International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in; editorijoer@gmail.com

Vol.5., Issue.3, 2017
May-June

RESEARCH ARTICLE

ISSN: 2321-7758

BUILT IN SELF TEST FOR SYSTEM ON CHIP MEMORY

B.RAJ KUMAR[1], B.N.S MUNDA[2]

[1]Dept of Electronics and Communication, National Institute of Technology, Jamshedpur, India
rajkumarbhogavelli@gmail.com

[2]Associate Professor, Dept of Electronics and Communications, National Institute of Technology, Jamshedpur, India

ABSTRACT

Miniaturization and integration of different cores onto a single chip are increasing the complexity of VLSI chips. To ensure that these chips operate as desired, they have to be tested at various phases of their development. Built-In Self-Test (BIST) is one technique which allows testing of VLSI chips from wafer-level to system-level. The basic idea of BIST is to build test circuitry inside the chip so that it tests itself along with the BIST circuitry. The idea of current research is to develop BIST configurations for testing memory cores and other regular structure cores in System-on-Chips (SoCs). BIST approach for testing memory cores and other regular structure cores in FPGAs is described in this thesis. Another approach which takes advantage of some of the architectural capabilities of Atmel SoCs to reduce test time is also described in this thesis.

Keywords: Built-in self test (BIST), SoC, Output response analyzer (ORA), (Advanced Virtual Reduced Instruction Set Computer (AVR)

INTRODUCTION

Since the arrival of the first transistor-based computer, high scale integration became one of the main concerns in the hardware design techniques. In the early 1970's relatively high levels of integration were achieved, but the continuing effort to miniaturize and build more complex digital circuitry remained one of the goals in leading computer construction and chip design [1]. As a result, semiconductor integration has progressed from Small Scale Integration (SSI) to Very Large Scale Integration (VLSI) and now to System Level Integration (SLI) or System-on-Chip (SoC) [1].

System-on-Chip (SoC)

SoC technologies are the consequent continuation of the Application Specific Integrated Circuit (ASIC) technology, whereas complex functions, that previously required heterogeneous components to be merged onto a printed circuit board, are now integrated within one single silicon IC or chip [2]. As device integration scales grew, the enhanced performance of memory, microprocessors and logic devices boosted the performance of the digital systems they constituted. However, performance increases in larger systems were hampered by speed limitations associated with the long and numerous interconnects between devices on the printed circuit board (PCB) and associated input/output (I/O) buffers on the chips. Closely related system functions must be combined on a single chip to eliminate this bottleneck and take full advantage of improvements in transistor switching speeds and higher integration scales. This is precisely the capability that SoC technology provides. Rapid advances in semiconductor processing technologies allowed the realization of complicated designs on the same IC. SoCs can be broadly classified into two categories: ASIC-based and Configurable or Programmable. While the Configurable SoCs (CSoC) can be customized to

different applications through embedded reconfigurable logic cores, ASIC-based SoCs cannot be customized. CSoCs combine the advantages of both ASIC-based SoCs and multi-chip board development using standard components [1]. The major general goal for the development of such application-tailored reconfigurable architectures is to realize adaptivity vs. power/performance/cost trade-offs by migrating functionality from ASICs to multi-granularity reconfigurable hardware [3].

2. BIST Approach for Free RAMs Using Embedded Processor Core

The idea of this approach is to generate TPG signals from the embedded processor core. As a result, this approach is applicable only to the Field Programmable System Level Integrated Circuit (FPSLIC). The processor is also responsible for running the BIST, retrieving the BIST results, diagnosing the results and reporting back the diagnostic results to a higher controlling device (PC for example). The embedded processor in the FPSLIC can write into the configuration memory of the FPGA. This capability of the processor is used in combining the three RAM BIST configurations into one configuration. The free RAMs are initially configured in dual-port synchronous mode for running BIST. Then RAMs and FPGA logic are reconfigured to test RAMs in single-port synchronous and asynchronous modes. Thus, by avoiding two of the three downloads, testing time can be reduced significantly (approximately 3 times). Since only one bit-stream has to be stored instead of three, memory requirements are also reduced by a factor of three. The TPG is very irregular in structure. The rest of the circuit containing ORA and RAMs and can be made regular. Thus, by making the BIST circuitry inside the FPGA regular, the entire BIST logic to be built inside the FPGA (RAMs, ORAs and interconnections) can be algorithmically configured by the processor. This further reduces testing time because no bit-stream needs to be downloaded into the FPGA.

