A STUDY OF ENERGY-AREA TRADEOFFS OF VARIOUS ARCHITECTURAL STYLES FOR ROUTING INPUTS IN A DOMAIN SPECIFIC RECONFIGURABLE FABRIC

Anil Yadav[1], Justin Stander[2], Alex K. Jones[2], and Gayatri Mehta[1]

[1]Department of Electrical Engineering, University of North Texas, Denton, TX, USA
[2]Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA, USA
gayatri.mehta@unt.edu

ABSTRACT

Coarse-grained reconfigurable fabrics (CGRF's) have great promise for achieving low-energy flexible designs for an application domain. However a universally accepted architecture for coarse-grained reconfigurable fabrics has not yet crystallized, and many architectural options are still un- der consideration by the research and industry community. One scientific question is how to efficiently route inputs through a CGRF. This paper addresses this question in part by exploring various alternative input solu- tions for a stripe-based fabric. Alternative architectural styles examined in this paper include (i) integrated constants (IC) approach where constants are loaded in the registers local to the functional units; (ii) inputs coming from the side (ICS) where both constants and variable inputs can be routed to the stripe directly where needed; (iii) ICS with extended vertical interconnect (ICS-EV); and (iv) a combination of dedicated pass gates (DPs) with standard, IC, ICS, and ICS-EV architecture styles. We implemented these architecture styles using 90 nm ASIC process from Synopsys. We perform a detailed area and energy analysis on these architectures and present quantitative results in this paper. We observed that the fabric with ICS and 50% DPs is the best among these options, providing 31% energy savings and 62% area savings over a baseline architecture for our benchmark set.

KEYWORDS

Reconfigurable computing, domain-specific architecture, reconfigurable architecture, coarse-grained fabric

1. INTRODUCTION

Reconfigurable devices mitigate many of the problems encountered with the development of Application Specific Integrated Circuits (ASICs) for hardware acceleration. For example, reconfigurable devices amortize the rapidly increasing mask and non-recurring engineering (NRE) costs over many more generic devices. Computer Aided Design (CAD) flows are often simplified for these de- vices. Thus, the design cycle is much reduced, which can significantly decrease the time to market.

The tradeoff for using these reconfigurable devices is a compromise in performance and most notably power/energy consumption. To reduce the energy consumption of a reconfigurable device, particular care must be given to designing both functional units and interconnect of the device.

Stripe-based fabrics in particular (e.g., see Figure 2) are quite promising due to their good fit to a data flow graph structure [5, 25, 8, 9]. When a data flow graph is mapped to a stripe-style structure, however, data dependency edges often traverse multiple rows. Mapping of a data flow graph onto a reconfigurable fabric is described in detail in Section 3.1. In these fabrics,

DOI : 10.5121/vlsic.2013.4107

arithmetic and logic units (ALUs) must often pass these values through without doing any computation. In other words, the ALU's function merely as pass-gates. It was observed for some of the signal and image processing applications, for example, that more than 50% of the functional units in the fabric were used for routing by configuring the ALU as a pass-gate as shown in Figure 1 [8].

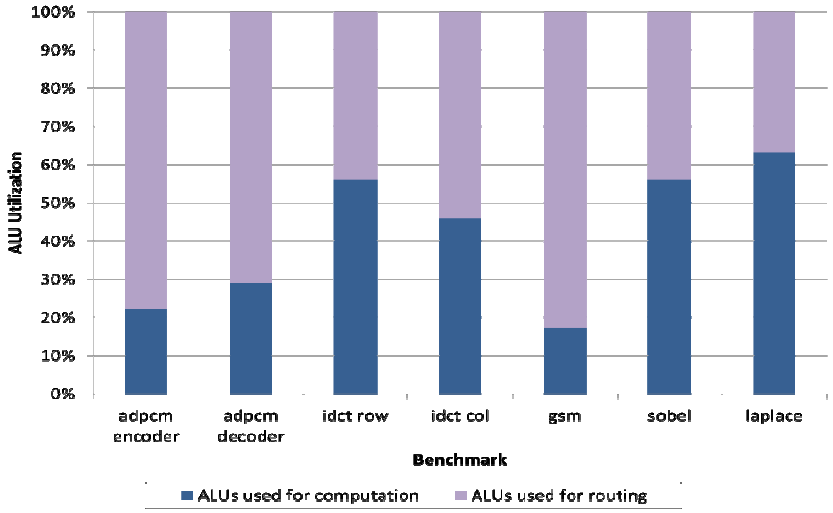

Figure 1: Comparison of ALUs used for routing and computation

However, these ALUs used as passgates are an area-inefficient and power- inefficient method for vertical routing. One alternative that has been studied is to use a simple routing structure that could only pass a value, i.e., a dedicated pass-gate. Using an ALU as a passgate requires over an order of magnitude more power than such a direct vertical route implementation. Previous research has found, for example, that an architecture that adds 50% DPs to an existing fabric provides 19% energy savings and 30% area savings [8].

There are a variety of possible ways to route inputs in the coarse-grained fabrics. It is not obvious which approach is better because additional hardware must be configured for some, we must understand how well it is utilized. To better understand the tradeoffs, we present a quantitative study of different architectures described briefly as follows. In this paper, we study (i) integrated constants (IC) approach where constants are loaded in the registers local to the functional units; (ii) inputs coming from the side (ICS) where both constants and variable inputs can be routed to the stripe directly where needed; (iii) ICS with extended vertical interconnect (ICS-EV); and (iv) a combination of dedicated pass gates (DPs) with standard, IC, ICS, and ICS-EV architecture styles. In the standard implementation, inputs are routed from the top of the fabric and functional units are used for passing information. This leads to the inefficient resource utilization because functional units that could have been used for actual computations are being used for pass operation. Since there are many inputs that stay constant during the execution cycle, they can be loaded to the registers local to the functional units. This approach will use additional registers for loading constants but can save some functional units for being used only for passing information. It provides 22% area savings and 13% energy savings on an average over the baseline. In the ICS architecture, we introduce small multiplexers to the inputs of each functional unit to provide flexibility to read inputs from the top row or directly from outside the fabric. We find that the use of such multiplexers allows substantial area savings through allowing smaller fabrics to carry the same benchmark suites. The small additional power and energy cost of the additional hardware is recovered easily through the fact that the overall fabric is smaller and fewer functional units are used as pass gates. This approach achieves 51% area savings and 27% energy savings over the

baseline. We extended the ICS approach by introducing multi-level vertical interconnect in the fabric. Now the functional unit can not only reach the functional units in the row above but can also reach the functional units in the grand-parent and great-grand parent rows in the same column. We use bigger multiplexers as compared to the ICS approach to provide that reachability but now we can implement the same benchmarks on even smaller fabrics. It provides 60% area savings and 27% energy savings over the baseline. In addition to these, we also studied the combination of adding dedicated vertical routes to the standard, IC, ICS, and ICS-EV techniques. Adding dedicated pass gates to these architectural options further increase the area and energy savings.

While our technique applies to stripe-based reconfigurable fabrics in general such as PipeRench ([13, 14]) and Kilocore ([4]), and conceptually to the larger class of coarse-grained reconfigurable fabrics, our technique is demonstrated using the low-energy domain specific fabric (DSF) target ([5]) shown in Figure 2.

