Advance in Electronic and Electric Engineering.
ISSN 2231-1297, Volume 3, Number 2 (2013), pp. 257-264

http://www.ripublication.com/aeee.htm

Dynamic Partial Reconfiguration in Low-Cost FPGAs

K. Bhuvaneswari[1] and V. Srinivasa Rao[2]

[1]ECE Dept, Shri Vishnu Engineering College for Women, Vishnupur, Bhimavaram, West Godavari District, Andhra Pradesh, INDIA.
[2]ECE Dept, Shri Vishnu Engineering College for Women, Vishnupur, Bhimavaram, West Godavari District, Andhra Pradesh, INDIA.

Abstract

Field Programmable Gate Array (FPGA) market is growing rapidly with various applications in different industries. There is a new concept evolving in FPGA industry called Dynamic Partial Reconfiguration (DPR) with has a greater exposure in different applications. Partial reconfiguration is nothing but reconfiguring the selected areas of an FPGA after its initial configuration at runtime. In this paper we reconfigure some specific region of the FPGA with a new functionality at runtime while the remaining areas remain static during this time. The complexities during the runtime can be simplified by a tool called PlanAhead which was introduced by Xilinx that is able to implement run time reconfigurable systems for all Virtex FPGAs. This results in low computational cost and low power FPGAs, PlanAhead is the first graphical environment for Partial Reconfiguration. In this context Partial Reconfiguration gives the flexibility for reducing the board space (effective utilization of resources), change a design in the field and also reduces the power consumption and delay.

Keywords: FPGA, Dynamic Partial Reconfiguration, PlanAhead.

1. Introduction

FPGAs provide users with an environment where the user is able to quickly develop and implement a circuit or module at low cost and fast turnaround time. [1] FPGA hardware allow the reconfiguration of a programmable logic partial reconfiguration [2] can also be used to allow larger complex designs to be implemented on devices with

fewer resources than would normally be required for a complete implementation. The most common method for implementation of partial reconfiguration is one in which a modular design approach is used. The logic is configured in a manner such that one module is dynamically reconfigured [5] while another module is retained in a static configuration for performing operations which do not change. Partial reconfiguration is the practice of reprogramming only a portion of an FPGA.Specifically,Dynamic Partial Reconfiguration denotes the ability to reprogram a portion of a circuit while it is operating. This is done without a need for the chip to power down or be reset. Partial reconfiguration can be implemented through the use of the Xilinx ISE tools in conjunction with PlanAhead software [8]. Partial reconfiguration generally requires the use of a modular design flow. Modular design [4] requires additional procedures for synthesis and implementation and adds complexity not found when utilizing the basic design flow. In general, modular design allows for simultaneous development of different modules which together complete the design of an FPGA. The modular design flow consists of two phases. PlanAhead's main purpose is to customize the way circuits designed in ISE are laid out on the FPGA as well as providing timing and placement analysis to improve circuit function. By using this tool users can group circuits and modules. The benefit of this is that if each module is in its own area, the task of partial reconfiguration becomes much easier as only one area is being programmed and will not affect any of the other sections. This task is known as floor planning. PlanAhead also provides a useful set of design rule checks which can be used to improve designs and provide suggestions to the designer.

2. Dynamic Partial Reconfiguration

Partial reconfiguration [4] is of two types namely dynamic partial reconfiguration and static partial reconfiguration. In this project we concentrate on the dynamic partial reconfiguration process. Dynamic partial reconfiguration is also known as active partial reconfiguration, permits to change a part of the device while the rest of an FPGA is still running. In other words, while the partial data is sent into the FPGA,the rest of the device will be still running and the new data is being configured into FPGA.There are two styles of dynamic partial reconfiguration[4] named as Difference based partial reconfiguration and Module based partial reconfiguration.Difference based partial reconfiguration can.

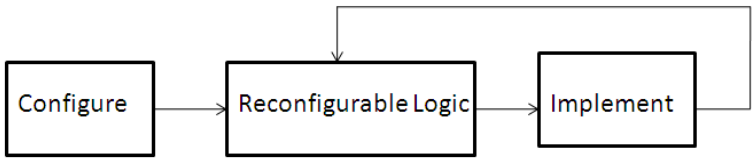

Figure 1: Dynamic Partial Reconfiguration.

be used when a small change is made to the design.It is especially useful in case of changing Look-Up Table(LUT) equations or dedicated memory blocks content. Module based partial reconfiguration uses modular design concepts to reconfigure large blocks of logic.

