International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 24 (2017) pp. 15261-15268

Design of Online Self-Testable Multi-core System using Dynamic Partial Reconfiguration of FPGA

G. Prasad Acharya
Associate Professor, Department of Electronics and Communication Engineering,
Sreenidhi Institute of Science and Technology, Hyderabad, Telangana, India.
Orcid Id: 0000-0003-2427-7541

Dr. M. Asha Rani
Professor, Department of Electronics and Communication Engineering,
Jawaharlal Nehru Technological University, Hyderabad - (JNTUH), Telangana, India.
Orcid: 0000-0002-0047-4236

Abstract

This paper presents a novel and efficient method of designing an online self-testable multi-core system. Testing of a Core Under Test (CoUT) in a massively multi-core system can be carried out while the system is operational, by assigning the functionality of the CoUT to one of the non-functioning/idle and pre-tested core. The methodology presented in this paper has been implemented taking a test setup by demonstrating the Dynamic Partial Reconfiguration (DPR) feature of latest 7 series FPGAs. The simulation results obtained from the experimental setup show that the utilization of a multi-core system can be significantly improved by effectively utilizing the idle core(s) to back up CoUT for concurrent on-line test without a significant hardware overhead and test latency.

Keywords: Partial Reconfiguration, test scheduling, on-line self test, Core under Test, partial bit stream.

INTRODUCTION

The rapid technological advancement has led the design of both Application Specific as well as General Purpose System-on-Chip devices within a single chip. Over the past two decades, SoC technology is integrating multiple parallel processing elements, called cores along with other hardware resources like embedded memory blocks, memory controllers, I/O circuitry, A/D and D/A interfaces, peripheral hardware and so on, into a single chip. The availability of multiple cores in a parallel-computing processor enables its Operating System (OS) to divide the given task into several independent threads and executing them in parallel so that the overall execution time of the task can be reduced.

The multi-core System on Chip integrates multiple homogenous/heterogeneous cores as processing elements and a specialized control circuitry that establishes the communication among the cores and I/O interfaces. With the advances in reconfigurable systems, the designer has a wide variety of choices, the most effective being Field Programmable Gate Arrays (FPGAs) and Complex Programmable Logic Devices (CPLDs) for rapid prototyping of their designs. The DPR methodology enables the designers to modify or partially reconfigure specific portion of FPGA as per designer's requirements on fly while keeping the other portion of FPGA unchanged [1-2]. The DPR finds wide applications in reconfigurable processors development, image and signal processing, surveillances and defense applications wherein a large amount of data is to be processed.

In a conventional method, testing of a system for its functionality can be carried out using offline test procedures either during boot up or by executing a specific test procedure by suspending its functional operation.[11-12] have presented cost-effective core testing in Network-on Chip (NoC) wherein a unicast based multicast scheme is proposed to deliver test patterns from Automatic Test Equipment (ATE) to all the cores, routers and interconnects. Test responses from all the cores under test for each test vectors are sent back to the ATE via a reverse path. This test methodology is based on offline testing wherein all the cores in the NoC must be active for uni-casting/multicasting during test process. The offline testing presented on the available literatures though focus on maximizing the fault coverage, they are incapable of detecting temporary faults because the behavior and occurrence of temporary faults are unpredictable. These temporary faults are better targeted with online testing. The work presented in this paper is aimed to facilitate online testing of multi-core systems and is motivated from the assumption that all the cores in the multi-core system are not always being involved in functional operation. The idle core(s) at a given time may be used as spare core(s) for testing of functional core(s). Though this methodology may not facilitate strictly online testing, but is aimed to facilitate online testing of cores whenever spare core(s) is(are) available during test.

The number of cores in a massively multi-core systems may range from 10s to 100s, wherein all the available cores may not be functioning simultaneously. The task scheduling feature available in the Operating System (OS) kernel of the

latest microcontroller based FPGA can be utilized for scheduling the cores for test. This approach will facilitate online testing of a functional core without suspending the functional operation of the system. In the proposed test methodology, a functional core is scheduled for online test by offloading its functional operation to one of idle and pre-tested cores so that all the temporary faults in the CoUT are detected. However this methodology may not seem to provide online testing in strict sense, but can be used for online testing of non-real time systems.

This paper is organized as follows. The overview of Dynamic Partial Reconfiguration (DPR) and related work is presented in the next section. The following Section presents the generalized FPGA based reconfigurable multi-core system architecture. The DPR design flow is presented in the next Section. The proposed online self-test methodology for multi-core system, the experimental setup, the simulation results and device utilization summery are presented in following sections.

Dynamic Partial Reconfiguration of FPGA and DPR Flow

In today's SoCs much functionality are embedded in a single processor which causes increase not only in the software and hardware complexity but also in the power consumption. The power consumption especially in hand-held battery operated devices like smart phones and PDAs which run many applications simultaneously is the key concern to be dealt with. The DPR feature available in the latest FPGAs, provides an extended re-programmability to the designer to dynamically alter the design modules in the Partially Reconfigurable Region (PRR) of the FPGA keeping the static region unchanged. The DPR allows the alteration of dynamically reconfigurable region (DRR) of FPGA by loading the bit files of Partially Reconfigurable Modules(PRM) keeping the other regions of FPGA fully functional [5,7,8]. Thus, DPR allows power optimization in such devices by dynamically configuring only the modules performing the intended applications while keeping other power hungry modules turned off.

