Modeling Dynamic Single-Cell and Coupling Faults Via Automata Models

Davit Hayrapetyan
Yerevan State University
Yerevan, Armenia
e-mail: davidhayrapetyan@ymail.com

Aleksandr Manukyan
Synopsys Inc.
Yerevan, Armenia
e-mail: aleksandr.manukyan@synopsys.com

ABSTRACT
The main purpose of this paper is to introduce a new method of static and dynamic fault representation with the help of DFA formalization. This method can be used during memory behavior simulation at register transfer level.

Keywords
Fault modeling, single-cell faults, dynamic faults, coupling faults, DFA.

1. INTRODUCTION
Abstraction of SRAM memory defects with faults started to be used since the design of first memory devices. One of the earliest methods base on finite state machine abstraction was introduced in [1]. Fault notations for static single-cell and coupling faults were also introduced and further enhanced in [2] and are known as <S/F/R> and <Sa, Sv/F/R> fault notations correspondingly. Furthermore the decrease of memory cell size led to new types of faults called dynamic [3] to be introduced and the fault notation was expanded in a way that allowed to interprete S, Sa and Sv as sequences of operations. There is a technique of fault injection in SPICE Netlist introduced in [4] that models the real memory cell defects and requires gate level simulation. Another approach of fault modeling using Mealy automata was introduced in [5]. Though this method uses a higher level of abstraction, than SPICE Netlist fault injection, it covers faults that are sensitized with a single operation and the problem of choosing initial memory state remains open. Adequate fault modeling might be crucial for development of efficient test algorithms for memory test and repair [6].
This is the structure of the paper. Section 2 suggests a new method of modeling the faults with deterministic finite automata (DFAs), that is based on <S/F/R> and <Sa, Sv/F/R> dynamic fault notations, supports any sequence of fault sensitizing operations and solves the problem of picking the initial state of memory cell. Section 3 suggests an application for modeling the fault in register transfer level (RTL) simulation. Future work is described in section 4.

2. MODELING FAULTS WITH DFA
The purpose of using DFA is that they allow analyzing test algorithms as words in a language of operations. We defined our model based on the DFA formal definition from [7].
Two DFAs will be introduced for single-cell and coupling faults correspondingly. Though single-cell faults can be also modeled using the second DFA for coupling faults.

2.1. Advantages of the DFA model
The state machine model of faults was introduced in [1].
This model can be represented as
SM = (Q, ∑, δ), where
Q = {0, 1}
∑ = {W0, W1}
δ: Q x ∑ => Q, ∀ q ∈ Q and ∀ a ∈ ∑

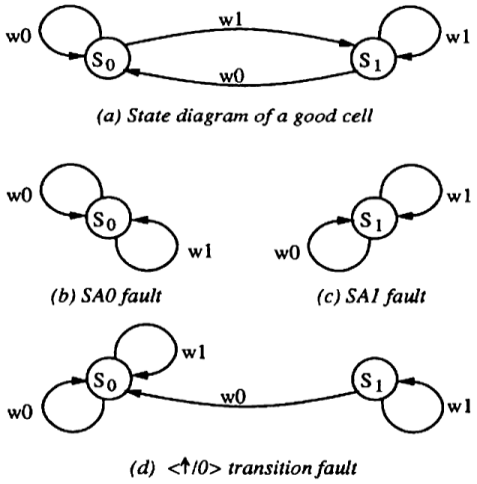

Figure 1. State machines for single-cell faults from [1]

The advantages of DFA model are the presence of initial and final states, which are necessary for modeling more complex faults.
It will be shown later that the DFA model extends this state machine model, thus DFA can model every fault that can be modeled with state machine.

2.2. Initial state
The problem of the choice of initial state was considered in our DFA model. The initial state of memory cell may impact its behavior, e.g. sequence of operations W0R0 (write 0, read value expecting 0) will trigger fault <0W0/1/0> on cell with initial state 0 and won't do that for initial state 1.
Thus each fault should be modeled twice for each initial state (single-cell faults) or fourth (coupling faults). To eliminate the necessity of modeling faults for each initial state, we considered using unknown initial state, i. e. the state could be either 0 or 1. Unknown states are widely used in RTL simulations of the circuits.

