Interconnect Customization for a Coarse-grained Reconfigurable Fabric *

Gayatri Mehta[1], Justin Stander[1], Mustafa Baz[2], Brady Hunsaker[2], and Alex K. Jones[1]
[1] Electrical and Computer Engineering [2] Industrial Engineering
University of Pittsburgh, {hunsaker,akjones}@engr.pitt.edu

Abstract

This paper describes several system-level interconnection strategies for a coarse-grained reconfigurable fabric designed for low-energy hardware acceleration. A small, representative sub-graph for signal and image processing applications is used to predict the success of mapping larger applications onto the fabric device with these different interconnection strategies, which include 32:1, 8:1, 5:1, 4:1, 3553:1 (3:1, 5:1, 5:1, 3:1) and 355:1 (3:1, 5:1, 5:1) cardinalities. Three mapping techniques are presented and used to complete mappings onto several of these fabric instances including a mixed integer linear programming technique, a constraint programming approach, and a greedy heuristic. We present results for area (in number of required rows), power, delay, and energy as well as run times for mapping a set of signal and image processing benchmarks onto each of these interconnects. Our results indicate that the 5:1 interconnect provides the best overall results and does not require any additional hardware resources than the baseline 4:1 technique. When compared with other implementation strategies, the reconfigurable fabric energy consumption, using 5:1-based interconnect, is within 5-10X of a direct ASIC implementation, is 10X better than an Virtex II Pro FPGA and is 100X better than an Intel XScale processor.

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 devices. Thus, the design cycle is much reduced, which can significantly decrease the time to market.

*This work was partially supported by the Technology Collaborative.

1-4244-0910-1/07/$20.00 ©2007 IEEE.

The tradeoff for using these reconfigurable devices is a compromise in performance and most notably power/energy consumption. To reduce the overhead of using a reconfigurable device, particular care must be given to designing the system-level interconnect of the device. It is the interconnect that largely determines the power and performance characteristics of the device (see Section 2.1).

This paper describes a system-level interconnect prediction strategy for the SuperCISC low-energy reconfigurable fabric target. In previous work, the fabric architecture was developed based on multi-bit functional units and multi-bit routing structures configured through multiplexers [13]. As part of the previous work, an architectural design space exploration was completed that determined parameters for the functional units and routing. However, the structure of the routing was considered only briefly. In this paper, a commonly recurring graph structure from applications in the multimedia and signal processing domains that is difficult to map onto the fabric is used to drive the interconnect strategy. This graphical structure is applied to several interconnection strategies within the fabric architecture to help determine their viability for larger applications.

The remainder of this paper is organized as follows: Section 2 provides some background material on the SuperCISC project and the mapping concept including a study of related work within these areas. The system-level interconnect architectures and evaluation strategies are described in detail in Section 3. Section 4 presents the three mapping strategies employed in this work. Power, delay, and energy results are presented in Section 5. Section 6 discusses several conclusions and considers future work.

2 Background and Literature Review

2.1 System Overview

While FPGAs are the most commonly used general purpose reconfigurable fabrics, they exhibit poor power characteristics. The dynamic power consumption in FPGAs has been shown to be dominated by interconnect power. For example, the reconfigurable interconnect in the Xilinx Virtex II FPGA consumes more than 70% of the total power dissipated in the device [16]. Contributing to the static power consumption of FPGAs are the SRAMs for programming the state of the device. This is exacerbated by the necessity of bit-level control for the computational and switch blocks. Additionally, because the device is designed to handle sequential logic, clock trees and storage registers are required, which also contribute to power consumption. Thus, to create a low-power computational fabric, it is desirable to remove or reduce as many of these power consuming characteristics as possible.

The SuperCISC low-power fabric was designed to operate within the SuperCISC processor architecture summarized in [10]. The idea is to accelerate the high incidence code segments (e.g. loops) that require large portions of the application runtime, called kernels, while assigning the control-intensive portion of the code to a core processor. These kernels are converted into entirely combinational hardware functions generated automatically from the C using a design automation flow [9]. Using hardware predication, a Control Data Flow Graph (CDFG) can be converted into a Super Data Flow Graph (SDFG) [9]. SDFG based hardware functions operate asynchronously from the processor core. Also by removing the sequential logic makes the hardware fabric for implementing the SDFG much simpler.

Due to certain assumptions of the SuperCISC flow, such as entirely combinational hardware, original computation from C implying 8, 16, and 32-bit operation granularities, it is possible to reduce high power characteristics of an FPGA for a more efficient reconfigurable device. Moving to multibit functional units significantly reduces routing complexity and leads to a lower power device. Removal of sequential logic eliminates clock trees and local storage also contributing to power reduction. Thus, the interconnect prediction in Section 3 begins with these assumptions.