Figure 1 illustrates Dynamic Partial Reconfiguration that allows the modification of PRR of an FPGA by loading partial bit files of PRM while keeping the functionality of Static Partition (SP) of the FPGA unchanged. The Internal Configuration Access Port (ICAP) which can operate at a frequency of up to 100 MHz in the latest Virtex FPGA, is the medium for inter-chip communication that transports the partial bit files from Block RAM into the FPGA.

Dynamic Partial Reconfiguration Flow

The flow chart shown in Figure 2 illustrates the DPR flow. The Xilinx Vivado Design suite allows creating a complex block level system design by integrating soft IP cores. Once the block design is completed and verified for interconnection among the IP cores, an HDL wrapper for the design is generated which is then synthesized to generate a design check points (.dcp file) using synthesis tools.

In DPR flow, a system architecture with a black box design, which is considered as a Static Partition of FPGA for the PRM, is synthesized and implemented using Place and route (PAR) to generate a blank configuration (stactic.dcp). The reconfigurable modules for different configurations as mentioned in the later section are synthesized and placed into the black box i.e., PRR portion of the FPGA. The implementation step in Xilinx tool flow invokes the Xilinx PAR tool to place the design elements at appropriate locations in the target FPGA and provides routing among them as per the constraints specified in synthesis design constraint (SDC) file. The bitstream generator takes the routed netlists to generate bitstream files. The partial bit files are generated separately for static and partially reconfiguration modules and stored in the Block RAM and then loaded into the FPGA through one of the following ports: Serial Port, JTAG Port, or Internal Configuration Access Port (ICAP).

The reconfigurable multi-core computing platform aims to speed up the host Processing System (PS) by incorporating multiple Processing Elements (PE). A PE is the Reconfigurable core and is resided in the Programmable Logic (PL) area of the FPGA. These cores can be partially or fully loaded into the FPGA as per designer's requirements by downloading the partial bit-stream files from Block-RAM using HWICAP port of the FPGA. The PS (the dual core ARM9 processor available in Zynq7 PFGA board) runs a simple software application to control the hardware circuitry implemented in the PL.

