Asynchronous Sequential Circuits

- Sequential circuits that are not synchronized by a clock – Asynchronous circuits
- Analysis of Asynchronous circuits
- Synthesis of Asynchronous circuits
- Hazards that cause incorrect behavior of a circuit

ASYNCHRONOUS SEQUENTIAL CIRCUITS

- Synchronous sequential circuits
  - state variables : F/Fs
  - controlled by a clock
  - operate in pulse mode
- Asynchronous sequential circuits
  - do not operate in synchronous with *clock signal.*
  - do not use F/Fs to represent state variables
  - Changes in state are dependent on whether each of inputs to the circuit has the logic level 0 or 1 at any given time
- To achieve reliable operation
  - the inputs to the circuit must change one at a time
  - there must be sufficient time between the changes in input signals to allow the circuit to reach a stable state
  - A circuit that adheres to these constraints is said to operate in the *fundamental* mode

Advantages of Asynchronous Circuits

- No clock skew (clock signal arrives at different time)
- Lower power (Synchronous : clock signal must be present every time and everywhere.
- Average-case performance VS worst-case performance
- Easing of global timing issues
- Partial optimization
- Better external input handling

Drawbacks

- More difficult to design
- Concerns for hazards and glitches
- Unsure about faster performance

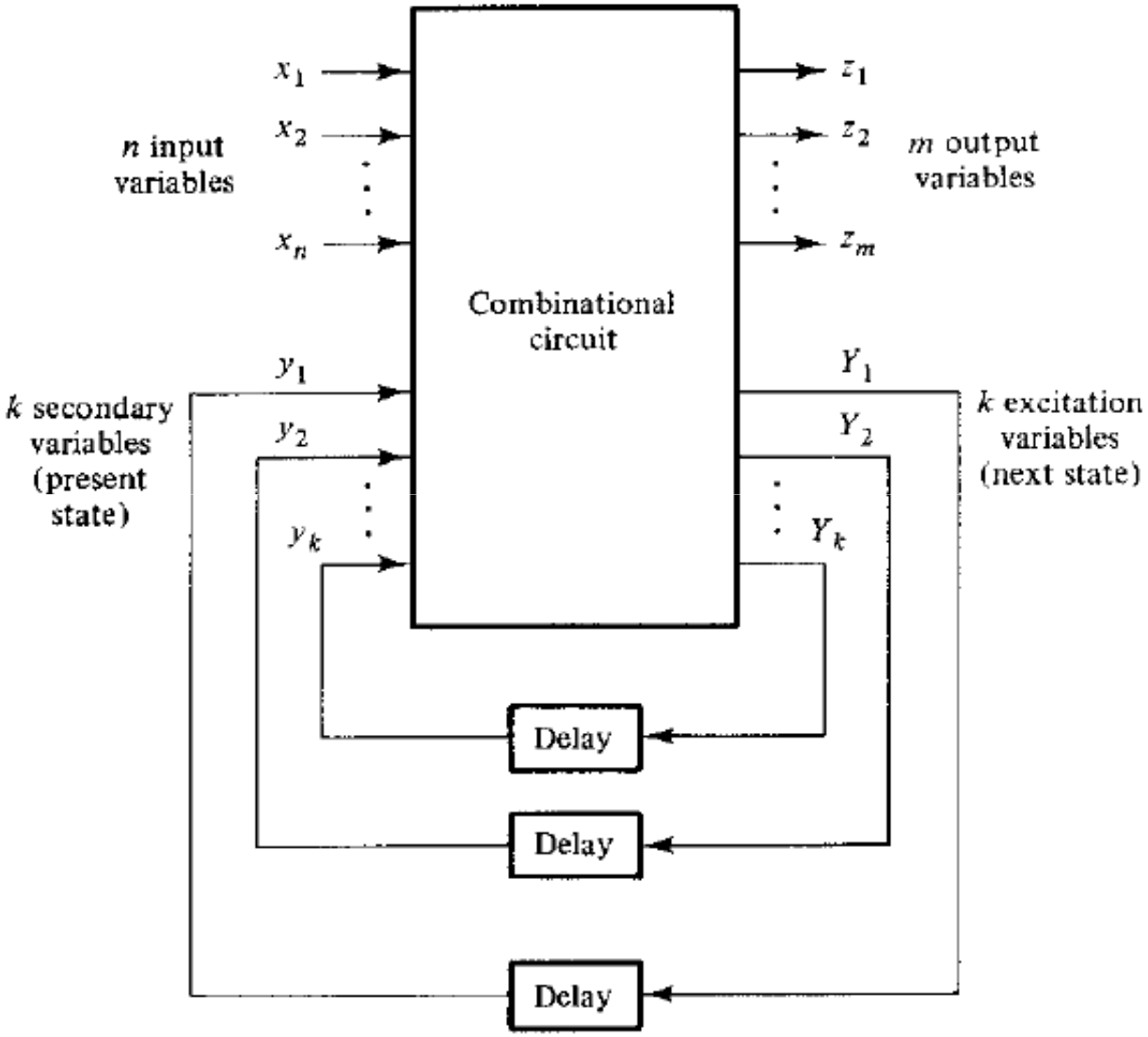
WHY ASYNCHRONOUS CIRCUITS ?

- Used when speed of operation is important
  - Response quickly without waiting for a clock pulse
- Used in small independent systems
  - Only a few components are required
- Used when the input signals may change independently of internal clock
  - Asynchronous in nature
- Used in the communication between two units that have their own independent clocks
  - Must be done in an asynchronous fashion

Mode of Operations

- Steady-state condition:
  - Current states and next states are the same
  - Difference between Y and y will cause a transition
- Fundamental mode:
  - No simultaneous changes of two or more variables
  - The time between two input changes must be longer than the time it takes the circuit to a stable state
  - The input signals change one at a time and only when the circuit is in a stable condition Fundamental Mode
- Pulse Mode:
  - the inputs and outputs are represented by pulses.
  - only one input is allowed to have pulse present at any time.
  - Similar to synchronous sequential circuits except without a clock signal.

General Block Diagram



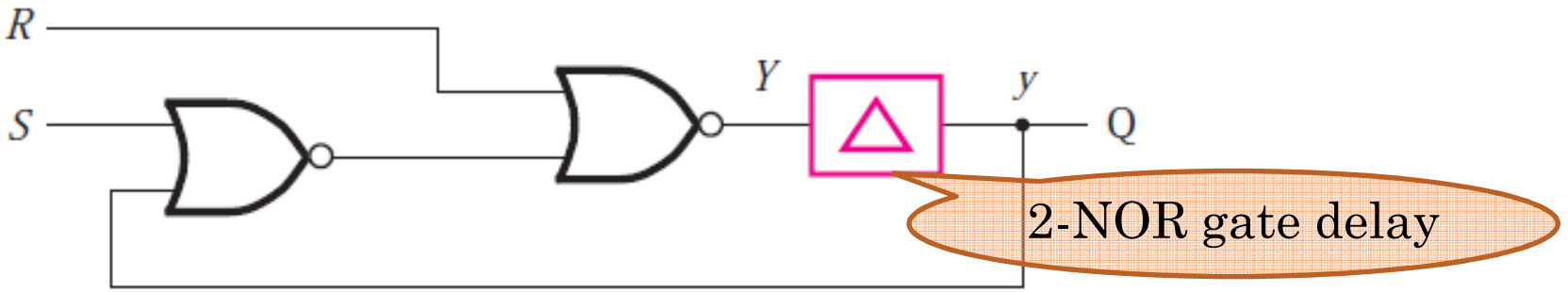
TERMINOLOGY

- Asynchronous circuits
  - state table -> flow table
  - state-assigned table -> transition table or excitation table
- will use the term flow table and excitation table

Steps in The Analysis Process

- Each feedback path is cut
  - A delay element is inserted at the point where the cut is made
  - A cut can be made anywhere in a particular loop formed by feedback connection, as long as there is only one cut per (state variable) loop
- Next-state and output expressions are derived from the circuit
- The excitation table is derived
- A flow table is obtained
- A corresponding state diagram is derived from the flow table if desired

ASYNCHRONOUS BEHAVIOR OF SR-LATCH



$$Y = \overline{\overline{(y + S)} + R}$$

$$= \overline{y'S' + R} = (y + S)R'$$

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) State-assigned table

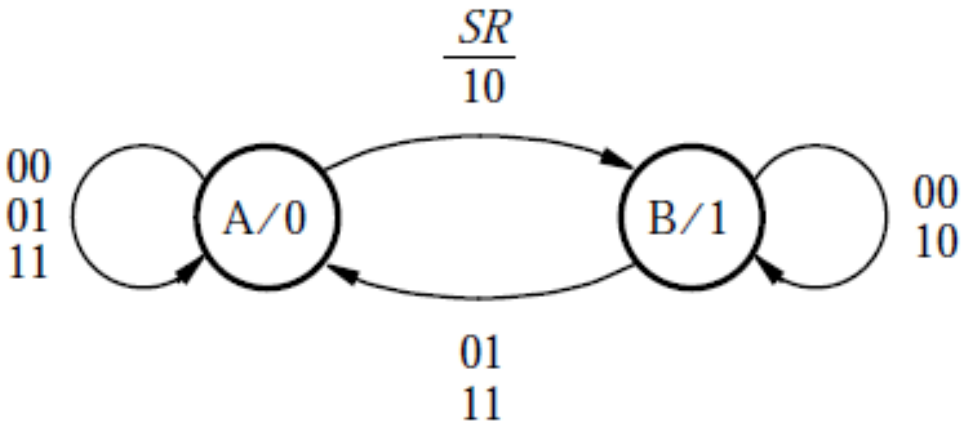
Stable state: given inputs, if a circuit reaches a state and remains in that state, then the state is said to be "stable".

Circles denote "stable" states, i.e., state "unchanged".

