Sequential Circuit Testing

Sequential Circuits and Finite State Machines

Priyank Kalla

Professor
Electrical and Computer Engineering, University of Utah
kalla@ece.utah.edu
http://www.ece.utah.edu/~kalla

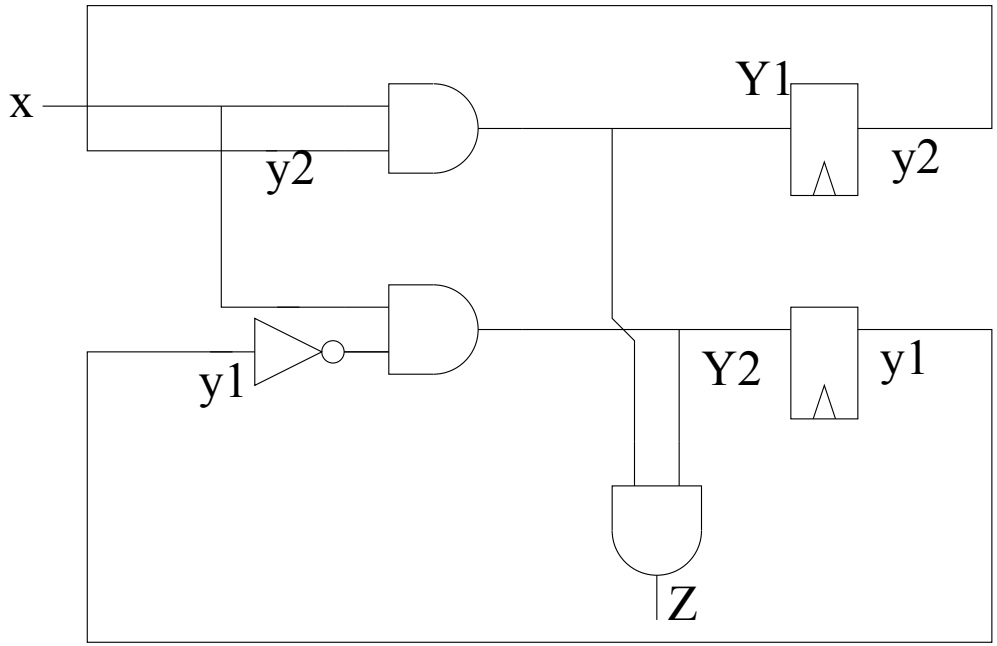
Slides updated Dec 1, 2021

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

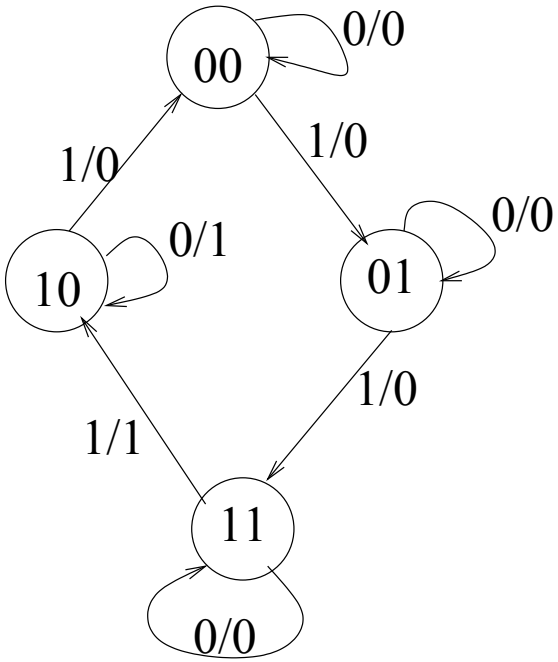
Sequential Circuits

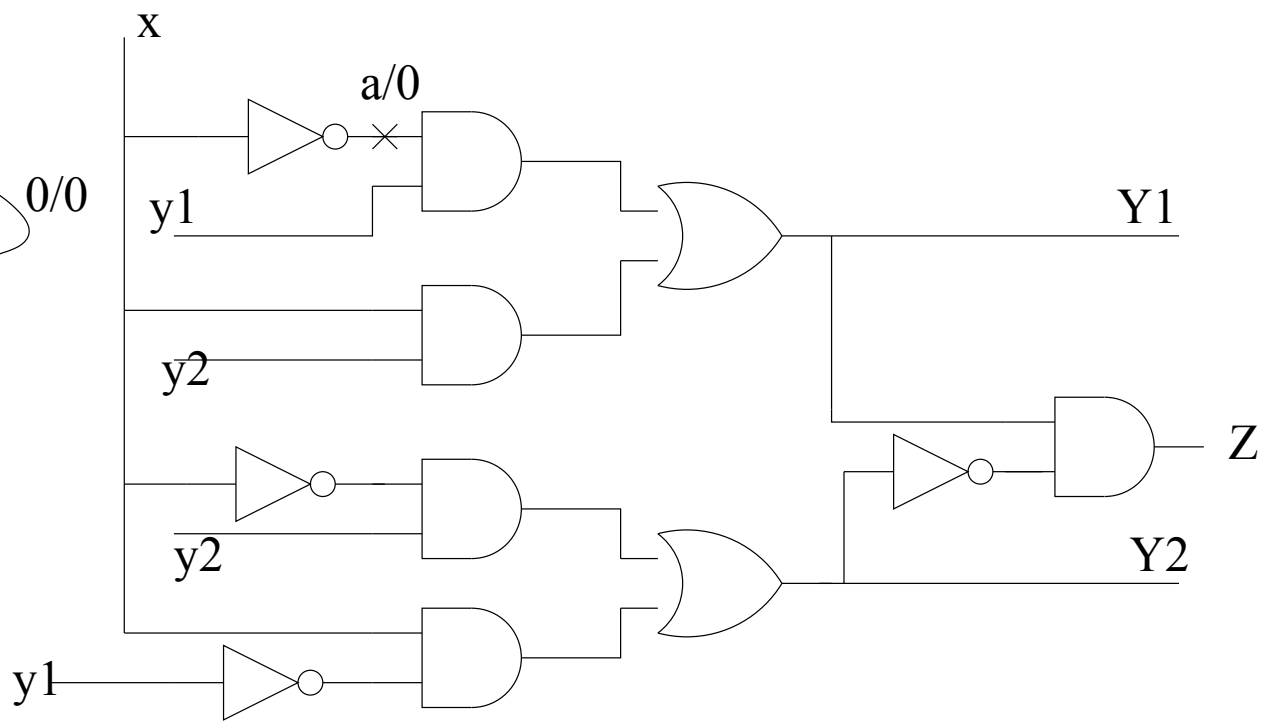
- We consider sequential circuits with edge-triggered D-flip-flops
- Underlying a sequential circuit is a finite state machine (FSM)
- We will consider only FSMs of the Mealy-type (Moore machine ATPG is similar)
- An example of sequential circuit:



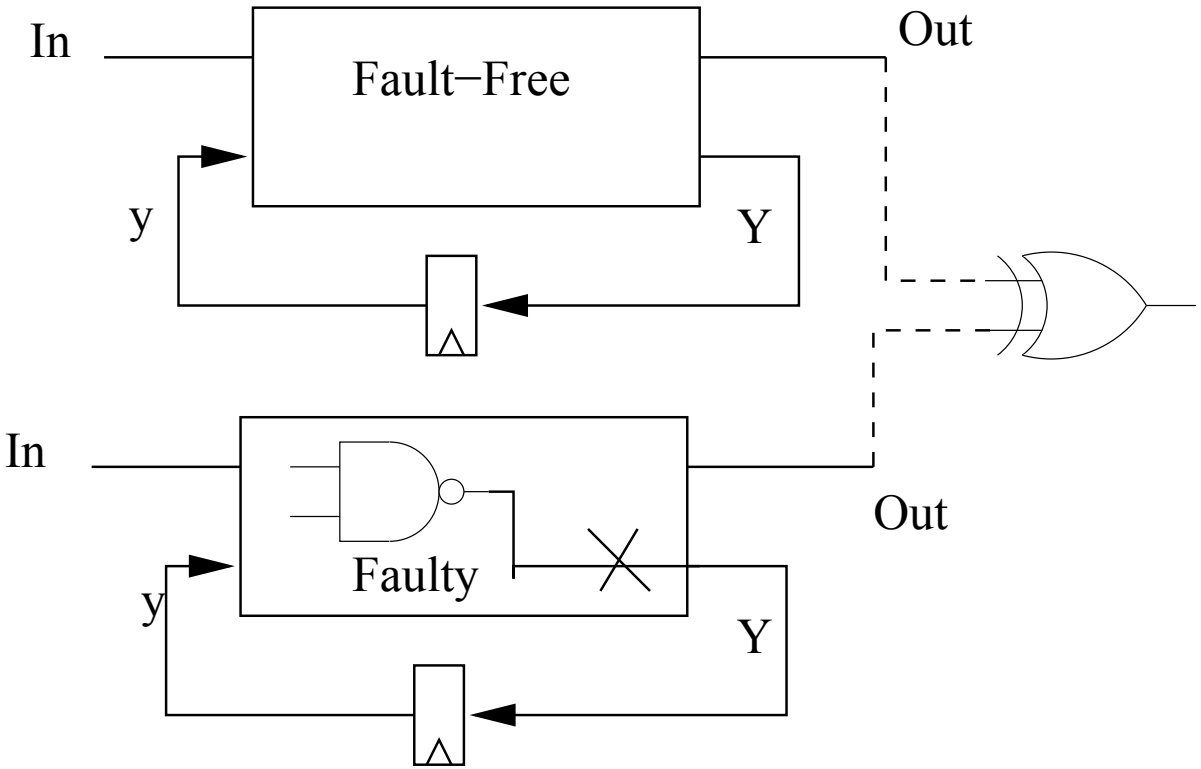
FSM and Circuit example

For ATPG, we are not given the FSM.
Given → only the circuit





Sequential Circuit Testing: Sequential Miter



- Find a **sequence of inputs** (or input vectors) that produces an output 1 at the miter output, sometime in the future clock cycles

FSM Equivalence

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 seqences.
- In absence of resets, how to synchronize both faulty and fault-free machines to the same states. Concept of synchronizing sequences.