The remainder of this paper is organized as follows: Section 2 provides some background material in the area of reconfigurable computing and coarse-grain architectures in general. An overview of the fabric target used in this paper to demonstrate the impact of the inputs coming from the side is presented in Section3. Section 4 includes results and an analysis of energy consumption for a suite of benchmark circuits. Section 5 discusses conclusions.

2. BACKGROUND AND LITERATURE REVIEW

A tremendous amount of effort has been devoted to the area of reconfigurable computing for application acceleration with custom hardware. While FPGAs are the most commonly used general purpose reconfigurable devices, they exhibit poor power characteristics.

Recently, the development and use of coarse-grained fabrics for computation- ally complex tasks has received a lot of attention as a possible alternative to FP- GAs. Many architectures have been proposed and developed both in academia and industry during the last two decades such as MATRIX ([3]), Garp ([23]), MorphoSys ([24], [26], [6]), RaPiD ([7], [15]), PipeRench ([13], [14], [16]), HF- PGA ([10]), Kilocore ([4]), Pact XPP ([29]), CFPA ([11]), Montium ([12],[17]), ADRES ([18]), SmartCell ([19], [1]), and the coarse-grained architectures devel- oped by [2],[27].

MATRIX (Multiple ALU architecture with Reconfigurable Interconnect eX- periment) [3] is comprised of a two-dimensional array of identical 8-bit functional units with a configurable network. Each functional unit consists of a 256x8-bit memory, an 8-bit ALU and a control logic. The Garp [23], the Chimaera [28], the MorphoSys [24], and the SuperCISC [20] architectures combine a reconfigurable computing device with a processor in order to do hardware acceleration. RaPiD (Reconfigurable Pipelined Datapath) [7, 15], mainly intended for computation- intensive applications, consists of a linear array of application-specific functional units. PipeRench [13, 14], Kilocore ([4]) have a striped configuration and is comprised of an interconnected network of configurable logic blocks and storage elements. It consists of a set of physical pipeline stages called stripes and each stripe contains a set of processing elements, register files, and an interconnec- tion network. The CFPA (Computational Field Programmable Architecture) [11] consists of Partial Add, Subtract, and Multiply (PASM) blocks for im- plementing data path operations of computational intensive applications. The PASM block operates on 4-bit operands and can be connected together to im- plement adders, subtracters, and multipliers of various sizes. The HFPGA (Hi- erarchical Field Programmable Gate Array) [10] allows the creation of coarse grain blocks built from traditional 4-input lookup tables. These coarse grain blocks have dedicated routing channels. ADRES ([18]) implemented and eval- uated several inter-connection topologies that includes simple mesh and more complex schemes, where one functional unit can transmit data to non-adjacent functional units in the same row or non-

adjacent functional units in the same column. Pact XPP Technologies [21] proposed the XPP architecture, which has a hierarchical array of coarse-grained adaptive computing elements called Processing Array Elements (PAEs) and a packet-oriented communication net- work. An XPP core is comprised of a rectangular array of ALU-PAEs and RAM-PAEs with I/O. These reconfigurable fabric architectures have sequential structure and use local registers or shared register files for storing data values. Of these, PipeRench and Kilocore are stripe-based coarse grain fabrics. These fabrics used pass register files to manage constants and pass computed values from one stripe to the other. [22] describes how to manage short-lived and long- lived values in coarse-grained fabrics. They discuss 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. In this paper, we present a detailed energy and area analysis of various architectural techniques including integrated constants, inputs coming form the side, the hybrid of IC and ICS approach with extended vertical interconnect and the combination of dedicated pass gates with standard, IC and ICS. Dedicated pass gates are also incorporated to reduce the usage of functional units as pass gates to pass computed values from one stripe (producer) to another stripe (consumer) (especially when the consumer is separated by multiple stripes from the producer).

In our previous research, we studied 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 [5, 25, 8, 9]. We attempted to minimize the cardinality of the interconnect and the number of operations supported by each ALU, and maximize the use of dedicated pass gates in the fabric. We observed that even with all of the optimizations a very large number of ALUs as pass-gates remain and results appear to be area-inefficient, which motivates the idea of exploring alternative approaches in this paper. To our knowledge, no one has yet presented a sys- tematic exploration of input routing alternatives as considered in this paper.

3. DOMAIN SPECIFIC FABRIC OVERVIEW

Stripe-based hardware fabrics are designed to easily map data flow graphs (DFGs) from the application onto the device. We illustrate our results by modifying the domain specific fabric architecture shown in Figure 2, although a similar approach could be used for other stripe-based architectures.

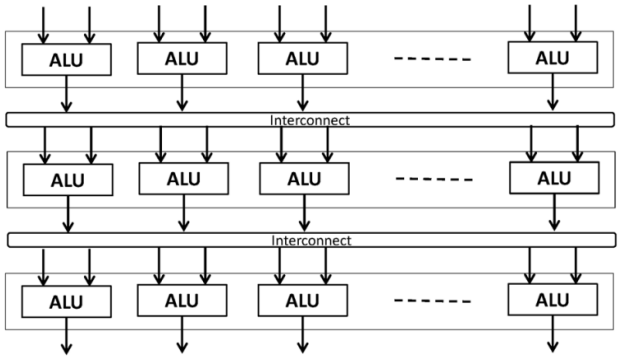

Figure 2: The fabric model is comprised of ALUs and a reconfigurable interconnect

For our examples, ALUs are organized into rows or *computational stripes* within which each functional unit operates independently. The results of these ALU operations are then fed into *interconnection stripes* constructed using multiplexers.

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 in each computational stripe. H determines the number of computational and interconnection stripes in the fabric model shown in Figure 2. The fabric architecture also has several early exit rows, spaced evenly in the device. For example, for a fabric with height 18, every alternate row is connected to the exit row. As soon as the output is computed, it can be sent to the nearest exit row which is connected to the final output of the device. If the output is available in row 9, it will go the nearest exit row 10 and then go the final output of the device. This saves a significant number of functional units in the successive rows being used to pass outputs down the rows.

3.1. Mapping of applications onto domain-specific reconfigurable fabric

A mapping of a data flow graph (DFG) onto a reconfigurable fabric consists of an assignment of operators in the DFG to ALUs in the reconfigurable fabric such that the logical structure of the DFG is preserved and the architectural constraints of the fabric are followed. This mapping problem is very critical to the use of the fabric because a mapping solution must be available each time the fabric is reprogrammed for a specific DFG. Because of the layered nature of the fabric, the mapping is also allowed to use ALUs as *pass-gates*, which take a single input and pass the input value to one or more outputs. In general, not all of the available ALUs and edges will be used. An example DFG and a corresponding mapping are shown in Figure 3 and Figure 4. The DFG from Figure 3 is implemented on a baseline architecture where inputs and constants are routed from the top of the fabric. 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 inputs and outputs are shown in white colored ovals. Consider an ALU in row 11 and column 10 i.e. ALU (11,10), shown in yellow color, one of its inputs is a constant and is being routed all the way from the top of the fabric. It uses 10 ALUs for just passing this input to the desired location. Obviously, routing alternatives for passing input values are needed.

This DFG has two outputs, one of which is computed and available very early in the 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.

3.2. Architectural exploration case studies

In order to conduct architectural exploration case studies, we selected a set of core signal processing benchmarks from MediaBench benchmark suite includ- ing the ADPCM encoder (enc), ADPCM decoder (dec), GSM channel encoder (gsm), and the MPEG II decoder (row, col). We added the Sobel (sob) and Laplace (lap) edge detection algorithms to the benchmark suite. We computed the number of operations and number of constants in each benchmark. Table 1 shows the number of operations and the number of constants contained in the 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 seen that a large of functional units are being wasted for routing inputs and constants. For example, in "enc", 105 pass gates are used to route only 3 inputs and 14 constants.

