# Optimizing Floating Point Units in Hybrid FPGAs

ChiWai Yu, Alastair M. Smith, Wayne Luk, *Fellow, IEEE*, Philip H. W. Leong, *Senior Member, IEEE*, and Steven J. E. Wilton, *Senior Member, IEEE*

***Abstract*—This paper introduces a methodology to optimize coarse-grained floating point units (FPUs) in a hybrid field-programmable gate array (FPGA), where the FPU consists of a number of interconnected floating point adders/subtracters (FAs), multipliers (FMs), and wordblocks (WBs). The wordblocks include registers and lookup tables (LUTs) which can implement fixed point operations efficiently. We employ common subgraph extraction to determine the best mix of blocks within an FPU and study the area, speed and utilization tradeoff over a set of floating point benchmark circuits. We then explore the system impact of FPU density and flexibility in terms of area, speed, and routing resources. Finally, we derive an optimized coarse-grained FPU by considering both architectural and system-level issues. This proposed methodology can be used to evaluate a variety of FPU architecture optimizations. The results for the selected FPU architecture optimization show that although high density FPUs are slower, they have the advantages of improved area, area-delay product, and throughput.**

***Index Terms*—Common subgraph extraction, field-programmable gate array (FPGA), floating point (FP).**

## I. Introduction

IN MODERN field-programmable gate arrays (FPGAs), coarse-grained elements such as memories and digital signal processors (DSPs) are embedded within a fine-grained programmable fabric. These fixed-functionality elements provide a high-throughput and cost-effective platform to develop applications [1].

Although coarse-grained units are more efficient than fine-grained units for implementing specific word-level operations, they are less flexible, and only benefit applications that can make use of them. Given this limitation, the optimization of coarse-grained elements becomes a critical issue. The computational speed of domain-specific applications can be further increased through additional embedded elements. For example, an application which demands high performance floating point (FP) computation can achieve better speed and density by incorporating embedded floating point units (FPUs) [2]–[4]. In the FP application domain, an FPGA can provide higher data throughput by using more and faster FPUs [5], [6]. In this work, an FPU is a number of fixed point wordblocks, floating point adders/subtracters (FAs), and floating point multipliers (FMs) [3]. Wordblocks (WBs) are used for general fixed point arithmetic and logical operations. Hard WBs, FAs and FMs are composed from non-programmable elements, resulting in a more compact block with higher speed, but less flexibility than fine-grained logic.

A method to optimize the architecture of a comprehensive set of FPUs is proposed. The number and interconnection of primitives are determined using common subgraph extraction to find efficient arithmetic units over a set of benchmark circuits [7]. Specifically, the contributions of this paper are as follows:

1) a methodology to optimize a floating point hybrid FPGA by considering both the internal architecture of FPUs and types of FPUs incorporated into the system;
2) a study of FPU architectures over a set of FP benchmark circuits;
3) a quantitative system-level analysis of resource trade-offs in FP hard cores;
4) an analysis of the benefits of merging different types of FPUs into a larger coarse-grained FPU.

A preliminary version of this work was presented in [8]. This paper revises the area and timing results by using the latest version of Synopsys Design Complier and further considers optimization by merging different types of FPUs into a larger coarse-grained FPU. This serves to reduce the number of distinct types of FPUs in the system, resulting in fewer constraints on the FPU placement and routing. We also discuss the data throughput of different systems in this paper.

## II. Background

An FPGA is an array of fine-grained configurable logic blocks interconnected in a hierarchical fashion. Commercial FPGAs contain coarse-grained blocks such as memories and multipliers for commonly used primitives [9] to improve efficient for specific functions. However, FPGAs have been shown to be approximately 20 times larger and 4 times slower than application-specific integrated circuits (ASICs) [1]. In order to reduce this gap, considerable research has been focused on identifying more flexible and efficient coarse-grained blocks, particularly for specialized application domains such as floating point computations. It is because implementing FP operations in fine-grained FPGA consumes a large amount of resources and a number of approaches to optimize FP operations in FPGAs have been proposed.

Manuscript received September 01, 2010; revised March 13, 2011; accepted April 14, 2011. Date of publication June 20, 2011; date of current version June 01, 2012. This work was supported in part by UK Engineering and Physical Sciences Research Council and by Canadian Commonwealth Postdoctoral Fellowship scheme.

C. Yu and W. Luk are with the Department of Computing, Imperial College London, London SW7 2BT, U.K. (e-mail: yuchiwai@gmail.com; wl@doc.ic.ac.uk).

A. M. Smith is with the Department of Electrical and Electronic Engineering, Imperial College London, South Kensington SW7 2BT, U.K. (e-mail: alastair.smith@imperial.ac.uk).

P. H. W. Leong is with the School of Electrical and Information Engineering, University of Sydney, Sydney, NSW 2006, Australia (e-mail: philip.leong@sydney.edu.au).

S. J. E. Wilton is with the School of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: stevew@ece.ubc.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2011.2153883

1063-8210/$26.00 © 2011 IEEE

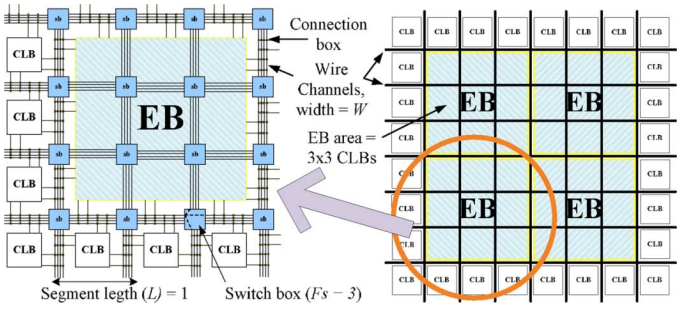



Fig. 1. Hybrid FPGA: EBs are surrounded by grid-based CLBs.

Our previous work [7] used common subgraph extraction to identify subcircuits which could be implemented as fused-arithmetic units to improve speed and area. Cevrero *et al.* [10] observed a common pattern for multi-input addition and introduced the field programmable compressor tree (FPCT) based on this pattern as an alternative to DSP blocks. The FPCT can perform DSP function in conjunction with a small amount of FPGA general logic, but was limited to fixed point. This paper employs a similar technique to determine FP common subgraphs and we focus on system-level tradeoffs.

Pipelined FP stages in custom computing machines have been studied and optimized using existing fine-grained resources [11]. However, the density and speed are poor compared with an ASIC. Langhammer's work on fused FP datapath synthesis for existing FPGAs reduces the logic resources and latency by sharing the normalization of different FP operators [5], [6], [12]. The FloPoCo compiler [13] generates a custom pipeline to optimize FP operators. Beauchamp *et al.* [2] introduced embedded variable length shifters in the FPGA fabric and added a 4:1 multiplexer to the cluster to reduce area and delay. A domain-specific hybrid FPGA architecture with embedded FPUs was presented in [3]. This architecture achieved an 18 times area reduction compared with a homogeneous FPGA. These studies only considered a particular FPU architecture, and did not optimize the combination of WBs, FAs, and FMs. This paper directly addresses this issue.