2.1 BIST Architecture: The architecture used is similar to the one used in the previous approach except that the TPG signals are generated by the processor. In dual-port mode, as in the previous approach, each ORA compares two adjacent RAMs as shown in Figure. In single-port mode, each ORA compares data from RAM with expected data generated by the processor as shown in Figure

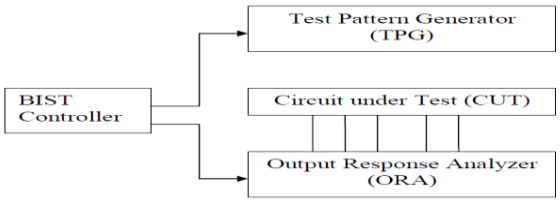


Figure 1: BIST Architecture

2.2 BIST Architecture for Single-port Modes : The BIST architecture for testing free RAMs in single-port synchronous and asynchronous modes is similar and is as shown in Figure. All RAMs are tested in parallel using a single TPG and the ORA compares data from RAMs with expected read data results generated by the TPG. The design of the single-bit ORA is shown in Figure. A tri-state buffer is required in this design as the write-data lines are used for both reading and writing data in single-port mode. The active high tri-state buffer in the ORA passes TPG data through when writing into the RAM and is tri-stated when reading from the RAM which allows the read data to be compared with expected data from the TPG. The ORA design for single-port mode, though not as simple as dual-port design, makes diagnosis of RAMs much simpler. Such a design is not used in dual-port mode because the generation of expected results by the TPG is more complicated as data can be read and written at the same time and also routing resources are not sufficient to implement such a design.

2.3 BIST Architecture for Dual-port Synchronous Mode: The BIST architecture used for testing free RAMs in dual-port synchronous mode is shown in Figure 3.1(a). All RAMs are tested in parallel using a single TPG and the ORA is designed to compare outputs of two adjacent RAMs. All RAMs except those on the rightmost and leftmost columns are compared by two ORAs. Two TPGs are generally used for this kind of BIST architecture to make sure that TPG is not faulty. But the Finite State Machine (FSM) based TPG is too large to replicate and fit inside the device. Therefore, it is assumed that the logic and routing resources are known to be fault-free as a result of previously executed BIST for programmable logic and routing resources. All the ORAs are connected in the form of a scan chain to

shift the BIST results out on (Advanced Virtual Reduced Instruction Set Computer).

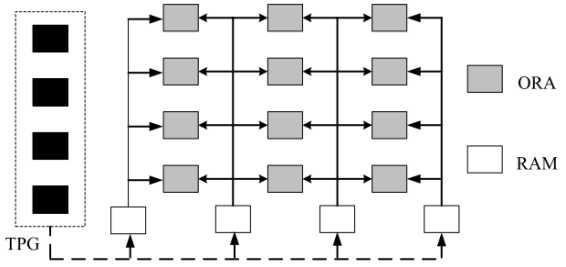


Figure 2: Dual-Port Free RAM BIST Architecture

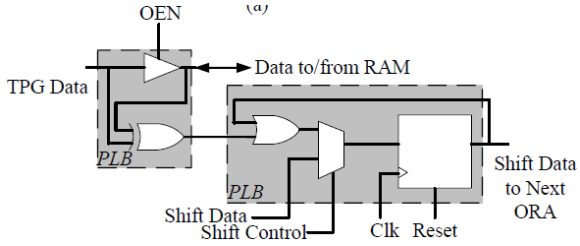


Figure 3: ORA Design.

BIST Approach for Free RAMs Using Embedded Processor Core The idea of this approach is to generate TPG signals from the embedded processor core. As a result, this approach is applicable only to the FPSLIC. The processor is also responsible for running the BIST, retrieving the BIST results, diagnosing the results and reporting back the diagnostic results to a higher controlling device (PC for example). The embedded processor in the FPSLIC can write into the configuration memory of the FPGA. This capability of the processor is used in combining the three RAM BIST configurations into one configuration. The free RAMs are initially configured in dual-port synchronous mode for running BIST. Then RAMs and FPGA logic are reconfigured to test RAMs in single-port synchronous and asynchronous modes. Thus, by avoiding two of the three downloads, testing time can be reduced significantly (approximately 3 times). Since only one bit-stream has to be stored instead of three, memory requirements are also reduced by a factor of three. The TPG is very irregular in structure. The rest of the circuit containing ORA and RAMs and can be made regular. Thus, by making the BIST circuitry inside the FPGA regular, the entire BIST logic to be built inside the FPGA (RAMs, ORAs and interconnections) can be algorithmically configured by the processor. This further reduces testing time because no bit-stream needs to be downloaded into the FPGA.

