Built-In Self-Test of Embedded SEU Detection Cores in Virtex-4 and Virtex-5 FPGAs

Bradley F. Dutton and Charles E. Stroud
Dept. of Electrical and Computer Engineering
Auburn University, Alabama

Abstract—*A Built-In Self-Test (BIST) approach is presented for the Internal Configuration Access Port (ICAP) and Frame Error Correcting Code (ECC) logic cores embedded in Xilinx Virtex-4 and Virtex-5 Field Programmable Gate Arrays (FPGAs). The Frame ECC logic facilitates the detection of Single Event Upsets (SEUs) in the FPGA configuration memory. The ICAP provides read and write access to the configuration memory from within the FPGA fabric, enabling embedded dynamic reconfiguration and fault-tolerant applications with memory scrubbing. Therefore, the fault-free operation of the ICAP and Frame ECC logic is critical for space and fault-tolerant applications that require detection and repair of SEUs. The BIST approach presented is applicable to all Virtex-4 and Virtex-5 FPGAs for both manufacturing and system-level testing of the ICAP and Frame ECC logic. The actual implementation of the BIST approach in Virtex-4 and Virtex-5 FPGAs and associated experimental results are discussed.* [1]

1 Introduction

The increased use of Field Programmable Gate Arrays (FPGAs) for implementing digital logic applications over the past two decades has been accompanied by increased concern about radiation effects; in particular, the effects of Single Event Upsets (SEUs). In addition to memory elements, such as flip-flops and random access memories (RAMs), the contents of the static random access memory (SRAM) used as the configuration memory to establish the overall application performed by the FPGA is also susceptible to SEUs. An SEU induced bit-flip in the SRAM configuration memory can alter the functionality of the FPGA. This makes SEUs of significantly more concern in FPGAs than in traditional application specific integrated circuits (ASICs). Radiation experiments indicate the SEU rate in FPGAs increased by a factor of 4.74 when design rules decreased from 600nm to 350nm with a corresponding reduction in Vcc supply voltage from 5V to 3.3V [1]. Xilinx Virtex-4 FPGAs are reported to have SEU FIT (failures in 10^9 hours) rates of 246 per million bits of configuration memory, and only 151 in Virtex-5 FPGAs [2]. This reduction in SEU FIT rate from Virtex-4 to Virtex-5 indicates that Xilinx is designing FPGA configuration memories to be more robust, as suggested in [3]. However, the largest FPGAs currently have configuration memories with up to 160 million bits [4]. As a result, some recent FPGAs, like Virtex-4 and Virtex-5, have incorporated additional logic that enables the detection of SEUs in the configuration memory. This logic can be used in conjunction with user-defined circuitry in the FPGA core to correct erroneous configuration memory bits that result from SEUs [5]. Approaches for on-line SEU detection and correction for Virtex-4 FPGAs have been proposed in [5] and [6] and for Virtex-5 FPGAs in [6] and [7]. All of these approaches assume that the embedded specialized cores for SEU detection, including the Internal Configuration Access Port (ICAP) and Frame Error Correcting Code (ECC) modules, are fault-free.

This paper presents an off-line BIST approach which completely tests the internal hardware mechanisms used for SEU detection and correction in the configuration memory of Xilinx Virtex-4 and Virtex-5 FPGAs. Since the FPGA is reconfigured for BIST only when testing is desired or required, there is no area or performance penalty incurred by the system application(s) normally executed in the FPGA. The only overhead for the BIST approach is the memory required to store one additional configuration used to configure the target device for BIST. The BIST approach is VHDL-based and is applicable to all production Virtex-4 and Virtex-5 devices. Furthermore, the BIST can be used for both manufacturing and system-level testing of the ICAP and Frame ECC logic. The paper begins with an overview of the ICAP and Frame ECC circuitry included in Virtex-4 and Virtex-5 FPGAs in Section 2. The test algorithm employed by the BIST approach to detect faults in parity-based ECC circuits is described in Section 3. Section 4 describes the method for generating and applying the test patterns to the ICAP and Frame ECC logic as well as the method used for output response analysis. Section 5 describes the actual implementation of the BIST approach in the fabric of Virtex-4 and Virtex-5 FPGAs along with experimental results. The paper is summarized and concludes in Section 6.

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.

