1-1-2010

Unidirectional Multi-Bit FPGA Architecture For Area Efficient Implementation of Datapath Circuits

Omesh Mutukuda
Ryerson University

UNIDIRECTIONAL MULTI-BIT FPGA ARCHITECTURE FOR AREA EFFICIENT IMPLEMENTATION OF DATAPATH CIRCUITS

by

Omesh Mutukuda

B. Sc. in Electrical and Computer Engineering,

University of Windsor, Windsor, ON, 2006

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Applied Science (MASc)

in the Program of

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2010

Author's Declaration

Abstract

UNIDIRECTIONAL MULTI-BIT FPGA ARCHITECTURE FOR AREA EFFICIENT IMPLEMENTATION OF DATAPATH CIRCUITS

Omesh Mutukuda

Master of Applied Science (MASc), 2010

Department of Electrical and Computer Engineering

Ryerson University

Field Programmable Gate Arrays (FPGAs) are increasingly being used to implement large datapath-oriented applications that are designed to process multiple-bit wide data. Studies have shown that the regularity of these multi-bit signals can be effectively exploited to reduce the implementation area of datapath circuits on FPGAs that employ the traditional bidirectional routing. Most of modern FPGAs, however, employ unidirectional routing tracks which are more area and delay efficient. No study has investigated the design of multi-bit routing resources that can effectively transport multiple-bit wide signals using unidirectional routing tracks. This paper presents such an investigation of architectures which employ multi-bit connections and unidirectional routing resources to exploit datapath regularity. It is experimentally shown that unidirectional multi-bit architectures are 8.6% more area efficient than the conventional architecture. Additionally, this paper determines the most are efficient proportion of multi-bit connections.

Acknowledgements

I would like to take this opportunity to express my gratitude and appreciation to my graduate supervisors, my family and friends.

It has been a privilege to work with Dr. Andy Ye and Dr. Gul Khan as they have made my post-graduate career a rewarding experience. I am especially grateful to Dr. Ye for his generosity, patience and keeping an open door in order to help me see through the challenges in my research. I would also like to thank Dr. Khan for the motivation and guidance on the practical aspects of my research. Their extensive knowledge and thoughtful advice will have made a remarkable influence on my entire engineering career.

This thesis would not have been possible if not for the moral and financial support from my parents Indra and Soma Mutukuda. Since childhood, my parents sacrificed their chances at many of life's opportunities (sometimes even their own happiness) in ensuring I had the freedom and ability to pursue my dreams. It is to them I would like to dedicate this thesis.

I would also like to thank my aunt Priyanthie for taking me out to enjoy the fine cuisine of Toronto and looking out for my general well being all while keeping a cheerful smile.

Finally, I would like to thank my friends David, Mustafa, Thuan and Sebastian for the good times full of youthful folly and hijinx. Their friendship, support and encouragement meant a great deal to me.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

FPGA	Field Programmable Gate Array
CLB	Configurable Logic Block
BLE	Basic Logic Element
MUX	Multiplexor
CAD	Computer-aided Design
VPR	Versatile Placement and Routing
CMS	Configuration Memory Sharing
MB-FPGA	Multi-bit Field Programmable Gate Array
LUT	Look-up Table
SRAM	Static Random Access Memory
ASIC	Application Specific Integrated Circuit
I/O	Input/Output

Chapter 1 : Introduction

Developed in 1984, Field Programmable Gate Arrays (FPGAs) are user programmable integrated circuit devices which have the ability to implement almost any digital circuit. FPGAs consist of blocks containing simple digital logic interconnected by channels of routing wires and programmable electrical switches. Circuits are implemented on FPGAs by programming the electrical switches to connect logic blocks together to form larger and more complex circuits. Programming requires only a few minutes thus making hardware verification and upgrades relatively effortless as design changes or corrections can be performed by reprogramming the FPGA. Comparable technologies such as standard cell-based ASICs (Application-Specific Integrated Circuit) require a new integrated circuit to be manufactured for each design. Additionally, FPGAs are programmed via specialized Computer-Aided Design (CAD) tools which are cheaper to obtain than comparable Electronic Design Automation (EDA) tools for ASIC designs. In summary, circuits designed on FPGAs have an advantage of lower non-reoccurring engineering (NRE) costs and shorter time-to-market. FPGAs are the ideal for implementation of small to medium volume designs or prototyping digital circuits prior to implementation using standard cell based technologies.

Modern FPGAs are densely packed devices, containing more than 800,000 logic elements with the ability to operate at clock speeds of 600MHz or more [1] [2]. Despite these features, the programmability of FPGAs incurs a cost. The addition of programmable logic and switches forces circuit implementations on FPGAs to be slower and consume more area than those implemented on ASICs. These programmable features contribute to the large size of FPGA implementations and thus make it more expensive to design applications for mass-production. Once programmed, it is observed that over 70% of the total FPGA area is devoted to the routing resources which interconnect the logic blocks [3]. The actual computational logic occupies the remaining FPGA area (which is typically less than 30%). This motivates FPGA designers to consider ways to reduce the area impact of routing resources as well as that of computational logic.

In general, FPGA are increasingly used to implement large arithmetic circuits which inturn demand, from FPGA vendors, an increase in the amount of logic elements in each generation of FPGAs. The wide range of applications employing arithmetic operations has also motivated FPGA vendors to include a variety of multiple-bit wide computing elements including embedded memory, Digital Signal Processors (DSP), processor cores and even hard intellectual property (IP) blocks.

Many of the applications implemented on FPGAs are arithmetic intensive and include large fractions of regularly structured components called datapaths. Typically, datapaths are composed of identical blocks of logic which process and transport several bits of data as a unit (called a *multi-bit signal*) at one time. The correlation between individual bit-based signals travelling from a common source to a common end introduces *regularity* among datapath circuits. This regularity can be preserved and exploited to increase FPGA area efficiency by employing multiple bit processing (multi-bit) architectural features. Specifically, multi-bit signals can be transported as a single unit on *multi-bit based routing tracks* to be processed by *multi-bit based logic blocks*. Unlike single-bit logic blocks which treat the signals at the input and output pins as independent bits of data, the signals at the input and output pins of multiple-bit logic blocks are processed as multiple-bit wide data. Subsequently, the signals at each pin are considered as unique bit positions of a wider multi-bit signal.

Since the release of Altera’s Stratix II series of FPGAs [4], most commercial FPGAs have replaced bidirectional routing resources with buffered unidirectional routing resources. Unidirectional routing resources have been shown to be more area efficient while reducing delay and wire capacitance over bidirectional routing resources [5]. The work done in [5] also outlines the two major optimizations of unidirectional routing over bidirectional routing: 1) Using pairs of directional wires 2) Replacing tri-state drivers and track to track connections with non-tristate drivers and multiplexers respectively. While multi-bit connections using bidirectional routing have resulted in FPGA area savings, the behaviour of multi-bit connections employing unidirectional components is unknown.

The multi-bit architecture described in this work is an adaptation of the MB-FPGA architecture presented in [6] and contains groups of multi-bit unidirectional routing buses as well as singular unidirectional tracks (independent tracks to route non-mult-bit signals). We experimentally evaluate the area efficiency and performance of this architecture based on the methodology proposed in [7].

In order to experimentally investigate the effects of multi-bit signals on unidirectional routing, a set of datapath-rich benchmark circuits are implemented on multi-bit and non-multi-bit architectures. To this end, the Virtual Place and Route (VPR) CAD tool [8] employing placement and routing algorithms is used while modifications are made to support both multi-bit and unidirectional architectural features. In order to preserve the regularity (amount of related signals travelling from a common source to a common destination) of the benchmark circuits, datapath-oriented synthesis [9] and packing [10] tools are used to generate the netlists used during the placement and routing operations in this thesis.

1.1 Thesis Motivation

The motivation of this thesis centres on investigating how using multi-bit based connections on unidirectional routing resources affect FPGA area efficiency. In this research we determine the theoretical area savings of a purely routing bus-based FPGA tile over that of a conventional tile; both of which employ unidirectional routing. We then introduce the multi-bit architecture which employs both bus-based and singular unidirectional routing elements. In practical FPGAs, architectural aspects such as channel width and I/O connectivity are pre-fabricated and therefore the goal of this research is to determine a suitable architecture to efficiently implement datapath-oriented applications. Previous studies [11] [12] [13] [14] [15] have proposed various FPGA architectures containing specialized computing elements that are designed to process multiple-bit wide data. None of the studies, however, have investigated the design of multi-bit routing resources that can effectively transport multiple-bit wide signals using unidirectional routing tracks.

1.2 Thesis Contribution

This thesis investigates the area and delay performance of a multi-bit FPGA architecture using unidirectional routing tracks and is the first study to do so. Furthermore, this research reflects the transition undertaken by academic and commercial FPGA designers towards conventional unidirectional routing architectures employing directional wires and non-tristate drivers.

In this research an analysis on the effect of routing resources on total FPGA area is presented. These area results are then compared to those of an equivalent architecture utilizing only routing buses and related connections. Both analyses take into account a transistor sizing of a modern process technology. Modifications to the routing resource graph and output display of the VPR tool are made to reflect the architecture under investigation. To conduct an empirical investigation, this work places and routes a set of pre-synthesized and pre-packed benchmark circuits of varying datapath regularities to measure the active area, track count and critical path delay. As it shall be shown, the multi-bit architecture is found to be more area efficient when implementing a wide range of datapath rich circuits over the conventional architecture without a significant effect on delay.

1.3 Thesis Organization

The thesis is organized as follows. Chapter 2 presents a background on FPGA architectures relevant and leading up to this work including two previous architectural studies regarding multi-bit based connections and unidirectional routing. Additionally, a review of the relevant academic CAD tools is presented. Chapter 3 analytically compares the theoretical advantages of an architecture using multi-bit connections over the conventional routing architecture. Chapter 4 describes the unidirectional multi-bit architecture including a description of logic and routing components modeled using a modern process technology (predictive technology models) to generate accurate area and delay results. Chapter 4 also presents the modeling of this architecture on a modified version of the VPR CAD tool (introduced in Chapter 2) to be used in evaluating the unidirectional multi-bit architecture described earlier in Chapter 4. Chapter 5 presents the results and analysis of the experiments. Finally, Chapter 6 concludes this thesis by providing a summary of the results and proposes future work.

Chapter 2 : Background

This chapter presents background information on the architectural aspects of FPGAs, the CAD flow and related academic CAD tools used to implement circuits. Section 2.1 briefly reviews the various programmable technologies followed a discussion on the structure of logic components and routing networks of the bidirectional FPGA architecture. Furthermore, two prior architectures relevant to this work are reviewed. Section 2.2 describes the typical CAD flow used to implement circuits on FPGAs with bidirectional single-bit routing. Lastly, the CAD flow to implement datapath circuits on bidirectional multi-bit routing resources will be reviewed.

2.1 FPGA Architecture

The majority of modern FPGAs, including devices from Xilinx [16] and Altera [1], rely on static random access memory (SRAM) based programming technology. SRAM cells are used to program FPGA logic and routing components such as pass transistors, multiplexors and look-up-tables (LUTs). Other programming technologies include one-time programmable antifuses,

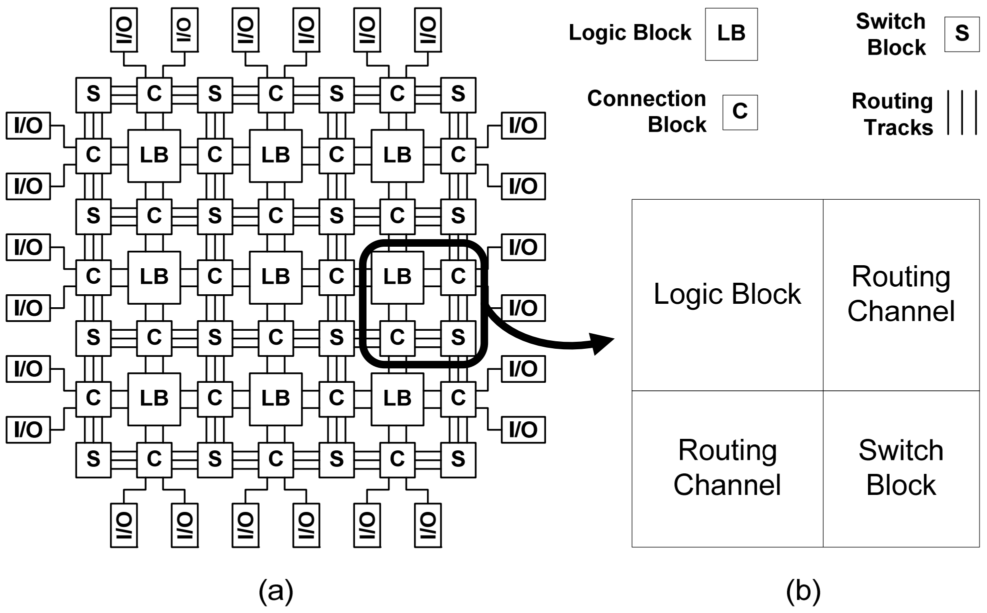


Fig. 2.1. Layout of an (a) Island-style FPGA and (b) a FPGA tile

erasable programmable read-only memories (EPROMs) and flash-erase EPROMs. This research focuses particularly on FPGAs employing SRAM cells.

The FPGA architectures considered in this research are based on the island style topology shown in Fig. 2.1 (a) which is segmented into *tiles* as shown in Fig. 2.1 (b). An FPGA tile consists of a *logic block*, one vertical and one horizontal *routing channel* containing a finite number of routing tracks and a *switch block* where the routing channels intersect. Connections between the routing tracks in a channel and the input/output pins of an adjacent logic block are made inside of a *connection block*. Finally the I/O blocks allow for external circuitry to interact with the FPGA.

2.1.1 Single-bit Bidirectional Architecture

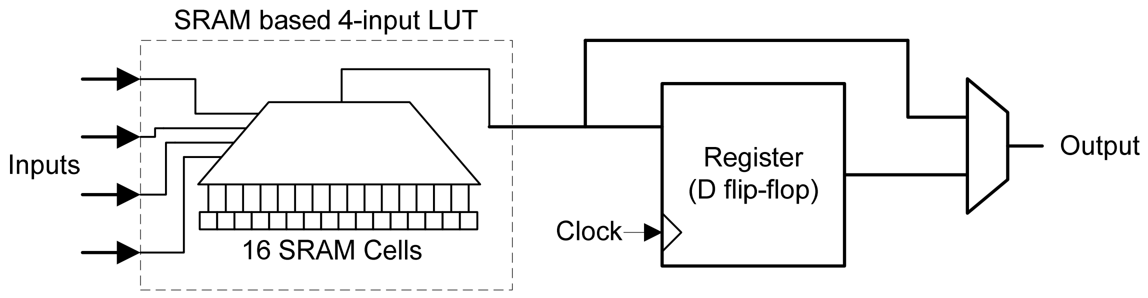


Fig. 2.2. Basic Logic Element (BLE)

2.1.1.1 Logic Block Structure

Logic blocks contain the basic digital logic required to implement sequential or combinational circuits. Most current commercial FPGAs and academic architectures employ logic blocks containing a LUT paired with a register element. As such, this research will focus on these LUT-based logic blocks. Fig. 2.2 illustrates how a 4 input LUT and a D flip-flop are used to form a *Basic Logic Element* (BLE) which is situated inside a logic block. In general, an k-input LUT is implemented using a 2^k:1 multiplexor and 2^k SRAM cells, where any function of

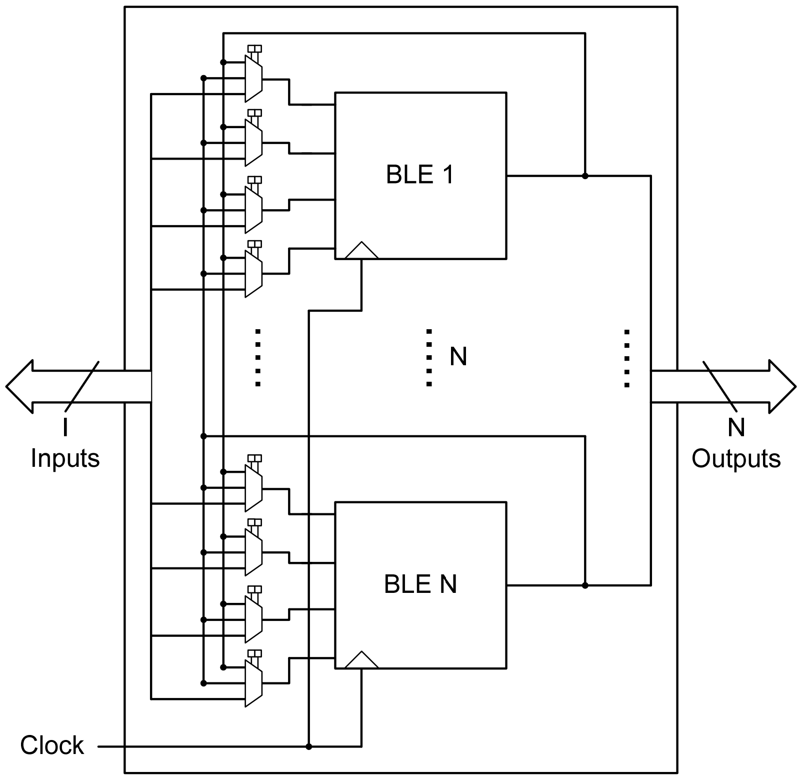


Fig. 2.3. Internal logic block structure

k-inputs (reflecting the truth-table values of the function) can be programmed into the 2^k SRAM cells. The inputs to the LUT connect directly to the selection inputs of the multiplexor and the output of the multiplexor is synonymous with the LUT output. The work in [17] has shown that using a 4-input LUT (4-LUT) in FPGAs achieves the greatest area-efficiency by providing the most functionality per pin. Additionally, attaching a D flip-flop to the output of the 4-LUT results in a registered BLE output and allows for the implementation of sequential logic. As shown in Fig. 2.2, the output of the BLE is wired in such a way that either a registered or unregistered version of the 4-LUT function implementation can be chosen. Therefore this research employs the 4-LUT based BLE shown in Fig. 2.2 for the remainder of this thesis.

A typical logic block contains several BLEs connected together by local routing resources and is alternatively known as either a *logic cluster* or *configurable logic block* due to its 'cluster-like' topology. While the terms logic block and logic cluster are synonymous for this architecture, Section 2.1.2.1 will show how several logic clusters can be combined to form a multi-bit logic block. As shown in Fig. 2.3, a logic cluster can be described as having *I* inputs, *N* BLEs of size 4 and *N* cluster outputs. Each BLE input has access to any of the logically equivalent cluster inputs through the use of a multiplexor. The same multiplexors also allow BLE outputs to connect to any BLE input. Logic clusters are know to be 'fully connected' when the flexibility of BLE inputs allows connections to any of the cluster inputs or BLE outputs.

2.1.1.2 Routing Architecture

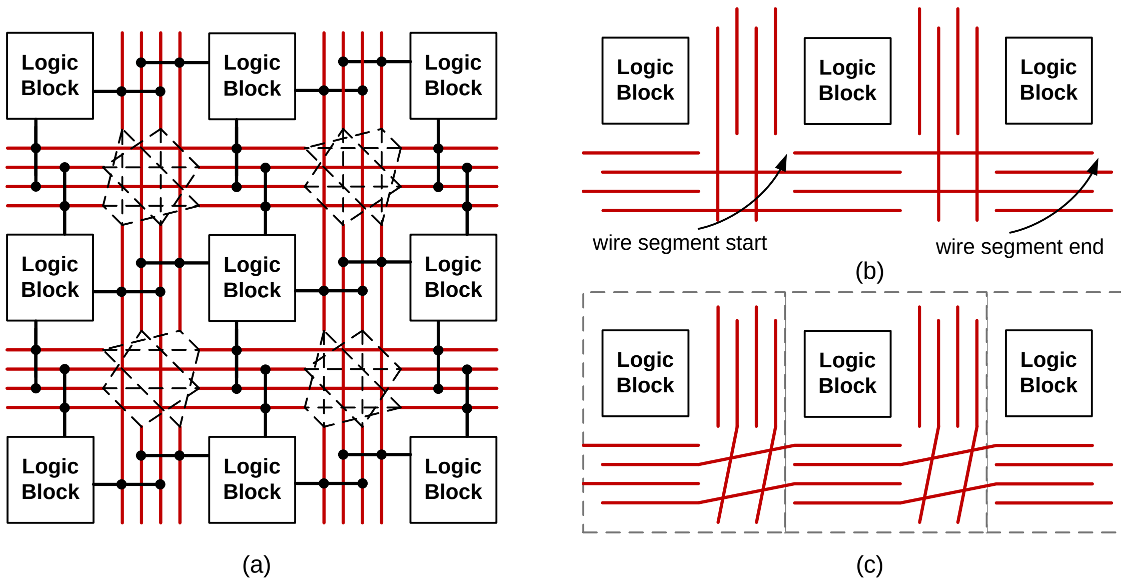


Fig. 2.4. FPGA routing architecture with wire segments of length 2

Routing tracks and programmable switches enable logic blocks to be interconnected and allow signals to traverse the FPGA. The routing tracks are contained in routing channels which run adjacent to the logic blocks both horizontally and vertically. Each channel contains a fixed number of routing tracks on which signals travel in a bidirectional fashion. The number of routing tracks in a channel is represented by *W* – its channel width. Every routing track is composed of a series of wire segments of length *L*, where the *L* is the number of logic blocks that each wire segment spans. Fig. 2.4 (a) shows a 3x3 size FPGA layout with each routing channel containing 4 routing tracks which consist of wire segments of length 2. It can be seen that the wire segments in every channel are staggered [18] (adjacent wire segments start and end at different locations in a channel), as some wires will continue through a switch block while others end. Fig. 2.4 (b) shows this staggering clearly as the logic block in the middle is able to connect to the adjacent logic blocks on either side using a single wire segment. Hence, the staggering of wire segments significantly increases the routing flexibility of the FPGA. Furthermore, Fig. 2.4 (c) shows wire segments which are staggered and shift upwards (called track shifts). Incorporating wire segments with track shifts and staggering start positions simplify the layout process of FPGAs, since a single tile is replicated to construct the complete FPGA. At the intersection of a vertical and horizontal channel, there exists a *switch block*, where individual wire segments are connected together allowing signals to continue on the same track or turn. The dashed lines in Fig. 2.4 (a) indicate connections from the end of one wire segment to the beginning of another inside a switch block. The I/O pins of logic blocks are connected to the routing network via programmable switches inside *connection blocks*.

Groups of connections within a switch block can be classified as either full-switch

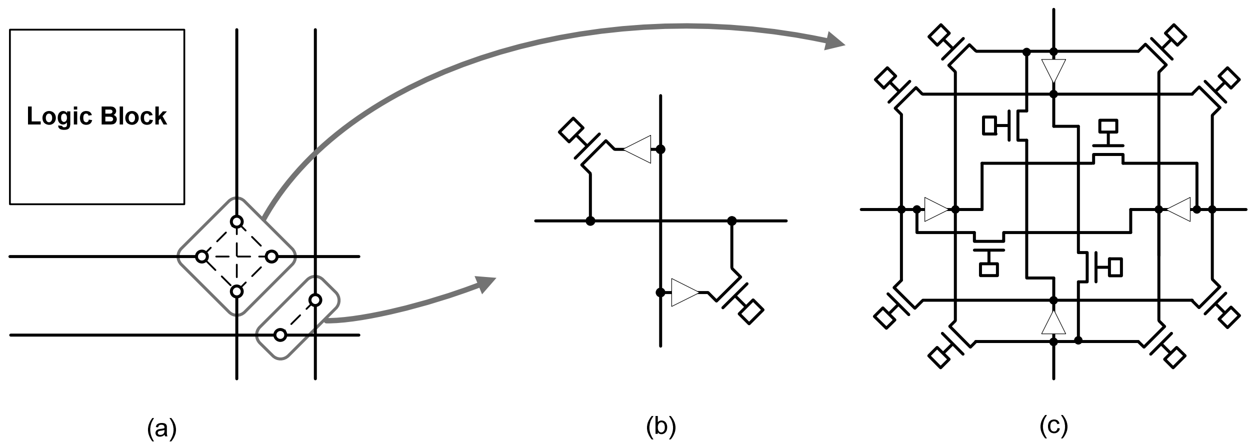
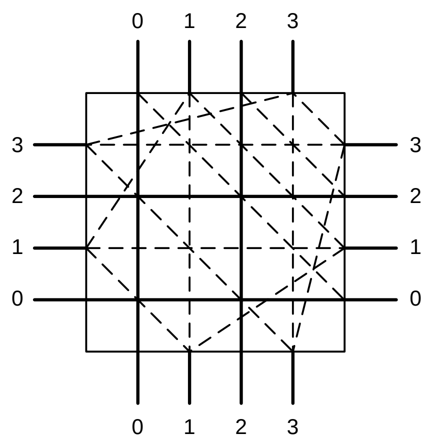


Fig. 2.5. A (a) switch block with (b) half and (c) full switch block connections



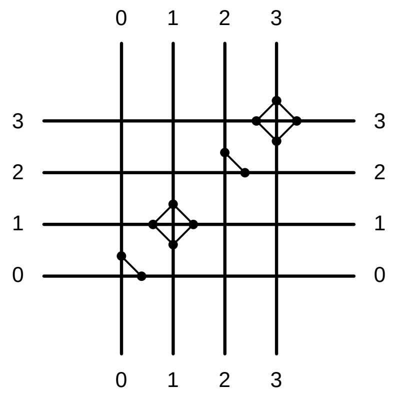


Fig. 2.6. Disjoint switch block topology (L=2) using half and full switch connections

connections or half-switch connections. Fig. 2.5 (a) illustrates these two types of switches in a simplified FPGA tile. Internal connections between wires that continue through a switch block are made using the half-switch shown in Fig. 2.5 (b). The full-switch pictured in Fig. 2.5 (c) shows the end point of a wire segment in any direction drives up to three other wire segments, all of which also end at this location. Both types of switch connections employ tri-state buffers.

