EXPLORATION OF ENERGY AND AREA EFFICIENT TECHNIQUES FOR COARSE-GRAINED RECONFIGURABLE FABRICS

Anil Yadav

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

December 2011

APPROVED:

Gayatri Mehta, Major Professor
Kamesh Namuduri, Committee Member
Mahadevan Gomathisankaran, Committee Member
Murali Varanasi, Chair of the Department of Electrical Engineering
Costas Tsatsoulis, Dean of College of Engineering
James D. Meernik, Acting Dean of the Toulouse Graduate School

Yadav, Anil. Exploration of Energy and Area Efficient Techniques for Coarse-Grained Reconfigurable Fabrics. Master of Science (Electrical Engineering), December 2011, 68 pages, 32 tables, 41 figures, references, 33 titles.

Coarse-grained fabrics are comprised of multi-bit configurable logic blocks and configurable interconnect. This work is focused on area and energy optimization techniques for coarse-grained reconfigurable fabric architectures. In this work, a variety of design techniques have been explored to improve the utilization of computational resources and increase energy savings. This includes splitting, folding, multi-level vertical interconnect. In addition to this, I have also studied fully connected homogeneous and heterogeneous architectures, and 3D architecture. I have also examined some of the hybrid strategies of computation unit's arrangements. In order to perform energy and area analysis, I selected a set of signal and image processing benchmarks from MediaBench suite. I implemented various fabric architectures on 90nm ASIC process from Synopsys. Results show area improvement with energy savings as compared to baseline architecture.

ACKNOWLEDGMENTS

This thesis would have remained a dream without the support of my mentor and graduate advisor Dr. Gayatri Mehta who had been a source of constant guidance and encouragement from the very early stage of this research and also for the extraordinary experiences throughout the work I have done under her supervision. Her scientific intuition has made her as an oasis of intuitive ideas and passion in science, which have also inspired and enriched my growth as a student, and as a researcher. Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Kamesh Namuduri, and Dr. Mahadevan Gomathisankaran for their encouragement, insightful comments, and hard questions.

My deepest gratitude goes to my parents (Mr. Omvir yadav and Mrs. Ramgiri Yadav) and my four lovely sisters (Anita, Kavita, Manju, Sarita) for their unflagging love and support throughout my life. I am indebted to my father for supporting me through my education overseas. I would also like to express my appreciation to the Department of Electrical Engineering and its staff for assisting me with the technical labs where I could work anytime during my research study. My thanks extend to my invaluable network of supportive, forgiving, generous, cheerful and loving friends without whom I could not have survived the process. I also express my appreciation to my girlfriend Akemi Kaushik whose dedication, unconditional love and persistent confidence in me, has taken the load off my shoulder. I would also appreciate everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

Last but not the least, thanks to God for my life & through all tests in the past years, while I was staying away from my family in the Unites states. You have made my life more bountiful. May your name be exalted, honored, and glorified.

CONTENTS

ACKNOWLEDGMENTS iii

LIST OF TABLES vi

LIST OF FIGURES ix

CHAPTER 1. INTRODUCTION 1

1.1. Fine Grain Reconfigurable Architectures 2

1.2. Coarse Grain Reconfigurable Architectures 3

1.3. Contribution of the Thesis 5

1.4. Organization of the Thesis 6

CHAPTER 2. BACKGROUND 7

2.1. Related Work 7

2.2. Domain Specific Fabric Overview 8

2.2.1. Mapping of Applications Onto Domain-Specific Reconfigurable Fabric 9

2.2.2. Design Case Studies 10

2.2.3. Fabric Architecture with Dedicated Pass Gates 12

2.2.4. Fabric Architecture with Inputs Coming from Side (ICS) 13

CHAPTER 3. ARCHITECTURE EXPLORATION 15

3.1. Split/Fold Architecture 15

3.1.1. Motivation 15

3.1.2. Split Architecture 17

3.1.3. Fold Architecture 20

3.2. Architecture with Multi-Level Vertical Interconnects 23

3.3. Architecture with Horizontal Interconnect 25
3.4. Fully Connected Architectures 28
3.4.1. Homogeneous Architecture 30
3.4.2. Heterogeneous Architecture 31
3.5. 3D CGRA Architecture 32
CHAPTER 4. EXPERIMENTS AND RESULTS 36
4.1. Experimetal Setup 36
4.1.1. Evaluated Applications 36
4.1.2. Experimental Methodology 36
4.2. Results 37
4.2.1. Split and Fold Architectures 37
4.2.2. Architectures with Multi-level Vertical Interconnect and Horizontal Interconnect 47
4.2.3. Fully Connected Homogeneous and Heterogeneous Architectures 52
4.2.4. Three Dimensional Architectures 56
CHAPTER 5. CONCLUSION AND FUTURE WORK 60
5.1. Conclusion 60
5.2. Future Work 61
BIBLIOGRAPHY 64

LIST OF TABLES

2.1 Number of operations, constants, inputs, pass gates in the data flow graphs of the benchmarks 11

4.1 Fabric size (Width x Height) for mapping various benchmarks onto various standard, standard-split and standard-fold architectures using the heuristic mapper. 38

4.2 Fabric size (Width x Height) for mapping various benchmarks onto various ICS, ICS-split and ICS-fold architectures using the heuristic mapper. 39

4.3 Minimum fabric size (Width x Height) for various fabric architectures. 40

4.4 Comparison of the total number of functional units in standard fabric architectures vs standard-split & standard-fold fabric architectures. 40

4.5 Comparison of the total number of functional units in ICS fabric architectures vs ICS-split & ICS-fold fabric architectures. 41

4.6 Number of ALUs used as pass gates in various standard (std), std-split, std-fold, ICS, ICS-split, ICS-fold fabric architectures. 42

4.7 Percentage area savings per benchmark mapped onto standard, DP, std-split, std-fold and hybrid architectures. (Negative numbers mean cost). 43

4.8 Percentage area savings per benchmark mapped onto ICS, DP, ICS-split, ICS-fold and hybrid architectures. (Negative numbers mean cost). 43

4.9 Energy savings (%) per benchmark mapped onto standard, DP, std-split, std-fold and hybrid architectures. (Negative numbers mean cost). 44

4.10 Energy savings (%) per benchmark mapped onto ICS, DP, ICS-split, ICS-fold and hybrid architectures. (Negative numbers mean cost). 45

4.11 Percentage savings in terms of number of functional units per benchmark mapped onto standard, DP, std-split, std-fold and hybrid architectures. (Negative numbers mean cost). 46

4.12 Percentage savings in terms of number of functional units per benchmark mapped onto ICS, DP, ICS-split, ICS-fold and hybrid architectures. (Negative numbers mean cost). 47

4.13 Fabric size (Width x Height) for mapping various benchmarks onto various ICS architectures , architectures with multi-level VI and architectures with HI using the heuristic mapper. 48

4.14 Minimum Fabric size (Width x Height) for various fabric architectures. 48

4.15 Comparison of the total number of functional units in ICS fabric architectures vs Architecture with multi-level vertical interconnect & Architecture with horizontal interconnect. 49

4.16 Number of ALUs used as pass gates in various ICS architectures, architecture with multi-level VI and architecture with HI. 49

4.17 Percentage area savings per benchmark mapped onto ICS, DPs, and hybrid architectures, architecture with multi-level VIs and HIs. (Negative numbers mean cost). 50

4.18 Energy savings (%) per benchmark mapped onto ICS, DPs, and hybrid architectures, architecture with multi-level VI and HI. (Negative numbers mean cost). 50

4.19 Percentage savings in terms of number of functional units per benchmark mapped onto ICS, DPs, and hybrid architectures, architecture with multi-level VIs and HIs. (Negative numbers mean cost). 51

4.20 Fabric size (Width x Height) for mapping various benchmarks onto various ICS architectures, fully connected homogeneous and heterogeneous architectures using the heuristic mapper. 52

4.21 Minimum Fabric size (Width x Height) for various fabric architectures. 52

4.22 Comparison of the total number of functional units in various ICS architectures, fully connected homogeneous and heterogeneous architectures. 53

4.23 Percentage area savings per benchmark mapped onto various ICS architectures, fully connected homogeneous and heterogeneous architectures. (Negative numbers mean cost). 54

4.24 Energy savings (%) per benchmark mapped onto various ICS architectures, fully connected homogeneous and heterogeneous architectures. (Negative numbers mean cost). 54

4.25 Percentage savings in terms of number of functional units per benchmark mapped onto various ICS architectures fully connected homogeneous and heterogeneous architectures. (Negative numbers mean cost). 55

4.26 Fabric size (Width x Height) for mapping various benchmarks onto various ICS architectures, three dimensional architecture using the heuristic mapper. 56

4.27 Minimum Fabric size (Width x Height) for various fabric architectures. 56

4.28 Comparison of the total number of functional units in various ICS architectures, three dimensional architectures. 57

4.29 Percentage area savings per benchmark mapped onto various ICS architectures, three dimensional architecture. (Negative numbers mean cost). 57

4.30 Energy savings (%) per benchmark mapped onto various ICS architectures, three dimensional architecture. (Negative numbers mean cost). 58

4.31 Percentage savings in terms of number of functional units per benchmark mapped onto various ICS architectures, three dimensional architecture. (Negative numbers mean cost). 58

LIST OF FIGURES

1.1 Flexibility vs energy tradeoff for ASICs, FPGAs and CGRAs. 2

1.2 Example of fine grain reconfigurable architecture (FPGA). 3

1.3 Example of coarse grain reconfigurable architecture (CGRA). 4

2.1 The fabric model is comprised of ALUs and a reconfigurable interconnects. 8

2.2 An example of a data flow graph (DFG). 10

2.3 Example mapping of the DFG in Figure 2.2 onto a stripe-based coarse-grained fabric. 11

2.4 A DFG shown in Figure 2.2 mapped on the architecture with 33% dedicated pass gates. 12

2.5 Multiplexer-based interconnector used for inputs coming from side (ICS) concept. 13

2.6 A DFG shown in Figure 2.2 mapped on the architecture with ICS. 13

3.1 A set of subfigures to demonstrate the implementation of diamond data flow on split and fold architecture. 15

3.2 A set of subfigures to demonstrate the implementation of wedge data flow on split and fold architecture. 16

3.3 Base CGRA architectures and CGRA architecture after split . 17

3.4 A set of subfigures to demonstrate the steps involved in getting an ICS-split architecture. 18

3.5 Schematic for 8:1 interconnect used in split architecture, fold achitecture, and architecture with horizontal interconnect. 18

3.6 A DFG shown in Figure 2.2 mapped on the ICS-split architecture. 19

3.7 A DFG shown in Figure 2.2 mapped on the ICS-split architecture with 33% dedicated pass gates. 20

3.8 Fabric model: Horizontal line show the point from where fabric is folded. 21

3.9 Final folded fabric model. 21

3.10 Base CGRA architectures and CGRA architecture after fold. 21

3.11 A DFG shown in Figure 2.2 mapped on the ICS-fold architecture. 22

3.12 A DFG shown in Figure 2.2 mapped on the ICS-fold architecture with 33% dedicated pass gates. 22

3.13 A simple example of computation. 23

3.14 Implementation of Figure 3.13 with vertical step connection. 23

3.15 Schematic for 5:1 Interconnect build using 4:1 multiplexers. 24

3.16 A fabric model comprises of ALUs with multi-level vertical interconnect. 25

3.17 A detailed version of multi-level vertical interconnect. 25

3.18 A DFG shown in Figure 2.2 mapped on the architecture with multi-level vertical interconnect. 26

3.19 A fabric model comprises of ALUs with horizontal interconnect. 27

3.20 A fabric model showing horizontal interconnect concept. 27

3.21 A DFG shown in Figure 2.2 mapped on the architecture with horizontal interconnect. 28

3.22 Fully connected architecture shown with its interconnects. 29

3.23 A DFG shown in Figure 2.2 mapped on the fully connected homogeneous architecture. 31

3.24 An example showing the stacking of 4x4 fabrics on top of each other. 32

3.25 Three Dimensional Fabric Architecture shown with various interconnects. 33

3.26 A set figures showing the placement of 4:1 interconnectors and the data main data flow in 3D architecrure. 34

3.27 A set of figure showing the various diagonals with hopping interconnection in the proposed 3D architecture. 34

3.28 A DFG shown in Figure 2.2 mapped on the fully connected homogeneous architecture. 35

4.1 Experimental flow. 37

5.1 Area consumption for various fabric architectures implemented on synopsys 90nm generic library. 61

5.2 Energy consumption for various fabric architectures implemented on synopsys 90nm generic library. 62

5.3 Energy Vs Area graph for various architectures as compare to ICS-No-DPs architecture. 63

CHAPTER 1

INTRODUCTION

The main concerns of modern day's computationally intensive applications such as image processing and recognition, streaming video, and signal processing are power and area consumed by the processing units. For these kind of applications, we require custom hardware which should be faster, more flexible, energy-efficient and requires less area. Computing architectures which are widely used can be divided into three categories: general-purpose processors (GPPs), application-specific integrated circuits (ASICs) and reconfigurable architectures (RAs). Unfortunately when speed is required, GPP cannot be used for complex applications. The application-specific integrated circuit is designed for a specific application or task, which is faster and energy-efficient, but it limits the flexibility; on the other hand RAs (FPGAs) are flexible, but require more energy as compared to ASICs. Since reconfigurable architectures provide the good flexibility, they are getting more attention. The granularity of the reconfigurable architectures is defined as the size on the smallest functional units. Based on granularity, reconfigurable architectures can be divided into two categories: fine-grained reconfigurable architectures (FPGAs) and coarse-grained reconfigurable architectures (CGRAs). Fine-grained architectures work at bit-level and that is why they can be reconfigured at the bit-level, whereas coarse-grained architectures work at multiple-bit datapaths that give the flexibility to reconfigure the whole processing units of the device.

Due to inflexibility of AISCs and poor energy-efficiency of FPGAs, CGRAs have been getting attention in computing world from a decade. Figure 1.1 depicts the tradeoff between energy and flexibility of various computing architectures. CGRA bridges the gap between FPGAs and ASICs, as they are promising good energy-efficiency and high flexibility.

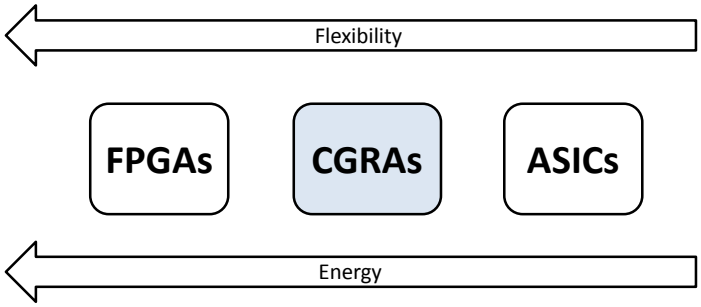


Figure 1.1: Flexibility vs energy tradeoff for ASICs, FPGAs and CGRAs.

1.1. Fine Grain Reconfigurable Architectures

Most of the FPGA architectures are known as fine-grain reconfigurable architectures. FPGAs have high granularity, which is also known as fine-grain. Fine-grain reconfigurable architectures are made up of configurable logic blocks (CLBs), programmable interconnectors for routing, and programmable input/output ports. Each CLB is made up of one or more than one k-input lookup tables which are interconnected through fast local interconnectors. Each k-LUT can implement any function with a single output and with k-inputs or less. Figure 1.2 shows a conventional island-style FPGA architecture, which is a widely accepted architecture model. The CLBs are placed in arrays which are surrounded by horizontal and vertical interconnectors. Both the interconnectors are connected through a programmable *switch block* to route the data between CLB-CLB, CLB-I/O. This flexibility of routing allows every CLB to reach every other CLB and I/O port. The main advantage of FPGAs over ASICs is that circuits and all the functions are fixed in the chip after fabrication and they cannot be altered again, whereas in FPGAs base gates and base cells are fixed, but the connections between them are programmable (not fixed), which helps to realize several required circuits while maintaining the high level of flexibility.

However, FPGAs have inherent disadvantages:

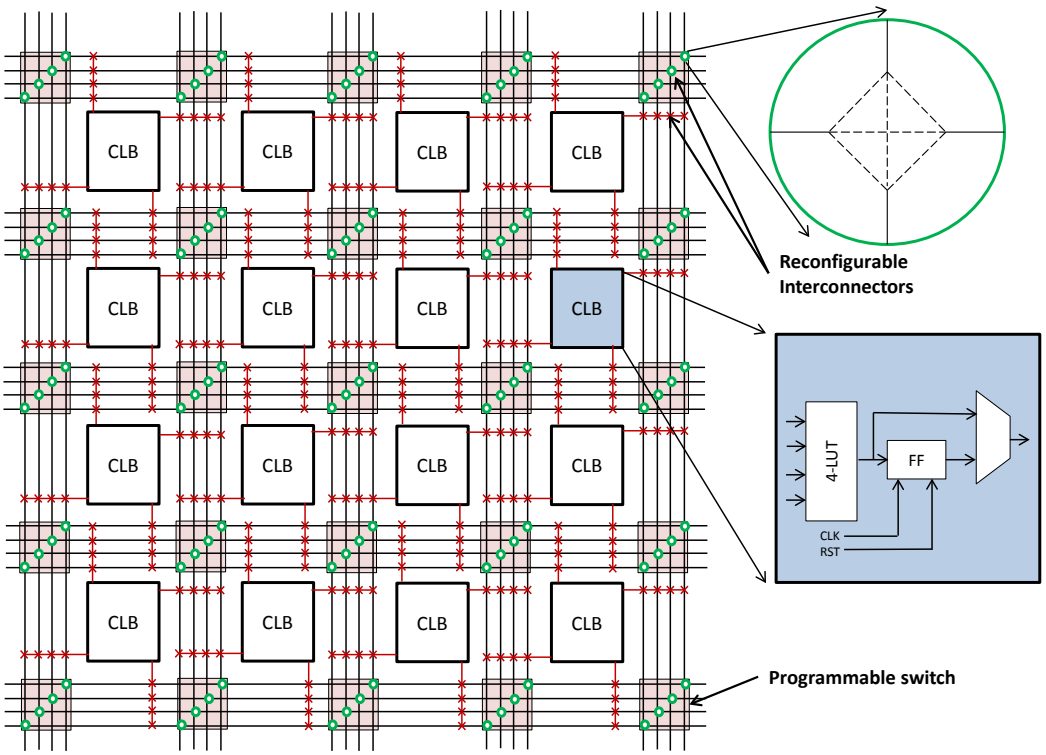


Figure 1.2: Example of fine grain reconfigurable architecture (FPGA).

Logic Granularity: Basically FPGAs are designed for logic replacement. As a result, it is inefficient to perform multimedia computations and complex signal processing.

