THE COARSE-GRAINED / FINE-GRAINED LOGIC INTERFACE IN FPGAS WITH EMBEDDED FLOATING-POINT ARITHMETIC UNITS

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

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

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

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

ABSTRACT

This paper examines the interface between fine-grained and coarse-grained programmable logic in FPGAs. Specifically, it presents an empirical study that covers the location, pin arrangement, and interconnect between embedded floating point units (FPUs) and the fine-grained logic fabric in FPGAs. The results show that (1) FPUs should be square, (2) FPUs should be positioned tightly near the center of the FPGA and (3) that the FPU pins should be arranged on four sides of the FPU.

1. INTRODUCTION

Significant improvements in the performance, logic density, and power efficiency of Field-Programmable Gate Arrays (FPGAs) have made them useful for implementing nearly any type of digital application. In early FPGAs, significant improvements were made by optimizing the fine-grained programmable logic and routing architecture of the FPGA. Today, further improvements are being made by embedding coarse-grained elements such as memories, multipliers, and processors within the fine-grained programmable fabric of the FPGA.

Coarse-grained elements can implement a specific function more efficiently than fine-grained programmable logic. However, since they are not as flexible, they only benefit applications which utilize them. This limits the types of embedded blocks which are commercially viable in general-purpose FPGAs to very common circuit elements such as memories, adders, and multipliers. For domain-specific FPGAs, however, additional embedded blocks may make sense. For example, an FPGA that is built specifically to implement applications containing a significant amount of floating point computation would benefit from embedded floating point units. This was explored in [1], in which a domain-specific FPGA that incorporates coarse-grained floating point units (FPUs) was described. The results in [1] show that the embedded floating point units lead to an 18 times density improvement for a set of floating point datapath circuits.

An important consideration when adding coarse-grained embedded elements to an FPGA is the interface between the coarse-grained and fine-grained resources. If this interface is not flexible enough, the usefulness of the embedded block will be reduced, since connections to and from the block will be expensive. On the other hand, if the interface is too flexible, it will require too much area and delay, possibly negating the density and performance advantages of including the embedded block, and resulting in unnecessary overhead for applications that do not use the embedded component.

In this paper, we examine this interface. We focus on architectural issues, such as the location of the embedded elements, and the interconnect between the embedded elements and the fine-grained fabric. Our approach is presented in the context of the embedded floating point blocks described in [1].

Specifically, the key contributions of this paper are:

- a set of parameters that describes the interface between coarse-grained and fine-grained programmable logic in FPGAs.
- an empirical framework to model the impact of coarse-grained architectural parameters in terms of performance, density, and power consumption.
- an empirical study that examines:
  1. where the coarse-grained FPUs should be embedded within FPGAs,
  2. where the pins of the FPUs should be on the periphery,
  3. how flexible the interconnect between the FPUs and the fine-grained logic should be,
  4. what shape the FPU should have.

Although the empirical study focuses on FPGAs with embedded FPUs, the conclusions of the study may be applicable to other types of embedded computational blocks.

This paper is organized as follows. Section 2 describes related work. Section 3 illustrates the interface between coarse and fine-grained logic and presents corresponding parameters to describe this interface. Section 4 then presents the empirical framework used to evaluate different interface schemes. Finally, Section 5 presents our results and analysis, and Section 6 summarizes our conclusions.

2. BACKGROUND

Conventional island-style FPGAs consist mainly of a fine-grained programmable fabric that is made up of configurable logic blocks (CLBs), programmable routing resources, and programmable I/Os. The CLBs consist of one or more k-input lookup tables (k-LUT) and fast local interconnect. Each k-LUT can implement any single output function with k inputs or less. The routing resources implement the interconnect between the CLBs and the I/Os.

A significant number of studies have focused on optimizing this type of FPGA architecture to minimize area, critical-path delay, and power consumption. As an example, the study described in [2] compares different aspects of segmented routing architectures, such as wirelength distribution, switch block implementation, and connection block flexibility, with the goal of creating a fast and area-efficient general-purpose FPGA architecture.

