

Topic 14

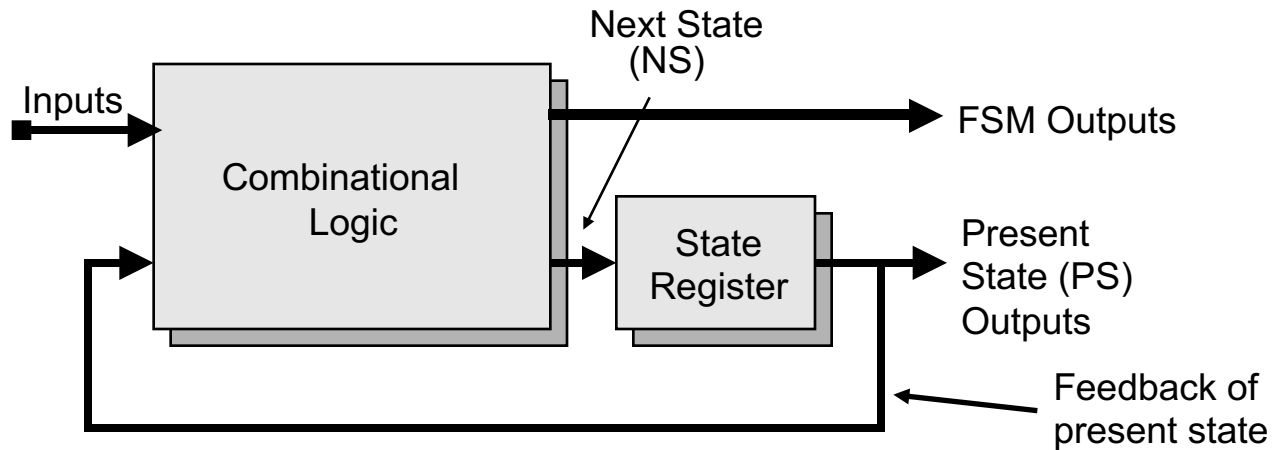
Timing Issues

Outline

- Timing requirements for FSM
- Setup time and hold time
- Asynchronous input and metastability
- Synchronizer
- Switch debouncing
- Clock skew
- Hazards

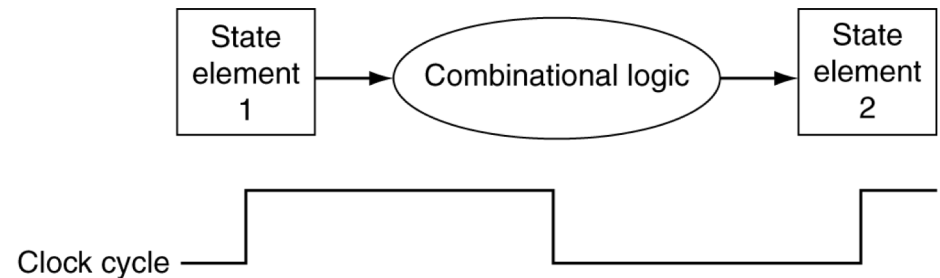
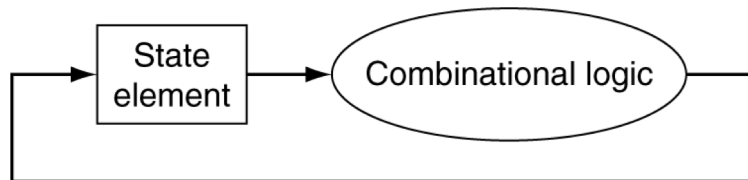
State Machine

- Finite State Machine



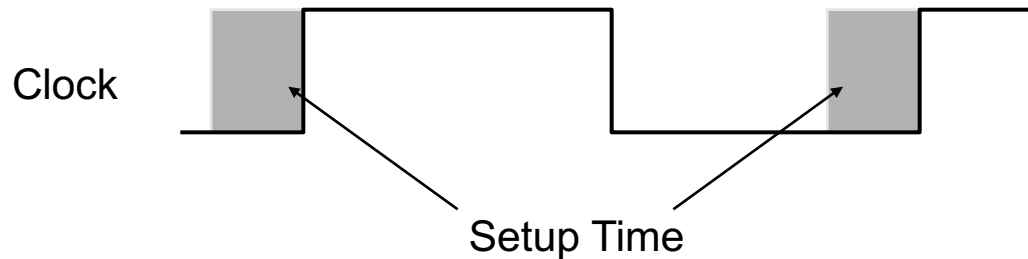
Clocking Methodology for FSM

- Combinational logic calculates data during clock cycles
 - Between clock edges
- Clock cycles should be
 - Long enough to allow combinational logic completes computation
 - Longest delay determines clock period
 - Short enough to ensure acceptable performance and to capture small changes on external inputs



Hardware Constraints – Setup and Hold

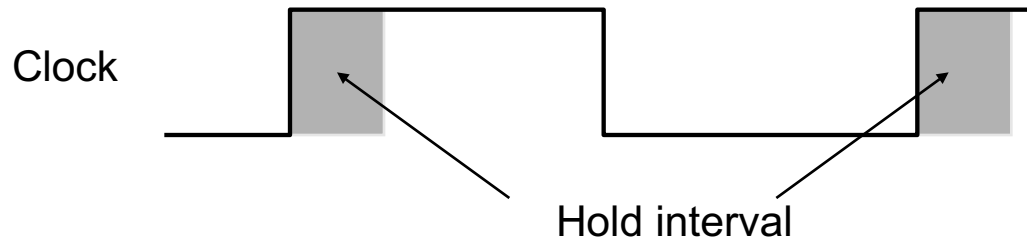
- An edge-triggered flip-flop will not operate correctly if the data is not stable for a sufficient time before and after the clock edge
- Storage element may be put in nondeterministic state
- Setup Time
 - Minimum time that data must be stable prior to the triggering edge



- Setup-time violations are caused by combinational paths that are long relative to the clock cycle
 - Fix: stretch clock, optimize combinational circuit, split combinational circuit, pipelining,

Hardware Constraints – Setup and Hold

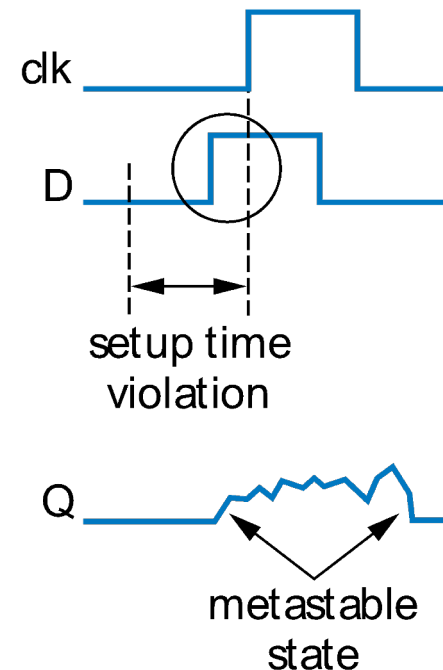
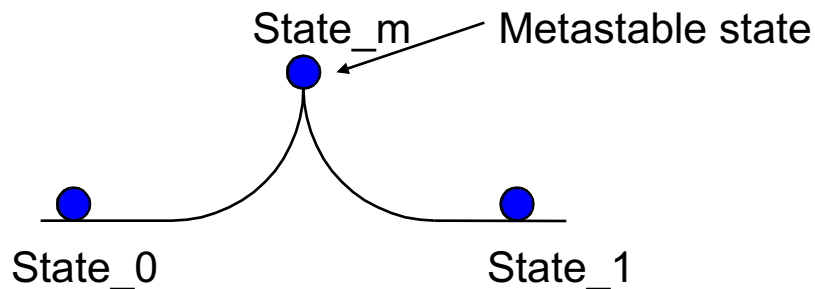
- Hold Time:
 - Minimum time that data must remain stable after the triggering edge



- Hold-time violations are caused by short paths that allow a signal to propagate from a source flip-flop to a destination flip-flop and change the data that was created in the previous cycle before the destination flip-flop has registered its output
 - Fix: insert buffers, use different flip flop
- Setup and hold time: ns