2.3. Reset operation
Reset operation is introduced here due to the requirement on dynamic faults [8], that fault can be triggered only if the sequence of operations is applied uninterruptedly, straightly on the memory cell, e.g. if fault is triggered after R0R0R0 sequence is applied on memory cell, but only two reads are being applied to the cell and then the operations are done to another cell, its state should be reset.
Reset operation should be explicitly passed to the DFA of faulty memory cell any time a switch to another cell is made from memory device top level.

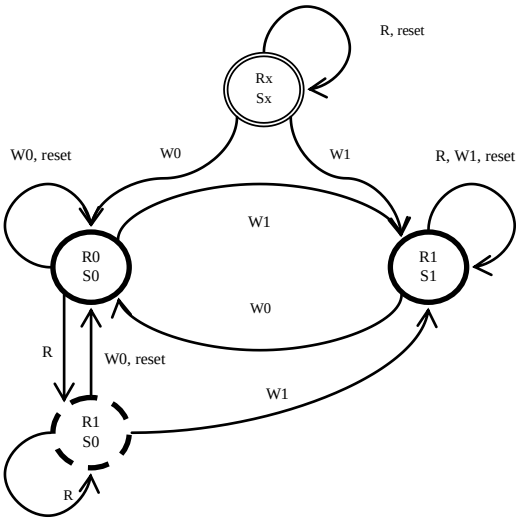

Figure 2. DFA for <0r0/0/1> fault.

2.4. A DFA model for single-cell faults

Below is the definition of DFA for single-cell dynamic faults:

DFA $A_{single\text{-}cell} = (Q, \sum, \delta, q_0, F)$

$Q = \{SXRX, S0R0, S1R1\} \cup \{FIS_i : FIS_i \in \{S0R0, S1R1\}, i = 1,\ldots,n$, where n is finite and $n \geq 0\} \cup \{FAS: FAS \in \{S0R0, S1R1, S0R1, S1R0\}\}$

$\sum = \{W0, W1, R, reset\}$

δ: Q x $\sum$ => Q, $\forall$ q $\in$ Q and $\forall$ a $\in$ $\sum$

$q_0 \in \{SxRx\}$

F = { FAS }

Here Q – the states of DFA, $\sum$ - input alphabet, δ – transition function, q_0 – initial state, F – final state.

Each state of DFA is defined as a couple <S, R> , where S – is the value of the state, R – the result returned by read operation on that state. Each of S and R can have 3 values {X, 0, 1} where X is used once in SXRX initial state and represents an unknown state, i. e. we have no idea what value is stored in the state, nor what will read operation return. FISi - fault intermediate states: their number may vary based on the number of operations in the fault sensitizing sequence. FAS –fault activation state: the state where the faulty behavior is triggered. Example of a single cell fault is shown in figure 2.

0 and 1 states of state machine model correspond to S0R0 and S1R1 in our DFA. Because of $Q_{SM} \subseteq Q_{DFA}$, $\sum_{SM} \subseteq \sum_{DFA}$, $\delta_{SM} \subseteq \delta_{DFA}$, DFA model is an extension for specified state machine.

2.5. A DFA model for coupling faults

The main difference of the second DFA is that each state as a triple $<S_a, S_v, R>$, where S_a – state of aggressor cell, S_v – state of victim cell, R – read operation returned value on victim cell.

The values are defined the same way as in 3.1.

DFA Acoupling = $(Q, \sum, \delta, q_0, F)$

$Q = \{ R\alpha S_a \beta S_v \alpha : \alpha, \beta \in \{0,1,x\}\} \cup \{FIS_i : FIS_i \in \{\{ R\alpha S_a \beta S_v \alpha : \alpha, \beta \in \{0,1\}\}, i = 1,\ldots,n$, where n is finite and $n \geq 0\} \cup \{FAS: FAS \in \{ R\alpha S_a \beta S_v \gamma : \alpha, \beta, \gamma \in \{0,1\}\}\}$

$\sum = \{W_a0, W_a1, W_v0, W_v1, R_a, R_v, reset\}$

δ: Q x $\sum$ => Q, $\forall$ q $\in$ Q and $\forall$ a $\in$ $\sum$

$q0 = \{ RxS_axS_vx \}$

F = { FAS }

The DFA model for coupling fault may be represented as a composition of two states and two blocks (figure 3).