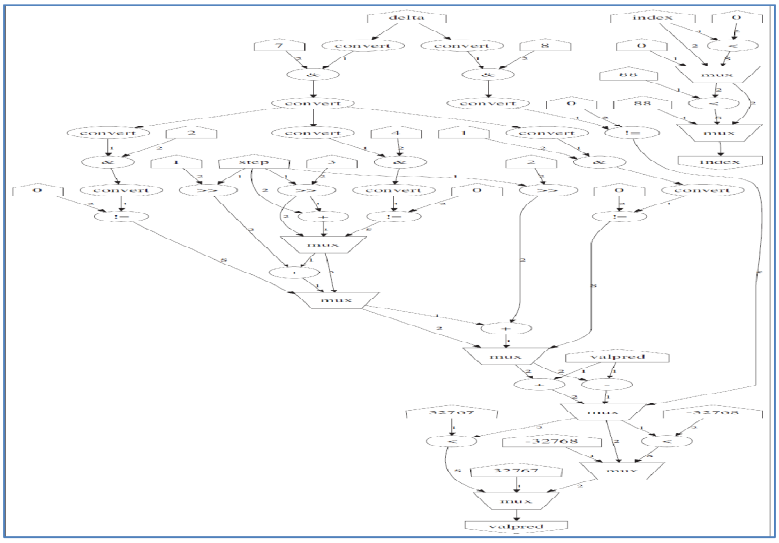

Figure 3: An example of a data flow graph (DFG).

3.3. Fabric architecture with dedicated pass gates (DP)

In order to reduce power consumption due to large numbers of ALUs being used as pass gates, the use of dedicated pass gates, which simply route data vertically from one row to the next have been explored ([8]). The dedicated pass gate can also be set to idle state when not being used. Figure 5 shows the data flow graph (DFG) from Figure 3 mapped onto the architecture with 33% DPs (1 out of 3). 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. Our goal here 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.

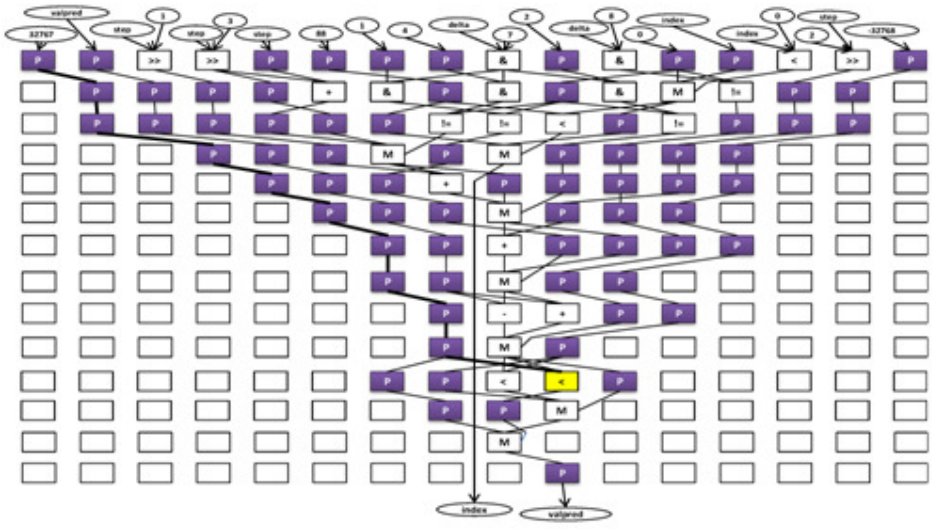

Figure 4: Example mapping of the DFG in Figure 3 onto a stripe-based coarse-grained fabric.

Table 1: Number of operations, constants, inputs, pass gates in DFGs 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	105	31	23	36	84	11	13

3.4. Fabric architecture with integrated constants (IC)

To implement the Integrated Constants (IC) architecture, we used a register to store a constant and a 2:1 multiplexer for each operand of an ALU as shown in Figure 6. Each multiplexer can take inputs from the stripe above and from a register. The first stripe of ALUs in the fabric architecture takes variable inputs from the top and constant inputs from the registers; the ALUs in the rest of the stripes can get their operands either from the predecessor stripe or from the register.

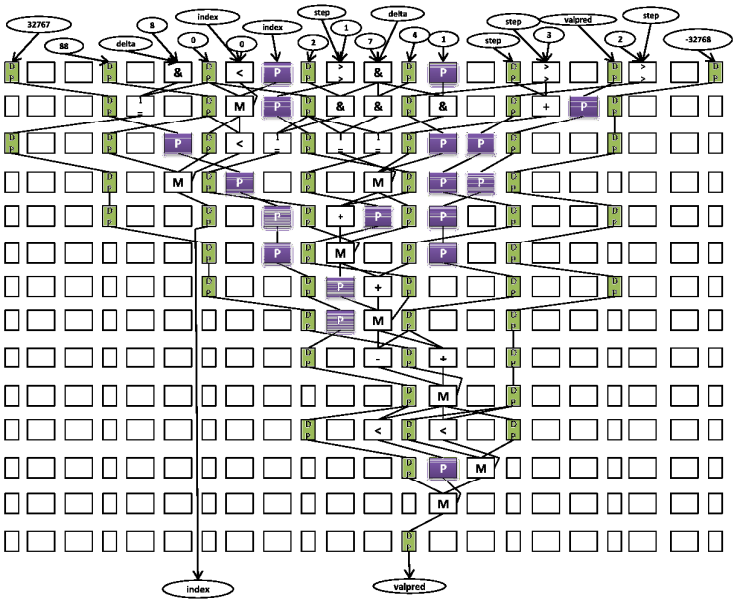

Figure 5: A DFG shown in Figure 3 mapped on the architecture with 33% DPs.

Figure 7 shows the DFG shown in Figure 3 mapped onto the architecture where constants are routed directly to the functional units where needed using registers. In order to keep the figures simple, we show the constants integrated inside the ALUs and variables are 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 16x14 standard fabric is using only 13x14 fabric with IC. It requires 19% fewer functional units to implement the same DFG onto the fabric with IC than the standard implementation.

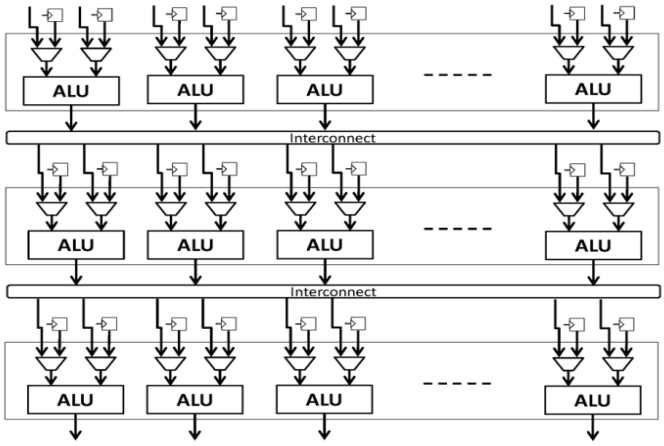

Figure 6: Fabric with Integrated Constants (IC).

