Design and Implementation of an Efficient Programmable Floating Point Unit with Coarse-Grained FPGA

Arun. A [1], Dr. K. S. Srinivasan [2], Dr. M. Devaraju [3]

1. Assistant Professor, Dept of ECE, Velammal Engineering College, Chennai
aruneswaran26@gmail.com
2. Turbo Machinery Institute of Technology and Science, Hyderabad
sshari_2003@yahoo.com
3. Professor, Dept.of ECE, RMK Engineering College, R.S.M.Nagar, Chennai
devarajum@yahoo.com

Abstract: The novel method is to optimize coarse-grained floating point units (FPUs) in a hybrid FPGA by employing common sub graph extraction to determine the number of floating point adders / subtracters (FAs), multipliers (FMs) and word blocks (WBs) in the FPUs. Single precision FP adders / subtracters (FAs) and FP multipliers (FMs), with normalization are generated using standard cell library design flow. This empirical method is used to examine the speed and area of different coarse-grained FPUs. The common sub graph extraction method is for floating point applications and tools, benchmarks and models that are used in hybrid FPGA. To explore the design of a hybrid FPGA based on common sub graph extraction and synthesis, a set of floating point designs are used as benchmark circuits. They are: (1) DSCG, a data path of digital sine-cosine generator (2) BFLY, the basic computation of Fast Fourier Transform (3) FIR 4, a 4-tap finite impulse response filter (4) ODE, a circuit to solve ordinary differential equations (5) MM 3, a 3x3 matrix multiplier (6) BGM, a circuit to compute Monte Carlo simulations of interest rate model derivatives, (7) Syn2, a circuit contains 5 FAs and 4 FMs (8) Syn7, a circuit contains 25 FAs and 25 FMs. syn2 and syn7 are two synthetic benchmark circuits generated by a synthetic benchmark circuit generator. These 8 single precision floating point benchmark circuits are not efficiently implemented in fine-grained FPGAs, since the floating point computation requires a great deal of fine-grained resources. We synthesize different combinations of floating point adders / subtracters, multipliers and registers into coarse grained blocks, which are embedded in a hybrid FPGA. Later the benchmark circuits with these coarse-grained embedded blocks (EBs) are evaluated by the Altera Quartus II Chip planner tool for area and timing analysis.
[Arun. A, K. S. Srinivasan, M. Devaraju. **Design and Implementation of an Efficient Programmable Floating Point Unit with Coarse-Grained FPGA.** *Life Sci J* 2013;10(3):1959-1966] (ISSN: 1097-8135). http://www.lifesciencesite.com. 290

Keywords: floating point units (FPU), fine-grained architecture, coarse-grained architecture

1. Introduction

FPGA technology has been widely adopted to speed up computationally intensive applications. In modern Field Programmable Gate Arrays (FPGAs), coarse-grained elements such as processors, memories and DSPs are embedded into the fine-grained programmable fabric; 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). And the FPU resources will be wasted for those applications, which do not have any floating point computations. 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.

A hybrid FPGA consists of a combination of coarse-grained and fine-grained reconfigurable elements. It provides a high-throughput and cost effective platform for designers to develop applications. Coarse-grained units are more efficient than fine-grained programmable logic for implementing specific word-level operations. And these coarse grained units are less flexible and only specific applications that can make use of them. Given this limitation, optimization of coarse-grained elements becomes a critical issue. Modern commercial FPGAs consist of commonly used coarse-grained elements such as DSPs and memories. The computational speed of domain-specific applications can be further increased by additional embedded elements. For example, the high performance floating point computation can achieve better speed and density by incorporating embedded

floating point units (FPUs) [2]. Floating point adders/subtracters (FAs) and floating point multipliers (FMs) contain several basic functional elements such as barrel shifter, adders and multipliers. Word-blocks (WBs) are used for the bitwise operation of the floating point number such as comparison, shifting, latch and logical operation. Constructing hard circuit, which are composed of basic functional elements, results will be a more compact block with higher speed but less flexibility. Grouping together optimized WBs, FAs and FMs to be a floating point unit (FPU) can further improve the speed and area, since the interconnects between them use bus based connection .

