

Soft Core Embedded Processor Based Built-In Self-Test of FPGAs

Bradley F. Dutton and Charles E. Stroud

*Dept. of Electrical and Computer Engineering
Auburn University, Alabama
duttobf@auburn.edu strouce@auburn.edu*

Abstract

This paper presents the first implementation of Built-In Self-Test (BIST) of Field Programmable Gate Arrays (FPGAs) using a soft core embedded processor for reconfiguration of the FPGA resources under test, control of BIST execution, retrieval of BIST results, and fault diagnosis. The approach was implemented in Xilinx Virtex-5 FPGAs but is applicable to any FPGA that contains an internal configuration memory access port.¹

1. Introduction

Built-In Self-Test (BIST) for Field Programmable Gate Arrays (FPGAs) exploits the reprogrammability of FPGAs to create BIST circuitry in the FPGA fabric during manufacturing and system-level off-line testing [1]. The only overhead is the external memory required to store the BIST configurations along with the time required to download and execute the BIST. No area overhead or performance penalties are incurred in the user function because the BIST logic is replaced by the intended system function after testing is complete. The BIST configurations are applicable to all levels of testing because they are independent of the intended system function and require no specialized external test fixture or equipment. Most research and development in BIST for FPGAs has focused on reducing the number of test configurations, reducing the size of test configuration files, and decreasing BIST execution time [2]-[8]. But the ever increasing complexity and level of integration in FPGAs has, with few exceptions, resulted in longer test times, more downloads, and more memory required for storing BIST configurations for each new generation of FPGA. However, the increasing size and complexity of FPGAs have also created opportunities for innovation in FPGA testing.

This paper presents the first implementation of BIST for FPGAs using a soft core embedded processor synthesized into the fabric of the FPGA under test. The approach reduces the number of configuration files required for BIST by exploiting the regularity of BIST structures to significantly compress and store partial configuration data in the embedded processor's program memory. The embedded processor controls and executes the BIST sequence, including retrieval and analysis (fault diagnosis) of BIST results, and reconfiguration of the FPGA for subsequent BIST configurations. This embedded processor based BIST approach is possible for two reasons: first, the growing size and complexity of FPGAs facilitates the inclusion of complex circuitry that only occupies a small percentage of the total configurable resources, leaving adequate area for BIST logic; and, secondly, the ability to access the configuration memory from inside the FPGA fabric has made possible internal reconfiguration and read back. The approach has been successfully implemented in Xilinx Virtex-5 but is applicable to any FPGA with internal configuration memory access.

2. Background

A number of BIST approaches have been developed for the configurable logic and memory resources in FPGAs [1]. Due to the programmable nature of resources to be tested, all BIST approaches for FPGAs require multiple configurations in order to obtain high fault

¹ This work was sponsored by the National Security Agency under contract H98230-04-C-1177 and supported in part by the National Science Foundation Grant CNS-0708962.

coverage. Generally, a BIST approach is organized into test *sessions* and *phases* [2]. Each test session consists of a set of test phases (test configurations) for a particular resource under test in order to test that resource in all modes of operation. For example, BIST of configurable logic blocks (CLBs) requires two test sessions. In the first test session, half of the CLBs are configured as blocks under test (BUTs), with the remaining half serving as comparison-based output response analyzers (ORAs) and test pattern generators (TPGs). In recent CLB BIST approaches, the TPGs are implemented in non-CLB resources freeing CLBs to function as additional ORAs such that circular comparison can be implemented, as illustrated in Fig. 1, where the outputs of each BUT in a row or column are monitored by two ORAs and compared to the outputs of two other identically configured BUTs [1]. This circular comparison in conjunction with multiple identically configured TPGs provides high diagnostic resolution with low probability of fault escape. In the second test session, the positions of the BUTs and ORAs are swapped, such that every CLB is configured as a BUT in one test session and as ORA in the other test session.