More recent work has focused on adding coarse-grained blocks within the fine-grained fabric. Examples of this include embedded arithmetic multipliers [3, 4] and embedded processors [4]. Coarse-grained blocks improve area and delay since they can implement specific functions more efficiently than the fine-grained logic [5]. On the other hand, coarse-grained blocks waste area when they are not used by an application. FPGAs vendors must consider this tradeoff to determine the type and number of coarse-grained blocks that should be embedded within their devices.

In order to take further advantage of coarse-grained blocks, domain-specific hybrid FPGAs target a specific application domain. In doing so, greater area and delay savings can be achieved for certain types of applications since the amount of coarse-grained logic can be tailored for those applications. A number of recent approaches have been proposed in the literature. In [6], a coarse-grained architecture with bus-based interconnect has been shown to reduce area for datapath circuits. In [7], a tool that generates a domain-specific reconfigurable fabric that is tailored to a specified set of application has been proposed. In [8], the QUKU architecture which merges coarse-grained reconfigurable processing element array and FPGA architectures has been described. This two-level reconfigurable architecture provides active support for fast and efficient dynamic reconfiguration. Enzler et. al. [9] has proposed a framework for the cycle-accurate performance evaluation of hybrid reconfigurable processors on the system level, which is based on data-streaming applications. In [1], a domain-specific hybrid FPGA architecture that targets floating point arithmetic applications by incorporating floating point units within a fine-grained programmable fabric has been presented; this architecture is shown to be 18 times more area-efficient than a purely fine-grained architecture for floating point arithmetic applications. One of the key parts of an FPGA with embedded coarse-grained blocks is the routing structure between the embedded blocks and the fine-grained logic resources. If the coarse-grained/fine-grained interface is not flexible enough, many applications will be unroutable. On the other hand, if the interface is overly flexible, the routing resources will be slower and consume more area than is necessary. Although a number of studies have proposed new coarse-grained blocks and hybrid FPGA architectures, few have examined the interface between the coarse-grained blocks and fine-grained fabric in significant detail. In [10], the local routing resources that connect CLBs to the FPGA routing resource are shared with the embedded blocks to minimize the overall area penalty when adding the embedded blocks. This technique, called *shadow clustering*, is useful for embedded blocks with similar I/O pin densities as the existing CLBs; however, for embedded blocks which has higher I/O pin densities than the existing routing resources are not sufficient. In [11], the interface between embedded memory blocks and fine-grained programmable logic is examined. Memories are quite different from computation blocks, and so we expect that the interface presented in [11] would not be suitable for our problem.

3. COARSE/FINE-GRAINED INTERFACE

In this section, we describe the architecture of the blocks used in this work. We first present our assumptions regarding the fine and coarse-grained logic and then give a description of a generic interface architecture with parameters that cover the space of architectures considered.

3.1. Fine-Grained FPGA Assumptions

We assume that the fine-grained resources in the FPGA consist of a grid of identical configurable logic blocks (CLBs), each containing N Basic Logic Elements (BLEs). Each BLE contains a k-LUT and flip flop. We assume that each CLB also contains support for carry chains, shift registers, internal multiplexers and XOR gates.

The CLBs are connected using horizontal and vertical channels, as described in [2]. Each channel contains W parallel routing tracks of length 1 and is connected to neighbouring CLBs using a connection block, and intersecting channels using a switch block. We use the subset switch block (also known as disjoint) with $Fc_{switch} = W$, $Fs = 3$, $Fc_{output} = 1$, $Fc_{input} = 1$ and $Fc_{pad} = 1$ [2].

3.2. Coarse-Grained Block Assumptions

We adopt the coarse-grained floating point blocks described in [1]. Each coarse-grained block contains two double precision floating point adders, two double precision floating point multipliers, and five wordblocks (each bit comprising a 4-LUT and register) which can efficiently implement operations such as addition and multiplexing as shown in Figure 1.

