A new hardware fault insertion scheme for system diagnostics verification

Benoit Nadeau-Dostie[‡], Harry Hulvershorn[†], Saman M. I. Adham[†]
[†]Northern Telecom, P.O. Box 3511, Station C, Ottawa, Ontario, Canada K1Y 4H7
[‡]LV Software (Canada) Inc., 1525 Carling Avenue, Suite 404, Ottawa, Ontario, Canada K1Z 8R9

Abstract

A new fault insertion method to help debug diagnostic software of telecommunications systems is described. The method makes use of Boundary Scan to inject multiple and un-correlated faults in a telecom system carrying traffic. Both hardware and software implementation aspects are discussed. The new method allows the use of structural test as part of diagnostics software to locate faults.

I. Introduction

Fault insertion (also known as fault injection) has been used for many years to evaluate the effectiveness of diagnostic software. Even though it is possible to perform this evaluation using simulation, it is usually difficult because of the absence of proper tools and models. A common way of performing this verification is still to inject faults around components of a board using switches or other similar hardware. Usually only a few carefully chosen faults could be inserted using this primitive mechanism. For each revision of a board, a few boards are selected and modified to add the fault injection means. This method is becoming less practical because it is becoming necessary to inject more faults in more complex systems and, at the same time, the physical access to components and tracks of the board is reduced. The process has several drawbacks. It is expensive since the modifications on the selected boards are done after the board is designed. Faults are injected into limited number of tracks on the board which in turn results in low fault coverage.

This is why alternate methods are needed in modern telecommunications systems. Recently, several authors demonstrated how Boundary Scan (Bscan) [4] could be used to insert faults in complex systems containing Application Specific Integrated Circuits (ASICs) [1-3]. We first review those schemes. We then describe a significant improvement over the method introduced by Wilcox et al [3]. This improvement is essential to the verification of diagnostic software making use of structural tests (including all the ones involving Bscan) to locate faults. The new scheme retains all the other benefits of Wilcox's. Multiple un-correlated faults can be injected at the same time while the system is running. No extra delay is inserted in the Bscan cells to inject the faults. Finally, existing Boundary scan cell layouts can be reused to implement a complete fault insertion cell. Those benefits are traded off against area.

A few applications of this new fault insertion method are demonstrated. Several aspects of the system software required to generate and apply faults is also described. Limitations of the scheme are discussed before we conclude.

II. Review of previous schemes

Sedmak [2] suggests three ways to inject faults at the output of ASICs. For example, all outputs of a given ASIC can be faulted using standard Bscan instructions (HIGHZ, CLAMP or EXTEST). Another way is to load

INTERNATIONAL TEST CONFERENCE

0-7803-2991-0/95 $4.00 © 1995 IEEE

the Bscan chain with faulty value(s) using the SAMPLE/ PRELOAD instruction followed by another instruction that could expose the values to the output pins. Unfortunately, no implementation is described to show how this can be done.

Chau [1] describes in detail how to modify a Bscan cell to inject faults. In his scheme, any combination of inputs or outputs can be faulted at the same time. The fault(s) is (are) injected in a way such that the performance of the circuit is not affected more than it would be due to the presence of Bscan alone (i.e. one multiplexor delay). The Bscan output register holds the flag indicating whether or not this input or output pin will be affected by a fault. The value of the faulty data itself can be determined in two ways. It can be driven from a bit of the Instruction Register of the Test Access Port (TAP) or from the shift register of the Bscan chain. In the first case, the value of the faulty data is unique for a given chip imposing constraints when multiple faults are considered. The second option is difficult to manage because it is not possible to scan in the faulty data values without corrupting the flags shifted in previously unless the update state of the TAP state machine is suppressed during this mode. Another problem is that the output of the faulty pins is rippling when the faulty data values are scanned in. This is unacceptable for our applications since the handling of a stuck-at 1 or a stuck-at 0 fault can differ in the diagnostic software program. The modification of the Bscan cell itself was not acceptable to us because of the large number of cells that would have needed to be modified. Finally, it is not possible to use structural tests, including an interconnect test, to locate the fault once its presence has been detected by the system.

Wilcox et al [3] describes another scheme that addresses most of the limitations of Chau's. The Bscan cells themselves are not modified. An example of an output Bscan cell is shown in Figure 1. The parallel latch of the Bscan cells is used to carry the faulty data value that needs to be inserted at the selected pad(s). The fault flag indicating whether a particular pin is faulted or not is stored in a separate register that is part of the Bscan chain.

