Hardware Resource Estimation for Heterogeneous FPGA-based SoCs

Mariem Makni
LAMIH, University of Valenciennes, France
mariem.makni@etu.univ-valenciennes.fr

Mouna Baklouti
National Engineering School of Sfax, Tunisia
mouna.baklouti@enis.rnu.tn

Smail Niar
LAMIH, University of Valenciennes, France
smail.niar@univ-valenciennes.fr

Mohamed Abid
National Engineering School of Sfax, Tunisia
mohamed.abid@enis.rnu.tn

ABSTRACT

The increasing complexity of recent System-on-Chip (SoC) designs introduces new challenges for design space exploration tools. In addition to the time-to-market challenge, designers need to estimate rapidly and accurately both performance and area occupation of complex and diverse applications. High-Level Synthesis (HLS) has been emerged as an attractive solution for designers to address these challenges in order to explore a large number of SoC configurations. In this paper, we target hybrid CPU-FPGA based SoCs. We propose a high-level area estimation tool based on an analytic model without requiring register-transfer level (RTL) implementations. This technique allows to estimate the required FPGA resources at the source code level to map an application to a hybrid CPU-FPGA system. The proposed model also enables a fast design exploration with different trade-offs through HLS optimization pragmas. Experimental results show that the proposed area analytic model provides an accurate estimation with a negligible error (less than 5%) compared to RTL implementations.

CCS Concepts

•**Hardware** → *Hardware-software codesign;*

Keywords

HLS; FPGA; SoC; Hardware Accelerator

1. INTRODUCTION

Hybrid CPU-FPGA based SoCs [1] have recently emerged as an attractive solution for data-intensive processing in embedded systems such as multimedia computing, automotive, etc. This solution offers the flexibility and scalability of a Field Programmable Gate Array (FPGA), while providing performance through their hard-cores [2]. The time-to-market (TTM) challenge requires a fast design space exploration (DSE) in order to achieve the application performance and constraints. Rapid estimation of both performance and area occupation of diverse applications is an important challenge for designers. Current research in performance and area analysis relies on register-transfer level (RTL) based synthesis flows to produce accurate timing and area estimates. Besides, many techniques require commercial high-level synthesis (HLS) tools to perform area and performance estimations [3]. Such techniques not only require significant effort and time but are also difficult to use, making large DSE and TTM costly.

Recently, analytic models have been proposed to overcome these problems. These models are based on a set of equations to estimate the area occupation for several applications on different architectural configurations. They are considered as an attractive alternative to low-level hardware (HW) description languages (Verilog/VHDL). RTL-based HW synthesis tools are very time consuming and error prone for today's complex designs. Furthermore, it is critical for designers to evaluate whether the implementation of the chosen application meets the area requirements on an FPGA platform. Therefore, the area analytic models allow to quantitatively estimate the FPGA resources (DSP, BRAMs, etc.) required to map a given application to a HW system in an early design stage. Several works have investigated the high-level estimation aspect for different heterogeneous architectures but mostly from a performance perspective [4][5][6].

In this paper, we present an analytic model for area estimation in hybrid CPU-FPGA systems. This estimation is only considered for the reconfigurable part (FPGA) since hard-cores are integrated into an ASIC part. HLS optimization pragmas, such as dataflow, loop pipelining and array partitioning are taken into account in the proposed model. We aim to estimate the area occupation of different SoC configurations to select the adequate architecture faster and easier without generating RTL implementations.

SAC 2017,April 03-07, 2017, Marrakech, Morocco
Copyright 2017 ACM 978-1-4503-4486-9/17/04. . . $15.00
http://dx.doi.org/10.1145/3019612.3019683

This paper has two-fold contributions:

- Modeling accurately HW resources (DSP, BRAM, LUTs, FF) of different applications mapped to hybrid CPU-FPGA systems.
- Estimating the impact of optimization pragmas on the area occupation for hybrid CPU-FPGA systems under FPGA resource constraints.

Our proposed tool estimates the area of different configurations with 5% average error compared to RTL-based HW implementations. To demonstrate both accuracy and efficiency of our approach, various benchmarks have been tested.

The remainder of this paper is organized as follows. Section 2 presents an overview of the different HLS optimization pragmas. Section 3 discusses related works. An overview of our proposed area estimation tool is illustrated in Section 4. Our developed analytic estimation model is discussed in Section 5. Experimental results of the area estimation tool are described in Section 6. Section 7 concludes the paper with a brief outlook on future work.