Routability: Because of bit-level operations, several functional units are needed to operate wide datapaths. This includes huge routing overhead and poor routability [2], [7]. In addition, introduced switches used to connect the wires will take both area and power.

Configuration Time: With fine granularity, the high volume of configuration data is needed for the fabric. The time to load such a high volume of configuration data ranges from hundreds of microsecond to hundreds of milliseconds.

1.2. Coarse Grain Reconfigurable Architectures

Coarse-grain reconfigurable architectures (CGRAs) try to overcome the disadvantages of FPGAs-based computing, which was discussed in Section 1.1. CGRAs are capable of

implementing high-level operators in their processing units, which can operate at a multiple-bit width datapath [26], [8], [1], [3]. Since interconnectors used in CGRAs are mainly buses, less configuration data is needed for CGRAs and that solves the long configuration time problem of fine-grain reconfigurable architectures. Communication resources in CGRAs have buses for the interconnections that are low in number but have a high granularity of communication lines, which is an advantage over fine-grain architectures.

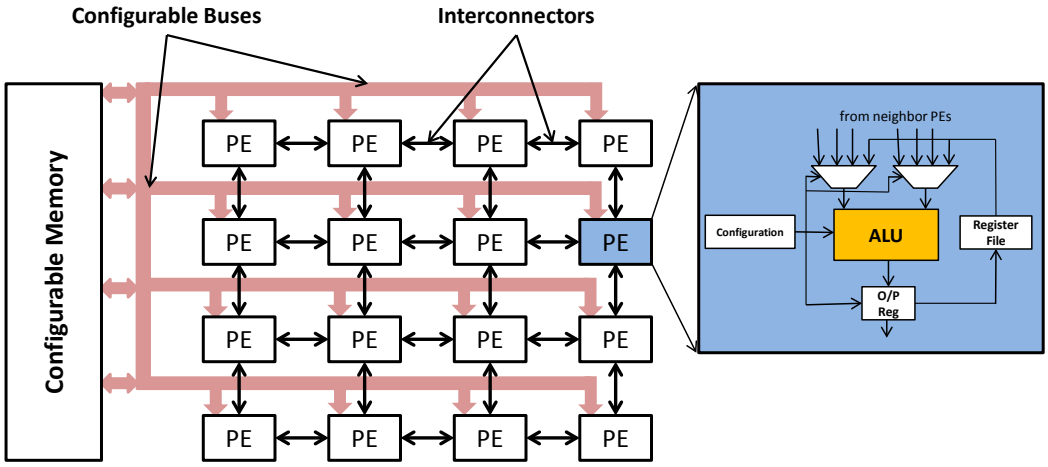


Figure 1.3: Example of coarse grain reconfigurable architecture (CGRA).

The designing and specifications of CGRAs is based on target application domain. In CGRA, processing elements (PEs) are placed in arrays, which are connected to each other with flexible topology. Each PE is made up of an arithmetic logic unit (ALU) that is capable of execute arithmatic operations (i.e. addition, multiplication, etc.) and logic operations (i.e. shift, etc.) or load/store. Each PE also has a register file to store the data or to load the data. Since CGRAs have more computational resources, thay provide high performance. Configurable buses connect the PEs to communication resources and the configurable memory. Figure 1.3 shows a basic CGRA architecture in which PEs are connected by a mesh-like interconnector. Many architectures have been proposed and developed both in academia and industry during the last decade such as MATRIX [1], Garp [4], MorphoSys [6], [5], RaPiD [8], [10], PipeRench [11], [13], [14], HFPGA [27], Kilocore [30], CFPA [28], Montium [22],[18]

ADRES [9], SmartCell [24] [20]. These architectures have sequential structure and use local registers or shared register files for storing data values.

1.3. Contribution of the Thesis

This work focused on area and energy optimization techniques for coarse grained reconfigurable fabric architectures. In this work, a variety of design techniques have being explored to improve the utilization of computational resources and increase energy savings. My studies are based on design domain specific fabrics which are developed with the goals of reducing power consumption and area optimization. Stripe-based fabric in particular are quite promising due to their good fit to a data flow graph structure [16, 23, 21, 15] but I have observed that the data flow graphs of the application domain have some common graphs which are narrower from one end and wide on other end, resulting in inefficient resource utilization. The main contribution of this thesis is to explore new fabric optimization techniques for coarse-grained fabrics in which ALUs of the fabric can be utilized more efficiently. This includes splitting, folding, multi-level vertical interconnect. In addition to this, I also studied fully connected homogeneous and heterogeneous architectures, and 3D architecture. In split architecture, instead of using one big fabric I used two smaller fabrics of different sizes which work together to perform computation. And in fold architecture, I reduced the total height of the fabric by accommodating working ALUs of the bottom rows in the idle locations of the top row. Hence the results of splitting and folding the fabric show reduction in power as well as area consumption. I also examined some of the hybrid strategies of computation unit's arrangements. To reduce the ALUs used as pass-gate I introduced multi-level vertical interconnect which routes the data more efficiently in vertical direction.

In addition to stripe based fabric architectures, I also studied non-stripe based fabric architectures with horizontal interconnect, fully connected homogeneous and heterogeneous architectures, and three dimensional (3D) architecture. These architectures have being explored to improve resource utilization. One of the interesting features of 3D architectures is that you can stack a layer of computational units on top of a two dimensional (2D) array.

This kind of arrangement gives you a dense packing of computational units. By using 3D architecture the number of neighbors is increased, and each computational unit can access large number of nearest neighbor computational units with shorter wire length.

1.4. Organization of the Thesis

The remainder of this thesis is organized as follows: Chapter 2 provides some background material in the area of reconfigurable computing and coarse-grain architectures in general. Then an overview on the domain-specific fabric is provided. After that application mapping onto domain-specific and their related results are discussed. Chapter 3 explores the various area and energu efficient techniques for coarse-grained reconfigurable fabrics. Chapter 4 includes an experimental setup, results and an analysis of area and energy consumption for a suite of image and signal proceesing benchmarks. Chapter 5 discusses the conclusions and future work.

CHAPTER 2

BACKGROUND

2.1. Related Work

Coarse-grained fabrics are comprised of multi-bit configurable logic blocks and multi-bit configurable interconnector. Many coarse grain reconfigurable architectures such as MATRIX [1], Garp [4], MorphoSys [6], [5], RaPiD [8], [10], PipeRench [11], [13], [14],HFPGA [27], Kilocore [30], CFPA [28], Montium [22],[18] ADRES [9], SmartCell [24] [20] have been proposed in recent years. Of these, PipeRench and Kilocore are stripe-based coarse grain fabrics. These fabrics use pass register files to manage constants and pass computed values from one stripe to another; [25] describes how to manage short-lived and long-lived values in coarse-grained fabrics. It discusses various architectural options for storing values when optimizing for area and energy. They consider constants as long-lived values and store them in register files. An application-specific optimization concept can also be used for CGRA architecture optimization[17], [29], [19], in which a large area consuming critical resources of process elements (PEs) can be shared whereas pipelining of the critical resources saves delay. [12] describes how fewer number of processing elements (PEs) with a enhanced interconnection lead to area and energy savings.

The domain-specific reconfigurable fabric (DSRF) considered here is also stripe-based, but it does not have storage elements. In previous research on this DSRF, the impact of varying different design parameters such as the width of the functional units, homogeneous vs. heterogeneous functional units, various functional unit implementation techniques, granularity of the interconnect, interconnect patterns, and horizontal and vertical routing onto physical characteristics like power, performance, and area has being studied [[16, 23, 21, 15]].

I observed that a large number of ALUs are still used for pass operation resulting in inefficient utilization of ALUs. There is a potential to make further improvement to increase the energy and area savings.

2.2. Domain Specific Fabric Overview

In stripe-based fabric, ALUs are organized into rows or *computational stripes* within which each functional unit operates independently and works in parallel. The results of these ALU operations are then fed into *interconnection stripes* constructed using multiplexers.

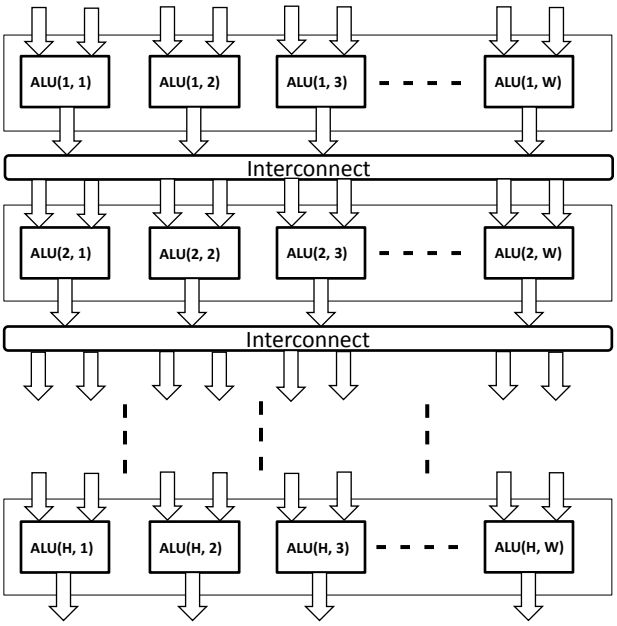


Figure 2.1: The fabric model is comprised of ALUs and a reconfigurable interconnects.

The fabric model was implemented in parameterized VHDL using the `generic` capability of the VHDL language. The fabric size is determined with the parameters specifying the width of the fabric W and height of the fabric H. W dictates the number of ALUs implemented in each computational stripe. H determines the number of computational stripes and $H - 1$ determines the number of interconnection stripes in the fabric model shown in Figure 2.1. To route the outputs more efficiently from different ALUs to the final output

port, several early exit rows are placed evenly in the device. For example, if fabric has height 18, each alternate row (i.e. 2, 4, 6, 8 ...18) is connected to early exit row. If the output is available in row 7, it will be passed to the next row (row 8), and from there it can be passed to final output row through early exit row. This technique saves a significant number of functional units in the fabric which is being used for passing the output till the end of the fabric.

2.2.1. Mapping of Applications Onto Domain-Specific Reconfigurable Fabric

The mapping of an application onto reconfigurable fabric means the assignment of different operations to different ALUs in the fabric in order to preserve the functionality of the application. Because of the layered structure of the fabric, ALUs are used as *pass-gates*, which take a single input and pass the input value to one or more ALUs of the next stripe. In general, not all of the available ALUs will be used for a single DFG mapping.

An example of DFG and its corresponding mapping on stripe-based reconfigurable fabric is shown in Figure 2.2 and Figure 2.3 respectively. The DFG shown in Figure 2.2 is implemented on our standard architecture where inputs and constants are routed from the top of the fabric. ALUs are shown in white colored squares with operators marked in them, ALUs used as pass gates are shown in blue color and labeled as "P" The inputs and outputs are shown in white colored ovals. Consider an ALU in row 14 and column 8 i.e. ALU(14,8), one of its inputs is a constant and is being routed all the way from the top of the fabric. It uses 13 ALUs for just passing this input to the desired location. A large number of hardware resources including computational units and interconnect get wasted in routing inputs from the top of the device.

This DFG has two outputs, one of which is computed and available early in the fabric (in row 12). Because of early exit rows in the fabric, this output can come out directly to the final output without using any ALUs in the successive stripes for the pass operation.

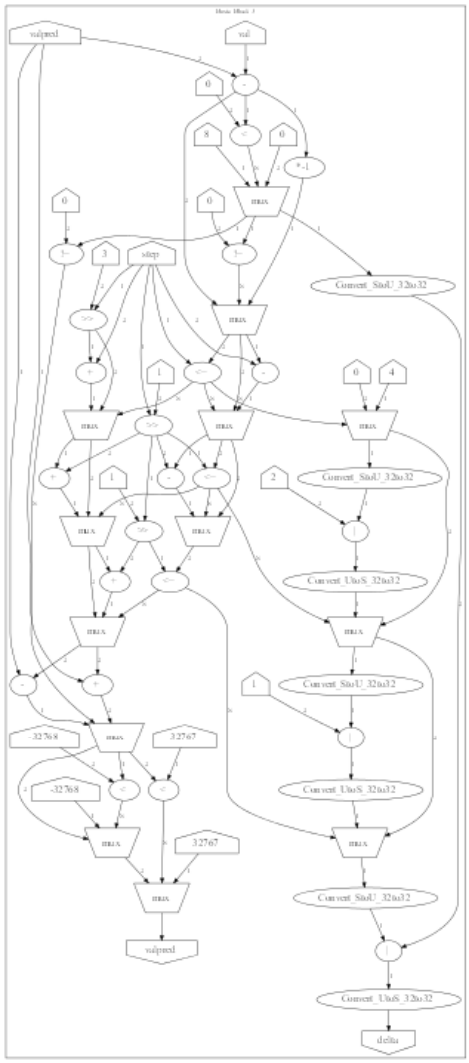


Figure 2.2: An example of a data flow graph (DFG).

2.2.2. Design Case Studies

In order to conduct architectural exploration case studies, I selected a set of core signal processing benchmarks from MediaBench benchmark suite including the ADPCM encoder (enc), ADPCM decoder (dec), GSM channel encoder (gsm), and the MPEG II decoder (row, col). I added the Sobel (sob) and Laplace (lap) edge detection algorithms to the benchmark suite I computed the number of operations and number of constants in each benchmark. Table 2.1 shows the number of operations and the number of constants contained in the

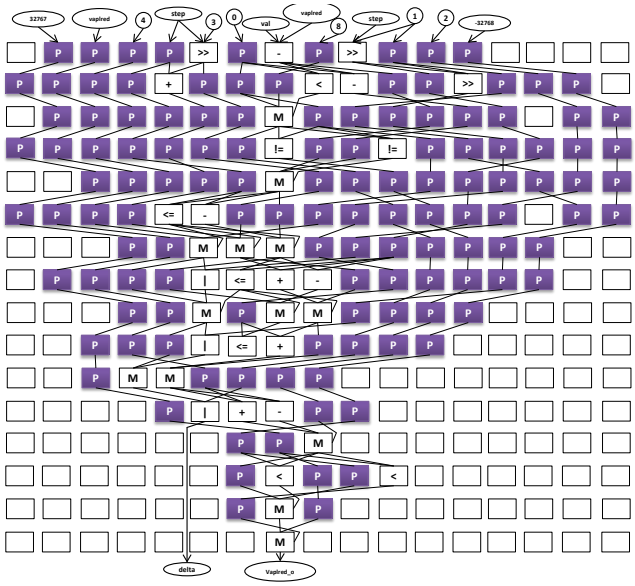


Figure 2.3: Example mapping of the DFG in Figure 2.2 onto a stripe-based coarse-grained fabric.

Table 2.1: Number of operations, constants, inputs, pass gates in the data flow graphs of the benchmarks

	enc	**dec**	**row**	**col**	**gsm**	**sob**	**lap**
Operations	36	29	52	61	29	24	29
Constants	14	20	23	32	20	10	5
Inputs	3	4	8	8	3	8	25
Pass gates	126	71	41	72	139	19	17

benchmark suite. Operations include only regular arithmetic, logic and shift operations such as addition, multiplication, AND, OR, right-shift, etc. It also shows the number of pass gates required to pass inputs and constants to the functional units where they are needed in the baseline architecture, as it can be seen that a large of functional units are being wasted for routing inputs and constants. For example, in "enc", 126 pass gates are used to route

only 3 inputs and 14 constants and some intermediate results. These numbers reflect the number of pass gates added to implement the data flow graphs on the reconfigurable fabric.

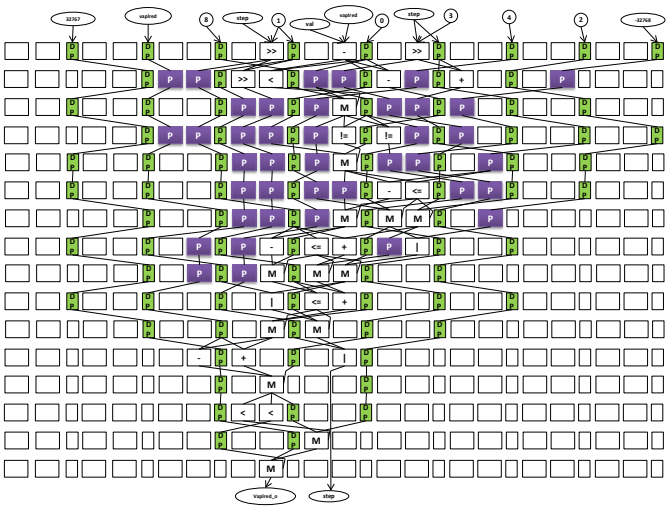


Figure 2.4: A DFG shown in Figure 2.2 mapped on the architecture with 33% dedicated pass gates.

2.2.3. Fabric Architecture with Dedicated Pass Gates

In our previous work, in order to reduce power consumption due to large numbers of ALUs being used as pass gates, the dedicated pass gates are introduced in the fabric, which simply route data vertically from one row to the next row [21]. The dedicated pass gate can also be set to idle state when not being used. For vertical routing, I varied the percentage of dedicated pass-gates at levels of 25% (1 out of 4), 33% (1 out of 3), and 50% (1 out of 2).

Figure 2.4 shows an example of a data flow graph (DFG) from Figure 2.2 mapped onto the architecture with 33% DPs. ALUs used as operators are shown in white colored squares with operators marked in them, ALUs used as pass gates are shown in blue color and labeled as "P", the dedicated pass gates are shown in green color and are labeled as "DP", and the white empty squares are idle. The goal of using dedicated pass-gates is to minimize the usage of ALUs for pass operations. As it can be seen that the number of ALUs used as pass

gates shown in blue color have been reduced from the baseline architecture but there are still many ALUs which are being used for pass operation.

2.2.4. Fabric Architecture with Inputs Coming from Side (ICS)

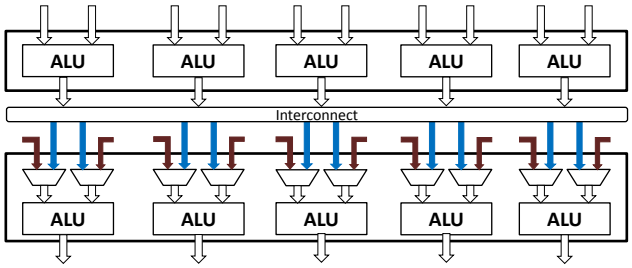


Figure 2.5: Multiplexer-based interconnector used for inputs coming from side (ICS) concept.

