Abstract - Complex, high performance multi-PCB, multi-bus, multi-FPGA electronic systems pose difficult challenges in design and prototype validation, and in production and field testing. High performance systems require at-speed test, in order to uncover static and dynamic faults. Self-test provides an attractive solution. Software-based and hardware-based BIST (built-in self test) can be used, namely for interconnections test. Hw-based solutions typically suffer from significant overhead. However, FPGA technology provides flexible, low-cost hw-based solutions, as unallocated device resources can be wisely used as TPG (Test Pattern Generators) and TAE (Test Answer Evaluators). The purpose of this paper is to present a new interconnect test methodology for complex multi-PCB, multi-bus and multi-FPGA electronic systems, aiming to uncover static and dynamic faults. On-chip ROM resources of the FPGA are configured to store deterministic test patterns (according to specific fault models). The test controller is also embedded in one of the FPGA devices. The interleave True/Complement algorithm is used to cover static faults (open, shorts and 2-nets coupling). For dynamic faults, the MA (Maximum Aggressor) fault model is used (glitches and delayed responses, due to cross coupling or other effects). A software tool has been developed to generate the test vectors needed to cover both types of faults. A case study, with 5 PCB and a back plane, 2 64-bit busses and 9 4-million FPGA is used to demonstrate the effectiveness of the proposed methodology.
dynamic faults, and even commercial solutions exist (see, e.g., [5]), the costs of TPG (Test Pattern Generation) and application are huge, as it requires long shift-in and shift-out operations, and fast test sequence application.

Interconnect testing has been long under research [7]-[10]. This has been performed to be accommodated in BST [9][10], even using at-speed testing [11]. Especially at chip level, interconnect (namely bus) testing has attracted much attention, and numerous methodologies and tools have been published, especially targeting crosstalk faults [12]-[18]. The majority of them use a hardware-based self-test approach. Some of them [14] use a software-based approach, in which an embedded processor core in a SoC (Systems on a Chip) tests for crosstalk effects in on-chip busses by executing a software program. The advantage is that no additional hardware is required (as it usually is required in hardware-based self-test approaches). However, using the on-board programmable hardware (the FPGA devices), a hardware-based self-test approach (such as the one we propose) may require no additional hardware.

The purpose of this paper is to present a new test methodology for complex multi-PCB, multi-bus and multi-FPGA electronic systems. Interconnect testing is addressed; however, the methodology can be used also for chip, board and system test. The methodology allows the detection of design or product, static or dynamic faults, and can be applied in design (or prototype) validation, and in production (or lifetime) testing.

The paper is organized as follows. In section 2, a review of previous work, as far as interconnect fault models are concerned, is briefly described. Section 3 introduces the interconnect test methodology and a software tool which has been developed to support it. In section 4, some results with a case study [23][24] with five PCB and a back plane, two 64-bit busses and 9 Xilinx™ xc2v4000-4bf957 FPGA (957 pins) are described to demonstrate the effectiveness of the proposed methodology. Finally, section 5 summarizes the main conclusions and points out directions for future work.

2. Interconnect Fault Models

In order to identify and screen out the most likely physical defects, and operation-dependent disturbances, which may lead to system errors, static and dynamic faults must be considered. Here, the concept of fault does not only encompass the impact of a physical defect on circuit behaviour, but also the impact of an operation dependent disturbance, such as crosstalk, on system behaviour.

2.1 Static Interconnect Faults

When testing a bus for static failures, it is common practice to consider open circuit’s faults and shorts between multiple wires. Open faults are usually modelled by single line stuck-at-1 (LSA1) or stuck-at-0 (LSA0) faults. Short-circuit faults are modelled, at logic level, by wired-AND (where logic 0 dominates) and wired-OR (where the logic 1 dominates) behaviour. In this work, the fault model used for the detection of such failures is the one proposed by Jutman [11]. The author suggests a test pattern generated by an algorithm referred as Interleaved True/complement sequence, as explained in section 3.2. Opens, shorts and 2-net coupling faults are covered.

2.2 Dynamic Interconnect Faults

In this work we first assume that dynamic faults in bus lines are transient (non permanent) faults, which are operation-dependent. In this fault type, we include those generated by cross-coupling capacitance and inductances between bus interconnect lines.