2 FRAME ECC AND ICAP LOGIC

Like any RAM, the configuration memory of an FPGA is partitioned into words, also referred to as *frames*, which represent the smallest addressable unit of the configuration memory for write and read operations. Virtex-4 and Virtex-5 frames consist of 1,312 bits [8]-[11]. Each frame includes a 12-bit field of 11 Hamming bits and an overall parity bit for to provide the potential for single error correction (SEC) as well as double error detection (DED) in the frame data. The parity and Hamming bits are generated external to the FPGA by the configuration bitstream generation software and are subsequently downloaded with the application specific configuration data to the FPGA configuration memory. An overall cyclic redundancy check (CRC) performed on the device during the download verifies the integrity of configuration data during download. However, system memory data subject to change during the operation of the FPGA, such as contents of block RAMs and look-up tables (LUTs) used as distributed RAMs, are not covered by the overall parity and Hamming bits.

Virtex-4 and Virtex-5 FPGAs provide a specialized core, called Frame ECC, for detection and identification of single-bit errors and detection of double-bit errors in the frame data [9][11]. The Frame ECC primitive, illustrated in Figure 1, has 11 syndrome outputs, an error output, and syndrome valid output. Each time that a frame is read from the configuration memory the Frame ECC module calculates the Hamming bits as well as overall parity for the frame data, and compares these bits with the Hamming bits and parity stored for that frame in the configuration memory. Based on this comparison, the Frame ECC module produces indications for no error, single-bit error, and double-bit error in addition to a syndrome indicating the location of single-bit errors. System memory element contents (for example, block RAMs, LUT RAMs, and flip-flops) are masked from the internal parity and Hamming calculation by the Frame ECC. The error codes for the Frame ECC are summarized in Table I.

TABLE I. Frame ECC Codes

Error Type	Condition (when *syndromevalid* = 1)
No bit error	Hamming match w/ no parity error
1-bit correctable error (SEC)	Hamming mismatch w/ parity error
2-bit error detection (DED)	Hamming mismatch w/ no parity error

A Hamming mismatch with an overall parity error indicates that a single-bit correctable error has occurred. In this case, the bit-wise exclusive-OR of the stored Hamming code and the regenerated Hamming code, which is called the *syndrome*, gives the location of the single-bit error. A Hamming mismatch (non-zero syndrome) and no overall parity error indicate a non-correctable double-bit error has occurred. In the case of a double-bit error, the frame data must be repaired with data from a reliable external source. Single-bit errors in the configuration memory can be repaired with additional user logic implemented in the FPGA fabric to flip the bit in error as was done in [5], [6], and [7].

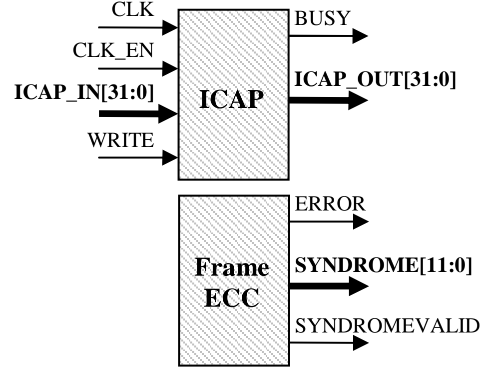

Figure 1. Frame ECC and ICAP Primitives

The SYNDROMEVALID output is asserted for one clock cycle per frame during a frame read operation to indicate that the SYNDROME and ERROR outputs are valid for the current frame [9][11]. The most significant bit of the SYNDROME[11:0] bus is the overall parity error indication. The ERROR output is asserted whenever a single-bit or double-bit error is detected. To distinguish between single-bit correctable errors and double-bit non-correctable errors, the user must add logic to determine the result based on the scenarios in the last two entries in Table I.

The ICAP provides access to status and control registers as well as to the configuration memory from the FPGA fabric [9][11]. The ICAP works like the external SelectMAP configuration interface except that it has separate 32-bit read and write buses, as opposed to a bidirectional 32-bit bus. The maximum operating frequency of the ICAP is 100 MHz, and it supports 8-bit, 16-bit, and 32-bit word sizes. Every device includes two ICAPs. However, both ports can not be used simultaneously. A bit in a control register is used to select whether the upper or lower ICAP is the active port.

3 TEST ALGORITHM

Hamming bits are parity calculated over a certain subset of bits in the configuration frame data. For example, the Hamming parity matrix in Table II can be extended to any number of data bits (D#) where the Hamming bits (H#) occupy the power-of-2 number locations in the counting sequence. Each Hamming bit is calculated by exclusive-ORing the data bits that have a logic 1 in the same row as that Hamming bit, yielding the logic equations shown in the lower half of the table for this example.

