Evaluating FPGA Virtex-II Board using Dynamic Partial Reconfiguration

Imran Hashmi
M.Sc Student

Department of Electrical
Engineering, UET Taxila, Pakistan.

Habibullah Jamal
Professor

Department of Electrical Engineering,
UET Taxila, Pakistan.

Tahir Muhammad
Lecturer

Department of Electrical Engineering,
UET Taxila, Pakistan.

Abstract

The Field Programmable Gate Array (FPGA) offer effective suppleness and performance because of reconfigurable hardware but consume more power in contrast to the Application Specific Integrated Circuit (ASIC). At run time reconfiguration of hardware in FPGAs can not only be very economical but can be real alternative for ASICs. The designers are reluctant to use Dynamic Partial Reconfiguration (DPR) in FPGA due to lack of adequate tools provided by the vendors. DPR has been in academic use for more over a decade. DPR offers reduction in power consumption, area, cost as well as increase in flexibility, efficiency and fault tolerance but has an application dependent overhead. In this work prior performance of DPR is evaluated using Xilinx Virtex II Pro in order to realize whether it is suitable for an application rather than at later complex design stages of a system design having the DPR employed. The evaluation is based on the reconfiguration speed and the resource utilization. The DPR shows an improvement of resource utilization by 22.5 % (in terms of slices) as well as speedup in comparison to Non-DPR design.

Keyword

Dynamic Partial Reconfiguration, Field Programmable Gate Array, Reconfigurable Architecture.

1 INTRODUCTION

Due to the flexibility and enhanced capacity of logic gates, Field Programmable Gate Arrays are becoming sustainable and alluring option for digital system design. They are used in different application domains as in bioinformatics and motion detector [1]. The high-end FPGAs e.g. Virtex II, Virtex 4, and Virtex 5 have multi millions gates of logic available for very large system implementations. For the implementation of reliable systems increasing attention has been given for new methodologies and techniques in the last decade. Applications requiring distance communication, fault relief, quick recovery and low power design are common nowadays e.g. space technology and telecommunication. Advantages offered by FPGAs such as programmability, flexibility, debugging, prototyping and parallel computation have provided an alternate for ASICs. The SRAM based FPGAs are configured by programming bits of SRAM being connected to the configuration points of a chip[1]. Run-time reconfiguration is used to dynamically re-design applications offering performance

improvements in multi traits such as speed, power, area, cost, and fault tolerance. High-end FPGAs have an appealing feature of DPR and an evolving interest is presented for DPR [2-4]. Due to lacking tools and support provided by the vendors, DPR is only in academia use for the past decade. Applications using DPR have shown power reduction, speed improvement, time-sharing of resources; economical cost and chip-area reduction reported in [5-10].

DPR can be used to reconfigure only a desire part of the reconfigurable portion of the device while the other part of the application continues to run. In reconfiguration process a bitstream is loaded to FPGA's configuration memory which holds reconfigurable data. This process overwrites the previous design using the newer reconfiguration data. The speed of reconfiguration is a key factor for the performance of time critical applications. The overall process demerits execution as time is required to download bitstream to configuration memory before execution. Typically reconfiguration time published in previous work correspond to only chip's configuration port [11]. The frameworks developed for performance evaluations is in order to leverage Partial Reconfiguration design flow at initial steps of the development [12]. For real system experimentation all the system components taking part in reconfiguration overhead have to be considered. When the assumptions are unrealistic the theoretical reconfiguration overhead can deviate in magnitudes [13-14]. A framework developed in [1] has shown the performance of DPR for various bitstream sizes using PLB-HWICAP. There are other possible reconfiguration design architectures of reconfiguration port ICAP such as DMA-HWICAP, BRAM-ICAP, PLB-HWICAP (Cache Enabled) which are evaluated in this work and have shown to improve the reconfiguration speed.

The contribution of this paper is as follow:

- ✓ Reconfiguration Time comparison for different design architectures of ICAP
- ✓ Maximum speed offered by ICAP design architectures
- ✓ Device Utilization for Power and Area concerns of chip

This paper is organized as follow. Section 2 provides a short view of related work on applications using DPR and several reconfiguration designs are discussed. Section 3 describes the experimental setup with sub-sections detailing on individual

processes showing the complete work presented in this paper. Section 4 shows and discusses the results obtained in the experiments with the concluding remarks in Section 5.

