New Solutions for Optimal Hardware Tests of Reconfigurable Hardware Systems

Asma Ben Ahmed[1,2], Olfa Mosbahi[1] and Mohamed Khalgui[1]
[1]*LISI Laboratory, National Institute of Applied Sciences and Technology , University of Carthage, Carthage, Tunisia*
[2]*Tunisia Polytechnic School, University of Carthage, Carthage, Tunisia*

Keywords: Embedded System, Reconfiguration, Hardware Verification, Testing, Fault Collapsing, Optimization.

Abstract: This research paper deals with Reconfigurable Hardware Systems (abbreviated, RHS) that should be adapted to their environment under well-defined conditions. A reconfiguration scenario is a run-time hardware operation allowing the addition/removal of hardware components. We classify the reconfiguration scenarios into three levels: Architectural, Structural and Data Reconfiguration Levels. We propose a new solution for optimal hardware tests of RHS based on the definition of new fault collapsing relationships termed Inter-Equivalence, Inter-Dominance and Redundancy.

1 INTRODUCTION

In recent years, embedded systems have become ubiquitous at a fast rate. An embedded system is a computing machine designed for specific functions within a larger system. Embedded systems often require stringent performance, energy efficiency and flexibility for multifunctional use. In order to cope effectively and timely with these requirements, reconfiguration of system components at different levels of abstraction has become a crucial task to change systems during their execution while preserving their availability. In the literature, we have two reconfiguration policies: static and dynamic reconfigurations such as static reconfigurations applied off line (Angelov et al., 2005), dynamic reconfigurations are applied at run-time in two forms: manual reconfigurations applied by users (Rooker et al., 2007) and automatic reconfigurations applied by intelligent agents (Khalgui, 2010). The reconfiguration can touch the software level (Gharbi et al., 2010) and the underlying hardware (Ahmadinia, 2007) .

Reconfigurable Hardware System (RHS) allows post-fabrication configurability, enabling single base hardware design to implement a variety of circuitries. Furthermore, it permits the system to transform the underlying hardware to be adapted to temporal requirements of an application when internal or external changes occur. The quality and the correctness of hardware parts are greatest concerns in embedded systems. Hardware verification such that hardware testing ensures that the hardware components contains no defects that could affect the products correct functioning (Steininger, 2000).

One approach for considerably reducing the cost of the testing process is fault collapsing. Fault collapsing is the process of generating a reduced fault set in a circuit using the Equivalence and the Dominance relationships and is classified as structural or functional (Prasad et al., 2002). These techniques can be said Intra-Equivalence and Intra-Dominance fault collapsing as well as they are interested in faults located at the same circuit.

For RHS, the number of faults can become very large. Our problem consists in reducing the hardware test cost of reconfigurable system. It is thus beneficial to minimize the set of faults whenever possible. Greater reduction is possible by defining new techniques reducing faults from different circuits of the device under test. Our proposition is original and different from all others since no one proposes hardware tests for RHS. In this research work, we propose a new solution for optimal hardware tests of RHS. The hardware test is addressed at the gate level and we consider the single stuck-at fault model. We define the RHS as a network of components interconnected via signals. To address all possible forms of reconfiguration, we classify the reconfiguration scenarios into three levels: the first level deals with addition/removal of gates. The second deals with activation/deactivation of internal signals between already used components and the third consists in selecting

Ben Ahmed A., Mosbahi O. and Khalgui M..
New Solutions for Optimal Hardware Tests of Reconfigurable Hardware Systems.
DOI: 10.5220/0005243902810288
In *Proceedings of the 5th International Conference on Pervasive and Embedded Computing and Communication Systems* (PECCS-2015), pages 281-288
ISBN: 978-989-758-084-0

data via a multiplexer. For the purpose of optimizing both the set of faults for the RHS and the set of test vectors for the test generation process, we propose new fault collapsing relationships termed Inter-Equivalence, Inter-Dominance and Redundancy reducing faults from different circuits of the RHS. Our experiments show that the size of collapsed fault set is reduced to just 8 faults for an initial collapsed fault set of 108 faults, when the Inter-circuits relations are considered. With the traditional relations, we obtain 63 faults. Therefore, the number of faults to be considered in test generation and fault diagnostic can be decreased considerably, so the overall run-time can be reduced.

The rest of this paper is organized as follows. Next section is devoted to the background. Section 3 introduces the case study. Section 4 presents the formalization of RHS while section 5 describes the new characterization of faults in RHS. Section 6 shows the implementation. Finally, section 7 presents the conclusion.

