Modeling Microprocessor Faults on High-Level Decision Diagrams

R. Ubar, J.Raik, A.Jutman, M.Jenihhin
Tallinn Technical University, Estonia

M.Instenberg, H.-D.Wuttke
Ilmenau Technical University, Germany
Outline

• Introduction
• Motivations and contributions
• Discussion: faults and tests
• Fault modeling with Decision Diagrams
• Modeling microprocessor faults
• Experimental results
• Conclusions
Introduction

• Fault models are needed for
  – test generation,
  – test quality evaluation and
  – fault diagnosis
• To handle real physical defects is too difficult
• The fault model should
  – reflect accurately the behaviour of defects, and
  – be computationally efficient
• Usually combination of different fault models is used
• Fault model free approaches (!)
Introduction

• Fault modeling levels
  – Transistor level faults
  – Logic level faults
    • stuck-at fault model
    • bridging fault model
    • open fault model
    • delay fault model
  – Register transfer level faults
  – ISA level faults (MP faults)
  – SW level faults
• Hierarchical fault handling
• Functional fault modeling

Low-Level models

High-Level models
Motivations

Current situation:
- The efficiency of test generation (quality, speed) is highly depending on
  - the description method (level, language), and
  - fault models
- Because of the growing complexity of systems, gate level methods have become obsolete
- High-Level methods for diagnostic modeling are today emerging, however they are not still mature

Main disadvantages:
- The known methods for fault modeling are
  - dedicated to special classes (i.e. for microprocessors, for RTL, VHDL etc. languages...), not general
  - not well defined and formalized
Contributions

• **High-Level Decision Diagrams** are proposed for diagnostic modeling of digital systems
• A novel DD-based node fault model is proposed
• The fault model is simple and formalized
• Traditional high-level fault models for different abstraction levels of digital systems can be replaced by the new uniform fault model
• As the result,
  – the complexity of fault representation is reduced, and
  – the speed of test generation and fault simulation can be increased
Register Level Fault Models

RTL statement:

\[ K: (If \ T, C) \rightarrow R_D \leftarrow F(R_{S1}, R_{S2}, \ldots R_{Sm}), \rightarrow N \]

Components (variables) of the statement:

- **K** - label
- **T** - timing condition
- **C** - logical condition
- **R_D** - destination register
- **R_S** - source register
- **F** - operation (microoperation)
- \( \leftarrow \) - data transfer
- \( \rightarrow N \) - jump to the next statement

RT level faults:

- **K** \( \rightarrow K' \) - label faults
- **T** \( \rightarrow T' \) - timing faults
- **C** \( \rightarrow C' \) - logical condition faults
- **R_D** \( \rightarrow R_D' \) - register decoding faults
- **R_S** \( \rightarrow R_S' \) - data storage faults
- **F** \( \rightarrow F' \) - operation decoding faults
- \( \leftarrow \) - data transfer faults
- \( \rightarrow N \) - control faults
- \( (F) \rightarrow (F)' \) - data manipulation faults
Dedicated functional fault model for multiplexer:
- stuck-at-0 (1) on inputs,
- another input (instead of, additional)
- value, followed by its complement
- value, followed by its complement on a line whose address differs in one bit
Hierarchical Fault Modeling

System: $F$

Module: $F_k$

Component: $F_{ki}$

Test

Fault model

Network of transistors

$W^d_{ki}$

$W^F_{ki}$

Network of components

$W^S_{ki}$

Network of modules

$W^F_k$

$W^S_k$

Higher level

Lower level

Bridge

Interface between levels

$WF_k$ interpretation:

Test – at the lower level

Fault model – at the higher level

System: $F_k$
Logic Level Faults on SSBDDs

Fault modeling on Structurally Synthesized BDDs:

\[ y = x_1 \lor x_2 (x_3 \lor x_4 x_5) \lor x_6 x_7 \]
Data Path in Digital Systems

Control Path

Data Path

<table>
<thead>
<tr>
<th>$y_1$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$M_1 = R_1$</td>
</tr>
<tr>
<td>1</td>
<td>$M_1 = IN$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y_2$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$M_2 = R_1$</td>
</tr>
<tr>
<td>1</td>
<td>$M_2 = IN$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y_3$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$M_3 = M_1 + R_2$</td>
</tr>
<tr>
<td>1</td>
<td>$M_3 = IN$</td>
</tr>
<tr>
<td>2</td>
<td>$M_3 = R_1$</td>
</tr>
<tr>
<td>3</td>
<td>$M_3 = M_2 \cdot R_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y_4$</th>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reset</td>
<td>$R_2 = 0$</td>
</tr>
<tr>
<td>1</td>
<td>Hold</td>
<td>$R_2 = R_2$</td>
</tr>
<tr>
<td>2</td>
<td>Load</td>
<td>$R_2 = M_3$</td>
</tr>
</tbody>
</table>
Decision Diagram of the Data Path