3. Implementation of BIST Approach

Initially free RAMs are configured to be tested in dual-port mode. The FPGAWE and FPGARE lines are used as clocks for running BIST and for retrieving BIST results, respectively. The Data bus is used for providing address, data and output enable signals to the free RAMs. Since the 8-bit wide data bus is not sufficient to provide all required signals.

On-Chip Diagnostics: AVR is not only capable of executing the BIST sequence and retrieving the BIST results but also capable of performing diagnostic procedures based on the BIST results for the identification of faulty RAMs in the FPGA core. The AVR, after running diagnostic procedures, identifies the location of the faulty RAM in terms of its X (column) and Y (row) coordinates. The AVR also identifies which bit(s) of the RAM is faulty. Since two different BIST architectures are used for testing free RAMs, two different diagnostic procedures were developed. In single-port test configuration, the ORA compare the expected results generated by the AVR with the data read from the RAMs Under Test (RUTs). Since the ORA incorporate a shift register, the BIST results latched in the ORA are retrieved by the AVR. Each bit retrieved corresponds to a single-bit of the 4-bit words of the RAM. The position of the ORA in the FPGA array, and the corresponding RAM with which it is associated, is determined by the ORA's position in the shift register. As a result of the ORA comparison of the RUTs output with the expected read results produced by the TPG, the diagnostic procedure for the single-port RAM modes of operation is straight forward. The diagnostic procedure looks for ORA failure indications (logic 1s) and translates the positions based on the shift register order to identify not only which RAMs are faulty but also which bits in a given RAM are faulty. Faulty ORA can mimic a fault in its corresponding RAM. This can be identified when PLBs are tested. In dual-port test configuration, since each ORA compares two adjacent RAMs a different diagnostic approach is used. The Multiple Faulty Cell Locator algorithm originally developed for diagnosing faulty PLBs in FPGAs is used for diagnosis of dual-port RAMs. This procedure is more complicated because it is possible that equivalent faults in two RAMs being compared by the same ORA will go undetected.

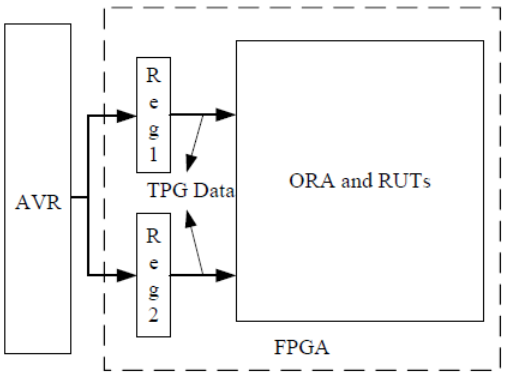


Figure 4: RAMBIST Implementation from AVR

Since all the RAMs except those at the leftmost and the rightmost edges of the FPGA are being observed by two sets of ORAs and being compared to a different RAM in each set of ORAs, it is highly improbable for the faulty RAMs to go undetected. This approach however loses diagnostic resolution for the RAMs at the leftmost and rightmost edges of the FPGA The goal was to develop BIST configurations for testing free RAMs in AT40K series FPGAs and AT94K series SOCs since they have embedded AT40K FPGA cores. Initially VHDL was used to design the BIST circuitry. This approach was useful only for pass/fail indication and not for diagnosis to indicate faulty RAMs due to lack of support from the synpaper tool for control of placement of RAMs relative to their associated ORAs. As a result, a combined VHDL-MGL approach was used to design the BIST circuitry. Three BIST configurations were developed to completely test free RAMs. The embedded microcontroller (AVR) in AT94K series SoCs can access the embedded FPGA core and can write into its configuration memory. This feature gave rise to an alternate BIST approach for SoCs. The AVR was used to control the BIST i.e., to start the BIST, retrieve the results after the BIST was completed and present the results to a higher controlling device (PC) which performed diagnosis based on BIST results. The same three BIST configurations were developed to test the free RAMs from the AVR. BIST circuitry implemented inside the FPGA can be made regular by moving the irregular TPG function into the AVR, leaving only the ORAs and RAMs in the FPGA. This gave rise to the possibility of combining the three BIST configurations into one. This was possible because regular BIST structure inside the FPGA is similar for all three configurations and can now easily be reconfigured by the AVR for the next mode of testing. Diagnosis was also moved from PC to AVR and thus a single configuration was developed which tests free RAMs completely and also performs diagnosis. A similar approach was used to test the embedded data SRAM shared by both AVR and FPGA.