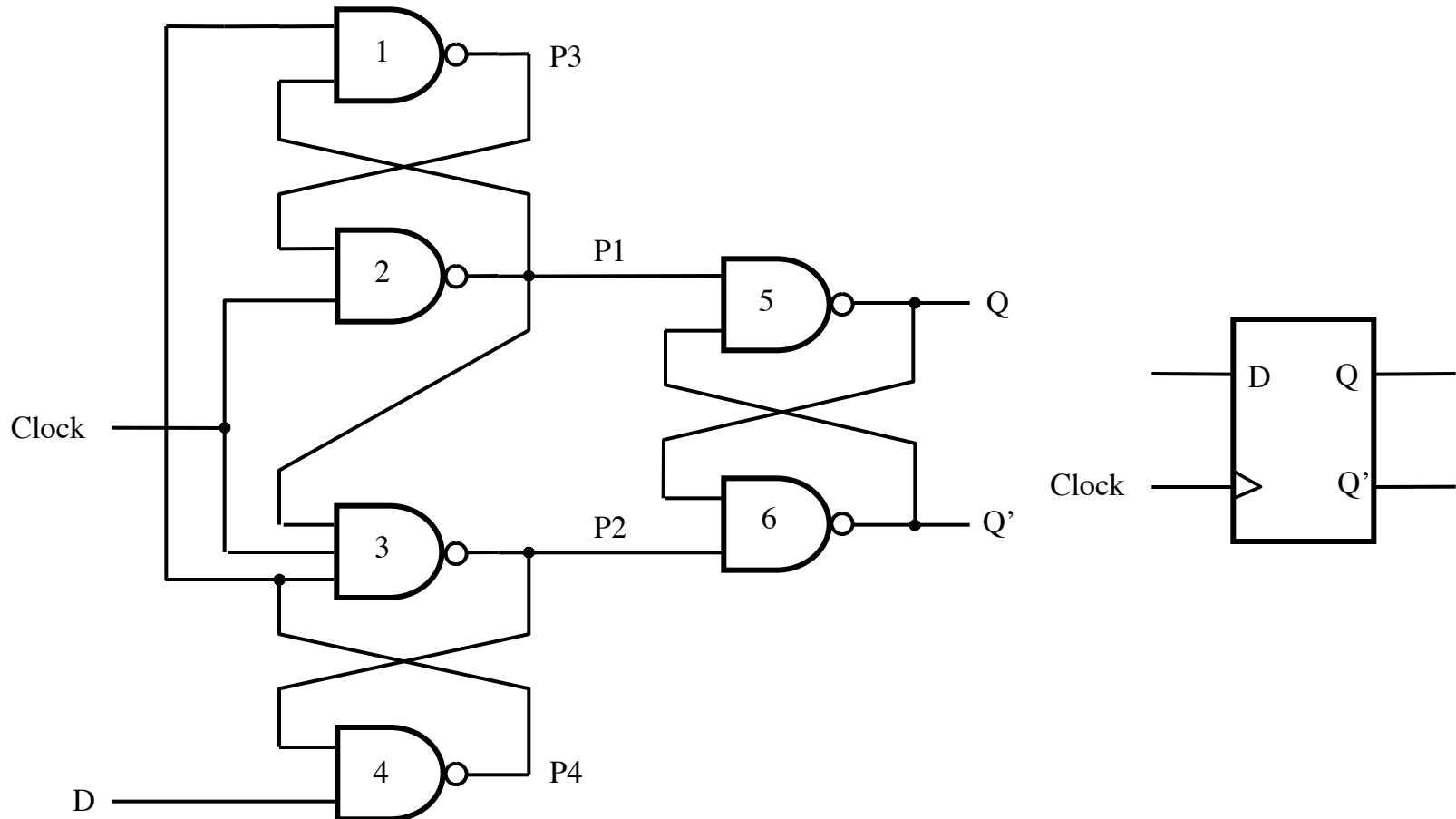
Metastability and Asynchronous Inputs

- If the setup or hold time of an edge-triggered flip-flop is violated the flip-flop may enter a metastable state.
 - Metastable state: Any flip-flop state other than stable 1 or 0
 - Eventually settles to one or other, but we don't know which
 - For internal circuits, we can make sure setup time is satisfied
 - But what if input comes from external (asynchronous) source, e.g., button press?



Metastable State

- Assuming $D = 0 \rightarrow 1$ while rising edge of clock ($0 \rightarrow 1$)
 - May oscillate and never stops if D and clock changes at exactly same time
 - In reality, DFF will settle eventually after some time (hopefully within a clock cycle)



Asynchronous Inputs

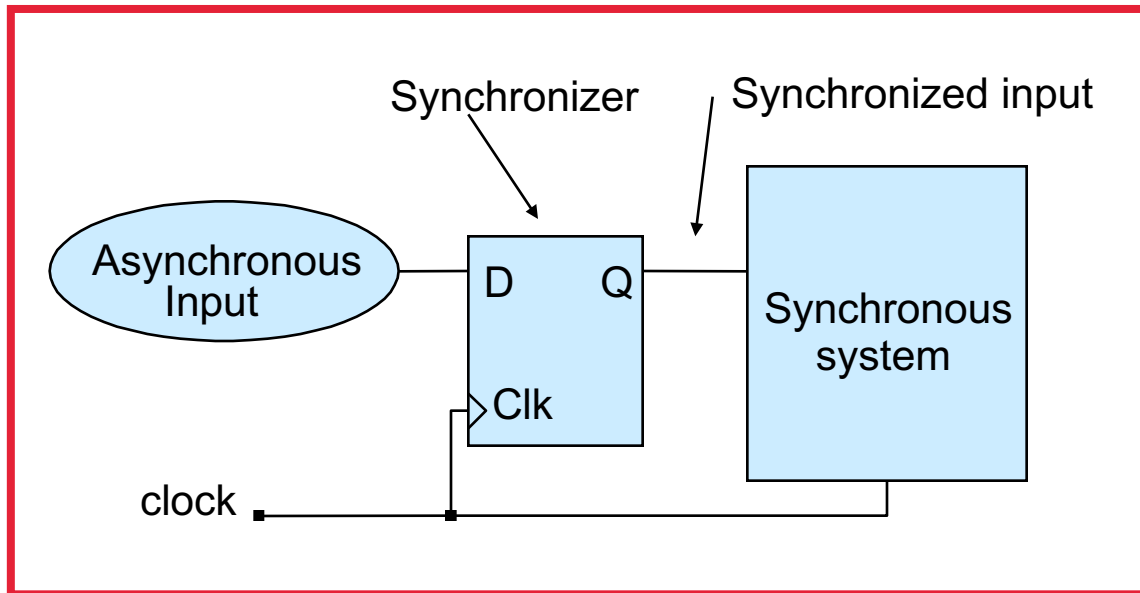
- In a fully synchronous system, all circuits are driven by a common clock, and the state registers may change only under the control of the clock
- Synchronous system may be driven by asynchronous inputs which may put flip flops into the metastable state
- Examples: Keyboard activity, push buttons, and interrupts to a computer

Problems and Solution

- Problems caused by asynchronous inputs
 - The **unpredictable arrival** of an asynchronous input may cause a setup condition to be violated.
 - **Glitches** on the asynchronous inputs may cause system failure.
 - Asynchronous **fanout path delays** might cause some input transitions to be "caught " by some devices and "missed" by others
- **General rule:** Ensure that all inputs are synchronous, i.e. synchronize them to the clock signal. Reset and set can be exceptions.

Synchronizer

- Use a D-type flip flop to synchronize the input.

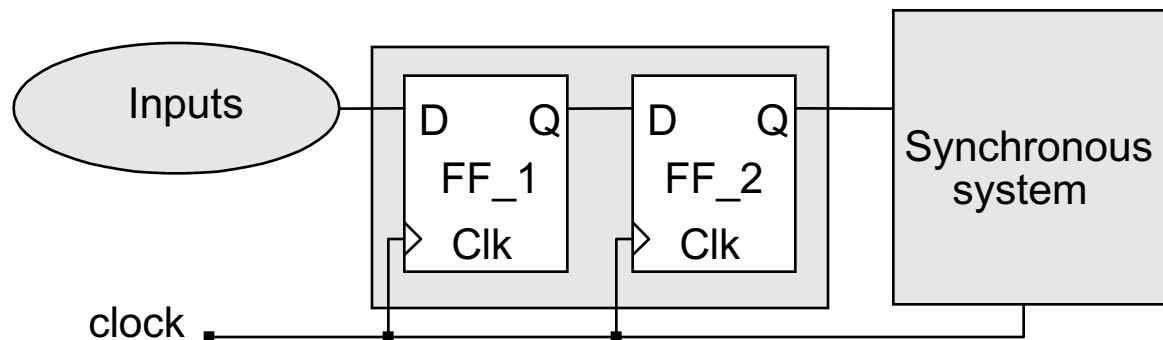


Synchronizer Failure

- The systems input is synchronized, but the setup and hold conditions of the synchronizing flip-flop may be violated.
 - The synchronizing flip-flop may enter a metastable state and disrupt system operation.
- The flip-flop may be in metastable state, then its state will be nondeterministic, its output may be interpreted as
 - “0”, or
 - “1”, or
 - enter a metastable state, output is not synchronized anymore
- A circuit that uses the output of a synchronizer that is in the metastable state is said to experience a **synchronizer failure**.
- To recover from a synchronizer failure
 - wait for the circuit to leave the metastable state, the time is unpredictable, or
 - execute a **reset** to put the system into a known state.

