Physical Defect Modeling for Fault Insertion in System Reliability Test*

Zhaobo Zhang[1], Zhanglei Wang[2], Xinli Gu[2] and Krishnendu Chakrabarty[1]
[1]Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708
[2]Cisco Systems, Inc., San Jose, CA 95134

Abstract

Hardware fault-insertion test (FIT) is a promising method for system reliability test and diagnosis coverage measurement. It improves the speed of releasing a quality diagnostic program before manufacturing and provides feedbacks of fault tolerance of a very complicated large system. Certain level insufficient fault tolerance can be fixed in the current system but others may require ASIC or overall system architectural modifications. The FIT is achieved by introducing an artificial fault (defect modeling) at the pin level of a module to mimic any physical defect behavior within the module, such as SEU (Single Event Upset) or escaped delay defect. We present a hardware architectural solution for pin fault insertion. We also present a simulation framework and optimization techniques for a subset of module pin selection for FIT, such that desired coverage are obtained under the constraints of limited FIT pins due to the costs of the associated implementation. Experimental results are presented for selected ISCAS and OpenCore benchmarks, as well as for an industrial circuit.

1. Introduction

Reliable systems employ error-handling techniques to ensure continuous operation in the presence of hardware errors. A typical hardware platform can encounter three types of hardware errors during operation: [1], [2]

(1) Hard Failure – These are caused by permanent and readily reproducible physical defects. Most defective parts affected by this type of errors can be screened by manufacturing testing but some may still be shipped to end customers.

(2) Intermittent Failure – These failures are caused by occasional faults that occur as a result of frequency, voltage or other environmental conditions that is reproducible under the same condition. This type of fault is usually attributable to corner conditions in design [3].

(3) Transient Failure – These are caused by "one time" and non-reproducible faults that are usually caused by high-energy particles (e.g. cosmic radiation, alpha particles) striking latches or memory cells and flipping circuit logic values. Cosmic particles are high-energy particles from outer space that cannot be prevented. Alpha particles originate within chip packaging materials and can be minimized via improvements in the manufacturing process. As semiconductor processes and core voltages keep shrinking, these particle-induced failures are becoming a pressing issue for hardware systems. Thus error detection and handling techniques are crucial to mitigate the consequences of such failures. These errors are also referred to as soft errors or single event upsets (SEU).

All current hardware platforms at "system companies" such as Cisco Systems, Inc. employ some level of error handling, from the simplest form of rebooting the box on low-end platforms to more complex hardware redundancy schemes and software support to recover from hardware errors in high-end platforms. In addition, such platforms use some level of diagnostic software to diagnose and repair boards in manufacturing and allow customers to quickly identify and repair faulty hardware components within their networks. The need to validate the above types of platform software requires the use of an error stimulus of some sort to insure both types of software perform as intended in the presence of faulty hardware. Runtime software must respond to hardware errors in the prescribed manner and diagnostics must quickly and correctly isolate errors to a minimal set of failing components. This error stimulus testing is called hardware fault-insertion test, and the artificial error insertion provides an indication of how the system will respond to real hardware errors.

Fault-insertion test and its variants have attracted considerable attention in the study of reliable system design [4], [5], [6]. The work in [4] characterized the effects of transient faults on a high-performance processor pipeline, including the analysis of fault masking and identification of the most vulnerable parts of the processor. In [5], the soft-error resilience results obtained from the statistical fault insertion were verified using proton-beam experiments. The work reported in [6] includes a tool, called DEPEND, which provides an integrated design and fault insertion environment for system dependability analysis.

Even though FIT has been widely used to assess system reliability, it needs to be revisited for high-density ICs and boards in use today. FIT is difficult to apply to these systems because of the prohibitively large fault space and the practical difficulties associated with the insertion of all the faults in the system. Selecting a group of faults to represent the complete set of faults and fully exercise the system is therefore extremely important for FIT-based system reliability assessment.

* The work of Z. Zhang and K. Chakrabarty was supported in part by a research grant from Cisco Systems.

978-1-4244-4867-8/09/$25.00 ©2009 IEEE

This paper aims at selecting the most effective set of module pins for fault insertion (FI), such that faulty values on these pins can model most physical defects that occur inside the module, thus mimicking the behaviors of hardware defects encountered in the customer environment. To achieve this goal, we have developed a simulation framework to study the propagation of physical defects on internal nodes to module pins. We have also developed optimization techniques, based on integer linear programming and heuristics to minimize the number of pins for FI and assign appropriate fault types for those selected pins. The pins targeted in this work are not limited to the pins at the chip level, but they also refer to the pins of sub-modules inside ASICs, since our method can be hierarchically applied to logic blocks within ASICs. The addition of error-forcing logic within ASICs and programmable devices permits an additional level of testing and reliability assessment.

The remainder of this paper is organized as follows. Section 2 provides an overview of FI test. Section 3 presents the fault-simulation framework and proposed selection method. Section 4 illustrates the design of experiments and experimental results for benchmark circuits as well as an industrial circuit. Section 5 concludes the paper.

2. Fault-insertion test

Hardware fault-insertion test (FIT) is a technique that intentionally inserts wrong values at pin/logic level to model the effects of manufacturing defects and in-field intermittent and transient errors [7], [8]. FIT is an essential tool to verify the robustness and error-handling capabilities of boards and systems and it is an accurate and objective method to verify the ability of diagnostics to accurately detect and diagnose failures at a board or system level [8]. FIT is a required system test step at companies such as Cisco Systems, Inc. because it is the only method that allows system hardware, integrated with system software, to be tested as a whole in the area of hardware-error handling. Error case studies can be done in simulation environments.

Hardware FIT verifies the error detection, handling, and recovery components of a system. This verification covers the following items [9]:

(1) Error Detection - the ability to detect abnormal operations as a result of hard, intermittent or transient errors within the design.

(2) Reconfiguration - in cases where redundancy may be in place, verifying that the involved reconfiguration components can correctly reconfigure the active faulty device to a redundant standby device.