TABLE II. Hamming Parity Matrix Example

H1	H2	D1	H3	D2	D3	D4	H4	D5	D6	D7	D8	D9	D10	D11
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
$H1 = D1 \oplus D2 \oplus D4 \oplus D5 \oplus D7 \oplus D9 \oplus D11$														
$H2 = D1 \oplus D3 \oplus D4 \oplus D6 \oplus D7 \oplus D10 \oplus D11$														
$H3 = D2 \oplus D3 \oplus D4 \oplus D8 \oplus D9 \oplus D10 \oplus D11$														
$H4 = D5 \oplus D6 \oplus D7 \oplus D8 \oplus D9 \oplus D10 \oplus D11$														

As a result, the Frame ECC logic consists mainly of parity generators. A parity generator is simply an exclusive-OR tree, and can be arranged in linear tree or balanced tree forms; both arrangements are C-testable with four test patterns if *and only if* the exact parity tree construction and interconnections are known for every gate in the tree [13][14]. However, for cases where the parity tree structure is unknown, a pseudo-exhaustive test set to detect all gate level single and multiple stuck-at faults is: 1) walk a single one through a field of zeros, and 2) all combinations of two ones in a field of zeros [15]. This set of test patterns also detects all bridging faults in the Hamming generation circuit and overall parity generation circuit [16]. Therefore, the number of test vectors, N_{TV}, required in terms of the number of inputs, N, to test any parity generator (regardless of structure) is given by:

$$N_{TV} = \binom{N}{2} + N = \frac{N^2 + N}{2} \tag{1}$$

For the Virtex-4 and Virtex-5 Frame ECC logic, which calculates Hamming and parity over 1312-bits, the number of test patterns required by Equation 1 is $N_{TV} = 861{,}328$.

It is interesting to note that the parity calculations could be performed sequentially (32-bits at a time), as opposed to in parallel based on the entire 1312-bit frame. This leads to a significant reduction in the amount of logic for the calculation of Hamming code bits and overall parity. By masking appropriate bits from the parity trees (forcing bits to logic 0 using a mask LUT in conjunction with AND gates) the entire set of calculations can be performed sequentially, one 32-bit word at a time, as illustrated in Figure 2. The sequential Hamming generator requires twelve 32-input parity trees (one for each Hamming bit and one for the overall parity bit) with the cumulative parity calculations stored in 12 flip-flops. The Hamming and overall parity bits stored in the middle word of the frame are latched for comparison with the regenerated bits to produce the syndrome and overall parity error. This sequential parity generation would require only about 372 XOR gates and 352 AND gates for the masks. Parallel calculation over the entire 1312 frame bits, on the other hand, would require approximately 8,516 XOR gates.

It is possible that the number of test vectors for the sequential Hamming and parity bit calculation circuit might be reduced from that given by Equation 1. However, the set of test vectors described previously will also ensure complete testing of the word counter, masking circuit, and flip-flops/latches used to perform the sequential Hamming calculation. This means the test pattern sequence is independent of the actual architecture of the Frame ECC circuit. In addition, the walking patterns in the set of test vectors will detect stuck-at and bridging faults in the ICAP.

4 BIST APPROACH

Our approach to testing the Frame ECC logic is to implement a customized embedded core in the FPGA fabric that will repetitively write and read a single frame of configuration memory via the ICAP with the set of test patterns described in Section 3. The target frame for the BIST is arbitrarily located in the programmable interconnect network to avoid any configuration memory bits that are masked from the Frame ECC circuitry as a result of potentially legitimate changes to LUT-RAMs and flip-flop contents [9][11]. The basic procedure is as follows:

Step 1. Write a configuration memory frame with a test pattern via the ICAP.
Step 2. Read the frame containing the test pattern, compacting the ICAP output response.
Step 3. Compact the output response of the Frame ECC when the syndrome is valid.
Step 4. Generate the next test pattern and repeat Steps 1 through 3 for all 861,328 test vectors.

Even using the 32-bit ICAP interface, this test sequence is time-intensive because each frame write and read requires a significant amount of overhead in terms of clock cycles. In our implementation of the BIST, there are 318 clock cycles of overhead for each of the 861,328 test patterns. Therefore, the actually test time is 318 times the number of test patterns (as will be discussed in Section 4.1), or 273,902,304 clock cycles. However, the amount of logic that is tested is not insignificant, and the Frame ECC logic is critical for space and fault-tolerant applications that rely on the detection and correction of SEUs during on-line operation.

