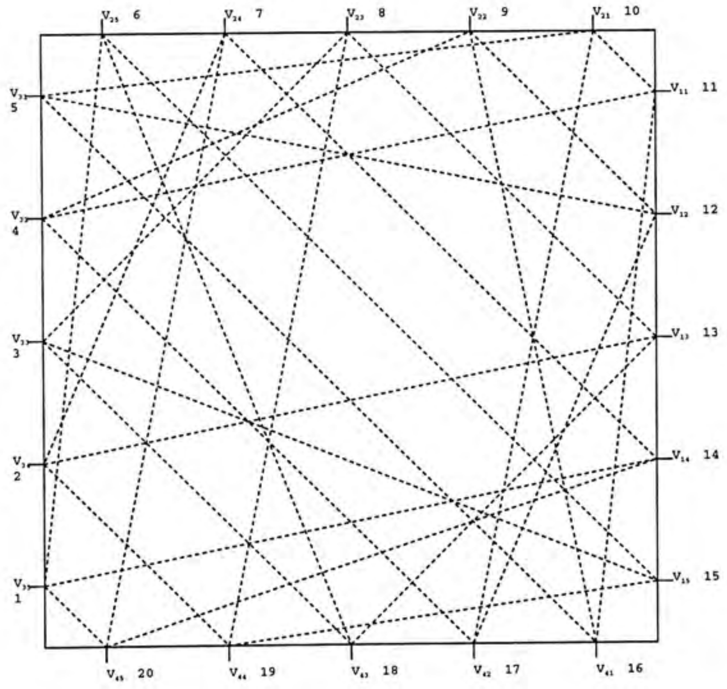


Figure D.5: H5 S-Box

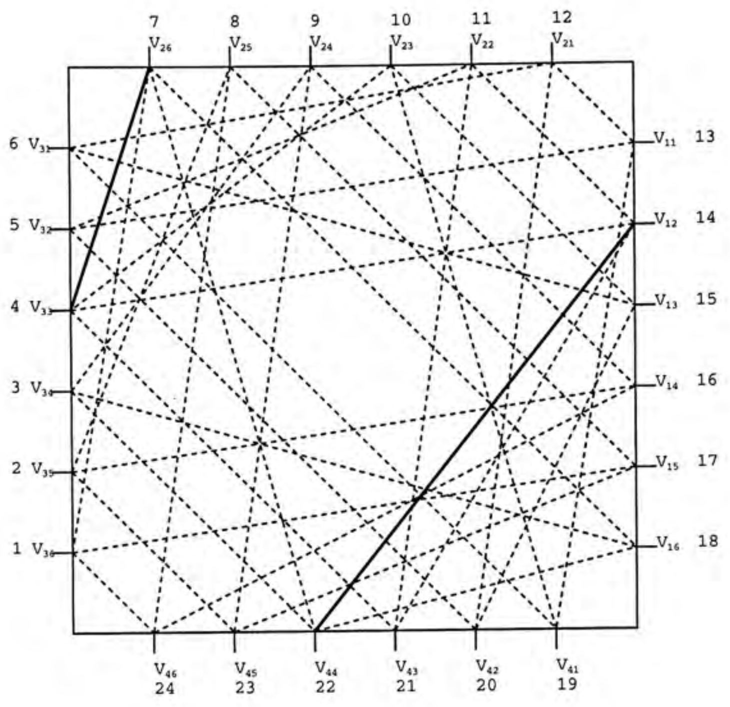


Figure D.6: H6 S-Box

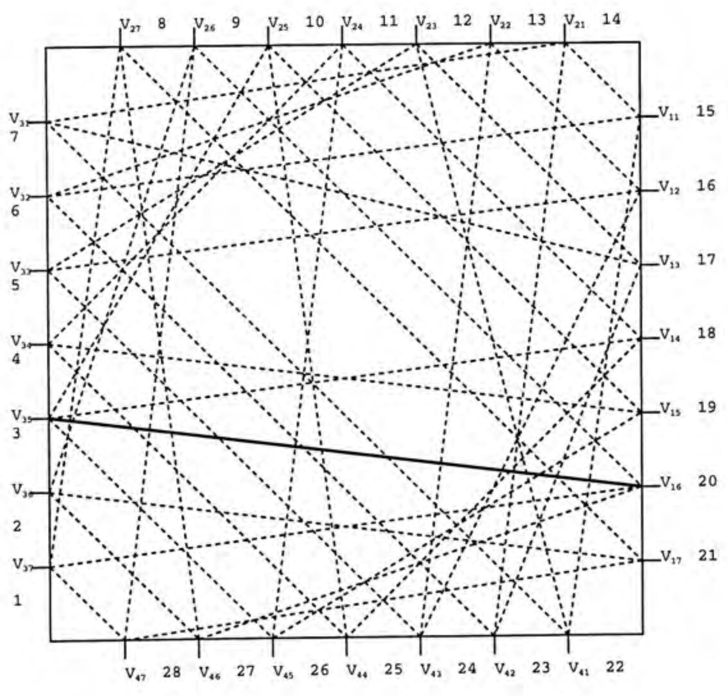

V_{27} 8 V_{26} 9 V_{25} 10 V_{24} 11 V_{23} 12 V_{22} 13 V_{21} 14
V_{31} 7
V_{32} 6
V_{33} 5
V_{34} 4
V_{35} 3
V_{36} 2
V_{37} 1
V_{11} 15
V_{12} 16
V_{13} 17
V_{14} 18
V_{15} 19
V_{16} 20
V_{17} 21
V_{47} 28 V_{46} 27 V_{45} 26 V_{44} 25 V_{43} 24 V_{42} 23 V_{41} 22