The number and arrangement of both full-switch and half-switch connections inside a switch block are the function of its topology. Furthermore, the topology describes how many and specifically which tracks an ending wire segment can connect to. The most common switch block topologies include the *disjoint* topology [19] and the W*ilton* topology [20]. The results in [7] show that the disjoint topology is more area-efficient than the Wilton topology for wire segment lengths greater than 2 and have the same area efficiency as the W*ilton* for segment lengths equal to 2. Hence, the disjoint topology is used in this work, since the focus of this work is on architectures containing wire of length 2 and laying the foundation of longer wire lengths in future work. Fig. 2.6 (a) and (b) illustrate the connections of the disjoint topology in a switch block using half-switch and full-switch connections. A wire segment ending at this switch block

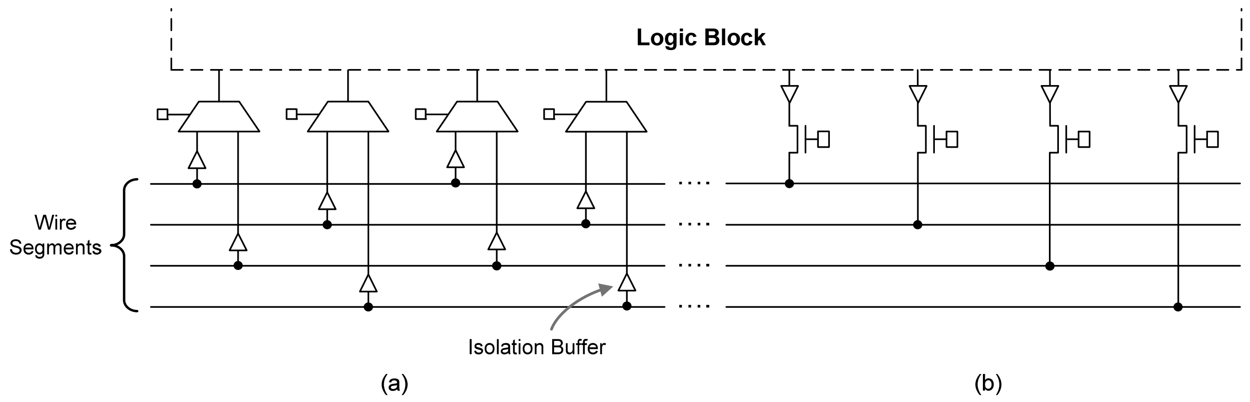


Fig. 2.7. Connection block with (a) input and (b) output connections

location connects to exactly 3 other wire segments beginning at this switch block with the same track position. To demonstrate this, take for example the horizontal wire segment 3 entering the switch block from the left in Fig. 2.6 (a). This segment connects to: the horizontal wire segment 3 on the opposite side, vertical wire segment 3 entering from the top and vertical wire segment 3 entering from the bottom. As mentioned before, full-switch connections are employed at wire segment endpoints, while half-switch connections occur at internal locations along a wire segment.

The final aspect in the architectural description of this FPGA is the connection block. The connection block house the individual connections from the routing tracks to the logic block output pins and the logic block input pins to the routing tracks. As shown in Fig. 2.7 (a), individual routing tracks connect to the inputs of a multiplexor whose output connects directly to a logic block input pin. The inputs and outputs are shown separately for clarity and simplicity. Notice in this example that each input connection multiplexor can select from 2 routing tracks in the channel. The selection of tracks is staggered to offer greater overall routing flexibility and reduce the amount of loading capacitance. In general, the proportion of tracks any input pin can connect to is denoted by F_{ci}. Additional buffers called *isolation buffers* are added to the input connections to isolate the routing tracks from the capacitive effects of the input connections. Fig. 2.7 (a) shows how logic block output pins connect directly to the routing tracks using a driving buffer to propagate the output signal and a SRAM based pass transistor switch to control the connection. Here, F_{co} represents the proportion of tracks the output pin can connect to.

2.1.2 Bidirectional Multi-bit Architecture

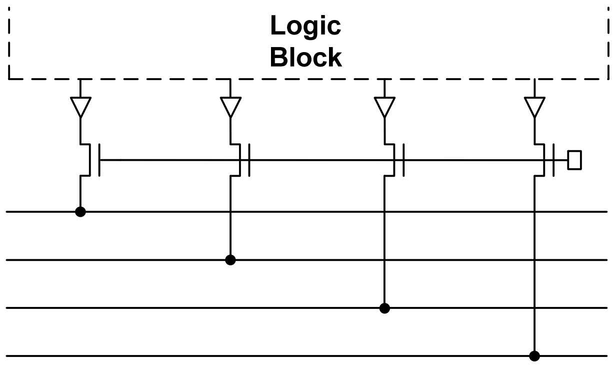


Fig. 2.8. Output connections using configuration memory sharing (CMS)

Prior research regarding an FPGA architecture optimized for datapath applications (DP-FPGA architecture) is described in [14]. The DP-FPGA study focused on designing heterogeneous blocks including a memory block, control block and a datapath block. The memory block could be configured to implement various types of memory architectures. The control and datapath blocks implemented non-datapath and datapath circuits respectively. The DP-FPGA architecture is the first study to make use of configuration memory sharing (CMS). As its name implies, configuration memory sharing involves sharing a single set of SRAM memory bits over multiple switch connections. Take for example the output connections in Fig. 2.7 (b) of the previous section. Fig. 2.8 shows how employing CMS on these output connections would reduce amount of SRAM bits required from 4 to 1 resulting in significant area savings. However,

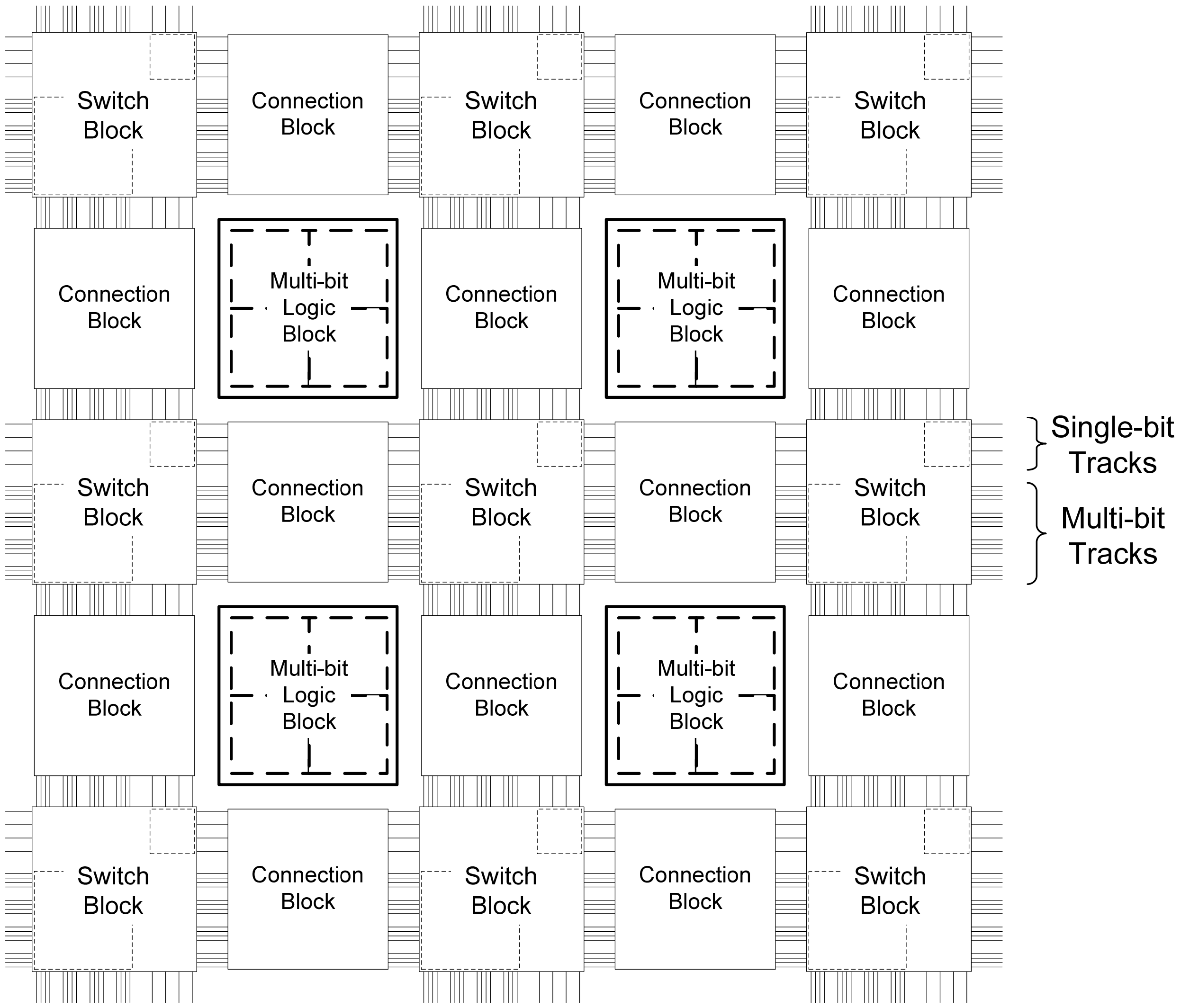


Fig. 2.9. Multi-bit FPGA

the DP-FPGA study did not specify the architecture in detail or investigate the effectiveness of CMS utilization on the routing architecture.

The *multi-bit* architecture, proposed and described in [21], is a continuation of the DP-FPGA work and is also based on the island style topology. Similar to the DP-FPGA architecture, the purpose of this architecture was to exploit the significant proportion of datapath signals in modern circuits to achieve overall FPGA area-efficiency. As Fig. 2.9 shows, the multi-bit architecture employs the same components described in the previous section (including logic

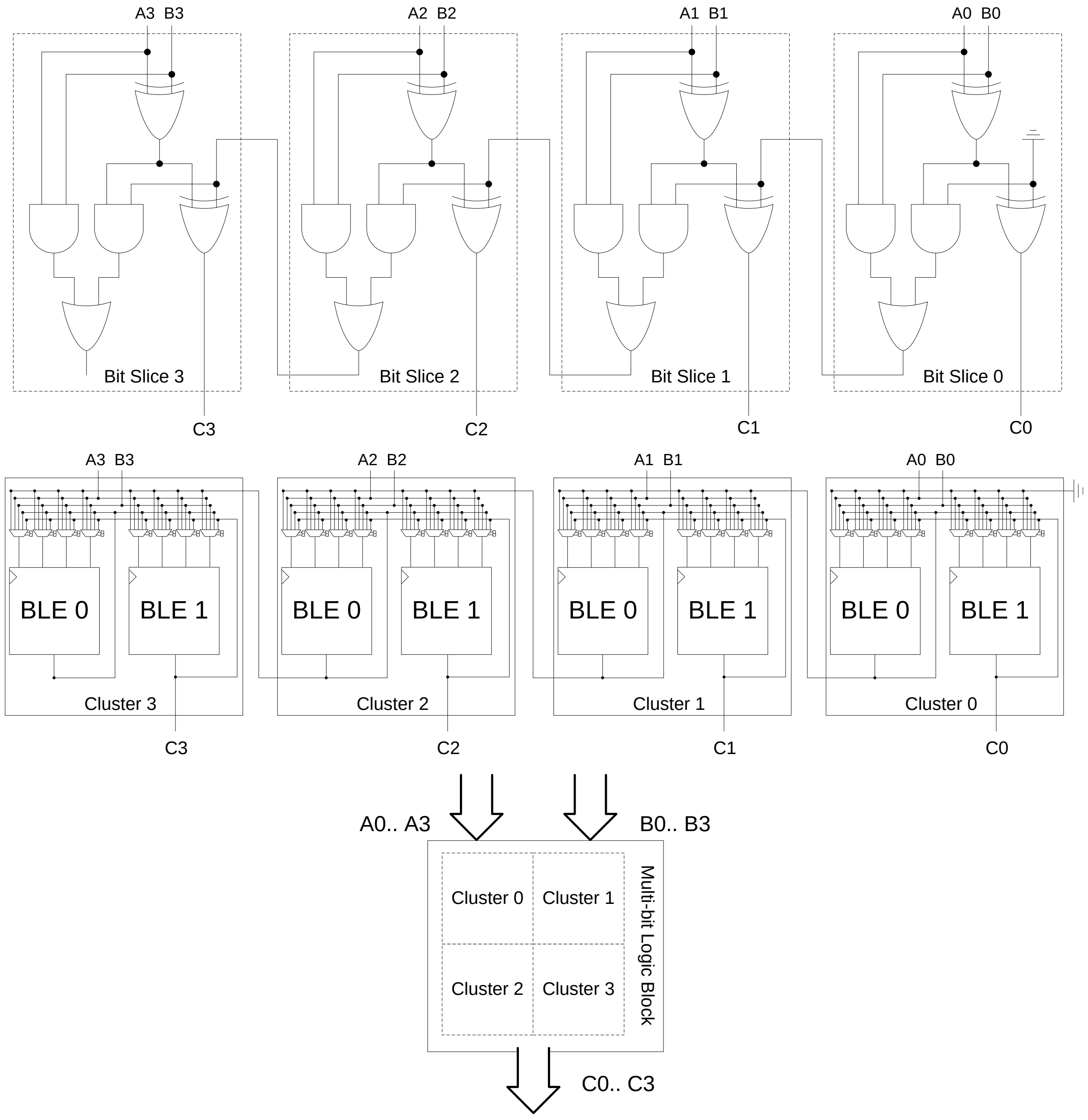


Fig. 2.10. Mapping of a 4-bit adder into a Multi-bit logic block

functions are somewhat different. The multi-bit architecture in Fig. 2.9 employs a special type of logic block called the multi-bit logic block which are connected to other multi-bit logic blocks using single bit routing tracks and groups of *routing buses* called multi-bit routing tracks.

2.1.2.1 Multi-bit Logic Block

Recall in Section 2.1.1.1, how several BLEs can be grouped together using internal routing to form a logic cluster (or logic block). Similarly a multi-bit logic block involves grouping *M* logic clusters together to process *M*-bit computation, where *M* is known as the *granularity* of the architecture. Take for example, a 4-bit addition operation with inputs A0-A3, B0-B3 and the sum C0-C3. Fig. 2.10 shows how this addition operation can be partitioned into bit-slices, where each bit-slice encompasses a 2-bit adder with a carry signal feeding the next bit-slice. Assuming the logic in each bit-slice can fit 2 interconnected BLEs, these two BLEs form a logic cluster. The resulting 4 logic clusters form a multi-bit logic block with 2 4-bit wide *input connection buses* and 1 4-bit wide *output connection bus*. In general, the number of inputs and outputs are still represented by *I* and *N* respectively. Recall from Section 2.1.1.1, the number of logic block outputs reflects the exact number of BLEs the logic block contains. In the multi-bit architecture, *N*, actually represents the number of output buses, while the total number of output pins is *N* x *M*. Similarly, *I* represent the number of input buses and the total number of input pins is *I* x *M*.

2.1.2.2 Multi-bit Routing

The routing network of the multi-bit architecture is similar in design to the bidirectional single-bit routing architecture described in Section 2.1.1.2. The difference lies in the addition of *M*-bit wide routing buses to the existing channels containing the *single-bit routing tracks* as described in Section 2.1.1.2. The individual tracks in a routing bus are known as *multi-bit routing tracks*. The multi-bit routing tracks transport the datapath oriented signals from one multi-bit logic block to another as a single unit. The single-bit routing tracks transport controls signals or any other single-bit signal throughout the FPGA. The amount of multi-bit routing

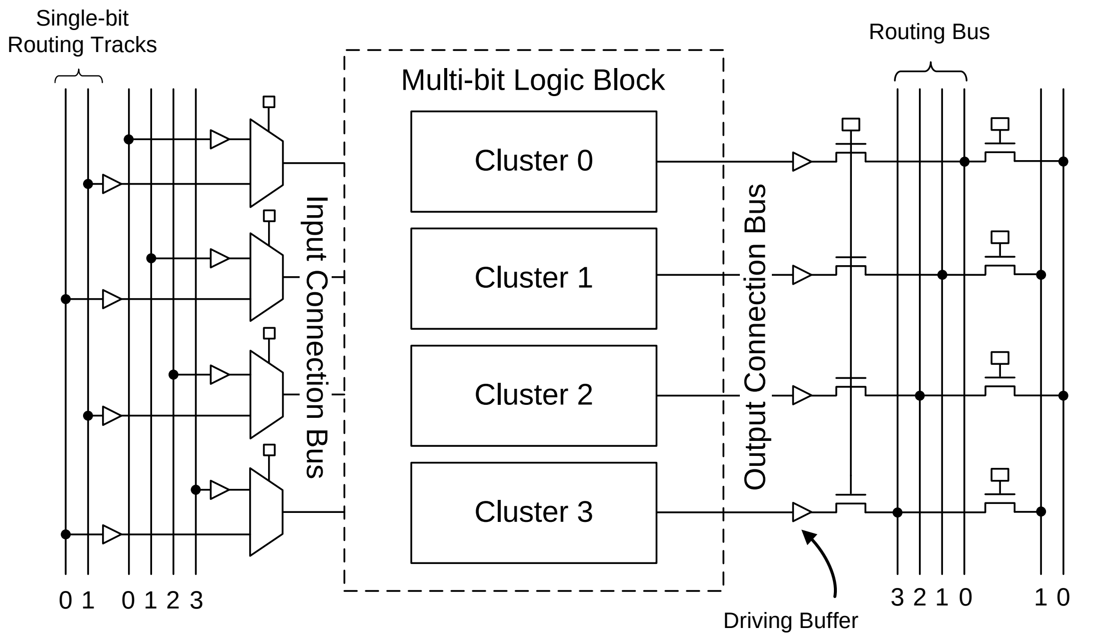


Fig. 2.11. Input and output connections of the connection block

tracks in a channel (the channel width) is represented by W_c, while the amount of single-bit routing tracks in a channel is defined as W_f.

A simplified version of the input and output connections of a multi-bit logic block is shown in Fig. 2.11 and shown separately for clarity. Each input pin of the multi-bit logic block is connected to a fixed number of single-bit tracks and multi-bit routing tracks. The proportion of single-bit tracks an input pin connects to is represented by F_{cif} while the number of routing buses that an input pin connects to is represented by F_{cic} [21]. Similarly the output pins are connected to a fixed proportion of single-bit tracks and routing buses represented by F_{cof} and F_{coc} respectively. While the single-bit tracks can follow any connection pattern, routing bus connectivity requires the input and output pins to be restricted to the specific bus index (or bit position) they represent. Take for example the output of cluster 0 in Fig. 2.11. Cluster output 0 is the 0th bit in the output connection bus; therefore, it may only connect to 0th track of the routing bus. Similarly, multi-bit logic block inputs must connect to routing buses with the same bus index.

The multi-bit switch block accommodates both single-bit switch connections and multi-bit switch connections. Within the switch block, each type of routing track is segregated from the other, meaning multi-bit tracks connect only to other multi-bit tracks and single-bit tracks connect only to other single-bit tracks. While this allows the flexibility to implement different switch block topologies for each routing track type, the disjoint switch topology [19] defines the switch arrangement for both multi-bit and single-bit switch connections. The single-bit tracks are connected together in the same manner as outlined in Section 2.1.1.2 using half and full switch block connections. Conversely the switches connecting each M-bit wide routing bus to another share a single set of configuration memory (CMS). Similar to the input and output connections, the multi-bit routing tracks in a bus must connect to other multi-bit tracks with the same index within a switch block.

2.1.3 Single-bit Unidirectional Routing Architecture

In both the bidirectional single-bit and multi-bit architectures, the bidirectional wire segments are connected together and driven using tri-state buffer switches inside a switch block (one for each direction). The study done in [5] demonstrates that 50% of these switches remain unused once the FPGA has been fully configured. Conversely, unidirectional wires only employ a single driver per direction per wire. This results in a much higher utilization of the programmable switches, especially if CAD tools can approximately match the amount of signals to the number of tracks in each direction. It is important to note that the number of tracks W inside a channel must be a multiple of 2. Given an FPGA tile with 1 track (2 wire segments connected using bidirectional switches); it can be observed that an equivalent tile with the same amount of

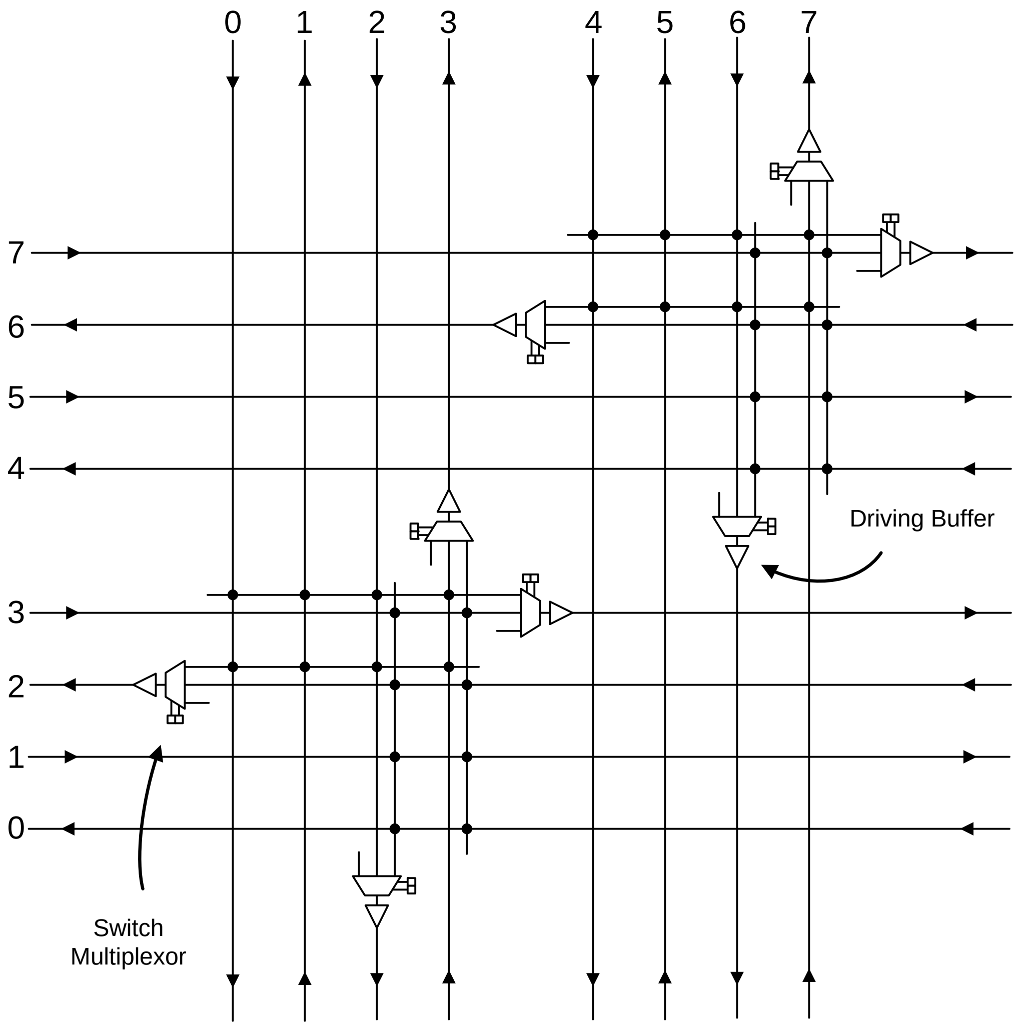


Fig. 2.12. Conventional switch block with unidirectional wires of length 2

unidirectional switches requires twice as many wires. The work in [5] shows that in order for unidirectional routing to be area efficient, the total number of routing tracks must be less than twice the amount of equivalent bidirectional tracks.

In this architecture, the programmable switches within the switch blocks (routing switches) are implemented using non-tristate drivers and multiplexors. Using non-tristate driver offers an advantage of greater area savings due to the removal of the SRAM based tri-state functionality. The multiplexor offers greater track connection flexibility within a switch block. The use of a multiplexor eliminates the need for full and half type connections as a multiplexor can switch between wire segment midpoints and ends. As in both the general and multi-bit architecture, the conventional architecture employs the disjoint topology inside switch blocks. Fig. 2.12 illustrates switch block containing unidirectional wire segments of length 2 and associated programmable switches. The staggered routing is exemplified in this example as tracks 2, 3, 6 and 7 in both channels end at this switch block, while tracks 1, 2, 4 and 5 continue through. The arrows at the fringes of the switch block mark the directions of each track. The multiplexors in Fig. 2.12 show connections from the midpoints and endpoints of nearby wires with a single input line and black circles to indicate individual connections. The unconnected input line represents connections to logic block outputs which will be discussed next.