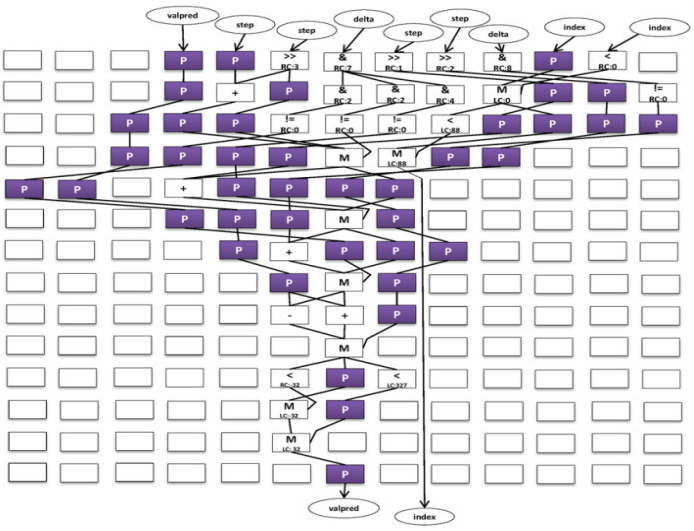

Figure 7: A DFG shown in Figure 3 mapped on the architecture with Integrated Constants (IC).

3.5. Fabric architecture with inputs coming from side (ICS)

To implement the ICS architecture, we used a 2:1 multiplexer for each operand of an ALU as shown in Figure 8. Each multiplexer can take inputs from the stripe above and from the side. The first stripe of ALUs in the fabric architecture takes all inputs from the top. No multiplexers are needed for the first ALU stripe. The ALUs in the rest of the stripes can get their operands either from the predecessor stripe or from the side. Each stripe has two busses, one for the left operand and one for the right operand. Inputs are stacked in a single multi-bit signal that is sent along the bus, and required inputs are selected from this value by the left or right multiplexer.

Figure 9 shows the DFG shown in Figure 3 mapped onto the architecture where inputs a constants are routed directly to the functional units where needed. The same graph which used 16x14 standard fabric is using only 4x14 fabric with ICS. It requires 75% fewer functional units to implement the same DFG onto the fabric with ICS than the baseline.

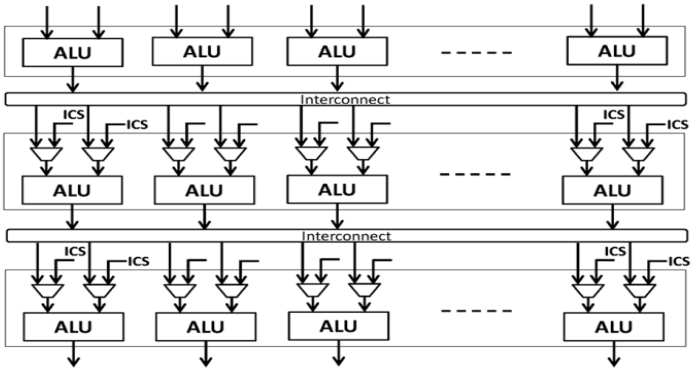

Figure 8: Fabric with Inputs Coming from Side (ICS).

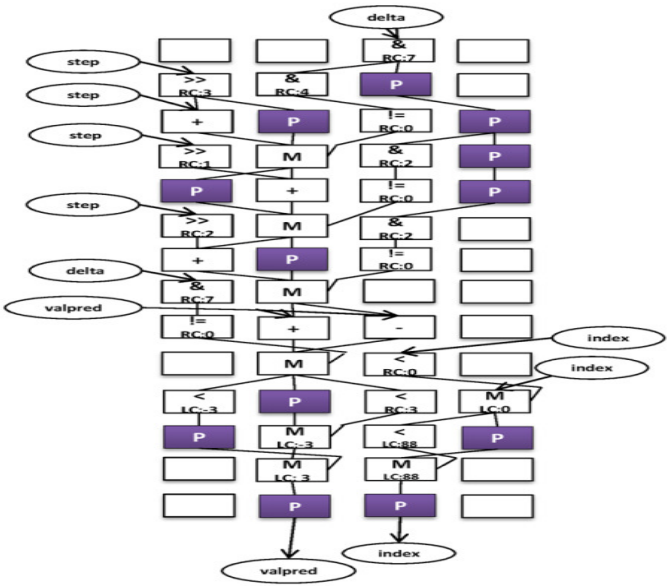

Figure 9: A DFG shown in Figure 3 mapped on the architecture with ICS.

3.6. Fabric architecture with ICS with extended vertical interconnect (ICS-EV)

To implement this architecture, we used a 4:1 multiplexer for each operand of an ALU as shown in Figure 10. Each operand can come either from the stripe above, the grandparent stripe ALU(same column), the great grandparent stripe ALU(same column), or from the

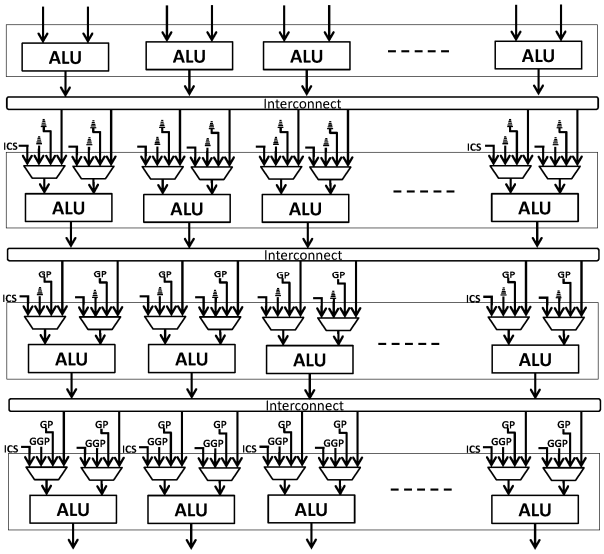

Figure 10: Fabric with IC and ICS with extended vertical interconnect (ICS-EV).

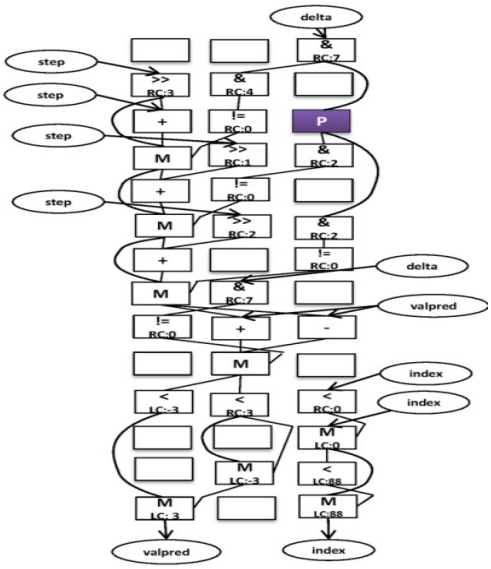

Figure 11: A DFG shown in Figure 3 mapped on the ICS-EV architecture.

side and the 4:1 multiplexer provides this flexibility and reachability. The first stripe of ALUs in the fabric architecture takes variable inputs from the top and constant inputs from the registers. Each stripe has two busses, one for the left operand and one for the right operand. Inputs are stacked in a single multi-bit signal that is sent along the bus, and required inputs are selected from this value by the left or right multiplexer.