Untestable Faults in Sequential Circuits

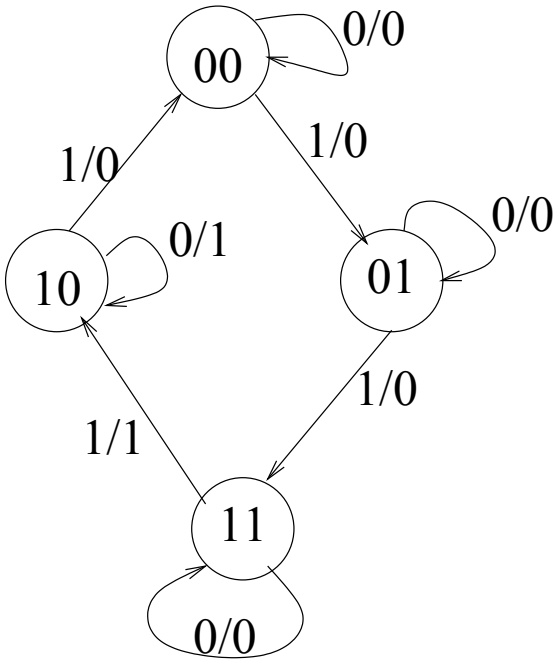
A fault in a sequential circuit can be untestable due to **four reasons**:

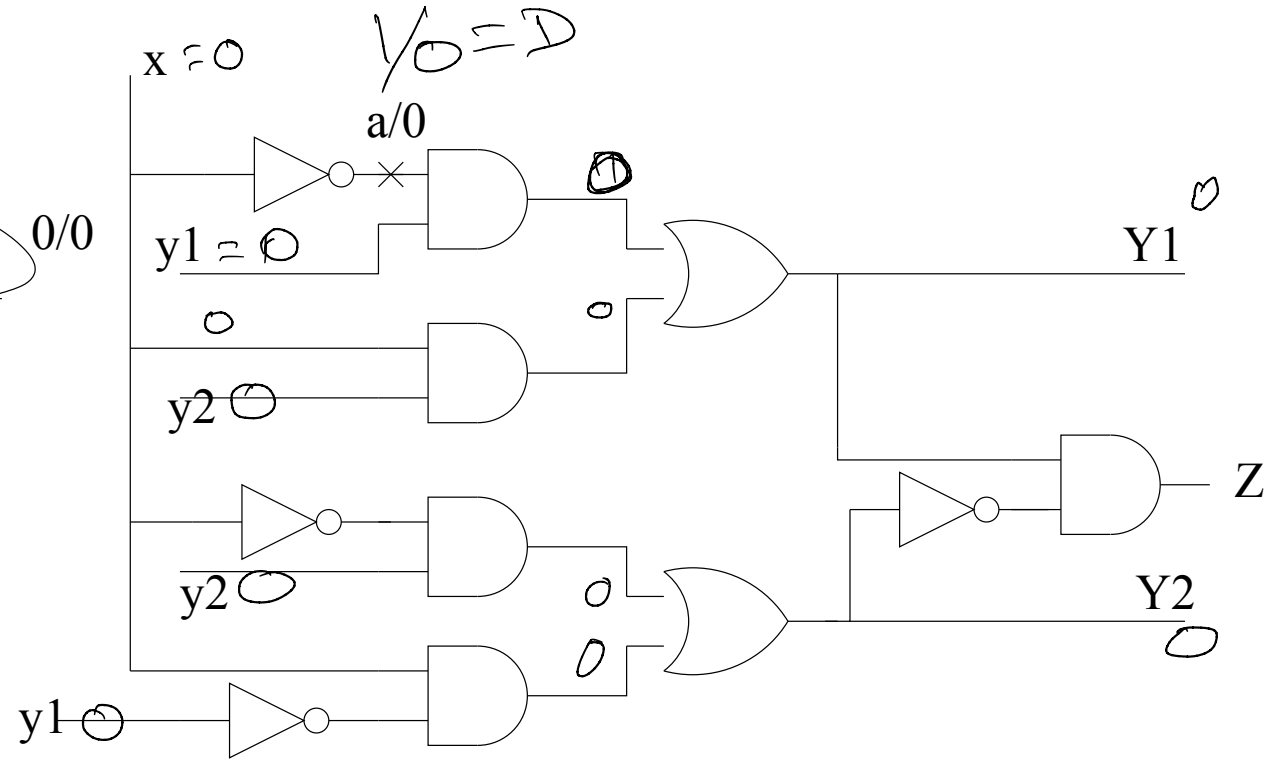
- Redundancy in a combinational logic
- Lack of a common synchronizing sequence that can drive both faulty and fault-free machines to a common (starting) state
- Sequential Redundancy: a fault causes a transition to a (faulty) state which may be equivalent to a fault-free state
- Fault excitation or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