2 BACKGROUND

In hardware testing, physical defects are abstracted into logical fault model. In fact, many fault models have been proposed such as stuck-at faults, bridging faults, delay fault models, etc (Abramovici and Menon, 1997). The most widely used model is the single stuck-at fault (Al-Asaad and Lee, 2002). To test a circuit under a given fault model, a set of input vectors is applied to the inputs of the circuit. Then, the output responses are compared with the fault-free responses to determine whether the circuit is faulty. Circuits that fail to produce correct responses are assumed to be faulty. An input vector is said to be a test vector if it produces a different output response from that of the fault-free circuit. Consequently, we have for an input vector, V, to be a test vector:

$$f_0(V) \oplus f_1(V)=1$$

where f_0 is the fault-free function and f_1 is the faulty function.

We present the gate level single stuck-at fault model that will be used in this paper. Two properties characterize a single stuck at fault, (i) just one line is faulty (ii) the faulty line is permanently set to constant logic value, either logic 0 or 1, referred to as stuck-at-0 (Line/0) or stuck-at-1 (Line/1), respectively. Under this model, the number of single stuck-at faults associated with each gate is twice the total number of inputs and outputs of the gates.
Fault collapsing reduces the number of faults using two relationships among faults: fault Equivalence and fault Dominance.

Equivalence fault collapsing: Two faults f_i and f_j in a circuit C are said to be equivalent (i) if the corresponding faulty output of the circuits C^{f_i} and C^{f_j}, respectively, are identical for every input vector applied to the circuit (ii) therefore f_i and f_j have exactly the same set of test vectors. Hence, the set of vectors that detect f_i, can also detect f_j. Consequently, if C^{f_i} is simulated then C^{f_j} has not to be simulated (Veneris et al., 2004).

Dominance fault collapsing: A fault f_i is said to dominate a fault f_j if all tests that detect f_j also detect f_i, but detecting f_i does not mean that f_j is also detected (Sethuram et al., 2008).

Note that no one treated the problem of fault collapsing for a RHS. All related works are interested in defining faults located at the same circuit. No relationships among faults of different circuits were given.

3 CASE STUDY

We present, in this section, a case study to expose our problem and to be assumed in the following as a running example. Let we assume a system with different behaviors. The system is considered as reconfigurable offering different services and implemented with different logics. Let C_1, C_2, C_i, ..., C_{10}, be combinational logic circuits with primary inputs, S_1, S_2, S_3, S_4 and one output Z_i, with $1 \leq i \leq 10$. We assume that the 10 circuits are constructed using gates AND, NAND, OR, NOR and XOR. Each circuit C_i implements a logic function, as shown in figure 1.

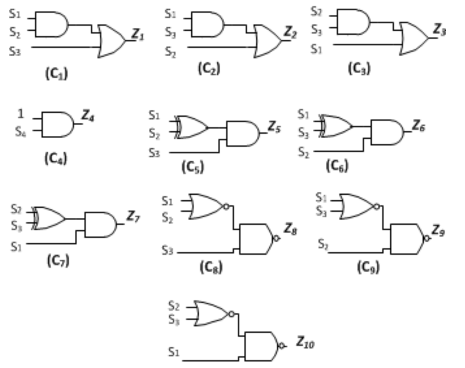

Figure 1: Logic functions implementing the system.

Assuming that each circuit C_i implements a reconfiguration scenario characterized by corresponding gates, signals and data. In order to make the system flexible, we propose to group these circuits in a single basic design where each circuit can be executed at a given time when a corresponding reconfiguration is applied. In fact, to apply a hardware reconfiguration, a multiplexer should be attached to the design such as writing a new set of values into the input selection reconfigures the hardware to implement a different circuit as shown in Figure 2.

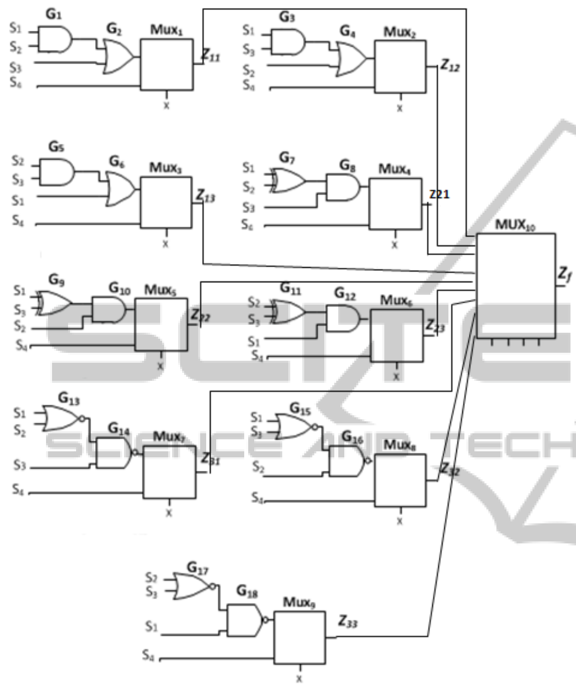

Figure 2: The proposed system.

