OPTIMIZING COARSE-GRAINED UNITS IN FLOATING POINT HYBRID FPGA

Chi Wai Yu [1], Alastair M. Smith[2], Wayne Luk[1], Philip H.W. Leong[3], Steven J.E. Wilton[2]

[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
{alastair, 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 introduces a novel methodology to optimize coarse-grained floating point units (FPUs) in a hybrid FPGA. We employ common subgraph extraction to determine the number of floating point adders/subtracters (FAs), multipliers (FMs) and wordblocks (WBs) in the FPUs. We first study the area, speed and utilization trade-off of the selected FPU subgraphs in a set of floating point benchmark circuits. We then explore the impact of density and flexibility of FPUs on the system in terms of area, speed and routing resources. We derive an optimized coarse-grained FPU by considering both architectural and system level issues. The results show that: (1) embedding more types of coarse-grained FPU in the system causes at most 21.3% increase in delay, (2) the area of the system can be reduced by 27.4% by embedding high density subgraphs, (3) the high density subgraphs requires 14.8% fewer routing resources.

1. Introduction

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 functional elements provide significant improvements in speed, logic density and power consumption for word-level computations. 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. However they are less flexible, and only benefit 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, an application which demands high performance floating point computation can achieve better speed and density by incorporating embedded floating point units (FPUs) [1].

Floating point adders/subtracters (FAs) and floating point multipliers (FMs) contain several basic functional elements such as barrel shifters, adders and multipliers. Wordblocks (WBs) are used for the bitwise operation of the floating point number such as comparison, shifting, latch and logical operation. Constructing hard circuit WBs, FAs and FMs, which are composed of basic functional elements, results in a more compact block with higher speed, but less flexibility. Optimizing these hard circuits is essential. 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 [1].

This paper assesses the impact of FPUs on hybrid FPGAs in terms of area, speed, routing resources and flexibility. The number and connection of WBs, FAs and FMs are determined by considering common subcircuits in a selected set of floating point benchmark circuits. The common subgraph extraction technique [2] has been employed to find out the fixed point arithmetic units which are common in a set of benchmark circuits. We adapt this method to study the combination of wordblocks, floating point addition and multiplication.

Specifically, this paper offers:

- a novel methodology to optimize the floating point hybrid FPGA by considering both internal architecture of FPUs and system level performance according to the mixture of FPUs.
- a study of internal architecture of FPUs. Common arithmetic subcircuits for floating point benchmark circuits are examined, and use these subcircuits to form a hardcore FPU,
- a quantitative system level analysis of speed, area and routing resource trade-off of common FP hardcores. By considering the speed, the area and the routing re-

source of a hybrid FPGA with selection of different hardcores, optimized designs can be obtained.

This paper is organized as follows. Section 2 describes related work. Section 3 illustrates the types of FPUs and the assumptions of the coarse-grained circuits. Section 4 then presents the empirical methodology used to extract common subcircuits and the evaluation schemes. Section 5 presents our results and analysis, and Section 6 summarizes our conclusions.

2. Background

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. Examples include embedded arithmetic multipliers and processors [5,6]. However, this hybrid FPGA architecture is on average approximately 20 times larger and 4 times slower than when implemented as ASIC [7]. In order to reduce this gap, considerable research has been focused on the architecture of hybrid FPGA. Jamieson and Rose [8] 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 datapath circuits [9].

Our previous work [2] detects common subcircuits that occur frequently in a variety of benchmark circuits. This is known as *common subgraph extraction*. Fused-arithmetic units generated by these common subcircuits 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 subgraphs similar to the fixed point approach. Instead of the improvement of particular cores, we focus on the system level trade-off in hybrid FPGAs.

Coarse-grained logic can be tailored for a certain domain-specific applications to achieve greater area and delay reduction. An example is an application which requires a significant amount of floating point computation. Implementing floating point operations in fine-grained FPGA technology consumes a large amount of logic and routing resources. Therefore, a number of recent approaches to optimize floating point operations in FPGA have been proposed. Pipeline stages of floating point arithmetic in custom computing machine have been optimized within existing fine-grained resources [10]. However, the density and speed are still worse than that implemented in ASIC.

