BIST based can Bus Control System Implemented into FPGA

Amit Kumar Bhadrawat, Sourabh Sharma

Abstract— Electronics components in many application required maximum level of fault tolerance and high reliability . Application like avionic, railway ,deep space mission can serve as an example of these applications. In these applications, electronics components are exhibited to the environment conditions, from among them especially cosmic radiation can have an undesired and destructive effect. In this paper,the design and implementation of BIST based CAN bus control system into FPGA is described. The bus control system uses CAN Aerospace application protocol .the fault tolerant features of the developed system are improved by BIST architecture. Then, experiments With SEU injection into the FPGA configuration memory with both non-TMR and BIST architectures are described, the results presented and evaluated.

***Keywords*- CAN bus, BIST, fault, fault tolerant, FPGA, TMR.**

I. INTRODUCTION

This paper is about fault tolerant CAN bus control system. we are using FPGA as a platform for implementation. We have used BIST method to reduce complexity of system. The complexity of digital systems have a significant impact on reliability and diagnostic features of these systems. FPGA-based systems are becoming increasingly popular for space-based applications due to their high-throughput capabilities and relatively low cost. When faults are detected in any part of the system implemented into FPGA then a possibility to reconfigure it and extend its lifetime exists. SRAM-based FPGAs are susceptible to radiation-induced Single Event Upsets. SEU causes the change in the state of a digital memory element caused by an ionizing particle. As the ionizing particle passes through the device, charge can be transferred from one node to another. This charge transfer can lower the voltage of a memory cell and change its internal state. SEU occurrence in FPGA memory can be seen as a big problem for many digital systems. Therefore, many FT techniques have been proposed and tested for mitigating SEUs in systems implemented into FPGAs.

II. FPGA

A Field-Programmable Gate Array is an FPD featuring a general structure that allows very high logic capacity. Whereas CPLDs feature logic resources with a wide number of inputs (AND planes), FPGAs offer more narrow logic resources. FPGAs also offer a higher ratio of flip-flops to logic resources than do CPLDs.

Manuscript Received on October 2014.

Amit Kumar Bhadrawat, M.Tech Student Trinity Institute of Technology and Research, Bhopal, India.

Sourabh Sharma, Asst. Prof., (EC) Department, Trinity Institute of Technology and Research, Bhopal, India.

Multiple methods of FPGA programming have been devised, including antifuses, SRAM, and EEPROM/FLASH. Currently, SRAM-based FPGAs are the most popular, due to the high number of reconfiguration cycles they support and the relative ease of programming. An illustration of a basic FPGA architecture is given in Figure.1. Switch boxes at wires segment intersections provide routability through the use of programmable interconnect points (PIPs), which are programmable connections between wires. The switch boxes and wire segments form an interconnect matrix. Logic blocks tap into this matrix using connector blocks (also known as input/output multiplexers).The logic blocks are multi-input, multi-output digital circuits capable of implementing both combinational and sequential designs. They are usually made up of lookup tables (small ROMs), multiplexers, and flip-flops, although alternative architectures have been devised. Multiple blocks are connected together through the programmable routing matrix to form complex designs, such as high-resolution multipliers and state machines .Specific FPGA architectures(i.e., Xilinx Virtex, Altera Apex) have additional features such as direct PLB to-PLB connections (not part of global routing matrix), carry logic (so that PLBs may be chained into adders), and multi-row and column length wires. These features fit within the framework given in Figure.1 by simply passing through connector and/or switchboxes.