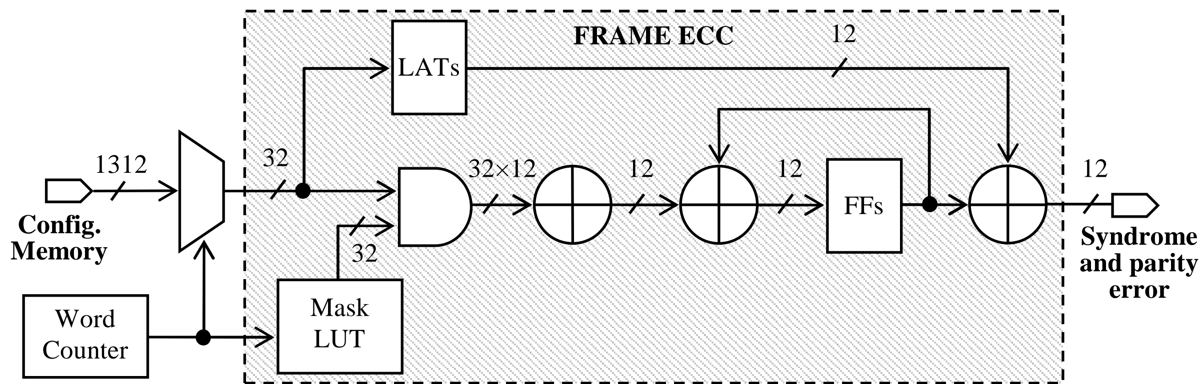

Figure 2. Sequential Hamming bit calculation

4.1 Test Pattern Generator

The test pattern generator (TPG) used to generate the parity tree test patterns is the largest component of the BIST architecture. It requires two 1,312-bit shift registers, 1,312 two-input OR gates, and a 32-bit 64-to-1 multiplexor array (the TPG is identical for both Virtex-4 and Virtex-5). In all, the TPG occupies about 1000 slices in Virtex-5 – 90% percent of all of the resources occupied by the BIST circuitry. Virtex-4 and Virtex-5 FPGAs incorporate several configuration registers to provide write/read access to the configuration memory. The Frame Address Register (FAR) stores the memory address to/from which frame data is written/read. The Frame Data Register Input (FDRI) and Frame Data Register Output (FDRO) registers facilitate input/output data to/from the configuration memory. There are other registers such as the status (STAT) register, the cyclic redundancy check (CRC) register, and the command (CMD) register which stores the next register operation to perform such as "Write FAR" or "Read FDR0". To write/read to/from the configuration memory, a combination of these registers must be used. In Virtex-4 and Virtex-5, the frame write and read instructions for the BIST are stored in a single 512×32-bit block RAM. The complete set of write and read instructions utilize about 10% of the Block RAM. The procedure for writing/reading to/from the configuration memory in the context of the BIST is illustrated in the pseudocode of Figure 3 and Figure 4, respectively.

```
Write_Test_Pattern (Test_Pattern, FRAME_ADDR){
        Write to Command RESET_CRC
        Write to ID Register DEVICE_ID
        Write to Command WCFG WRITE_CONFIG_MEM
        Write to Frame Address FAR FRAME_ADDR
        Write to Frame Data Input FDRI 82 words
        for(i=0; i<41; i++){
                Write word(i) of Test_Pattern
        }
        for(i=0; i<41; i++){
                Write pad word 0x00000000
        }
        Write NO-OP
        Write NO-OP
        Write to CRC 0x0000DEFC
}
```

Figure 3. Test pattern write sequence via ICAP interface

```
Read_Test_Pattern (FRAME_ADDR){
        Write to Command READ_CONFIG_MEM WCFG
        Write to Frame Address FAR FRAME_ADDR
        Read Frame Data Output FDRO 82 words
        for(i=0; i<41; i++){
                // Discard pad frame
        }
        for(i=0; i<41; i++){
                // Enable MISR to compact output
                // of FrameECC and ICAP
        }
        Write NO-OP
        Write NO-OP
}
```

Figure 4. Test pattern read sequence via ICAP interface

In both Virtex-4 and Virtex-5, the frame address selected as the write/read destination for the test patterns cannot contain LUT-RAM or flip-flop configuration bits because these bit locations are masked in the Frame ECC logic during read back (due to the fact that these bits can change after configuration if the capture command is decoded via the configuration interface or if the capture input to the capture primitive is asserted) [9][11]. Additionally, no BIST logic or routing resources can be located in the target reconfiguration memory region. Otherwise, the test logic could overwrite and modify parts of its own architecture. To eliminate the risk of overwriting the configuration of BIST logic or routing, the target configuration memory frame is located in the routing resources in the leftmost column of I/O Tiles (however, any frame containing only routing resources and not utilized for the BIST logic could be used). In Virtex-4, the target configuration frame is arbitrarily located in the leftmost column of I/O Tiles in the 16 rows below the center line. In Virtex-5, the target configuration frame is arbitrarily located in the lower 20 rows of the leftmost column of I/O Tiles. To avoid the target frame resources, the BIST logic is physically constrained to the right half of the target device during placement and routing. Additionally, before synthesizing the BIST, the Block RAM contents may require a minor modification. The Block RAM contents are device dependent, since the correct device ID must be written to the ID register before data can be written to the configuration memory via the ICAP. This is to ensure that a configuration file formatted for one device is not written, by mistake, to the wrong device.

