DOMAIN-SPECIFIC HYBRID FPGA: ARCHITECTURE AND FLOATING POINT APPLICATIONS

Chun Hok Ho[1], *Chi Wai Yu*[1], *Philip H.W. Leong*[2], *Wayne Luk*[1], *Steven J.E. Wilton*[3]

[1]Department of Computing
Imperial College London
London, England
{cho,cyu,wl}@doc.ic.ac.uk

[2]Dept. of Computer
Science and Engineering
Chinese University of Hong Kong
Hong Kong
phwl@cse.cuhk.edu.hk

[3]Dept. of Electrical
and Computer Engineering
University of British Columbia
Vancouver, B.C., Canada
stevew@ece.ubc.ca

ABSTRACT

This paper presents a novel architecture for domain-specific FPGA devices. This architecture can be optimised for both speed and density by exploiting domain-specific information to produce efficient reconfigurable logic with multiple granularity. In the reconfigurable logic, general-purpose fine-grained units are used for implementing control logic and bit-oriented operations, while domain-specific coarse-grained units and heterogeneous blocks are used for implementing datapaths; the precise amount of each type of resources can be customised to suit specific application domains. Issues and challenges associated with the design flow and the architecture modelling are addressed. Examples of the proposed architecture for speeding up floating point applications are illustrated. Current results indicate that the proposed architecture can achieve 2.5 times improvement in speed and 18 times reduction in area on average, when compared with traditional FPGA devices on selected floating point benchmark circuits.

1. INTRODUCTION

FPGA technology has been widely adopted to speed up computationally intensive applications. Most current FPGA devices employ an island-style fine-grained architecture, with additional fixed-function heterogeneous blocks such as multipliers and block RAMs; these have been shown to have severe area penalties compared with standard cell ASICs [1]. In this work, we propose domain-specific coarse-grained architectures which can have advantages in speed, density and power over more conventional heterogeneous FPGAs. One key issue associated with such an approach is identifying the correct amount of coarse-grained logic necessary to enhance the performance of an application without adversely affecting area and flexibility. For example, an application that demands high performance floating point computation can potentially achieve better speed and density by introducing dedicated embedded floating point units (FPUs). However, for those applications which do not have any floating point computations, the FPU resources will be wasted. To address this issue, we advocate domain-specific FPGAs with flexible, parameterised architectures that can be generated to address application sets that are smaller than those targeted by conventional FPGAs, but possibly larger than that of ASICs.

We introduce a hybrid FPGA model in which both fine-grained and coarse-grained units are considered important. Given a domain-specific application requirement, a reconfigurable fabric consisting of both types of units is generated, the coarse-grained units being used for the datapath and fine-grained units for control and bit-oriented operations. A model is also introduced that allows us to search for the best proportion of each type of fabric, and a method for rapidly evaluating the performance of the architecture is employed.

The key contributions of this paper are:

- A generic hybrid FPGA architecture that supports configurable resources of multiple granularity that can be customised for different applications.
- Use of this architecture to design a domain-specific hybrid FPGA for various floating point computations.
- Demonstration that a single configuration of a floating point specific hybrid FPGA is able to achieve improvements in both speed and area compared with commercial and proposed reconfigurable devices on selected floating point benchmarks.

The rest of this paper is organised as follows. Section 2 presents related work and illustrates certain commonly employed FPGA fabric architectures. Section 3 illustrates the hybrid FPGA architecture optimised for floating point computations; the issues and challenges associated with its design flow are also discussed. Section 4 demonstrates a methodology to model the proposed architecture. Section 5 contains results and analysis, and Section 6 concludes the paper.

2. BACKGROUND

2.1. Related work

FPGA architectures containing coarse-grained units have been reported in the literature. Compton and Hauck propose a domain-specific architecture which allows the generation of a reconfigurable fabric according to the needs of the application [2]. Ye and Rose suggest a coarse-grained architecture that employs bus-based connections, achieving a 14% area reduction for datapath circuits [3].