Figure D.7: H7 S-Box

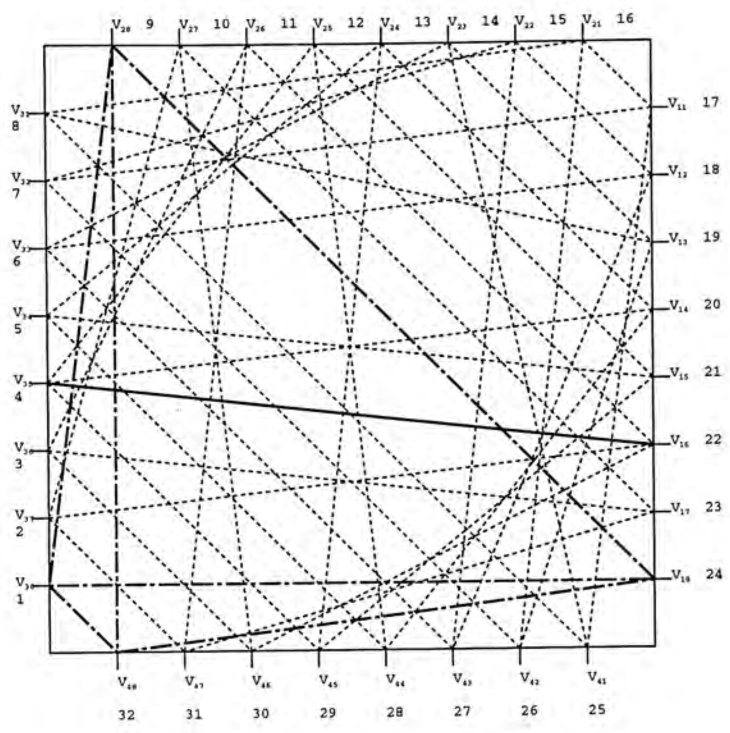


Figure D.8: H8 S-Box

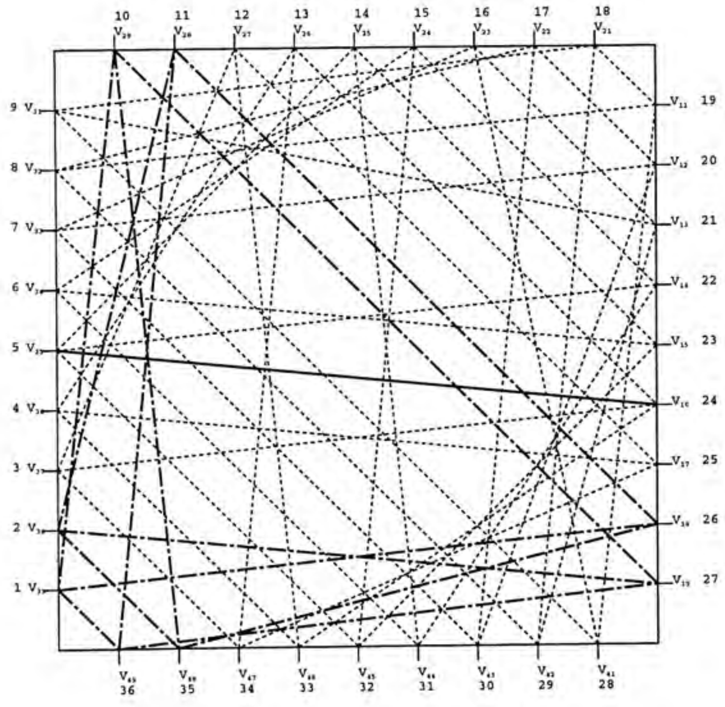


Figure D.9: H9 S-Box

Appendix E

