EE/CE 6301: Advanced Digital Logic

Bill Swartz

Dept. of EE
Univ. of Texas at Dallas

Session 10

Asynchronous Design

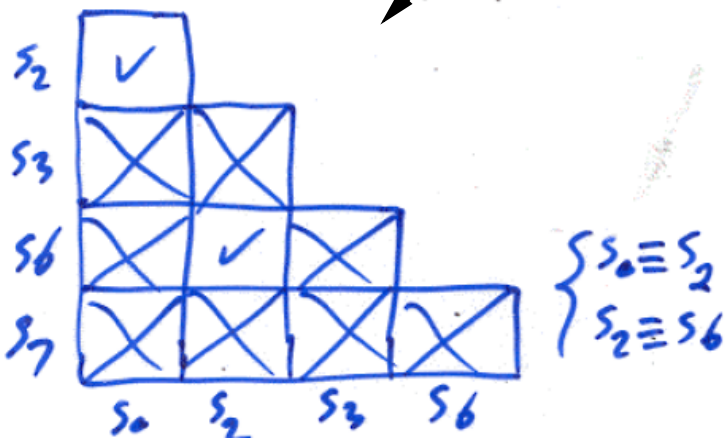
Adapted from the work of M Nourani to whom I am grateful

Optimization Techniques

State Minimization – Implication Chart

- Similar to synchronous design, the implication chart method can be used for state minimization

	CLK D			
S	00	01	11	10
S0	S2 , 01	S6 , 01	(S0), 01	(S0), 01
S2	(S2), 01	S6 , 01	— , —	S0 , 01
S3	(S3), 10	S7 , 10	— , —	S0 , 01
S6	S2 , 01	(S6), 01	S7 , 11	— , —
S7	S3 , 10	(S7), 10	(S7), 10	(S7), 10
	S*, Q /Q			



	CLK D			
S	00	01	11	10
SB	(SB), 01	S6 , 01	(SB), 01	(SB), 01
S3	(S3), 10	S7 , 10	— , –	SB , 01
S6	SB , 01	(S6), 01	S7 , 11	— , –
S7	S3 , 10	(S7), 10	(S7), 10	(S7), 10
	S*, Q /Q			

State Minimization

- State minimization does not have transitive property.
- A typical procedure for state minimization
  - Use implication chart to identify pairs of equivalent states
  - Use a method (e.g. clique identification in graph representation) to identify compatible (equivalent) states
  - Use a method (e.g. covering table similar to Phase II in Q-M) to identify the maximal compatibles (minimum number of states). Be careful about next-state consistency.

State Minimization - Example

Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

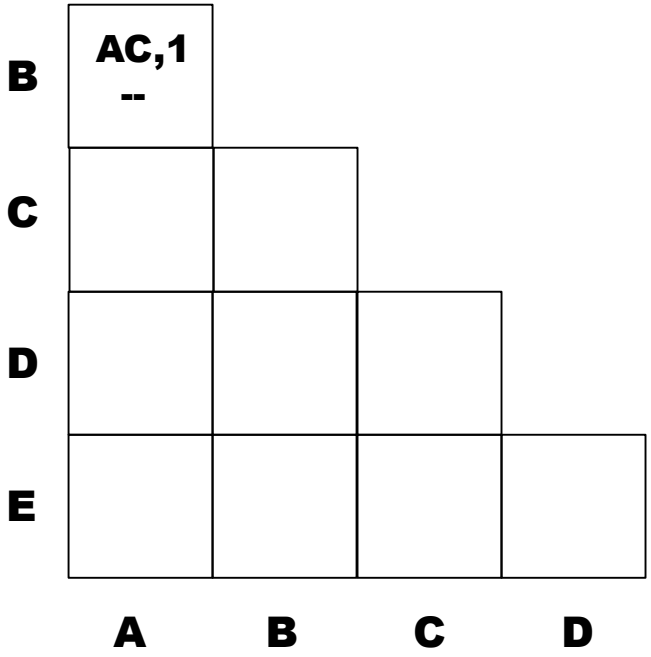
Map states to flow table

Flow table

State Minimization – Implication Chart

Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Flow table



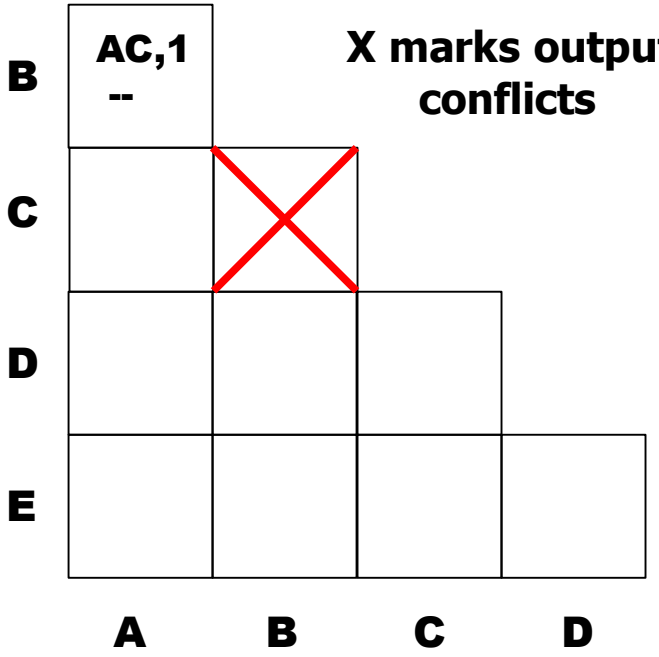
Implication Chart
First pass

A and B are equal if A = C due to X = 0

A and B are equal if B=B when X = 1
But this is an identity and therefore not a constraint

State Minimization – Implication Chart

Present state	Next state	
	$X=0$	$X=1$
A	A,–	B,–
B	C, 1	B,–
C	D, 0	–,–
D	–,–	B, 1
E	A, 0	C,–



X marks output conflicts

B and C can never be equal since outputs conflict when X = 0

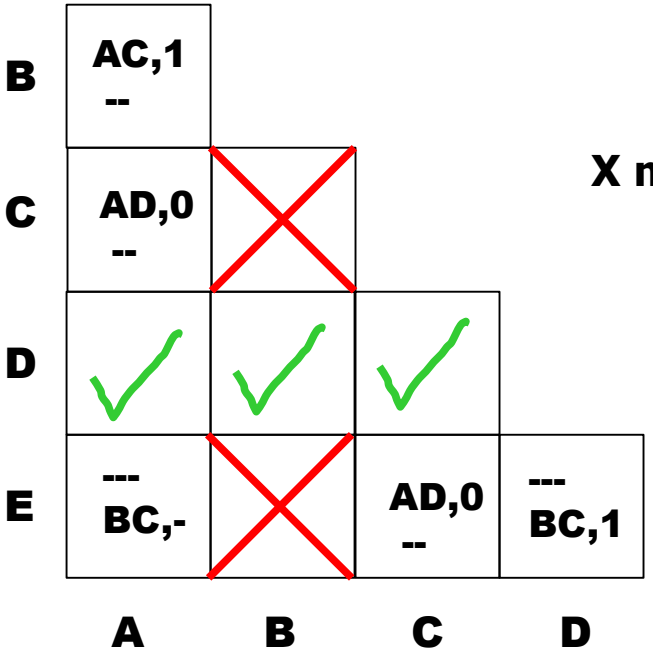
No need to go any further

Flow table

Implication Chart First pass

State Minimization – Implication Chart

Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –



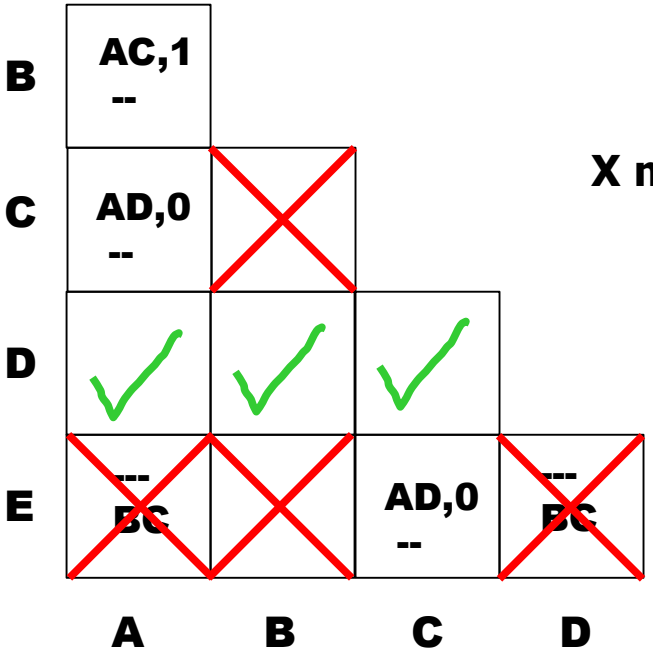
X marks output conflicts

Flow table

Implication Chart
First pass

State Minimization – Implication Chart

Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –



X marks output conflicts

Flow table

Implication Chart

Second Pass Transitive Closure

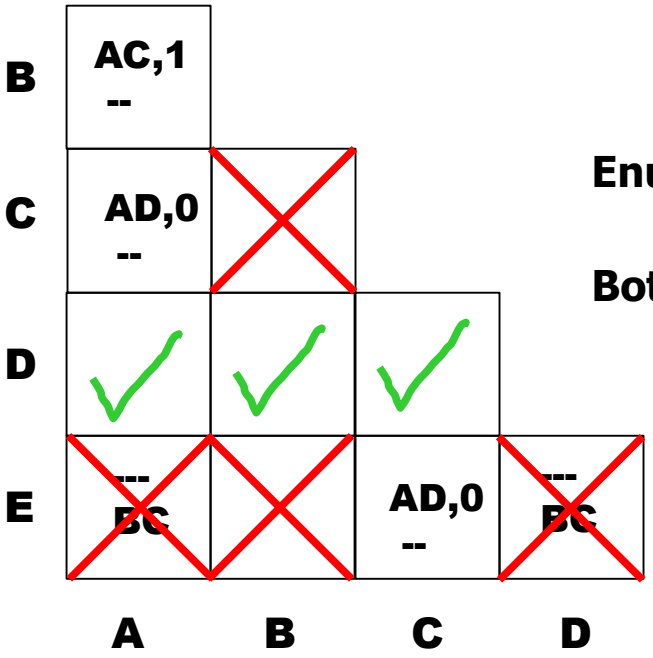
Is any checked boxes a next state?

If true, mark that box as well.

In this case, we have to check for BC and BE

State Minimization – Compatible Pairs

Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –



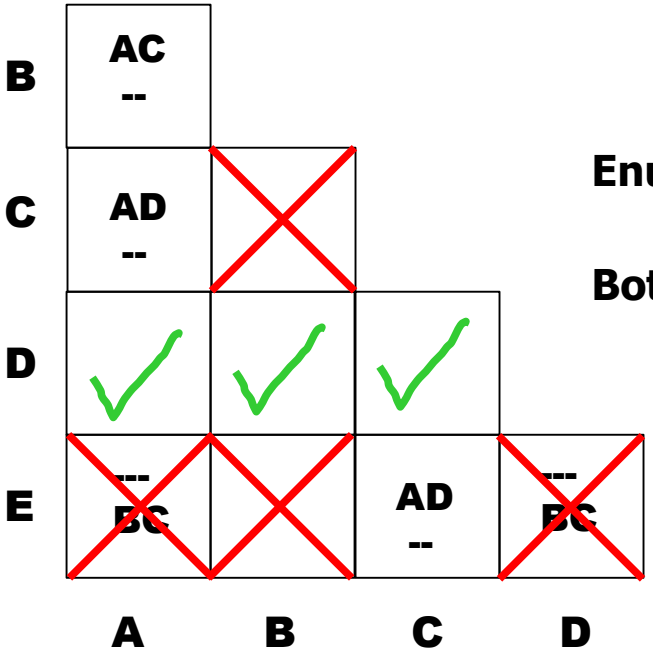
Enumerate Chart
Right to Left
Bottom to Top

Flow table

From Column D, nothing compatible
From Column C, we have CE and CD
From Column B, we get BD
From Column A, we get AD, AC, and AB

State Minimization – Incompatible Pairs

Present state	Next state	
	$X=0$	$X=1$
A	A, –	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –



Enumerate Chart
Right to Left
Bottom to Top

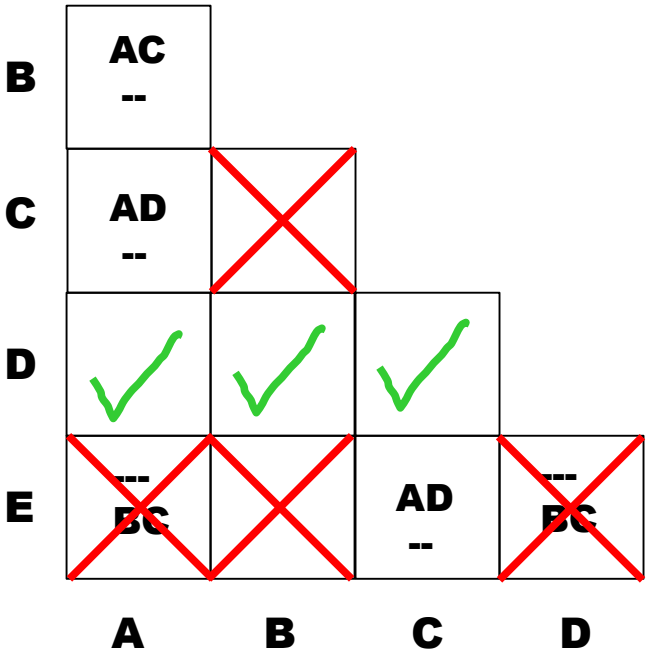
Flow table

From Column D, we find DE
From Column C, everything compatible
From Column B, we find BE, and BC
From Column A, we get AE

State Minimization - Result

Present state	Next state	
	$X=0$	$X=1$
A	A, –	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Flow table



Implication Chart

D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)

Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)

Incompatible Pairs

Cliques

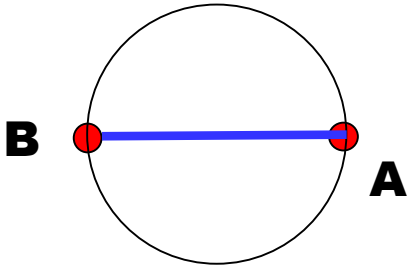
Graph Representation –

Fully connected graph

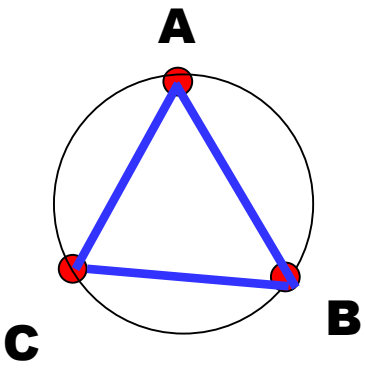
Chord Graphs

A

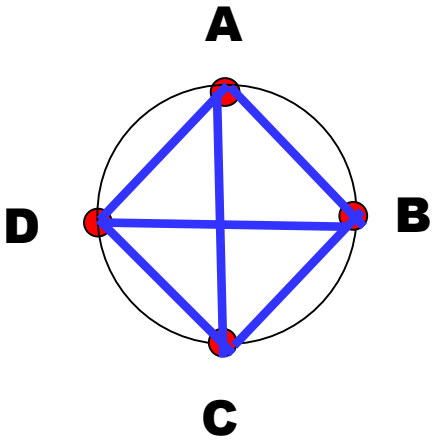
1-clique



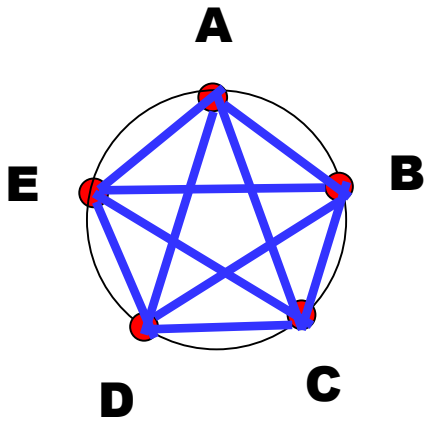
2-clique



3-clique



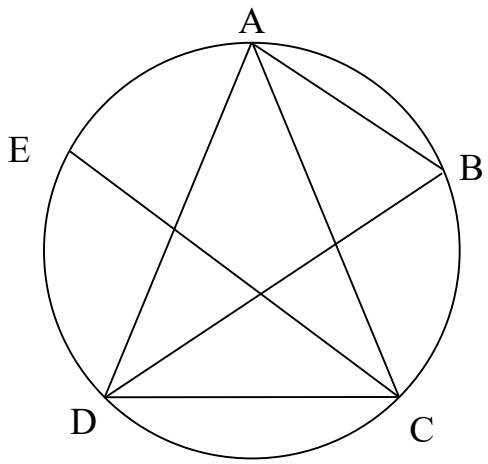
4-clique



5-clique

State Minimization – Compatible Cliques

Graph Representation – build chord graph

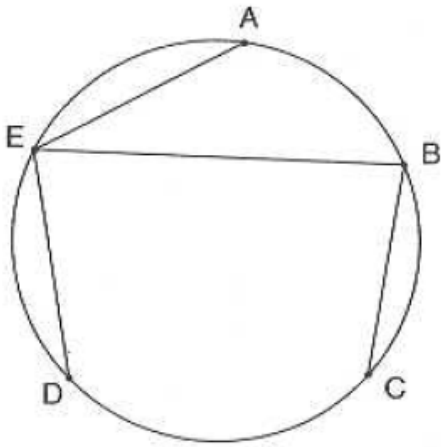


From Column D, nothing compatible
From Column C, we have CE and CD
From Column B, we get BD
From Column A, we get AD, AC, and AB

State Minimization – Incompatible Cliques

Graph Representation – build chord graph

From Column D, we find DE
From Column C, everything compatible
From Column B, we find BE, and BC
From Column A, we get AE



Incompatibles

(DE) (BE) (BC) (AE)

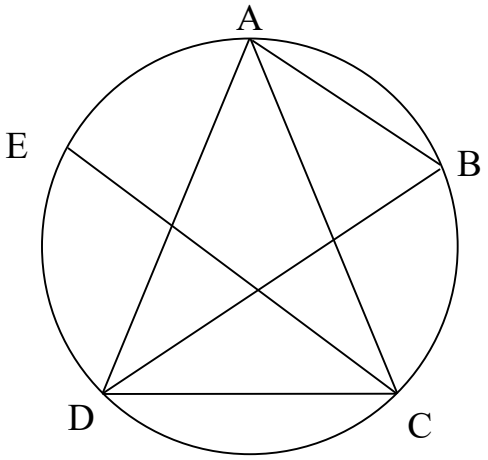
State Minimization – Trial and Error

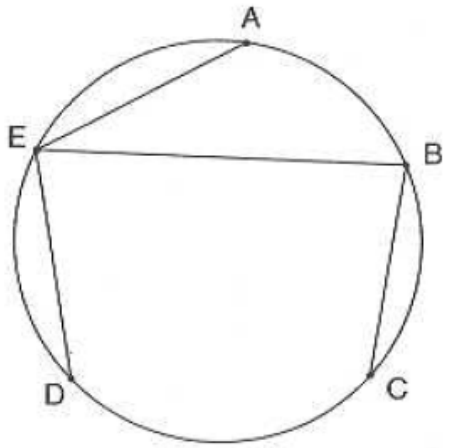
Graph Representation

D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)
Compatible Pairs

Cummulative Set here!

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)
Incompatible Pairs





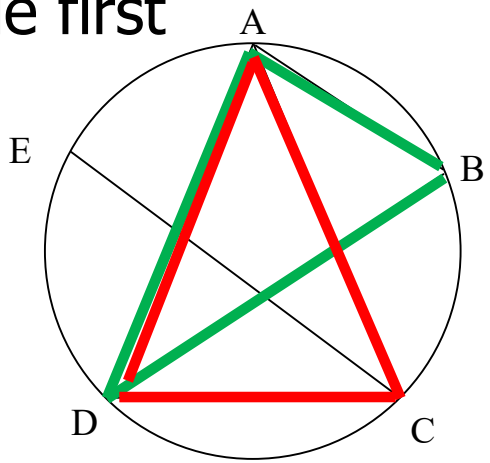
Incompatibles
(DE) (BE) (BC) (AE)

- Note carefully that three conditions must be satisfied:
  1. Completeness: all states must be covered
  2. Minimality: the smallest number of compatibles required is chosen.
  3. Consistency: Next states of each selected maximal compatibles (cliques) is contained by another maximal compatible within the selected set.

Order is trial and error

Greedy Assignment

- Choose largest clique first

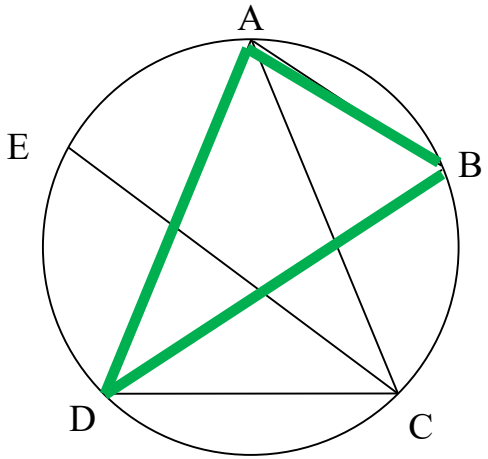


Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Two choices

Greedy Assignment

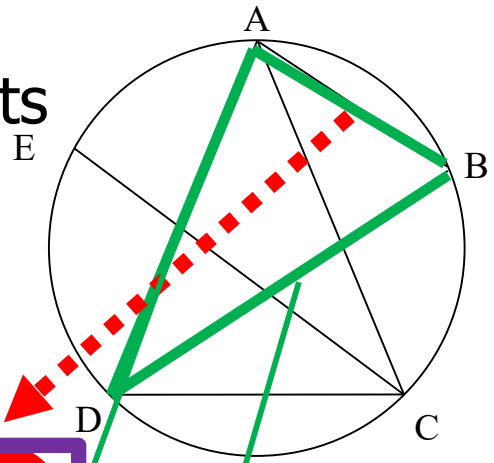
- Go green arbitrarily



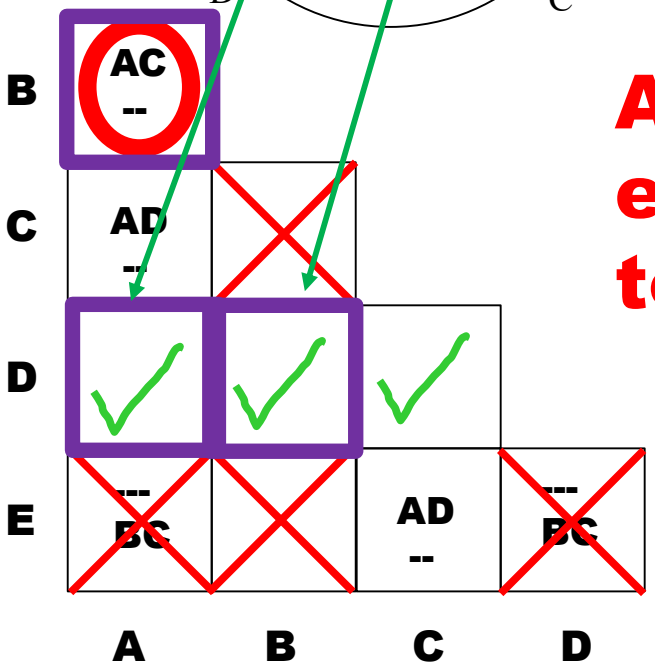
Present state	Next state	
	$X=0$	$X=1$
A	A, –	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Greedy Assignment

- Go green arbitrarily
- Check the constraints



Present state	Next state	
	$X=0$	$X=1$
A	A,–	B,–
B	C, 1	B,–
C	D, 0	–,–
D	–,–	B, 1
E	A, 0	C,–

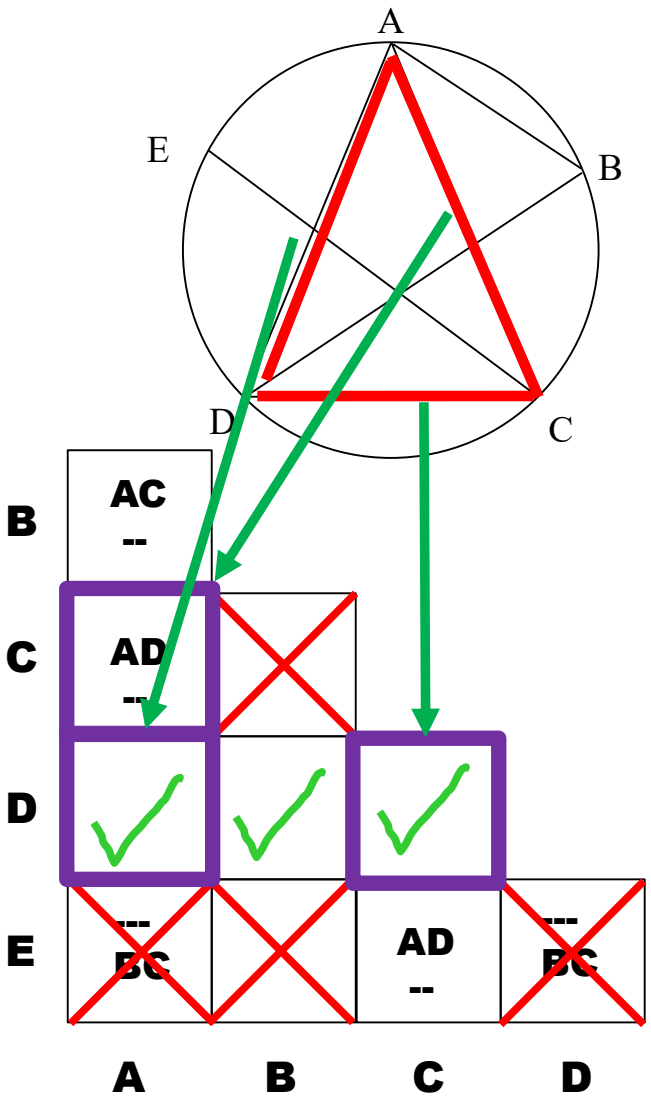


A is not equivalent to C here!