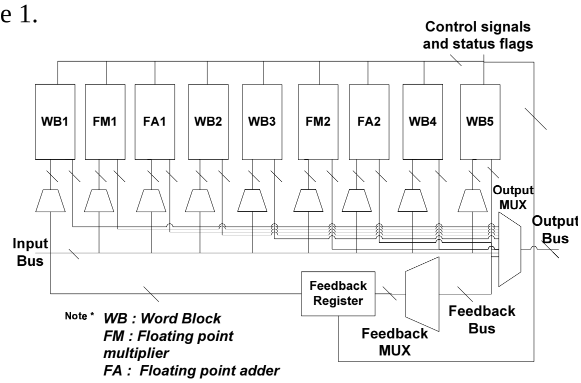


Fig. 1. coarse-grained unit modelled in this paper

3.3. Coarse-Grained Interface

Based on our detailed area model, we estimate that our embedded block (EB) consumes roughly the same amount of area as 182 tiles. Each tile represents a CLB and its associated interconnect, buffer and configuration bit. To embed an EB, we remove a 13x14 grid of CLBs, and replace them with a single EB. Figure 2 shows an example of replacing 3x3 grid of CLBs by a single EB. We assume that the EB pins connect to the routing architecture through connection blocks, similar to those used for CLBs. Although other connection patterns are possible (see [11], for example), this pattern allows us to minimize the number of changes to the existing FPGA routing architecture, so that we can leverage the significant amount of previous work on FPGA routing structures. We also assume that the gridded routing fabric extends over the embedded block, as shown in Figure 2. Given the large number of metal layers available in modern CMOS processes, it is reasonable that tracks can easily be placed on top of the embedded blocks. In Figure 2, the four switch blocks required at the interface of the horizontal and vertical channels must co-exist with the embedded block; the embedded block, which takes the same area as nine CLBs, includes the area of these four switch blocks. Although it would be possible to consider architectures in which the grid is ”broken” [12], it would require changes to the detailed routing architecture.

3.4. Interface Parameters

In this paper, we consider a range of interface architectures. To describe the space of architectures that we consider, we define the following parameters:

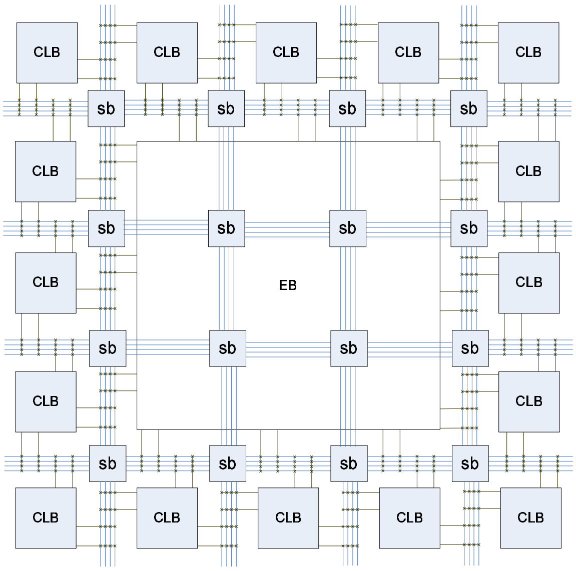
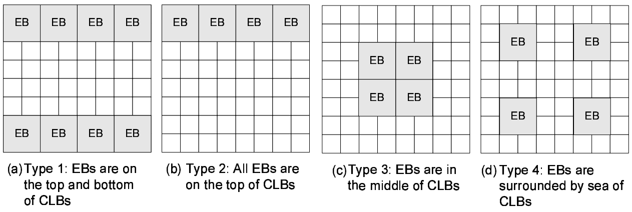


Fig. 2. connection between coarse and fine-grained units through switch box (sb)

1. EB position: The embedded blocks can be placed in various places within the FPGA. In this paper, we consider the positions as shown in Figure 3.