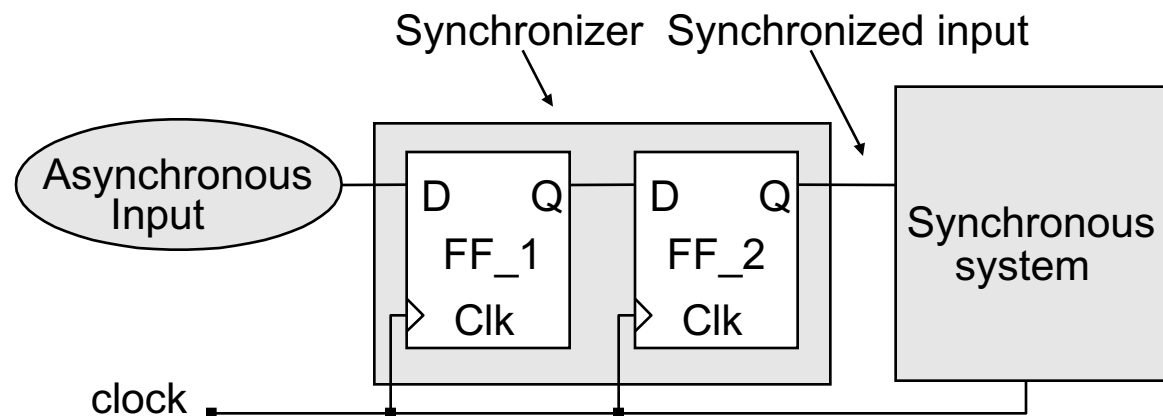
Reduce Possibility of Synchronizer Failure

- Use flip-flops having the shortest possible setup and hold intervals
- stretch the clock period to allow more time for the circuit to recover
 - Output finally settles in 0 or 1
 - But means slower circuit
- Insert **double-synchronizing D-type flip-flops** after the asynchronous inputs



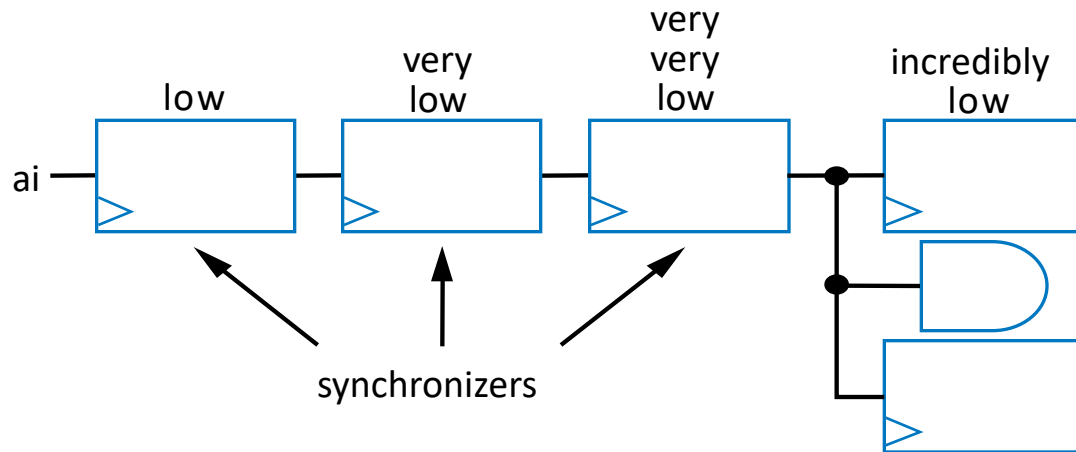
Double DFF Synchronizer

- The first flip-flop might enter a metastable state, but has a clock period to recover
- Both flip-flop must enter a metastable state to have a synchronizer failure
- It is less likely that FF_2 will see an input that is not valid, thus it is less likely that FF_2 will enter a metastable state
 - Typically, 1 clock cycle is long enough for 1st DFF to recover
 - Output of 2nd DFF is synchronized and stable, but **unpredictable**



Synchronizer with More FFs

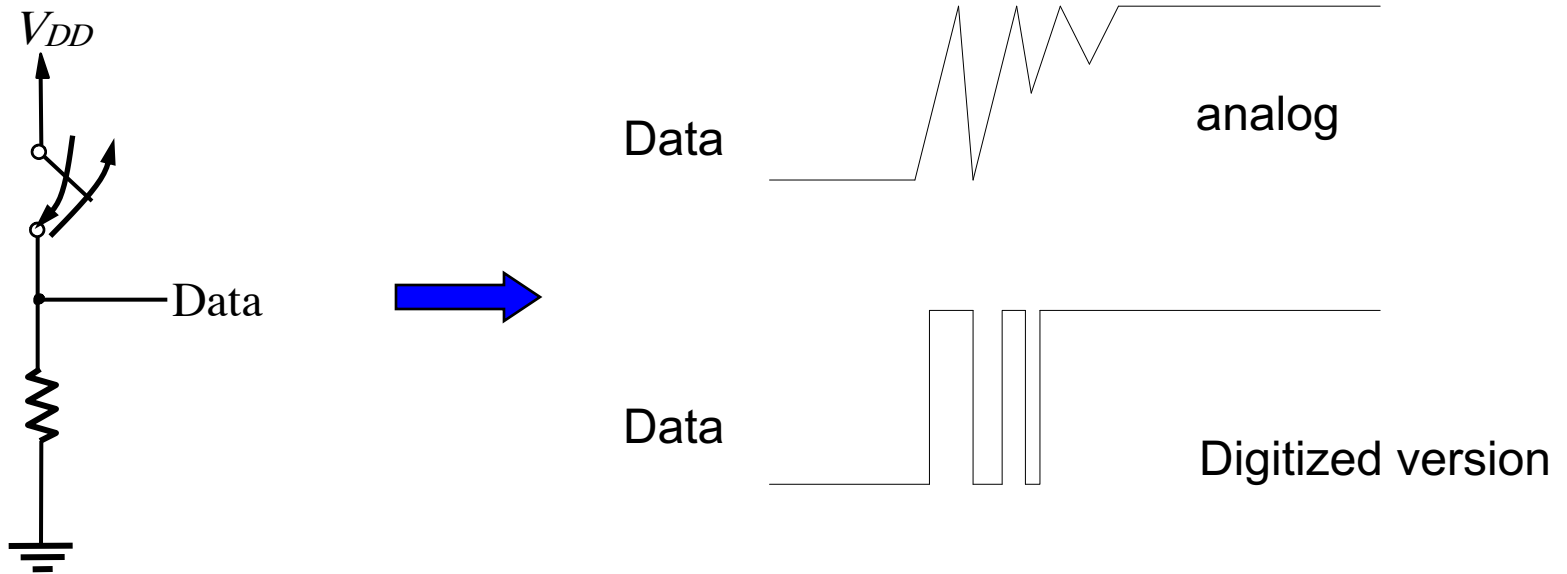
- More FFs is unnecessary



- However, synchronizer failure can never be 100% avoided
 - We can just maximize the Mean Time Between Failures (MTBF) -- a number often given along with a circuit

Bouncing Switch

- Asynchronous input may be produced by a mechanical switch or a push button (mechanical component)
- When changing from one position to another, the switch may bounce away from the contact point

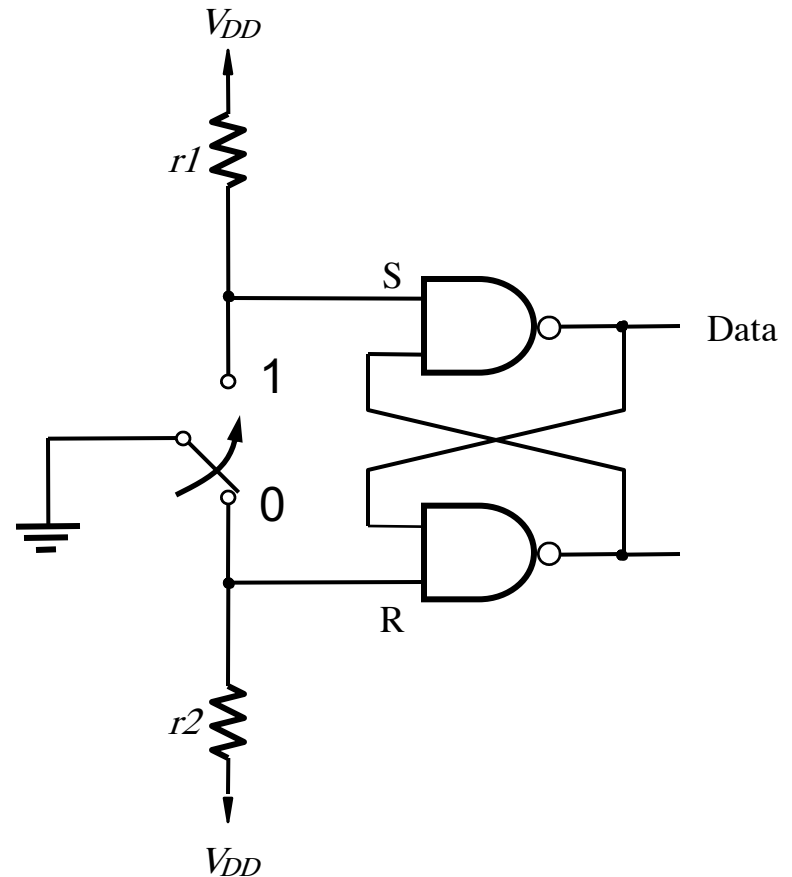


Switch Debouncing

- Debouncing circuit: SR latch
- When throw is at position 0,
 $S = 1, R = 0, \text{Data} = 0$
- When throw changes to position 1
 $R = 1, S = (0 \longleftrightarrow 1), \text{Data} = 1$
- Data is latched regardless of S
 assuming switch won't bounce
 back to position 0 and touch it

