Digital Testing: Design Representation and Fault Detection

Outline
- Design Representation Model
- Switching functions
- Boolean Difference
- Finite State Machines
- Tabular Representation
- Graphical Representation
- Graphs
- Binary Decision Diagrams
- Netlists
- Hardware Description Languages

Design Paradigm
- The design representation space consists of domains and levels
- Behavioral domain most abstract
- Structural domain specifies the architecture
- Physical domain includes the transistors and layout

Domains and Levels

<table>
<thead>
<tr>
<th>Domain</th>
<th>Behavioral</th>
<th>Structural</th>
<th>Physical</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>System Specifications</td>
<td>Blocks</td>
<td>Chip</td>
</tr>
<tr>
<td>RTL</td>
<td>RTL Specifications</td>
<td>Registers</td>
<td>Macro Cells</td>
</tr>
<tr>
<td>Logic</td>
<td>Boolean Functions</td>
<td>Logic Gates</td>
<td>Standard Cells</td>
</tr>
<tr>
<td>Circuit</td>
<td>Differential Equations</td>
<td>Transistors</td>
<td>Masks</td>
</tr>
</tbody>
</table>

Domains

- Behavioral Domain
  \[ a = b \cdot c \]
  \[ z = \neg (a \cdot d) \]
- Structural Domain
- Physical Domain

Levels

System Level

Gate Level

Register Level

Circuit Level
Fault detection and redundancy

Definition:
A test vector t detects a fault t if \( Z_f(t) \neq Z(t) \)
\( Z(x) \) - logic function of circuit N
\( x \) - input vector
\( t \) - a specific input (test) vector
\( N_f \) - a faulty circuit (N changes as a result of fault f)

Example:
\( Z_1 = x_1 \cdot x_2 \)
\( Z_{1f} = x_1 + x_2 \)
\( Z_2 = x_2 \cdot x_3 \)
\( Z_{2f} = (x_1 + x_2) \cdot x_3 \)
\( Z(011) = 01, Z_{1f}(011) = 11 \Rightarrow t = 011 \) detects f

The set of all tests that detect f is given by
\( Z(x) \oplus Z_f(x) = 1 \)

Fault detection and redundancy

Example:
Consider test \( t = 011 \)

SPECIFIC-FAULT ORIENTED TEST SET GENERATION

TWO FUNDAMENTAL TEST GENERATION STEPS
1. Activate, Excite, Provok e or Setup the Fault
2. Make Fault Observable, Fault Sensitization
   * Find Primary Input Values that Cause Error Signal in Faulty Circuit
   * For Single-Stuck-at-f Fault — Place \( \neg \) at Fault Site
3. Propagate the Resulting Error to a Primary Output
   * Path Sensitization
   * Find Primary Input Values that Sensitize Error Signal to Primary Output

Fault detection and redundancy

a set of inputs which detect all possible (detectable) faults is called a complete detection test set
an input \( b = (b_1 \ldots b_n) \) distinguishes a fault \( \alpha \) from another fault \( \beta \) if \( Z_\alpha \neq Z_\beta \) or \( Z_\alpha \oplus Z_\beta = 1 \)
a set of tests which distinguish all pairs of fault is called a complete location test set
**Fault detection and redundancy**

Example:
Consider fault $\alpha = x_2 s a_1$ and $\beta = x_3 s a_0$
• find $Z_1$, $Z_\alpha$ & $Z_\beta$
• check if (101) detects $Z_\alpha$
• check if (101) distinguishes $Z_\alpha$ & $Z_\beta$

\[
\begin{align*}
Z &= [(x_1 x_2)'(x_2 x_3)']' \\
&= x_1 x_2 + x_2 x_3 \\
&= x_2 (x_1 + x_3)
\end{align*}
\]

$\alpha = x_2 s a_1$ and $\beta = x_3 s a_0$
$Z \oplus Z_\alpha | (1,0,1) = Z \oplus (x_1 + x_3) = 1 \oplus 1 = 1$
$Z_\alpha \oplus Z_\beta | (1,0,1) = Z \oplus (x_1 x_2) = 1 \oplus 0 = 1$
We see that the same vector $x = (1,0,1)$ distinguishes these two faults

---

**Fault detection and redundancy**

Example:
If only $f$ out of $x$ faults have been detected by a test then “test coverage” is $t_c = f/x \leq 1$

One-dimensional path sensitization
A line whose value changes in the presence of the fault is sensitized to the fault by the test $t$. A path composed of sensitized lines is called a sensitized path.