In [1], 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. In [11], the best interfacing between FPUs and fine-grained elements has been determined. 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 subgraph extraction and the hybrid FPGA exploration tool called VPH [12].

3. Approach

This section describes the architecture of the resources used in this work. First, we present the architecture of a hybrid FPGA and its fine-grained elements. Then we describe the different potential coarse-grained FPUs and the assumption of their configuration.

3.1. Hybrid FPGA and Fine-grained Assumption

A Hybrid FPGA consists of both coarse-grained and fine-grained components, which are connected by routing tracks. Our fine-grained fabric consists of a grid of identical configurable logic blocks (CLBs). Each CLB contains k-LUTs, flip flops (FFs), support for fast carry chains, internal multiplexers and XOR gates. The coarse-grained embedded blocks (EBs), such as embedded memories and multipliers, are surrounded by CLBs as shown in Figure 1. As an area model, we use the Virtex II CLB with area 10,912μm^2 and feature size 0.15μm for our fine-grained fabric. The FPGA is island-style, with horizontal and vertical channels connecting the CLBs [3]. The channel contains W parallel routing tracks of length 1 and is connected to neighbouring CLBs using a connection block.

3.2. Coarse-grained Block Assumption

Single precision FP adders/subtracters (FAs) and FP multipliers (FMs), with normalization are generated using standard cell library design flow. They are synthesized by Synopsys Design Complier with a 130nm process. A word-block (WB) is a LUT and flip flop based unit. It carries out bitwise shifting, comparison, latch and logical operations such as *and*, *or* and *xor*.

In our previous work [1], WBs, FAs and FMs are connected by bus based wires to construct a more compact

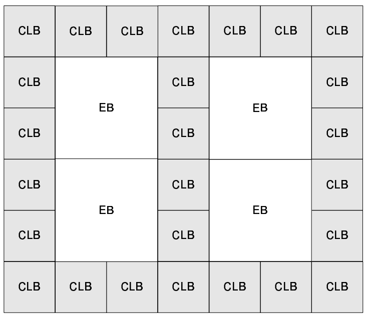


Figure 1. Hybrid FPGA: Embedded blocks (EBs) are surrounded by grid based CLBs

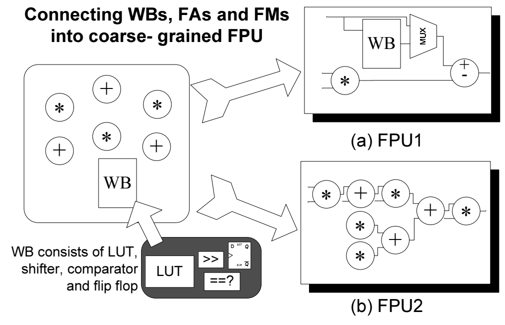

Connecting WBs, FAs and FMs into coarse- grained FPU
WB
WB consists of LUT, shifter, comparator and flip flop
LUT
>>
==?
MUX
(a) FPU1
(b) FPU2

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

coarse-grained FPU. This local bus based connection can save a large amount of the routing resources in an FPGA. Figure 2 shows an example of this arrangement.

3.3. Interface of Coarse-grained Blocks in Hybrid FPGA

The coarse-grained blocks are able to connect to fine-grained resources in various ways. Our previous work [11] shows that the best interface between coarse-grained and fined-grained elements in a hybrid FPGA 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.

3.4. 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 an FPU (Figure 2 (b)), greater area and speed improvement can be achieved, but the whole FPU is wasted if not used. In Figure 2 (a), FPU1 with fewer WBs, FAs and FMs would waste less resources if not used, but we need a large amount of fine-grained elements to support WBs, FAs or FMs. Therefore, choosing a suitable number of WBs, FAs and FMs is important. We consider the FPU in the following parameters:

1. 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.

2. System level optimization:

Based on the different FPU architectures from common subgraphs, 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 elements 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 also 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.

4. Methodology

We employ an empirical methodology to examine the speed and area of different coarse-grained FPUs. This section describes a common subgraph extraction methodology for floating point applications, and the tools, benchmarks and models that are used.