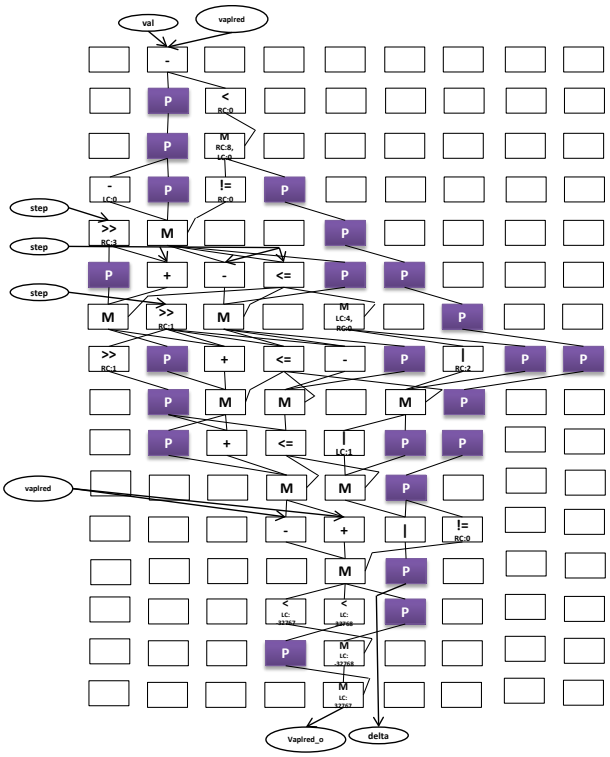


Figure 2.6: A DFG shown in Figure 2.2 mapped on the architecture with ICS.

In our previous work, instead of routing inputs and constants from the top of the fabric I fed them directly to the ALUs where they are needed. By routing inputs and constants efficiently from the side, I had achieved significant improvement in area and energy savings. For example, Figure 2.6 shows the DFG shown in Figure 2.2 mapped onto the architecture where inputs and constants are routed directly to the functional units where needed. In order to keep the figures simple, I show the constants integrated inside the ALUs and the variables in bubbles off to the sides. Constants are labeled within an ALU as "LC" and "RC". "LC" stands for a left constant, and it means that the left operand of the ALU is a constant. "RC" stands for a right constant, and it means that the right operand of the ALU is a constant. The same graph which used 17x16 standard fabric is using only 9x16 fabric having ICS. It requires 47% fewer functional units to implement the same DFG onto the fabric having ICS than the standard implementation.

CHAPTER 3

ARCHITECTURE EXPLORATION

3.1. Split/Fold Architecture

3.1.1. Motivation

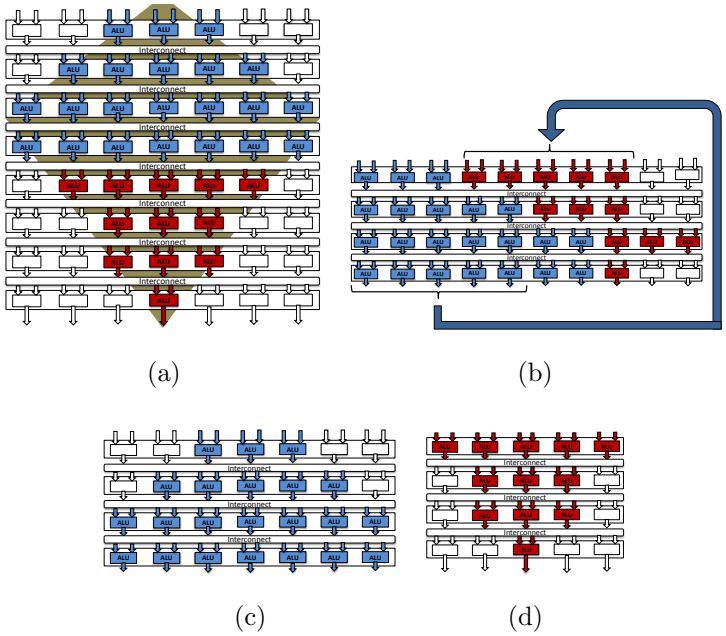


Figure 3.1: A set of subfigures to demonstrate the implementation of diamond data flow on split and fold architecture: (a) Diamond data-flow.; (b) shows fabric after folding the architecture whereas (c) and (d) are showing left and right fabric for (a).

I have observed that the data flow graphs of the application domain examined here have some common structures like wedge, diamond, trapezoid, etc. When I considered these structures, I could see that the structures got narrower as we move from top to bottom, resulting in inefficient resource utilization. ALUs of the fabric can be utilized more efficiently with reduction in the number of ALUs of the fabric after splitting/folding the fabric. I have

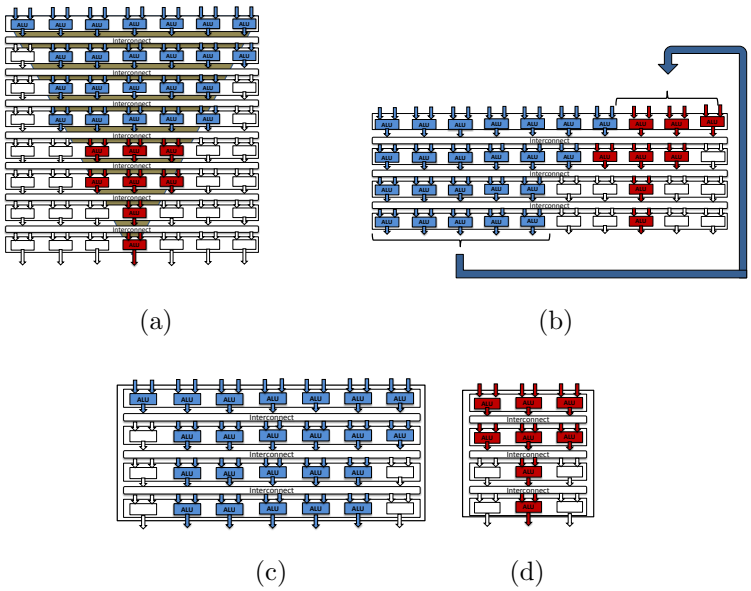
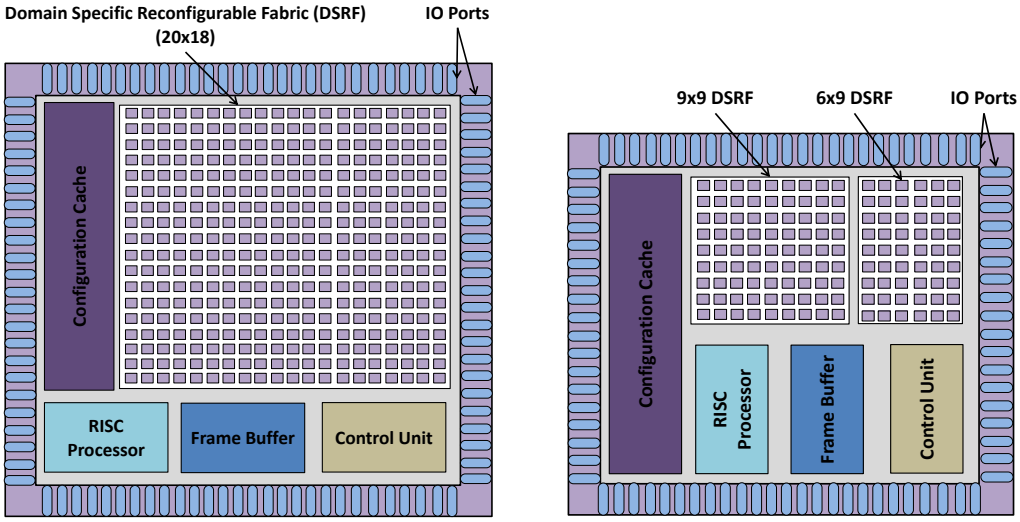


Figure 3.2: A set of subfigures to demonstrate the implementation of wedge data flow on split and fold architecture: (a) Merged data-flow.; (b) shows fabric after folding the architecture whereas (c) and (d) are showing left and right fabric for (a).

considered only two shapes in this thesis to explain the concept of the folding and splitting. Figure 3.1 shows the diamond structure and its corresponding split and fold architectures. Figure 3.2 shows the wedge structure and its corresponding split and fold architectures.

The main motivation behind splitting of the fabric is that, instead of using one big fabric, I can use two smaller fabrics of different sizes which work together to compute the desired output. By splitting the diamond structure, I get 14% theoretical area savings and by splitting the wedge structure, I get 29% theoretical area savings. I have used folding to get rid of idle ALUs of the bottom rows and utilized the idle ALUs of the top row. In the folding I have adjusted the processing ALUs of the bottom rows in the top rows. The results of folding shows huge reduction in height of the fabric with a small increase in width of the fabric. By folding the diamond structure, I get 29% theoretical area savings and by folding the wedge structure I get 29% theoretical area savings. These results motivated us to use these techniques for CGRA architecture.

3.1.2. Split Architecture



(a) Base CGRA architecture.

(b) Split CGRA architecture.

Figure 3.3: Base CGRA architectures and CGRA architecture after split .

To demonstrate the steps involved in getting a split architecture, consider an 8x8 fabric shown in Figure 3.4a. ALUs used for computations and passing data down are shown as blue-colored and red-colored boxes and the empty boxes are idle ALUs. As you can see, the graph has fewer ALUs doing work at the top and the bottom, as compared to the middle part of the fabric, where there are many ALUs that are idle and can be used in a more efficient manner. In order to achieve a split architecture, I first shift the working ALUs to either left/right as much as I can. In this example, I have shifted the working ALUs to the left side. After shifting, I have the idle ALUs on the right hand side of the fabric as shown in Figure 3.4b. Next, I split the fabric into two unequal parts. The horizontal cut (dotted line) shows the point from where the fabric can be split and the vertical dotted lines show the new maximum widths of the split fabrics. In the bottom stripes, very few ALUs are being used for computation. Due to this, I can make the bottom fabric much smaller as compared to the top fabric. Figure 3.4c and Figure 3.4d shows the split fabric architecture which consists of *left fabric* and *right fabric* with sizes 7x4 and 4x4 respectively. The results

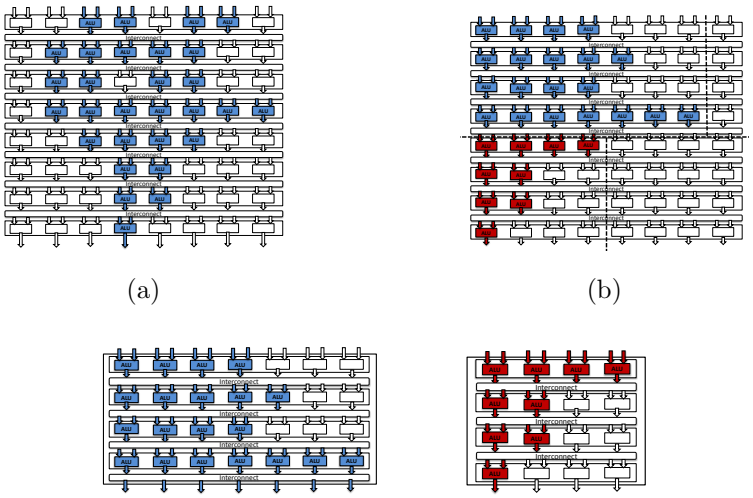


Figure 3.4: A set of subfigures to demonstrate the steps involved in getting an ICS-split architecture: (a) Basic Fabric Model.; (b) Fabric model after shifting the working ALUs towards left.; (c) and (d) left and right fabric after Splitting.

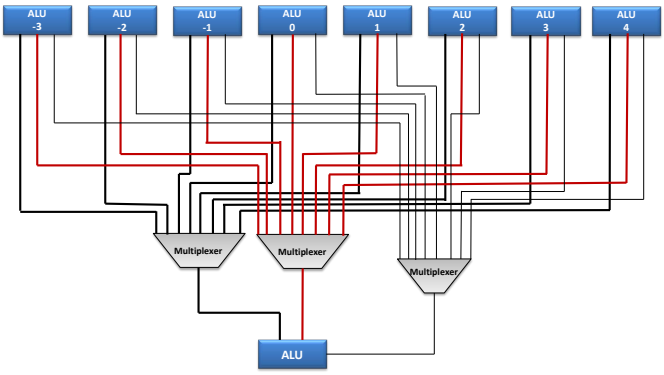
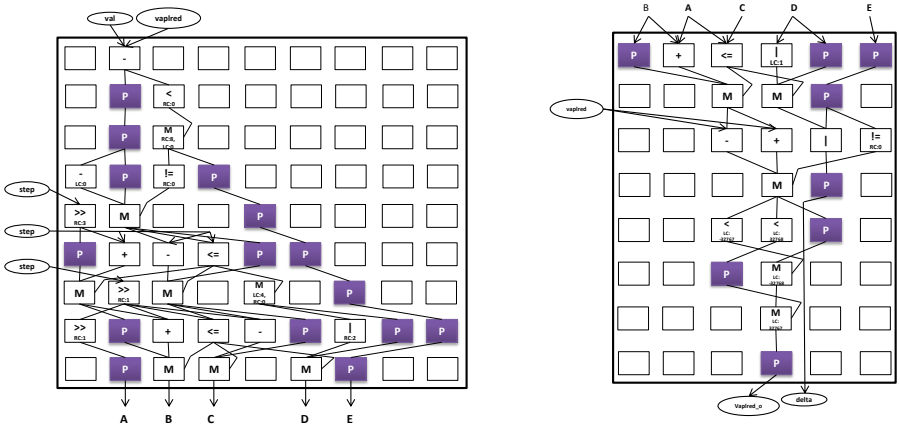


Figure 3.5: Schematic for 8:1 interconnect used in split architecture, fold achitecture, and architecture with horizontal interconnect.

of computations from the *left fabric* are fed to the *right fabric*, as shown, and the final output will be taken out from the bottom of the *right fabric*; the results can also be taken out from the *left fabric*. To route the output more efficiently from the different ALUs to the final

output port, several early exit rows are placed evenly in the *left fabric* as well as the *right fabric*. For example, if fabrics have height 9, each row (except the 1st row) is connected to early exit rows. If the output is available in any row (except the 1st row), it can be passed to the final output row through the early exit row. If output is available in row 1, then it will be passed to the row 2 and from there it can be further passed to final output port. This technique saves a significant number of functional units in the fabric which is being used for passing the output to the end of the fabric. In this example, our original graph has 64 total functional units, whereas the split fabric has 44 total functional units.

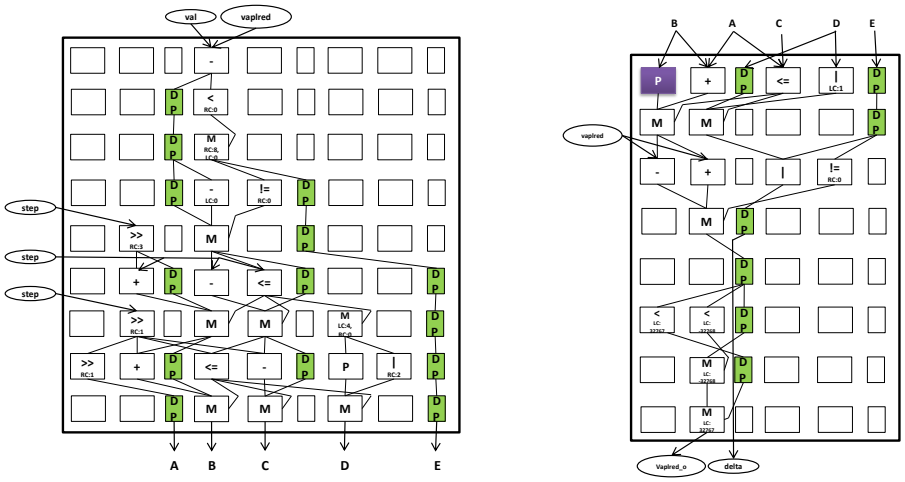


(a) Left fabric.

(b) Right fabric.

Figure 3.6: A DFG shown in Figure 2.2 mapped on the ICS-split architecture.

I implemented the graph from Figure 2.2 using split architecture as shown in Figure 3.6. The two split fabrics (*left fabric* and *right fabric*) are connected through interconnection. The signals A, B, C, D and E in Figure 3.6a shows the results of the *left fabric* which are fed to the *right fabric* as shown in Figure 3.6b. This DFG has two outputs, one of which is computed and available early in the *right fabric* (in row 4). Because of early exit rows in the fabric, this output can come out directly to the final output without using any ALUs in the successive stripes for the pass operation. The size of the ICS graph was 9x16, when the same graph is implemented on split architecture having ICS uses 9x9 and 6x8



(a) Left fabric.

(b) Right fabric.

Figure 3.7: A DFG shown in Figure 2.2 mapped on the ICS-split architecture with 33% dedicated pass gates.

fabrics and requires 10% fewer functional units as compared to the baseline (ICS-No-DPs) architecture. Figure 3.7 shows the implementation of the same DFG (Figure 2.2) onto split architecture fabric with 33% DPs. It requires 40% fewer functional units than the baseline implementation.

3.1.3. Fold Architecture

As I can see in Figure 3.8, there are some idle ALUs in the top rows of the fabric which can be used for computations. The blue and red colored ALUs are doing computations and the empty ALUs are the idle ones. In order to reduce the total number of ALUs in the fabric, I can cut the fabric using a horizontal dotted line and the idle ALUs in the top rows can now be used to do operations done by the red-colored ALUs. I call this architecture "fold architecture" because the fabric is folded from the row where the cut is made and the outputs of the blue-colored ALUs are fed back to the red-colored ALUs. The final folded fabric is shown in Figure 3.9. During the first execution cycle, blue-colored operations will do computations and red-colored ALUs stay idle, and in the second execution cycle, the

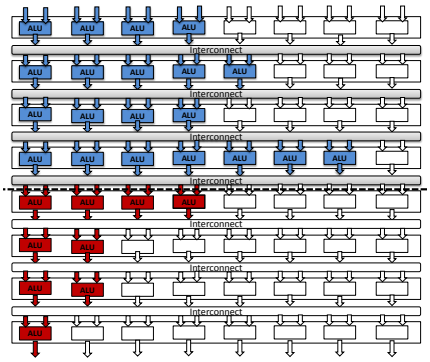


Figure 3.8: Fabric model: Horizontal line show the point from where fabric is folded.