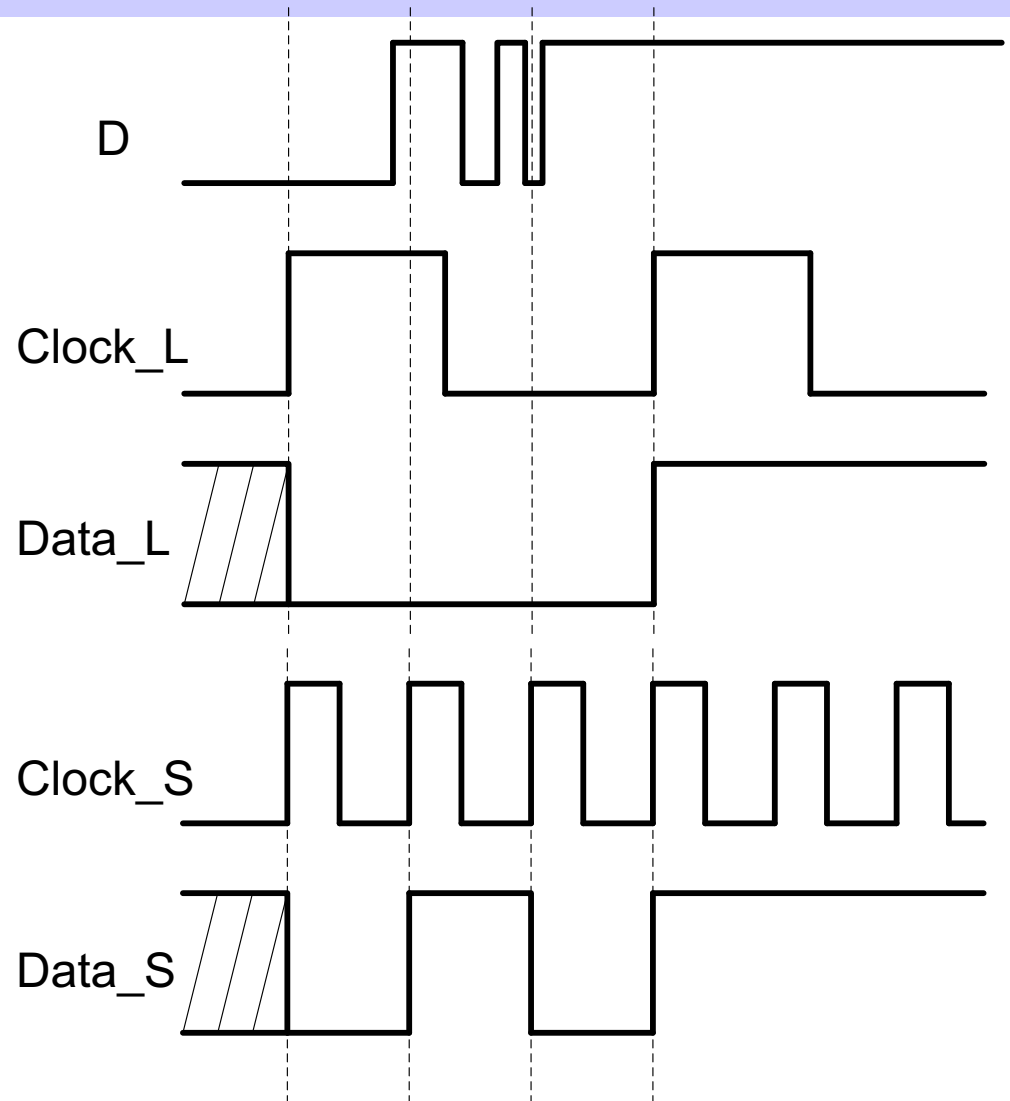
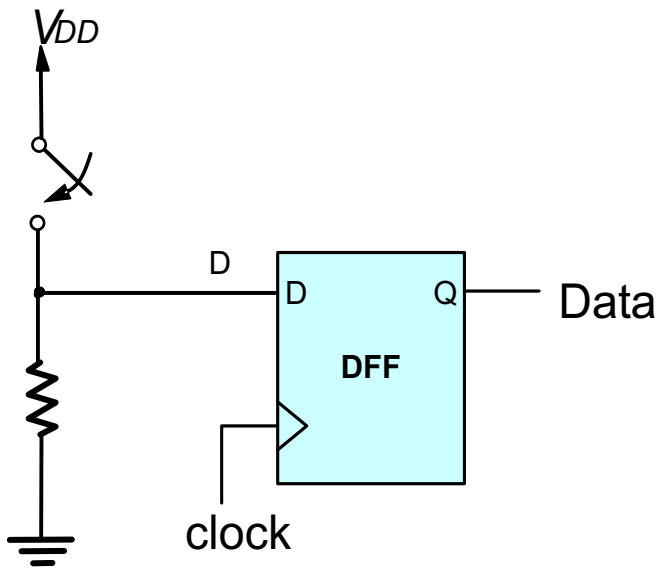
S	R	Q	Q ⁺
0	0	0	X
0	0	1	X
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

not allowed
 set
 reset
 hold



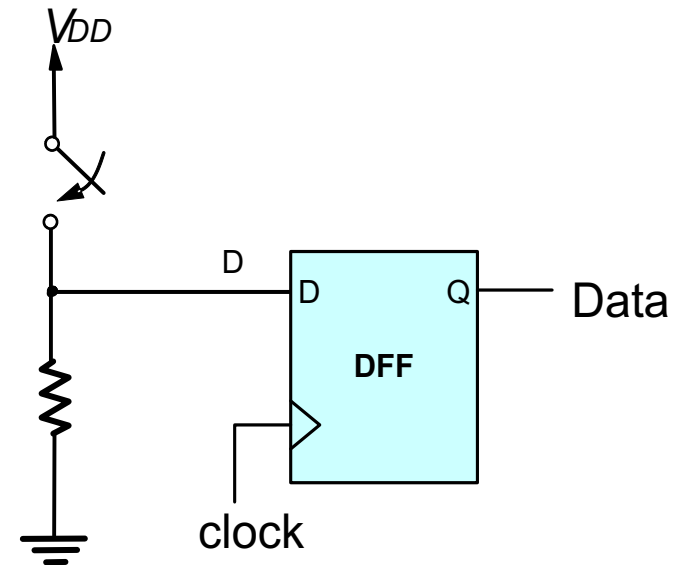
Switch Debouncing

- D Flip Flop may be used for switch debouncing
- But clock needs careful design



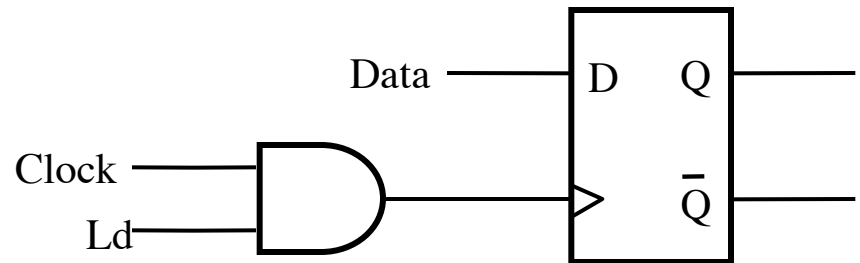
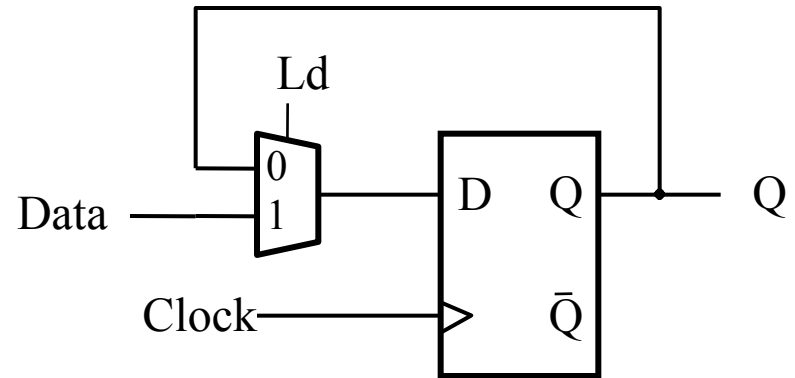
Switch Debouncing

- The clock signal triggering the D flip flop should be slow enough to allow the switch to settle
- The clock should be fast enough to allow the asynchronous input to take effect as soon as possible
- Bouncing time: ms
- The D flip flop may be replaced by a double DFF synchronizer
 - thus both synchronizing and debouncing will be taken care of by one circuit