Fig. 1. Output Bscan cell

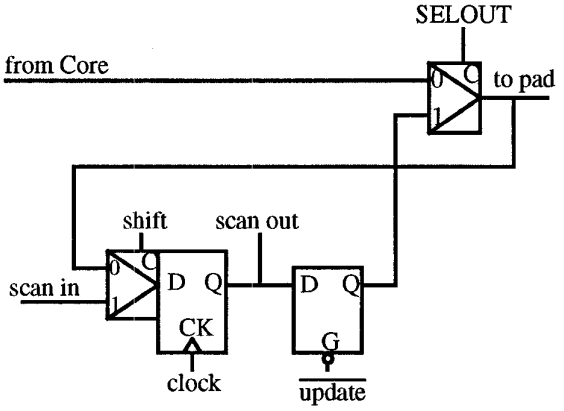


Fig. 2. shows this additional register and additional logic contained in a separate cell. Figure 3 illustrates how the two cells are connected together.

Fig. 2. Fault insertion cell (Wilcox92)

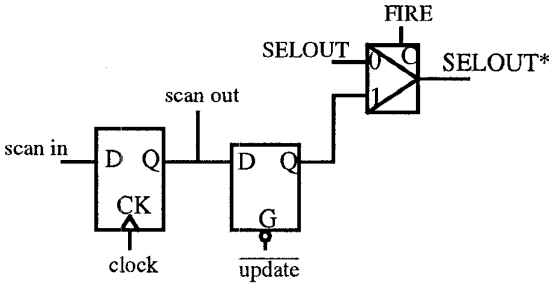
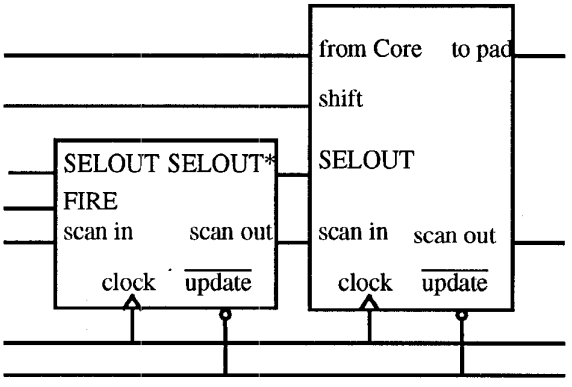


Fig. 3. Bscan cell with fault insertion



In a conventional Bscan implementation, the SELOUT control signal selecting the system data or the data value of the Bscan cell is common to all Bscan cell and is only activated when an interconnect test is performed (EXTEST instruction of the IEEE 1149.1 standard) or when it is necessary to maintain all chip outputs at a particular value (CLAMP instruction). However, when a fault insertion cell is present, the global signal SELOUT can be replaced with the output of the latch located in the fault insertion cell (fault flag) using a multiplexer controlled by a global signal called FIRE (Fault Insertion Register Enable) generated by the Test Access Port (TAP). Fire is activated (logic 1) when a fault insertion instruction that also selects the Bscan chain is loaded into the instruction register. When FIRE is active (logic 1), the fault flag determines whether a fault is inserted at this pad. The faulty value is determined by the output of the latch located in the Bscan cell itself. At least a flip-flop, a latch and a multiplexer is needed for each potential fault site. Similar implementations are available for input and bidirectional cells.

This cell allows multiple and un-correlated faults to be injected at the same time while the system is running, does not add further delays to the signal path and does not require modifications on the Bscan cell itself. At the time this fault insertion scheme was introduced in the company, the diagnostic software was still mainly based on functional tests. However, as the structural tests (e.g. interconnect test, logic BIST, scan) became easier to use at the system level, it became clear that the fault insertion scheme described above had a serious limitation. It is not possible to find a fault that is inserted because it disappears when structural tests are applied. For this reason, a new fault insertion cell is required.

III. New fault insertion scheme

Even though several implementations can address the limitation of the fault insertion scheme of Wilcox et al, many constraints limit the solution space. The new Bscan hardware supporting fault insertion needs to be pin compatible with the previous implementation for easy migration of old designs. Also, the solution needs to be compatible with the tool verifying the implementation and extracting the Bscan chain and fault insertion information.