Primitive minimal 4-way global routing Structures

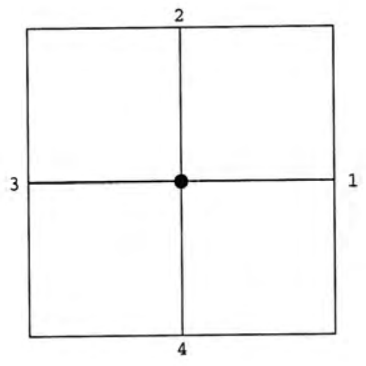


Figure E.1: GR_1^1

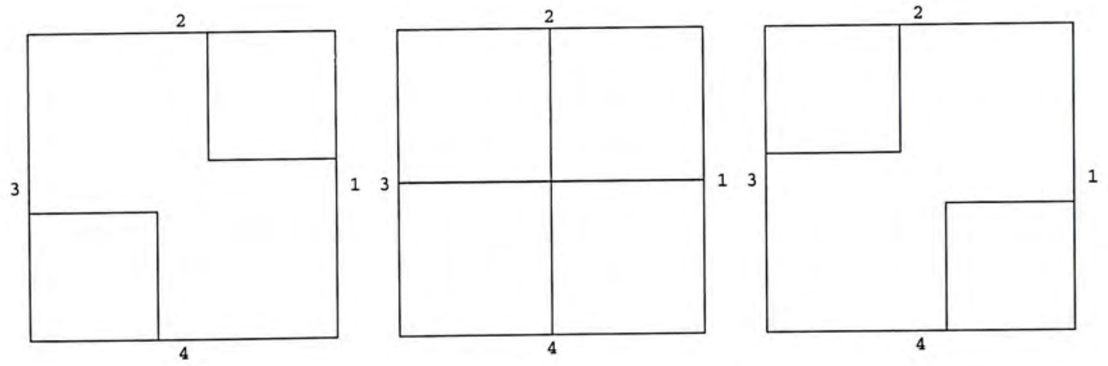


Figure E.2: $GR_{2,1}^1, GR_{2,2}^1, GR_{2,3}^1$