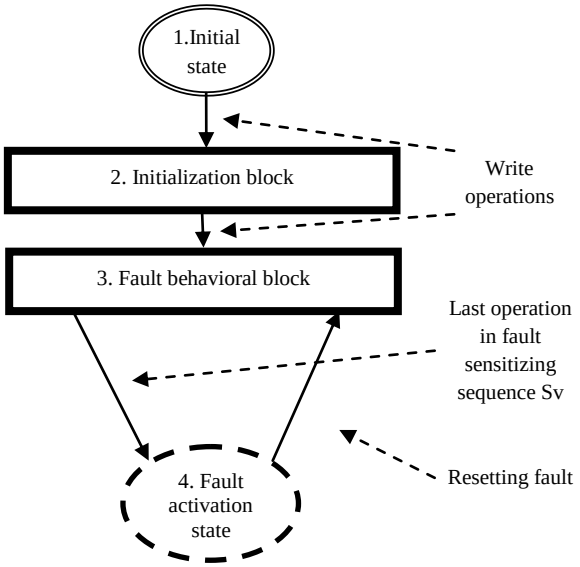

Figure 3. Structure of coupling fault DFA.

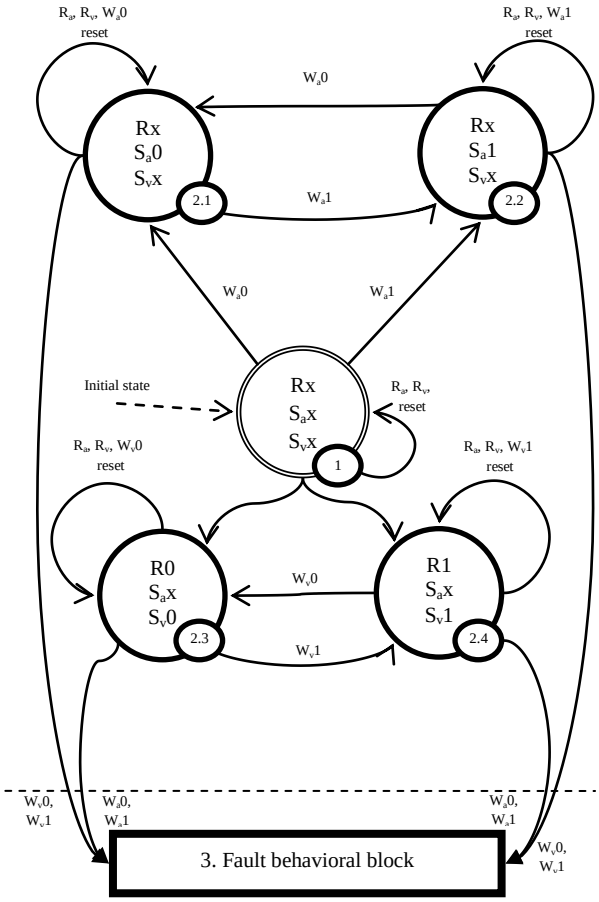

Figure 4. Initial state and initial block of any coupling fault DFA.

In this paper, for convenience, we will consider the DFA of <0w1; 1w0r0/0/1> fault.

2.5.1 Initial state and initialization block

These two blocks have the same structure for all coupling faults.

It should be noted that initial unknown state is a combination of two initial unknown states of aggressor and victim cells, thus it leads to 4 more unknown states {2.1, 2.2, 2.3, 2.4} to be used in DFA (figure 4).

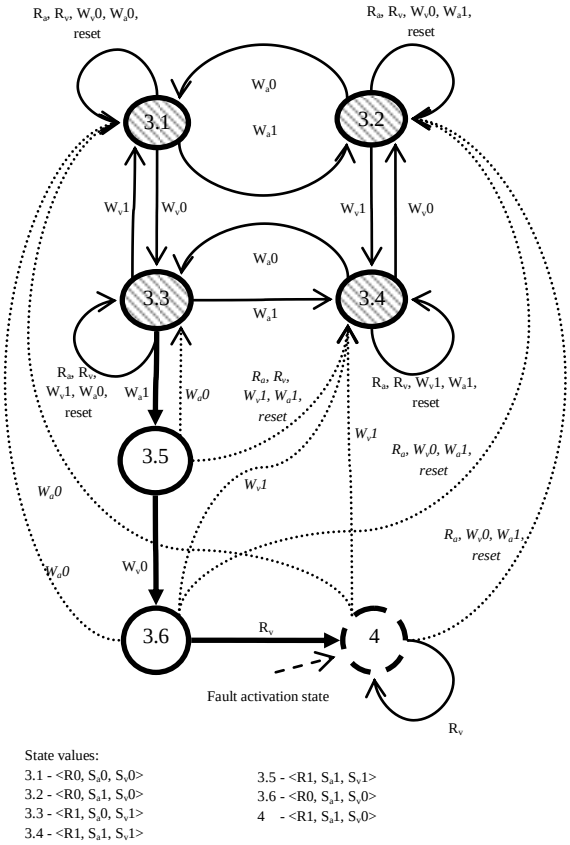

Figure 5. Fault behavioral block and final state of <0w1; 1w0r0/0/1> fault DFA.