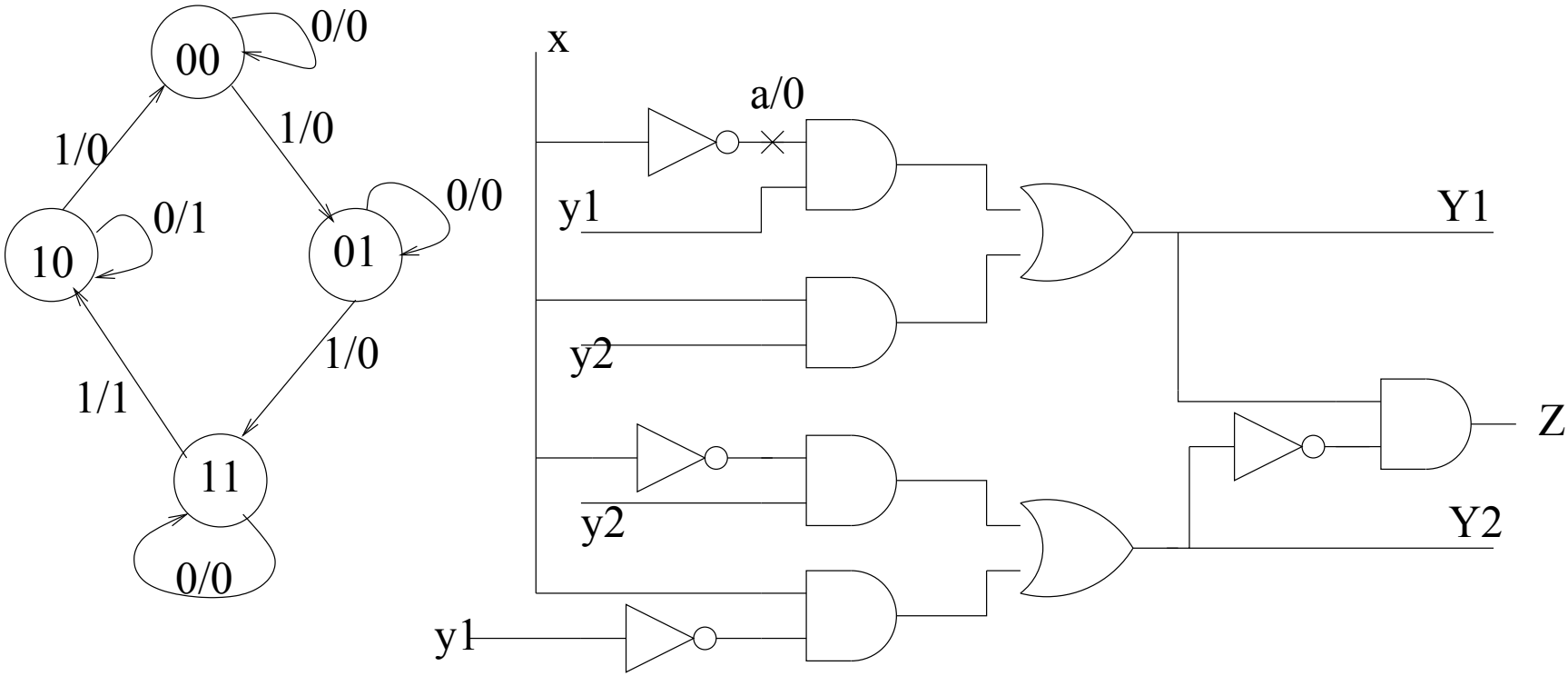
Test: $x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1$

$X_0 = 0$ takes me back to (00) → can't get out ....