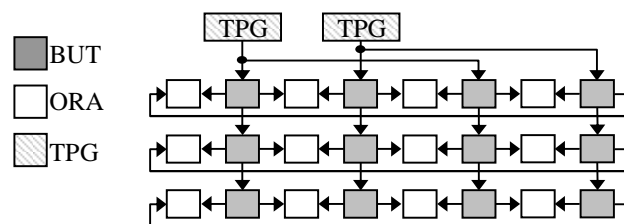


Figure 1. Configurable logic block (CLB) BIST architecture

BIST control, including downloading the initial BIST configuration, executing the BIST sequence, retrieval of results, fault diagnosis based on failing results, and reconfiguration of subsequent BIST phases, has traditionally been achieved via interface to an external BIST controller. However, the increased complexity of FPGAs, large number of test configurations associated with various programmable resources, and speed limitations of external download interfaces result in long manufacturing test times and limit practicality of system-level testing. Various approaches have been investigated to reduce the overall test time while achieving high quality tests. Beyond minimizing the number of test phases, partial reconfiguration reduces test time by reconfiguring only the resources under test for various modes of operation once the overall test structure has been downloaded into the device. BIST configurations that have been recently developed for Virtex-4 and Virtex-5 FPGAs include a single-bit pass/fail output to eliminate retrieval of ORA contents for passing test phases or when fault diagnosis is not desired [5]-[8]. When failures are observed, partial configuration memory read back can be used to obtain the ORA contents to diagnose the faulty resource(s) for fault tolerant applications. Beyond these techniques, the only new development in FPGA BIST has been introduction of embedded processor based approaches.

Prior work in embedded processor based BIST includes system-on-chip (SoC) testing with hard core microprocessors [9] but did not address testing of FPGAs or FPGA cores in SoCs. The first embedded processor based BIST approach for FPGAs was developed to minimize test time, number of downloads, and complexity of the external BIST controller by relocating BIST reconfiguration, control, and diagnosis to the dedicated hard core embedded processor in the Atmel AT94K series configurable SoC [3][4]. The device consists of an FPGA core, various RAMs, and an 8-bit Advanced Virtual RISC (AVR) microcontroller [10]. Sets of BIST configurations were developed to test each of the various programmable resources in the FPGA core including CLBs, RAMs, IOBs, and programmable routing network [3][4]. The embedded processor was used to configure the FPGA for each test session, execute the BIST sequence, retrieve BIST results from the ORAs, and perform diagnosis based on failing BIST results. This embedded processor based BIST approach achieved a total test time speed-

up of about 43.5 over the tradition approach of downloading each BIST configuration [4]. External memory requirements for storing BIST configurations were reduced by a factor of about 158 because only a single program needed to be downloaded into the AVR program memory, from which all BIST configurations were generated and executed.

While this embedded processor based BIST approach was practical for system-level testing, the approach was developed specifically for AT94K devices such that application to other FPGAs is limited due to reliance on the hard core processor with dedicated program memory. Some hardcore processors (such as the PowerPC in Virtex-4 and Virtex-5 FX series FPGAs) do not have a dedicated program memory and must use programmable resources in the FPGA. Soft core processors, on the other hand, can be implemented in most FPGAs such that a soft core processor based approach would be applicable to a wider range and variety of FPGAs and applications. The primary requirement is that the FPGA include an internal configuration access port (ICAP) to provide processor access to the configuration memory.

The configuration memories of Virtex-4 [11] and Virtex-5 [12] FPGAs are partitioned into frames, where each frame has a fixed length of 1,312 bits, or forty-one 32-bit words. A frame is the smallest addressable segment of the configuration memory; therefore all memory read/write operations must be performed on whole frames. This means that individual FPGA resources cannot be reconfigured without also providing explicit reconfiguration data for other FPGA resources that occupy the same frame. In Virtex-5, a frame contains the configuration data for 20 rows of CLBs and (I/O) tiles, or 5 rows of block RAMs and DSPs tiles in the same column. In Virtex-4, a frame contains configuration data for 16 rows of CLBs and I/O tiles, or 4 rows of block RAMs and DSP tiles.