## III. Framework

### A. Hybrid FPGA

A hybrid FPGA consists of coarse-grained and fine-grained components, which are connected by routing tracks. Our fine-grained fabric consists of an array of identical configurable logic blocks (CLBs), each containing $N$ basic logic elements (BLEs). Each BLE contains $k$-LUTs, flip flops (FFs), support for fast carry chains, internal multiplexers and XOR gates. This architecture is similar to the Xilinx Virtex II *slice* [9]. The coarse-grained embedded blocks (EBs), such as memories and multipliers, are surrounded by CLBs.

An example of our hybrid FPGA showed in Fig. 1 embeds four EBs are positioned tightly in the center, each taking the place of $3 \times 3$ array of tiles and surrounded by CLBs. They are connected by uniform width vertical and horizontal wires. The channel contains $W$ parallel routing tracks of segment length $L = 1$ and is connected to neighboring CLBs or EBs using a connection box. A switch box is located at the intersection of each segment channel and offers each incoming wire the ability to connect to three other wire segments ($Fs = 3$) [14]. We use the subset switch box (disjoint switch box). All CLBs, EBs and I/O pads are fully connected to connection boxes, i.e., $Fc_{\text{output}} = 1$, $Fc_{\text{input}} = 1$, and $Fc_{\text{pad}} = 1$ [14]. In addition, since the programmable routing resources consume about 70% of the area in a die [1], we add 70% extra area to the coarse-grained block for the vertical, horizontal routing tracks and switches. In [15], the area of the routing switches in the coarse-grained block reduces routing area and does not affect the area of block when $W$ is less than 33 for 0.13-$\mu$m process. We assume that eight-metal layer process technology is used.

Based on the physical die area and photomicrograph of a Virtex II device described in [16], we estimate that 60% of the total die is used for logic blocks. Our area model uses a featue size of 0.15-$\mu$m and assumes that each CLB in our fine-grained fabric has the same area as the Virtex II CLB (10 912 $\mu\text{m}^2$). Each CLB has two 4-LUTs ($N = 2$), 13 input pins, 5 output pins, and the maximum combinational delay of 0.45 ns.

We estimate the corresponding resistances, capacitances and intrinsic delay of a tri-state buffer via SPICE simulations. We also estimate the area of the tri-state buffer with 5 times minimum driving strength as being 34.5 times the minimum transistor area, and the wire width and spacing as being 1.5 times the width of a minimum transistor. We use a segment length $L$ of 4 which gives a good area-delay product [14]. Finally, we estimate the routing area of our architecture using the same model as [14]. This involves summing the area of multiplexers and tri-state buffers used in the circuit.

### B. Coarse-Grained Block

A coarse-grained FPU is composed of FAs, FMs, and WBs. The FAs and FMs are double precision (64 bit) and fully IEEE 754 compatible including all four rounding modes, denormalised numbers and status signals [17].

As described in [18], each WB contains $N$ identical bitblocks, each consisting of two 4-input LUTs and a reconfigurable register. The value of $N$ depends on the bit-width of the FPU. Bitblocks within a wordblock are all controlled by the same set of configuration bits, which perform the same function. A wordblock can efficiently implement operations such as fixed point addition and multiplexing. Therefore, some bit-level operations in FP primitives can be supported inside the FPU, to reduce the need for communication between the FPU and the fine-grained fabric. Chong *et al.* [4] suggested a similar idea of including fixed point units in an FPU, but with a different architecture. In our previous work [3], WBs, FAs, and FMs are connected using a local bus. This avoids using the fine-grained routing resources of the FPGA for connections that can be implemented inside the FPU, see Fig. 2.

### C. Interface

Coarse-grained blocks are able to connect to fine-grained resources in various ways. Coarse-grained blocks are usually large and have high I/O density, resulting in a disruption in the routing fabric, similar to that of MegaRAM blocks in the Altera Stratix III device [19]. Our previous work [15], [20] shows that the

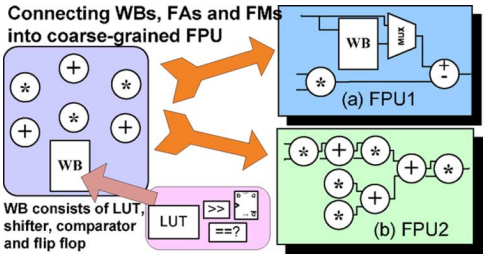



Fig. 2. Connecting WBs, FAs, and FMs into different coarse-grained FPUs.

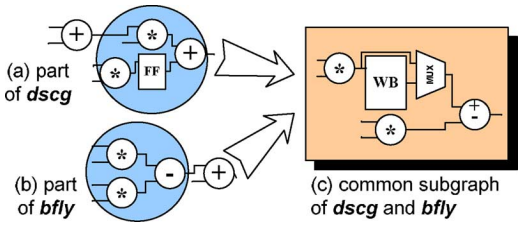



Fig. 3. Common subgraph extraction for an FP applications.

traditional column-based hybrid FPGA architecture is not particularly suitable for large coarse-grained blocks and concludes that coarse-grained blocks should be: 1) square in aspect ratio; 2) closely packed together; 3) positioned near the center of the FPGA; and 4) have I/O pins arranged on all four sides of the block. The interface between coarse-grained blocks and fine-grained blocks in this paper follows these recommendations.

## IV. FPU Optimizations and Methodology

This section considers three types of optimizations: the internal structure of the FPU, optimizations for density and flexibility, and the merging of FPUs into larger composite structures. The techniques described here are not restricted to our specific architecture; they can be extended to optimize the other FPU architecture such as that in [4].

### A. Internal Optimization of FPUs

As described in Section III, our assumed FPU consists of floating point adders and multipliers (FAs and FMs) as well as fixed point coarse-grained WBs. The first optimization we describe is the optimization of the exact number of each type of subunit within each FPU, as well as the pattern in which these subunits are connected. Fig. 2 shows an example of two different potential FPU architectures with different internal structures.

To derive candidate FPU structures, we employ *common subgraph extraction*. In this methodology, we systematically analyze a set of benchmarks, and extract patterns of FP operations that commonly appear in these circuits. Fig. 3 is an example of a common subgraph of two circuits (*dscg* and *bfly* benchmarks). A single unit which combines the common FP operations can be extracted, see Fig. 3(c).