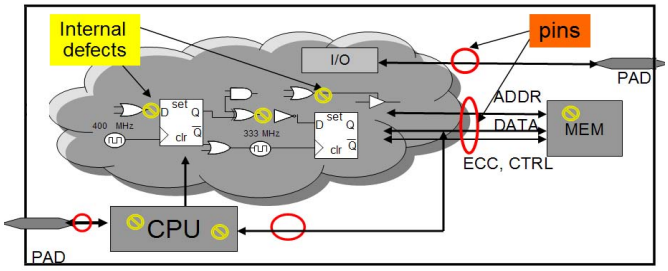

Figure 1: Illustration of fault insertion at the pin level.

(3) Notification/Alarming - the hardware and software components used to report an error condition to higher levels of software control and network operation personnel.

Hardware FIT in a system can be divided into three general categories: physical FI, software-implemented FI and simulated FI [10]. Physical FI is accomplished at the physical level, whereby the hardware is deliberately targeted by subjecting it to environmental stress conditions such as heavy ion radiation, electromagnetic interference, etc. Software-implemented FI, also called fault emulation, consists of reproducing at a software level, the errors that would have been produced if faults occurred in the hardware, e.g., the modification of memory data and the mutation of the application software or the lowest service layers [10]. For the simulated FI technique, the system under test is simulated in another computer system. The faults are induced by altering the logical values during simulation. Our work belongs to the category of simulated FI.

In simulated FI, one of the most difficult problems is that the fault space is extremely large. A fault inserted in a system can be specified by at least four parameters, namely fault insertion time, fault location, fault type, and fault duration. Taking all these parameters into account, the fault space is nearly infinite. To cope with the large size of the fault space, several methods have been described in the literature. In [11], a 3-stage random sampling technique is developed; however, a drawback of random selection is the high probability that the injected faults will remain latent, therefore leading to no-response fault insertion experiments. Another selection mechanism reported in [12] is based on fault-list generation. It is an ad hoc technique that depends on the analysis of dataflow and the instruction-set architecture.

In this paper, an implementation of FIT based on “pin-level forcing” is presented, as shown in Figure 1. The key idea is to construct fault models at the pin level of a module to represent physical defects inside the module. By applying FI at the pin level, instead of traditional full module FI, our methods shrink the large fault space in terms of fault location. This improves the efficiency of credible system reliability assessment and diagnosis coverage measurement.

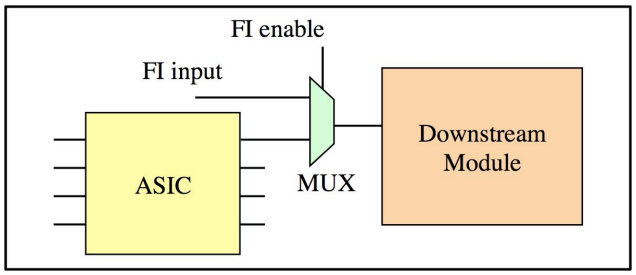

Figure 2: A typical hardware implementation of multiplexer-based FI.

For the hardware implementation of FI, several techniques are currently in use, such as manual FI, probe-based FI [8], boundary-scan logic based FI [7], [8], [9], and multiplexer-based FI. A simple example of a multiplexer-based FI implementation is illustrated in Figure 2. Faults are inserted at a pin of an ASIC using a FI logic element (multiplexer). If fault insertion is enabled, the faulty value is passed out through the multiplexer, otherwise the fault-free value is selected.

FIT is resource-limited in terms of silicon area, time, equipment, personnel and what is actually possible to test from one design to the next. More specifically, at the ASIC level, usually a small portion of chip IOs and internal nodes are equipped with FI logic elements to avoid excessive area and timing overhead. Hence, the selection of these FIT-enabled IOs or internal nodes becomes critical: these nodes should be able to represent as many effects of hard and transient faults as possible that occur inside the chip. The physical defect modeling proposed in this paper not only accelerates fault simulation in system-reliability assessment, but it also provides guidelines on the placement of FI logic elements in an actual hardware implementation.

3. Physical defect modeling for FIT

To construct an effective physical defect model, a selection scheme has been developed, whereby the most appropriate fault locations and fault types can be specified for FI. In our method, it is not necessary for the pin-level model to extract the exact response due to a physical defect. We want to generalize the similarities of the effects caused by physical defects, so that it is possible to use one fault at the pin level to represent many physical defects inside the module, thus reducing FI effort both in the simulation process and in the hardware implementation.

The construction of physical defect modeling begins with physical defect injection and concludes with an optimization step based either on integer linear programming or heuristics. Fours steps involved in the process of model construction are shown in Figure 3. According to the fault simulation results and constrained optimization algorithms, a minimum subset of output pins can be selected and the most appropriate fault types can be assigned to those pins for FI.

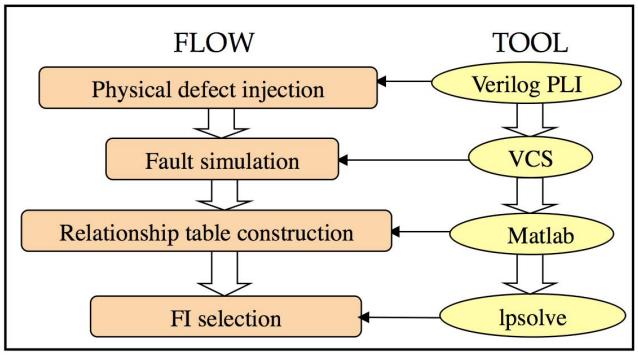

Figure 3: The steps involved in physical defect modeling.

Physical defect injection is first simulated, since FI at the pin level depends on the impact of the physical defects inside the module. Defects are injected using the Verilog Programming Language Interface (PLI). Based on our simulation infrastructure, the defect location, type, duration, start and end time all can be specified by the user. Fault simulation runs in VCS. Based on the faulty response of the circuit, a *relationship table* between injected defects and errors at the pin level (pin-faults) can be constructed. After the relationship table is constructed, the optimization problem of pin-fault selection for FIT can be formulated by an integer linear programming (ILP) model and solved by *lpsolve*, which can be used for solving pure linear, (mixed) integer/binary, semi-continuous and special ordered sets models [15]. In order to make the method scalable for large designs, a heuristic method is also developed to handle large data sets. Details of each step are presented in the following subsections.