Specifically this paper offers:

- A generic hybrid FPGA architecture that supports configurable resources with multiple granularity that will be customised for different applications.
- A domain-specific hybrid FPGA for various floating point computations will be designed from this architecture.
- 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.

2. Background

2.1. Related work

In the past decade, there has been much research on optimizing conventional island-style fine-grained FPGAs. The fine-grained elements consist of one or more *k*-input lookup tables (*k*-LUTs) and fast local interconnects. The studies include different aspects of segmented routing architectures for interconnect between fine-grained resources [3], and the effect of LUT and cluster size of the fine-grained elements [4]. The goal of these studies is to create a fast and area-efficient general purpose FPGA architecture.

Today, adding coarse-grained blocks within fine-grained fabric to improve area and speed is a common technique, since coarse-grained blocks implement specific functions more efficiently than fine-grained fabric. However, this hybrid FPGA architecture is on average approximately 20 times larger and 4 times slower than when implemented as ASIC [5]. In order to reduce this gap, considerable research has been focused on the architecture of hybrid FPGA. Jamieson and Rose [6] propose *shadow clustering* to minimize the overall area penalty by sharing local routing resources of fine-grained elements with embedded blocks. Also a coarse-grained architecture with bus-based interconnect has been shown to reduce area for data path circuits [7].

In Hybrid FPGA common sub-circuits that occur frequently in a variety of benchmark circuits. This is known as *common sub-graph extraction*. Fused-arithmetic units generated by these common sub-circuits get up to 3.3 times in speed and 19.7 times in area for average improvement of particular silicon cores. This paper employs this technique to determine floating point common sub-graphs similar to the fixed point approach. Instead of the improvement of particular cores, we focus on the system level tradeoff in hybrid FPGAs. A novel domain-specific hybrid FPGA architecture which embedded FPUs within fine-grained fabric has been presented; this architecture has advantage of 18 times area reduction when compared to a purely fine-grained architecture for floating point applications. However, these studies only accounted for the particular FPU architecture; they do not evaluate the combination of WBs, FAs and FMs according to different applications. This paper examines this relationship and investigates the effect of FPUs on a selected set of applications by using common sub-graph extraction and the hybrid FPGA exploration tool Altera Quartus II Chip planner.

2.2. FPGA architectures

The heart of a FPGA is a reconfigurable fabric. The fabric consists of arrays of fine-grained or coarse-grained units. Normally a fine-grained unit usually implements a single function and has a single bit output. The most 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 to make a cluster in a hardwired manner. These results in little loss in flexibility but can reduce area and routing resources within the fabric [8].

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, by combining several functions such as those in an arithmetic logic unit (ALU) and outputs are often multi-bit. They can be parameterised in terms of features such as bus-width and functionality. 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 Stratix III device has embedded fixed-function multipliers, embedded DSP units with multipliers and 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

3.1 Introduction

Hybrid FPGA is nothing but, it is connected by routing tracks and combination of both coarse grained and fine grained components. Fine grained consists of combination of Configurable logic blocks (CLBs). Each Configurable logic block contains look up tables (LUTs) and flip-flops (FFs) which supports fast carry chains, internal multiplexers and XOR gates. The coarse-grained embedded blocks (EBs), where as embedded memories and multipliers, are surrounded by CLBs and Single precision floating point adders (FA), floating point subtracters (FS) and FP multipliers (FMs), and a word-block (WB) is a LUT and flip-flop based unit. It carries out shifting, comparison, latch and logical operations.

