International Journal of Latest Research in Engineering and Technology (IJLRET)
ISSN: 2454-5031
www.ijlret.com ǁ PP. 117-123

Implementation of Optimized ALU for Digital System Applications using Partial Reconfiguration

NAVEEN K H[1], Dr. JAMUNA S[2], BASAVARAJ H[3]
[1](PG Scholar, Dept. of Electronics and Communication, Dayananda Sagar College of Engineering, Bangalore, India, Email ID: naveenkhss@gmail.com)
[2](Professor, Dept. of Electronics and Communication, Dayananda Sagar College of Engineering, Bangalore, India, Email ID: jamuna-ece@dayanandasagar.edu)
[3](Assistant professor, School of Engineering and Technology, Jain University, Bangalore, India, Email ID: basavaraj.ec@gmail.com)

ABSTRACT: Arithmetic Logic Unit (ALU) is an essential building block of any Central Processing Unit (CPU). Modern CPU's perform multiple operations which results in complex ALU design. This motivated us to design and implement an optimized ALU using partial reconfiguration. Partial Reconfiguration is well-defined as the ability of a single system to get reconfigured to accomplish numerous applications. Newer Xilinx FPGAs (few Spartan & Virtex series) offers the opportunities to be reconfigured by Partial Reconfiguration (PR) technique. The internal blocks of the ALU designed in this paper includes Adder, Subtractor, multiplier, shifter and Logical Block. The performance with respect to area and delay constraints is evaluated. The simulation results obtained validate the proposed concept.

KEYWORDS: Arithmetic Logic Unit, Central Processing Unit, Field programmable gate arrays, Partial Reconfiguration.

1. INTRODUCTION

In general, reconfiguration of Field Programmable Gate Array (FPGA) is done by resetting the device. While in partial reconfiguration (PR), the FPGA can be reconfigured during runtime. Partial Reconfiguration is the method of reconfiguring only a certain portion of the hardware circuitry without interrupting the rest of the operations [1]. High end FPGAs like Virtex and Spartan series manufactured by Xilinx vendors has provided such features. High-level applications and algorithms in different areas have been attempting to incorporate the partial reconfiguration concept into their design to boost their performance. NASA deep space mission is one of the typical examples of such scenarios, which requires high reliability involving mission safety or other critical tasks. Such applications rely increasingly on FPGAs to support their computing requirements.

SRAM based FPGA is a multiprocessing device where the modules operate concurrently within the same chip. The benefit of such device is that its memory can be reconfigured at any time. Partial reconfiguration enhances the SRAM based FPGA by reconfiguring only certain portion of the chip without resetting the device [2].

ALU is the arithmetic and logic unit of a processor. It contributes highest power density location and occupies more circuit area [3]. PR concept is been used in this paper to design an optimized ALU that satisfy the performance requirements.

This paper is organised into 5 sections, Section 1 gives the introduction, Section 2 describes the related work carried out in understanding the PR concept, Section 3 explains the design and implementation of ALU using PR concept, Section 4 shows the simulation results and section 5 concludes the paper.

2. RELATED WORK

2.1 Partial Reconfiguration Design Flow

PR design flow is divided as difference based and module based design flows. In general, modular design flow allows for simultaneous development of different modules with together complete the design of an FPGA. This design procedure is most commonly used by engineering and programming teams who must coordinate efforts to save time and money. Modular design also allows a team to modify non-working or unstable modules without affecting those which are functioning properly. Difference-based partial reconfiguration follows a different design flow. This approach requires the creation of a bit stream file which only includes design differences from one design to another.

