EECS 579: Logic and Fault Simulation

- **Simulation**: Use of computer software models to verify correctness
- **Fault Simulation**: Use of simulation for fault analysis and ATPG

![Simulation Diagram]

- **Emulation**: Hardware-assisted simulation using FPGA-based emulators

**Simulation**

**Uses**
- Reduce/eliminate need to build physical prototypes
- Alternative design comparison
- Functional design verification
- Timing/performance design verification
- Fault analysis and test generation

**Simulation Levels**

- Analog (continuous)
- Digital (discrete)
- Device layout
Logic Simulation

- Component types
  Gates, flip-flops, switches, macro elements

- Signal accuracy
  0, 1, X (unknown) plus a few others

- Timing accuracy (delay models)
  Transport: unit, nominal
  Rise/fall delay

- Simulator structures
  Compiled-code (compiler-driven)
  Non-compiled event-driven

Example 1

Event 1
Compiled-Code Simulation

- Circuit model is executable code.
- Circuit elements are program routines linked to reflect circuit structure.
- Simulation steps
  - Levelize and code circuit
  - Compile and execute
- Characteristics
  - All signals are evaluated in all time periods
  - Fast simulation of simpler circuits
  - Inefficient with complex signal/timing
  - Frequent recompilation often necessary

(Code to compute x)

(Code to compute y)

LDA x  Load x from memory into CPU accumulator A
AND y  Compute A.y; put in A
NOT    Compute complement of A
STA z  Store A in z

(To next element)
Compiled-Code Simulator

procedure compiled_sim
    Read circuit description;
    Break feedback and levelize circuit;
    Generate code for gates;
    Initialize signal values;
    for each input signal vector do
        for each logic level do
            for each gate do execute gate code (simulate);
        end
    end
    Output desired results;
end
end procedure

Event-Driven Simulation

- Circuit model is a data structure consisting of set of linked tables specifying components, their attributes, and connections
- Separate simulation control program processes the circuit
- Characteristics
  - Signal timing calculations are event-directed
  - Handles complex signal/timing models
  - Circuit models are easy to change
  - Supports forward/backward tracing
  - Slower for large, simple models
### Event-Driven Simulation

#### Name   | Type  | Inputs   | Outputs | Delay  | Other attributes
--- | --- | --- | --- | --- | ---
G1  | NAND | A,B     | FF1-J, Z6 | 10,15 | ...  
G2  | NOT  | A       | FF1-K    | 10,10 | ...  
FF1 | JKFF | G1, CLK, G2 | Z7, Z8 | 10    | ...  
A   | PI   |          | G1, G2   |        |      
Z6  | PO   | G1      |          |        |      

etc.

#### Event-Driven Simulation

#### Name   | Type  | Fan-in | Fan-out | Fan-in | Fan-out | Delay  | Other attributes
--- | --- | --- | --- | --- | --- | --- | ---
G1  | NAND | P1     | P2      | 2      | 2       | 10,15  | ...  
G2  | NOT  | P3     | P4      | 1      | 1       | 10,10  | ...  
FF1 | JKFF | P5     | P6      | 2      | 2       | 10     | ...  
A   | PI   | P7     | 1       |        |        |        |      
Z6  | PO   | P8     |         |        |        |        |      

etc.

#### Fan-in/ fan-out table

- P1
  - A
  - B
  - FF1-J
  - Z6
- P2
  - A
  - FF1-K
- P3
  - G1
- P4
  - CLK
- P5
  - G2

etc.
Event-Driven Simulation

Example 1

![Circuit Diagram]

Event 1

Event-Driven Simulator

procedure \textit{table\_sim}

Read circuit description;
Construct circuit data structures (tables);
Initialize signal values and event time $t$

\[\text{repeat} \{\text{with set of current events}\} \]
Select event and examine its fanout list;
\[\text{while} \ (\text{unevaluated fanout signals remain}) \ \text{do}\]
Evaluate and schedule fanout events;
Output desired results; \textbf{end}
\[\text{until} \ (\text{no more current events})\]
Increment $t$ to next event time;

\textbf{end procedure}
Time Simulation

- Time driven: simulate entire circuit at every time step
- Event driven: emulate active parts of circuit only when a signal change or event occurs.