MOORE FSM MODEL

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

(a) State table



(b) State diagram

$$Y = \overline{\overline{(y + S)} + R}$$

$$= \overline{y'S' + R}$$

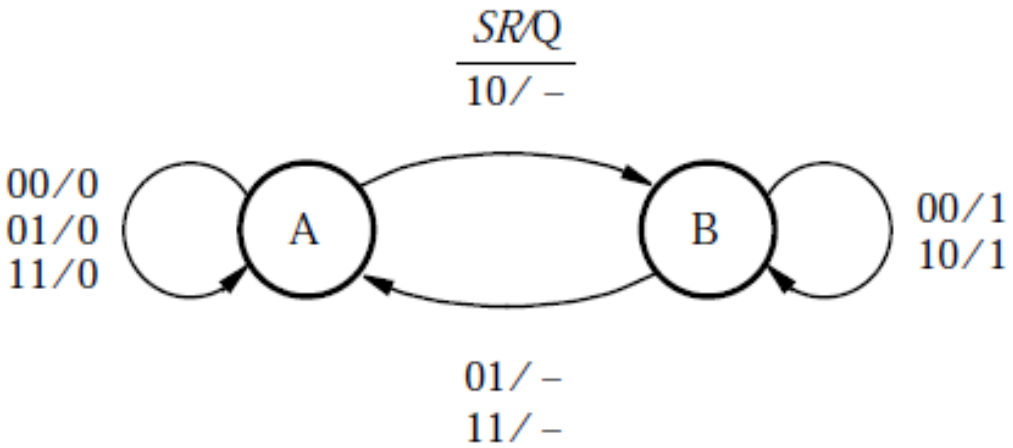
$$= (y + S)R'$$

$$Q = y$$

MEALY FSM

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

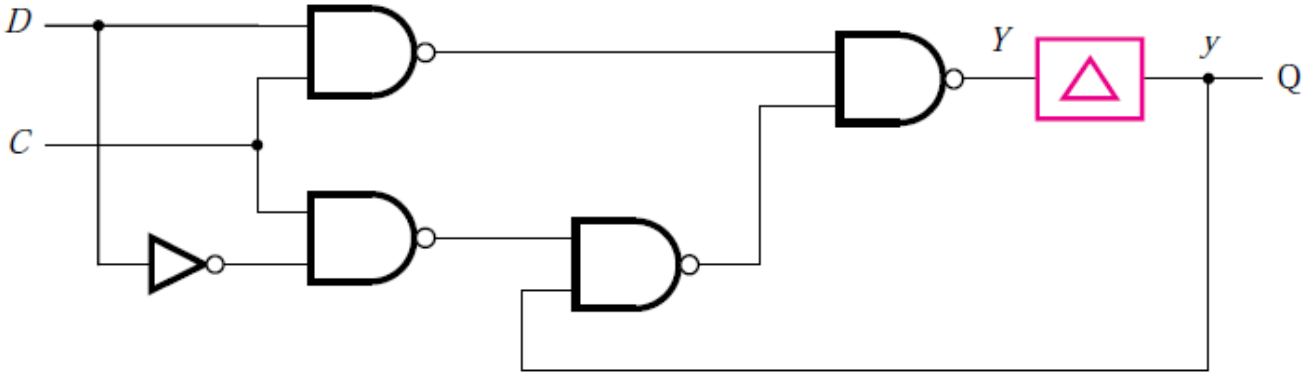
(a) State table



(b) State diagram

Unspecified output :
At state A: input 10 changes to state B, Y=1, Q=1 after reaching state B.
No need to change it beforehand.

GATED D-LATCH



$$Y = \overline{\overline{CD} \cdot \overline{CD'} \cdot y}$$

$$= \overline{(C'+D') \cdot \overline{(C'+D) \cdot y}}$$

$$= \overline{(C'+D') \cdot (\overline{(C'+D)} + y')}$$

$$= \overline{(C'+D') \cdot (CD'+y')}$$

$$= CD + C'y + Dy$$

$$Q = y$$

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

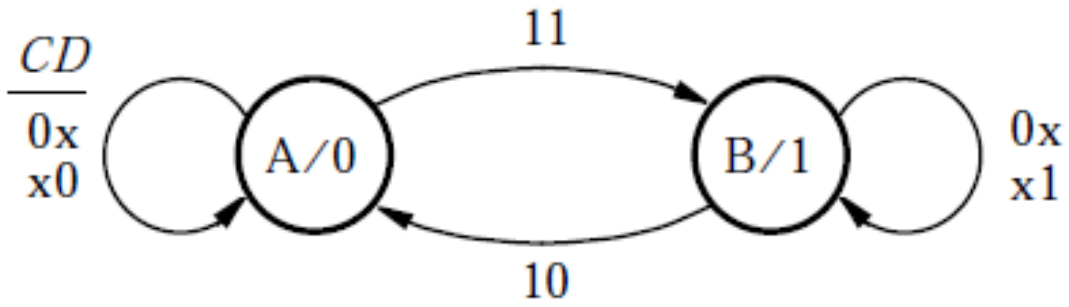
(b) Excitation table

Consensus term

GATED D-LATCH (2)

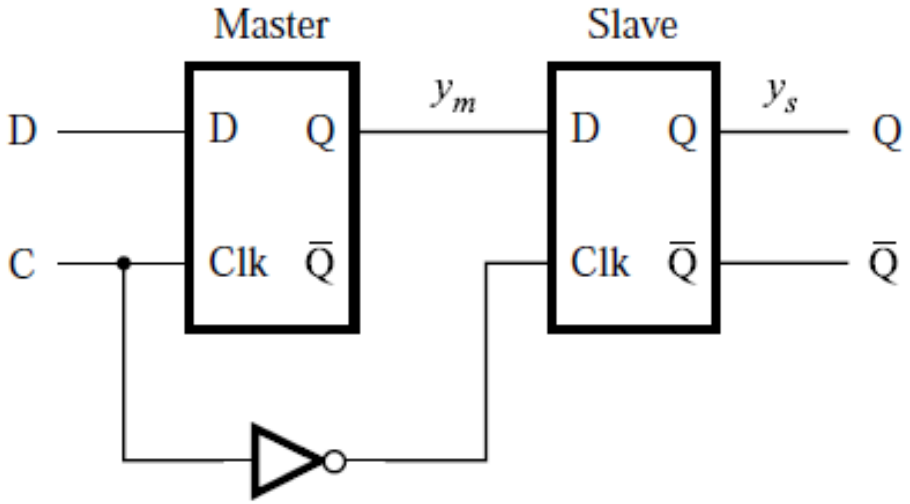
Present state	Next state				
	$CD = 00$	01	10	11	Q
A	(A)	(A)	(A)	B	0
B	(B)	(B)	A	(B)	1

(c) Flow table



(d) State diagram

MASTER-SLAVE D FLIP-FLOP



$$Y_m = CD + C' y_m$$

$$Y_s = C' y_m + C y_s$$

$$Q = y_s$$

Present state $y_m y_s$	Next state $Y_m Y_s$				Output Q
	$CD = 00$	01	10	11	
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) Excitation table

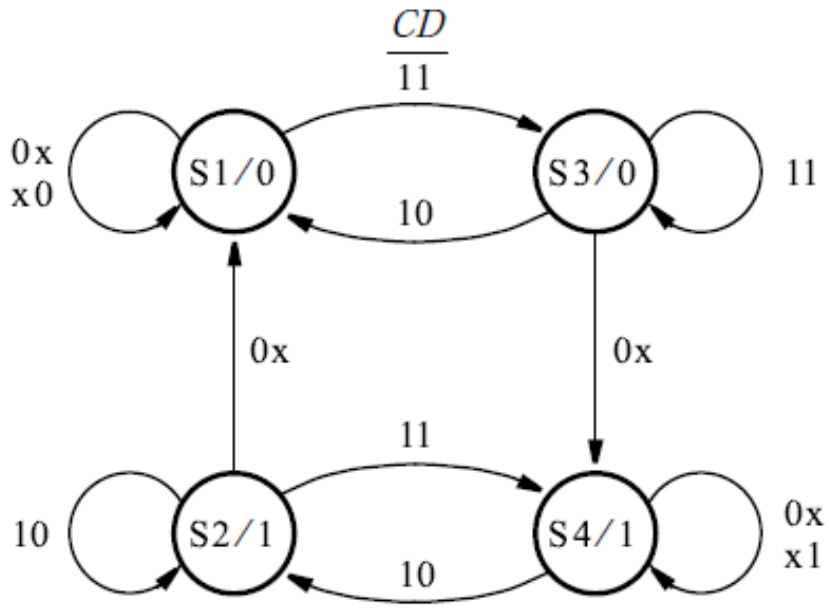
MASTER-SLAVE D FLIP-FLOP (2)

Present state	Next state				Output Q
	$CD = 00$	01	10	11	
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

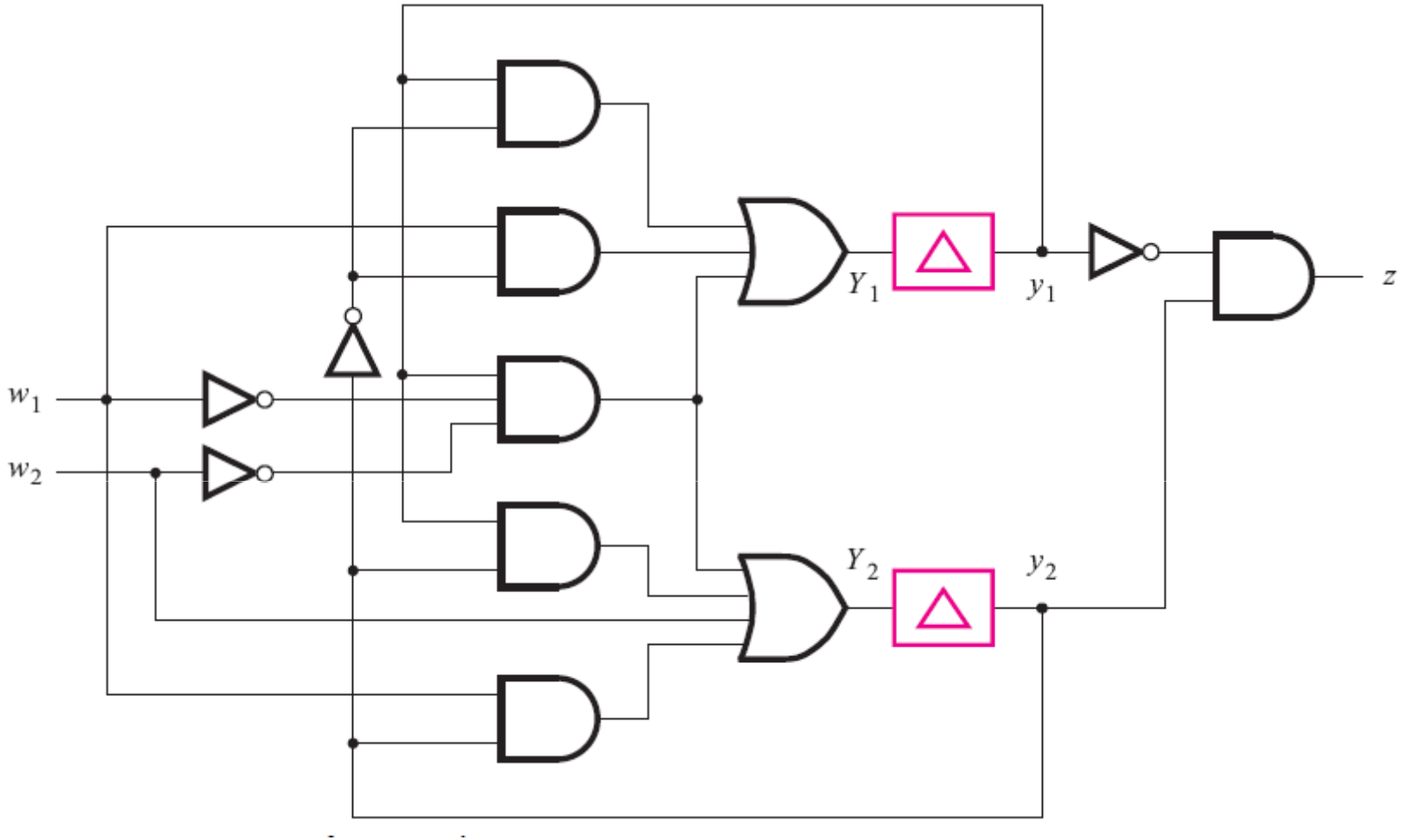
Present state	Next state				Output Q
	$CD = 00$	01	10	11	
S1	(S1)	(S1)	(S1)	S3	0
S2	S1	—	(S2)	S4	1
S3	—	S4	S1	(S3)	0
S4	(S4)	(S4)	S2	(S4)	1

(c) Flow table with unspecified entries



Unspecified entries :
At state S2: stable with input CD=10. Thus, cannot achieve CD=01. (2 inputs cannot change at the same time.)