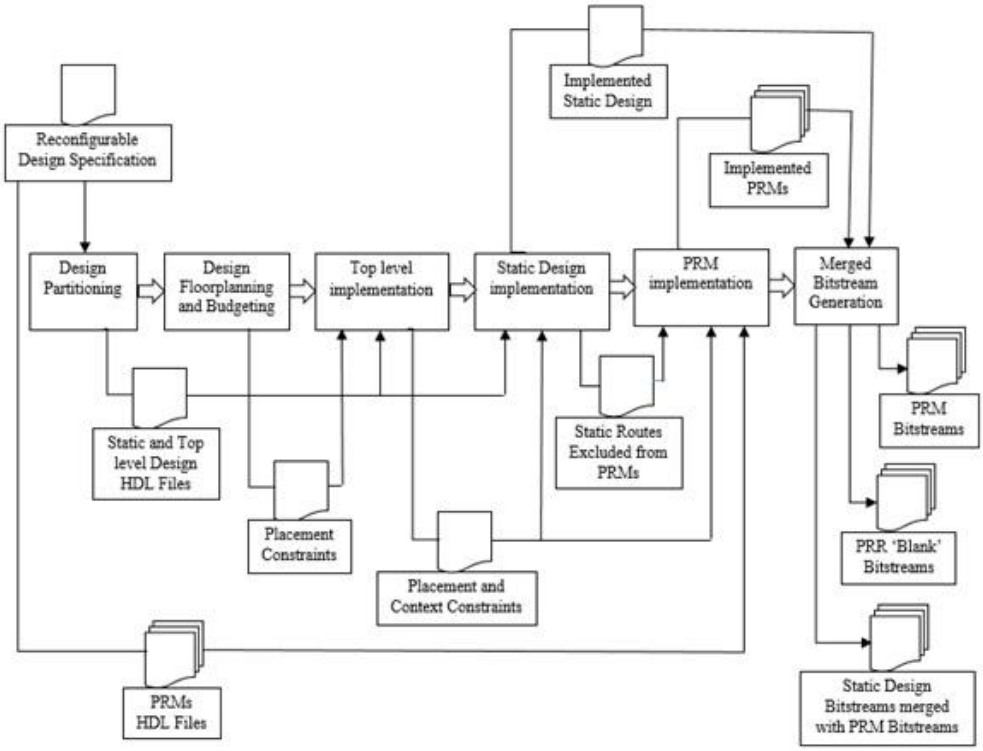

Figure 1: Partial Reconfiguration design flow.

PR flow physically divides the device regions as static region and dynamic region. Static portion of the device is programmed at start up and never changes, whereas dynamic portion of the device will be reconfigured dynamically multiple times with different designs. The configuration data produced to program a reconfigurable device is known as bitstream. PR flow will generate two bitstreams, one for static region and another for dynamic region. There will be multiple PR bitstreams, one for each design that can be dynamically loaded to the device.

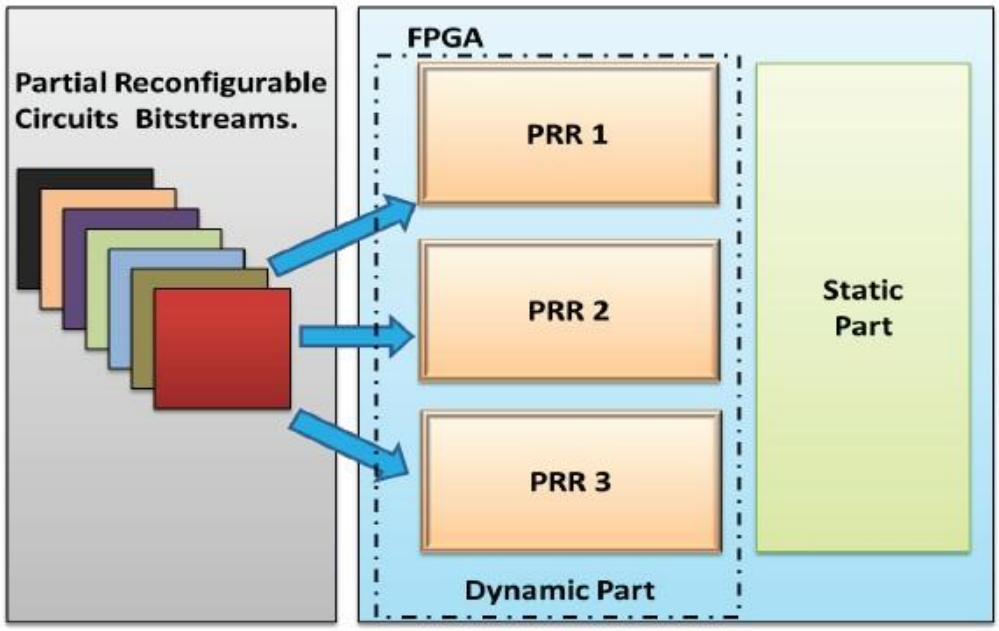

Figure 2: Partial Reconfiguration FPGA showing static and dynamic regions.

