Software Engineering 2DA4

Slides 9: Asynchronous Sequential Circuits

Dr. Ryan Leduc

Department of Computing and Software
McMaster University

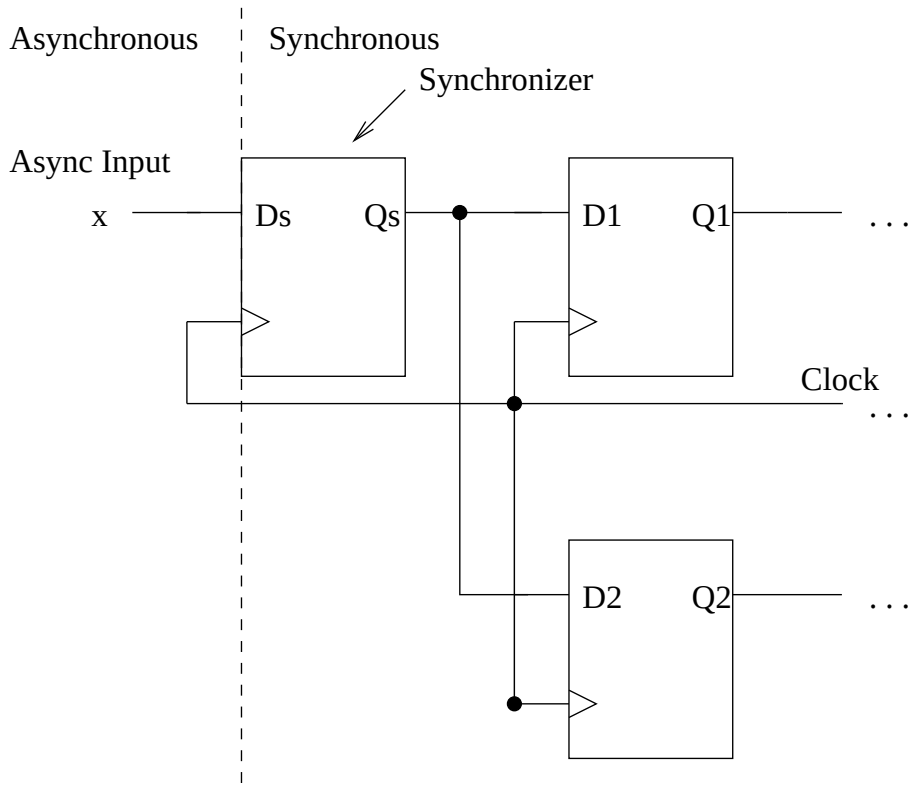
Material based on S. Brown and Z. Vranesic, *Fundamentals of Digital Logic with Verilog Design, 3rd Ed.*

Asynchronous Inputs

- Synchronous sequential circuits use flip-flops to store the current state.
- State changes occur on the positive or negative clock edge.
- For synchronous inputs, changes occur shortly after the active clock edge since inputs of one circuit are typically outputs of another synchronous circuit driven by same clock.
- Asynchronous inputs may change at any time. e.g. pushbutton inputs, outputs from a circuit driven by a different clock.
- Problems:
  a) Propagation delays can cause signals to be interpreted as different values in different parts of the circuit.
  b) Asynchronous inputs may violate the setup and hold times of flip flops.

Synchronizing Inputs

- Always put asynchronous inputs into a synchronizer.
- If the asynchronous input x was connected directly to D1 and D2, propagation delays could cause Q1 and Q2 to latch different values.



Metastability

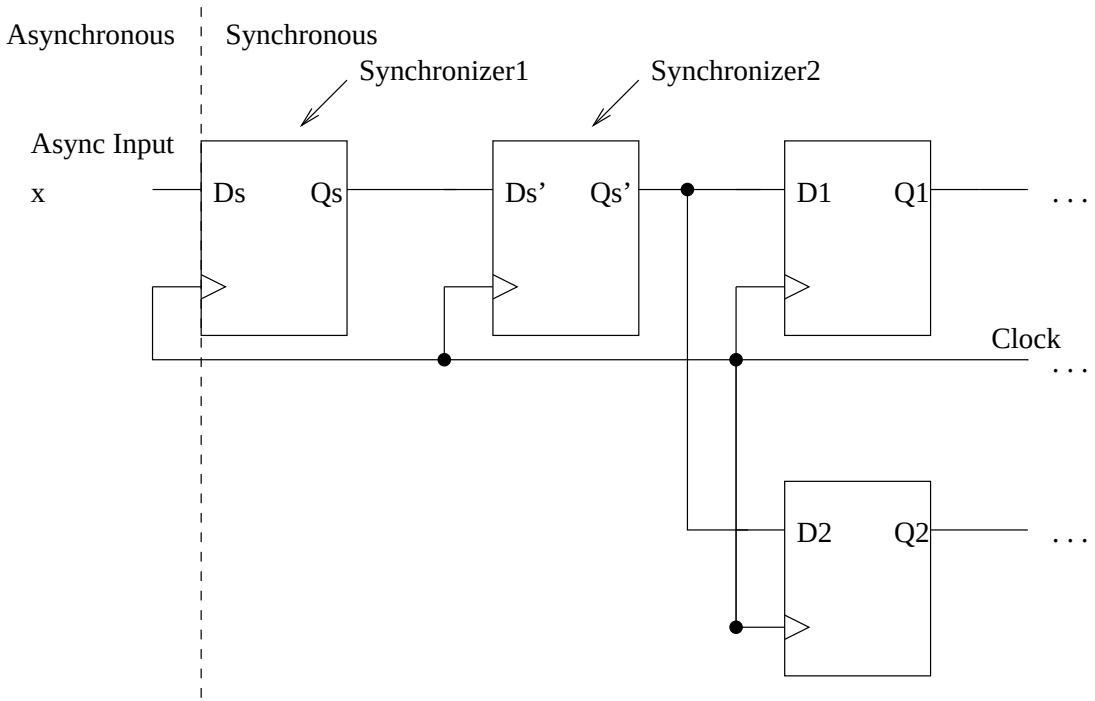
- **Q:** What happens when flip-flop setup and hold times are violated for synchronizer?
- **A:** Flip-flop may enter metastable state where logic value is neither 1 nor 0. Flip-flop will eventually resettle to either 1 or 0 after an indeterminate amount of time.
- If such a *synchronizer failure* occurs, the only guaranteed way to recover is to reset the entire circuit!
- While probability of synchronizer failure is usually small, it can never be eliminated. Good design though can reduce the probability of such failures.