Two modifications to the existing Bscan hardware needs to be implemented to support the new fault insertion scheme. The fault insertion cell and the Test Access Port (TAP) are modified to provide a "locked" mode that allows to maintain a fault during a structural test.

The modification to the fault insertion cell is shown in Figure 4. Two OR gates are added to disable the update of both the fault flag and the Bscan cell output latch. The update clock would normally update those latches at the end of a shift operation during an EXTEST or INTEST instruction wiping out the fault that we are trying to insert.

Fig. 4. New fault insertion cell

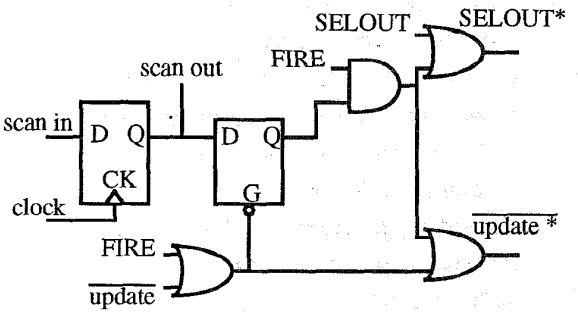
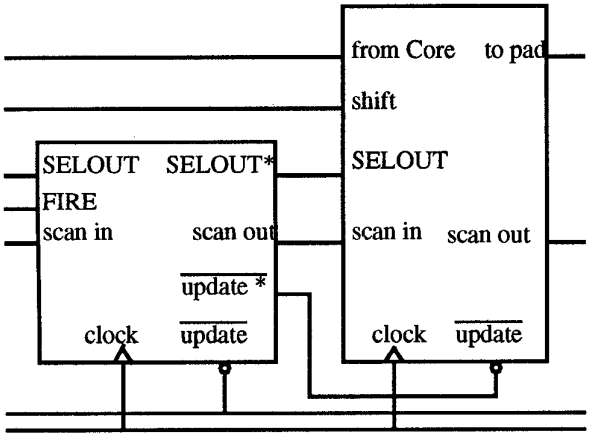


Figure 5 illustrates how the new fault insertion cell is connected to the Bscan cell. Since the Bscan and fault insertion cells are always grouped together when they are used, it can be seen that the group is pin compatible with the previous cell. Only, one additional internal signal ($\overline{\text{update *}}$) is needed.

Similar cells can be built for input and bidirectional pads. For inputs, the SELOUT signal becomes SELIN and both SELOUT and SELIN signals are generated for bidirectional pads, inserting the fault in both the input and output directions.

The TAP is modified to force the FIRE signal active (logic 1) when the fault insertion instruction is loaded into the Instruction Register (IR). FIRE stays active "sticky" until the next fault insertion instruction is loaded into IR where it is deactivated (logic 0). This toggling effect allows diagnostics software to use the IR for other purposes (e.g EXTEST, RUNBIST) as long as it does not make use of the fault insertion instruction. The FIRE output can be wired back to a status bit of the TAP to verify its condition. Both the synchronous and asynchronous reset of the TAP will reset the FIRE output. This new TAP is also pin-compatible with older versions.

Fig. 5. Bscan cell with fault insertion



IV. Applications

In this section, we describe how to insert faults using the cells described above. The effect of the faults inserted at inputs, outputs, bidirectional and internal nodes of an ASIC is also discussed.

The sequence to inject faults is the same for all cases.

Make sure the ASICs that need a different fault pattern are not in fault insertion mode.

This can be done in two ways. The TAP can be reset in a synchronous or asynchronous fashion, or the fault insertion instruction (FI) can be used to toggle the FIRE output of the TAP assuming the fault insertion mode was previously active. It is recommended to observe the state of the FIRE output via a status bit of the TAP if the system software can not keep track of the number of times the instruction was used. This toggle mode allows a local modification of the fault pattern whereas the reset will affect all chips on the board. It is also the only way to preserve the pin compatibility of the TAP with previous versions.

Load the boundary scan chain, containing the FI latches, of all chips intended to inject faults with the appropriate fault flags and fault data values.

The other chips should be programmed to select their bypass register to accelerate the set-up of the fault pattern.

Activate the fault insertion mode using the FI instruction.