2.2 Approaches For Partial Reconfiguration

There are three approaches to design Reconfigurable Systems
1. Reconfiguration using JTAG
2. Reconfiguration using RS232
3. Auto Reconfiguration

2.2.1 Reconfiguration using JTAG

To begin testing the design, the initial static_full.bit file was downloaded to the development board through the JTAG chain. JTAG is the simplest programming procedure completed through the Xilinx iMPACT software. However, JTAG programming requires a separate computer or board to load the files through the JTAG chain. Partial bit files are then downloaded to the board in the same manner. When a partial bit file is selected to load, it is not necessary to power down the board or to re-initialize the JTAG chain. The FPGA will continue to operate non reconfigurable regions while the new partial bit file is loaded [4].

2.2.2 Reconfiguration using RS232

The RS-232, is a legacy, full duplex, wired, Asynchronous Serial Communication Interface. It extends the UART communication signals for external data communication. RS 232 interface defines various handshaking and control signals for communication apart from transmit and receive signal lines for data communication.

2.2.3 Auto Reconfiguration

The FPGA reconfigures itself to accomplish usefulness, either after predefined time length or in light of a status flag. Microblaze delicate processor and ICAP interface are utilized for this circumstance as well, yet the utilization of UART interface is not compulsory.

3. EXPERIMENTAL DETAILS

ALU is the integral part of the Central Processing Unit which performs arithmetic and logical operations. The main operation of any processing unit is fetching, decoding and executing an instruction. The fetching of instruction is done by the instruction fetch unit, the decoding of instruction is performed by the decode unit (or control unit) which generate appropriate control signals for ALU to carry-out particular operation of the instruction.

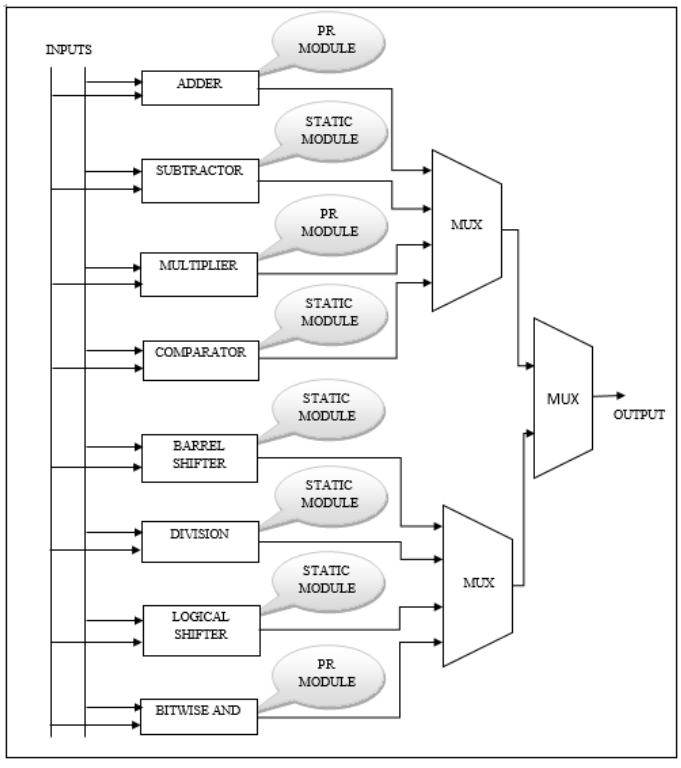

Figure 3: Block diagram of ALU.

Fig.3 shows the internal structure of an 8 bit ALU which includes adder, subtractor, multiplier, comparator, barrel shifter, division, logical shifter and bitwise AND. We have considered subtractor, comparator, barrel shifter, division and logical shifter modules as static modules. These static modules operations are not interrupted during the runtime. The remaining modules like adder, multiplier and bitwise AND blocks are considered as PR modules. These PR modules can be reconfigured with the necessary modules to satisfy the requirement of the system.