Further Example



$$Y_1 = y_1 y_2' + w_1 y_2' + w_1' w_2' y_1$$

$$Y_2 = y_1 y_2 + w_1 y_2 + w_2 + w_1' w_2' y_1; z = y_1' y_2$$

FURTHER EXAMPLE (2)

Present state y_2y_1	Next state $w_2w_1 = 00$	01	10	11	Output z
	Y_2Y_1	Y_2Y_1	Y_2Y_1	Y_2Y_1	
00	(00)	01	10	11	0
01	11	(01)	11	11	0
10	00	(10)	(10)	(10)	1
11	(11)	10	10	10	0

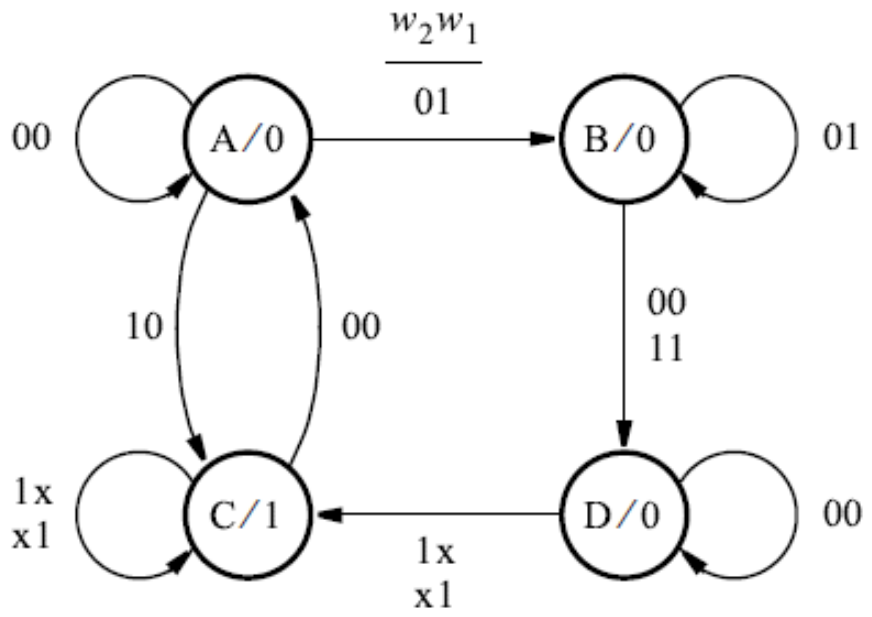
(a) Excitation table

Present state	Next state $w_2w_1 = 00$	01	10	11	Output z
A	(A)	B	C	D	0
B	D	(B)	D	D	0
C	A	(C)	(C)	(C)	1
D	(D)	C	C	C	0

(b) Flow table

FURTHER EXAMPLE (3)

Present state	Next state				Output
	$w_2w_1 = 00$	01	10	11	z
A	(A)	B	C	–	0
B	D	(B)	–	D	0
C	A	(C)	(C)	(C)	1
D	(D)	C	C	C	0



Unspecified entries :
At state B: stable with input w_2w_1=01. Thus, cannot achieve w_2w_1=10. (2 inputs cannot change at the same time.)

Synthesis of Asynchronous Circuits

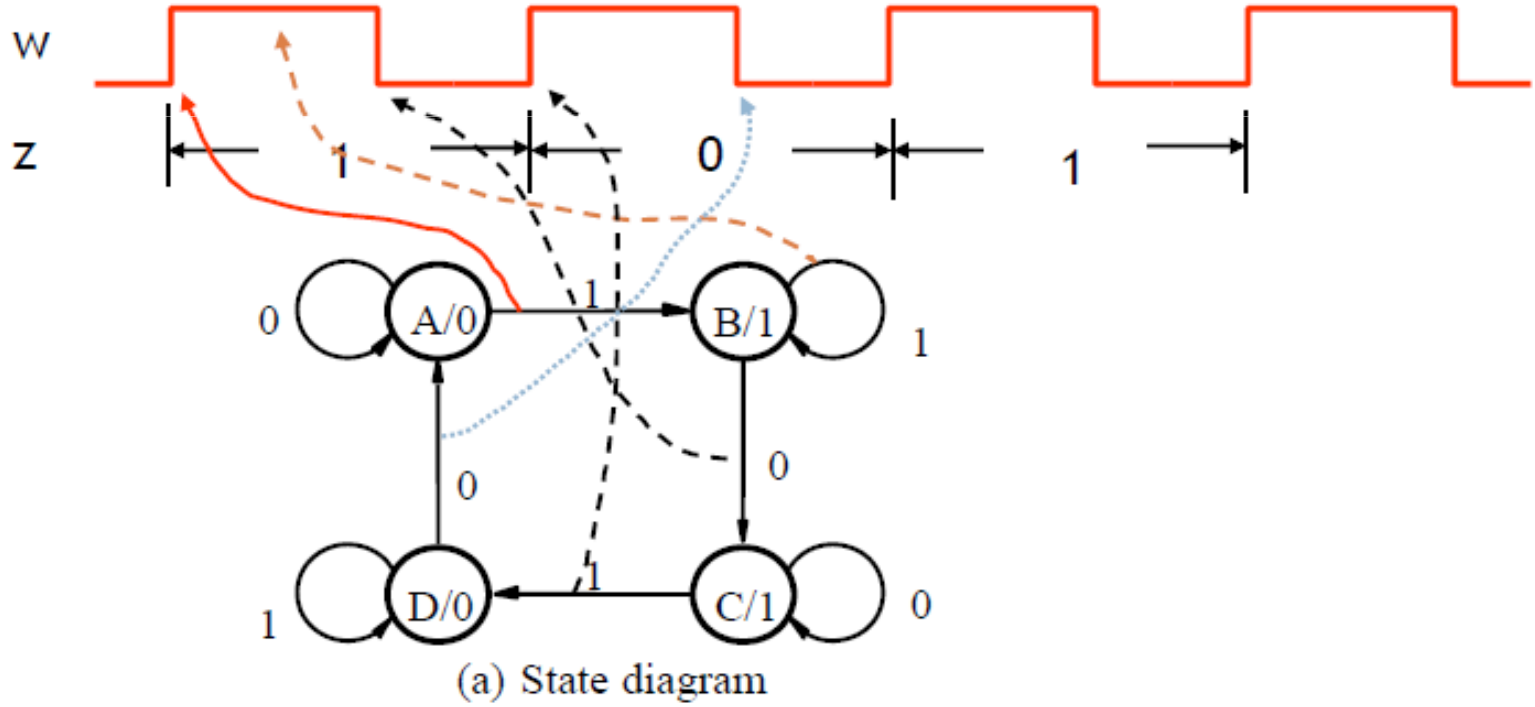
the same basic steps used to synthesize the synchronous circuits

1. Devise a state diagram for an FSM
2. Derive the flow table and reduce the number of states if Possible
3. Perform the state assignment and derive the excitation table
4. Obtain the next-state and output expressions
5. Construct a circuit that implements these expressions

- ❖ Reverse of Analysis

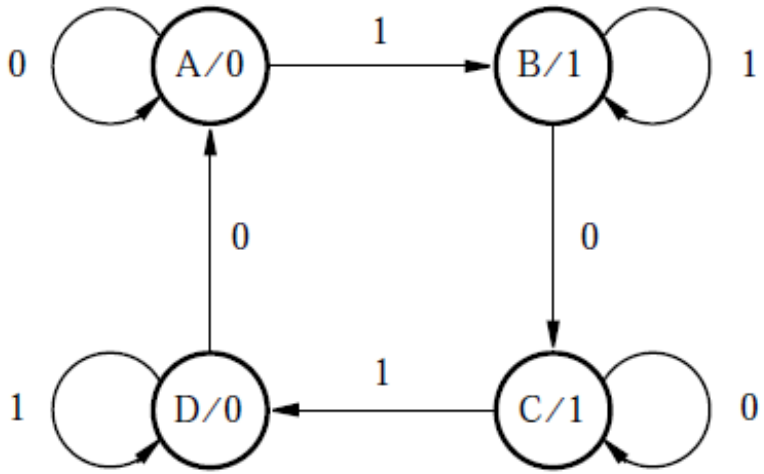
EXAMPLE: SERIAL PARITY GENERATOR

- input w : pulses are applied to w
- output z
- z=1 if the number of previously applied pulses is odd



(a) State diagram

EXAMPLE: SERIAL PARITY GENERATOR (2)



(a) State diagram

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 ASSIGNMENT

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

(b) Good state assignment

- State assignment (a) has a major flaw
- state D =11 : w=0 → state A
- y_2y_1=11 → y_2y_1=00
- the values of the next-state variables determined by the networks of logic gates with varying delays
  - suppose y_1 changes first
    - y_2y_1=10 -> state C(10)
    - state C is stable when w=0
  - suppose y_2 changes first
    - y_2y_1=01-> state B (01)
    - try to change to y_2y_1=10 when w=0
    - if y_1 changes first, y_2y_1=00
  - *race condition occurs*

Race Condition

A race is said to occur if between next set of states **S_q and present states s_q,** there are two or more bits change (two or more latches undergoes change in Q output) at the memory or delay section.

Race condition arises due to variable number of delays of different latches, which lead to intermediate states.

