EXPLOITING PARTIALLY RECONFIGURABLE FPGA FOR PERFORMANCE ADJUSTMENT IN THE RVC FRAMEWORK

Manel Hentati, Yassine Aoudni, Jean François Nezan, Mohamed Abid

EXPLOITING PARTIALLY RECONFIGURABLE FPGA FOR PERFORMANCE ADJUSTMENT IN THE RVC FRAMEWORK

Manel Hentati[1,2] *, Yassine Aoudni* [2] *, Jean-François Nezan*[1] *and Mohamed Abid* [2]

[1] INSA/IETR, 20, av des Buttes de coesmes CS 14315 F-35043 RENNES, France
[2] ENIS/CES, Route Soukra B.P. 1173, 3038 Sfax, Tunisie

ABSTRACT

In this paper, we present a method to implement a specific algorithm using the RVC framework and the dynamic partial reconfiguration (DPR). The DPR is a technique allowing to replace modules in a design at run-time. While, the RVC framework is based on the use a specific language for writing dataflow models called RVC-CAL. The studied algorithm is a Hadamard transform. Several dataflow models of the Hadamard transform can be used in the design process in order to favor speed or power consumption. We show the steps required to implement and switch between two dataflow models (a sequential model and a pipelined model) of the Hadamard transform. Our design allows to user to choose one of two architectures according her requirements of low power and high speed.

Index Terms— Dynamic Partial Reconfiguration, RVC framework, Hadamard transform, FPGA, performance.

1. INTRODUCTION

The last generation of FPGA from Xilinx adds a new feature which is dynamic partial reconfiguration (DPR). DPR is the ability to change some processes of an FPGA device while other processes continue in the rest of the device [5]. To apply the DPR functionality, the designer requires a hardware description of the process using an HDL language like VHDL or VERILOG. But, this description is very hard to achieve due to the increasing the complexity of signal processing algorithms. As solution, designer can describe the application process at a higher level and use tools to generate automatically the hardware description. In our work, we propose to use RVC-CAL language which is a textual and a domain specific language for writing dataflow applications [3]. It is normalized by the MPEG community through the MPEG-RVC standard [7]. This standard provides a framework to define different codes by combining communicating blocks developed in RVC-CAL. This language is supported by several tools including OpenDF [4] for the simulation and CAL2HDL [6] [**?**], ORCC [2] for the automatic code generation (VERILOG, VHDL ...).

The main contribution of this paper is to propose a method which allows adjusting performances according the system exigences (power, speed). Our approach consist of applying DPR between two modules which have different architectures (pipelined and sequential). Using the proposed method, the designer can implement one of two architectures depending on her requirement of low power or high speed. As we know sequential architecture allows saving power and pipelined architecture increases the speed. The proposed design allows using multiple version of the same algorithm. If designer need to reconfigure a full decoder, he should adjust the new implementation taking into account several parameters (latency / speed / power consumption) in the chosen algorithms. However, this requires to store bitstreams in the memory of the FPGA or to recover bitstreams on the network before reconfiguration. This paper is organized as follows: Section 2 presents Hadamard transform. In the Section 3, we expose the needed steps for the implementation of a reconfigurable system in RVC framework. The experimental results are reported in Section 4. Section 5 outlines the conclusion with planned future work.

2. HADAMARD TRANSFORM

The Hadamard transform is used in many image and video coders notably the LAR (Locally Adaptive Resolution) [1]. The Hadamard transform derives from a generalized class of the Fourier transform. It consists of a multiplication of a 2^mx2^m matrix by an Hadamard matrix (H_m) that has the same size. Here are examples of Hadamard matrices.

H_0 is the identity matrix so $H_0 = 1$.

For any $m > 0$, H_m is then deducted recursively by (1).

$$H_m = \frac{1}{\sqrt{2}} \begin{bmatrix} H_{m-1} & H_{m-1} \\ H_{m-1} & -H_{m-1} \end{bmatrix} \quad (1)$$

In our work, we develop the Hadamard transform (H2) with sequential and pipelined architectures. After that, we apply DPR between these two architectures. We note Hadamard-seq for sequential architecture and Hadamard-pip for pipelined architecture. Both Hadamard-seq and Hadamard-pip architectures have 16 inputs and 16 outputs. But, Hadamard-seq is composed of one actor which includes

17 actions and a finite state machine to ensure the sequential execution of actions. The first action may read inputs and the other actions allow computing outputs. Each computed action comprises a set of arithmetical operations (subtraction, addition) to produce one output. During the execution of Hadamard-seq, we always have one active action for each state. In the first state, we read the inputs values and save its. Then, we produce outputs sequentially in the others states. Figure 1 shows the sequential execution for computed actions.