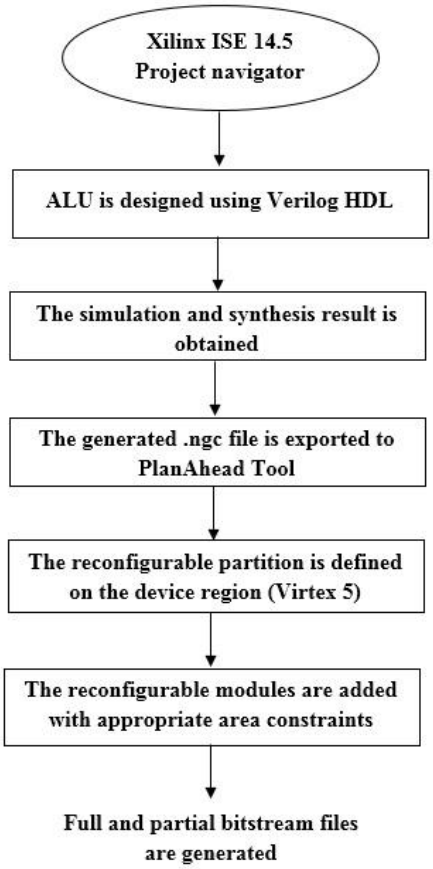

Figure 4: Partial reconfiguration design flow for ALU.

Fig.4 shows the methodology employed in implementation of ALU using PR concept. The first step is to synthesize the netlist from the HDL sources. These files are further used in the implementation process. The generated netlist files is exported to PlanAhead tool. In PlanAhead tool the device region is partitioned into static and PR regions. The partial reconfiguration process requires separate netlists for the static (top level) design and for PR modules. A netlist is generated for each implementation of the partial reconfiguration partition used in the design. Full and partial bitstreams are generated for different configuration of PR modules [1].

4. SIMULATION RESULTS

The simulation result obtained from Xilinx ISIM for 8 bit ALU design comprising of adder, subtractor, multiplier, comparator, barrel shifter, division, logical shifter and bitwise AND is shown in Fig.5.
Inputs A=00001000, B=00000100, Cin=0.
Outputs sum=00001100, difference=00000100, product=0000000000100000, comparator Alarger=1, barrel shifter=01000000, division=00000010, logical shifter=10000000, bitwise AND=00000000.

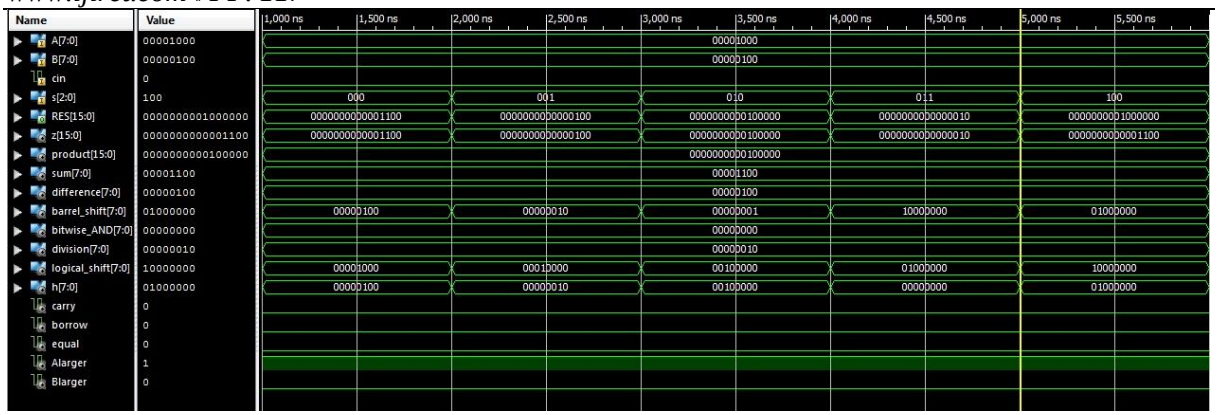

Figure 5: Simulation result.

The summary of resource utilization for synthesized ALU is shown in Fig.6. Number of slice LUTs and delay with respect to different configurations of PR modules is shown below.

Device utilization summary(estimated values)				
Logic utilization/Different configurations using PR	Configuration1 Ripple carry adder Array multiplier Bitwise AND	Configuration2 Carry look ahead adder Multiplier Bitwise XOR	Configuration3 Carry save adder Booth multiplier Bitwise XNOR	Configuration4 BCD adder Vedic multiplier Bitwise NOR
Number of slice LUTs used	298	287	316	328
Number of bonded IOBs used	36	36	36	36
Delay	18.702ns	19.312ns	18.516ns	18.815ns

Figure 6: Synthesis summary.