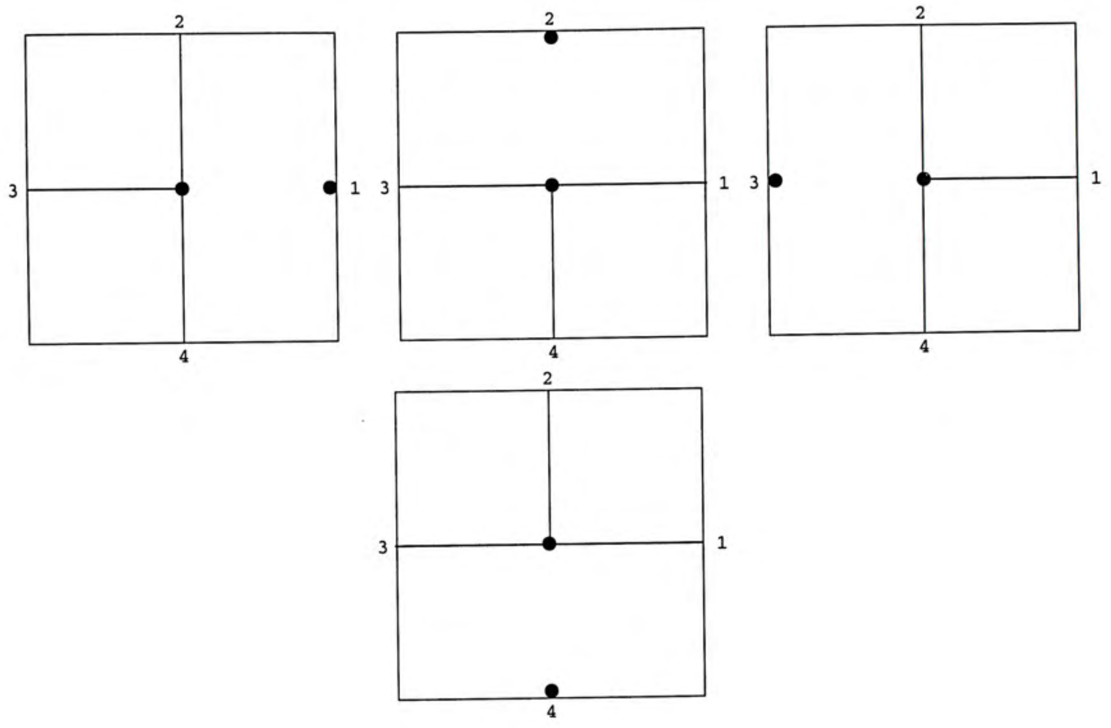


Figure E.3: $GR^1_{3,1}, GR^1_{3,2}, GR^1_{3,3}, GR^1_{3,4}$

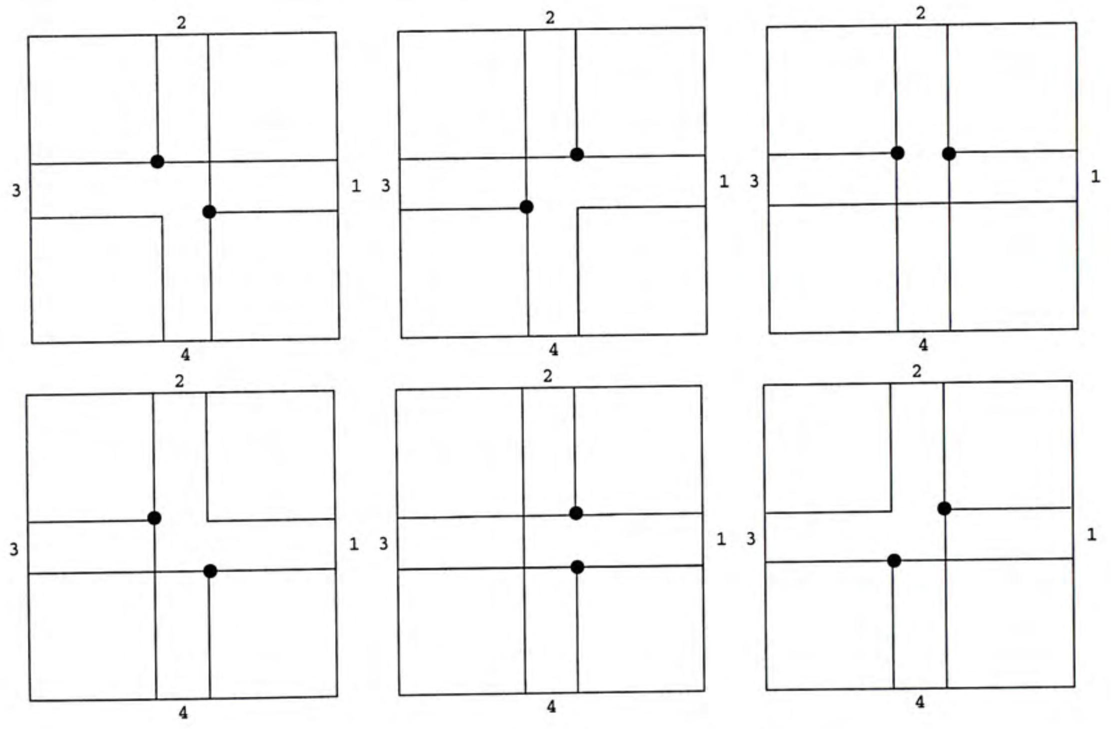


Figure E.4: $GR^2_{1,1}, GR^2_{1,2}, GR^2_{1,3}, GR^2_{1,4}, GR^2_{1,5}, GR^2_{1,6}$

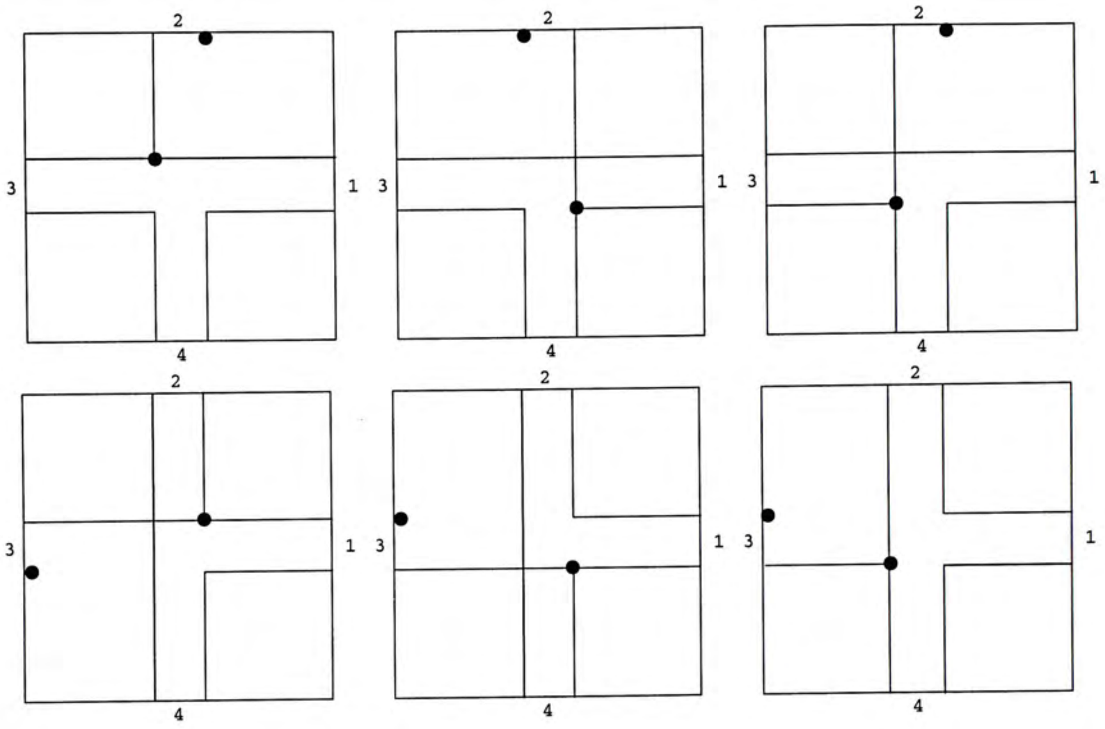


Figure E.5: $GR^2_{2,1}, GR^2_{2,2}, GR^2_{2,3}, GR^2_{2,4}, GR^2_{2,5}, GR^2_{2,6}$