Figure 11 shows the DFG shown in Figure 3 mapped onto the architecture where inputs are constants are routed directly to the functional units where needed. The same graph which used 16x14 standard fabric is using only 3x14 fabric with hybrid approach. It requires 81% fewer functional units to implement the same DFG onto this new architecture compared to the baseline.

3.7. Fabric architecture with ICS with horizontal interconnect (ICS-HI)

To implement this architecture, we used a 4:1 multiplexer for each operand of an ALU as shown in Figure 12. Each operand can come either from the stripe above, the left ALU (same row), the right ALU(same row), or from the side and the 4:1 multiplexer provides this flexibility and reachability. The first stripe of ALUs in the fabric architecture takes variable inputs and constants from the top and results from the neighbor ALUs. Each stripe has two busses, one for the left operand and one for the right operand. Inputs are stacked in a single multi-bit signal that is sent along the bus, and required inputs are selected from this value by the left or right mulitplexer.

Figure 13 shows the DFG shown in Figure 3 mapped onto the architecture where inputs and constants are routed directly to the functional units where needed. The same graph which used 16x14 standard fabric is using only 6x6 fabric with hybrid approach. It requires 84% fewer functional units to implement the same DFG onto this new architecture compared to the baseline.

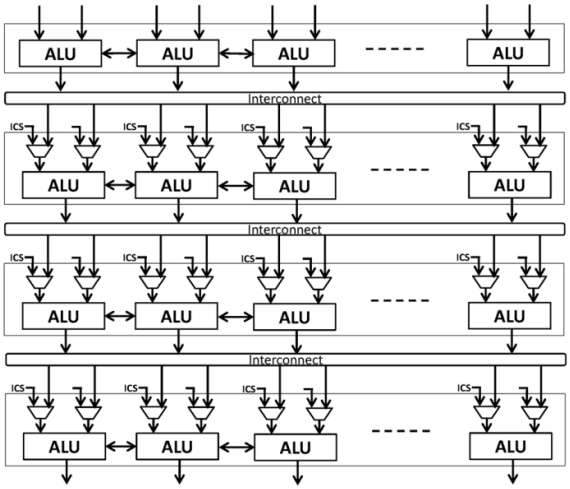

Figure 12: Fabric with IC and ICS with horizontal interconnect (ICS-HI).

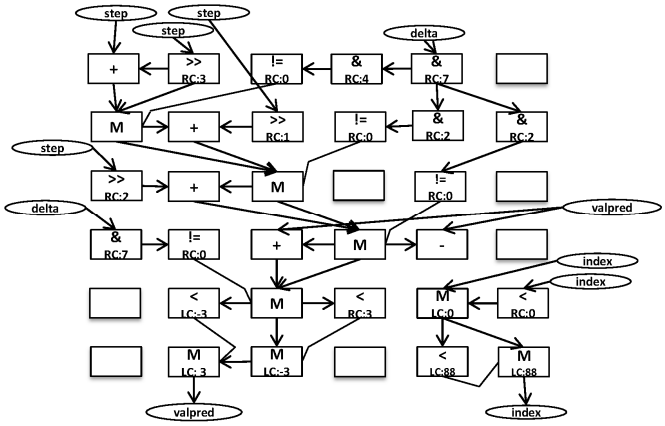

Figure 13: A DFG shown in Figure 3 mapped on the ICS-HI architecture.

4. RESULTS

We performed detailed area and energy analysis on various architectural options including standard, IC, ICS, ICS-EV, and a combination of dedicated pass gates with these approaches.

Table 2 provides a summary of the size requirements of the seven signal and image processing benchmarks mentioned in Section 3.2 mapped to various fabric architecture styles. The fabric size is given by Width x Height. When we compare the various architecture alternatives with the baseline, the benchmarks can fit in smaller width fabric. The benchmarks with more number of constants such as "enc", "dec", "col", and "gsm" show large area improvements. For example, "gsm" implemented on standard fabric with 33% DPs was using 16-wide fabric whereas the same benchmark when implemented on the fabric with ICS takes only 3-wide fabric.

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. The benchmarks can be mapped onto smaller size fabric for ICS architectures as compared to the

Table 2: Fabric size (Width x Height) for mapping benchmarks onto various fabric architectures.

	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
IC-No DPs	12x16	13x14	12x10	13x12	11x18	8x10	13x8
IC-25% DPs	9x16	9x14	12x10	12x12	11x18	8x10	13x8
IC-33% DPs	11x16	10x14	12x10	11x12	9x18	8x10	13x8
IC-50% DPs	9x18	7x14	11x12	11x16	7x18	8x10	13x10
ICS-No DPs	9x16	4x14	9x10	9x12	5x18	7x10	8 x10
ICS-25% DPs	7x16	4x14	8x10	8x12	3x18	6x10	8x10
ICS-33% DPs	6x16	4x14	8x10	7x12	3x18	6x10	8x10
ICS-50% DPs	5x16	3x14	7x10	6x12	3x18	6x10	7x10
ICS-EV-No DPs	5x16	3x14	8x10	7x12	3x18	5x10	6x12
ICS-EV-25% DPs	4x16	3x14	8x10	7x12	3x18	5x10	6x12
ICS-HI-No DPs	8x9	6x6	8x10	9x7	11x7	4x8	8x6
ICS-HI-25% DPs	7x9	6x6	8x10	9x7	11x7	4x8	8x6

Table 3: Area (in terms of number of functional units) for mapping benchmarks onto several fabric architectures.

	enc	dec	row	col	gsm	sob	lap
std-No DPs	272	224	180	240	324	100	120
std-25% DPs	256	210	130	180	270	90	104
std-33% DPs	256	196	156	192	288	100	112
std-50% DPs	160	140	132	216	198	80	130
IC-No DPs	192	182	120	156	198	80	104
IC-25% DPs	144	126	120	144	198	80	104
IC-33% DPs	176	140	120	132	162	80	104
IC-50% DPs	180	98	132	176	126	80	130
ICS-No DPs	144	56	90	108	90	70	80
ICS-25% DPs	112	56	80	96	54	60	80
ICS-33% DPs	96	56	80	84	54	60	80
ICS-50% DPs	80	42	70	72	54	60	70
ICS-EV-No DPs	80	42	80	84	54	50	72
ICS-EV-25% DPs	64	42	80	84	54	50	72
ICS-HI-No DPs	72	36	63	77	32	42	48
ICS-HI-25% DPs	63	36	63	77	32	42	48

standard architectures as shown in Table 4, 5, 6 and 7. For example, the benchmarks implemented on standard architecture with no DPs used 20x18 size fabric whereas the same set of benchmarks can now be implemented on 9x18 fabric with ICS.

Table 4: Minimum Fabric size (Width x Height) for combination of IC and DP.

Architecture	Fabric size(std)	Fabric size(IC)	% Savings
No DPs	20x18	13x18	35
25% DPs	16x18	13x18	19
33% DPs	16x18	13x18	19
50% DPs	13x18	13x18	0

Table 5: Minimum Fabric size (Width x Height) for ICS with DP.

Architecture	Fabric size(std)	Fabric size(ICS)	% Savings
No DPs	20x18	9x18	55
25% DPs	16x18	8x18	50
33% DPs	16x18	8x18	50
50% DPs	13x18	7x18	46

Table 6: Minimum Fabric size (Width x Height) for ICS-EV.