The Hadamard-pip architecture is composed of 12 actors.

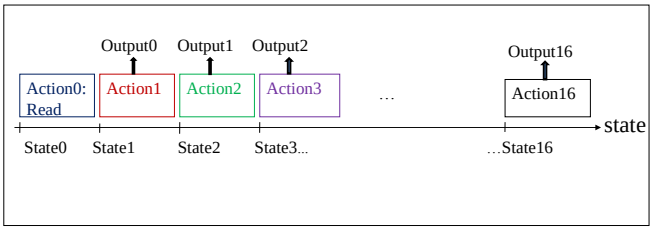

Fig. 1. Sequential architecture for 4x4 Hadamard transform

The first four actors are running in parallel to calculate intermediate results. After that, the second four actors are executed. Finally, the last four actors are running to produce all the outputs. Figure 2 illustrates the pipelined architecture.

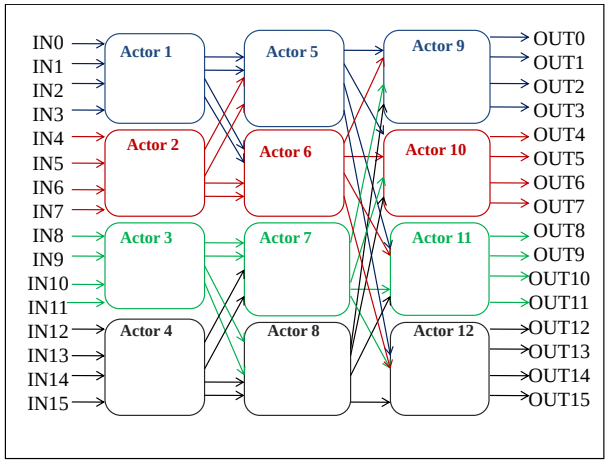

Fig. 2. Pipelined architecture for 4x4 Hadamard transform

3. HARDWARE IMPLEMENTATION OF THE HADAMARD TRANSFORM

Applying the partial reconfiguration flow using Xilinx FPGAs requires design tools and methodologies that exploit the partial reconfiguration capabilities. In our approach, we use this set of Xilinx tools version 12.3: synthesis tool Xilinx ISE (Integrated Synthesis Environment), PlanAhead for placement /routing and EDK (embedded development kit) helps to build an on-chip system includes the MicroBlaze processor [8] and internal/external peripherals. Basically, the adopted architecture is built around a processor core (Microblaze) which manages the implementation of reconfigurable modules. An UART provides serial communication between PC and FPGA. A timer to measure the reconfiguration time. A flash memory to store bitstreams partially reconfigurable. A HWICAP to download partial reconfigurable bitstream from flash memory to the FPGA. This component is delivered by Xilinx and it is operating at 100 MHZ and has 32-bits data port. All the peripherals are connected on the bus PLB. Figure 3 shows our adopted architecture.

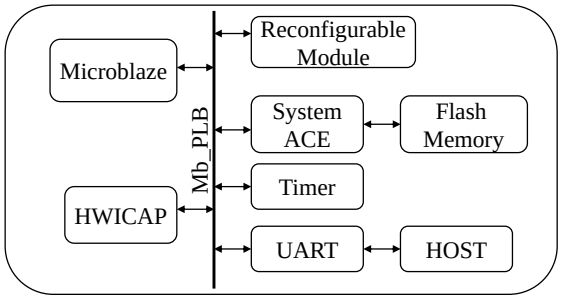

Fig. 3. Adopted reconfigurable architecture

4. RESULTS

Our design is implemented in the Xilinx Virtex-5 XC5VLX110T-1ff1136. The board includes 69120 registers, 17920 Blocks RAMs, 69120 LUTs and 64 DSP48ES. Table 1 shows the synthesis results of reconfigurable modules in terms of chip area, maximum frequency and minimum period.

By making a comparison between these two modules in terms of hardware resources requirement and maximum frequency, we note Hadamard-pip uses more logic cells than Hadamard-seq and it has the highest maximum frequency, because the pipelined architecture needs more resources than

Table 1. Synthesis results

	Hadamard-pip	Hadamard-seq
Slices	739	702
LUTs	897	479
Registers	674	181
Maximum Frequency(MHZ)	241,433	91,384
Minimum Period (ns)	4,182	10,94

the sequential architecture. However, it is faster than the sequential architecture.