A unique feature of our approach is the manner in which we handle fixed point operations using wordblocks in the FPU. Thus, creation of the FPU requires considering both fixed point and floating point functions in the application circuits. Our approach is to include a multiplexer to the input of the FA or FM in the common subgraph when there is a fixed point operation

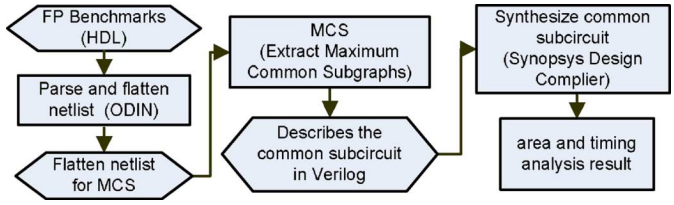



Fig. 4. Common subgraph extraction design flow.

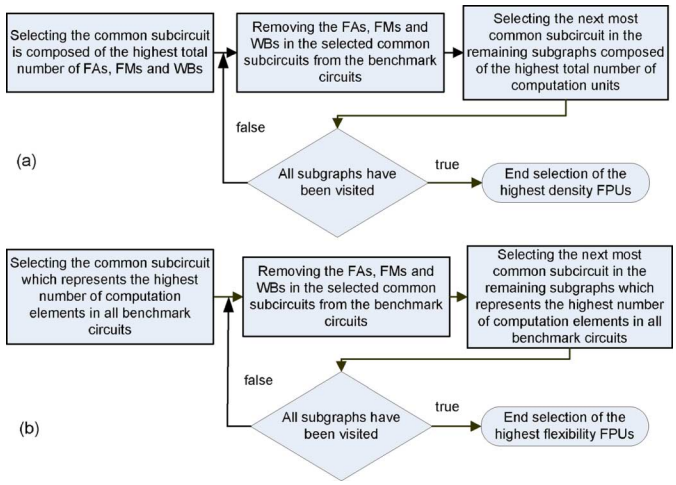



Fig. 5. Flow of the selection of (a) the highest density FPUs and (b) the highest flexibility FPUs in hybrid FPGAs.

such as FF, XOR, and AND connecting to the FA or FM in one of the analyzed benchmarks. This multiplexer allows selecting internal signal from WBs implementing a fixed point operation, internal signal from FM or FA implementing a FP operation, or external signal. The combination of fixed point WBs with FAs and FBs leads to more efficient circuit implementions, which reduces the slow communication between the FPU and the fine-grained fabric.

In order to evaluate candidate FPU architectures, we employ the flow in Fig. 4 to obtain area and timing estimates for each candidate. Benchmark circuits are written in Verilog. ODIN [21] is used to parse and flatten the circuits. The flattened netlist is then fed into a maximum common subgraph (MCS) generation stage to extract the common subgraphs over a set of circuits, as described in Section V. We describe the coarse-grained FPU in another Verilog file. The FPU is then synthesized using the Synopsys Design Complier V-2008.09 and a 0.13-$\mu$m process. We obtain the area and delay of the FPU and use this information to evaluate its performance using VPH, described later.

### B. System-Level Optimizations

The experiments in Section V involve two candidate FPU architectures: one optimized for density and one optimized for flexibility, see Fig. 5.

The first FPU we consider is one which has been optimized for overall density. An FPU with more computational elements achieves a greater reduction in area since connections between components in the FPU can be made locally. However, larger blocks may require more routing resources for connections to fine-grained blocks, and may lead to a reduction in flexibility

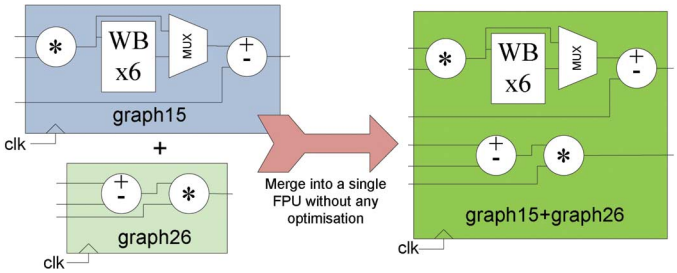



Fig. 6. Merging *graph15* and *graph26* into a larger FPU.

since it is difficult to reuse them in other applications. As shown in Fig. 5(a), in our flow to construct an FPU optimized in this way, we choose common subcircuits which are as large as possible. We then remove the selected subgraph pattern from each benchmark circuit, and repeat until all subgraphs are visited.

The second FPU we consider is one optimized for overall flexibility. As shown in Fig. 5(b), the difference between this flow and the previous one is that when we optimize for overall flexibility, we choose common subcircuits which appear in as many user benchmark circuits as possible.

These two candidate FPUs represent two extremes. In Section V, we present results only for these two extremes. However, clearly, there are other candidates that can be developed by combining the two optimization goals, and this is an interesting area for future work.

### C. Optimization by Merging FPUs

The final type of optimization we consider is the merging of two different types of FPUs into one composite FPU. The more distinct types of FPUs exist in an FPGA, the more placement constraints the architecture imposes, since each subgraph in the circuit may only be able to be implemented in one of the FPU types. By combining FPUs into larger FPUs, these placement constraints may be relaxed, leading to more efficient implementations. Fig. 6 shows an example of merging *graph15* and *graph26* into a larger composite FPU (*graph15* + *graph26*). This is similar to the approach of Quinnel *et al.* [22] who merged FP multiply and addition to a fused multiply-add unit in the FPU of the AMD processor. This architecture is 30% to 70% faster than the original one.

In this work we do not consider routing or logic resource sharing when merging FPUs into composite structures (except for the clock pin). We expect sharing resources may lead to improved efficiency; this is an interesting area for future work.

When merging FPUs, there are two important considerations. First, merging may lead to reduced placement constraints, leading to a reduction in the overall wirelength of the implemented circuit. The position of various FPUs are fixed so only the same FPU type can be swapped during the placement stage. This leads to inflexible placement and may introduce long wires between FPUs, see Fig. 7(a). Merging different FPU types into a larger FPU may lead to a better placement and reduce the wirelength. Fig. 7(b) shows an example of merging FPU1 and FPU3. FPU1_3 can be swapped to optimize connections such as $Net1$ and $Net3$. $Net2$ becomes a short wire and the delay of the circuit is reduced.

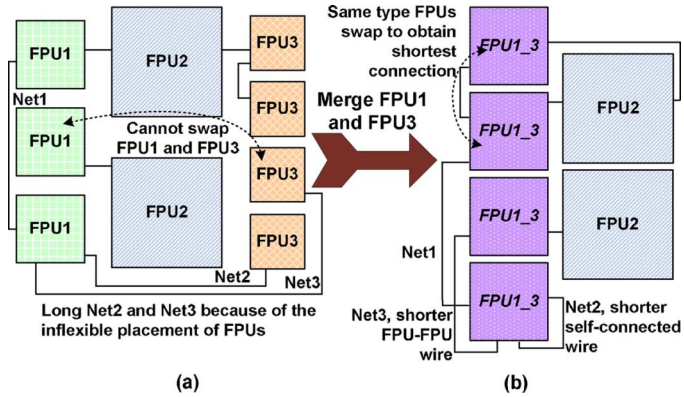



Fig. 7. Merging different types of FPUs can obtain better placement and reduce wirelength. (a) FPU1 and FPU3 cannot be swapped to reduce the length of wires. (b) Merging FPU1 and FPU3 can reduce the length of wires.