(a) Type 1: EBs are on the top and bottom of CLBs

(b) Type 2: All EBs are on the top of CLBs

(c) Type 3: EBs are in the middle of CLBs

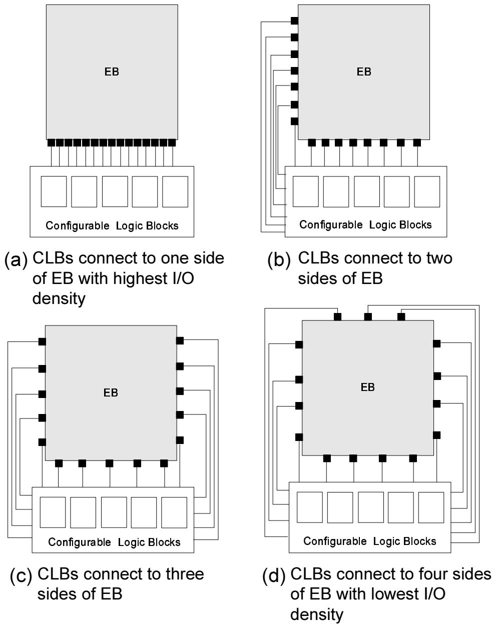
(d) Type 4: EBs are surrounded by sea of CLBs

Fig. 3. various positions of the EBs relative to the fine-grained CLBs

2. Pin Location: Figure 4 shows several strategies for positioning the pins of each EB. Strategy (a) has the highest I/O density, but may be suitable if signals from the I/O block are to be combined using a small set of CLBs. Strategies (b), (c), (d) have lower I/O density, but may result in longer connections if signals from more than one side of the EB are to be connected to the same CLB(s).

3. Channel Width: The width of the channels surrounding the EB has a significant impact on the routability of the device. Since our EB has a large number of pins, congestion around the EB may happen so it is desirable to relieve this congestion by using wider channels.

4. Shape: Several layouts of each embedded block are possible. We consider various aspect ratios.


EB
Configurable Logic Blocks
EB
Configurable Logic Blocks
EB
Configurable Logic Blocks
EB
Configurable Logic Blocks

(a) CLBs connect to one side of EB with highest I/O density

(b) CLBs connect to two sides of EB

(c) CLBs connect to three sides of EB

(d) CLBs connect to four sides of EB with lowest I/O density

Fig. 4. different pin positions in EB

4. METHODOLOGY

We employ an empirical methodology to examine the impact of the interface parameters described in the previous section. This section describes the benchmark circuits, the CAD tools, and the model that are used.

4.1. Domain-specific Benchmark Circuits

We use six double precision floating point benchmark circuits [13]. They are: (1) *bfly*, the basic component of Fast Fourier Transform: $z = y + x * w$ using complex numbers, (2) *dscg*, a digital sine-cosine generator, (3) *fir4*, a 4-tap finite impulse response filter, (4) *mm3*, a 3x3 matrix multiplication circuit, (5) *ode*, an ordinary differential equation solver, (6) *bgm*, a datapath to compute Monte Carlo simulations of interest rate model derivatives priced under the Brace, Gątarek and Musiela (BGM) framework.

These benchmarks are chosen since they each involve a significant amount of floating point computation. Since *bfly*, *dscg*, *fir4*, *ode* and *mm3* contain a small number of fine-grained units, each core is replicated four times and are connected together. For example, a *dscg* benchmark contains four *dscg* cores connected together. All circuits use a single global clock. The number of FPUs and CLBs used for each benchmark circuit is shown in Table 1.

4.2. VPH: Versatile Place and Route for Hybrid FPGAs

We use the evaluation tool VPH to explore our architectures. VPH is a modified version of the VPR tool, with support for embedded blocks, complex logic blocks, carry chains, and constraint files [14]. In the VPH design flow, shown in Figure 5, applications and coarse-grained elements are written in a high level hardware description language (VHDL) and synthesized to a mapped library netlist in VHDL format using Synplicity's Synplify Premier 8.5 tool. The library netlist contains the usage and connection of simple units such as registers, LUTs, internal multiplexors and internal inverters. The basic logic block packing tool, VPHpack, packs these units into basic logic elements (BLEs). VPHpack clusters BLEs into CLBs.

