EAI Endorsed Transactions
on Context-aware Systems and Applications
Research Article EAI.EU

Automatic FPGA-based Hardware Accelerator Design: A Case Study with Image Processing Applications

Cuong Pham-Quoc[1,2*]

[1]Ho Chi Minh City University of Technology (HCMUT), 268 Ly Thuong Kiet, District 10, Ho Chi Minh City, Vietnam
[2]Vietnam National University – Ho Chi Minh City, Thu Duc District, Ho Chi Minh City, Vietnam

Abstract

We present a case study of automatic FPGA-based hardware accelerator design using our proposed framework with the image processing domain. With the framework, the ultimate systems are optimized in both performance and energy consumption. Moreover, using the framework, designers can implement FPGA platforms without manually describing any hardware cores or the interconnect. The systems offer accelerations in execution time compared to traditional general purpose processors and accelerator systems designed manually. We use two applications in the image processing domain as experiments to report our work. Those are Canny edge detection and jpeg converter. The experiments are conducted in both embedded and high-performance computing platforms. Results show that we achieve overall speed-ups by up to 3.15× and 2.87× when compared to baseline systems in embedded and high-performance platforms, respectively. Our systems consume less energy than other FPGA-based systems by up to 66.5%.

Keywords: FPGA-based design framework, Hardware accelerator, Image processing

Received on 10 April 2020, accepted on 10 May 2020, published on 12 May 2020

doi: 10.4108/eai.12-5-2020.164497

* Corresponding author: cuongpham@hcmut.edu.vn

1. Introduction

In reccent years, FPGAs (Field programmable gate arrays) have been considering as a promising approach to overcome many obstacles in SoC designs such as time-to-market, power consumption as well as flexibility. However, FPGA designs still suffer from higher NRE (non-recurring engineering) cost than general pupose procesosrs or require knowledge of both hardware and software. These issues usally prevent the use of FPGA in real application domains such as image processing, voice recognization, artificial intelligent, or machine learning. Although a number of toochains have been proposed to pursuade designers to exploit FPGAs advantages like high-level synthesis tools, designers still need hardware knowledge and skills.

Our previous work presented in [1] proposed an automatic framework for designing and implementing FPGA-based hardware accelerator systems with an optimized hybrid interconnect to reduce data communication overhead. When using the framework, designers are helped to build FPGA accelerator systems without much hardware knowledge and skills. The proposed framework already solved a number of issues from which other similar toolchains suffer. For example, they could be used for only a dedicated application domain [2][3][4][5][6] or used for different applications without an interconnect optimized [7][8]. One of the most important contribution of our framework is to optimize the interconnect of hardware cores because data communication is one of the two main sources of overhead in multicore systems [9]. Each interconnection type, such as crossbar, bus or network-on-chip, offers different advantages while suffering various disadvantages [10]. Therefore, a hybrid interconnect consisting of multiple interconnect architectures is an appropriate approach for keeping performance of an FPGA-based hardware accelerator improved further.

In this paper, we report a case study for the image processing domain. We use the design framework to implement two different applications, the Canny edge detection and the jpeg image converter. We discuss design steps and conduct experiments with the systems in both embedded and high-performance computing platforms. We analyze experimental results of both the systems in terms of execution time and energy consumption. We then compare our system with general purpose processors and traditional FPGA-based accelerators.

The main contributions of this paper are summarized into two folds.

- We first briefly introduce a case study of the image processing domain with two applications when using our proposed framework;
- We analyze and compare results of the systems designed by the framework and other systems.

The rest of the paper is organized as follows. Section 2 quickly presents the framework designed and reported in our previous work. Section 3 discusses steps in developing two applications in the image processing domain with our framework. We present our experimental results with two computing platforms in Section 4. Finally, the paper contributions are concluded in Section 5.

2. The automatic design framework

In our previous work [1], we already proposed an automatic framework for designing an FPGA-based hardware accelerator with a hybrid interconnect. The framework allows designers to develop a hardware accelerator system for a particular application without much knowledge and working effort at the hardware level. Figure 1 depicts the design flow of the proposed framework. Although the design flow includes five automatic processing steps, designers are also able to interfere to manually make further improvements.