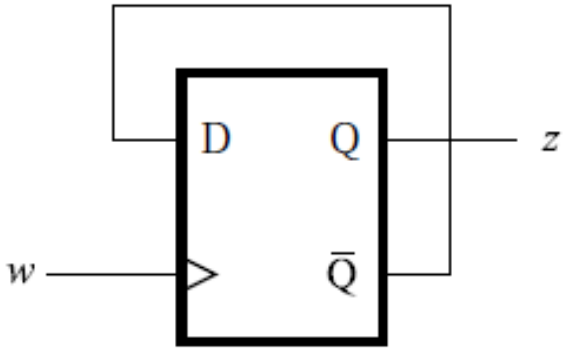
NEXT-STATE EQ'S & CIRCUIT DIAGRAM

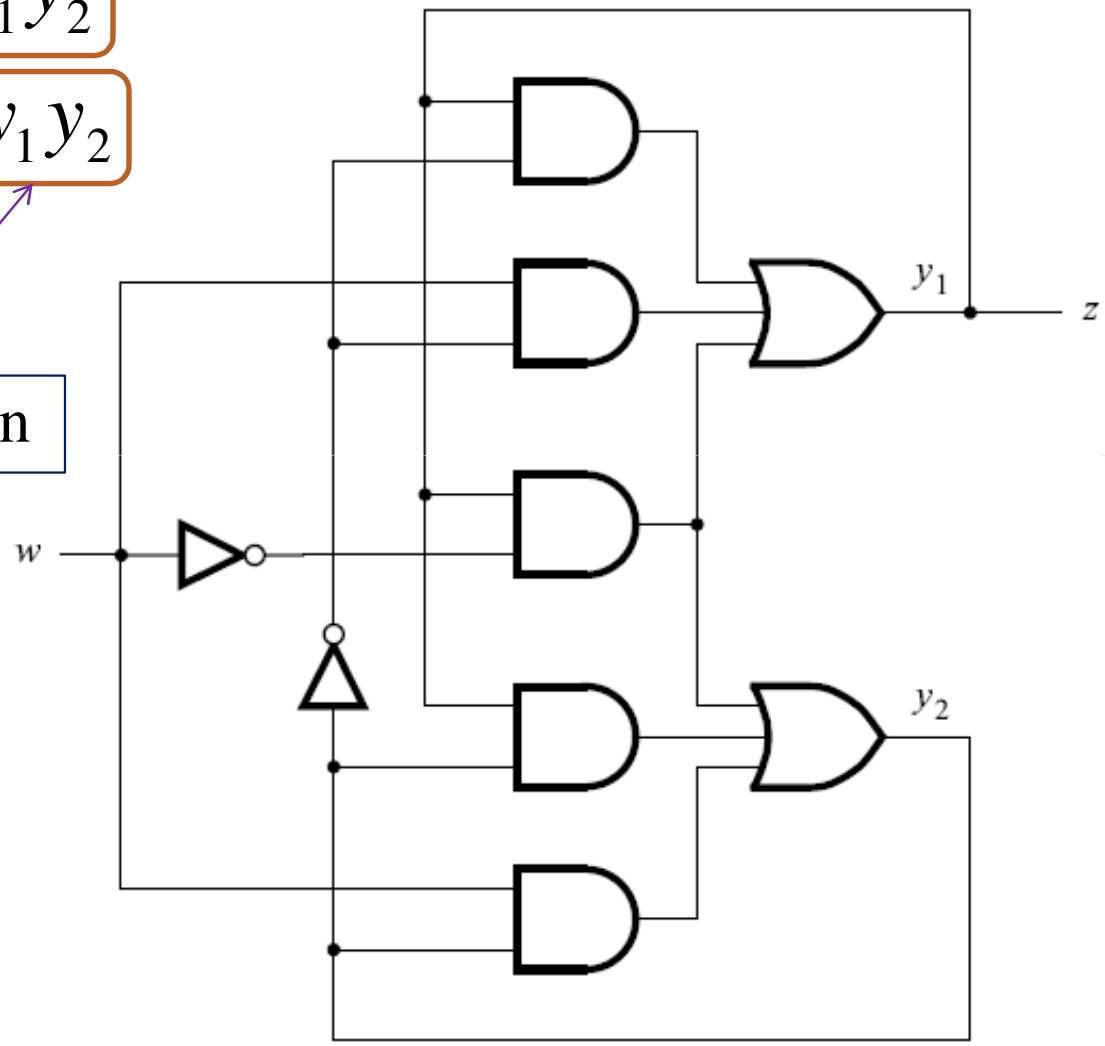
$$Y_1 = wy_2' + w'y_1 + y_1y_2$$

$$Y_2 = wy_2 + w'y_1 + y_1y_2$$

$$z = y_1$$

Hazard prevention

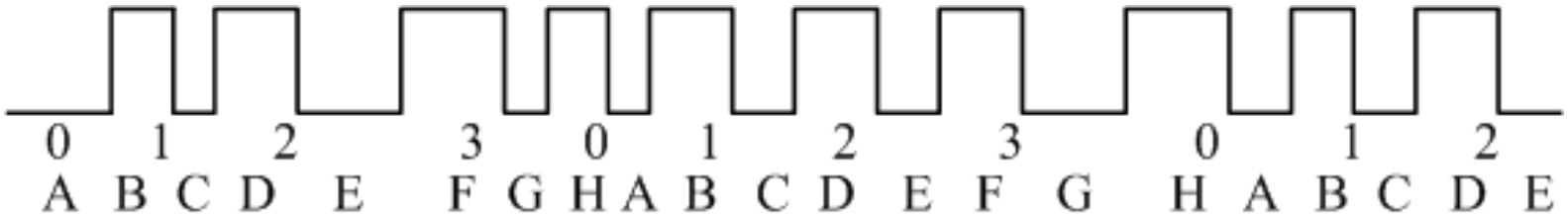


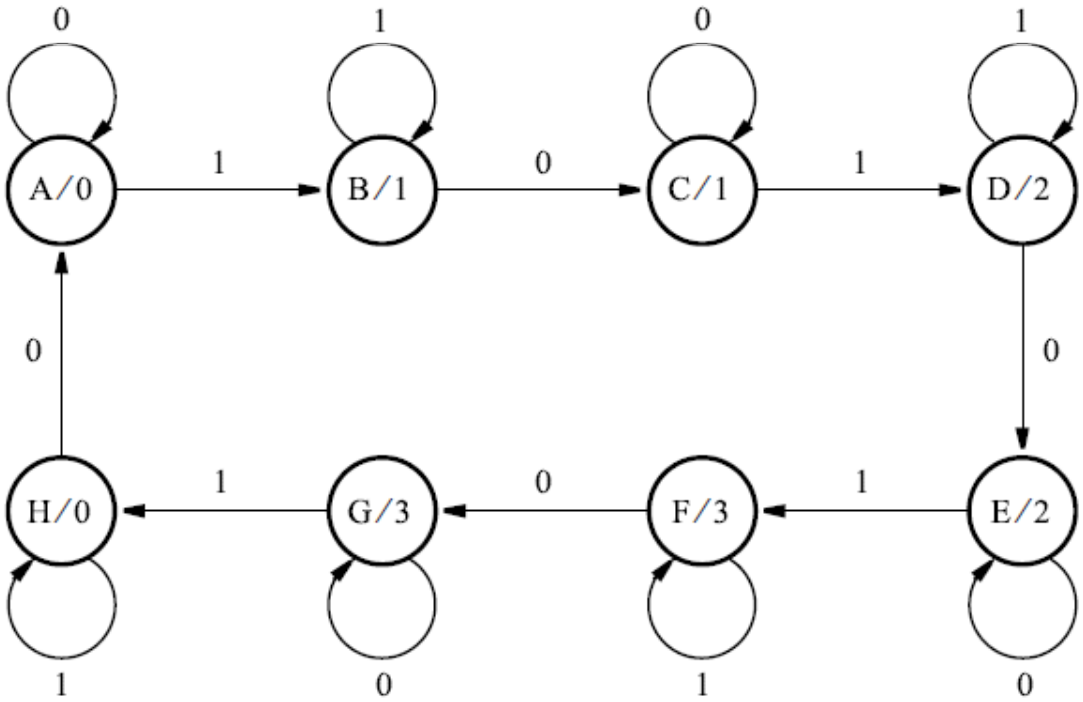


Synchronous Solution

MODULO-4 UP COUNTER

- Input w: 1 cause A→B, and stay there.
- w=0 cause B→C and so on





Flow Table & Excitation Table

Present state	Next state		Output
	$w = 0$	$w = 1$	z
A	(A)	B	0
B	C	(B)	1
C	(C)	D	1
D	E	(D)	2
E	(E)	F	2
F	G	(F)	3
G	(G)	H	3
H	A	(H)	0

(a) Flow table

Present state	Next state		Output
	$w = 0$	$w = 1$	
$y_3y_2y_1$	$Y_3Y_2Y_1$		z_2z_1
000	(000)	001	00
001	011	(001)	01
011	(011)	010	01
010	110	(010)	10
110	(110)	111	10
111	101	(111)	11
101	(101)	100	11
100	000	(100)	00

(b) Excitation table

NEXT STATE & OUTPUT EQ'S

$$Y_1 = w' y_1 + w y_2 y_3 + w y_2' y_3' + y_1 y_2 y_3 + y_1 y_2' y_3'$$

$$Y_2 = w y_2 + w' y_1 y_3' + y_1' y_2 + y_2 y_3'$$

$$Y_3 = w y_3 + y_1 y_3 + y_1' y_2 w' + y_2 y_3$$

$$z_1 = y_1$$

$$z_2 = y_1 y_3 + y_1' y_2$$

- Exercise: Draw the circuit diagram.

State Reduction

- For simpler implementation (fewer FF's and so on)
- Two step approach
  - Partitioning (Same as the synchronous case)
    - Equivalent states : equivalent k-successors and equivalent stable next-state with same input.
  - Use **merger diagram**

COMPATIBILITY OF STATES

Definition: *Two states (rows in a flow table), S_i and S_j , are said to be compatible if there are no state conflicts for any input valuation. Thus for each input valuation, one of the following conditions must be true:*

- *both S_i and S_j have the same successor, or*
- *both S_i and S_j are stable, or*
- *the successor of S_i or S_j , or both, is unspecified.*

Moreover, *both S_i and S_j must have the same output whenever specified.*

Compatible States Example

Present state	Next state				Output z
	$w_2w_1 = 00$	01	10	11	
A	(A)	H	B	–	0
B	F	–	(B)	C	0
C	–	H	–	(C)	1
D	A	(D)	–	E	1
E	–	D	G	(E)	1
F	(F)	D	–	–	0
G	F	–	(G)	–	0
H	–	(H)	–	E	0

Primitive Flow Table

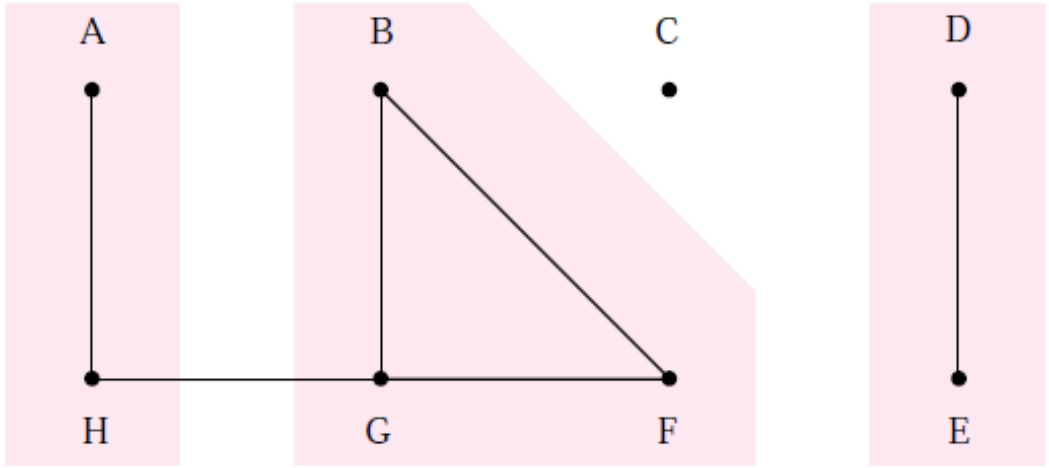
Compatible pairs:

(A,H), (B,F),
(B,G), (D,E),
(F,G), (G,H).