Let F = $\{f_1, f_2, \ldots, f_n\}$ be the initial fault list that contains all possible stuck-at faults for the proposed system. According to the proposed case study, the initial set of faults F contains 108 faults. The authors propose in (Agrawal et al., 2003) a fault collapsing using two relationships among faults in one circuit: fault equivalence and fault dominance. By applying these relationships, we can reduce the number of faults about 58.3%. We obtain 63 faults.

When comparing faults from different circuits, redundancy of faults is possible so that inter-circuits relations among faults do exist. As the number of faults has a strong influence on the costs which must be paid for the test pattern generation, it is interesting to identify inter-circuits relations and to generate an optimum fault list and an optimum set of test vectors. Therefore, new relations should be proposed. This paper is original since it deals with reconfigurable systems, and defines new relations between faults in order to optimize the test process.

4 CONTRIBUTION: RECONFIGURABLE HARDWARE SYSTEM RHS

In our research, we are interested in Reconfigurable Hardware Systems that we denote by RHS in the following. RHS is considered as a network of gates interconnected via signals and expressed in terms of data. A hardware reconfiguration will be any operation allowing the activation (addition) and deactivation (removal) of hardware components at run-time. Therefore, at a given time (t), a change may affect the gates, the signals or the input data. In order to switch from a reconfiguration to another, a multiplexer is attached to the system such that multiplexer inputs present the different circuits implementing the different reconfigurations. Assigning new values into the selector pins reconfigures the system to implement a different circuit. To address all possible forms of reconfiguration, we define three levels: (i) *Architectural reconfiguration* triggered by creating, deleting or updating gates, (ii) *Structural reconfiguration* allowing activation/deactivation of internal signals between already used components, (iii) *Data reconfiguration* allowing selection of data via a multiplexer. This original classification covers all possible forms of reconfigurations to dynamically change the behavior of an RHS depending on the modifications that occurred in its environment.

In order to control the increasing complexity of the RHS and to cover all forms of reconfiguration, a hierarchical design is becoming more and more important. We propose a design with three levels of hierarchy: the top level is the architecture, the second level is the structure and the third level presents data.
1) Architectural Reconfiguration: consists in transforming an architecture into another one at the same level of abstraction. Such reconfiguration is applied by the addition and/ or the removal of gates.

Running Example 1:
According to the case study, the change from C_1 to C_8 represents an architectural reconfiguration scenario occurred by the removal of gates AND and OR and the addition of gates NOR and NAND. We switch from C_1 to C_5 by the addition of the XOR gate (see Figure 3).

2) Structural Reconfiguration: modifies the structural level by the activation or the deactivation of signals between already used gates as shown in Figure 4. We keep the same gates but the signals between gates are modified.

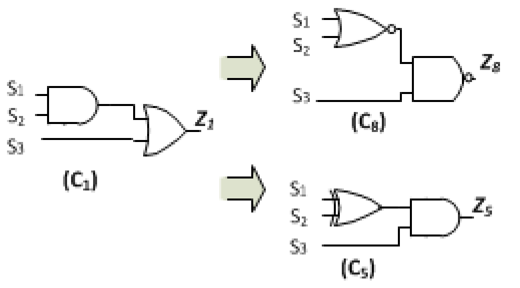

Figure 3: Architectural reconfiguration.

Running example 2:
Through the example given in Figure 4, we notice that AND and OR gates are retained. Z_2 and Z_3 are expressed as follows: Z_2=$S_1.S_3$+S_2 ; Z_3=$S_2.S_3$+ S_1

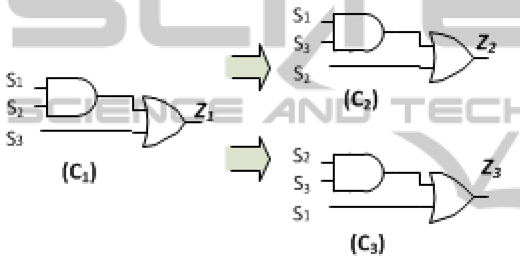

Figure 4: Structural reconfiguration.

3) Data reconfiguration: consists in adding a multiplexer (data selector) in order to express the system differently.

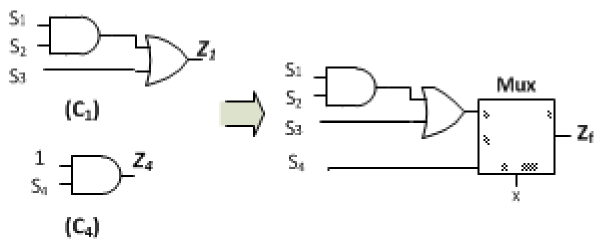

Figure 5: Data reconfiguration.

RHS is defined by the set of triples α, β and φ as follows:

RHS ={α, β, φ } where α = {{components}}, β = {{Signals}} and φ = {{Data}}.

At a given time (t), the system is characterized as follows: RHS(t) = {X, Y, Z}, where $X \in$ α ; $Y \in$ β ; $Z \in$ φ. X, Y and Z are respectively the set of gates, the set of signals and the data set that have to implement the system at a given time (t).
Let Ω be the initial fault list that contains all stuck-at faults of the proposed system. Ω is initially equal to 108 faults.