2. BACKGROUND

Computational platforms such as the recent Xilinx SoC designs [2][7], are limited in terms of HW resources. In particular, BRAM and DSP blocks are the most critical resources to analyze since they are the fewest resources in most FPGA devices. Exploring the different SoC configurations to measure the FPGA resource occupation can be a complex and tedious task for designers as they need to run a full synthesis on each iteration of the SoC design. To tackle these issues, this paper proposes a fast method to obtain detailed HW resource estimations. Dataflow, loop pipelining and array partitioning are considered as the most prominent SW pragmas in modern HLS tools, such as Xilinx Vivado [8]. These pragmas have significant impact on HW resource utilization. Applying multiple pragmas produces various implementations with different performance/area tradeoffs. Hence, our proposed area model supports these three pragmas while enabling a fast architectural exploration to find the best design for a given application. The optimization pragmas, considered in this work, are detailed as follows [9]:

Loop pipelining: is a *fine grain* pipelining applied at the loop level, allowing parallel executions of loop iterations. When enabled, the HW performance is determined by a constant initiation interval (II) of the loop. II is defined as the number of clock cycles between the start times of consecutive loop iterations. This pragma provides higher throughput with less execution time.

Dataflow: In contrast to the loop pipelining pragma, dataflow exploits a *coarse grain* parallelism at the level of functions. Thus, it increases the concurrency between the different functions within the application, which can be consequently pipelined and executed in parallel.

Array partitioning: allows to split the arrays into multiple smaller arrays stored in multiple memory banks in the BRAMs. This pragma has the advantage of improving the memory bandwidth by increasing the number of load/store ports. In this work, we assume that each memory bank has only 2 read ports and 1 write port. Similar to Vivado HLS, our proposed tool supports *block*, *cyclic* and *complete* array partitioning strategies with different partition factors [8].

3. RELATED WORKS

Area estimation for FPGA-based SoCs has been studied by several researchers. Current state-of-the-art techniques primarily rely on creating RTL implementations to estimate the area occupation. However, this method is tedious and time consuming, especially for large and complex designs. It takes from minutes to hours to generate, simulate, and synthesize RTL to evaluate the area of each SoC configuration.

In some of the existing works [4][5][6], the authors ignore the FPGA resource constraints. In [10], the authors propose DSE techniques for applications that consist of multiple nested loops with or without data dependencies. However, in their work, area information are obtained from HLS tool invocations. Instead of invoking HLS tools for each design, we propose a fast and high-level area estimation tool without requiring RTL implementations. This helps designers to reduce the total run-time to perform a large design space. Holland et al. [11] present a methodology, called RC Amenability Test (RAT), for rapidly analyzing an application's design compatibility to a specific FPGA platform. Their paper discusses the common difficulties encountered during several FPGA application migration projects. However, their methodology is based on RTL implementations, a time consuming task to estimate the area occupation.

In contrast to the aforementioned methods, our proposed tool is a high-level estimation tool. It takes unmodified and high-level descriptions of algorithms, to generate dynamic data dependence graphs (DDDGs) as representations of the different Hardware Accelerators (HAs). In [12], the authors present an area estimation model for HAs. However, no details are given for the used analytic estimation model. The authors only compare between their results and the ones obtained from Vivado HLS to validate their area model. They estimate HA area within 7% average error compared to RTL implementations. Another area estimation model is presented in [13] to determine the required FPGA resources in order to hard-wire an algorithm. However, it is based on a static analysis. In comparison, our proposed methodology is based on a dynamic analysis. Furthermore, we aim to estimate not only the area of the computation part but also the area of the communication part including the data exchange between processors and HAs. We use the Advanced eXtensible Interface (AXI4) stream protocol [14], which is considered the most efficient design interconnect to communicate between the processor and the HA in the recent heterogeneous systems. Conducted experimental results show that the area estimated by our tool is considerably close to FPGA-based RTL implementations.