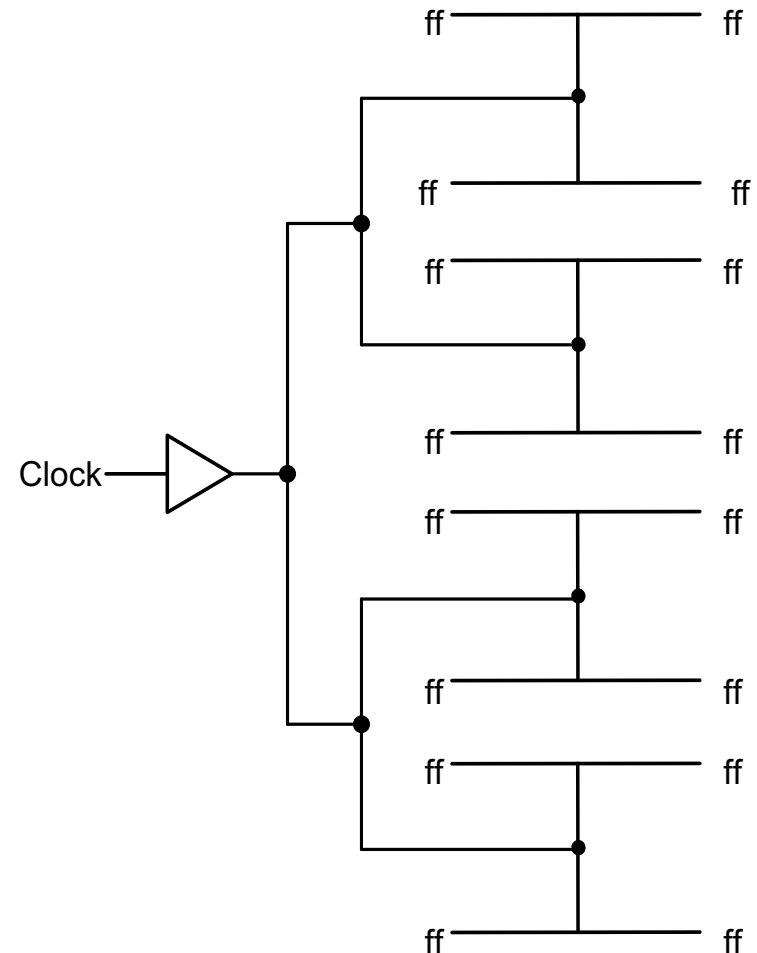
Clock Skew

- Example: add synchronous control to a D flip flop
 - Choice 1: control the input by mux
 - Choice 2: gated clock
- **Gated clock is not a good practice**
 - Flip flops that have gated clock will observe the clock change later than flip flops that have direct clock;
 - The situation in which the clock signal arrives at different flip flops at different times is known as **clock skew**
- Clock skew may cause synchronous circuit behaves asynchronously



Minimize Clock Skew

- Clock skew should be minimized as much as possible
 - H-Tree: carefully designed distribution network try to create the same length from each flip flop to the clock source
 - If same length may not be achieved, buffers are inserted on the short paths to even the delay
- Same for the global asynchronous reset



Glitches and Static Hazards

- The output of a combinational circuit may make a transition even though the patterns applied at its inputs do not imply a change. These unwanted transitions are called **glitches**.
- A circuit in which a glitch may occur is said to have a hazard.
- static 1-hazard

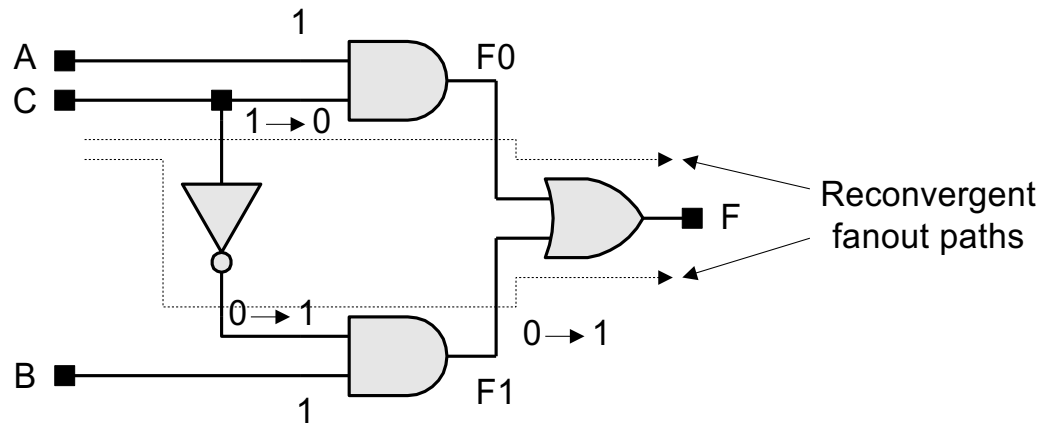


- static 0-hazard



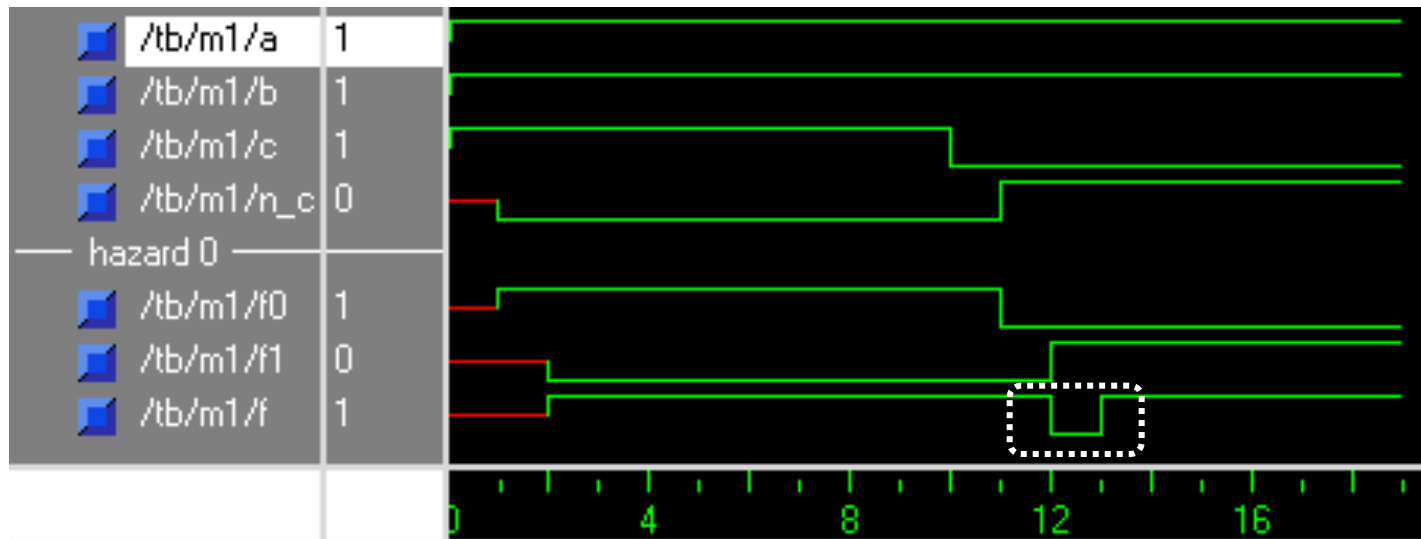
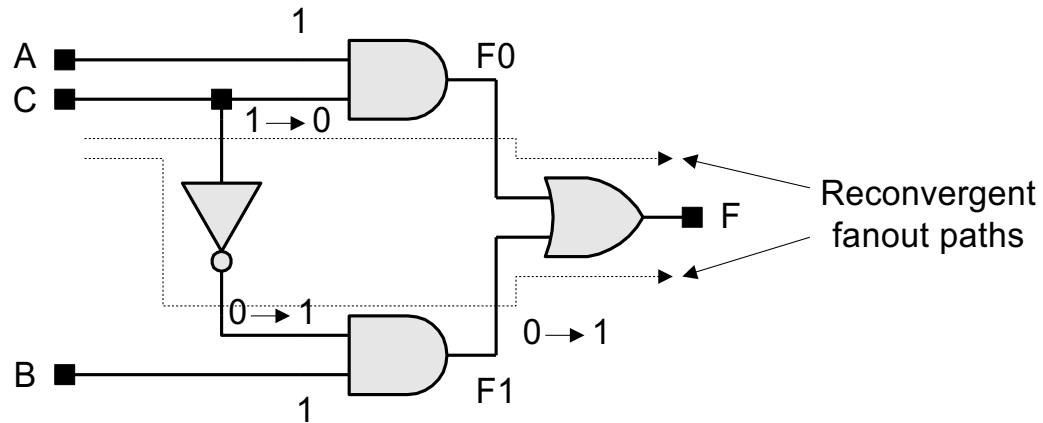
Static Hazards

- Static hazards are caused by different propagation delays on the **reconvergent fanout** paths
- Example: $F = AC + BC'$