Running Example 3:
In the example shown in Figure 5 , if x=0 then Z_f= $S_1.S_2$+S_3. if x=1 then Z_f=S_4.

Running Example 4: Based on these scenarios, the RHS example is composed of three main architectures ($Archi_1$, $Archi_2$, $Archi_3$). Each architecture comprises three different structures $Struct_{ij}$, with $1 \leq i \leq 3$; $1 \leq j \leq 3$. Each structure can be expressed in terms of S_1, S_2 and S_3 or in terms of S_4 depending on input selection values. 16-to-1 Multiplexer is added to our model in order to enable reconfigurations. Each circuit presenting the inputs of 16-to-1 multiplexer is expressing architectural, structural and data reconfiguration. For the totality of our system, we have 18 possible reconfigurations as shown in Figure 6.

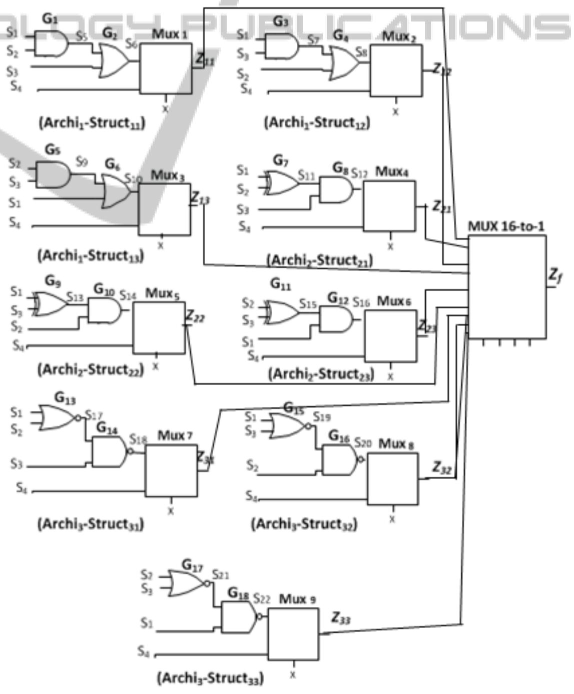

Figure 6: Reconfigurable Hardware System.

5 CONTRIBUTION: NEW CHARACTERIZATION OF FAULTS IN RHS

5.1 Classification of Faults

We define three new relations between faults: fault

Running Example 5:
Continuing from the running example n4, α, β and φ can take the following values:
α = {{$G_1(11)$, $G_2(11)$, $Mux_1(11)$}, {$G_3(12)$, $G_4(12)$, Mux_2}, {$G_5(13)$, $G_6(13)$, $Mux_3(13)$}, {$G_7(21)$, $G_8(21)$, $Mux_4(21)$}, {$G_9(22)$, $G_{10}(22)$, $Mux_5(22)$}, {$G_{11}(23)$, $G_{12}(23)$, $Mux_6(23)$}, {$G_{13}(31)$, $G_{14}(31)$, $Mux_7(31)$}, {$G_{15}(32)$, $G_{16}(32)$, $Mux_8(32)$}, {$G_{17}(33)$, $G_{18}(33)$, $Mux_9(33)$}}
β ={{S_1, S_2, S_3, S_4, S_5, S_6, Z_{11}}, {S_1, S_2, S_3, S_4, S_7, S_8, Z_{12}}, {S_1, S_2, S_3, S_4, S_9, S_{10}, Z_{13}}, {S_1, S_2, S_3, S_4, S_{11}, S_{12}, Z_{21}}, {S_1, S_2, S_3, S_4, S_{13}, S_{14}, Z_{22}}, {S_1, S_2, S_3, S_4, S_{15}, S_{16}, Z_{23}}, {S_1, S_2, S_3, S_4, S_{17}, S_{18}, Z_{31}},{S_1, S_2, S_3, S_4, S_{19}, S_{20}, Z_{32}},{S_1, S_2, S_3, S_4, S_{21}, S_{22}, Z_{33}}}
φ= {{S_1, S_2, S_3}, {S_4}}

Running Example 6:
Continuing from the previous running example, we suppose that the composition $Archi_1$-$Struct_{11}$ with x=0, initially, implements the RHS then we have
RHS(t_1) = {{G_1, G_2, Mux_1}, {S_1, S_2, S_3, S_4, S_5, S_6, S_{11}}, {S_1, S_2, S_3}}.
At a given time t_2, a change may affect the architectural level. The $Archi_2$-$Struct_{21}$ is deployed. The RHS is expressed as follows:
RHS(t_2) = {{G_7, G_8, Mux_4}, {S_1, S_2, S_3, S_4, S_{11}, S_{12}, Z_{21}}, {S_1, S_2, S_3}}.
At a given time t_3, a structural reconfiguration can occur: $Archi_2$-$Struct_{23}$ is deployed.
RHS(t_3) = {{G_7, G_8, Mux_4}, {S_1, S_2, S_3, S_4, S_{15}, S_{16}, Z_{23}}, {S_1, S_2, S_3}}.
Note that just internal signals are modified.
At t_4, if the selector input x switch from x=0 to x=1 then a data reconfiguration occurs.
RHS(t_4) = {{G_7, G_8, Mux_4}, {S_1, S_2, S_3, S_4, S_{15}, S_{16}, Z_{23}}, {S_4}}

Inter-Equivalence, fault Inter-Dominance and fault Redundancy.