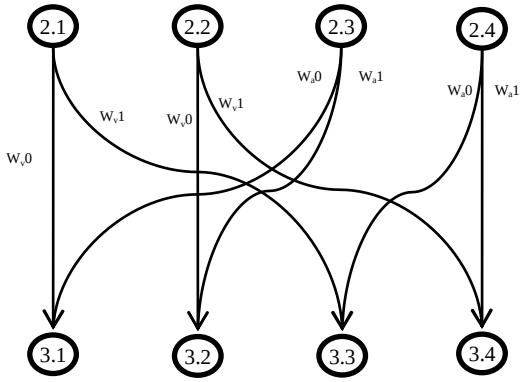

Figure 6. Connections between initialization block and fault behavioral block for <0w1; 1w0r0/0/1> fault.

2.5.2 Fault behavioral block and fault activation state

Shaded states {3.1, 3.2, 3.3, 3.4} in the figure 5 are present in any coupling fault DFA. On the other hand states {3.5, 3.6} are intermediate states ensuring that fault will be activated after a certain sequence of operations described in the fault notation.

The bolded way leading from state 3.3 to FAS 4 corresponds to the sequences of 0w1 and 1w0r0 from the fault notations. If any other operation is applied, the sequence of fault activating operations is being reset, thus the applied operations leads to one of the four {3.1, 3.2, 3.3, 3.4} states. Finally, read operation needs to be applied to FAS in order to observe the fault.

3. APPLICATION

3.1. Converting DFA to Mealy state machine

Introduced DFAs can be easily converted to Mealy state machine (SM) by adding $\Omega = \{-, 0, 1\}$, and λ: Q x $\sum$ => Ω, $\forall$ q $\in$ Q and $\forall$ a $\in$ $\sum$, where Ω – output alphabet, λ – output function, and λ on { w_a0, w_a1, w_v0, w_v1, reset } returns "-", and returns "0" or "1" on

r_v - depending on R value of the state

r_a – depending on S_a value of the state

3.2. Injecting faults via memory top level design Verilog test bench

The proposed fault models can effectively be implemented using System Verilog, without affecting the SoC design. The only thing that is required for implementation of fault model is the knowledge of memory module positioning in the SoC hierarchy and the names of memory pins.

The fault models are positioned outside of SRAM memory model, capturing memory data input, output and address buses, write-enable (WE) pin and operating under the memory clock frequency (figure 7). Fault module uses Mealy SM for fault representation. Each fault module is attached to a specific memory address, i.e. triggering the fault state machine, based on address bus value. Applied operation is being passed to fault module via WE pin. If it is 'read' operation, output bus value is being determined by Mealy SM, if fault address is being passed via address bus, and is determined by memory address bus otherwise.

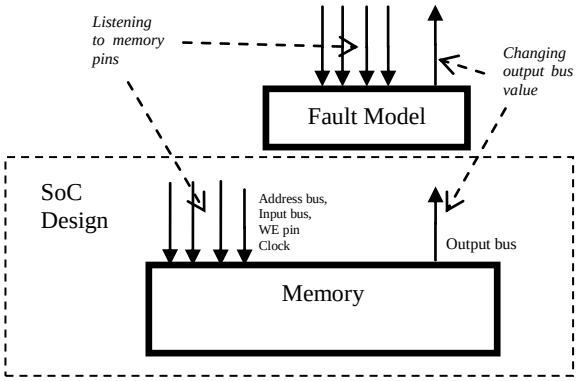

Figure 7. Fault Mealy SM modeled in RTL.

3.3. Implementation

Each state of Mealy SM is modeled as a descriptor as shown in figure 8 (example for single-cell faults).

Mealy SM is modeled as a table (array) of such descriptors, a pointer to the table, two variables storing initial and final states of Mealy SM. Such model is guaranteed to work with O(1) access time if implemented in System Verilog.