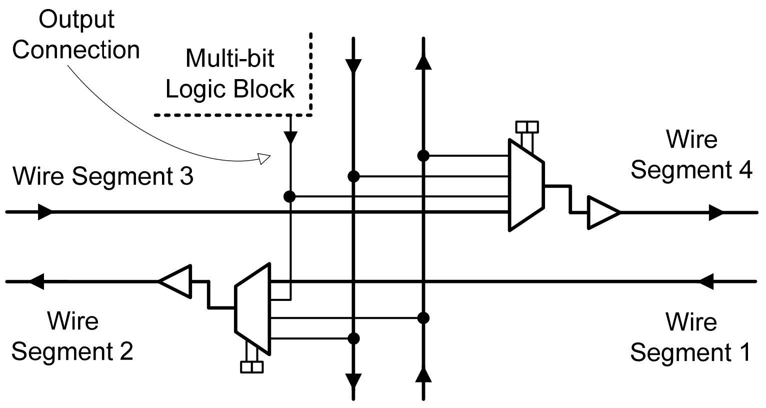


Fig. 2.13. Switch block connections for a horizontal channel

The connection blocks in the conventional architecture lack direct connections to tracks from logic block outputs, containing only input connections. Due to design restrictions, the outputs of multi-bit logic blocks can only connect to the multiplexors of wire segments that begin nearby. Fig. 2.13 shows a multiplexor base switch and its connectivity to a multi-bit logic output in a horizontal channel. The SRAM cells can be programmed to select between the multi-bit logic block output and any of the three track to track connections. The input connections are structured exactly as in the single-bit bidirectional architecture.

2.2 FPGA CAD Algorithms and Tools

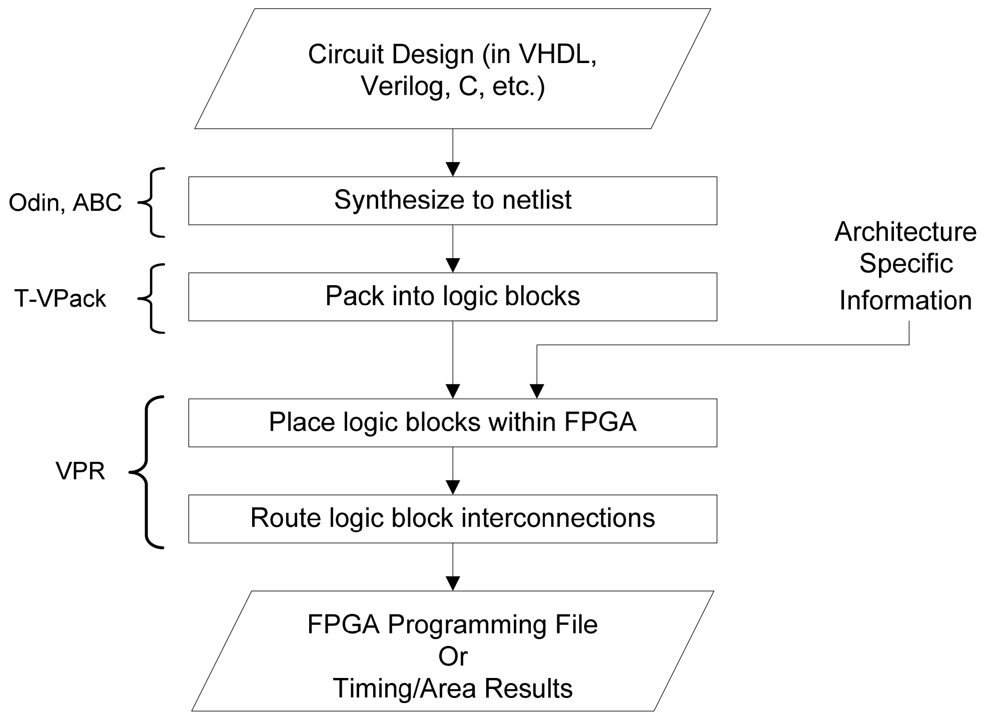

Circuit Design (in VHDL, Verilog, C, etc.)
Odin, ABC
Synthesize to netlist
Architecture Specific Information
T-VPack
Pack into logic blocks
VPR
Place logic blocks within FPGA
Route logic block interconnections
FPGA Programming File Or Timing/Area Results

Fig. 2.14. FPGA CAD flow

In order to practically implement circuit designs on an FPGA, the use of CAD tools are necessary. CAD tools allow a designer to input a high-level description of a circuit which is then converted into a programming file or alternatively, in the case of VPR, a textual and graphical description of the FPGA implementation. The programming file specifies the state of every programmable switch and configuration bit on the FPGA. These CAD tools, both commercial and academic, follow a set of steps called a CAD flow as illustrated in Fig. 2.14. This section will first briefly describe the *synthesis*, *packing*, *placement* and *routing* elements of the CAD flow. A more in-depth discussion of the *routing* step, where the architectural changes of this research are made, will then be presented.

2.2.1 Synthesis & Packing

Given a circuit, described in either a hardware description language or schematic form, the synthesis stage converts such a description into a netlist of simple gates (NAND, NOR, Inverter, etc.). The netlist of gates then undergoes a process of optimization, independent of any technology process, where the logic is reduced to a minimum or near-minimum amount of gates. This process of logic minimization must be completed within a reasonable amount of computing time. The new optimized netlist of gates is then technology-mapped into look-up tables and flip-flops [22] [23] [24]. If in the case of the VPR CAD flow, the circuit is described in Verilog hardware description language, the open-source tool Odin [25] can be used to initially synthesize the circuit into a basic netlist of gates. This netlist can then be optimized using the ABC [26] synthesis and verification tool.

The second step in the CAD flow, called packing, groups LUTs and flip-flops (up to a limit of *N*) into logic clusters. The packer must take into consideration pre-determined architectural parameters such as the maximum number of inputs, the maximum number of outputs and whether clock signals are needed (and how many distinct clock signals) to implement sequential logic. The packing operation attempts to minimize the amount of routing connections between logic blocks. In addition, the packing process attempts to pack as many highly connected LUTs into logic blocks (up to their limit *N*) therefore reducing the total number of total logic blocks required. Timing driven packing involves attempting to reduce the number of connections between logic clusters on the critical path in addition to the packing process described above. In the case of the VPR CAD flow, T-VPack is the stand alone packing tool described in [27] which can perform timing or non-timing driven packing.

2.2.2 Placement & Routing

The placement and routing operations in Fig. 2.14 are performed by a single program – VPR and will be described in further detail in this section. Also illustrated in Fig. 2.14 are two necessary inputs to this process: 1) The packed and optimized logic block netlist from the previous step and 2) a file containing architectural specific information about the target FPGA – the architectural description file.

VPR Placement

The placement process of VPR is conducted using the simulated annealing algorithm [28]. This algorithm is derived from the annealing process in metallurgy wherein metals are heated and then slowly cooled to alter the materials' physical properties (improve strength, soften for cutting and shaping etc.). Initially the placer will randomly position all the logic blocks in the netlist to available locations on an FPGA. It will then move a randomly selected logic block either by swapping it with another logic block's physical location or moving it to an unoccupied spot to create a new placement. The move is then evaluated based on a *cost function* which measures the quality the new placement against the previous placement. The process is repeated multiple times in order to reach a final placement solution. After each iteration, a decision is made to either keep or reject each move based on the change in the cost before and after the move. If the cost is less after the move, it will always be accepted. Should the cost increase, the move may still be accepted even though the placement is worse in the hope that subsequent moves may yield a better placement result. This probability is modeled mathematically and a key parameter called the *temperature* determines how likely a move is accepted despite the cost increase the placement causes. Initially, the temperature is set very high allowing almost all moves to be accepted. The temperature is then gradually reduced at each subsequent move allowing fewer bad moves to be accepted till at a temperature of zero, where only good moves are accepted. The rate at which the temperature is reduced and the number of moves attempt at each set temperature value is defined in the *annealing schedule*.VPR uses an adaptive annealing schedule, which adapts to a wide range of FPGA architectures, cost functions and circuit sizes [7] while consuming a reasonable amount of computation time.

VPR Routing

The VPR router has the flexibility of implementing a variety of architectures through the use of directed graphs. A directed graph, or *routing-resource graph* (as it is know in VPR) [29], is generic and simple enough for any architecture to be represented. Given the architectural description file, an internal graph generator creates a detailed routing resource graph representation of the desired FPGA [7]. The architectural description file (described fully in [7]) contains parametric information of the architecture such as the number of logic block input pins (*I*), the switch block topology and F_c values for logic block outputs. The actual routing operation, graphical output and performance analysis all rely on the routing-resource graph. Should the need arise to use a new FPGA architecture, only the routing-resource graph requires

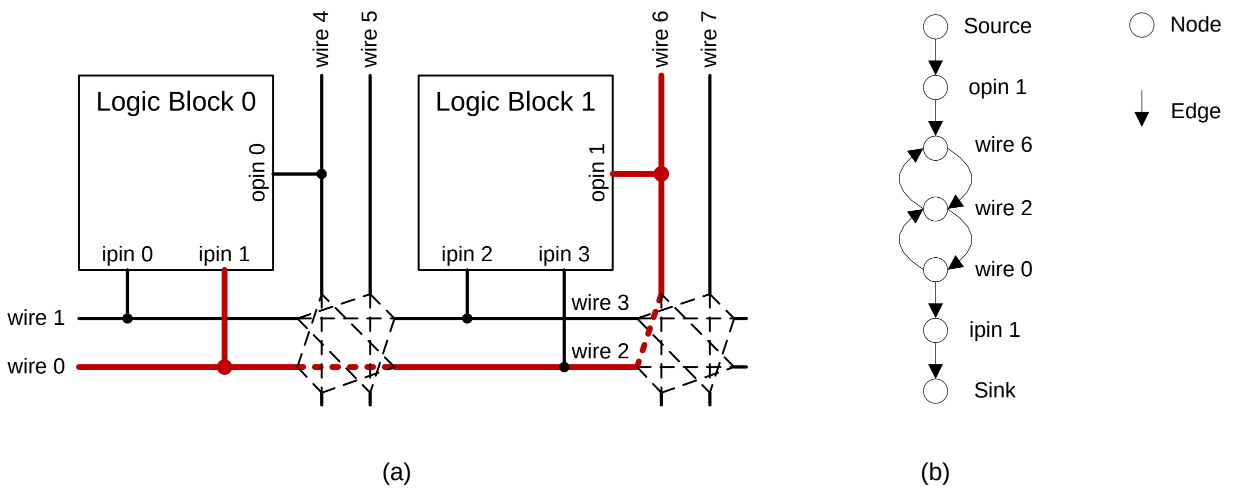


Fig. 2.15. Modelling (a) a connection between two output pins on (b) a directed graph

modification. This is particularly convenient since this research involves investigating the effects of unidirectional and multi-bit architectural features.

Inside the routing-resource graph, wires and logic block pins are represented by *nodes* while connections between two nodes (or switches) are represented by *edges*. Fig. 2.15 (a) illustrates a path originating from ‘opin 1’ of logic block 1 and terminating at ‘ipin 1’ of logic block 0, where the resources required to connect these two points are highlighted in red. Fig. 2.15 (b) shows the directed graph equivalent of the path in Fig. 2.15 (a). Recall in Section 2.1.1.1, the input and output pins of logic blocks are logically equivalent. This logical equivalence is represented in the routing-resource graph by having connections originating and terminating at common logical points called *sources* and *sinks* respectively. Accordingly, the pins ‘opin 1’ and ‘ipin 1’ are labeled as the source and sink respectively in Fig. 2.15 (b). Since Fig. 2.15 (a) depicts a bidirectional routing architecture, the connections between wire 6, wire 2 and wire 0 are drawn as pairs of directed edges in Fig. 2.15 (b).

VPR uses the negotiated congestion based method of the Pathfinder algorithm [29] with optimizations to better optimize delay. When searching for the best path for each net, conflicts will inevitably occur when choosing routing resources which can only be assigned to a single net. When a particular routing resource is temporarily assigned to several nets, it is called a *congested* routing resource. The Pathfinder algorithm executes multiple routing iterations to resolve this contention for routing resources. In the first routing iteration, each connection is routed on a minimum delay path and ignores routing congestion. In subsequent routing iterations, each net is ripped up and re-routed using a maze routing algorithm [30] until all occurrences of congestion are resolved. After each iteration, the costs of congested routing resources are increased to give more timing-critical nets priority when multiple nets compete for the same routing resources. These costs take into account the net delays (Elmore delay of each source to sink connection) and congestion of routing resources from all previous iterations.

2.2.3 MB-FPGA CAD Flow

The MB-FPGA CAD flow is similar to the CAD flow described in Fig. 2.14 but also contains support for recognizing datapath components and preserving their regularity through the CAD process. The distinguished features of each step in the CAD flow are outlined as follows:

- The EMC (Enhanced Module Compaction) synthesis algorithm [9] attempts to optimize logic across bit-slices for area efficiency while retaining their regularity.
- The CNG (Coarse-Grain Node Graph) packing algorithm [10] packs identical LUTs and register elements (flip flops) from neighboring bitslices. The packer optimizes the packed clusters for area efficiency and delay of critical nets.
- The MB-FPGA Placer [31] is similar to the VPR placer described in Section 2.2.2 as it contains the same annealing schedule and cost functions. The MB-FPGA placer differs in that it moves blocks of logic (within a square grid) on two hierarchal levels: as bit-sized clusters and multi-bit sized logic blocks.
- The CGR (Coarse-Grain Resource) Routing Algorithm routes two different types of tracks (multi-bit and single-bit). The router compensates overused tracks by committing signals to under used tracks, even if it means routing single-bit signals on multi-bit tracks (or vice versa – multi-bit wide signals on single-bit tracks). In doing so, the router balances the overall usage of all routing tracks.

This chapter has presented background information on basic FPGA architectural components along with a review of previous work on mult-bit architectures and unidirectional routing architectures. It is shown how the proposed features of both these architectures can improve routing area efficiency. The information on the VPR and MB-FPGA CAD flows will serve as a basis for the design of a unidirectional multi-bit compatible CAD tool for empirical evaluation.

Chapter 3 : Unidirectional Bus Connections

This chapter defines the conventional unidirectional routing architecture and discusses unidirectional bus connections in detail. The theoretical area savings of an ideal architecture using unidirectional bus connections is compared to those of the conventional routing architecture. This serves as a motivation to empirically determine the actual performance of the unidirectional multi-bit architecture in Chapter 6.

3.1 The Conventional Routing Architecture

The conventional architecture employs the same multi-bit logic block of granularity *M* as discussed in Section 2.1.2.1. The choice of using a multi-bit logic block over the general logic block (consisting of a single cluster) is justifiable for two reasons: 1) Consider if both the unidirectional multi-bit architecture and the conventional architecture employ multi-bit logic blocks. Therefore the investigation can focus on the effect of the routing fabric on area-efficiency rather than a 2 dimensional problem involving variations in both routing and logic block design. 2) In addition to the studies in [11], [12], [13], [14], [15] and [21] which employ multi-bit logic blocks, commercial FPGAs from Xilinx [16] and Altera [1] already use similar block types, such as DSPs and multi-bit addressable memory blocks, to process multi-bit signals. Motivated by this trend, this work compares the efficiency of the conventional architecture against the unidirectional multi-bit routing architecture for connecting multi-bit logic blocks.

Unidirectional routing tracks and related switch block resources are employed in the conventional architecture with routing channels consisting of *W* unidirectional routing tracks. Recall from Section 2.1.3, *W* must always be an even number of tracks to accommodate signals travelling in both the forward and reverse directions. In this research, *M*=4, *L*=2, *I*=10 and *N*=4 are used because the previous work in [21] has shown this combination of values results in good area results. Since this architecture uses a multi-bit logic block, it is important to note that *N* and *I* actually represent the number of output buses and input buses of size *M* respectively. However since the conventional architecture does not include any routing bus tracks to specifically route datapath signals, all individual input and output bus connections are treated as fully independent one-bit wide input and output connections. The total number of input connections is $\left\lceil F_{ci} \times \frac{W}{2} \right\rceil \times 2$ where F_{ci} represents the fraction of routing tracks each input pin connects to. The total number of output connections is $\left\lceil F_{co} \times \frac{W}{2} \right\rceil \times 2$ where F_{co} represents the fraction of routing tracks each output pin connects to. Recall from Section 2.1.3, the output connections can only be made to the routing switch multiplexors of wire segments that begin nearby. Also recall from Section 2.1.1.2, in order to create a tile-based FPGA design, the starting points of wire segments must be staggered and shifted in fixed intervals. These two factors determine the number of available routing tracks that an output pin can connect to. The number of wire segments beginning in switch blocks on either side of a logic block is represented by $\frac{W}{L}$. The parameters F_{ci} and F_{co} are set to 0.5 and 0.25 respectively and the disjoint switch block topology is used.

3.2 Unidirectional Routing Bus Connections and Their Advantages

In this research, the active area (the area occupied by transistors) is measured to estimate the total FPGA area including computational logic and routing resources. Specifically, this area is measured in terms of minimum-width transistor area and calculated using the following equation:

$$\text{Area} = \sum_{\text{All Transistors}} \left(0.5 + \frac{\text{Drive Strength of Current Transistor}}{2 \times \text{Drive Strength of Minimum Width Transistor}} \right)$$

While the previous work in [21] has confirmed the large impact of bidirectional single-bit routing resources on total FPGA area (previously discovered by [3]), no study has found the impact of unidirectional single-bit routing resources on total FPGA area. Therefore, Table 3.1 summarizes the routing resource area (input and switch block connections) and the total area of an FPGA tile (including the area of a single-bit logic block) employing unidirectional single-bit routing resources for increasing values of *W*.

Table 3.1 Impact of Routing on Total FPGA Area

W	A_{input}	$A_{sw.block}$	$A_{routing}$	A_{FPGA}	$A_{routing} / A_{FPGA}$
4	396	262	658	8,507.72	7.73%
8	873	524	1397	9,246.96	15.11%
12	1350	845	2195	10,044.64	21.85%
16	1587	1049	2635	10,485.44	25.13%
20	2064	1369	3433	11,283.12	30.43%
24	2301	1573	3874	11,723.92	33.04%
28	2538	1894	4432	12,281.61	36.08%
32	2775	2098	4872	12,722.40	38.30%
36	**3252**	**2418**	**5670**	**13,520.09**	**41.94%**
40	**3489**	**2622**	**6111**	**13,960.88**	**43.77%**
48	**3963**	**3146**	**7109**	**14,959.36**	**47.52%**
52	**4200**	**3467**	**7667**	**15,517.05**	**49.41%**
64	**4911**	**4195**	**9106**	**16,956.32**	**53.70%**
88	6574	5768	12342	17,754.01	55.78%
100	7285	6613	13898	20,191.77	61.12%
120	8470	7866	16336	21,747.94	63.90%

The transistor sizing of routing resources for all area calculations in Table 3.1 are based on the 90nm process specifications in [32]. The sizes of transistors within a logic block are determined by following the methodology described in [7]. Columns 2 and 3 list the input and switch connection area (both of which are illustrated later in Fig. 3.2) while column 4 lists the total area of all routing resources. Column 5 lists the total FPGA area including that of the multi-bit logic block while the final column shows the percentage of total FPGA area that the routing resources occupy. It can be observed that the input connection area, the output connection area and the percentage of total area that routing resources occupy in Table 3.1 increase as a function of *W*. The routing resources consume between 8% and 20% of total FPGA area for small channel widths. For large channel widths (ie. Over *W* = 64 routing tracks), the programmable routing resources occupy a substantial amount (over 54%) of the total FPGA area. The bold rows in Table 3.1 represent a typical track count range (32 to 64) for circuits with the given architectural values where the routing resources consume from 38% to 54% of the total FPGA area. It is important to note that the contribution to area savings from unidirectional routing as described in [5] accounts for the lower proportional values of routing area results (Column 6) with respect to the results obtained from the bidirectional architecture in [21].

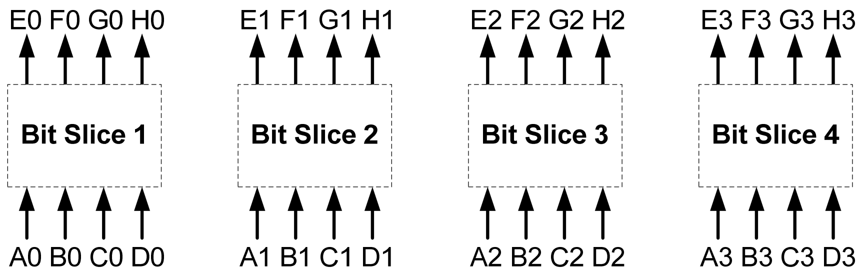


Fig. 3.1. Bit-slice partitioned datapath circuit

It is possible to alleviate some of this area by replacing conventional unidirectional tracks with multi-bit oriented unidirectional routing buses that employ multi-bit based connections to more efficiently transport multi-bit signals from a common source to a common destination. In order to clearly illustrate the advantages of multi-bit routing, we first consider mapping a generic datapath circuit onto a conventional FPGA tile. The circuit is segmented into 4 bit-slices in

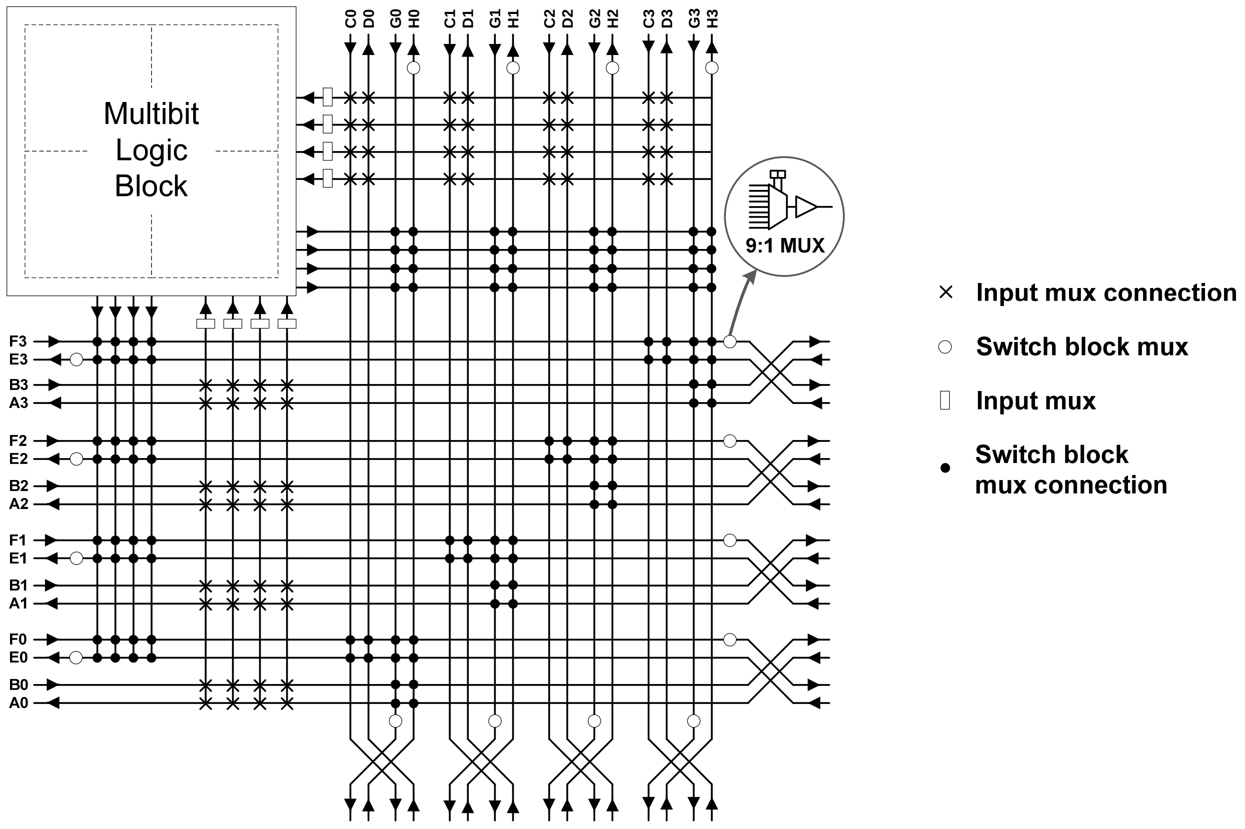


Fig. 3.2. Bit-slice circuit implementation on a conventional FPGA tile

which each bit-slice has 4 inputs and 4 outputs as shown in Fig. 3.1. Assuming the computational logic of each bit-slice can fit within a single logic cluster, for M=4, a multi-bit logic block is used to house the 4 logic clusters containing the entire datapath circuit. Fig. 3.2 shows, at a minimum, a 16-bit wide routing channel is required to transport all the signals to and from the multi-bit logic block. Each of the white circles in Fig. 3.2 represent a routing switch where each switch includes a *X*:1 multiplexor and its associated driving buffer. *X* represents the amount of multiplexor input connections which consist of all the black circles on the associated track (line segment) and the track itself. These switches are arranged in accordance with the disjoint topology. The white squares represent input multiplexors of size *Y*:1 where *Y* equals the number of connections between tracks in a channel and a particular input pin (marked with an ‘x’). This example assumes 50% connectivity of the inputs and 100% connectivity of the outputs wherever possible (F_{ci} = 50% and F_{co} = 100%). According to Fig. 3.2, there are 16 switch block connections and 16 input connections. Each switch block connection includes a 9:1 multiplexor while each input connection includes an 8:1 multiplexor. The multiplexors situated at the switch block each incorporate 4 output connections, 4 connections from adjacent tracks in the perpendicular channel and the track continuing straight through the multiplexor (hence 9 multiplexor inputs in total). While the bit-slices of Fig. 3.1 show a total of 16 input and output connections, Fig. 3.2 shows only 8 connections in total. The remaining 8 connections would be distributed among the other two sides of the multi-bit logic block (not pictured) and were also intentionally left out to preserve the clarity of the illustration.