4. THE PROPOSED AREA ESTIMATION TOOL

Our proposed area estimation tool, illustrated in Figure 2, takes as inputs algorithms described in high-level languages (C/C++). It uses a dynamic analysis and DDDG graph

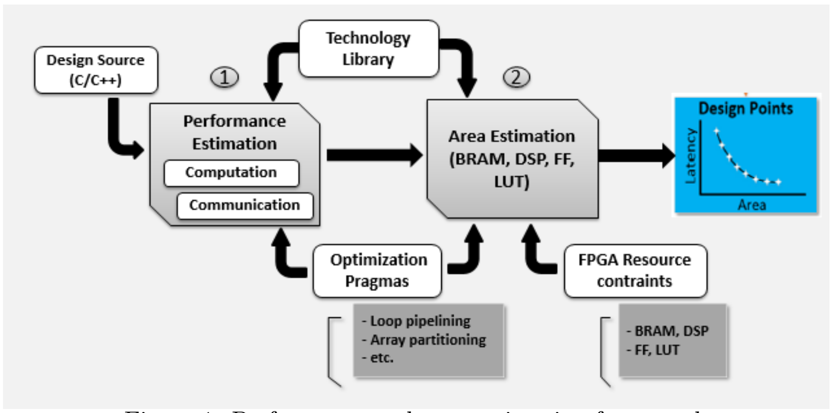

Figure 1: Performance and area estimation framework

generated from the program trace as a representation of the HA implementation [6]. A DDDG is an acyclic graph, in which the nodes represent the different operations including computation and memory instances, while the edges represent the dynamic data dependencies between the operations. Our proposed tool is integrated into a framework to estimate both performance and area occupation for a large number of design points with different pragmas, as shown in Figure 1.

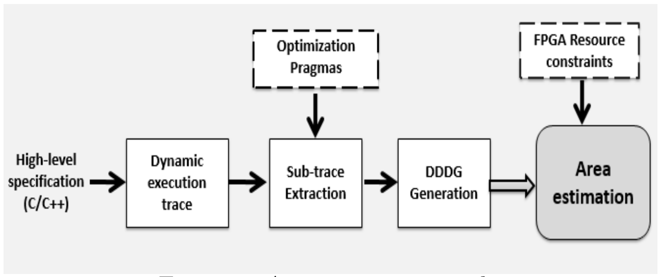

Figure 2: Area estimation tool

The proposed framework (Figure 1) consists of three important steps: computation estimation [6], communication estimation, which is out of the scope of this paper, and area estimation. In our proposed area model, designers can choose the pragmas for their SoC design. They can also perform a large DSE to evaluate all possible SoC configurations, in order to find the best design that matches the application characteristics and the architecture features. As shown in Figure 2, we use the optimization pragmas (loop pipelining, etc.) chosen by the designers to extract the subtrace and generate the DDDG graph. In contrast to existing works, our proposed tool only focuses on the relevant subtrace instead of modeling the whole application trace. Based on the DDDG, we model the HA area without generating RTL implementations. The proposed area estimation tool achieves a fast runtime (Table 1) even for applications with relative large input data size and different pragmas. Consequently, we may rapidly evaluate different heterogeneous parallel architectures including the Xilinx Zynq 7000 AP SoCs [2] that combine ARM-based symmetric multiprocessing and an FPGA.

5. ANALYTIC AREA ESTIMATION MODEL

In this paper, we estimate the area occupation of the HAs with the following assumptions: (1) HW functional units associated with nodes (DDDG) follow the default setting of Xilinx Vivado HLS [8] including FPGA resource occupation. For instance, we assume that a 32-bit floating-point addition node is mapped to a pipelined floating-point add (FA) unit consuming 2 DSP and zero BRAM; (2) Each memory bank supports only 2 reads and 1 write simultaneously; (3) Resource constraints are modeled for DSP, BRAM, LUT and FF. Our developed tool applies optimization pragmas as well as resource constraints to the DDDG graph to explore a large space with different trade-offs (Figure 2). We rigorously validated our estimation results against RTL implementations of the HAs for a wide range of applications, including streaming embedded applications. The experimental results show that our proposed tool can rapidly model area within 5% average error, compared to traditional RTL-based design flows.