Definition 1: Fault Inter-Equivalence
Let V^{f_i} and V^{f_j} be two sets of vectors that detect, respectively, f_i and f_j. Two faults f_i and f_j in different circuits C_i and C_j are said to be Inter-Equivalent (i) if they are detected by exactly the same set of vectors $V^{f_i} = V^{f_j}$, (ii) the input-output logic behaviors of $C_i^{f_i}$ and $C_j^{f_j}$ are not necessary identical. If two faults f_i and f_j are Inter-Equivalent, then only one of these faults, say f_i, needs to be retained in the fault list.

Definition 2: Fault Inter-Dominance
Let f_i and f_j be two faults in different circuits C_i and C_j. Let V^{f_i} be the set of all the vectors that detect the fault f_i. Similarly, let V^{f_j} be the set of the vectors that detect the fault f_j. The Fault f_i is said to Inter-Dominate the fault f_j (i) if V^{f_i} contains V^{f_j} that means $V^{f_j} \subset V^{f_i}$, (ii) the vectors that detect f_j do not necessary imply identical values at the corresponding faulty outputs of $C_i^{f_i}$ and $C_j^{f_j}$. If a fault f_i dominates a fault f_j, then each vector that detects f_j detects f_j. Therefore, f_i can be deleted from the fault list.

Definition 3: Fault Redundancy
A fault f_i is called redundant in different circuits C_i and C_j (i) if it is located at the same primary signal used by the circuits C_i and C_j. If the fault appears in the circuit C_i, then it will also appear at the same signal of C_j (ii) if the fault is at an internal signal of the circuit C_i and the same signal is present in C_j, then f_i is said to be redundant in C_i and C_j when f_i is detected by the same set of vectors. From the Redundancy fault set, only one fault can be retained to represent all faults in the corresponding fault set.

5.2 Application to Case Study: Minimization of Faults for RHS

Consider the proposed system shown in Figure 6. It has a total of 108 single stuck-at faults that can be reduced to 63 faults if we use the Intra-Equivalence and the Intra-Dominance relationships. All faults collapsing results are shown in Table 1: the number of collapsed faults and the Collapse Ratio are given. The latter is defined as (Bushnell, 2001):

$$\text{Collapse Ratio} = \frac{set of collapsed faults}{set of all faults}$$

Table 1: Fault collapsing results.

Circuit name	All faults	Number of collapsed faults (Collapse Ratio)	
		Intra-Equivalence	Intra-Dominance
$Archi_1$-$Struct_{11}$	12	4(0.33)	2(0.16)
$Archi_1$-$Struct_{12}$	12	4(0.33)	2(0.16)
$Archi_1$-$Struct_{13}$	12	4(0.33)	2(0.16)
$Archi_2$-$Struct_{21}$	12	2(0.16)	1(0.083)
$Archi_2$-$Struct_{22}$	12	2(0.16)	1(0.083)
$Archi_2$-$Struct_{23}$	12	2(0.16)	1(0.083)
$Archi_3$-$Struct_{31}$	12	4(0.33)	2(0.16)
$Archi_3$-$Struct_{32}$	12	4(0.33)	2(0.16)
$Archi_3$-$Struct_{33}$	12	4(0.33)	2(0.16)

A collapse ratio around 0.4 is quite typical for Intra-Equivalence and Dominance collapsing. However, considering Inter-circuits relationships, which is the main topic of this paper, may provide smaller collapsed fault set.