Table 1: BIST and Diagnosis Summary

function	Execution cycles	Data memory(bytes)	Program memory(bytes)
BIST	100	464	18
diagnosis	28	332	33
total	128	796	51

Due to limitations imposed by the AVR architecture, three configurations were required to completely test the data SRAM. The VHDL-only approach did not yield any benefits for Atmel FPGAs. However, due to better synpaper tool support, the VHDL approach seemed worth experimenting on Xilinx FPGAs. This approach yielded good results on Xilinx FPGAs by controlling the placement of RAMs with respect to their associated ORAs. A portable VHDL code was thus created to test embedded block RAMs and LUT RAMs in all families of FPGAs from Xilinx. A total of 9 BIST configurations were developed for completely testing block RAMs in all families of FPGAs from Xilinx and another 3 configurations.

4. Results & Observations

It was observed that the architecture of an FPGA has a significant impact on BIST development. FPGAs using two different architectures were considered in this paper. Atmel FPGAs use fine-grained architecture as opposed to Xilinx FPGAs which use coarse-grained architecture. Such a problem can occur with coarse-grained FPGAs as well when logic or routing resources are used almost completely.

Placement and routing problems did not occur with Xilinx FPGAs when testing block RAMs. However, LUT RAM testing caused placement and routing issues, as almost 100% of logic resources were used. Routing issues were solved once placement of RUTs and ORAs were defined with a constraint file. TPG signals become heavily loaded, particularly when testing all the memory components in a large FPGA

with a single BIST configuration. All Xilinx FPGAs support boundary-scan with facilities for access to the FPGA core logic and this enabled usage of boundary-scan signals for downloading, running and controlling the BIST. This provides a common interface for BIST independent of the package being tested. Due to lack of access to the FPGA core by the boundary scan in Atmel devices, different I/O pins had to be used in different packages for running BIST. Atmel SoCs support writing into FPGA configuration memory but do not support reading of configuration memory or reading the contents of storage elements in the device. As a result, ORAs were required to be configured as a scan chain to shift out the results after running BIST. The length of the frames varies with the device and typically contains a few hundreds bits. Although Xilinx FPGAs have read-back capability, the rame-level segmentation makes read-back complicated, as post processing of results read back is required to extract the exact ORA data and, therefore, doesn't reduce testing significantly.

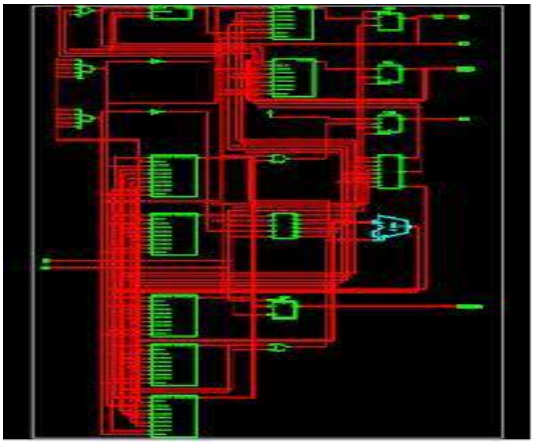
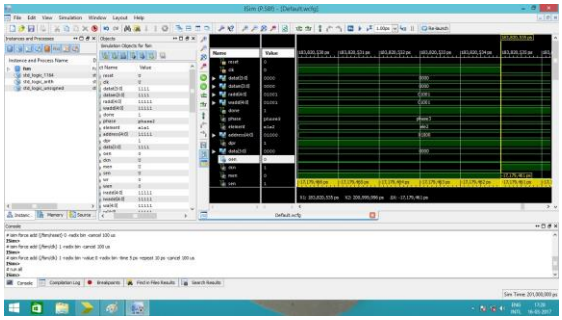