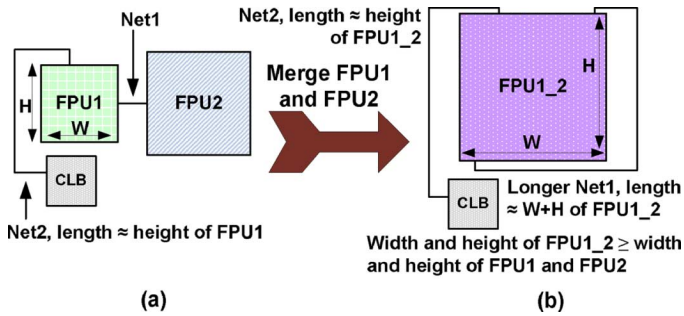



Fig. 8. Increase wirelength when merging FPUs. (a) Shorter cross FPU/CLB wires. (b) Longer cross FPU/CLB wires, more long self-connected wires.

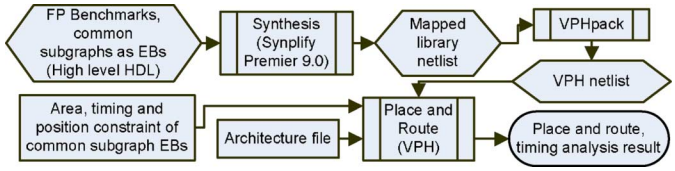


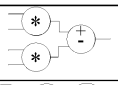
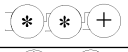
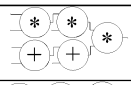
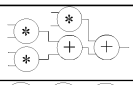
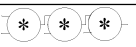
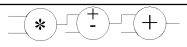
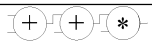
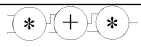
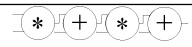
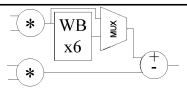
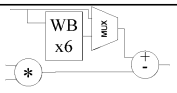
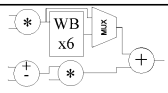
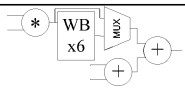
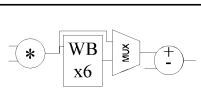
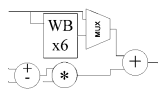
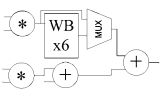
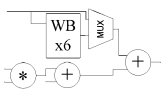
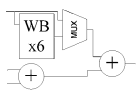
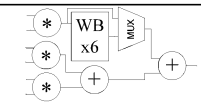
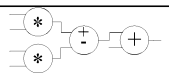
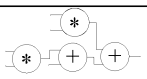
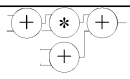
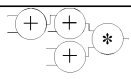
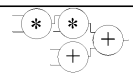
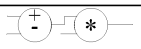
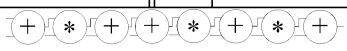
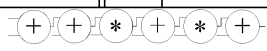
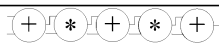
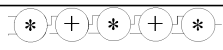
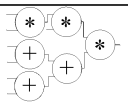
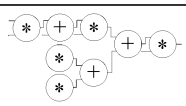
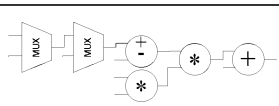
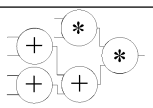
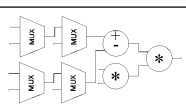
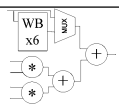
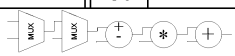
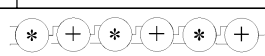
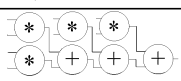

Fig. 9. Design flow for common subgraph EBs using VPH.

Second, a larger FPU will require more chip area, and may lead to an increase in overall wirelength. This is illustrated in Fig. 8. The original $Net1$ between CLB and FPU1, and $Net2$ between FPU1 and FPU2 are short, see Fig. 8(a). In Fig. 8(b), the width and height of the merged FPU1_2 is larger, leading to longer nets and hence increased delay.

### D. Evaluation Methodology

In Section V, we evaluate these optimizations in the context of a complete FPGA. We use the VPH tool [23] for place and route. VPH is a modified version of the VPR tool that supports EBs, memories, multipliers, carry chains, and user constraints. The tool flow is illustrated in Fig. 9. Circuits described in VHDL are synthesized using Synplify Premier 9.0. VPHpack packs and clusters LUTs and registers into CLBs. The area, timing and position of EBs are specified in a user constraint file. The architecture file contains the architectural information including the delay of the LUTs and registers. VPH performs placement, routing and estimates the area and delay for each circuit.

TABLE I
COMMON SUBGRAPH STRUCTURE OCCURRED IN BENCHMARK CIRCUITS (WBX6 MEANS THERE ARE SIX WBS CONNECTED SERIALLY)

| 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 |  | | | | | | |

## V. EVALUATION

This section introduces the FP benchmarks used; evaluates the area and delay impact of internal and system-level optimizations of coarse-grained FPUs based on common subgraphs; and finally optimizes the systems by merging different FPUs into a larger FPU. We assume that the channel width is 1.2 times the minimum channel width required to implement each circuit in all experiments.

### A. FP Benchmark Circuits

To explore the design of a hybrid FPGA based on common subgraph extraction and synthesis, a set of FP designs are used as benchmark circuits. They are: 1) $dscg$, a datapath of four digital sine-cosine generators; 2) $bfly$, the basic computation of fast Fourier transform: $z = y + x * w$ where inputs and output are complex numbers; 3) $fir$, four 4-tap finite impulse response filters; 4) $ode$, four circuits to solve ordinary differential equations; 5) $mm3$, four $3 \times 3$ matrix multipliers; 6) $bgm$, a circuit to compute Monte Carlo simulations of interest rate model derivatives; 7) $syn2$, a circuit containing 5 FAs and 4 FMs; and 8) $syn7$ a circuit containing 25 FAs and 25 FMs. $syn2$ and $syn7$ are two synthetic benchmark circuits generated by a synthetic benchmark circuit generator. These eight double precision FP benchmark circuits are not efficiently implemented in fine-grained FPGAs, since the FP computation requires a great deal of fine-grained resources.