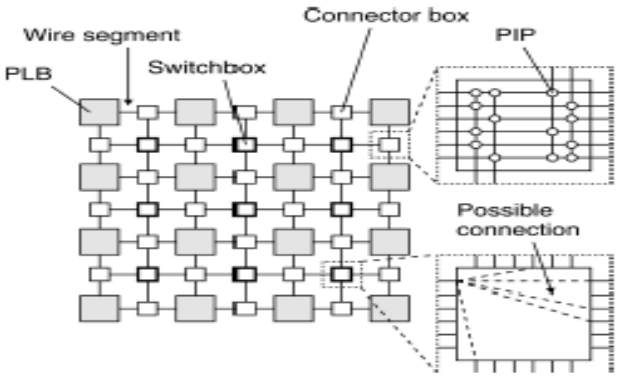

Fig. 1: Generic FPGA Architecture

III. FAULT

A. Causes Of Degradation

There are two types of faults which can affect FPGAs. These are highly relevant as some of the techniques which have been developed in response to them can also be applied to faults caused by degradation. The first of these is manufacturing defects. Manufacturing defects can be exhibited as circuit nodes which are stuck-at 0 or 1 or switch too slowly to meet the timing specification. Defects also

affect the interconnect network and can cause short or open circuits and stuck open or closed pass transistors. The second type of fault which is widely discussed in relation to FPGAs comprises of Single Event Upsets (SEUs) and Single Event Transients (SETs) caused by certain types of radiation. This is of particular concern to aviation, nuclear research and space applications where devices are exposed to higher levels of radiation. The most commonly considered failure mode is the flipping of an SRAM cell in the configuration memory. This causes an error in the logic function which persists until the configuration memory is refreshed in a process known as scrubbing. this recovery method is not applicable to permanent faults caused by degradation, ways of detecting SEU faults are relevant. Figure 2. shows the failure rate after the chip has left the factory and before the end of its life is typically constant and is due to environmental stresses. The time axis may be compressed, possibly significantly, if the device is used in a harsh environment

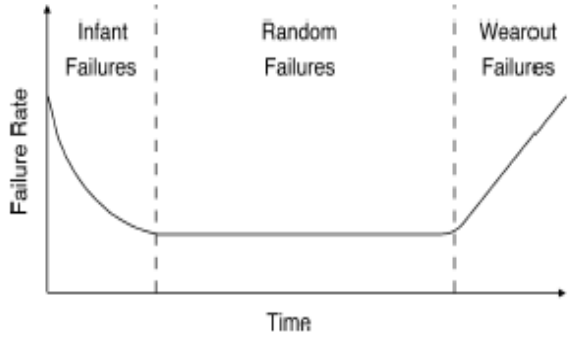

Fig. 2. System failure rate during the life cycle of an FPGA.

IV. LEVEL OF FAULT TOLERANCE

FPGA fault tolerance methods can be divided into two groups, based on the level of abstraction at which faults are tolerated. The first group attempts to deal with faults at the level of the FPGA hardware. The second group of methods takes a higher-level approach, tolerating faults at the level of the FPGA configuration

A. Device Level

Attempt to construct a fault-free array from a larger array containing faulty resources. When faults are discovered, permanent routing and/or logic changes are made within the FPGA, selecting redundant hardware resources to replace the faulty ones. Because alterations must be made at the hardware level, DL methods are generally only used for yield enhancement. However, the fact that modifications are made at the hardware level means that the tolerated faults are transparent to the end-user tools

B. Configuration Level

Methods treat the FPGA as a set of abstract resources, often represented as a graph structure, without considering the actual physical structure of the device. When a circuit is placed-and-routed, fault-free resources are selected from the set of available resources. The status of resources as faulty vs. fault-free is considered each time a circuit is placed-and-routed, so CL methods are able to tolerate new faults in the field. Obviously CL methods are not transparent to the tools, so tolerance of new faults requires additional configuration time.

V. METHODS FOR FAULT TOLERANT

Device-level fault tolerance was initially conceived as a method for yield enhancement .The first configurable VLSI arrays were not reconfigurable, so faults could not be tolerated dynamically. However, it was perfectly feasible to create a fault-free array out of a larger array containing faulty processing elements by making permanent configuration changes during manufacture. Configuration-level fault tolerance attempts to map a system function to a set of fault-free resources. Taking the system function into account allows methods to make more informed decisions about how faults should be dealt with. A device-level method must consider any resource that is faulty to be unusable .A configuration-level method, however, may decide that a given fault does not affect the system, and that the resource is still usable. For instance, if a LUT bit is stuck at '1', and the system function calls for that particular bit to have a value of '1', then the fault does not affect the system and can be ignored. This reasoning also holds true for interconnect .Configuration-level FT does have drawbacks, the most notable being that an external processor is generally required to analyze and reconfigure the system.