Benchmarks	bgm	dscg	bfly	ode	mm3	fir4
No. of CLB	6433	647	790	336	773	180
No. of FPU	7	8	8	8	8	8

Table 1. Number of FPU and CLB used in each benchmark circuit

A user constraint file (.ucf) is used to specify the FPGA area and the absolute position of each embedded block. A separate constraint file for each embedded block is used to specify the area, the pin position and the timing information for the EB; the area and delay information for each block is obtained using Synopsys Design Compiler V-2004.06. As in VPR, an architecture file specifies the fine-grained FPGA's architectural parameters, such as timing delay of the LUT. Using these files, the VPH tool performs placement, routing, and timing analysis to produce area and delay estimates for each benchmark circuit.

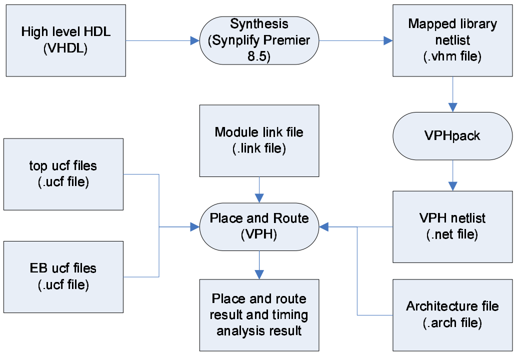

High level HDL (VHDL)
Synthesis (Synplify Premier 8.5)
Mapped library netlist (.vhm file)
VPHpack
Module link file (.link file)
top ucf files (.ucf file)
Place and Route (VPH)
VPH netlist (.net file)
EB ucf files (.ucf file)
Place and route result and timing analysis result
Architecture file (.arch file)

Fig. 5. Design flow of exploration using VPH

5. RESULTS AND DISCUSSION

In this section, the impact of the interface parameters in Section 3 on hybrid FPGAs is studied. In the experiments conducted, the default architecture parameters are: (1) CLB with 2× 4-LUTs, (2) type 3 EB position (Figure 3) as this gives best performance for the first experiment, (3) channel width 80; since the maximum I/O density of the EBs is 42 pins per slice width, we choose 80 to be the channel width to facilitate routing, (4) EB size of 13×14 CLBs.

5.1. EB Position Results

We first examine how the position of the EBs affects the overall performance of the device. As shown in Figure 3, we consider positioning the EBs both around the periphery of the device, as well as in the centre. Intuitively, positioning the EBs in the centre will lead to shorter wirelengths for wires that connect multiple EBs. However, positioning the EBs around the periphery may cause less congestion since the EBs will be more spread out.

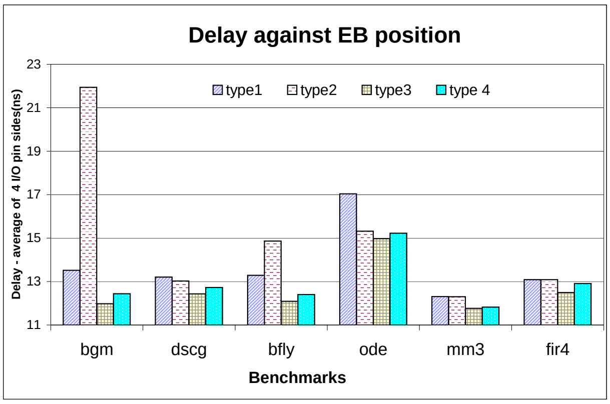


Fig. 6. Delay against various EBs positions, as defined in Figure 3

Figure 6 shows the results for each of the positioning strategies described in Figure 3. The best strategy is type 3, in which the EBs are in the centre of the device, surrounded by a sea of CLBs. The critical path of our circuits tend to include nets that connect multiple EBs; thus placing the EBs close to each other is beneficial.

5.2. Pin Location Results

We next consider the effect of I/O pin position on the periphery of each EB. As shown earlier, pins can be distributed evenly around the EB, or can be concentrated on one or more sides of the block. Intuitively, distributing the pins evenly will lead to a lower I/O density, possibly reducing congestion but may lead to longer wirelengths if pins from more than one side of the EB are connected.