Reducing Metastability Problems

- Synchronizer failure can be reduced by the following methods:
  - Using faster flip-flops with smaller setup and hold times (reduces size of vulnerable time window).
  - Lengthening the system clock period or, if possible, sampling the input at a lower frequency (reduces number of vulnerable time windows).

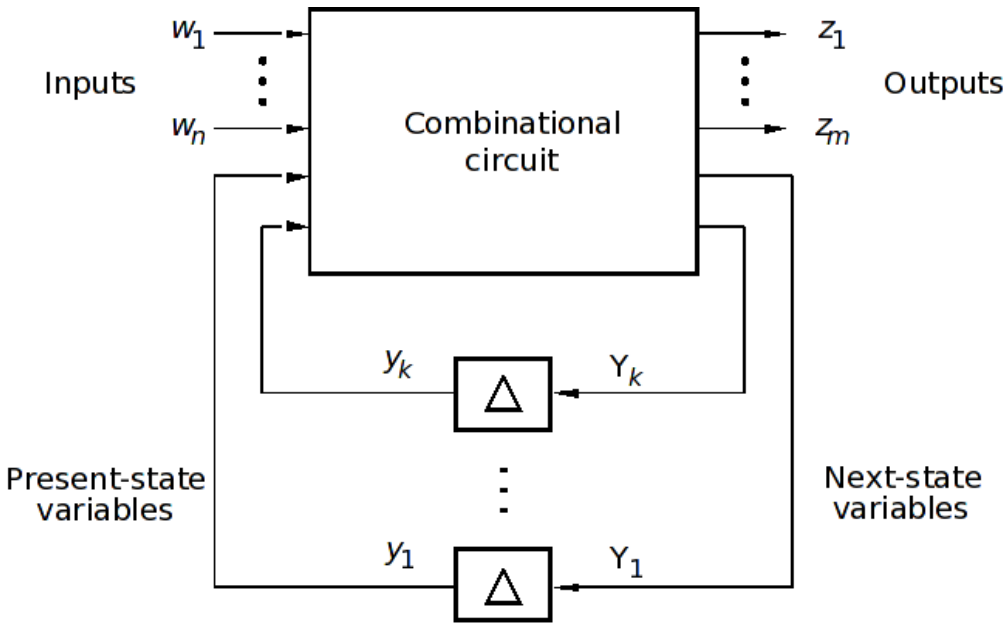
Reducing Metastability Problems - II

- Another common method uses a 2-bit shift register as synchronizer.
- If 1st synchronizer flip-flop enters metastable state, it will usually settle to a proper logic value before next active clock edge.
- This introduces a delay of one clock period on input.



Asynchronous vs. Synchronous Sequential Circuits

- For the general model of a sequential circuit, inputs and current state are used by combinational circuits to compute outputs and next state.
- After a time delay (say Δ) the next state becomes the current state.
  - For synchronous circuits, Δ = clock period
  - For asynchronous circuits, Δ = total propagation delay



Asynch vs. Synch Sequential Circuits - II

- Asynchronous circuits change when inputs change while synchronous circuits change on clock edge.
- Synchronous circuits assume that inputs do not change too close to active clock edge.
- Main assumptions for Asynchronous circuits:
  1. Only 1 input changes at a time
  2. Input changes occur sufficiently far apart to allow the circuit to reach a stable state before the next change.
- A stable state is a state the asynchronous circuits will stay in once reached, until another input changes.

Asynch vs. Synch Sequential Circuits - III

- Asynchronous advantages:
  - **Speed:** no clock involved; speed only depends upon propagation delays.
  - **Flexibility:** different parts of an asynchronous system can operate at different speeds (each limited by their propagation delay) while in synchronous systems, clock freqequency has to accommodate slowest part.
  - **Power usage:** distributing clock signal to all parts of an synchronous system adds to power usage, up to 30-40% for a high performance circuit.
- Asynchronous disadvantages:
  - **Design complexity:** difficult to design, and limited tool support.
  - **Glitches:** race conditions, "glitches" can cause problems if circuits not carefully designed.