A. Fault Detection

The first step of a fault-tolerant scheme is fault detection. Fault detection has two purposes; firstly, it alerts the supervising process that action needs to be taken for the system to remain operational and, secondly, it identifies which components of the device are defective so that a solution can be determined. These two functions may be covered simultaneously, or it may be a multi-stage process comprising of different strategies. Fault detection methods can be categorized into three broad types:

1. Redundant/concurrent error detection uses addit- ional logic as a means of detecting when a logic function is not generating the correct output.
2. Off-line test methods cover any testing which is carried out when the FPGA is not performing its operational function.
3. Roving test methods perform a progressive scan of the FPGA structure by swapping blocks of functionality with a block carrying out a test function.

B. Fault Tolerance

Hardware level repair has the advantage of being transparent to the configuration. This makes repair a simple process, as the repair controller does not need any knowledge of the placement and routing of the design. Another benefit is that the timing performance of the repaired FPGA can be guaranteed, as any faulty element will be replaced by a pre-determined alternative.Hardware level fault tolerance has a drawback in that it can tolerate just a low number of faults for a given overhead and there are likely to be certain patterns of faults which cannot be tolerated. Fault tolerant methodologies will focus on area overhead, system performance impact, reconfiguration complexity,

and tolerable fault patterns. Reconfiguration complexity will only be used to compare configuration-level methodologies since DL methodologies do not involve any explicit reconfiguration of the circuit.

VI. CONTROLLER AREA NETWORK (CAN)

CAN is a serial communications protocol which efficiently supports distributed real time control with a very high level of security. Its domain of application ranges from high speed networks to low cost multiplex wiring. In automotive electronics, engine control units, sensors, anti-skid systems, etc. are connected using CAN with bitrates up to 1 Mbit/s. At the same time it is cost effective to build into vehicle body electronics, e.g. lamp clusters, electric windows etc. To achieve design transparency and implementation flexibility CAN has been subdivided into different layers.

- the (CAN-) object layer
- the (CAN-) transfer layer
- the physical layer

CAN has the following properties:

- prioritization of messages
- guarantee of latency times
- configuration flexibility
- multicast reception with time synchronization
- system wide data consistency
- multimaster
- error detection and signalling
- automatic retransmission of corrupted messages as soon as the bus is idle again
- distinction between temporary errors and permanent failures of nodes and autonomous switching off of defect nodes Message transfer is manifested and controlled by four different frame types:

A DATA FRAME carries data from a transmitter to the receivers.

A REMOTE FRAME is transmitted by a bus unit to request the transmission of the DATA FRAME with the same IDENTIFIER.

An ERROR FRAME is transmitted by any unit on detecting a bus error.

An OVERLOAD FRAME is used to provide for an extra delay between the preceding and the succeeding DATA or REMOTE FRAMEs.

DATA FRAMEs and REMOTE FRAMEs are separated from preceding frames by an INTERFRAME SPACE.

Flow of architecture is shown in fig.3.

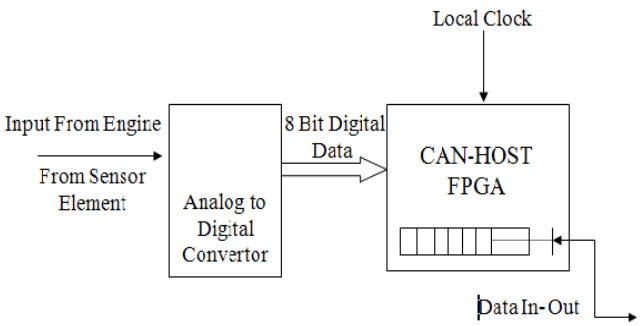

Fig. 3: Flow of Architecture of CAN

Input is taken from any sensor placed in the car (eg: engine) and provided to ADC, at the output of ADC we get a 8 bit digital value which is fed to the CAN host (CAN controller). The CAN host decodes the output obtained from the ADC and forms a particular message using CAN format and transmits it using CAN protocol.