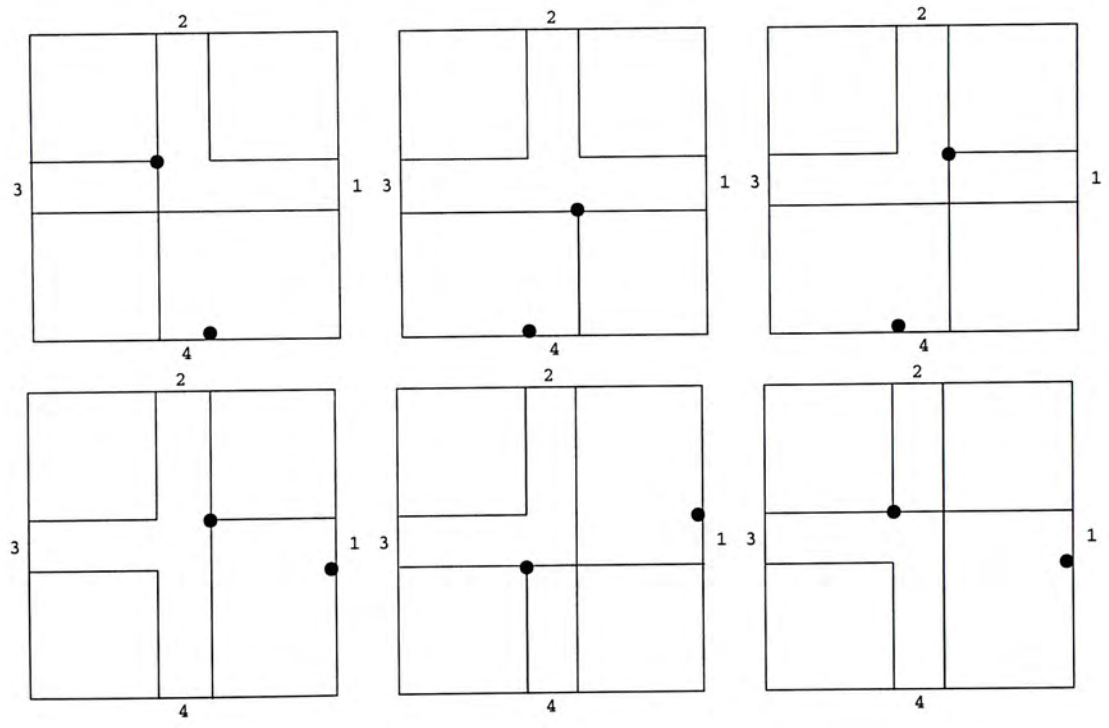


Figure E.6: $GR^2_{2,7}, GR^2_{2,8}, GR^2_{2,9}, GR^2_{2,10}, GR^2_{2,11}, GR^2_{2,12}$