3. EAPR Design Flow

The dynamic partial reconfiguration is supported on all types of Virtex series device.Early Access Partial Reconfiguration[10] is the latest design flow used for partial reconfiguration. The EAPR flow allows signals in the base design to cross through a partially reconfigurable region without busmacro.This improves timing performance and simplifies the process of building a PR design. Static module is the design remains in operation during the PR process.

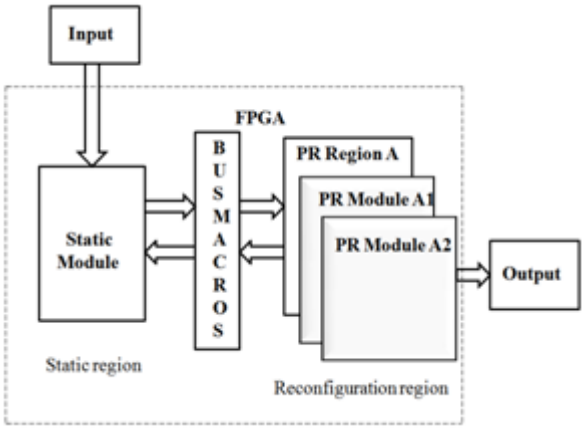

Figure 2: Architecture of EAPR design.

Partial Reconfiguration Module(PRM)[6] can be swapped in and out of the device.Multiple PRMs can be defined for a specific region.Partial Reconfiguration Region(PRR) is the part of the FPGA that is set aside for partial reconfigurable modules.More than one PRRs can be set for reconfiguration.Here a Proxy LUT is used which automatically connects the reconfigurable modules to the static module.This was possible in Virtex4 and above series of devices.There is no need to use busmacros for the purpose of connecting the modules which is the greatest advantage.The EAPR design flow can be implemented using PlanAhead[8].

4. Implementation with Planahead

The EAPR design flow is shown in the figure 3.This can be implemented using PlanAhead tool[8].

i) HDL design description: The HDL design description for the top module,static module and the reconfigurable module are implemented here. The top module doesnot consist of any logic. It contains only I/O instantiations, static modules, clock signals etc..The static module is nothing but the base module which remains in the stable state during the partial reconfiguration state. The HDL description for the reconfigurable modules is implemented.These make use of the clock signals from the top level modules.

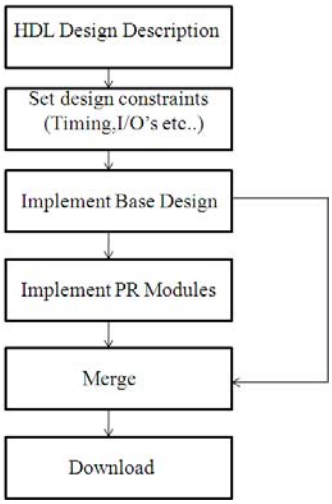

Figure 3: EAPR design flow.

ii)Set design constraints: The area group, reconfiguration mode,timing constraint and location constraints are defined here.

*iii) Implement base module:*The static modules will be created.The constraints files will be checked whether it has been properly created.

*iv)Implement PR modules:*There are various steps to be performed for creating a PR module

- Create a Reconfigurable Partition(RP).
- Add a Reconfigurable Module(RM)
- Define P block ranges for the reconfigurable partitions.
- Run pre-specific DRC checks.
- Implement configurations.
- Verify the configuration.

v) Merge: As soon as the configurations have been checked, all the base and the PR modules will be merged in order to complete a design. Furtherly many partial bit streams for each PRM and initial full bit streams are created to configure the FPGA. These partial bit files will be further downloaded onto the FPGA.

5. Results

In this paper two different modules have been taken and they are reconfigured on the FPGA at runtime. The effective utilization of the resources will be taken place due to this runtime reconfiguration process. Here the functionality of one module can be replaced with the functionality of another module. So any kind of module can be reconfigured with a different functionality on FPGA dynamically. Reconfiguration time can be greatly reduced. The reconfigurable module-Vedic multiplier is designed and this module will be reconfigured in the place of an array type of multiplier. The simulation and synthesis results for the Vedic and Array multipliers are shown in the following figures:

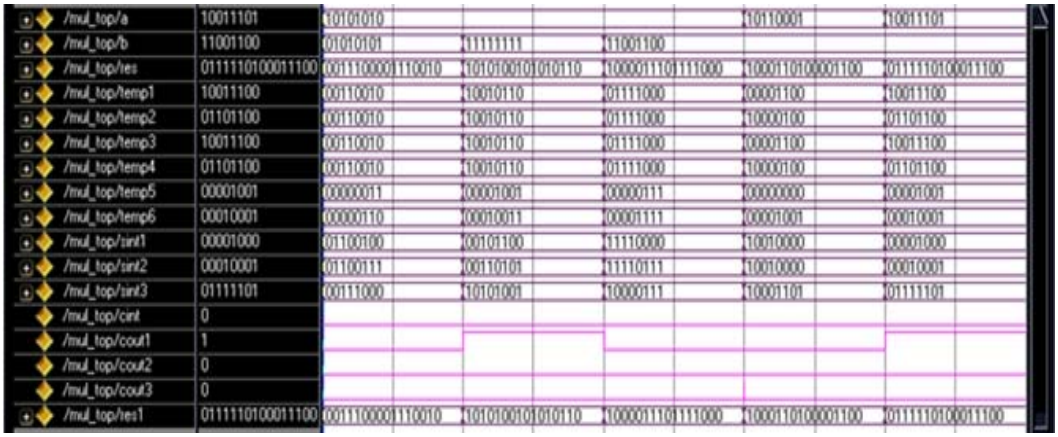

Figure 4: Simulation result for 8×8 Array type of multiplier.

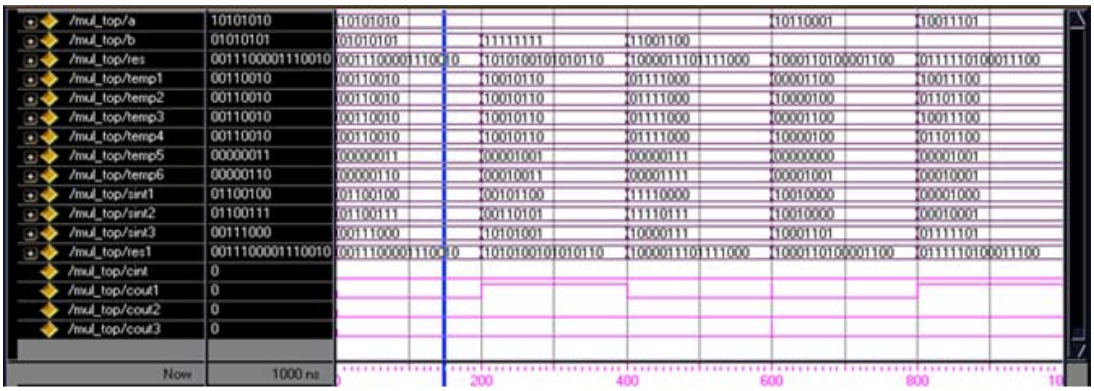

Figure 5: Simulation result for 8×8 Vedic multiplier using Urdhva Tiryagbhyam Sutra.

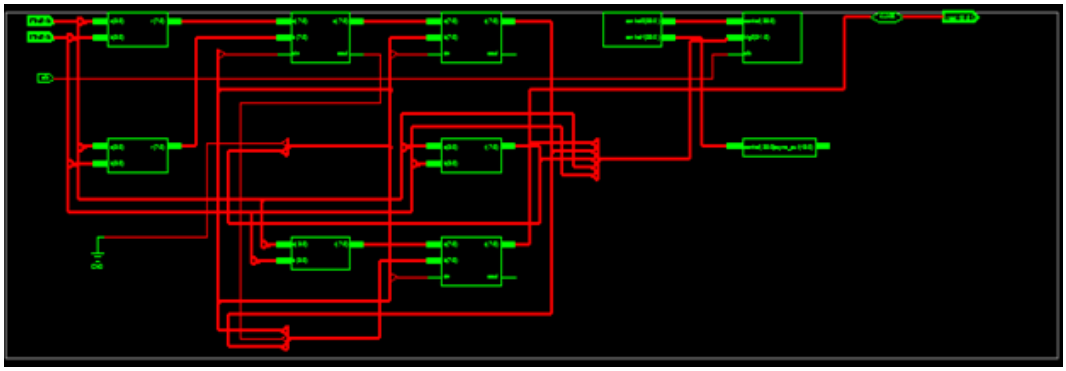

Figure 6: RTL Schematic of 8×8 Vedic multiplier.