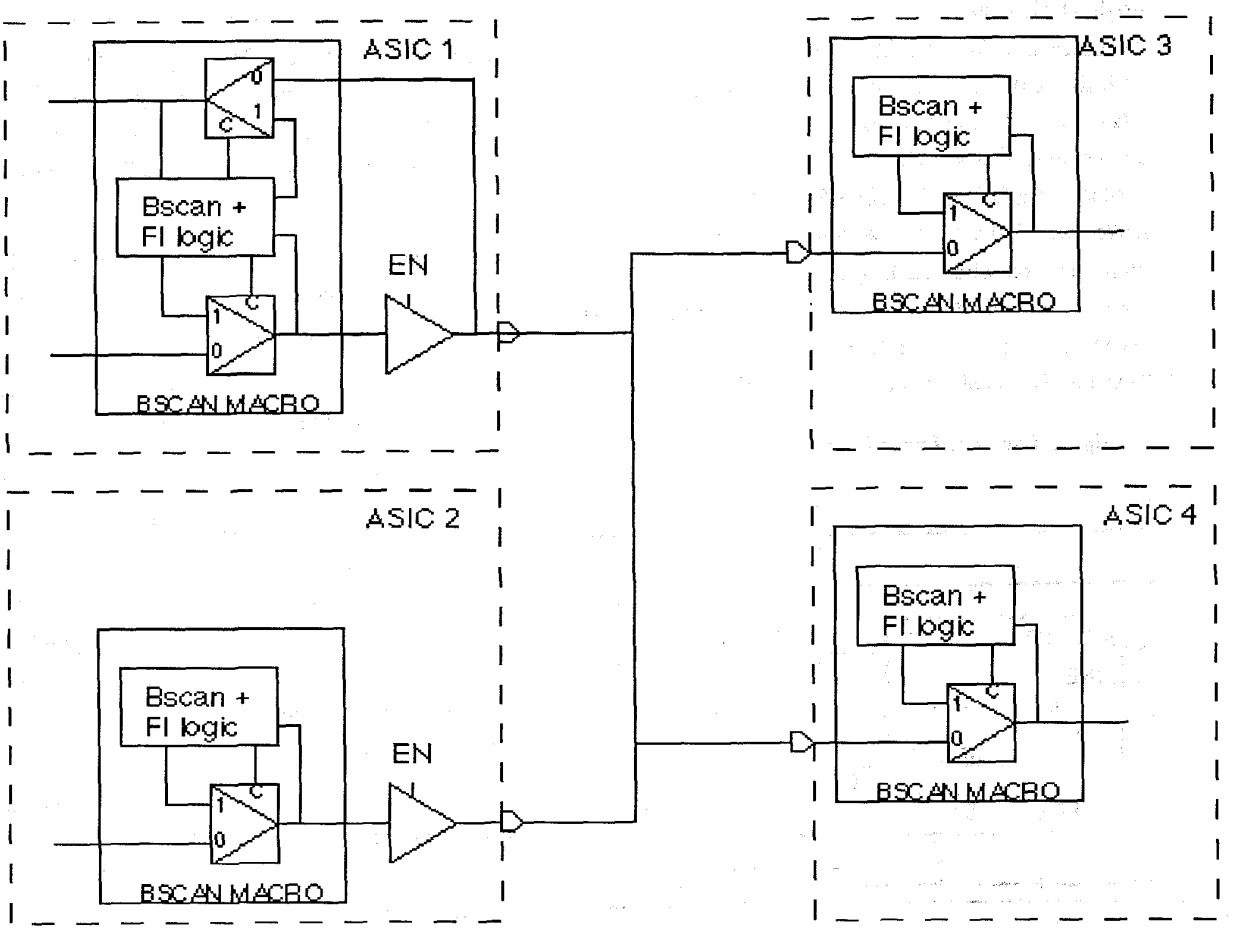
After activation, the system should detect the fault(s) and enter the diagnostics mode. The system can use any structural test modes to detect the fault(s). However, the TAP can not be reset at any time and the boundary scan chain should not be selected using the FI instruction. After successful diagnosis, the fault pattern can be changed by starting over at step 1.

The next five subsections explain specific situations that occur when faults are injected at chip inputs, outputs or internal nodes. Note that the faulty behavior described is specific to the basic Bscan cell shown above. Other implementations will lead to a different behavior. For example, if the data captured by the Bscan cell is the system data BEFORE it is intercepted by the Bscan multiplexer (i.e. signal coming from core is connected to the flip-flop instead of the output of the multiplexer in Fig. 2.) the fault will only affect one mode of operation of the Bscan cell (e.g. internal test or external test). This is an advantage is some cases. In general, we prefer the implementation shown in Fig. 2. because of the higher fault coverage achievable through structural tests only.