Applications are firstly analyzed to determine their total input/output data size and computational density. In this work, we develop an efficient analytic model using a set of equations to estimate the area occupation of the application based upon different parameters generated from the DDDG graph, such as number of loop levels, loop bounds, input data size, etc. Our analysis allows then to estimate the application's HW usage in order to reject unfeasible designs. In addition, the AXI4 stream interface [14] used to exchange data between the processor and the HA using DMA (Direct Memory Access), can also consume a significant number of FPGA resources. This quantity depends on the number of input data arrays, denoted nIA and can be estimated by Equation 1. DMA_{area} represents the FPGA resources required by a DMA controller.

$$Com_{area} = (nIA) * DMA_{area} \tag{1}$$

In the following subsections, we will use nFA, nFS, nFM and nFD to represent respectively the total number of floating addition, floating subtraction, floating multiplication and floating division operations required for the application execution. These values are obtained from the DDDG graph generated from the dynamic execution trace (Figure 2). N_{op} represents the number of *computation* operation nodes (multiplication, addition, etc.), while N_m represents the total number of *memory* operations extracted from the DDDG graph. N_{load} and N_{store} represent the number of memory *load* and *store* operations respectively.

Let's consider a nested loop $K = \{K_1, .., K_i, .., K_L\}$, where K_L is the innermost loop level. B_K is the bound of the nested loop K. The iteration latency IL is the number of clock cycles required to perform a single iteration of the loop. L_K is the number of loop levels in the nested loop K. The number of single-level loops in an application is represented by S_l. In this paper, the total number of nested loops in a given application is represented by n. The constants used in analytic equations, denoted Ci, are determined by thoroughly analyzing the results obtained from Vivado HLS tool.

5.1 BRAM Estimation

Measuring the required block RAM (BRAM) resources is an important metric in the area estimation process. Based on experimental results, we note that the BRAM resources only depend on the array partitioning pragma. The memory bandwidth can be improved by splitting up the original arrays into multiple independent memory banks. Similar to Vivado HLS [8], our proposed tool supports *block*, *cyclic* and *complete* array partitioning strategies. It enables array partitioning by mapping addresses of memory nodes (load and store) in the DDDG graph to memory banks.

Our developed BRAM Resource Estimation algorithm, denoted BRE, allows to rapidly estimate the required BRAM resources to map the application to a HA implementation. It takes the parameters nA, T, Pf, S_A and S_{BRAM} as inputs and generates $BRAM_T$, which is the estimated total amount of BRAM required to implement the application. When applying the array partitioning pragma, we should specify the array A, the partition factor Pf, and the array partitioning type T, where $T = \{cyclic, complete, block\}$. Varying the partition factor Pf as well as the arrays data size, may increase or decrease the required BRAM resources. The S_A, respectively $S_A * Bits$, parameter represents the size of an array A, measured in *words*, respectively in *bits*. nA represents the number of arrays in a given application. S_{BRAM} is the size of a BRAM (in *bits*) in FPGA. It is a generic parameter that can be set by the designer. $BRAM_b$ is the number of BRAMs per memory bank.

In this work, BRAM 18Kbit was considered in the BRE algorithm. Each BRAM in FPGA can store 18432 *bits* [7]. Following the Vivado HLS settings, each array can be implemented as one BRAM with 2 read ports and 1 write port. Therefore, our proposed tool only allows up to 2 read or 1 write memory accesses for the same memory bank. The developed BRE algorithm (Algorithm 1) is useful for *cyclic* and *block* types. In fact, no BRAMs are required for the *complete* type, since in this type, the array is completely split into individual registers. As a result, we use Flip-Flops (FF) instead of BRAMs.

As shown in Line 9 of Algorithm 1, before multiplying the number of BRAMs per bank with the total number of memory banks or Pf, we should round $BRAM_b$. After that, once we verify that the obtained number of $BRAMs$ is a power of two, then $BRAM_T$ allocated to A is equal to $BRAMs$. Otherwise, we should find the nearest power of two of $BRAMs$, as shown in Line 14 of Algorithm 1.

Algorithm 1 BRAM Resource Estimation (BRE) Algorithm

```
Input: $nA$, $T$, $Pf$, $S_A$, $S_{BRAM}$
Output: Total BRAMs $BRAM_T$
 1: begin
 2: $BRAM_T \leftarrow 0$;
 3: $BRAM_b \leftarrow 0$;
 4: $R \leftarrow 0$;
 5: $BRAMs \leftarrow 0$;
 6: for $i \leftarrow 1$ to $nA$ do
 7:    if $T == cyclic$ or $T == block$ then
 8:       $BRAM_b \leftarrow (S_A * Bits)/(pf * S_{BRAM})$;
 9:       $R \leftarrow Round(BRAM_b)$;
10:       $BRAMs \leftarrow pf * R$;
11:       if $BRAMs == power(2, x)$ then
12:          $BRAM_T \leftarrow BRAMs$;
13:       else
14:          $BRAM_T \leftarrow 2^{round[log_2(BRAMs)]}$;
15:       end if
16:    else
17:       $BRAM_T \leftarrow 0$;
18:    end if
19: end for
20: return $BRAM_T$;
```