### B. FPU Architecture Optimization

We determine the common subgraphs of FP in the benchmark circuits. The common subgraphs are shown in Table I, and are found using the common subgraph extraction technique described in Section IV. This technique can enumerate all possible common subgraphs; we only include subgraphs with two or more nodes. *graph2,3,4,6,9,10,23,24,25,27,28,30,31,32,33,34,35,39,40,41* occur in two benchmark circuits; *graph5,8,13,22,36* occur in three benchmark circuits; *graph14,16,17,20,38* are common in four benchmark circuits; *graph1,7,18,21,26,37* are common in five benchmark circuits; *graph11,19,29* are used in six benchmark circuits; and *graph12,15* are used in seven benchmark circuits. Since our standard cell library is for a 0.13-$\mu$m process and our fine-grained fabric modelled in VPH uses a 0.15-$\mu$m process, normalized area (area/feature size squared) is used. The equivalent area of the FPU in CLB is rounded to an integer value. The minimum channel width $W$ is larger than 33, therefore no additional area for switches inside FPU is required as suggested in [15].

Table II shows the frequency, normalized area, delay, number of input/outputs, and latency of the common subgraphs selected via system-level optimization. Embedding more FAs and FMs in an FPU can achieve an area reduction compared to those FPU with less FAs and FMs. It is because all the elements are compacted into a single unit.

TABLE II
STATISTICS FOR SELECTED SUBGRAPHS

(The feature size (L) of the coarse-grained units is 0.13$\mu$m. Virtex II CLB: area is 10,912$\mu$m$^2$, feature size is 0.15$\mu$m, normalized area is 485,013, the timing constraint for synthesizing the units is 2ns, 70% extra area is added to coarse-grained unit)

| Graph no. | Occurrence in benchmarks | | | | | | | | area(A) ($\mu$m$^2$) | normalized area=A/L$^2$ | area in CLB | delay (ns) | no. input | no. output | latency (cycles) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | dscg | bfly | fir | ode | mm3 | bgm | syn2 | syn7 | | | | | | | |
| WB | No separate WB embedded | | | | | | | | 37,135 | 2,197,321 | 7.7 | 0.76 | 104 | 38 | 1 |
| FA | 4 | 4 | 3 | 3 | 2 | 9 | 5 | 25 | 75,681 | 4,478,179 | 15.7 | 2.88 | 67 | 39 | 6 |
| FM | 4 | 4 | 4 | 2 | 3 | 11 | 4 | 25 | 214,348 | 12,683,315 | 44.5 | 2.95 | 67 | 39 | 6 |
| 12 | 8 | 8 | 12 | 4 | 4 | 7 | 0 | 16 | 547,774 | 32,412,679 | 113.6 | 3.17 | 127 | 77 | 18 |
| 15 | 8 | 8 | 12 | 4 | 8 | 9 | 0 | 16 | 548,265 | 32,441,718 | 113.7 | 3.17 | 127 | 77 | 18 |
| 20 | 0 | 0 | 4 | 0 | 4 | 2 | 0 | 2 | 1,021,944 | 60,470,082 | 212.0 | 3.09 | 226 | 98 | 18 |
| 26 | 8 | 0 | 0 | 4 | 0 | 5 | 1 | 15 | 282,039 | 16,688,697 | 58.5 | 2.95 | 100 | 46 | 12 |
| 37 | 0 | 4 | 4 | 0 | 4 | 2 | 0 | 2 | 829,449 | 49,079,801 | 172.0 | 3.17 | 193 | 91 | 18 |
| 41 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1,505,000 | 89,053,247 | 312.1 | 2.95 | 397 | 109 | 54 |

TABLE III
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 | 4 | 2 | 8 | 8 | 1 | 4 | 4 | 8 | 3 | 8 | 8 |
| **Benchmark** | **Utilization rate (%)** | | | | | | | | | | | | | |
| dscg | 64 | 64 | 50 | 0 | 0 | 100 | 100 | 0 | 0 | 0 | 100 | 0 | 100 | 100 |
| bfly | 64 | 64 | 50 | 0 | 0 | 100 | 100 | 0 | 0 | 100 | 50 | 0 | 50 | 50 |
| fir | 48 | 64 | 75 | 0 | 0 | 0 | 50 | 0 | 100 | 0 | 50 | 0 | 0 | 0 |
| ode | 48 | 32 | 25 | 0 | 0 | 100 | 50 | 0 | 0 | 0 | 50 | 0 | 100 | 50 |
| mm3 | 32 | 48 | 25 | 100 | 0 | 0 | 12.5 | 0 | 100 | 0 | 0 | 0 | 0 | 0 |
| bgm | 36 | 44 | 43.75 | 50 | 0 | 0 | 25 | 100 | 25 | 0 | 12.5 | 0 | 0 | 25 |
| syn2 | 20 | 16 | 0 | 0 | 50 | 50 | 37.5 | 0 | 0 | 0 | 0 | 33.33 | 50 | 37.5 |
| syn7 | 100 | 100 | 100 | 0 | 100 | 87.5 | 87.5 | 100 | 0 | 50 | 100 | 100 | 50 | 62.5 |

### C. System-Level Optimization

We evaluate the impact of embedding the new FPUs into a hybrid FPGA. Based on the optimization parameters in Section IV, the delay, area and routing resources of purely FA/FM FPGA, and mixture of subcircuits are examined. In the purely FA/FM FPGA, 25 FA, and 25 FM are used. We select two systems based on density and flexibility of FPU which are compared to the purely FA/FM FPGA. These two architectures are the extreme cases so the boundaries of performance can be explored.

*1) Density:* The architecture optimized for density is constructed by creating an FPU that contains as many FAs, FMs, and WBs, as possible. For example, *graph41* has 11 nodes that are a combination of FAs and FMs, contains most computation elements among subgraphs in Table II. Since the density and area reduction of this subcircuit is greatest compared to having separate FA, FM, and WB, this set of subcircuits may be the best choice to reduce the area of the hybrid FPGA. The selection is based on the scheme in Fig. 5(a).

We choose 7 types of FPUs: ***graph41, graph20, graph37, graph12, graph26, FM,*** and ***FA*** as subcircuits to embed in the hybrid FPGA (*FPGA_41_20_37_12_26*).

*2) 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 II, based on the flow in Fig. 5(b). *graph12* has the highest occurrence rate (16 times) among all the subgraphs. We choose five types of FPUs: ***graph12***, ***graph15***, ***graph26***, ***FM***, and ***FA*** to embed in the hybrid FPGA (*FPGA_12_15_26*).

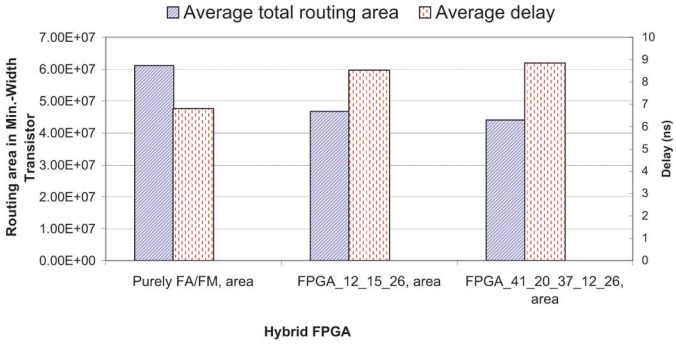



Fig. 10. Delay and average routing area using different types of FPUs.