Greedy Assignment

- Try red clique

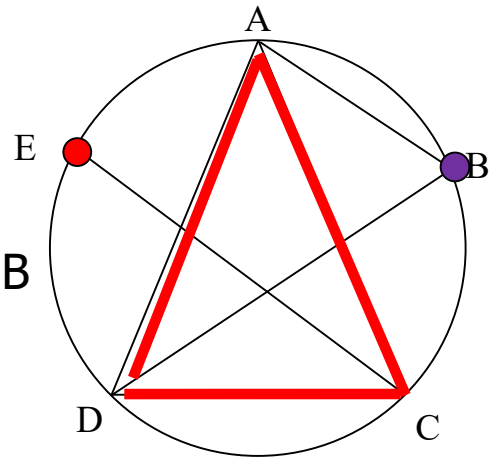
Present state	Next state	
	$X=0$	$X=1$
A	A,–	B,–
B	C, 1	B,–
C	D, 0	–,–
D	–,–	B, 1
E	A, 0	C,–



AC is equivalent if A=D. Yes!! Everything consistent!

Greedy Assignment

- Remove red clique



- No way to combine E and B

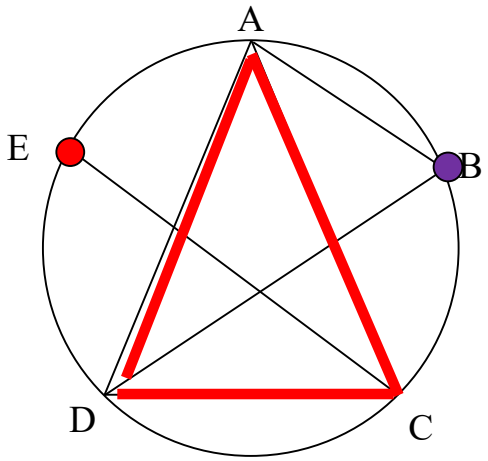
Present state	Next state	
	$X=0$	$X=1$
A	A,–	B,–
B	C, 1	B,–
C	D, 0	–,–
D	–,–	B, 1
E	A, 0	C,–

Closure Table

Maximal Compatibles	Set of Next States	
	X=0	X=1
ACD	AD,0	B,1
B	C,1	B,-
E	A,0	C,-

Greedy Assignment

- Now rename states:
  - ACD = P
  - B = Q
  - E = R



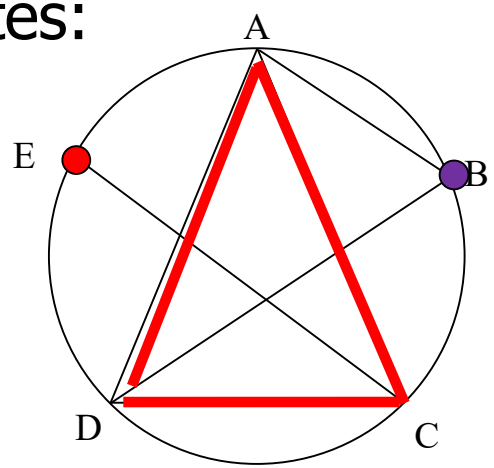
Present state	Next state	
	$X=0$	$X=1$
A	A,–	B,–
B	C, 1	B,–
C	D, 0	–,–
D	–,–	B, 1
E	A, 0	C,–

Closure Table

Maximal Compatibles	Set of Next States	
	X=0	X=1
~~ACD~~ P	~~AD~~ P,0	~~B~~ Q,1
~~B~~ Q	~~C~~ P,1	~~B~~ Q,-
~~E~~ R	~~A~~ P,0	~~C~~ P,-

Greedy Assignment

- Now find stable states:



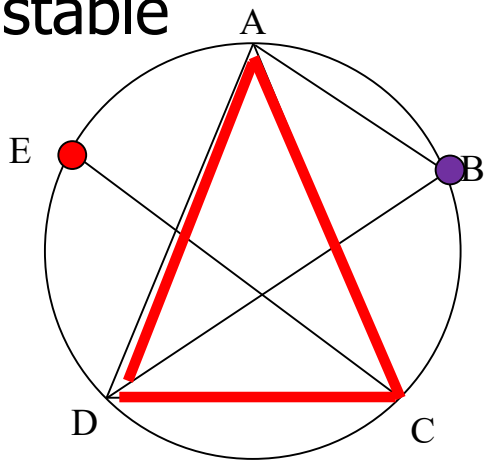
Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Closure Table

Maximal Compatibles	Set of Next States	
	X=0	X=1
~~ACD~~ P	~~AD~~ P,0	~~B~~ Q,1
~~B~~ Q	~~C~~ P,1	~~B~~ Q,-
~~E~~ R	~~A~~ P,0	~~C~~ P,-

Greedy Assignment

- Strike rows with no stable states:



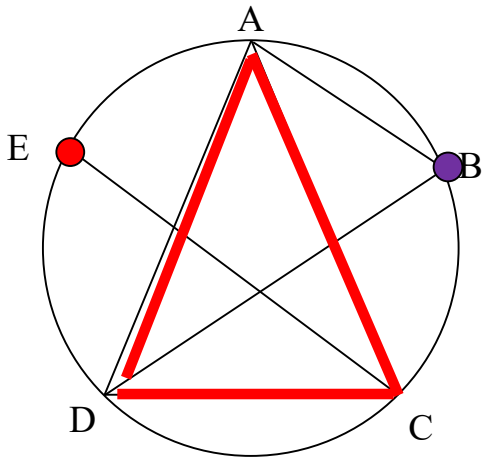
Present state	Next state	
	$X=0$	$X=1$
A	A,–	B,–
B	C, 1	B,–
C	D, 0	–,–
D	–,–	B, 1
E	A, 0	C,–

Closure Table

Maximal Compatibles	Set of Next States	
	X=0	X=1
~~ACD~~ P	~~AD~~ P,0	~~B~~ Q,1
~~B~~ Q	~~E~~ P,1	~~B~~ Q,-
~~E~~ R	~~A~~ P,0	~~E~~ P,-

Greedy Assignment

- Final table



Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Closure Table

Maximal Compatibles	Set of Next States	
	X=0	X=1
~~ACD~~ P	~~AD~~ P,0	~~B~~ Q,1
~~B~~ Q	~~C~~ P,1	~~B~~ Q,-

Greedy Assignment

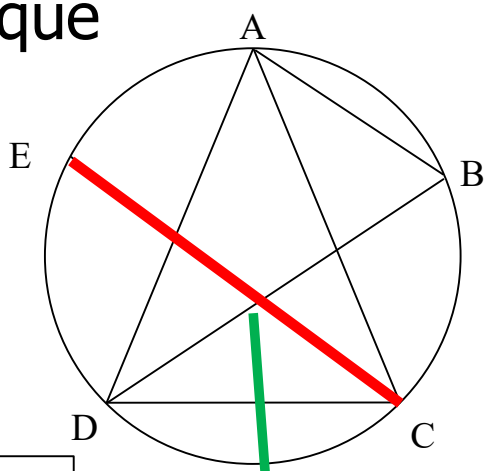
- Behavior

Closure Table

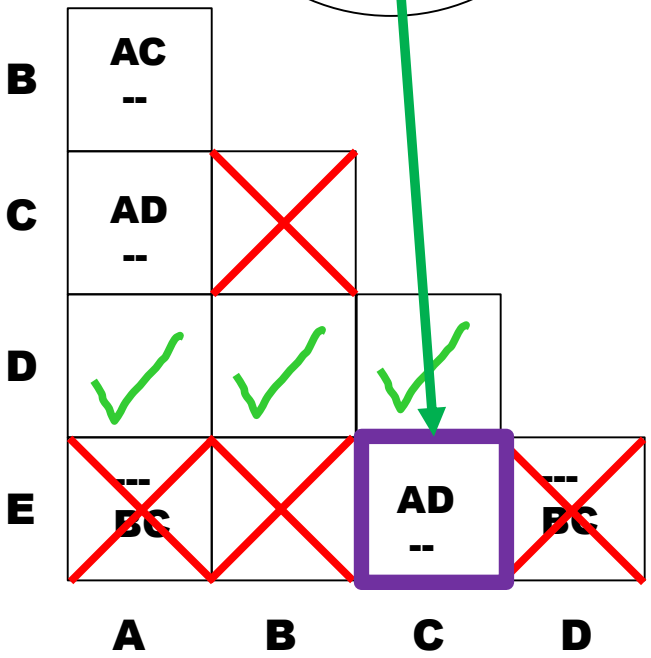
Maximal Compatibles	Set of Next States	
	X=0	X=1
~~ACD~~ P	~~AD~~ P,0	~~B~~ Q,1
~~B~~ Q	~~C~~ P,1	~~B~~ Q,-

Non Greedy Assignment

- Try non-maximal clique
- Pick CE



Present state	Next state	
	$X=0$	$X=1$
A	A, –	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

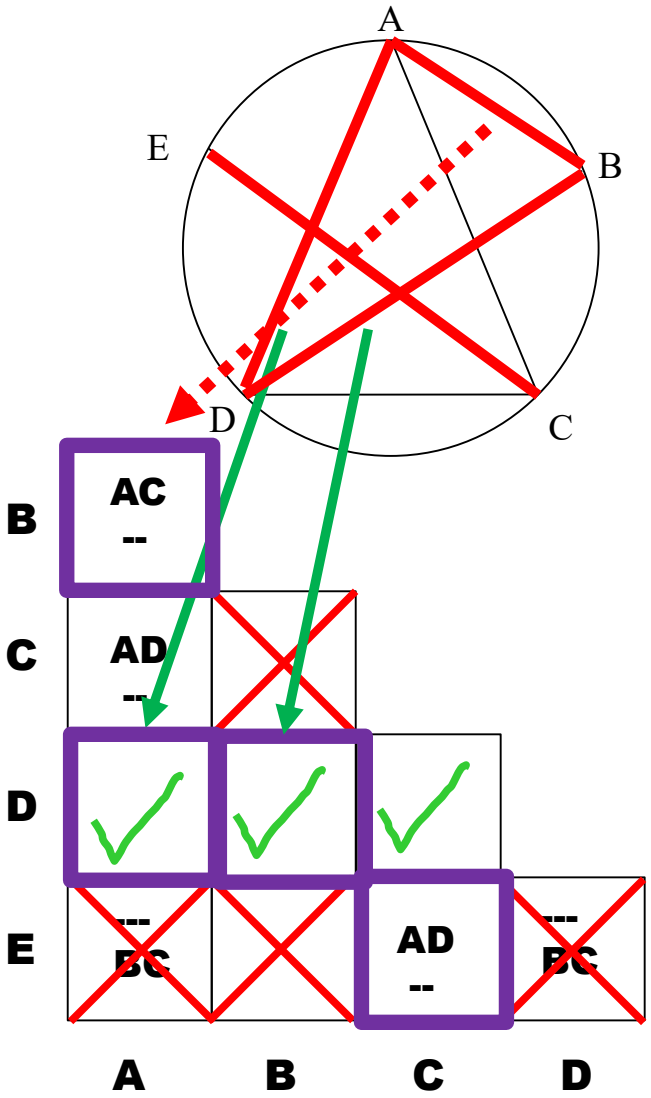


CE is equivalent if A=D. Implies we should pick ADB or AD next

Non Greedy Assignment

- Try ABD

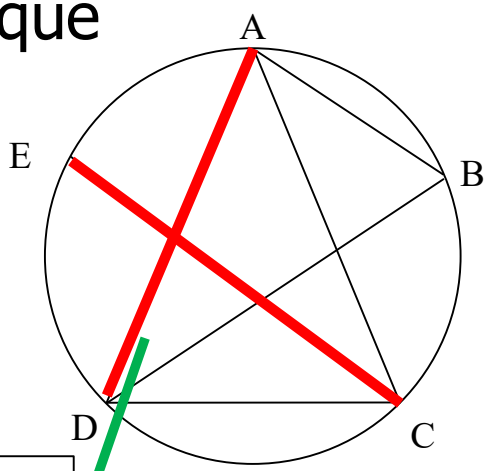
Present state	Next state	
	$X=0$	$X=1$
A	A,–	B,–
B	C, 1	B,–
C	D, 0	–,–
D	–,–	B, 1
E	A, 0	C,–



No! AB requires A=C and not possible

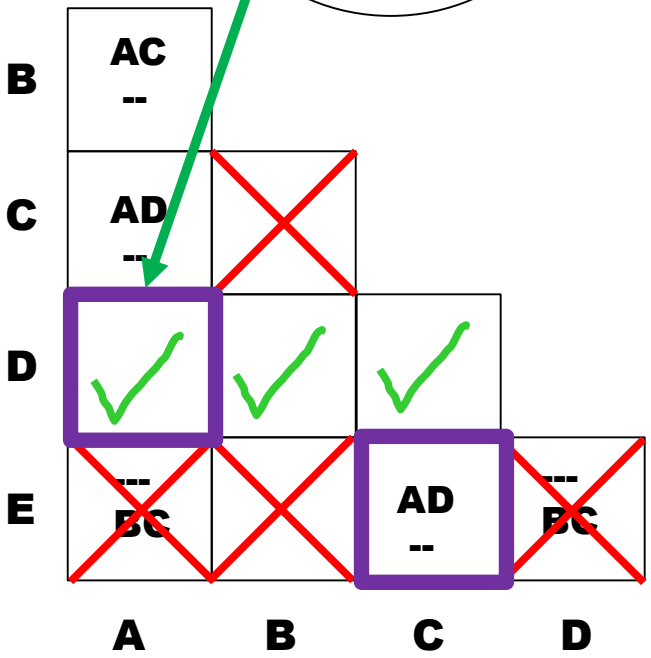
Non Greedy Assignment

- Try non-maximal clique
- Pick CE



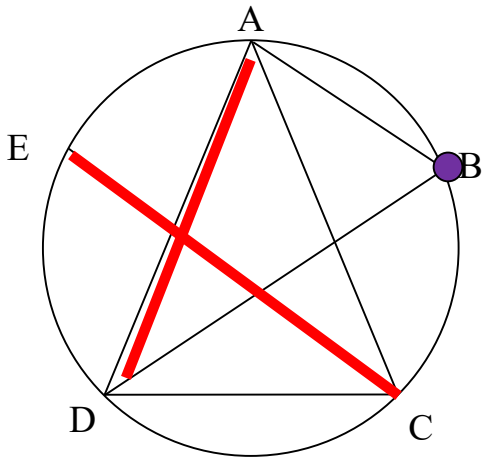
OK
Left with B

Present state	Next state	
	$X=0$	$X=1$
A	A, –	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –



Non Greedy Assignment

- Build table



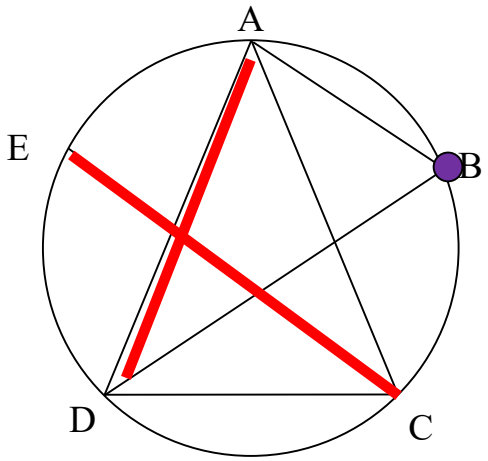
Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Closure Table

Maximal Compatibles	Set of Next States	
	X=0	X=1
CE	AD,0	C,-
AD	A,-	B,1
B	C,1	B,-

Non Greedy Assignment

- Rename states
- CE = P
- AD = Q
- B = R



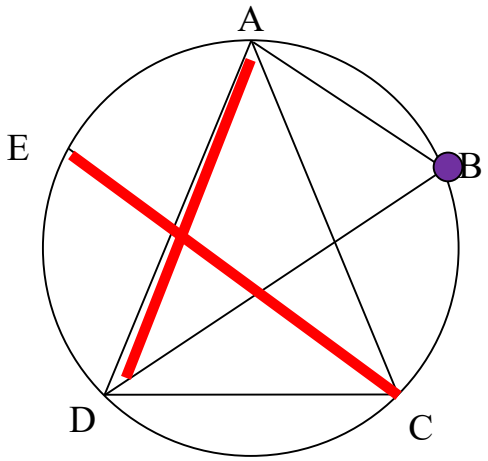
Present state	Next state	
	$X=0$	$X=1$
A	A,–	B,–
B	C, 1	B,–
C	D, 0	–,–
D	–,–	B, 1
E	A, 0	C,–

Closure Table

Maximal Compatibles	Set of Next States	
	X=0	X=1
~~CE~~ P	~~AD~~ Q,0	~~C~~ P,-
~~AD~~ Q	~~A~~ Q,-	~~B~~ R,1
~~B~~ R	~~C~~ P,1	~~B~~ R,-

Non Greedy Assignment

- Find stable states



Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Closure Table

Maximal Compatibles	Set of Next States	
	X=0	X=1
~~CE~~ P	~~AD~~ Q,0	~~C~~ P,-
~~AD~~ Q	~~A~~ Q,-	~~B~~ R,1
~~B~~ R	~~C~~ P,1	~~B~~ R,-

State Minimization – Try again Pick CE

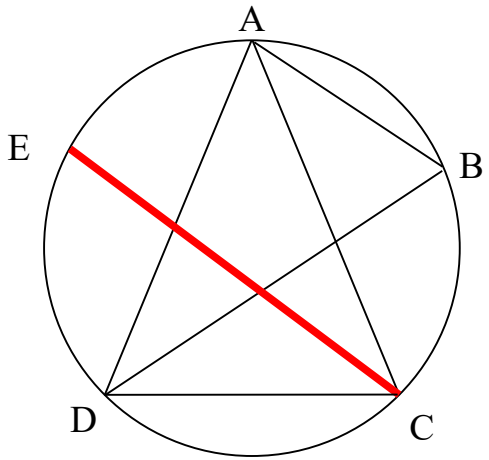
Graph Representation

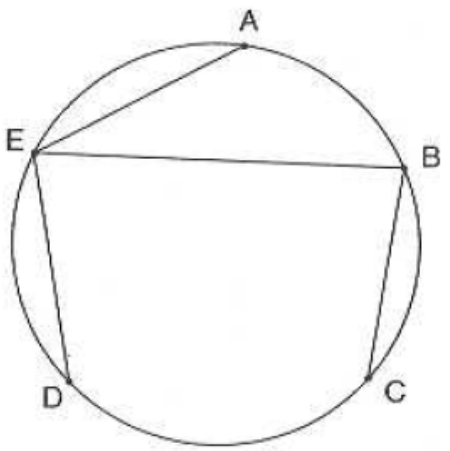
D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)

Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)

Incompatible Pairs





Warning: Book's method

Incompatibles

(DE) (BE) (BC) (AE)

Present State	Next State	
	X=0	X=1
A	A, -	B, -
B	C, 1	B, -
C	D, 0	- , -
D	-, -	B, 1
E	A, 0	C, -

Flow table

Combine CE

Present State	Next State	
	X=0	X=1
A	A, -	B, -
B	C, 1	B, -
CE	AD, 0	C , -
D	-, -	B, 1

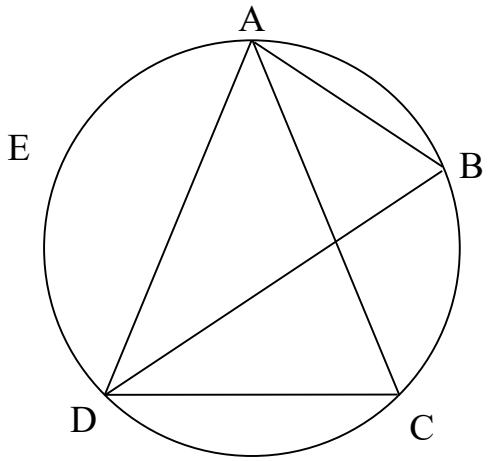
Closure Table

State Minimization – Update Graph

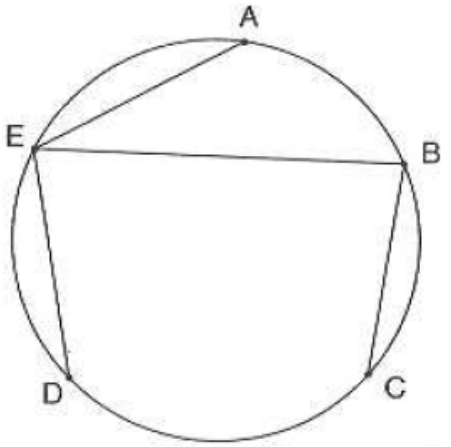
Graph Representation

D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)
Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)
Incompatible Pairs



Mark or delete edge from graph



Incompatibles
(DE) (BE) (BC) (AE)

Present State	Next State	
	X=0	X=1
A B C D E	A, - C, 1 D, 0 -, - A, 0	B, - B, - - , - B, 1 C, -

Flow table

Combine CE

Present State	Next State	
	X=0	X=1
A B CE D	A, - C, 1 AD, 0 -, -	B, - B, - C , - B, 1

Closure Table

State Minimization – Pick ABD

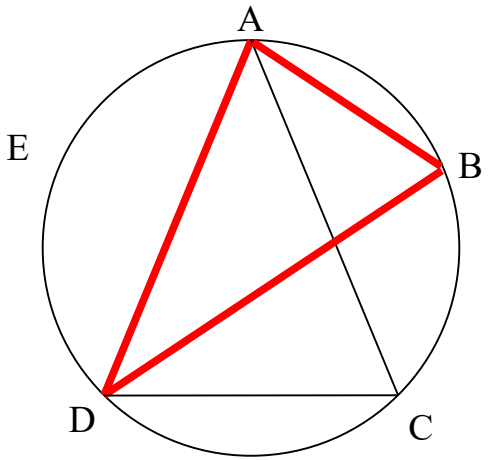
Graph Representation

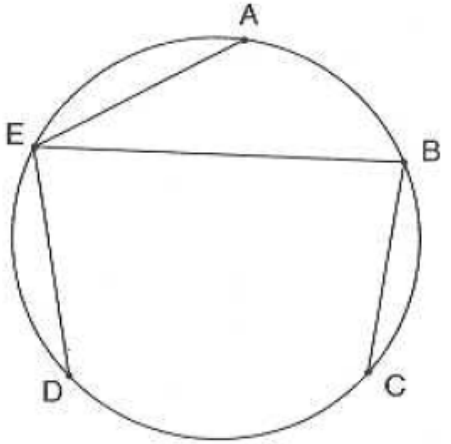
D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)

Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)

Incompatible Pairs





Incompatibles

(DE) (BE) (BC) (AE)

Present State	Next State	
	X=0	X=1
A B CE D	A, - C, 1 AD, 0 -, -	B, - B, - C , - B, 1

Flow table

Combine ABD

→

Present State	Next State	
	X=0	X=1
ABD CE	AC, 1 AD, 0	B, 1 C , -

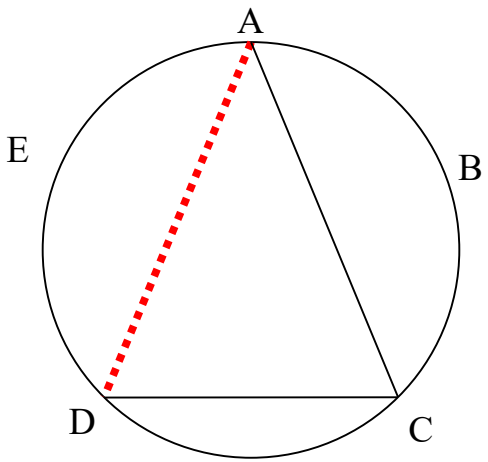
Closure Table

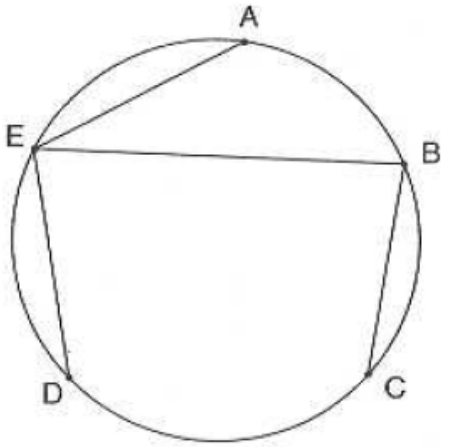
State Minimization – Update Graph

Graph Representation

D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)
Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)
Incompatible Pairs





We would be done but how do we represent the next state AC here???