First example: When reset state exists

$X_0=1$, $Y_1=0$, $Y_2=1$, $Z=0$. T.F.1 : $y_1=0, y_2=1$

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

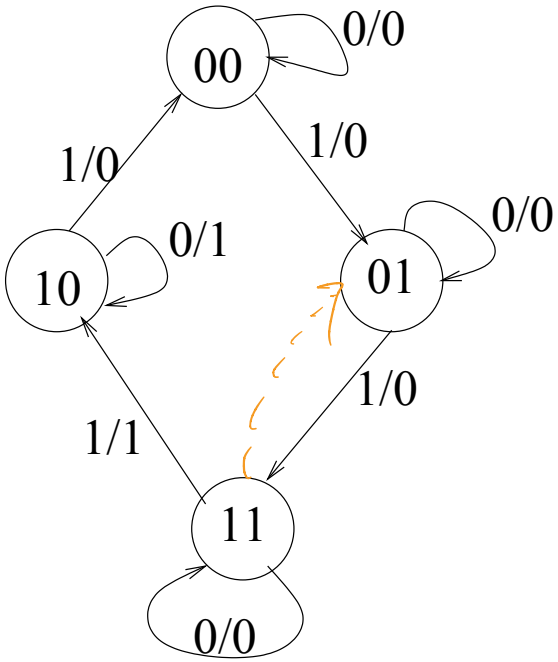


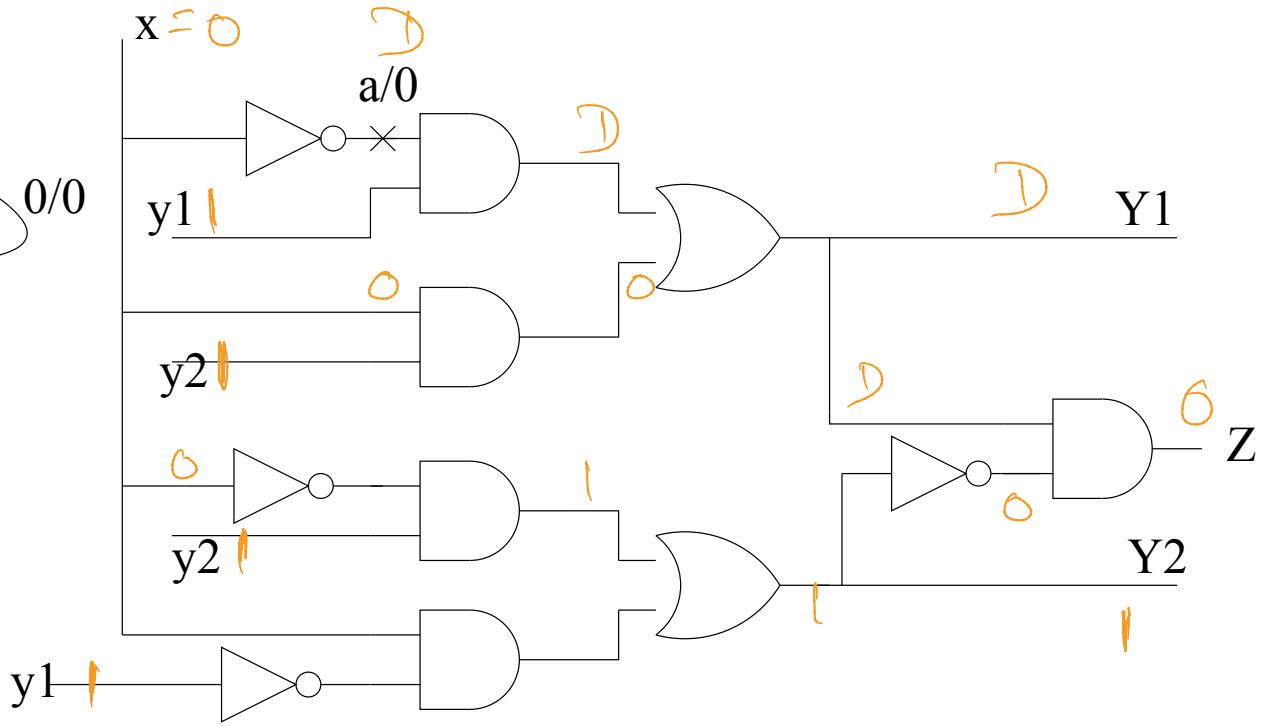
Test: $x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1$

