Agenda

- Review of Sequential Circuits and Mealy Finite State Machines (FSM)
- The concept of FSM equivalence
- Fault excitation and propagation conditions in FSM
- Reset states, synchronizing sequences and redundant states
- Untestable faults in sequential circuits
Sequential Circuits

- We consider sequential circuits with edge-triggered D-flip-flops
- Underlying a sequential circuit is a finite state machine (FSM)
- We will consider only FSMs of the Mealy-type (Moore machine ATPG is similar)
- An example of sequential circuit:
For ATPG, we are not given the FSM. Given only the circuit:

**FSM and Circuit example**
Find a sequence of inputs (or input vectors) that produces an output 1 at the miter output, sometime in the future clock cycles.
Two Machines, $\mathcal{M}_1$ and $\mathcal{M}_2$, are equivalent if

- They are identical; or
- They have identical states but different encoding; or
- $\mathcal{M}_1 \subseteq \mathcal{M}_2$ or vice-versa; or
- They have different reachable states but same distinguishable states (same condition as above); or
- Different unreachable states, and unreachable states are a *don’t care* condition

Prove that two machines (sequential circuits) produce the same output response on application of all possible input sequences
Sequential ATPG is Harder than Combinational

- Given: A sequential circuit, generate ATPG. Sometimes reset (starting) state available, sometimes not!
- Procedure: Activate a fault, and propagate it to PO
- Bottlenecks: Fault activation requires an “activation state”. How to get to the activation state? Fault can be propagated to the next-state line, but not to PO? Unroll the machine, or in other words, traverse to other states.
- How to distinguish between Faulty and Fault-free machines? The concept of state distinguishing sequences.
- In absence of resets, how to synchronize both faulty and fault-free machines to the same states. Concept of synchronizing sequences.
A fault in a sequential circuit can be untestable due to **four reasons:**

- Redundancy in a combinational logic
- Lack of a common synchronizing sequence that can drive both faulty and fault-free machines to a common (starting) state
- Sequential Redundancy: a fault causes a transition to a (faulty) state which may be equivalent to a fault-free state
- Fault excitation or propagation requires getting to an unreachable state

We will look at all these cases!
First example: When reset state exists

- Start the machine in a reset state (00), traverse FSM
- At some point in the future, faulty and fault-free machines will diverge

Test: \( x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1 \)

\( x_0 = 0 \) takes me back to (00) \( \Rightarrow \text{can't get out} \)
First example: When reset state exists

\[ x_0 = 1, \; y_1 = 0, \; y_2 = 1, \; Z = 0. \]

- Start the machine in a reset state (00), traverse FSM.
- At some point in the future, faulty and fault-free machines will diverge.

Test: \( x_0 = 1, \; x_1 = 1, \; x_2 = 0, \; x_3 = 1 \)

\[ \Rightarrow \; y_1 = 1, \; y_2 = 1, \; Z = 0 \]
First example: When reset state exists

- Start the machine in a reset state (00), traverse FSM
- At some point in the future, faulty and fault-free machines will diverge

Test: \( x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1 \)

\( Y_1 = \overline{Y_0} \quad Y_2 = 1 \)
First example: When reset state exists

TF3: $y_1 = D, y_2 = 1$

- Start the machine in a reset state (00), traverse FSM
- At some point in the future, faulty and fault-free machines will diverge

Test: $x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1$
Meaning of the above Test

- Fault $D$ or $\overline{D}$ gets into next state line.
- Faulty and Fault-free machines go to different states.
- Machine operation diverges....
- Sometime in the future, you can catch that effect.
- You can distinguish between faulty and fault-free states of the machine.
- How? Using state distinguishing experiments.
When there is no reset state?

- Power up the machine, it can be in any (unknown) state
- Does there exist a sequence of inputs that can drive the machine to some – singleton – state?
- Some machines have a synchronizing sequence, others do not
- Given an FSM, how to find a synchronizing sequence?
- For ATPG: we are not given the FSM, only the circuit
- For ATPG: We need to find a synchronizing sequence for both the faulty and fault-free machines that can drive both to a common state. This can be hard.
Synchronizing Sequence on an FSM

Table: State Transition Table

<table>
<thead>
<tr>
<th>P.S.</th>
<th>Next State, Z $x = 0$</th>
<th>Next State, Z $x = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B, 0</td>
<td>D, 0</td>
</tr>
<tr>
<td>B</td>
<td>A, 0</td>
<td>B, 0</td>
</tr>
<tr>
<td>C</td>
<td>D, 1</td>
<td>A, 0</td>
</tr>
<tr>
<td>D</td>
<td>D, 1</td>
<td>C, 0</td>
</tr>
</tbody>
</table>

[Diagram showing state transitions]
This machine has no Synchronizing Sequence

P. Kalla (Univ. of Utah)  Sequential ATPG

Slides updated Dec 1, 2021  16 / 19
ATPG without Reset State

\[ TFO: (y_1 = 0, y_2 = 1) \]
No reset state. Start w/ Fault excitation state: \( y_1 = 0, y_2 = 1 \).

\( x = 1 \) propagates the fault effect!

Problem: How to get to the fault excitation state?

Answer: Sequential backtrack! Unroll the m/c backwards until you can get unknown values in FFs. This implies there exists a state (backward in time from F.E. state) where you can “control” the FFs (control the state).

Called Self-Initializing tests! This test implies that both faulty and fault-free machines can be initialized to a “common initial state”; i.e., you can get a common starting point.