Event-Driven Simulation
1. Simulate the current event
2. Scan the circuit for implied events and schedule them
3. Advance simulation time to the next event

- Event scheduling methods
    - Fast, large storage needs (static allocation)
  - Linear list (linked list)
    - Slow, efficient storage (dynamic allocation)
  - Timing wheel: indexed-list that trades off speed and storage

Event Scheduling

Linked List

<table>
<thead>
<tr>
<th>Time</th>
<th>Line</th>
<th>Value</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X0</td>
<td>0</td>
<td>P1</td>
</tr>
<tr>
<td>10</td>
<td>Y4</td>
<td>X</td>
<td>P17</td>
</tr>
<tr>
<td>15</td>
<td>Y5</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

Other events at $t = 0$
**Event Scheduling**

**Timing Wheel**

<table>
<thead>
<tr>
<th>Time</th>
<th>Link</th>
<th>Event data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Avoids long searches but is subject to overflow

---

**Fault Simulation**

**Definition**
- Simulation designed for fault analysis and ATPG

**Applications**
- Evaluate effects, e.g., criticality of given faults
- Evaluate coverage of a given test pattern set
  - To aid in test pattern set generation
  - To construct fault dictionaries
  - To estimate manufacturing yield

\[
DL = 1 - Y^{1-d}
\]

*DL* = probability of shipping a defective IC  
*Y* = probability that a manufactured IC is OK  
*d* = defect coverage of testing approximated by fault coverage
Fault Simulation

Fault Models
- Single stuck-line fault model is standard: any logic line may be stuck at 0 or 1; the logic elements are fault free
- Others: delay faults, short/open faults, etc.

Fault Simulation Methods
- Serial: one fault is simulated at a time
- Parallel: \( n \) faults are simulated at a time
  - “Parallel” method
  - Deductive simulation
  - Concurrent simulation
  - etc.
- Statistical: fault sampling

Parallel Method
- Each \( n \)-bit word in host computer stores \( m \) \( k \)-bit values, \( n > mk \)
- Exploits logical instruction set of host
- Relatively inflexible and usually requires multiple passes

\[
\begin{array}{cccc}
4 & 3 & 2 & 1 \\
z/1 & z/0 & y/1 & x1/1 \\
\end{array}
\]

Fault positions

To Evaluate \( z \)

\[
\begin{array}{cccc}
0 & 1 & 1 & 0 \\
\end{array}
\]
Mask \( I_z \): fault positions for \( z \)

\[
\begin{array}{cccc}
4 & 3 & 2 & 1 \\
1 & 0 & 0 & 0 \\
\end{array}
\]
Mask \( S_z \): fault values

Basic evaluation: \( z = x1.y \)
Fault insertion: \( z^* = z \cdot I_z + I_z \cdot S_z \)
Fault Simulation

Deductive Method

- The “signal” on each simulated line is a list $L$ of all faults causing errors on that line.
- All faults can be simulated in one pass

\[
\begin{align*}
L_a &\quad 0 \\
L_b &\quad 1 \\
L_c &\quad 0 \\
&\quad \ldots \\
F &\quad z \\
&\quad L_a = \text{list of faults causing errors on line a} \\
&\quad \overline{L}_a = \text{list of faults not causing errors on line a}
\end{align*}
\]

Deductive Fault Simulation

Example

\[
\begin{align*}
&\quad L_a = \{a_0\}; \quad L_b = \{b_0\}; \quad L_d = \{d_1\} \\
&\quad L_c = L_a \cap L_b \cup \{c_1\} = \{c_1\} \\
&\quad L_e = L_c \cup L_d \cup \{e_1\} = \{c_1,d_1,e_1\} \\
&\quad L_f = \{f_0\}; \quad L_g = L_e \cup \{g_1\} = \{c_1,d_1,e_1,g_1\} \\
&\quad L_h = \{h_0\}; \quad L_i = \{i_0\}; \quad L_d = \{d_1\} \\
&\quad L_j = L_e \cup \{j_1\} = \{c_1,d_1,e_1,j_1\}
\end{align*}
\]
### Deductive Fault Simulation

**Example (cont’d)**

![Logic Diagram]

\[
L_k = L_i \cup L_h \cup \{k_1\} = \{i_0, h_0, k_1\}
\]

\[
L_l = L_j \cup \{l_1\} = \{c_1, d_1, e_1, j_1, l_0\}
\]

\[
L_m = L_k \cap \overline{L_l} \cup \{m_1\} = \{i_0, h_0, k_1, m_1\}
\]

\[
L_n = L_g \cap \overline{L_l} \cup \{n_1\} = \{c_1, d_1, e_1, g_1, n_1\}
\]

\[
L_p = L_m \cup L_n \cup \{p_1\} = \{c_1, d_1, e_1, g_1, n_1, i_0, h_0, k_1, m_1, p_1\}
\]

---

### Emulation

**Approach**

- Map target design into a network of field-programmable gate array (FPGA) ICs
- The programmed FPGA forms a hardware prototype of the design and can run at effective clock speed of, say, 50 MHz
- The emulated circuit can be instrumented as a logic analyzer to capture normal or faulty behavior

**Drawback**

- Limited ability to do performance measurement
**Emulation**

---

**FPGA: Xilinx 3000 series**

- Logic block
- Programmable connections

---

**Emulation System**

- Host computer
- Host interface
- Config. system
- Programmable interconnect