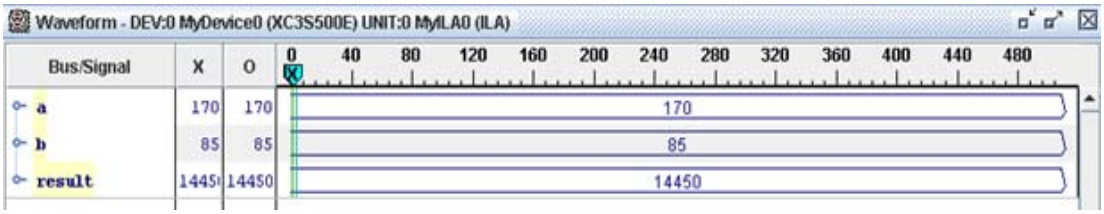

Figure 7: Runtime output view in chipscope.

Table 1: Comparison between multipliers.

Device Utilization	**Array Type OF 8×8 Vedic Multiplier**	**8×8 VEDIC Multiplier Using Urdhva Tiryagbhyam Sutra**
Selected Device	3S500EFG320-5	3S500EFG320-5
Number of Slices	90 OUT OF 4656 1%	91 OUT OF 4656 1%
Number of 4 Input LUTS	157 OUT OF 9312 1%	161 OUT OF 9312 1%
Number of IOS	17	17
Number of bonded IOBS	17 OUT OF 232 7%	18 OUT OF 232 7%
Delay	22.995NS AT 43.48MHZ	20.477NS AT 48.83MHZ

6. Conclusions

In this manner, we showed that the design flow methodology EAPR have the clear advantage over the existing FPGA design methodology. This methodology is also applicable to detect some kind of problems in the FPGA architecture with the help of redundancy at the runtime. Partial reconfiguration begins new advantages to encryption implementation with the use of an FPGA.

References

[1] Krzysztof Jozwik, Hiroyuki Tomiyama, Masato Edahiro,Shinya Honda and Hiroaki Takada,(June 2012)"*Comparison of Preemption Schemes For Partially Reconfigurable FPGAS*", IEEE Embedded Systems Letters, **vol.4**, *no.2*.

[2] Kyprianos Papadimitriou, Student Member IEEE, Tonis Anyfantis and Apostolos Dollas, Senior Member, IEEE,(June 2010*),"An Effective Framework to Evaluate Dynamic Partial Reconfiguration in FPGA Systems"*, IEEE transactions on Instrumentation and Measurement, **vol. 59**, no.6.

[3] Zine EL Abidine Alaoui Ismailia ns Ahmedmoussa,(2009*),"Self-partial and Dynamic Reconfiguration Implementation for AES using FPGA"*, IJCSI International Journal of Computer Science Issues,**Vol.2**.

[4] Wang Lie, Wu Feng-yan,(2009*),"Dynamic Partial Reconfiguration in FPGAs"*, Third International Symposium on Intelligent Information Technology Application,**vol.2**,IEEE.

[5] Eric J.Mcdonald,(July 2008)*" Runtime FPGA Partial Reconfiguration"*, IEEE A&E systems Magazine.

[6] S.S.Shriramwar and Kartik Ingole,(May 2012*),"Partial Reconfiguration for Signal Processing Application using FPGA"*,International Journal ofAdvances in Engineering &Technology,**Vol.3**,Issue 2, pp. 680-684.

[7] Matthew G.Paris,*(*2008*),"Optimizing Dynamic Logic Realizations for Partial Reconfiguration of Field Programmable Gate Arrays"*,B.S.University of Louisville.

[8] Xilinx,Inc.*Partial Reconfiguration Tutorial PlanAhead Design Tool*, (May 8,2012).UG743**(V14.1).**

[9] Xilinx,Inc.*Overview of PartialReconfiguration Flow,PlanAhead software tutorial*,(July23,2010),**(v12.2).**

[10] Xilinx,Inc.UG208,*Early Access Partial Reconfiguration userguide* http://www.xilinx.com.2006.

[11] Swami Bharati Krishna Tirtha,*Vedic Mathematics*. Delhi: Motilal Banarsidass Publishers, (1965).

[12] Ming-Chen Wen,Sying-Jyan Wang, and Yen-Nan Lin,(10-12 May 2005*),"Low Power Parallel Multiplier with Column Bypassing"*,Electronics letters,**vol.41**,Issue Page(s):581-583.

[13] Kamboh, Hamid M,Khan Shoab A,(2012),*"FPGA Implementation of fast adder"*,7th International Conference on Computing and Convergence Technology(ICCCT),Page(s):1324-1327, IEEE.

[14] Xilinx Inc,*Chipscope Pro Software and Cores userguide* www.xilinx.com/chipscope_pro_sw_cores_ug029.pdf, UG029 **(v9.2),** May 30, 2007.