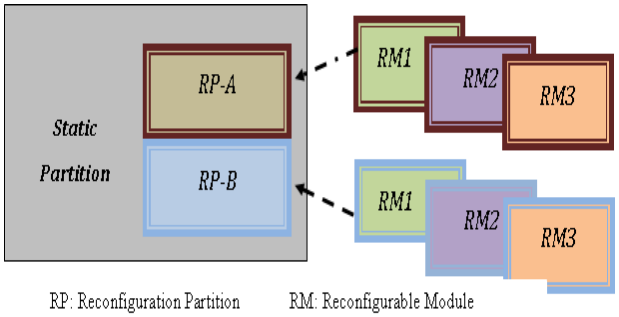

Figure 1: Illustration of Dynamic Partial Reconfiguration

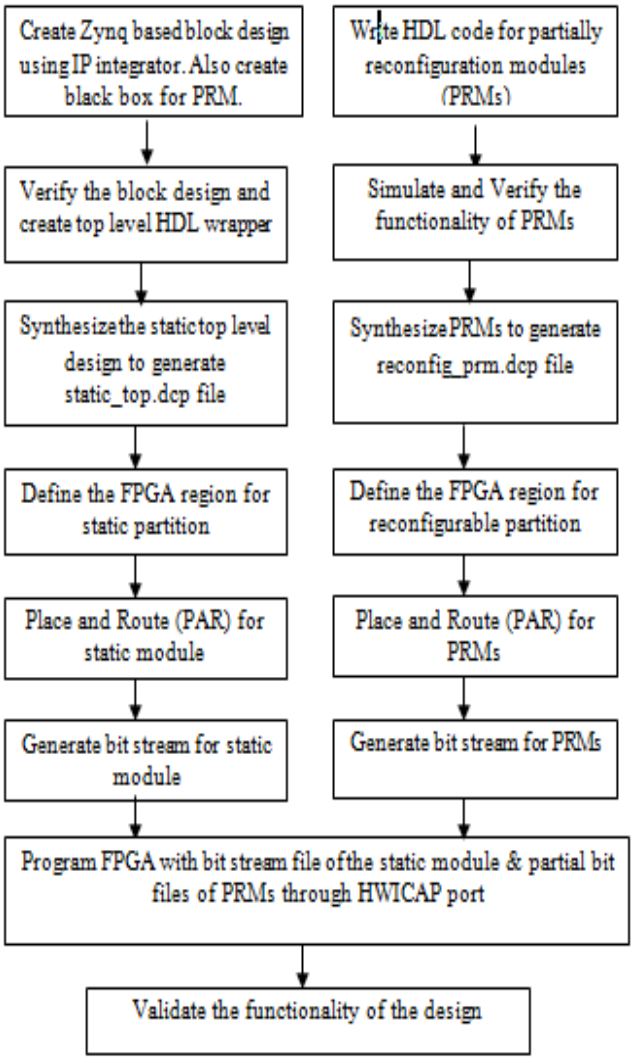

Figure 2: Flowchart for DPR using Xilinx Vivado tool

Reconfigurable Multi-Core System Architecture

A complex system that performs massively parallel applications need to support a large number of PEs up to 100 or even more. The reconfigurable platform supports the existence of a variety of PEs (smaller PEs to bigger PEs) based on the applications and their complexity. To support fine-grained reconfigurable granularity, the FPGA fabric is distributed into a fixed-size Reconfigurable Processing Clusters (RPC) and the reconfiguration is done at the cluster-level.

The architecture of the reconfigurable platform shown in Figure 3, consists of tightly coupled FPGA with a host system through a high speed PCI bus. The Virtex-7 FPGA from Xilinx (Xc7z020) consists of dual core Cortex-M3 ARM9 processor as a high performance PS which communicates to PL portion of FPGA through AXI bus with variable bandwidth [6]. The reconfiguration partition of FPGA consists of RPCs which are interconnected through Cluster Interconnect Area (CIA). Each RPC contains two or more PRMs along with local and shared cluster memory and an interface with CIA. The operating system on the host system runs a software module called Hypervisor which performs the following operations: (i) assigning the applications to one of the cores in the instantiated RPC and recording its performance (ii) issuing of commands for the reconfiguration engine that controls the generation and loading of configuration bit-stream of the selected PRM into the dynamic partition of the FPGA. The reconfiguration engine provides an interface or abstraction layer between the OS of the PS and the reconfiguration process and resides in the static region of the FPGA. Static Module(s) which remains unchanged during dynamic reconfiguration is implemented in the static portion of the FPGA.

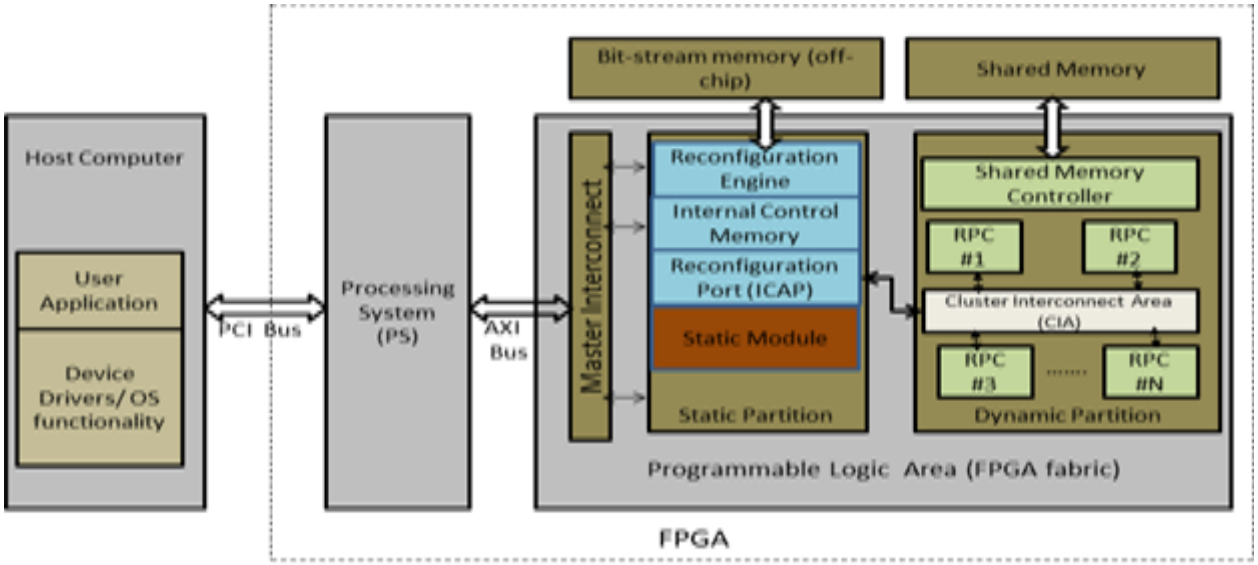

Figure 3: Architecture of an FPGA based Reconfigurable System

Proposed Online Self-Test Methodology in Multi-Core System

The multi-core System consists of multiple number of homogeneous/heterogeneous cores, referred as Processing Elements (PE). The processing speed of such systems are accelerated due to the fact of splitting a given tasks into a number of data-independent threads and scheduling them for execution among the available parallel cores. This parallel architecture can be exploited for online test [3] of Processing Elements (cores) by utilizing the idle core as back-up core for CoUT.