↳ $Y_1=1$, $Y_2=1$, $Z=0$

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



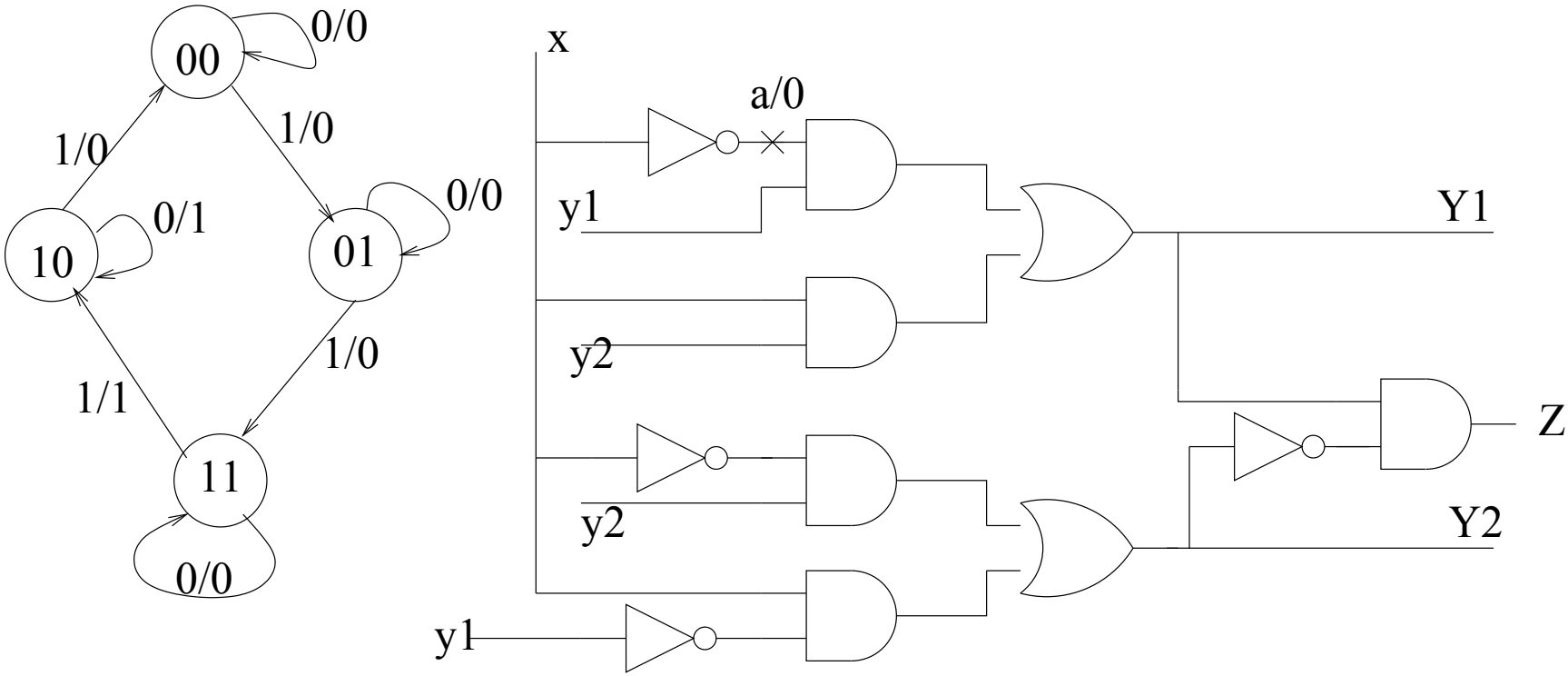


Test: $x_0 = 1, x_1 = 1, x_2 = 0, x_3 = 1$

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: $x_0 = 1, x_1 = 1, x_2 = 0, \underline{x_3 = 1}$

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 faulty-free states of the machine.
- How? Using state distinguishing experiments.

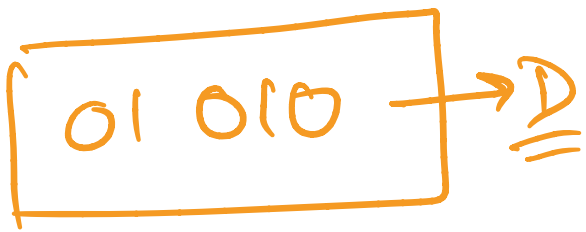
When there is no reset state?