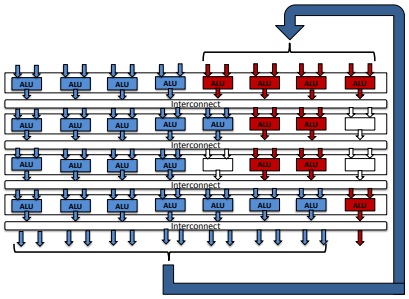
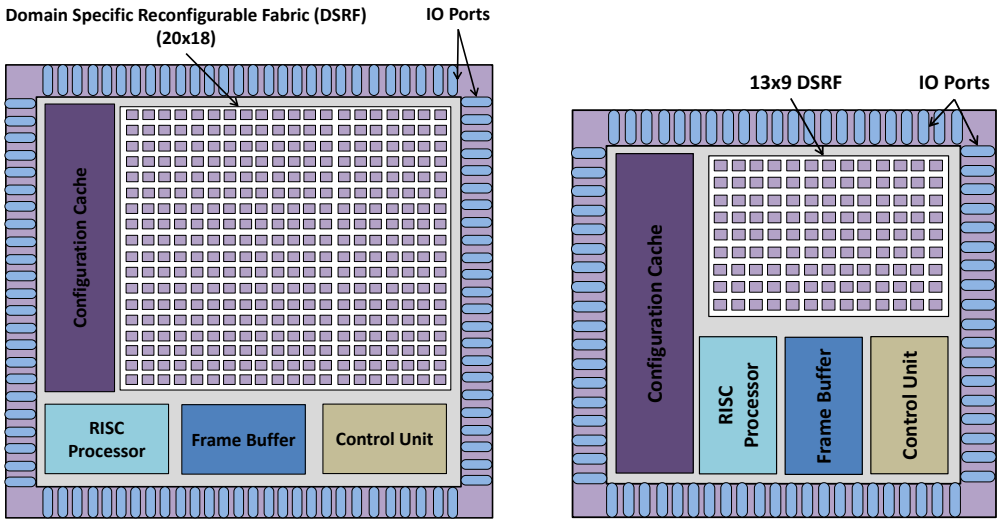


Figure 3.9: Final folded fabric model.



(a) Base CGRA Architecture.

(b) Fold CGRA Architecture.

Figure 3.10: Base CGRA architectures and CGRA architecture after fold.

blue-colored ALUs stay idle and the red-colored ones do the actual computations. The fold architecture has multiplexers at the top of the first ALU stripe that provides flexibility of getting inputs from top/outside and intermediate results from the bottom stripe. Now, if

we compare Figure 3.4a and Figure 3.9 we observe that the fabric size is reduced from 8x8 to 8x4, which shows 50% savings in terms of total number of functional units.

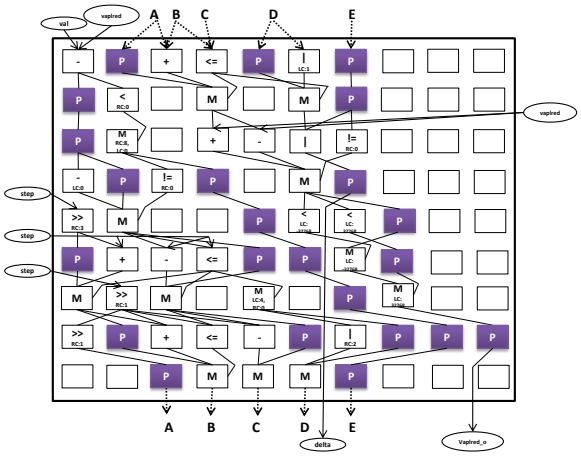


Figure 3.11: A DFG shown in Figure 2.2 mapped on the ICS-fold architecture.

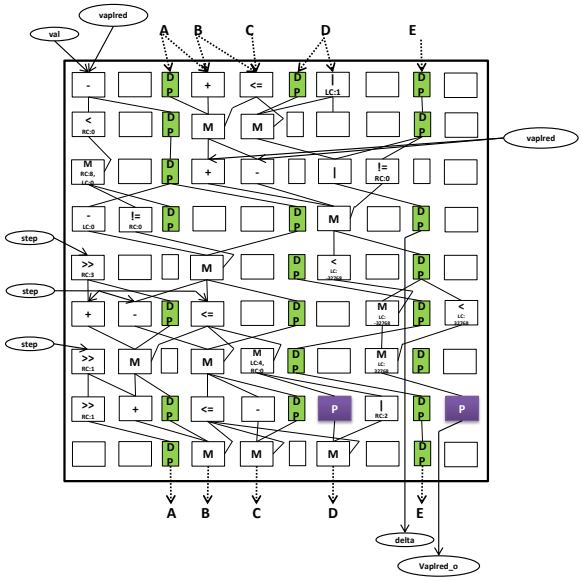


Figure 3.12: A DFG shown in Figure 2.2 mapped on the ICS-fold architecture with 33% dedicated pass gates.

Figure 3.11 shows the DFG shown in Figure 2.2 mapped onto the architecture where I have folded the fabric from the 9th row. In the first execution cycle, the intermediate results A, B, C, D, and E are taken out and sent back to the top computational stripe for the second cycle of execution and final outputs are taken out. The inputs and constants are routed directly to the functional units where needed.

The same graph that used 9x16 ICS fabric can be implemented on 10x9 fabric when I introduce ICS with folding. It will require 38% fewer functional units to implement the same DFG onto the fold architecture as compared to the ICS (ICS-No-DPs) implementation. Figure 3.12 shows the implementation of the same DFG (Figure 2.2) onto ICS-fold fabric with 33% DPs; it requires 50% fewer functional units as compared to baseline architecture.

3.2. Architecture with Multi-Level Vertical Interconnects

By looking into various benchmarks of reconfigurable fabric architecture, I observed that some intermediate results are not consumed in the stripe next to the ALUs that produced that result. To route that result to the intended ALUs, I have to use ALU only for routing that result which is a penalty in terms of area as well as in terms of energy. Architecture with multi-level vertical interconnect is a complementary approach to reduce the number of ALUs used for pass operation. This technique can route the results in a more efficient way.

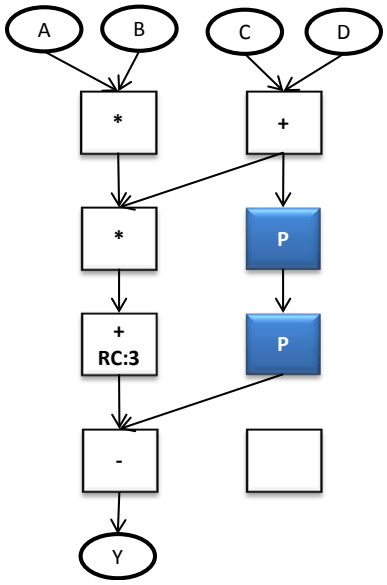


Figure 3.13: A simple example of computation.

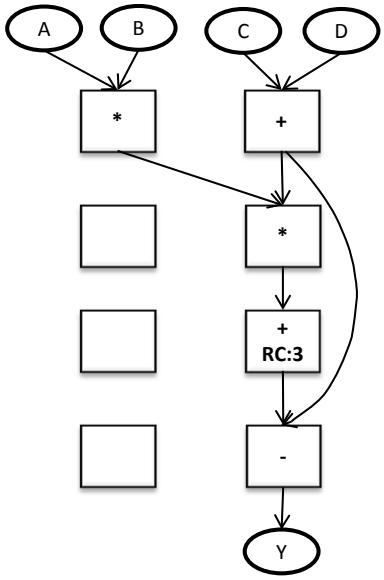


Figure 3.14: Implementation of Figure 3.13 with vertical step connection.

For example, Figure 3.13 shows a series of operations to compute a desired output "Y" on a 2x4 fabric. The result obtained from the addition of "C" and "D" is later used in multiplication and subtraction operations. To route that intermediate result to the subtraction operation, two ALUs are used for pass operation. By adding multi-level vertical interconnect, we can route the intermediate result directly to the desired ALU for subtraction operation and save ALUs used for pass operation as shown in Figure 3.14.

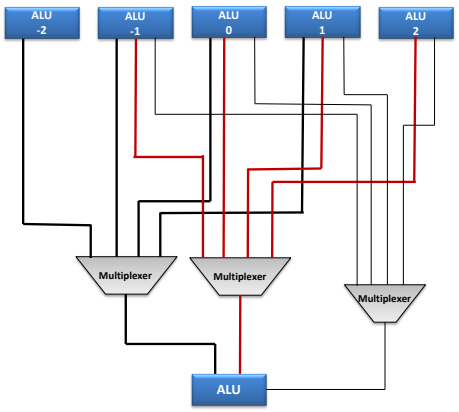


Figure 3.15: Schematic for 5:1 Interconnect build using 4:1 multiplexers.

To implement fabric architecture using multi-level vertical interconnect, I have used a 4:1 multiplexer for each operand of an ALU. Each 4:1 multiplexer can take one input from the parent stripe above using another 5:1 interconnect, one from the side (ICS), one from the grandparent stripe's ALU (from same column), and one from the great-grandparent stripe's ALU (from same column). One input out of these four inputs is used by the ALU. The obtained result from the ALU will be available at all the possible destinations which can be used by the intended ALU. The same result can be used by all possible intended ALUs which are connected to the producer ALU. The functionality will remain the same, but it will just be connected to the ALUs in a more flexible way. In other words, ALUs will work independently but they will have more vertical reachability in the same column. Architecture with multi-level vertical interconnect will have an additional interconnectivity between grandparent-to-grandchild and great-grandparent to great-grandchild of the same column.

As a result, this approach can reduce the number of ALUs used for routing and will increase the routing flexibility. In architecture with multi-level vertical interconnect, most of the intermediate results can be sent to intended ALUs directly.

Figure 3.16 shows our technique and the interconnection I have used for vertical step architecture. In order to keep the figure simple, I have shown arrows which are depicting

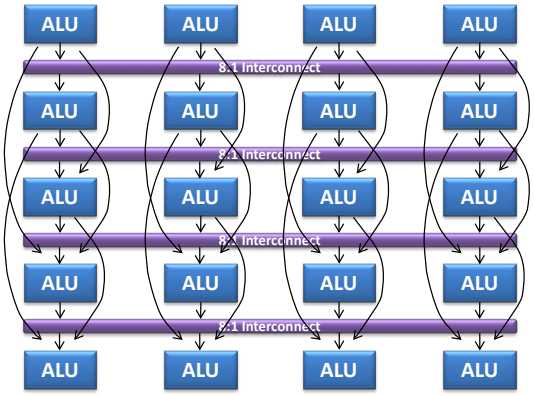


Figure 3.16: A fabric model comprises of ALUs with multi-level vertical interconnect.

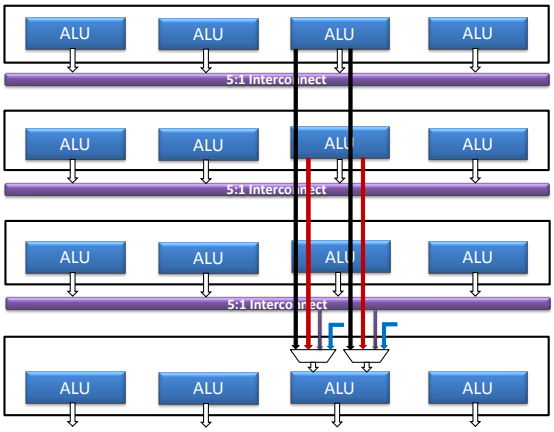


Figure 3.17: A detailed version of multi-level vertical interconnect.

all possible vertical routing interconnection except the ICS (Input Coming from Side). Figure 3.17 shows the actual interconnection of an ALU with all possible ALUs. Each operand of an ALU is coming from the multiplexer which is connected to a total of 7 different ALUs and one external input port (input coming from side or ICS). Out of 7 connected ALUs, 5 of them are from the parent stripe which is connected through the 5:1 interconnect, and one is from the grandparent stripe's ALU (from same column), and one is from the great-grandparent stripe's ALU (from same column).

In our research, I have achieved a significant improvement in area and energy savings by using architecture with multi-level vertical interconnect. For example, Figure 3.18 shows the DFG shown in Figure 2.2 mapped onto the architecture with multi-level vertical interconnect. The same graph that used 9x16 ICS fabric is using only 5x16 fabric. It requires 47% fewer functional units as compared to ICS implementation.

3.3. Architecture with Horizontal Interconnect

In addition to multi-level vertical interconnect between stripes, I have also studied the impact of horizontal interconnect between the adjacent ALUs of the same stripe onto energy

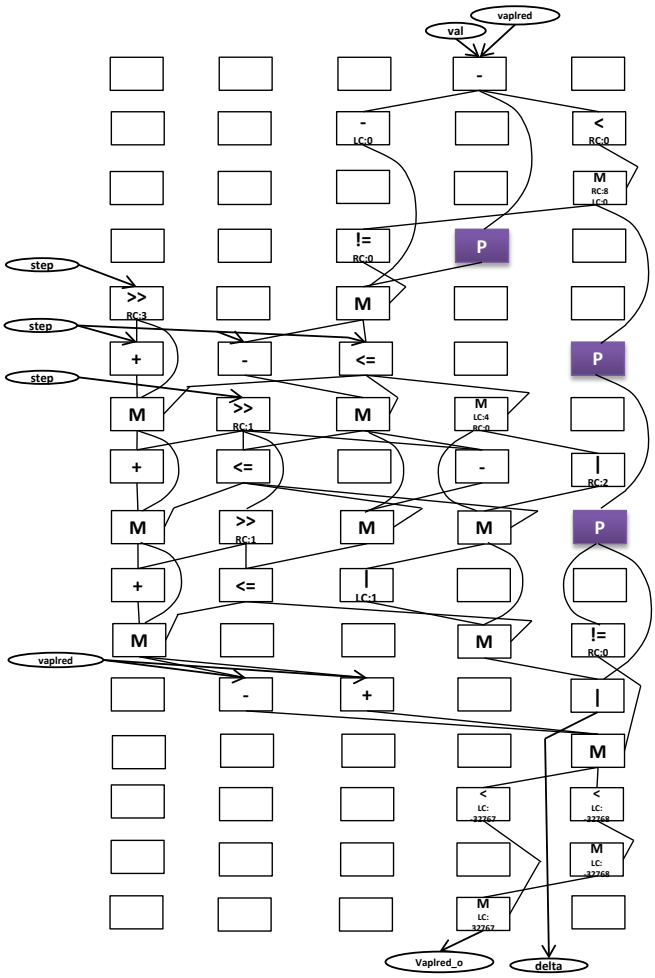


Figure 3.18: A DFG shown in Figure 2.2 mapped on the architecture with multi-level vertical interconnect.

and area. To implement architecture using horizontal interconnection I have used a 4:1 multiplexer for each operand of an ALU. Each 4:1 multiplexer can take input from the stripe above, from the side, from the adjacent right ALU, and from the adjacent left ALU. One input out of these four inputs is used by the ALU at a time. Obtained result from the ALU will be available at all the destinations which can be used by the intended ALU. The same result can be used by all possible intended ALUs which are connected to the producer ALU. In this architecture, we can wrap the ALUs in the stripes because it will restrict the ALUs to work in parallel. In this architecture, the functionality will remain the same, but the ALUs

will be connected in a more flexible way. In other words ALUs will work independently but they will have reachability to the other ALUs of the same row.

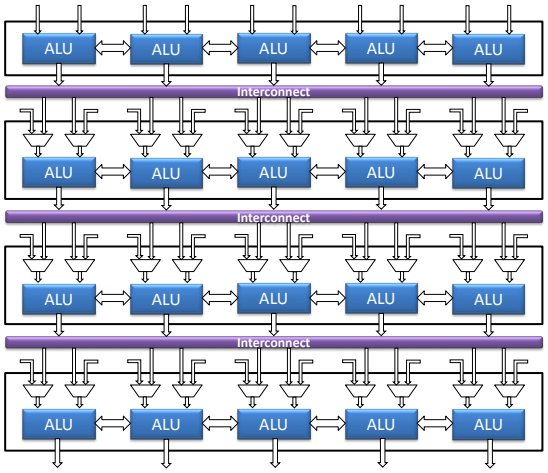


Figure 3.19: A fabric model comprises of ALUs with horizontal interconnect.

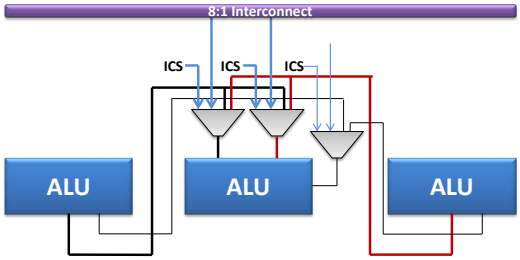


Figure 3.20: A fabric model showing horizontal interconnect concept.

Figure 3.19 shows the fabric model comprised of ALUs with horizontal interconnections. In order to keep the figure simple, I have shown the horizontal left-right arrow depicting the interconnection between two adjacent ALUs of the same ALU stripe. Figure 3.20 shows the actual interconnection of an ALU with all possible ALUs. Each operand of the center ALU is connected to total 10 different ALUs and one external input port (input coming from side (ICS)). Out of 10 connected ALUs, 8 of them are from parent stripe which is connected through the 8:1 interconnect and one is from the left adjacent ALU, and one is from the right adjacent ALU. The third multiplexer is used to provide a selector signal when the ALU is used for multiplexer operation.

This approach can reduce the height of the fabric and its overall fabric size by moving certain operations up in the fabric. This technique saves a significant number of functional units in the fabric which is being used for passing the output to the end of the fabric.

In our research, I have achieved a significant improvement in area and energy savings by using architecture with horizontal interconnect. For example, Figure 3.21 shows the DFG

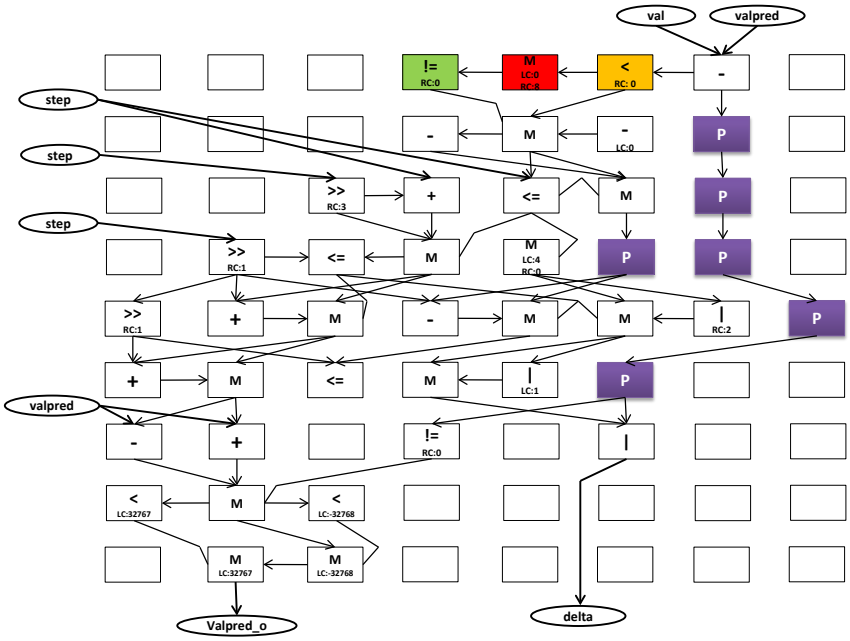


Figure 3.21: A DFG shown in Figure 2.2 mapped on the architecture with horizontal interconnect.

shown in Figure 2.2 mapped onto the architecture with horizontal interconnect. Consider red-colored ALU, the parent ALU of green-colored ALU, and child of yellow-colored ALU are in the same row. Once the parent ALU (yellow in color) computes the result it will pass it to the next ALU(red in color) which will be processed and passed to its child ALU(Green in color). The same graph which used 9x16 ICS fabric is using only 8x9 fabric. It requires 49% fewer functional units to implement the same DFG onto the architecture with horizontal interconnect than the ICS implementation.