Fig. 3.2 illustrates the two essential details of a practical design using one common tile layout as discussed in Section 2.1.1.2. The first being a staggered starting position of wires leaves tracks labeled A0-A3, B0-B3, C0-C3 and D0-D3 (in this example) without any switch block connections. These tracks do not begin on this tile and therefore do not have any switches associated with them to make connections. This leads to the next detail of requiring track shifts between pairs of 2*L* wires as shown at the bottom and right edges of Fig. 3.2. Specifically in this example, track shifts occur in groups of 4 wires. Additionally, in order to achieve a tile-based design with unidirectional wires, the channel width must be a multiple of 2*L*.

Alternatively Fig. 3.3 illustrates an architecture which replaces the routing tracks in each channel of Fig. 3.2 with 4 4-bit wide routing buses and groups the existing input and output connections into 4-bit wide input buses and output buses. The same multi-bit logic block of the previous example is used. Multi-bit based connection patterns are then used to connect the input/output connection buses and routing buses together. In particular, a bit in one bus can only be connected to a bit of the same bit position from another bus. Like the conventional tile of the

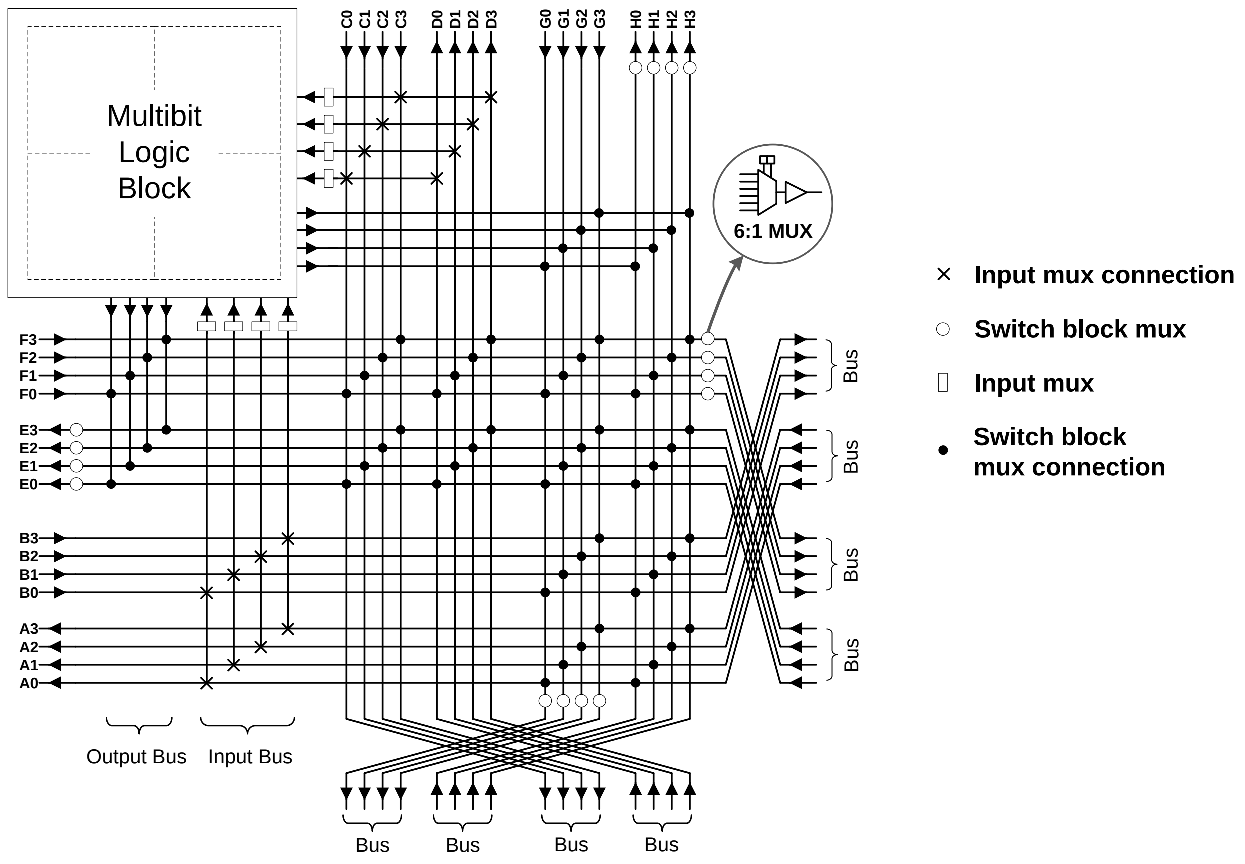


Fig. 3.3. Bit-slice circuit implementation on a FPGA tile with routing buses

previous example, this example assumes F_{ci} = 50% and F_{co} = 100%. Additionally, Fig. 3.3 illustrates a tile employing track shifts and staggered start positions of wires as seen in the previous example. This time, however, the staggering of wires occur in groups of *M* (or as buses) while the track shifts are made between groups of 2*L* buses (between a group of 4 buses in this example) as shown in Fig. 3.3. Notice how both the tiles in Fig. 3.2 and Fig. 3.3 require the same number of switches (white circles and white squares) and routing tracks to implement the circuit. Fig. 3.3 however requires smaller input and switch block multiplexors, specifically of size 2:1 and 6:1 respectively. This constitutes a 75% reduction in input multiplexor size and a 33% reduction in the switch block multiplexor size. This reduction occurs due to a much sparser switch block and input connection pattern where bit positions in one bus only connect to the same bit positions in another bus.

Given the area results in Table 3.1 for a conventional tile, the same methodology is used to generate results for a purely routing bus based tile. Table 3.2 lists the active area of a conventional tile and that of a unidirectional routing bus-based tile for increasing values of W and W_{bus}, where is the number of equivalent M-bit wide routing buses. The area calculations use M=4, N=4, I=10, L=2, F_{ci} = .5, F_{co} = .25 and a disjoint switch topology. For the purpose of this

Table 3.2 Conventional and Multi-bit Area of an FPGA Tile

Conventional		**Bus-based**		**A_{BUS} / A_{BIT}**
W	**A_{BIT}**	**W_{BUS}**	**A_{BUS}**	
8	9,246.96	2	8,685.98	94%
16	10,485.44	4	9,155.50	87%
24	11,723.92	6	10,096.50	86%
32	12,722.40	8	10,425.64	82%
40	13,960.88	10	10,732.42	77%
48	14,959.36	12	11,025.08	74%
56	15,957.84	14	11,307.98	71%
64	16,956.32	16	11,583.74	68%
72	18,194.81	18	13,113.11	72%
80	19,193.29	20	13,379.11	70%
88	20,191.77	22	13,641.64	68%
96	21,190.25	24	13,901.29	66%
104	22,188.73	26	14,158.53	64%
112	23,187.21	28	14,413.72	62%
120	24,185.69	30	14,667.13	61%

analysis, these examples are assumed to only implement circuits containing M-bit wide interconnected datapath components. As shown in column 5, the use of buses to route datapath signals can reduce area by 23% (for *W*=40 channel width) and 30% for (*W*=80). Larger area savings can be obtained for larger channel widths.

In summary, this chapter has presented a theoretical analysis on the effect of employing unidirectional routing buses to implement an ideal (only *M*-bit wide multi-bit signals) datapath circuit. In doing so, the conventional architecture has been defined for comparison to an architecture employing purely unidirectional routing buses. When employing only unidirectional routing buses a reduction of 23% to 30% in total FPGA area can be seen. It is shown that this area savings is attributed to a reduction in multiplexor sizes in the unidirectional routing bus-based architecture (75% reduction in input multiplexor size and a 33% reduction in routing switch multiplexor). Furthermore, the conventional architecture described earlier is modeled later in this thesis for an empirical comparison against the unidirectional multi-bit architecture (defined in Chapter 4).

Chapter 4 : Unidirectional Multi-bit Architecture & Modeling

This chapter describes the unidirectional multi-bit architecture given the motivations and background information in the previous chapters. First, Section 4.1 describes the architecture in detail, including the types of resources and connectivity. Section 4.2 presents a description of the buffer and transistor sizing information based on a modern technology process, in order to generate accurate results. Section 4.3 outlines the parameters and their values used to describe the architecture parametrically. Finally, Section 4.4 describes how this architecture is modeled on a modified version of the VPR CAD tool.

4.1 Unidirectional Multi-bit Architecture

As shown thus far, implementing ideal datapath circuits on a purely bus based routing architecture can significantly improve the area efficiency of FPGAs. Practical datapath circuits also contain highly regular components which can be adequately partitioned into bit-slices as demonstrated in the examples, in the previous chapter. These bit-slices can then in turn be placed and routed using only multi-bit logic and routing resources. However, practical circuits also contain irregular signals (single-bit wide signals, control signals or multi-bit wide signals that shift bit positions between their source logic blocks and their destination logic blocks). Implementation of circuits containing non-datapath logic which use irregular signals is achieved by using the individual clusters within the multi-bit logic block to implement a segment of the non-datapath logic. Conversely, routing the associated irregular signals on purely bus based routing resources can cause a loss in area efficiency. To accommodate these irregular signals, pairs of conventional routing tracks (using the same connection patterns as those used in the conventional unidirectional routing architecture) are used to augment the routing buses to form the multi-bit routing architecture [21]. For the remainder of this thesis the conventional routing tracks will be referred to as singular tracks.

The unidirectional multi-bit architecture is composed of multi-bit logic blocks interconnected by vertical and horizontal channels of routing tracks. The routing channels consist of both singular tracks and *M*-bit wide buses of channel widths W_f and W_c respectively. The multi-bit logic blocks each contain *M* configurable logic blocks whose input and output connections connect directly to those of the multi-bit logic block. Each configurable logic block is composed of *N* basic logic elements (BLE) and share *I* inputs.

Although previous works (DP-FPGA [14] and MB-FPGA [6]) have employed configuration memory sharing (CMS) among their bus based routing connections, the work done in [33] shows experimentally that using CMS routing has no significant advantage on area savings for bidirectional multi-bit architectures. Additionally, when routing circuits with a small percentage of multi-bit signals, the architecture employing CMS routing consumes more area than the same architecture employing non-CMS routing (each connection is controlled by an individual switch). Therefore, this research opts-out of employing configuration memory sharing on multi-bit routing connections. Practically, modifications to the MB-FPGA place and route tool by [33] already introduce the boolean variable *CMSSw* to control the application of configuration memory sharing. This *CMSSw* controls whether CMS is applied on multi-bit switch block connections. The MB-FPGA tool reads a value of either 1 (CMS) or 0 (no CMS – value which is set for this research) from the architecture file and builds the architecture accordingly.

Fig. 4.1 illustrates a simplified input connection block consisting of a single input bus of size M=4. The horizontal routing channel contains two 4-bit buses travelling in opposite directions as well as two singular tracks also travelling in opposite directions. Each individual connection of the input bus employs a multiplexor which selects between routing bus connections and singular connections. The size of each multiplexor is dependent on the combined values singular input connections and multi-bit input connections and determined by F_{cic} and F_{cif}. It can be seen in Fig. 4.1 that application of CMS to the input connection buses is not possible if selection of both multi-bit bus connections and singular connections are required. In the even singular connections are made, the input connection bus no longer functions as a 'bus', rather a group of individual connections which require independent control of each multiplexor. Given that two types of routing tracks (multi-bit and singular) are present within a routing channel, the length of each routing type can be specified. L_f denotes the length of singular tracks while L_c denotes the length of multi-bit routing tracks. Also pictured are isolation buffers which are used to isolate the tracks from the electrical effects of the input connections. While not pictured, additional input buses would be distributed evenly along the four sides of the multi-bit logic block. This is possible due to the logical equivalency between the input buses of the multi-bit logic blocks.

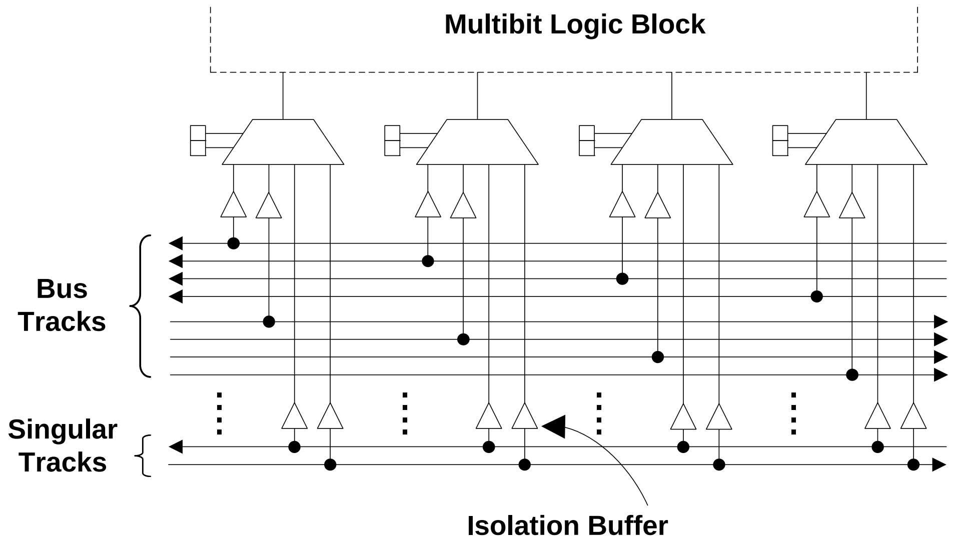


Fig. 4.1. Multi-bit logic block input connections

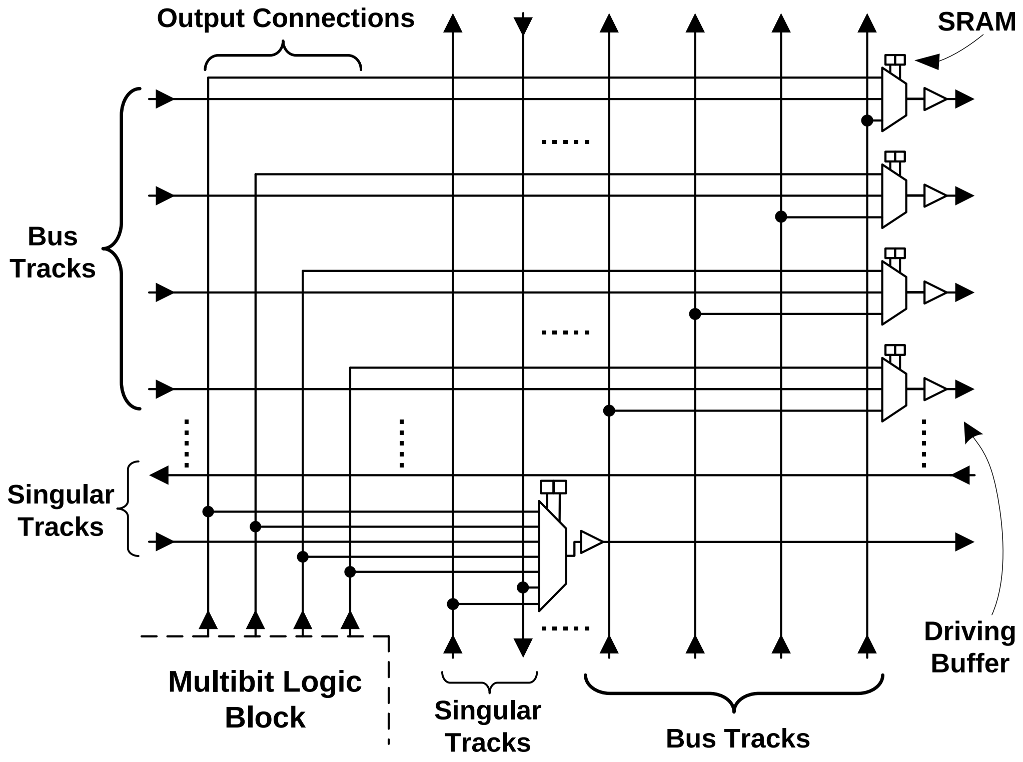


Fig. 4.2. Switch block with multi-bit logic block output connections

Fig. 4.2 illustrates the switch block connections for a single direction on a horizontal channel. Unlike Fig. 4.1, only a single bus travelling in one direction is pictured (while two singular unidirectional tracks are still included) in Fig. 4.2 to simplify the example. Also pictured is a single group of output connections from the multi-bit logic block output pins. The switch block multiplexors illustrate the three types of possible switch block connections: connections from tracks in the vertical channel, output connections from the nearest multi-bit logic blocks and a connection for the track travelling through the switch. Note, all tracks travelling to the right are assumed to end at this switch block location. Similar to the previous example, the logical equivalence among the output buses allow additional output buses (though not pictured) to be distributed evenly along the multi-bit logic block sides.

4.2 Buffer and Transistor Sizing

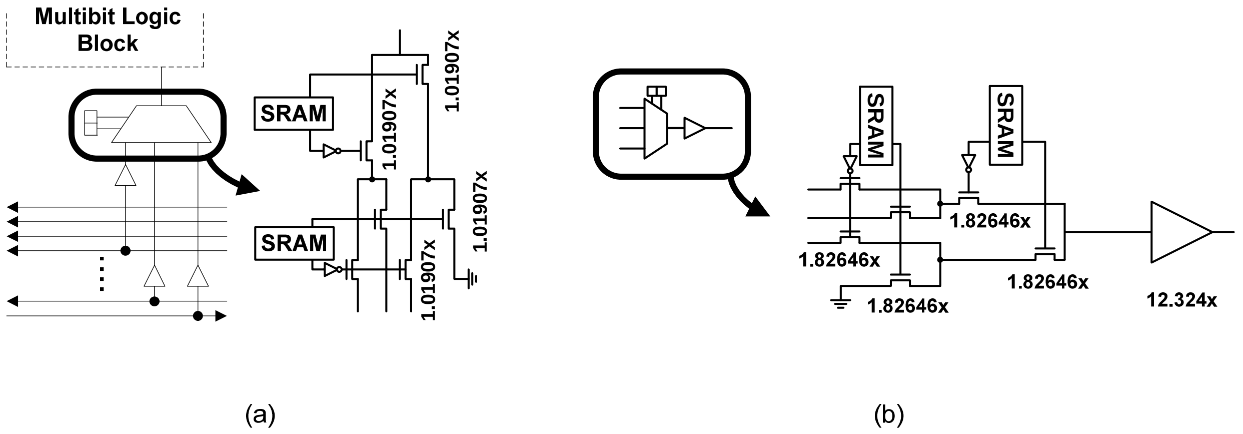


Fig. 4.3. Input and routing switch multiplexors implemented as a pass-transistor tree

In order to generate realistic data on the behavior of the unidirectional multi-bit architecture on unidirectional routing, logic and routing components must be modeled based on a modern process technology. In this research, accurate area and timing estimates for 90nm CMOS technology are employed and optimized for N=4, I=10 and L=2. The area and delay information is extracted from [32] and [34], whose transistor-level models are based on the Berkeley Predictive Technology Model (BPTM) [35]. As shown in Fig. 4.3 (a), the input multiplexors are built as a tree of pass-transistors and sized 1.01907 times that of a minimum-width transistor. Similarly, the switch block multiplexors are built as trees of pass-transistors adjoined by a driving buffer. As shown in Fig. 4.3, each transistor is 1.82646 times that of a minimum-width transistor while the driving buffer is designed as a three-stage buffer of size 12.324 minimum-width transistor area units. In both examples, each SRAM cell controlling the multiplexor select lines are assumed to be composed of 6 minimum-width transistors. The delay of an input connection starting from the routing track through the isolation buffer and the multiplexor to the multi-bit logic block is 0.07428ns. The switch block switch consisting of a multiplexor and driving buffer have an intrinsic delay of 0.07115ns.

Table 4.1 Internal Multi-bit Logic Block Delays

Delay Description	**Delay (ns)**
BLE output to CLB ouput pin	0
CLB input pin to BLE input	0.6077
BLE output to BLE input in the same CLB	0.05793
BLE input to BLE in combinational mode	0.2391
BLE input to storage component within BLE in sequential mode	0.2347
BLE storage component to BLE input in sequential mode	0.140

Furthermore Table 4.1 lists the delay values obtained from [32] for signals travelling through various logic block components such as input pins, output pins, BLEs and internal storage components (flip-flops). Also listed are timing estimates for specific paths when the BLEs are in sequential or combinational states.

4.3 Parameters

Overall, there are 12 variables used to parametrically describe the unidirectional multi-bit architecture. These parameters can be categorized as follows: multi-bit logic block parameters, routing track dimensions and connection parameters. N, I, k, M as defined before describe the size of the multi-bit logic block, the number of BLEs and their size. L_f, L_c, W_f and W_c describe the dimensions of the routing tracks. Finally F_{cif}, F_{cic}, F_{cof}, F_{coc} and T_s define the input and switch block connectivity of the routing tracks. T_s specifically describe the number of programmable connections and their topological arrangement within the switch block.

These parameters allow for a highly realistic modelling of the unidirectional multi-bit architecture. However, the combination of these parameters generates an extremely large design space requiring exploration that is both difficult and beyond the scope of this study. Therefore, most of these parameters are selected based on values determined to be optimal from previous architectural FPGA studies. Internal logic block parameters N and I are set to 4 and 10 respectively as the work in [36] has shown these to be efficient for bidirectional CLB based FPGAs. Additionally the value of k is set to 4 since [17] and [37] have shown a size 4 LUT yields a minimum in total routing area. The granularity M is set to 4 since it has been experimentally shown to yield the most area efficient results by [21]. T_s, for the unidirectional multi-bit architecture, is the disjoint switch block topology as it is ideal for segmented architectures [7] and widely used. $F_{cif} = F_{cic} = .5$, $F_{cof} = F_{coc} = .25$, $L_c = 2$ and $L_f = 2$ are used in this work. The studies done in [7] and [21] find these parameter values result in efficient area results for both singular-bidirectional and multi-bit-bidirectional architectures. While typically W_f and W_c are dependent variables whose values are determined by the binary search algorithm in the routing portion of the CAD flow.

4.4 Routing Resource Graph Generation

Many of the modifications to the VPR CAD tool involve changes to the routing resource graph. Specifically, the algorithms tasked with automatically generating the architecture based on architectural parameters (outlined in Section 4.3) are modified. The routing architecture is generated as directed graph where components including wires segments, input pins and output pins are represented as nodes while connections between these components are represented by edges. Once generated the process of routing a given circuit can proceed.

Initially, all data structures necessary for the storage of nodes and edges are allocated and initialized. One particular data structure, *seg_details*, is loaded with information about each wire segment. Examples of such information including wire segment length, start location (in x,y coordinates), end location (in x,y coordinates) and bit-wise position on a bus (if the wire segment is a multi-bit track) to name a few [6] [8]. This data structure was modified to include aspects of unidirectional routing such as directionality and a new switch type (multiplexor based) [8].

