Design Space Exploration for Low-Power Reconfigurable Fabrics

Gayatri Mehta, Raymond R. Hoare, Justin Stander*, and Alex K. Jones
{gmehta,hoare,akjones}@ece.pitt.edu, *jns36@pitt.edu
Department of Electrical and Computer Engineering
University of Pittsburgh, Pittsburgh, PA 15261 USA

Abstract— **This paper presents a parameterizable, coarse-grained, reconfigurable fabric model that attempts to maintain Field Programmable Gate Array (FPGA)-like programmability and Computer Aided Design (CAD), with Application Specific Integrated Circuit (ASIC)-like power characteristics for Digital Signal Processing (DSP) style applications. Using this model, architectural design space decisions are explored in order to define an energy-efficient fabric. The impact on energy and performance due to the variation of different parameters such as datawidth and interconnection flexibility has been studied. The multiplexer cardinality usage has also been studied by mapping some of the signal processing applications onto the fabric. The results point to the use of power optimized 32-bit width computational elements interconnected by low cardinality multiplexers like 4:1 multiplexers.**

I. Introduction

Hardware acceleration using Field Programmable Gate Arrays (FPGAs) has become increasingly popular for computationally intensive Digital Signal Processing (DSP) applications. Unfortunately, while FPGAs have a reasonably tractable Computer Aided Design (CAD) flow and performance, they have poor power characteristics when compared to direct Application Specific Integrated Circuit (ASIC) fabrication. However, ASICs require more complex CAD than FPGAs and large Non-Recurring Engineering (NRE) costs.

A reconfigurable device that exhibits ASIC-like power qualities and FPGA-like costs and tool support is desirable to fill this void. Several coarse-grained fabric architectures have been proposed during the last decade such as MATRIX [1], RaPiD [2], PipeRench [3]. Most of these have been focused on performance and area-efficient architectural techniques with the notable exception of reduced power consumption.

This paper proposes an approach to reconfigurable fabric architectural space exploration with an emphasis on both performance and energy efficiency. A parameterizable reconfigurable fabric model is presented that allows design parameters to be adjusted within the architecture. The impact of varying different design parameters such as the width of functional units, and the granularity of interconnect are studied for their implications on power and performance.

The remainder of this paper is organized as follows: Section II describes relevant previous work particularly concentrating on the design space exploration of the reconfigurable computational fabrics. The fabric architecture and configurable parameters are described in Section III. Results for design space exploration are presented in Section IV. Conclusions and future work are discussed in Section V.

II. Related Work

Several methods have been proposed in the past few years for design space exploration of reconfigurable architectures [4–6]. However, these methods are either too technology-dependent or too architecture-dependent. Due to this drawback, these methods can not explore a large domain of the design space. In order to overcome this limitation, Bossuet et al [7] proposed a design space exploration method which can be used to cover a wide domain of reconfigurable fabrics, from fine-grained to coarse-grained fabrics, as well as heterogeneous fabrics. They used the architectural processing use rate and the communication hierarchical distribution as metrics to investigate a power-efficient architecture. However, our work is focused on the study of the impact of varying different design parameters such as the data width of the basic functional units, and the granularity of interconnect on power as well as performance of the reconfigurable architectures.

The proposed low-power fabric was designed to operate within the SuperCISC processor architecture. The SuperCISC processor was developed with a 4-way very long instruction word (VLIW) core with a shared register file [8]. The idea is to accelerate the high incidence code segments (e.g. loops) that require large portions of the application runtime, called kernels, while also accelerating the remaining non-kernel code with the VLIW. These kernels are converted into entirely combinational hardware functions generated automatically from the C using a design automation flow [8]. In order to create a combinational hardware function, a technique called hardware predication is employed to remove the need for sequential logic. As a result, the Super Data Flow Graph (SDFG) can be transformed into a combinational hardware implementation.