---

**Fault detection and redundancy**

Path sensitization algorithm
I. specify inputs to generate at the site of the fault.
II. propagate error to the output
III. specify inputs to obtain signal values needed in II

---

**Element evaluation**

Truth tables
• requires $2^n$ entries
Input Scanning is simpler
• gates described by
  - $c --$ controlling value
  - $i --$ inversion

<table>
<thead>
<tr>
<th>$c$</th>
<th>$x$</th>
<th>$c \oplus i$</th>
<th>$c$</th>
<th>$x$</th>
<th>$c \oplus i$</th>
<th>$c'$</th>
<th>$c'$</th>
<th>$c' \oplus i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>OR</td>
<td>NAND</td>
<td>NOR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Fault detection and redundancy**

**Lemma:**
Gate with c, i - controlling and inversion values
• all inputs of G sensitized to f have the same value (say a)
• all not sensitized inputs have value c'
• the output of Gate is equal a ⊕ i

<table>
<thead>
<tr>
<th>c x x</th>
<th>c ⊕ i</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>0 0</td>
</tr>
<tr>
<td>OR</td>
<td>1 0</td>
</tr>
<tr>
<td>NAND</td>
<td>0 1</td>
</tr>
<tr>
<td>NOR</td>
<td>1 1</td>
</tr>
</tbody>
</table>

The rules for error propagation with sensitized inputs equal to a

<table>
<thead>
<tr>
<th>c i</th>
<th>Gate</th>
<th>Other inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>AND</td>
<td>all must be 1</td>
<td>a</td>
</tr>
<tr>
<td>0 1</td>
<td>NAND</td>
<td>all must be 1</td>
<td>a'</td>
</tr>
<tr>
<td>1 0</td>
<td>OR</td>
<td>all must be 0</td>
<td>a</td>
</tr>
<tr>
<td>1 1</td>
<td>NOR</td>
<td>all must be 0</td>
<td>a'</td>
</tr>
</tbody>
</table>

**Detectability**
• if no test can detect fault f => f is **undetectable**
• such a circuit is **redundant**
• undetectable fault can prevent detection of another fault

Example:
- b sa0 is detected by t = 1101
Detectability

Example: \( b \text{ sa0 is no longer detected by } t=1101 \) if \( a \text{ sa1 is present} \)

\[ \begin{align*}
A = 1 & \quad a = 0/1 \\
C = 0 & \quad b = 1/0
\end{align*} \]

Detectability

Redundant circuit can always be simplified by removing a gate or gate input

**Rules**

<table>
<thead>
<tr>
<th>Undetectable fault</th>
<th>Simplification rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND (NAND) input sa1</td>
<td>remove input</td>
</tr>
<tr>
<td>AND (NAND) input sa0</td>
<td>remove gate, replace by 0(1)</td>
</tr>
<tr>
<td>OR (NOR) input sa0</td>
<td>remove gate, replace by 1(0)</td>
</tr>
<tr>
<td>OR (NOR) input sa1</td>
<td>remove gate, replace by 1(0)</td>
</tr>
</tbody>
</table>

Redundancy may be used to avoid hazards

Example: Consider: \( b=c=1 \), \( a \) changes from 1 to 0

\[ \begin{align*}
a = 1 & \quad b = 1/0 \\
c = 1 & \quad b = 1/0
\end{align*} \]

boolean Algebra

- \( f(x) \), has a range, \( B \), and a domain, \( B^n \), where \( B = \{0, 1\} \) and \( : B^n \rightarrow B \).
- For any element \( c \in B \), the constant function is \( f(x) = c \), where \( x \in B^n \).
- For any \( x \in B^n \), the projection function is \( f(x_i) = x_i \).
- The set of variables \( \{x_1, x_2, \ldots, x_n\} \) is called the support of the function.
- If \( g \) and \( h \) are \( n \)-variable functions, then the functions \( g + h, g \cdot h, \) and \( g' \) are defined.

Properties of Boolean functions

- \( g \cdot h(x_1, \ldots, x_n) = g(x_1, x_2, \ldots, x_n) \cdot h(x_1, x_2, \ldots, x_n) \)
- \( g(x_1, \ldots, x_n) = g(x_2, \ldots, x_n) \cdot h(x_1, x_2, \ldots, x_n) \)
- \( g(x_1, \ldots, x_n) = g(x_1, x_2, \ldots, x_n) \cdot h(x_1, x_2, \ldots, x_n) \)

There is only a finite set of distinct functions of \( n \)-variable.