Implementation and Bitstream generation of PR modules with different configuration applied for ALU design is shown in Fig.7 and Fig8.

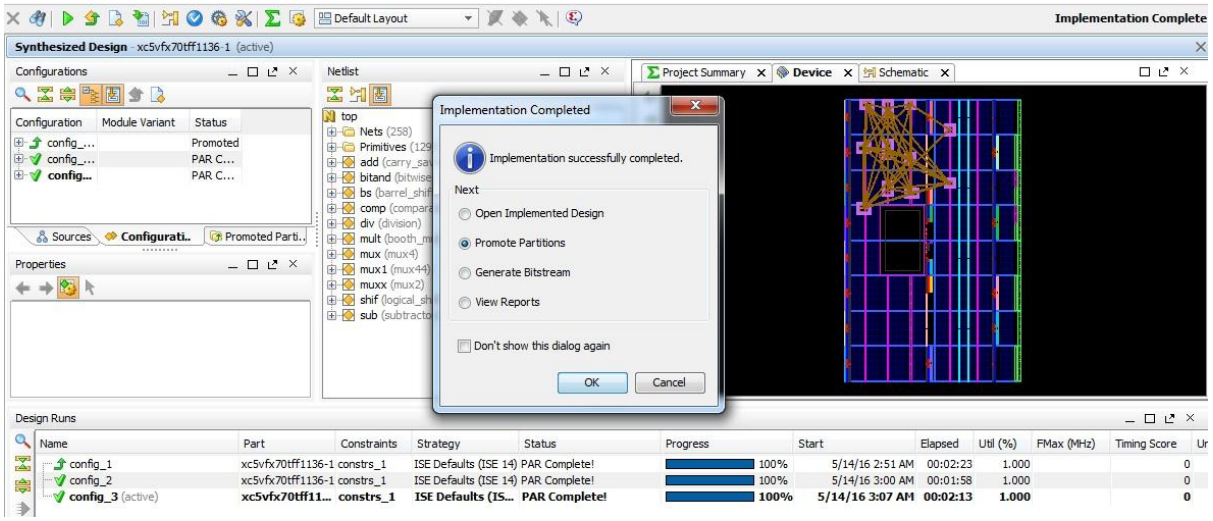

Figure 7: Implementation of PR modules with different configurations.