3.4. Fully Connected Architectures

In our previous architectures (Standard, Split, Fold, architecture with horizontal interconnect and multi-level vertical interconnect), I have used the unidirectional interconnectors like “8:1 Interconnector”, “8:3 Interconnector” and “5:3 Interconnector”, which restrict the data flow, and data can flow from top to bottom. Second drawback is that, the ALUs are wrapped into stripes that force them to work in parallel. If in any case, when we have to

rerun a single ALU, we have to rerun the whole stripe resulting inefficient power utilization. Thirdly, suppose there are only 3 ALUs out of 8 in a single stripe are used at a time then there is no other way to use these 5 idle ALUs in the same run, since we have unidirectional data flow, and ALUs are working in parallel. To improve these drawbacks and provid the required reconfigurable functionality, I have used a bigger reconfigurable interconnector.

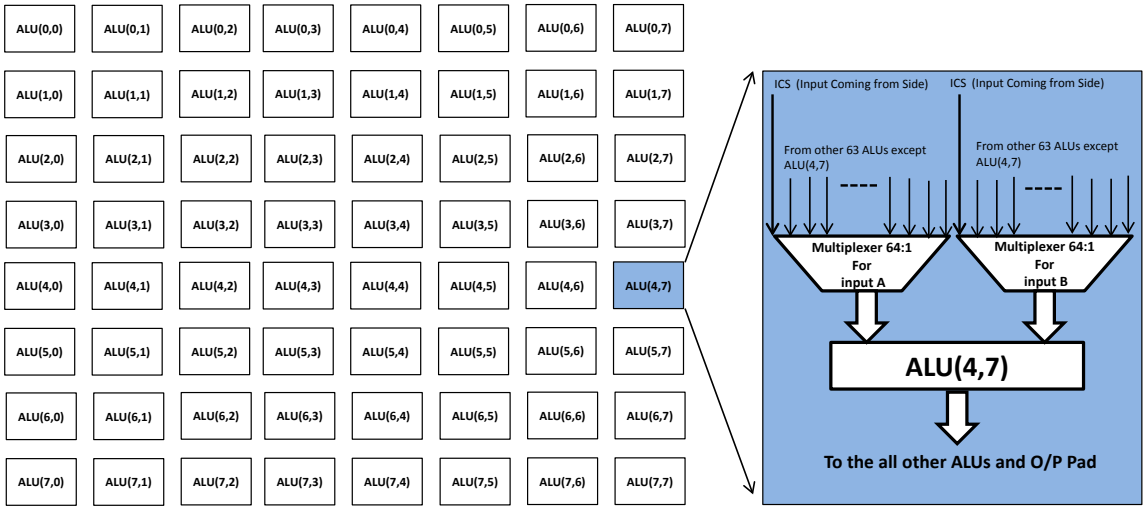


Figure 3.22: Fully connected architecture shown with its interconnects.

The proposed architecture as shown in Figure 3.22 is comprised of 8x8 fabric. In order to reduce the number ALU in the fabric, it is neccessary to provide a wide range of interconnect to support the various benchmarks. Each operand of an ALU is connected to other 63 ALUs through a 64:1 multiplexer. Since each ALU is connected to other 63 ALUs, multiplexer have an extra input node that is connected to the external input port, i.e. inputs can come from the side (ICS). One input out of these sixty four inputs is used by the ALU in each cycle. The obtained result from any ALU will be available at all the destinations that can be used by intended ALUs. The result obtained from a particular ALU can't be used by itself, but can be used by other 63 ALUs of the fabric. Under fully connected architecture I will discuss two architectures (i) fully connected homogeneous architecture and (ii) fully connected heterogeneous architecture. In homogeneous architecture each ALU can support

same type and same number of operations, which I have used in ALUs of earlier architectures, whereas in heterogeneous architecture, ALUs can support only 3 operations including noop (No Operation). Each row of the fabric consists of same type of ALUs. ALUs of different rows will have different combination of operations. Since each ALU is working independently, the data will follow ZigZag-path.

To route the output more efficiently from the different ALUs to the final output port, several early exit rows and early exit columns are placed evenly in the fabric. For example, if the fabric size is 8x8, each row and each column is connected to early exit row and early exit column respectively. If two outputs are available in same column but in different rows (let say column 2 and row 7 and row 5), one output can passed to the final output port through early exit row and other can be passed through early exit column. This technique saves a significant number of functional units in the fabric which are being used for passing the output till the end of the fabric.

3.4.1. Homogeneous Architecture

In this architecture, same ALUs are used through out the fabric with new type of arrangement. Each ALU can support same number of operations and same type of operations. The only difference between this architecture and previous ones is the way in which ALUs are arranged in the fabric. Each ALU is independent, they are physically placed in a square fabric (i.e. 8x8) but they are non-stripes based architecture. They have their separate independent interconnections which are connected to all other ALUs in the fabric.

In our research, I have achieved a significant improvement in area by using fully connected homogeneous architecture. For example, Figure 3.23 shows the DFG shown in Figure 2.2 mapped onto fully connected homogeneous architecture. In order to keep the figure simple, I have shown the data flow with arrow lines of different colors. If producer and consumer are 2 or more row apart or 2 or more column apart I have used red arrow and green arrow respectively to show the connectivity else I have used black arrows. The same graph which used 9x16 ICS fabric is using only 8x5 fabric. It requires 77% fewer functional units to

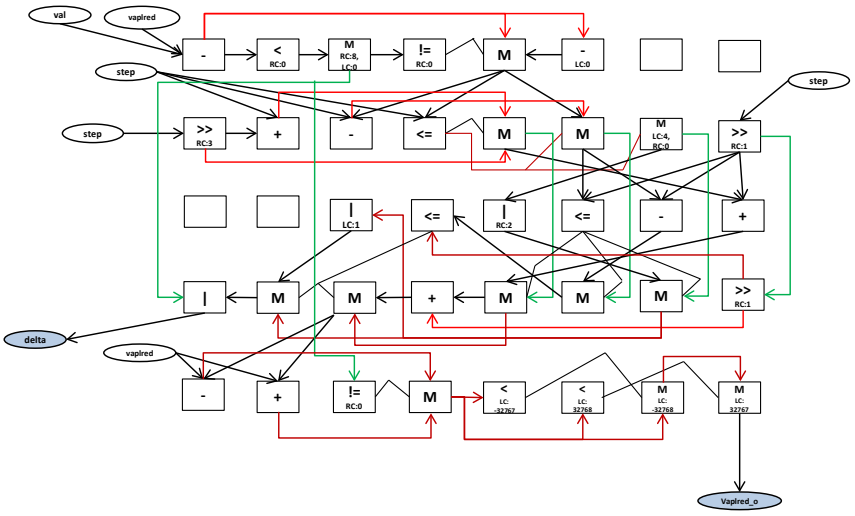


Figure 3.23: A DFG shown in Figure 2.2 mapped on the fully connected homogeneous architecture.

implement the same DFG onto fully connected homogeneous architecture than the ICS implementation.

3.4.2. Heterogeneous Architecture

In our previous architectures (Split, Fold, architecture with horizontal interconnect and vertical interconnect), I have used the ALUs which can support 15 different operations. One out of 15 operations is used in each cycle and other will remain idle. I have focused on seven different benchmarks to do case study in earlier architectures. I have observed that not all the 15 operations are used in a single benchmark but all of them are used in total seven different benchmarks I have studied. Thus, I concluded that there is a room to utilize the operations in a single ALU more efficiently. In order to do so, I came up with a novel heterogeneous architecture in which I don't have to support 15 different operations. ALUs in the new heterogeneous architecture will support fewer operations as compared to 15 operation with a penalty of additional interconnect. The reason of having fewer operations in ALU is that, any operation inside an ALU with fewer operations will take less power than

the same operations will take in an ALU with more number of operations. Fewer operations per ALU also provide area savings.

3.5. 3D CGRA Architecture

The wire length and the interconnection consume a large portion of total area of the fabric array. Three dimensional (3D) architectures which contains multiple functional layers, mitigate many of the limitations of two dimensional (2D) architecture like functionality, performance and density [31]. The 2D architectures has limitations because of long wire length which consumes a large portion of the power as well as the area. Since in 3D architecture the number of neighbors will increase, each computational unit can access large number of nearest neighbor computational units with shorter wire length. By bringing the computational units closer together in three dimensional architecture we can achieve a huge reduction in the size of the reconfigurable interconnectors required for routing the data [32], [33].

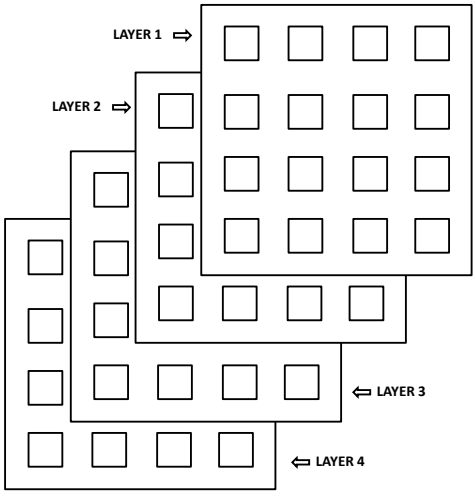


Figure 3.24: An example showing the stacking of 4x4 fabrics on top of each other.

Figure 3.24 shows the stacking of four 4x4 fabric array on top of each other. Top layer of the 3D architecture is named as layer 1, then the second layer then the third layer and the bottom layer is named as layer 4. Figure 3.25 shows the detailed version of the 3D

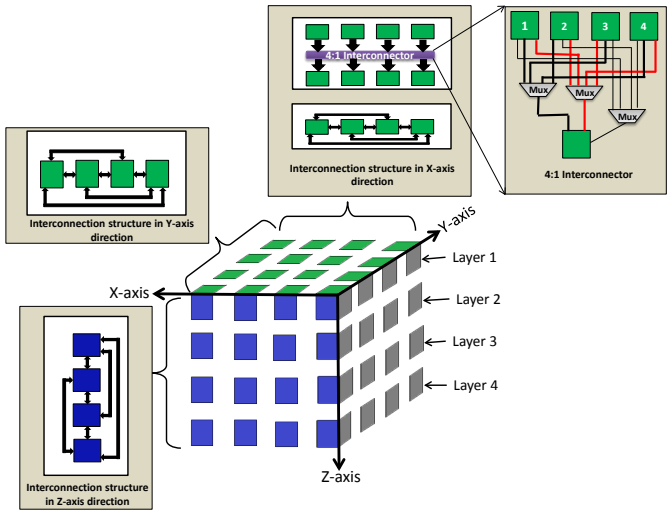
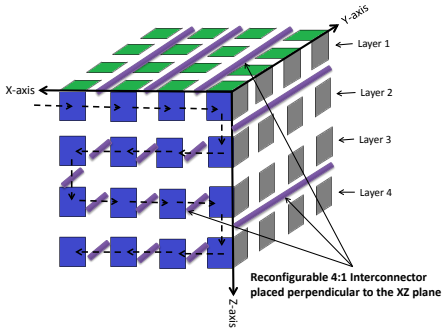


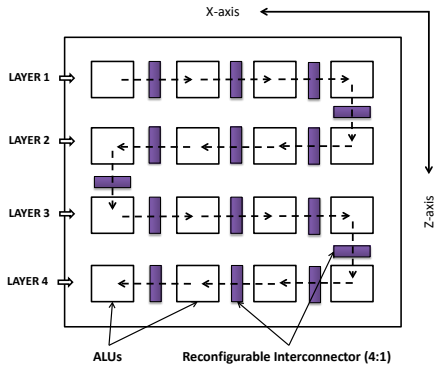
Figure 3.25: Three Dimensional Fabric Architecture shown with various interconnects.

architecture and the various interconnections I have used. XZ plane is shown by the blue colored ALUs, XY planes is shown by the green colored ALUs and YZ plane is shown with gray colored ALUs where columns are placed along with X-axis, rows are placed along with Y-axis and Z-axis gives the depth of the fabric. In any direction (each row, each column, each depth, each diagonal of each plane, and each diagonal of the 3D architecture (Cube architecture)) if 3D architecture have four ALUs in a straight line I have connected them with hopping interconnections. Between two rows I have used unidirectional 4:1 interconnect. Figure 3.26 show the direction of 4:1 interconnection as well as main data flow in 3D architecture.

Figure 3.27 shows the various diagonal in the three dimensional architecture and the hopping interconection structure for them. Figure 3.27a shows the diagonals of a plane or a layer. Each plane has 4 layers resulting in 12 layers in 3D architecture (i.e. XY-plane =4 , XZ-plane = 4, YZ-plane = 4). Figure 3.27b shows the diagonal of the cubicle structure. 3D architecture has 4 diagonal which also have hopping interconnections. Since diagonal is connecting the ALUs of different layer they are shown in different color.

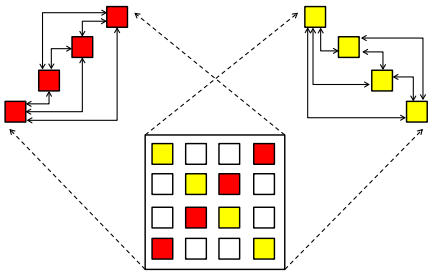


(a) 3D architeccture with 4:1 interconnector

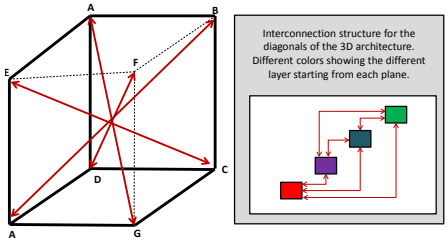


(b) XZ-plane view of the architecture showing placements of 4:1 interconnection.

Figure 3.26: A set figures showing the placement of 4:1 interconnectors and the data main data flow in 3D architecrure.



(a) Hopping Interconnection for Diagonals of a Layer



(b) Hopping Interconnection for Diagonals of the 3D Architecture.

Figure 3.27: A set of figure showing the various diagonals with hopping interconnection in the proposed 3D architecture.

The 3D fabric architecture also have several early exit rows, placed evenly in the fabric. Since 3D architecture have six faces, out of those six faces three faces have early exit rows which are placed in front of each ALU. Three faces I have chosen for this are adjacent to each other. Each face have 16 ALUs so 16 early exit ports are there, a total of 48 early exit rows are in 3D architecture. The main reason of using early exit rows in each direction is to

avoid some conflicts occurring when different final results are computed in the same row and same column. This will give the flexibility to take those results more efficiently and connect them to the final output port of the device.

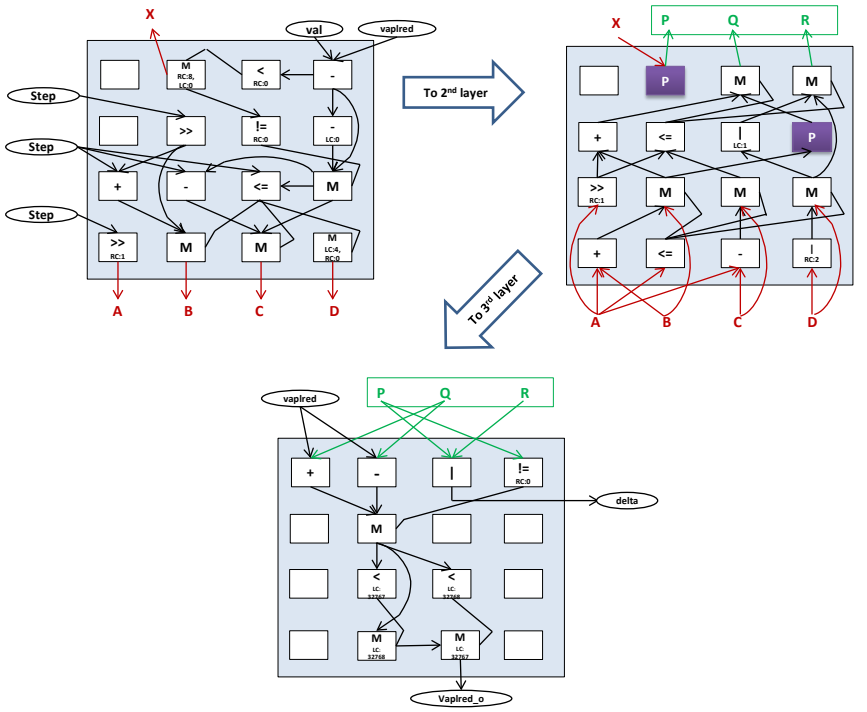


Figure 3.28: A DFG shown in Figure 2.2 mapped on the fully connected homogeneous architecture.

In our research I have achieved a significant improvement in area by using 3D architecture. For example Figure 3.28 shows the DFG shown in Figure 2.2 mapped onto 3D architecture. The intermediate outputs (shown in red color) of layer one are fed to the layer two, intermediate outputs (shown in green color) of layer two are fed to the layer three. The same graph which used 9x16 ICS fabric is using only 4x4x3 fabric. It requires 67% fewer functional units to implement the same DFG onto 3D architecture as compared to the ICS implementation.

CHAPTER 4

EXPERIMENTS AND RESULTS

4.1. Experimetal Setup

4.1.1. Evaluated Applications

In order to conduct the experiments, I selected a set of core signal processing benchmarks from MediaBench benchmark suite including the ADPCM encoder (enc), ADPCM decoder (dec), GSM channel encoder (gsm), and the MPEG II decoder (row, col). I added the Sobel (sob) and Laplace (lap) edge detection algorithms to the benchmark suite. Operations in these benchmarks include only regular arithmetic, logic and shift operations such as addition, multiplication, AND, OR, right-shift, etc.

4.1.2. Experimental Methodology

VHDL is used as a hardware description language because of the flexibility to exchange among the environments. The fabric models are implemented in parameterized VHDL using the generic capability of the VHDL language. I used Synopsys Design Compiler to synthesize our designs and Synopsys PrimePower is used for the power estimation.