3.1. Physical defect injection

A defect injection environment is designed by using Verilog Procedural Interface (VPI), which allows us to inject faults into a Verilog model of the underlying hardware. VPI is the third-generation Verilog programming language interface (PLI). [16] It uses a C-platform and consists of a set of accesses and routines that can be called from standard C programming language functions. These routines interact with the instantiated simulation modules and users can customize routines to control the nets in the compiled module, such as observing or forcing the logical values of the nests through Verilog tasks.

In our simulation framework, five VPI routines are written to inject five different types of defects. The following is the list of the function and arguments for each routine:

1. **$stuck_at (net, start_time, end_time, stuck_value):**
   It forces the value of 'net' to 'stuck_value' at 'start_time' and releases forcing at 'end_time'.
2. **$flip (net, start_time):**
   It changes the value of the 'net' to the complementary value at 'start time'. It is a one-time event, and there is no continuous forcing on that net.

3. **$bridging(net, dominant_net, start_time, end_time):**
   It forces the value of 'net' to be the same with the value of 'dominant_net' at 'start_time' and releases the forcing at 'end_time'.
4. **$glitch (net, start_time, pulse_width):**
   It adds a pulse to 'net' at 'start_time' and the width of the pulse is defined by 'pusle_width'.
5. **$delay (port, rising_delay, falling_delay):**
   It applies transition delay on 'port'; the rising-edge delay and falling-edge delay can be specified by 'rising_delay' and 'falling_delay'.

The above VPI routines can be integrated with a Verilog-based testbench and used to perform defect injection for arbitrary circuits. In Section 4, simulation results for stuck-at faults (stuck-at-0, stuck-at-1) as well as the complementation (flip) of the value on an internal line are presented. Stuck-at faults are used to mimic permanent defects, such as opens and shorts, that can occur in manufacturing. Flip faults are used for modeling transient faults, which is a common concern in modern system reliability evaluation [4], [5].

3.2. Fault Simulation

Fault simulation is typically used for the development of manufacturing tests. It determines the coverage for a given set of input stimuli and for a given fault model or a given fault list. With the help of other programs, it can produce a set of vectors with a given fault coverage for manufacturing test [17]. In this work, serial fault simulation is used to collect the information of faulty circuit response caused by injected defects. By injecting internal defects one by one and comparing the output responses of fault circuits with those of fault-free circuits, the impacts/mismatches at output pins caused by the internal defects are obtained. To better reflect the effects of physical defects, fault simulation is performed at the gate level and functional patterns are applied for the purpose of system reliability evaluation.

In fault simulation, to save storage space, only the mismatches between faulty circuits and fault free circuits are stored. More specifically, when the output response in fault simulation is different from the expected fault-free value, we save a record of the current pin identifier, injected defect identifier, test pattern identifier, faulty value, and expected value at that pin. According to the recorded mismatch information, we can determine which defects can be modeled by a specified pin-fault and construct the relationship table in the next step.

3.3. Relationship table (RT) construction

The relationship table is a two-dimensional matrix, similar to a fault dictionary. It is used to reflect the fact that a pin-fault can represent a defect. Here we assume a pin is able to represent an internal defect if the defect can manifest itself at that pin. Before providing more details on the RT, we first clarify the definition of pin-faults and three types of coverage that are considered in this paper.

3.3.1. Definition of pin-fault

To describe the faulty behavior at output pins, three types of pin-faults are defined: pin-stuck-at-0, pin-stuck-at-1 and pin-flip. They are abstracted from practical scenarios of FIT hardware implementation. Each pin-fault is associated with a time frame in terms of several clock cycles. Within the time frame, the logical value of an output pin is monitored. If the value meets the following constraints, a pin-fault occurs at that pin.

- **Pin-stuck-at-0**
  The logical value at the pin is either "*D*" or "0" and at least one "*D*" occurs in that time frame, where a D corresponds to 1/0 (1 in the good circuit, 0 in the faulty circuit)
- **Pin-stuck-at-1**
  The logical value at the pin is either "$\overline{D}$" or "1" and at least one "$\overline{D}$" occurs in that time frame, where a $\overline{D}$ corresponds to 0/1 (0 in the good circuit, 1 in the faulty circuit).
- **Pin-flip**
  The logical value at the pin is either "$\overline{D}$" or "*D*" and the value keeps flipping between "$\overline{D}$" and "*D*" in that time frame.

In the above fault models, we require that at least one "D" or "$\overline{D}$" occurs within the time frame. This guarantees that the pin has been corrupted in that period. For example, if the logical value of a pin stays "0" during fault simulation, while the golden value of the pin is always "0" in that period, there is no pin-stuck-at-0 fault occurring at that pin. In the remainder of this paper, the time frame is called a fault period.

3.3.2. Coverage measures

Due to our special definition of pin-faults, the internal defect coverage based on pin-faults is different from the traditional fault coverage measure. To be specific, three types of coverage of interest are listed below:

$$\text{Defect coverage} = \frac{\#\,\text{Detectable defects}}{\#\text{Total defects}}$$

$$\text{FI-Defect coverage} = \frac{\#\,\text{FI-Detectable defects}}{\#\text{Detectable defects}}$$

$$\text{Selection coverage} = \frac{\#\,\text{Selected-FI-Detectable defects}}{\#\text{FI-Detectable defects}}$$

Total defects refer to all the injected defects. Detectable defects are those defects that cause different logical values from golden values at output pins. FI-Detectable defects are those defects that result in pin-faults occurring at output pins. Moreover, if any pin corrupted by a FI-Detectable defect is selected for FI, this FI-Detectable defect is a Selected-FI-Detectable defect. Based on these definitions, we aim at defining an appropriate pin-fault model to achieve high FI-Defect coverage, and maximizing selection coverage.