Asynchronous Terminology

- For asynchronous circuits we use the following terminology:

  Flow Table: refers to state table.

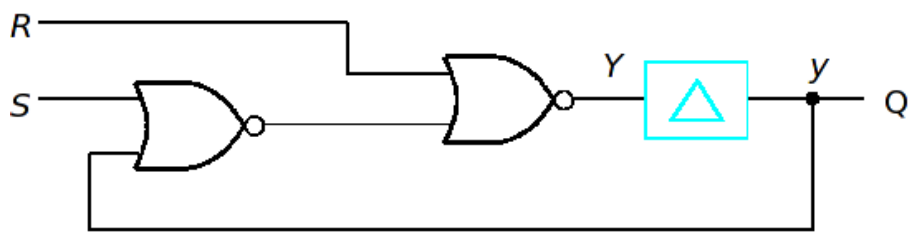
  Excitation Table: refers to state-assigned table.

- A state in a flow table is *stable* for particular set of inputs when the Next state = Current state.

- Otherwise the state is unstable and will change. We denote stable states in a flow table by circling them (e.g. (A)).

SR-latch as Asynchronous Sequential Circuit

- To obtain state variables, we use the natural gate delay of circuit.
- We cut the feedback loop and insert a delay element.
- Creates a delay of time $\triangle$, equal to the combined propagation delay of the two NOR gates.
- We then treat the NOR gates as ideal gates with zero delay.
- We take y (output of delay element) to be the present state, and Y to be our next state variable (input to delay element).
- After time delay $\triangle$, y is assigned the value of Y.



(b) Circuit with modeled gate delay

SR-latch as Asynch Sequential Circuit - II

- We take Q to be our output variable, where $Q = y$.
- We now derive an equation for Y in terms of S, R. and y.
- Taking point A as output of leftmost NOR gate, we have:

$$A = \overline{(S + y)}$$

- Thus: $Y = \overline{R + \overline{(S + y)}} = \overline{R} \cdot (S + y)$
- Using this equation, we can construct the excitation table below.

Present state y	Next state SR = 00	01	10	11
	Y	Y	Y	Y
0	(0)	(0)	1	(0)
1	(1)	0	(1)	0

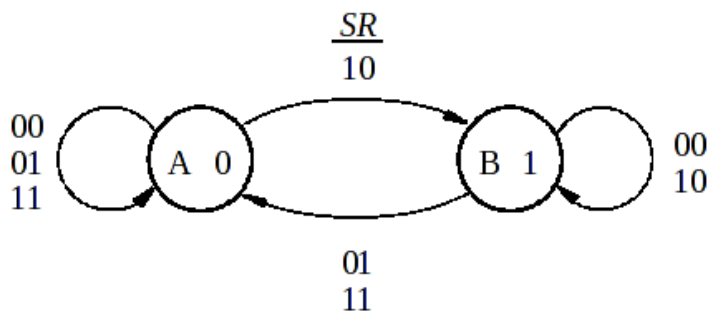
(b) Excitation table

SR-latch as Asynch Sequential Circuit - III

- If we take $y = 0$ to be state A, and $y = 1$ to be state B, we can convert the excitation table to the flow diagram below.
- From the flow table, we can easily derive the state diagram below.
- Note that asynchronous circuits have no RESET state.
- Their initial state is random.

Present state	Next state				Output
	SR = 00	01	10	11	
A	(A)	(A)	B	(A)	0
B	(B)	A	(B)	A	1

(a) Flow



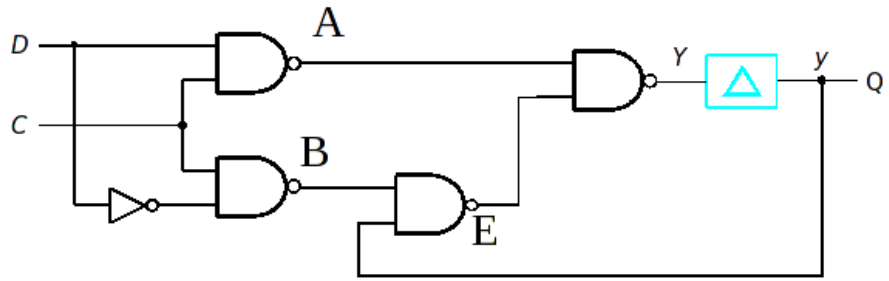
(b) State diagram

Analysis of Asynchronous Sequential Circuits

- An asynchronous sequential circuit can be analyzed by the following steps:
  1. Cut each feedback path and insert a delay.
  2. Determine Next State and Output expressions from circuit.
  3. Derive excitation table (async state assignment table) from Next State and Output equations.
  4. Obtain a flow table (async state table) by assigning state labels to each of the state encodings.
  5. Draw FSM from flow table.

Analysis of Gated D Latch

- Derive equations for output Q and next state Y for gated D latch and use to derive excitation, flow and state diagram.