2.2 Reconfigurable Fabric Target

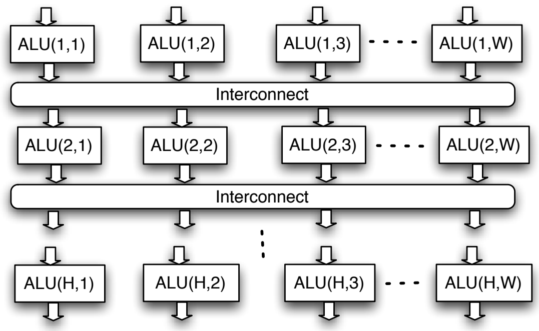


Figure 1. The fabric model is comprised of ALUs and a reconfigurable interconnect.

SDFGs retain a data flow structure allowing computational results to be computed in one arithmetic and logic unit (ALU) and flow onto others in the system. The proposed reconfigurable fabric model is designed to mimic this computational style. As shown in Figure 1, 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 model does not include registers or other internal storage, and does not permit feedback between stripes.

The fabric model was implemented in parameterized VHDL using the `generic` capability of the VHDL language. For example, each ALU in the fabric can be represented by a number of parameters such as the number of operands O, data width of each operand DW, the number of operations OP. The multiplexer cardinality C determines the width of each multiplexer and as a result connectivity of the interconnection stripe.

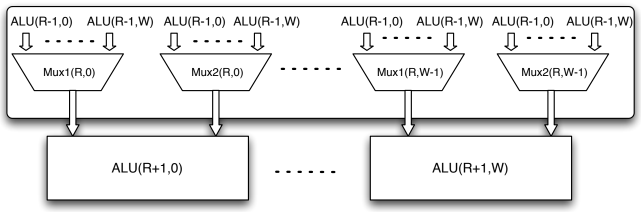


Figure 2. Interconnect stripe feeding ALUs.

The fabric size is determined with the parameters specifying the width of the fabric W, height of the fabric H, and data width DW. W dictates the number of ALUs in each computational stripe. The number of multiplexers in each interconnection stripe is a function of both W and O as the input to each ALU operand is configurable. This is shown in Figure 2. H determines the number of computational and interconnection stripes in the fabric model. Thus, with the parameterizable model, a fabric contains WxH DW-wide ALUs segregated into H computational stripes. These computational stripes are interconnected by $H - 1$ stripes each containing OxW DW-wide $C : 1$ multiplexers.

As the fabric model was designed for implementation of SDFGs, in addition to the removal of need for internal storage, one of the operations implemented within the ALU is *hardware predication*. This operation requires a third, single-bit operand to be included in the ALU. This bit specifies which of the two input operands to propagate to the output, acting as a selector. Thus, the interconnection stripe contains a third set of single-bit $C : 1$ multiplexers for controlling this operand.

2.3 Acceleration with Custom Hardware

Recently, a tremendous amount of effort has been devoted to the area of reconfigurable computing particularly stressing the development and use of coarse-grained fabrics for computationally complex tasks. Many architectures have been proposed and developed both in academia and industry during the last decade such as MATRIX, Garp, RaPiD, PipeRench, Elixent, Pact XPP, and the FPOA.

Unlike MATRIX [14] whose basic functional unit consists of an 8-bit ALU and a SRAM, the basic functional unit in our fabric is a coarse-grained ALU having variable data width. However, for this paper we fix the ALU data width to be 32-bits. Our approach differs from GARP [8] insomuch as we tailor the hardware co-processor to the application domain. Compared to RaPiD [4], which has smaller RAMs and registers to store data and intermediate results, our fabric is purely combinational. The programmable connections in the data-path interconnect in our fabric are modeled as multiplexers somewhat similar to those in RaPiD. Unlike RAP [5] whose ALUs are arranged in a chess board style, the fabric model used in our research has a striped configuration like that of PipeRench [15] but without register files.

Several methods have been proposed in the past few years for design space exploration of reconfigurable architectures [1, 2, 6]. However, these methods are either too technology-dependent or too architecture-dependent. Bossuet, et al [3] proposed a design space exploration method that can be used to cover a wide domain of reconfigurable fabrics. They used the architectural processing use rate and the communication hierarchical distribution as metrics to investigate a power-efficient architecture. In contrast, our work studies the impact of varying the interconnect strategy based on a representative sub-graph to reduce the power/delay of coarse-grained reconfigurable architectures.