III. Parameterized Reconfigurable Fabric Model

SDFGs retain a data flow structure allowing computational results to be computed in one 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 fabric model is

1-4244-0054-6/06/$20.00 ©2006 IEEE

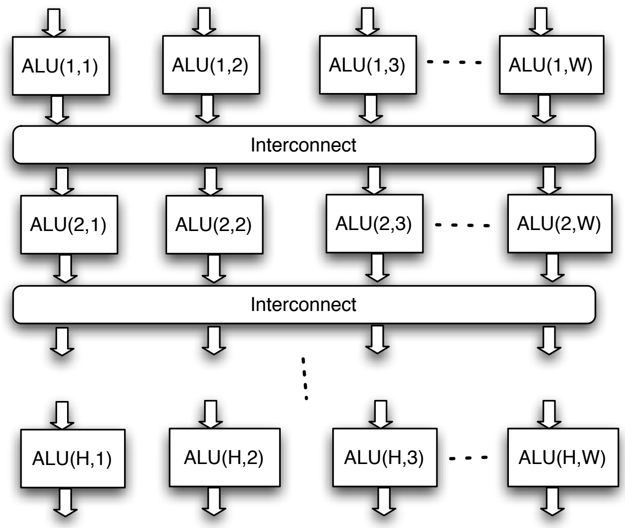


Fig. 1. The fabric model is comprised of Arithmetic Logic Units and a rec onfigurable interconnect.

purely cominational. The fabric model was implemented in

TABLE I

PARAMETERS OF THE FABRIC MODEL. FOR THE DESIGN SPACE EXPLORATION CONSIDERED HERE, FIXED OR LIMITED PARAMETERS ARE INDICATED IN **BOLD**.

Global Parameters	
ALU Datawidth	DW= **{8,16,32}**
Fabric Parameters	
Width of the fabric	W
Height of the fabric	H
Arithmetic and Logic Unit Parameters	
Number of operands	O= **2**
Number of operations	OP= **20**
Interconnect Parameters	
Multiplexer cardinality	C= **{2,4,8,16}**

parameterized VHDL using the `generic` capability of the VHDL language. Several parameters were included in the fabric model and are listed in Table I.

The fabric size is determined with the parameters specifying the width of the fabric W, height of the fabric H, and datawidth DW. W dictates to 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 each 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

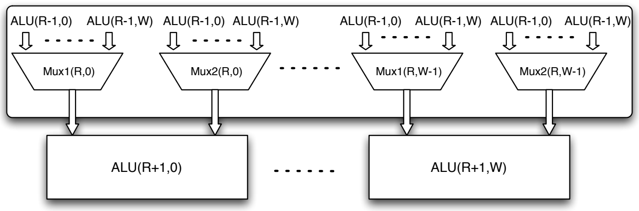


Fig. 2. The multiplexer-based interconnection stripe structure.

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.

IV. RESULTS

To begin studying the impact of various parameters of the fabric, the ALU and multiplexer elements were studied individually. The cardinality of the multiplexers was varied from 2 to 32 in powers of 2 and profiled for power consumption. Different architectural techniques for implementation of the ALU are also displayed including a study of varying the width of ALU. A technique similar to [9] was used to power profile various functional units and interconnect multiplexers with different sets of values of average input signal probability p, average transition density d and spatial correlation s. The details of power modeling and analysis can be found in [10]. Finally, the study of multiplexer cardinality usage is done based a set of core signal processing benchmarks, selected from the MediaBench benchmark suite.

A. Multiplexer Cardinality Impact on Power

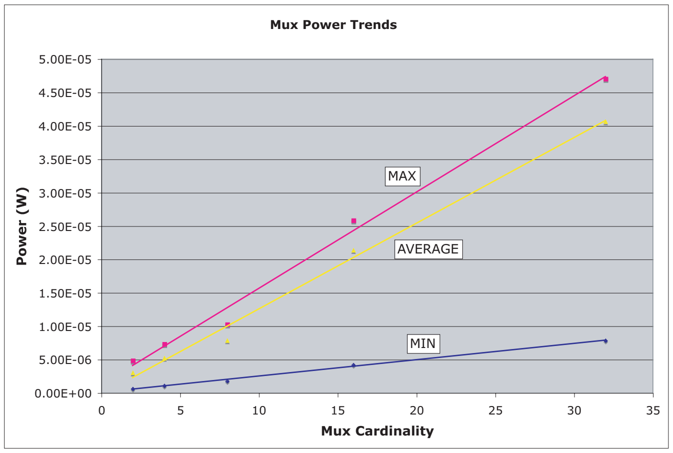


Fig. 3. Power consumption in multiplexers of different cardinalities. Results for a standard cell 160nm OKI ASIC process.