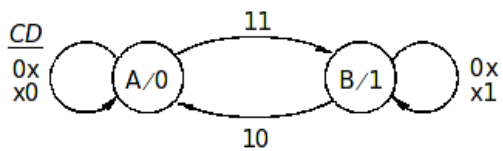
(a) Circuit

Present state y	Next state CD = 00	01	10	11	
	Y	Y	Y	Y	
0	(0)	(0)	(0)	1	0
1	(1)	(1)	0	(1)	1

(b) Excitation table

Present state	Next state CD = 00	01	10	11	Q
A	(A)	(A)	(A)	B	0
B	(B)	(B)	A	(B)	1

(c) Flow table



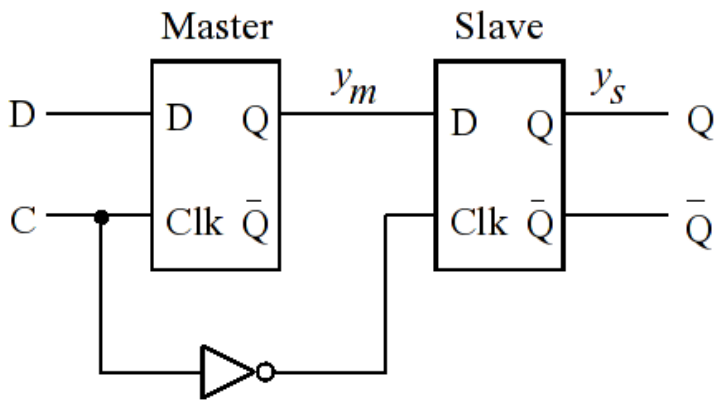
(d) State diagram

Analysis of Master-Slave D Flip-flop

- Derive equations for output Q and next state variables Y_m and Y_s for master-slave D flip-flop, and use them to derive excitation, flow and state diagram.

- We will use y_m and y_s as our current state variables, and use the next state equations from gated D latch as our starting point (see discussion in class).

- NOTE: input C is the clock in a synchronous setting, but here it is just another input.



Analysis of Master-Slave D Flip-flop - II

- Using derived equations, we can fill in the tables below.

Present state $y_m\ y_s$	Next state $Y_m\ Y_s$ for CD = 00	01	10	11	Output Q
00	(00)	(00)	(00)	10	0
01	00	00	(01)	11	1
10	11	11	00	(10)	0
11	(11)	(11)	01	(11)	1

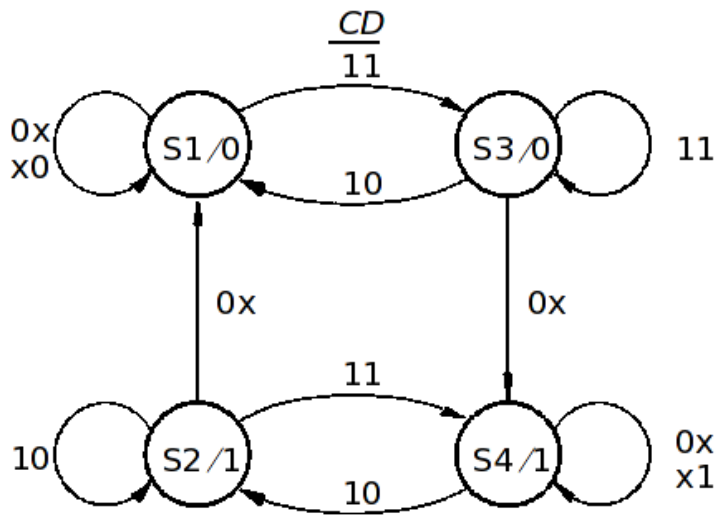
(a) Excitationtable

Present state	Next state for CD = 00	01	10	11	Output Q
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	S1	(S2)	S4	1
S3	S4	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

(b) Flowtable

Analysis of Master-Slave D Flip-flop - III

- Using flow table, we can construct state diagram below.
- Read example 9.3 in text on own.



Unspecified Entries

- For synthesis, can simplify by adding unspecified states when a particular input combination can't occur at a given state.
- Consider present state S2, and input combination $CD = 01$.
- This combination will never occur in normal operation, as S2 is only stable for $CD = 10$ and $CD = 01$ requires two input changes, both of which are not stable.
- For i.e., if we have path $CD = 10$ to $CD = 00$, this would take us to state S1, before we got change $CD = 01$ both stable for S1.

Present state	Next state CD = 00	01	10	11	Output Q
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	S1	(S2)	S4	1
S3	S4	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

(b) Flow table

Present state	Next state CD = 00	01	10	11	Output Q
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	–	(S2)	S4	1
S3	–	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

(c) Flow Table with unspecified entries

Synthesis of Asynchronous Sequential Circuits

1. Derive state diagram.

2. Derive flow table (reduce states: we won't cover this).

3. Perform state assignments and derive excitation table.

   Ensure state variables do not contain *race conditions.*

4. Obtain next state and output expressions and ensure that they do not contain hazards (Section 9.6).

5. Construct circuit that implements these expressions.

Asynchronous Design Example: Serial Parity Generator

- **Problem:** Design an asynchronous sequential circuit that implements a FSM that acts as an even serial parity generator.
- Circuit receives series of pulses on input w.
- When odd number of pulses have been received, output z is 1.
- When even number of pulses received, output z is 0.
- NOTE: in an ansynchronous circuit, we have no timing information.
- To detect a pulse, we need to detect when input goes from 0 to 1 (start of pulse), and then when it goes from 1 to 0 (end of pulse).
- See design of state diagram in class on board.