4.1. Floating Point Benchmark Circuits

To explore the design of a hybrid FPGA based on common subgraph extraction and synthesis, a set of floating point designs are used as benchmark circuits. They are: (1) *dscg*, a datapath of digital sine-cosine generator, (2) $bfly$, the basic computation of Fast Fourier Transform: $z = y + x * w$ where inputs and output are complex numbers, (3) $fir4$, 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) *syn*2, a circuit contains 5 FAs and 4 FMs, (8) *syn*7 a circuit contains 25 FAs and 25 FMs. *syn*2 and *syn*7 are two synthetic benchmark circuits generated by a synthetic benchmark circuit generator, based on [13]. 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 of fine-grained resources. The use of fine-grained CLB resources to implement these applications is summarised in Table 4.1. The tool VPH is used to do clustering, placement and routing of the circuits.

Table 1. The delay and the number of CLB used in each benchmark circuit

Circuits	Area (in CLB)	Delay (ns)
dscg	4,759	15.54
bfly	4,984	16.3
fir4	4,551	13.9
ode	4,100	10.68
mm3	3,328	14.54
bgm	16,300	21.29
syn2	5,555	12.81
syn7	30,245	30.4

4.2. 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 VPH evaluation tool for area and timing analysis.

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 [2]. We enhance this method for floating point application which supports FA, FM and WB extraction.

Figure 3 is an example of a common subgraph of two circuits. Figure 3(a) and (b) are part of *dscg* and *bfly* respectively. The common floating point operations can be extracted as a single unit as shown in Figure 3(c).

In the tool flow of common subgraph extraction as shown in Figure 4, floating point benchmark circuits are written in Verilog. Icarus Verilog [14] and ODIN [15] 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 subgraphs in these benchmark circuits.

The common subgraphs cover FP operations such as the example shown in Figure 3(c). With the connection information of WBs, FAs and FMs, we describe the coarse-grained FPU of common subcircuit in another Verilog file. The FPU, which consists of complex FA and FM circuits, is then synthesized by Synopsys Design Complier with 130nm process. We obtain the area and delay of the this FPU and use this information to evaluate the FPGA by VPH.

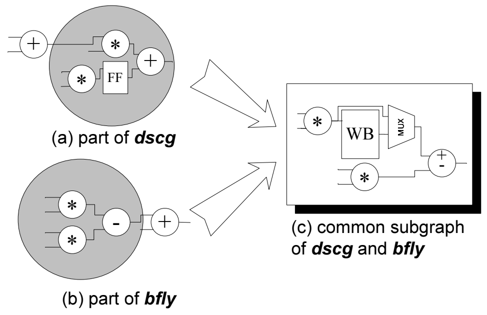
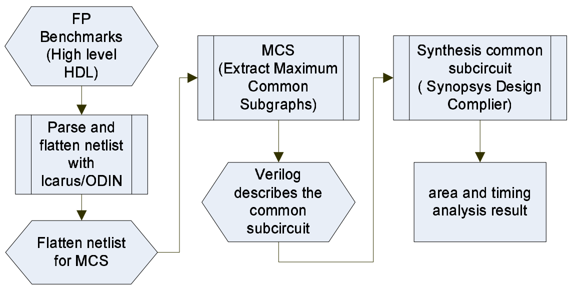


Figure 3. Common subgraph extraction for WB, FA and FM in FP applications



FP Benchmarks (High level HDL)
Parse and flatten netlist with Icarus/ODIN
Flatten netlist for MCS
MCS (Extract Maximum Common Subgraphs)
Verilog describes the common subcircuit
Synthesis common subcircuit (Synopsys Design Complier)
area and timing analysis result

Figure 4. Common subgraph extraction design flow

VPH: Versatile Place and Route for Hybrid FPGA

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 evaluation tool VPH [12] to explore this novel hybrid FPGA architecture. VPH is a modified version of the VPR tool, with support for embedded blocks (EBs), memories, multipliers, carry chains and user constraints. In the VPH design flow in Figure 5, benchmark circuits described in a hardware description language (HDL) are synthesized to a mapped library netlist in VHDL format using Synplicity's Synplify Premier. Various units such as LUTs and registers are included in the library netlist. VPHpack 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. The architecture file contains the information of the architectural parameters of the fine-grained elements, such as delay of LUT and register. The VPH tool performs placement, routing and timing analysis using the packed benchmark netlists, constraint file and architecture file. The tool finally estimates the area and delay for each benchmark circuit.