Both Virtex-4 and Virtex-5 FPGAs include several configuration registers to access the configuration memory, including Frame Address Register (FAR), Frame Data Register Input (FDRI), and Frame Data Register Output (FDRO) which facilitate writing/reading data to/from a specific frame of configuration memory. There are other registers for functions such as status (STAT), cyclic redundancy check (CRC), command (CMD), etc. To access the configuration memory, a combination of these registers must be used. These registers are normally accessible from both Boundary Scan and SelectMAP configuration interfaces but are also accessible via the ICAP located inside fabric. The ICAP works like the external SelectMAP configuration interface except that it has separate 32-bit write and read buses, as opposed to a bidirectional 32-bit bus. The maximum ICAP clock frequency is 100 MHz.

3. Embedded BIST Architecture

The soft core embedded processor based BIST approach for FPGAs incorporates additional logic in the FPGA fabric along with the BIST logic to perform tasks typically assigned to an external BIST controller or computer. The embedded BIST approach offers several advantages over the external BIST approach. First, the 32-bit ICAP configuration interface is used for reconfiguration, eliminating the test time penalties associated with the lower speed serial Boundary Scan interface. Secondly, the total number of external download configurations is reduced to one per test session. In addition, all control of the BIST configurations and sequences can be implemented in the embedded controller. Diagnostic procedures can also be performed by the embedded BIST controller, further reducing the complexity of the external BIST controller in fault tolerant applications and providing considerable speed-up when compared to Boundary Scan based read back and diagnosis.

The implementation of the embedded processor BIST approach in Virtex-5 FPGAs incorporates elements of both hardware and software design to achieve an architecture that is general enough for any Virtex-5 device as well as for any BIST approach for the resources in Virtex-5 FPGAs. The design is applicable to any Virtex-5 device with only minor modifications to system software and no modifications to system hardware. Furthermore, the

design can easily be extended to Virtex-4 devices for similar improvements in test time. To minimize the number of external downloads per test session, the embedded processor based BIST hardware must fit in one half of the smallest supported device. The embedded processor core must also be capable of storing configuration data for all of the subsequent test phases for each test session in memory in the FPGA fabric using Block RAMs or distributed RAMs. Finally, the core must support interfaces for connecting with the ICAP and BIST circuitry. There are a variety of designs which can be used for the embedded processor ranging from fast, full-custom register transfer level (RTL) designs, to highly configurable general purpose soft core microprocessors. While RTL level designs are useful for simple repetitive tasks, this approach is not very efficient for supporting multiple device architectures of a variety of BIST approaches. Such an approach requires a different hardware configuration for each device and for each BIST session, which requires a significant amount of hardware development time when compared with other, more general purpose software based approaches. Another option is to use a general purpose processor in the form of a “soft” intellectual property (IP) core. One of the simplest and most efficient general purpose architectures available for Xilinx FPGAs is the PicoBlaze 8-bit microcontroller [13]. The PicoBlaze occupies one block RAM and approximately 50 slices in Virtex-5 FPGAs – much less than half of an array in the smallest Virtex-5 device. The PicoBlaze is supported by a simple assembler and software simulator. However, the program memory in the PicoBlaze is limited to 1024 stored instructions and scratch-pad memory is limited to 64 Bytes. The 8-bit architecture also creates timing penalties when interfacing with the 32-bit ICAP port because each ICAP write requires a minimum of four PicoBlaze instructions of two clock cycles each. To improve timing for ICAP operations, a 32-bit architecture is best for embedded BIST applications in Virtex-4 and Virtex-5 devices. One IP core that meets the requirement for a BIST controller is the MicroBlaze soft core processor which is a highly configurable 32-bit general purpose RISC microprocessor for Xilinx FPGAs [14]. The MicroBlaze also includes an optional, pre-engineered, interrupt driven ICAP hardware interface. The MicroBlaze can be configured with up to 64 kB of program and initializable data memory in Virtex-5 FPGAs that is implemented in the FPGA fabric in Block RAMs. The processor can be modified by the addition of custom peripherals on the processor local bus (PLB). These features led to selection of MicroBlaze as the embedded processor in our implementation.

The basic architecture for the embedded processor BIST approach is illustrated in Fig. 2 for CLB BIST where half of the FPGA array is used for processor and additional hardware resources and the other half of the array contains the CLB BIST configuration. Custom memory-mapped registers are included in the MicroBlaze VHDL model for interfacing with the BIST circuitry. The processor interfaces directly with the ICAP for reconfiguration of the BIST array and read back of BIST results. To test all CLBs in the FPGA, a second configuration is generated with locations of BIST logic and embedded processor swapped, as shown in Fig. 2b. For some resources, such as I/O tiles or CRC modules, it is possible to test all of the resources simultaneously by placing the MicroBlaze around the BIST circuitry.