Merger Diagram

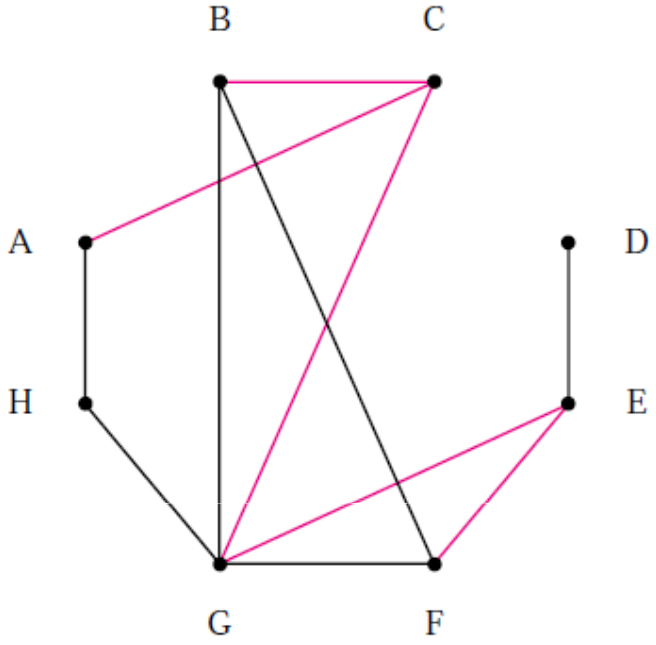
- Used to represent the *compatibility* relationship.
- Procedure
  - Each row of the flow table is represented as a point, labeled by the name of the row.
  - Aline is drawn connecting any two points that correspond to compatible states (rows).
- The reduced flow table can be derived from the merger diagram by choosing the "best" merging possibility.

Merger Diagram for Previous Compatible States Example



Present state	Next state				Output
	$w_2w_1 = 00$	01	10	11	z
A	(A)	(A)	B	D	0
B	(B)	D	(B)	C	0
C	–	A	–	(C)	1
D	A	(D)	B	(D)	1

Reduced Moore-type Flow Table



Complete Merger Diagram (Include Mealy-type FSM)
Black line : Moore-type

State Reduction Procedure

1. Use the partitioning procedure to eliminate the equivalent states in a primitive flow table.
2. Construct a merger diagram for the resulting flow table.
3. Choose subsets of compatible states that can be merged, trying to minimize the number of subsets needed to cover all states. Each state must be included in only one of the chosen subsets.
4. Derive the reduced flow table by merging the rows in chosen subsets.
5. Repeat steps 2 to 4 to see whether further reductions are possible.

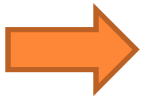
State Reduction Example 1

Present state	Next state $w_2w_1 = 00$	01	10	11	Output z
A	(A)	F	C	–	0
B	A	(B)	–	H	1
C	G	–	(C)	D	0
D	–	F	–	(D)	1
E	G	–	(E)	D	1
F	–	(F)	–	K	0
G	(G)	B	J	–	0
H	–	L	E	(H)	1
J	G	–	(J)	–	0
K	–	B	E	(K)	1
L	A	(L)	–	K	1

$$P_1 = (AG)(BL)(C)(D)(E)(F)(HK)(J)$$

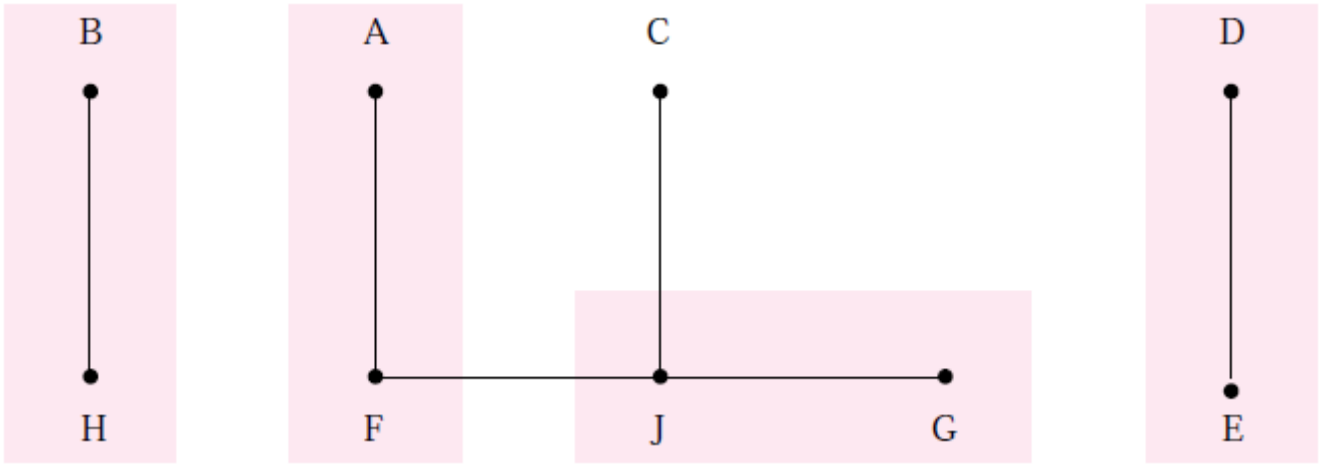
$$P_2 = (A)(G)(BL)(C)(D)(E)(F)(HK)(J)$$

$$P_3 = P_2$$



Present state	Next state $w_2w_1 = 00$	01	10	11	Output z
A	(A)	F	C	–	0
B	A	(B)	–	H	1
C	G	–	(C)	D	0
D	–	F	–	(D)	1
E	G	–	(E)	D	1
F	–	(F)	–	H	0
G	(G)	B	J	–	0
H	–	B	E	(H)	1
J	G	–	(J)	–	0

State Reduction Example 1 (2)

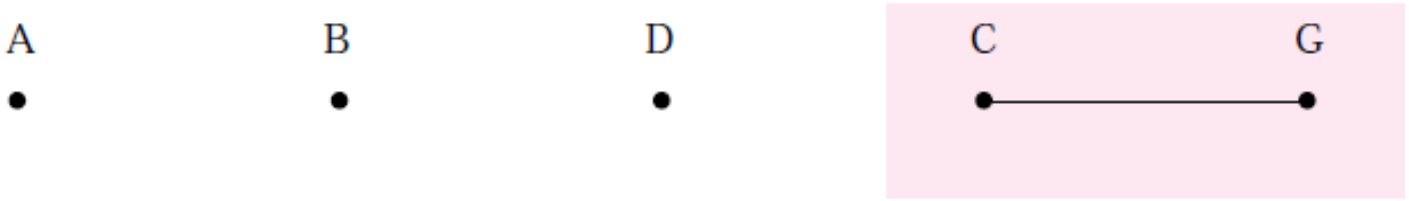


Merger Diagram

Present state	Next state $w_2w_1 = 00$	01	10	11	Output z
A	(A)	(A)	C	B	0
B	A	(B)	D	(B)	1
C	G	—	(C)	D	0
D	G	A	(D)	(D)	1
G	(G)	B	(G)	—	0

Reduced Flow Table

State Reduction Example 1 (3)



Merger Diagram for reduced flow table

Present state	Next state $w_2w_1 = 00$	01	10	11	Output z
A	(A)	(A)	C	B	0
B	A	(B)	D	(B)	1
C	(C)	B	(C)	D	0
D	C	A	(D)	(D)	1

Finalized Flow Table

State Reduction Example 2

Present state	Next state $w_2w_1 = 00$	01	10	11	Output z
A	(A)	B	C	–	0
B	F	(B)	–	H	0
C	F	–	(C)	H	0
D	(D)	G	C	–	1
E	A	(E)	–	H	0
F	(F)	E	C	–	0
G	D	(G)	–	H	0
H	–	G	C	(H)	1

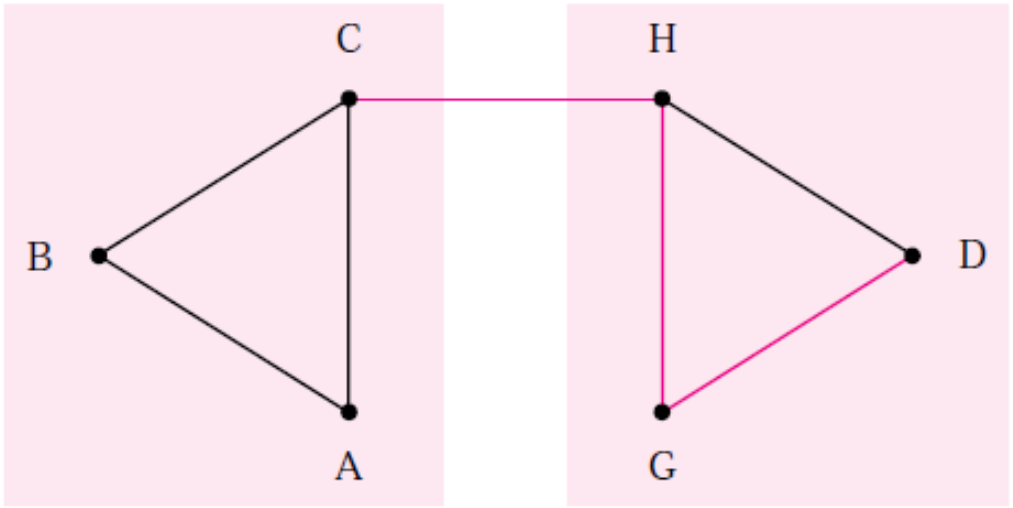
$$P_1 = (AF)(BEG)(C)(D)(H)$$

$$P_2 = (AF)(BE)(G)(C)(D)(H)$$

$$P_3 = P_2$$

Present state	Next state $w_2w_1 = 00$	01	10	11	Output z
A	(A)	B	C	–	0
B	A	(B)	–	H	0
C	A	–	(C)	H	0
D	(D)	G	C	–	1
G	D	(G)	–	H	0
H	–	G	C	(H)	1

State Reduction Example 2 (2)