2 RELATED WORK

The previously reported evaluations of DPR of FPGA for power reduction and chip size (area) are in [7-8, 15-16]. Two cores of ICAP are available for reconfiguration. A low performance OPBHWICAP and high performance XPSHWICAP in Xilinx design. A number of reconfiguration tactics are used for speed comparison such as ICAP, JTAG, and selectMAP[7]. A good speed for reconfiguration has been shown by optimizing the PLB ICAP design but resource consumption is not presented [17-18]. A complete evaluation is presented in this work for design architectures of ICAP such as DMA-HWICAP, BRAM-HWICAP, PLB-HWICAP, and OPB-HWICAP.

process. Last section ICAP design provides explanation of design architectures of reconfiguration port ICAP.

3.1 Creating Partial Bitstream

There are two approaches supported by Xilinx for generation of partial bitstream i) Difference based ii) Module based [19]. In our design we have used difference based. The difference based design is for smaller designs while the module based is for larger and complex designs. Partial bitstream is created using Xilinx ISE 10.1 design tools and PlanAhead. The HDL design flow used is shown in Figure 1. The design is described using Verilog language in hierarchical manner. The Top-Level module contains the system design, Static Modules, and Partial Reconfigurable Module (PRM). All I/O of design, Bus Macros (BM), GCLK, DCMs, Buffers, Static and Partial Modules of design are instantiated as black box in Top-Level module.

Figure 1: HDL Design Flow

3 EXPERIMENTAL SETUP

The experiments were performed using XC2VP30 Xilinx Virtex II Pro with the aid of logic analyzer and PCs. The experimental setups are used to measure time components contributing to total reconfiguration time. The next section, Creating Partial Bitstream, discusses the flow of HDL Design and bitstream generation using Xilinx ISE 10.1. Later section, System Operational Flow emphasizes on execution of reconfiguration

After the synthesis of the design, description (.ngc) file is created which is used in place and routing for defining timing and area constraints. Non-DPR design is implemented for verification and resource utilization comparison. Timing and area constraints are verified for Non-DPR design before stepping in DPR complex design. The *static.used* file generated after implementing static module is later used when implementing PRM to avoid overlapping of areas. Then merging of static and partial modules is performed and bitstreams are generated for individual PRM.

3.2 System Operational Flow

The reconfiguration flow used is shown in Figure 2. The FPGA runs and executes the initial configured application and waits for the reconfiguration call. As soon as the reconfiguration event is detected the PowerPC reads the data from the memory, where initially the bitstream is placed. The data is transferred to PowerPC memory. In a single transaction the size of data transferred depends on the processor array. Then the data is transferred from processor to the configuration cache (BRAM) of the HWICAP. The data is written to FPGA's configuration memory using ICAP as soon as BRAM is full. The end of configuration is indicated by the use of pad frame. As soon as the configuration is completed the pipeline terminates. FPGA executes the application using the newly downloaded PRM.

3.3 ICAP Design

Two cores OPB-HWICAP and XPS-HWICAP for Xilinx designs are used mainly for the DPR. A bridge is required to interface low performance OPB with high performance PLB bus as in Figure 3. The data is stored in configuration cache (BRAM) using OPB. BRAM is dual port memory. ICAP is used to load data to FPGA's configuration memory. While using XPS-HWICAP the DP-BRAM is replaced by FIFO write/read and register's group shown in Figure 4. To enhance the transfer rate of data from processor to ICAP port XPS-HWICAP is used along with DMA. The use of DMA function, shown in figure 5 on XPS-HWICAP increases the efficiency of data transfer from embedded processor to ICAP.