3.3.3. Relationship table construction

Based on our optimization goal and the requirements of optimization methods, two relationship tables are constructed. One is group-fault versus defect RT, reflecting whether a group of pins with a designated fault type can represent a physical defect. An example is shown in Table 1.

In the group-fault versus defect RT, the rows stand for the identifier of a group with a designated fault type, and the columns of the table list the internal defects. The entries in the table are either "1" or "0". A value of "1" implies that the defect on the column can be represented by the group with the associated fault type on that row. Likewise, an entry of "0" implies that the group with the designated fault cannot represent the defect, which therefore means that type of pin-fault does not appear at any output pin in that group during fault simulation. In Table 1, there are three groups of output pins and eight defects. If we intend to select a set of groups and corresponding fault types to cover all the defects, we can choose group1 with pin-s-a-0 fault, group1 with pin-flip faults and group 2 with pin-flip fault to cover all the defects. A total of eight defects are covered by the selected rows, highlighted in Table 1.

The group versus defect RT reflects whether a group of output pins can represent a defect, without a specified fault type. An example is shown in Table 2. In this table, the entry is "1" if any pin of the group corresponding to the row is able to represent the defect corresponding to the column. If the entry is "0", it means that the group listed on that row cannot represent the defect, irrespective of the type of fault inserted on the pins in that group.

The reason why we assign a group of pins, instead of a pin, to each row in the relationship table is that the adjacent module pins are typically assembled in a group for FI in practical implementation. For example, in the hardware implementation of FIT at a system company, a group of pins are connected to the same type of FI logic element and all pins in that group are selected at once.

Table 1: An example of group-fault versus defect RT.

	d1	d2	d3	d4	d5	d6	d7	d8
group1, pin-s-a-0	1	0	1	0	0	0	0	0
group1, pin-s-a-1	0	0	0	1	0	1	1	1
group1, pin-flip	1	1	1	0	0	0	0	0
group2, pin-s-a-0	1	1	0	0	0	0	0	0
group2, pin-s-a-1	0	0	0	0	0	1	0	0
group2, pin-flip	0	0	0	1	1	0	1	0
group3, pin-s-a-0	0	1	0	0	0	0	0	0
group3, pin-s-a-1	0	0	0	0	0	1	0	0
group3, pin-flip	0	0	0	1	0	0	1	0

Table 2: An example of group versus defect RT.

	d1	d2	d3	d4	d5	d6	d7	d8
group1	1	1	1	1	0	1	1	1
group2	1	1	0	1	1	1	1	0
group3	0	1	0	1	0	1	1	0

3.4. Optimization algorithms for FI selection

In this section, optimization techniques, based on integer linear programming (ILP) and heuristics, are presented for minimizing the number of output pins for FI and for assigning fault types for those selected pins, such that desired coverage levels are obtained under constraints on the number of selected pins and the fault-insertion cost.

3.4.1. Integer linear programming model

Using fewer FI logic elements is the best way to save FI cost in the hardware implementation of FIT. To minimize the set of groups of pins for FIT, a two-step selection method is developed: first, select the group of pins without consideration of fault types; second, select the fault type for each selected group. This selection method can solve the single pin selection problem as well, when the group size is equal to one.

Step 1. Select the group of output pins

In the first step, we ignore fault types of the groups, and instead only minimize the number of groups to achieve the desired selection coverage. After the smallest set of groups is found, we use the second step selection to assign fault types for the selected groups. The minimum set of groups can be obtained by solving the following group-selection ILP model. Let the number of output pins in the circuit under test be p, and the group size be m. Therefore, the number of groups is p-m+1. Let the number of injected defects be n.

Group-selection ILP model:

Objective function: Minimize $\sum_{i=1}^{p-m+1} g_i$

Constraints: $\sum_{i=1}^{p-m+1} (g_i y_{ik}) \geq d_k,\ \forall k,\ k \in \{1,...,n\}$ (1)

$$\sum_{k=1}^{n} d_k \geq C \quad (2)$$

$$\sum_{j=i}^{i+m-1} g_j \leq 1 \quad \forall i \in \{1,..., p-2m+2\} \quad (3)$$

$$g_i,\ d_k \in \{0,1\}$$

$$C = \lceil \text{selection coverage} \times \#\text{FI-Detected defects} \rceil$$

In this model, our objective is to minimize the summation of g_i, where g_i is an indictor to indicate if group i is selected. If group i is selected, g_i =1, otherwise g_i =0. By minimizing the summation of g_i, we can get the smallest set of groups.

Constraints (1) and (2) ensure that the selected groups achieve the desired selection coverage. In constraint (1), the constant y_{ik} is the entry on the row i and k column in the group versus defect RT. The variable d_k can be either 1 or

0. If d_k =1, it means that defect k is successfully represented by at least one selected group. To satisfy constraint (1), at least for one group, $g_i y_{ik}$ is 1, when d_k =1. If d_k =0, constraint (1) is always satisfied. The constant C is the least integer greater than the product of selection coverage and the number of detectable defects by all the groups. By constraint (1) and (2), we do not specifically confine which defect should be represented, but the total number of detectable defects by selected groups should be greater than C, which ensures that the desired selection coverage can be achieved. In constraint (3), the summation is less than and equal to one, which implies that the groups consisting of the same pins cannot be simultaneously selected. This constraint is used to guarantee that there are no overlapped pins in the selected group, because one pin cannot belong to two FI logic elements in a practical implementation.

There are two special cases for this ILP model. One is that 100% selection coverage is required. Under this scenario, the model regresses to a simple coverage problem. For any defect k, d_k =1. Constraint (2) is always satisfied. There is no change for constraint (3). The other special case is single-pin selection. In this case, group size is equal to one, and constraint (3) is always satisfied.