5. Results

In this section, the internal and system level optimization of coarse-grained floating point units are studied. The com-

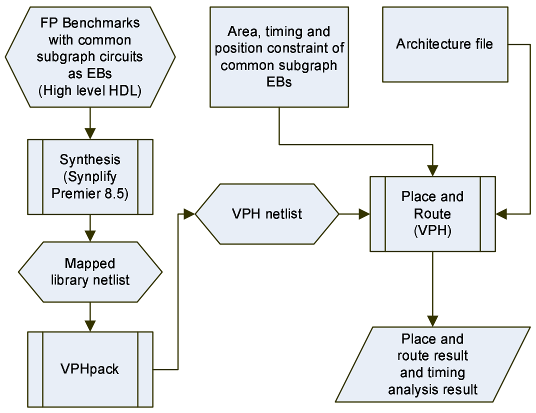

FP Benchmarks with common subgraph circuits as EBs (High level HDL)
Area, timing and position constraint of common subgraph EBs
Architecture file
Synthesis (Synplify Premier 8.5)
VPH netlist
Place and Route (VPH)
Mapped library netlist
VPHpack
Place and route result and timing analysis result

Figure 5. Design flow for common subgraph EBs using VPH

mon subgraphs of the floating point benchmark circuits are examined to optimize the architecture of the FPU. After that we explore the area and delay impact of making the common subgraphs coarse-grained units in the hybrid FPGA. The default architecture parameters of the experiments conducted are: (1) CLB with 2x 4-LUTs, (2) the channel width is 50 to facilitate routing.

5.1. Internal Optimization of FPU

We determine the common subgraphs of floating point arithmetic in the benchmark circuits. The hard FPU has specific ordering of WBs, FAs and FMs such that it can be reused by different benchmark circuits. The common subgraphs are shown in Table 5.1, which are found by common subgraph extraction described in Section 4. The hard FPU is obtained by 130nm process, however the CLB feature size modelled in VPH is 150nm. Therefore, the normalized area (Area/Feature size squared) is used to obtain the equivalent area in terms of CLBs. The performance of the various common subcircuits is examined.

Table 5.1 shows the occurrence in benchmarks, normalized area, delay, number of input/output and latency of each common subgraph found. It is obvious that more FAs and FMs embedded in an FPU can achieve higher area reduction since all the elements are compacted into a single unit. The average area of the FPUs is 9.4% less than that without clustering of WBs, FAs and FMs. But the delay after grouping is 14.9% more than pure FAs and FMs. The difference in the delay is because the timing report of single FA or FM does not account for the output to input delay when connecting them together. Therefore we should investigate the performance of the system rather than the single unit.

5.2. System Level Optimization

After we determine the common subcircuits, we evaluate the impact of these subcircuits to the systems. Based on the optimization parameter in Section 3.4. The delay, area and routing resources of purely FA/FM system, and mixture of subcircuits are examined. In the purely FA/FM system, 25 x **FA** and 25 x **FM** are used in the hybrid FPGA.

Density of FPU

We select a set of common subcircuits with more FAs, FMs and WBs. For example, *Graph 41* has 11 nodes that are a combination of FAs and FMs, contains most computation elements among subgraphs in Table 5.1. Since the density and area reduction of this individual subcircuit is greatest compared to separate FA, FM and WB, this set of subcircuits can be the best choice to reduce the area of the hybrid FPGA. The selection scheme is:

1. Selecting the common subcircuit with highest number of FAs, FMs and WBs,
2. Removing the FAs, FMs and WBs in the selected common subcircuits from the benchmark circuits,
3. Selecting the next most common subcircuit in the remaining subgraphs with highest number of computation units,
4. Repeating *2-3* until all subgraphs have been visited.

We choose 7 types of FPUs: **Graph 41**, **Graph 20**, **Graph 37**, **Graph 12**, **Graph 26**, **FM** and **FA** as subcircuits to embed in the hybrid FPGA (*FPGA_41_20_37_12_26*).