Next, the switch block connection patterns for the entire FPGA are generated. As mentioned in Section 4.3, the disjoint switch block topology is used in both conventional and multi-bit architectures. The switch blocks are generated by specifying each connection in the switch block with respect to the disjoint switch pattern. The connections can be made in an organized and predictable fashion by identifying three distinct switch block types in an FPGA: the core block, the corner block and the fringe block [8]. The core block is the most common type of switch block which consists of wires segments ending, beginning or passing through each of the four sides. Fig. 4.4 illustrates the VPR graphical output of a corner block for the unidirectional multi-bit architecture. If Fig. 4.4 is viewed in colour, then the purple lines indicate multi-bit connections while the green lines indicate singular connections. If not viewed in colour, the multi-bit connections account for the 8 tracks from the left on the vertical channel and bottom on the horizontal channel. The remaining tracks are singular tracks. The dark arrows indicate wire segments (the word segments is sometimes dropped to ease discussion) which end at this switch block location while the light grey arrows represent wires which pass through this switch block. The directions of the arrows indicate the directions in which the wires are driven. Fig. 4.4 also displays the multiplexors (black trapezoids with select lines) at the start positions of wire segments beginning at this switch block. Notice that the amount of wires which end on each side equals the amount of wires which start on each side. The staggering of wire segments results in an ideal N-to-N assignment of connections (for each type of routing track) involving ending

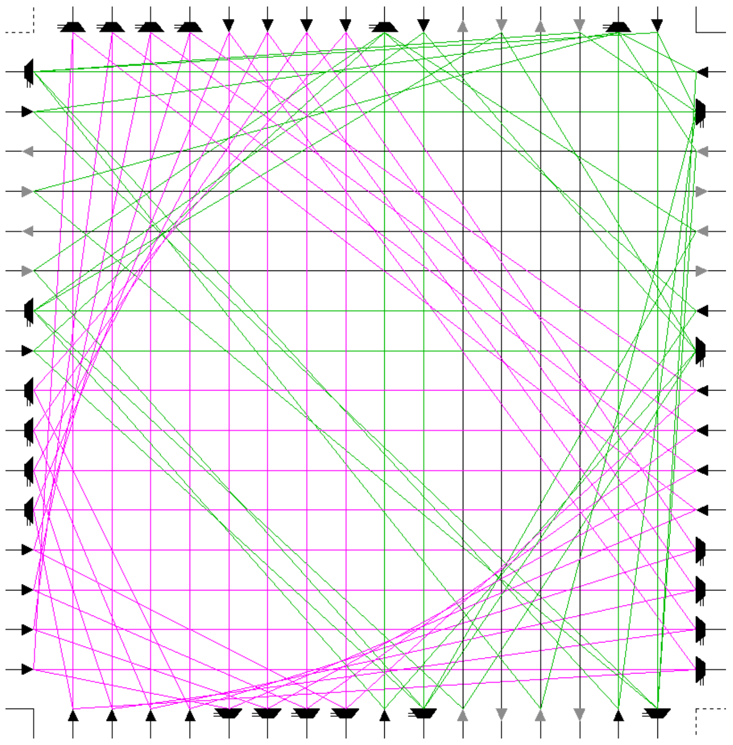


Fig. 4.4. Core switch block (disjoint topology)

wires and multiplexors of starting wires. The corner block as the name implies is the set of connections existing on the four corners of the FPGA. A corner block is similar to a core block but only contains two sides where all the wires on both sides either end or begin. Fig. 4.5 (a) illustrates the corner switch block with multi-bit and singular connections. Similar to the core block, each of the two sides contains the same number of ending wires and starting wires allowing for an N-to-N assignment of perfectly balanced connections. Additionally, Fig. 4.5 (a) clearly shows how mutli-bit connections and singular connections are segregated from each other. Finally, Fig. 4.5 (b) illustrates a fringe block which consists of three sides and is positioned along the edges of the FPGA. Two of the sides are similar to the core and corner block in that each type of routing track has an equal number of starting and ending wires. One side however, contains a different number of ending and beginning wires. In this case, staggering ensures M-to-N connectivity between the multiplexors of the starting wires and the ending wires

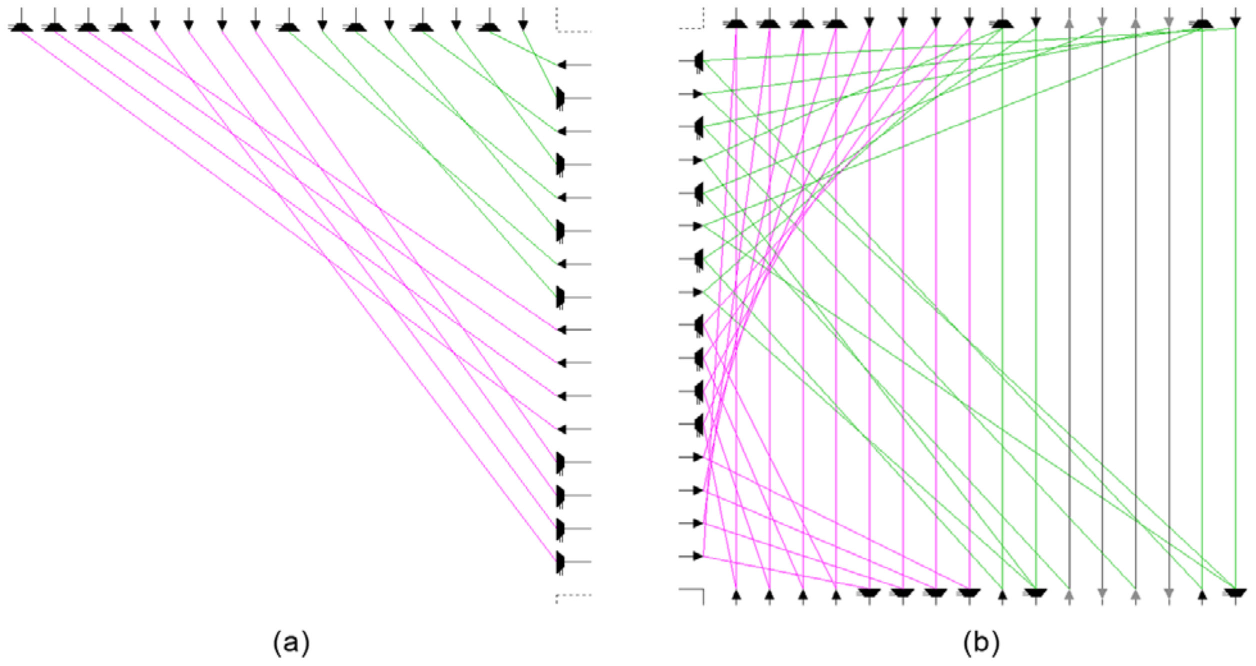


Fig. 4.5. Corner (a) and fringe (b) switch blocks (disjoint topology)

of each routing track type. This results in the amount of inputs between multiplexors being imbalanced by no more than one connection. This imbalance can be seen in Fig. 4.5 (b) where the multiplexors on the left side of the switch block have 1 less connection than the multiplexors on the top and bottom sides. Notice that for all three types of switch blocks, the multi-bit bus connections are made in a manner that maintains their bit positions. The algorithm which generates these connection patterns for each switch block on the FPGA is outlined in Fig. 4.6. The algorithm begins with determining the switch block type and then proceeds to indentify which wires end and start at this switch block location for each of the applicable sides along with their respective directions. The algorithm then iterates for each side, the generation of the connections for the remaining three sides (2 sides for fringe or 1 side for corner) depending on the switch block type. Sub-processes in generating connections are performed independently for multi-bit buses and singular tracks as indicated in Fig. 4.6.

```
FOR each switch block in the FPGA
    Determine switch type (core, fringe or corner)

    FOR tracks of each type (multi-bit and singular)
        FOR each side
            Determine ending wire segments and direction
            Determing starting wires segments and direction
        END FOR
    END FOR

    FOR each side with starting wires
        Determine the orientation of the other 3 sides (cw, ccw, opp)

        FOR each of the 3 sides with ending wires
            IF side is clock-wise (cw)
                FOR tracks of each type (multi-bit and singular)
                    Generate connections between ending and
                    starting wires as per switch block type
                END FOR
            END IF
            IF side is counter clock-wise (ccw)
                FOR tracks of each type (multi-bit and singular)
                    Generate connections between ending and
                    starting wires as per switch block type
                END FOR
            END IF
            IF side is opposite (opp)
                FOR tracks of each type (multi-bit and singular)
                    Generate connections between ending and
                    starting wires as per switch block type
                END FOR
            END IF
        END FOR
    END FOR
END FOR
```

Fig. 4.6. Switch block connection pattern generation

The same switch multiplexors are referenced again as edges are generated to represent connections from output pins. First, the absolute values for F_{cof}, F_{coc}, F_{cif} and F_{cic} are determined from the fractional values (in architecture file) since the number of mutli-bit and singular tracks are known. The equations used to calculate the absolute values of each are listed below:

$$F_{cif\_abs} = \left\lceil F_{cif} \times \frac{W_f}{2} \right\rceil \times 2, \qquad F_{cic\_abs} = \left\lceil F_{cic} \times \frac{W_c/M}{2} \right\rceil \times 2$$

$$F_{cof\_abs} = \left\lceil F_{cof} \times \frac{W_f}{2} \right\rceil \times 2, \qquad F_{coc\_abs} = \left\lceil F_{coc} \times \frac{W_c/M}{2} \right\rceil \times 2$$

These calculations consider that the absolute values F_{cof}, F_{coc}, F_{cif} and F_{cic} must be a multiple of 2. Additionally these values must at a minimum be a value of 2, due to unidirectional routing requirements. It is important to note that F_{cic_abs} and F_{coc_abs} represent the number of bus connections while F_{cif_abs} and F_{cof_abs} represent the number of singular connections. The edges are then generated from output pins to an F_{cof_abs} or F_{coc_abs} amount of drivers on wire segments traversing both directions.

Fig. 4.7 shows a typical VPR graphical output of a routing channel with input and output connection buses from two multi-bit logic blocks (located at the top and bottom). The output bus connections are shown as two 4-bit wide buses (in red) from the right side of the figure, while the remaining 4-bit wide buses (in blue) from the left are the input buses. Although output

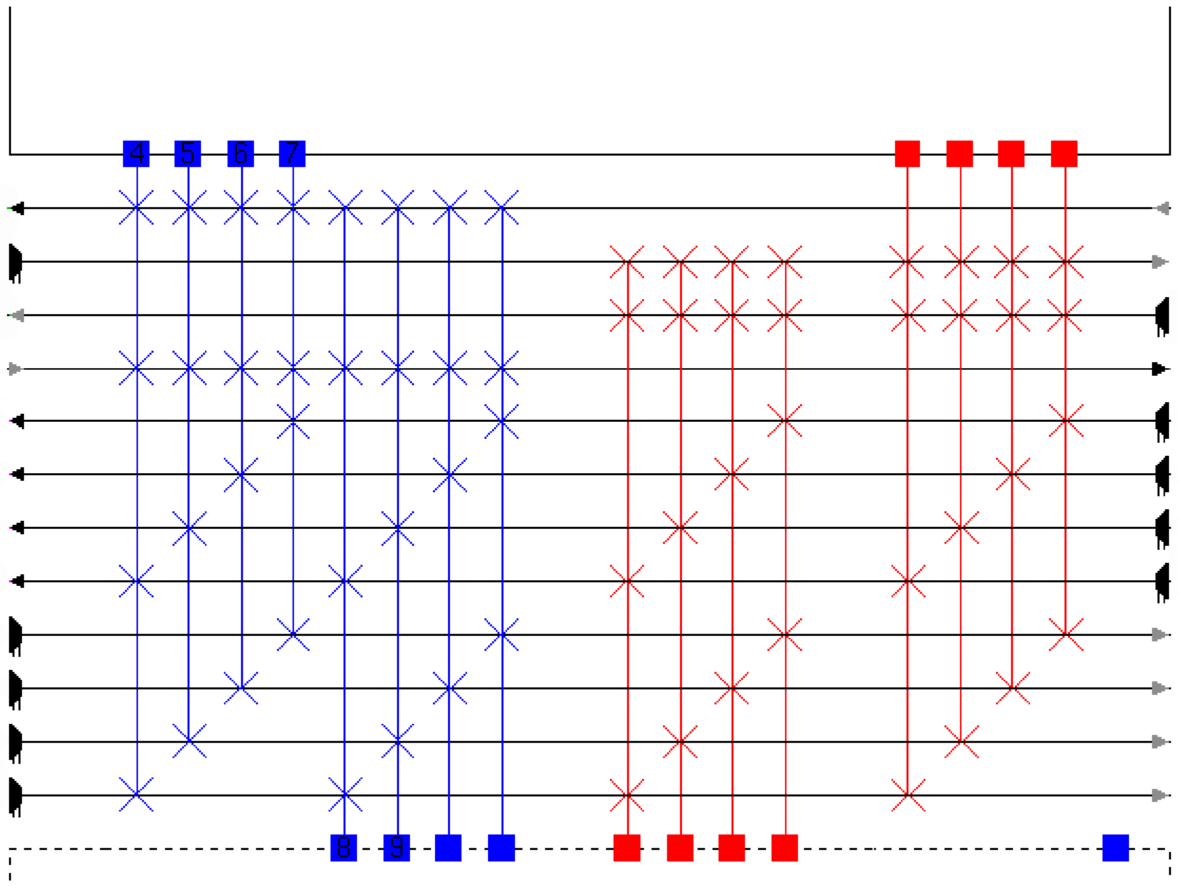


Fig. 4.7. Input and Output connection buses from VPR (MB-FPGA) output

connections electrically connect directly to routing switch multiplexors, this becomes visually congested for large designs. Therefore, these connections are drawn perpendicular to the tracks that begin at either side of a logic block even though logically the connections are made to the routing switch multiplexors. Notice in Fig. 4.7, that the first and fourth singular wires from the top are not connected to the output buses as these wires do not begin at this location.

Finally the switch block patterns discussed earlier in this chapter are used to create the edges between wire segments within the switch block. Furthermore the values of F_{cif_abs} and F_{cic_abs} are used to create the appropriate amount of edges representing connections from the routing tracks to multi-bit logic block input pins.

In conclusion, this chapter has described the unidirectional mutli-bit architecture. The parameters defining this architecture are described along with references to previous works that have found them to be most optimal. Finally, a detailed description of the process and algorithms used to model the unidirectional multi-bit architecture in a CAD tool are presented. This CAD tool is then used to empirically determine the area and delay efficiency of the unidirectional multi-bit architecture.

Chapter 5 : Experimental Results

This chapter will present a description of the experiments conducted using the modified CAD tools as well as an analysis of the results. The purpose of these experiments is to address the following questions:

1. How area efficient (or inefficient) is the unidirectional multi-bit architecture in comparison to the conventional architecture?
2. What proportion of multi-bit routing tracks results in the greatest average area-efficiency given the sample set of benchmark circuits?
3. How does the timing performance of the unidirectional multi-bit architecture compare to that of the conventional one?

Section 5.1 describes the experimental setup. Section 5.2 discusses the effects of unidirectional multi-bit connections on area efficiency and address questions 1 and 2. Finally Section 5.1 presents results pertaining to the timing analysis and track count which ultimately addresses question 3.

5.1 Experimental Setup

To experimentally evaluate the effect of multi-bit connection patterns on the area efficiency of unidirectional routing architectures, 15 benchmark circuits [6] consisting of datapath components from Sun Microsystems' Pico-Java processor [38] are implemented. Each of these circuits are synthesized and mapped onto multi-bit logic blocks using datapath-oriented synthesis and packing tools described in Section 2.2.3. In this investigation, the benchmark circuits are implemented on both conventional and unidirectional multi-bit architectures to compare their performance. In order to fairly assess the area results, the same routing tool (the multi-bit routing tool described in Chapter 5) is used for every experiment, eliminating any effects arising due to routing algorithm variations. The 13 architectural parameters outlined and rationalized in Section 4.3 are entered appropriately in the architectural file. Table 5.1 summarizes all these architectural parameters and their values.

Table 5.1. Architecture Parameters and Values

Parameter Type	Parameter	Value
Multi-bit Logic Block Parameters	N	4
	I	10
	k	4
	M	4
Routing Track Dimensions	L_f	2
	L_c	2
	W_f	-
	W_c	-
Connection Parameters	F_{cif}	.5
	F_{cic}	.5
	F_{cof}	.25
	F_{coc}	.25
	T_s	disjoint

Analysis of both implementations is achieved by constraining the routing bus channel width W_c and then attempt to successfully route the circuit with a minimum number of singular tracks W_f using the binary search algorithm of the router. The conventional implementation involves constraining W_c to zero, thereby routing the benchmark circuits using only singular unidirectional tracks. Conversely the unidirectional multi-bit architecture is evaluated over a range of values of W_c. Here, the router is executed for fixed values of W_c starting with 2M bus tracks (8 tracks in this experiment) and incremented by 2M tracks to an upper limit of 120 tracks (30 buses). Each of these circuit implementations are then sorted according to various percentile ranges. Each percentile range represents the proportion of routing bus tracks as a function of the total routing tracks in a routing channel. The minimum area results are chosen for all 15 benchmark circuits and arithmetically averaged for each percentile range. This allows a designer to gauge how many tracks to include in physical hardware for the given sample of circuits. Similarly the minimum amount of total track segments are determined for each circuit implementation and arithmetically averaged for each percentile range. Finally the best critical-path delays of the multi-bit implementation are determined for each circuit and compared against the conventional implementations.

5.2 Effect of Routing Buses on Area

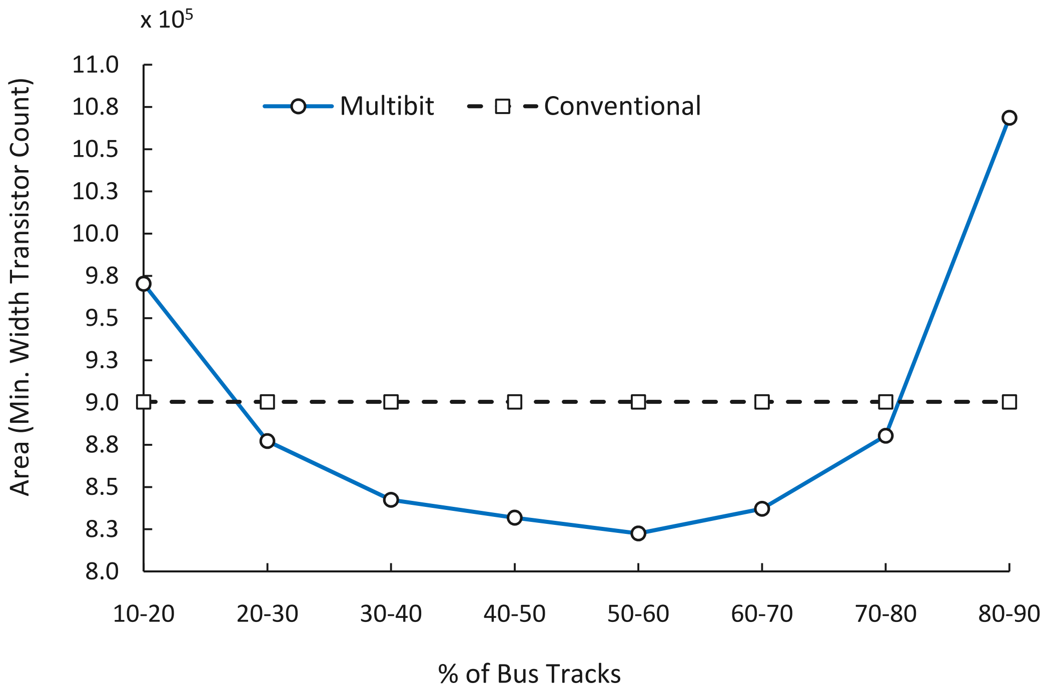


Fig. 5.1. Area as a function of the percentage of multi-bit tracks

Fig. 5.1 is a graph of the average area consumed by 15 benchmark circuits over a series of 8 percentile ranges. The percentile ranges are (10% to 20%], (20% to 30%], (30% to 40%], (40%

to 50%], (50% to 60%], (60% to 70%], (70% to 80%] and (80% to 90%] The solid curve represents the average multi-bit implementation area while the dashed curve represents the average conventional implementation area. Notice, the percentile range (0% to 10%] is not present in the plot since there are only a few circuit designs in the benchmark set utilizing this range of routing bus tracks. When 10% - 30% of the tracks in a channel are routing bus tracks, an increase in area is observed. This occurs due to the inability of input pins to connect such few routing buses at the current value of F_{cic}. The generated number of routing buses remains unused while the router attempts to reroute these signals using singular routing tracks, resulting in an increase routing area. A similar observation is made for the bidirectional multi-bit architecture in [21]. The 50% - 60% range of routing bus tracks achieves the greatest area efficiency with an 8.6% routing area reduction over the conventional architecture. As the percentage of bus tracks increase past 70%, the number of constrained bus tracks will exceed the amount actually required by each circuit by a factor of 2 (recall unidirectional routing requires an even number of tracks/buses). The router uses these excess bus tracks to route singular signals, resulting in drastically higher area consumption.

Table 5.2 displays the best implementation area results of each benchmark circuit for both multi-bit and conventional implementations. The results are then categorized into percentile ranges based on the regularity of each circuit. These categories are listed in Column 1. Column 3 lists this regularity (in ascending order) as the percentage of total signals in each circuit that are grouped into 4-bit wide multi-bit buses. Columns 4 and 5 list the effective routing areas of circuit implementations on both the unidirectional multi-bit architecture and conventional architecture. Arithmetic averages are computed and displayed for each percentile range and implementation type. As shown, almost all benchmark circuits routed on the multi-bit architecture are more area efficient than those routed on the conventional architecture for every proportion of multi-bit signals listed in Table 5.2. Column 6 lists the percentage difference in routing area of the unidirectional multi-bit implementation over the conventional one. Every circuit implemented using unidirectional multi-bit resources consumes less routing area with the exception of the circuit *code_seq_dp*. Additionally, it can be seen that larger circuits containing a higher proportion of datapath components tend to realize larger area savings.

Table 5.2. Routing Area vs. Proportion of Routing Buses Per Circuit

% Range of Multibit Signals	**Benchmark Circuit**	**% of Multibit Signals**	**Routing Area (min. width transistor count)**		**% Difference in Area over Conventional Implementation**
			Multibit	**Conventional**	
10-20%	multmod_dp	18.71	912194	946289	3.6
	Average		***912194***	*946289*	*3.6*
20-30%	prils_dp	29.14	214759	233488	8.0
	code_seq_dp	29.37	315211	298276	-5.7
	Average		***264985***	*265882*	*0.3*
30-40%	exponent_dp	31.36	238851	274516	13.0
	incmod	37.41	431668	465967	7.4
	pipe_dpath	39.42	217886	255391	14.7
	Average		***296135***	*331958*	10.8
40-50%	smu_dpath	41.08	332538	353157	5.8
	imdr_dpath	42.88	571594	628280	9.0
	mantissa_dp	44.11	7.81E+05	8.56E+05	8.7
	icu_dpath	48.60	1590000	1910000	16.8
	ucode_dat	48.93	952775	1140000	16.4
	Average		***845655.2***	*977410*	13.5
50-60%	ex_dpath	50.25	4.01E+06	4.88E+06	17.8
	rsadd_dp	51.06	147503	1.67E+05	11.5
	dcu_dpath	54.01	871663	1010000	13.7
	Average		***1.68E+06***	*2.02E+06*	17.0
60+%	ucode_reg	65.64	6.88E+04	8.33E+04	17.4
	Average		***6.88E+04***	*8.33E+04*	17.4

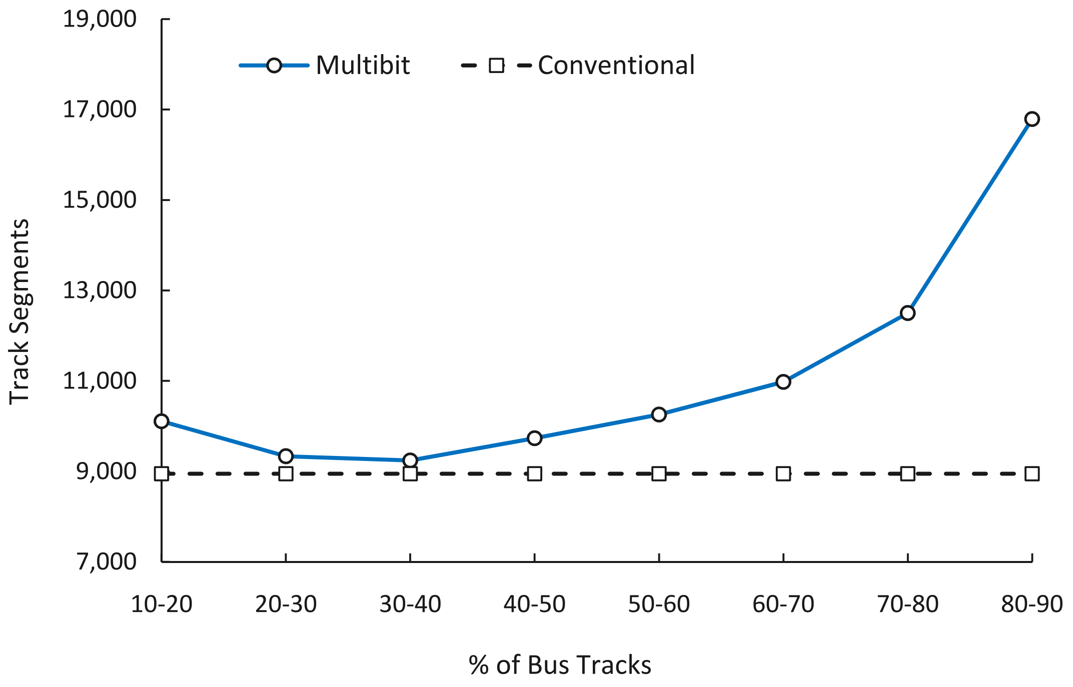


Fig. 5.2. Number of track segments as a function of the percentage of multi-bit tracks

5.3 Delay and Track Segment Results

Fig. 5.2 plots the number of wire segments utilized per circuit, averaged over 15 benchmark circuits as a function of the percentage of routing bus tracks. The total number of track segments is an ideal metric to monitor as it takes into account both channel width and size of the FPGA. It is observed that the best unidirectional multi-bit architecture (with 50% to 60% multi-bit tracks) employs 14.6% more track segments over the conventional architecture. Fig. 5.2 shows that even though the total number of routing segments of the unidirectional multi-bit architecture (for the 20% to 80% range of multi-bit tracks) is greater, routing area savings can still be achieved. This is due to the multi-bit connections employing smaller multiplexors than the conventional ones. The difference in multiplexor size is attributed to the sparse input and output connection patterns of the unidirectional multi-bit architecture where buses are connected to other buses on the same bit positions.

The focus of this research is on FPGA area efficiency. Although no theoretical gains in timing performance can be hypothesized, it is essential to determine how critical path delay is affected by the implementation of multi-bit routing resources. Specifically, it is important to determine whether the gains in area efficiency described in Section 5.2 come at the cost of decreased performance. Table 5.3 summarizes the critical path delays of the entire set of benchmark circuits implemented on the most area efficient conventional and multi-bit architectures. The geometric mean is calculated for each type of architectural implementation and is displayed. As shown by the geo-mean values, the multi-bit architectures perform slightly better than the conventional architecture. Furthermore, each of the multi-bit circuit implementations performs equally or demonstrates a slight performance increase.

Table 5.3. Critical Path Delays of Benchmark Circuits

Benchmark Circuit	**Critical Path Delay (s)**	
	Conventional	**Multi-bit**
code_seq_dp	6.05E-09	5.36E-09
dcu_dpath	3.38E-09	2.76E-09
ex_dpath	1.64E-08	1.62E-08
exponent_dp	8.20E-09	8.00E-09
icu_dpath	1.17E-08	1.17E-08
imdr_dpath	1.55E-08	1.54E-08
incmod	1.59E-08	1.48E-08
mantissa_dp	4.35E-09	3.90E-09
multmod_dp	1.29E-08	1.20E-08
pipe_dpath	6.09E-09	5.85E-09
prils_dp	9.78E-09	7.08E-09
rsadd_dp	1.34E-08	1.32E-08
smu_dpath	1.28E-08	1.27E-08
ucode_dat	3.58E-09	3.12E-09
ucode_reg	1.42E-09	1.42E-09
Geometric Mean	***7.78E-09***	*7.21E-09*

In conclusion, this chapter has addressed the question of routing area efficiency of the unidirectional multi-bit architecture. The experiments described in this chapter have also determined the percentage range of multi-bit routing tracks necessary to achieve the greatest average area efficiency. The critical path delays of the best 15 benchmark circuits have shown that the unidirectional multi-bit architecture performs equally or slightly better than the conventional architecture. Furthermore, Appendix A contains the results for the same experiments conducted in this chapter but with F_{cif} and F_{cic} set to values of 0.4. Finally, Appendix B contains the best (least active routing area) FPGA implementations for each of the 15 benchmark circuits.