Serial Parity Generator: tables

- From state diagram on board, we can write out flow diagram below.
- We will see on next slide why first state assignment is bad.

Present State	Next state $w = 0$	Next state $w = 1$	Output z
A	(A)	B	0
B	C	(B)	1
C	(C)	D	1
D	A	(D)	0

(b) Flow table

Present state y_2y_1	Next state $w = 0$, Y_2Y_1	Next state $w = 1$, Y_2Y_1	Output z
00	(00)	01	0
01	10	(01)	1
10	(10)	11	1
11	00	(11)	0

(a) Poor state assignment

Present state y_2y_1	Next state $w = 0$, Y_2Y_1	Next state $w = 1$, Y_2Y_1	Output z
00	(00)	01	0
01	11	(01)	1
11	(11)	10	1
10	00	(10)	0

(b) Good state assignment

State Assignments for Asynchronous FSMs

- Figure below contains two different possible state assignments for serial parity FSM.
- **Problem:** table (a) requires transition

$$y_2 y_1 = 11 \xrightarrow{w=0} y_2 y_1 = 00$$

- This requires y_1 and y_2 to change at exactly the same time!
- Since the circuit is not ideal, y_1 and y_2 will not change at the same time.

Present state $y_2 y_1$	Next state $Y_2 Y_1$, $w = 0$	Next state $Y_2 Y_1$, $w = 1$	Output z
00	(00)	01	0
01	10	(01)	1
10	(10)	11	1
11	00	(11)	0

(a) Poor state assignment

Present state $y_2 y_1$	Next state $Y_2 Y_1$, $w = 0$	Next state $Y_2 Y_1$, $w = 1$	Output z
00	(00)	01	0
01	11	(01)	1
11	(11)	10	1
10	00	(10)	0

(b) Good state assignment

State Assignments for Asynchronous FSMs - II

- **Case 1:** y_1 changes 1st. Circuit changes to $y_2y_1 = 10$ corresponding to state C and then stays in C producing wrong output.
- **Case 2:** y_2 changes first. Circuit changes to $y_2y_1 = 01 =$ state B. Then tries to change to state $C = 10$. This requires $01 \rightarrow 10$, another simultaneous change. This means that y_2 must change again.
- Since we are assuming y_2 changed first, y_1 should complete its change to 0 first (before y_2 can change again), bringing the state to 00.

Present state y_2y_1	Next state Y_2Y_1 $w = 0$	Next state Y_2Y_1 $w = 1$	Output z
00	(00)	01	0
01	10	(01)	1
10	(10)	11	1
11	00	(11)	0

(a) Poor state assignment

Present state y_2y_1	Next state Y_2Y_1 $w = 0$	Next state Y_2Y_1 $w = 1$	Output z
00	(00)	01	0
01	11	(01)	1
11	(11)	10	1
10	00	(10)	0

(b) Good state assignment

State Assignments for Asynchronous FSMs - III

- As state $00 = A$ is correct and stable for $w = 0$, we arrive at the correct answer.
- The correct outcome depends on which variable changes first. This is referred to as a race condition.

Present state y_2y_1	Next state Y_2Y_1, $w = 0$	Next state Y_2Y_1, $w = 1$	Output z
00	(00)	01	0
01	10	(01)	1
10	(10)	11	1
11	00	(11)	0

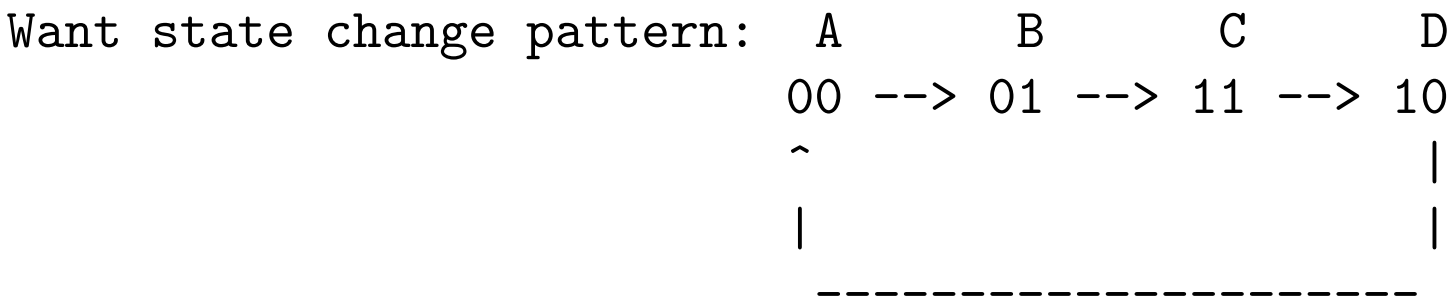
(a) Poor state assignment

Present state y_2y_1	Next state Y_2Y_1, $w = 0$	Next state Y_2Y_1, $w = 1$	Output z
00	(00)	01	0
01	11	(01)	1
11	(11)	10	1
10	00	(10)	0

(b) Good state assignment

Eliminating Race Conditions

- Treat state variables like inputs to circuit:
- Only allow one variable to change at a time (grey code!).