Test Framework

A typical homogenous multi-core DSP system shown in Figure 4 is designed to exploit the proposed test methodology in this work. The system consists of four identical DSP Sub-Systems as Processing Elements (PEs) and each subsystem consists of ADDSUB_MACRO cell that can perform complex DSP MAC operations on integer and floating point numbers. However the data size for each ADDSUB_MACRO cell considered in this simulation work is 4-bits, it supports variable data size of 1 to 48-bits. Each sub system is equipped with the proposed embedded BIST circuitry that facilitates self-testing.

Embedded BIST Circuitry

The BIST circuitry for the DSP sub-system shown in Figure 5 utilizes on-chip ROM-1 to store predefined test patterns and on-chip ROM-2 to store expected responses of the DSP sub-system for the test pattern applied. Either functional inputs or test patterns from ROM-1 are applied as the inputs to the DSP sub-system based on Test Mode (TM) selection signal. The address generator shown in the Figure 5 generates addresses for both ROM-1 and ROM-2 to access test patterns and test responses during BIST operation. The Output Response Analyzer (ORA) employs a comparator that compares the CoUT response with the expected response to detect the presence of faults in the CoUT.

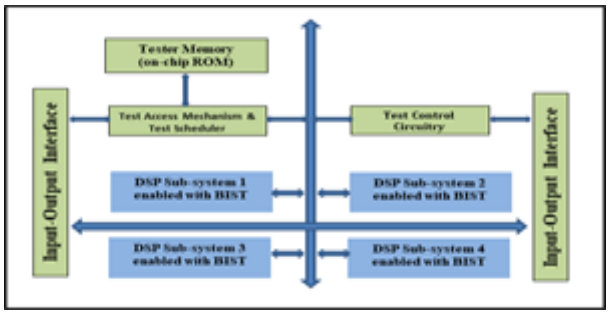

Figure 4: A typical DSP system consisting of Parallel Cores (DSP Sub-Systems)

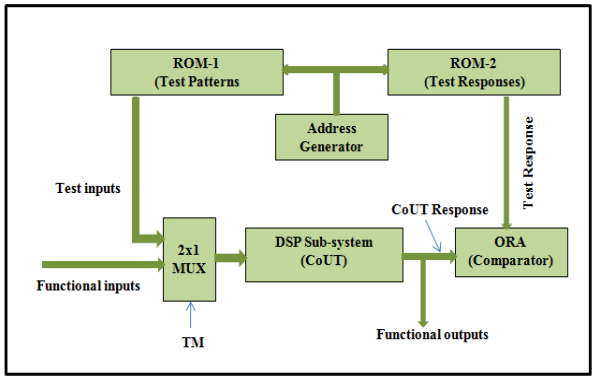

Figure 5: A BIST-enabled DSP Sub-System #N

Test Access Mechanism (TAM) bus is used to transport test patterns from ROM-1 to CoUT and test responses from CoUT to ORA. Test scheduler schedules (one after another) all the cores for test by assigning the functional inputs of CoUT to one of the idle and pre-tested core. The proposed test methodology may not be utilized when all the cores are engaged in functional operation.

Proposed Online Self Test Methodology

The proposed test methodology facilitates online testing of one core at a time in a multi-core system while the system is functional. It is assumed that all the cores in a massively multi-core system will not involved in functional operation simultaneously. The unused/idle core(s) is(are) first subjected to test using built-in self test circuitry. The pre-tested idle cores are utilized as spare/standby cores when other cores are tested one after another.

The task of scheduling a core for test is carried out by a dedicated block called Test scheduler. The test scheduling procedure for the typical multi-core DSP system [Figure 4] is demonstrated by the flowchart shown in Figure 6.

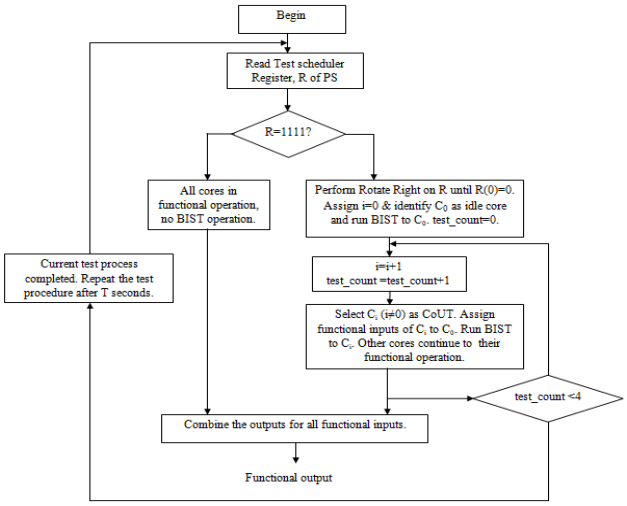

Figure 6: Flowchart for test scheduling

EXPERIMENTAL SETUP

To implement and evaluate the proposed test methodology for the reconfigurable multi-core framework, a Zynq based Processing System (PS) is created using IP-Integrator in Xilinx Vivado environment and then prototyped in Xilinx Virtex-7 FPGA (Xc7z020 Evaluation board) . The Zynq board consists of two ARM9 Processors with associated L1/L2 Cache and Memory Controllers as the Processing System (PS) and Programmable Logic (PL) area which is the heart of FPGAs. Hardware Internal Configuration Access Port (HWICAP), an IP Core from Xilinx allows the embedded microprocessor to access the configuration memory for reconfiguring FPGA through its ICAP signals. This interface supports high data bandwidth (AXI4-Stream) between a master module and many slave modules connected in daisy-chain configuration. The PS accesses the configuration memory through HWICAP core for dynamically reconfiguring the PRMs [9].