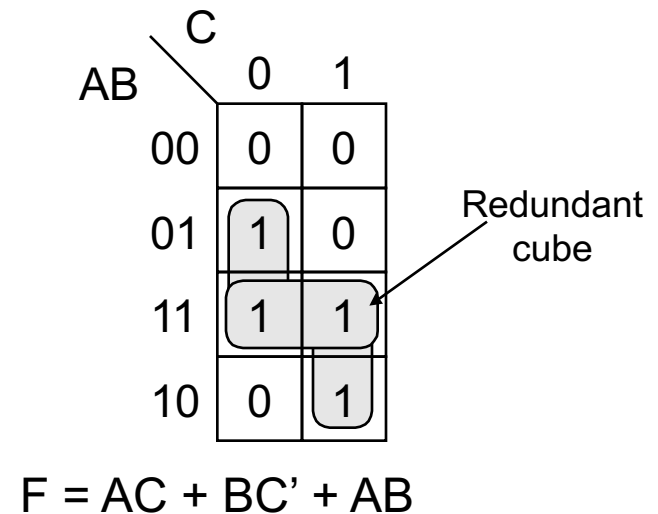
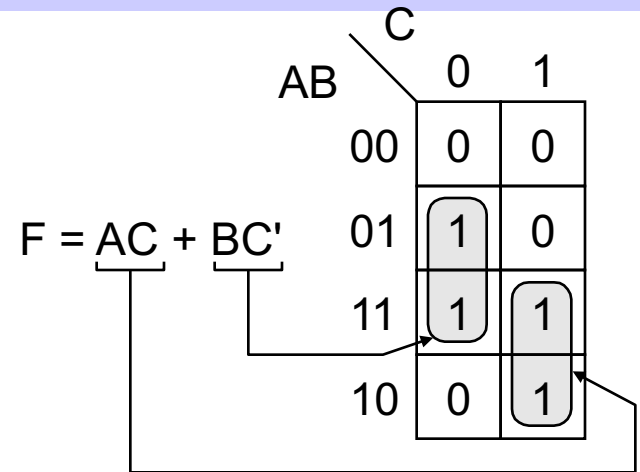
- Initial inputs: $A = 1, B = 1, C = 1$ and $F = 1$
- New inputs: $A = 1, B = 1, C = 0$ and $F = 1$
- With non-zero propagation delays, the path to F1 will be longer than the path to F0, causing a change in C to reach F1 later than it reaches F0.

Static Hazards Example



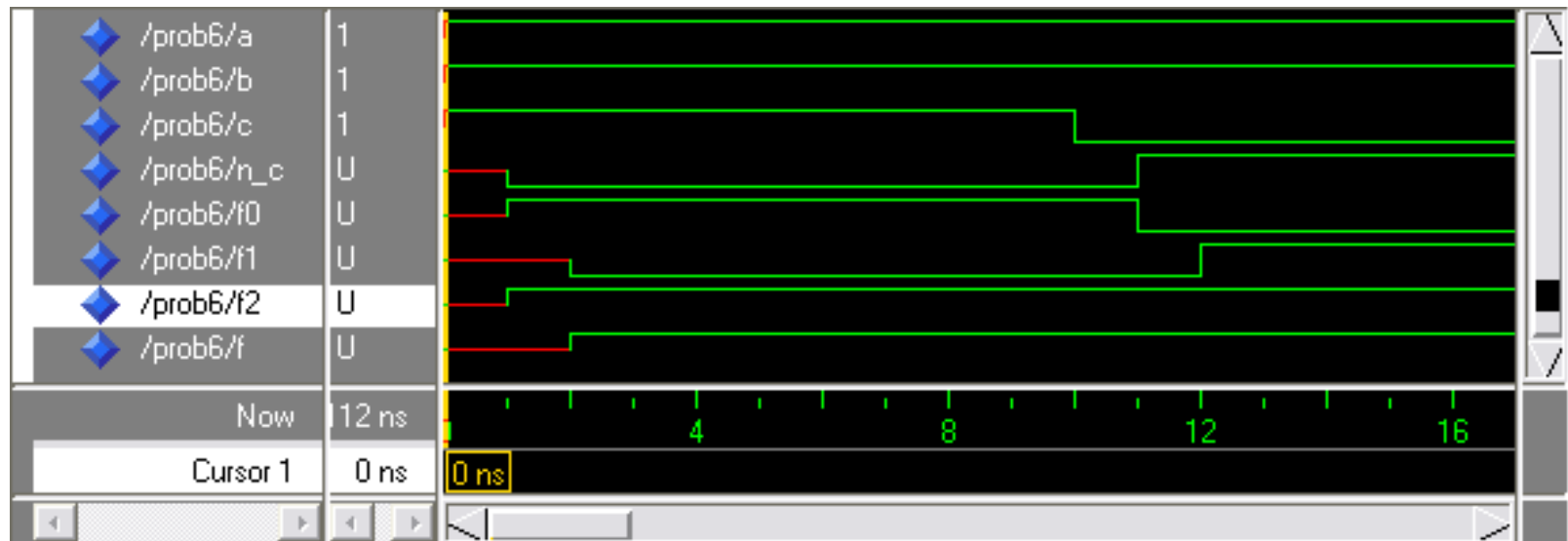
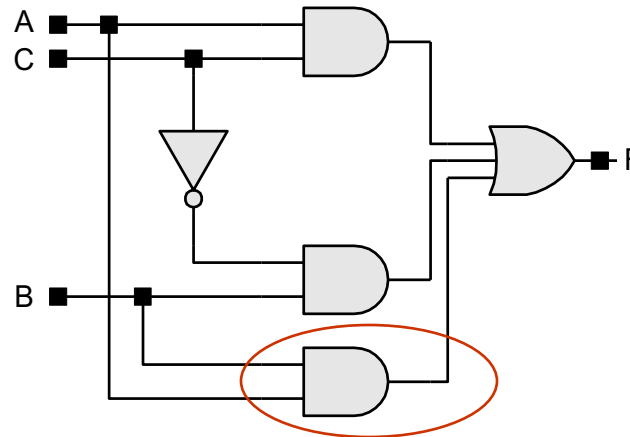
Static Hazards

- Hazards might not be significant in a synchronous sequential circuit if the clock period can be stretched
- A hazard is problematic in an asynchronous circuit (including Mealy FSM)
- If transition is within the same group in the K-map, hazard won't occur
 - will occur when transition from one group to another
- **Hazard Removal**
A static hazard can be removed by using a redundant PI



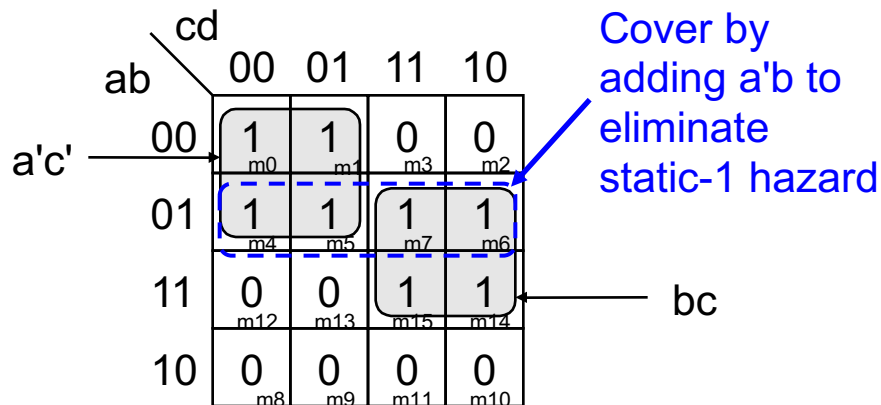
Static Hazards Example

Static 1-Hazard-Free circuit

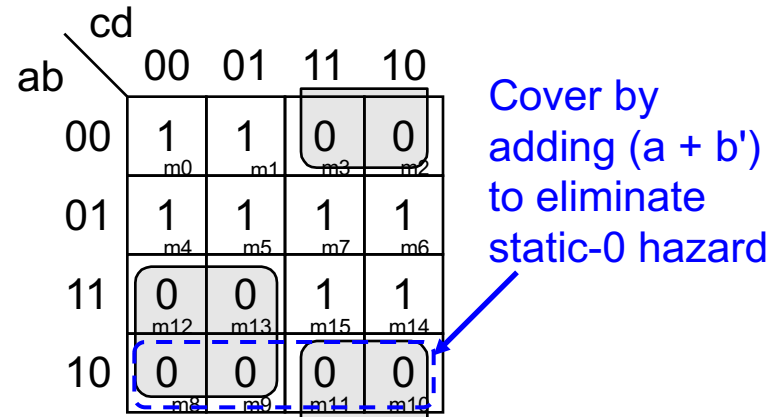


Static Hazards

- Static 0 hazard might not be removed in a static 1 hazard free circuit
- To eliminate static 0 hazard:
 - method 1: cover the adjacent 0s in the corresponding POS expression
 - method 2: first eliminate the static 1 hazards. Then verify in the K-map if 0 hazards have been eliminated already