Although one FI logic element is usually added to a group of pins in a practical FIT implementation, the optimization results for single pin selection can be used to derive a lower bound on the number of FI logic elements needed for group selection with different group size. The lower bound is given by the following formula:

$$\#\text{FI\_group\_m} \geq \text{lower bound} = \left\lceil \frac{\#\text{FI\_single}}{m} \right\rceil$$

Where #FI_group_m is the minimum number of FI logic elements needed for group selection; the group size is m; #FI_single is the minimum number of FI logic elements needed for single-pin selection.

Step 2. Select fault type for the selected group

In this step, our goal is to assign an appropriate fault type for each selected group. The overhead for different fault-type insertions varies. Stuck-at fault has lower FI cost, since stuck-at fault insertion can be performed by simply soldering a switch between V_{DD} or Gnd and the net under test, and closing the switch. This method is much cheaper, compared with loading a flipped signal to the faulty pin. The costs associated with different fault-type insertion are taken into account by adding weights corresponding to different fault types. Assuming the cost of stuck-at fault insertion is 1 and the cost of flip fault insertion is 2, the fault type selection ILP model is shown below.

Fault-type selection ILP model:

Objective function: Minimize $\sum_{i=1}^{2(p\text{-}m+1)} s_i + 2 \sum_{i=2(p\text{-}m+1)+1}^{3(p\text{-}m+1)} s_i$

Constraints: $\sum_{i=1}^{3(p\text{-}m+1)} (s_i y'_{ik}) \geq d_k \quad \forall k, k \in \{1,\ldots,n\}$ (4)

$$\sum_{k=1}^{n} d_k \geq \text{C} \quad (5)$$

$$s_i, d_k \in \{0,1\}$$

$$\text{C} = \lceil \text{selection coverage} \times \#\text{FI detected defects} \rceil$$

In the objective function, the variable s_i is an indicator for denoting if the group with the associated fault on the row i is selected, if the group with the associated fault is selected, s_i=1, otherwise s_i=0. The first part of the objective function is the summation of the indictors for pin-s-a-0 and pin-s-a-1 faults. The weights of those stuck-at faults indicators are 1. The second part is the summation of the indicators for pin-flip fault. The weights of those pin-flip fault indicators are 2.

Similarly, constraints (4) and (5) ensure that the selected groups with associated fault types can model a sufficient number of defects to achieve desired selection coverage. The constant y'_{ik} is the entry on the row i and column k in the updated group-fault v.s. defect RT. (The group-fault v.s. defect RT is updated by deleting all the unselected groups in step one.) The meanings of parameters p, n and m are the same with those in the first step and the variable d_k has the same meaning as well. The optimization results for the second ILP model provide the final fault-insertion decision, including the groups where the FI logic elements should be inserted and the fault types for each selected group. The overall procedure for the two-step selection technique is summarized below.

1. Prepare relationship tables:
   RT1: Group versus Defect RT
   RT2: Group-fault versus Defect RT
2. Construct the first model, group-selection ILP model using RT1
3. Solve the group-selection problem using the first ILP model
4. Update RT2
5. Construct the second model, fault-type selection ILP model using updated RT2
6. Solve the fault-type selection problem using the second ILP model

3.4.2. Heuristic algorithm (HA)

The increasing complexity of integrated circuits is resulting in a large number of physical defects in a chip or a module within a chip. In order to model the effect of these defects on chip/module pins, the optimization algorithm should be computationally feasible and able to handle large data sets to meet the requirement of state-of-the-art hardware FIT. To model as many defects as possible and handle large data sets, an efficient and scalable heuristic algorithm is presented next to select the subset of groups. The ILP models presented earlier are useful for deriving lower bounds using LP-relaxation, since linear programming problems can be solved in polynomial time [18].

In the following procedure, we describe a greedy heuristic algorithm to select the groups associated fault types for FIT. The heuristic algorithm is applied to the group-fault versus defect RT. HA first selects the group-fault that can model the largest number of defects, and then deletes the detected defects from the RT; it repeats this "selection-deletion" process to the updated RT until the required selection coverage is achieved.

HA Procedure: select groups and fault types for FI

```
1.  load group-fault versus defect RT  mat(i,j)
2.  while (#Selected-FI-Detectable defects < C)
3.  {
4.        find the row t with max sum(mat(:,t)');
5.        for all columns    % delete the detected defects
6.        {
7.             if mat(t,j)==1
8.              mat(:,j)=0;
9.         }
10.       % delete adjacent rows
11.       mat(t-m:t+m,:)=0
12.       update #Selected-FI-Detectable defects
13. }
```

4. Experiment Results

In this section, experimental results for an ISCAS benchmark circuit (c3540) is firstly presented to illustrate the flow of our optimization method, and then results for an OpenCores benchmark circuit (USB 1.1 function core) and an industrial RISC CPU core are reported. The latter two circuits represent modern communication and information processing systems. All experiments were performed on a pool of state-of-the-art servers running Linux. The Verilog simulator for all simulations is VCS (Y-2006.06-SP1-16). The ILP model was solved using lpsolve (5.5.0.14) [15]. For each circuit, error distribution at output pins is first illustrated. Next, the tradeoff curves between the number of selected output pins and the selection coverage is provided. Finally, the selection results of fault locations and fault types are presented.

4.1. Results for c3540

In the c3540 circuit, there are 50 inputs, 22 outputs, and 1647 internal wires. 169 ATPG patterns are generated for stuck-at faults using Fastscan. We use a stuck-at test set here for the sake of illustration. In actual practice, functional patterns based on actual data traffic must be used. Three types of defects, namely s-a-0, s-a-1, and flip, are injected on all the internal nets, one by one. The simulation time for handling all the faults is around one hour. The preparation of the relationship table takes nearly one minute of CPU time. After this, the ILP solver takes only a few seconds to complete the selection. For this example, heuristic algorithm runs about 2 times faster.