Flexibility of FPU

If we can reuse all the subcircuits for all applications, the area of the hybrid FPGA may be reduced. We select a set of subcircuits which have highest occurrence rate in the benchmark circuits from Table 5.1, and based on the following rules:

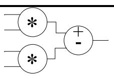
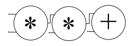
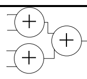
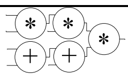
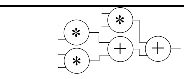
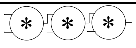
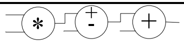
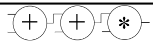
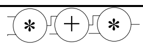
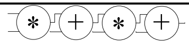
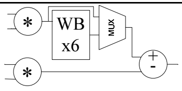
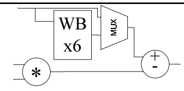
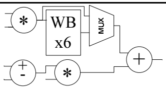
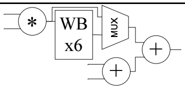
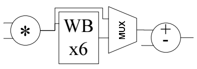
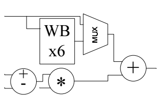
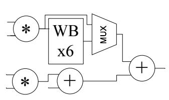
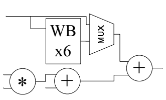
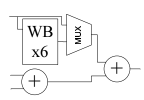
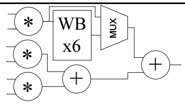
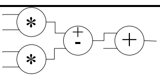
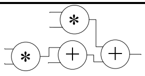
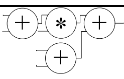
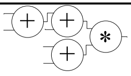
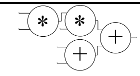
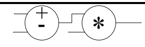
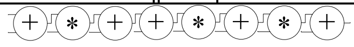
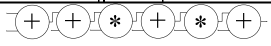
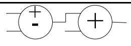
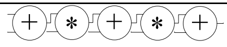
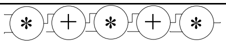
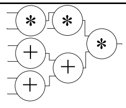
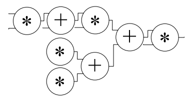
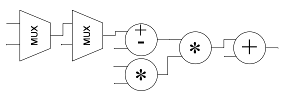
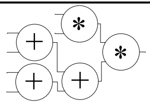
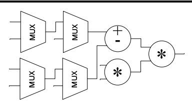
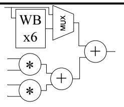
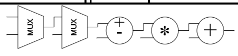
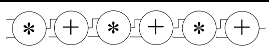
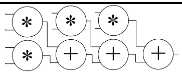
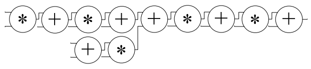
1. Selecting the common subcircuit which represents the highest number of computation elements in all benchmark circuits,
2. Removing the FAs, FMs and WBs in the selected common subcircuits from the benchmark circuits,
3. Selecting the next most common subcircuit in the remaining subgraphs which represents the highest number of computation elements in all benchmark circuits,
4. Repeating *2-3* until all subgraphs have been visited.

We finally choose 5 type of FPUs: **Graph 12**, **Graph 15**, **Graph 26**, **FM** and **FA** to embed in the hybrid FPGA (*FPGA_12_15_26*).

Discussion

From the three hybrid FPGAs we selected : (a) Purely FA/FM FPGA, (b) *FPGA_12_15_26* and
(c) *FPGA_41_20_37_12_26*, we examine the area, delay and routing resources impact to the applications. The utilization rate of the subcircuits in each hybrid FPGA for benchmark circuits is shown Table 5.2. An obvious result is that,

Table 2. The common subgraph structure occurred in benchmark circuits

no.	Subcircuits	no.	Subcircuits	no.	Subcircuits	no.	Subcircuits	no.	Subcircuits
1		2		3		4		5	
6		7		8		9		10	
11		12		13		14		15	
16		17		18		19		20	
21		22		23		24		25	
26		27				28			
29		30				31			
32		33				34			
35		36				37			
38				39					
40				41					

embedding FPUs can achieve at least 31.5 times area and 2.9 times delay reduction compared to purely fine-grained FPGA (Table 4.1).