2.4 Mapping Problem Statement

In order to use our fabric with a given benchmark circuit, it is necessary to map the circuit onto the fabric. Such a mapping consists of an assignment of operators in the circuit to ALUs of the fabric such that the logical structure of the circuit is preserved and the parameters of the fabric are respected, particularly the width, height, and interconnect design. This mapping problem is central to the use of the fabric, and we consider it in several forms described as follows. Our approaches for solving the mapping problems appear later in Section 4. All of the problems assume a fixed fabric width and interconnect design.

Minimum Size Mapping: In order to reduce power consumption, it is desirable to use as few rows in the fabric as possible. Given a fabric width, fabric interconnect design, and circuit to be mapped, the Minimum Size Mapping problem is to find a mapping that uses the minimum number of rows in the fabric. The mapping may use pass gates as necessary.

Feasible Mapping: Given a fully-specified fabric and a circuit, Feasible Mapping is the problem of identifying any mapping that preserves the logical structure of the circuit and respects the fabric parameters. Note that for a given height, width, and interconnect design, some circuits may have no feasible mapping.

Feasible Mapping with Fixed Rows: One of the more complicated parts of creating a mapping is the introduction of pass gates to fit the row-wise structure of the fabric. A successful approach that we have used is to work in two stages. In the first stage, pass gates are introduced heuristically and operators are assigned to rows so that all edges go from one row to the next. The second stage assigns the operators to columns so that the fabric interconnect is respected. This second stage is called Feasible Mapping with Fixed Rows. Note that depending on the interconnect design, there may or may not exist such a feasible mapping.

Optimal Graded-Cost Mapping: In this problem, we assume that a full interconnect is available in the fabric, but we assign different costs to each ALU depending on what size of interconnect it needs for its inputs. Interconnects with larger fan-in are given a higher cost than those with lower fan-in, on a graded scale. The problem is to find a mapping with minimum total cost. Our reason for investigating this problem is to see how often large interconnects are needed and to gain insight into possible interconnect designs. We will solve this problem and show the results in Section 3.2.

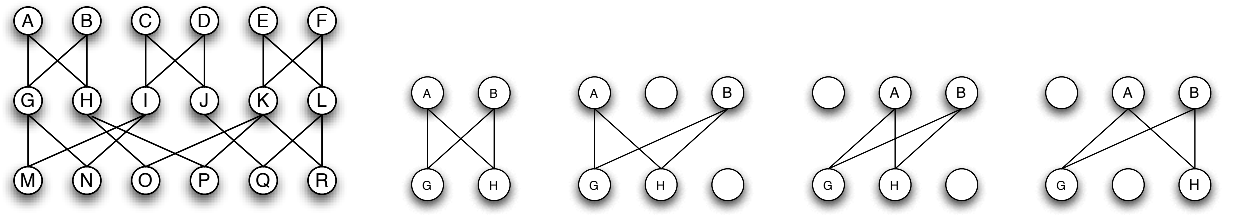
3 System-level Interconnection

A fundamental problem that we explore in this paper is the Fabric Design Problem: Given a set of representative benchmarks, determine a good fabric design. This is a subjective problem in that it may involve tradeoffs between the interconnect design and the size of the fabric.

3.1 Benchmark Driven Interconnect

Architectural innovations are often developed in somewhat of a vacuum, without consideration of the tools needed to program the architectures and without consideration of the needs of the applications to run on the architectures. For a reconfigurable fabric such as the architecture described here, the temptation is to create an interconnect that has a regular structure. However, depending on the needs of the applications to be implemented, this regular structure may not be appropriate.

As we describe in Section 2.2 our reconfigurable fabric consists of a multiplexer-based interconnect. Based on our initial design space exploration studies, a 4:1 interconnect was selected to create enough flexibility in the routing while reducing power consumption and delay in the fabric [12, 13]. An example of this interconnection is shown in Figure 3. Each operand of the ALU has a multiplexer in this configuration, including the third operand for the predica-



(a) Problem graph. (b) Standard. (c) Top shifted right. (d) Top two shifted right. (e) Bottom shifted left.

Figure 4. A problem graph and different ways to overlap node pairs with 4:1 connectivity.

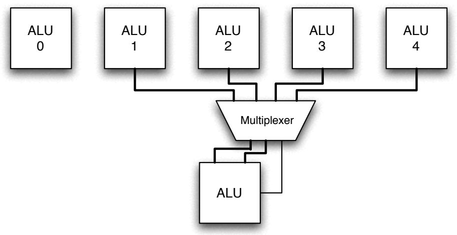


Figure 3. Connectivity using 4:1 multiplexers.

tion operation. In this configuration, there are four possible locations to read the input operands from the previous row.