Inter-circuits fault collapsing reduces the number of faults using three relationships among faults: Inter-Equivalence, Inter-Dominance and Redundancy. For $Archi_1$, when comparing faulty versions of $Struct_{11}$, $Struct_{12}$ and $Struct_{13}$, faults $S_3/1(12)$ from $Struct_{12}$ and $S_3/1(13)$ from $Struct_{13}$ are dropped since they are Inter-Equivalent to $S_2/1(11)$ and $S_1/1(11)$, respectively. Also, faults $S_3/0$ (11), $S_2/0(12)$ and $S_1/0(13)$ are dropped since they Inter-Dominate $S_1/0(12)$, $S_1/0(11)$ and $S_1/1(11)$, respectively. Faults $S_1/0(12)$, $S_1/1(12)$, $S_4/0(12)$, $S_4/1(12)$, $S_2/1(13)$, $S_4/0(13)$ and $S_4/1(13)$, which are derived from primary signals, are dropped since they are redundant in $Struct_{12}$ and $Struct_{13}$. The remaining faults for $Struct_{11}$ are thus $S_1/0$, $S_1/1$, $S_2/1$, $S_4/0$, $S_4/1$. This set of faults covers all faults for $Struct_{12}$. For $Archi_1$-$Struct_{13}$, only the fault $S_2/0$ remains. The same process is repeated for the other architectures. With respect to the proposed model, our hierarchical approach to fault collapsing is as follows: we first look for Inter-Equivalence, Inter-Dominance and Redundancy relationships among all faults of different structures of the same architecture. This process is called Intra-Architecture fault collapsing. The latter is then followed by Inter-Architectures fault collapsing where we look for Inter-Equivalence, Inter-Dominance and Redundancy relationships among remaining faults of different architectures. By applying the proposed approach, we obtain the following results as shown in Tables 2, 3 and 4. The Tables present, respectively, Intra-Architecture fault collapsing results for $Archi_2$, Intra-Architecture fault collapsing results for $Archi_3$ and finally Inter-Architectures fault collapsing results for $Archi_1$-$Archi_2$-$Archi_3$.

Table 2: Intra-Architecture fault collapsing results for $Archi_2$.

Initial fault list For:		
$Archi_2$-$Struct_{21}$=$\{S_1/0, S_1/1, S_2/0, S_2/1, S_3/0, S_3/1, S_4/0, S_4/1, S_{11}/1\}$		
$Archi_2$-$Struct_{22}$=$\{S_1/0, S_1/1, S_2/0, S_2/1, S_3/0, S_3/1, S_4/0, S_4/1, S_{13}/1\}$		
$Archi_2$-$Struct_{23}$=$\{S_1/0, S_1/1, S_2/0, S_2/1, S_3/0, S_3/1, S_4/0, S_4/1, S_{15}/1\}$		
Number of collapsed faults (Collapse Ratio)		
Inter-Equivalence	Inter-Dominance	Redundancy
2(0.074)	0(0.0)	14(0.51)
Remaining faults:		
$Archi_2$-$Struct_{21}$=$\{S_1/0, S_1/1, S_2/0, S_2/1, S_3/0, S_3/1, S_4/0, S_4/1, S_{11}/1\}$		
$Archi_2$-$Struct_{22}$=$\{S_{13}/1\}$		
$Archi_2$-$Struct_{23}$=$\{S_{15}/1\}$		

Table 3: Intra-Architecture fault collapsing results for $Archi_3$.

Initial fault list For:		
$Archi_3$-$Struct_{31}$=$\{S_1/0, S_1/1, S_2/0, S_3/1, S_4/0, S_4/1\}$		
$Archi_3$-$Struct_{32}$=$\{S_1/0, S_1/1, S_2/1, S_3/0, S_4/0, S_4/1\}$		
$Archi_3$-$Struct_{33}$=$\{S_1/0, S_1/1, S_2/0, S_3/0, S_4/0, S_4/1\}$		
Number of collapsed faults (Collapse Ratio)		
Inter-Equivalence	Inter-Dominance	Redundancy
4(0.22)	0(0.0)	8(0.44)
Remaining faults:		
$Archi_3$-$Struct_{31}$=$\{S_1/0, S_1/1, S_2/0, S_3/1, S_4/0, S_4/1\}$		
$Archi_3$-$Struct_{32}$=$\{\}$		
$Archi_3$-$Struct_{33}$=$\{\}$		

Table 4: Inter-Architectures fault collapsing results for $Archi_1$-$Archi_2$-$Archi_3$.

Initial fault list For:		
$Archi_1$-$Struct_{11}$=$\{S_1/0, S_1/1, S_2/1, S_4/0, S_4/1\}$		
$Archi_1$-$Struct_{12}$=$\{\}$		
$Archi_1$-$Struct_{13}$=$\{S_2/0\}$		
$Archi_2$-$Struct_{21}$=$\{S_1/0, S_1/1, S_2/0, S_2/1, S_3/0, S_3/1, S_4/0, S_4/1, S_{11}/1\}$		
$Archi_2$-$Struct_{22}$=$\{S_{13}/1\}$		
$Archi_2$-$Struct_{23}$=$\{S_{15}/1\}$		
$Archi_3$-$Struct_{31}$=$\{S_1/0, S_1/1, S_2/0, S_3/1, S_4/0, S_4/1\}$		
$Archi_3$-$Struct_{32}$=$\{\}$		
$Archi_3$-$Struct_{33}$=$\{\}$		
Number of collapsed faults (Collapse Ratio)		
Inter-Equivalence	Inter-Dominance	Redundancy
1(0.04)	5(0.21)	9(0.39)
Remaining faults:		
$Archi_1$-$Struct_{11}$=$\{S_1/0, S_1/1, S_2/1, S_4/0, S_4/1\}$		
$Archi_1$-$Struct_{12}$=$\{\}$		
$Archi_1$-$Struct_{13}$=$\{S_2/0\}$		
$Archi_2$-$Struct_{21}$=$\{S_{11}/1\}$		
$Archi_2$-$Struct_{22}$=$\{\}$		
$Archi_2$-$Struct_{23}$=$\{\}$		
$Archi_3$-$Struct_{31}$=$\{S_3/1\}$		
$Archi_3$-$Struct_{32}$=$\{\}$		
$Archi_3$-$Struct_{33}$=$\{\}$		