The whole experiment process consists of three phases (i) synthesis phase (ii) simulation phase (iii) power analysis phase. Synthesis phase is one of the most important steps involved in any design. Synthesis is an automatic method of converting a higher level of abstraction to a lower level of abstraction. In other words the synthesis process converts register transfer level (RTL) descriptions to gate-level netlists. These gate-level netlists can be optimized for area, speed, testability, etc. I synthesized the fabric VHDL into synopsys 90nm generic library using Synopsys Design Compiler. In simulation phase the post-synthesis design (Gate-level Netlist) was simulated in Mentor Graphics ModelSim to calculate the delay of each design and these simulations were used as stimulus to the Syopsys PrimePower tool to

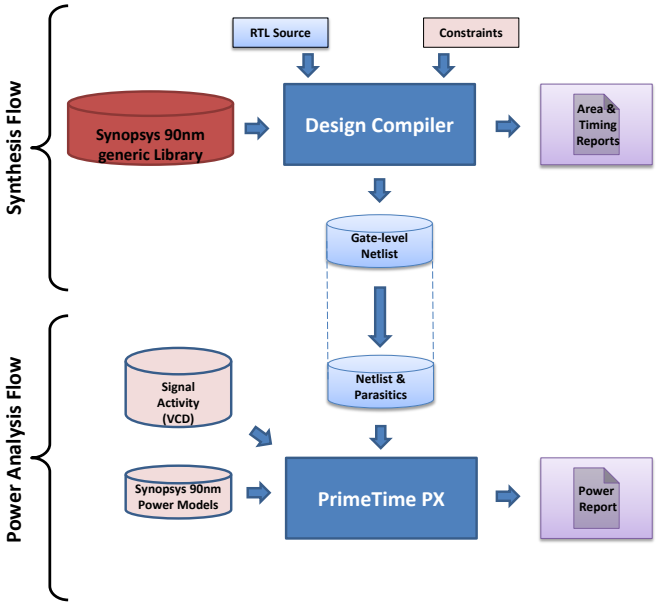


Figure 4.1: Experimental flow.

estimate the power consumption of the device. The Energy was calculated by computing the product of the power and delay of the design. Figure 4.1 shows the experimental flow, the area and timing reports are based on the results of Synopsys Design Compiler, delay is based on simulation from ModelSim and power number is based on the results given by Synposys PrimeTime tool.

4.2. Results

4.2.1. Split and Fold Architectures

4.2.1.1. Fabric Sizes

Table 4.1 and Table 4.2 provides a summary of the size requirements of the seven signal and image processing benchmarks Section 4.1.1 mapped to the fabric for different standard and ICS architectures. The fabric size is given by Width x Height. As we can see that by folding and by splitting the size of the fabric get reduced and the benchmarks can fit in smaller fabric. The benchmarks with higher number of constants such as “enc”, “dec”, “col”, and “gsm” show large area improvements with Splitting and Folding techniques. Folding gives us

Table 4.1: Fabric size (Width x Height) for mapping various benchmarks onto various standard, standard-split and standard-fold architectures using the heuristic mapper.

	enc	**dec**	**row**	**col**	**gsm**	**sob**	**lap**
std-No DPs	17x16	16x14	18x10	20x12	18x18	10x10	15x8
std-25% DPs	16x16	15x14	13x10	15x12	15x18	9x10	13x8
std-33% DPs	16x16	14x14	13x12	16x12	16x18	10x10	14x8
std-50% DPs	10x16	10x14	11x12	12x18	11x18	8x10	13x10
std-split-No DPs	13x9	14x9	17x9	19x9	10x9	10x9	15x8
	10x8	4x4	4x2	5x4	12x9	0x0	0x0
std-split-25% DPs	9x9	10x9	12x9	13x9	9x9	8x9	13x8
	7x8	3x4	4x2	4x4	9x9	0x0	0x0
std-split-33% DPs	9x9	9x9	11x9	12x9	8x9	8x9	13x8
	6x8	4x4	4x2	4x4	8x9	0x0	0x0
std-split-50% DPs	7x9	6x9	11x9	11x9	6x9	8x9	12x9
	5x8	2x4	5x4	8x9	6x9	0x0	0x0
std-fold-No DPs	19x9	17x9	21x9	23x9	16x9	10x9	15x8
std-fold-25% DPs	16x9	12x9	15x9	17x9	12x9	8x9	13x8
std-fold-33% DPs	16x9	12x9	15x9	14x9	12x9	8x9	13x9
std-fold-50% DPs	12x9	8x9	16x9	14x9	10x9	8x9	12x9

better improvements in all the benchmarks. For example, "enc" implemented on standard fabric with 33% DPs was using 16x16 fabric and the same benchmark after folding takes 16x9 and after splitting takes two fabrics with size 9x9 & 6x8. When the same benchmark is implemented on the fabric with ICS & 33% DPs takes 6x16 fabric whereas the splitting of the same benchmark it can be implemented onto two fabrics of size 6x9 & 4x8. When we fold the architecture, same benchmark can be implemented on smaller fabric with size 8x9.

Once all benchmarks were mapped to a fabric using a particular architecture, the fabric size was fixed to the smallest size that could fit all seven benchmarks. For various standard architectures, I have fixed the fabric sizes to 20x18, 16x18, 16x18, and 13x18 for std-No DPs, std-25%DPs, std-33%DPs, and std-50%DPs respectively. Table 4.3 shows the final fabric sizes of the various Split and Fold architecture. The benchmarks can be mapped onto smaller

Table 4.2: Fabric size (Width x Height) for mapping various benchmarks onto various ICS, ICS-split and ICS-fold architectures using the heuristic mapper.

	enc	**dec**	**row**	**col**	**gsm**	**sob lap**	
ICS-No DPs	9x16	4x14	9x10	9x12	5x18	7x10	8x10
ICS-25% DPs	7x16	6x14	8x10	8x12	3x18	8x10	8x10
ICS-33% DPs	6x16	4x14	8x10	7x12	3x18	6x10	8x10
ICS-50% DPs	5x16	3x14	7x10	6x12	3x18	6x10	7x10
ICS-split-No DPs	9x9	4x9	9x9	9x9	4x9	7x9	8x9
	6x8	4x4	4x2	4x4	4x9	0x0	0x0
ICS-split-25% DPs	7x9	4x9	8x9	8x9	3x9	6x9	8x9
	5x8	3x4	3x2	4x2	3x9	0x0	0x0
ICS-split-33% DPs	6x9	4x9	8x9	7x9	3x9	6x9	8x9
	4x8	2x4	4x2	4x4	3x9	0x0	0x0
ICS-split-50% DPs	5x9	3x9	7x9	6x9	3x9	6x9	7x9
	4x8	2x6	3x2	3x6	3x9	0x0	1x2
ICS-fold-No DPs	10x9	8x9	13x9	13x9	6x9	7x9	8x9
ICS-fold-25% DPs	9x9	6x9	10x9	11x9	5x9	6x9	8x9
ICS-fold-33% DPs	8x9	6x9	10x9	11x9	4x9	6x9	8x9
ICS-fold-50% DPs	7x9	5x9	9x9	9x9	4x9	6x9	7x9

size fabric with folding and splitting having ICS architectures as compared to the folding of standard architectures. For example, the benchmarks implemented on standard architecture with no DPs used 20x18 size fabric and the same set of benchmarks can be implemented on 23x9 fabric when we fold the fabric. The same benchmarks can be implemented with combination of 19x9 & 12x9 fabrics when we split the standard fabric, but when we introduce the ICS along with folding, the same benchmarks can be implemented 13x9 fabric whereas using ICS along with splitting reduces the fabric to 9x9 & 6x9.

Table 4.4 and Table 4.5 shows the comparison between the base fabric size and the fabrics we will get after splitting or folding the base architectures.

Table 4.4 shows the reduction in the total number of functional units in fabric architecture with std-split & std-fold architectures Vs standard implementation. On an average, the

Table 4.3: Minimum fabric size (Width x Height) for various fabric architectures.

Architecture	Fabric size (std)	Fabric size std-split	Fabric size std-fold
No DPs	20x18	19x9	23x9
		12x9	
25% DPs	16x18	13x9	17x9
		9x9	
33% DPs	16x18	12x9	16x9
		8x9	
50% DPs	13x18	11x9	16x9
		8x9	

(a) Standard Architectures.

Architecture	Fabric size ICS	Fabric size ICS-split	Fabric size ICS-fold
No DPs	9x18	9x9	13x9
		6x9	
25% DPs	8x18	8x9	11x9
		5x9	
33% DPs	8x18	8x9	11x9
		4x9	
50% DPs	7x18	7x9	9x9
		4x9	

(b) ICS Architectures.

Table 4.4: Comparison of the total number of functional units in standard fabric architectures vs standard-split & standard-fold fabric architectures.

Architecture	Standard	std-split	% Savings
No DPs	360	279	23
25% DPs	288	198	31
33% DPs	288	180	38
50% DPs	234	171	27

(a) Standard-Split.

Architecture	Standard	std-fold	% Savings
No DPs	360	207	43
25% DPs	288	153	47
33% DPs	288	144	50
50% DPs	234	144	38

(b) Standard-Fold.

number of functional units gets reduced by 38% and 50% in std-split and std-fold fabrics both having 33% DPs as compared to standard architecture having 33% DPs. Table 4.5 shows the reduction in the total number of functional units in fabric architecture with ICS-split & ICS-fold architectures Vs ICS implementation. On an average, the number of functional units gets reduced by 20% and 29% in ICS-split and ICS-fold fabrics as compared to ICS architecture.

Table 4.5: Comparison of the total number of functional units in ICS fabric architectures vs ICS-split & ICS-fold fabric architectures.

Architecture	ICS	ICS-split	% Savings
No DPs	162	129	20
25% DPs	144	117	19
33% DPs	144	108	25
50% DPs	126	99	21

(a) ICS-split.

Architecture	ICS	ICS-fold	% Savings
No DPs	162	117	28
25% DPs	144	99	31
33% DPs	144	99	31
50% DPs	126	81	36

(b) ICS-fold.

I have also looked at the utilization of ALUs for pass operation for various fabric architecture implementations. Table 4.6 shows the total number of ALUs used as pass gate in various fabric architecture. The numbers of ALUs used as pass gate have significantly been reduced when we split or fold the standard fabric. The number of ALUs used as pass gates have significantly been reduced when we compare the ICS, ICS-fold architectures, ICS-fold architectures having a combination of ICs and DPs features with the baseline architectures. Consider the case of "gsm". When we mapped this benchmark onto the standard fabric with no dedicated pass gates, 139 out of 360 ALUs were being used for pass operation. When we split the architecture, the numbers of ALUs being used as pass gates were reduced to 94 and after folding they get reduced to 62. In the Standard architecture, when we introduced ICS It gives the better result than other options.

4.2.1.2. Area Evaluation

This section presents area savings for the suite of benchmarks for splitting & folding of standard architecture and splitting & folding of ICS architecture.

Table 4.11 shows the area savings per benchmark for various standard fabric implementations. Our baseline architecture is "std-No-DPs". We compared every standard, standard-split and standard fold architectural option with our reference baseline architecture to obtain savings. On an average, the standard fabric architecture with 50% DPs achieves 19% area

Table 4.6: Number of ALUs used as pass gates in various standard (std), std-split, std-fold, ICS, ICS-split, ICS-fold fabric architectures.

	enc	dec	row	col	gsm	sob	lap
std-No DPs	126	71	41	72	139	19	17
std-25% DPs	65	23	14	26	69	4	2
std-33% DPs	41	16	3	14	50	1	1
std-50% DPs	19	4	7	6	22	0	1
std-split-No DPs	91	52	43	66	94	18	17
std-split-25% DPs	54	31	18	28	55	1	1
std-split-33% DPs	34	18	5	16	31	1	0
std-split-50% DPs	16	5	2	3	8	0	0
std-fold-No DPs	91	52	43	63	62	18	17
std-fold-25% DPs	56	31	18	28	38	1	1
std-fold-33% DPs	40	17	4	16	31	1	0
std-fold-50% DPs	20	4	2	0	6	0	0
IC-No DPs	21	12	4	4	9	5	5
IC-25% DPs	6	0	0	0	1	0	0
IC-33% DPs	2	0	0	0	1	0	0
IC-50% DPs	0	0	0	0	0	0	0
ICS-split-No DPs	23	10	8	8	8	4	4
ICS-split-25% DPs	7	2	2	3	0	0	0
ICS-split-33% DPs	2	0	2	2	0	0	0
ICS-split-50% DPs	0	0	0	0	0	0	0
ICS-fold-No DPs	23	10	4	8	8	4	4
ICS-fold-25% DPs	5	2	0	0	3	0	0
ICS-fold-33% DPs	2	0	0	0	0	0	0
ICS-fold-50% DPs	0	0	0	0	0	0	0

improvement as compared to the baseline architecture. std-split architecture with 50% dedicated pass gates provides 40% area savings whereas std-fold architecture with 50% dedicated pass gates provides 40% area savings as compared to the baseline architecture.

Table 4.12 shows the area savings per benchmark for various ICS fabric implementations. Our baseline architecture is "ICS-No-DPs". We compared every architectural option with our reference baseline architecture to obtain savings. The baseline architecture with 50%

Table 4.7: Percentage area savings per benchmark mapped onto standard, DP, std-split, std-fold and hybrid architectures. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
std-No DPs	-	-	-	-	-	-	-	-
std-25% DPs	4	4	26	23	15	8	12	13
std-33% DPs	3	10	11	18	8	-2	4	7
std-50% DPs	38	34	23	4	35	16	-14	19
std-split-No DPs	27	37	10	20	39	10	0	21
std-Split-25% DPs	48	53	34	45	49	27	12	38
std-Split-33% DPs	51	55	39	47	54	27	11	40
std-split-50% DPs	59	70	30	25	64	25	5	40
std-fold-No DPs	37	31	-6	13	55	9	-1	20
std-fold-25% DPs	45	50	23	34	66	26	11	36
std-fold-33% DPs	45	50	22	45	65	26	10	38
std-fold-50% DPs	56	65	15	44	70	23	4	40

Table 4.8: Percentage area savings per benchmark mapped onto ICS, DP, ICS-split, ICS-fold and hybrid architectures. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	21	-1	9	9	39	14	-1	13
ICS-33% DPs	32	-3	9	21	39	13	-2	16
ICS-50% DPs	41	22	19	30	38	10	8	24
ICS-split-No DPs	10	7	1	10	20	10	10	10
ICS-split-25% DPs	27	13	12	25	39	22	9	21
ICS-split-33% DPs	39	19	9	27	39	21	8	23
ICS-split-50% DPs	44	27	20	30	39	19	15	28
ICS-fold-No DPs	37	-30	-31	-9	40	22	9	5
ICS-fold-25% DPs	42	1	-3	6	48	21	8	18
ICS-fold-33% DPs	49	0	-3	6	59	21	7	20
ICS-fold-50% DPs	54	15	4	20	58	18	17	27

dedicated pass gates provides 24% area savings base architecture. The ICS-split and ICS-fold architectures with 25% dedicated pass gates provide area savings of 21% and 18% respectively. The ICS-split with 50% dedicated pass gates provides 28% area savings and the ICS-fold with 50% dedicated pass gates provides 27% area savings as compared to the baseline architecture.

4.2.1.3. Energy Evaluation

Table 4.9: Energy savings (%) per benchmark mapped onto standard, DP, std-split, std-fold and hybrid architectures. (Negative numbers mean cost).

	enc	**dec**	**row**	**col**	**gsm**	**sob**	**lap**	**average**
std-No DPs	-	-	-	-	-	-	-	-
std-25% DPs	21	28	4	6	20	7	13	14
std-33% DPs	30	29	5	9	27	9	13	18
std-50% DPs	37	38	3	6	30	8	9	19
std-split-No DPs	20	18	-1	2	21	1	0	9
std-split-25% DPs	33	28	2	7	28	9	13	17
std-split-33% DPs	38	29	5	9	33	10	13	20
std-split-50% DPs	44	38	3	9	47	10	9	23
std-fold-No DPs	19	17	-2	2	37	0	-3	10
std-fold-25% DPs	30	26	1	6	38	8	11	17
std-fold-33% DPs	33	33	4	9	36	9	10	19
std-fold-50% DPs	42	43	2	9	51	9	6	23

This section presents area savings for the suite of benchmarks for splitting & folding of standard architecture and splitting & folding of ICS architecture.

I have also conducted energy simulations on a subset of architectures discussed in this subsection for comparative study of the energy consumed by the device. The energy results for different standard fabric architectures are shown in Table 4.9 and the energy results for different ICS fabric architectures are shown in Table 4.10. The standard fabric with 33% DPs shows an energy improvement of 18%. There is a minimal power and energy cost to pay for the additional hardware required to route the intermediate signals in Split & Fold

Table 4.10: Energy savings (%) per benchmark mapped onto ICS, DP, ICS-split, ICS-fold and hybrid architectures. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	13	14	1	4	9	7	3	7
ICS-33% DPs	19	17	0	4	9	9	3	8
ICS-50% DPs	19	16	0	2	8	8	3	7
ICS-split-No DPs	-3	4	-2	-2	1	1	2	0
ICS-split-25% DPs	12	16	0	3	9	4	5	7
ICS-split-33% DPs	18	16	-1	3	8	4	6	8
ICS-split-50% DPs	19	18	0	2	9	4	2	8
ICS-fold-No DPs	-1	3	-3	3	0	-1	-5	-1
ICS-fold-25% DPs	16	15	-2	3	10	1	-2	6
ICS-fold-33% DPs	18	16	-2	3	7	1	-2	6
ICS-fold-50% DPs	19	16	-1	2	8	1	-4	6

fabric but the same benchmarks can now be implemented on much smaller fabric that leads to savings in the energy consumption. The splitting of standard fabric with 33% DPS shows 20% energy improvement whereas folding of standard fabric with 33% DPS shows 19% energy saving as compared to baseline (std-No-DPs) architecture.

The ICS fabric with 33% DPs shows an energy improvement of 8% as compared to the baseline architecture (ICS-No-DPs), whereas by having a combination of 50% DPs and ICS in the ICS-split and ICS-fold hybrid architectures, we achieved energy savings of 8% and 6% respectively as compared to the baseline (ICS-No-DPs) architecture with a significant improvement in terms of area. There is a minimal power and energy cost to pay for the additional hardware.