- Table (b) uses this state assignment.

Present state y_2y_1	Next state Y_2Y_1 $w = 0$	Next state Y_2Y_1 $w = 1$	Output z
00	(00)	01	0
01	10	(01)	1
10	(10)	11	1
11	00	(11)	0

(a) Poor state assignment

Present state y_2y_1	Next state Y_2Y_1 $w = 0$	Next state Y_2Y_1 $w = 1$	Output z
00	(00)	01	0
01	11	(01)	1
11	(11)	10	1
10	00	(10)	0

(b) Good state assignment

Implementing FSM

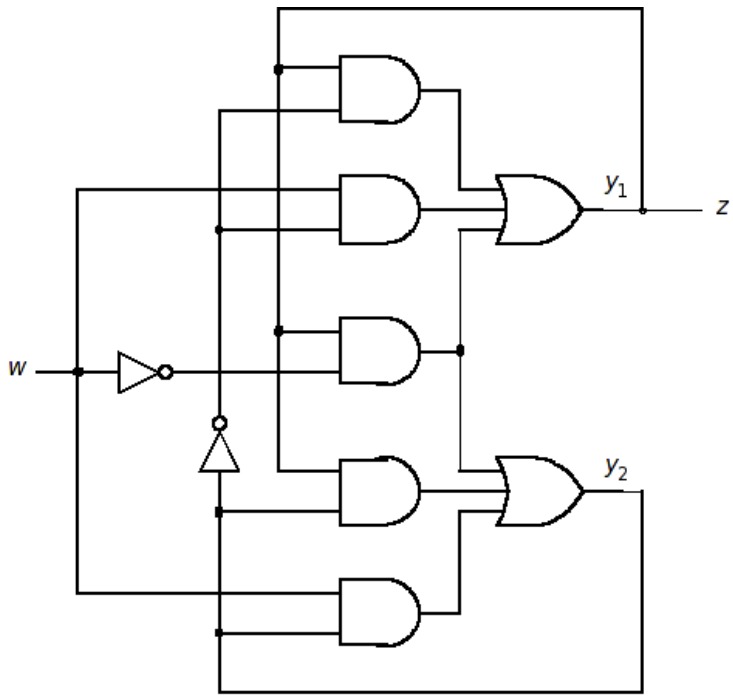
- From excitation table (b), we can derive our output and next state equations.
- For our output, we have: $z = y_1$.
- For our next state equations we have:

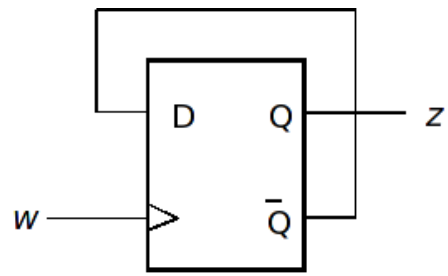
$$Y_1 = w\overline{y_2} + \overline{w}y_1 \quad + \quad y_1\overline{y_2}$$

$$Y_2 = \underbrace{wy_2 + \overline{w}y_1}_{k-maps} \quad + \quad \underbrace{y_1y_2}_{\text{stop hazards}}$$

Implementing FSM - II

- Below is circuit for FSM.
- On the right is the synchronous equivalent.





Hazards

- In an asynchronous sequential circuit, want to avoid *glitches* on signals.
- A glitch is when a signal temporary takes on the wrong value:

```
        _
_______| |_______   should always be zero.
```

- Glitches caused by structure of circuit and propagation delays are called hazards.

Types of Hazards

- There are two types:

Static Hazards: When signal is not suppose to change its value in response to a specific change in an input, but instead momentarily does change.

Dynamic Hazards: This is when a signal is suppose to change value, but there is a small oscillation.

(a) Static hazard

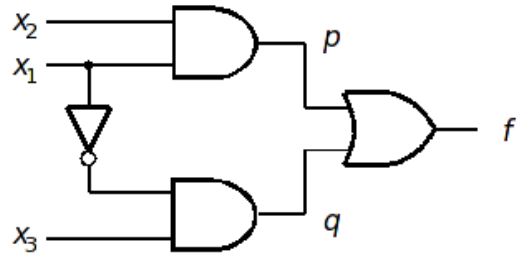
(b) Dynamic hazard

Static Hazards

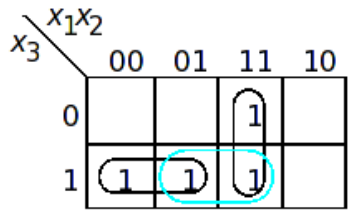
- A change to a primary input often has more than one path of propagation to an output.
- When one path has a longer propagation delay than the others, we may find a static hazard.
- This can be eliminated by examining the k-map of the output.
- A potential hazard exists whenever two ajacent ones (or 0's if we are doing a product-of-sum implementation) are not covered by a common product term (sum term for product-of-sum).
- To guarantee no static hazards, obtain a cover such that each pair of adjacent one's (zero's) is covered by a common product term (sum term).