From the three hybrid FPGAs we have selected: 1) **Purely FA/FM FPGA**; 2) ***FPGA_12_15_26***; and 3) ***FPGA_41_20_37_12_26***, we examine their impact of area and delay on the applications. The utilization rate of the subcircuits in each hybrid FPGA for benchmark circuits is shown in Table III. The two selection methods are greedy, and do not consider any connections of the chosen subgraphs.

*3) Delay Impact:* Fig. 10 shows that a purely FA/FM hybrid FPGA achieves the highest speed. The delay of purely FA/FM FPGA is 20.1% and 23% less than *FPGA_12_15_26* and *FPGA_41_20_37_12_26*, respectively. We have found 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

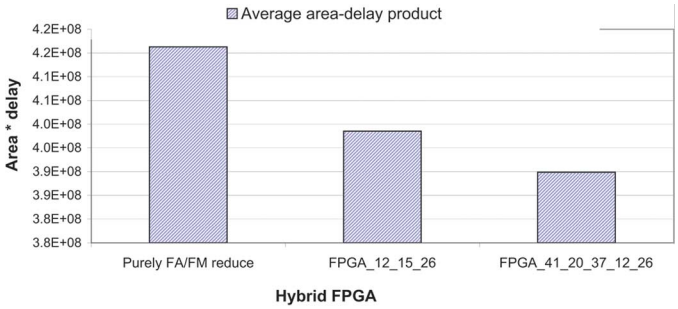



Fig. 11. Area-delay product of different types of FPGA.

architectures, they cannot be swapped to get better placement. For example, we cannot swap *graph41* and *graph20*, but two *graph12*'s can be swapped. The purely FA/FM system has the smallest number of distinct types of subcircuits. Therefore, due to the reduced placement constraints, FPUs in a purely FA/FM system have more freedom to be moved. This can be reflected by the wirelength, where the wirelength of purely FA/FM is 6.7% shorter than *FPGA_12_15_26* and 21% shorter than *FPGA_41_20_37_12_26*.

*4) Routing Area Impact:* On average, the total routing area of *FPGA_41_20_37_12_26* and *FPGA_12_15_26* are 27.9% and 23.5% less than the purely FA/FM FPGA as shown in Fig. 10. In *FPGA_41_20_37_12_26* and *FPGA_12_15_26*, most of the connections are in the FPUs therefore they can use less routing resources for interconnection. *FPGA_41_20_37_12_26* is the most compact and consumes less total CLBs (including the area of FPUs). Therefore, it uses less total routing area.

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

*5) Area-Delay Product Impact:* We present the overall area-delay product of the systems in Fig. 11. *FPGA_41_20_37_12_26* achieves the best area-delay product. It is 2.2% and 6.4% better than *FPGA_12_15_26* and purely FA/FM FPGA, respectively. *FPGA_41_20_37_12_26* is slower than the other systems; it consumes less routing resources. Overall, *FPGA_41_20_37_12_26* is the best for both speed and area. We believe that between the two extreme systems, the lower density FPU should achieve higher speed while the less flexible FPU should consume more routing area.

### D. Optimization by Merging FPUs

As explained in Section IV, an FPGA with many distinct types of FPUs may impose a large number of placement constraints, which may lead to an increase in wirelength. We evaluate the extent to which FPU merging may improve the overall wirelength, and delay of circuits implemented on a hybrid FPGA.

*1) Merging Scheme:* We merge the FPUs with similar number of subcircuits in the FPGAs stated in Table III which minimizes the waste of unused FPUs. For example, in *FPGA_41_20_37_12_26*, *graph37* occurs four times and *graph26* occurs three times. We merge them into a single FPU *graph26* + *graph37*. We consider five different merge scheme

TABLE IV
DIFFERENT FPUS MERGED IN THE THREE FPGAS

| Merge scheme | Pure FA/FM FPGA | FPGA_12_15_26 | FPGA_41_20_37_12_26 |
|---|---|---|---|
| A | (1)FA (2)FM | (1)graph12 (2)graph15 (3)graph26 (3)graph26 (4)FA (5)FM | (1)graph12 (2)graph20 (3)graph26 (4)graph37 (5)graph41 (6)FA (7)FM |
| B | (1)FA +FM | (1)graph12 (2)graph15 (3)graph26 (4)FA+FM | (1)graph12 (2)graph20 (3)graph26 (4)graph37 (5)graph41 (6)FA+FM |
| C | (1)5*FA+ 5*FM | (1)graph12 (2) FA+FM (3)graph15+graph26 | (1)graph12 (2)graph20 (3)graph26+graph37 (4)graph41 (5)FA+FM |
| D | (1)7*FA+ 7*FM | (1)graph12+graph15+ graph26+FA+FM | (1)graph12+graph20+ graph26+graph37+ graph41+FA+FM |
| E | (1)10*FA+ 10*FM | (1)4*graph12+graph15+ graph26+2*FA+2*FM | (1)2*graph12+graph20+ graph26+graph37+ +graph41+2*FA+2*FM |

TABLE V
STATISTIC OF THE MERGED FPUS IN THE FPGAS

| System: Purely FA/FM FPGA | | | | |
|---|---|---|---|---|
| Merged FPU type | Area in CLB | Delay (ns) | No. FPU | I/O density |
| FA+FM | 56 | 2.95 | 25 | 13.43 |
| 5*FA+5*FM | 289 | 2.95 | 5 | 29.46 |
| 7*FA+7*FM | 400 | 2.95 | 4 | 35.04 |
| 10*FA+10*FM | 576 | 2.95 | 3 | 41.70 |
| **System: FPGA_12_15_26** | | | | |
| Merged FPU type | Area in CLB | Delay (ns) | No. FPU | I/O density |
| FA+FM | 56 | 2.95 | 8 | 13.43 |
| graph15+graph26 | 169 | 3.17 | 4 | 11.62 |
| graph12+graph15 graph26+FA+FM | 342 | 3.19 | 16 | 18.014 |
| 4*graph12+graph15 graph26+2*FA+2*FM | 756 | 3.19 | 4 | 24.73 |
| **System: FPGA_41_20_37_12_26** | | | | |
| Merged FPU type | Area in CLB | Delay (ns) | No. FPU | I/O density |
| FA+FM | 56 | 2.95 | 8 | 13.43 |
| graph26+graph37 | 225 | 3.17 | 4 | 12.47 |
| graph12+graph20+graph41+ graph37+graph41+FA+FM | 930 | 3.19 | 8 | 24.11 |
| 2*graph12+graph20+graph26+ graph37+graph41+2*FA+2*FM | 1406 | 3.19 | 4 | 32.01 |

for the evaluation. These schemes are denoted A, B, C, D, and E in Table IV. The FPGA in scheme A contains smaller FPUs while the FPGA in scheme E composes larger merged FPUs. Table V shows the area, delay and the number of FPUs embedded in the FPGAs. We examine the impact on area, delay and wirelength of the merged FPUs in the hybrid FPGAs based on the these FPU results. The five schemes were selected to demonstrate the significant changes in performance. For example in the pure FA/FM system, $\mathrm{FA}+\mathrm{FM}$ and $2*\mathrm{FA}+2*\mathrm{FM}$ system have the same area and delay, therefore we do not show those schemes in this paper.