5.2 LUT Estimation

To estimate the total number of the Lookup table (LUT) resources (LUT_T) of a given application, we sum three important parameters: LUT_m, LUT_{op} and LUT_{ex}, as presented in Equation 2. LUT_m (Equation 3) corresponds to the LUT consumed by the multiplexer resources. LUT_{op} (Equation 4) represents the number of LUT consumed by the computation operation nodes, while LUT_{ex} (Equation 5) represents the LUT resources used by any expressions such as multipliers, adders and comparators. These information are automatically generated from our proposed area estimation tool (Figure 2). α and β are two constants. B_K is the bound of the nested loop K, where B_K can be represented as $B_K = 2^e + c$.

$$LUT_T = LUT_{op} + LUT_m + LUT_{ex} \tag{2}$$

$$LUT_m = \alpha(N_{store} + N_{op}) + V(\sum_{K=1}^{K=n} L_K + S_l) + \beta N_{load} \tag{3}$$

Where $N_{op} = nFA + nFM + nFS + nFD$
$V = e + 1$; $\alpha = 32$; $\beta = 14$

$$LUT_{op} = nFA*C1+nFM*C2+nFD*C3+nFS*C4 \tag{4}$$

$C1$, $C2$, $C3$ and $C4$ are four constants, which represent the number of LUT resources required to perform respectively a single *FA*, *FM*, *FD* and *FS* operations.

$$LUT_{ex} = (S_l + \sum_{K=1}^{K=n} L_K)(V1 + V2 + V3) \quad (5)$$

Where $B_K = 2^e + c$; $V1 = e + 1$
$V2 = e * 2$; $V3 = e + 2$

5.3 FF Estimation

Quantifying the necessary number of Flip-Flops (FF) depends on various parameters such as the type of an operation (load/store, computation, etc.), the total number of operations required to execute an application, the number of the loop levels, etc. Each operation is represented by a node in the DDDG graph, as illustrated in Figure 2. Based on an empirical study, the total number of FF resources (FF_T) can be estimated using Equation 6.

$$FF_T = FF_r + FF_{op} \quad (6)$$

FF_{op} (Equation 7) is the FF consumed by the arithmetic operations extracted from the DDDG graph, where $C5$, $C6$, $C7$ and $C8$ are constants.

$$FF_{op} = nFA * C5 + nFM * C6 + nFD * C7 + nFS * C8 \quad (7)$$

FF_r (Equation 8) represents the FF consumed by the register resources used to map an application to an FPGA. It also includes the FF consumed by the memory load/store instructions.

$$FF_r = \alpha(N_m + N_{op}) + IL + (2V \sum_{K=1}^{K=n} L_K) + S_l V \quad (8)$$

Where $N_{op} = nFA + nFS + nFD + nFM$
$N_m = N_{load} + N_{store}$; $B_K = 2^e + c$
$V = e + 1$; $\alpha = 32$

5.4 DSP Estimation

The DSP resource consumption depends on the total amount of compute operations (*subtraction*, *multiplication*, *addition* and *division*) required to execute an application. To estimate the total DSP resources of a specific SoC design, we measure the computation operation nodes within the generated DDDG graph. $C9$, $C10$ and $C11$ are three constants, which represent the number of DSP resources required to perform respectively a single *FA*, *FM* and *FS* operations. The total DSP resources (DSP_T) required to implement an application on an FPGA is estimated using Equation 9.

$$DSP_T = nFA * C9 + nFM * C10 + nFS * C11 \quad (9)$$

In this paper, we estimate the area occupation for hybrid CPU-FPGA architectures under the FPGA resource constraints. The Area Efficiency, denoted AE, is defined using Equation 10, where $BRAM$, DSP, FF and LUT represent the available BRAM, DSP, FF and LUT resources of a given FPGA platform, while $bram$, dsp, ff and lut are FPGA resources consumed by the current implementation.

$$AE = max(\frac{bram}{BRAM}, \frac{dsp}{DSP}, \frac{ff}{FF}, \frac{lut}{LUT}) \quad (10)$$

A given HA implementation can be fit into the FPGA if and only if $AE \leq 1$. Consequently, generated area results are equal to the FPGA resources required by the current configuration. Otherwise, the design exceeds the FPGA resource capacity, and the corresponding configuration is automatically rejected from the design space.