Incompatibles
(DE) (BE) (BC) (AE)

Present State	Next State	
	X=0	X=1
A B CE D	A, - C, 1 AD, 0 -, -	B, - B, - C , - B, 1

Combine ABD

Present State	Next State	
	X=0	X=1
ABD CE	AC, 1 AD, 0	B, 1 C , -

We would be done But how do we represent the

You can see this in the graph representation. Edge AD is missing from the clique in graph.

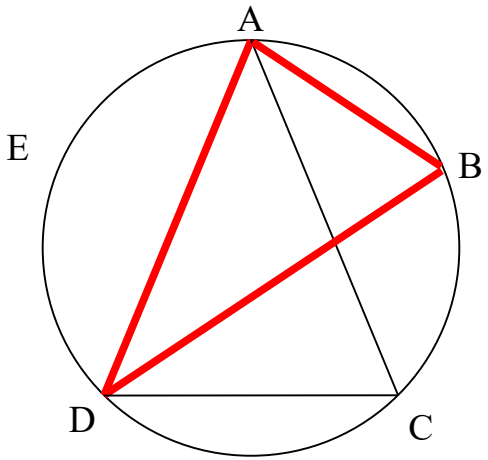
Closure Table

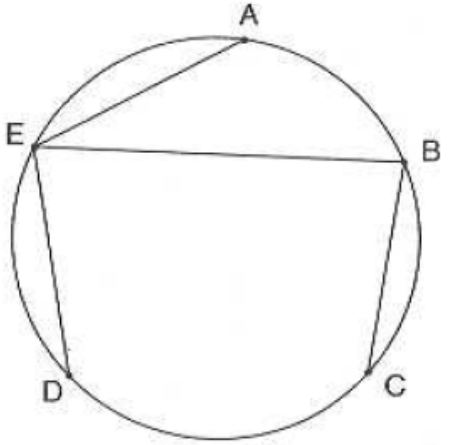
State Minimization – Pick ABD redo

Graph Representation

D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)
Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)
Incompatible Pairs





Incompatibles
(DE) (BE) (BC) (AE)

Present State	Next State	
	X=0	X=1
A B CE D	A, - C, 1 AD, 0 -, -	B, - B, - C , - B, 1

Combine ABD →

Present State	Next State	
	X=0	X=1
A ABD CE C D	A, - AC, 1 AD, 0 D, 0 -, -	B, - B, 1 C , - - , - B, 1

Don't remove rows until the end

Closure Table

State Minimization – Update graph

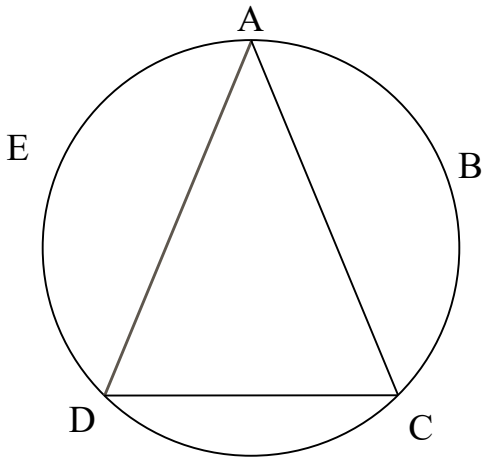
Graph Representation

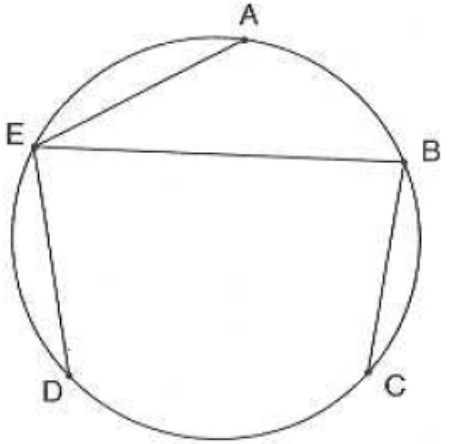
D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)

Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)

Incompatible Pairs





Incompatibles

(DE) (BE) (BC) (AE)

Present State	Next State	
	X=0	X=1
A ABD CE C D	A, - AC, 1 AD, 0 D, 0 -, -	B, - B, 1 C , - - , - B, 1

Combine ABD

Present State	Next State	
	X=0	X=1
A ABD CE C D	A, - AC, 1 AD, 0 D, 0 -, -	B, - B, 1 C , - - , - B, 1

Delete only edges not in common with the remaining cliques

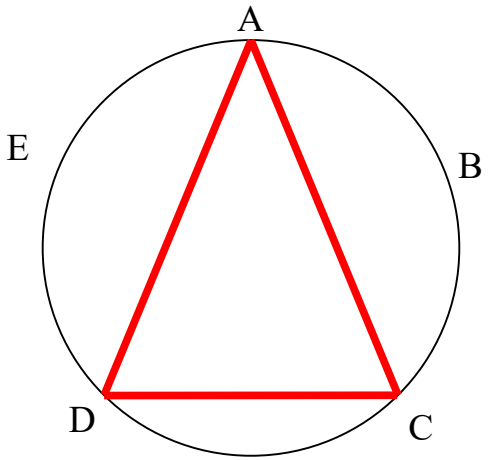
Closure Table

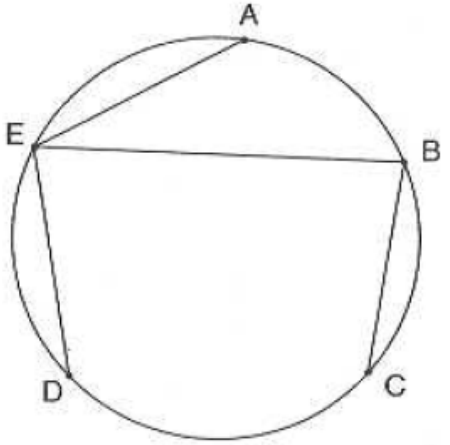
State Minimization – Pick ACD

Graph Representation

D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)
Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)
Incompatible Pairs





Incompatibles
(DE) (BE) (BC) (AE)

Present State	Next State	
	X=0	X=1
A ABD CE C D	A, - AC, 1 AD, 0 D, 0 -, -	B, - B, 1 C , - - , - B, 1

Combine ACD →

Present State	Next State	
	X=0	X=1
A ABD ACD CE C D	A, - AC, 1 AD, 0 AD, 0 D, 0 -, -	B, - B, 1 B, 1 C , - - , - B, 1

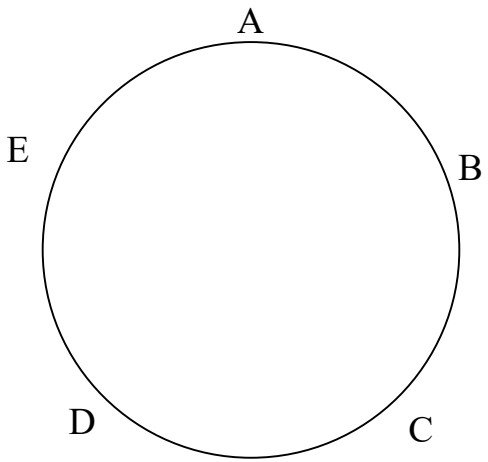
Closure Table

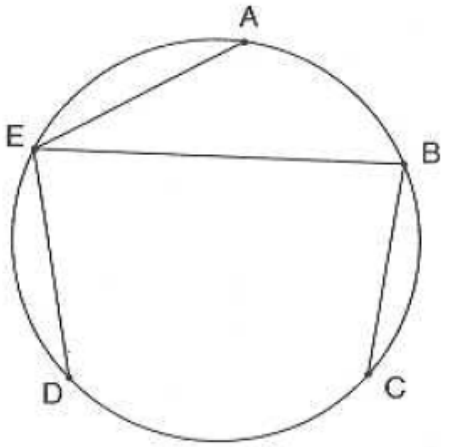
State Minimization – Update Graph

Graph Representation

D –
C (CE) (CD)
B (CE) (CD) (BD)
A (CE) (CD) (BD) (AD) (AC) (AB)
Compatible Pairs

D (DE)
C (DE)
B (DE) (BE) (BC)
A (DE) (BE) (BC) (AE)
Incompatible Pairs





Incompatibles
(DE) (BE) (BC) (AE)

Present State	Next State	
	X=0	X=1
A ABD CE C D	A, - AC, 1 AD, 0 D, 0 -, -	B, - B, 1 C , - - , - B, 1

Combine ACD

→

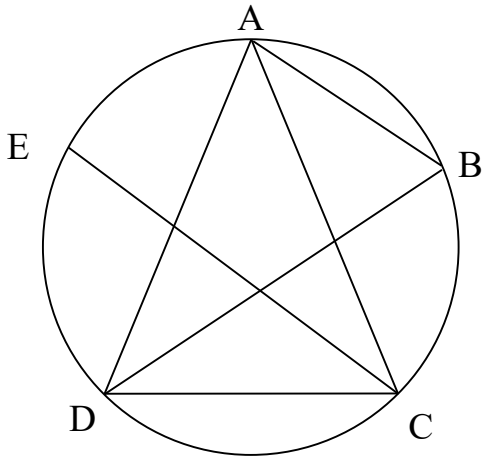
Present State	Next State	
	X=0	X=1
~~A~~ ABD ACD CE ~~C~~ ~~D~~	~~A, -~~ AC, 1 AD, 0 AD, 0 ~~D, 0~~ ~~-, -~~	~~B, -~~ B, 1 B, 1 C , - ~~- , -~~ ~~B, 1~~

Closure Table

No chords left!
Remove covered rows

Conditions for State Reduction (cont.)

- Rename states
- P = ABD
- Q= ACD
- R = CE



Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Maximal compatibles	Sets of next states	
	$X=0$	$X=1$
ABD	AC,1	B,1
ACD	AD,0	B,1
CE	AD,0	C,--

Closure Table

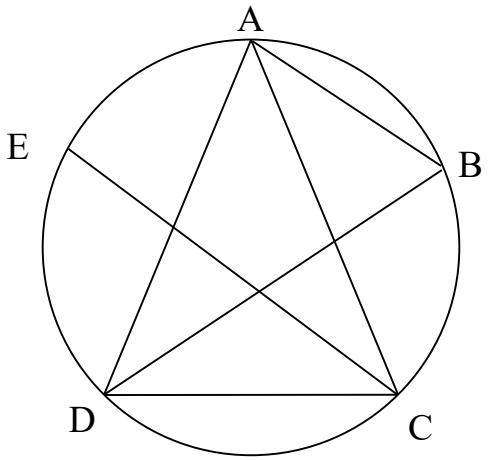
How do we know how to name the next state? Renamed state must cover or contain original state

Present state	Next state	
	$X=0$	$X=1$
P	Q,1	P,1
Q	Q,0	P,1
R	Q,0	R,–

Reduced Table

Conditions for State Reduction (cont.)

- Considering condition 3:



Present state	Next state	
	$X=0$	$X=1$
A	A,–	B, –
B	C, 1	B, –
C	D, 0	–, –
D	–, –	B, 1
E	A, 0	C, –

Maximal compatibles	Sets of next states	
	$X=0$	$X=1$
ABD	AC,**1**	B,**1**
ACD	AD,**0**	B,**1**
CE	AD,**0**	C,--

Closure Table

- **Can be Q or P because AD is contained in (ABD) as well as (ACD)**

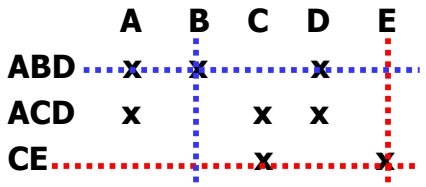
Present state	Next state	
	$X=0$	$X=1$
P	Q,1	P,1
Q	Q,0	P,1
R	Q,0	R,–

Reduced Table

Can we reduce more?

- Note carefully that three conditions must be satisfied:
  1. Completeness: all states must be covered
  2. Minimality: the smallest number of compatibles required is chosen.
  3. Consistency: Next states of each selected maximal compatibles (cliques) is contained by another maximal compatible within the selected set.

If we ignore condition 3:



Present State	Next State	
	X=0	X=1
ABD=P	AC,1	B,1
CE=R	AD,0	C, --

Present State	Next State	
	X=0	X=1
P	?,1	P,1
R	P,0	R, --

- Which one?
  - A is contained in P
  - C is contained in R

Requirements for Proper Operation

1. Only one input change at a time (fundamental mode assumption).
2. The state assignment (transition table) is free of critical race
3. The excitation logic is hazard free
4. The minimum propagation delay is satisfied
5. The maximum propagation delay is satisfied

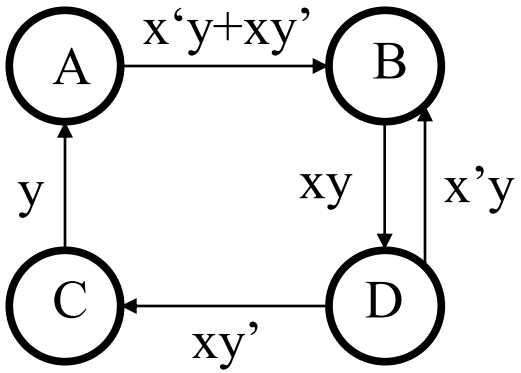
Race-Free State Code Assignment

- Example of a flow table

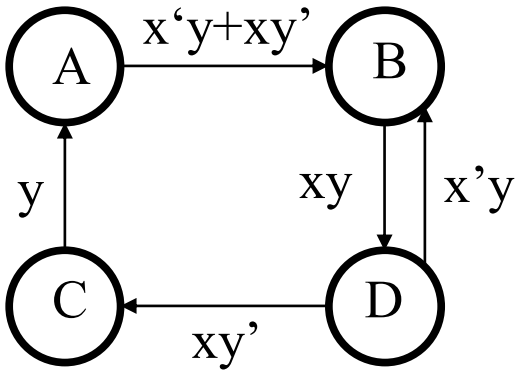
	xy			
S	00	01	11	10
A	(A)	B	(A)	B
B	(B)	(B)	D	(B)
C	(C)	A	A	(C)
D	(D)	B	(D)	C
	S*			

- State Diagram

Contains information of flow table in graphical form



Race-Free State Code Assignment



- Need to assign each of the states an unique binary pattern

State Assignment Guidelines

- For m states, we need at least r bits such that

$$2^{r-1} < m \le 2^r \quad or \quad r = \lceil \log_2 m \rceil$$

- Number of ways to select m codes out of 2^r codes

$$\binom{2^r}{m} = \frac{2^r!}{m!(2^r - m)!}$$

- Number of ways to assign m selected codes to m state is m!.
- Overall, number of ways to assign r-bit codes to m states

$$\frac{2^r!}{m!(2^r - m)!} \cdot m! = \frac{2^r!}{(2^r - m)!}$$

- Only some of these will be race-free. For previous example, m=4, r=2, we have total of 24 assignments 8 of which are race-free and only 3 unique

State Assignment

- Not all state assignments are unique or distinct
- Complementing a bit does not increase complexity of logic
- Similarly, swapping columns of bits does not change circuit complexity
- For two state circuits, only one unique assignment
- For three and four state circuits, there are only 3 unique assignments out of 24
- Number of unique state assignments is given by:

$$m_{unique} = \frac{(2^r - 1)!}{\left((2^r - m)!\, r!\right)}$$

State Assignment

- Number of unique state assignments is given by:

$$m_{unique} = \frac{(2^r - 1)!}{\left((2^r - m)!\, r!\right)}$$

States	Flip-flops	Possible states	Unique states
2	1	2	1
3	2	24	3
4	2	24	3
5	3	6,720	140
6	3	20,160	420
7	3	40,320	840
8	3	40,320	840
9	4	415×10^7	10,810,800
10	4	219×10^8	75,675,600

Adjacent States

- Each bit of the encode state may be represented by a flip-flop bit ε_i
- An encode of the state forms a vector of length *r*:

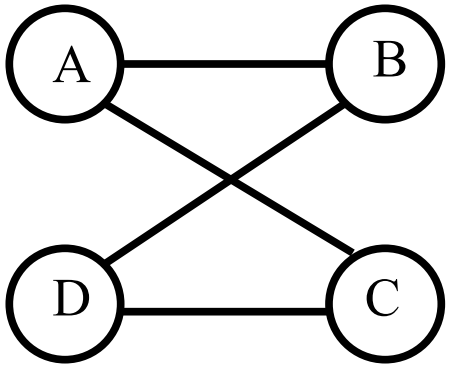
$$\varepsilon_{r-1}\varepsilon_{r-2}\cdots\varepsilon_1\varepsilon_0$$

- Two state vectors are said to be ***adjacent*** if they differ in only one bit position
- Two states, A and B, are said to be adjacent if their state vectors are adjacent

Adjacency Graph

- Edge between adjacent states
- Let

State	ε_1	ε_0
A	0	0
B	0	1
C	1	0
D	1	1



This is not the state transition graph!

Unique assignments of 4 states

A	B	C	D
B	C	D	A
C	D	A	B
D	A	B	C
D	C	B	A
C	B	A	D
B	A	D	C
A	D	C	B

A	B	D	C
B	D	C	A
D	C	A	B
C	A	B	D
C	D	B	A
D	B	A	C
B	A	C	D
A	C	D	B

A	D	B	C
D	B	C	A
B	C	A	D
C	A	D	B
C	B	D	A
B	D	A	C
D	A	C	B
A	C	B	D

Shift sequence by one bit to left and rotate in from right

Mirror image of first sequence around y axis

- Unique vectors are {ABCD, ABDC, and ADBC}
- Move D to unique position in ring sequence
- 24 total assignment
- 3 unique – only 3 possible unique rings
- Gray code (named after Frank Gray of Bell Labs 1947)

Gray code assignment

- Given previous encoding:

A	B	D	C
B	D	C	A
D	C	A	B
C	A	B	D
C	D	B	A
D	B	A	C
B	A	C	D
A	C	D	B

State	ε_1	ε_0
A	0	0
B	0	1
D	1	1
C	1	0

Races

- A *race* exists when two or more secondary variables must change when the circuit makes a transition from one stable state to another
- If the circuit reaches the correct stable state, it is *non-critical*
- If the circuit reaches an erroneous state, the race is said to be *critical*
- A circuit is said to have a *cycle* when it goes thru a unique sequence of unstable states

Transitions

- May go thru cycle

	x	
$\varepsilon_1\varepsilon_0$	**0**	**1**
00	10	00
01	01	11
11	01	10
10	11	00

- 01 -> 11 -> 10 -> 00
- 00 -> 10 -> 11 -> 01

Transitions

- Oscillation

	x	
$\varepsilon_1\varepsilon_0$	**0**	**1**
00	--	01
01	(01)	11
11	--	10
10	--	00

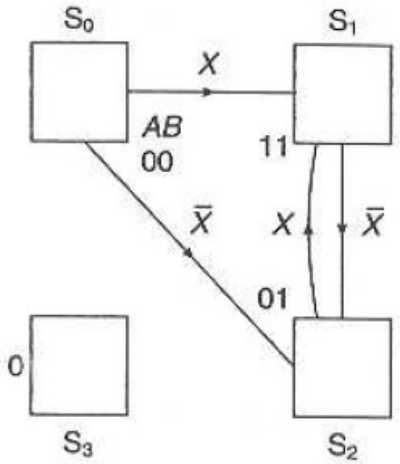
- (01) -> 11 -> 10 -> 00 -> 01 -> 11 -> …

Assignment Rules

- How do we avoid critical races and oscillations?

Example with Critical Race

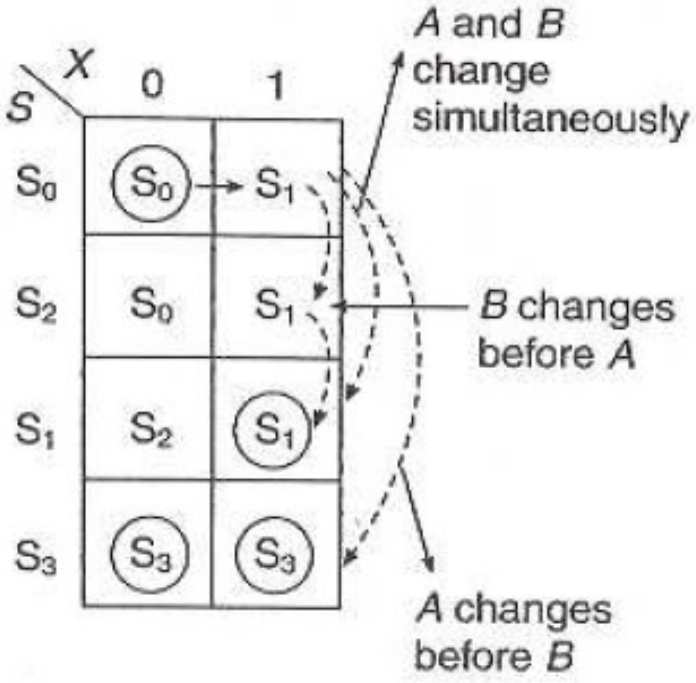
- State diagram



Confusing! Why?
Bad convention

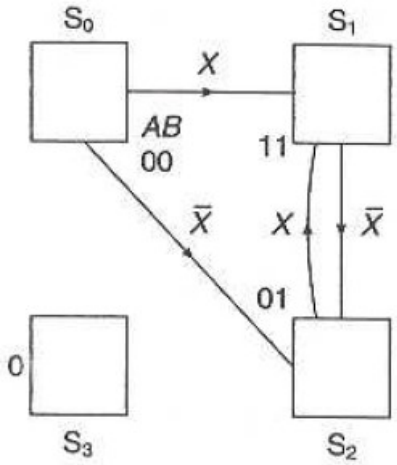
Bad engineering practice

- State table

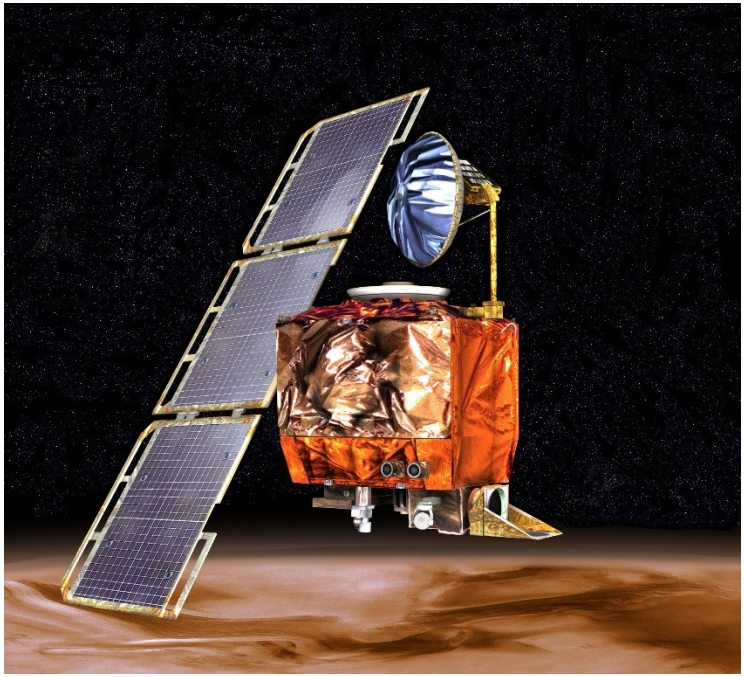


Example with Critical Race

- State diagram uses S0, S1, … Sn to denote state
- Implies that this is the encoding
- Look state S1 is encoded 11
- Bad convention



Without convention



Without convention

- In Sept 1999, NASA lost a $125 million Mars orbiter because a Lockheed Martin engineering team used English units of measurement while the agency's team used the more conventional metric system for a key spacecraft operation, according to a review finding released Thursday.
- The units mismatch prevented navigation information from transferring between the Mars Climate Orbiter spacecraft team in at Lockheed Martin in Denver and the flight team at NASAs Jet Propulsion Laboratory in Pasadena, California.
- In the end, the orbiter got a big headache after hitting the surface and all was lost.

