Logic Simulation

- Purpose
  - Design Verification
  - Performance Evaluation
  - Evaluation of Alternative Designs
  - Debugging
    - observe internal signals
    - can change clock rate, gate delays
    - can start the simulation in any desired state
    - can be interfaced to partial designs

Outline

- Types of Simulation
- Logic Simulation in Presence of Unknowns
- Simulation of Synchronous Sequential Circuits
- Gate Evaluation Methods
- Event-Driven Simulation
- Hazard Detection
Types of Simulation

- Compiled Code
  - functional verification, timing not incorporated
- Table Driven
  - primitives are evaluated using tables
- Event Driven (activity-directed)
  - 1% - 10% of lines are active in a simulation
- Time Driven / Cycle Based

**Event:** change in signal line --> “active line”
Evaluate gates only when inputs change.
Level of simulation depends on level of the model.

Unknown Value

- Unlike real circuits, which have only two signal values (0, 1), a simulator cannot restrict itself to 0, 1. Why?  (0, 1, U, Z, Rising, Falling)
- At the start, flip-flops are in an “unknown” state.
- Formally, unknown is a set of two values {0,1}
  - **Boolean Operation** `op`
    - 0 `op` U = {0} `op` {0,1} = {0} `op` {0, 1} = {0} `op` 0 = 0
    - 1 `op` U = {1} `op` {0,1} = {0,1} = U
**Unknown Value**

- A subtle difference between unknown U and don’t care X
- In Boolean algebraic operations, they are the same, but when differences do exist, care must be taken.

**Logic Simulation**

in Presence of Unknowns

- Use high-level Boolean functions of modules.
Coding for Three Values

- Values: 0, 1, U
- An encoding: 00, 11, 01

<table>
<thead>
<tr>
<th>AND</th>
<th>0</th>
<th>1</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>x</td>
</tr>
<tr>
<td>x</td>
<td>0</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AND</th>
<th>00</th>
<th>11</th>
<th>01</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>01</td>
</tr>
</tbody>
</table>

bit-wise AND on the codes

<table>
<thead>
<tr>
<th>AND</th>
<th>0: 00</th>
<th>AND</th>
<th>U: 01</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: 11</td>
<td>1: 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0: 00</td>
<td>U: 01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OR</th>
<th>0: 00</th>
<th>OR</th>
<th>1: 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: 11</td>
<td>1: 11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1: 11</td>
<td>1: 11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NOT 00 --> 00 --> 11
NOT U --> 01 --> 10 swap

Simulation of Synchronous Sequential Circuits

![Diagram of sequential circuit](image)

Procedure CKT1
inputs A, B; static Q;
begin
  E = B NAND Q
  F = A OR B
  Q = F
end

Every call advances the clock by 1.

Assume data is stable at inputs to flip-flops when clock signal arrives.
Gate Evaluation Methods

Controlling value and Inversion value:
- Controlling value controls the output of the gate
- If there is a controlling input, output of gate is \( c \oplus i \)
  otherwise \( c' \oplus i \)

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

Gate Evaluation Methods

- Input scanning
  - Look for controlling value \( c \)
  - If found then output = \( c \)
  - else if unknown found then output = \( X \)
  - else output = \( c \)

- Count based
  - \( C_0 = \) count of 0’s
  - \( C_x = \) count of x’s
  - if \( C_0 > 0 \) then output 0
  - else if \( C_x > 0 \) then output \( x \)
  - else output 1

- Table based -- indexed look-up
Table-Based Indexed Look-up

- \{0, 1, x\}^3 = 27 possible vectors
- \(2^6 = 64\) entries in table, since 0,1,\(x\) require 2 bits

Index \(I = \begin{array}{ccc}
A & B & C \\
\end{array}\)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D[I]</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
<td>11</td>
<td>1</td>
</tr>
</tbody>
</table>

Of 64 entries, 27 are useful

**zoom tables**: gate type is part of index

---

Event-Driven Simulation

- All \(\tau = 0\): zero delay simulation
  - easiest, fastest
- All \(\tau = 1\): unit delay simulation
  - moderate
- Different \(\tau\)'s: variable delay, assignable delay simulation
  - hardest, slowest
Event-Driven Simulation

- Different gates may have different delays
  - In the project you will implement a zero-delay simulator
  - Logic Simulation Algorithm with Delay
    Two-pass strategy when delay is not zero:
    - Need to schedule events, that is, changes of logic values
    - Need to keep a queue of activated gates

Observation on Logic Simulation Algorithm with Delay

- Correct only if an event at time $t+k$ does not schedule an event earlier than an already scheduled event for the same gate.
- Example:
  - Event list:
    0: (a,0)
    1: (a,1)
    6: (e,0)
    2: (d,0)
    4: (e,1)  Need to cancel 6: (e,0)
Zero-Delay vs. Unit-Delay Simulation

- In unit-delay simulation, gate A is evaluated 3 times.
- In zero-delay simulation, gate A is evaluated once.
  - zero-delay gives steady-state values of nodes.

---

Zero-Delay Simulation Example

- Conceptually, one event queue per level.
Assigned-Delay Simulation Example

Time \( t_1 \)
- Pass 1: \( a \neq a' \) --> \( (a'1) \) is an event, add \( g \) to Activated
- \( c = c' \) --> \( (c'1) \) is an event, add \( h \) to Activated
- \( d = d' \) --> \( (d'0) \) is an event, add \( h \) to Activated
- \( e = e' \) --> \( (e'1) \) is an event, add \( i \) to Activated

Time \( t_4 \)
- Pass 1: \( h' = h \), so no event
- \( i' \neq i \) --> \( (i',0) \) is an event, add \( I, m \) to Activated

Pass 2: \( l' = 1 \) at time \( 4+4 = 8 \)
- \( m' = 0 \) at time \( 4+4 = 8 \)
Assigned-Delay Simulation Example

Event List

- $t_1$: $(a',1), (c',1), (d',0), (e',0)$
- $t_6$: $(g',1)$
- $t_4$: $(h',1), (i',0)$
- $t_8$: $(l',1), (m',0)$
- $t_7$: $(j',0), (k',0)$
- $t_{10}$: $(l',1)$

Must schedule since there may be $(h,0)$ scheduled before time $t_4$

Time Event

- $t_6$: $(g',1)$ means at time $t_6$, $g$ will take on value 1.

---

Unit-Delay Simulation Example

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```

```
0 --> 1
1 --> 0
```
Hazard Detection

| A 0 --> 1 slow | 001 |
| B 1 --> 0 fast | 100 |
| 111 |
| A 0 --> 1 fast | 011 |
| B 1 --> 0 slow | 110 |
| 101 |

- Important for unclocked flip-flops (latches)
  - \( t \): \( A=0, B=1 \) \( \rightarrow 1 \)
  - \( t' \): \( A=U, B=U \) \( \rightarrow U \)
  - \( t+1 \): \( A=1, B=0 \) \( \rightarrow 1 \)

- Hazard exists if and only if the output sequence has 0U0 or 1U1