The results are shown in Table 2 and Table 3. The critical path of the circuit is slightly smaller if all pins are placed on a single side of the embedded block. In several of our benchmarks, the critical path includes a path from one EB, through a register in a CLB, into another EB. These connections are shorter if the pins are close together. On the other hand, Table 3 shows that the routing demand in each channel can be reduced by distributing the pins evenly around each EB. Compared to the configuration in which all pins are on one side of the block, evenly distributing the pins reduces the channel width by 44%. We conclude that this is the best choice.

I/O pos.	1 side	2 sides	3 sides	4 sides
Density (per clb)	42	21	14	11
Circuits	delay in ns (Deviation from 1 side)			
bgm	11.94(0%)	11.99(0.3%)	12.01(0.6%)	11.99(0.4%)
dscg	12.42(0%)	12.51(0.7%)	12.26(-1.3%)	12.53(0.9%)
bfly	12.09(0%)	12.13(0.3%)	12.06(-0.3%)	12.09(0%)
ode	14.83(0%)	14.93(0.7%)	15.04(1.4%)	15.11(1.9%)
mm3	11.70(0%)	11.83(1.1%)	11.67(-0.3%)	11.86(1.4%)
fir4	12.32(0%)	12.42(0.8%)	12.53(1.6%)	12.71(3.1%)

Table 2. Critical path delay for different EB's I/O positions

I/O pos.	1 side	2 sides	3 sides	4 sides
Circuits	Min. channel width (Deviation from 1 side)			
bgm	46(0%)	35(-22%)	32(-29%)	25(-44%)
dscg	43(0%)	33 (-23%)	32 (-26%)	32 (-26%)
bfly	44 (0%)	35 (-20%)	33 (-25%)	32 (-27%)
ode	46 (0%)	38 (-17%)	37 (-20%)	38 (-17%)
mm3	42 (0%)	40 (-5%)	26 (-38%)	24 (-43%)
fir4	43 (0%)	33 (-23%)	32 (-26%)	30 (-30%)

Table 3. Minimum channel width for different I/O configurations

5.3. Interconnect Flexibility

We next consider the width of the channels surrounding the EBs. Intuitively, there is a high pin density on each side of each EB; this may place additional demands on the routing fabric near the EBs. If the fabric cannot provide the required flexibility, circuitous routes may be required, leading to increased delay.

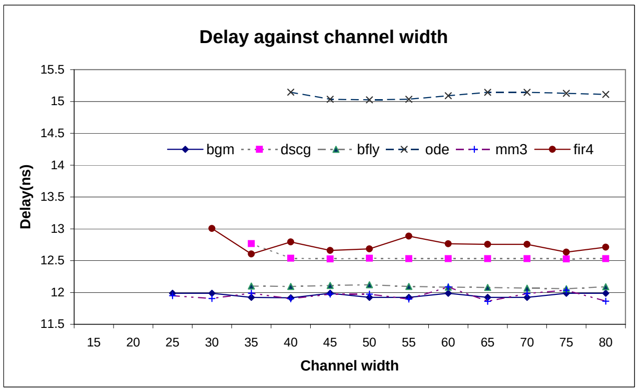


Fig. 7. Delay against channel width

The results in Figure 7 show the effect of EB to CLB channel width on delay. For routable circuits, rather surprisingly, the variation is less than 3%. We believe this is due to critical paths being routed efficiently so once the circuit is routable, channel width does not affect delay.

5.4. EB Aspect Ratio

Finally, we consider how the aspect ratio of each EB affects the overall performance of the FPGA. In this experiment, the area of EB is fixed, but the aspect ratio is changed. Intuitively, changing aspect ratio will change the distance between pins on different EBs; this leads to change in the delay of the nets connecting these pins.