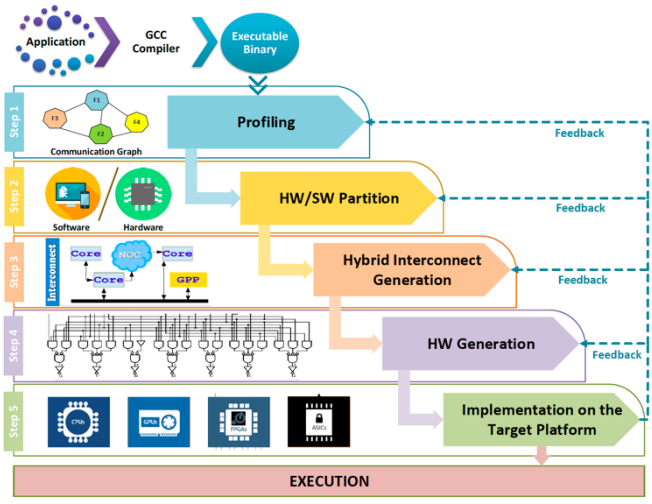

Figure 1. The proposed framework proposed in [1]

In this framework, a target application (developed in high level programming language) is profiled in Step 1 to collect execution time of functions in the application. This profiling step also creates a communication graph to represent data communication of functions inside the application. Based on this data communication graph, a hybrid interconnect for hardware kernels as well as between the hardware device and the host processor can be defined appropriately.

The application is then partitioned into hardware and software parts in Step 2. Computationally intensive functions should be candidates for accelerating with hardware kernels in FPGA. Moreover, based on the data communication graph, non-intensive functions may also be implemented in FPGA for reducing off-chip data communication.

As stated above, the data communication graph is used to define the most suitable hybrid interconnect for hardware kernels in Step 3. The hybrid interconnect may comprise bus, crossbar, network-on-chip, or shared buffer depended on the communication patterns of the kernels. The main purpose of this step is to reduce data communication overhead while keeping hardware resources usage for the interconnect minimized. That, in turn, will save energy consumption of the system since the less resources are used, the more energy is saved.

The selected functions are then synthesized by high-level synthesis (HLS) toolchains (in this case we use the Vivado HLS from Xilinx) to create hardware kernels described by hardware description languages (in our work, we prefer Verilog-HDL). With the support of HLS tools, functions written in high level programming languages can be compiled to Verilog-HDL automatically. This solves one of the most difficult issues that designers usually have.

Finally, the entire system is developed, synthesized and mapped to hardware platforms by tools provided by FPGA manufacturers (in our case, we use Xilinx Vivado since Xilinx platforms are targeted). Based on the resources availability of the target platforms, computationally intensive kernels can be replicated to further improve overall performance.

As summarized above, designers do not need to interfere with the framework much because all steps can process automatically. However, in case designers would involve themselves to modify some kernels or the interconnect, they are still able to do.

3. Case study

In this section, we introduce our hardware accelerator systems developed for the two applications, Canny edge detection [11] and jpeg image converter [12] using the framework.

3.1. The Canny edge detection

The main purpose of this application is to detect edges of images by applying a number of operators like Gaussian

filter, gradient calculation, non-maximize suppression, and finally hysteresis thresholding. The application was first introduced in 1986 and coded by ANSI C. The profiling results indicate that three operators Gaussian filter, gradient calculation and non-maximize suppression are the most computationally intensive. The framework also generates the data communication graph shown in Figure 2.

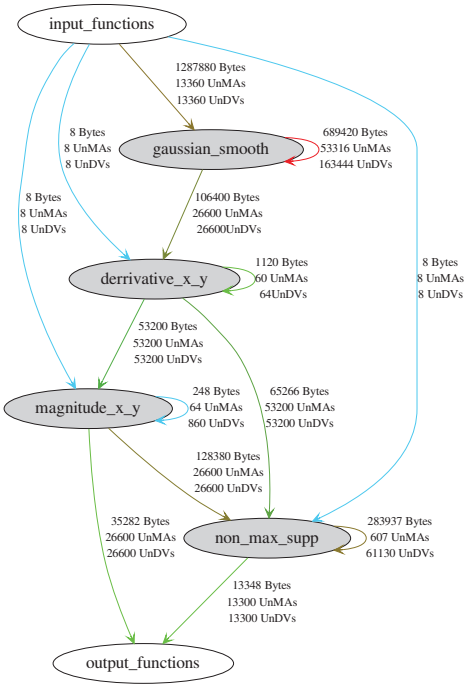

Figure 2. Data communication graph of the Canny edge detection application

According to the profiling results, three aforementioned operators (implemented as functions *gaussian_smooth*, *magnitude_x_y*, *non_max_supp*) are good candidates to be accelerated by hardware kernels. However, as illustrated in the data communication graph, the *derivative_x_y* function should also be accelerated by hardware kernels because there is a huge amount of data transferred between this function and other candidates. Therefore, all the four functions are implemented as accelerators. Other procedures of the application are kept processing on the general-purpose processor.