Architecture	Fabric size (std)	Fabric size (ICS-EV)	% Savings
No DPs	20x18	8x18	55
25% DPs	16x18	8x18	50

Table 7: Minimum Fabric size (Width x Height) for ICS-HI.

Architecture	Fabric size (std)	Fabric size (ICS-HI)	% Savings
No DPs	20x18	11x9	73
25% DPs	16x18	11x9	66

Table 8 shows the percentage savings in terms of number of functional units per benchmark mapped onto standard, IC, ICS, and hybrid architectures. We computed the number of functional units required to map each benchmark for a particular architecture. We then compared every architectural option with our reference baseline architecture to obtain savings. The IC architecture requires 27% fewer functional units compared to the baseline. The ICS architecture provides savings of 52% in terms of functional units compared to the standard architecture. The ICS-EV architecture requires 62% fewer functional units than the baseline architecture. The combination of ICS and 50% DPs needs 64% fewer functional units as compared to the baseline.

Table 8: Percentage area savings in terms of number of functional units per benchmark mapped onto standard, DP, ICS and hybrid architectures.

	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
IC-No DPs	29	19	33	35	39	20	13	27
IC-25% DPs	47	44	33	40	39	20	13	34
IC-33% DPs	35	38	33	45	50	20	13	33
IC-50% DPs	47	56	27	27	61	20	-8	31
ICS-No DPs	47	75	50	55	72	30	33	52
ICS-25% DPs	59	75	56	60	83	40	33	58
ICS-33% DPs	65	75	56	65	83	40	33	60
ICS-50% DPs	71	81	61	70	83	40	42	64
ICS-EV-No DPs	71	81	56	65	83	50	40	64
ICS-EV-25% DPs	76	81	56	65	83	50	40	65
ICS-HI-No DPs	74	84	65	68	90	58	60	71
ICS-HI-25% DPs	77	84	65	68	90	58	60	72

Table 9: Number of ALUs used as pass gates in standard (std), integrated constants (IC), inputs coming from the side (ICS) and combination of DPs with std, IC, and ICS 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	5	2
std-33% DPs	40	19	15	15	46	2	1
std-50% DPs	21	5	10	8	25	3	2
IC-No DPs	67	40	18	28	67	8	4
IC-25% DPs	34	15	6	14	20	3	2
IC-33% DPs	20	6	6	5	9	1	0
IC-50% DPs	26	9	9	8	13	0	1
ICS-No DPs	21	12	4	4	9	5	5
ICS-25% DPs	6	3	0	0	0	0	0
ICS-33% DPs	2	0	0	0	0	0	0
ICS-50% DPs	0	0	0	0	0	0	0
ICS-EV-No DPs	3	1	1	1	0	0	2
ICS-EV-25% DPs	0	0	0	0	0	0	0
ICS-HI-No DPs	5	0	3	3	0	3	0
ICS-HI-25% DPs	0	0	0	0	0	0	0

Using the parameterized fabric model described in Section 3, we generated various instances of fabric architectures. We synthesized the fabric VHDL into Synopsys cell-based ASIC design with a feature size of 90 nm using Synopsys Design Compiler. Figure 14 shows the area consumption of standard, dedicated pass gates, ICS, and hybrid architectures having both ICS and DPs. The hybrid architecture with ICS and 50% DPs consumes least area. This architecture provides 61% area savings compared to the standard architecture with no DPs.

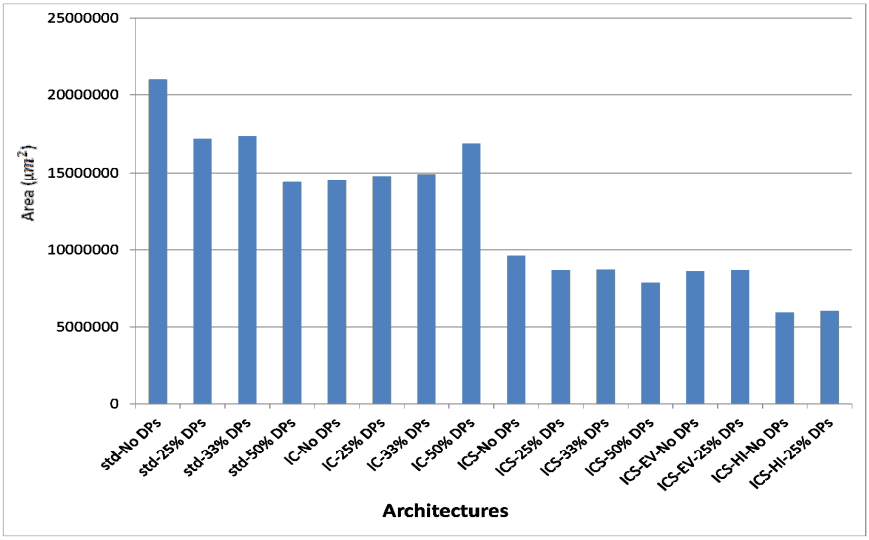

Figure 14: Area consumption for various fabric architectures implemented on Synopsys 90nm ASIC process.

We also examined the utilization of ALUs for pass operation for various fabric architecture implementations. In Table 9, we compare std, IC, ICS, ICS-EV, and a combination of DPs with these techniques. The number of ALUs used as pass gates has been reduced significantly when we compare the architectures having a combination of ICS and DPs 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 added 33% dedicated pass gates to the architecture, the number of ALUs being used as pass gates was reduced to 46. When we introduced ICS also in the fabric, ALUs are no longer required for passing values down in the fabric. Even in the hybrid architecture with IC and ICS and extended vertical interconnect, only 2 functional units are used for passing information.

We also conducted energy simulations on the architectures discussed in this paper. The energy results are shown in Figure 15. For each architecture, we compute energy for all the benchmarks examined and then compute average consumption over all the benchmarks. The combination of ICS and DPs consume least energy consumption. The hybrid-EV architecture also shows similar average energy consumption as ICS and DPs combination. This architecture does not use any dedicated pass gates to pass information from producer to the consumer. Instead it has extended vertical interconnect that increase the reachability of the functional units. The energy savings results for different fabric instances are shown in Table 10. *Energy was calculated by computing the product of the power and delay of the design*. To calculate the power and delay of the design, the fabric VHDL is synthesized into Synopsys cell-based ASIC design with a feature size of 90 nm using Synopsys Design Compiler. The post-synthesis design was simulated in Mentor Graphics ModelSim to calculate the delay of each design and these simulations were used as stimulus to the Synopsys PrimeTime-PX tool to estimate the power consumption of the device. The fabric with IC provides energy savings of 13% as compared to the standard fabric. With

ICS, we achieved energy savings of 27% as compared to the baseline architecture. By having a combination of DPs and ICS, we achieved energy savings upto 32% as compared to the baseline architecture averaged over all benchmarks. ICS-EV fabric provides energy savings of 27%.

Table 10: Energy savings (%) per benchmark mapped onto standard, DP, ICS and hybrid architectures.

	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
IC-No DPs	24	18	3	7	26	5	10	13
IC-25% DPs	39	35	6	10	46	8	12	22
IC-33% DPs	44	40	6	12	50	10	17	26
IC-50% DPs	32	39	1	7	49	10	9	21
ICS-No DPs	54	45	7	12	58	6	6	27
ICS-25% DPs	60	52	8	15	62	10	9	31
ICS-33% DPs	63	54	7	15	62	10	11	32
ICS-50% DPs	63	54	7	13	62	10	6	31
ICS-EV-No DPs	64	53	6	13	61	10	4	30
ICS-EV-25% DPs	65	53	6	13	61	10	7	31
ICS-HI-No DPs	62	57	6	13	62	10	8	31
ICS-HI-25% DPs	63	57	6	13	62	10	8	31

Table 11: Percentage area savings per benchmark mapped onto standard, DP, ICS and hybrid architectures.