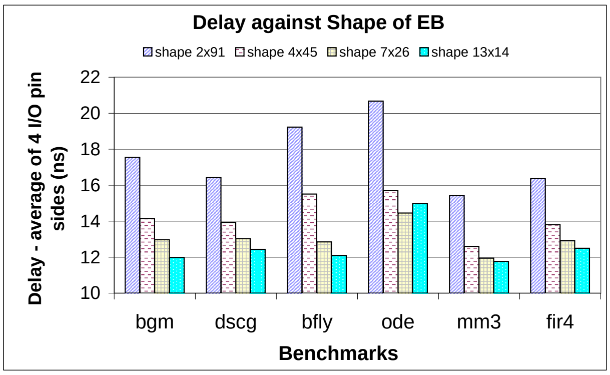


Fig. 8. Delay against various EBs' shape

We modify the shape of the EBs from rectangular (2x91) to square (13x14); the width and height are counted in the number of CLBs. The results in Figure 8 show that square EBs are the most efficient for all applications and result in a 23% speed improvement compared to the 2x91 shape. Square EBs lead to a better worst-case delay between the EBs, shortening the critical path in our benchmark circuits.

6. CONCLUSION

This paper investigates the architecture of the programmable interconnect between coarse-grained blocks and the fine-grained fabric in domain-specific FPGA with embedded floating point blocks. Specifically, we examine the position of the embedded blocks (EBs) within the FPGA, the placement of the pins on the periphery of the EB, the width of the routing channels surrounding the EB, and the aspect ratio of the EB. We find that (a) the EBs should be positioned close to each other in the middle of the chip, (b) the EB's pins should be distributed evenly around the EB, (c) the width of the channels surrounding the EB have little impact on circuit speed, and (d) a square EB leads to the most efficient implementations. Although our results are specific to the architecture studied, we believe they can be applied to FPGAs containing other types of embedded blocks. Current and future work includes extending our methodology to cover other embedded blocks, and generalising our model to support multiple types of embedded blocks.

Acknowledgement. The support of UK Engineering and Physical Sciences Research Council (EP/D060567/1 and EP/C549481/1), Celoxica and Xilinx is gratefully acknowledged.

References

[1] C.H. Ho, C.W. Yu, P.H.W. Leong, W. Luk and S.J.E. Wilton, "Domain-Specific Hybrid FPGA: Architecture and Floating Point Applications," in *Proc. FPL*, 2007, pp. 196 – 201.

[2] V. Betz, J. Rose, and A. Marquardt, "Architecture and CAD for Deep-Submicron FPGAs," *Kluwer Academic Publishers*, 1999.

[3] Altera Corp., "Stratix III Device Handbook, Vol.1," 2006.

[4] Xilinx Inc., "Virtex-5 Family Overview - LX, LXT, and SXT Platforms," 2007.

[5] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *IEEE Trans. CAD*, vol. 26, no. 2, 2007, pp. 203–215.

[6] A. Ye, J. Rose, and D. Lewis, "Architecture of Datapath-Oriented Coarse-Grain Logic and Routing for FPGAs," in *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*, 2003, pp. 61–64.

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

[8] S. Shukla, N.W. Bergmann and J. Becker, "QUKU: A Coarse Grained Paradigm for FPGA," *Pro. Dagstuhl Seminar 06141, 2006*.

[9] R. Enzler, C. Plessl and M. Platzner, "System-level performance evaluation of reconfigurable processors," *Microprocessors and Microsystems*, vol. 29, no. 2-3, pp. 63–73, 2004.

[10] P. Jamieson and J. Rose, "Enhancing the Area-Efficiency of FPGAs with Hard Circuits Using Shadow Clusters," in *Proc. ICFPT*, 2006, pp. 1–8.

[11] S.J.E. Wilton, J. Rose, and Z.G. Vranesic, "The Memory/Logic Interface in FPGAs with Large Embedded Memory Arrays," *IEEE Trans. on Very-Large Scale Integration System*, vol. 7, no. 1, 1999.

[12] T. Wong and S. Wilton, "Placement and routing for non-rectangular embedded programmable logic cores in SoC design," in *Proc. ICFPT*, 2004, pp. 65 –72.

[13] C.H. Ho, P.H.W. 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.

[14] C.W. Yu, "A Tool for Exploring Hybrid FPGAs," in *Proc. FPL PhD forum*, 2007, pp. 509 – 510.