While many structures can be successfully mapped using this interconnection strategy, this configuration is somewhat limiting, as it provides only a 4-way fanout from one node to other nodes in the circuit. Additionally, a subgraph that appears frequently in signal and image processing applications is shown in Figure 4(a). Between the first and second rows, this graph consists of three pairs of nodes (A,B), (C,D), and (E,F) that communicate with three other pairs of nodes (G,H), (I,J), and (K,L) in the subsequent stage. However, between the second and third stage nodes G-H are grouped into three different pairs (G,I), (H,K), and (J,L), which communicate with three new pairs in the third stage (M,N), (O,P), and (Q,R). While separately, either of these rows could be implemented with the configuration in Figure 3, the edges connecting H with P and J with R are not possible in this configuration.

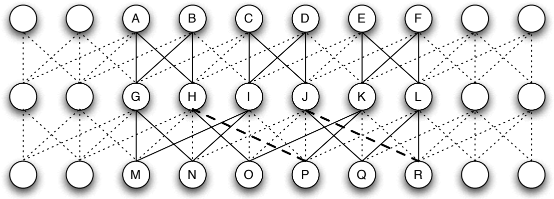


Figure 5. Attempt to embed the graph from Figure 4(a) into 4:1 multiplexer interconnect.

The graph in Figure 4(a) is not a directed minor of the graph created by the connectivity in the fabric supplied by the 4:1 multiplexers as oriented in Figure 3. Two edges represented by dashed lines, as shown in Figure 5, are not available in this structure. This is not solved by permuting the nodes. Figure 4 presents all four orientations possible for two pairs of nodes with dependency edges under this connectivity. Unfortunately, none of these configurations can be overlapped with another pair of nodes, which shows that permutation of the nodes is not possible. To effectively map this structure, what is required is effectively a 5:1 multiplexer. However, a true 5:1 multiplexer would require an additional control bit and an additional level of logic, which is undesirable from a power and performance perspective.

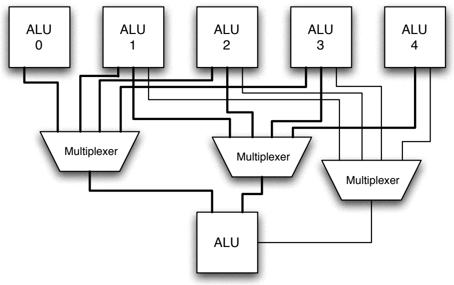


Figure 6. Schematic for a 5:1 multiplexer equivalent using 4:1 multiplexers

The connectivity shown in Figure 6 is a compromise that allows an emulation of a 5:1 multiplexer without increasing the architectural complexity beyond 4:1 multiplexers. In this case, the three internal ALUs, 1-3, are shared on both operand's multiplexers. The outermost ALUs, 1 and 5, are only available on the left and right operand multiplexer, respectively. The rationale for this is that if an operand is placed to the far left or right ALU, the other operand cannot occupy the same space, thus there is no conflict for the resource. The biggest limitation to this approach is that for non-commutative operations such as subtract, there is some restriction as to which operand may be retrieved from the far left or far right.

As shown in Figure 7, it is relatively easy to embed the graph from Figure 4(a) into the connectivity provided by Figure 6. Based on our study of several applications graphs, we have found that the limited 5:1 connectivity provides a good baseline to relatively easily map. The limitation of

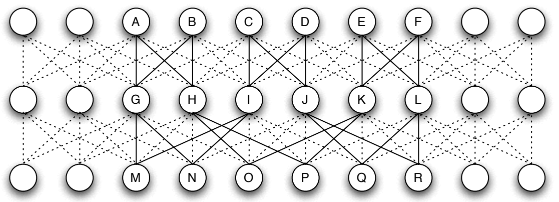


Figure 7. Embedding the graph from Figure 4(a) with the connectivity in Figure 6.

non-commutative operation mapping can be overcome by providing a separate operation that executes the operation right to left in addition to left to right.

3.2 Optimizing the Interconnect

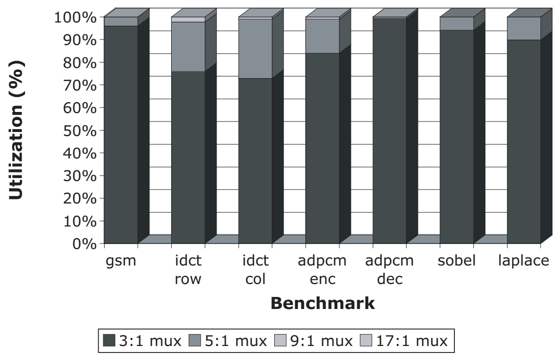


Figure 8. Multiplexer cardinality usage.