In existing systems, the coarse-grained and fined-grained elements in a hybrid FPGA for floating-point applications are FPUs are in square, FPUs should be positioned tightly near the centre of the FPGA, the FPU pins should be arranged on four sides of the FPU. In fine-grained elements not much to support WBs, FAs, or FMs. In our system, we use a large amount of fine-grained elements to support WBs, FAs or FMs. if we use more WBs, FAs and FMs are inside a FPU greater area and speed improvement can be achieved, but the whole FPU is wasted if not used. First, we consider the performance of individual FPUs by connecting the elements using connection patterns. It is based on the different FPU architectures from common sub-graphs, so the performance of the hybrid FPGAs is defined by the density of FPU and flexibility of FPU. The common sub-graph extraction can potentially be implemented as a hard Embedded blocks (EBs) to speed up the computation and it represents the functionality shared across the benchmark circuits. The flattened net-list is then fed into the program maximum common sub-graph (MCS) generation stage to extract the common sub-graphs in these benchmark circuits. We enhance this method for floating point application which supports FA, FM and WB extraction. The common sub-graphs cover FP operations with the connection information of WBs, FAs and FMs, the coarse-grained FPU of common sub circuit in HDL file. Then synthesized by Altera Quartus II to obtain the area and delay of the FPU and use this information to evaluate the FPGA by place and route.

By place and route, this novel hybrid FPGA architecture will explore. The benchmark circuits described in a hardware description language (HDL) are synthesized to a map net-list in HDL format various units such as LUTs and registers are included in the library net-list. And place and route packs the clusters into fine-grained elements called configurable logic blocks (CLBs). Area, timing and position of the EBs are specified in a user constraint file. The architecture file contains the information of the architectural parameters of the fine-grained elements, such as delay of LUT and register. The placement and routing also timing analysis by using the packed benchmark net-lists, finally we estimate the area and delay for each benchmark circuit.

3.2 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 and 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 data path portions. The data path typically contains floating point operators such as adders, subtracters and multipliers, and occasionally square root and division operations. The data path often occupies most of the area in an implementation of the application. Existing FPGA devices are not optimized for floating point computations; floating point operators consume a significant amount of FPGA resources.

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 data path can often be pipelined and routing within the data path may be uni-directional in nature. Occasionally there is feedback in the data path for some operations such as accumulation. The control circuit is much simpler than the data path 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.

Table 1: Coarse-grained unit parameters

Symbol	Parameter description
D	Number of blocks (Including FPUs, word blocks)
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

From the above analysis, some basic requirements for floating point hybrid FPGA architectures will be done.

- 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

3.3 Architecture

Figure 1 shows a top-level block diagram of our hybrid FPGA architecture. It performs 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. Four input LUT-based fine-grained units, similar to Altera Stratix III slices are hence employed. However, the proposed FPGA hybrid modelling allows us to adopt other architectures such as the 6 input LUTs in Altera Stratix III. We believe the same trends would be seen as we migrate to smaller technologies and more modern FPGA architectures.

The data path 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 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.

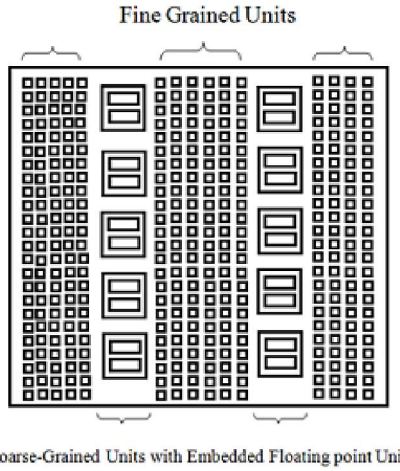

Figure 1: Floating point hybrid FPGA architecture.

A word-block contains *N* identical bit-blocks, and is similar to published designs [9]. A bit-block contains two 4-input LUTs and a reconfigurable output register. The value of *N* depends on the size of the FPU. Bit-blocks within a word-block are all controlled by the same set of configuration bits, so all bit-blocks within a word-block perform the same function. A word-block can efficiently implement operations such as addition and multiplexing. Similar to FPUs, word-blocks 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 the 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 a FPU can exploit the bus-oriented routing resources in the coarse-grained blocks.

The floating point multiplier block is a fixed-function block and 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.