A. Injection of faults at inputs

A fault can be injected at the input of ASIC 3 or 4 (see Figure 6) using input-only pads by loading a 1 in the fault insertion register associated with that input as well as the faulty data value in the corresponding Bscan register. The

Fig. 6. Fault insertion scenarios



fault will be diagnosed as being potentially a cold solder joint, broken wire bond or a dead input pad during the interconnect test. However, the internal scan or BIST test will also fail leading to the conclusion that it is definitely the pad and that the ASIC should be removed.

B. Injection of faults at outputs

The fault injection mechanism is similar to the one described above for inputs. However, note that a fault injected using ASICs 1 or 2 will only be visible from ASICs 3 and 4 if the pad driver matched with the BSCAN macro is enabled through functional means during normal operation of the system. This is because pad driver enable macros are typically not equipped with fault insertion capability in order to avoid potential contention with other drivers on a same bus that could cause permanent damage to the devices involved. On multi-source nets, like our example, the fault will look like an intermittent one during the normal operation of the system.

A stuck-at fault on a board net driven by more than one source can be emulated by inserting faults at all sources. However, the diagnosis will be that all pad drivers are stuck and that all chips where faults have been injected should be removed. However, a "real" stuck-at fault on a board net would lead to a different result. The fault would not cause the internal test of any of the ASICs to fail. Only the interconnect test would fail and would point to the stuck net. Note that this is again an artifact of the Bscan cell used and that the alternate implementation discussed at the beginning of this section would have a

behavior that mimics more closely the one of the "real" stuck-at fault.

C. Bidirectional pins

Our bidirectional Bscan cells only use one flip-flop to sample data coming in and out the chip. Its design resembles the one shown in figure 10-34, page 10-39, in the 1149.1 standard [4]. As mentioned above, the fault insertion cell will cause the fault to be inserted in both the input and output direction. So, the fault will propagate from the input to the core of the chip and it will propagate to the other chips when the pad is enabled as it was the case for output only pads.

D. Injection of faults at internal nodes:

Internal nodes can be faulted using a BSCAN input cell equipped with this new fault insertion cell, if desired. These cells must be concatenated to the BSCAN chain, adding to its length. Traditionally, very few faults were injected at internal nodes because of the difficulty of getting access to the signals controlling the boundary scan cells or because of the relatively high routing cost of those signals throughout the chip. Strictly speaking, the additional BSCAN cells required for faulting internal nodes are in violation of the IEEE 1149.1 standard. However, the impact of this non-compliance is minimal and doesn't cause problems with commercial software used to generate interconnect test patterns.

V. Software Support

Software must be developed to exploit the fault insertion capabilities designed within the ASIC hardware. Throughout the development of the fault insertion capability, several software architectures were experimented with, all with the same fundamental architectural issue: the fact that inserting a fault may impact the ability to remove the fault and restore normal system operation. This obvious fact greatly impacts the physical, communication, and software architectures used.

A. Physical access

Since the mechanism to fault ASIC pins is accessed from the 1149.1 port, a board test bus must connect all 1149.1 compliant ASICs together. Mastership of this bus can be provided either through an external 1149.1 controller and connector, or an on card 1149.1 test bus master device. Either method translates processor write cycles into the serial test data stream required to enable the fault on an ASIC pin under the control of either an in-system or off system processor. The availability of an in-system test bus master is a requirement when structural testing is included as part of the system's test and diagnostic software, which leverages the use of the new fault insertion scheme within the devices.

B. Communication architecture

Once physical access to each boards' 1149.1 bus has been provided, the method of communicating fault insertion commands or vectors within the system has to be determined. Similar to the architectures for conducting system level structural testing, either a centralized or distributed architecture can be implemented. [5]

In a centralized architecture, the system processor is responsible for all 1149.1 activities within a system implying that the system processor has direct access to the all circuit card 1149.1 test bus master across either a dedicated backplane test bus, or using the system communication bus. If the system communication path is used, NO faults that affect communication between the system processor and the test bus master may be inserted. Use of a dedicated backplane test bus alleviates this concern, and allows for the implantation of a more generic software architecture. The major advantage of a centralized architecture is that a single point of access is used to insert faults within the entire system. Of great concern however is the load placed on the system processor while the fault is being inserted, and especially when attempts are made to remove the fault, at which time the system processor is pre-occupied with taking corrective action as a result of the fault.