4.2 Output Response Analyzer

Since only one Frame ECC component is included in every Virtex-4 and Virtex-5 device, comparison-based output response analysis of identical blocks under test (BUT) is not possible. Furthermore, comparison with stored good circuit output responses is not practical, since the 861,328 12-bit syndromes could not be stored on the device. Instead, a 32-bit multiple input signature register (MISR) with internal feedback and primitive characteristic polynomial is employed to compact the Frame ECC output responses into a final signature. The MISR characteristic polynomial, $P(x)$, is given by:

$$P(x) = x^{32} + x^{28} + x^{27} + x + 1 \qquad (2)$$

At the conclusion of the BIST, the signature in the MISR is compared with the known good circuit signature stored in the BIST logic, producing a single-bit pass/fail output. Additionally, the MISR is configured in a scan chain such that the signature can be retrieved via Boundary Scan for comparison with the good circuit signature. Any mismatch of the good circuit signature and the signature obtained by the BIST indicates a faulty circuit response. It should be noted that all MISRs have some probability of signature aliasing and fault escape. Signature aliasing occurs when a faulty circuit produces the same signature as the fault-free circuit. However, signature aliasing is extremely unlikely for properly designed MISRs. The classical approximation for the probability of fault aliasing is 2^{-n}, where n is the degree of the MISR's primitive polynomial [17]. Therefore, the

probability of signature aliasing is approximately 1 in 4.3 billion for the 32-bit MISR described by Equation 2.

The ICAP is tested by adding another identical 32-bit MISR to observe the ICAP outputs during the BIST sequence. This MISR, which is enabled when the ICAP read input is asserted, will detect any stuck-at faults as well as any bridging faults in the ICAP inputs and outputs. The MISR used to detect faults in the ICAP uses a similar on-chip comparison with the known good ICAP signature to produce a pass/fail output that is logically ORed with the pass/fail output of the Frame ECC MISR and comparison circuit, as illustrated in Figure 5. A simultaneous test of the ICAP and Frame ECC is logical since the ICAP would almost certainly be used for any space or fault-tolerant application that actively detects and corrects SEUs. However, because each device includes two ICAPs, only one of the ICAPs may be tested per BIST configuration in our current approach. Both ICAPs can be tested by simply generating, downloading, and executing two BIST configurations that alternate between the two ICAPs. It may also be possible to modify the BIST architecture such that both ICAPs are tested during the same configuration by using the top ICAP for the first half of the BIST sequence and switching to the bottom ICAP during the remainder of the BIST sequence, for example. This would require two additional instructions to write a logic 1 to the ICAP_SELECT bit in the control register, enabling access via the lower ICAP.

4.3 Additional Logic

In addition to the TPG and MISRs, the BIST architecture includes a custom soft-core embedded processor to control the BIST sequence execution. The processor is modeled in VHDL and is implemented entirely in configurable logic blocks. It controls the ICAP read/write signal and clock enable, the TPG/Block RAM multiplexor select inputs, and the TPG clock enable. The processor also includes three counters for addressing the instruction Block RAM, the TPG multiplexor, and for frame read timing. A block diagram of the ICAP and Frame ECC BIST architecture, including (from left to right) the TPG, circuits under test (CUT) and MISR output response analyzers, is shown in Figure 5. The input/output behavior of the architecture is discussed in Section 5.

5 IMPLEMENTATION RESULTS

The entire BIST circuit is implemented in VHDL, and only one configuration download is required for the BIST application. Some minor architectural differences between Virtex-4 and Virtex-5 devices require changes to the VHDL model for the two families of devices. First, before writing to the configuration memory, a device ID check must be performed by writing the correct device ID to the IDCODE register. This prevents accidental configuration with a bitstream formatted for another device. Any attempt to write the configuration memory without a successful device ID check will cause the FPGA to attempt a fallback reconfiguration [9][11]. The device IDs are kept in a look-up table specific to Virtex-4 or Virtex-5 and are synthesized with the design as a constant. Second, the frame address register is formatted differently for Virtex-4 and Virtex-5, requiring a modification to the stored target frame address. Finally, the input/output ordering for the ICAP in Virtex-5 is byte-swapped, compared to the Virtex-4 ICAP. Therefore, we maintain two VHDL BIST models, one for Virtex-4 and one for Virtex-5 with each model supporting all devices within that particular family.