VII. BUILT-IN-SELF TEST

On-line testing is fast becoming a basic feature of digital systems, not only for critical applications, but also for highly-available applications. To achieve the goals of high error coverage and low error latency, advanced hardware features for testing and monitoring must be included. One such hardware feature is built-in self-test (BIST), a technique widely applied in manufacturing testing BIST is a design-for-testability technique that places the testing functions physically with the CUT, as illustrated in **Figure .4.** In normal operating mode, the CUT receives its inputs *X* from other modules and perform the function for which it was designed. In test mode, a test pattern generator circuit TPG applies a sequence of test patterns to the CUT, and the test responses are evaluated by a response monitor (RM). In the most common type of BIST, test responses are compacted in RM to form (fault) *signatures*. The response signatures are compared with reference signatures generated or store on chip, and the error signal indicates any discrepancies detected .Four primary parameters must be considered in develop in BIST methodology for digital systems.

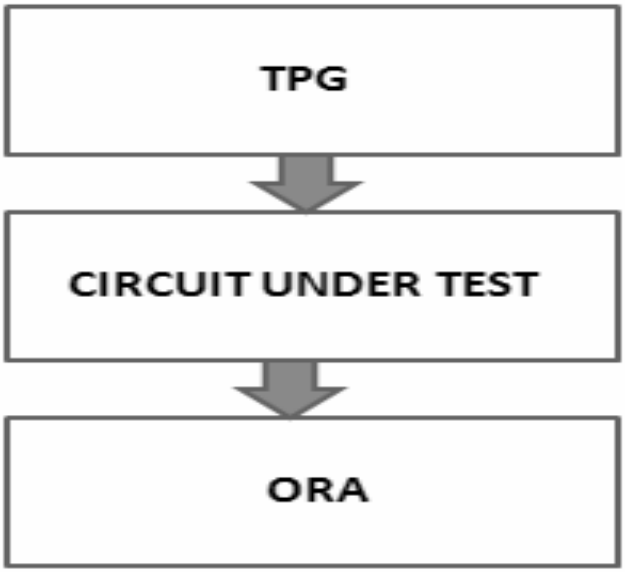

Fig. 4: Generic Flow of BIST

Fault coverage: This is the fraction of faults of interest that can be exposed by the test patterns produced by TG and detected by RM.

Test set size: This is the number of test patterns produced by the TG, and is closely linked to fault coverage. Generally, large test sets imply high fault coverage. However, for on-line testing, test set size must be kept small to reduce *FL* and *EL*. *Hardware overhead*: The extra hardware needed for BIST is considered to be overhead. In most digital systems, high hardware

overhead is not acceptable, as discussed earlier.

Performance penalty: This refers to the impact of BIST hardware on normal circuit performance such as its worst-case (critical) path delays. Overhead of this type is sometimes more important than hardware overhead.

VIII. MOTIVATION AND GOALS OF THE RESEARCH

With growing interest in the use of SRAM based FPGAs in space and other radition environments , there is a greater need for efficient and effective fault tolerant design techniques specific to FPGAs. Triple modular redundancy is common fault mitigation technique for FPGAs and has been successfully demonstrated by several organizations. Although TMR has been shown to significantly improve design reliability ,it carries a high overhead cost .At a minimum ,full TMR of a design requires three times the hardware to implement three identical copies of a given circuit .in addition additional logic is required to implement the majority logic voters .In the worst case , TMR can require up to six times the area of the original circuit .the additional hardwar resources required to triplicate the original circuit result in other secondary problems such as increased power and slower timing. TMR is a static hardware redundancy scheme for masking single faults in a digital circuit .Fig.5 .