Access to systems having distributed processing capabilities varies greatly. Use of an external 1149.1 controller falls into the category of distributed access, since the responsibility of generating the fault insertion data, and driving the data onto the 1149.1 test bus no longer resides with the system processor. If control of the 1149.1 bus resides within the system, a distributed architecture dictates that each card has local software access to the board 1149.1 bus. If a single point of access is desired, access to the system is still provided through the system processor, but faults can now be controlled using high level commands between the system processor and the target circuit card. This necessitates that the software on the target card be able to generate the fault insertion vector locally: if the communication between the processors consisted of the actual fault insertion vectors, there would be no reduction in load on the system processor when compared to the centralized architecture. Similar to the centralized architecture, no fault may affect the communication between the system processor and the target circuit card processor.

Another alternative using a distributed architecture is to communicate to each circuit card across a local debug port (e.g. RS232), which would permit either high level fault insertion commands, or fault insertion vectors to be transferred to the circuit card, without intervention from the system processor. The downside to this approach is that the number of physical connections to the system which may become inhibitive. (1 per card)

Finally a network layer using either a private LAN or ethernet must be added to either architecture to allow fault insertion testing of systems without having to be physically located alongside the equipment. This permits the evaluation of the network impact of a fault within a particular system from a central location. Test cases have in fact been run where the switch being faulted was located hundreds of Kilometers from the site the testing was being performed.

C. Pattern generation and application

The final issue in architecting the software for fault insertion is the generation of the fault insertion vector. The capability of generating the serial test vector data can reside within the system, or the computer controlling the fault insertion activity. The selected implementation is dependent on the bandwidth available between the controlling computer and the system.

The simplest approach is to generate the serial data external to the system, and transfer the generated vector across the network to the system processor, or target circuit card, depending on the system architecture selected. This solution requires the greatest bandwidth, but simplifies the system resident software which only has to transfer the incoming data to the target 1149.1 test bus master.

The amount of data transferred across the network, and within a system can be greatly reduced by improving the software within the system. If the system resident software contained information which included the instruction required to select a device boundary scan chain, the instruction to globally enable a fault within a device, and the 'no faults' value of the boundary scan chain, only the position of bits on the boundary scan chain that must be toggled to insert a specified fault (2 per fault) must be transferred. Another benefit to this approach is that intermittent faults can be emulated within system software, since the system has the knowledge required to globally enable and disable faults.

Finally, fault insertion can be entirely implemented within the system software. In addition to the information required presented in the previous implementation, software would have access to the boundary scan signal to fault insertion cell mapping required to assert faults based on the specification of a device pin within the command. This is the most logical solution for systems that already include software access to structural test capabilities. Since fault insertion is a procedure to be conducted only within a lab environment, all supporting data required for fault insertion is downloaded into the system prior to beginning the fault insertion exercise, preventing

inadvertent access to fault insertion once the equipment is in the field. Faults are inserted based on commands received from the controlling computer.

D. General observations regarding fault insertion

When fault insertion was first architected for use within a system, the intent was to use fault insertion only as part of system software verification and regression testing. New software diagnostics would be run against a pre-selected set of faults to ensure that these faults were detected and that there was no degradation in the fault diagnostic capabilities between successive releases of software. Since boundary scan fault insertion coupled with the system software provided an easily accessed mechanism by which any pin level fault could be asserted, it soon became widely used as part of the diagnostic development process. Software designers could now verify the reaction of the system to any fault scenario, and ensure that the diagnostics being developed properly reacted to the fault prior to release of the software for verification testing. This flexibility directly impacted the quality of the diagnostics, since the actual reaction of the system to a fault could be observed, rather than having to guess at how the system would react to the fault. Many theories regarding system functionality in the presence of faults were disproved by actually inserting the fault, as reflected by the number of queries whether the fault insertion software was in fact faulting the correct signal. The sheer number of faults that could be asserted also impacted the quality of diagnostics. Rather than just a select number of faults being asserted, hundreds of faults were verified for proper diagnostic resolution. Since fault insertion is used to verify diagnostics, software developers insisted that fault insertion be able to affect structural tests as well as functional tests, resulting in the new fault insertion scheme presented within this paper.