One memory mapped write-only (WO) register, shown in Table 1, is included for control of the BIST circuitry and sequence. The outputs of the register are connected directly to inputs to the BIST logic, but not all of the register bits are utilized in any one BIST session. One read-only (RO) register, also shown in Table 1, is included at the same memory-mapped address as the output register. The inputs to this register are connected directly to outputs of the BIST logic. Each register is general enough to be applicable to all BIST configurations that have been developed for Virtex-5.

Because many BIST configurations must be executed for a different minimum number of clock cycles to achieve the intended fault coverage, there is the need for a hardware timer for BIST execution. Therefore, a 16-bit down counter is included in addition to the RO and WO

BIST control registers. The counter is initialized by writing to the lower 16-bits of the BIST control register. The counter automatically counts down to zero, setting the *cnt_eq* bit when zero. The *cnt_eq* bit is used to enable the BIST logic and can be polled in software to determine when the BIST is complete. The counter clock and BIST clock can share the MicroBlaze clock or can be clocked independently at a higher clock frequency by connecting the BIST clock input and 16-bit counter clock to an independent BIST clock source.

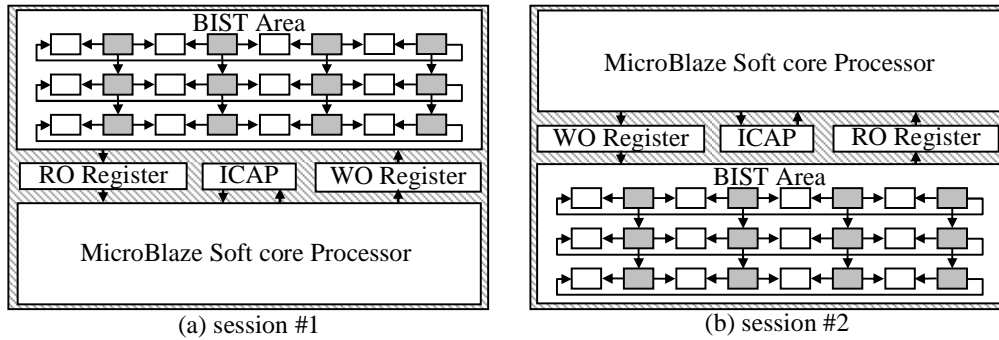


Figure 2. Embedded soft core processor based BIST architecture

Table 1. BIST control registers

Write-only register								Read-only register			
31:24	23:21	20	19	18	17	16	15:0	31:3	2	1	0
control	reserved	done	reset	start	tdi	clk_en	cnt_init	reserved (read as 0)	cnt_eq	BIST_done	tdo

4. Software Development

One important feature of this BIST approach stems directly from the generality of the embedded processor. Namely, that only the software changes from one BIST session to the next; the hardware remains unchanged for any and all BIST sessions. The software can be efficiently constructed in a manner that exploits the regularity of BIST configurations, and only the code for a particular BIST session need be compiled and programmed in the MicroBlaze program memory since a new download is performed at the start of each test session. Each BIST for a specific resource is composed of a set of phases, with each phase corresponding to a reconfiguration of the resources under test. Each phase comprises writes of entire set of frames of data to configuration columns that control the resources under test. Therefore, only certain portions of the partial reconfiguration files must be stored because the array-half, row, and column locations of the resources under test can be determined algorithmically based on the particular device in which BIST is implemented. The algorithm for the embedded BIST reconfiguration process is shown in Fig. 3. The algorithm for frame address generation using multi-frame write operations, given configuration row and destination minor address for frame data previously written to FDRI, is also shown in Fig. 3.