Merger Diagram

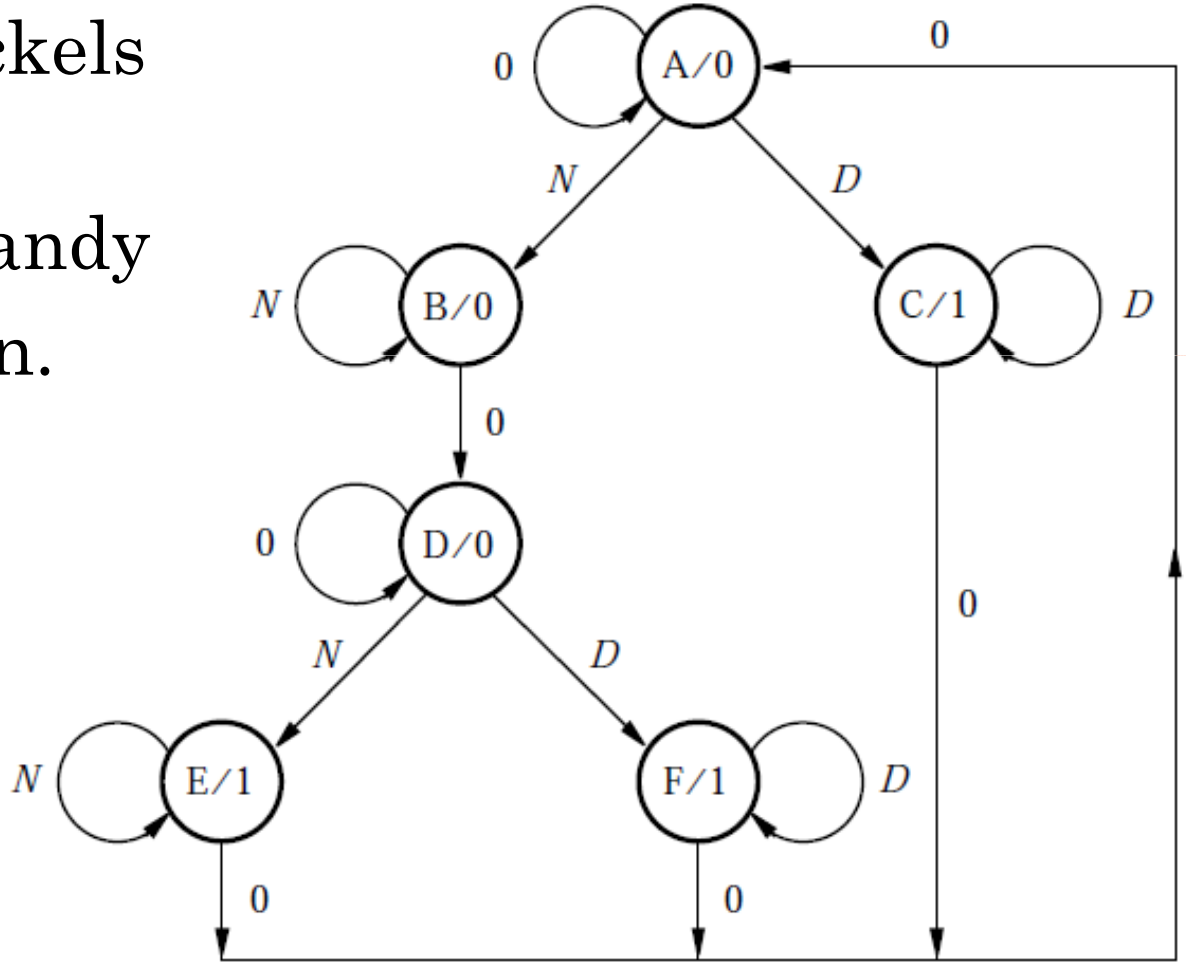
Present state	Next state				Output z			
	$w_2w_1 = 00$	01	10	11	00	01	10	11
A	(A)	(A)	(A)	D	0	0	0	—
D	(D)	(D)	A	(D)	1	0	—	1

Reduced Flow Table

An Asynchronous Vending Machine

Specification A candy vending machine with

- accepts only nickels and dimes
- 10 cents for 1 candy
- No change given.



Initial State Diagram

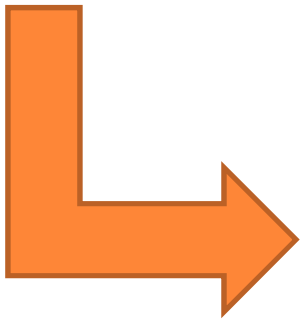
An Asynchronous Vending Machine (2)

Present State	Next state				Output
	$DN = 00$	01	10	11	z
A	(A)	B	C	–	0
B	D	(B)	–	–	0
C	A	–	(C)	–	1
D	(D)	E	F	–	0
E	A	(E)	–	–	1
F	A	–	(F)	–	1

$$P_1 = (AD)(B)(CF)(E)$$

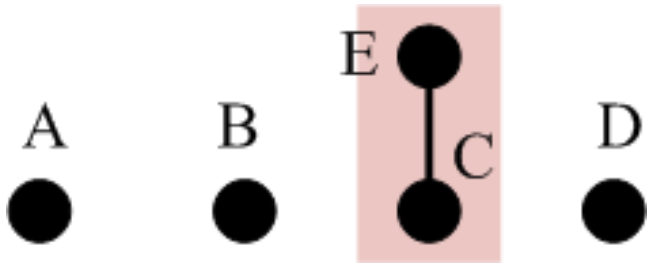
$$P_2 = (A)(D)(B)(CF)(E)$$

$$P_3 = P_2$$



Present state	Next state				Output
	$DN = 00$	01	10	11	z
A	(A)	B	C	–	0
B	D	(B)	–	–	0
C	A	–	(C)	–	1
D	(D)	E	C	–	0
E	A	(E)	–	–	1

An Asynchronous Vending Machine (3)



Merger Diagram

Present state	Next state				Output
	DN = 00	01	10	11	*z*
A	(A)	B	C	–	0
B	D	(B)	–	–	0
C	A	(C)	(C)	–	1
D	(D)	C	C	–	0

Reduced Flow Table

State Assignment

- To achieve reliable operation of the circuit, the state variables should change their values one at a time in controlled fashion. This can prevent the race.
- Hamming distance : the number of different bits, e.g., 0100,0011 → distance 3.
- Ideal state assignment: Hamming distance of 1 for all transitions from one stable state to another. (may not be possible!!!)

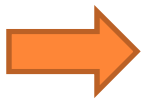
TRANSITION DIAGRAM

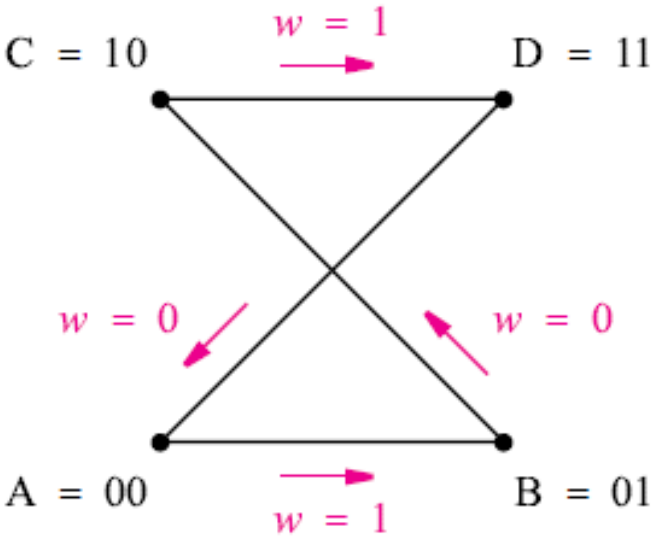
- Or *state-adjacency diagram*, used for depicting the state transitions to provide a convenient aid in searching for a suitable state assignment.
- A good state assignment results if the transition diagram does not have any **diagonal** paths.
- Condition: Must be possible to *embed* the transition diagram onto a *k-dimensional cube*

TRANSITIONS EXAMPLE

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

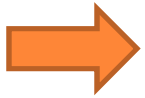
(a) Poor state assignment

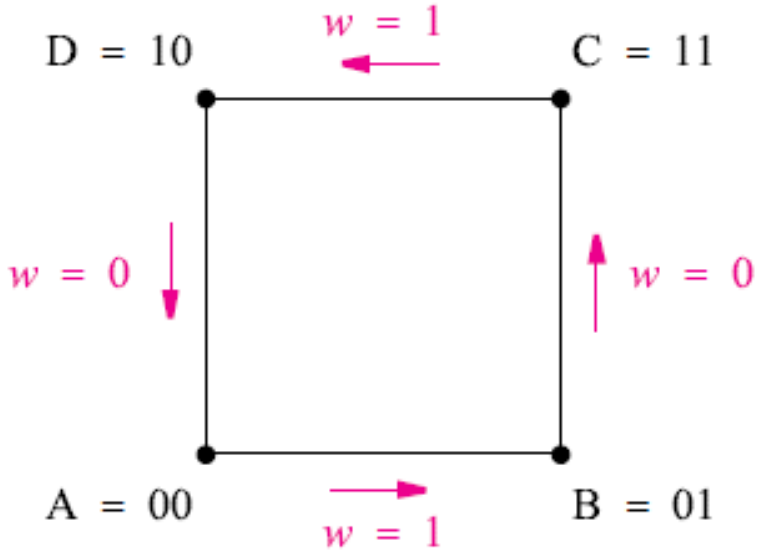




Present state y_2y_1	Next state Y_2Y_1 $w = 0$	Next state Y_2Y_1 $w = 1$	Output z
00	(00)	01	0
01	11	(01)	1
11	(11)	10	1
10	00	(10)	0

(b) Good state assignment





RACE CONDITION

- A **race condition** occurs in an asynchronous circuit when 2 or more state variables change in response to a change in the value of a circuit input.
- Unequal circuit delays may imply that the 2 or more state variables may not change simultaneously – this can cause a problem.
- Assume two state variables change…
  - If the circuit reaches the same final, stable state regardless of the order in which the state variables change, then the race in **non-critical.**
  - If the circuit reaches a different final, stable state depending on the order in which the state variables change, then the race is **critical.**
- We need to avoid critical races for predictability and to ensure our circuit does the intended function!

NON-CRITICAL RACE

- Assume current state is 00, and input changes from 0 →1. This requires y_1y_2 to change from 00 → 11.

		x	
	y1y2	0	1
curr state	00	00	11
	01	00	01
	11	00	01
	10	00	11

- Depending on circuit delays, several possible transition sequences:
  - 00 -> 11 -> 01 (simultaneous change for y1 and y2).
  - 00 -> 01 (y2 changes first).
  - 00 -> 10 -> 11 -> 01 (y1 changes first)
- In all cases, we end up in the same stable state, so the race is **non-critical**.