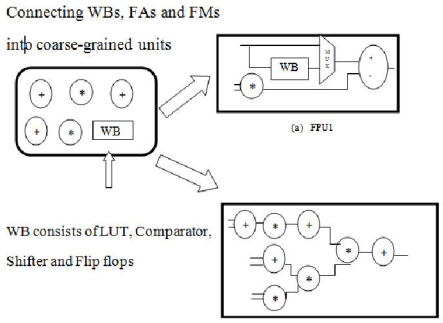

Figure 2: Connecting WBs, FAs and FMs into different coarse-grained FPUs.

3.4. Interface of Coarse-grained Blocks in Hybrid FPGA

The coarse-grained blocks are able to connect to fine-grained resources in various ways. The best interface between coarse-grained and fine-grained elements in a hybrid FPGA [10] for floating-point applications are: (1) FPUs are square (2) FPUs should be positioned tightly near the centre of the FPGA (3) The FPU pins should be arranged on four sides of the FPU. The interface between coarse-grained blocks and fine-grained blocks in this paper is assumed to follow the above configuration.

4. Optimization Parameters

In this paper, we optimize the internal connection structure and the number of the WBs, FAs and FMs. If more WBs, FAs and FMs are inside a FPU (Fig 2), greater area and speed improvement can be achieved, but the whole FPU is wasted if not used of WBs, FAs and FMs is important. We consider the FPU in the following parameters:

(a) Internal optimization of FPU:

The WBs, FAs and FMs in FPUs can be connected in different orders as shown in Figure 2. We consider the performance of individual FPUs by connecting such elements using commonly found connection patterns.

(b) System level optimization:

Based on the different FPU architectures from common subgraph, we optimize the performance of the hybrid FPGAs by selecting the FPUs in the following ways:

Density of FPU: The FPU consists of more computation an element achieves greater reduction since all elements can be closely packed. However, this may require more routing resources for the connection between the coarse-grained block and fine-grained block. And the flexibility decreases, since it is difficult to reuse in another application.

Flexibility of FPU: FPUs are wasted when not used. The FPUs can be reused across different applications. Therefore, embedding high flexibility can reduce the area waste for unused FPUs.

5. Modelling of Hybrid FPGA

5.1 Introduction

A method [9] is used to model floating point hybrid FPGAs with different architectural parameters and coarse-grained blocks. This approach is general and can be used to model any FPGA provided that a floor planner and a timing analysing tool are available for that device. In this method, 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 modeled 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.

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 Altera Quartus II 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 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. During the first step, we create a HDL description of the control logic part of the application circuit. We then add additional statements which instantiate the coarse-grained units 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 Stratix III required implementing 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, 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 netlist 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. After generating the netlist for the targeted FPGA, a User Constraint File (UCF) which forces the netlist to be located in a particular column is created. The final area and timing results will be obtained from the Altera Quartus II Place and Route tool. 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. After generating the netlist netlist for the targeted FPGA, a User Constraint File (UCF) which forces the netlist 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 and Altera Quartus II Synthesis, Place and Route tools can be used in the modelling of the hybrid FPGA and it can produce a realistic comparison to existing FPGA devices.

5.2. Tool Flow

We adopt common subgraph extraction to detect the most frequently used arithmetic units in floating point benchmark circuits, such as floating point adders/subtracters, multipliers and registers. Then we synthesize different combinations of these units into coarse-grained blocks, which are embedded in a hybrid FPGA. After that, the benchmark circuits with these coarse-grained embedded blocks (EBs) are evaluated by the Altera Quartus II tool for area and timing analysis.