VI. Limitations

The size increase of the Bscan and Fault Insertion cells combined is the main potential problem with this scheme. The size of the Bscan cells can increase by as much as 50 to 100% depending on the pad type (e.g. input, output or bidirectional). Selective use of the fault insertion cell is sometimes required to alleviate this problem. For example, wide busses will only have a few pads with fault insertion capability. This way, the cost can be limited to a small fraction of the chip area (1-2%). Table 1 presents the area overhead of adding fault insertion to several ASICs.

Table 1: Area Overhead of Fault Insertion

Device Name	# of Signal Pins	# of FI Cells	Total # of Gates	% Area of FI
A	135	125	1035658	1.93%
B	145	113	168148	1.08%
C	126	69	57768	1.91%

The Bscan register is nearly twice as long when fault insertion is used on all pads. We examined the possibility of defining a separate register for the fault flags. However, there are compatibility issues with the tools we are using to perform the extraction of the fault flags and the internal test (BIST) modes. Selective use also addresses this problem.

There are some minor compliance issues with the IEEE 1149.1 standard controlling the implementation of Bscan. Strictly speaking, there is definitely an issue in the case this fault insertion macro is used to fault internal nodes that are not connected to pads. The more regular case where the fault insertion macro is used to insert faults at pads is more subtle but doesn't cause problems with commercial board test pattern generators.

The diagnosis of faults injected at the pads using the boundary scan chain will always cause the failure of both the interconnect test and internal scan/BIST test leading to the conclusion that the ASIC is at fault. This is not always desirable. The alternate type of Bscan cell described above will circumvent this problem. Another workaround is to:

Order the structural tests used to diagnose the faulty system such that the interconnect test is always run before

the ASIC tests. Exit the diagnostics procedure if a failure is encountered.

Extend the Bscan chain to include at least one internal fault. This way, a fault can be injected so that only this ASIC will report an error OR use the alternate type of Bscan cell for at least one input.

VII. Conclusions

A new fault insertion scheme to help verify diagnostic software of telecommunications systems was described. The fault insertion feature has been successfully used to accelerate, by several months, the initial debugging of the diagnostic software as well as evaluating the quality of new diagnostic software releases.

The scheme makes use of Boundary Scan to insert faults asynchronous to a system carrying traffic. The faults injected remain active even when structural tests are applied to diagnose the source of the problem, an essential feature to the verification of diagnostic software of new telecommunications systems making extensive use of BIST. In fact, this scheme should be general enough to be applicable to other types of systems. The new scheme retains all the other benefits of [3]. Multiple un-correlated faults can be injected at the same time while the system is running. No extra delay is inserted in the Bscan cells to inject the faults. Finally, existing Boundary scan cell layouts can be reused to implement a complete Bscan cell with fault insertion capability.

Application of the fault insertion hardware was described as well as the main aspects of software support required at the system level to make use of the fault insertion capability. Several implementation options were presented for physical access of the fault insertion hardware, communication of fault insertion commands within the system and fault insertion pattern generation.

Fault insertion allowed us to improve significantly the quality of the system diagnostic software in considerably less time than with the traditional hardware fault insertion methods. The benefits of the new scheme were traded off against area and Bscan chain length. The cost could be limited to a small fraction of the chip area (1-2%) by making selective use of the fault insertion cell without reducing significantly the number of critical nodes that needed to be faulted.

References

[1] Savio Chau, "Fault Injection Boundary Scan Design for Verification of Fault Tolerant Systems", International Test Conference 1994, Washington D.C., pp. 677-681.

[2] Richard Sedmak, "Boundary-Scan: Beyond Production Test", 12th VLSI Test Symposium (1994), Cherry Hill N.J., pp. 415-420.

[3] Phil Wilcox, Jim Hjartarson, Robert Hum, "Software Verification by fault insertion", U.S. patent no. 5,130,988 (1992).

[4] "IEEE Standard Test Access Port and Boundary-Scan Architecture", IEEE Std 1149.1a-1993, October 1993.

[5] Saman Adham, Paul Soong, Harry Hulvershorn, "Linking Diagnostic Software to Hardware Self Test in Telecom Systems", International Test Conference 1995