No modification to the MicroBlaze hardware is required for support of other BIST approaches such as DSP and Block RAM BIST [5][6]. However, the control bits [31:24] of the WO BIST register are used during these test sessions to control the TPG mode. The outputs of these register bits are connected directly to the mode control inputs of the TPG when the MicroBlaze hardware and BIST hardware are merged. Block RAM and DSP embedded BIST architectures are otherwise arranged identically to the CLB BIST architecture shown in Fig. 2 with the BIST circuitry occupying one-half of the device.

Reconfiguration files are generated in a manner that allows full or partial reconfiguration from an external memory without the need for an “intelligent” controller. While ideal for systems containing only non-volatile memory and an FPGA, the partial reconfiguration files are too large to be directly stored in the program or data memory of an embedded processor. For example, the total size of the 5 partial reconfiguration files for CLB BIST in half of an

array of a small Virtex-5 device (LX30T) is 41,360 Bytes – exceeding the maximum 32 kB of data memory that can be allocated for MicroBlaze. Partial reconfiguration files are also device dependent since the size of the reconfiguration file is proportional to the device size. Hence, compression of partial reconfiguration files is required for the embedded processor.

Overall BIST algorithm	Addressing algorithm w/ multi-frame write
<pre> for all test phases do for all configuration <i>rows</i> in BIST <i>half</i> do for all frames in reconfiguration structure do construct configuration frame multi-frame write to all RUTs in <i>half</i> & <i>row</i> end for end for execute BIST phase get BIST results end for set done bit in WO control register </pre>	<pre> for all configuration columns do if column is block under test then for all minor addresses in compressed config do multi-frame write to <i>row</i>, <i>column</i>, & <i>minor</i> end for end if end for </pre>

Figure 3. Embedded processor BIST algorithms

Our compression scheme exploits four features of Virtex-5 partial reconfiguration files to compress the data for storage in the embedded processor program memory and eliminate device dependencies. First, each configuration file contains certain instructions, such as those for multi-frame writes to the configuration memory, which are repeated many times during download. Since, in the embedded processor BIST approach, the download is executed under the control of the embedded processor, ICAP instructions can be stored once and regenerated when needed. Second, multi-frame writes can only occur in one configuration row in Virtex-5 devices. For BIST configurations, which create identical configurations in BUTs and ORAs in every configuration row, the overhead of multi-frame write instructions can be eliminated by storing frame data only once for one configuration row; the structure of the partial reconfiguration file can be regenerated by repetitively writing the frame data and frame addresses to the ICAP inside of a software loop for all configuration rows. Third, because one frame of configuration data spans 20 rows of identical resources under test, 2 to 4 words of frame data are repeated 10 to 20 times (in a repeating sequence) in each 41-word frame for BIST. Therefore, only 2 to 4 words of configuration data need to be stored for each frame in the partial reconfiguration file. The frame can be reconstructed in its entirety from the smallest repeating set of 32-bit words. Finally, the partial reconfiguration file includes the addresses of every frame to which frame data must be written for each configuration row. Again, due to the regularity of the BIST structure, only the minor addresses in the first BUT column for each configuration frame need to be stored. The remaining addresses can be regenerated algorithmically (Fig. 3) given the locations of resources under test in the FPGA fabric. We constructed a program to automatically extract only the essential data from every partial reconfiguration files for any BIST session using the compression methods described above. The program generates a C header file with a data structure containing only essential data from the compressed partial reconfiguration file. The data structure declaration is shown in Fig. 4, where the constant NRECONFIG is the number of test phases for the BIST session.

When the compression program was used to compress the 5 partial reconfiguration files for a CLB BIST session in a Virtex-5 LX30T, the total size of the files reduced from 41,360 Bytes to 820 Bytes. Table 2 shows the size of the original compressed partial reconfiguration files and the size of the essential data in compressed form for different BIST sessions for Virtex-5. The original file size given in the table is for an LX30T and the size of the file will increase in proportion to the number of configuration rows in a given device. However, the size of the essential data in compressed format is independent of the device size. Fig. 5 illustrates these device dependencies of reconfiguration file sizes for the smallest and largest devices in each Virtex-5 family of devices (LXT, SXT, FXT, and TXT).

```

struct framedata {
    unsigned int numword;           // # of words
    unsigned int word[MAXWORD];   // config data
    unsigned int numminor;        // # of addresses
    unsigned int minor[MAXMINOR]; // minor addr
};
struct partialconfig {
    unsigned int numframe;        // # frames
    struct framedata frame[MAXFRAME]; // frames
} config[NRECONFIG] = {
// compressed frame data placed here by program
};
    
```