Static Hazard Example

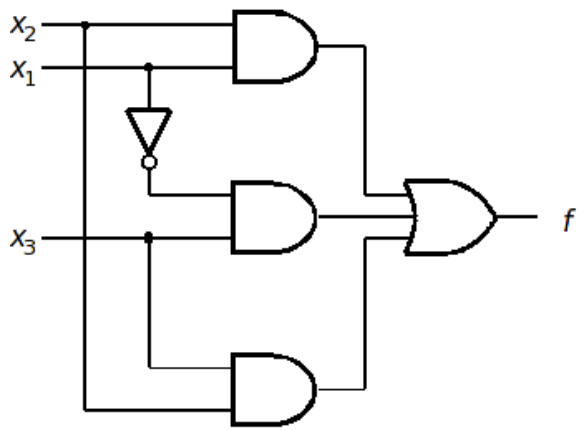
- Top circuit is the minimal k-map version and it contains a static hazard as shown on next slide.
- Minimal circuit (black groupings) have two adjacent 1 terms (x_1x_2 goes from 01 to 11), thus could have static hazard.
- In k-map, the blue grouping is added to ensure no hazards.
- Bottom circuit contains all three terms, thus no static hazards.



(a) Circuit with a hazard



(b) Karnaugh map

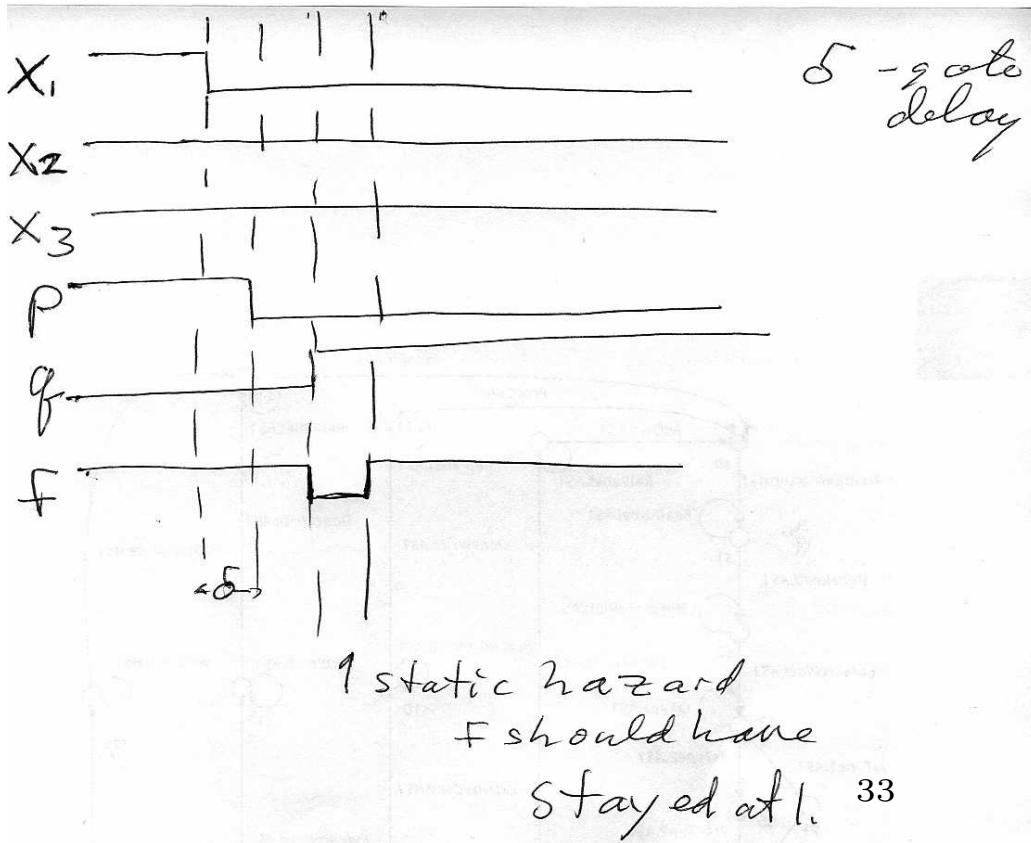


(c) Hazard-free circuit

Static Hazard Example - II

- Using a k-map, we can see if a circuit might have static hazards, and how to ensure they don't.
- The kmap also indicates where to look for a static hazard (input combinations that go between two adjacent one terms that are not covered by a common product term.)