6. EXPERIMENTAL RESULTS

The main goal of our proposed tool is to efficiently explore the design space and provide an accurate area estimation for hybrid CPU-FPGA architectures. We estimate the required FPGA resources (BRAM, DSP, LUT, FF) through an analytic model. In order to validate our approach, we have implemented a collection of benchmarks from Polybench suite [15] and CHStone [16]. The different benchmarks have been mapped on several HA architectures, running at 100 MHz, over a wide range of input data sizes. We use Xilinx Vivado HLS version 2014.4 and Xilinx ZC702 Evaluation Kit [7] to validate our estimations.

6.1 Rapid Estimation

In this paper, we describe a fast high-level area estimation tool exploring different HAs with respect to FPGA resource constraints. Our proposed analytic model helps designers to obtain an accurate area estimation in an early design stage without generating RTL implementations. Table 1 shows the DSE time of the proposed area estimation tool (Figure 2). The results have been compared to Vivado HLS tool, for the same design space. Table 1 lists three selected benchmarks (MM, BICG and CONV3D) in column 1, while column 2 shows the input data size used for each one. Column 3 presents the total number of explored configurations for each benchmark varying pipeline options, array partitioning factors and types. As an example, for MM benchmark, we explored 119 design points with different pragma combinations in just 10 seconds. Our proposed tool is based on accurate modeling techniques and optimization pragmas, and can provide considerable fast area estimations for many designs compared to Vivado HLS tool.

Table 1: Estimated FPGA resources for MM, BICG and CONV3D: Vivado HLS tool vs. proposed tool

Application	**Input Size**	**Design Space**	**DSE Time**	
			Vivado HLS	Proposed Tool
MM	128*128	119	18 hours	10 seconds
BICG	256*256	133	1 day	17.50 seconds
CONV3D	32*32*32	120	22 hours	11 seconds

An example of MM design space is shown in Figure 3. The X-Axis shows some selected MM design configurations. The Y-Axis denotes the DSE time of each configuration in milliseconds (ms).

Each configuration, denoted (di_pj_ak), is expressed as follows:

- di: dataflow, which is disabled when $i = 0$; otherwise it is enabled.
- pj: pipeline level, which is disabled when $j = 0$; otherwise, j represents the pipeline level indicating that

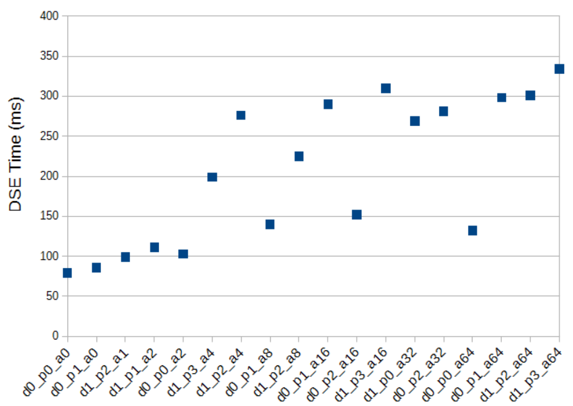

Figure 3: DSE time results for the MM benchmark with different (dataflow_pipeline_arrayPartitioning) configurations

this pragma is applied to the loop level j of the nested loop.

- ak: array partitioning factor, which is disabled when $k = 0$; otherwise, k represents the number of factor.

The results confirm that our proposed tool can rapidly estimate the area occupation for a large DSE with different pragma combinations. In addition, it is worth to mention that the proposed area model, presented in Section 5, is validated on other computational platforms, such as Xilinx Virtex6, Virtex7, etc.