Convention to minimize errors

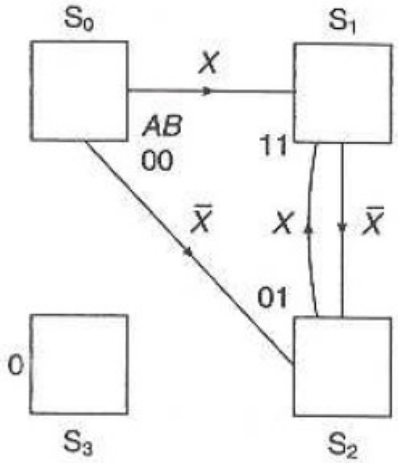
- Moore machine directly uses state encoding
- Natural biases
- How can we do this applied to states?

Convention in this class

- Perhaps, states all are capital alphabetic letters starting from A, ie, A, B, C, D, …
- All circuit inputs or literals should be italicized lower case to denote the difference
- Encoding flip-flops bits are denoted with script letter ε_i

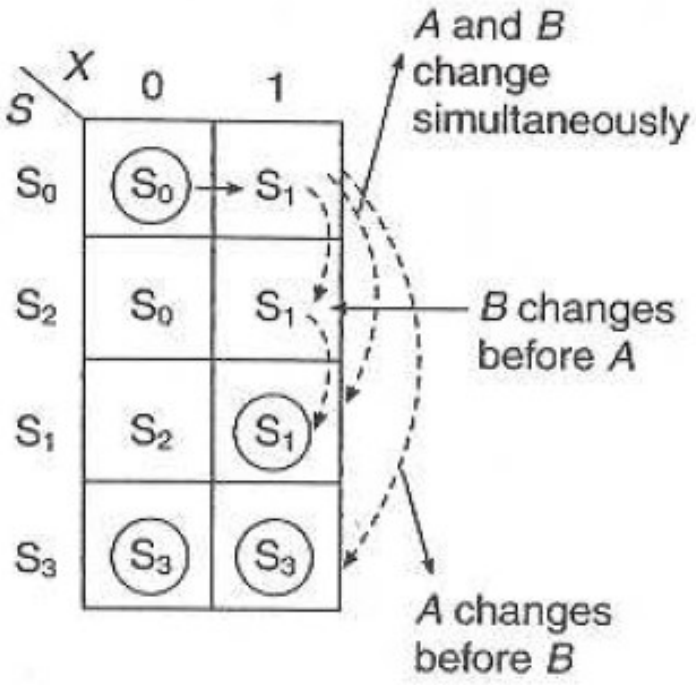
Example with Critical Race

- State diagram



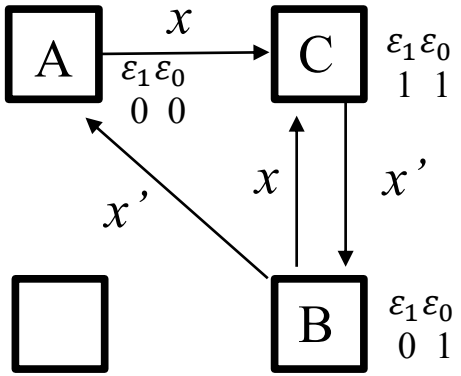
Now is it clear why it is confusing?

- State table



Example with Critical Race

- State diagram



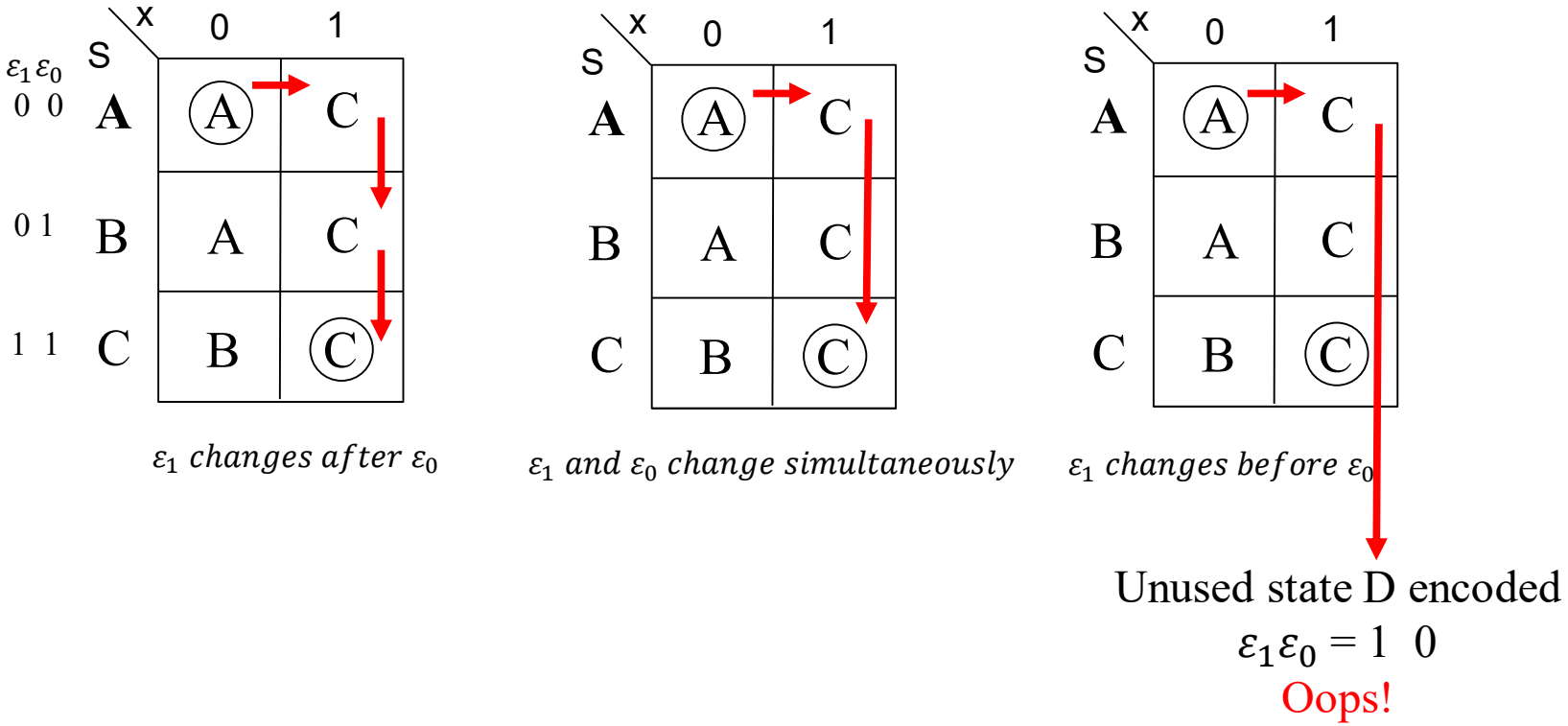
S \ x	0	1
A	(A)	C
B	A	C
C	B	(C)

- Unused encoding $\varepsilon_1\varepsilon_0$

  1 0

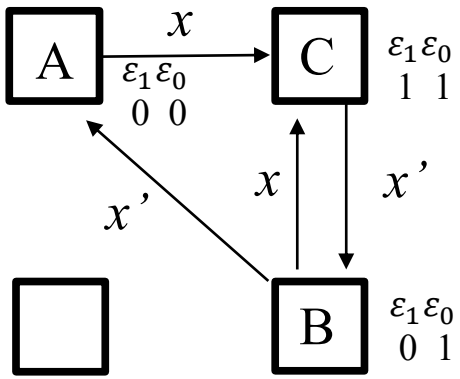
Example with Critical Race

- Now transition from state A with input change to *x*

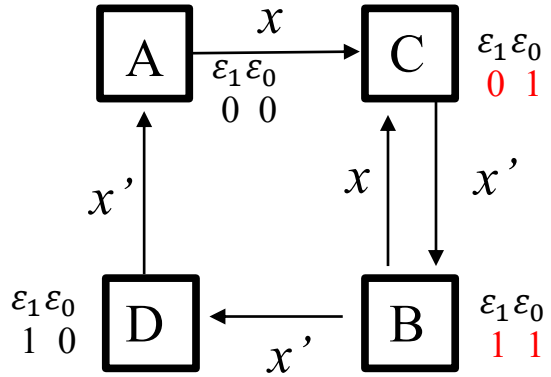


Adding Dummy States

- Original state diagram



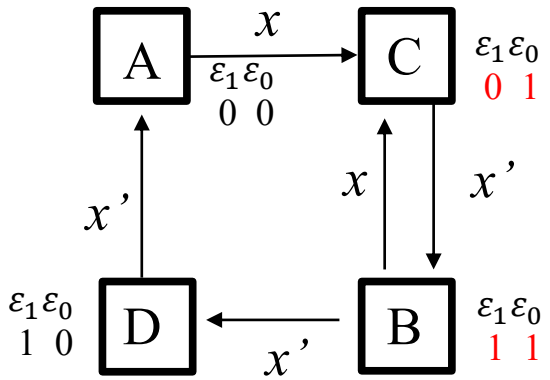
- Adding a dummy state to make it race-free and change encoding



New State Table

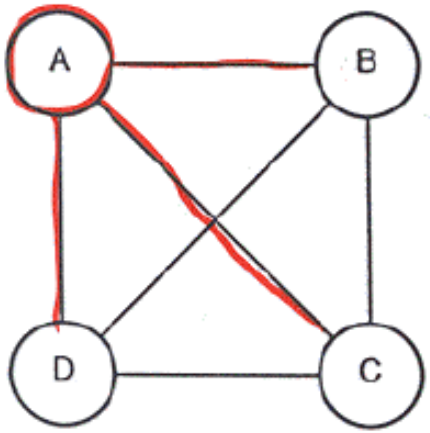
- Now table looks like this:

	S \ x	0	1
$\varepsilon_1\varepsilon_0$ 0 0	A	(A)	C
1 1	B	D	C
0 1	C	B	(C)
1 0	D	A	(D)



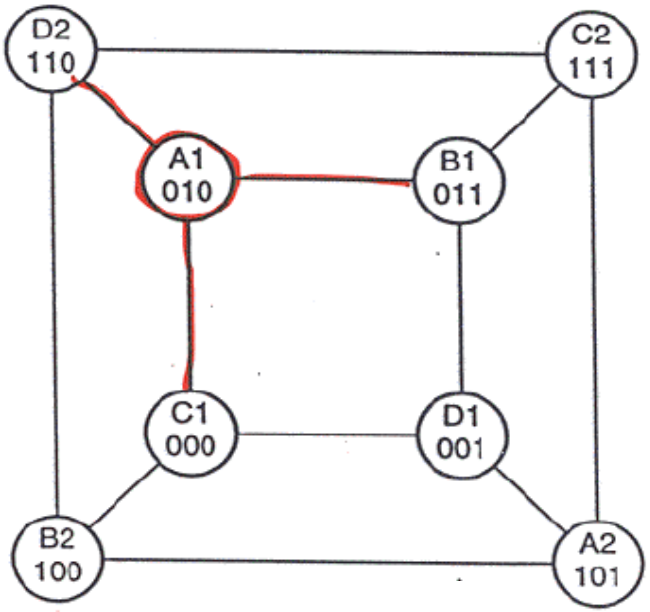
A Worst Case Scenario

- 4-state adjacency diagram



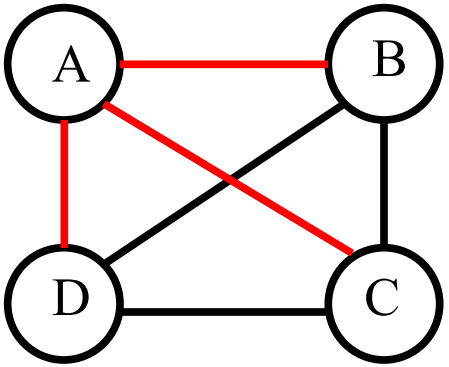
Wrong! This is not the adjacency graph

- Assignment using pairs of equivalent states

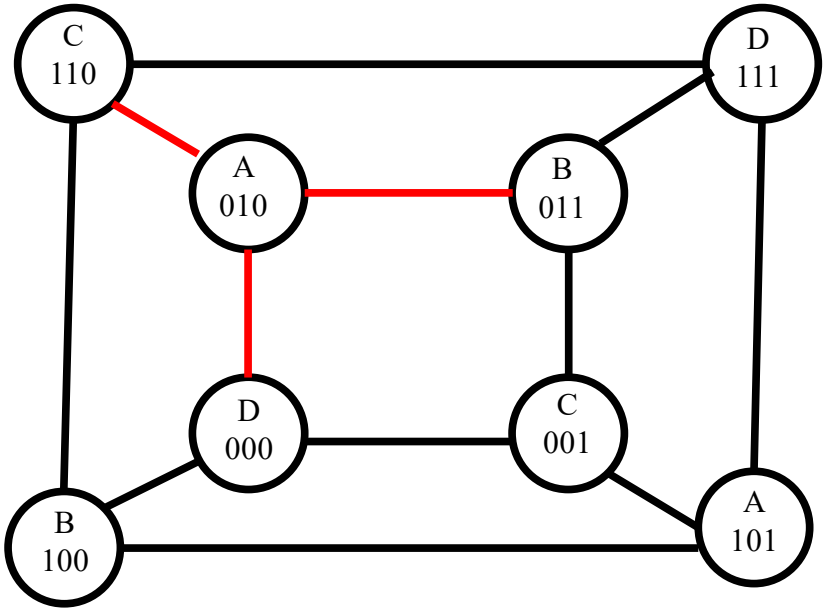


A Worst Case Scenario

- 4-state state transition graph



- Assignment using pairs of equivalent states



State Assignment Guidelines (cont.)

- For a general flow table with 2^n rows (states), we can define a race-free assignment (not necessarily minimum length) using 2n-1 bits.

- Need to assign tables with more than two rows!

Assignment Methods

- Need to avoid critical races
- Proposed Methods:
  - Shared-Row
  - Multiple-Row
  - One-Hot
  - "Race-Free State Assignments for Synthesizing Large-Scale Asynchronous Sequential Logic Circuits", P.D. Fisher and Sheng-Fu Wu, IEEE Transactions on Computers, September 1993 (vol. 42 no. 9), pp. 1025-1034.

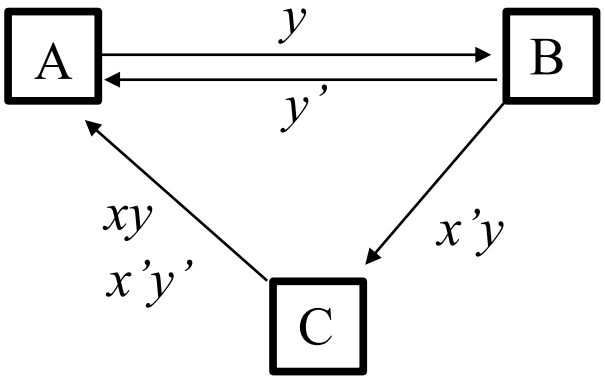
Shared-Row Method

- Include an extra row in the flow table to introduce a cycle into each transition that contributes a critical race
- New row is “shared” by two stable states

What constitutes a critical race?

- More than one different stable state in a column
- Columns with only one stable state can be removed from critical race consideration

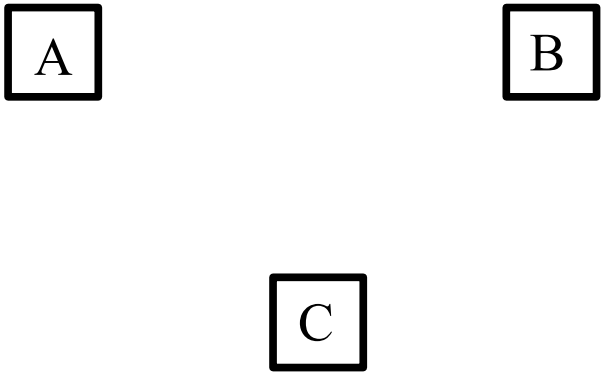
	xy			
S	**00**	**01**	**11**	**10**
A	(A)	B	B	(A)
B	A	C	(B)	A
C	A	(C)	A	C



What constitutes a critical race?

- Every column has one stable state
- No critical races
- Any arbitrary assignment will work

	xy			
S	**00**	**01**	**11**	**10**
A	**A**	**B**	**B**	**A**
B	**A**	**C**	**B**	**A**
C	**A**	**C**	**A**	**C**

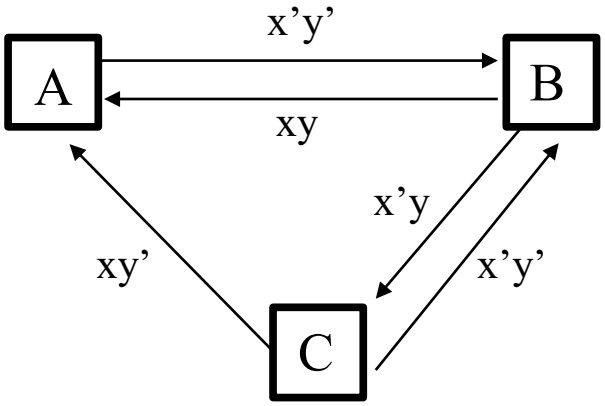


- However could go outside the table

Shared-row state assignment

- Multiple stable states in a column

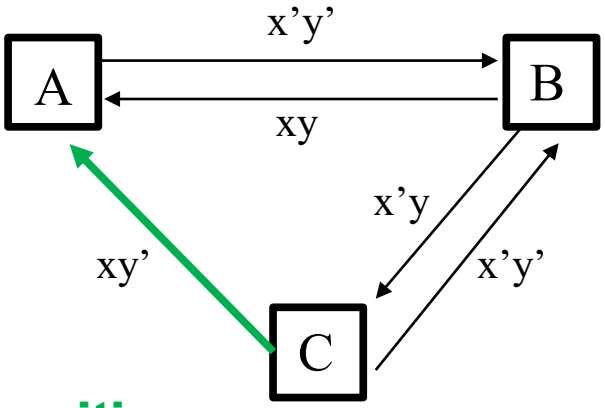
	xy			
S	**00**	**01**	**11**	**10**
A	**B**	**A**	**A**	**A**
B	**B**	**C**	**A**	**B**
C	**B**	**C**	**C**	**A**



Shared-row state assignment

- Multiple stable states in a column
- Encoding 1

		xy			
S	***enc***	**00**	**01**	**11**	**10**
A	**00**	**B**	**A**	**A**	**A**
B	**01**	**B**	**C**	**A**	**B**
C	**11**	**B**	**C**	**C**	**A**



Desired transition

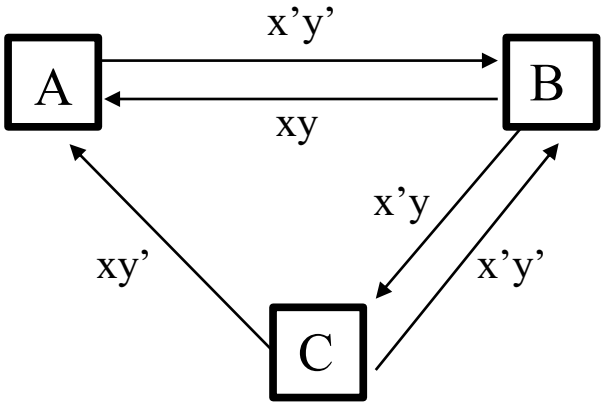
C : xy changing from 11 to 10
This implies C -> A
Encoded as 11->00
Suppose 11->01->00
C never makes it to A but stops at B encoded as 01!!!
Suppose 11->10->00 Encoding is out of the table!

Shared-row state assignment

- Multiple stable states in a column
- Encoding 2

		xy			
S	***enc***	**00**	**01**	**11**	**10**
A	**00**	**B**	**A**	**A**	**A**
B	**01**	**B**	**C**	**A**	**B**
C	**10**	**B**	**C**	**C**	**A**

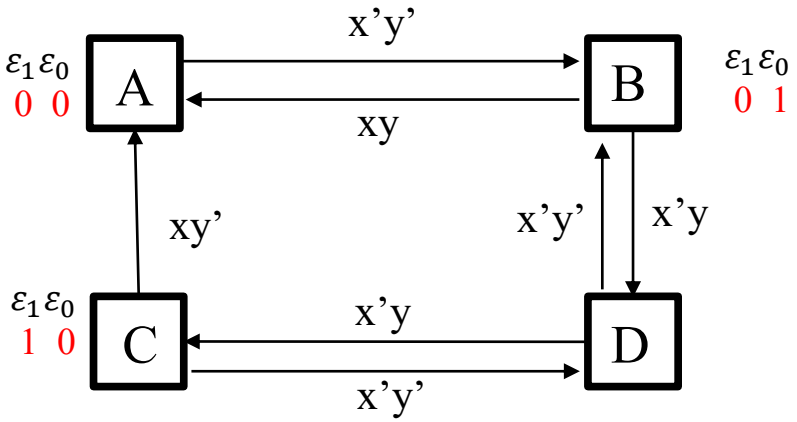
Desired transition



B : xy changing from 00 to 01
This implies B -> C
Encoded as 01->10
Suppose 01->11->10 Encoding 11 is out of table!
Suppose 01->00->10
B never makes it to C but stops at A encoded as 00!!!