The final fault list is thus $\{S_1/0(11)$, $S_1/1(11)$, $S_2/1(11)$, $S_4/0(11)$, $S_4/1(11)$, $S_2/0(13)$, $S_{11}/1(21)$,

$S_3/1(31)\}$. Hence, by using Inter-Circuits fault collapsing we are able to reduce the number of faults from 63 to 8. This is, in effect, a 0.87 reduction from the Intra-collapsing fault list which is divided between 0.63 for Intra-Architecture fault collapsing and 0.23 for Inter-Architecture fault collapsing.

In the next section, we propose the algorithms for Inter-Circuits fault collapsing.

6 IMPLEMENTATION

Based on the new definitions of fault collapsing, we present in this section the proposed algorithms. Let $F_A = \{f_{A1}, f_{A2}, \ldots, f_{An}\}$ be the set of possible faults in circuit C_A. We suppose that T_A is the list of positions of input values that generate different outputs from the corresponding responses of the fault-free circuit C_A. T_{Ai} is the corresponding fault positions of fault f_{Ai}. Let $F_B = \{f_{B1}, f_{B2}, \ldots, f_{Bn}\}$ be the set of possible faults in circuit C_B and T_B is the list of positions of faults for circuit C_B. Algorithm 1 is developed in order to reduce the set of faults between two circuits C_A and C_B using the Inter-Equivalence relation. We propose algorithm 2 to decrease the fault set using the Inter-Dominance relationship. Finally, algorithm 3 presents fault collapsing through the Redundancy relation.

7 CONCLUSION AND FUTURE WORK

In this paper, we propose three new fault collapsing relations reducing faults from different circuits of the RHS: fault Inter-Equivalence, fault Inter-Dominance and fault Redundancy. This new classification of faults reduces considerably the number of faults which leads to optimize the set of vectors to be considered in the test generation. Consequently, we optimize the overall cost of the hardware test for the reconfigurable systems. In our future work, we plan to continue our research with the Automatic Test Pattern Generation (ATPG) for the RHS.

Algorithm 1: Inter_Equivalence_Fault_Collapsing.

```
Inputs:
$F_A \leftarrow \{f_{A1}, f_{A2}, f_{Ai}, \ldots, f_{An}\}$
i = 1 ... k : the set of faults in $C_A$
$T_A \leftarrow \{T_{A1}, T_{A2}, T_{Ai}, \ldots, T_{An}\}$
i = 1 ... k : the list of positions of faults for $C_A$
$F_B \leftarrow \{f_{B1}, f_{B2}, f_{Bj}, \ldots, f_{Bn}\}$
j = 1 ... h : the set of faults in $C_B$
$T_B \leftarrow \{T_{B1}, T_{B2}, T_{Bj}, \ldots, T_{Bn}\}$
j = 1 ... h : the list of positions of faults for $C_B$
Inter_equi← {equi_list [line 1], equi_list [line 2],...,
equi_list [line l]}; initially empty
Begin
For i = 1 ... sizeof($T_A$)
For j = 1 ... sizeof($T_B$)
if the elements in $T_A$[i] are the same than
the elements in $T_B$[j]
then add line in Inter_equi
with $F_A$[i]: top of the list [line]
$F_B$[j]: end of the list [line]

        Remove $T_B$[j] from $T_B$
Remove $F_B$[j] from $F_B$
End if
End For
End For
End
```

Algorithm 2: Inter_Dominance_Fault_Collapsing.

```
Inputs:
$F_A \leftarrow \{f_{A1}, f_{A2}, f_{Ai}, \ldots, f_{An}\}$
i = 1 ... k : the set of faults in $C_A$
$T_A \leftarrow \{T_{A1}, T_{A2}, T_{Ai}, \ldots, T_{An}\}$
i = 1 ... k : the list of positions of faults for $C_A$
$F_B \leftarrow \{f_{B1}, f_{B2}, f_{Bj}, \ldots, f_{Bn}\}$
j = 1 ... h : the set of faults in $C_B$
$T_B \leftarrow \{T_{B1}, T_{B2}, T_{Bj}, \ldots, T_{Bn}\}$
j = 1 ... h : the list of positions of faults for $C_B$
Inter_dom← {dom_list [line 1], dom_list [line 2],...,
dom_list [line l]}; initially empty
Begin
For i = 1 ... sizeof($T_A$)
For j = 1 ... sizeof($T_B$)
if the elements in $T_A[i] \subset T_B[j]$
then add line in Inter_dom
with $F_A$[i]: top of the list [line]
$F_B$[j]: end of the list [line]

        Remove $T_B$[j] from $T_B$
Remove $F_B$[j] from $F_B$
End if
End For
End For
End
```