	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
IC-No DPs	25	13	29	31	35	15	8	22
IC-25% DPs	43	39	28	35	34	13	6	28
IC-33% DPs	29	31	27	40	45	13	5	27
IC-50% DPs	26	51	18	18	56	11	-21	23
ICS-No DPs	46	75	49	54	72	29	32	51
ICS-25% DPs	58	74	54	59	83	39	31	57
ICS-33% DPs	63	74	54	64	83	38	31	58
ICS-50% DPs	69	80	59	68	82	36	38	62
ICS-EV-No DPs	70	81	55	64	83	49	39	63
ICS-EV-25% DPs	76	81	54	64	83	49	38	63
ICS-HI-No DPs	73	83	64	67	90	57	59	70
ICS-HI-25% DPs	76	83	64	67	90	56	58	71

4.1. Energy vs Area Tradeoffs

This section presents energy vs area tradeoffs for the suite of benchmarks for standard, IC, ICS, and hybrid architectures. Table 10 shows the energy savings per benchmark for various fabric implementations. Our baseline architecture is “std-No DPs”. We compare all architectural options with our reference baseline architecture to obtain savings. On an average, the architecture with integrated constants provides 13% energy savings, the ICS and ICS-EV architectures pro- vide 27% energy savings compared to the baseline. The combination of DPs and ICS can provide upto 32% energy savings over the baseline architecture.

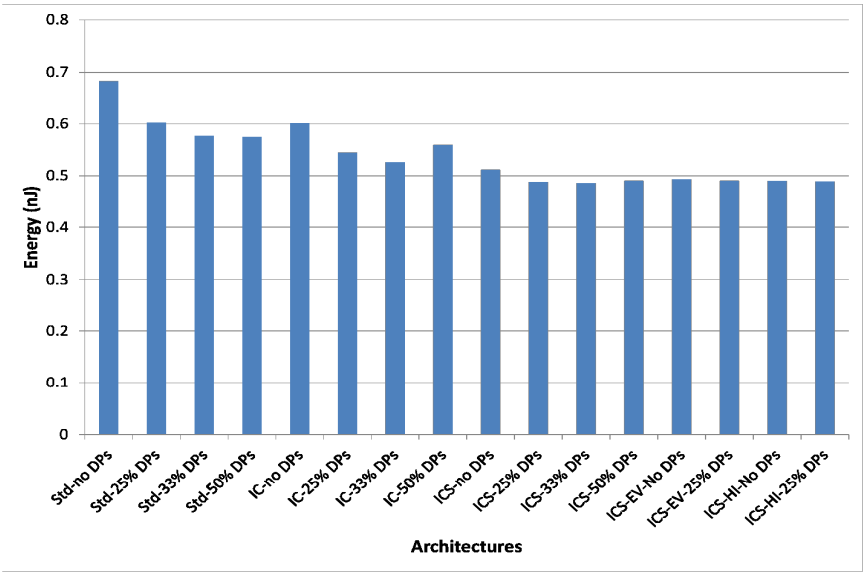

Figure 15: Energy comparison of standard, IC, ICS, ICS-EV, and hybrid architectures.

Table 11 shows the percentage area savings. IC fabric provides 22% area savings as compared to the standard implementation. The ICS and ICS-EV architectures achieve 51% and 60% area savings respectively. The combination of ICS and DPs provides upto 62% energy savings when compared with baseline.

Figure 16 shows the energy and area savings for various fabric architecture implementations for a suite of signal and image processing applications exam- ined here. For each architecture, we show the energy and area savings that we achieve over the baseline architecture averaged over all the benchmarks. The ICS option achieves more energy and area savings as compared to the IC architecture. When ICS is combined with DPs, the level of energy and area improvements get even higher. The same level of area and energy savings can also be achieved using ICS-EV style.

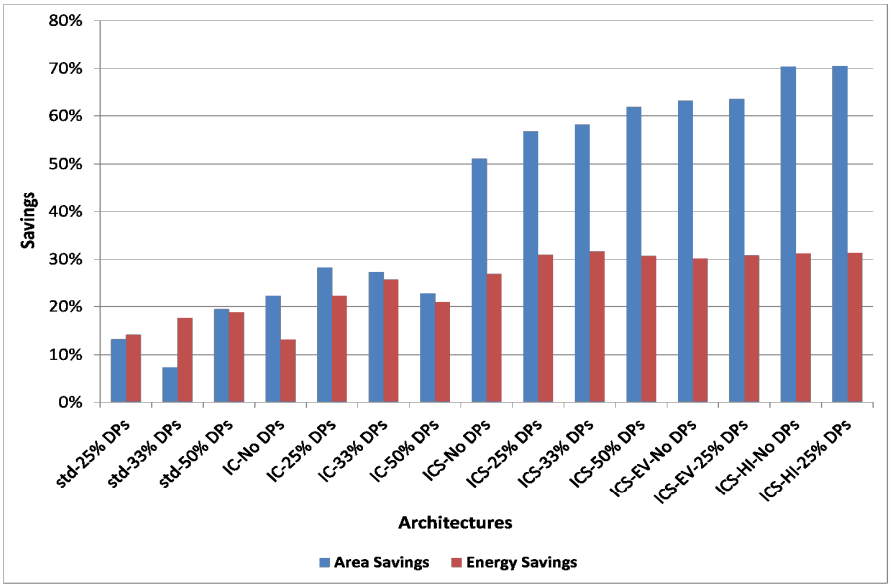

Figure 16: Area and energy savings for various fabric architectures compared to the baseline.

5. CONCLUSIONS

In this paper, we discussed various styles of routing constants and variable inputs in a stripe-based coarse grained reconfigurable fabric including (i) integrated constants (IC) approach where constants are loaded in the registers local to the functional units; (ii) inputs coming from the side (ICS) where both constants and variable inputs can be routed to the stripe directly where needed; (iii) ICS with extended vertical interconnect (ICS-EV); and (iv) a combination of dedicated pass gates (DPs) with standard, IC, ICS, and ICS-EV architecture styles. We implemented these architecture styles using 90 nm ASIC process from Synopsys. We performed a detailed area and energy analysis on these architectures using signal processing benchmarks from Mediabench benchmark suite and some of the image processing applications. We observed that the fabric with ICS and 50% DPs is the best among these options, providing 31% energy savings and 62% area savings over a baseline architecture for our benchmark set.