Shared-row state assignment

- Arbitrarily assign encodings from both ends

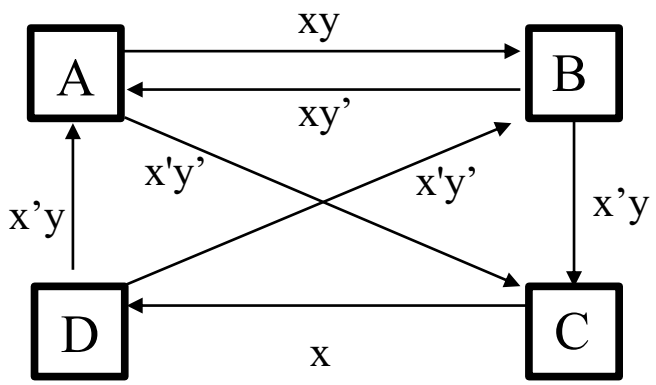


		xy			
S	***enc***	**00**	**01**	**11**	**10**
A	**00**	**B**	**A**	**A**	**A**
B	**01**	**B**	**D**	**A**	**B**
C	**10**	**D**	**C**	**C**	**A**
D	**11**	**B**	**C**	**-**	**-**

- Efficient but difficult (trial and error)

Shared-row Example

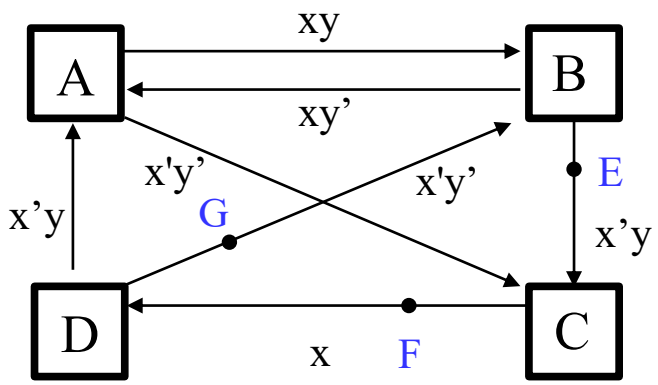
- 4 Columns are all critical



	xy			
S	**00**	**01**	**11**	**10**
A	**C**	**A**	**B**	**A**
B	**B**	**C**	**B**	**A**
C	**C**	**C**	**D**	**D**
D	**B**	**A**	**D**	**D**

Shared-row Example

- Insert new nodes in edges to remove races



Use K map to work out adjacencies

	xy			
z	**00**	**01**	**11**	**10**
0	**A**	**B**	**E**	**C**
1	**D**	**G**		**F**

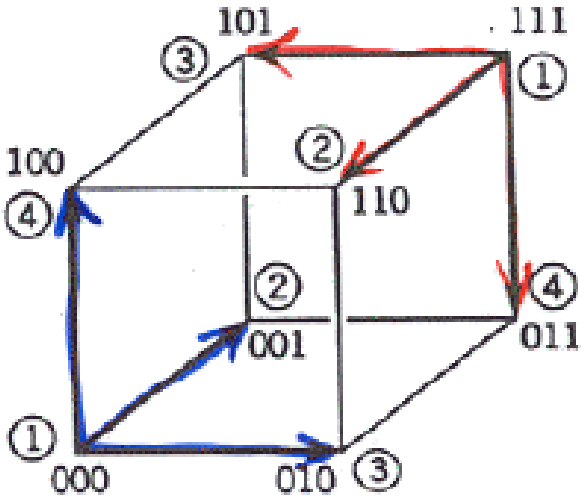
		xy			
S	**enc**	**00**	**01**	**11**	**10**
A	**000**	**C**	**A**	**B**	**A**
B	**001**	**B**	**C**	**B**	**A**
C	**010**	**C**	**C**	**D**	**D**
D	**100**	**B**	**A**	**D**	**D**
E	**011**	**-**	**C**	**-**	**-**
F	**110**	**-**	**-**	**D**	**D**
G	**101**	**B**	**-**	**-**	**-**

Multiple-Row Method

- Notice how the K-map helped us

- Universal assignment for 4 states

	yz			
x	**00**	**01**	**11**	**10**
0	**1**	**3**	**2**	**4**
1	**2**	**4**	**1**	**3**



Multiple-Row Method

- But the universal assignment is not unique

	yz			
x	**00**	**01**	**11**	**10**
0	**1**	**3**	**2**	**4**
1	**2**	**4**	**1**	**3**

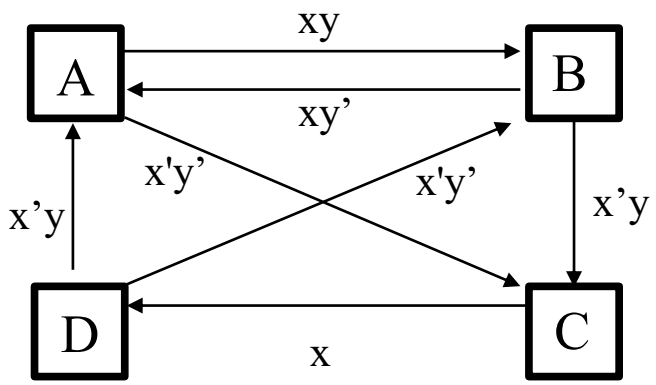
	yz			
x	**00**	**01**	**11**	**10**
0	**1**	**2**	**3**	**4**
1	**3**	**4**	**1**	**2**

Multiple-Row Method

- Start with one row per state. If the state transition has multiple bit changes, duplicate the row and use the universal assignment to assign the states.

Multiple-row Example Revisited

- 4 Columns are all critical



	xy			
S	**00**	**01**	**11**	**10**
A	**C**	**A**	**B**	**A**
B	**B**	**C**	**B**	**A**
C	**C**	**C**	**D**	**D**
D	**B**	**A**	**D**	**D**

Multiple-row Example Revisited

- Duplicate rows and use adjacency table

	xy			
S	**00**	**01**	**11**	**10**
A_1	C_2	A_1	B_1	A_1
A_2	C_1	A_2	B_2	A_2
B_1	B_1	C_1	B_1	A_1
B_2	B_2	C_2	B_2	A_2
C_1	C_1	C_1	D_1	D_1
C_2	C_2	C_2	D_2	D_2
D_1	B_2	A_1	D_1	D_1
D_2	B_1	A_2	D_2	D_2

	xy			
	00	**01**	**11**	**10**
0	A_1	B_1	C_1	D_1
1	C_2	D_2	A_2	B_2

Look in the universal table and find adjacency

Multiple-row Example Revisited

- Duplicate rows and use adjacency table

		xy			
S	***enc***	**00**	**01**	**11**	**10**
A_1	000	C_2	A_1	B_1	A_1
A_2	111	C_1	A_2	B_2	A_2
B_1	001	B_1	C_1	B_1	A_1
B_2	110	B_2	C_2	B_2	A_2
C_1	011	C_1	C_1	D_1	D_1
C_2	100	C_2	C_2	D_2	D_2
D_1	010	B_2	A_1	D_1	D_1
D_2	101	B_1	A_2	D_2	D_2

	yz			
x	**00**	**01**	**11**	**10**
0	A_1	B_1	C_1	D_1
1	C_2	D_2	A_2	B_2

Multiple-row Example Revisited

- Show encoding and reorder rows

		xy			
S	***enc***	**00**	**01**	**11**	**10**
A_1	000	C_2	A_1	B_1	A_1
B_1	001	B_1	C_1	B_1	A_1
C_1	011	C_1	C_1	D_1	D_1
D_1	010	B_2	A_1	D_1	D_1
B_2	110	B_2	C_2	B_2	A_2
A_2	111	C_1	A_2	B_2	A_2
D_2	101	B_1	A_2	D_2	D_2
C_2	100	C_2	C_2	D_2	D_2

	yz			
x	**00**	**01**	**11**	**10**
0	A_1	B_1	C_1	D_1
1	C_2	D_2	A_2	B_2

Multiple-row state assignment

- Can be extended to any number of states using a predefined table
- Inefficient but straight forward

	wx			
yz	**00**	**01**	**11**	**10**
00	A_1	B_1	C_1	D_1
01	E_1	F_1	G_1	H_1
11	C_2	D_2	A_2	B_2
10	G_2	H_2	E_2	F_2

Other state assignment methods

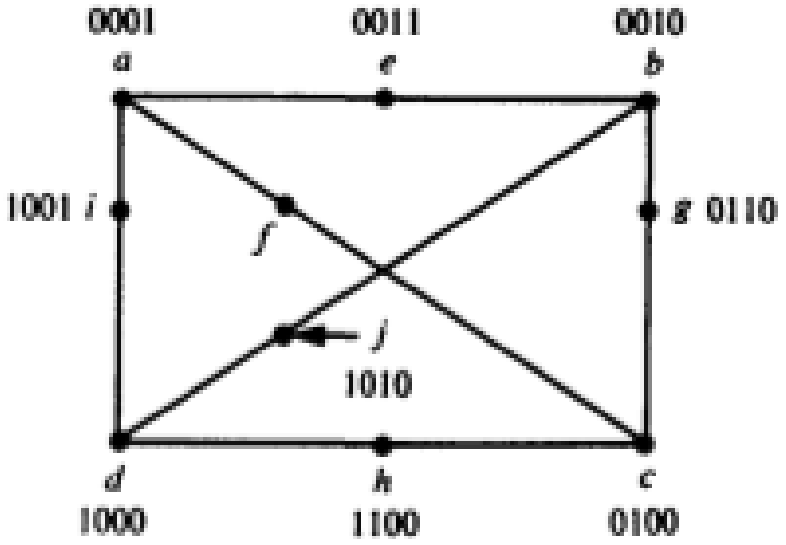
- One-Hot
- Each state has one 1 in it
- A: 0001
- B: 0010
- C: 0100
- D: 1000

- State transitions become new states which are the logical OR of the two final states

One-hot method

- Insertion into every edge

		x_1x_2 00	01	11	10
0001	*a*	¢ *f*	*a*	¢ *e*	*a*
0010	*b*	*b*	*c g*	*b*	*a e*
0100	*c*	*c*	*c*	¢ *h*	¢ *h*
1000	*d*	¢ *j*	*a i*	*d*	*d*
0011	*e*			*b*	*a*
0101	*f*	*c*			
0110	*g*		*c*		
1100	*h*			*d*	*d*
1001	*i*		*a*		
1010	*j*	*b*			



Race-Free State Assignments for Synthesizing Large-Scale Asynchronous Sequential Logic Circuits

- P.D. Fisher and Sheng-Fu Wu, IEEE Transactions on Computers, September 1993 (vol. 42 no. 9), pp. 1025-1034.

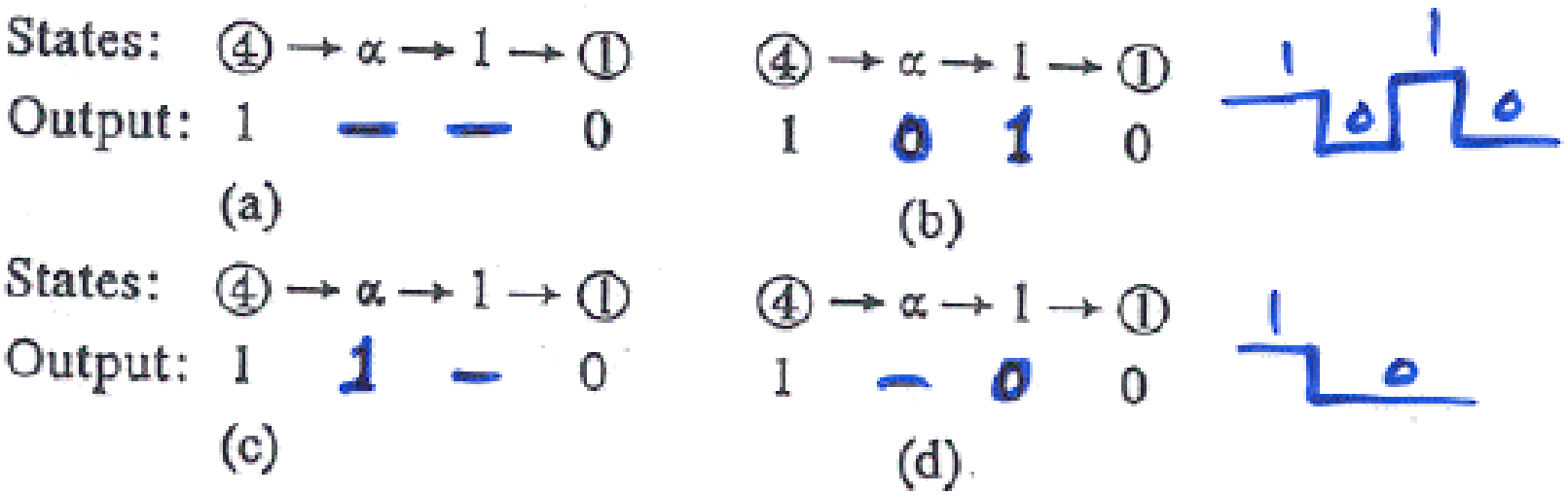
Hazards

Hazard

- Temporary wrong output calculation due to propagation delay
- Static
  - Static-1 starts in 1, ends up in 1 but transitions to 0
  - Static-0 starts in 0, ends up in 0 but transitions to 1
- Dynamic
  - Multiple transitions
  - Can't occur if no static hazards exist

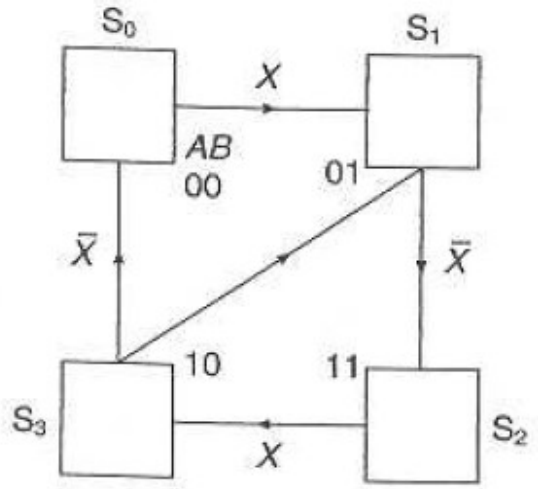
Avoiding Output Glitches

- Sometimes you need to change the don't-cares to care situations to avoid possible glitches. Otherwise, after optimization some glitches may be generated.
- Example: (a) original, (b) decided by optimizer, (c) and (d) guided by replacing some of don't-cares.

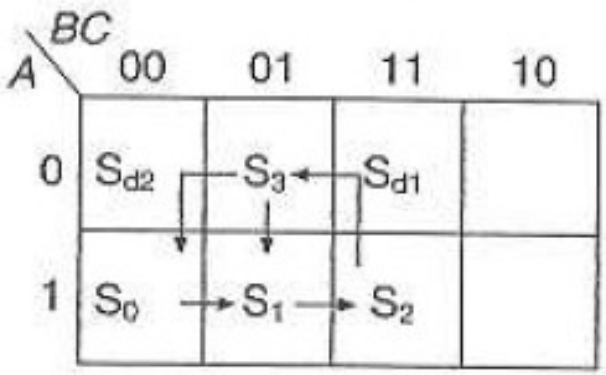


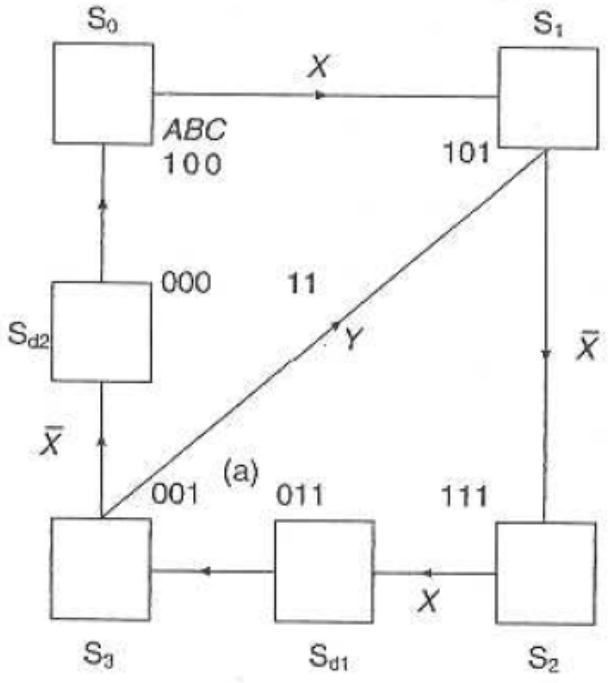
Adding Dummy State – Example

- Theoretically 2 bits are enough to encode 4 states. But it is not race-free.



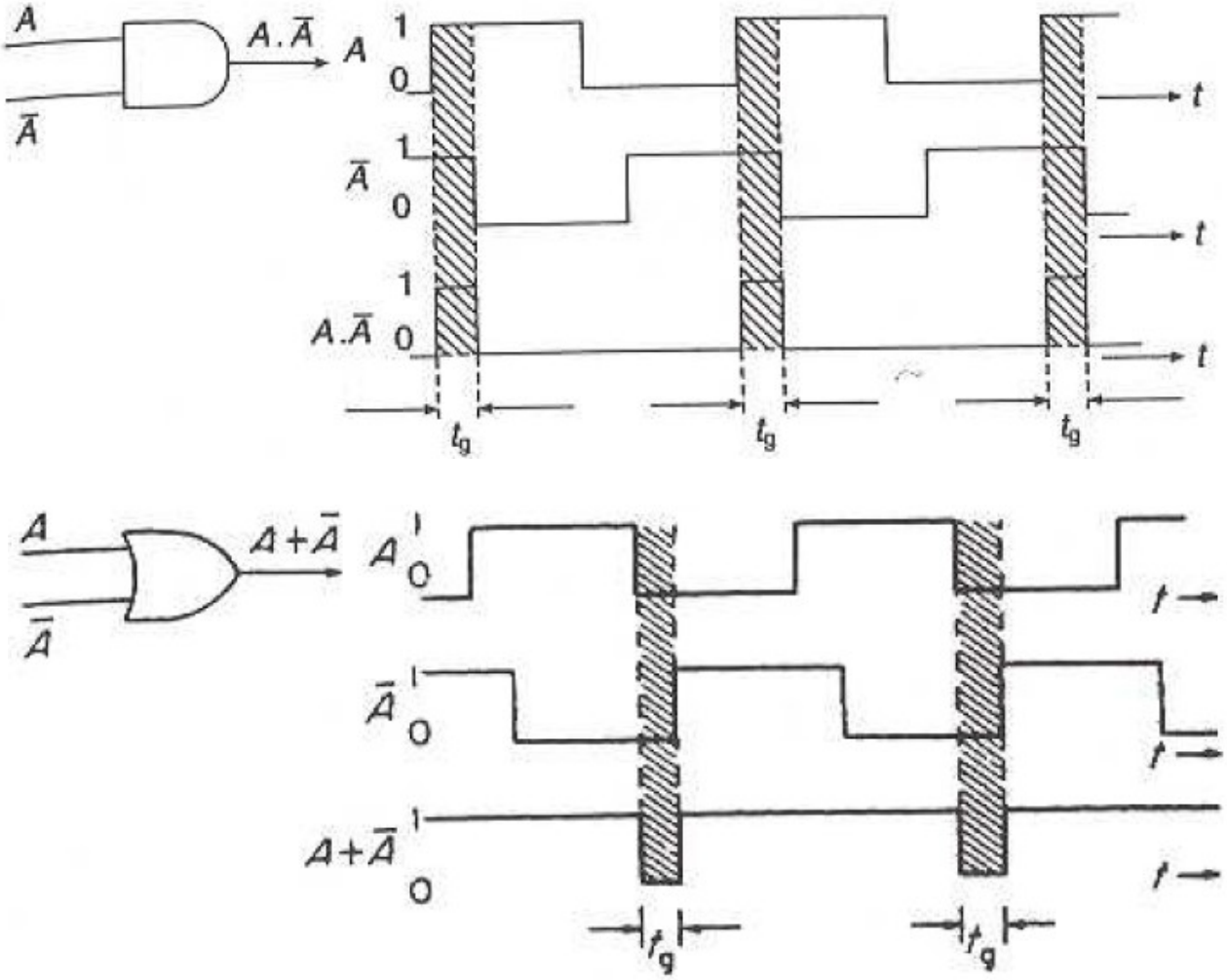
- Using 3 bits





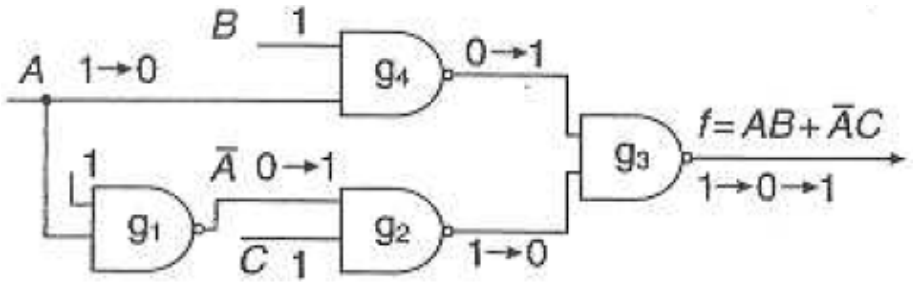
Static Hazard on the Outputs

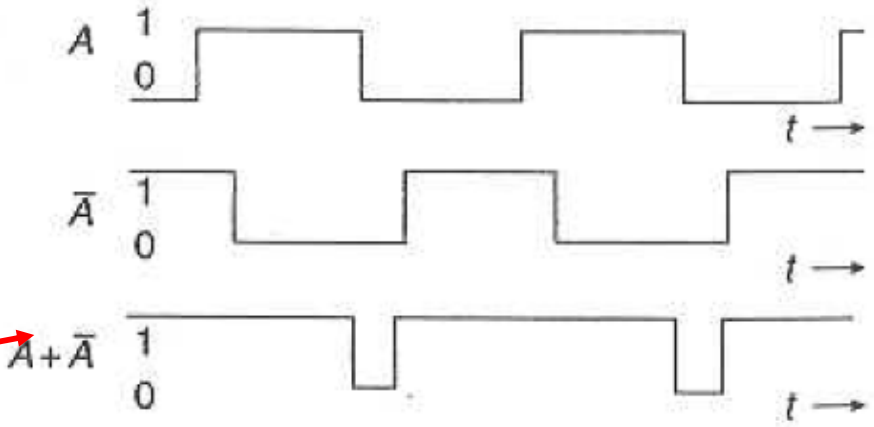
- If the Boolean expression is reduced to AA’ or A+A’, the possibility of static hazard (glitch) exist



Static Hazard - Examples

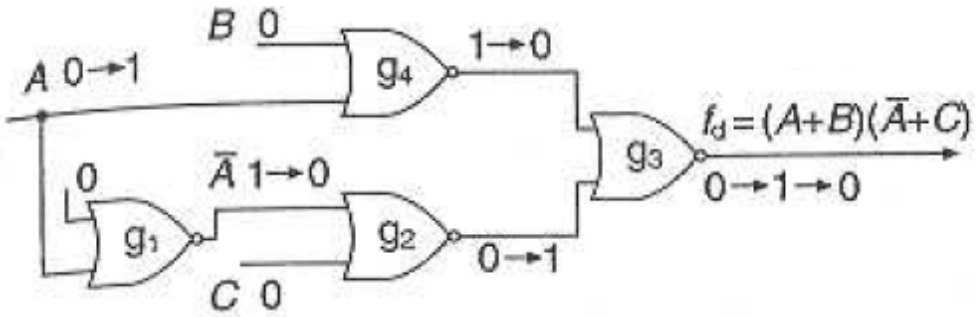
- When B=C=1, f is reduced to A+A'

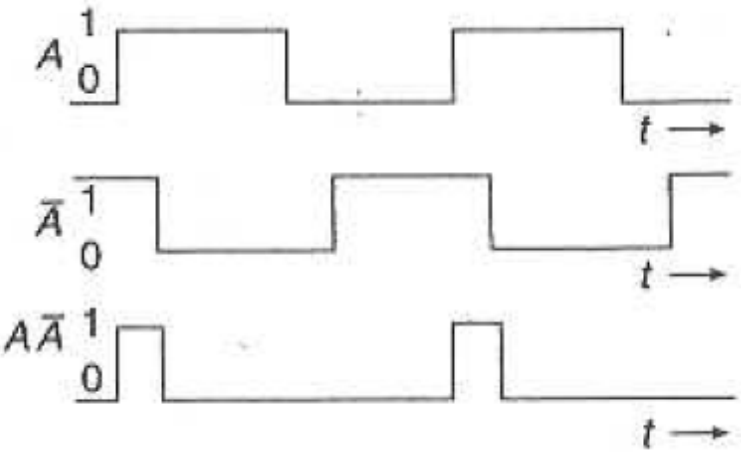




1-hazard

- When B=C=0, f is reduced to AA'