Algorithm 3: Redundancy_Fault_Collapsing.

```
Inputs:
$F_A \leftarrow \{f_{A1}, f_{A2}, f_{Ai},\dots, f_{An}\}$
i = 1 ... k : the set of faults in $C_A$
$T_A \leftarrow \{T_{A1}, T_{A2}, T_{Ai},\dots, T_{An}\}$
i = 1 ... k : the list of positions of faults for $C_A$
$F_B \leftarrow \{f_{B1}, f_{B2}, f_{Bj},\dots, f_{Bn}\}$
j = 1 ... h : the set of faults in $C_B$
$T_B \leftarrow \{T_{B1}, T_{B2}, T_{Bj},\dots, T_{Bn}\}$
j = 1 ... h : the list of positions of faults for $C_B$
Redundancy_List:initially empty
Begin
SignalsA $\leftarrow$ Extract signals from $F_A$ where SignalsA[i]
is the corresponding signals used by $F_A$[i]
SignalsB $\leftarrow$ Extract signals from $F_B$ where SignalsB[j]
is the corresponding signals used by $F_B$[j]

  For i = 1 ... sizeof($F_A$)
For j = 1 ... sizeof($F_B$)
if $F_A$[i] is the same than $F_B$[j]
if the corresponding SignalsA[i] and SignalsB[j]
are the same primary signal
then add line in Redundancy_List
with $F_A$[i]: top of the list [line]
$F_B$[j]: end of the list [line]

        Remove $T_B$[j] from $T_B$
Remove $F_B$[j] from $F_B$
Remove SignalsB[j] from SignalB
else
if the elements in $T_A$[i] are the same than
the elements in $T_B$[j]
then add line in Redundancy_List
with $F_A$[i]: top of the list [line]
$F_B$[j]: end of the list [line]
Remove $T_B$[j] from $T_B$
Remove $F_B$[j] from $F_B$
Remove SignalsB[j] from SignalB

    End if
End For
End For
End
```

REFERENCES

Abramovici, M. and Menon, P. R. (1997). Fault simulation on reconfigurable hardware. In *5th IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '97), 16-18 April 1997, Napa Valley, CA, USA*, pages 182–191.

Agrawal, V. D., Prasad, A. V. S. S., and Atre, M. V. (2003). Fault collapsing via functional dominance. In *ITC*, pages 274–280.

Ahmadinia, A. (2007). Optimal free-space management and routing-conscious dynamic placement for reconfigurable devices. *IEEE Trans. Computers*, 56(5):673–680.

Al-Asaad, H. and Lee, R. (2002). Simulation-based approximate global fault collapsing.

Angelov, C., Sierszecki, K., and Marian, N. (2005). Design models for reusable and reconfigurable state machines. In *EUC*, pages 152–163.

Bushnell, M. L. (2001). Essentials of electronic testing for digital, memory and mixed-signal vlsi circuits.

Gharbi, A., Khalgui, M., and Ahmed, S. B. (2010). Optimal model checking of safe control embedded software components. In *Proceedings of 15th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2010, September 13-16, 2010, Bilbao, Spain*, pages 1–8.

Khalgui, M. (2010). Nces-based modelling and ctl-based verification of reconfigurable embedded control systems. *Computers in Industry*, 61(3):198–212.

Prasad, A. V. S. S., Agrawal, V. D., and Atre, M. V. (2002). A new algorithm for global fault collapsing into equivalence and dominance sets. In *ITC*, pages 391–397.

Rooker, M. N., Snder, C., Strasser, T., Zoitl, A., Hummer, O., and Ebenhofer, G. (2007). Zero downtime reconfiguration of distributed automation systems: The epsiloncedac approach. In Mavrk, V., Vyatkin, V., and Colombo, A. W., editors, *HoloMAS*, volume 4659 of *Lecture Notes in Computer Science*, pages 326–337. Springer.

Sethuram, R., Bushnell, M. L., and Agrawal, V. D. (2008). Fault nodes in implication graph for equivalence/dominance collapsing, and identifying untestable and independent faults. In *26th IEEE VLSI Test Symposium (VTS 2008), April 27 - May 1, 2008, San Diego, California, USA*, pages 329–335.

Steininger, A. (2000). Testing and built-in self-test - a survey. *Journal of Systems Architecture*, 46(9):721–747.

Veneris, A. G., Chang, R., Abadir, M. S., and Amiri, M. (2004). Fault equivalence and diagnostic test generation using atpg. In *ISCAS (5)*, pages 221–224.