<table>
<thead>
<tr>
<th>$y_1$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$M_1 = R_1$</td>
</tr>
<tr>
<td>1</td>
<td>$M_1 = IN$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y_2$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$M_2 = R_1$</td>
</tr>
<tr>
<td>1</td>
<td>$M_2 = IN$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y_3$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$M_3 = M_1 + R_2$</td>
</tr>
<tr>
<td>1</td>
<td>$M_3 = IN$</td>
</tr>
<tr>
<td>2</td>
<td>$M_3 = R_1$</td>
</tr>
<tr>
<td>3</td>
<td>$M_3 = M_2 * R_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$y_4$</th>
<th>Operation</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reset</td>
<td>$R_2 = 0$</td>
</tr>
<tr>
<td>1</td>
<td>Hold</td>
<td>$R_2 = R'_2$</td>
</tr>
<tr>
<td>2</td>
<td>Load</td>
<td>$R_2 = M_3$</td>
</tr>
</tbody>
</table>
Faults and High-Level Decision Diagrams

RTL-statement:
\[ K: (\text{If } T, C) \quad R_D \leftarrow F(R_{S1}, R_{S2}, \ldots, R_{Sm}), \rightarrow N \]

Terminal nodes

Nonterminal nodes

RTL-statement faults:
- data storage, data transfer, data manipulation faults
- label, timing condition, logical condition, register decoding, operation decoding, control faults

K: (If T, C) \quad R_D \leftarrow F(R_{S1}, R_{S2}, \ldots, R_{Sm}), \rightarrow N
Faults and High-Level Decision Diagrams

**RTL-statement:**

\[ K: (\text{If } T, C) \quad R_D \leftarrow F(R_{S1}, R_{S2}, \ldots R_{Sm}), \quad \rightarrow N \]

**Nonterminal nodes**

RTL-statement faults:
- label,
- timing condition,
- logical condition,
- register decoding,
- operation decoding,
- control faults

**Terminal nodes**

RTL-statement faults:
- data storage,
- data transfer,
- data manipulation faults
Fault Modeling on DDS

Binary DD
with 2 terminal nodes and 2 outputs from each node

General case of DD
with $n \geq 2$ terminal nodes and $n \geq 2$ outputs from each node
Fault Model for Decision Diagrams

• Each path in a DD describes the behavior of the system in a specific mode of operation
• The faults having effect on the behaviour can be associated with nodes along the path
• A fault causes incorrect leaving the path activated by a test
Fault Model for Decision Diagrams

D1: the output edge for \( x(m) = i \) of a node \( m \) is always activated

D2: the output edge for \( x(m) = i \) of a node \( m \) is broken

D3: instead of the given edge, another edge or a set of edges is activated
Microprocessor Modeling with S-Graphs

Instruction set of a Microprocessor:

I₁: MVI A,D  A ← IN
I₂: MOV R,A  R ← A
I₃: MOV M,R  OUT ← R
I₄: MOV M,A  OUT ← A
I₅: MOV R,M  R ← IN
I₆: MOV A,M  A ← IN
I₇: ADD R    A ← A + R
I₈: ORA R    A ← A ∨ R
I₉: ANA R    A ← A ∧ R
I₁₀: CMA A,D A ← ¬ A
Test Generation for Microprocessors

High-Level DDs for a microprocessor (example):

Instruction set:

1. MVI A, D  \( A \leftarrow \text{IN} \)
2. MOV R, A  \( R \leftarrow A \)
3. MOV M, R  \( \text{OUT} \leftarrow R \)
4. MOV M, A  \( \text{OUT} \leftarrow A \)
5. MOV R, M  \( R \leftarrow \text{IN} \)
6. MOV A, M  \( A \leftarrow \text{IN} \)
7. ADD R  \( A \leftarrow A + R \)
8. ORA R  \( A \leftarrow A \lor R \)
9. ANA R  \( A \leftarrow A \land R \)
10. CMA A, D  \( A \leftarrow \neg A \)

DD-model of the microprocessor:
Decision Diagrams for Microprocessors

High-Level DD-based structure of the microprocessor (example):

DD-model of the microprocessor:
Microprocessor Fault Model

Faults affecting the operation of microprocessor can be divided into the following classes:

- addressing faults affecting register decoding;
- addressing faults affecting the instruction decoding and -sequencing functions;
- faults in the data-storage function;
- faults in the data-transfer function;
- faults in the data-manipulation function.
Microprocessor Fault Model

For multiplexers under a fault, for a given source address any of the following may happen:

F1: no source is selected
F2: wrong source is selected;
F3: more than one source is selected and the multiplexer output is either a wired-AND or a wired-OR function of the sources, depending on the technology.
Microprocessor Fault Model

For demultiplexers under a fault, for a given destination address:

F4: no destination is selected

F5: instead of, or in addition to the selected correct destination, one or more other destinations are selected
Microprocessor Fault Model

Addressing faults affecting the execution of an instruction may cause the following fault effects:

F6: one or more microorders not activated by the microinstructions of I

F7: microorders are erroneously activated by the microinstructions of I

F8: a different set of microinstructions is activated instead of, or in addition to, the microinstructions of I
Microprocessor Fault Model

The data storage faults:

- F9: one or more cells stuck at 0 or 1;
- F10: one or more cells fail to make a 0→1 or 1→0 transitions;
- F11: two or more pairs of cells are coupled;

For buses under a fault:

- F12: one or more lines stuck at 0 or 1;
- F13: one or more lines form a wired-OR or wired-AND function due to shorts or spurious coupling.
Test Generation on DDS

**Binary DD**
with 2 terminal nodes and
2 outputs
from each node

**General case of DD**
with $n \geq 2$ terminal nodes and
$n \geq 2$ outputs
from each node
Hierarchical Test Generation on DDs

Hierarchical test generation with DDs: Scanning test

Single path activation in a single DD
Data function $R_1 \times R_2$ is tested

Data path

Test program: Control: $y_1 y_2 y_3 y_4 = x032$
Data: For all specified pairs of $(R_1, R_2)$

Low level test data (constraints $W$)
Test Generation on High Level DDs

High-level test generation with DDs:

Multiple paths activation in a single DD
Control function $y_3$ is tested

Data path

Test program: Control: For $D = 0,1,2,3$: $y_1 y_2 y_3 y_4 = 00D^2$
Activating high-level faults: Data: Solution of $R_1 + R_2 = IN \neq R_1 \neq R_1 * R_2$
Test Generation for Microprocessors

DD-model of the microprocessor:

Scanning test program for adder:
Instruction sequence $T = I_5(R)I_1(A)I_7I_4$
for all needed pairs of $(A,R)$
Test Generation for Microprocessors

DD-model of the microprocessor:

Conformity test program for decoding I:
Instruction sequence $T = I_5 I_1 D I_4$
for all $D \in \{I_1 - I_{10}\}$ at given $A, R, IN(3)$
Experimental results

<table>
<thead>
<tr>
<th>BW</th>
<th>ATPG</th>
<th>Time</th>
<th>Test</th>
<th>Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>HTPG</td>
<td>0.08</td>
<td>224</td>
<td>900</td>
</tr>
<tr>
<td></td>
<td>Synopsys</td>
<td>0.29</td>
<td>46</td>
<td>855</td>
</tr>
<tr>
<td>8</td>
<td>HTPG</td>
<td>0.10</td>
<td>224</td>
<td>1612</td>
</tr>
<tr>
<td></td>
<td>Synopsys</td>
<td>0.75</td>
<td>64</td>
<td>1531</td>
</tr>
<tr>
<td>16</td>
<td>HTPG</td>
<td>0.13</td>
<td>224</td>
<td>3016</td>
</tr>
<tr>
<td></td>
<td>Synopsys</td>
<td>1.86</td>
<td>73</td>
<td>2861</td>
</tr>
<tr>
<td>32</td>
<td>HTPG</td>
<td>0.15</td>
<td>224</td>
<td>5908</td>
</tr>
<tr>
<td></td>
<td>Synopsys</td>
<td>5.57</td>
<td>84</td>
<td>5607</td>
</tr>
</tbody>
</table>

HTPG – high level  
Synopsys – gate level

Gate-level fault coverage – 100%
Conclusions

- Different fault models for different representation levels of digital systems can be replaced on DDs by the uniform node fault model.
- It allows to represent groups of structural faults through groups of functional faults.
- As the result, the complexity of fault representation can be reduced, and the simulation speed can be raised.
- The fault model on DDs can be regarded as a generalization:
  - of the classical gate-level stuck-at fault model, and
  - of the known higher level fault models.

www.pld.ttu.ee/~raiub/