Chapter 6 : Conclusion

6.1 Summary

This study has explored the effect on FPGA area efficiency of multi-bit connections using unidirectional routing in order to efficiently implement arithmetic intensive circuits. To this end, this work is the first to study the effects of implementing datapath circuits on unidirectional multi-bit architecture. Initially a simple theoretical datapath circuit is mapped onto conventional and bus-only architectures where the total area results of each are compared. From these results, the estimates and limits on area efficiency by using routing buses are found. In order to accommodate the use of non-ideal signals in modern circuits, pairs of singular signals are added to the routing buses to form the unidirectional multi-bit routing architecture.

The actual effectiveness of this multi-bit architecture is determined by comparing the implementation area of 15 benchmark circuits mapped on multi-bit and conventional architectures. It is found that the best architecture consists of 50% to 60% routing bus tracks with an average routing area reduction of 8.6% over the best conventional architecture. The track segment results show that despite a higher amount of total routing segments, the unidirectional multi-bit architecture still achieves routing area savings from the use of smaller multiplexors due to sparse connection patterns. Finally, larger circuits containing a higher proportion of datapath components tend to realize larger area savings.

The best critical-path delays of the multi-bit and conventional implementations are determined for comparison. It is found that each of the multi-bit circuit implementations performs (timing wise) equally or demonstrates a slight performance increase.

6.2 Future Work

The work done in this thesis is considered an initial investigation into unidirectional multi-bit routing. Several opportunities to conduct research arise from this thesis which should be pursued.

One such line of research is the full or partial exploration of the design space that is defined by the architectural parameters M, L_f, L_c, F_{cif}, F_{cic}, F_{cof} and F_{coc}. As noted in Section 4.3, the values of these parameters in this research are intelligently selected based on previous FPGA studies which have shown them to be optimal. Although the results in this thesis are a positive indication of possible reductions in routing area consumption, further investigation into this large design space may reveal more optimal values of these parameters.

The configuration memory sharing of routing connections was not employed in this research as previous research on bidirectional multi-bit architectures has shown area *increases* from CMS connectivity. However, an actual comparison of unidirectional multi-bit architectures with and without CMS should be performed to conclusively determine their usefulness.

Lastly, new academic CAD tools support the use of heterogeneous logic blocks or 'hard blocks' (ie. memory or multipliers) intermixed with standard logic blocks. These blocks can be of various sizes (with restrictions) but would have to be a multiple of the area occupied by a multi-bit logic block. Research could be conducted into creating or modifying the existing architecture to practically accommodate heterogeneous logic blocks, multi-bit logic blocks, unidirectional multi-bit routing tracks and routing connections.

Appendix A: Results for $F_{cif} = .4$ and $F_{cic} = .4$

This appendix presents the area, delay and channel width results for the following architectural parameters $N = 4$, $I = 10$, $k = 4$, $M = 4$, $\boldsymbol{F_{cif} = F_{cic} = .4}$, $F_{cof} = F_{coc} = .25$, $L_c = 2$, $L_f = 2$ and Ts = disjoint. These results supplement those presented in Section 5.2 and Section 5.3. Section A.1 presents results in a format identical to Fig. 5.1 and Table 5.2. Similarly Section A.2 presents results in a format identical to Fig. 5.2 and Table 5.3.

A.1 Area Results

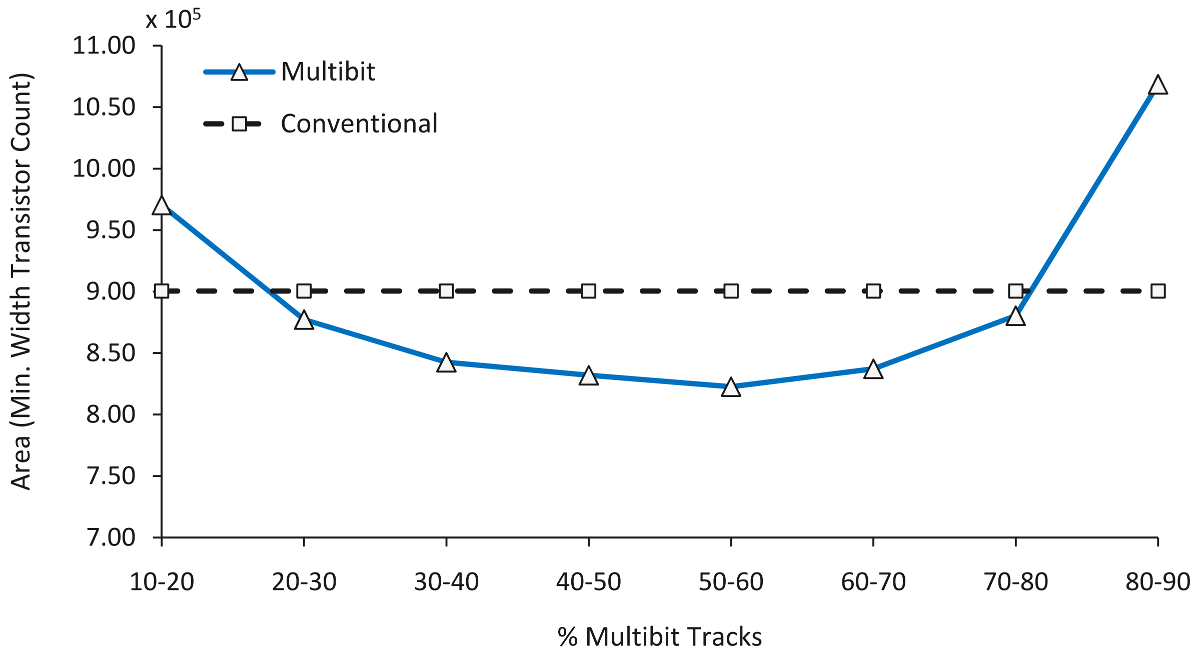


Fig. A.1. Area as a function of the percentage of multi-bit tracks ($F_{cif} = F_{cic} = .4$)

Table A.1. Routing Area vs. Proportion of Routing Buses per Circuit ($F_{cif} = F_{cic} = .4$)

% Range of Multibit Signals	**Benchmark Circuit**	**% of Multibit Signals**	**Routing Area (min. width transistor count)**	
			Multibit	**Conventional**
10-20%	multmod_dp	18.71	848639	897133
	Average		***848639***	*897133*
20-30%	prils_dp	29.14	198782	216042
	code_seq_dp	29.37	306395	275055
	Average		*252589*	***245549***
30-40%	exponent_dp	31.36	240505	252498
	incmod	37.41	412007	432337
	pipe_dpath	39.42	213892	240417
	Average		***288801***	*308417*
40-50%	smu_dpath	41.08	301166	332538
	imdr_dpath	42.88	549582	589023
	mantissa_dp	44.11	763716	810918
	icu_dpath	48.60	1539910	1841830
	ucode_dat	48.93	860405	1091800
	Average		***802956***	*933222*
50-60%	ex_dpath	50.25	3738560	4411750
	rsadd_dp	51.06	134185	148802
	dcu_dpath	54.01	823324	1050760
	Average		***1565356***	*1870437*
60+%	ucode_reg	65.64	60799	76219
	Average		***60799***	*76219*

A.2 Delay and Channel Width Results

The first result presented here is the channel widths (independent of FPGA size) of both the unidirectional multi-bit architecture and the conventional architecture. Furthermore Table A.2 lists the critical path delays for both unidirectional multi-bit and conventional circuit implementations.

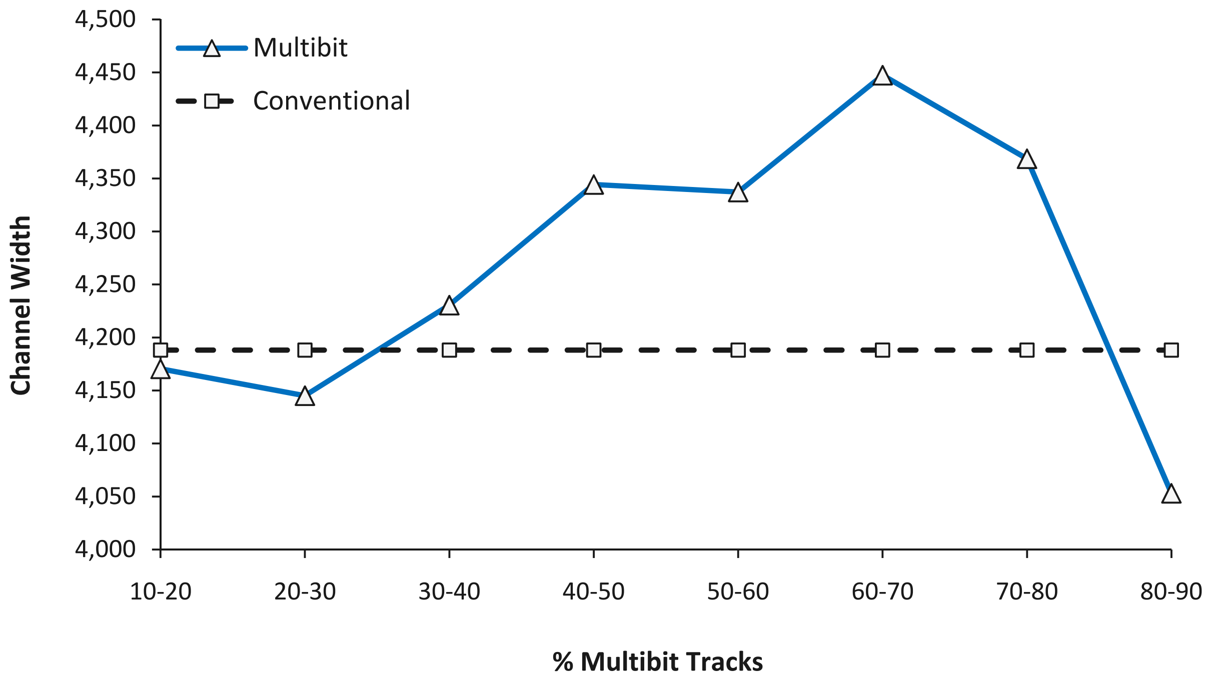


Fig. A.2. Channel width as a function of the percentage of multi-bit tracks ($F_{cif} = F_{cic} = .4$)

Table A.2. Critical Path Delays of Benchmark Circuits ($F_{cif} = F_{cic} = .4$)

Benchmark Circuit	**Critical Path Delay (s)**	
	Conventional	**Multibit**
code_seq_dp	6.05E-09	5.36E-09
dcu_dpath	3.38E-09	2.76E-09
ex_dpath	1.64E-08	1.62E-08
exponent_dp	8.20E-09	8.00E-09
icu_dpath	1.17E-08	1.17E-08
imdr_dpath	1.55E-08	1.54E-08
incmod	1.59E-08	1.48E-08
mantissa_dp	4.35E-09	3.90E-09
multmod_dp	1.29E-08	1.20E-08
pipe_dpath	6.09E-09	5.85E-09
prils_dp	9.78E-09	7.08E-09
rsadd_dp	1.34E-08	1.32E-08
smu_dpath	1.28E-08	1.27E-08
ucode_dat	3.58E-09	3.12E-09
ucode_reg	1.42E-09	1.42E-09
Geometric Mean	***7.77889E-09***	*7.20765E-09*

Appendix B: Graphical Outputs of Circuit Implementations

This appendix presents the graphical layouts from the modified MB-FPGA CAD tool of placed and routed circuit nets. Each figure displays the best (least active routing area) FPGA implementations for each of the 15 benchmark circuits.