Crosstalk between a pair of interconnect lines can cause two distinct effects: a signal glitch and a delayed transition of the signal. These two effects depend on the type of signal transition on the lines in question, as shown in Figure 1 a) and b) respectively. Note that, by considering a delayed transition, this fault can describe the impact of (1) a crosstalk effect, or (2) any physical defect or operation-dependent disturbance which modifies the circuit timing response. Operation-dependent disturbances can result from thermal effects, circuit switching activity, or aging. This is why it is very relevant to perform interconnects delay tests during product lifetime.

![Figure 1- Glitch and delay faults in interconnect lines, in the presence of a coupling impedance, Z.](image)

In order to detect such faults, the Maximal Aggressor (MA) fault model has been proposed [13]. This model defines four types of faults, due to the effects generated by crosstalk errors: the positive glitch (g_p), negative glitch (g_n), the rising delay (d_r) and the falling delay (d_f). The interconnect line in which an erroneous logic value can occur is referred to as the victim. The remaining interconnect lines inducing the crosstalk error in the victim are described as the aggressors.

Figure 2 shows the necessary transitions in the victim and aggressors lines, in order to activate each of the four types of faults in victim line, \( Y_v \).

![Figure 2 - Maximum Aggressor tests for victim line \( Y_v \).](image)
Each conductive line can become a victim. Fault detection requires a two-vector sequence for each MA fault. However, some test vectors are identical. Hence, when fault detection of all four fault types is targeted, test vector compaction can be performed, in order to create a shorter test sequence. The total test length (TL) measures the number of individual test vectors required to complete a sequence, referred as the test pattern. For this fault model, $TL = 6N$ vectors are required for the detection of the four types of faults [17] (where N is the number of lines). According to [13], with this set of vectors, 100% crosstalk fault coverage can be achieved.

3. Interconnect Test Methodology

3.1 Test Methodology

The goal of interconnect testing is to verify that the physical infrastructure (which allows data and control signals to flow among complex devices) is defect-free and does not induce signal integrity problems. At system level, such infrastructure may encompass physical busses (sets of adjacent conductive layers running in parallel), transceivers and board connectors. The physical infrastructure connecting the various FPGA may exhibit different lengths, inducing variable transmission delays. Hence, bus protocols must guarantee that signal launch and capture (at the Bus I/O) leads to correct operation. The protocol, and its architecture, should, thus, also be tested at-speed.

At high-frequency, transceivers may cause signal reflections, which may modify signal delays. Connectors and device I/O pins may exhibit resistive contacts, whose resistance may vary with temperature. This effect may also cause abnormal signal delays. Communication paths (point-to-point or device to device communication) need to be tested, at-speed, and in a manner that resembles, as much as possible, normal operation. Hence, system clock must be used. The test pattern generation (TPG) process must use all fault models required to describe all likely effects which may compromise correct operation.

In a complex, multi-PCB, multi-bus and multi-FPGA electronic system, complex devices are programmable devices (FPGA). For interconnects test, device dialog must be tested. Hence, a specific FPGA device (emitter) can be programmed to act as TPG, and the receiver device as TAE (Test Answer Evaluator). Self-test control is also embedded in a specific FPGA system device. This is the first underlying principle of the proposed methodology.

The communication path between each pair of devices must be tested in both directions, as, in general, bidirectional signal flow may occur in normal operation. Hence, each FPGA in the system may act as emitter or receiver device (Figure 3). As the communication channels must deliver the same data they receive, both the emitter and receiver devices must store the same data. Hence, the content of the embedded memory (typically, a configured ROM) of each FPGA must be identical. Once the TPG process is carried out (and tuned to uncover the most likely interconnect faults), test data storage (test pattern) is identical, reused in each programmable device.

This is the second underlying principle of the proposed methodology.

When a specific FPGA acts as TPG, it broadcasts the test data in one or more system’s busses. System clock is used in test application. Data includes a header, which allows individual FPGA connected to the bus(es) to understand if they should read bus data, or not. When a specific FPGA is acting as TAE, it receives data from the bus (the test pattern) and compares it with the data stored in its embedded memory. Moreover, in a multi-bus environment, different busses may have different widths. Hence, the FPGA configured ROM must have a width which corresponds to the system’s bus with the largest width.