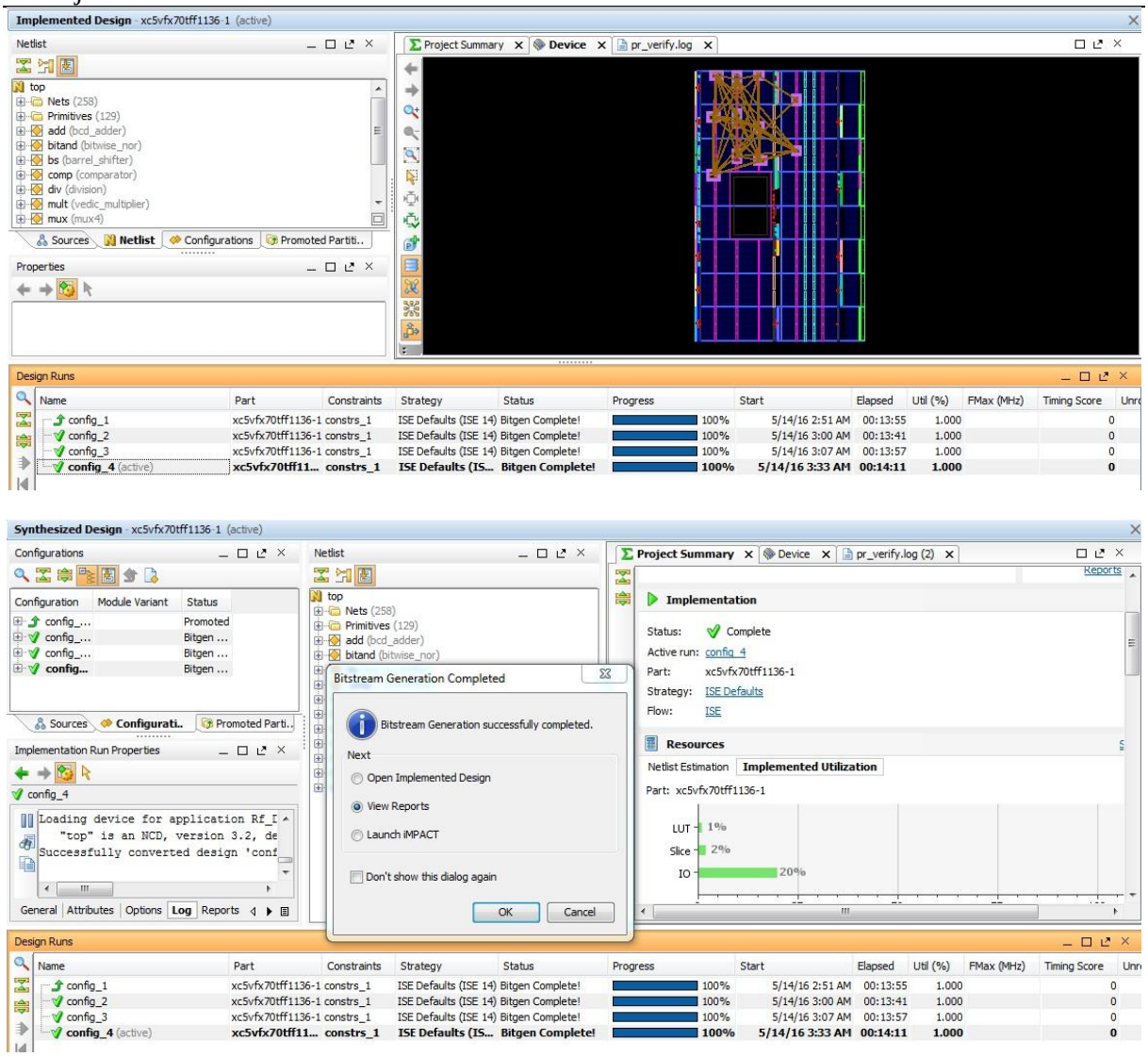

Figure 8: Bitstream generated for different configurations of PR modules.

5. CONCLUSION

This paper presents design, simulation and implementation of ALU using partial reconfiguration. The objective was to design and implement an optimized ALU to satisfy the area and delay constraints by reducing the overall reconfiguration time of the system. We have generated full and partial bitstream files for four different configurations of PR modules. This time multiplexing optimized ALU can be used in several applications such as digital signal processors, digital design, etc.

6. REFERENCES

[1]. Xilinx, Inc. "Partial Reconfiguration User Guide." Xilinx UG702. July 6, 2011. http://www.xilinx.com/support/documentation/sw_manuals/xilinx13_2/ug702.pdf

[2]. Farid Lahrach, Abderrahim Doumar and Eric Chatelet,"Fault Tolerance of SRAM-based FPGA Via Configuration Frames", Design and Diagnostics of Electronic circuits and Systems (DDECS), *IEEE 14th International Symposium*, on page(s): 139 – 142, Print ISBN**:** 978-1-4244-9755-3, April 2011.

[3]. Akshay dhenge, Abhilash kapse, Sandip kakde, "VLSI Implementation of Area Optimized ALU using GDI Technique", *Proceedings of International Conference on Research in Electrical, Electronics & Mechanical Engineering, Dehradun, 26th April-2014, ISBN: 978-93-84209-11-7.*

[4]. Kunal Yogeshkumar Parikh, "Detailed Design Flow for Partial Reconfiguration", *International Journal of Combined Research & Development (IJCRD) eISSN:2321-225X;pISSN:2321-2241 Volume: 2; Issue: 2; February – 2014.*

[5]. Maysam Sarfaraz, *"Educational Applications of Partial Reconfiguration of FPGAs", T*he University of Tennessee at Chattanooga, Tennessee, May 2011

[6]. Sheetal U. Bhandari, Shaila Subbaraman, Shashank Pujari and Rashmi Mahajan, "Internal dynamic partial reconfiguration for real time signal processing on FPGA*", Indian Journal of Science and Technology Vol. 3 No. 4 (Apr. 2010) ISSN: 0974- 6846.*