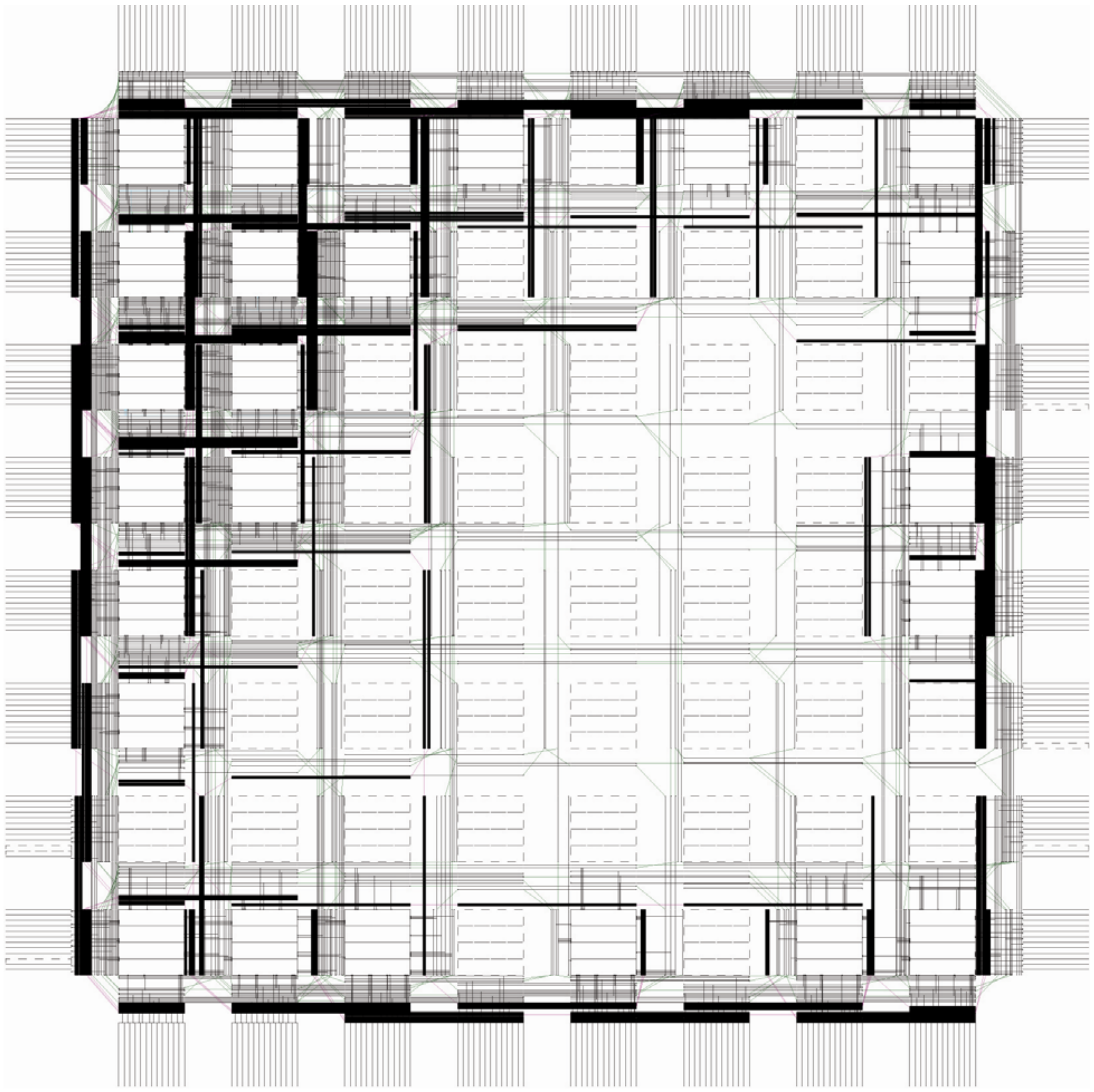


Fig. B.1. Routed nets of circuit: code_seq_dp

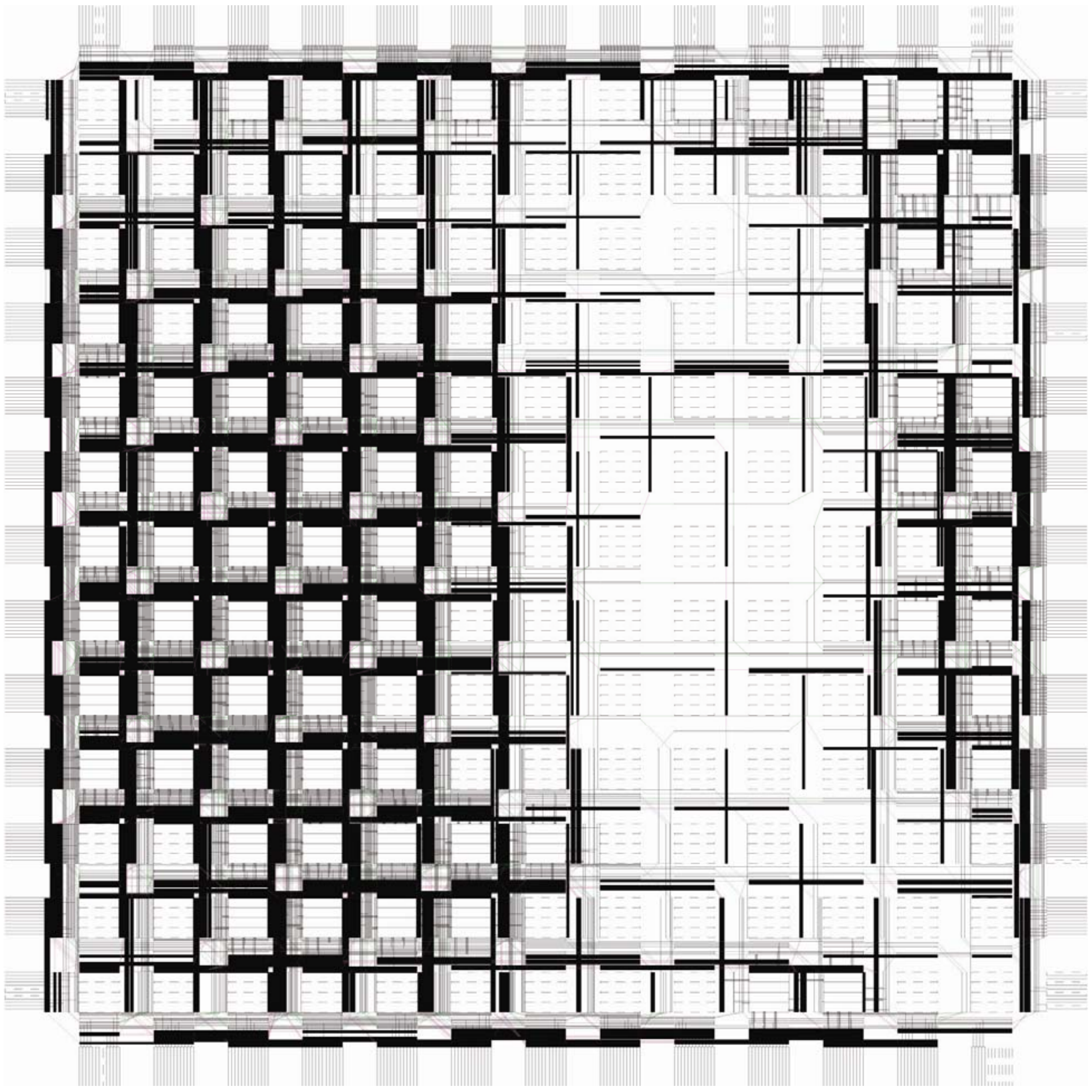


Fig. B.2. Routed nets of circuit: dcu_dpath

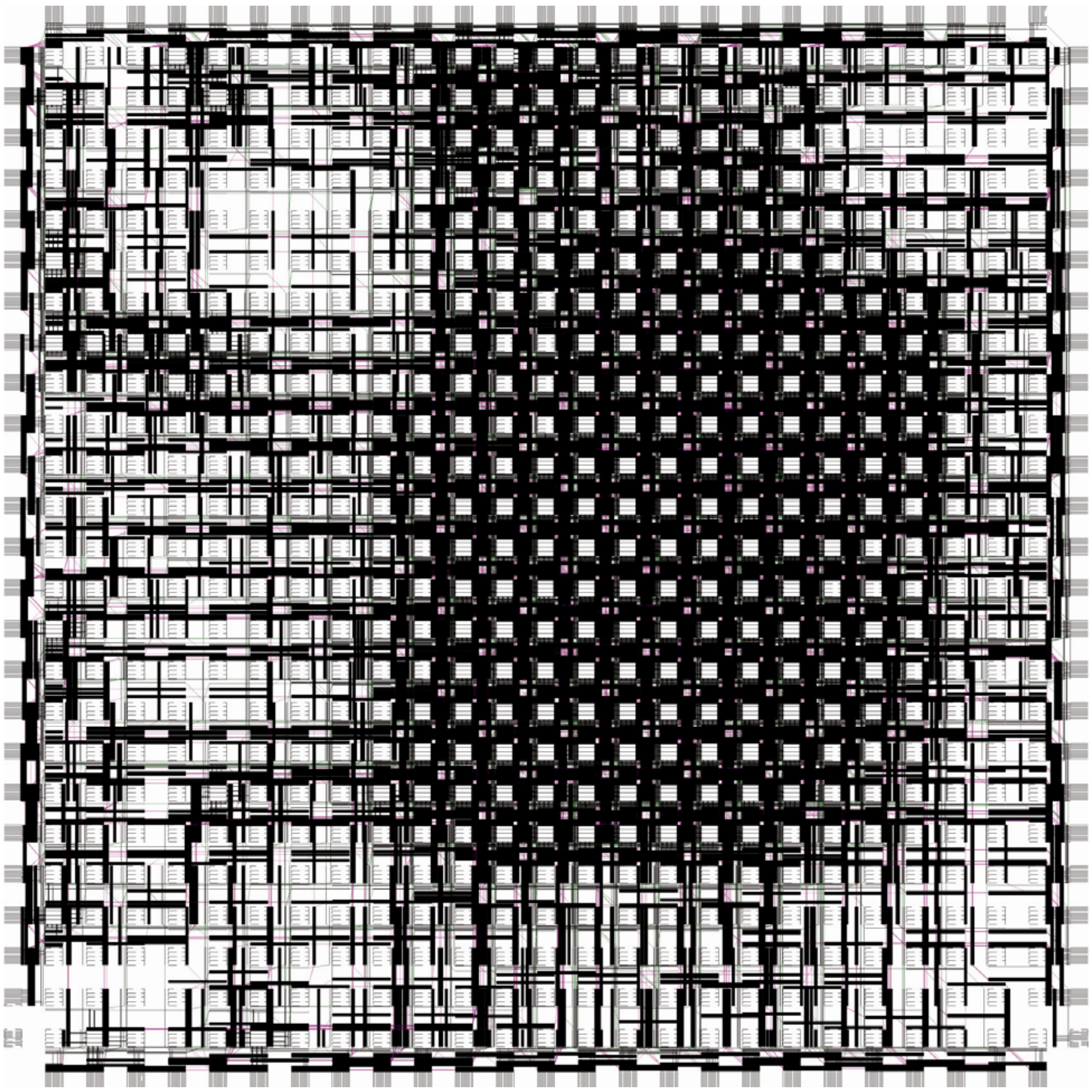


Fig. B.3. Routed nets of circuit: ex_dpath

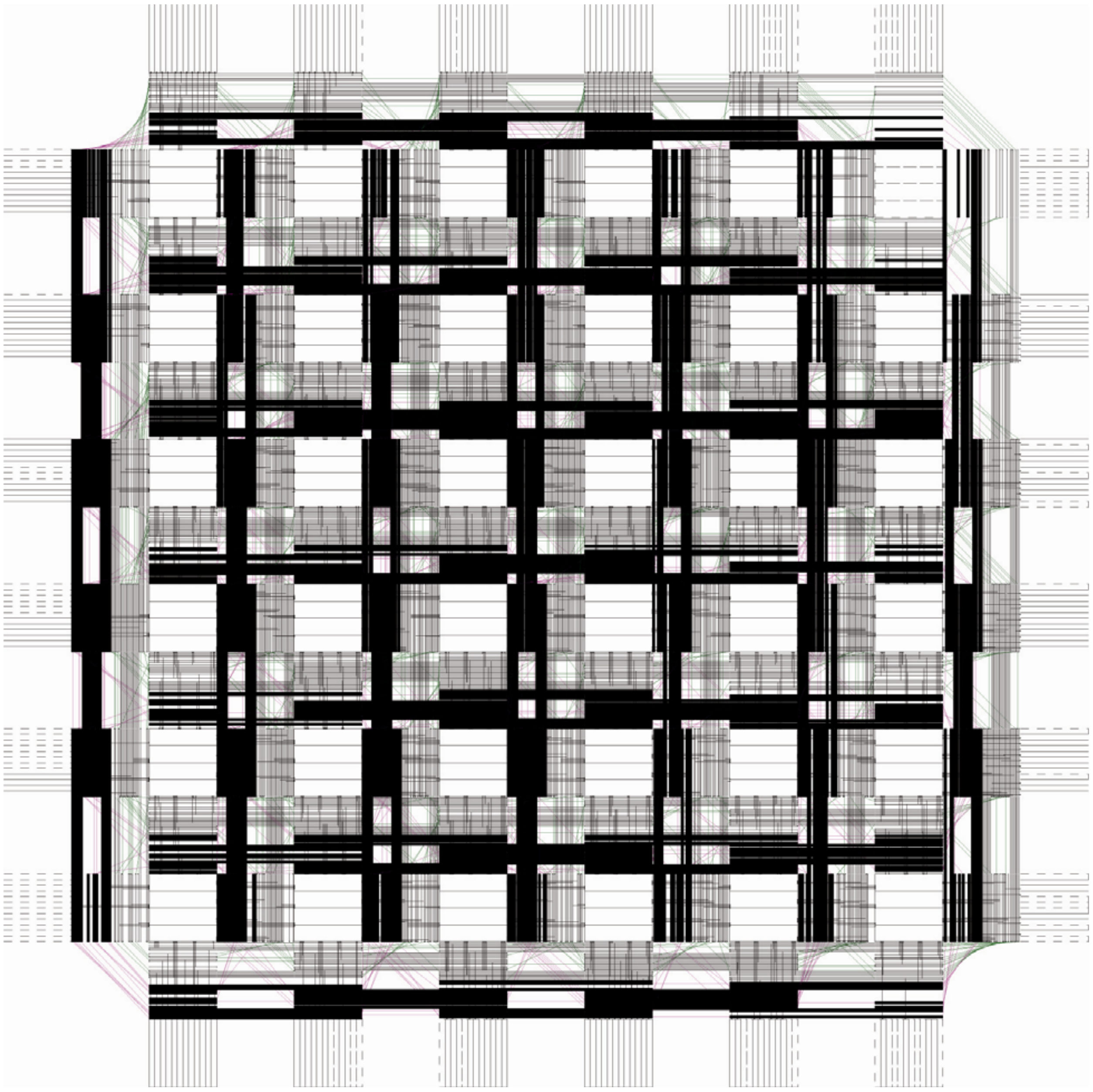


Fig. B.4. Routed nets of circuit: exponent_dp

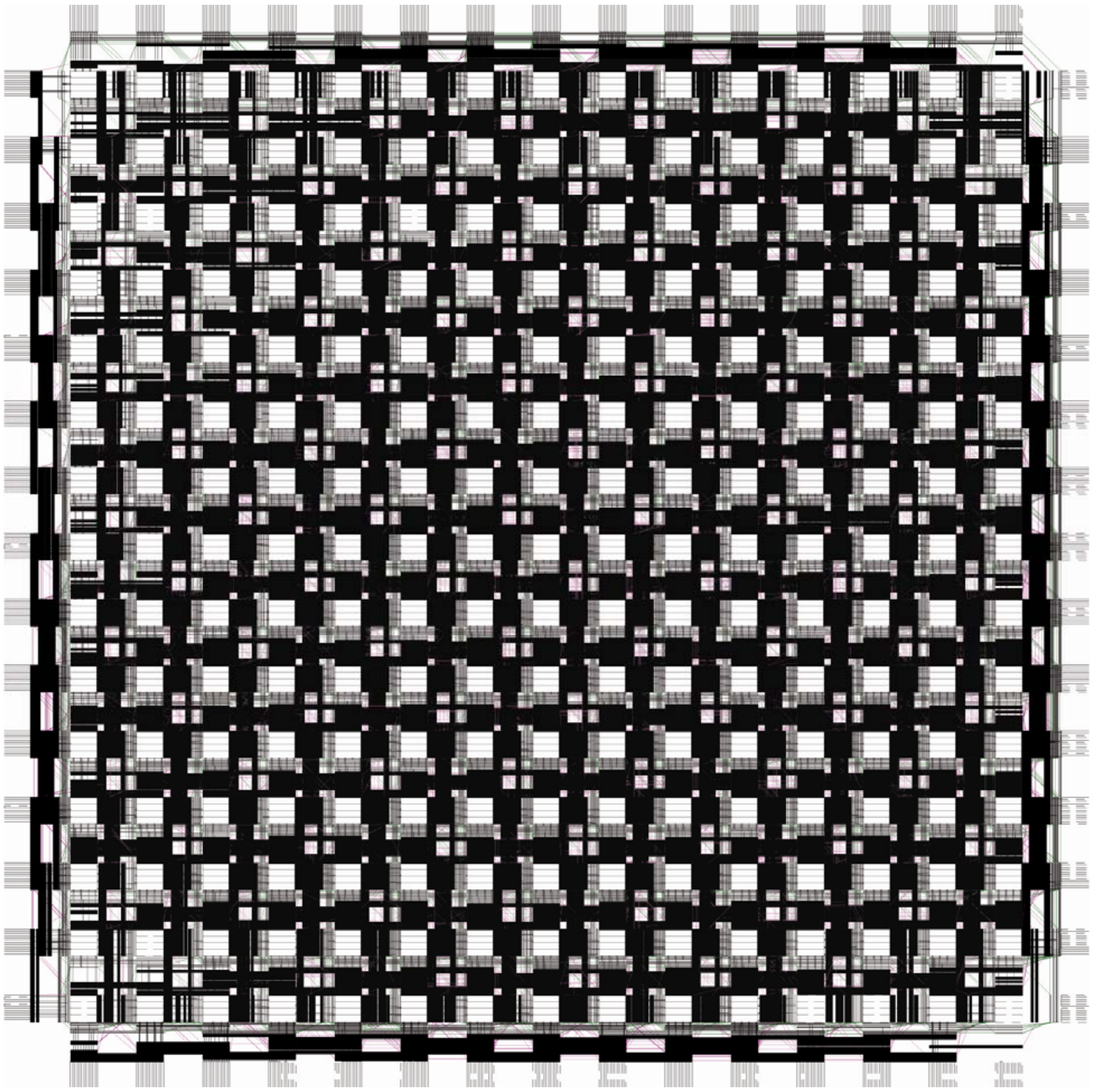


Fig. B.5. Routed nets of circuit: icu_dpath

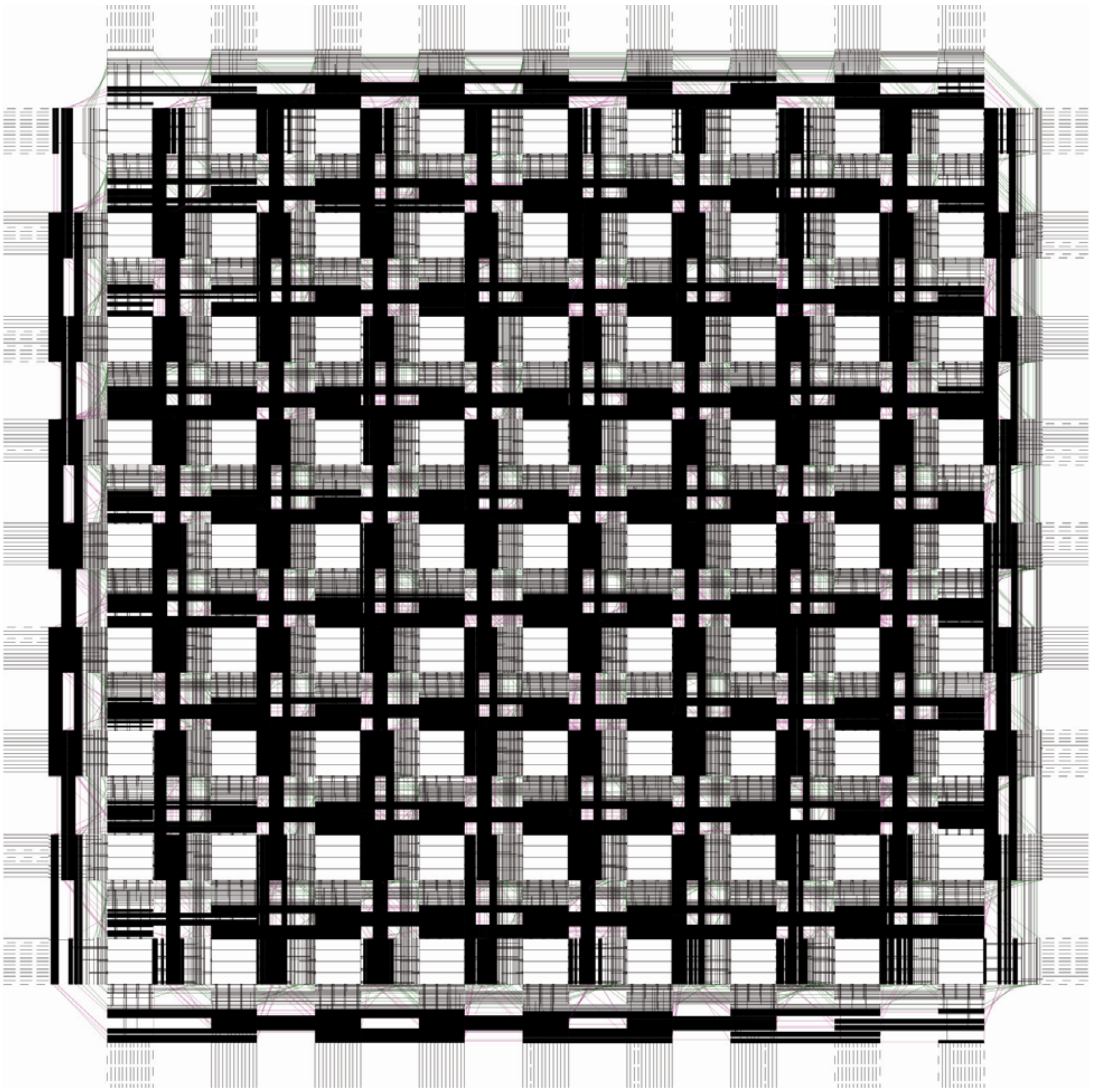


Fig. B.6. Routed nets of circuit: imdr_dpath

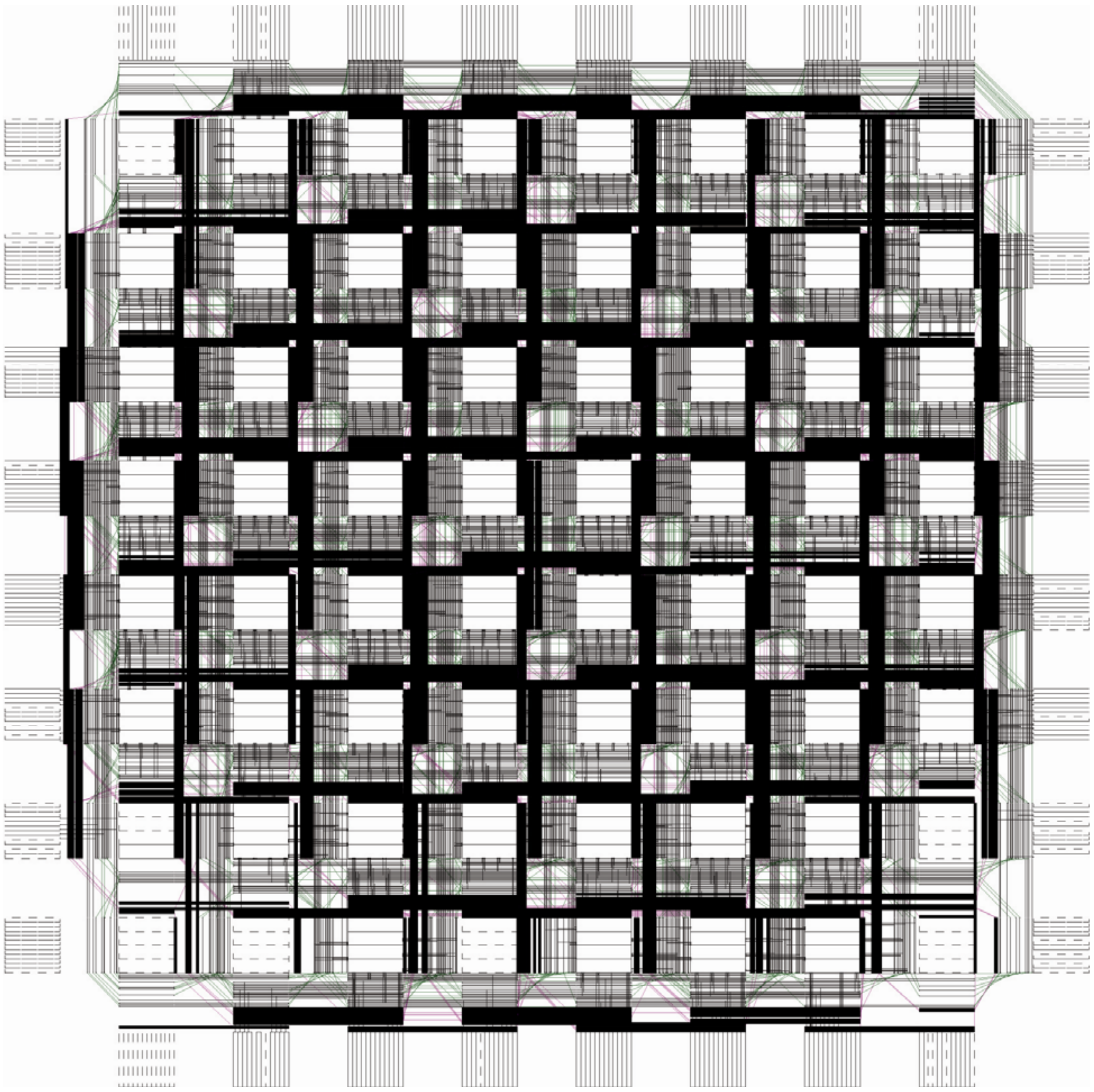


Fig. B.7. Routed nets of circuit: incmod

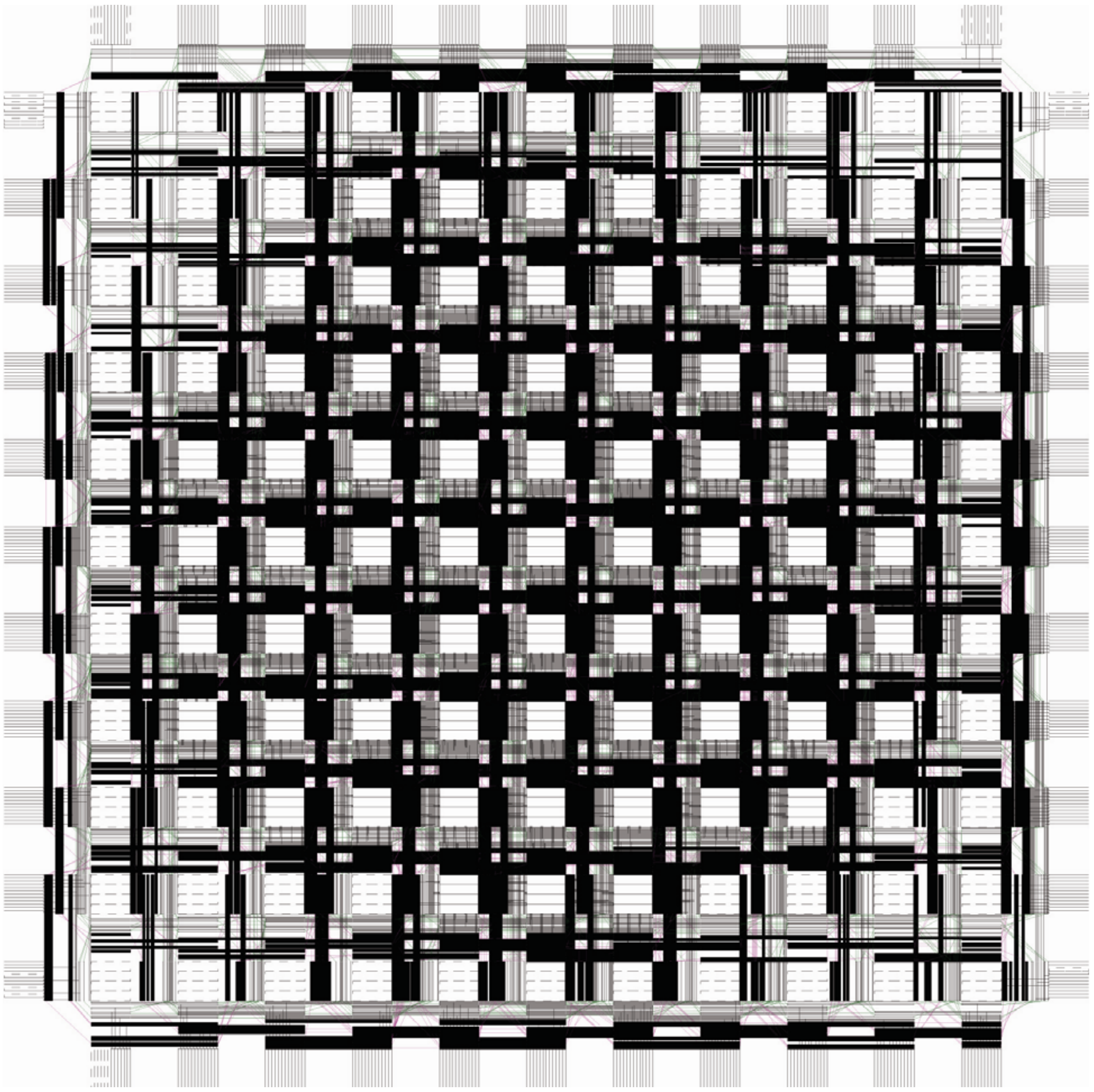


Fig. B.8. Routed nets of circuit: mantissa_dp

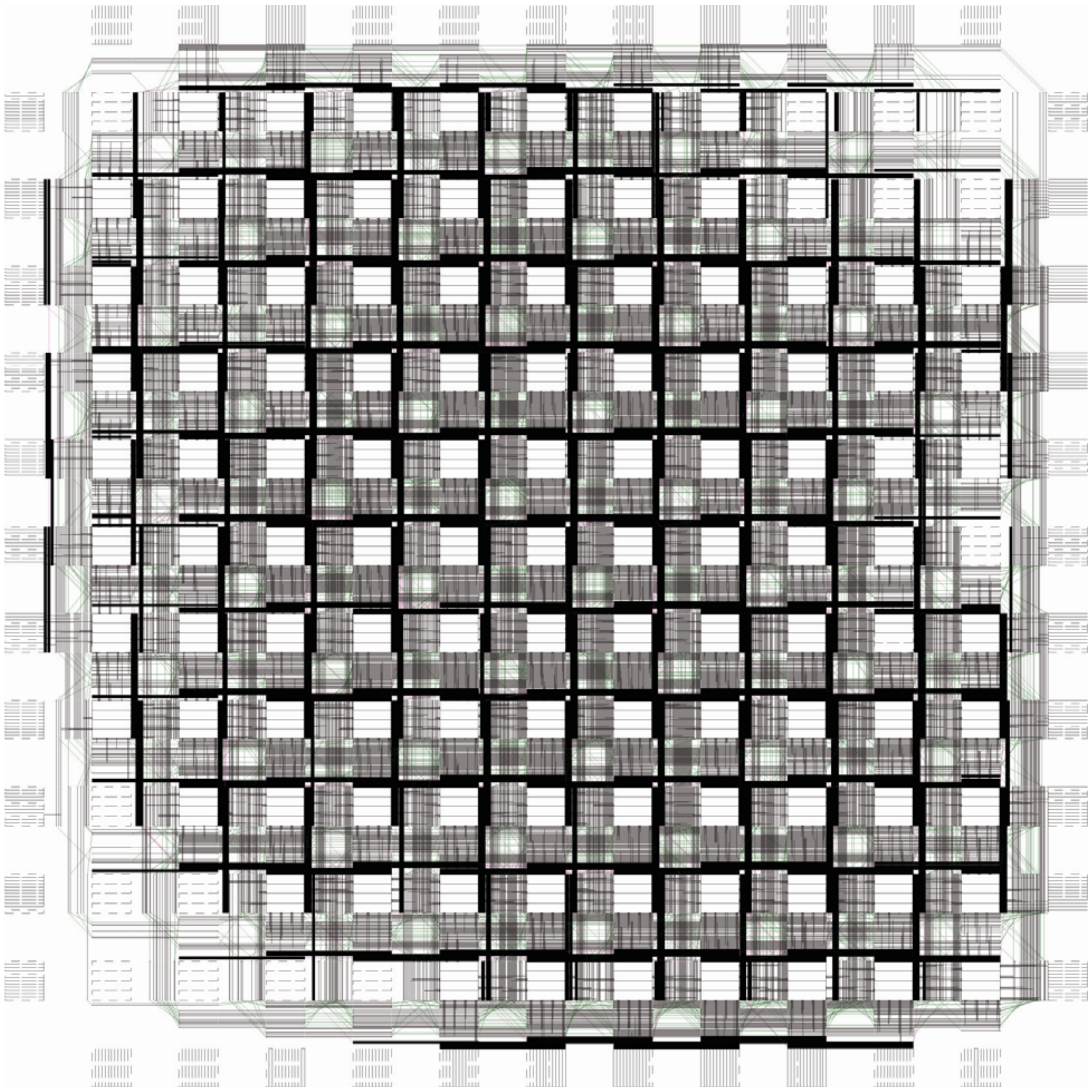


Fig. B.9. Routed nets of circuit: multmod_dp

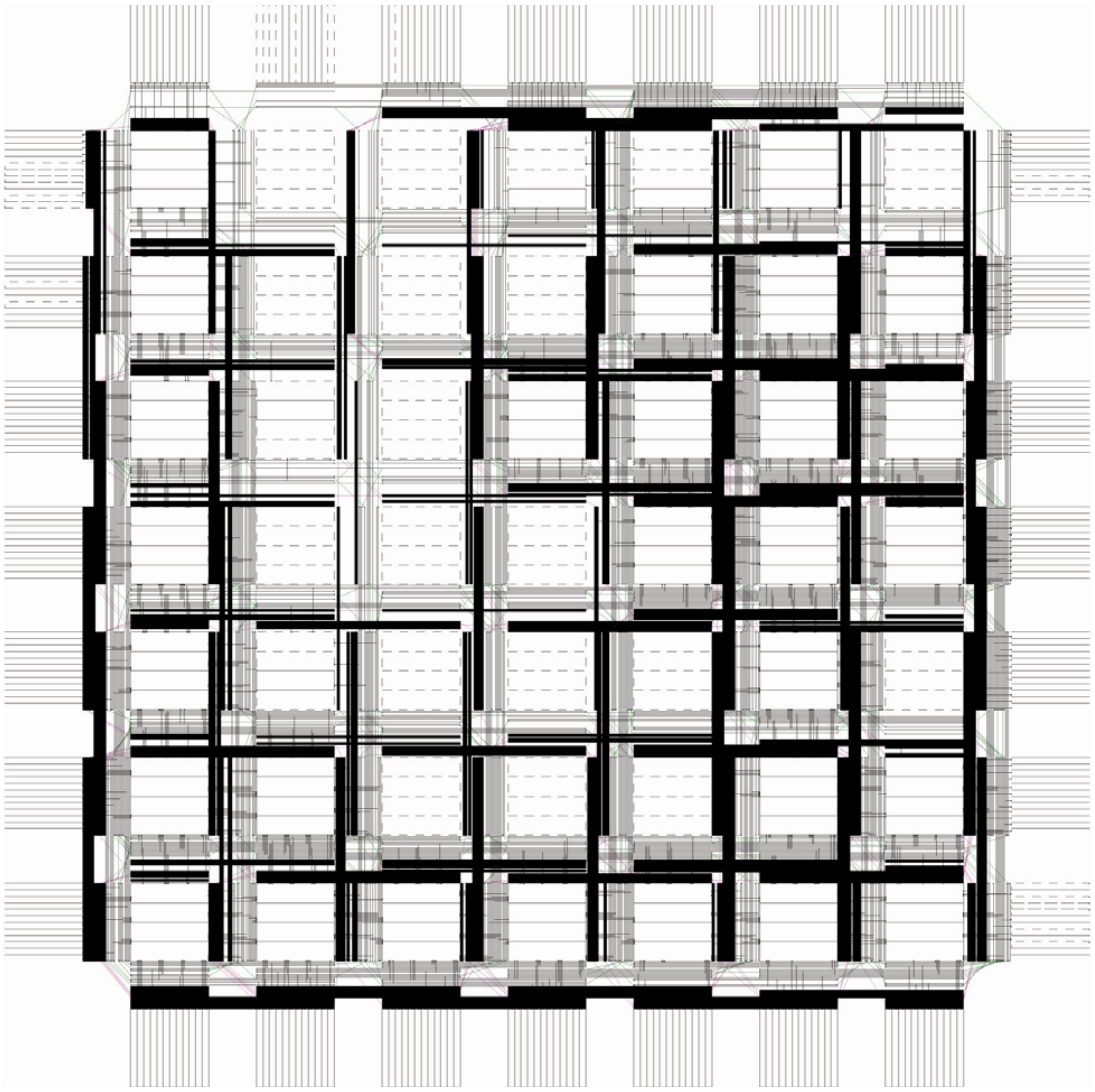


Fig. B.10. Routed nets of circuit: pipe_dpath

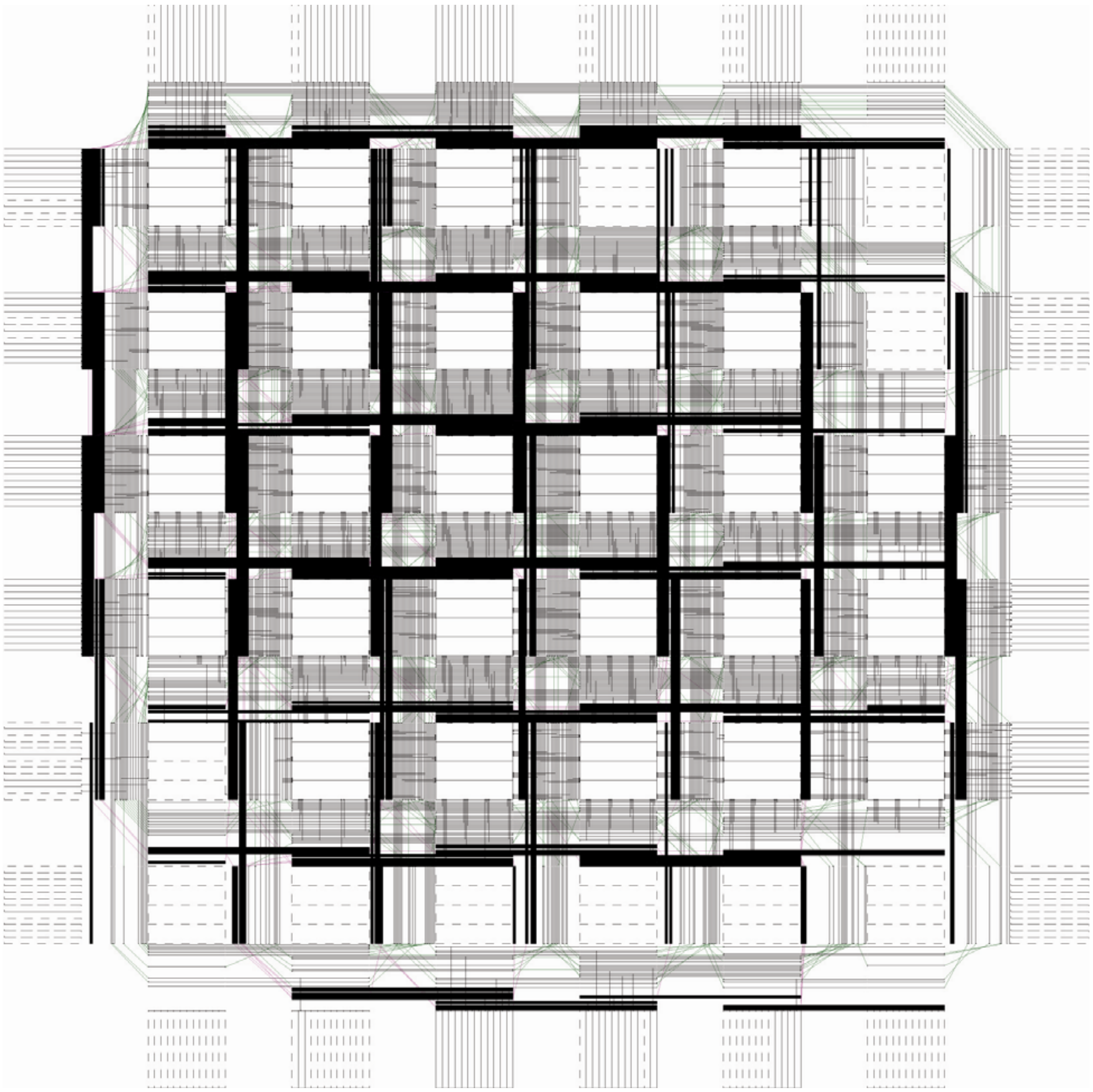


Fig. B.11. Routed nets of circuit: prils_dp

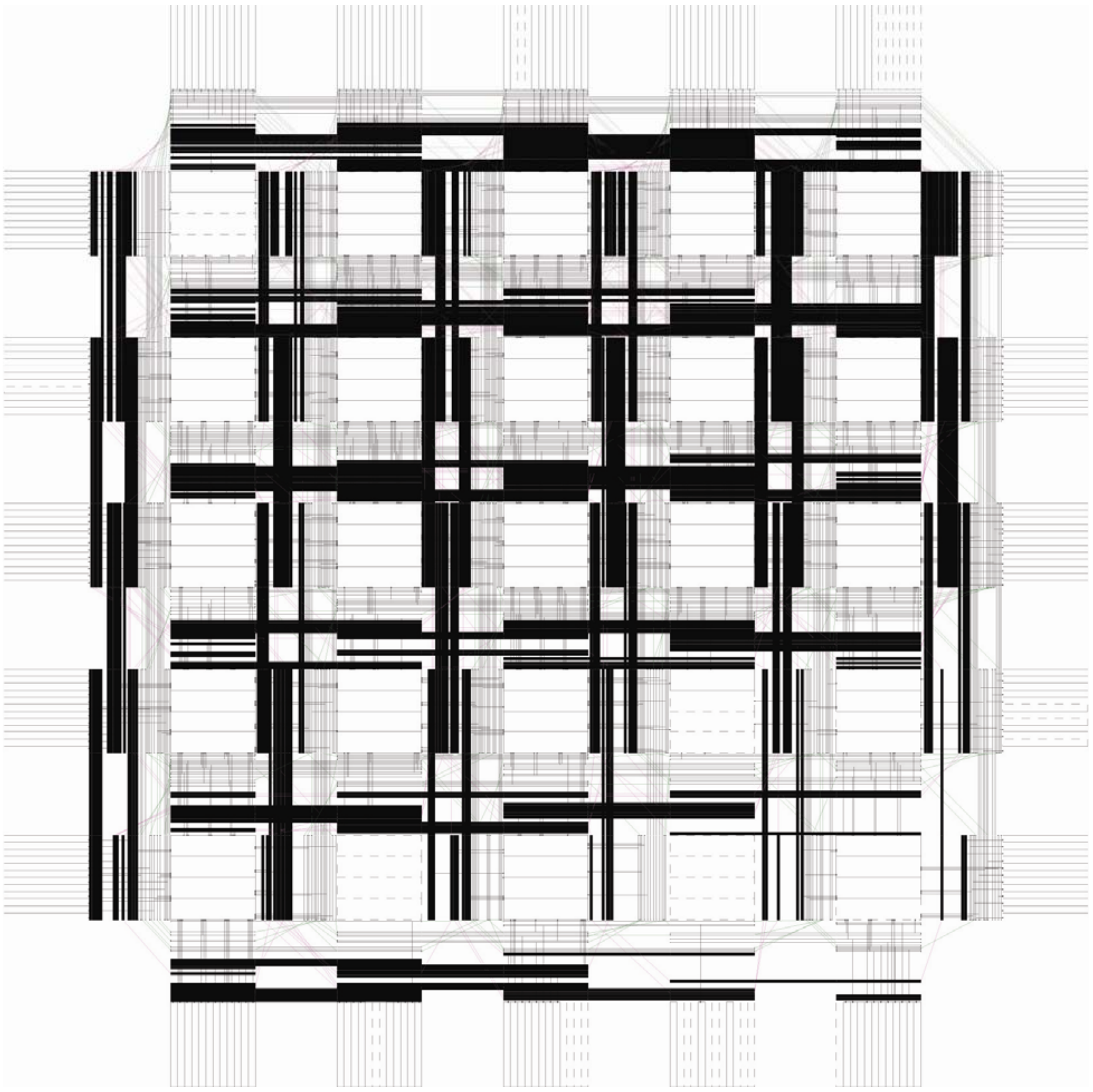


Fig. B.12. Routed nets of circuit: rsadd_dp

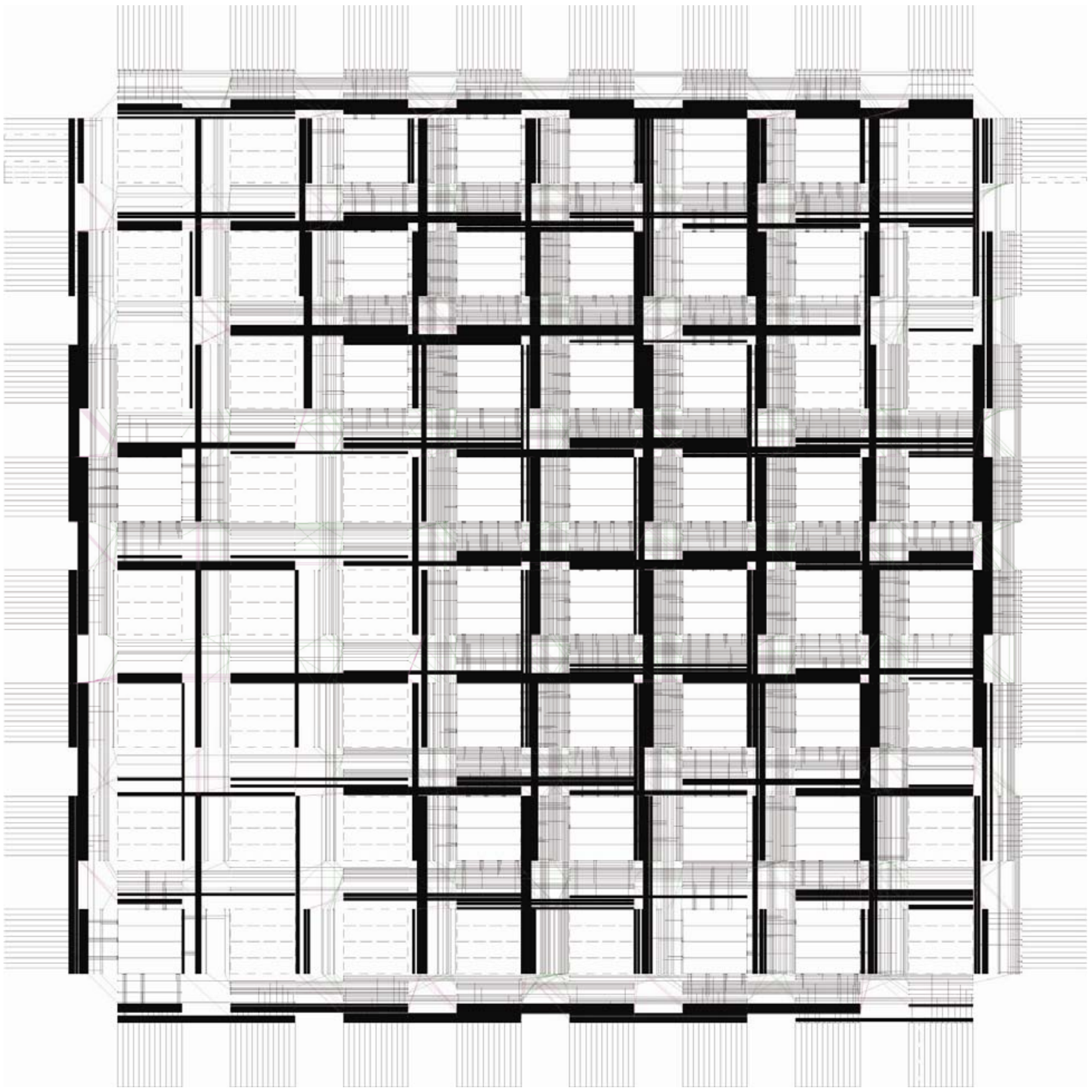


Fig. B.13. Routed nets of circuit: smu_dpath

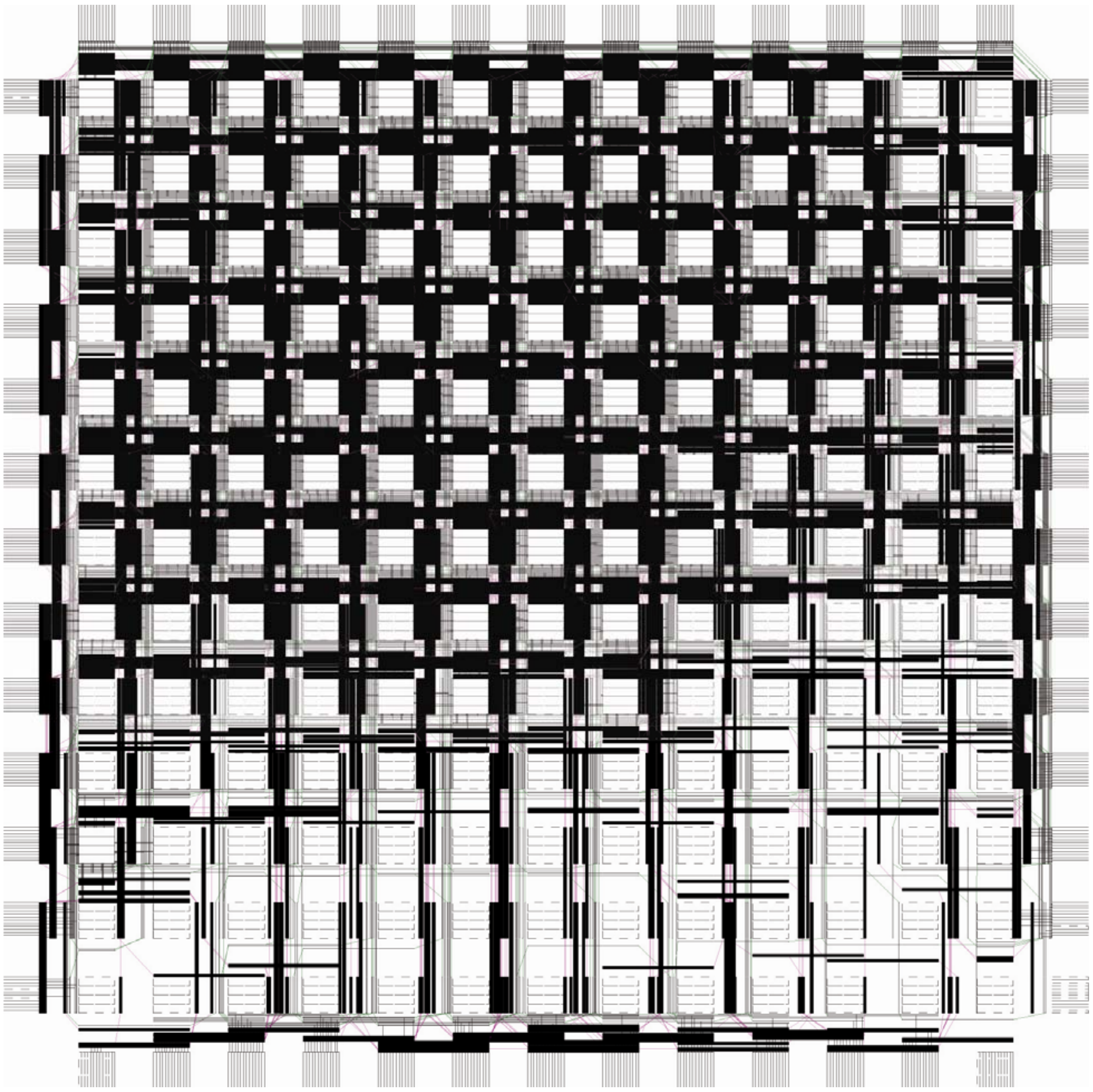


Fig. B.14. Routed nets of circuit: ucode_dat

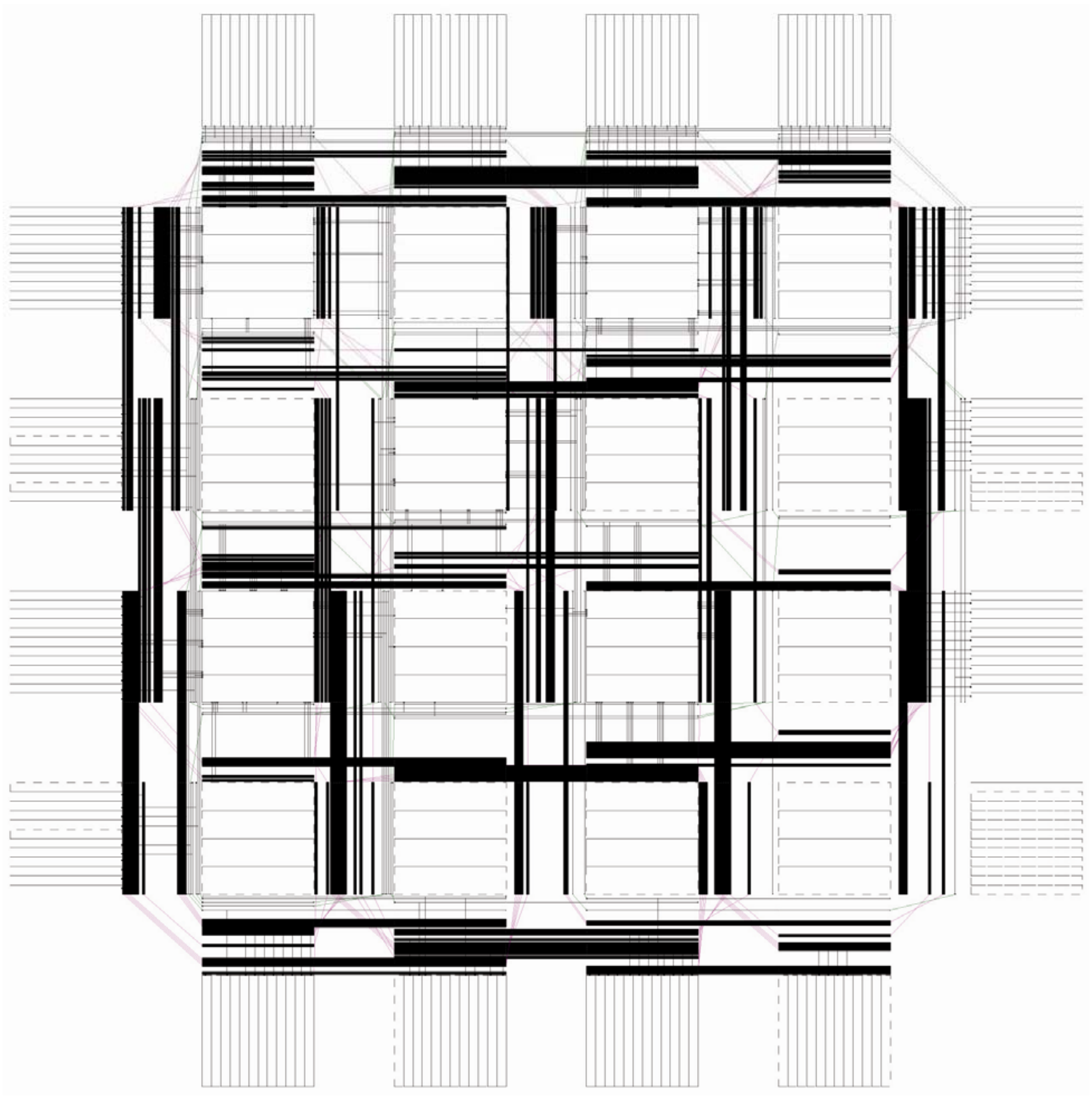


Fig. B.15. Routed nets of circuit: ucode_reg

References

[1] *“Altera Documentation Library,”* Altera Corporation, San Jose, CA, 2004 [Online]. Available: http://www.altera.com

[2] *“Xilinx UG190 Virtex-5 FPGA User Guide*,” Xilinx Inc., San Jose, CA, 2004 [Online]. Available: www.xilinx.com/support/documentation/user_guides/ug190.pdf

[3] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "Architecture of Field-Programmable Gate Arrays," *Proc. IEEE,* vol. 81, pp. 1013-1029, Jul. 1993.

[4] D. Lewis *et al.*, "The Stratix II logic and routing architecture," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays,* 2005, pp. 14-20.

[5] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and single-driver wires in FPGA interconnect," in *Proc. Int. Conf. Field-Programmable Tech.,* 2004, pp. 41-48.

[6] A. Ye, "Field-Programmable Gate Array Architecture and Algorithms Optimized for Implementing Datapath Circuits," Ph.D. dissertation, Univ. of Toronto, Toronto, ON, Canada, 2004.

[7] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep Submicron FPGAs*. Norwell, MA: Kluwer, 1999.

[8] J. Luu *et al.*, "VPR 5.0: FPGA cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2009, pp. 133-142.

[9] A. Ye, J. Rose, and D. Lewis, "Synthesizing datapath circuits for FPGAs with emphasis on area minimization," in *Proc. Int. Conf. Field-Programmable Tech.*, 2002, pp. 219 - 226.

[10] A. Ye and J. Rose, "Using multi-bit logic blocks and automated packing to improve field-programmable gate array density for implementing datapath circuits," in *Proc. Int. Conf. Field-Programmable Tech.*, 2004, pp. 129-136.

[11] C. Ebeling, D.C. Cronquist, and P. Franklin, "RaPiD - Reconfigurable Pipelined Datapath," in *Proc. Int. Workshop Field-Programmable Logic Appl.*, 1996, pp. 126-135.

[12] D.C. Chen and J.M. Rabaey, "A reconfigurable multiprocessor IC for rapid prototyping of algorithmic-specific high-speed DSP data paths," *IEEE J. of Solid-State Circuits*, vol. 27, pp. 1895-1904, Dec. 1992.

[13] K. Leijten-Nowak and J. van Meerbergen, "An FPGA architecture with enhanced datapath functionality," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2003, pp. 195-204.

[14] D. Lewis and D. Cherepacha, "DP-FPGA: An FPGA Architecture Optimized for Datapaths," *J. VLSI Des.*, vol. 4, pp. 329-343, 1996.

[15] A. Marshall *et al*., "A reconfigurable arithmetic array for multimedia applications," in *Proc. ACM/SIGDA Int. Symp. Field Programmable Gate Arrays*, 1999, pp. 135-143.

[16] *"Xilinx Data Sheets,"* Xilinx Inc., San Jose, CA, 2004 [Online]. Avaliable: http://www.xilinx.com