A block level design of top module as shown in Figure 7, is created by integrating Zynq7 Processor as the Processing System and the associated IP cores. The HWICAP port of Zynq7 processor is used to load partial bit files of PRMs into FPGA. An Integrated Logic Analyzer (ILA) and dbug module are integrated along with zynq7 processor. ILA is an IP core available in the library of Vivado design suite is used to monitor the internal signals of a design. The dbug module is used by the users to debug their design quickly, easily and more effectively.

The entire reconfiguration process is carried out as five different configurations (as mentioned below in this section) of the typical DSP sub-system. The DSP sub-system under each configuration is synthesized; implemented, partial bit files are generated and then stored in Block RAM (BRAM).

The synthesized netlist for the typical DSP sub-system described in the section 4.1 is shown in Figure 8. The elaborated netlist inside the rectangular box in the figure illustrates the BIST enabled DSP sub-system as shown in Figure 5 as discussed in the section 4.1.

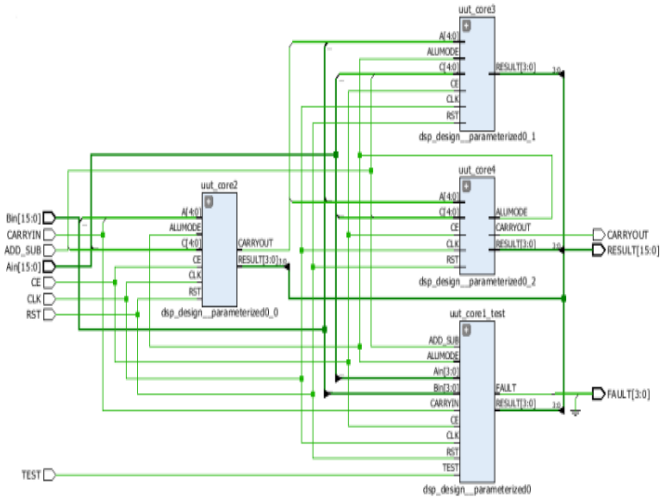

Figure 7: Top level synthesized design of DSP system

The following configurations of the typical DSP sub-system are synthesized individually and their design checkpoint (.dcp) files are created. These design check-points are implemented and verified by PAR tool and corresponding partial bit files are generated and stored in ROM. These partial bit files during FPGA programming, are downloaded into the FPGA through HWICAP port for partially reconfiguration.

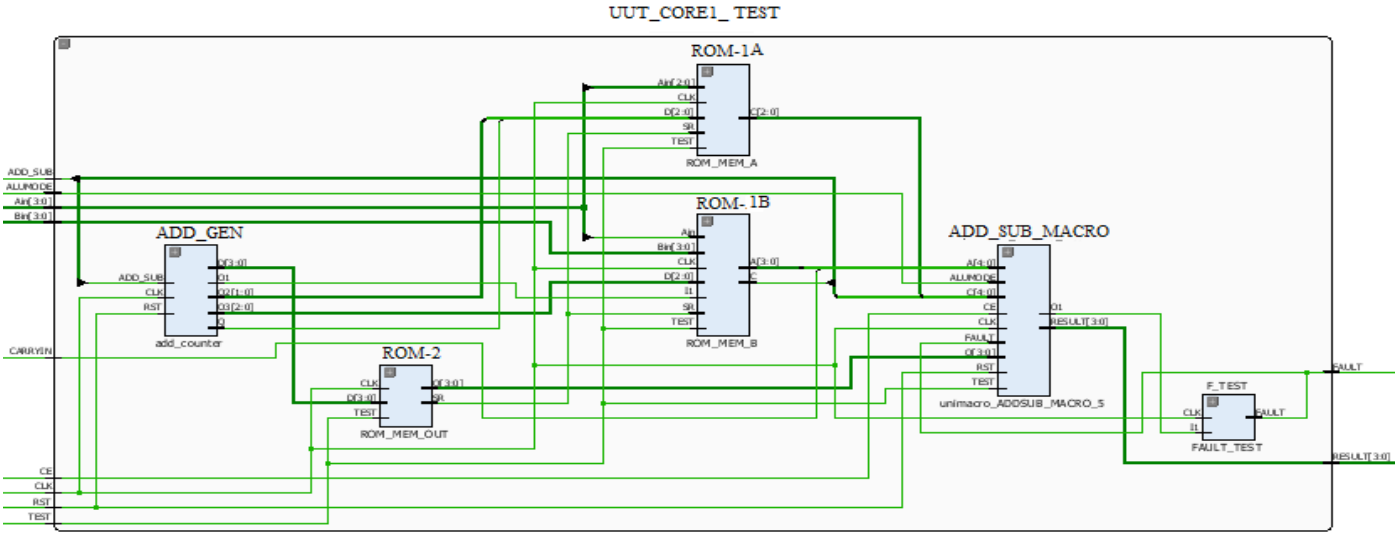

Figure 8: Synthesized netlist of the typical DSP sub-system (dsp_configuration1.dcp file)