Table 2 gives power consumption of the two reconfigurable modules. The results are obtained via XPower which is the power estimator tool of Xilinx. The approach used by this tool consists of providing information including the number of LUTs, the number of registers and the clock frequency to determine the power consumed by the FPGA for a given temperature. The XPower tool is based on an Excel spreadsheet. It receives input values about the types and number of used FPGA resources such as LUTs, DCM, BRAM, DSP, etc. Also, the designer can set some values and types of used resources such that temperature, frequency and FPGA type, etc. We note total power consumption for Hadamard-pip

Table 2. Power consumption results

	Total power (mw)	Dynamic power (mw)	leakage power (mw)
Hadamard-seq	934	91	843
Hadamard-pip	959	115	844

is greater than total power consumption for Hadamard-seq, because Hadamard-pip uses more logic cells than Hadamard-seq. So, we conclude that the saving area on chip indicates directly the optimized power consumption.

According these results, we are able to confirm the DPR technique is a promising solution to decrease total power consumption.

The table 3 presents the reconfiguration time. The practical value is obtained by a Xilinx IP named xps-timer and the theoretical value is computed using the equation (2):

$$t = \frac{bitstream * 8}{32 * 100} \quad (2)$$

The number 100 presents the ICAP frequency in MHZ and the number 32 is the ICAP bandwidth of the data port.

The bitstreams size of both Hadamard-pip and Hadamard-seq is equal to 192512 bytes (188 ko).

The difference between the theoretical and practical value is

Table 3. Reconfiguration Time

	Practical value (μs)	Theoretical value (μs)
Hadamard-pip	483,73	481,28
Hadamard-seq	486,82	481,28

the time that puts the processor to send to HWICAP a request to load a configuration. This time depend on size of partial bitstreams and the performance of used HWICAP.

5. CONCLUSION

This paper presented a design approach based on DPR functionality to implement two Hadamard modules. These two modules are described with two different architectures: pipelined and sequential using RVC-CAL dataflow language. The main advantage of this paper appears in the ability of proposed approach to adjust the system according the constraints design : speed and power. Indeed, our method allows switching between sequential architecture to save power and pipelined architecture to provide high speed. In this work, we focused on the Hadamard transform. This is a simple case study, but, the concept can easily be extended to provide more useful and complex RVC applications. We measured the reconfiguration time theoretically and practically using a Xilinx IP (xps-timer). We note that, this time depends on HWICAP used and bitstreams size.

In future work, to continue the use DPR in the RVC technology by applying this technique between two different RVC decoders. These two decoders will be placed in same reconfigurable partition regions in FPGA. We plan also to improve the tools which automatically implement the decoder by integrating DPR functionality.

6. REFERENCES

[1] O. Deforges, M. Babel, L. Bedat, and J. Ronsin *Color LAR Codec: A Color Image Representation and Compression Scheme Based on Local Resolution Adjustment and Self-Extracting Region Representation* , IEEE Trans. Circuits Syst. Video Techn., vol. 17, no. 8, pp. 974-987, 2007.

[2] IJ. W. Janneck, M. Mattavelli, M. Raulet, and M. Wipliez, *Reconfigurable video coding: a stream programming approach to the specification of new video coding standards*,in MMSys 10: Proceedings of the first annual ACM SIGMM conference on Multimedia systems. New York, USA: ACM, pp. 223234, 2010..

[3] ISO/IEC FDIS 23001-4: 2009, *Information Technology - MPEG systems technologies - Part 4: Codec Configuration Representation*, 2009.

[4] S. Bhattacharyya, G. Brebner, J. Eker, J. Janneck, M. Mattavelli, C. von Platen, and M. Raulet, *OpenDF - A Dataflow Toolset for Reconfigurable Hardware and Multicore Systems*, First Swedish Workshop on Multi-Core Computing, MCC , Ronneby, Sweden, November 27-28, 2008.

[5] P. Lysaght, B. Blodget, J. Mason, B. Bridgford, and J. Young, *Enhanced Architectures, Design Methodologies and CAD Tools for dynamic reconfiguration of XILINX FPGAs*,in 16th Int. Conf. on Field Programmable Logic and Applications (FPL2006), pp. 12-17, 2006 .

[6] Cal2HDL-openforge source" Available from:: http://openforge.sourceforge.net. [Accessed: December 2010]

[7] Ihab Amer, Christophe Lucarz, Ghislain Roquier,Marco Mattavelli, Mickal Raulet, Jean-François Nezan, and Olivier Deforges *Reconfigurable Video Coding on Multicore*, IEEE Signal Processing magazine, 2009.

[8] MicroBlaze Processor Reference Guide Embedded Development Kit EDK 10.1i, UG081 (v9.0), 2008.