The study of embedded heterogeneous blocks for the acceleration of floating point computations has been reported by Roseler and Nelson [4] as well as Beauchamp et. al. [5]. Both studies conclude that employing heterogeneous blocks in designing an FPU can achieve area saving and increased clock rate over a fine grained approach.

In earlier work, we describe a methodology to estimate the impact of incorporating an embedded block in an existing FPGA [6]. In this paper, we employ this methdology to estimate the impact of including a floating-point coarse-grained embedded core.

2.2. FPGA architectures

The heart of an FPGA is a reconfigurable fabric. The fabric consists of arrays of fine-grained or coarse-grained units. A fine-grained unit usually implements a single function and has a single bit output. The most common fine-grained unit is a K-input lookup table (LUT), where K typically ranges from 4 to 6. The LUT can implement any boolean equation of K inputs. This type of fabric is called a LUT-based fabric. Several LUT-based cells can be joined in a hardwired manner to make a cluster. This results in little loss in flexibility but can reduce area and routing resources within the fabric [7].

A coarse-grained unit is usually less flexible and typically much larger than a fine-grained one, but is often more efficient for implementing specific functions. The coarse-grained unit is usually programmable to some degree, combining several functions such as those in an arithmetic logic unit (ALU). Outputs are often multibit. They can be parameterised in terms of features such as bus-width and functionality. As an example, the ADRES architecture [8] assumes that the wordlength and the functionality of a coarse-grained unit is the same as the attached processor. We have also proposed a word-based synthesisable architecture, and show that it has large improvements in area over a similar fine-grained approach [9].

Heterogeneous functional blocks are found on commercial FPGA devices. For example, a Virtex II device has embedded fixed-function 18-bit multipliers and a Xilinx Virtex 4 device has embedded DSP units with 18-bit multipliers and 48-bit accumulators. The flexibility of these blocks is limited and it is less common to build a digital system solely using these blocks. When the blocks are not used, they consume die area and contribute to increased delay without adding to functionality.

As shown in the above examples, FPGA fabric can have different levels of granularity. In general, a unit with smaller granularity has more flexibility, but can be less effective in speed, area and power consumption. Fabrics with different granularity can coexist as evident in many commercial FPGA devices. Most importantly, the above examples illustrate that FPGA architectures are evolving to be more coarse-grained and application-specific. The proposed architecture in this paper follows this trend, focusing on floating point computations.

3. HYBRID FPGA ARCHITECTURE

Requirements

Before we introduce the floating point hybrid FPGA architecture, common characteristics of what we consider a reasonably large class of floating point applications which might be suitable for signal processing, linear algebra and simulation are first described. Although the following analysis is qualitative, it is possible to develop the hybrid model in a quantitative fashion by profiling application circuits in a specific domain.

In general, FPGA based floating point application circuits can be divided into control and datapath portions. The datapath typically contains floating point operators such as adders, subtracters, and multipliers, and occasionally square root and division operations. The datapath often occupies most of the area in an implementation of the application. Existing FPGA devices are not optimised for floating point computations; floating point operators consume a significant amount of FPGA resources. For instance, if the embedded DSP48 block is not used, a double precision floating point adder requires 701 slices on a Xilinx Virtex 4 FPGA, while a double precision floating point multiplier requires 1238 slices on the same device [10].

The floating point precision is usually a constant within an application. The IEEE 754 standard is almost always used, especially the single precision format (32-bit) or double precision format (64-bit). The interconnection can be bus-oriented. The datapath can often be pipelined and routing within the datapath may be uni-directional in nature. Occasionally there is feedback in the datapath for some operations such as accumulation.

The control circuit is much simpler than the datapath and therefore the area consumption is typically lower. Control is usually implemented as a finite state machine and most synthesis tools can produce an efficient mapping from the boolean logic of the state machine into fine-grained FPGA resources.

Based on the above analysis, the following presents some basic requirements for floating point hybrid FPGA architectures.