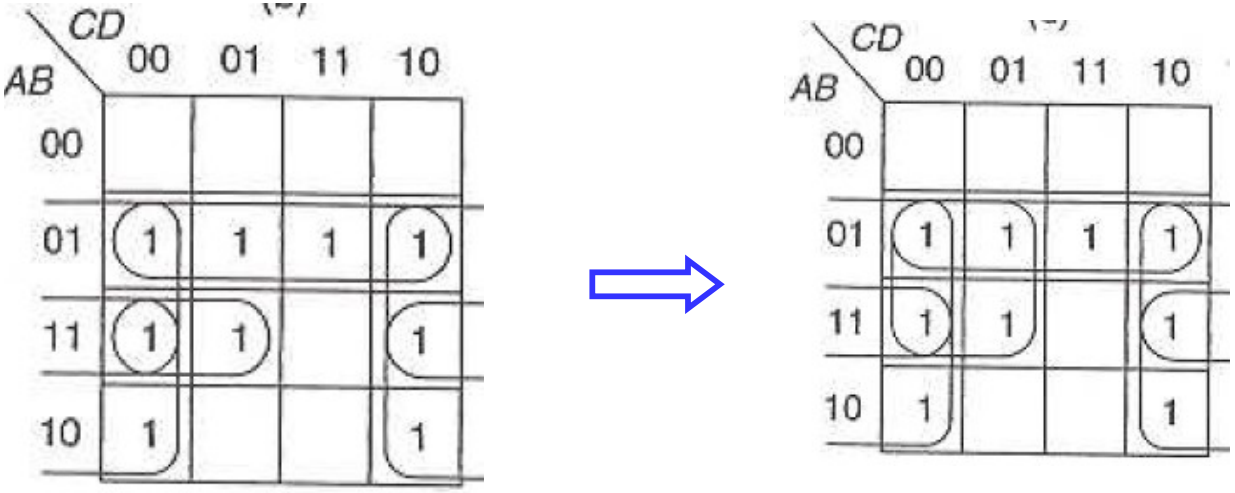




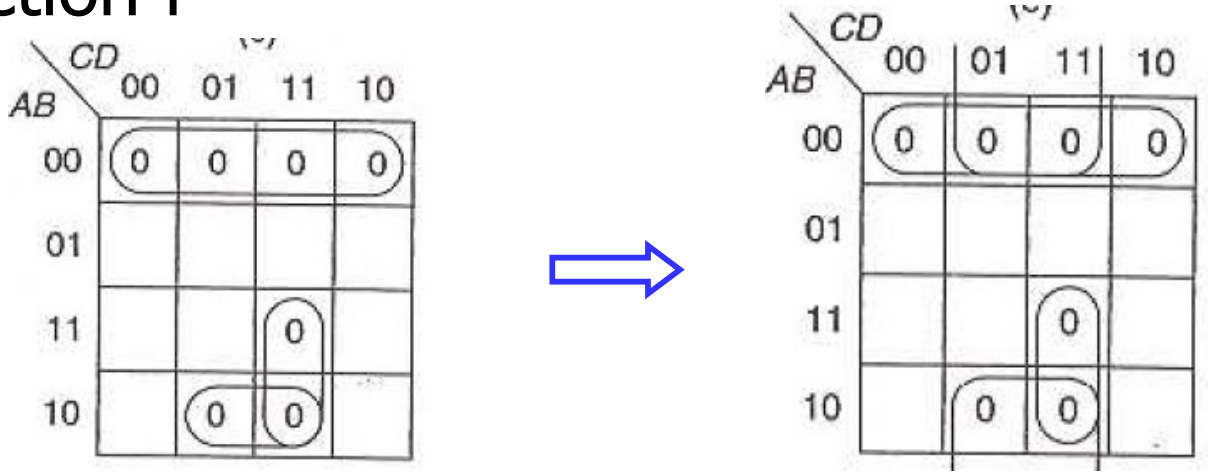
0-hazard

Hazard-Free Implementation

- Function f=ABC'+(A+B)(A'+D')

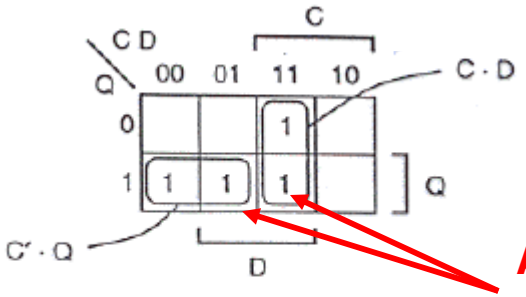


- Function f'

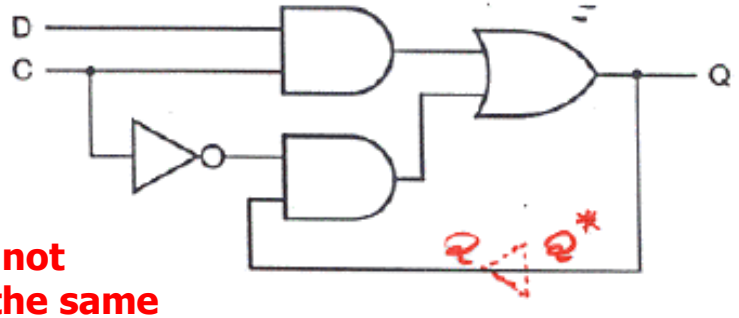


Hazard Free Implementation – D-Latch

- To be on the safe side and avoid undesired transitions, hazard-free equations for state variables and output functions should be used.
- Example: unreliable D-latch (subject to static-1 hazard)

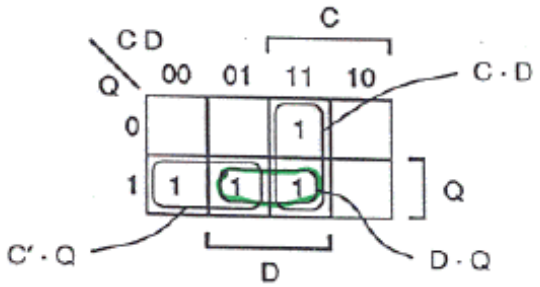


$Q^* = C \cdot D + C' \cdot Q$



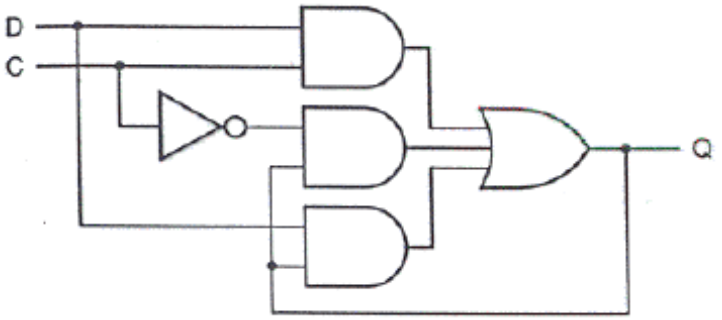
Adjacent 1s not covered by the same Prime Implicant

- Reliable D-latch (hazard eliminated)



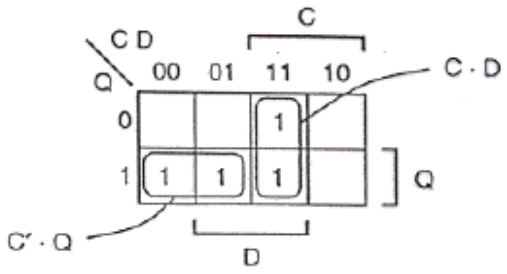
$Q^* = C \cdot D + C' \cdot Q + D \cdot Q$

Redundant Term

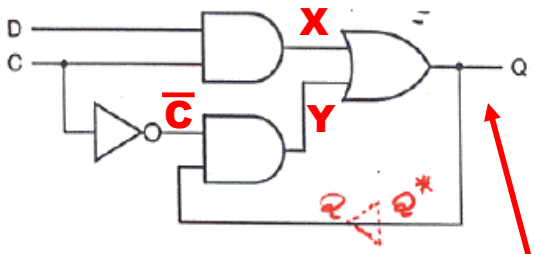


Hazard Free Implementation D-Latch (cont.)

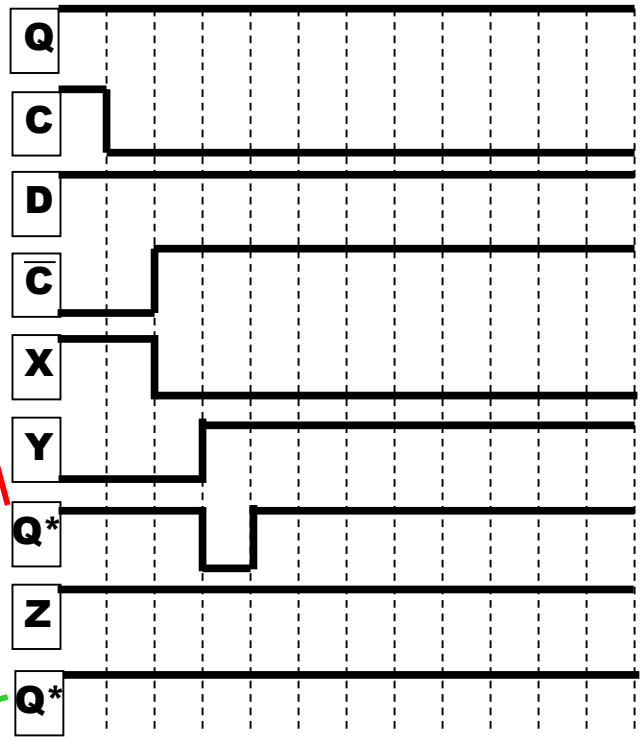
- QCD: 111→101 ; Assume 1 time unit delay for each gate

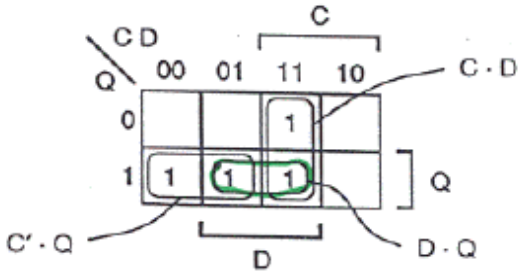


$Q^* = C \cdot D + C' \cdot Q$

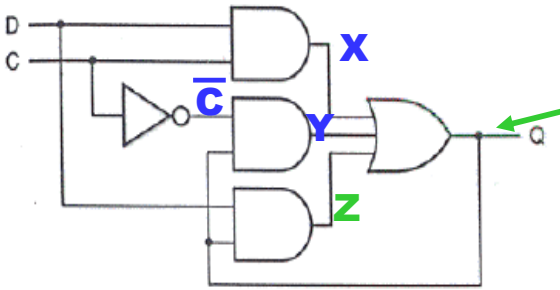


1 Hazard



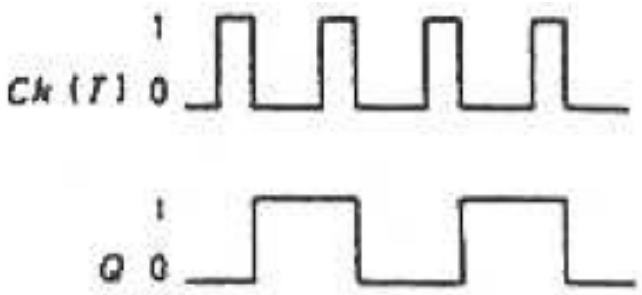


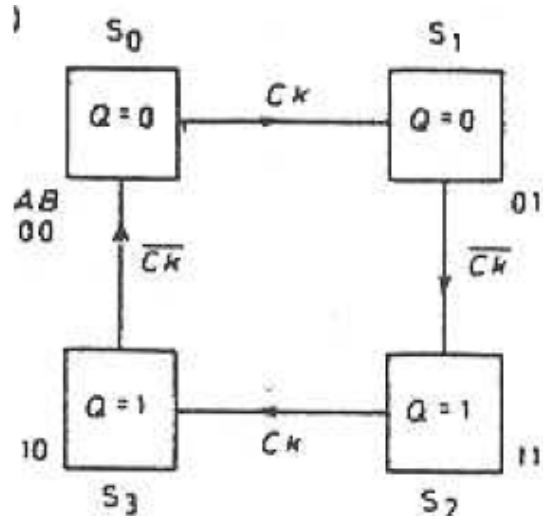
$Q^* = C \cdot D + C' \cdot Q + D \cdot Q$

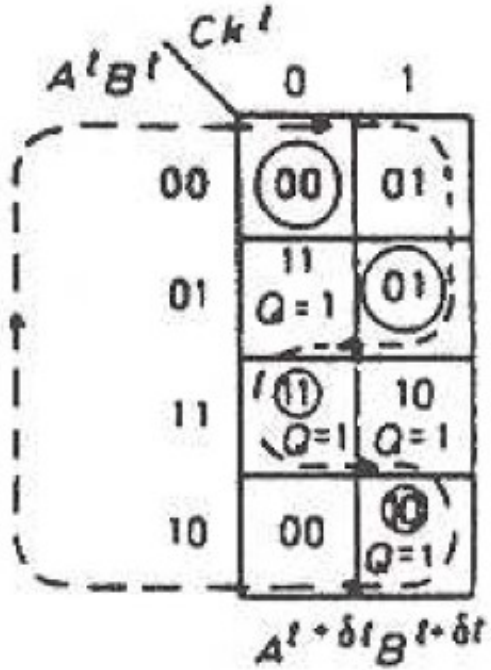


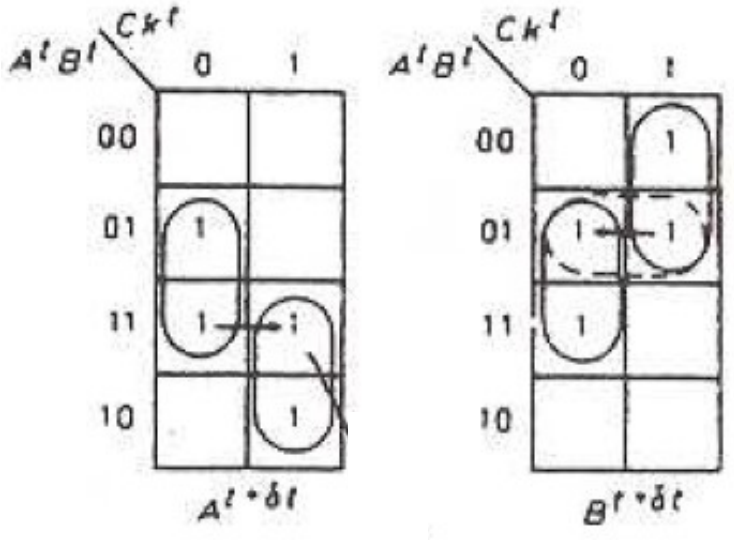
Hazard Free Implementation - T-FF

- Behavior of T Flip-Flop



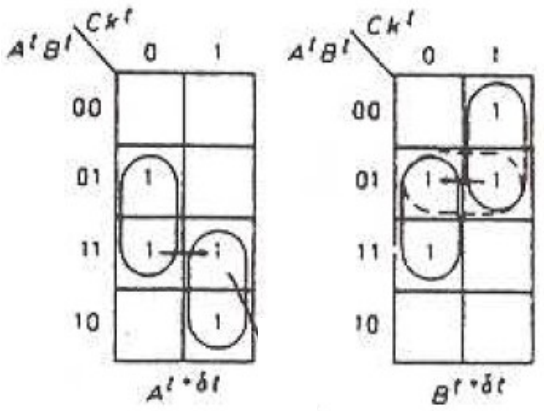


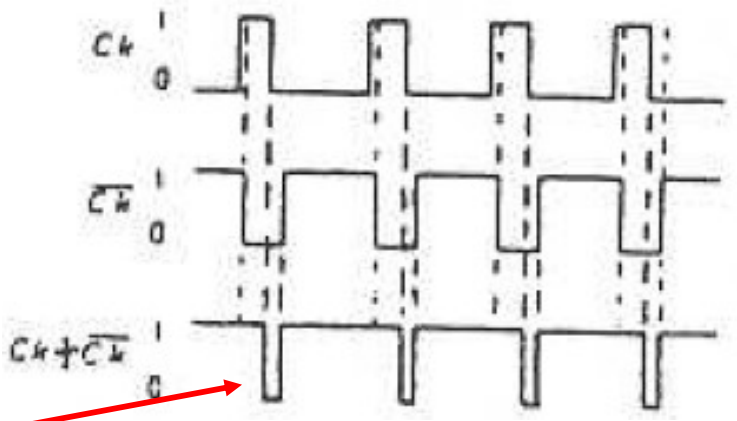




Hazard Free Implementation - T-FF

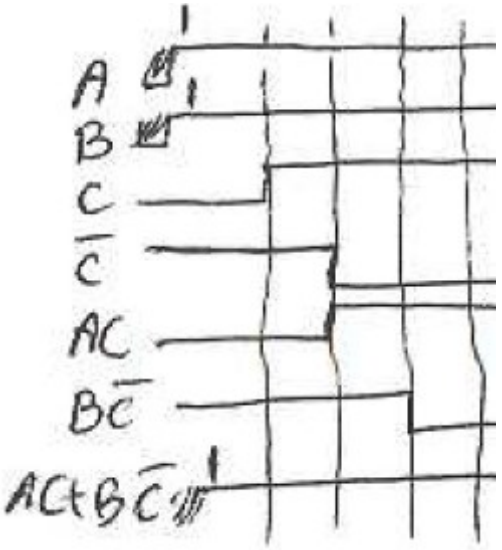
- Possibility of hazard



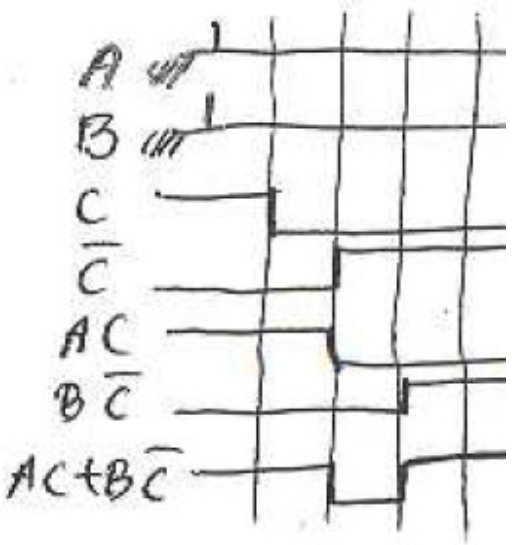


Only C: 1→0 causes 1-hazard

ABC: 110→111 (No hazard)

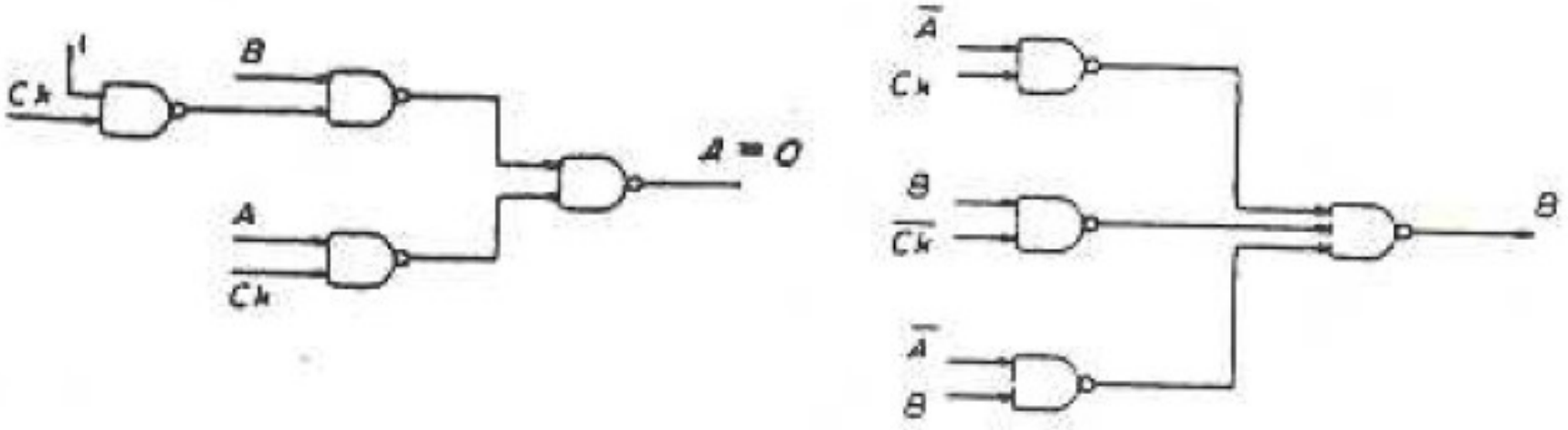


ABC: 111→110 (1-hazard)



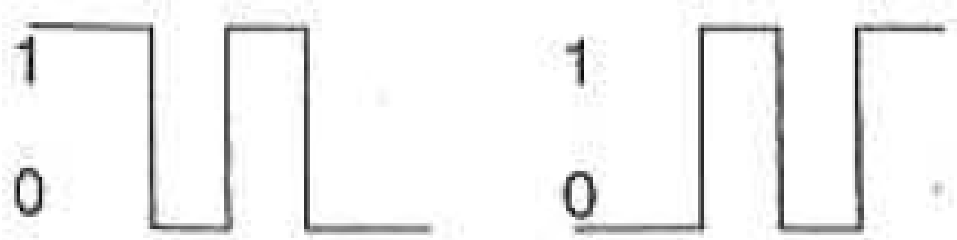
Hazard-Free Implementation – T-FF

- One possible implementation



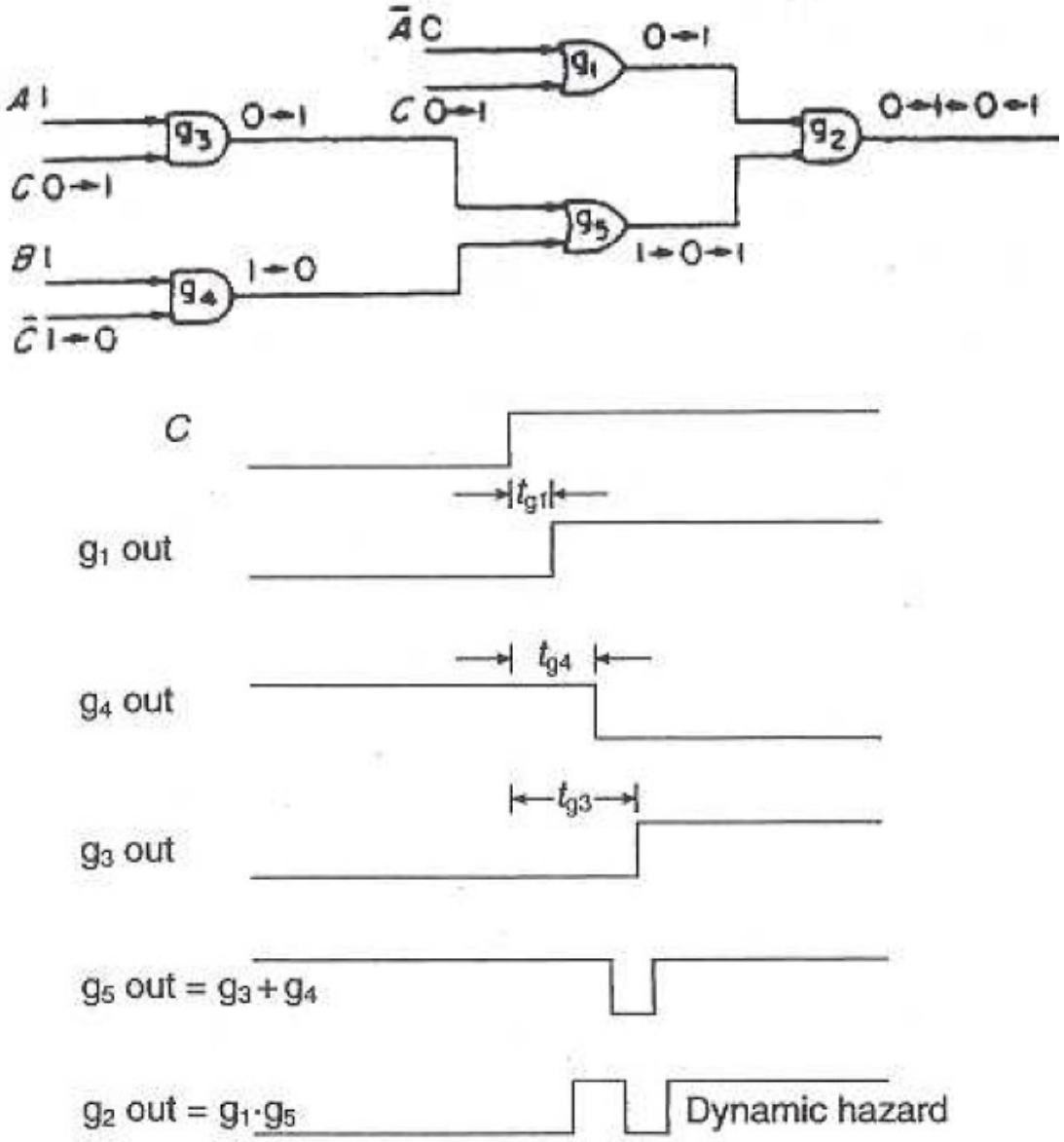
Dynamic Hazard on the Outputs

- Those cases that
  - —instead of 1→0 we get 1→0→1→0
  - —Instead of 0→1 we get 0→1→0→1



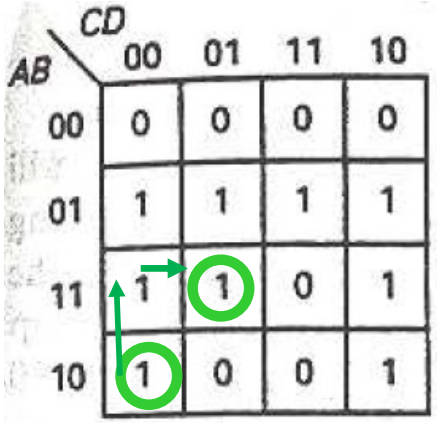
- In SoP and PoS circuits, if there are no static hazards, then the circuit will have no dynamic hazards.