The power required in the interconnect depends heavily on the cardinality of the multiplexers in the interconnection stripe. To model this in part, the power impact of the cardinality of the multiplexer is shown by the trends from Figure 3. As shown in the figure, the maximum, minimum and average powers consumed in the multiplexers increases linearly with the cardinality. While not shown explicitly, reducing the cardinality also reduces the delay of the multiplexer. Thus, reducing the multiplexer complexity is desirable if warranted by the needs of the applications.

B. Architecture of the ALU

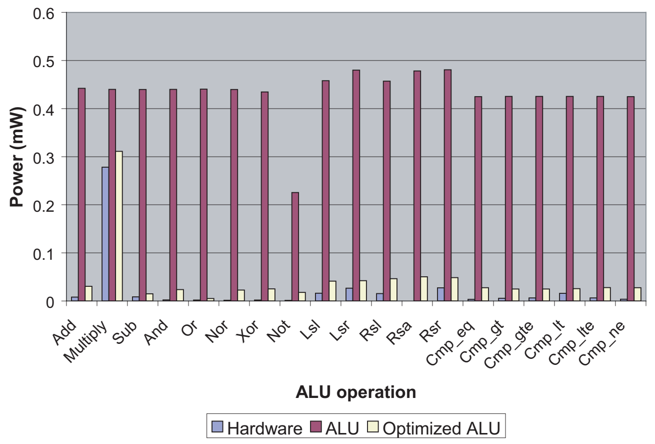


Fig. 4. Power results for several different functional unit implementation techniques. Results for a standard cell 160nm OKI ASIC process.

The power consumed in the fabric is also heavily dependent on the ALU power consumption. Several architectural techniques for implementing computations in the functional units were profiled including a high performance ALU, a power optimized ALU and individual functional units directly. These blocks were synthesized using 160nm OKI standard cells. The results are shown in Figure 4.

The $Hardware$ bar corresponds to the power results for the blocks independently synthesized for each operation. The ALU bar corresponds to a synthesizable ALU built structurally from the Mentor Moduleware components. It executes each function in parallel and selects the result using a multiplexer after the computation completes. Finally, the $Optimized\ ALU$ represents a low-power ALU in which latches are used at the input to each operation to avoid unnecessary switching of the rest of the hardware blocks when only a single operation is executed. The optimized ALUs are used as computational elements of the fabric.

The datawidth of each functional unit has a significant impact on power dissipation of the fabric. Thus, 8, 16 and 32-bit ALUs, which are candidates to be used as computational elements, have been power profiled for several ALU operations.

The results shown in Figure 5 reveal that as datawidth decreases by half, power dissipation also decreases by nearly 50% for all cases excepting multiplication. Because combinational multiplication has a stacking complexity it grows faster than other functional units, and the power reflects that

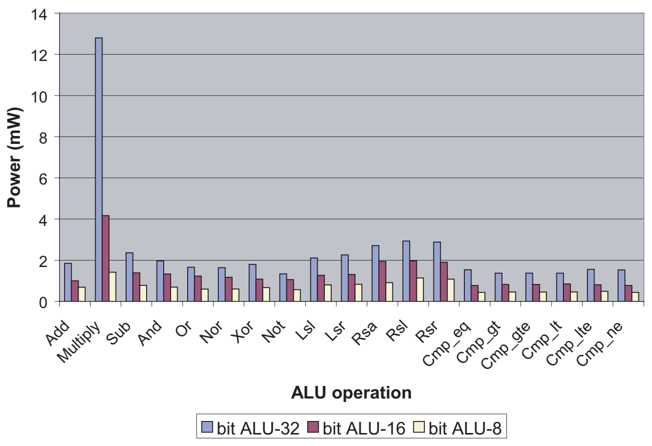


Fig. 5. Power consumption of ALU for different datawidths.