*2) Delay and Wirelength Impact:* Fig. 12 shows there is a maximum delay reduction in the five merge schemes. The delay is the critical path after place and route, averaged over the eight benchmarks. Merge scheme A is the original FPGA without any merged FPUs. In scheme B, a purely FA/FM hybrid FPGA reduces delay by 3.1% and *FPGA_41_20_37_12_26* reduces it by 3.5%, while *FPGA_12_15_26* achieves a 1.6% delay reduction. Further increases in the size of merged FPUs increases the delay due changes in width $(W)$ and height $(H)$ of the merged FPUs. Table VI shows the average wirelength and the maximum

TABLE VI
AVERAGE WIRELENGTH OF THE THREE FPGAS IN DIFFERENT MERGE SCHEMES

| | Purely FA/FM FPGA | | | FPGA_12_15_26 | | | FPGA_41_20_37_12_26 | | |
|---|---|---|---|---|---|---|---|---|---|
| Merge Scheme | Average wirelength | Max. W+H of all FPUs | Max. W+H of merged FPUs | Average wirelength | Max. W+H of all FPUs | Max. W+H of merged FPUs | Average wirelength | Max. W+H of all FPUs | Max. W+H of merged FPUs |
| A | 24.08 | 13 | – | 25.80 | 21 | – | 30.48 | 35 | – |
| B | 23.46 | 15 | 15 | 29.11 | 21 | 15 | 32.19 | 35 | 15 |
| C | 26.87 | 34 | 34 | 28.97 | 26 | 26 | 28.97 | 35 | 30 |
| D | 28.63 | 40 | 40 | 35.75 | 37 | 37 | 43.55 | 61 | 61 |
| E | 28.96 | 48 | 48 | 37.16 | 55 | 55 | 43.55 | 75 | 75 |

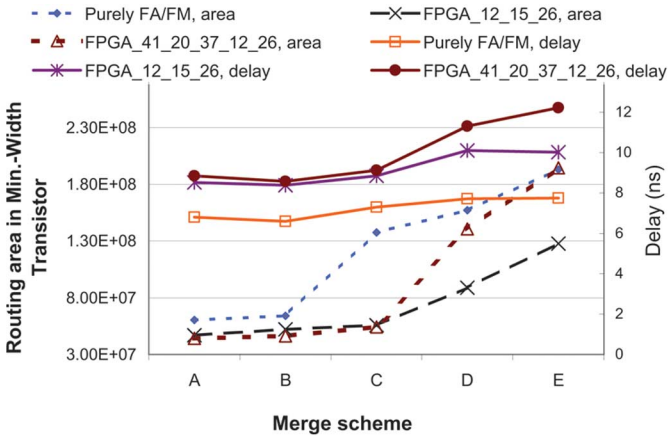



Fig. 12. Delay and routing area of FPGAs using different merging methods.

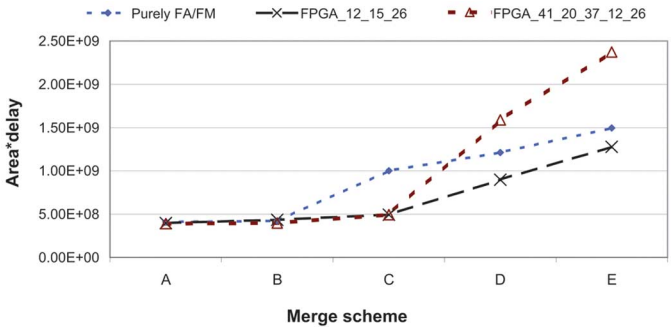



Fig. 13. Area-delay product of FPGAs using different merging methods.

$W + H$ of the FPUs and merged FPUs. The wirelength generally increases with larger FPUs. When the $W+H$ of the merged FPUs are shorter than the average wirelength, the speed can be improved because of the both inter and intra-FPUs are short as described in Fig. 7. Once the $W + H$ of the merged FPUs exceed the average wirelength, the inter-FPU wires are short, but the intra-FPGA wires are long as shown in Fig. 8, causing a decrease in speed.

*3) Area Impact:* Fig. 12 shows the total routing area used in each scheme. *FPGA_41_20_37_12_26* has the smallest area. Larger and more compact FPUs have higher I/O density as shown in Table V. For example, FA and FM originally have 7.7 and 12.6 I/O pins per CLB length, respectively. The I/O density of merged FA + FM FPU is 13.43, which is 74% more than FA and 6.6% more than FM. Therefore, large FPUs in scheme B, C, D, and E require larger routing area.

*4) Area-Delay Product Impact:* Finally, Fig. 13 shows the area-delay product of each scheme. The area-delay product of scheme A and B are similar in all the three systems. Scheme B has an advantage in speed which can compensate the loss in routing area. Scheme A is opposite to scheme B, which has better area but is slower. Scheme C, D, and E include more compact FPUs; the area gained in individual FPUs cannot compensate the loss in speed and routing area. As a result, the larger merged FPUs cause a worse area-delay product.

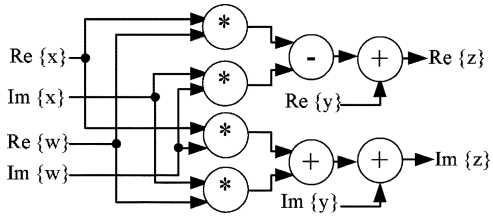



Fig. 14. Circuit diagram of one core of *bfly*.

## E. Throughput of FP Computation

Throughput of the FP computation is a major concern for designing FP applications. We show that the high density system *FPGA_41_20_37_12_26* is 27.9% smaller than the others. We can embed more FPUs in the high density system compared to the other systems on an fixed-area FPGA. As a result, it can provide more FP operators for computation at the same time and achieve higher data throughput. For example, one core of the benchmark *bfly* (shown in Fig. 14) requires four FAs and four FMs. The purely FA/FM FPGA can implement six *bfly* cores, while *FPGA_41_20_37_12_26* can implement eight *bfly* cores. An application requires to compute more than eight *bfly* is possible to achieve about 33% higher data throughput by using high density FPGA than using purely FA/FM FPGA.