During the hybrid interconnect generation step, the most suitable interconnect is designed for data communication among the four hardware kernels above. Please note that, the communication infrastructure for transferring data between the processor and kernels is already defined by the target platform used for building the system. In this application, a shared buffer is used for the interconnect of the *gaussian_smooth* and *derivative_x_y* accelerators while a network-on-chip (NoC) is used for transferring data among the *derivative_x_y* and the rest two kernels.

As discussed above, in this work we target Xilinx platforms for implementing our systems; we therefore use Xilinx Vivado HLS to generate Verilog-HDL description modules for our hardware kernels. Figure 3 shows a part of the Verilog-HDL description generated by Xilinx Vivado HLS for the *gaussian_smooth* function.

```
1  // ==============================================================
2  // RTL generated by Vivado(TM) HLS - High-Level Synthesis from C, C++ and SystemC
3  // Version: 2012.3
4  // Copyright (C) 2012 Xilinx Inc. All rights reserved.
5  //
6  // ==============================================================
7
8  `timescale 1 ns / 1 ps
9
10 (* CORE_GENERATION_INFO="GSKernel_inst,GSKernel,{component_name=GSKernel_inst,
11 HLS_INPUT_TYPE=c,HLS_INPUT_FLOAT=1,HLS_INPUT_FIXED=0,HLS_INPUT_PART=xc6vlx760ff1760-1,
12 HLS_INPUT_CLOCK=10.000000,HLS_INPUT_ARCH=others,HLS_SYN_CLOCK=8.010000,
13 HLS_SYN_LAT=-1,HLS_SYN_TPT=none,HLS_SYN_MEM=0,HLS_SYN_DSP=58,HLS_SYN_FF=9875,HLS_SYN_LUT=12589}" *)
14
15 module GSKernel (
16         ap_clk,
```

Figure 3. Part of the gaussian kernel generated by Vivado HLS

In this work, we implement applications on both embedded system with the Xilinx ML510 board [13] and high-performance computing system with Micron HC-2ex [14]. Therefore, the final step needs to be performed for two different target platforms. For the embedded platform, because there exists only one FPGA device, we are able to build the system with 5 kernels. Hence, only the *gaussian_smooth* kernel is duplicated. Meanwhile, the high-performance computing platform consists of 4 modern FPGA devices that can host more kernels. Consequently, we replicate the kernels up to 64 accelerators in total.

The system finally is synthesized and mapped to FPGA devices by the Xilinx Vivado toolchain. This final step is technology dependent. Resources usage and other parameters are reported in detail in Section 4.

3.2. The jpeg image converter

The jpeg image converter is the second application used as our case study. The main purpose of this application is to encode bitmap images to the jpeg format. The application was implemented in ANSI C and reported in the benchmark in [12]. The main part of the application includes four computationally intensive functions. Those are *huff_dc_dec*, *huff_ac_dec*, *dquantz_lum*, and *j_rev_dct*.

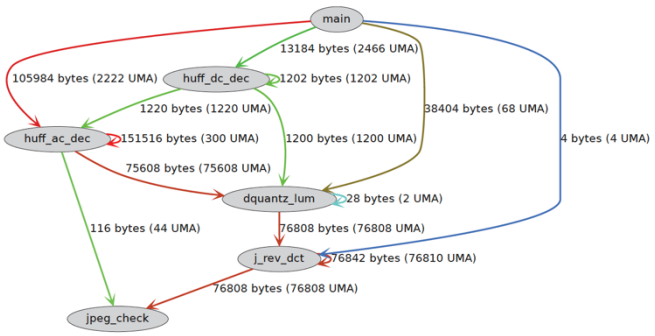

Figure 4. Data communication graph of the jpeg image converter application

Figure 4 illustrates the data communication graph mainly focusing on the four mentioned functions. All those functions are accelerated by hardware kernels. Similar to the Canny edge detection application, a NoC should be used for data communication of the three first kernels while a shared local buffer involves in transferring data between the *dquantz_lum* and *j_rev_dct* accelerators. Similar to the systems for the Canny edge detection application, Vivado HLS is also used for generating Verilog-HDL descriptions for these functions from the C code.

This application is also implemented in both the above embedded and high-performance computing systems. With the embedded system, the *huff_ac_dec* kernel is duplicated only while there are 64 kernels built in the high-performance computing system. Figure 5 illustrates the architecture of the jpeg application when implemented on the embedded platform.