Configuration 1. All the cores are functional. No BIST operation.
Configuration 1. Core1 is under Test and remaining cores are functional.
Configuration 2. Core2 is under Test and remaining cores are functional.
Configuration 3. Core3 is under Test and remaining cores are functional.
Configuration 4. Core4 is under Test and remaining cores are functional.

SIMULATION RESULTS & DISCUSSIONS

The simulation results obtained for the test setup explained in section 5 has demonstrated the following four test cases: (a) All cores are functional (configuration 1) (b) handing over of functional inputs of Core1 to Core4 when Core1 is selected as CoUT (c) No faults being detected in Core1 (d) injection and detection of faults in Core1.

The red color arrows in the simulation results shown in Figure 9 illustrates case (a). The inputs taken for the simulation include CARRYIN=0, TEST=0 (functional mode), ADD_SUB=1 (addition operation), configuration mode selection input, cm=000 (Configuration mode 1, all cores are functional), Ain=0xaaaa and Bin=0x5555 (input signals on which an addition operation to be performed). The obtained functional output is RESULT=0xffff with CARRYOUT=0.

Under the inputs cm=001 (configuration 1), Ain=0x0aaa and Bin=0x0555, it is observed that Core4 is idle and hence shares (case b) the entire functional inputs along with Core2 and Core3. Core1 now receives the test patterns from test_pattern memory (ROM1) which is addressed through addr. This mode of configuration is shown by white colored dotted arrows in Figure 9 where the functional inputs are Ain=0x0aaa and Bin=0x0555, module inputs after the insertion of test patterns for Core1 are Ain2=0xaaa7 and Bin=0x5555 shown by first set of dotted arrows and subsequently. The module output is RESULT=0xfffc (yellow colored arrow). The output of CoUT (Core1) is 0xC and is equal to expected response (test_out=0xC). As long as CoUT output is equal to test_out, the CoUT is assumed to have no faults (case c).

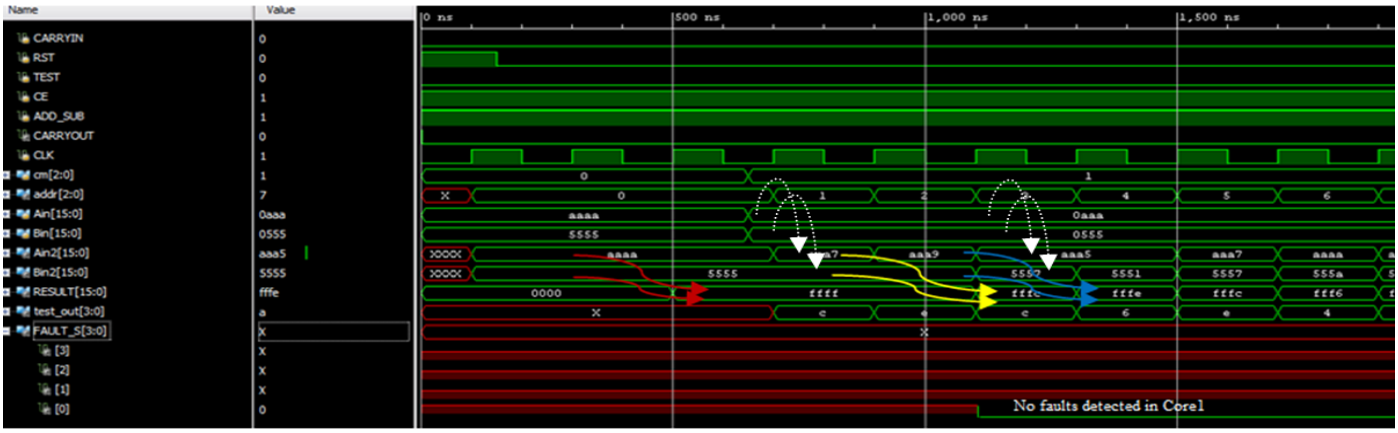

Figure 9: Simulation results for test cases (a), (b) and (c)