Figure 2: Reconfiguration Process Flow

Figure 3: OPB-HWICAP Architecture

Figure 4: XPS-HWICAP Architecture

DMA has two interfaces one for the commands and other for the data transfer. Further reduction in communication overhead can be achieved using burst transmission support of PLB interface. Figure 6 shows the configuration efficiency of ICAP primitive evaluated by using dedicated BRAM for bit stream storage. BRAM must have size enough to hold the bitstream. The bitstreams are loaded in dedicated BRAM instead of memory. The data is available for reconfiguration which excludes required time for transfer of data of PRM from memory to HWICAP. PLB IP (Intellectual Property) is used to load data to BRAM. The highest speed for reconfiguration achieved is using BRAM-HWICAP design. This design architecture is well suited to small designs and those applications requiring fast switching of design modules. But there is a tradeoff for this achieved speed in term of extra utilization of resources. So power consumption and area of chip also increases with increase in resource utilization.

Figure 5: DMA-HWICAP Architecture

Figure 6: BRAM-HWICAP Architecture

4 RESULTS AND DISCUSSION

4.1 Reconfiguration Time Measurements

The experiments were carried out using PowerPC and MicroBlaze running at 300 MHz and 100 MHz respectively. The speed has been increased as the ICACHE and DCACHE of PowerPC are enabled. The reconfiguration time (RT) is proportional to size of bitstream. Bitstreams size 548 KB, 125 KB, and 99 KB were used. The size of processor array and of configuration cache does also have an impact on the configuration speed. Here the size of processor array and configuration cache is kept constant. So the linear increase in RT is observed. The rows 3, 4, 5 and 6 of Table 1 show that the speed is higher when using the PowerPC compared to MicroBlaze. Hardcore processor is much faster than the softcore processor. The DMA function on XPSICAP provides much increase in speed compared to the trivial design architectures of ICAP. Moreover, BRAM-HWICAP design architecture provides highest speed but with a penalty of physical resource utilization. Graphical comparison of speed and reconfiguration time is shown in Figure 7 and Figure 8.

4.2 Device Utilization

The device utilization has made it clear that the area of chip and resources requirement using DPR is reduced. From Table 2 the slice utilization for non DPR design is 22.5 % greater than that of DPR design. The utilization under title DPR contains the utilization of both static and partial modules. The LUTs and FFs utilizations are also less. But the BRAM utilization is very near to that of Non-DPR design. The reason is that the use of BRAM as memory to hold bitstream increase the speed greatly but the cost is paid in term of more resource utilization and so the area of chip is also increased.

Figure 7: Speed of ICAP Design Architectures

Table 1: Reconfiguration and Speed of Bitstreams

Sr.No	ICAP Design	Bitstream 1 Reconf. Time (ms)	Bitstream 2 Reconf. Time (ms)	Bitstream 3 Reconf. Time (ms)	Avg. Speed MB/Sec	Max. Speed MB/Sec
1	OPB-ICAP _(Cache_disabled)	344	73	56.5	1.67	1.75
2	PLB-ICAP _(Cache_disabled)	267	53	42.8	2.24	2.32
3	OPB-ICAP _(Cache_enabled)	128	30	19.9	4.52	4.98
4	PLB-ICAP _(Cache_enabled)	59.8	14.46	9.6	9.37	10.31
5	OPB-ICAP _(MicroBlaze)	170.6	43.2	28.8	3.2	3.5
6	PLB-ICAP _(MicroBlaze)	79.7	23.8	13.4	6.5	7.3
7	DMA-ICAP	4.8	0.8544	0.7654	129.78	143.96
8	BRAM-ICAP	2.3	0.4835	0.4283	242.6	254.4

Power consumption, cost and chip-area are proportional to the resource utilization. Graphical representation of Device Utilization is shown in Figure 9. The values are bit overlapped but they can be verified from the table.

Figure 8: Reconfiguration Time Vs Bitstream

Figure 9: Device Utilization

Table 2: Device Utilization

Sr.No	Resource	DPR Design Utilization (Static + Partial)	Non PR Utilization	Total Available Resource	Non PR %Utilization	DPR %Utili- zation
1	Slices	1410+2056=3466	6545	13696	47.8	25.3
2	LUTs	2045+3503=5548	10819	27392	39.5	20.3
3	FFs	1581+1003=2584	5015	27392	18.3	9.5
4	IOs	35+0=35	64	416	15.4	8.3
5	BRAMs	47+38=85	97	136	71.3	62.5

5 CONCLUSIONS

In this paper an evaluation of Virtex-II Pro using DPR is performed. This evaluation can help designers to decide whether the architecture is suitable for their desired application(s). It has been observed that speed of reconfiguration is proportional to size of the bitstream. BRAM-HWICAP offers highest speed but increases resource utilization. It is suitable for small and speedy application designs. A hardcore (PowerPC) processor also offers more speed than softcore processor (MicroBlaze). Power, area and cost reduction is possible with reduction in resource utilization. Overall, the resources utilized using DPR are less compared to that of Non-DPR.