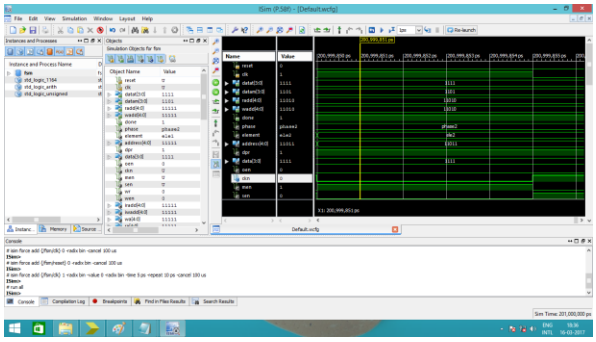
Figure 5: Schematic of ORA & RUT

4.1 Memory With Out Faults:

When data writing into the memory



4.2 Memory with faults:



5. Conclusions & Future scope

BIST configurations for testing memory components in commercially available FPGAs and SoCs are presented in this paper. Two different approaches were followed for developing BIST configurations to separately deal with two important features: portability of BIST development and testing time. BIST configurations developed were used to test memory components in AT40K series FPGAs and AT94K series SoCs from Atmel and Spartan II, Virtex II series FPGAs and SoCs from Xilinx. A summary of the paper, observations made during BIST development and suggestions for future research.

To conclude the paper, a few suggestions for improvements in the current BIST approach and also some areas that can be explored are discussed. Comparison with adjacent elements detects all possible faults in the RAMs except for the case where all elements have equivalent faults but fails to uniquely diagnose the results in cases where three or more adjacent elements being compared have equivalent faults. Comparison with adjacent elements was preferred over expected data comparison in some cases in this paper as the latter approach consumed more logic and routing resources and did not fit in some devices. Virtex II Pro SoCs have embedded Power PC microprocessors similar to the If a slice can be modeled using VHDL in such a way that the tool recognizes the model as a slice, BIST development can be reduced significantly by following the approach used for LUT RAM testing and logic BIST can be designed using VHDL alone and by controlling the physical placement of logic blocks and ORAs.

Bibliography

[1]. Arnaldo,B., “Systems on Chip: Evolutionary and Revolutionary Trends", 3rd International

Conference on Computer Architecture (ICCA'02), pp: 121-128, 2009.

[2]. J. Becker, "Configurable Systems-on-Chip (CSoC)", Proc. IEEE Integrated Circuits and Systems Design Symposium, pp: 379-384, 2002.

[3]. M. Rabaey, "Experiences and Challenges in System Design", Proc. IEEE Computer Society Workshop, pp: 2-4, 2010.

[4]. J. Becker and M. Vorbach, "Architecture, Memory and Interface Technology Integration of an Industrial/Academic Configurable System-on-Chip (CSoC)", Proc. IEEE. Computer Society Annual Symposium, pp: 107-112, 2009

[5]. S. Knapp and D. Tavana, "Field Configurable System-On-Chip Device Architecture", Proc. IEEE Custom Integrated Circuits Conference, pp: 155-158, 2000.

[6]. K. Kawana, H. Keida, M. Sakamoto, K. Shibata and I.Moriyama, "An Effcient Logic Block Interconnect Architecture for User-Reprogrammable Gate Array", Proc. IEEE Custom Integrated Circuits Conference,

[7]. H. Verma, "Field Programmable Gate Arrays", IEEE Potentials, Vol. 18, No. 4, pp: 34-36,

[8]. S.J.E Wilton, "Embedded Memory in FPGAs: Recent Research Results", Proc. IEEE Pacific Rim Conference, pp: 292-296.

[9]. S.J.E. Wilton, "Implementing Logic in FPGA Memory Arrays: Heterogeneous Memory Architectures", Proc. IEEE Field-Programmable Technology, pp: 142-147, 2002.

[10]. "International Technology Roadmap For Semiconductors (ITRS) 2000 Update. Technical Report", ITRS, 2000.

[11]. V. Ratford, "Self-Repair Boosts Memory SoC Yields", Integrated System Design, Sept 2001.

[12]. A. Benso, S. Carlo, G. Natale, P. Prinetto, and M. Bodoni, "Programmable Built-in Self-Testing of Embedded RAM Clusters in System-on-Chip Architectures", IEEE Communications Magazine, Vol. 41, No. 9, pp: 90-97, Sept 2003