4.2.1.4. Energy Evaluation

This section presents energy vs area tradeoffs for the suite of benchmarks for standard, dedicated pass gate, std-split, std-fold, ICS, ICS-split, ICS-fold, and hybrid fabric architectures. Our baseline architecture for various implementation for standard architecture is

Table 4.11: Percentage savings in terms of number of functional units per benchmark mapped onto standard, DP, std-split, std-fold and hybrid architectures. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
std-No DPs	-	-	-	-	-	-	-	-
std-25% DPs	6	6	28	25	17	10	13	15
std-33% DPs	6	13	13	20	11	0	7	10
std-50% DPs	41	38	27	10	39	20	-8	24
std-split-No DPs	28	37	11	20	39	10	0	21
std-split-25% DPs	50	54	36	45	50	28	13	39
std-split-33% DPs	53	57	41	48	56	28	13	42
std-split-50% DPs	62	72	34	29	67	28	10	43
std-fold-No DPs	37	32	-5	14	56	10	0	19
std-fold-25% DPs	47	52	25	36	67	28	13	38
std-fold-33% DPs	47	52	25	48	67	28	13	40
std-fold-50% DPs	60	68	20	48	72	28	10	44

"std-No-DPs" and our baseline architecture for various implementation for ICS architecture is "ICS-No-DPs". I compared every architectural option with our reference baseline architecture to obtain savings. On an average, the standard fabric architecture with 50% DPs achieves 19% energy improvement, 19% area savings and requires 24% fewer functional units as compared to the baseline architecture. std-split architecture with 50% dedicated pass gates provides 23% energy savings, 40% area savings and requires 43% fewer functional units as compared to the baseline architecture whereas std-fold architecture with 50% dedicated pass gates provides 23% energy savings, 40% area savings and requires 44% fewer functional units as compared to the baseline architecture. The ICS architecture with 50% dedicated pass gates provides 7% energy savings, 24% area savings and requires 27% fewer functional units as compared to its base architecture. The ICS-split with 25% dedicated pass gates provides 7% energy savings, 21% area savings and requires 22% fewer functional units as compared to the baseline architecture. The ICS-split with 50% dedicated pass gates

Table 4.12: Percentage savings in terms of number of functional units per benchmark mapped onto ICS, DP, ICS-split, ICS-fold and hybrid architectures. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	22	0	11	11	40	14	0	14
ICS-33% DPs	33	0	11	22	40	14	0	17
ICS-50% DPs	44	25	22	33	40	14	13	27
ICS-split-No DPs	10	7	1	10	20	10	10	10
ICS-split-25% DPs	28	14	13	26	40	23	10	22
ICS-split-33% DPs	40	21	11	27	40	23	10	25
ICS-split-50% DPs	47	30	23	33	40	23	19	31
ICS-fold-No DPs	38	-29	-30	-8	40	23	10	6
ICS-fold-25% DPs	44	4	0	8	50	23	10	20
ICS-fold-33% DPs	50	4	0	8	60	23	10	22
ICS-fold-50% DPs	56	20	10	25	60	23	21	31

provides 8% energy savings, 28% area savings and requires 31% fewer functional units as compared to the baseline architecture whereas The ICS-fold with 50% dedicated pass gates provides 6% energy savings, 27% area savings and requires 31% fewer functional units as compared to the baseline architecture.

4.2.2. Architectures with Multi-level Vertical Interconnect and Horizontal Interconnect

4.2.2.1. Fabric Sizes

Table 4.13 provides a summary of the size requirements of the seven signal and image processing benchmarks Section 4.1.1 mapped to the fabric for various ICS architectures, architecture with multi-level vertical interconnect and architecture with horizontal interconnect. As we can see that the result we are getting with the help of 50% DPs in ICS architecture can be obtained by using multi-level vertical interconnects. When we use horizontal interconnect in the architecture having ICS, benchmarks can be fit on much smaller fabrics as compare to ICS architectures. For example, "enc" implemented on ICS fabric

Table 4.13: Fabric size (Width x Height) for mapping various benchmarks onto various ICS architectures , architectures with multi-level VI and architectures with HI using the heuristic mapper.

	enc	**dec**	**row**	**col**	**gsm**	**sob**	**lap**
ICS-No DPs	9x16	4x14	9x10	9x12	5x18	7x10	8x10
ICS-25% DPs	7x16	6x14	8x10	8x12	3x18	8x10	8x10
ICS-33% DPs	6x16	4x14	8x10	7x12	3x18	6x10	8x10
ICS-50% DPs	5x16	3x14	7x10	6x12	3x18	6x10	7x10
Architecture with multi-level VI	5x16	3x14	8x10	8x12	3x18	5x10	6x12
Architecture with HI	8x9	6x6	9x7	11x7	4x8	7x6	8x6

with 50% DPs was using 5x16 fabric, the same benchmark takes 5x16 fabric when we used multi-level vertical interconnect whereas architecture having horizontal interconnect takes 8x9 fabric.

Table 4.14: Minimum Fabric size (Width x Height) for various fabric architectures.

Architecture	**Fabric size**	**Fabric size**	**Fabric size**
	ICS	Architecture with multi-level VI	Architecture with HI
No DPs	9x18	8x18	11x9
25% DPs	8x18	-	-
33% DPs	8x18	-	-
50% DPs	7x18	-	-

Table 4.14 shows the final fabric sizes of the various ICS architectures, architectures having multi-level VI and architectures having HI.

Table 4.15 shows the comparison between the ICS fabric size Vs the fabrics we will get after using multi-level vertical interconnect and horizontal interconnect.

As Table 4.16a shows, the number of functional units get reduced by 11% when we use the multi-level vertical interconnect (VI) to route the data vertically in the fabric as compared to base architecture (ICS-NO-DPs). Table 4.16b shows the number of functional

Table 4.15: Comparison of the total number of functional units in ICS fabric architectures vs Architecture with multi-level vertical interconnect & Architecture with horizontal interconnect.

Architecture	ICS	Architecture with multi-level VI	% Savings
No DPs	162	144	11

(a) Architecture with multi-level vertical interconnect.

Architecture	ICS	Architecture with HI	% Savings
No DPs	162	99	39

(b) Architecture with horizontal interconnects.

Table 4.16: Number of ALUs used as pass gates in various ICS architectures, architecture with multi-level VI and architecture with HI.

	enc	dec	row	col	gsm	sob	lap
IC-No DPs	21	12	4	4	9	5	5
IC-25% DPs	6	0	0	0	1	0	0
IC-33% DPs	2	0	0	0	1	0	0
IC-50% DPs	0	0	0	0	0	0	0
Architecture with multi-level VI	4	1	1	1	0	0	2
Architecture with HI	5	0	3	3	0	1	0

units get reduced by 39% when we use the horizontal interconnects (HIs) as compared to base architecture (ICS-NO-DPs).

Table 4.16 shows the total number of ALUs used as pass gate in various fabric architecture. The number of ALUs used as pass gates has significantly been reduced when we used the multi-level VI or HI in the fabric. Results shows that complementary approaches for routing the data provides the improvement in the fabric size as well as the number of ALUs used as pass-gate also reduced significantly. On the other hand the additional hardware we required for these techniques are much smaller than the conventional technique.

Table 4.17: Percentage area savings per benchmark mapped onto ICS, DPs, and hybrid architectures, architecture with multi-level VIs and HIs. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	21	-1	9	9	39	14	-1	13
ICS-33% DPs	32	-3	9	21	39	13	-2	16
ICS-50% DPs	41	22	19	30	38	10	8	24
Architecture with multi-level VI	47	28	14	14	42	31	13	27
Architecture with HI	49	35	29	28	64	39	39	41

4.2.2.2. Area Evaluation

Table 4.19 shows the area savings per benchmark for various ICS fabric implementations. Our baseline architecture is "ICS-No-DPs". I compared every architectural option with our reference baseline architecture to obtain savings. The baseline architecture with 50% dedicated pass gates provides 24% area savings as compared to base architecture. The architectures with multi-level VI provide area savings of 27% and the architectures with HI provides 41% area savings.

4.2.2.3. Energy Evaluation

Table 4.18: Energy savings (%) per benchmark mapped onto ICS, DPs, and hybrid architectures, architecture with multi-level VI and HI. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	13	14	1	4	9	7	3	7
ICS-33% DPs	19	17	0	4	9	9	3	8
ICS-50% DPs	19	16	0	2	8	8	3	7
Architecture with multi-level VI	30	25	3	5	14	7	5	13
Architecture with HI	18	22	-1	2	9	4	2	8

Table 4.19: Percentage savings in terms of number of functional units per benchmark mapped onto ICS, DPs, and hybrid architectures, architecture with multi-level VIs and HIs. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	22	0	11	11	40	14	0	14
ICS-33% DPs	33	0	11	22	40	14	0	17
ICS-50% DPs	44	25	22	33	40	14	13	27
Architecture with multi-level VI	44	25	11	11	40	29	10	24
Architecture with HI	50	36	30	29	64	40	40	41

The energy results for different ICS fabric architectures and architecture with multi-level VI and architecture with HI are shown in Table 4.18. The ICS fabric with 33% DPs shows an energy improvement of 8% as compared to the baseline architecture(ICS-No-DPs), whereas by having multi-level VI in the architectures, we achieved energy savings of 13% and by having HIs in the architectures, we achieved energy savings of 8% as compared to the baseline (ICS-No-DPs).

4.2.2.4. Energy Evaluation

This section presents energy vs area tradeoffs for the suite of benchmarks for ICS hybrid fabric architectures, architecture with multi-level VI and HI. Our baseline architecture for various implementation is "ICS-No-DPs". We compared every architectural option with our reference baseline architecture to obtain savings. The ICS architecture with 50% dedicated pass gates provides 7% energy savings, 24% area savings and requires 27% fewer functional units as compared to its base architecture. Whereas architecture having multi-level VI provides 13% energy savings, 27% area savings and requires 24% fewer functional units as compared to the baseline architecture and architecture having HIs provides 8% energy savings, 41% area savings and requires 41% fewer functional units as compared to the baseline architecture.

Table 4.20: Fabric size (Width x Height) for mapping various benchmarks onto various ICS architectures, fully connected homogeneous and heterogeneous architectures using the heuristic mapper.

	enc	**dec**	**row**	**col**	**gsm**	**sob**	**lap**
ICS-No DPs	9x16	4x14	9x10	9x12	5x18	7x10	8x10
ICS-25% DPs	7x16	6x14	8x10	8x12	3x18	8x10	8x10
ICS-33% DPs	6x16	4x14	8x10	7x12	3x18	6x10	8x10
ICS-50% DPs	5x16	3x14	7x10	6x12	3x18	6x10	7x10
fully connected Homogeneous Architecture	8x5	8x4	8x7	8x8	8x4	8x3	8x4
fully connected Heterogeneous Architecture	8x8	6x7	8x6	8x6	6x8	7x6	8x5

4.2.3. Fully Connected Homogeneous and Heterogeneous Architectures

4.2.3.1. Fabric Sizes

Table 4.20 provides a summary of the size requirements of the seven signal and image processing benchmarks Section 4.1.1 mapped to the fabric for various ICS architectures, fully connected homogeneous and heterogeneous architectures. Results shows that the fabric size can be reduced significantly by using fully connected interconnections in the fabric as compared to using ICS and combination of ICS and DPs in the fabric. For example, "lap" implemented on ICS fabric with 50% DPs was using 7x10 fabric, the same benchmark can be implemented on 8x4 and 8x5 fabric when we use fully connected homogeneous and heterogeneous architectures respectively.

Table 4.21: Minimum Fabric size (Width x Height) for various fabric architectures.

Architecture	**Fabric size**	**Fabric size**	**Fabric size**
	ICS	fully connected homogeneous architecture	fully connected heterogeneous architecture
No DPs	9x18	8x8	8x8
25% DPs	8x18	-	-
33% DPs	8x18	-	-
50% DPs	7x18	-	-

Table 4.21 shows the final fabric sizes of the various ICS architectures, fully connected homogeneous and heterogeneous architectures. The benchmarks can be mapped onto smaller size fabric having fully connected homogeneous or heterogeneous fuctional units as compared to the ICS architectures. For example, the benchmarks implemented on ICS architecture with no DPs used 9x18 size fabric and the same set of benchmarks can be implemented on 7x18 fabric when we use 50% DPs in ICS architecture whereas benchmarks can be implemented on a 8x8 fabric with fully connected homogeneous and heterogeneous architectures which are much smaller in size as compared to base architecture.

Table 4.22: Comparison of the total number of functional units in various ICS architectures, fully connected homogeneous and heterogeneous architectures.

Architecture	**ICS**	**Homogeneous Architecture**	**% Savings**
No DPs	162	64	61

(a) fully connected homogeneous architecture.

Architecture	**ICS**	**Heterogeneous Architecture**	**% Savings**
No DPs	162	64	61

(b) fully connected heterogeneous architecture.

Table 4.22 shows the comparison between the ICS fabric size Vs fully connected homogeneous and heterogeneous architectures. Results shows, the number of functional units get reduced by 61% when we used fully connected homogeneous and heterogeneous architectures instead of base architecture (ICS-NO-DPs). In real fully connected heterogeneous architecture occupy less area as compared to fully connected homogeneous architecture because of less number of operations inside the ALUs. Because of fully connected ALUs, there is not a single ALU used as PASS-gate to route the data.

4.2.3.2. Area Evaluation

Table 4.25 shows the area savings per benchmark for various ICS architectures, fully connected homogeneous and heterogeneous architectures. Fully connected homogeneous and heterogeneous architectures shows huge area savings as compared to baseline architecture.

Table 4.23: Percentage area savings per benchmark mapped onto various ICS architectures, fully connected homogeneous and heterogeneous architectures. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	21	-1	9	9	39	14	-1	13
ICS-33% DPs	32	-3	9	21	39	13	-2	16
ICS-50% DPs	41	22	19	30	38	10	8	24
fully connected Homogeneous Architecture	57	12	4	9	45	47	38	30
fully connected Heterogeneous Architecture	65	40	56	63	58	51	61	56

The best area savings for various ICS architectures is given by ICS architecture with 50% dedicated pass gates which provides 24% area savings as compared to base architecture whereas fully connected homogeneous architecture provides 30% area savings and fully connected heterogeneous architecture provides 55% area saving as compared to base architecture.

4.2.3.3. Energy Evaluation

Table 4.24: Energy savings (%) per benchmark mapped onto various ICS architectures, fully connected homogeneous and heterogeneous architectures. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	13	14	1	4	9	7	3	7
ICS-33% DPs	19	17	0	4	9	9	3	8
ICS-50% DPs	19	16	0	2	8	8	3	7
fully connected Homogeneous Architecture	-76	-94	-40	-40	-65	-47	-128	-70
fully connected Heterogeneous Architecture	-107	-136	-2	-6	-60	-17	-95	-60

The energy results for various ICS architectures, fully connected homogeneous and heterogeneous architectures are shown in Table 4.24. The energy consumption is increased

Table 4.25: Percentage savings in terms of number of functional units per benchmark mapped onto various ICS architectures fully connected homogeneous and heterogeneous architectures. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	22	0	11	11	40	14	0	14
ICS-33% DPs	33	0	11	22	40	14	0	17
ICS-50% DPs	44	25	22	33	40	14	13	27
fully connected homogeneous architecture	77	43	38	41	64	66	60	55
fully connected heterogeneous architecture	56	25	47	56	47	40	50	46

due to huge interconnector being used here to route the data. Each benchmark is showing a penalty in terms of energy consumption, on average 60% energy consumption in fully connected heterogeneous architecture is increased as compared to baseline architecture.

4.2.3.4. Energy Evaluation

This section presents energy vs area tradeoffs for the suite of benchmarks for various ICS architectures, fully connected homogeneous and heterogeneous architectures. The ICS architecture with 33% dedicated pass gates provides 8% energy savings, 7% area savings and requires 10% fewer functional units as compared to its base architecture. Whereas fully connected homogeneous architecture provides 30% area savings and requires 55% fewer functional units with a penalty of 70% in terms of energy as compared to the baseline architecture, and fully connected heterogeneous architecture provides 56% area savings and requires 46% fewer functional units with a penalty of 60% in terms of energy as compared to the baseline architecture.

Table 4.26: Fabric size (Width x Height) for mapping various benchmarks onto various ICS architectures, three dimensional architecture using the heuristic mapper.

	enc	**dec**	**row**	**col**	**gsm**	**sob**	**lap**
ICS-No DPs	9x16	4x14	9x10	9x12	5x18	7x10	8x10
ICS-25% DPs	7x16	6x14	8x10	8x12	3x18	8x10	8x10
ICS-33% DPs	6x16	4x14	8x10	7x12	3x18	6x10	8x10
ICS-50% DPs	5x16	3x14	7x10	6x12	3x18	6x10	7x10
3D Architectures	4x4x3	4x4x3	4x4x4	4x4x4	4x4x3	4x4x2	4x4x2

4.2.4. Three Dimensional Architectures

4.2.4.1. Fabric Sizes

Table 4.26 provides a summary of the size requirements, results shows that the fabric size is reduced when we used 3D architecture instead of using ICS and combination of ICS and DPs in the fabric. For example, "lap" implemented on ICS fabric with 50% DPs was using 7x10 fabric (i.e 70 ALUs), the same benchmark can be implemented on 4x4x2 fabric (i.e.32 ALUs) when we use 3D architecture.

Table 4.27: Minimum Fabric size (Width x Height) for various fabric architectures.

Architecture	**Fabric size** ICS	**Fabric size** 3D Architecture
No DPs	9x18	4x4x4
25% DPs	8x18	-
33% DPs	8x18	-
50% DPs	7x18	-

Table 4.27 shows the final fabric sizes of the 3D architecture and various ICS architectures. Results shows that by placing ALUs closer with proper interconnect the size minimum size required to map the all benchmarks can be reduced to much smaller fabric. For example, the benchmarks implemented on ICS architecture with no DPs used 9x18 size fabric and the same set of benchmarks can be implemented on 7x18 fabric when we use 50% DPs in ICS

architecture whereas benchmarks can be implemented on a 4x4x4 3D fabric which provides 61% area savings in terms of functional units. The numbers of ALUs used as pass-gate are also reduced to zero because of enhanced interconnections of the 3D architecture.

Table 4.28: Comparison of the total number of functional units in various ICS architectures, three dimensional architectures.

Architecture	ICS	3D Architecture	% Savings
No DPs	162	64	61

4.2.4.2. Area Evaluation

Table 4.31 shows the area savings per benchmark for 3D architecture and various ICS architectures. 3D architecture shows a huge area savings as compared to baseline architectures. As per results, the best area savings for various ICS architectures is given by ICS architecture with 50% dedicated pass gates which provides 24% area savings as compared to base architecture whereas 3D architecture provides 38% area savings as compared to base architecture.