# VI. CONCLUSION

This paper proposes a novel methodology to determine optimized coarse-grained FPUs in hybrid FPGAs, based on common subgraph extraction. Floating point circuits are not efficiently implemented in fine-grained FPGA and we have demonstrated the impact of embedding multiple types of FPUs on a hybrid FPGA on speed and area. We explore the internal and system-level optimization of the FPU. The effect of merging different coarse-grained types into larger FPUs is also studied. We observe that: 1) the speed of the system is the highest for implementations involving only FAs and FMs; 2) higher density subgraphs produce greater reduction on area; 3) they provide the best area-delay product; and 4) merging of FPUs can improve the speed of hybrid FPGAs, but results in consuming more area. Our research reveals that high density FPUs contribute to high system performance and high

data throughput. Future work includes generalizing our model to support multiple types of embedded blocks for different application domains.

## REFERENCES


[1] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, 2007.

[2] M. J. Beauchamp, S. Hauck, K. D. Underwood, and K. S. Hemmert, "Architectural Modifications to Enhance the Floating-Point Performance of FPGAs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 2, pp. 177–187, Feb. 2008.

[3] C. H. Ho, C. W. Yu, P. H. W. Leong, W. Luk, and S. J. E. Wilton, "Floating-point FPGA: Architecture and modeling," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 2, pp. 1709–1718, Dec. 2009.

[4] Y. J. Chong and S. Parameswaran, "Flexible multi-mode embedded floating-point unit for field programmable gate arrays," in *Proc. FPGA*, 2009, pp. 171–180.

[5] S. S. Demirsoy and M. Langhammer, "Cholesky decomposition using fused datapath synthesis," in *Proc. FPGA*, 2009, pp. 241–244.

[6] M. Langhammer and T. VanCourt, "FPGA floating point datapath compiler," in *Proc. FCCM*, 2009, pp. 259–262.

[7] A. M. Smith, G. A. Constantinides, and P. Y. K. Cheung, "Fused-arithmetic unit generation for reconfigurable devices using common subgraph extraction," in *Proc. ICFPT*, 2007, pp. 105–112.

[8] C. W. Yu, A. M. Smith, W. Luk, P. H. W. Leong, and S. J. E. Wilton, "Optimizing coarse-grained units in floating point hybrid FPGA," in *Proc. ICFPT*, 2008, pp. 57–64.

[9] Xilinx, Inc., San Jose, CA, "Virtex-II plaform FPGAs: Complete data sheet," 2005. [Online]. Available: http://direct.xilinx.com/bvdocs/publications/ds031.pdf

[10] A. Cevrero, P. Athanasopoulos, H. Parandeh-Afshar, A. K. Verma, H. S. A. Niaki, C. Nicopoulos, F. K. Gurkaynak, P. Brisk, Y. Leblebici, and P. Ienne, "Field programmable compressor trees: Acceleration of multi-input addition on FPGAs," *ACM Trans. Reconfig. Technol. Syst.*, vol. 2, no. 2, pp. 1–36, 2009.

[11] G. Govindu, L. Zhuo, S. Choi, and V. Prasanna, "Analysis of high-performance floating-point arithmetic on FPGAs," in *Proc. Parallel Distrib. Process. Symp.*, 2004, pp. 149–156.

[12] M. Langhammer, "Floating point datapath synthesis for FPGAs," in *Proc. FPL*, 2008, pp. 355–360.

[13] F. de Dinechin, C. Klein, and B. Pasca, "Generating high-performance custom floating-point pipelines," in *Proc. FPL*, 2009, pp. 59–64.

[14] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, 1999.

[15] C. W. Yu, W. Luk, S. J. E. Wilton, and P. H. W. Leong, "Routing optimization for hybrid FPGAs," in *Proc. ICFPT*, 2009, pp. 419–422.

[16] C. Yui, G. Swift, and C. Carmichael, "Single event upset susceptibility testing of the Xilinx Virtex II FPGA," presented at the Military Aerosp. Appl. Program. Logic Conf., Laurel, MD, 2002.

[17] R. Usselmann, "Floating point unit," 2005. [Online]. Available: http://www.opencores.org/project.cgi/web/fpu/overview

[18] S. Wilton, C. Ho, P. Leong, W. Luk, and B. Quinton, "A synthesizable datapath-oriented embedded FPGA fabric," in *Proc. FPGA*, 2007, pp. 33–41.

[19] D. Lewis, E. Ahmed, D. Cashman, T. Vanderhoek, C. Lane, A. Lee, and P. Pan, "Architectural enhancements in Stratix-III™ and Stratix-IV™," in *Proc. FPGA*, 2009, pp. 33–42.

[20] C. W. Yu, J. Lamoureux, S. J. E. Wilton, P. H. W. Leong, and W. Luk, "The coarse-grained/fine-grained logic interface with embedded floating-point arithmetic units," *Int. J. Reconfig. Comput.*, vol. 2008, 2008, Article ID 736203.

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

[22] E. Quinnell, E. E. Swartzlander, and C. Lemonds, "Bridge floating-point fused multiply-add design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 12, pp. 1727–1731, Dec. 2008.

[23] C. W. Yu, "A tool for exploring hybrid FPGAs," in *Proc. FPL*, 2007, pp. 509–510.


**ChiWai Yu** received the B.Eng. degree from Computer Science and Engineering Department, Chinese University of Hong Kong, Hong Kong, in 2000, and the Ph.D. degree from the Department of Computing, Imperial College London, London, U.K., in 2010, respectively.

In 2010, he joined the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, as a Research Associate. His research interest is in FPGA architecture and reconfigurable application.

**Alastair M. Smith** received the M.Eng. (Honors) and Ph.D. degrees in electrical and electronic engineering from Imperial College London, London, U.K., in 2003 and 2007, respectively.

He worked as a post-doctoral fellow with The University of British Columbia, Vancouver, BC, Canada, from 2007 to 2008. After another period as a Research Associate with the Circuits and Systems Research Group, Imperial College London, in 2010, he joined PA Consulting's Communications and Electronic System Practice.

**Wayne Luk** (F'09) is a Professor of computer engineering with Imperial College London, London, U.K. He was a Visiting Professor with Stanford University, Stanford, CA. His research interests include theory and practice of customizing hardware and software for specific application domains, such as multimedia, networking, and finance.

**Philip H. W. Leong** (SM'02) received the B.Sc., B.E., and Ph.D. degrees from the University of Sydney, Sydney, NSW, Australia.

In 1993, he was a Consultant with ST Microelectronics, Milan, Italy, where he worked on advanced flash memory-based integrated circuit design. From 1997 to 2009, he was with the Chinese University of Hong Kong. He is currently an Associate Professor with the School of Electrical and Information Engineering, University of Sydney. He is also a Visiting Professor with Imperial College London, London, U.K., and the Chief Technology Consultant to Cluster Technology.

**Steven J. E. Wilton** (SM'03) received the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 1992 and 1997, respectively.

In 1997, he joined the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, where he is currently a Professor. During 2003 and 2004, he was a Visiting Professor with the Department of Computing, Imperial College London, London, U.K., and at the Interuniversity MicroElectronics Center (IMEC), Leuven, Belgium.