Result 1, area impact: Figure 6 shows the number of CLBs used in each application with different types of graphs. *FPGA_12_15_26* reduces area by 18.8% and *FPGA_41_20_37_12_26* reduces area by 27.4% when compared to purely FA/FM hybrid FPGA's area in average. The purely FA/FM system does not have wordblocks and as such it requires a lot of fine-grained CLBs as registers and logical operators. *FPGA_41_20_37_12_26* embeds high density subcircuits, which are not flexible to the benchmarks, some of the FPUs are thus wasted, as they are not used. However, the individual subcircuits obtain highest area reduction to overcome the waste of area when not used. Therefore, it is the best to reduce area in hybrid FPGA.

Result 2, delay impact: Figure 7 shows purely FA/FM hybrid FPGA achieves highest speed. The delay of purely FA/FM FPGA is 16.7% and 21.3% less than *FPGA_12_15_26* and *FPGA_41_20_37_12_26* respectively. We discover that embedding more coarse-grained FPUs types causes a decrease in speed. The critical path is dominated by the connection between two FPUs. This path can only be optimized by moving the FPUs close together. Since the various FPUs have different architectures, they cannot be swapped to get better placement. For example, we cannot swap **Graph 41** and **Graph 20**, but two **Graph 12** can be swapped. The purely FA/FM system has least types of subcircuits. Therefore, due to the placement constraints of the different FPUs type, FPUs in a purely FA/FM system have more freedom to move around to achieve higher speed.

Result 3, routing resource impact: On average, the channel width of *FPGA_41_20_37_12_26* is 14.8% less than the purely FA/FM FPGA and 9% less than *FPGA_12_15_26* as shown in Figure 8. The FPUs in *FPGA_41_20_37_12_26* are most compact, most of the connection is in the FPUs, therefore it must use less routing resources for connection.

From the above result, different mixtures of coarse-grained subcircuits can achieve different aspects of optimization in hybrid FPGAs. As a result, we could use a suitable set of subcircuits to obtain a particular optimization goal.

6. Conclusion

This paper illustrates the adoption of common subgraph extraction to determine optimized floating point coarse-grained blocks in hybrid FPGAs. Floating point circuits

Table 3. Statistic of subgraphs. The feature size (L) of the coarse-grained units is 0.13μm. Virtex II CLB: area is 10,912μm^2, feature size is 0.15μm, normalized area is 485,013