- A number of coarse-grained floating point addition and multiplication blocks are necessary since most computations are based on these primitive operations. Floating point division and square root operators can be optional, depending on the domain-specific requirement.
- Coarse-grained interconnection, fabric and bus-based operations are required to allow efficient implementation and connection between fixed-function operators.
- Dedicated output registers for storing floating point values are required to support pipelining.
- Fine-grained units and suitable interconnections are required to support implementation of state machines and bit-oriented operations. These fine-grained units should be accessible by the coarse-grained units and vice versa.

Architecture

Figure 1 shows a top-level block diagram of our hybrid FPGA architecture. It employs an island-style fine-grained FPGA structure with dedicated columns for coarse-grained units. Both fine-grained and coarse-grained units are reconfigurable. The coarse-grained part contains embedded fixed-function floating point adders and multipliers.

The top-level architecture is inspired by existing commercial FPGAs. However, the proportion of coarse-grained blocks can be customised to meet design requirements. The island-style architecture with standard interconnect structures such as connection and switch boxes are used to implement the fine-grained fabric.

Throughout this paper, we employ a 130nm technology. To make our results consistent, we build our architecture around the Virtex II device since it employs a comparable process technology (0.15μm/0.12μm). Four input LUT-based fine-grained units, similar to Xilinx Virtex II slices, are hence employed. However, the proposed FPGA hybrid modelling discussed in Section 4 is general and allows us to adopt other architectures such as the 6 input LUTs in Virtex 5 and Stratix III. We believe the same trends would be seen as we migrate to smaller technologies and more modern FPGA architectures.

The datapath for the floating point units is implemented using coarse-grained logic. The coarse-grained logic consists of a number of coarse-grained units embedded into the fine-grained fabric. The architecture of the coarse-grained units, inspired by previous work [3, 9], is shown in Figure 2. It is parameterised to support different proportions of fine and coarse-grained elements, the parameters being detailed in Table 1. There are D blocks in a unit, P of them are floating point multipliers, another P of them are floating point adders and the rest ($D - 2P$) are wordblocks.

Symbol	Parameter Description
D	Number of blocks (Including FPUs, wordblocks)
N	Bit Width
M	Number of Input Buses
R	Number of Output Buses
F	Number of Feedback Paths
P	Number of Floating Point Adders and Multipliers

Table 1: Parameters for the coarse-grained unit.

The floating point multiplier block is a fixed-function block. The floating point adder block can be configured for either floating point addition or subtraction. This is achieved by XORing the sign bit with the configuration bit. Each FPU has a reconfigurable registered output and associated control input and status output signals. The control signal is a write enable signal that controls the output register. The status signals report the FPU's status flags and include those defined in IEEE standard as well as a zero and sign flag. The fine-grained unit can monitor these flags as routing paths exist between them.

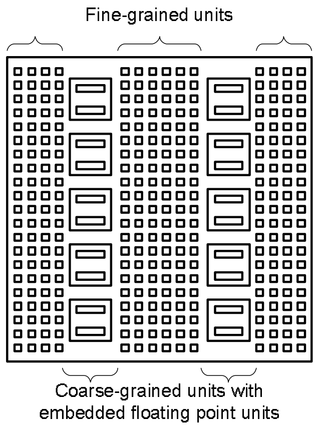

Figure 1: Architecture of the floating point hybrid FPGA.

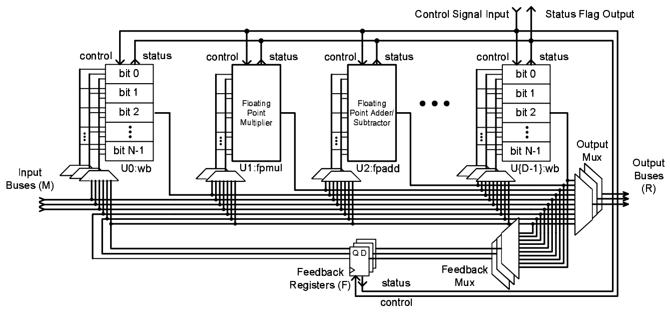

Figure 2: Architecture of the coarse-grained unit.