6.2 Estimation Accuracy

In order to evaluate the estimation accuracy of our area analytic model, we use the Vivado HLS tool and the Xilinx ZC702 Evaluation Kit [7] to generate the same HW configurations and compare the obtained results of the FPGA resource estimations, while exploring different optimization pragmas. Figure 4 presents the experimental results of our proposed area estimation tool and Vivado HLS compared to real HW implementations for the different benchmarks. We can note that our proposed tool results are very close to the ones obtained from the real HW implementations considering the optimization pragmas (dataflow, loop pipelining and array partitioning). For Vivado HLS, the estimation error of FF resources achieves about 10% in GEMM benchmark. The HLS technique cannot provide accurate estimations for the required FPGA resources (FF, LUTs) due to its static analysis and lack of memory information (number of load/store instructions, etc.). The static analysis causes false dependencies between the operation nodes within the control data flow graphs. Therefore, it introduces large inaccuracies in the estimated FPGA resources. In this work, both BRAM and DSP resources estimated by our proposed tool and Vivado HLS match the real implementation results very closely. They are accurately estimated **(0% estimation error)** in the different benchmarks.

As illustrated in Figure 4, the proposed model serves well in the different configurations and results in an average estimation error less than 5% compared to the real HW implementations. This high estimation accuracy of our proposed tool can be explained as follows: In contrast to Vivado HLS tool, our proposed tool provides fast and accurate estimations for different sizes of input data. It should also be noted that unlike the existing state of-the-art methods, our proposed analytic model is based on a dynamic analysis that encompasses the runtime information to obtain true dependences between the operation nodes and therefore accurately estimate the FPGA resources of the system. This also obviates the need for generating RTL implementations to improve the estimation accuracy, thereby resulting in a fast as well as a reliable DSE framework. The proposed tool explores rapidly various SoC configurations in the order of seconds to minutes, over a large design space using different pragma combinations, as illustrated in Table 1.

Figure 5 shows the estimated area occupation and the total execution time results in milliseconds (ms) in log scale of the three benchmarks: MM, GEMM and BICG with (W) and without (Wo) optimization pragmas. Here, we use the AXI4 stream protocol to communicate between the processor and the HA. The experimental results demonstrate that applying optimization pragmas presents a performance/area occupation trade-off. As shown in Figure 5, optimizing the application using pragmas, such as loop pipelining, array partitioning and dataflow, considerably reduces the total execution time compared to the results without (Wo) pragmas. However, more logic resources are required.

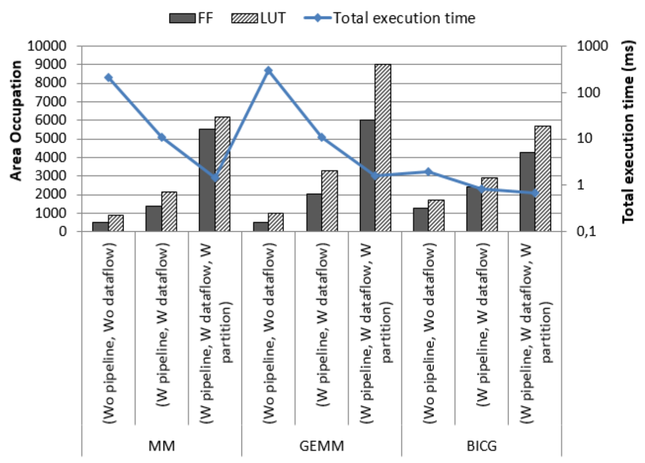

Figure 5: Estimated area occupation and total execution time of MM, GEMM and BICG benchmarks with different pragma combinations

We believe that our approach can be also applied to other FPGA devices such as Altera FPGA Platform. At a high level, Altera SDK for Open Computing Language (OpenCL) [17][18] translates an OpenCL kernel to a HW circuit that executes on the FPGA. Similar to Vivado HLS tool, recent Altera FPGA devices support optimization pragmas for different application domains, such as loop unrolling, pipelining, etc.

7. CONCLUSION

In this paper, we proposed a new area estimation tool based on a high-level analytic model. The tool has been used to quickly estimate the required FPGA resources (LUT, FF, DSP, BRAM) of different hybrid CPU-FPGA architectures