In the interconnect test mode, one of the FPGA devices will to act as master, being configured to have the off-line BIST controller. Being the master does not necessarily means that it will act as TPG; instead, it will control the test session and provide its result.

Depending on the purpose of the test process, additional functionality may be included in the FPGA firmware. For instance, if diagnosis is targeted, a given FPGA may collect information not only on pass/fail, but also on which communication path fails, and which are the failing vectors. Also, it may identify if the failure occurs every time the test is run, or if it starts failing after running the test in loop for a long time (due to power consumption problems), and so on.

The use of an FPGA memory, like ROM, for test pattern storage is a very important advantage of the proposed methodology. In fact, not only it uses usually unallocated FPGA resources, but also it adds flexibility to the test process. With a single firmware, different types of tests can be performed, allowing the detection of different types of faults (and this increasing test quality), by simply changing the contents of the FPGA embedded memory.

The use of the set of FPGA devices as an embedded tester can be envisaged in two scenarios. First, if unassigned resources are available, ROM and any additional test functionality can be configured with the mission functionality. In such scenario, interconnects test sessions can be off-line launched whenever the system is idle. Second, the test functionality (including test pattern storage) may be configured as an additional configuration, just for validation, debug and test purposes, even in an integrated test solution for device, board, interconnects and system test. In this second
scenario, the mission functionality is configured in all FPGA devices after the test scenario is used.

### 3.2 Test Generation for Interconnect Faults

In Figure 4, several test pattern generation techniques to uncover static interconnect faults are depicted for a 4-bit bus. The rationale for dynamic faults TPG is shown in Figure 2.

**Figure 4** – Bus test vector generation: a) counting sequence; b) true/complement sequence; c) proposed sequence for static faults.

The **Counting sequence (CS)**, proposed by Kautz [6], leads to shorts detection, and single LSA0, LSA1 detection (if 00…00 and 11…11 vectors are excluded). For the counting sequence, the total test length (TL) value is $TL = \left\lceil \log_2(N+2) \right\rceil$, where $N$ is the number of nets, or of bit lines in the bus. Obviously, TL is the closest integer equal or greater than $\log_2(N+2)$, and the same comment applies for the test lengths associated with the remaining sequences. It is depicted in Figure 4 a). The counting sequence suffers from an aliasing problem [19]. Hence, a new algorithm has been proposed.

**Counting sequence plus complement**, proposed by Wagner [9], is an extension of the CS, leading to an aliasing free test set, covering the same type of faults, and not requiring the exclusion of 00…00 and 11…11 vectors. For this test pattern, $TL = 2\left\lceil \log_2(N) \right\rceil$. This new sequence is also known as True/Complement (T/CS), and it is shown in Figure 4 b).

The **Walking sequence** has been proposed by Hassan [10], with better diagnosis capabilities [19]. It requires $TL = 2N$ test vectors.

The **Interleaved True/complement sequence (IT/CS)** (Figure 4 c)) is a modification of the True/Complement algorithm [11]. This sequence is, in turn, a modification of the Counting Sequence algorithm. The number of vectors for testing using the Interleaved True/Complement sequence is $TL = 2\left\lceil \log_2(N) \right\rceil$, where $N$ is the number of interconnect lines to test. In this work, we use the IT/CS sequence.

### 3.3 Software Tool for TPG

A proprietary application software has been developed to automatically generate the interconnect test patterns to be stored in each FPGA embedded memory. This application has as inputs the following variables: (1) the number of bits to be tested (bus width), (2) the memory size, (3) the data format and (4) the fault model(s) for which test vectors must be generated. At present, the software tool supports the IT/CS static fault model, and the MA dynamic fault model. The output result is a VHDL file containing the complete test set to be stored in the FPGA memory, in the format of a .coe file (Xilinx™ Coefficient File - COE), whose format has been devised by Xilinx™ [21]. In Figure 5 an example of a .coe file is shown, with initialization keywords and memory content. More details of the COE format are given in [22].