A wordblock contains N identical bitblocks, and is similar to published designs [9]. A bitblock contains two 4-input

LUTs and a reconfigurable output register. The value of N depends on the size of the FPU. Bitblocks within a wordblock are all controlled by the same set of configuration bits, so all bitblocks within a wordblock perform the same function. A wordblock, which includes a register, can efficiently implement operations such as addition and multiplexing. Similar to FPUs, wordblocks generate status flags such as MSB, LSB, carry out, overflow and zero which are connected to the fine-grained blocks.

Apart from the control and status signals, there are M input buses and R output buses connected to the fine-grained units. The routing layout assumes that a block can only accept inputs from the left, simplifying the routing. To allow more flexibility, F feedback registers have been employed so that a block can accept the output from the right block through the feedback registers. For example, the first block can only accept input from input buses and feedback registers, while the second block can accept input from input buses, the feedback registers and the output of the first block. The feedback registers latch the output of a block and forward it to another block. Each floating point multiplier is logically located to the left of a floating point adder so that no feedback register is required to support multiply-and-add operations. The coarse-grained units can support multiply-accumulate functions by utilising the feedback registers.

Switches in the coarse-grained unit are implemented using multiplexers and are bus-oriented. A single set of configuration bits is required to control these multiplexers, improving density compared to a fine-grained fabric. For the same reason, the FPUs are embedded in the coarse-grained units rather than distributed over the FPGA, such that an FPU can exploit the bus-oriented routing resources in the coarse-grained blocks.

4. MODELLING OF A HYBRID FPGA

A methodology, building on our earlier work [6, 9], is used to model floating point hybrid FPGAs with different architectural parameters and coarse-grained blocks as described in Section 3. This approach is general and can be used to model any FPGA provided that a floorplanner and a timing analysing tool are available for that device. In this methodology, an existing fine-grained commercial FPGA is used. Fine-grained blocks in our hybrid FPGA are directly mapped to the corresponding logic cells on the commercial FPGA.

The area and delay for the embedded coarse-grained units are first estimated by synthesising the design using a standard cell flow. They are then modelled in a commercial FPGA by employing blocks of logic cells with similar delay and area. The corresponding vendor's CAD tools are then used to estimate the delay and area of the hybrid FPGA. Overheads such as crossing clock domains are not considered in this work, nor are alternative approaches such as full custom design.

We employ a parameterised synthesisable IEEE 754 compliant floating point library in our experiments. The library supports four rounding modes and denormalised numbers. A floating point multiplier and floating point adder are generated and synthesised using a standard cell library design flow. The Synopsys Design Compiler is used for synthesis. During synthesis, retiming optimisation is enabled to obtain better results.

While a custom layout design for the coarse-grained unit can achieve much higher density and better speed, it is time consuming to design a coarse-grained unit for each set of architectural parameters. To allow us to explore different parameterised coarse-grained units, we employ a synthesisable flow which supports different granularities. To determine suitable parameters for generation of coarse-grained units, we first decide on an initial set of parameters and try to map a set of benchmark circuits to the units. Two parameters determine whether the architecture is best-fit. The first is the number of coarse-grained units required to implement the circuit. The second is the percentage of blocks used in a unit.

The best-fit architecture can be determined by varying the parameters to produce a design with the least number of units with maximum density on the benchmark circuits. Extra wordblocks are added to the design, allowing more flexibility for implementing other circuits outside of the benchmark set. Manual mappings are performed for each benchmark. Once the parameters are determined, a Verilog netlist is generated and synthesised together with soft-core FPUs using the Synopsys Design Compiler (a 130nm process is assumed throughout). Area information is obtained from the tool directly. Timing information, however, cannot be determined before programming the configuration bits.