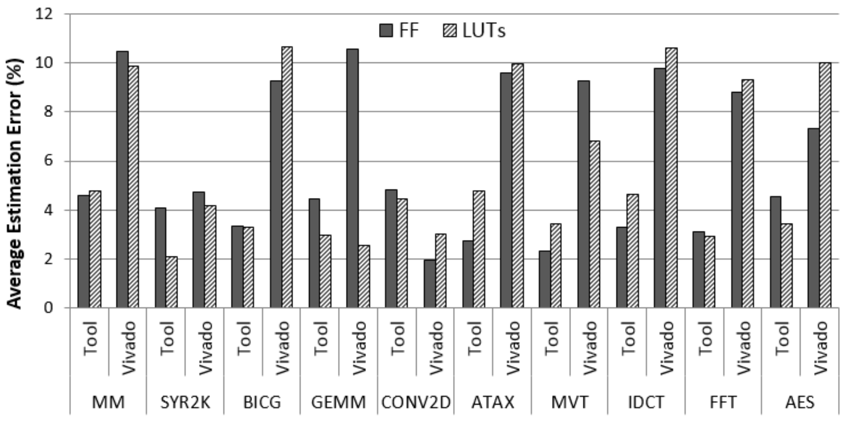

Figure 4: Our proposed tool and Vivado HLS estimation vs. real HW implementation with dataflow, loop pipelining and array partitioning

in an early design stage without generating RTL implementations. The comparison of our estimation results to RTL implementations confirmed both efficiency and accuracy of our proposed tool with an average area estimation error less than 5%. In addition, the proposed tool has the advantage of exploring and evaluating several HW configurations with different trade-offs through optimization pragmas. Dataflow, loop pipelining and array partitioning pragmas are those explored in the paper. In addition, our tool respects the FPGA resource constraints of the target platforms.

As extensions of this work, we project to test our tool on more real applications. We can also explore new pragmas in order to improve the performance of the system. In addition, we project to extend the tool by integrating energy consumption of various SoC designs with different trade-offs through optimization pragmas.

8. REFERENCES

[1] R. Jidin, D. Andrews, and D. Niehaus, "Implementing Multithreaded System Support for Hybrid FPGA/CPU Computational Components," in *International Conference on Engineering of Reconfigurable Systems and Algorithms*, pp. 116–122, June 2004.

[2] "Zynq Ultrascale+ MPSoC." http://www.xilinx.com/products/silicondevices/soc/zynqultrascale-mpsoc.html, 2016.

[3] A. Canis *et al.*, "LegUp: High-level Synthesis for FPGA-based Processor/Accelerator Systems," in *International symposium on Field programmable gate arrays (FPGA)*, 2011.

[4] D. Gonzalez *et al.*, "Coarse-grain performance estimator for heterogeneous parallel computing architectures like Zynq All-Programmable SoC," in *International Workshop on FPGAs for Software Programmers (FSP)*, 2015.

[5] H. Liu *et al.*, "On learning-based methods for design space exploration with high-level synthesis," in *Design Automation Conference (DAC)*, 2013.

[6] G. Zhong *et al.*, "Lin-analyzer: A high-level performance analysis tool for FPGA-based accelerators," in *Design Automation Conference (DAC)*, 2016.

[7] "Xilinx inc." http://www.xilinx.com, 2015.

[8] Xilinx, *Vivado Design Suite User Guide: High-Level Synthesis*, July 2012.

[9] N. Razvan *et al.*, "A Survey and evaluation of FPGA High-Level synthesis tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 35, pp. 1591 – 1604, December 2015.

[10] G. Zhong *et al.*, "Design space exploration of multiple loops on FPGAs using high level synthesis," in *IEEE International Conference on Computer Design (ICCD)*, October 2014.

[11] B. Holland *et al.*, "RAT: A methodology for predicting performance in application design migration to FPGAs," in *the 1st international workshop on High-performance reconfigurable computing technology and applications (HPRCTA)*, 2007.

[12] Y. S. Shao *et al.*, "Aladdin: A Pre-RTL, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *International Symposium on Computer Architecture (ISCA)*, 2014.

[13] A. Smith *et al.*, "Area estimation and optimisation of FPGA routing fabrics," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2009.

[14] ARM, "AMBA Specification (Rev 2.0)." http://www.arm.com/.

[15] L. Pouchet, "PolyBench/C 3.2.." http://web.cse.ohio-state.edu/pouchet/software/polybench/.

[16] "CHStone benchmark." http://www.ertl.jp/chstone/.

[17] "Altera sdk for opencl." https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/opencl-sdk/aocl_optimization_guide.pdf, 2015.

[18] D. Chen and D. Singh, "Fractal Video Compression in OpenCL: An Evaluation of CPUs, GPUs, and FPGAs as Acceleration Platforms," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2013.