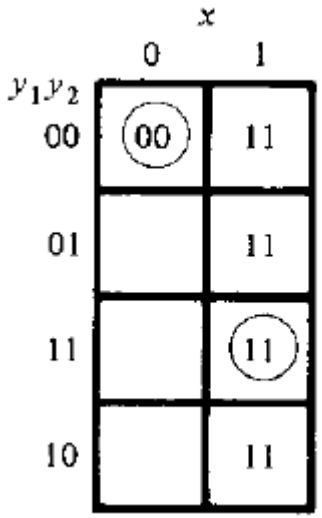
CRITICAL RACE

- Assume current state is 00, and input changes from 0->1. This requires y_1y_2 to change from 00->11.

curr state \ x	y1y2	0	1
	00	(00)	11
	01	00	11
	11	00	(11)
	10	00	(10)

- Possible transition sequences:
  - 00->11 (y1 and y2 change simultaneously)
  - 00->01->11 (y2 changes first)
  - 00->10 (y1 changes first)
- So, depending on which state variable changes first, we can get into **a different stable state – the race is critical.**

ADDITIONAL EXAMPLE

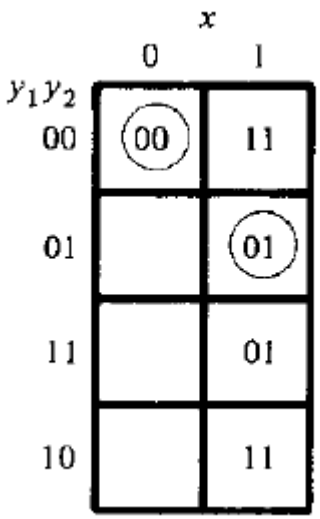


(a) Possible transitions:

00 → 11

00 → 01 → 11

00 → 10 → 11



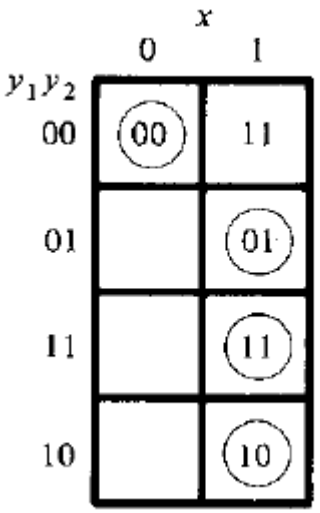
(b) Possible transitions:

00 → 11 → 01

00 → 01

00 → 10 → 11 → 01

Non-critical

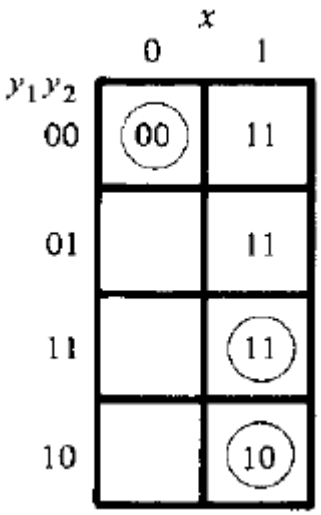


(a) Possible transitions:

00 → 11

00 → 01

00 → 10



(b) Possible transitions:

00 → 11

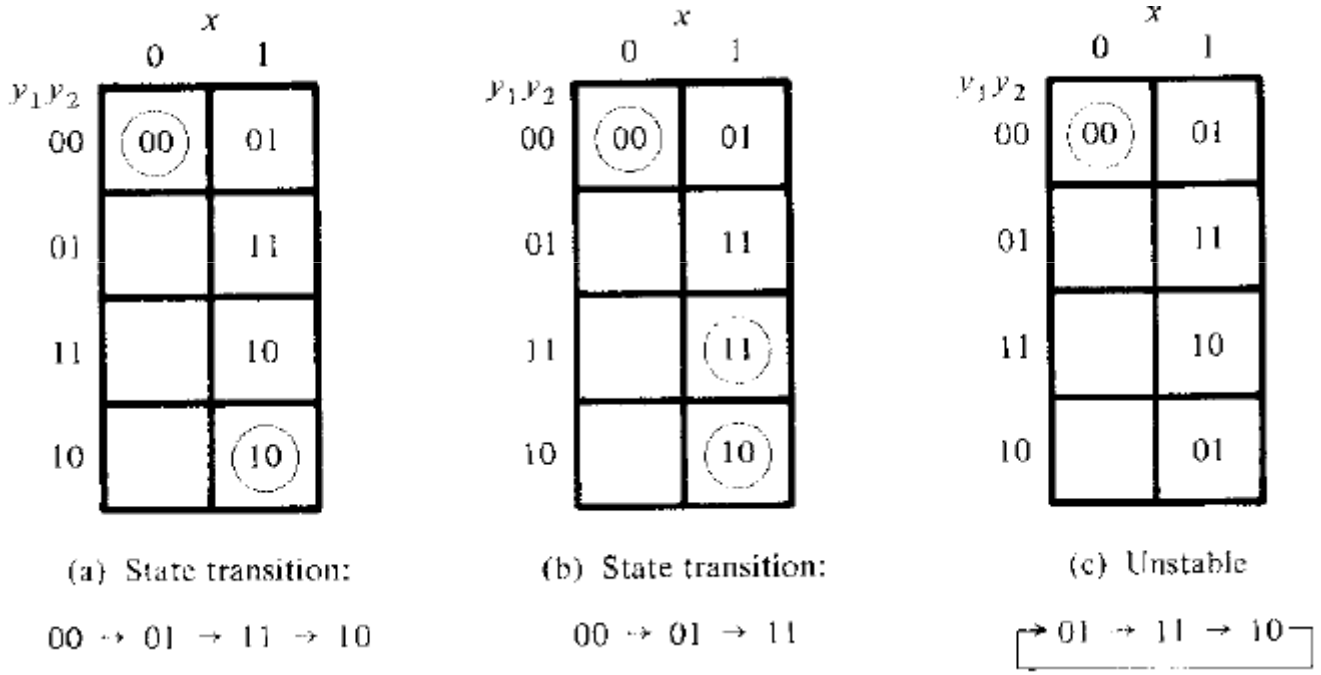
00 → 01 → 11

00 → 10

Critical

CYCLES

- A cycle occurs when a circuit goes through a unique sequence of **unstable** states.



y_1y_2 \ x	0	1
00	(00)	01
01		11
11		10
10		(10)

(a) State transition:

00 → 01 → 11 → 10

y_1y_2 \ x	0	1
00	(00)	01
01		11
11		(11)
10		(10)

(b) State transition:

00 → 01 → 11

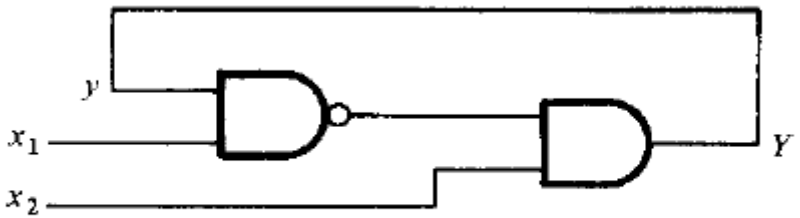
y_1y_2 \ x	0	1
00	(00)	01
01		11
11		10
10		01

(c) Unstable

→ 01 → 11 → 10 →

INSTABILITY

- When x_1x_2=11, y=1, then Y=0, $Y \neq y$.
- Then y=0 $\rightarrow$ Y=1, $Y \neq y$.
- If each gate has a 5-ns propagation delay, Y=0 for 10 ns, and Y=1 for 10 ns, resulting in 50-MHz clock signal!



(a) Logic diagram

y \ x_1x_2	00	01	11	10
0	(0)	1	1	(0)
1	0	(1)	0	0

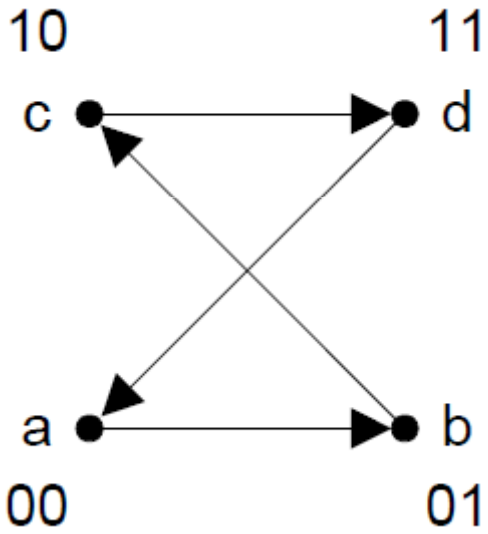
(b) Transition table

Race-Free State Assignment

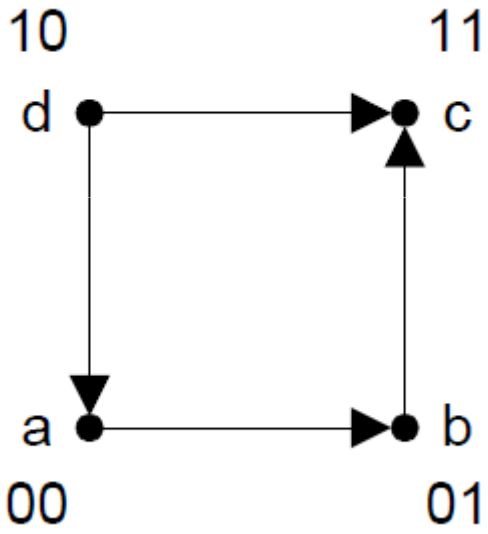
- Can prevent races (pre-emptive) by performing state assignment such that transitions from **one stable state to another stable state only require one state variable to change at a time.**

curr state	next state	
	x=0	x=1
a	(a)	b
b	c	(b)
c	(c)	d
d	a	(d)

Flow Table



BAD
State Assignment



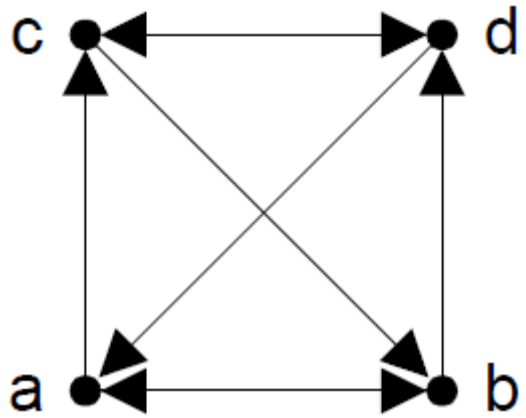
Good
State Assignment

Method One for Race Free State Assignment

- Given a flow table, try and "embed" the symbolic states into the co-ordinates of a "n-dimensional" cube such that the path from stable state to stable state:
  - Is direct along a single edge of the cube, or
  - Goes through newly introduced unstable states along edges of the cube.