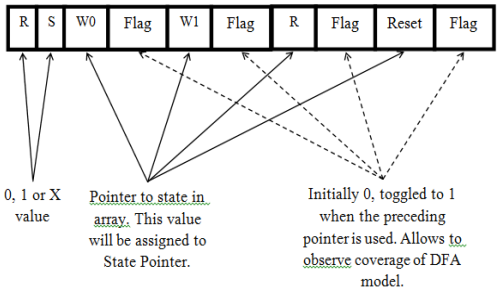

Figure 8. Mealy SM state descriptor for single-cell fault.

The flags of each descriptor are initial assigned with 0 value. A flag of a descriptor is being set to 1 if corresponding operation is being accepted by Mealy SM. That allows monitoring of Mealy SM coverage during the simulation. Here is an approach for implementing single-cell faults with two always blocks in top-level test bench (assuming read operation value can be observed to DATA_OUTPUT bus on clock negedge, if operation is applied on clock posedge):

```
always @(posedge `MEMORY_HIER_PATH.CLK)
begin
    `MEMORY_HIER_PATH.ADDR is tracked. Reset operation
   is called if address is being changed from the address of the fault
   fault_cell_address.
   if(`MEMORY_HIER_PATH.ADDR == fault_cell_address)
     begin
       Determining the operation based on value of
      `MEMORY_HIER_PATH.WE pin. Value of write operation
      is obtained from `MEMORY_HIER_PATH.DATA_INPUT.
      Value of read operation is stored for using in the next always
      block.
     end
end

always @(negedge `MEMORY_HIER_PATH.CLK)
begin
    If read operation was applied, force stored read value on
   `MEMORY_HIER_PATH.DATA_OUTPUT after a small delay
   to ensure that race condition won't occur.
end
```

4. FUTURE WORK

This paper offers test models for static and dynamic single-cell and coupling faults. There are other type of faults to be considered [9], such as linked fault. As it is shown in [10] all known fault can be divided into fault families and represented with periodicity table. Complete solution will be extending the introduced DFA model to cover the periodicity table.

5. CONCLUSION

An approach for modeling dynamic single-cell and coupling faults via automata models was introduced in this paper. This model was compared to other known models, and the advantages were shown. An implementation was proposed, that does not affect the simulation time. Based on structured representation of the faults in the proposed model, it is planned to extend the model to involve other known faults in periodic manner.

REFERENCES

[1] A.J. van de Goor, "Testing Semiconductor Memories: Theory and Practice", pp. 45-54, 1991.

[2] Z. Al-Ars, A.J. Van de Goor, J. Braun, D. Richter "A memory specific notation for fault modeling", Proceedings 10th Asian Test Symposium, pp. 43-48, 2001.

[3] A.J. van de Goor and Z. Al-Ars, "Functional Fault Models: A Formal Notation and Taxonomy", In Proc. of IEEE VLSI Test Symposium, pp. 281-289, 2000.

[4] G. Harutyunyan, G. Tshagharyan, V. Vardanian, Y. Zorian, "Fault Modeling and Test Algorithm Creation Strategy for FinFET-Based Memories", IEEE 32nd VLSI Test Symposium, 2014

[5] A. Benso, A. Bosio, S. Di Carlo, G. Di Natale, P. Prinetto, "Automatic March Tests Generation for Static and Dynamic Faults in SRAMs", Dipartimento di Automatica e Informatica Politecnico di Torino, IEEE Proceedings of the European Test Symposium, 2005

[6] Y. Zorian, S. Shoukourian, "Embedded-memory test and repair: infrastructure IP for SoC yield", IEEE Design & Test of Computers, 2003

[7] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, "Automata Theory, Languages, and Computation" 3rd edition, pp. 45-46, 2006

[8] Luigi Dilillo, Patrick Girard, Serge Pravossoudovitch, Arnaud Virazzel, "Efficient Test of Dynamic Read Destructive Faults in SRAM Memories", Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier

[9] S. Hamdioui , Z. Al-Ars, A.J. van de Goor, M. Rodgers , "Linked faults in random access memories: concept, fault models, test algorithms, and industrial results", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (Volume: 23, Issue: 5), 2004

[10] G. Harutyunyan, S. Shoukourian, V. Vardanian, "Extending fault periodicity table for testing faults in memories under 20nm", Design & Test Symposium (EWDTS), 2014