6. REFERENCES

- [1] K. Papadimitriou, A. Dollas, and S. Hauk. 2010. An effective framework to evaluate dynamic partial reconfiguration in FPGA systems. *Instrumentation and Measurement, IEEE Transactions*.
- [2] Line P. L. D., 2008., Xilinx honored for enabling technology in the ALICE experiment at CERN. Xilinx Inc., San Jose, CA.
- [3] Compton K. and S. Hauck., 2002. Reconfigurable computing: a survey of systems and software. *ACM Computing Surveys*
- [4] Press Release, 2006. ISR and Xilinx Roll Out Ready-to-Wear SDR. Xilinx Inc. San Jose. [online] Available www.fpgajournal.com
- [5] Gholamipour A. H. Hamid E. Ahmed E. and Fadi K. 2009. Size-reconfiguration delay tradeoffs for a class of DSP blocks in multi-mode communication systems. In *Field Programmable Custom Computing Machines*.
- [6] Paulsson k. Hubner M. and Becker J. 2008. Cost-and power optimized FPGA based system integration: methodologies and integration of a low-power capacity-based measurement application on Xilinx FPGAs. In *Design, Automation and Test in Europe*.
- [7] McDonald E. J. 2008. Runtime FPGA partial reconfiguration. In *IEEE Aerospace Conference*.
- [8] Noguera J. and Kennedy I. O 2007. Power reduction in network equipment through adaptive partial reconfiguration. In *International Conference on Field Programmable Logic and Applications*.
- [9] Hanho L. 2007. A self-reconfigurable adaptive FIR filter system on partial reconfiguration platform. *IEICE transactions on information and systems*.
- [10] Kao C. 2005. Benefits of partial reconfiguration. *Xcell journal*.
- [11] Papadimitriou, K., Dollas A. and Hauck S. 2011 Performance of partial reconfiguration in FPGA systems: A survey and a cost model. *ACM Transactions on Reconfigurable Technology and Systems*.
- [12] Hsiung P. A., Lin C. S., and Liao C. F., 2008. Perfecto: A systemc-based design-space exploration framework for dynamically reconfigurable architectures. *ACM Transactions on Reconfigurable Technology and Systems*.
- [13] Griesse B., Erik V., Mario P. and Ulrich R., 2004. Hardware support for dynamic reconfiguration in reconfigurable SoC architectures. In *Field Programmable Logic and Application*.
- [14] Galindo J., Eric P., Brad L. and Gene R., 2008. Leveraging firmware in multichip systems to maximize FPGA resources: An application of self-partial reconfiguration. In *international conference on Reconfigurable Computing and FPGAs*.
- [15] Choi C. S. and Lee H., 2006. An reconfigurable FIR filter design on a partial reconfiguration platform. In *international conference on Communications and Electronics*.
- [16] Paulsson K., Michael H., Salih B. and Jurgen B., 2007. Exploitation of run-time partial reconfiguration for dynamic power management in Xilinx spartan III-based systems. *Reconfigurable system on chip Montpellier, France*.
- [17] Claus C., Florian H. M., Johannes Z. and Walter S., 2007. A new framework to accelerate Virtex-II Pro dynamic partial self-reconfiguration. In *IEEE Parallel and Distributed Processing Symposiu*.
- [18] Claus C., Zhang B., Stechele W., Braun L., Hubner M. and Becker J., 2008. A multi-platform controller allowing for maximum dynamic partial reconfiguration throughput. In *international conference on Field Programmable Logic and Applications*.
- [19] Xilinx.Inc. www.xilinx.com/support/documentation/xapp290.pdf [Online].