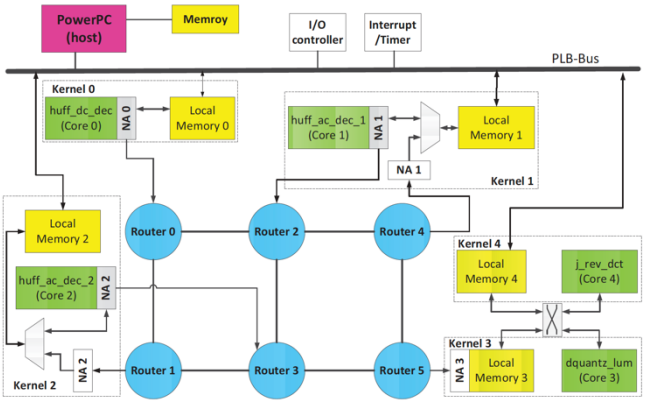

Figure 5. The architecture for jpeg application in the embedded platform

4. Experiments

In this section, we present our experiments to verify the FPGA-based accelerator systems reported above. Kernels performance as well as overall system performance are presented. Energy consumption compared to baseline systems is shown also.

4.1 Experimental setup

As presented above, the embedded and high-performance computing systems used as our target platforms are ML510 and Micro HC-2ex, respectively. The ML510 board consists of only one Xilinx xc5vfx130t FPGA device. Inside the FPGA device, there exist two embedded hardwired PowerPC processors that are used as host processors to process the software part of applications. Hardware accelerators are mapped into the reconfigurable area of the device. In this paper, we configure the processors functioning at 400 MHz while hardware kernels can work at 100 MHz only due to huge amount of reconfigurable logic resources used. To compare performance, the applications at first are executed on the host processor. They then are processed by the entire accelerator systems, i.e., hardware kernels process computationally intensive functions while software part still is kept executing on the host.

For the high-performance computing system, Micron HC-2ex (formerly Convey) is used as our experiment platform. The system includes of four Virtex-6 xc6vlx760 FPGA devices and one Intel Xeon X5670 processor. While the host processor can function at 2.93 GHz, the accelerators are set to work at 200 MHz. Similar to the embedded system, we first execute the applications on the host processor with full parallelization, i.e., the whole 12 cores of the processor are used to process the applications. They then are processed by the entire accelerator systems to compare execution time.

4.2 Experimental results

In this section, we discuss our experimental results with the two aforementioned systems when processing the two applications. Performance of kernels and entire systems as well as energy consumption and resources usage are analyzed.

Performance analysis

Table 1 depicts the kernels and the overall accelerator systems speed-ups when compared to their host processors (PowerPC for the embedded system and Intel Xeon for the high-performance computing system) in the third and the fourth columns, respectively. The table also presents speed-ups compared to baseline systems (the baseline systems are the traditional accelerator systems without helps of our framework for optimizing data communication but including replicated kernels for a fair comparison) in the fifth and the sixth columns, respectively.

Table 1. Speed-ups comparison between our systems and others

Platform	App.	w.r.t host processors		w.r.t baseline systems	
		kernels	overall	kernels	overall
EMB(1)	Canny	3.88×	3.15×	2.12×	1.83×
	jpeg	2.55×	2.33×	3.08×	2.87×
HPC(2)	Canny	2.62×	2.61×	1.55×	1.54×
	jpeg	1.96×	1.45×	1.93×	1.42×

(1) EMB: Embedded system; (2) High-performance computing system

As shown in the table, compared to both the host processors and the baseline systems, our systems designed with framework achieve better performance in term of execution time; in other words our systems outperform the others in term of execution time. Our systems process the applications up to 3.15× faster than general-purpose processors and up to 2.87× faster than baseline systems.

Table 2. Hardwar resources usage for the systems

Platform	Applications	Type	Baseline	Ours	Energy saved
Embedded	Canny	LUT	9,296	15,227	49.7%
		Reg	12,707	18,865	
	jpeg	LUT	11,755	20,837	66.5%
		Reg	11,910	20,900	
High-performance computing	Canny	LUT	74,965	90,789	54.3%
		Reg	48,994	54,849	
	jpeg	LUT	86,125	101,980	51.8%
		Reg	64,716	71,527	

Resources usage & energy consumption analysis

We use synthesis reports from the Xilinx tools and XPower analyzer also from Xilinx to extract hardware resources usage and power consumption of the systems. We compare our systems with the baseline systems in terms of resources utilization and energy consumption saved when using our systems instead of the baseline ones.