**Figure 5** - COE file example

The organization of the .coe file is done in such a way that one of the sub-patterns is stored in one side of the memory, e.g., in the high memory space address, while the second sub-pattern is stored in the other side, e.g., in the low memory space address. This way, the controller can access both sets, requiring minimal logic.

The test pattern may include, at present, two sub-patterns, one associated with the detection of the listed faults of each fault model. The user’s interface of the software tool for TPG is depicted in Figure 6.

The memory, in turn, can be implemented using the wizards existing in the development tool used by the developer. The developer can choose the type of memory that is best suited to his (her) application and to the architecture of the FPGA selected for the design, provided that this is compatible with the format generated by the computer application.

**Figure 6** – ROM generator interface
4. Results

The proposed test methodology has been used with a complex data acquisition system [23][24], to perform interconnects test. The System Under Test (SUT) has five PCB interconnected by a back plane housing two 64-bit busses. Nine Xilinx™ xc2v4000-4bf957 FPGA (957 pins) carry out the intended functionality. The SUT dialogues with a front-end sensing sub-system and with a PC for image reconstruction.

The electrical test of the SUT boards is carried out prior to FPGA’s functional test. Some defects have been detected and corrected in the system prototype, during the test phase of the FPGAs, mainly opens and bridges in the transceivers pin-outs. These errors have not been detected in the electrical test of the boards.

After this initial phase, interconnects test has been carried out. As shown in Figure 7, in this case study all FPGA communicate through the two busses. Two reasons justify this solution. First, the need to transmit two types of data packages, namely, one with 4 Bytes (event identifier) and another type up to 81 Bytes (event data) of information. The second reason is related with the required bandwidth (around 250 MBytes/s). Transceivers are used as FPGA gateways to the internal busses. Transceivers serve also as buffers for these busses.

The application software generates a sequence of 7+384 test vectors to uncover the static and dynamic faults (IT/CS and MA fault models, respectively). This means that, although the total test application time required verifying all bidirectional communication paths among the nine FPGA, through the two busses, may be large, the ROM requirements, to be configured in each FPGA, are extremely low ((391x64) bit).

Test is started by the Master FPGA, actuated through a PC on special on-board test connector. All FPGA read the vectors travelling in each bus, and decode the respective header, in order to identify if the vector is a Start Test command, Test Data from Master FPGA or Test Data from one of the other FPGA (Figure 8). Test data is then compared with the content of the internal ROM.

5. Conclusions

In this paper, a new test methodology for complex multi-PCB, multi-bus and multi-FPGA electronic systems is proposed. The novel methodology takes full advantage of FPGA technology. In fact, the presence of multiple FPGA devices allows us to configure built-in test functionality, for at-speed testing. Emphasis has been given in this paper on interconnects testing. However, the methodology encompasses all chip, board and system-level testing.
The new methodology drives the development of test data to be stored in an embedded memory of each FPGA (typically, a ROM), using unallocated resources. All FPGA devices store identical data. Each FPGA will act as TPG or TAE, in order to test all bidirectional paths among complex devices, through system busses, transceivers and board connectors. Hence, the total physical path is tested, at-speed, either at production stage, or during lifetime testing. One FPGA will act as master, being configured to have additional functionality, like the test controller. Test data consist of deterministic test vectors, generated to detect static and dynamic interconnect faults. Application software has been developed to generate such data, in .coe data format, to be used directly with Xilinx FPGA. At present, two fault models are supported: IT/CS for static faults and MA for crosstalk faults. However, the methodology can be applied for any improved fault model, or test generation algorithm.

The proposed interconnect test methodology has been validated in hardware, using a case study, a complex interconnect fault model determined through systematic analysis of the switching activity of adjacent lines, and weakly on the activity of distant wires. This fact can relax the constraints for test vector generation, leading to shorter test sequences. Results will be reported in the future.

**Acknowledgements**

The work reported in this paper has been partially supported by AdI (Portuguese Innovation Agency) in the scope of the PET II Project 70/00327. The work of C. Leong was supported by FCT (Portuguese Foundation for Science and Technology), under grant SFRH/BD/31646/2006.

**References**