[17] J. Rose, R.J. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: the effect of logic block functionality on area efficiency," *IEEE J. Solid-State Circuits*, vol. 25, pp. 1217-1225, Oct. 1990.

[18] S. Brown, M. Khellah, and Z. Vranesic, "Minimizing FPGA Interconnect Delays in Array-Based FPGAs," in *CICC*, 1994, pp. 181-184.

[19] H. Hseih *et al*., "Third-generation architecture boosts speed and density of field-programmable gate arrays," in *Proc. IEEE Custom Integrated Circuits Conf.*, 1990, pp. 31.2/1 - 31.2/7.

[20] S. Wilton, "Architecture and Algorithms for Field-Programmable Gate Arrays with Embedded Memories," University of Toronto, Ph.D. Dissertation 1997.

[21] A. Ye and J. Rose, "Using bus-based connections to improve field-programmable gate-array density for implementing datapath circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Sys.*, vol. 14, pp. 462-473, May 2006.

[22] K. Chen, J. Cong, Y. Ding, A. B. Kahng, and P. Trajmar, "DAG-Map: Graph-Based FPGA Technology Mapping for Delay Optimization," *Design Test of Comput., IEEE*, pp. 7 -20, Sept. 1992.

[23] R. J. Francis, J. Rose, and K. Chung, "Chortle: a technology mapping program for lookup table-based field programmable gate arrays," in *Proc. 27th Design Automation Conf.* , 1990, pp. 613 -619.

[24] S. Cho, S. Chatterjee, A. Mishchenko, and R. Brayton, "Efficient FPGA Mapping using priority cuts," in *Proc. FPGA ‘07*, 2007.

[25] P. Jamieson and J. Rose, "A Verilog RTL synthesis tool for heterogeneous FPGAs," in *IEEE Field Programmable Logic and Applications*, 2005, pp. 305-310.

[26] Berkeley Logic Synthesis and Verification Group. (2005). *ABC: A System for Sequential Synthesis and Verification* [Online]. Avaliable: http://www.eecs.berkeley.edu/~alanmi/abc/

[27] A. Marquardt, V. Betz, and J. Rose, "Using Cluster-Based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density," in *ACM/SIGDA Int. Symp. FPGAs*, 1999, pp. 37-46.

[28] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi, "Optimization by Simulated Annealing," in *Science*, vol. 220, pp. 671-680, 1983.

[29] C. Ebeling, L. McMurchie, S.A. Hauck, and S. Burns, "Placement and Routing Tools for the Triptych FPGA," in *IEEE Trans. VLSI Syst.*, pp. 473-482, Dec. 1995.

[30] C.Y. Lee, "An Algorithm for Path Connections and its Applications," *IRE Trans. on Electron. Comput.*, vol. EC-10, pp. 346-365, 1961.

[31] A. Ye and J. Rose, "Measuring and utilizing the correlation between signal connectivity and signal positioning for FPGAs containing multi-bit building blocks," in *Proc. Int. Con. Field Programmable Logic and Applications,* 2005, pp. 159-166.

[32] I. Kuon and J. Rose. (2008, February). *iFAR – intelligent FPGA Architecture Repository* [Online]. Available: http://www.eecg.utoronto.ca/vpr/architectures/

[33] P. P. Chen and A. Ye, "The Effect of Multi-Bit Correlation on the Design of Field-Programmable Gate Array Routing Resources," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 99, p. 1, October 2009.

[34] I. Kuon and J. Rose, "Area and delay trade-offs in the circuit and architecture design of FPGAs," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2008, pp. 149-158.

[35] W. Zhao and Y. Cao, "New Generation of Predictive Technology Model for Sub-45nm Design Exploration," *IEEE Trans. on Electron. Devices*, vol. 53, pp. 585-590, Nov. 2006.

[36] V. Betz and J. Rose, "How much logic should go in an FPGA logic block," *IEEE Des. Test of Comput. Mag.*, vol. 15, pp. 10 -15, 1998.

[37] E. Ahmed and J. Rose, "The Effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Trans. VLSI Syst.*, vol. 12, pp. 288 - 298, Mar. 2004.

[38] Pico-Java Processor Design Documentation, Sun Microsystems, 1999.

[39] G. Lemieux and D. Lewis, "Circuit design of routing switches," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2002, pp. 19-28.

[40] G. Lemieux and S. Brown, "A detailed router for allocating wire segments in field-programmable gate arrays," in *Proc. ACM/SIGDA Physical Design Workshop*, 1993, pp. 215-226.