Dynamic Hazard - Example



Functional Hazard on the Outputs

- This is due to multiple-bit change (or sequence of single-bit change according to the fundamental mode assumption)
- They appear as 0-hazard or 1-hazard
- Example

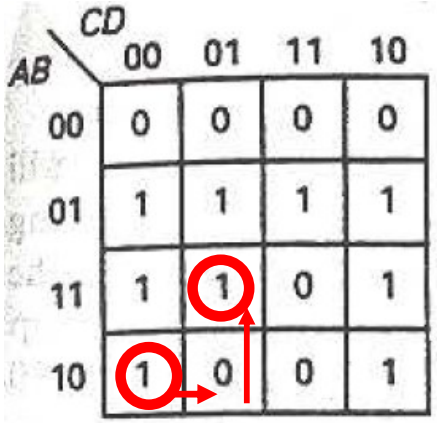


Static 1-hazard occurs

1. B and D change simultaneously:
   $ABCD = 1000 \rightarrow 1101$
   $f = 1 \quad \rightarrow 1$
2. B changes before D:
   $ABCD = 1000 \rightarrow 1100 \rightarrow 1101$
   $f = 1 \quad \rightarrow 1 \quad \rightarrow 1$

Functional Hazard on the Outputs

- This is due to multiple-bit change (or sequence of single-bit change according to the fundamental mode assumption)
- They appear as 0-hazard or 1-hazard
- Example



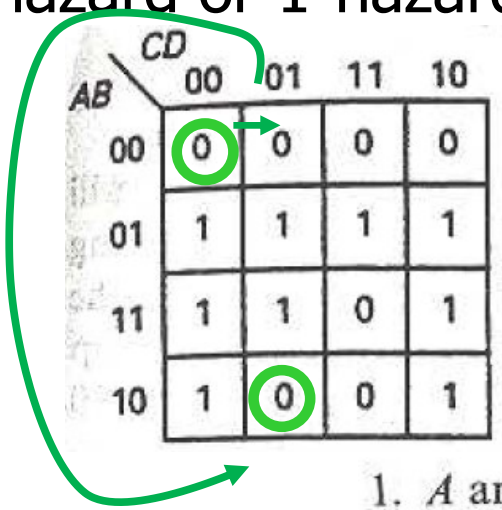
AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	0	1
10	1	0	0	1

1. B and D change simultaneously:
   $ABCD = 1000 \rightarrow 1101$
   $f = 1 \quad \rightarrow 1$
2. B changes before D:
   $ABCD = 1000 \rightarrow 1100 \rightarrow 1101$
   $f = 1 \quad \rightarrow 1 \quad \rightarrow 1$
3. D changes before B:
   $ABCD = 1000 \rightarrow 1001 \rightarrow 1101$
   $f = 1 \quad \rightarrow 0 \quad \rightarrow 1$

Oh, no! If ABCD=1001 is a transient state (which it is BTW), output should be a 1 not a 0

Functional Hazard on the Outputs

- This is due to multiple-bit change (or sequence of single-bit change according to the fundamental mode assumption)
- They appear as 0-hazard or 1-hazard
- Example



AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	0	1
10	1	0	0	1

Static 0-hazard occurs

1. A and D change simultaneously

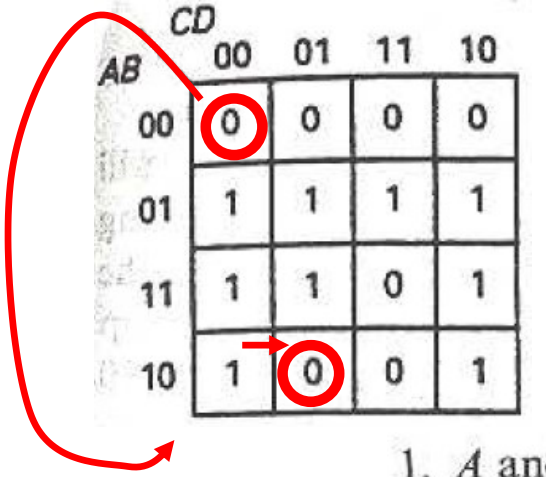
$$ABCD = 0000 \rightarrow 1001$$
$$f = 0 \quad \rightarrow 0$$

3. D changes before A

$$ABCD = 0000 \rightarrow 0001 \rightarrow 1001$$
$$f = 0 \quad \rightarrow 0 \quad \rightarrow 0$$

Functional Hazard on the Outputs

- This is due to multiple-bit change (or sequence of single-bit change according to the fundamental mode assumption)
- They appear as 0-hazard or 1-hazard
- Example



AB \ CD	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	1	1	0	1
10	1	0	0	1

Static 0-hazard occurs

Not enough info to make intelligent decision. What variables are state and what are inputs? ABCD=1000 was a stable state in static 1-hazard case.

1. A and D change simultaneously
   $ABCD = 0000 \rightarrow 1001$
   $f = 0 \quad \rightarrow 0$
2. A changes before D
   $ABCD = 0000 \rightarrow 1000 \rightarrow 1001$
   $f = 0 \quad \rightarrow 1 \quad \rightarrow 0$
3. D changes before A
   $ABCD = 0000 \rightarrow 0001 \rightarrow 1001$
   $f = 0 \quad \rightarrow 0 \quad \rightarrow 0$

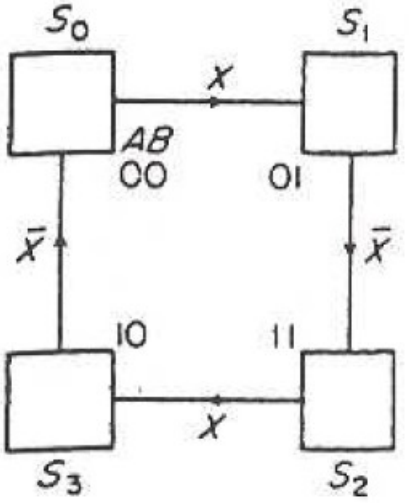
Essential Hazards

- This is caused by a race between
  - An input signal and
  - A state variable
- Usually, if input transition time (t_i) is bigger than state transition time (t_t), incorrect circuit operation will occur.

- Example: Suppose we are in S_0 and X:0→1 occurs.
  - Correct operation ($t_i<t_t$)

    X:0→1 and AB:00→01 (S_0→S_1)
  - Incorrect operation ($t_i>t_t$)

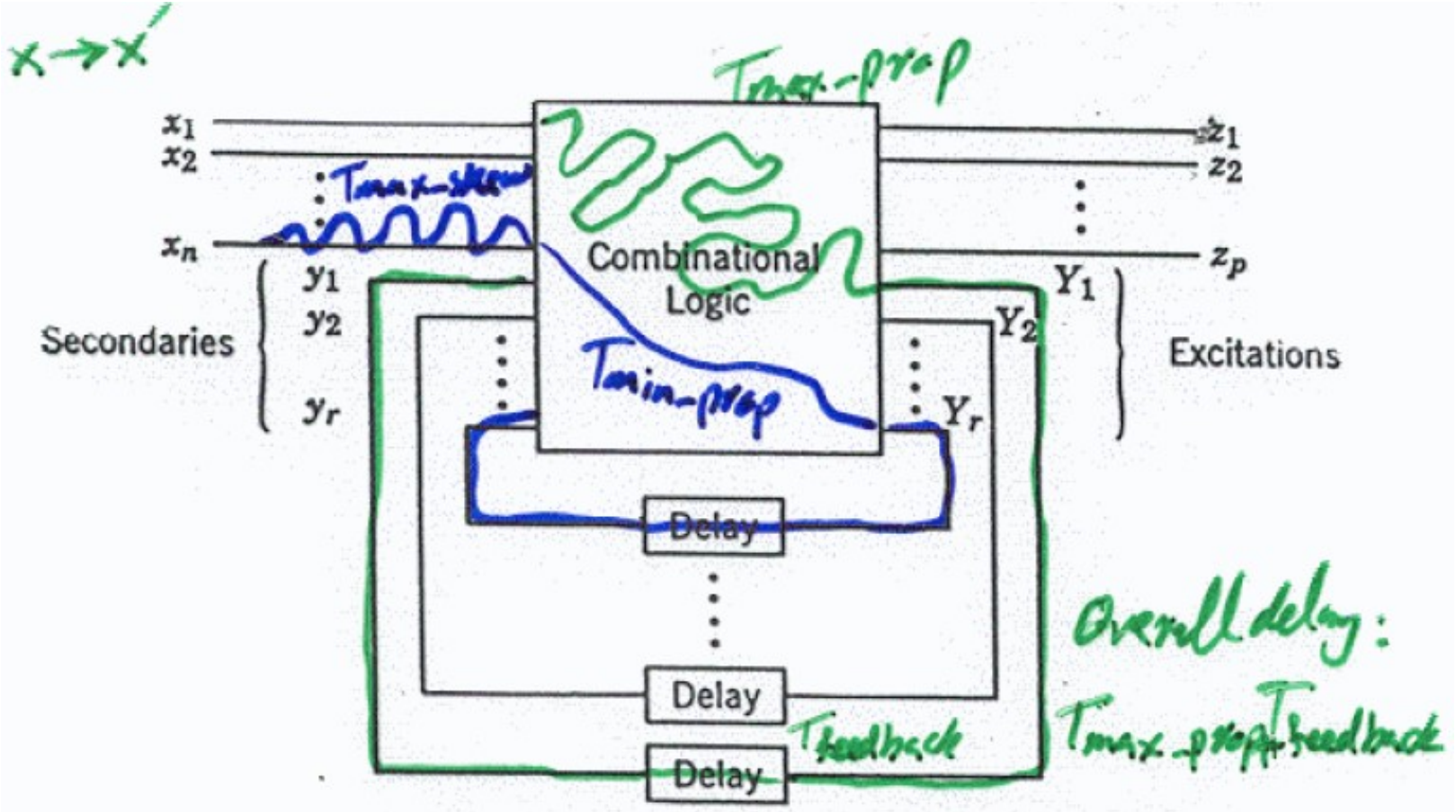
    X:0→0→1 and AB:00→01→11 (S_0→S_1→S_2)



Timing Issues

Timing Requirement – One Change at a Time

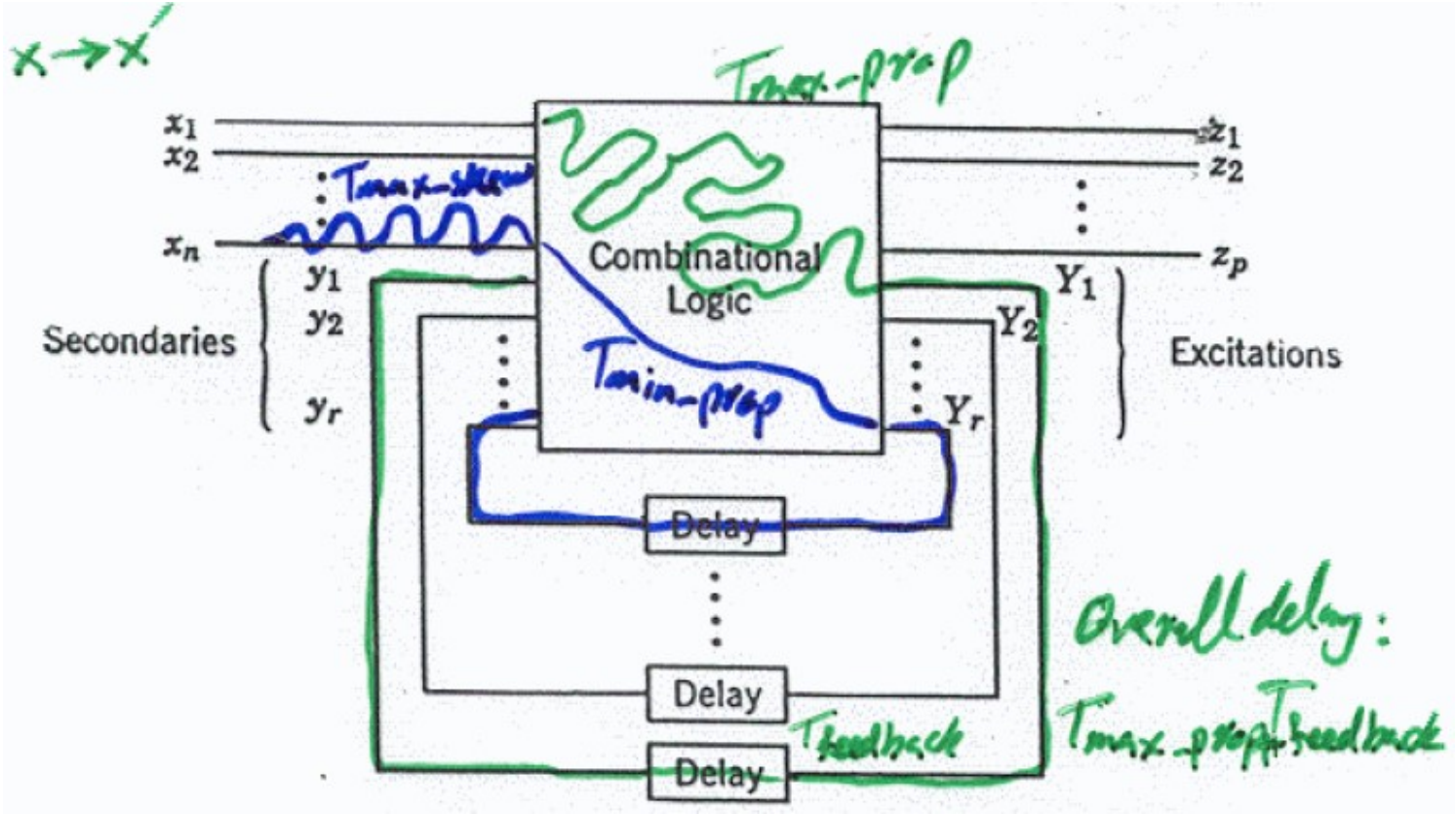
- Only one input signal changes at a time, with a minimum bound between successive input changes



Timing Requirement – Maximum Delay

- Maximum propagation delay through excitation logic and feedback paths is less than the time between successive input changes

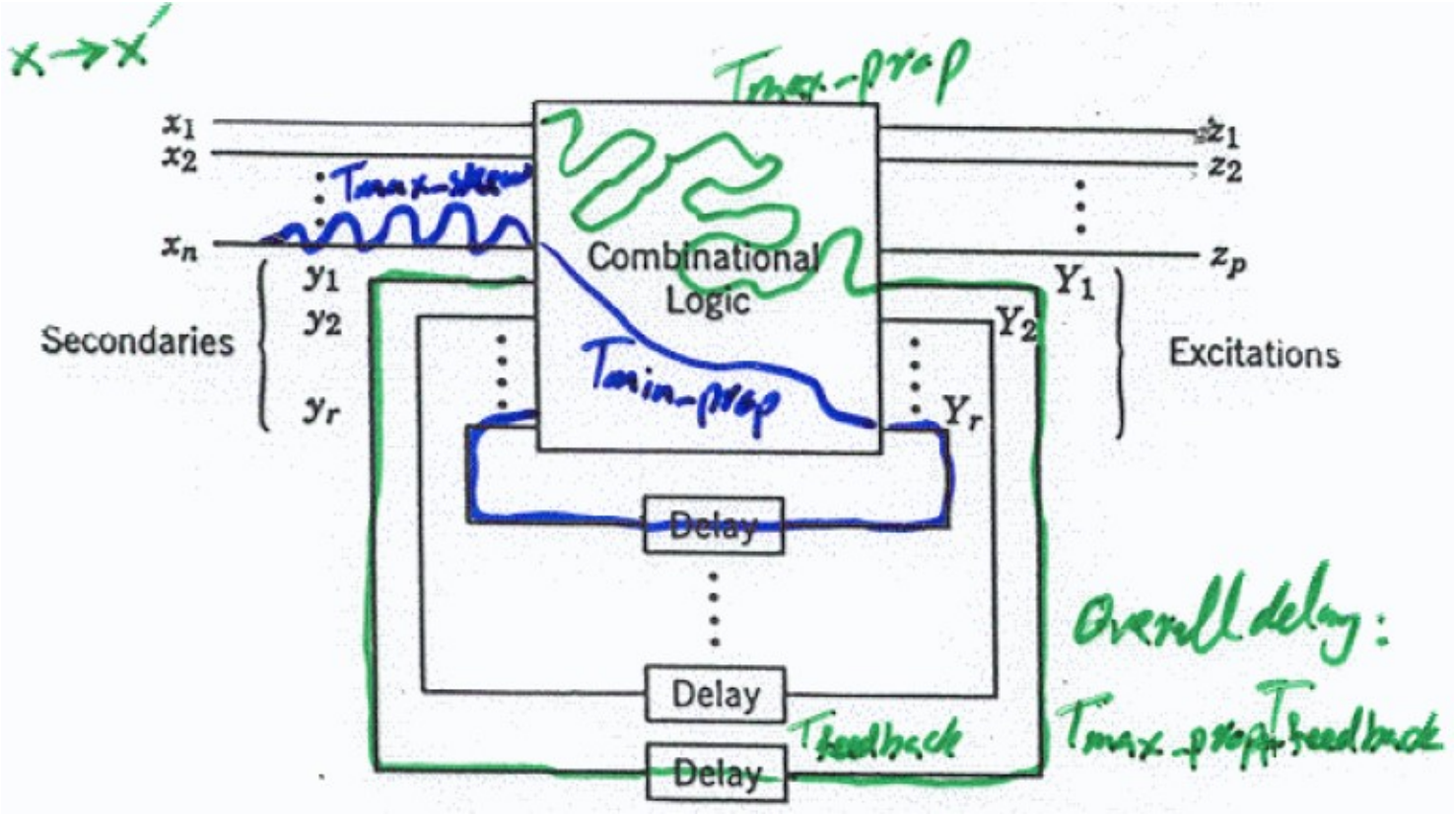
$$T_{max\_prop} + T_{feedback} < T_{min\_input} = 1/f_{max\_input}$$



Timing Requirement – Minimum Delay

- Minimum propagation delay through excitation logic and feedback paths is greater than the maximum timing skew through input logic

$$T_{min\_prop} + T_{feedback} > T_{max\_skew\_input}$$



Design Procedure - Pulse Catching Example

Basic Design Steps

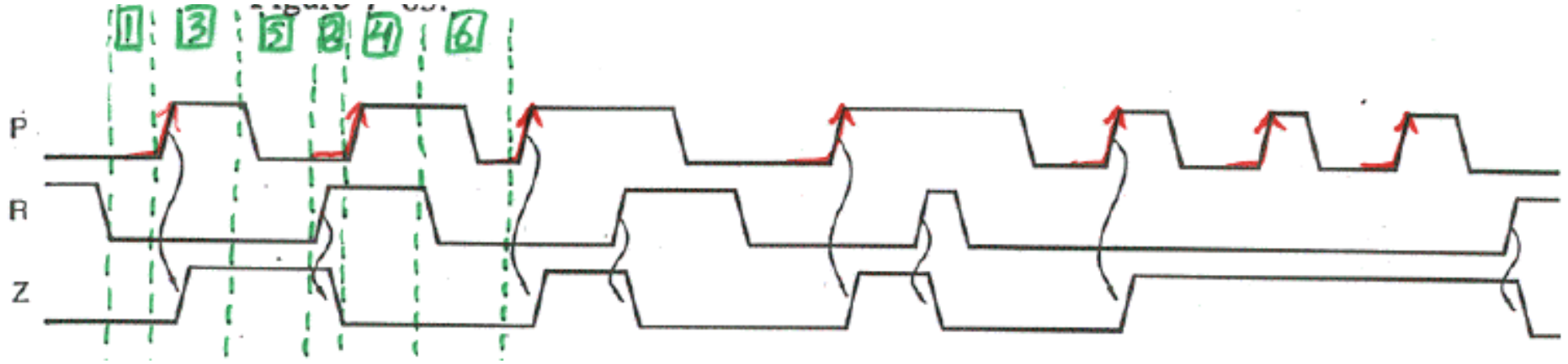
1. Construct the primitive flow table from the circuit's word description.
2. Minimize the number of states in the flow table.
3. Find a race-free assignment for the coded states, adding auxiliary states or splitting the states as required.
4. Construct the transition table.
5. Construct the excitation maps and obtain a hazard-free realization of the equations.
6. Draw the logic diagram.
7. Check for essential hazards. Modify the circuit if needed.

A Pulse-Catching Circuit Example

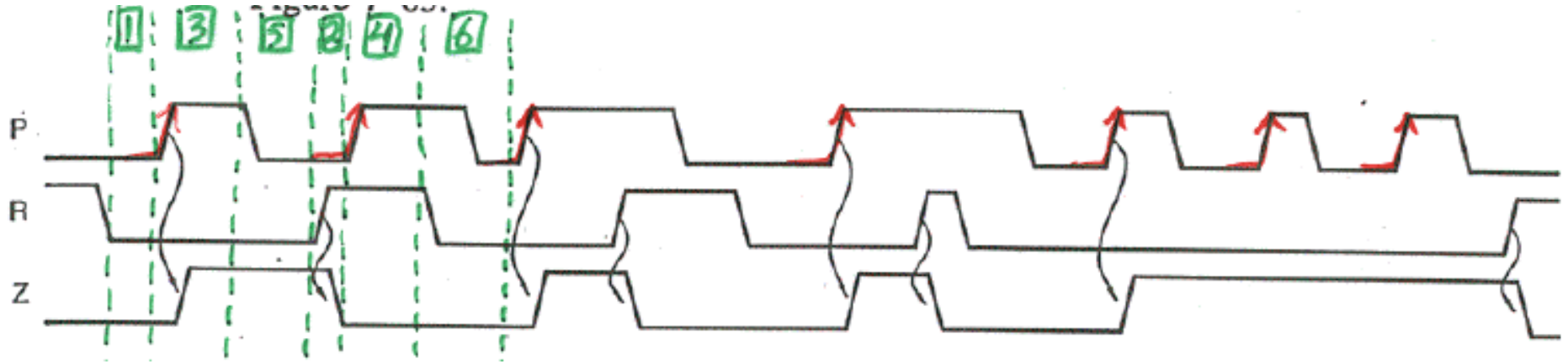
- Design an asynchronous (feedback) circuit with two inputs P (pulse) and R (reset) and a signal output Z that is normally zero. The output should become 1 whenever a 0→1 transition occurs on P, and should be reset to 0 whenever R is 1.