Figure 4 shows the error distribution on output pins. The X-axis lists the output pins. The Y-axis shows the number of defects captured by the corresponding output pin. In Figure 4, the number of defects captured by each output pin is different, which shows that there is significant difference between the output pins in terms of their ability to "capture" defects. This skewed distribution provides valuable evidence that it is meaningful to select the most effective set of output pins to represent defects; otherwise, if the distribution is uniform, there is no need to select an optimal set of output pins, because all the outputs are equivalent in respect of representing internal defects. Moreover, we find that the distributions for different types of defects are similar. More defects can be captured by pin 1 and pin 2. From this direct observation, pin 1 and pin 2 seem to be good choices for FI. This observation is corroborated by the selection results (derived from the optimization flow) shown in Table 3 and Table 4.

Figure 5 shows that tradeoff between the number of selected output pins and selection coverage. The curve is obtained by setting different selection coverage values in the ILP model and calculating the minimum number of output pins for each case. In Figure 5, we see that if we want to achieve 100% selection coverage, 21 FI logic elements are needed to fault 21 output pins out of 22

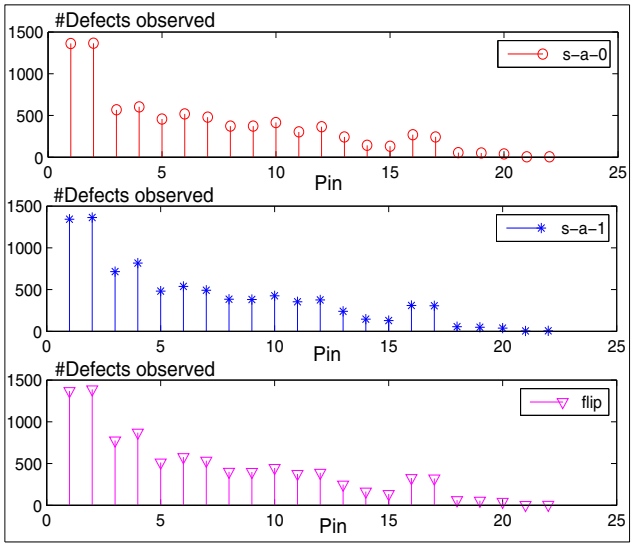

Figure 4: c3540 benchmark: error distribution on output pins.

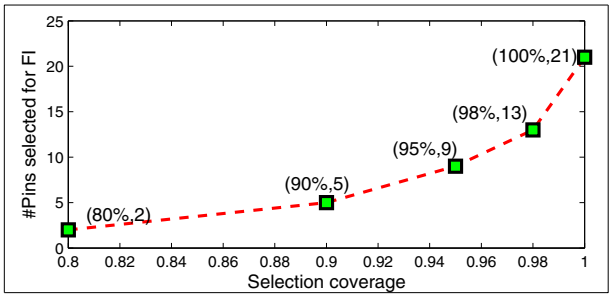

Figure 5: Tradeoff between #FIs and selection coverage.

output pins. This solution costs too much since it faults nearly all the output pins, which is not a good choice in practical FIT. If we sacrifice selection coverage slightly, 95% selection coverage can be achieved by adding FI logic to only 9 pins. This trade-off graph provides guidance for how many insertions are needed to achieve the desired coverage level, and for a fixed number of insertions, what level of selection coverage can be obtained.

Table 3 shows the single-pin and group selection results using the ILP-based and HA methods. The fault period (FP) is set to 8 time units. The selection coverage is set to 95%. The FI-Defect coverage is 62.4%. The results tell us where we should insert faults or add FI logic elements and what type of faults should be inserted. For example, the second column indicates that FI is done for 13 single pins; namely pins1, 2, 3, 5, 6, 11, 15, 16, 17, 18, 19, 20 are selected for pin-s-a-0 fault insertion; pins 1, 2, 3, 5, 6, 8, 11, 16, 17, 18, 19, 20 are selected for pin-s-a-1 fault insertion; pins 1, 2 are selected for pin-flip fault insertion. Fewer pin-flip faults are selected, since the cost of pin-flip-insertion is higher. The third column shows the group-selection results obtained using the ILP model for group size *m* equal to 2. In group selection, group *i* consists of pin *i* and pin *i*+1, if the group size is 2. The fourth and fifth columns list the HA selection results. The bottom row shows the total number of added FI logic elements. The heuristic algorithm provides a solution that uses more pins. Both optimization methods select Pin 1 and Pin 2 to insert faults, which is consistent with the error distribution in Figure 5.

To analyze impact of different fault periods in defect modeling, fault period is changed from 8 time units to 4 time units in Table 4. The FI-Defect coverage is improved to 94.6%. For single pin selection, at least 9 FI logics are needed. Complete selection results are list in Table 4.

Table 3: FI selection results for c3540: Part 1.

FP=8	ILP		Heuristic	
	single pin	Group in 2	single pin	Group in 2
Pin-s-a-0	1,2,3,5,6, 11,15,16,1 7,18,19,20	1,3,5,11,15, 17,19	1,2,3,5,6, 11,12,15, 16, 17,19	1,3,5,11, 15,18,
Pin-s-a-1	1,2,3,5,6,8 ,11,16,17, 18,19,20	1,5,11,15, 17,19	2,5,6,8,13,1 4,17,18, 20	3,5,7,13, 17,19,
Pin-flip	1,2	1	1	1
#FIs	13	7	16	10

Table 4: FI selection results for c3540: Part 2.

FP=4	ILP		Heuristic	
	Single pin	Group in 2	single pin	Group in 2
Pin-s-a-0	1,2,3,5,11, 14,16,20	1,4,10,15, 19	1,2,5,6,11,1 4,16,	1,5,10,15
Pin-s-a-1	1,2,3,5,11, 14,19,20	1,4,10,15, 19	1,2,3,10,11, 18,19,20	1,3,10, 17,19
Pin-Flip	1,2,11,14, 19,20	1	2	-
#FIs	9	5	12	7

4.2. Results for the USB circuit

In the USB circuit, there are 248 inputs, 116 outputs, and 7110 internal nets at the gate level. Functional patterns are taken from the OpenCores benchmark website. As before, three types of defects are injected. The simulation time needed to handle all the faults is nearly 8 hours. The generation of the relationship table costs takes 10 minutes. Following this, the ILP solver takes only 1 minute to complete the selection. As in the case of c3540, the heuristic algorithm runs about 2 times faster. The error distribution is shown in Figure 6, the tradeoff curve is shown in Figure 7, and the selection results are listed in Table 5.