While we have demonstrated that our 5:1 multiplexer is an effective interconnect for signal and image processing applications, it may be possible to further optimize this strategy. We mapped our benchmark set to a fabric with fully interconnected stripes (e.g. any ALU from a previous stripe could be connected to any ALU in the current stripe). Using a mixed-integer linear programming (MILP)[1] formulation that provided an increasing penalty to using 5:1, 9:1, and 17:1 routes or higher, we created the multiplexer usage statistics in Figure 8. The IP technique eliminated need for nearly all multiplexers greater than 5:1 for most of the cases.

Consider the goal to replace one-third of the 5:1 multiplexers with 3:1 multiplexers, built from mirrored 2:1 multiplexers. Now, consider a structure where we have two 5:1 multiplexers adjacent, followed by a 3:1 which is repeated. We call this a 355:1 interconnect. The graph from Figure 4(a) can be directly mapped.

If we increase the goal to replace one-half of the 5:1 multiplexers with 3:1 multiplexers, the first case to be considered is alternating both types of multiplexers. If we consider the possible pair communications from Figure 4, with the inclusion of 5:1 multiplexers we can now also consider the mirror of Figure 4(c) - 4(e) in addition to a few more configurations such as the (H,K) and (O,P) node pairs from Figure 4(a). To overlap any of these node pairs requires two adjacent spaces with 5:1 multiplexer connectivity, which explains in part why the embedding from Figure 7 is relatively straightforward. However, it is possible to replace half of the 5:1 multiplexers with 3:1 multiplexers and map the graph if we alternate in groups of two. We call this a 3553:1 interconnect and the solution is shown in Figure 9.

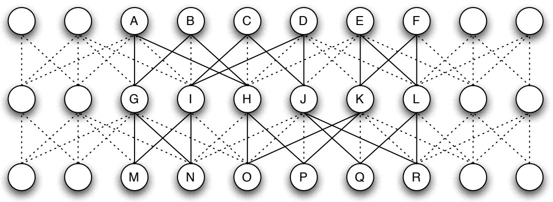


Figure 9. Embedding the graph from Figure 4(a) into a 3553:1 interconnect.

4 Mapping Strategies

As discussed in Section 2.4, a central problem in working with the architecture we discuss here is that of mapping a circuit onto the fabric. In this section we discuss the approaches we used to solving the various forms of the mapping problem.

4.1 Mixed Integer Linear Program

MILP is a common modeling and solution technique for combinatorial optimization. We used MILP to solve Feasible Mapping with Fixed Rows as well as for Optimal Graded-Cost Mapping. The objective function used in MILP is to minimize the number of required edges that are outside the interconnect design. If the MILP formulation finds a solution with objective value zero, then the solution is a valid mapping for the given interconnect design. If the optimal objective value is greater than zero, then there is no feasible mapping for that interconnect design. We used CPLEX 9.0 to solve the MILP. The details of the IP formulation can be found in [11].

4.2 Constraint Program

Constraint programming is a modeling and solution methodology based on the ideas of intelligent enumeration and reduction of the search space through careful analysis of constraints. It is most often applied to feasibility problems. We developed a constraint program to solve the problem of Feasible Mapping with Fixed Rows and then used this model in a heuristic approach to Minimum Size Mapping. The model was implemented in the open-source Mozart/Oz environment. The details can be found in [11].

[1]We use the MILP and IP interchangably to refer to an mixed-integer linear program.

4.3 Greedy Heuristic

The Greedy Heuristic Mapper follows a top-down mapping approach to provide a Feasible Mapping for any given benchmark. Starting with the top row, it completely places each individual row using a limited look-ahead of two rows. After each row is mapped, the mapper will not modify the mapping of any portion of that row. While the limited information available to the mapper does not often allow it to produce Optimal Mappings or Minimum-Size Mappings, its relative simplicity provides a decidedly short runtime. By default it tries to map the given benchmark to a fabric with width equal to the largest individual row, and height equal to the longest path through the graph. Although the width is static throughout a single mapping, the height can increase as needed. The details can be found in [11].

5 Results

In order to evaluate power and performance, a set of core signal processing benchmarks were selected from MediaBench benchmark suite including 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. Using the SuperCISC compilation flow [9], computational kernels were extracted for these applications and converted into SDFGs, which we used as our benchmark circuits. These SDFGs were then mapped to the fabric model using the IP program, constraint program, and greedy heuristic as described in Section 4. We targeted fabric hardware using 32:1, 8:1, and 4:1 multiplexers. We also targeted 5:1, 3553:1 and 355:1 multiplexer-based interconnects as described in Section 3. A detailed study of varying other parameters of the fabric such as the bit-width the ALUs can be found in [13].