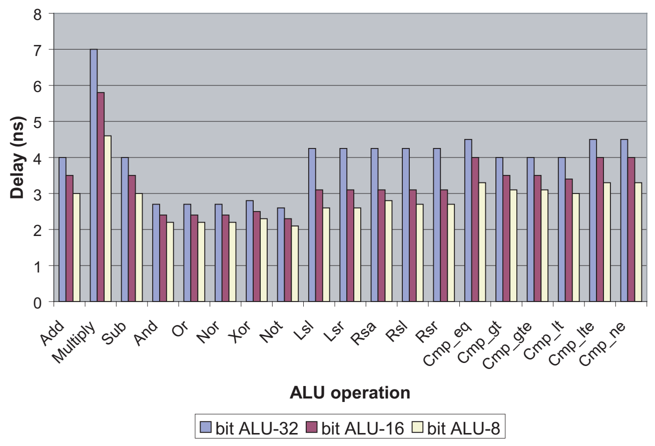


Fig. 6. Delay results of ALU for different datawidths.

decreasing to about 30% of the 32-bit version. There is a similar power trend between 16-bit and 8-bit operations. Figure 6 describes the latency of each bit-width. While as expected, the latency is lowest for the 8-bit ALU operations the change is only a nominal decrease over a 16-bit ALU. Even compared to a 32-bit ALU, the delay improvement is less than 50% at the cost of 3/4 of the bandwidth for the computation.

The energy results of ALUs of different datawidths are shown in Figure 7. This chart includes the power consumption of using a 32-bit wide ALU to compute both 16-bit and 8-bit values in comparison to computing them directly on a 16-bit or 8-bit wide ALU. This was done to calculate the energy consumption overhead of a 32-bit ALU used for lower bit width operations

Consider the dedicated width case (from left to right, the first, second, and fourth bars of each operation), as datawidth decreases by half, energy dissipation also decreases almost by half for most of the ALU operations except multiplier. In the case of multiplier, the energy is reduced to almost one-fourth when datawidth decreases by half. However, the overhead in

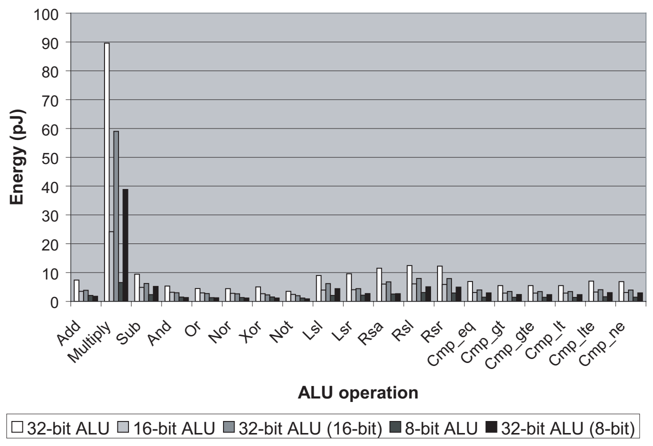
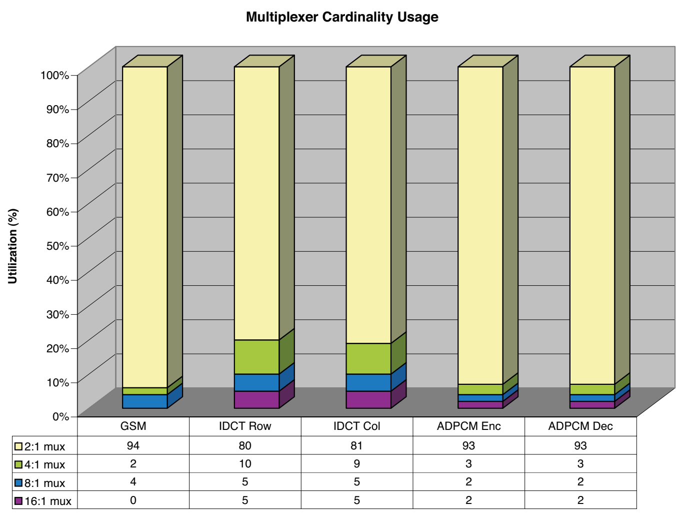