In Figure 7, we can see our selection method is very effective for the USB circuit. With only one FI logic element, we can obtain 95% selection coverage. That most effective pin is pin 115, which is the most suitable for defect modeling via FI; see Figure 6. There are nearly1600

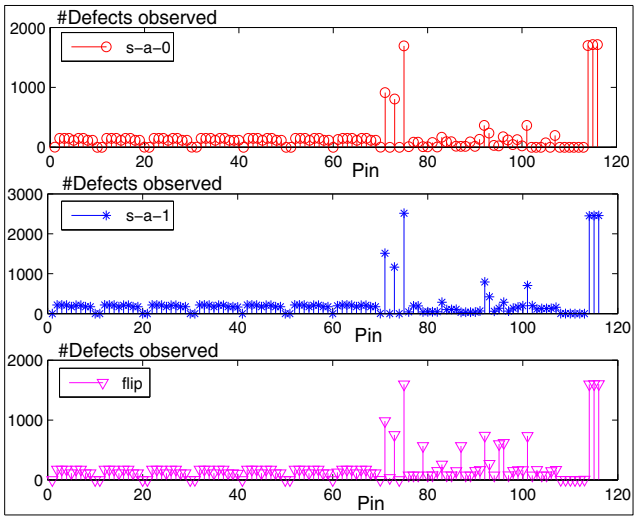

Figure 6: USB benchmark: error distribution on output pins.

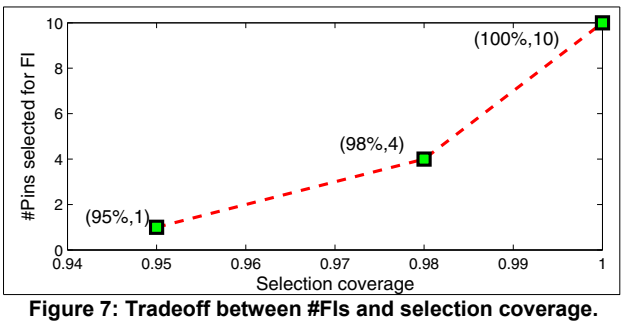

Figure 7: Tradeoff between #FIs and selection coverage.

Table 5: FI selection results for the USB circuit.

FP=8	ILP			Heuristic
	single pin	Group in 2	Group in 3	Single pin
Pin-s-a-0	115	114	113	75,115
Pin-s-a-1	70,71,75,95,96,99,108,109,114	70,74,95,98,108,	69,73,94,97,107,	71,75
Pin-flip	115	114	113	70,92,95,9698,99,108,109,114,115
#FIs	10	6	6	12

internal stuck-at faults manifesting at pin 115. However, there are no more than 200 faults showing up at any pin between pin1 and pin70. Compared with random selection, our selection technique is about 8 times more effective, assuming that random selection selects a pin between pin1 and pin70 with equal probability. The selection results in Table 5 are based on 100% selection coverage; a total of 10 output pins are selected for single pin selection. The fault period is set to 8 clock cycles. The FI-Defect coverage is as high as 98%, so there is no need to decrease the fault period for improving FI-Defect coverage. The lower bound on the number of FI logic elements for group selection and the heuristic selection results can be analyzed as before. For the USB circuit, we also find that the number of selected groups is 6, irrespective of whether the group size is 2 or 3. In this case, to decrease the impact on system performance due to fault insertion, we can place two pins in a group.

4.3. Results for the CPU circuit

The CPU circuit is a 32-bit RISC processor used by a system company. There are 68 inputs and 98 outputs. Three types of defects, including s-a-0, s-a-1 and flip, are injected on the 4764 internal nets. Functional patterns are used during fault simulation. The simulation time required for handling all the defects is around 6 hours. The generation of the relationship table takes nearly 5 minutes. Finally, the ILP solver takes only a few seconds to complete the selection, and the heuristic algorithm runs about 2 times faster. The defect distribution is shown in Figure 8, the tradeoff curve is illustrated in Figure 9, and selection results are listed in Table 6 and Table 7.

From the tradeoff curve, we find that at least 33 pins are needed to add FI logic elements, if we would like to obtain 100% selection coverage. Table 6 lists the selection results with 95% selection coverage. The fault period is again set to 8 clock cycles. FI-Defect coverage is 72.1%. The selection results show that no pin-flip faults are needed for the CPU circuit; stuck-at fault insertions are sufficient.

In Table 7, FI-Defect coverage is the same with that in Table 6, but the fault period is shorten to 4 clock cycles. In this experiment, we find that fewer FI logic elements are needed for the same FI-Defect coverage level, when pin-faults with shorter fault period are employed.

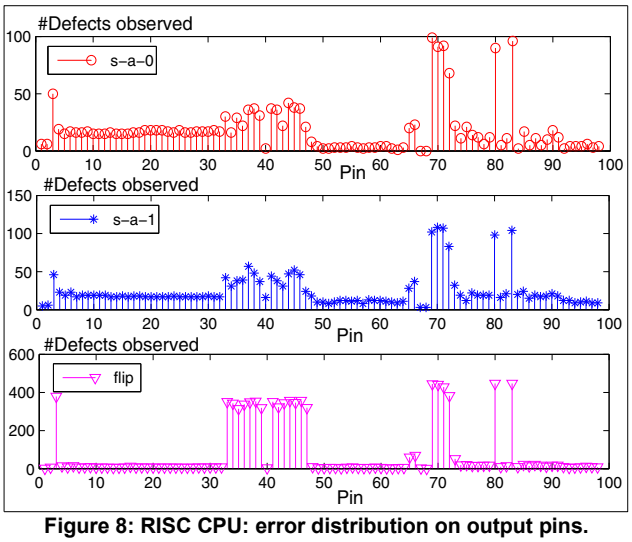

Figure 8: RISC CPU: error distribution on output pins.