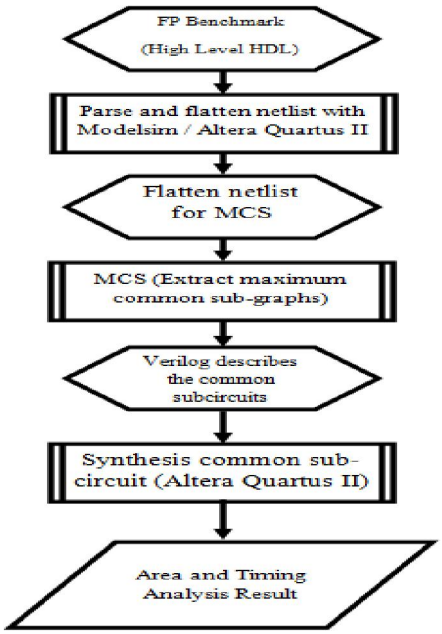

Figure 3: Common subgraph extraction design flow

(a) Common Subgraph Extraction:

Floating point applications have common characteristics for floating point computations. A common subgraph in these applications represents functionality shared across the benchmark circuits. The subgraph can potentially be implemented as a hard EB to speed up the computation. Efficiency can usually be improved by combining similar FP operations into the same core, by common subgraph extraction [11].

In the tool flow of common subgraph extraction as shown in Figure 3, floating point benchmark circuits are written in Verilog. Modelsim and Altera Quartus II are used to parse and flatten the Verilog benchmark circuits. The flattened netlist is then fed into the program Maximum Common Subgraph (MCS) generation stage to extract the

common subgraph in these benchmark circuits. With the connection information of WBs, FAs and FMs, we describe the coarse-grained FPU of common sub-circuit in another Verilog file. The FPU, which consists of complex FA and FM circuits, is then synthesized by Altera Quartus II with 110 nm process. We obtain the area and delay of this FPU and use this information to evaluate the FPGA by Altera Quartus II Place and Route. After we have determined the FP coarse-grained blocks by common subgraph extraction, such blocks are interfaced to the fine-grained FPGA. We use the Altera Quartus II tool to explore this novel hybrid FPGA architecture.

In the Altera Quartus II design flow in Figure 4, benchmark circuits described in a hardware description language (HDL) are synthesized to a mapped library netlist in VHDL format using Altera Quartus II. Various units such as LUTs and registers are included in the library netlist. Altera Quartus II netlist pack packs and clusters these simple units into fine-grained elements called configurable logic blocks (CLBs). Area, timing and position of the EBs are specified in a user constraint file.

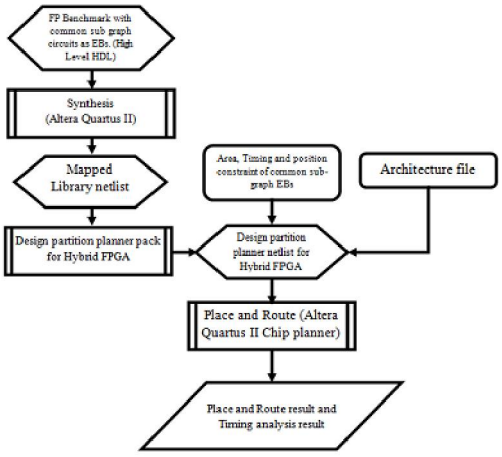

Figure 4: Design flow for common subgraph EBs using Altera Quartus II

The architecture file contains the information of the architectural parameters of the fine-grained elements, such as delay of LUT and register. The Altera Quartus II tool performs placement, routing and timing analysis using the packed benchmark netlist, constraint file and architecture file. The tool finally estimates the area and delay for each benchmark circuit.

6. Result

A set of benchmark applications are mapped to the proposed floating point hybrid FPGA, and the results are compared to a Stratix III device. All FPGA results are obtained using the Altera Quartus II for synthesis and place and route. Six benchmark circuits are used in this study [6]. Five of them are computational kernels and one is a Monte Carlo simulation data path. We have chosen these with simple but are not very efficiently implemented on general-purpose FPGA devices and 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 data path 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. The physical die area of a Stratix III device has been reported [13], 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 Stratix III device is 8,192 μm^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 1,051,011 μm^2. We further assume 15% overhead after place and route based on our experience [9]. The area values are normalised against the feature size (0.12μm). The number of equivalent slices is obtained through the division of coarse-grained unit area by slice area. 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 Stratix III slice employs smaller transistors (0.11μm) than those used for building the coarse-grained unit (0.12μm), we do not scale the timing of the coarse-grained unit and therefore conservative timing results are reported. We use EP2S15 as the host FPGA for the floating point hybrid FPGA. We assume that 8 double precision coarse-grained blocks are embedded into this FPGA. The coarse-grained blocks constitute 15% of the total area in an EP2S15 device. Benchmark circuits are implemented on the same device and the results are shown in Table 3. The FPU values for the EP2S15 device are estimated from the distribution of LUTs, which is reported by the Altera Quartus II synthesis tool. The logic area 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 80% of the user circuits. For example, the circuit *bfly* has 6 FPUs which consume 78% of the total FPGA area. It can fit into 2 coarse-