Table 1 provides a summary of the area requirements of the benchmarks mapped to the fabric using the different multiplexing cardinalities and using the previously mentioned mapping strategies. Table 2 summaries the runtimes required by the mapping algorithms for each multiplexing cardinality. All items marked '-' were not able to be mapped. Once all benchmarks were mapped using a specific mapping technique to a fabric with a particular interconnect, the fabric size was fixed to the smallest size that could fit all seven benchmarks.

In the subsequent sections we study the power, delay, and energy impact of varying the interconnect strategy within the fabric. To calculate the power and delay of the design, the fabric was synthesized into an Oki cell-based ASIC design with a feature size of 0.16 μm 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 PrimePower tool to estimate the power consumption of the device. Energy was calculated by computing the product of the power and delay of the design.

Table 1. Final size (W x H) for various mapping strategies onto different interconnects.

	gsm	sob	lap	row	col	enc	dec
32:1							
Greedy	14x18	10x9	15x8	17x10	20x12	17x16	16x13
8:1							
IP	20x18	14x9	20x8	19x10	20x13	20x16	20x13
Const.	18x18	13x9	17x9	18x10	20x13	19x16	16x13
Greedy	14x18	10x9	15x8	17x10	20x12	17x16	16x13
5:1							
IP	19x18	17x9	20x8	17x10	20x13	20x16	20x13
Const.	18x18	13x9	15x9	18x10	20x13	18x16	16x13
Greedy	20x19	10x9	20x8	17x13	20x19	20x18	16x13
4:1							
IP	20x18	16x9	15x8	-	-	-	19x13
Const.	18x18	13x9	18x10	20x13	20x17	19x19	16x13
Greedy	20x21	10x9	20x8	17x18	20x28	20x20	16x15
3553:1							
IP	20x18	-	17x8	-	-	-	17x13
Const.	17x18	-	17x9	16x22	-	-	16x13
Greedy	20x22	20x10	20x9	17x19	20x29	17x30	20x21
355:1							
IP	20x18	20x9	15x8	-	-	-	20x13
Const.	16x18	11x9	16x11	17x10	-	18x16	16x13
Greedy	14x19	20x10	20x9	20x17	20x24	20x22	16x15

5.1 8:1 mappings

The size of the 8:1 multiplexer-based interconnection fabric was set to 20x18 as each of the three mapping strategies was able to map all seven benchmarks within a fabric of this size. Table 3 summarizes the power, delay, and energy results for mapping to this particular fabric instance. In most cases, the results are pretty consistent across all mappers.

5.2 5:1 mappings

While the greedy heuristic performs reasonably well for 8:1 mappings, it does not perform as well for 5:1 mappings. The first indication, is that while the constraint and IP programming approach require no increase over the 20x18 fabric used in the 8:1 case, the greedy heuristic requires a 20x19 device. This leads to a power and delay increase. It should be noted that this improvement comes at a cost of at best a 10X additional mapping time. The summary of all power, delay, and energy results are shown in Table 4.

5.3 4:1, 3553:1 and 355:1 mappings

One of the successes of the 5:1 mappings, described in Section 5.2 is that with no additional architectural complexity, the interconnect flexibility is much greater than the baseline 4:1 multiplexing. This can be seen in Tables 1

Table 2. Runtime (seconds) of various mapping strategies onto different interconnects.

	gsm	sob	lap	row	col	enc	dec
32:1							
Greedy	< 1	< 1	< 1	< 1	< 1	< 1	< 1
8:1							
IP	514	37	13	460	894	306	155
Const.	1	< 1	27	1	10	1	1
Greedy	< 1	< 1	< 1	< 1	1	4	< 1
5:1							
IP	1801	6162	46	2049	7587	2746	960
Const.	1	< 1	38	27	5	213	1
Greedy	3	1	1	4	7	10	3
4:1							
IP	2613	104	158	-	-	-	1257
Const.	1	< 1	62	87	307	86	1
Greedy	4	1	1	17	19	16	2
3553:1							
IP	1351	-	181	-	-	-	539
Const.	25	-	23	512	-	-	1
Greedy	48	1	2	8	25	64	7
355:1							
IP	581	82	62	-	-	-	451
Const.	30	< 1	93	23	-	2	1
Greedy	2	< 1	< 1	4	7	7	3

and 2, where the numbers of required rows are consistently lower for 5:1, and many 4:1 mappings are not possible to be obtained in a reasonable amount of time using the IP formulation. The results for mapping to 3553:1 and 355:1 multiplexing-based interconnect are shown in Table 5. Because the IP and constraint programming solutions were unable to map several instances, a fixed size device was unable to be established and thus the power, delay, and energy results cannot be included. The results show that mappings to these interconnects were difficult, and that all mappers required some compromise. The greedy heuristic was forced to add too many rows that counteracted the savings due to the simplified interconnect.