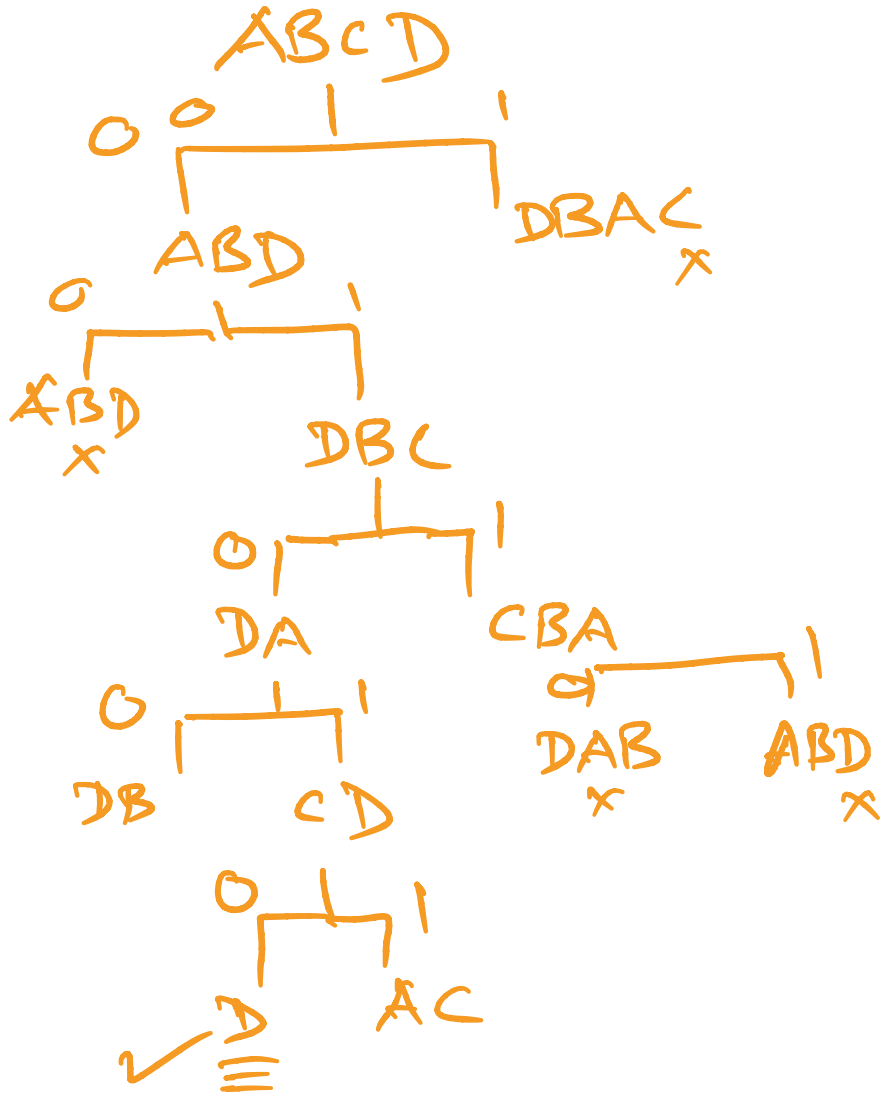
- Power up the machine, it can be in any (unknown) state
- Does there exist a sequence of inputs that can drive the machine to some – singleton – state?
- Some machines have a synchronizing sequence, others do not
- Given an FSM, how to find a synchronizing sequence?
- For ATPG: we are not given the 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

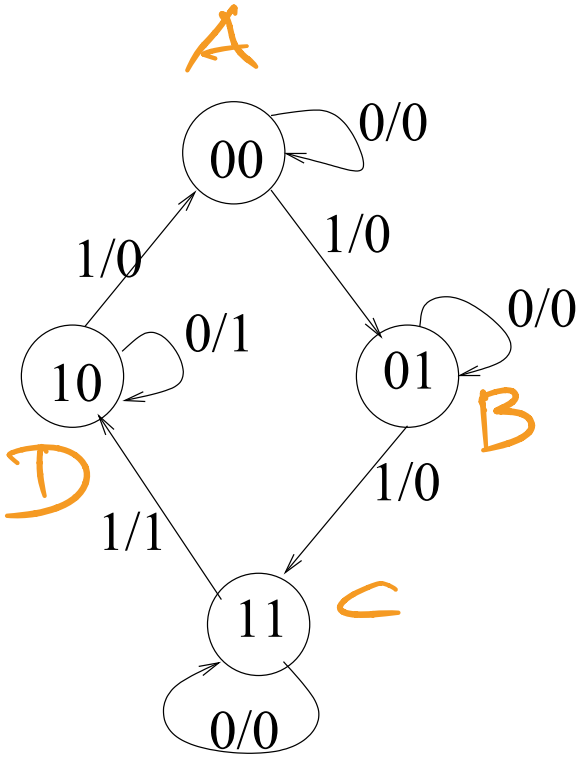
Table: State Transition Table

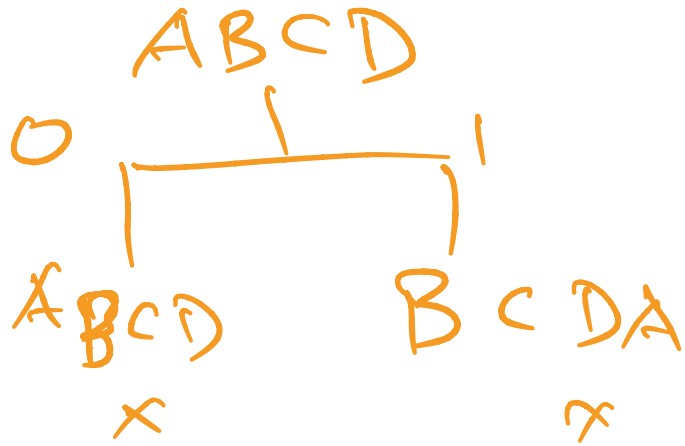
P.S.	Next State, Z	
	$x = 0$	$x = 1$
A	B, 0	D, 0
B	A, 0	B, 0
C	D,1	A,0
D	D,1	C,0