Figure 4. Compressed BIST partial reconfiguration structure in C

Table 2. Compressed partial reconfiguration data size

BIST Session	Number of BIST Sessions	Number of BIST Reconfigurations	Original File Size (Bytes)	Compressed Size (Bytes)
CLB East	2	5	41,360	820
CLB West	2	5	41,360	820
LUT-RAM	2	4	10,944	1,232
I/O Logic	1	5	11,308	1,236
I/O SerDes	1	8	94,432	2,680
CRC	1	1	4,716	184
DSP	1	9	28,836	1152
Block RAM	2	5	285,040	4920
ECC RAM	2	2	19,384	1200
FIFO	2	3	29,076	1800
FIFO ECC	2	1	9,692	600

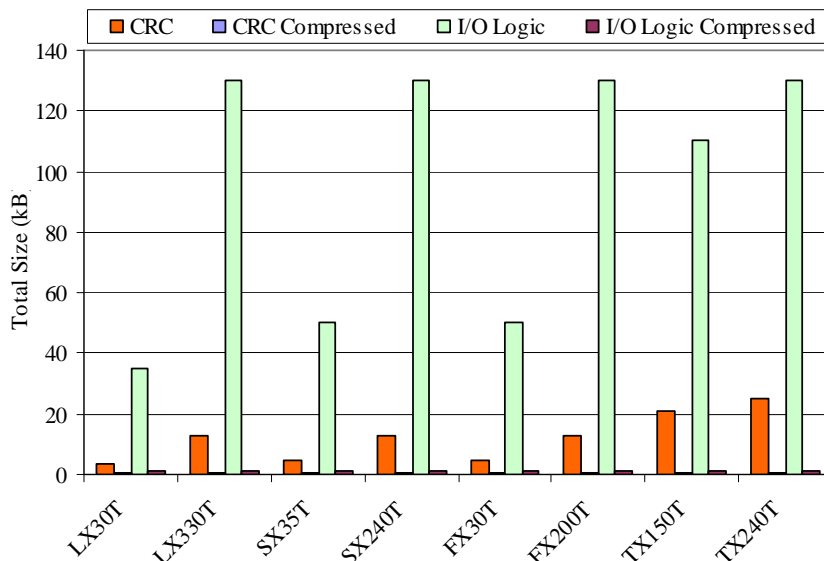


Figure 5. Original reconfiguration file sizes and compressed data structure sizes for one CRC BIST and a set of 5 I/O Logic BIST partial reconfigurations

Read back and diagnosis of BIST phases is performed by software in the embedded BIST processor when fault diagnosis is desired for a given application. The basic idea is to read back every frame of configuration memory containing an ORA flip-flop. The ORA flip-flop contents are then stored in an array in the processor data memory. An ORA contains a logic 0 when a failure is detected, otherwise a logic 1. Since the locations of ORAs are known for every BIST session in any device, the frame addresses of ORA flip-flops can be generated algorithmically during read back. The diagnostic algorithm [1] for circular comparison is easily implemented in the embedded processor to identify faulty resources. When combined with the 32-bit parallel access to the configuration memory, read back and diagnosis via the

embedded processor provides a substantial improvement in test time when compared to serial access via Boundary Scan.