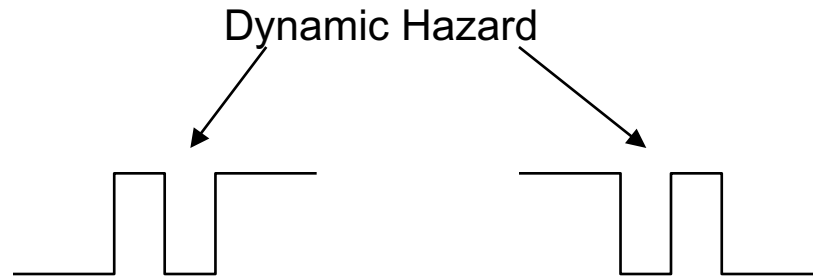
Eliminating static 1 hazard
 $F = a'c' + bc + a'b$



Eliminating static 0 hazard
 $F = (a' + c)(b + c')(a' + b)$
 $= a'c' + bc + a'b$

Dynamic Hazards

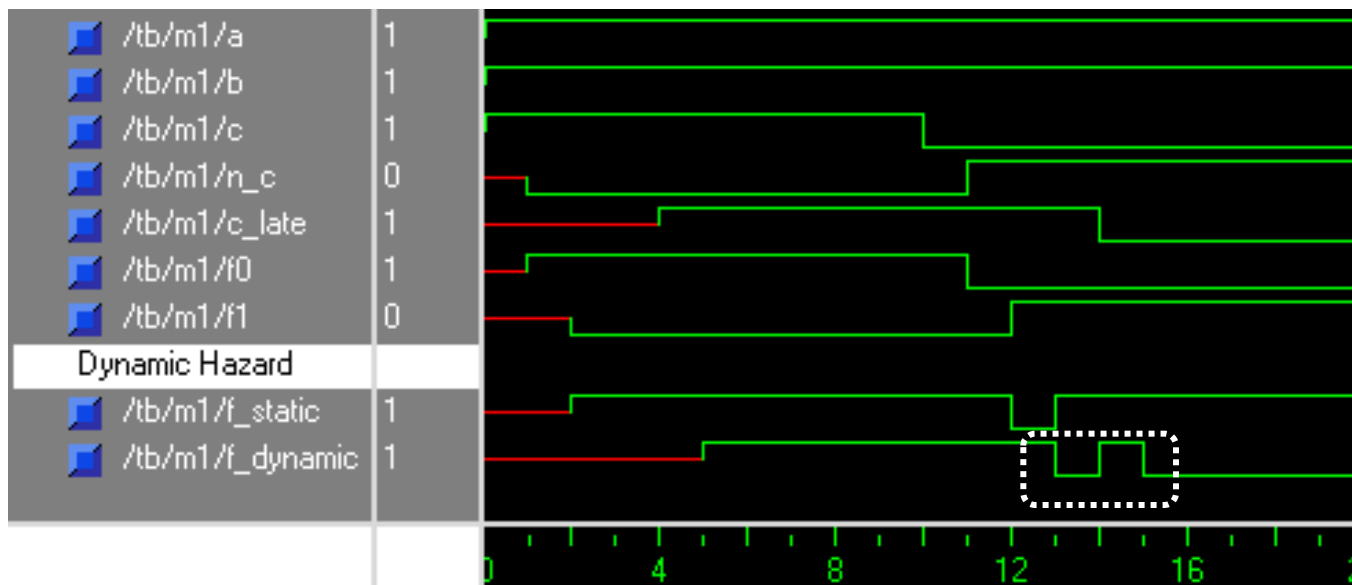
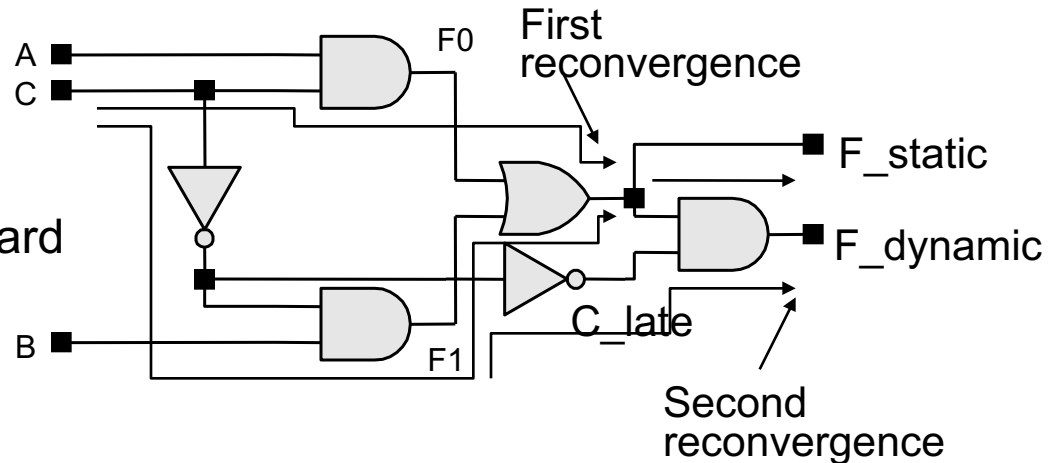
- Dynamic Hazard: output changes more than once as a result of a single output change



- Dynamic hazards are a consequence of multiple static hazards caused by multiple reconvergent paths in a multi-level circuit.
- **Hazard Removal:**
 - Elimination of all static hazards eliminates dynamic hazards.

Dynamic Hazards Example

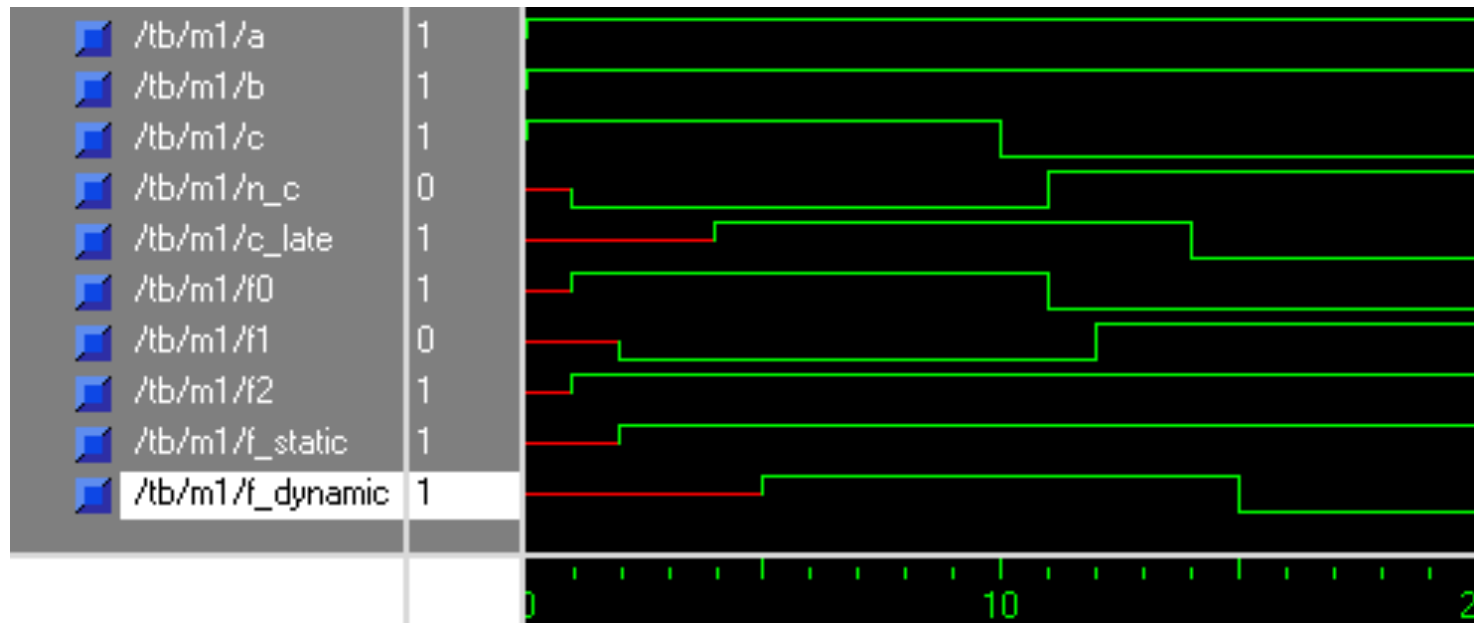
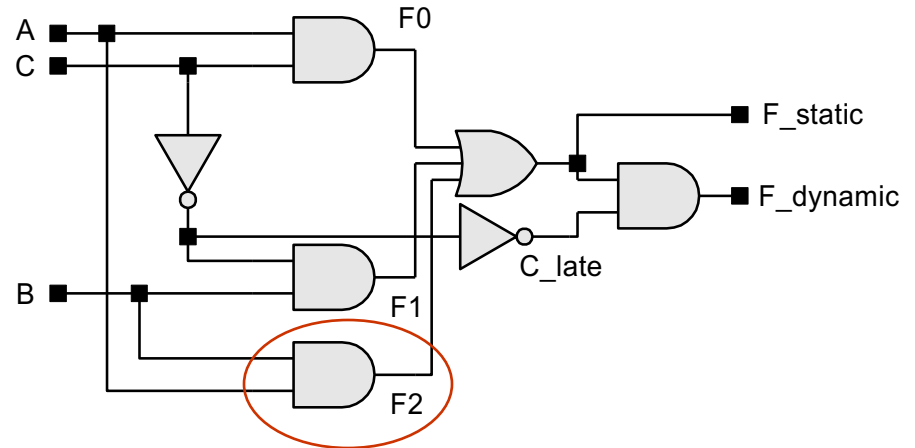
F_static has a static hazard
F_dynamic has a dynamic hazard