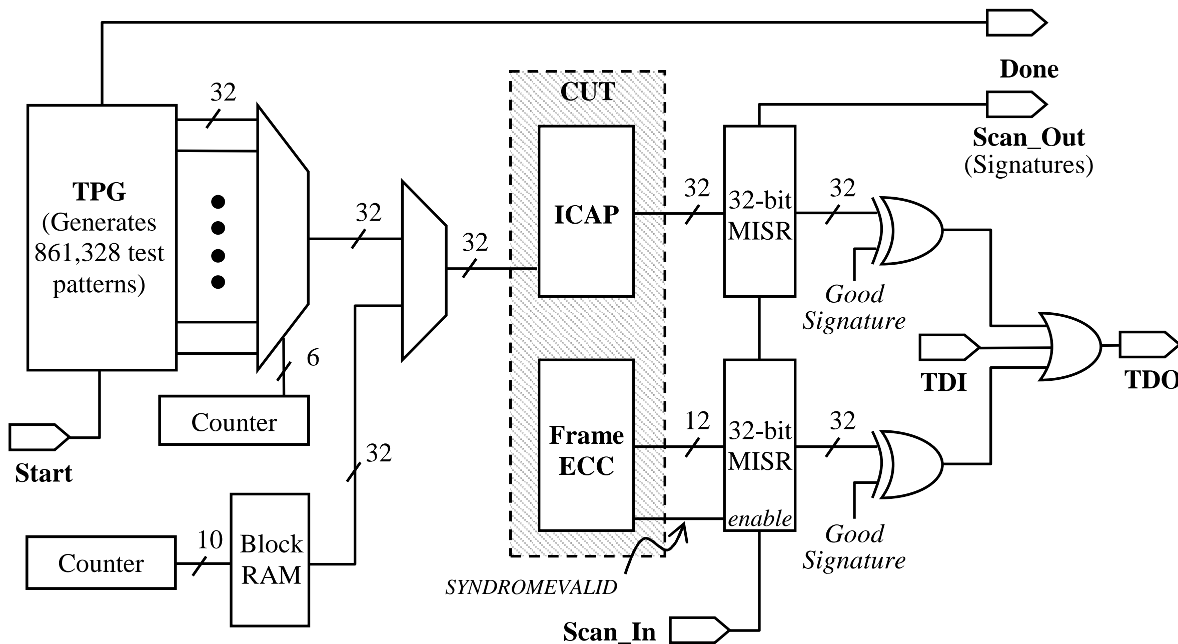

Figure 5. ICAP and Frame ECC BIST Architecture.

There are six primary inputs and three primary outputs for the BIST architecture. The VHDL component declaration illustrating these primary inputs and outputs of the BIST configuration is given in Figure 6. It should be noted that the four inputs associated with the MISR scan chain, *Scan_Clock*, *Scan_Mode*, *Scan_In*, and *Scan_Out*, are included only for design verification. Therefore, only three primary inputs and two primary outputs are required for a typical application.

The *Clock* input can be a free-running system clock or can be supplied by the Boundary Scan interface via TCK (DRCK internally). The maximum BIST clock frequency when the clock is supplied externally is 100 MHz, which corresponds to the maximum ICAP clock frequency. When the clock is supplied by Boundary Scan, the maximum BIST clock frequency is limited to 50 MHz which corresponds to the maximum TCK clock frequency. It should be noted, however, that the BIST logic in the FPGA fabric can actually operate well above the maximum configuration frequency of 100 MHz in all Virtex-4 and Virtex-5 devices based on timing analysis of the synthesized and routed design.

```
component Frame_ECC_BIST is
port(        Clock : in  std_logic;
               TDI : in std_logic;
             Start : in  std_logic;
           Scan_In : in std_logic;
         Scan_Mode : in std_logic;
        Scan_Clock : in std_logic;
               TDO : out std_logic;
              Done : out std_logic;
          Scan_Out : out std_logic);
end component Frame_ECC_BIST;
```

Figure 6. BIST VHDL Component Declaration.

The *Start* signal is an active-high, asynchronous signal which enables the execution of the BIST sequence. The *Start* signal should be asserted for a minimum of three cycles of *Clock* to begin the BIST sequence, but then may be de-asserted or may be left asserted. The BIST will start and run automatically to completion after download by tying the *Start* signal to logic 1 in the top-level VHDL model. Toggling the *Start* signal low and then high after the completion of the BIST will clear the MISRs and cause the entire BIST sequence to repeat. This feature can be used to check for reproducible BIST results during design verification. The *Scan_Mode* input places both MISRs in a scan mode. With *Scan_Mode* asserted, the *Scan_In* input is an optional input to the MISR scan chain, which can be used in conjunction with *Scan_Out* (the output of the MISR scan chain) for loading and retrieving signatures during design verification. The input *TDI* and output *TDO* provide a single-bit pass/fail result for the BIST. As illustrated in Figure 6, *TDI* is one input to a 3-input OR gate, with the other two inputs coming from the outputs of the MISR signature comparators. When both MISRs contain the good circuit signatures, *TDO* (the output of the OR gate) will equal *TDI*. However, if either MISR does not contain the good circuit signature, the output of the functional OR will be logic 1, regardless of the state of *TDI*. The *Done* output is asserted when the BIST sequence is complete. When the *Done* signal is asserted, the pass/fail result is valid on the *TDO* output. The BIST sequence, after download (and without tying *Start* to logic 1), is as follows:

Step 1. Assert the *Start* input.
Step 2. Wait for the *Done* signal to be asserted.
Step 3. Drive *TDI* low, poll *TDO* (should be logic 0).
Step 4. Drive *TDI* high, poll *TDO* (should be logic 1).

The BIST is interpreted as passing if the *TDO* output presents a logic 0 in Step 3 and a logic 1 in Step 4. This ensures that the *TDO* output is not stuck in the fault-free state due to a fault in the FPGA. Optionally, the contents of the two 32-bit MISRs may be scanned out and verified by external comparison to the known good circuit signatures.

The total execution time for the BIST with an external 100 MHz clock is 2.739 seconds. The BIST has been downloaded, executed and verified on Virtex-4 FX12, SX35, and LX60 devices and on Virtex-5 LX30T, LX50T, SX35T, and SX50T devices using both Boundary Scan and external clock and control. Due to the differences in the configuration interfaces, Virtex-4 and Virtex-5 produce different good circuit signatures, as reflected in Table III. Figures 7 and 8 show the ICAP and Frame ECC BIST implemented in the smallest Virtex-4 (FX12) and Virtex-5 (LX20T) devices, respectively. As can be seen in both figures, the BIST circuitry easily fits in programmable logic resources in the right hand half of the array. This shows that the BIST can be implemented in all other Virtex-4 and Virtex-5 devices, all of which have larger arrays than those illustrated in Figures 7 and 8. The target configuration frame areas that should be avoided by constraining the design placement are also illustrated in the figures.

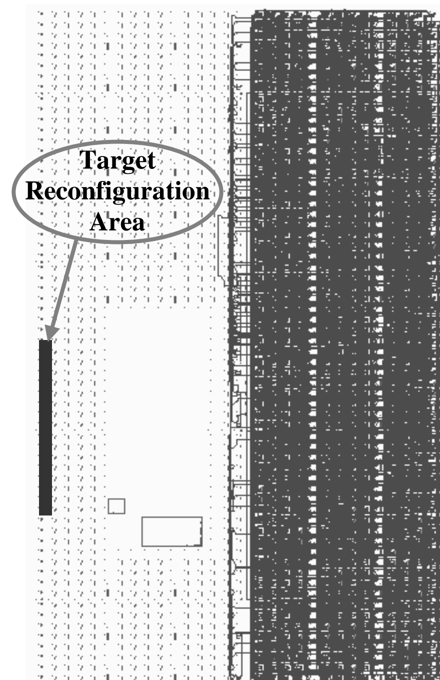

Figure 7. Virtex-4 FX12 with ICAP/Frame ECC BIST

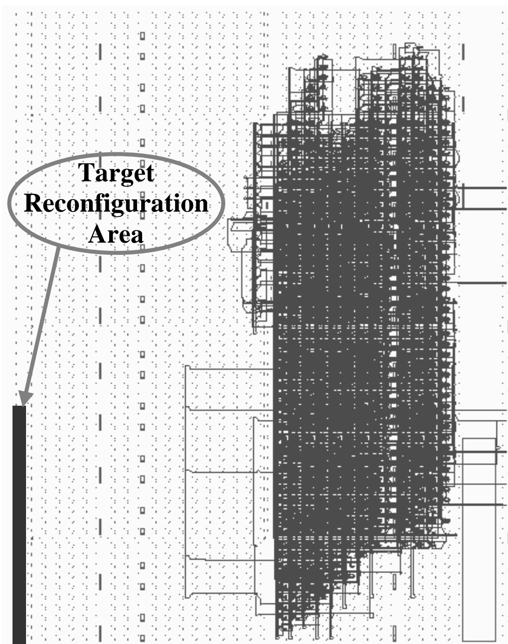

Figure 8. Virtex-5 LX20T with ICAP/Frame ECC BIST

Table III summarizes the actual implementation of BIST circuitry in Virtex-4 and Virtex-5 FPGAs. This includes the number of slices occupied by the BIST circuitry, the number of lines of VHDL code for the complete BIST circuit, and the total test time (excluding initial configuration time) at the maximum operating frequency of 100 MHz. The primary reason for the difference in the number of logic slices is due to the fact that Virtex-5 incorporates four 6-input LUTs and four flip-flops per slice while Virtex-4 slices incorporate only two 4-input LUTs and two flip-flops. As a result, a Virtex-5 slice has twice the logic of a Virtex-4 slice – hence, Virtex-4 requires at least twice the number of slices. The smaller LUTs in Virtex-4 account for the additional slices. The 32-bit good circuit signatures for the Frame ECC and ICAP modules are also included in Table III.