5. Design Flow and Implementation Results

The embedded BIST processor design flow, illustrated in Fig. 6a, is more complex than the traditional BIST design flow due to the inclusion of the MicroBlaze processor and BIST session specific software. Generating the embedded processor based BIST configurations requires inputs from three sets of source files. First, the C source file for the specific BIST approach (e.g. CLB, DSP, block RAM, etc.) is compiled to an executable linkable file (ELF) format. The MicroBlaze hardware is modeled in VHDL and synthesized using the Xilinx ISE design flow. The placement of the MicroBlaze logic is constrained to one half of the device. The placed, unrouted design is then converted to Xilinx Design Language (XDL) format. The BIST logic is generated concurrently by the BIST generation program which produces an unrouted XDL description of the BIST circuitry. The placement BIST array is constrained to the other half of the FPGA. The BIST XDL description and the MicroBlaze XDL description are merged by concatenating the two XDL files and connecting primary inputs and outputs of the BIST logic to the WO and RO BIST control registers included in the MicroBlaze logic to form the complete unrouted embedded processor based BIST configuration in XDL format. Finally, the complete hardware portion of the design is converted to an NCD format and routed, from which the bitstream configuration file is generated using the Xilinx BitGen program. At this point, the compiled software in ELF format is translated into Block RAM initialization values in the bitstream download file using the Xilinx Data2Mem program. At this point, the compiled software in ELF format is translated into Block RAM initialization values in the bitstream download file using the Xilinx Data2Mem program.