Table 2 presents the hardware resources utilization for one FPGA device in different systems with the two applications. We report the two most important values of resources including Look-up Tables (LUT) and Registers (Reg). Please note that, we only show resources directly used for the processing of our applications, i.e., we do not take resources used for managing the systems into account although they exist in the systems like debugger module or I/O handling modules. According to the table, our systems in both platforms need more resources than the baseline ones for all the experiments. The main reason for this issue is the hybrid interconnect used.

The table also presents energy reduction that our systems offer when compared to the baseline systems. Please note that the energy consumption is estimated by power consumption (reported by Xilinx XPower analyzer) and execution time. According to the table, although we need more resources than the baseline systems due to the hybrid interconnect; our systems use less energy than the baseline ones by up to 66.5%.

5. Conclusions

In this paper, we summarize the automatic FPGA-based hardware accelerator design framework proposed in our previous work. We then present in detail the accelerator systems for two applications belonging the image processing domain. The case study proves that the framework can help designers reduce the NRE cost and efforts in developing FPGA-based systems. Moreover, with the support of the framework, we achieve better overall performance compared to both the host processors only and the baseline systems. We also manage to save up to 66.5% of energy consumption when compared to the baseline systems. Although more resources are needed by our systems, more energy can be saved. Saving energy is one of the most critical issues for green computing in this era.

Acknowledgements.

This research is funded by Ho Chi Minh City University of Technology - VNU-HCM under grant number **To-KHMT-2018-03**.

References

[1] C. Pham-Quoc, “Design Framework for FPGA-based Hardware Accelerator with Hybrid Interconnect,” in proceedings of 2019 6th NAFOSTED Conference on Information and Computer Science (NICS), December 2019, Hanoi, Vietnams

[2] C. Pham-Quoc, B. Kieu-Do, and T. Ngoc Thinh, “An fpga-based seed extension ip core for bwa-mem dna alignment,” in 2018 International Conference on Advanced Computing and Applications (ACOMP), Nov 2018, pp. 1–6.

[3] C. Pham-Quoc, B. Tran-Thanh, and T. N. Thinh, “A scalable fpga-based floating-point gaussian filtering architecture,” in 2017 International Conference on Advanced Computing and Applications (ACOMP), Nov 2017, pp. 111–116.

[4] L. Song, J. Mao, Y. Zhuo, X. Qian, H. Li, and Y. Chen, “Hypar: Towards hybrid parallelism for deep learning accelerator array,” in 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), Feb 2019, pp. 56–68.

[5] M. Torabzadehkashi, S. Rezaei, A. Heydarigorji, H. Bobarshad, V. Alves, and N. Bagherzadeh, “Catalina: In-storage processing acceleration for scalable big data analytics,” in 2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Feb 2019, pp. 430–437.

[6] C. Pham-Quoc, B. Nguyen, and T. N. Thinh, “Fpga-based multicore architecture for integrating multiple ddos defense mechanisms,” SIGARCH Comput. Archit. News, vol. 44, no. 4, pp. 14–19, Jan. 2017.

[7] D. Pnevmatikatos, K. Papadimitriou, T. Becker, P. Baehm, A. Brokalakis, K. Bruneel, C. Ciobanu, T. Davidson, G. Gaydadjiev, K. Heyse, W. Luk, X. Niu, I. Papaefstathiou, D. Pau, O. Pell, C. Pilato, M. Santambrogio, D. Sciuto, D. Stroobandt, T. Todman, and E. Vansteenkiste, “Faster: Facilitating analysis and synthesis technologies for effective reconfiguration,” Microprocessors and Microsystems, vol. 39, no. 4, pp. 321 – 338, 2015.

[8] D. Glick, J. Grigg, B. Nelson, and M. Wirthlin, “Maverick: A standalone cad flow for xilinx 7-series fpgas,” in Proceedings of the 2019 ACM/SIGDA International Symposium on Field-

Programmable Gate Arrays, ser. FPGA '19. New York, NY, USA: ACM, 2019, pp. 306–307.
[9] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," ACM Trans. Archit. Code Optim., vol. 7, no. 1, pp. 4:1–4:28, May 2010.
[10] C. Pham-Quoc, Z. Al-Ars, and K. Bertels, "Heterogeneous hardware accelerators interconnect: An overview," in 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013), June 2013, pp. 189–197.
[11] J. Canny, "A computational approach to edge detection," Pattern Analysis and Machine Intelligence, pp. 679 –698, 1986.
[12] J. Scott, L. H. Lee, J. Arends, and B. Moyer, "Designing the lowpower M_CORE architecture," in IEEE Power Driven Microarchitecture Workshop, 1998.
[13] Xilinx, "Ml510 reference design," 2009
[14] Micron, "Hybrid core computer," 2012