5.4 Fabric Technology Comparison

Figure 10 shows a comparison of the best energy results for each type of interconnect, be it from the constraint, IP, or greedy mapping approach. Based on these results, clearly the 5:1 interconnect is the best overall solution for energy.

To provide some context, we compare the "best" fabric architecture with 5:1 multiplexing interconnect to implement the design on other digital hardware technologies, shown in Figure 11. Delay of the FPGA-based design was computed using post place-and-route simulation in ModelSim and power was estimated in Xilinx XPower using the results of the delay simulations. The delay of the XScale processor was calculated using the SimpleScalar ARM simulator and the XTREM [7] tool to estimate the power consumed by the processor. As before, energy is computed as the product of the delay and power.

Table 3. Power, delay, and energy results for an 8:1 multiplexer-based interconnect.

	gsm	sob	lap	row	col	enc	dec
Power (mW)							
IP	22.19	5.68	4.42	35.74	31.87	20.68	3.52
Const.	22.27	5.81	5.03	36.57	31.95	19.84	3.38
Greedy	21.45	5.65	4.30	36.22	29.09	20.08	3.50
Delay (ns)							
IP	48.4	28.6	22.2	37.5	47.2	39.5	35.4
Const.	48.4	28.6	25.4	37.5	47.2	39.5	35.4
Greedy	48.4	28.6	22.2	37.5	44.0	39.5	35.4
Energy (pJ)							
IP	1110	244	172	1858	1753	910	151
Const.	1114	250	196	1902	1757	873	145
Greedy	1073	243	168	1883	1600	884	151

Table 4. Power, delay, and energy results for a 5:1 multiplexer-based interconnect.

	gsm	sob	lap	row	col	enc	dec
Power (mW)							
IP	19.6	5.0	4.0	31.8	28.9	17.4	2.9
Const.	19.4	5.0	4.4	31.2	27.2	17.7	2.7
Greedy	19.3	4.7	3.9	39.4	33.4	18.0	2.7
Delay (ns)							
IP	42.0	25.3	22.1	34.2	42.5	36.8	30.8
Const.	42.0	25.3	22.1	34.2	42.5	36.8	30.8
Greedy	43.7	26.6	23.4	40.6	49.0	38.0	32.1
Energy (pJ)							
IP	843	188	137	1464	1415	662	107
Const.	832	185	150	1436	1334	672	100
Greedy	868	181	139	1889	1704	722	105

This chart is shown with a logarithmic scale for the energy. The energy required by the optimized fabric is within about 5-10X of an ASIC implementation and is 10X better than a Xilinx Virtex II Pro FPGA and 100X better than an Intel XScale processor operating at 733 MHz. A more detailed discussion of the tradeoffs of the reconfigurable fabric versus the ASIC, FPGA and embedded processor implementations can be found in [13].

6 Conclusions

In this paper we describe several multiplexer-based system level interconnection strategies (32:1, 8:1, 5:1, 4:1, 3553:1, 355:1) for a reconfigurable fabric that are predicted to work well for signal and image processing applications. The results presented here indicate that the 5:1 interconnect provides the best overall results and does not require any additional hardware resources than the baseline 4:1 technique. When compared with other implementation strategies, the reconfigurable fabric energy consumption, using 5:1-based interconnect, is within 5-10X of a direct ASIC implementa-

Table 5. Power, delay, and energy results for 3553:1 and 355:1 interconnects.

3553:1	gsm	sob	lap	row	col	enc	dec
Power (mW)							
Greedy	16.2	3.5	3.0	40.1	35.3	20.2	3.4
Delay (ns)							
Greedy	50.1	27.8	24.6	52	63.0	52.0	43.5
Energy (pJ)							
Greedy	937	183	146	2445	2259	1068	175
355:1							
Power (mW)							
Greedy	15.9	4.71	3.31	39.8	33.5	17.38	2.81
Delay (ns)							
Greedy	44.43	27.8	24.6	43.09	55.23	41.84	32.05
Energy (pJ)							
Greedy	795	207.24	135.71	2109.4	1876	782	123.64

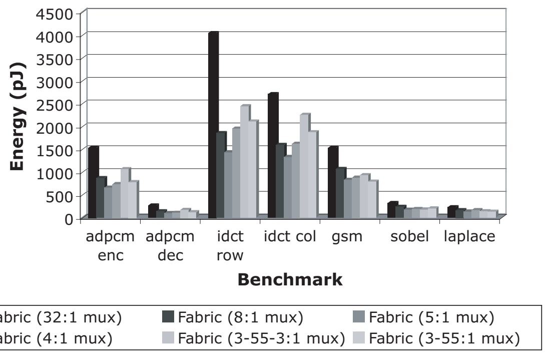


Figure 10. Energy comparison of benchmarks mapped to various interconnects.