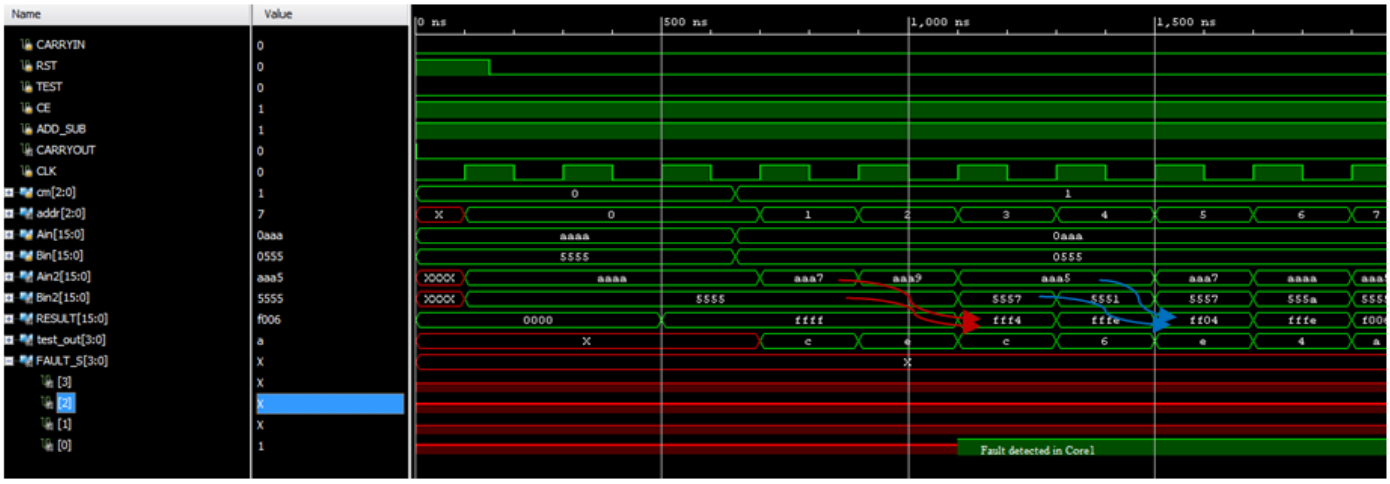

Figure 10: Simulation results for test case (d) as described in Section 6

For the simulation of case (d), some faults are injected in CoUT (Core1 in this case). It can be observed in the simulation waveform shown in Figure 10 that the CoUT output is not matching with the expected output (test_out) as illustrated by blue colored arrows thereby giving an indication of presence of fault in it (FAULT_S[0]).

Figure 11 shows the routed netlist of the entire design. The yellow color rectangular box, labeled as rM_dsp_inst inside another rectangular box pblock_rM_dsp_inst defines the DPR region wherein the typical reconfigurable DSP System is placed and routed.

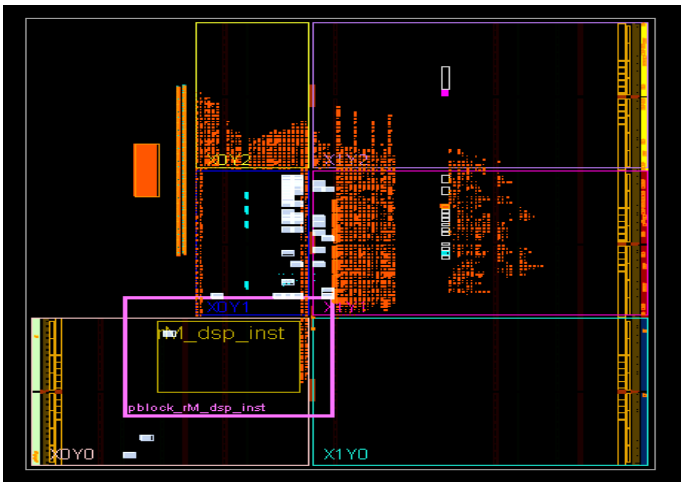

Figure 11: Routed Netlist in the FPGA

The device hardware utilization summery by the various configurations of the top level design is presented in Table 1.

The top_module consists of Zynq7 based PS (System_i), a reconfigurable DSP module (rm_dsp_inst), an ILA instant (ila_i) and a debug module (dbug_hub). The device utilization summery by each sub modules are reported in Table 1. Since the typical DSP system simulated in this work is a homogenous multi-core system, there is no significant hardware overhead (<1%) when all the cores are functional (Configuration 1- No BIST operation) and at least one core is under test (Configuration 2 to 5). However the device utilization remains the same irrespective of the core selected for test in a homogenous system, it varies from Configuration 2 to 5 when heterogeneous system is considered.

The timing report presented in Table 1 only reports the timing delays (in terms of negative slack and hold slack) for the PRM only and remains same for all the configurations. Only a marginal test latency can be observed if the scheduling time of the ARM OS kernel and the delays due to HWICAP ports to read the configuration files (partial bit files) is taken into account.

CONCLUSION

DPR allows reconfiguration of hardware dynamically into the PR region of an FPGA through its HWICAP port, while the static region of FPGA will remain unchanged. In this work, the DPR feature of latest FPGAs is efficiently utilized for online testing of each core in a multi-core System one after another by handing over the functionality of CoUT to idle/pre-tested core. The partial bit files of various configurations of the typical multi-core DSP system are loaded into the Zynq7 FPGA board through HWICAP port to verify their functionality. The simulation results obtained in this work shows a marginal increase in hardware overhead and test latency which is traded off with the online self-test capability of the system.