During manual mapping, a set of configurations is generated and can be used in timing analysis. We use the case analysis feature provided in the Synopsys Design Compiler which takes configuration bits into account in the timing analysis.

The architectural parameters: 9 blocks ($D = 9$), 4 input buses ($M = 4$), 3 output buses ($R = 3$), 3 feedback registers ($F = 3$), 2 floating point adders and 2 floating point multipliers ($P = 2$) are determined empirically by trial-and-error as explained above. We generate double precision coarse-grained fabrics so the buswidth is 64.

LUT-based fine-grained units are mature in terms of architecture and design flow. They have been widely adopted in commercial FPGAs. We have employed a methodology called virtual embedded blocks (VEB) [6] to model fine-grained units in our architecture. The VEB flow allows the evaluation of embedded elements on FPGA devices by creating dummy logic cells that model the timing and area of the embedded elements.

During the first step, we create an HDL description of the control logic part of the application circuit. We then add additional statements which instantiate the coarse-grained units explicitly, as well as the signals between the fine-grained and coarse-grained units. The design is then synthesised on the target device and a device-specific netlist is generated. The synthesis tool considers the coarse-grained unit as a black box. The area utilisation is computed by determining the number of slices in Virtex II [11] required to implement the application.

The second step is to obtain the timing and area models for each instantiated coarse-grained unit as described earlier. With this information, a VEB netlist can be compiled by generating dummy cells with appropriate area and delay. Special consideration is given to the interface between fine-grained units and coarse-grained units to make sure that the corresponding VEB model has sufficient I/O pins to connect to the fine-grained routing resources. This can be verified by keeping track of the number of inputs and outputs which connect to the global routing resources in a slice. For example, it is not possible to have a VEB model which has area of 4 slices but demands 33 inputs and 9 outputs, as we assume one slice in Virtex II can only support 8 inputs and 2 outputs. Also, as we cannot route the configuration clock and configuration input pin to a coarse-grained unit, there are two programming pins connected to the I/O of the host FPGA which act as the configuration port for the coarse-grained unit.

After generating the VEB netlist for the targeted FPGA, a User Constraint File (UCF) which forces the VEB to be located in a particular column is created. We then use the vendor's place and route tool to obtain the final area and timing results. This represents the characterisation of a circuit implemented on the hybrid floating point FPGA with fine-grained units and routing resources exactly the same as the targeted FPGA.

Using commercial FPGA fine-grained units in this manner has several advantages, since commercial quality synthesis and place and route tools can be used in the modelling of the hybrid FPGA. It can produce a realistic comparison to existing FPGA devices. Furthermore, optimisations such as retiming are available.

5. RESULTS

A set of benchmark applications are mapped to the proposed floating point hybrid FPGA, and the results are compared to a Virtex II device. This section introduces the circuits and gives an example of mapping one of the circuits. A double precision floating point hybrid FPGA is assessed. All FPGA results are obtained using the Synplicity Synplify Premier 8.5 for synthesis and Xilinx ISE 8.1i design tools to place and route. All ASIC results are obtained using Synopsys Design Compiler V-2004.06.

Six benchmark circuits are used in this study [6]. Five of them are computational kernels and one is a Monte Carlo simulation datapath. We have chosen these circuits since they are simple but are not very efficiently implemented on general-purpose FPGA devices. We expect these applications to yield better timing and density on a floating point hybrid FPGA.

The *bfly* benchmark performs the computation $z = y + x * w$ where the inputs and output are complex numbers; this is commonly used within a Fast Fourier Transform computation. The *dscg* circuit is the datapath of a digital sine-cosine generator. The *fir4* circuit is a 4-tap finite impulse response filter. The *mm3* circuit performs a 3-by-3 matrix multiplication. The *ode* circuit solves an ordinary differential equation. The *bgm* circuit computes Monte Carlo simulations of interest rate model derivatives priced under the Brace, Gątarek and Musiela (BGM) framework.