METHOD ONE EXAMPLE

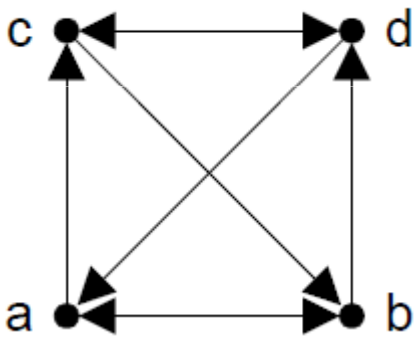
curr state	next state			
	x1x2=00	x1x2=01	x1x2=10	x1x2=11
a	(a)	(a)	c	b
b	a	(b)	d	(b)
c	(c)	b	(c)	d
d	c	a	(d)	(d)

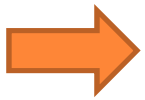


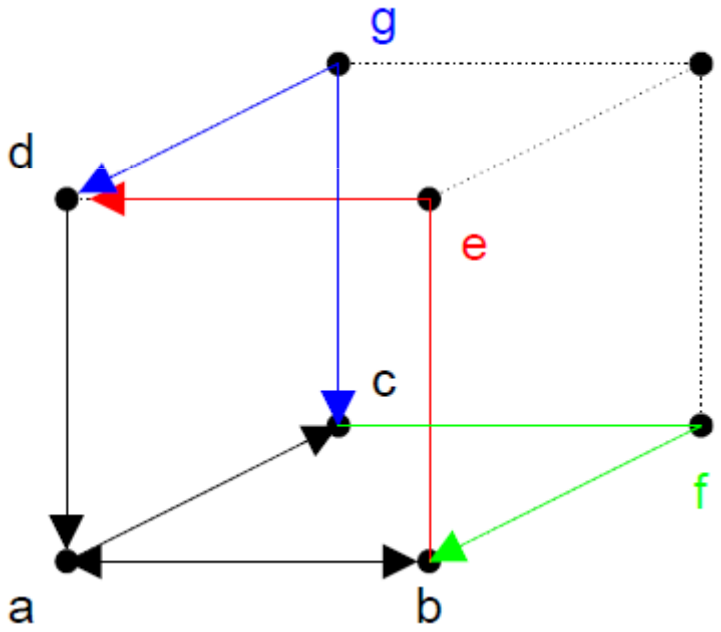
- Can attempt to embed 4 states into a 2-dimensional cube and draw the transition diagram:
  - No state assignment that has only one state variable changing at a time.

METHOD ONE EXAMPLE (CONT)

- Solution is to introduce additional, unstable states and use them as "intermediate" states during transitions.
  - i.e., embed the 4-states into the corners of a 3-dimensional cube.

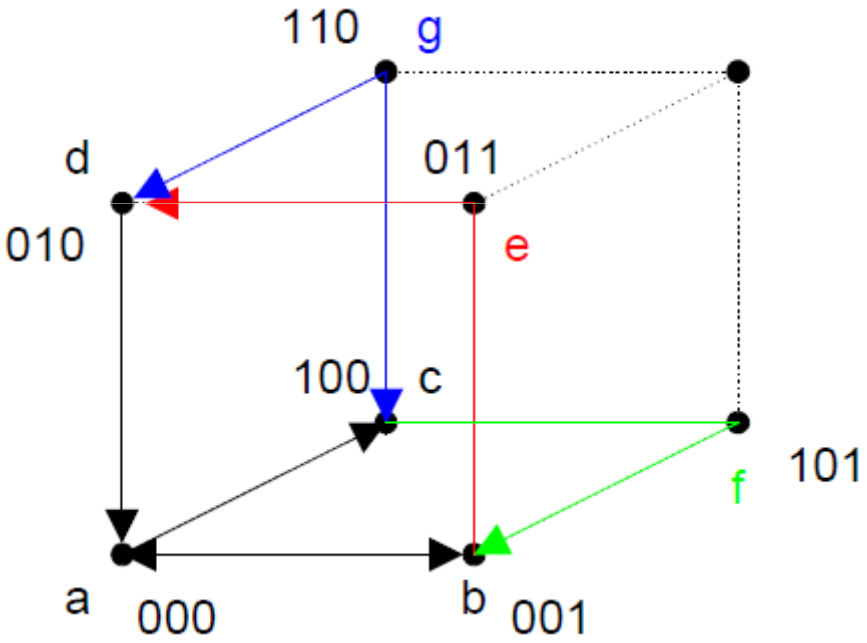






METHOD ONE EXAMPLE (CONT)

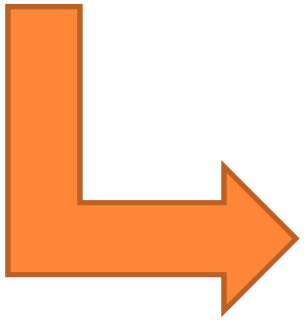
- All transitions are made properly by introducing extra unstable states (Note: the new extra states are always unstable!!!):



EXAMPLE : RACE-FREE FLOW TABLE

- Can now see the original flow table, and an expanded flow table (extra unstable states) that has a race-free state assignment (see previous slide!):

curr state	next state			
	x1x2=00	x1x2=01	x1x2=10	x1x2=11
a	(a)	(a)	c	b
b	a	(b)	d	(b)
c	(c)	b	(c)	d
d	c	a	(d)	(d)



curr state	next state			
	x1x2=00	x1x2=01	x1x2=10	x1x2=11
a	(a)	(a)	c	b
b	a	(b)	e	(b)
c	(c)	b	(c)	g
d	g	a	(d)	(d)
e	-	-	d	-
f	-	b	-	-
g	c	-	-	d

Method Two for Race Free State Assignment

- Method useful for flow tables with <= 4 states.
- Replace a state with multiple (two) equivalent states
  - Note: Outputs must be the same for the **equivalent** states!!!

METHOD TWO EXAMPLE

- Consider the following flow table, and another (larger table) with equivalent states:

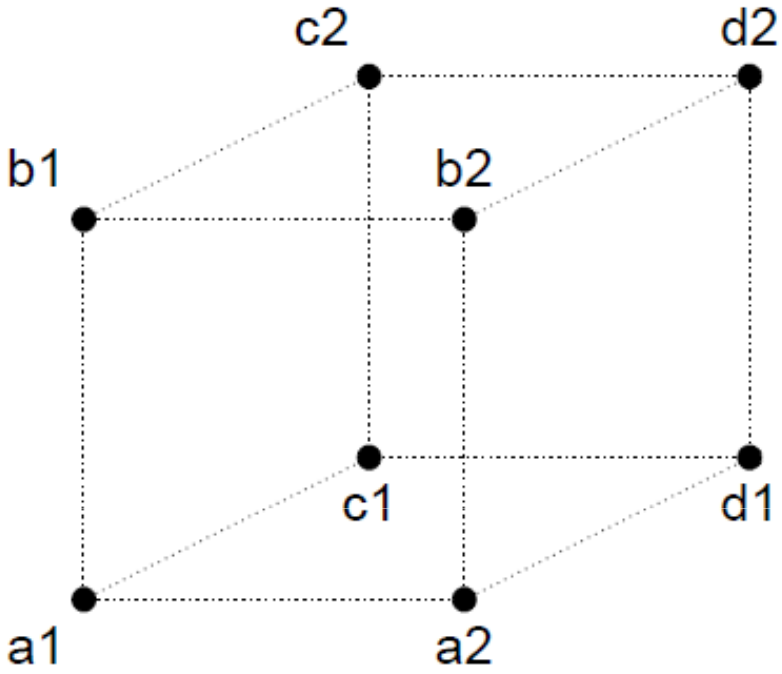
curr state	next state				output
	x1x2=00	x1x2=01	x1x2=10	x1x2=11	
a	(a)	(a)	c	b	0
b	a	(b)	d	(b)	1
c	(c)	b	(c)	d	0
d	c	a	(d)	(d)	1

curr state	next state				output
	x1x2=00	x1x2=01	x1x2=10	x1x2=11	
a1	(a1)	(a1)	c1	b1	0
a2	(a2)	(a2)	a1	b2	0
b1	a1	(b1)	b2	(b1)	1
b2	a2	(b2)	d2	(b2)	1
c1	(c1)	c2	(c1)	d1	0
c2	(c2)	b1	(c2)	d2	0
d1	c1	d2	(d1)	(d1)	1
d2	c2	d1	(d2)	(d2)	1

- The flow table with equivalent states permits a race free state assignment.
- We can see this by looking at the states on a 3-dimensional cube.

METHOD TWO EXAMPLE (CONT)

curr state	next state				output
	x1x2=00	x1x2=01	x1x2=10	x1x2=11	
a1	(a1)	(a1)	c1	b1	0
a2	(a2)	(a2)	a1	b2	0
b1	a1	(b1)	b2	(b1)	1
b2	a2	(b2)	d2	(b2)	1
c1	(c1)	c2	(c1)	d1	0
c2	(c2)	b1	(c2)	d2	0
d1	c1	d2	(d1)	(d1)	1
d2	c2	d1	(d2)	(d2)	1



- With equivalent states, we always have 1 of the 2 equivalent states directly adjacent to every other state!

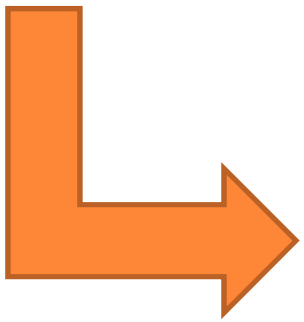
Method Three for Race-Free State Assignment

- Can use the idea of one-hot encoding to get a race free state assignment…
- Given n-states, let the i^{th} state be encoded as 0…010..0 where the 1 is in the i^{th} location.
- For a transition from i^{th} stable state to j^{th} stable state, introduce unstable state with encoding 0..010..010..0 where the 1s are in the i^{th} and j^{th} position.

Method Three Example

- Consider the following flow table, with one-hot state assignment:
- Look for transitions and introduce new unstable states, as required using one-hot encoding scheme.

	curr state	next state			
		x1x2=00	x1x2=01	x1x2=10	x1x2=11
0001	a	(a)	(a)	c	b
0010	b	a	(b)	d	(b)
0100	c	(c)	b	(c)	d
1000	d	c	a	(d)	(d)



	curr state	next state			
		x1x2=00	x1x2=01	x1x2=10	x1x2=11
0001	a	(a)	(a)	e	f
0010	b	f	(b)	g	(b)
0100	c	(c)	h	(c)	i
1000	d	c	j	(d)	(d)
0101	e	-	-	c	-
0011	f	a	-	-	b
1010	g	-	-	d	-
0110	h	-	b	-	-
1100	i	c	-	-	d
1001	i	-	a	-	-

Asynchronous Sequential Circuits Summary

- Analysis:
  - From the logic diagram, determine input, output, state variable.
  - Derive Boolean functions.
  - Obtain excitation table, flow table, state diagram.
- Synthesis (Design):
  - Determine state diagram and primitive flow table.
  - Reduce states if possible.
  - Assign states and obtain excitation table.
  - Derive Boolean functions and design circuit.