Graph	Occurrence in benchmarks								area(A)	Normalized	Area in	delay	no.	no.	latency
no.	dscg	bfly	fir4	ode	mm3	bgm	syn2	syn7	(μm^2)	area(A/L^2)	CLB	(ns)	input	output	(cycles)
WB	No separate WB embedded								22,009	1,302,307	3	0.59	104	38	1
FA	4	4	3	3	2	9	5	25	59,151	3,500,059	7	2.77	67	39	6
FM	4	4	4	2	3	11	4	25	114,696	6,786,745	14	3.18	67	39	6
1	0	2	1	0	1	2	0	4	263,520	15,592,899	32	3.17	133	53	12
2	0	0	0	1	0	0	0	8	184,811	10,935,562	23	3.41	133	53	18
3	0	0	0	0	0	0	1	3	163,017	9,645,976	20	3.33	133	53	12
4	0	0	0	0	0	0	1	4	419,114	24,799,645	51	3.69	199	67	18
5	0	0	1	0	0	2	0	2	415,381	24,578,757	51	3.99	199	67	18
6	0	0	0	0	0	0	1	7	282,878	16,146,627	33	3.52	133	53	18
7	0	2	1	0	1	2	0	8	207,722	12,291,242	25	3.27	133	53	18
8	0	0	0	0	0	1	1	8	211,799	12,532,485	26	3.31	133	53	18
9	0	0	0	0	0	4	0	7	259,231	15,339,112	32	3.38	133	53	18
10	0	0	0	0	0	3	0	5	309,120	18,291,124	38	3.51	166	60	24
11	2	2	1	0	1	2	0	4	413,951	24,494,142	51	4.06	160	84	18
12	2	2	3	1	1	7	0	16	309,754	18,328,639	38	3.66	127	77	18
13	2	0	0	0	0	1	0	4	470,076	27,815,147	57	4.95	193	91	18
14	0	0	1	0	1	2	0	6	368,998	21,834,201	45	4.06	160	84	18
15	2	2	3	1	2	9	0	16	314,496	18,609,230	38	3.53	127	77	18
16	2	0	0	1	0	3	0	8	366,819	21,705,266	45	5.82	160	84	18
17	0	0	1	0	1	2	0	5	450,525	26,658,284	55	3.54	193	91	18
18	0	1	1	0	1	2	0	8	366,819	21,705,266	45	3.95	160	84	18
19	0	1	1	0	1	2	1	8	271,785	16,081,952	33	3.71	127	77	12
20	0	0	1	0	1	2	0	2	543,269	32,146,094	66	3.54	226	98	18
21	0	2	1	0	1	2	0	2	309,120	18,291,124	38	3.34	166	60	18
22	0	0	1	0	0	2	0	5	309,120	18,291,124	38	3.34	166	60	18
23	0	0	0	0	0	1	0	5	268,377	15,880,295	33	4.54	166	60	18
24	0	0	0	0	0	0	1	5	268,377	15,880,295	33	3.54	166	60	18
25	0	0	0	0	0	0	1	6	309,120	18,291,124	38	3.51	166	60	18
26	2	0	0	1	0	5	1	15	166,254	9,837,514	20	3.39	100	46	12
27	0	0	0	0	0	1	0	1	567,414	33,574,792	69	3.55	298	88	48
28	0	0	0	0	0	1	0	3	417,493	24,703,727	51	3.76	232	74	36
29	0	2	1	0	1	2	1	8	106,534	6,303,786	13	3.44	100	46	12
30	0	0	0	0	0	1	0	4	352,290	20,845,562	43	3.83	199	67	30
31	0	0	0	0	0	1	0	4	415,381	24,578,757	51	3.98	199	67	30
32	0	0	0	0	0	0	1	2	460,638	27,256,686	56	3.87	232	74	18
33	0	0	0	0	0	1	0	1	639,636	37,848,284	78	4.68	298	88	30
34	0	0	0	1	0	0	0	4	311,337	18,422,307	38	3.89	232	60	18
35	0	0	0	0	0	0	1	3	364,272	21,554,556	44	4.54	199	67	18
36	0	0	0	1	0	0	1	8	268,323	15,877,100	33	3.40	265	53	12
37	0	1	1	0	1	2	0	2	463,914	27,450,532	57	3.74	193	91	18
38	2	0	0	1	0	3	0	8	210,981	12,484,082	26	4.02	199	53	18
39	0	0	0	0	0	1	0	2	463,216	27,409,230	57	3.75	232	74	36
40	0	0	1	0	0	0	0	2	550,086	32,549,467	67	4.49	265	81	24
41	0	0	0	0	0	1	0	1	794,002	46,982,366	97	3.68	397	109	54

Table 4. The utilization rate of subcircuits in the three hybrid FPGAs

Hybrid FPGA	**1. Purely FA/FM FPGA**		**2. FPGA_12_15_26**					**3. FPGA_41_20_37_12_26**						
Graph no.	**FA**	**FM**	**12**	**15**	**26**	**FA**	**FM**	**41**	**20**	**37**	**12**	**26**	**FA**	**FM**
No. of subcircuits in FPGA	25	25	16	2	2	7	7	1	1	2	8	3	4	5
Benchmark	**Utilization rate (%)**													
dscg	16	16	12.5	0	0	28.57	28.57	0	0	0	25	0	50	40
bfly	16	16	12.5	0	0	28.57	28.57	0	0	50	12.5	0	0	40
fir4	12	16	18.75	0	0	0	14.29	0	100	0	12.5	0	0	0
ode	12	8	6.25	0	0	28.57	14.29	0	0	0	12.5	0	50	20
mm3	8	12	6.25	50	0	0	14.29	0	100	0	0	0	0	0
bgm	36	44	43.75	100	0	0	28.57	100	100	0	12.5	0	0	40
syn2	20	16	0	0	50	57.14	42.86	0	0	0	0	33.33	100	60
syn7	100	100	100	0	100	100	100	100	0	100	100	100	100	100