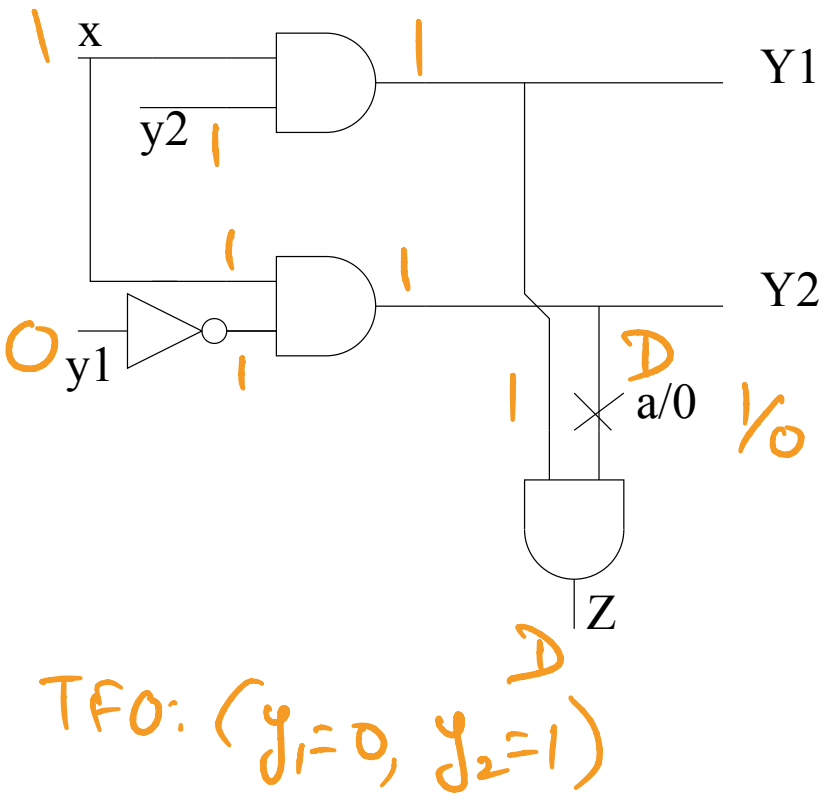


This machine has no Synchronizing Sequence





ATPG without Reset State

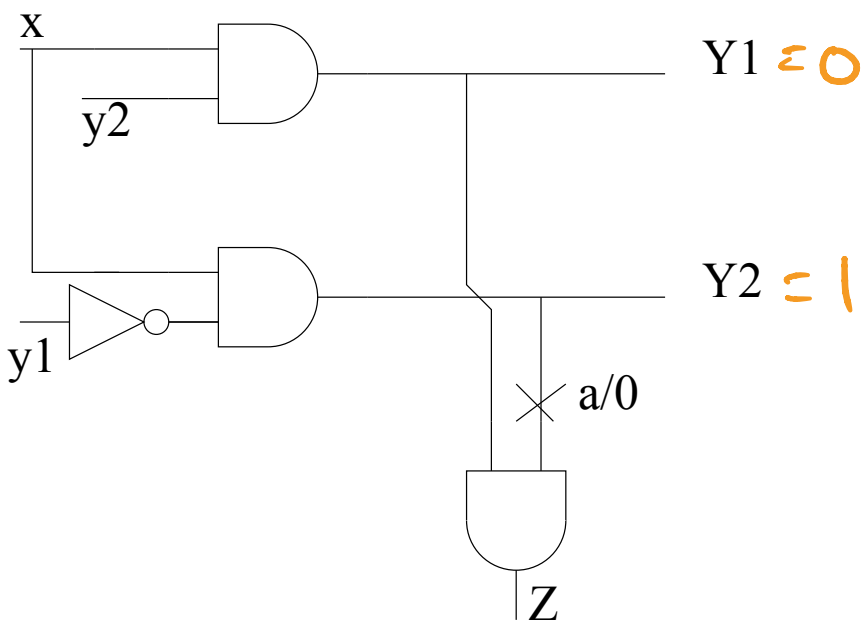


No Reset State Example: Test Exists

- 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.

ATPG without Reset State

Go back.



ATPG without Reset State