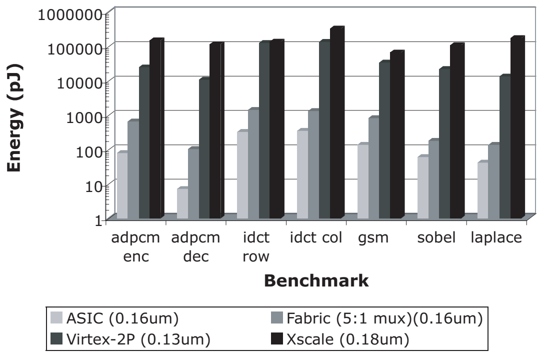


Figure 11. Energy comparison for various hardware and software implementations.

tion, is 10X better than an Virtex II Pro FPGA and is 100X better than an Intel XScale processor.

Our planned future work is to investigate further different interconnect structure patterns for reconfigurable architectures. We also plan to improve the mapping techniques and by doing so, we expect to further improve power/performance results.

References

[1] P. Benoit, G. Sassatelli, L. Torres, D. Demigny, M. Robert, and G. Cambon. Metrics for reconfigurable architectures characterization: Remanence and scalability. In *Reconfigurable Architecture Workshop*, 2003.

[2] S. Bilavarn, G. Gogniat, J. L. Philippe, and L. Bossuet. Fast prototyping of reconfigurable architectures from a C program. In *IEEE Symposium on Circuits and Systems*, 2003.

[3] L. Bossuet, G. Gogniat, and J.-L. Philippe. Generic design space exploration for reconfigurable architectures. In *Proc. of the Reconfigurable Architectures Workshop (RAW)*, 2005.

[4] C. Ebeling, D. C. Cronquist, and P. Franklin. Rapid - reconfigurable pipelined datapath. In *Proc. of FPL*, 1996.

[5] Elixent. The reconfigurable algorithm processor. `http://www.elixent.com/`.

[6] R. Enzler, T. Jeger, D.Cottet, and G. Troster. High-level area and performance estimation of hardware building blocks on FPGAs. In *Proc. of FPL: Forum on Design Language*, 2000.

[7] C. Gilberto, M. Martonosi, J. Peng, R. Ju, and G. Lueh. XTREM: A power simulator for the Intel XScale core. In *Proc. ACM LCTES*, 2004.

[8] J. R. Hauser and J. Wawrzynek. Garp: A MIPS processor with a reconfigurable coprocessor. In K. L. Pocek and J. Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 12–21, Los Alamitos, CA, 1997. IEEE Computer Society Press.

[9] R. Hoare, A. K. Jones, D. Kusic, J. Fazekas, J. Foster, S. Tung, and M. McCloud. Rapid VLIW processor customization for signal processing applications using combinational hardware functions. *EURASIP Journal on Applied Signal Processing*, 2006:Article ID 46472, 23 pages, 2006.

[10] A. K. Jones, R. Hoare, D. Kusic, G. Mehta, J. Fazekas, and J. Foster. Reducing power while increasing performance with superCISC. *ACM Transactions on Embedded Computing Systems (TECS)*, 5(3):658–686, 2006.

[11] A. K. Jones, G. Mehta, J. Stander, M. Baz, and B. Hunsaker. Interconnect predicition and customization for a hardware fabric. *IEEE Transactions on VLSI*, submitted. preliminary edition published in technical report TR-ECE-2006-07-001.

[12] G. Mehta, R. R. Hoare, J. Stander, and A. K. Jones. Design space exploration for low-power reconfigurable fabrics. In *Proc. of the Reconfigurable Architectures Workshop (RAW)*, 2006.

[13] G. Mehta, J. Stander, J. Lucas, R. R. Hoare, B. Hunsaker, and A. K. Jones. A low-energy reconfigurable fabric fo the supercisc architecture. *Journal of Low Power Electronics*, 2(2), August 2006.

[14] E. Mirsky and A. Dehon. Matrix: A reconfigurable computing architecture with configurable instruction distribution and deployable resources. In *Proc. of FCCM*, April 1996.

[15] H. Schmit, D. Whelihan, A. Tsai, M. Moe, B. Levine, and R. R. Taylor. Piperench: A virtualized programmable datapath in 0.18 micron technolog. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2002.

[16] L. Sheng, A. S. Kaviani, and K. Bathala. Dynamic power consumption in virtex-II FPGA family. In *FPGA*, 2002.