REFERENCES

[1]. Jitendra B. zalke, Sandeep Kumar Pandey, "Dynamic Partial reconfigurable embedded system to achieve Hardware flexibility using 8051 based RTOS on Xilinx FPGA", International Conference on Advances in Computing, Control, and Telecommunication Technologies, pp 684-686, 2009

[2]. Abelardo Jara-Berrocal, Ann Gordon-Ross, "VAPRES: A Virtual Architecture for Partially Reconfigurable Embedded Systems, DATE10 © 2010 EDAA

[3]. Shaon Yousuf, Adam Jacobs, Ann Gordon-Ross Partially Reconfigurable System-on-Chips for Adaptive Fault Tolerance, 2011 IEEE

[4]. Chuan Hong, Khaled Benkrid, Xabier Iturbe, Ali Ebrahim, and Tughrul Arslan Efficient On-Chip Task Scheduler and Allocator for Reconfigurable Operating Systems, IEEE EMBEDDED SYSTEMS LETTERS, VOL. 3, NO. 3, pp. 85-88, SEPTEMBER 2011

[5]. Kizheppatt Vipin, Student Member, IEEE, and Suhaib A. Fahmy , ZyCAP: Efficient Partial Reconfiguration Management on the Xilinx Zynq, IEEE Embedded Systems Letters, VOL. 6, NO. 3, pp. 41-44,September 2014

[6]. UG585: Zynq-7000 All Programmable SoC Technical Reference Manual Xilinx Inc., Mar. 2013.

[7]. B. Krill, A. Ahmad, A. Amira, H. Rabah, An efficient FPGA-based dynamic partial reconfiguration design flow and environment for image and signal processing IP cores Elsevier, Signal Processing: Image Communication 25 (2010)377–387

[8]. Xie Di, Shi Fazhuang, Deng Zhantao, He Wei, A Design Flow for FPGA Partial Dynamic Reconfiguration, Second International Conference on Instrumentation & Measurement, Computer, Communication and Control, pp. 119-123, 2012.

[9]. Julien Delorme, Amor Nafkha, Pierre Leray, Christophe Moy, New OPBHWICAP interface for real-time Partial reconfiguration of FPGA, pp.386-391, 2009 IEEE

[10]. PG193: Partial Reconfiguration Controller v1.0, Vivado Design Suite, 2015.
[11]. Dong Xiang and Ye Zhang, Cost-effective power-aware core testing in NOCs based on a new unicast-based multicast scheme," IEEE Trans. Computer-Aided Design, vol. 30, no. 1, pp. 135-147, Jan. 2011.
[12]. Dong Xiang, Krishndu Chakrabarty, Hideo Fujiwara, Multicast-based testing and thermal-aware test scheduling for 3D ICs with a stacked network-on-chip," IEEE Trans. Computers, vol. 65, no. 9, pp. 2767-2779, Sept. 2016.

Table 1: Device Utilization summery and Timing Report

Configuration 1 (All the cores are operational)												
	Device Utilization Report								Timing Report			
Name	# Slice LUTs	# Slice	# LUTs as logic	# LUT as memory	# Block RAM tile	# DSPs	# BSCANE2	# ICAPE2	WNS (ns)	TNS (ns)	WHS (ns))	THS (ns)
System_i	1021	369	957	64	1	0	0	1	-	-	-	-
rm_dsp_inst	8	35	8	0	0	4	0	0	-	-	-	-
ila_inst	4025	1489	3027	998	6	0	0	0	-	-	-	-
dbug_hub	326	195	302	24	0	0	1	0	-	-	-	-
top_module	**5380**	**2069**	**4294**	**1086**	**7**	**4**	**1**	**1**	**23.64**	**0.00**	**0.09**	**0.00**
Configuration 2 to configuration 5(at least one core under Test)												
Name	# Slice LUTs	# Slice	# LUTs as logic	# LUT as memory	# Block RAM tile	# DSPs	# BSCANE2	# ICAPE2	WNS (ns)	TNS (ns)	WHS (ns))	THS (ns)
System_i	1021	369	957	64	1	0	0	1	-	-	-	-
rm_dsp_inst	24	16	24	0	0	4	0	0	-	-	-	-
ila_inst	4025	1489	3027	998	6	0	0	0		-	-	-
dbug_hub	326	195	302	24	0	0	1	0	-	-	-	-
top_module	**5396**	**2088**	**4310**	**1086**	**7**	**4**	**1**	**1**	**23.64**	**0.00**	**0.09**	**0.00**
% of additional resources	0.30	0.92	0.37	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

BSCANE2: 7-series FPGA Boundary Scan User Instruction
ICAPE2: 7-series Internal Configuration Access Port
WNS: Worst Negative Slack TNS: Total Negative slack
WHS: Worst Hold Slack THS: Total Hold Slack
System_i: Processing System (PS) instant
rm_dsp_int: Top level reconfigurable module
ila_inst: Integrated Logic analyzer inst
dbug_hub: debug core inside ILA