A Boolean function can be expressed in different forms, for instance

\[ f(x_1, x_2, x_3) = x_1 \cdot x_2 + x_2 = x_1 + x_2 \]
Boolean difference

- Definition: The Boolean difference of \( f(x) \) is equal \( \Delta(f) = \frac{df(x)}{dx} = f(x) \oplus f(x') \)
- An equivalent definition results from the following Shannon's law \( f(x) = x \cdot f(1) + x' \cdot f(0) \)
- Lemma: \( f(x) \oplus f(x') = f(0) \oplus f(1) \)
- Then the Boolean difference is

\[
\begin{array}{c}
\text{Definition:} \\
\Delta(f) = \frac{df(x)}{dx} = f(x) \oplus f(x') \\
\text{An equivalent definition results from the following Shannon's law:} \\
f(x) = x \cdot f(1) + x' \cdot f(0) \\
\text{Lemma:} f(x) \oplus f(x') = f(0) \oplus f(1) \\
\text{Then the Boolean difference is:}
\end{array}
\]

Fault Detection

As an example, let us consider the function \( f(x) = g(x) + x_3 \)
- Thus \( \frac{df(x)}{dx} = x_3 \oplus (x_1 \cdot x_2) = 1 \)
- If \( x_1 = 1 \) and \( x_2 = 0 \)
- For the SA1 and SA0 faults on \( x_3 \), the patterns are then \( x_3 = 0 \) and \( x_3' = 1 \)
- For the first fault, we then must have \( x_3 \oplus x_1 x_2 = 1 \). This results in three patterns: \( x_1 x_2 x_3 = (001, 011, \text{or } 101) \)
- and for the other fault, we have \( x_3 = 0 \) and \( x_3' = 1 \)

Fault Detection: Example

- We will repeat this calculation for stuck-at faults on \( x_3 \).
- First, we calculate the Boolean difference: \( \frac{df(x)}{dx} = g(x) \oplus 1 = x_3 \cdot x_1 x_2 \).
- Then we equate its products with \( x_3 \) and \( x_3' \) to 1. The patterns to detect the faults \( x_3/0 \) and \( x_3/1 \) are then \( x_3 \cdot x_1 x_2 = 1 \) and \( x_3' \cdot x_1 x_2 = 1 \).
- For the first fault, we then must have \( x_3 \cdot x_1 x_2 = 1 \). This results in three patterns: \( x_1 x_2 x_3 = (001, 011, \text{or } 101) \)
- and for the other fault, we have \( x_3 = 0 \) and \( x_3' = 1 \)
Boolean difference

Example:

\[ T_1 = x_1'(df/dx_1) = x_1'(f(0) \oplus f(1)) \]
\[ = x_1'(x_1 \oplus (x_2 + x_3)) \]
\[ = x_1'(x_1'x_2 + x_1'x_3 + x_1x_2' + x_1x_3') \]

\( x=0001 \) is a solution which sensitizes the path \( x_G G_G G \)

Boolean difference

Example: find the Boolean difference of \( f \) w.r.t. \( x_2 \)

\[ df \]
\[ dx_2 = f(0) \oplus f(1) = x_1 \oplus x_1 = 0 \]

\( T_0 = 0 \) \( \frac{df}{dx_2} = 0 \) \( \rightarrow \) Not testable

\( T_1 = x_2 \frac{df}{dx_2} = 0 \)

Boolean difference

Theorem:
The set of all tests which detect \( h \ sa0 \) is defined by:

\[ T_0 = h(x) \frac{df(x, h)}{dh} \]

And \( h \ sa1 \) is defined by:

\[ T_1 = \overline{h(x)} \frac{df(x, h)}{dh} \]
Boolean difference

Example:
Find all $h_{sa0}$ tests

$$f = h + x_1x_4 + x_2x_3$$
where $h = x_1x_2$

$$T_h = h \frac{df}{dh}$$

$$= h[(f(h=0) \oplus f(h=1)]$$

$$= h(x_1x_4 + x_2x_3) \oplus 1$$

$$= h_x_1x_4 + h_x_2x_3$$

$$= n_1x_1\overline{r_1} + n_2x_2$$

$$= x_1 \overline{x_2}$$


Boolean difference

Boolean difference can be formed by concatenating Boolean differences (Simple chain rule)

$$\frac{dZ}{dx} = \frac{dZ}{dy} \frac{dy}{dx}$$

In the previous example we have:

$$f = G_x + G_y, \; G_x = G_x \overline{x_4}, \; G_y = G_y \overline{x_4}$$

$$\frac{df}{dG_x} \frac{dG_x}{dx} = \frac{df}{dG_y} \frac{dG_y}{dx}$$

$$= \overline{G} \cdot x_1 = \overline{x_2} \cdot x_2 = (x_1 + x_2)x_1 = x_1$$


Boolean difference

Test for multiple faults

for $x_1 = 0 \& x_2 = 0$

$$T_{0,0} = x_1x_2(f(x_1,x_2) \oplus f(\overline{x_1},\overline{x_2}))$$

$$= x_1x_2(f(00) \oplus f(11))$$

these results can be extended to more than 2 faults

$x_1, x_2, x_3 = 0$

$$T_{0,0,0} = x_1x_2 \cdots x_n(f(x_1,x_2,\cdots x_n) \oplus f(\overline{x_1},\overline{x_2},\cdots \overline{x_n}))$$

$$= x_1x_2 \cdots x_n(f(1\cdots 1) \oplus f(00\cdots 0))$$


Boolean difference

Test for multiple faults

for $x_1 = 1 \& x_2 = 1$

$$T_{1,1} = x_1x_2(f(x_1,x_2) \oplus f(\overline{x_1},\overline{x_2}))$$

$$= x_1x_2(f(10) \oplus f(01))$$

for $x_1 = 0 \cup x_2 = 0$

$$T_{0,0} = f \oplus f(00)$$

$$= x_1x_2 \frac{df}{dx_1x_2} + x_1 \frac{df}{dx_1} + x_2 \frac{df}{dx_2}$$
Finite State Machine

- A finite state machine is formally expressed as a 6-tuple \((\mathcal{I}, \mathcal{S}, \delta, S_0, \mathcal{O}, \lambda)\), where
  - \(\mathcal{I}\) is the input alphabet, that is, a finite non-empty set of inputs
  - \(\mathcal{S}\) is the finite and non-empty set of states
  - \(\delta: \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{S}\) is the next state function
  - \(S_0 \subseteq \mathcal{S}\) is the set of initial states
  - \(\mathcal{O}\) is the output alphabet
  - \(\lambda: \mathcal{S} \times \mathcal{I} \rightarrow \mathcal{O}\) is the output function for a Mealy machine [Mealy 1955] and,
    \(\lambda: \mathcal{S} \rightarrow \mathcal{O}\) is the output function for a Moore machine [Moore 1956].

Tabular Representation

<table>
<thead>
<tr>
<th>State</th>
<th>I = 0</th>
<th>I = 1</th>
<th>Next State</th>
<th>Present State</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>C</td>
<td>0</td>
<td>D</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>D</td>
<td>1</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>0</td>
<td>0</td>
<td>D</td>
</tr>
</tbody>
</table>

Table 3.3 State Table for FSM

<table>
<thead>
<tr>
<th>Present</th>
<th>Next</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>C</td>
</tr>
<tr>
<td>0</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>B</td>
<td>0</td>
</tr>
</tbody>
</table>

Tabular Representation

<table>
<thead>
<tr>
<th>Minterm</th>
<th>F1</th>
<th>F2</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>100</td>
<td>d</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.2 Combinational Functions Defined in Tabular Form.

Graphical Representation (FSM)
Graphs

(a) (b) (c) (d)

\[ x_1 x_2 + x_2 x_3 \]

Binary Decision Diagram

(a) (b) (c)

\[ f = x_1 + x_2 \]

Test Generation with BDD

\[ f = x_1 x_2 x_3 + x_1 x_2 x_1 \]

to test for \( x_1 \) sa faults get
\[ \frac{df}{dx_1} = x_2 x_2 x_3 + x_1 x_2 x_1 \]

Netlists

CMOS inverter

- \text{MN} 3 2 0 0 \text{CMOSN} \ W=1.8U, L=0.5U
- \text{MP} 3 2 1 1 \text{CMOSP} \ W=5.4U, L=0.5U
- CL 3 0 2pF

\text{Voltages used}

- \text{VDD} \ 1 0 DC 5V
- \text{Vin} 2 0 0 3.3 0.2

\text{Models used}

- \text{MODEL CMOSNI} \ \text{NMOS} \ \text{LEVEL}=3 \ \text{PHI}=0.700000 \ \text{TOX}=9.6000E-09 \ \text{J}=0.200000U
- \text{MODEL CMOSP} \ \text{PMOS} \ \text{LEVEL}=3 \ \text{PHI}=0.700000 \ \text{TOX}=9.6000E-09

\text{Weff} = \text{Wdrawn} - \text{Delta}_W

The suggested \( \text{Delta}_W \) is \( 4.1080E-07 \)
**Hardware Description Languages (HDL)**

- Developed originally at universities,
  - AHDL (Hill 91)
  - ISP (Barbacci 77)

- Characteristics: Representation
  - of hierarchical design description
  - at different levels
    - Behavioral, RTL and Gate

- Major languages: VHDL, Verilog-HDL, Hardware C