In the mapping of each circuit, we assume that the two floating point multipliers in the coarse-grained unit are located at the second and the sixth block. The two floating point adders are located in the third and the seventh block. All other parameters are given in Section 4.

The physical die area of a Virtex II device has been reported [11], and the normalisation of the area of coarse-grained unit is estimated in Table 2. We assume that 60% of the total die area is used for slices; the rest of the area is due to I/O pads, block memory, multipliers etc. This means that the assumed area of our Virtex II device is 10,912μm^2. This number is normalised against the feature size (0.15μm). A similar calculation is used for the coarse-grained units. The synthesis tool reports that the area of a double precision coarse-grained block is 1,256,570μm^2. We further assume 15% overhead after place and route based on our experience [9]. The area values are normalised against the feature size (0.13μm). The number of equivalent slices is obtained through the division of coarse-grained unit area by slice area. This shows that the double precision coarse-grained unit would take up 176 slices. The values in the sixth and seventh columns represent the number of I/O required, while the values in brackets indicate the maximum number of I/O allowed for the area in slices.

Although a Virtex II slice employs smaller transistors (0.12μm) than those used for building the coarse-grained unit (0.13μm), we do not scale the timing of the coarse-grained unit and therefore conservative timing results are reported.

We use XC2V3000-6-FF1152 as the host FPGA for the floating point hybrid FPGA. We assume that 12 double precision coarse-grained blocks are embedded into this FPGA. The coarse-grained blocks constitute 15% of the total area in an XC2V3000 device. The mapping is performed as described in Section 4. Benchmark circuits are implemented on the same device and the results are shown in Table 3.

The FPU values for the XC2V3000 device (seventh column) are estimated from the distribution of LUTs, which is reported by the synthesis tool. The logic area (eighth column)

Fabric	Area (A) (μm^2)	Feature Size (L) (μm)	Normalised Area (A/L^2)	Area in Slices	Input Pin	Output Pin
Virtex II Slice	10,912	0.15	485,013	1	8(8)	2(2)
DP-CGU	1,445,056	0.13	85,506,242	176	285 (1408)	258(352)

Table 2: Normalisation on the area of the coarse-grained units against a Virtex II slice. DP stands for double precision floating point arithmetic. CGU stands for coarse-grained unit. 15% overheads are applied on the coarse-grained units as shown in the second column.

	Double precision floating point hybrid FPGA					XC2V3000-6-FF1152				Reduction	
Circuit	number of CGU	CGU area (slices)	FGU area (slices)	Total Area (slices)	Delay (ns)	FPU area (slices)	Logic area (slices)	Total Area (slices)	Delay (ns)	Area (times)	Delay (times)
bfly	2	352 (2.5%)	213 (1.49%)	565 (3.9%)	9.02	12,813 (89%)	920 (6%)	13,733 (96%)	24.57	24.3	2.72
dscg	2	352 (2.5%)	309 (2.16%)	661 (4.6%)	10.11	9,287 (65%)	327 (2%)	9,614 (67%)	22.78	14.5	2.25
fir4	2	352 (2.5%)	19 (0.13%)	371 (2.6%)	9.06	11,143 (78%)	147 (1%)	11,290 (79%)	23.68	30.4	2.61
mm3	2	352 (2.5%)	290 (2.02%)	642 (4.5%)	8.9	8,071 (56%)	818 (6%)	8,889 (62%)	23.40	13.8	2.63
ode	2	352 (2.5%)	193 (1.35%)	545 (3.8%)	9.74	7,933 (55%)	305 (2%)	8,238 (57%)	21.93	15.1	2.25
bgm*	7	1232 (8.6%)	578 (4.03%)	1,810 (12.6%)	10.00	29,758 (208%)	539(4%)	30,207 (211%)	24.34	16.7	2.43
									Geometric Mean:	18.3	2.48