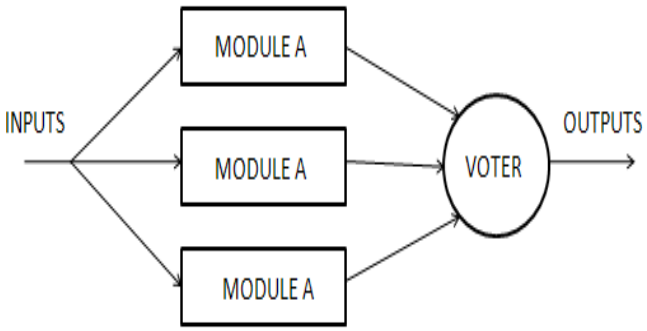

Fig. 5: Generic Flow of TMR

Depicts the traditional method for TMR. A failure in any one of the three circuits copies will be masked by the majority voter output. In order for TMR to work properly in an FPGA there should be no more than one upset in the configuration memory at any given time. More than one upset could defeat completely the majority voters and result in a functional error.Two reasons exist why the TMR based design can be possibly attacked more often: 1) TMR based designs need a greater area than the non-TMR design; 2) TMR based design contain voter which is not in our experiments protected against defects in any way. As soon as the voter is attacked, the design is completely corrupted and does not work. We worked with two versions of the design, non-TMR based design and BIST based under the assumption that we have no clear information about the relation between the position of particular bit in the bitstream and the function implemented in FPGA.The goal of the research was to verify how successful the injection into both versions of the implementations is the goal is to verify this hypothesis and gain precise data. The results of our targeted activities in the area of CAN bus control system design and the verification of its resilience against SEU attacks will show in next research paper.

IX. CAN BUS CONTROL SYSTEM DESIGN

The implemented CAN Bus Control System allows to connect FPGA-based systems through CAN bus which is interfaced on the small PCB (Printed Circuit Board) module by the SPI interface. This module consists mainly of CAN Bus transceiver and CAN controller MCP2515 with integrated SPI interface. It implements CAN protocol version 2.0B with maximal communication speed 1MB/s. MCP2515 contains 2 buffers for received frames and 3 buffers for transmitted frames. It also contains several filters and masks for control of receiving process. The MCP2515 circuit is controlled via the SPI interface, several instructions can be used for reading or writing from/to registers and buffers. The SPI interface supports 0/0 and 1/1 modes, it is able to communicate with maximal clock 10 MHz. Asynchronous events on the CAN bus are handled by interrupt system. Our CAN control system supports standard 11-bit ID of transmitted CAN frames. The CAN ID value is also used to define priorities. The frames with lower ID have higher priority and are transferred preferentially. CAN frame contains 8B data field without any information about its meaning. To increase information value and the usability of CAN frame, CANAerospace application protocol was used . Protocol definition is widely open to user defined message types and protocol implementations. CANAerospace message extends data field in the CAN frame. Message is dived into header and data part, its specific structure is shown in Fig. 6. Message header contains Node ID for identification of transmitting or addressed station, Data type for the definition of message data format and size, Service ID for specification of used node service and Message code for order identification during sequential transfer of messages. CANAerospace protocol uses CAN ID for the identification of 7 basic types of messages and their priority. Each type of message has allocated specific channel defined by the range of IDs.

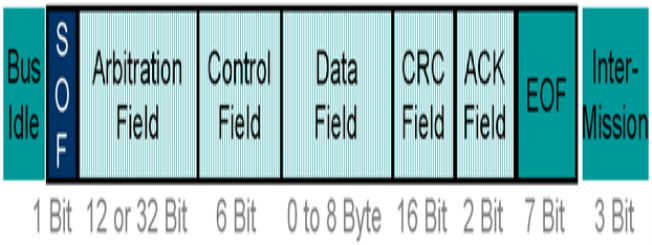

Fig. 6: Format of CAN Aerospace Message

The architecture of the control system is composed of application and communication part. In the communication part, CAN bus is controlled by CAN CTRL unit which uses the MCP2515 driver and SPI Master unit for communicating with the CAN module and controlling the MCP2515 circuit by SPI instructions. The main function of the CAN CTRL unit is to read and write CAN frames, to interrupt handling and provide configuration sequence for the MCP2515 circuit. The application part is formed by the CANAerospace calc. It provides basic mathematical functions for distributed computing via the CAN bus. Each function is operating as CANAerospace service in one defined channel. Except of mathematical functions, the

calc implements basic IDS service for its identification as is required by the CANAerospace protocol for each such application.