TABLE III. ICAP and Frame ECC BIST Summary

	Virtex-4	**Virtex-5**
# of logic slices	2546	1010
# lines of VDHL	1125	1125
Total test time	2.739 sec.	2.739 sec.
Frame ECC signature	0x9BC92CDB	0x969C47DD
ICAP signature	0xB3FFB18B	0x31D989BD

6 CONCLUSIONS

This paper has presented a BIST approach for the ICAP and Frame ECC modules in Virtex-4 and Virtex-5 FPGAs. These modules are critical components used for SEU detection and correction in the configuration memory of FPGAs for space and fault-tolerant applications. The BIST approach was developed in VHDL and is applicable to all Virtex-4 and Virtex-5 devices, and the only overhead is the memory required to store the BIST configuration and downtime for the test application. The total test time is independent of the size of the FPGA. However, when using compressed configuration bitstream files, the download time can vary with the size of the FPGA depending on the physical constraints applied during synthesis. The BIST can be periodically downloaded and executed in systems which rely on the Frame ECC and ICAP logic for on-line detection and correction of SEUs to guarantee the fault-free operation of these resources. The approach has been implemented, downloaded, and verified on a variety of Virtex-4 and Virtex-5 devices.

REFERENCES

[1] M. Ohlsson, P. Dyreklev and K. Johansson, “Neutron Single Event Upsets in SRAM-Based FPGAs,” *Proc. IEEE Nuclear and Space Radiation Effects Conf.*, pp. 177-180, 1998.

[2] A. Lesea, “Continuing Experiments of Atmospheric Neutron Effects on Deep Submicron Integrated Circuits,” WP286 (v1.0), Xilinx Inc., 2008.[2]

[3] A. Lesea, P. Alfke, “Xilinx FPGAs Overcome the Side Effects of Sub-90 nm Technology,” WP256 (v1.0.1), Xilinx Inc., March 2007. [2]

[4] *Virtex-6 Family Overview*, DS150 (v1.0), Xilinx Inc., 2009.[2]

[5] L. Jones, “Single Event Upset (SEU) Detection and Correction Using Virtex-4 Devices,” Application Note XAPP714 (v 1.5), Xilinx Inc., 2007.[2]

[6] B. Dutton and C. Stroud, “Single Event Upset Detection and Correction in Virtex-4 and Virtex-5 FPGAs,” *Proc. ISCA International Conf. on Computers and Their Applications*, pp. 57-62, 2009.

[7] K. Chapman and L. Jones, “SEU Stratagies for Virtex-5 Devices,” XAPP864 (v1.0.1), Xilinx Inc., March 2009.[2]

[8] *Virtex-4 FPGA User Guide*, UG070 (v2.5), Xilinx Inc., 2008.[2]

[9] *Virtex-4 FPGA Configuration User Guide*, UG071 (v1.1), Xilinx Inc., 2008.[2]

[10] *Virtex-5 FPGA User Guide* UG190 (v4.2), Xilinx Inc., 2008.[2]

[11] *Virtex-5 FPGA Configuration User Guide*, UG191 (v3.2), Xilinx Inc., 2008.[2]

[12] J. Heiner, N. Collins, and M. Wirthlin, “Fault-tolerant ICAP Controller for High-Reliable Internal Scrubbing,” *IEEE Aerospace Conf.*, pp. 1-10, 2008.

[13] D. Bossen, D. Ostapko, and A. Patel, “Optimum test patterns for parity networks,” *Proc. AFIPS Fall 1970 Joint Comput. Conf.*, pp. 63-68, 1970.

[14] W-B Jone and C-J Wu, "Multiple fault detection in parity checkers," *IEEE Trans. on Computers*, vol.43, no.9, pp.1096-1099, 1994.

[15] S. Mourad and E. McCluskey, “Testability of parity checkers,” *IEEE Trans. on Industrial Electronics*, vol. 36, no. 2, pp. 254-262, 1989.

[16] L-T Wang, C. Stroud, and N. Touba, *System-on-Chip Test Architectures*, San Francisco, CA: Morgan Kaufmann, 2007.

[17] C. Stroud, *A Designer’s Guide to Built-In Self-Test*, Boston, MA: Springer, 2002.

[2] Available at www.xilinx.com