REFERENCES

[1] Cao Liang, Xinming Huang, Mapping Parallel FFT Algorithm onto Smart- Cell Coarse-Grained Reconfigurable Architecture, Application-specific Systems, Architectures and Processors, 2009. ASAP 2009. 20th IEEE In- ternational Conference on , pp.231-234, (2009)

[2] Yoonjin Kim, Mahapatra R.N., A New Array Fabric for Coarse-Grained Reconfigurable Architecture, Digital System Design Architectures, Methods and Tools (DSD '08), 11th EUROMICRO Conference on , pp.584-591, (2008)

[3] 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, pp.157-166, 17-19 (1996)

[4] B. Levine, HASTE: Hybrid Architectures with a Single Transformable Executable, Ph.D. dissertation, Department of ECE, CMU., (2005)

[5] Gayatri Mehta, Hoare R.R., Stander J., Jones A.K., A Low-Energy Re- configurable Fabric for the SuperCISC Architecture, 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '06) , pp.309-310 (2006)

[6] 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, (2000)

[7] Ebeling Carl, Cronquist Darren C., Franklin Paul , RaPiD - Reconfigurable Pipelined Datapath, In: Proceedings of the 6th International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Com- pilers, pp. 126–135 (1996)

[8] Gayatri Mehta, Alex K. Jones, Justin Stander, Mustafa Baz, Brady Hunsaker, Interconnect Customization for a Hardware Fabric, In: ACM Transactions on Design Automation for Electronic Systems (TODAES), vol 14, pp 11:1–11:32 (2009)

[9] Mehta, G., Ihrig, C.J., Jones A.K., Reducing energy by exploring heterogeneity in a coarse-grain fabric,In: Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on , pp.1-8, 14-18 (2008)

[10] Aggarwal A.A., Lewis D.M., Routing architectures for hierarchical field programmable gate arrays, In: Computer Design: VLSI in Computers and Processors (ICCD '94). Proceedings., IEEE International Conference on , pp.475-478, (1994)

[11] Kaviani A., Vranesic D., Brown, S. , Computational field programmable architecture, In: Custom Integrated Circuits Conference, 1998. Proceedings of the IEEE 1998 , pp.261-264, (1998)

[12] Heysters P.M., Smit G.J.M. , Mapping of DSP algorithms on the MON- TIUM architecture, In: Parallel and Distributed Processing Symposium, 2003. Proceedings. International pp. 6 , (2003)

[13] Schmit H., Whelihan D., Tsai A., Moe M., Levine B., Reed Taylor R. PipeRench: A virtualized programmable datapath in 0.18 micron technology, In: Custom Integrated Circuits Conference, Proceedings of the IEEE (2002), pp. 63- 66, (2002)

[14] Schmit H., Whelihan D., Tsai A., Moe M., Levine B., Reed Taylor R. , PipeRench: A virtualized programmable datapath in 0.18 micron technology, In: Custom Integrated Circuits Conference, 2002. Proceedings of the IEEE (2002), pp. 63- 66, (2002)

[15] Ebeling C., Cronquist D.C., Franklin P., Secosky J., Berg, S.G. , Mapping applications to the RaPiD configurable architecture, In: FPGAs for Custom Computing Machines (1997) Proceedings., The 5th Annual IEEE Symposium on ,pp. 106-115, (1997)

[16] B. Levine, H. Schmit, Implementation of Target Recognition Applications using Pipelined Reconfigurable Hardware In: Proceedings of Military and Aerospace Applications of Programmable Devices and Technologies Interna- tional Conference, (2003)

[17] P.M. Heysters, Coarse-Grained Reconfigurable Computing for Power Aware Applications, In: Proc. ERSA, pp.272-272 (2006)

[18] Frank Bouwens, Mladen Berekovic, Andreas Kanstein, Georgi Gaydadjiev, G.: Architectural Exploration of the ADRES Coarse-Grained Reconfigurable Array, In: ARC 2007. LNCS pp. 1–13 (2007)

[19] Cao Liang, Xinming Huang, SmartCell: An Energy Efficient Coarse- Grained Reconfigurable Architecture for Stream-Based Applications, EURASIP Journal on Embedded Systems, vol. 2009, Article ID 518659, 15 pages, (2009)

[20] Alex K. Jones, Raymond Hoare, Dara Kusic, Joshua Fazekas, and John Foster, An FPGA-based VLIW processor with custom hardware execution. Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays (FPGA '05), pp. 107–117, (2005)

[21] PACT-XPP, XPP-lib Core Overview, http://www.pactcorp.com/

[22] 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 , pp.380-387, (2010)

[23] Hauser J.R., Wawrzynek J., , Garp: a MIPS processor with a reconfigurable coprocessor, In: FPGAs for Custom Computing Machines, (1997), The 5th Annual IEEE Symposium on , pp.12-21, 16-18 (1997)

[24] H. Singh , Morphosys: An Integrated Re-configurable Architecture, the NATO RTO Symposium on System Concepts and Integration (1998)

[25] Gayatri Mehta, and Raymond R. Hoare, Justin Stander and Alex K. Jones, Design Space Exploration for Low-Power Reconfigurable Fabrics, In: Proceedings of IPDPS (2006).

[26] G. Lu, M. Lee, H. Singh, N. Bagherzadeh, F.J. Kurdahi, E.M.C. Filho, MorphoSys: A Reconfigurable Processor Trageted to High Performance Image Application, IPPS/SPDP Workshops, pp.661-669 (1999)

[27] C. Brunelli, F. Garzia, D. Rossi, and J. Nurmi, A coarse-grain reconfigurable architecture for multimedia applications supporting subword and floating-point calculations, Journal of Systems Architecture - Embedded Systems Design, pp.38-47, (2010)

[28] S. Hauck, T. W. Fry, M. M. Hosler, J. P. Kao, The Chimaera Reconfigurable Functional Unit, IEEE Symposium on FPGAs for Custom Computing Machines, pp. 87-96, (1997)

[29] M. Petrov, T. Murgan, F. May, M. Vorbach, P. Zipf, and M. Glesner, "The XPP Architecture and Its Co-simulation Within the Simulink Environment", in Proc. FPL, pp.761-770, (2004).

Authors

Anil Yadav: Anil Yadav is currently working in John Deere. He received his M.S. in Electrical Engineering at the University of North Texas, Denton, TX. He received his B.E. in Electronics and Communications from Rajiv Gandhi Technical University, Bhopal, India in 2005. His interests are in low-power and area-efficient reconfigurable architectures.

Justin Stander: Justin Stander is currently working in General Dynamics C4 systems, Pittsburgh, PA. He received his M.S. and B.S. in computer engineering from the University of Pittsburgh in 2007 and 2005 respectively. His interests are in compilation and computer aided design for low-power reconfigurable computing fabrics.

Alex K. Jones: Alex K. Jones is currently an Associate Professor of Electrical and Computer Engineering at the University of Pittsburgh, PA. He received his Ph. D. and M.S. degrees in 2002 and 2000, respectively, in electrical and computer engineering at Northwestern University. He received his B.S. in 1998 in physics from the College of William and Mary in Williamsburg, Virginia. His research interests include compilation techniques for behavioral and low-power synthesis, embedded systems, radio frequency identification (RFID), and high-performance computing.

Gayatri Mehta: Gayatri Mehta is currently as an Assistant Professor in the department of Electrical Engineering at the University of North Texas, Denton, TX. She received her Ph.D. in Electrical and Computer Engineering, M.S. in Telecommunications from the University of Pittsburgh in 2009 and 2003 respectively, M.Tech in Microelectronics from Panjab University, India in 2001, and B.Tech in Electronics and Communication from the National Institute of Technology, India in 1999. She is a recipient of Junior Faculty Summer Research Fellowship at the University of North Texas in 2010. She is an IEEE member. Her research interests are broadly in the areas of Reconfigurable Computing, Low-Power VLSI Design, System on a Chip Design, Electronic Design Automation, Embedded Computing, Wearable Computing, and Energy Harvesting.