A. BIST Implementation of CAN Bus Control

For the experiments, the CAN bus control system was implemented as BIST system to increase fault tolerant parameters, the architecture is shown in Fig 2. The control system is included into CAN host unit. The unit can be replicated and all its inputs can be interconnected. The units have the following input signals: ADC_in, inject_fault, frame_tx tx_ctrl,clk, and rst asynchronous reset. All the outputs of CAN host units are connected to the inputs of comparator which identifies the correct inputs and propagates them to its outputs. If, the output of comparator is "0" i.e. system_ok ="0", then out will take out from second alternate path and if , here again system_ok ="0" then actuator transfer system control to the manual mode. Flow diagram shown in fig.7.

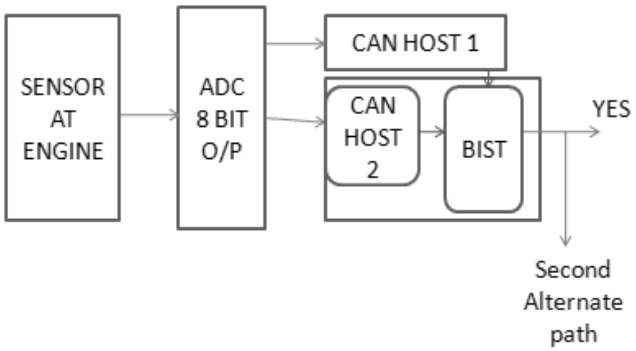

Fig. 7: Flow of BIST implementation of CAN Bus Control

X. CONCLUSION

In this research paper , we are analyzing fault tolerant properties of BIST based CAN control system design . We have Compare result of our analysis with TMR-based design and non-TMR based design and also compare with Evaluation factors, like area ,cost, speed , complexity of system ,power consumption. We are expecting as our theoretical analysis this design reduce system complexity ,And our proposed method will provide online fault detection using BIST technique and improve fault tolerant properties by providing alternate path when faulty value comes out.

REFERENCES

1. Microchip Technology Inc, "MCP2515 - Stand-Alone CAN Controller with SPI Interface," November 2005.
2. Robert Bosch GmbH, "CAN Specification 2.0," BOSCH, Stuttgart, Technical specification, 1991.
3. Michael Stock, "CANAerospace - Interface specification for airborne CAN applications V 1.7," Stock Flight Systems, 82335 Berg/Farchach,
4. G. Asadi, S. G. Miremadi, H. R. Zarandi, and A. Ejlali, "Evaluation of fault-tolerant designs implemented on sram-based fpgas," in Proceedings of the 10th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC'04). Washington, DC, USA: IEEE Computer Society, 2004, pp. 327–332.
5. J. A. Cheatham, J. M. Emmert, and S. Baumgart, "A survey of fault tolerant methodologies for fpgas," ACM Trans. Des. Autom. Electron. Syst., vol. 11, no. 2, pp. 501–533, 2006.
6. Emmert j. and Bhatia D.K. 1997 "Partial reconfiguration of FPGA mapped design with application for fault tolerance and yield enhancement.in proceedings of the 7th international workshop on field prommeble logic and application 141-150.
7. Fussele D.and vaema p. 1982.fault tolerance wafer scale architector for FPGA.in proceeding of the 9th annual symposium on computer architecture 190-198.
8. Green J.W. and gamal A.E. 1984 configuration of FPGA arrays in the presence of defect 4(oct) 697-717.