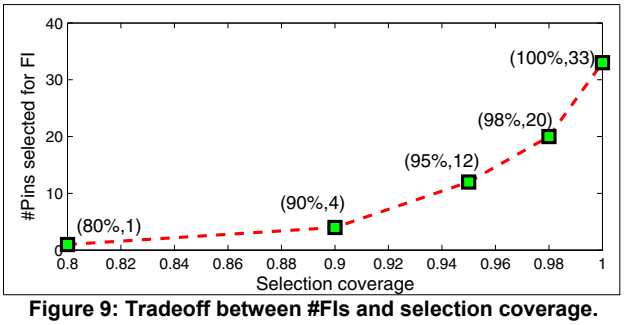

Figure 9: Tradeoff between #FIs and selection coverage.

Table 6: FI selection results for the CPU circuit: Part 1.

FP=8	ILP			Heuristic
	Single pin	Group in 2	Group in 3	single pin
Pin-s-a-0	66,70,71,79,81,84,85,89,92,96	70,72,75,81,84,88,91	70,75,84,88,91,	66,70,71,73,79,81,8485,89,92,96,
Pin-s-a-1	72,75	72,75	70,75	75
Pin-flip	-	-	-	-
#FIs	12	7	5	12

Table 7: FI selection results for the CPU circuit: Part 2.

FP=4	ILP			Heuristic
	single pin	Group in 2	Group in 3	single pin
Pin-s-a-0	70,71,81,85,87,89,91	70,74,84,89	70,84,89	70,71,81,85,88,90,92,96
Pin-s-a-1	75	70,74	70	75
Pin-flip	-	-	-	-
#FIs	8	4	3	9

Finally, for cross-validation purpose, another set of functional patterns is applied to the RISC CPU core. The goal is to verify that the selection results exhibit a fair degree of pattern independence. Experiments are repeated in the same manner and the selection results are similar. For single-pin selection, the most effective pins from before, namely pin 70, 75, 84, 85, 87, 89, 91, are selected

again. The reason why these pins are more representative than the others is that they have larger fan-in cones.

5. Conclusions

In this paper, a simulation framework and optimization techniques have been presented to guide the selection of several important parameters in hardware FIT, such as the number of FI logic elements, FI-Defect coverage, fault period, and the group size of pins. The selection method utilizes ILP-based optimization and a greedy heuristic algorithm to determine the optimal FI locations and fault types to mimic the effects of the defects within the module. By selecting the optimal FI locations, the efficiency of FIT-based system reliability assessment is improved. Several interesting observations have been made for test cases, and these observations provide valuable guidelines on the best way to implement hardware FIT.

References

[1] C. Constantinescu, "Impact of deep submicron technology on dependability of VLSI circuits", *Proc. International Conference on Dependable Systems and Networks*, pp. 205–209, 2002.

[2] N. Lopez-Benitez, K.S. Trivedi, "Multiprocessor performability analysis", *IEEE Transactions on Reliability*, vol. 42, no. 4, pp. 579–587, 1993.

[3] A. Correcher, E. Garcia, F. Morant, E. Quiles and R. Blasco-Gimenez, "Intermittent failure diagnosis in industrial processes", *Proc. IEEE International Symposium on Industrial Electronics*, pp. 723–728, 2003.

[4] N. J. Wang, J. Quek, T. M. Rafacz, S. J. Patel, "Characterizing the effects of transient faults on a high-performance processor pipeline", *Proc. IEEE International Test Conference*, pp. 61–70, 2004.

[5] P. Kudva, J. Kellington, P. Sanda, R. McBeth, J. Schumann, and R. Kalla, "Fault injection verification of IBM POWER6 soft error resilience", *Architectural Support for Gigascale Integration (ASGI) Workshop*, San Diego, CA, 2007.

[6] K. K. Goswami, R. K. Iyer, and L. Young, "DEPEND: A simulation-based environment for system level dependability analysis", *IEEE Transactions on Computers*, vol. 46, no. 1, pp. 60–74, 1997.

[7] B. Huang, M. Rodriguez, M. Li and C. Smidts, "On the development of fault injection profiles", *Proc. Reliability and Maintainability Symposium*, pp. 226–231, 2007.

[8] B. Eklow, A. Hosseini, K. Chi, S. Pullela, T. Vo and H. Chau, "Simulation based system level fault insertion using co-verification tools", *Proc. IEEE International Test Conference*, pp.704–710, 2004.

[9] S. Chau, "Fault injection boundary scan design for verification of fault tolerant systems", *Proc. IEEE International Test Conference*, pp. 677–682, 1994.

[10] D. Gil, J. Gracia, J. C. Baraza, P. J. Gil, "A study of the effects of the transient fault injection into the VHDL model of a fault-tolerant microcomputer system", *IEEE International On-Line Testing Workshop*, pp. 73–79, 2000.

[11] C. Constantinescu, "Inferring coverage probabilities by optimum 3-stage sampling", *Reliability and Maintainability Symposium*, pp. 22–25, 1996.

[12] D. T. Smith, B. W. Johnson, J. A. Profeta, "System dependability evaluation via a fault list generation algorithm", *IEEE Transactions on Computers*, vol. 45, no. 8, pp. 974–979, 1996.

[13] "IEEE standard test access port and boundary-scan architecture", *IEEE Standard 1149.1a*, 1993.

[14] R. Sedmak, "Boundary-scan: beyond production test", *Proc. IEEE VLSI Test Symposium*, pp. 415–420, 1994.

[15] lpsolve. http://sourceforge.net/projects/lpsolve

[16] "IEEE standard Verilog hardware description language", *IEEE Standard 1364*, 2001.

[17] M. L. Bushnell and V. D. Agrawal, "*Essentials of Electronic Testing*", Norwell, MA: Kluwer, 2005.

[18] D. Bertsimas and J. Tsitsiklis, "*Introduction to Linear Optimization*", Belmont, MA: Athena Scientific, 1997.