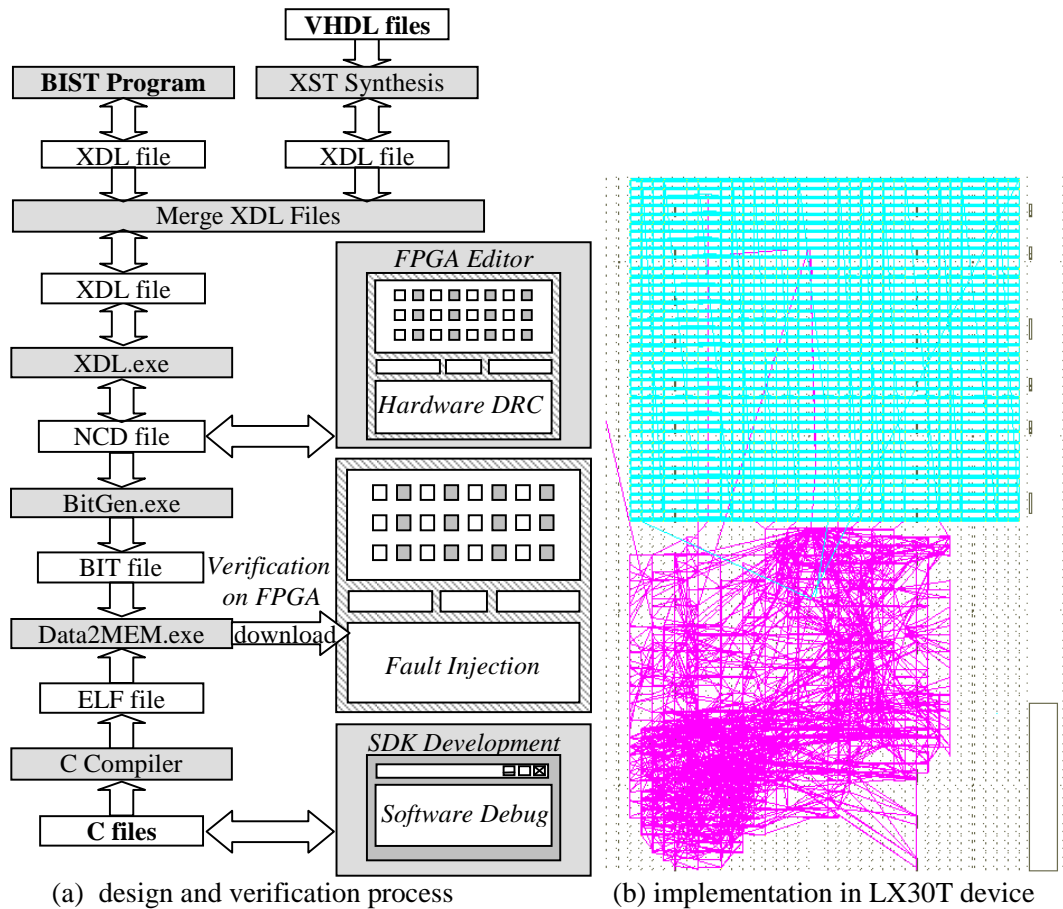


Figure 6. Embedded Process BIST design implementation

The embedded processor based BIST approach has been successfully implemented for BIST in Virtex-5 FPGAs. The unrouted embedded processor based BIST configuration for the CLBs implemented in the top of a Virtex-5 LX30T is shown in Fig. 6b. Two such configurations are implemented to fully test the CLBs with the locations of the BUTs and ORAs swapped, and another two configurations are required to test the bottom half CLBs. For the purpose of embedded BIST, the MicroBlaze processor is configured with a hardware integer multiplier, five stage pipeline, and 64 kB of on-chip program and data memory (configured in Block RAMs). In Virtex-5 devices, the MicroBlaze with ICAP interface and BIST control registers occupies three DSPs, 16 block RAMs, and 400 CLBs. The percentage of utilized resources is less than 50% in the smallest Virtex-5 device (LX20T). Timing analysis indicates that the maximum operating frequency of the MicroBlaze processor when constrained to one-half of a device is greater than 100 MHz in all Virtex-5 devices. Therefore, all ICAP operations can be performed at the maximum frequency of 100 MHz.

6. Conclusions

We have presented the first embedded soft core processor based FPGA BIST approach. The approach reduces the number of external configurations of the FPGA during any BIST session to a maximum of two (one for each half of the array); however, many resources can be tested in a single BIST session. The embedded processor performs reconfiguration of the resources under test at the maximum allowable clock frequency and data width. Read back of ORA contents can be performed when fault diagnosis is desired for fault-tolerant applications. The soft core processor approach was implemented in Virtex-5 FPGAs using the MicroBlaze processor. However, the overall approach is applicable to any FPGA with internal write and read access to the configuration memory.

References

- [1] L-T Wang, C. Stroud, and N. Touba, *System-on-Chip Test Architectures*, San Francisco, CA: Morgan Kaufmann, 2007.
- [2] M. Abramovici and C. Stroud, "BIST-Based Test and Diagnosis of FPGA Logic Blocks, *IEEE Trans. on VLSI Systems*, vol. 9, no. 1, pp. 159-172, 2001.
- [3] C. Stroud, S. Garimella and J. Sunwoo, "On-Chip BIST-Based Diagnosis of Embedded Programmable Logic Cores in System-On-Chip Devices," *Proc. ISCA Int. Conf. on Computers and Their Applications*, pp. 308-313, 2005.
- [4] J. Sunwoo and C. Stroud, "BIST of Configurable Cores in SoCs Using Embedded Processor Dynamic Reconfiguration," *Proc. Int. SoC Design Conf.*, pp. 174-177, 2005.
- [5] B. Garrison, et. al., "Built-In Self-Test for Memory Resources in Virtex-4 FPGAs," *Proc. ISCA Int. Conf. on Computers and Their Applications*, pp. 63-68, 2009.
- [6] M. Pulukuri and C. Stroud, "Built-In Self-Test of Digital Signal Processors in Virtex-4 FPGAs," *Proc. IEEE Southeastern Symp. on System Theory*, pp. 34-38, 2009.
- [7] B. Dutton and C. Stroud, "Built-In Self-Test of Configurable Logic Blocks in Virtex-5 FPGAs," *Proc. IEEE Southeastern Symp. on System Theory*, pp. 230-234, 2009.
- [8] B. Dutton and C. Stroud, "Built-In Self-Test of Programmable Input/Output Tiles in Virtex-5 FPGAs," *Proc. IEEE Southeastern Symp. on System Theory*, pp. 235-239, 2009.
- [9] R. Rajsuman, "Testing a System-On-Chip with Embedded Microprocessor," *Proc. IEEE Int. Test Conf.*, pp. 499-508, 1999.
- [10] "AT94K Series Field Programmable System Level Integrated Circuit," DS1138, 2005.²
- [11] "Virtex-4 FPGA Configuration User Guide," UG071(v1.1), Xilinx, 2008.³
- [12] "Virtex-5 FPGA Configuration User Guide," UG191(v2.7), Xilinx, 2008.³
- [13] "PicoBlaze 8-bit Embedded Microcontroller User Guide," UG129 (v1.1.2), Xilinx, 2008.³
- [14] "MicroBlaze Processor Reference Guide," UG081(v.9.0), Xilinx, 2008.³

² Available at www.atmel.com

³ Available at www.xilinx.com