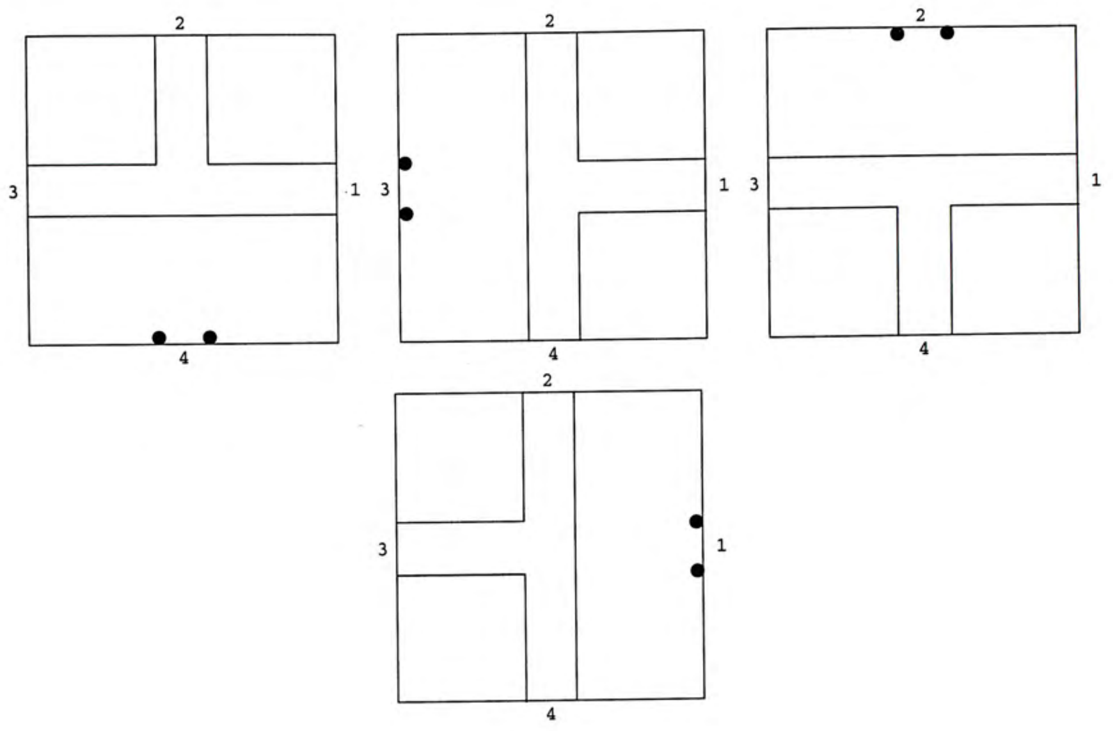


Figure E.7: $GR^2_{3,1}, GR^2_{3,2}, GR^2_{3,3}, GR^2_{3,4}$

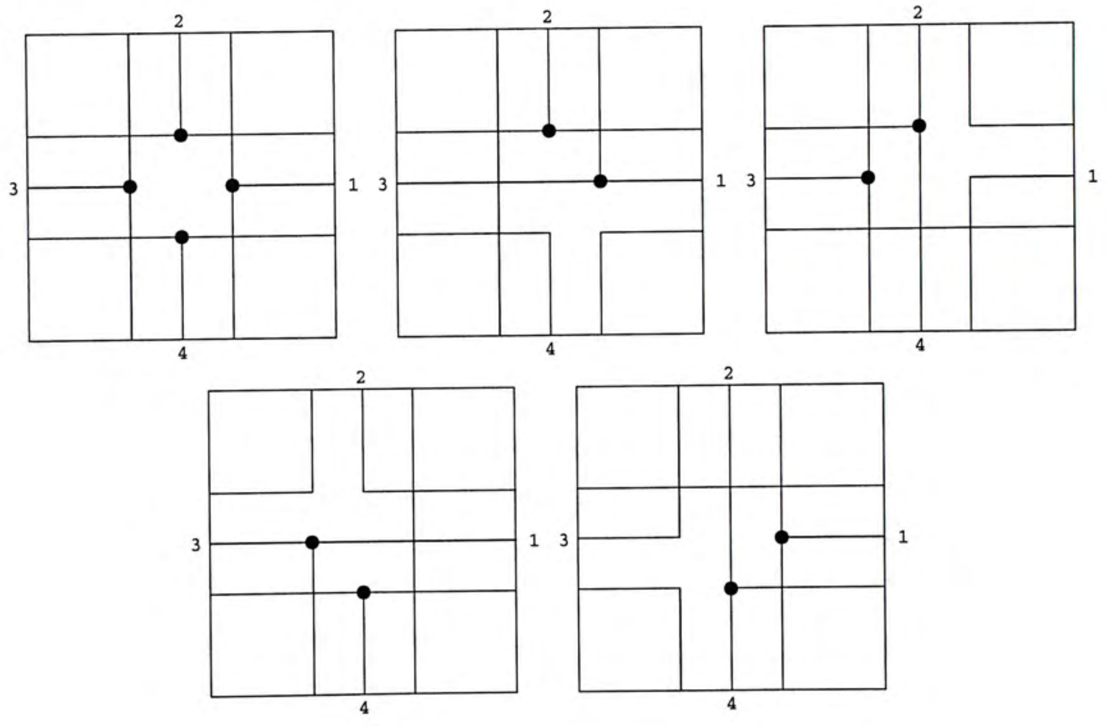


Figure E.8: $GR^3_1, GR^3_{2,1}, GR^3_{2,2}, GR^3_{2,3}, GR^3_{2,4}$