grained units, which constitute just 2.5% of the total FPGA area. The *bgm* circuit cannot fit in an EP2S15 device but it can be tightly packed into 7 coarse-grained units. Thus the circuit can fit in the hybrid FPGA in which the size is same as the EP2S15 device. Delay is reduced by 3.0 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 21 times. The saving is achieved by (1) embedded floating point operators, (2) efficient directional routing and (3) sharing configuration bits.

Table 2: Normalized area of the coarse-grained units against a Stratix III slice

Fabric	Area(A) (μm^2)	Feature size (L)(μm)	Normalised Area (A/m^2)	Area in slices	Input pin	Output pin
Stratix III slice	8.192	0.12	358,123	1	6(6)	2(2)
DP-CGU	1,051,01	0.11	78,506,546	158	215(1218)	186(312)

Table 3: Double precision floating point hybrid FPGA results

	Double precision floating point Hybrid FPGA					EP2S15				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	278(2.5%)	183(1.49%)	436(3.9%)	9.02	11,183(88%)	880(5%)	11377(91%)	21.57	21.4	2.18
dscg	2	278(2.5%)	289(2.16%)	566(4.6%)	11.11	8,887(62%)	289(2%)	9146(62%)	18.18	12.5	1.98
fir4	2	278(2.5%)	12(0.13%)	250(2.4%)	9.56	10,014(75%)	118(1%)	10299(72%)	20.68	28.8	2.16
mm3	2	278(2.5%)	290(2.02%)	524(4.59%)	7.9	7851(53%)	708(6%)	8089(58%)	20.18	13.9	2.16
cde	2	278(2.5%)	193(1.35%)	435(3.8%)	8.94	7339(51%)	215(2%)	7828(33%)	18.88	14.8	1.98
bgm*	6	1052(2.5%)	575(4.05%)	1180(8.8%)	9.10	21887(190%)	395(4%)	24887(180%)	24.14	16.6	2.03
									Geometric mean	18	2.08

7. Conclusion

We present a hybrid FPGA architecture which involves a combination of reconfigurable fine-grained and coarse-grained units dedicated to floating point computations. We show that the proposed floating point hybrid FPGA would be an 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 customisation for a large number of domain-specific applications.

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] 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.

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

[4] 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.

[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] P. Jamieson and J. Rose, .Enhancing the Area-Efficiency of FPGAs with Hard Circuits Using Shadow Clusters, in *Proc. ICFPT*, 2006, pp. 1.8.

[7] A. Ye, J. Rose, and D. Lewis, .Architecture of Data path-Oriented Coarse-Grain Logic and Routing for FPGAs, in *Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)*, 2003, pp. 61.64.

[8] 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.

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

[10] C. W. Yu, Julien Lamoureux, S. J. E. Wilton, P. H. W. Leong, and Wayne Luk, .The Coarse-Grained/Fine-Grained Logic Interface with Embedded Floating-Point Arithmetic Units,. in *Proc. SPL*, 2008, pp. 63.68.

[11] Alastair M. Smith, George A. Constantinides and Peter Y.K. Cheung, .Fused-Arithmetic Unit Generation for Reconfigurable Devices using Common Subgraph Extraction,. In *Proc. ICFPT*, 2007, pp. 105.112.

[12] 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.

[13] Floating-Point-Compiler-Increasing-Performance-With-Fewer-Resources.pdf http://www.altera.com.

5/20/2013