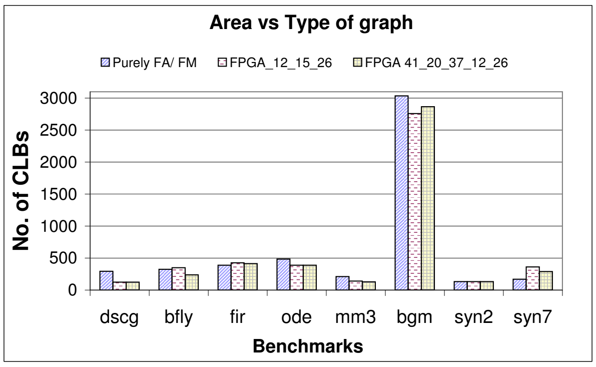


Figure 6. The number of CLBs used by different type of embedded FPUs

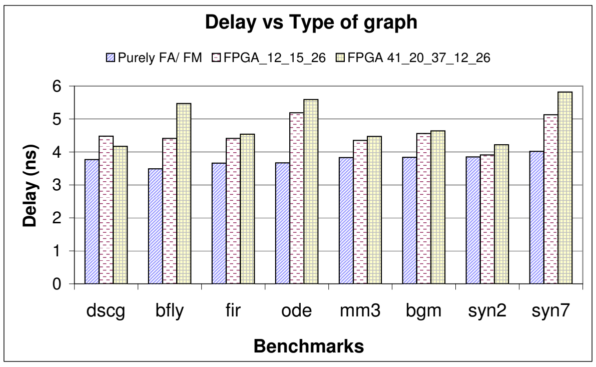


Figure 7. The delay of benchmark circuits by using different type of FPUs

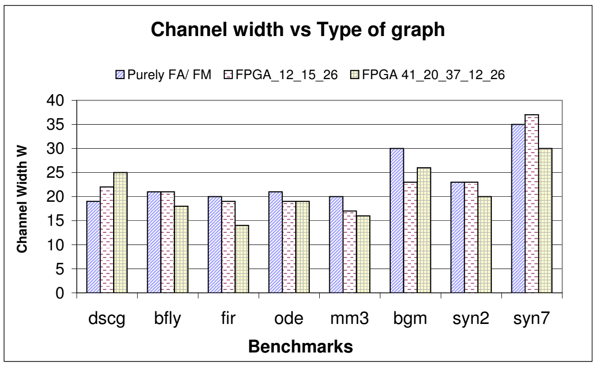


Figure 8. The routing resources of benchmark circuits by using different type of FPUs

are often not efficiently implemented in fine-grained FPGA technology. This paper has shown the impact of embedding different and multiple types of coarse-grained blocks on a floating point hybrid FPGA. We have found that, (1) the speed of the system is the highest with FAs and FMs only implementation, (2) higher density subgraphs produce greater reduction on the area of the system, and (3) they require less routing resources. We can optimize specific parameters such as area, delay and routing resources for the hybrid FPGA based on the above results. Current and future work includes generalising our model to support multiple types of embedded blocks for different application domains, and develop algorithm to map the subcircuits into different benchmark circuits.

Acknowledgement. The support of UK Engineering and Physical Sciences Research Council (EP/C549481/1, EP/D060567/1 and EP/D062322/1) and Canadian Commonwealth Postdoctoral Fellowship scheme (Canadian Bureau for International Education/CBIE) 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] 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.

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

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

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

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

[9] 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.

[10] G. Govindu, L. Zhuo, S. Choi, and V. Prasanna, "Analysis of High-performance Floating-point Arithmetic on FPGAs," in *Proc. Parallel and Distributed Processing Symposium*, 2004, pp. 149–156.

[11] 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.

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

[13] P. D. Kundarewich and J. Rose, "Synthetic circuit generation using clustering and iteration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 23, no. 6, pp. 869–887, 2004.

[14] ICARUS, Verilog http://www.icarus.com/eda/verilog.

[15] P. Jamieson and J. Rose, "A verilog RTL synthesis tool for heterogeneous FPGAs," in *Proc. FPL*, 2005, pp. 305–310.