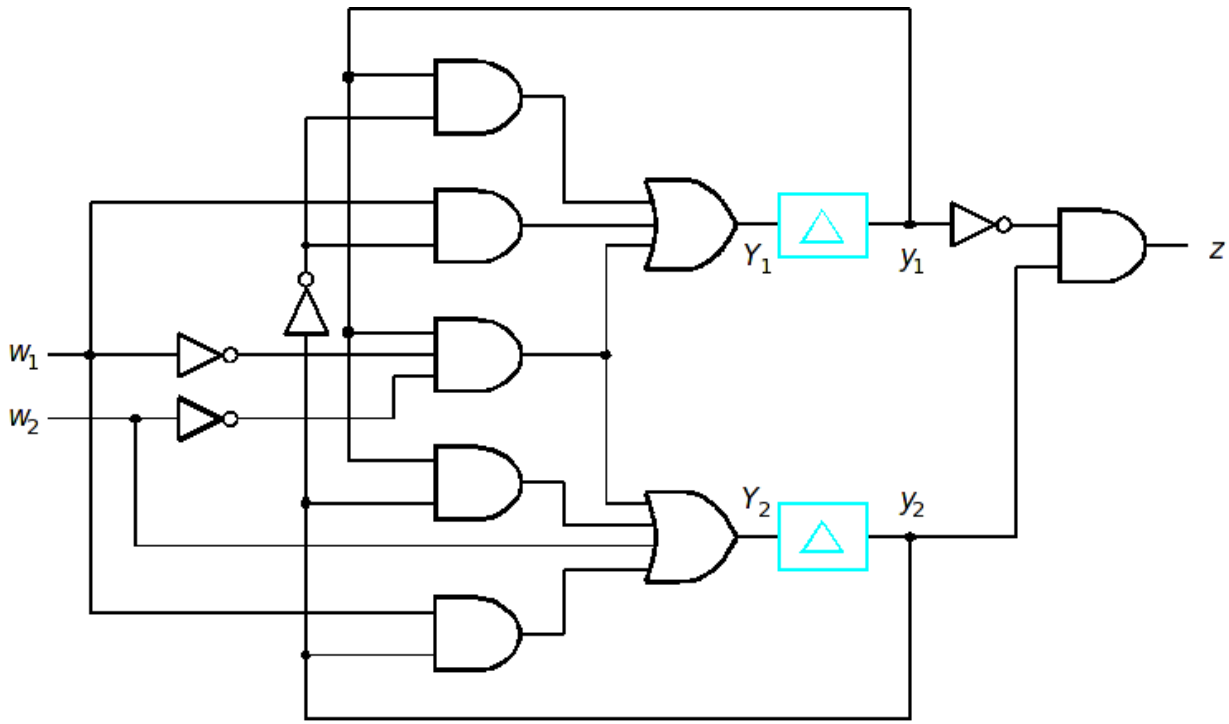
- For the hazard to exist, we need two paths with different propagation delays.
- We can use a timing diagram to show the existence of a hazard as below.



Checking State Variables for Static Hazards

- This means checking the next state variables (ie. Y_1, not y_1).
  - Ignore ouput logic.
  - Ignore feedback path. Treat present state variables as "just another input" to your circuit.
  - Derive equations for each next state variable and write down its k-map.
  - Analyze each k-map for potential static hazards.

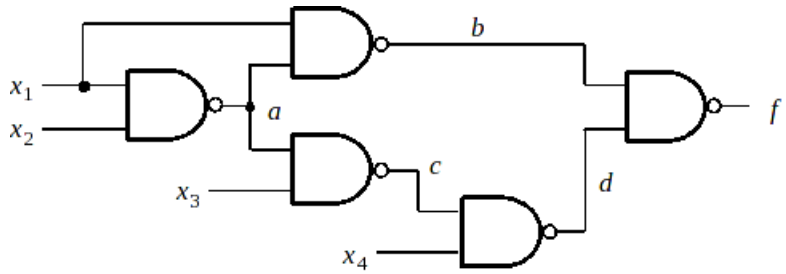
Checking State Variables for Static Hazards - II



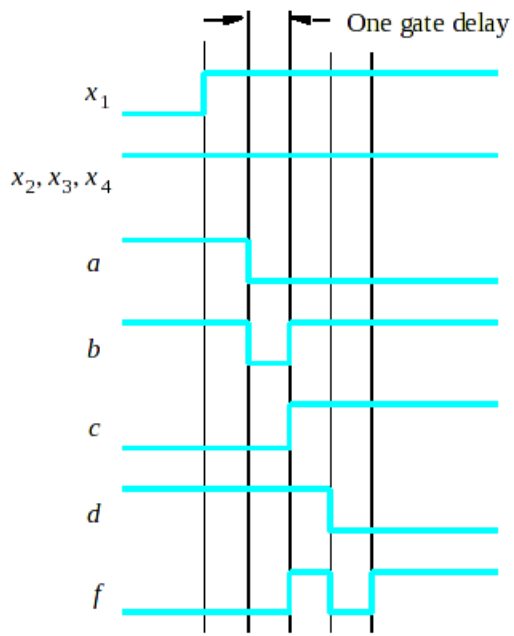
Dynamic Hazards

- Figure shows an example of a dynamic hazard.
- A dynamic hazard is caused by the structure of a circuit.

- It's caused by a circuit with more than two levels, in which changes to an input have more than one path to propagate along.
- In Figure, there are three paths.
- A circuit with a dynamic hazard must also contain a static hazard. The figure has a static hazard at point b.
- To avoid dynamic hazards, design two-level circuits with no static hazards.



(a) Circuit



(b) Timing diagram