Table 3: Double precision floating point hybrid FPGA results. CGU stands for coarse-grained unit and FGU stands for fine-grained unit.Values in the brackets indicate the percentages of slices used in a XC2V3000 device. *Circuit *bgm* cannot be fitted in a XC2V3000 device. The area and the delay are obtained by implementing on a XC2V6000 device.

is obtained by subtracting the FPU area from the total area reported by the place and route tool. As expected, the FPU logic occupies most of the area, typically more than 90% of the user circuits. Although the *bfly* circuit cannot fit in an XC2V3000 device, it can be tightly packed into a few coarse-grained blocks. For example, the circuit *bfly* has 8 FPUs which consume 89% of the total FPGA area. They can fit into 2 coarse-grained units, which constitute just 2.5% of the total FPGA area. Delay is reduced by 2.5 times on average. As the critical paths are in the FPU, improving the timing of the FPU through full-custom design would further increase the overall performance. The area reduction is significant: the proposed architecture can reduce the area by 18 times. The saving is achieved by (1) embedded floating point operators, (2) efficient directional routing and (3) sharing configuration bits.

6. CONCLUSION

We present a hybrid FPGA architecture which involves a combination of reconfigurable fine-grained and coarse-grained units dedicated to floating point computations. A parameterisable modelling framework is proposed which allows us to explore different configurations of this architecture. We show that the proposed floating point hybrid FPGA enjoys improved speed and density over a conventional FPGA for a variety of applications. Current and future work includes developing automated design tools supporting facilities such as partitioning for coarse-grained units, and exploring further architectural customisations for a large number of domain-specific applications.

Acknowledgements

The authors gratefully acknowledge the support of the UK EPSRC (grant EP/C549481/1 and grant EP/D060567/1).

References

[1] I. Kuon and J. Rose, “Measuring the gap between FPGAs and ASICs,” in *Proc. FPGA*. New York, NY, USA: ACM Press, 2006, pp. 21–30.

[2] K. Compton and S. Hauck, “Totem: Custom Reconfigurable Array Generation,” in *Proc. FCCM*, 2001, pp. 111–119.

[3] A. Ye and J. Rose, “Using Bus-Based Connections to Improve Field-Programmable Gate-Array Density for Implementing Datapath Circuits,” *IEEE Trans. VLSI*, vol. 14, no. 5, pp. 462–473, 2006.

[4] E. Roesler and B. Nelson, “Novel Optimizations for Hardware Floating-Point Units in a Modern FPGA Architecture,” in *Proc. FPL*, 2002, pp. 637–646.

[5] M. Beauchamp, S. Hauck, K. Underwood, and K. Hemmert, “Embedded floating-point units in FPGAs,” in *Proc. FPGA*, 2006, pp. 12–20.

[6] C. Ho, P. Leong, W. Luk, S. Wilton, and S. Lopez-Buedo, “Virtual Embedded Blocks: A Methodology for Evaluating Embedded Elements in FPGAs,” in *Proc. FCCM*, 2006, pp. 35–44.

[7] E. Ahmed and J. Rose, “The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density,” *IEEE Trans. VLSI*, vol. 12, no. 3, pp. 288–298, March 2004.

[8] B. Mei, S. Vernalde, D. Verkest, H.D. Man, and R. Lauwereins, “ADRES: An Architecture with Tightly Coupled VLIW Processor and Coarse-Grained Reconfigurable Matrix,” in *Proc. FPL*, 2003, pp. 61–70.

[9] S. Wilton, C. Ho, P. Leong, W. Luk, and B. Quinton, “A Synthesizable Datapath-Oriented Embedded FPGA Fabric,” in *Proc. FPGA*, 2007, pp. 33–41.

[10] Xilinx Inc., *Floating-Point Operator v1.0*. Product Specification, 2005.

[11] C. Yui, G. Swift, and C. Carmichael, “Single event upset susceptibility testing of the Xilinx Virtex II FPGA,” in *Military and Aerospace Applications of Programmable Logic Conference (MAPLD)*, 2002.