4.2.4.3. Energy Evaluation

The energy results for 3D architecture and various ICS architectures are shown in Table 4.30. On an average, ICS architecture having 33% DPs provides 8% energy improvement

Table 4.29: Percentage area savings per benchmark mapped onto various ICS architectures, three dimensional architecture. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	21	-1	9	9	39	14	-1	13
ICS-33% DPs	32	-3	9	21	39	13	-2	16
ICS-50% DPs	41	22	19	30	38	10	8	24
3D Architecture	63	5	21	34	41	49	55	38

Table 4.30: Energy savings (%) per benchmark mapped onto various ICS architectures, three dimensional architecture. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	13	14	1	4	9	7	3	7
ICS-33% DPs	19	17	0	4	9	9	3	8
ICS-50% DPs	19	16	0	2	8	8	3	7
3D Architecture	25	20	0	4	9	2	1	9

Table 4.31: Percentage savings in terms of number of functional units per benchmark mapped onto various ICS architectures, three dimensional architecture. (Negative numbers mean cost).

	enc	dec	row	col	gsm	sob	lap	average
ICS-No DPs	-	-	-	-	-	-	-	-
ICS-25% DPs	22	0	11	11	40	14	0	14
ICS-33% DPs	33	0	11	22	40	14	0	17
ICS-50% DPs	44	25	22	33	40	14	13	27
3D Architecture	67	14	29	41	47	54	60	45

as compared to base architecture whereas the 3D architecture provides 9% energy improvement as compared to base architecture.

4.2.4.4. Energy Evaluation

This section presents energy vs area tradeoffs for the suite of benchmarks for 3D architecture and various ICS architectures. The ICS architecture with 33% dedicated pass gates provides 8% energy savings, 7% area savings and requires 10% fewer functional units as compared to its base architecture. When we increase the dedicated pass gates from 33% to 50% in ICS architecture, it will provides 7% energy savings, 24% area savings and requires 27% fewer functional units as compared to base architecture whereas 3D architecture provides

9% energy savings, 38% area savings and requires 45% fewer functional units as compared to its base architecture.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this chapter, I am giving an overview and conclusion of my presented work in Section 5.1. Finally, in Section 5.2 I will discuss further improvement and studies that can be done in future.

5.1. Conclusion

The objective of Coarse-Grained Reconfigurable Architecture (CGRA) is to achieve low-power customized hardware which can support a large range of applications. This thesis presented some methodologies to optimize Coarse-Grained Reconfigurable Architecture (CGRA) in terms of area as well as energy. In stripe-based coarse grained fabrics, the typical design approach of routing inputs from the top of the fabric has a high power and area cost, as large numbers of computational units are used as pass gates and the number of idle computational units remains idle which cost in terms of area as well as energy. In my work, I had investigated a complimentary design approached to get rid of ALUs which are not used for any application. I had explored the impact of splitting, folding and new approaches on energy and area of the domain-based coarse-grained reconfigurable fabric.

Figure 5.1 shows the area consumption of different architectures implementated on Synopsys 90nm Generic Library. Figure 5.2 shows the energy consumption of different architectures implementated on Synopsys 90nm Generic Library. Figure 5.3 shows the energy vs area graph for various architectures I have discussed in this thesis. On an average, The ICS-Split architecture having ICS and 33% DPs provides 20% energy savings, 8% area savings, and requires 25% fewer functional units than the baseline ICS implementation. The ICS-Fold architecture with ICS and 50% dedicated pass gates achieves 6% energy improvement, 27% area improvement, and requires 31% fewer functional units compared to the ICS

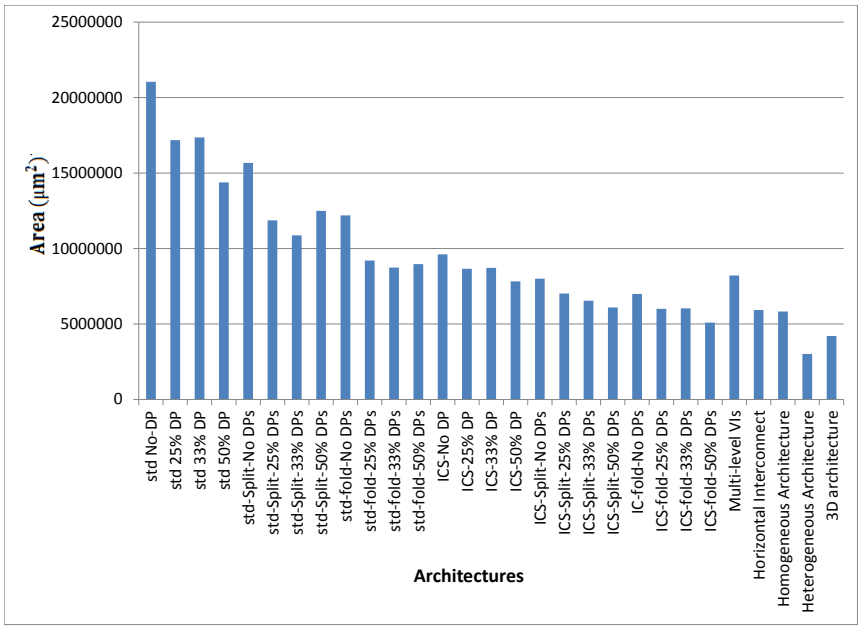


Figure 5.1: Area consumption for various fabric architectures implemented on synopsys 90nm generic library.

architecture. When I used multi-level vertical inteconnect along with ICS (inputs coming from side), I have achieved 13% energy savings, 27% area savings, whereas architecture with horizontal inteconnector provedes 8% energy savings, and 41% area savings as compared to baseline architecture. The fully connected homogeneous and heterogeneous architecture provides 30% and 56% area savings respectively with cost of 70% and 60% energy savings respectively. 3D architecture provides 9% energy improvement, 38% area improvements and requires 45% fewer functional units as compare to baseline architecture.

In terms of area, fully connected heterogeneous architecture provides best result whereas in terms of energy, split and fold architectures with 50% DPs provides best results. 3D architecture provide a good balance between energy and area.

5.2. Future Work

I have focused on overall area and energy optimization for my architectures. Unfortunately, much important information is still hidden in the proposed architectures. For

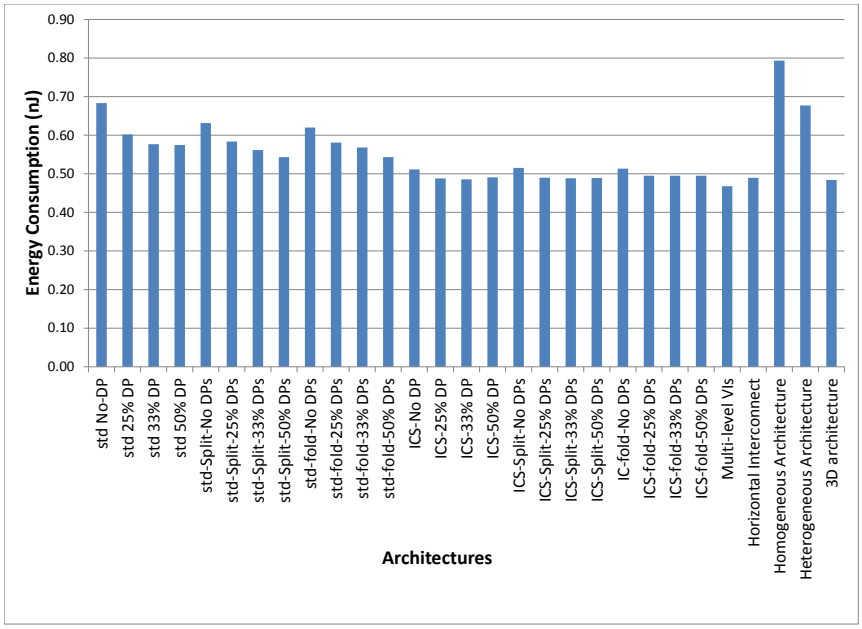


Figure 5.2: Energy consumption for various fabric architectures implemented on synopsys 90nm generic library.

example, wire lengths and the energy consumed by them. There is still room for further optimizations that can be done, for example the energy consumed by the interconnectors is much more than the energy consumed by the functional units especially in fully connected homogeneous and heterogeneous architectures. In future work, the optimizations for the interconnectors have to be done. The proposed three dimensional architecture can be improved in many ways, for example we can use the register with each ALU to store the data or results for bigger applications. We can also use the optimizations techniques like resource sharing for further improvements.

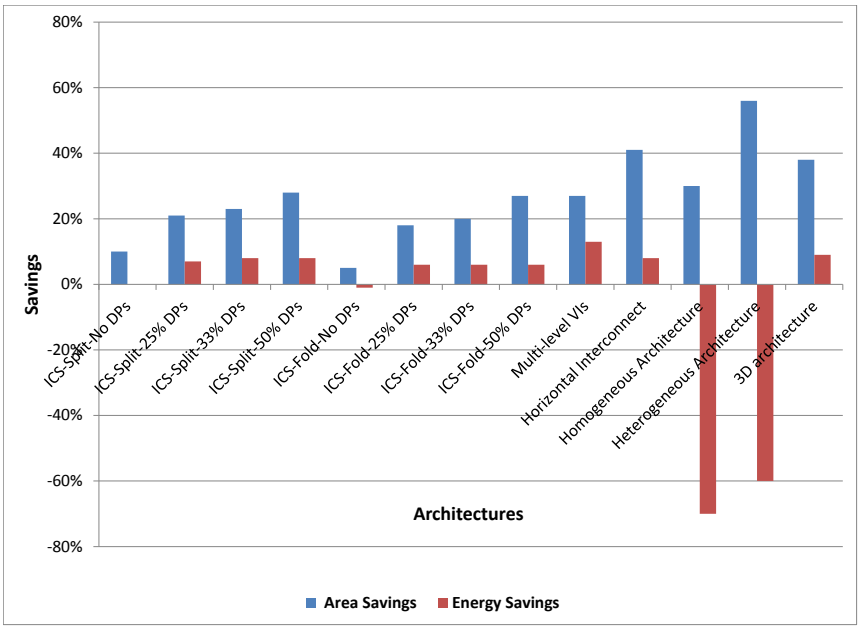


Figure 5.3: Energy Vs Area graph for various architectures as compare to ICS-No-DPs architecture.

BIBLIOGRAPHY

[1] Mirsky, E.; DeHon, A.; , "MATRIX: a reconfigurable computing architecture with configurable instruction distribution and deployable resources," FPGAs for Custom Computing Machines, 1996. Proceedings. IEEE Symposium on , vol., no., pp.157-166, 17-19 Apr 1996.

[2] Hartenstein, R.; , "The microprocessor is no longer general purpose: why future reconfigurable platforms will win," Innovative Systems in Silicon, 1997. Proceedings., Second Annual IEEE International Conference on , vol., no., pp.2-12, 8-10 Oct 1997.

[3] Alan Marshall, Tony Stansfield, Igor Kostarnov, Jean Vuillemin, and Brad Hutchings; "A reconfigurable arithmetic array for multimedia applications" In Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays (FPGA '99). ACM, New York, NY, USA, pp.135-143, 1999.

[4] Hauser, J.R.; Wawrzynek, J.; , "Garp: a MIPS processor with a reconfigurable coprocessor," FPGAs for Custom Computing Machines, 1997. Proceedings., The 5th Annual IEEE Symposium on , vol., no., pp.12-21, 16-18 Apr 1997.

[5] Singh, H.; Ming-Hau Lee; Guangming Lu; Kurdahi, F.J.; Bagherzadeh, N.; Chaves Filho, E.M.; , "MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications," Computers, IEEE Transactions on , vol.49, no.5, pp.465-481, May 2000.

[6] Singh, H.; Ming-Hau Lee; Guangming Lu; Kurdahi, F.J.; Bagherzadeh, N.; Filho, E.M.C.; , "MorphoSys: a reconfigurable architecture for multimedia applications," Integrated Circuit Design, 1998. Proceedings. XI Brazilian Symposium on , vol., no., pp.134-139, 30 Sep-3 Oct 1998.

[7] Andre Maurice Dehon. "Reconfigurable Architectures for General-Purpose Computing" Ph.D. Dissertation. Massachusetts Institute of Technology, 1996.

[8] Carl Ebeling, Darren C. Cronquist, and Paul Franklin; "RaPiD - Reconfigurable Pipelined Datapath" In Proceedings of the 6th International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers (FPL '96), Reiner W. Hartenstein and Manfred Glesner (Eds.). Springer-Verlag, London, UK, pp.126-135, 1996.

[9] Frank Bouwens, Mladen Berekovic, Andreas Kanstein, and Georgi Gaydadjiev. 2007. Architectural exploration of the ADRES coarse-grained reconfigurable array. In Proceedings of the 3rd international conference on Reconfigurable computing: architectures, tools and applications (ARC'07), Springer, pp.1-13, 2007

[10] Ebeling, C.; Cronquist, D.C.; Franklin, P.; Secosky, J.; Berg, S.G.; , "Mapping applications to the RaPiD configurable architecture," FPGAs for Custom Computing Machines, 1997. Proceedings., The 5th Annual IEEE Symposium on , vol., no., pp.106-115, 16-18 Apr 1997.

[11] Levine, B.; Schmit, H.; , "PipeRench: Power & Performance Evaluation of a Programmable Pipelined Datapath," presented at Hot Chips 14, Palo Alto, CA, Aug 2002.

[12] Yoonjin Kim; Mahapatra, R.N.; , "A New Array Fabric for Coarse-Grained Reconfigurable Architecture," Digital System Design Architectures, Methods and Tools, 2008. DSD '08. 11th EUROMICRO Conference on , vol., no., pp.584-591, 3-5 Sept. 2008.

[13] Schmit, H.; Whelihan, D.; Tsai, A.; Moe, M.; Levine, B.; Reed Taylor, R.; , "PipeRench: A virtualized programmable datapath in 0.18 micron technology," Custom Integrated Circuits Conference, 2002. Proceedings of the IEEE 2002 , vol., no., pp. 63- 66, 2002.

[14] Levine, B.; Schmit, H.; , "Implementation of Target Recognition Applications using Pipelined Reconfigurable Hardware," In Proceedings of Military and Aerospace Applications of Programmable Devices and Technologies International Conference,2003.

[15] Mehta, G.; Ihrig, C.J.; Jones, A.K.; , "Reducing energy by exploring heterogeneity in a coarse-grain fabric," Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on , vol., no., pp.1-8, 14-18 April 2008.

[16] G. Mehta, J. Stander, J.M. Lucas, R.R. Hoare, B. Hunsaker, and A.K. Jones, "A Low-Energy Reconfigurable Fabric for the SuperCISC Architecture", presented at J. Low Power Electronics, 2006, pp.148-164.

[17] Chulsoo Park; Yoonjin Kim; Kiyoung Choi; "DOMAIN-SPECIFIC OPTIMIZATION OF RECONFIGURABLE ARRAY ARCHITECTURE".

[18] Heysters, P.M. Coarse-Grained Reconfigurable Computing for Power Aware Applications. In Proceedings of ERSA. pp.272-272, 2006.

[19] Yoonjin Kim; Mahapatra, R.N.; Kiyoung Choi; , "Design Space Exploration for Efficient Resource Utilization in Coarse-Grained Reconfigurable Architecture," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.18, no.10, pp.1471-1482, Oct. 2010

[20] Cao Liang; Xinming Huang; , "Mapping Parallel FFT Algorithm onto SmartCell Coarse-Grained Reconfigurable Architecture," Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE International Conference on , vol., no., pp.231-234, 7-9 July 2009.

[21] G. Mehta, J. Stander, M. Baz, B. Hunsaker, and A.K. Jones, "Interconnect customization for a hardware fabric", presented at ACM Trans. Design Autom. Electr. Syst., 2009.

[22] Heysters, P.M.; Smit, G.J.M.; , "Mapping of DSP algorithms on the MONTIUM architecture," Parallel and Distributed Processing Symposium, 2003. Proceedings. International , vol., no., pp. 6 pp., 22-26 April 2003.

[23] Mehta, G.; Hoare, R.R.; Stander, J.; Jones, A.K.; , "Design space exploration for low-power reconfigurable fabrics," Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International , vol., no., pp.4 pp., 25-29 April 2006.

[24] Cao Liang; Xinming Huang; , "SmartCell: A power-efficient reconfigurable architecture for data streaming applications," Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on , vol., no., pp.257-262, 8-10 Oct. 2008.

[25] Van Essen, B.; Panda, R.; Wood, A.; Ebeling, C.; Hauck, S.; , "Managing Short-Lived and Long-Lived Values in Coarse-Grained Reconfigurable Arrays," Field Programmable Logic and Applications (FPL), 2010 International Conference on , vol., no., pp.380-387, Aug. 31 2010-Sept. 2 2010.

[26] Reiner W. Hartenstein, Thomas Hoffmann, and Ulrich Nadeldinger; "Design-Space Exploration of Low Power Coarse Grained Reconfigurable Datapath Array Architectures". In Proceedings of the 10th International Workshop on Integrated Circuit Design, Power and Timing Modeling, Optimization and Simulation (PATMOS '00), Dimitrios Soudris, Peter Prisch, and Erich Barke (Eds.). Springer-Verlag, London, UK, pp.118-128, 2000.

[27] Aggarwal, A.A.; Lewis, D.M.; , "Routing architectures for hierarchical field programmable gate arrays," Computer Design: VLSI in Computers and Processors, 1994. ICCD '94. Proceedings., IEEE International Conference on , vol., no., pp.475-478, 10-12 Oct 1994.

[28] Kaviani, A.; Vranesic, D.; Brown, S.; , "Computational field programmable architecture," Custom Integrated Circuits Conference, 1998. Proceedings of the IEEE 1998 , vol., no., pp.261-264, 11-14 May 1998.

[29] Yoonjin Kim; Kiemb, M.; Park, C.; Jinyong Jung; Kiyoung Choi; , "Resource sharing and pipelining in coarse-grained reconfigurable architecture for domain-specific optimization," Design, Automation and Test in Europe, 2005. Proceedings , vol., no., pp. 12- 17 Vol. 1, 7-11 March 2005

[30] Rapport, Inc , "Kilocore," website: `http://www.rapportincorporated.com`.

[31] Beyne, E.; , "3D interconnection and packaging: impending reality or still a dream?," Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International , vol., no., pp. 138- 139 Vol.1, 15-19 Feb. 2004.

[32] Rahman, A.; Das, S.; Chandrakasan, A.P.; Reif, R.; , "Wiring requirement and three-dimensional integration technology for field programmable gate arrays," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.11, no.1, pp.44-54, Feb. 2003

[33] Michael J. Alexander , James P. Cohoon , Jared L. Colflesh , John Karro , Edward L. Peters , Gabriel Robins "Placement and Routing for Three-Dimensional FPGAs", 1996.