Dynamic Hazards Example

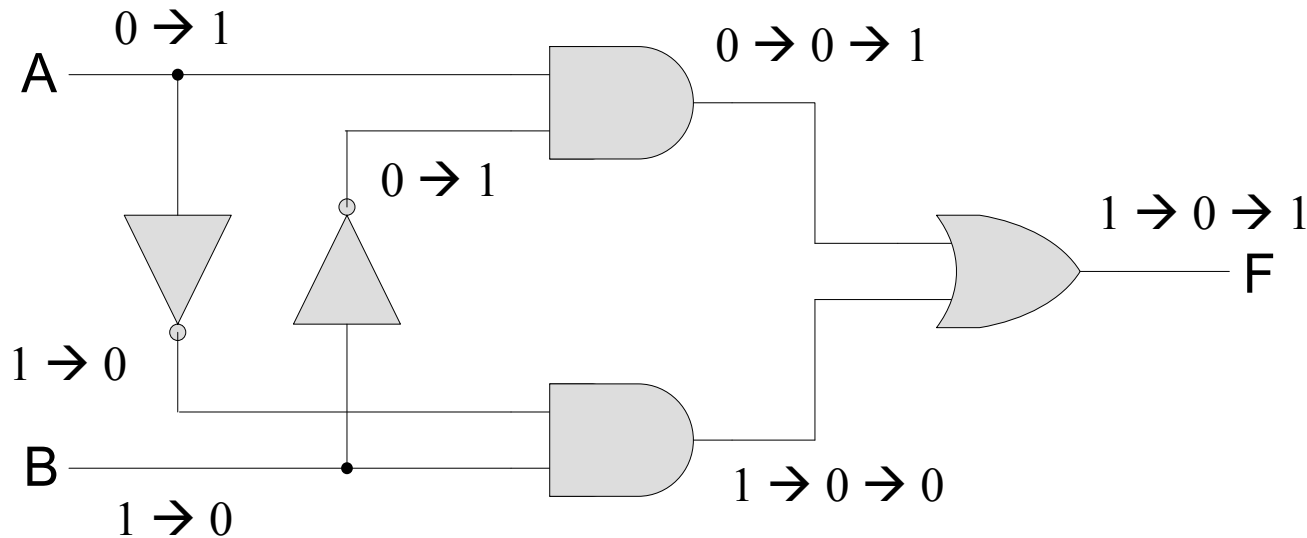
		C	
		0	1
AB	00	0	0
	01	1	0
	11	1	1
	10	0	1

Redundant group
Eliminates both
1-hazard and
0-hazard

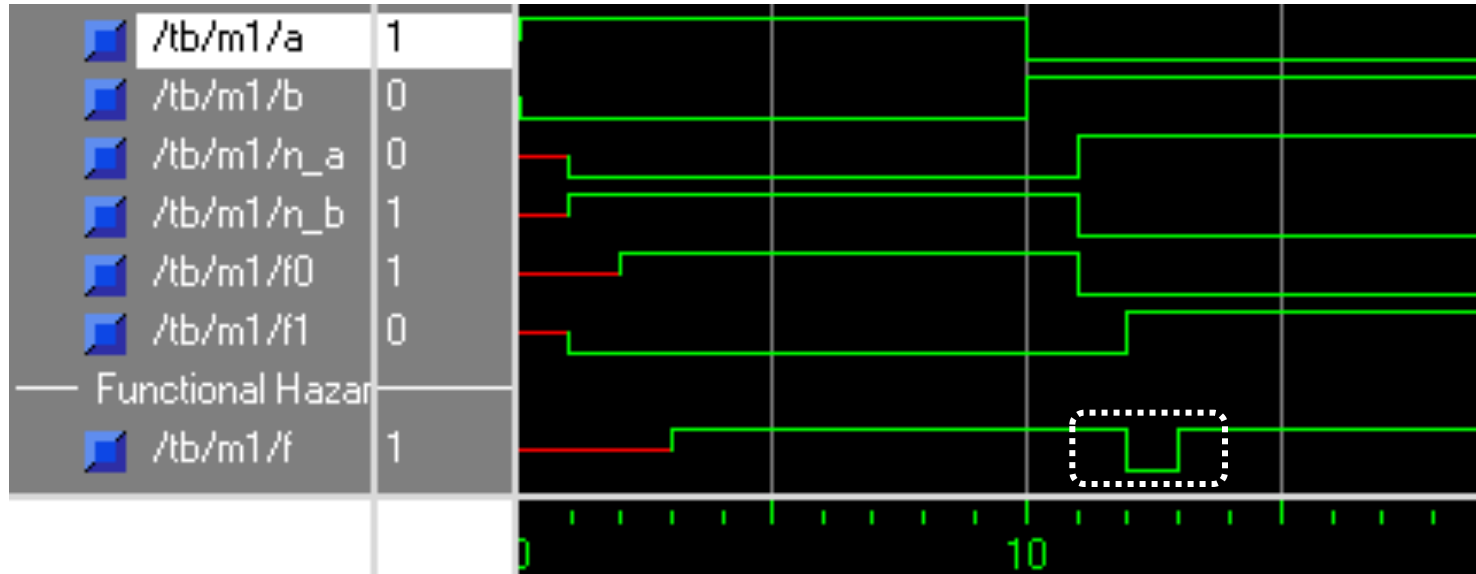


Functional Hazard

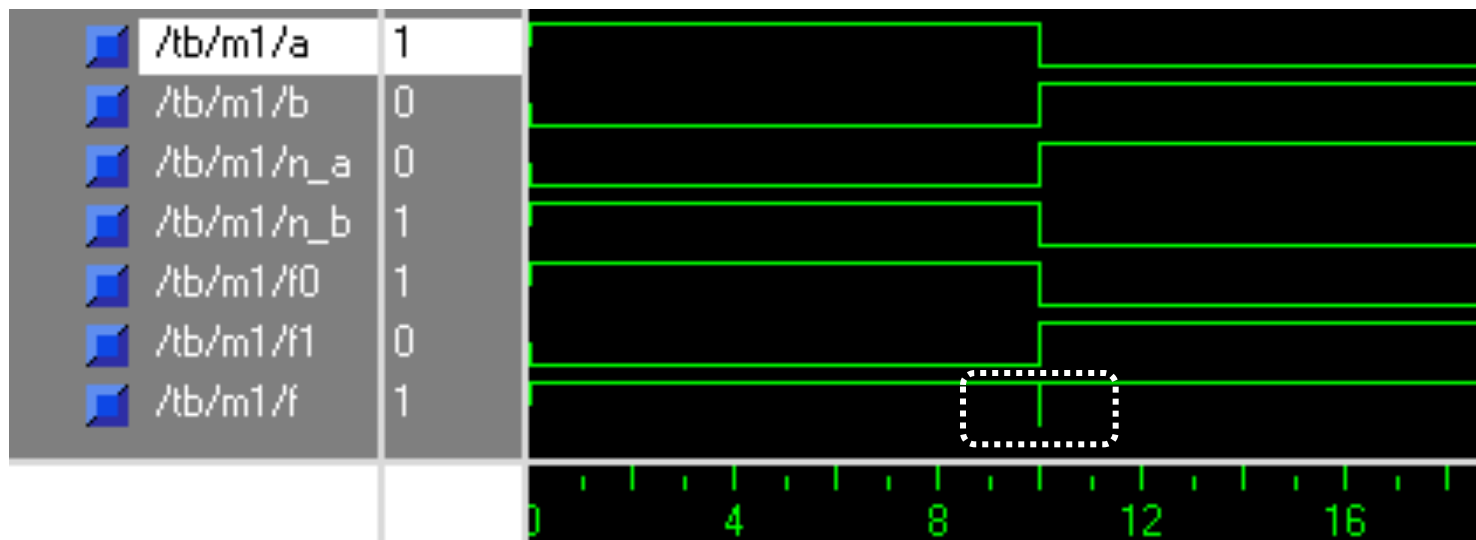
- Solutions to both static hazard and dynamic hazard are based on the assumption that only one input is changing at a time
- When more than one input changes occur at the same time, the hazard caused is more complicated to remove
- Example:



Functional Hazard



Simulation
with delay



Simulation
without delay.
Many sim
tools show
this to warn
for functional
hazard

Functional Hazard

- Functional hazards are unsolvable hazards which occur when more than one input variable changes at the same time
- Hazards such as functional hazards can not be logically eliminated as the problem lies with actual specification of the circuit
- K-map view of the functional hazard