Fig. 7. Comparison of Energy consumption of 32-bit ALU, 16-bit ALU, 32-bit ALU used for 16-bit operations, 8-bit ALU and 32-bit ALU used for 8-bit operations.

using a 32-bit ALU for 16-bit operations as compared to a 16-bit ALU is 2.5X on the average. The same trend is observed if we compare the energy results of a 32-bit ALU used for 8-bit operations and a 8-bit ALU. However, when the multiplier is removed from consideration, the overheads are much lower. While, it appears that the overhead actually decreases for some of the logic operations, this is unlikely, and the difference between the two calculations does not exceed the estimated inaccuracy from using the PrimePower simulation over a SPICE level simulation.

C. Multiplexer Cardinality Usage



	GSM	IDCT Row	IDCT Col	ADPCM Enc	ADPCM Dec
2:1 mux	94	80	81	93	93
4:1 mux	2	10	9	3	3
8:1 mux	4	5	5	2	2
16:1 mux	0	5	5	2	2

Fig. 8. Multiplexer cardinality usage.

In order to study the multiplexer cardinality usage, SDFGs of the benchmark circuits were mapped to the fabric model by hand and through an automated mapping flow. The multiplexers were then categorized based on whether they would have been satisfied with a smaller multiplexer from the group of 2:1, 4:1, 8:1 and 16:1 multiplexers. For this analysis, the multiplexers are left justified, e.g. a 2:1 mux can select from the value just overhead, or one element to the right. A 4:1 mux selects from just overhead, and up to 3 values to the right, etc. As shown in Figure 8, the connections satisfied with 2:1 multiplexers is significant, 88%. The need for 16:1 multiplexers is only 3%. Since hardware predication operation is done in some of the benchmarks, we can not use 2:1 multiplexers for 3-operand operations. So, our analysis suggests that most of the connections can be justified using lower cardinality multiplexers like 4:1 multiplexers. Due to the trend of power consumption versus multiplexer cardinality shown in Figure 3, it is desirable to utilize 4:1 multiplexers wherever possible. Even though results suggested to use lower cardinality multiplexers, fixing the multiplexer cardinality was not possible due to the maturity of our tool flow, however, we expect to pursue this in our future work.

V. Conclusion and Future Work

In this paper, we described a generic and parameterized fabric model that exhibits ASIC-like power characteristics and FPGA-like programmability and tool support. The design space exploration results suggested the use of power optimized 32-bit width computational elements interconnected by low cardinality multiplexers like 4:1 multiplexers.

Our planned future work is to develop a mapper that can handle limited cardinality routing. By doing so, we expect to further improve power and performance results.

References

[1] E. Mirsky and A. Dehon, “Matrix: A reconfigurable computing architecture with configurable instruction distribution and deployable resources,” in *in Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines*, April 1996.

[2] C. Ebeling, D. C. Cronquist, and P. Franklin, “Rapid - reconfigurable pipelined datapath,” in *in the 6th International Workshop on Field-Programmable Logic and Applications*, 1996.

[3] 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.

[4] 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.

[5] R. Enzler, T. Jeger, D.Cottet, and G. Troster, “High-level area and performance estimation of hardware building blocks on fpgas,” in *Field-Programmable Logic and Applications Forum on Design Language*, 2000.

[6] 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.

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

[8] 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*, 2005, to appear.

[9] X. Liu and M. C. Papaefthymiou, “A markov chain sequence generator for power macromodeling,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, July 2004.

[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)*, 2005, to appear.