A Pulse-Catching Circuit Example

- Design an asynchronous (feedback) circuit with two inputs P (pulse) and R (reset) and a signal output Z that is normally zero. The output should become 1 whenever a 0→1 transition occurs on P, and should be reset to 0 whenever R is 1.
- Typical behavior



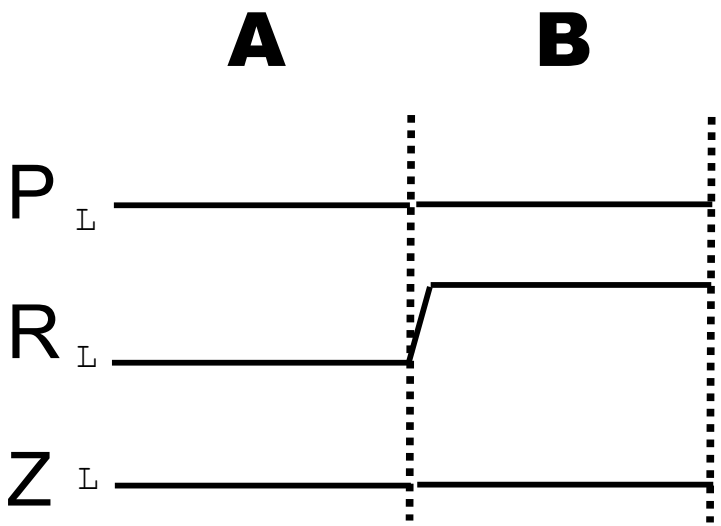
Pulse-Catching – Flow Table



- The primitive flow table

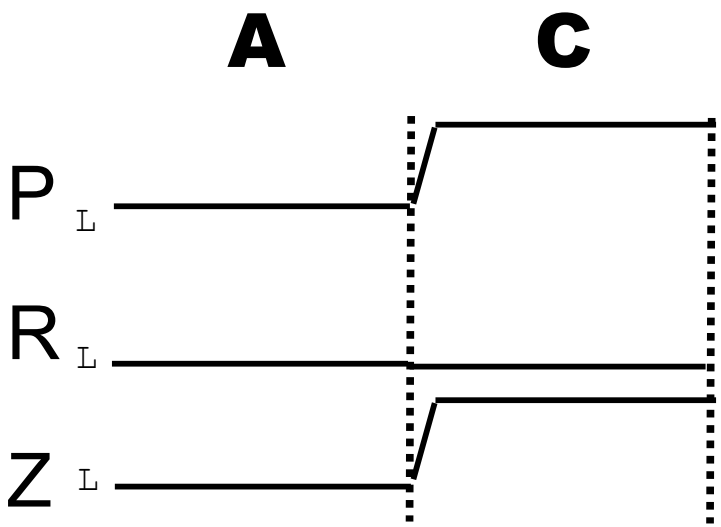
Meaning		S	P R 00	01	11	10	Z
Idle, waiting for pulse	1	IDLE	IDLE	RES1	—	PLS1	0
Reset, no pulse	2	RES1	IDLE	RES1	RES2	—	0
Got pulse, output on	3	PLS1	PLS2	—	RES2	PLS1	1
Reset, got pulse	4	RES2	—	RES1	RES2	PLSN	0
Pulse gone, output on	5	PLS2	PLS2	RES1	—	PLS1	1
Got pulse, but output off	6	PLSN	IDLE	—	RES2	PLSN	0
			S*				

Pulse-Catching – Flow Table



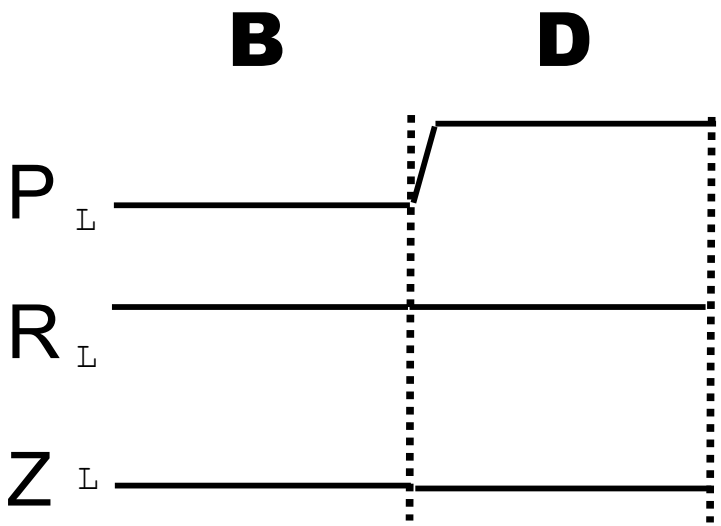
P R Z		00	01	11	10
0 0, 0	A	A,0	B,0		
0 1, 0	B		B,0		

Pulse-Catching – Flow Table



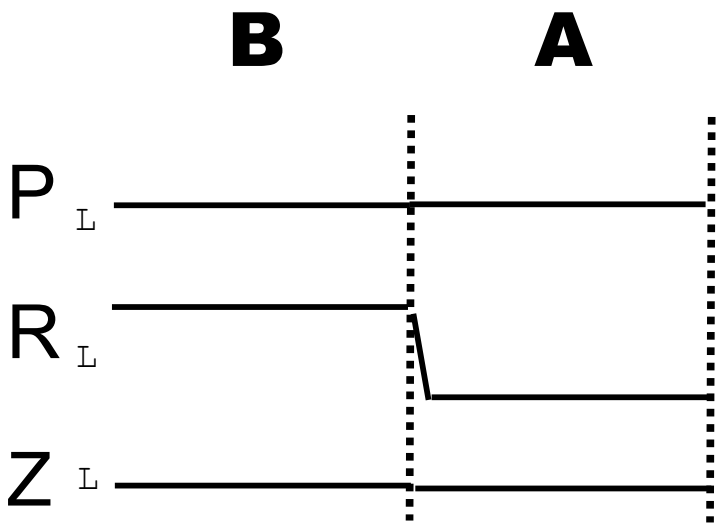
P R Z		00	01	11	10
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B		B,0		
1 0, 1	C				C,1

Pulse-Catching – Flow Table



P R Z		P R 00	01	11	10
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B		B,0	D,0	
1 0, 1	C				C,1
1 1, 0	D			D,0	

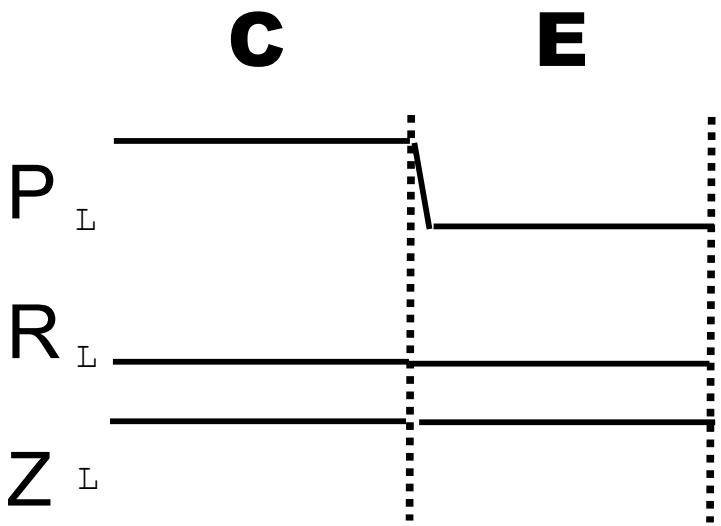
Pulse-Catching – Flow Table



P R

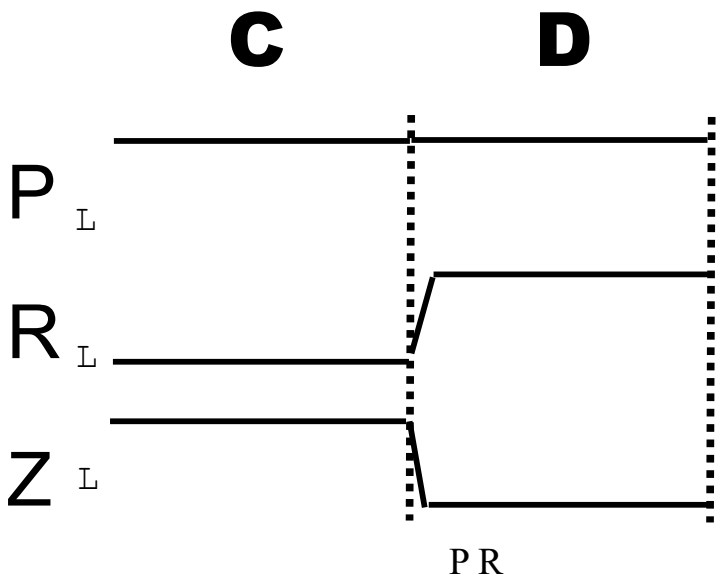
P R Z		00	01	11	10
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C				C,1
1 1, 0	D			D,0	

Pulse-Catching – Flow Table



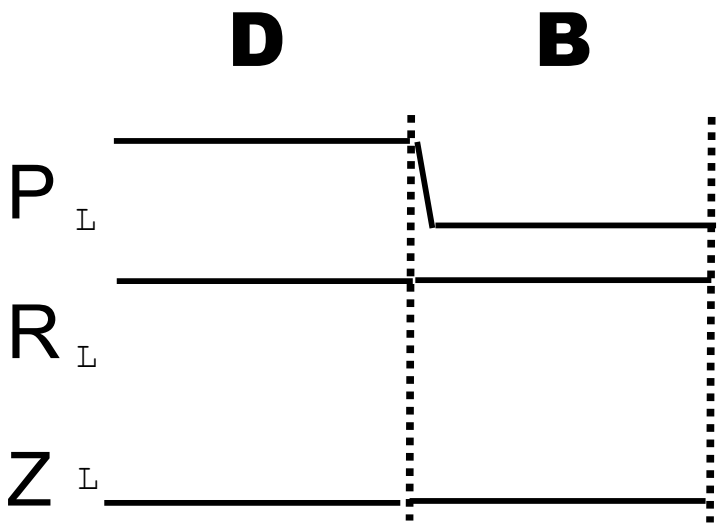
P R Z		00	01	11	10
		P R			
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C	E,1			C,1
1 1, 0	D			D,0	
0 0, 1	E	E,1			

Pulse-Catching – Flow Table



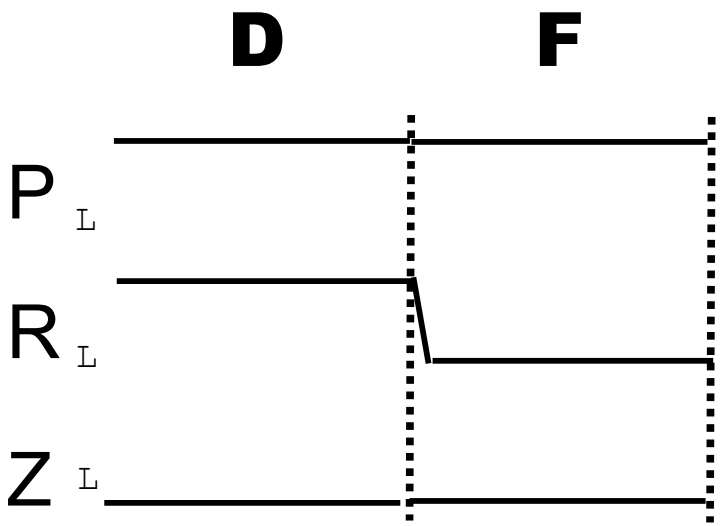
P R Z		00	01	11	10
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C	E,1		D,0	C,1
1 1, 0	D			D,0	
0 0, 1	E	E,1			

Pulse-Catching – Flow Table



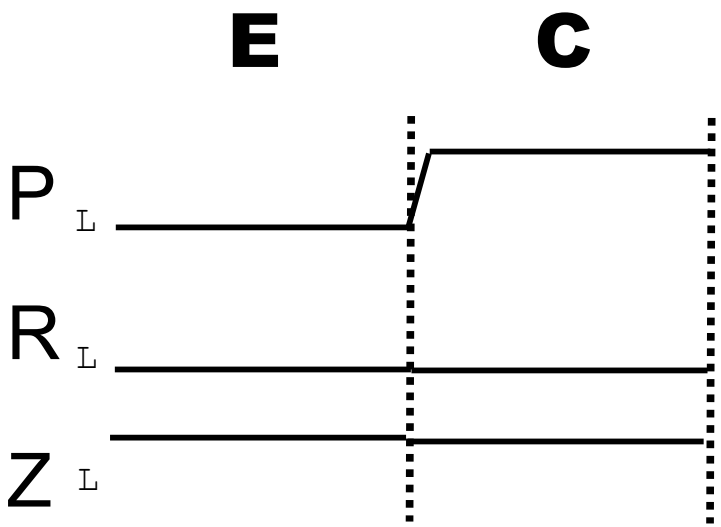
		P R			
P R Z		00	01	11	10
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C	E,1		D,0	C,1
1 1, 0	D		B,0	D,0	
0 0, 1	E	E,1			

Pulse-Catching – Flow Table



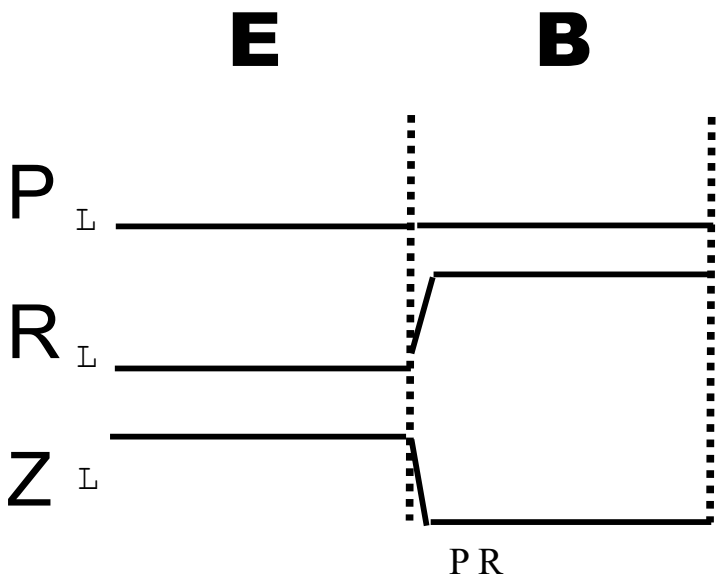
		P R			
P R Z		00	01	11	10
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C	E,1		D,0	C,1
1 1, 0	D		B,0	D,0	F,0
0 0, 1	E	E,1			
1 0, 0	F				F,0

Pulse-Catching – Flow Table



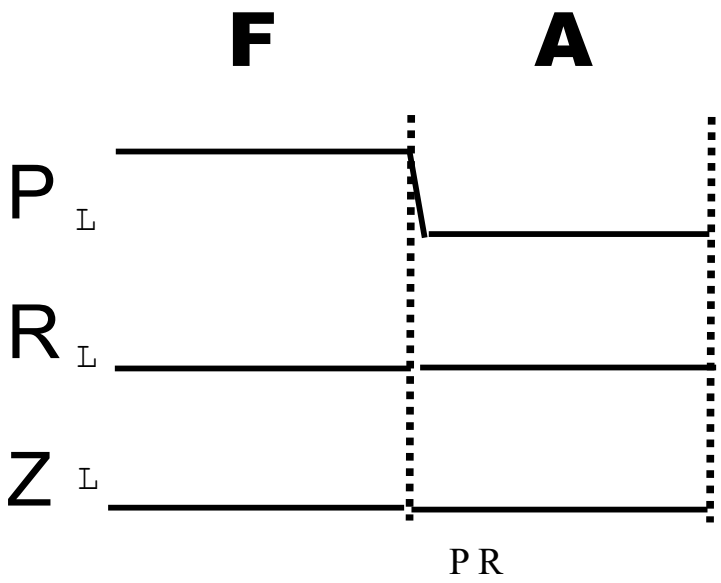
P R Z		00	01	11	10
		P R			
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C	E,1		D,0	C,1
1 1, 0	D		B,0	D,0	F,0
0 0, 1	E	E,1			C,1
1 0, 0	F				F,0

Pulse-Catching – Flow Table



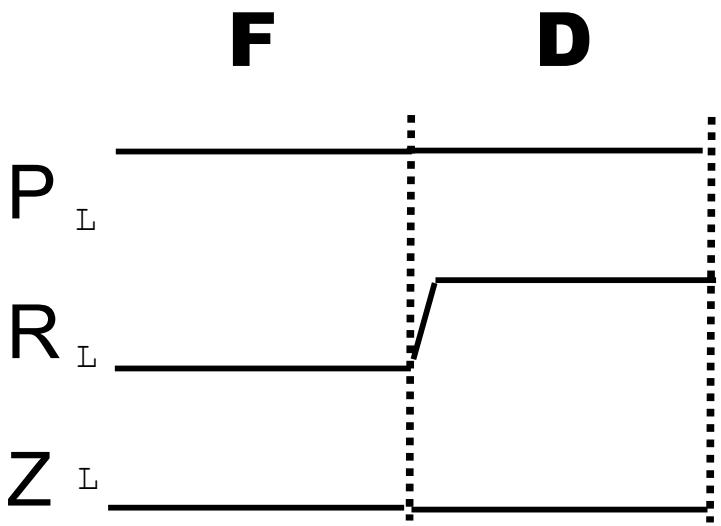
P R Z		00	01	11	10
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C	E,1		D,0	C,1
1 1, 0	D		B,0	D,0	F,0
0 0, 1	E	E,1	B,0		C,1
1 0, 0	F				F,0

Pulse-Catching – Flow Table



P R Z		00	01	11	10
		P R			
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C	E,1		D,0	C,1
1 1, 0	D		B,0	D,0	F,0
0 0, 1	E	E,1	B,0		C,1
1 0, 0	F	A,0			F,0

Pulse-Catching – Flow Table



P R Z		00	01	11	10
		P R			
0 0, 0	A	A,0	B,0		C,1
0 1, 0	B	A,0	B,0	D,0	
1 0, 1	C	E,1		D,0	C,1
1 1, 0	D		B,0	D,0	F,0
0 0, 1	E	E,1	B,0		C,1
1 0, 0	F	A,0		D,0	F,0

Pulse-Catching – Flow Table

- The primitive flow table

State		P	R	
	00	01	11	10
Idle = A	A,0	B,0	- -	C,1
Res1 = B	A,0	B,0	D,0	- -
Pls1 = C	E,1	- -	D,0	C,1
Res2 = D	- -	B,0	D,0	F,0
Pls2 = E	E,1	B,0	- -	C,1
Plsn = F	A,0	- -	D,0	F,0

Pulse-Catching – Implicant Table

State		P	R	
	00	01	11	10
Idle = A	A,0	B,0	- -	C,1
Res1 = B	A,0	B,0	D,0	- -
Pls1 = C	E,1	- -	D,0	C,1
Res2 = D	- -	B,0	D,0	F,0
Pls2 = E	E,1	B,0	- -	C,1
Plsn = F	A,0	- -	D,0	F,0

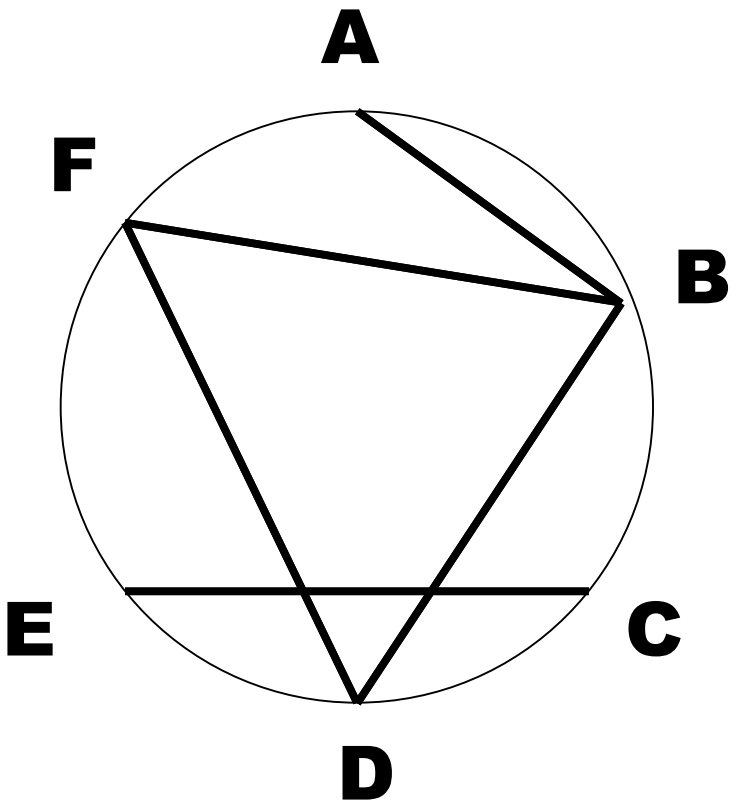
B	A,0=A,0 B,0=B,0 -- = D,0 C,1 = --				
C	A,0=E,1	A,0=E,1			
D	A,0=-- B,0=B,0 --=D,0 C,1=F,0	A,0=-- B,0=B,0 D,0=D,0 --=F,0	E,1=-- --=B,0 D,0=D,0 C,1=F,0		
E	A,0=E,1	A,0=E,1	E,1=E,1 --=B,0 D,0=-- C,1=C,1	--=E,1 B,0=B,0 D,0=-- F,0=C,1	
F	A,0=A,0 B,0=-- --=D,0 C,1=F,0	A,0=A,0 B,0=-- D,0=D,0 --=F,0	E,1=A,0	--=A,0 B,0=-- D,0=D,0 F,0=F,0	E,1=A,0
	A	B	C	D	E

Pulse-Catching – Implicant Table

B	A,0=A,0 B,0=B,0 -- = D,0 C,1 = --				
C	A,0=E,1	A,0=E,1			
D	A,0=-- B,0=B,0 --=D,0 C,1=F,0	A,0=-- B,0=B,0 D,0=D,0 --=F,0	E,1=-- --=B,0 D,0=D,0 C,1=F,0		
E	A,0=E,1	A,0=E,1	E,1=E,1 --=B,0 D,0=-- C,1=C,1	--=E,1 B,0=B,0 D,0=-- F,0=C,1	
F	A,0=A,0 B,0=-- --=D,0 C,1=F,0	A,0=A,0 B,0=-- D,0=D,0 --=F,0	E,1=A,0	--=A,0 B,0=-- D,0=D,0 F,0=F,0	E,1=A,0
	A	B	C	D	E

Pulse-Catching – Chord Graph

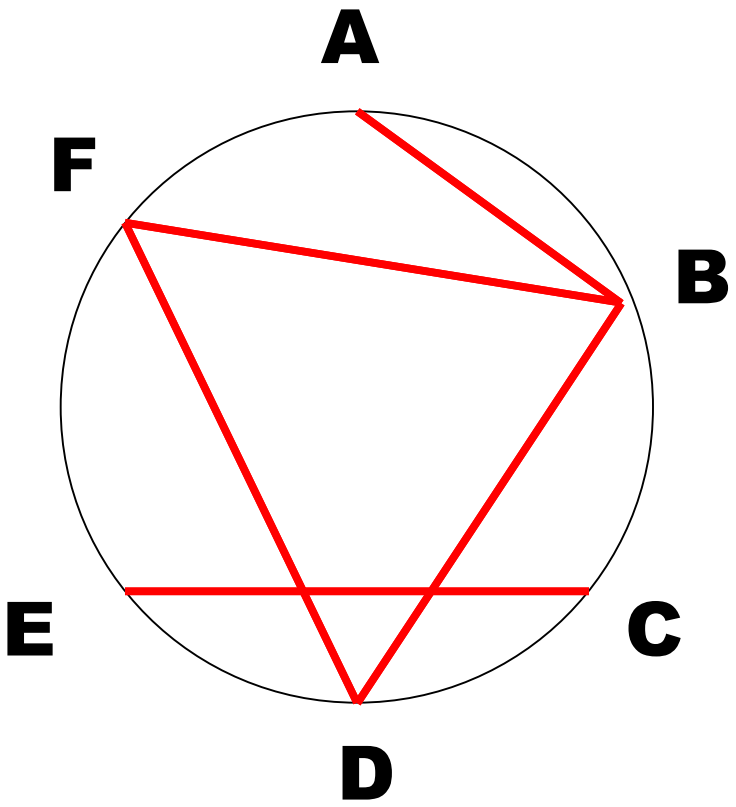
B	A,0=A,0 B,0=B,0 -- = D,0 C,1 = --				
C	A,0=E,1	A,0=E,1			
D	A,0=-- B,0=B,0 --=D,0 C,1=F,0	A,0=-- B,0=B,0 D,0=D,0 --=F,0	E,1=-- --=B,0 D,0=D,0 C,1=F,0		
E	A,0=E,1	A,0=E,1	E,1=E,1 --=B,0 D,0=-- C,1=C,1	--=E,1 B,0=B,0 D,0=-- F,0=C,1	
F	A,0=A,0 B,0=-- --=D,0 C,1=F,0	A,0=A,0 B,0=-- D,0=D,0 --=F,0	E,1=A,0	--=A,0 B,0=-- D,0=D,0 F,0=F,0	E,1=A,0
	A	B	C	D	E



Pulse-Catching – Chord Graph

B	A,0=A,0 B,0=B,0 -- = D,0 C,1 = --				
C	A,0=E,1	A,0=E,1			
D	A,0=-- B,0=B,0 --=D,0 C,1=F,0	A,0=-- B,0=B,0 D,0=D,0 --=F,0	E,1=-- --=B,0 D,0=D,0 C,1=F,0		
E	A,0=E,1	A,0=E,1	E,1=E,1 --=B,0 D,0=-- C,1=C,1	--=E,1 B,0=B,0 D,0=-- F,0=C,1	
F	A,0=A,0 B,0=-- --=D,0 C,1=F,0	A,0=A,0 B,0=-- D,0=D,0 --=F,0	E,1=A,0	--=A,0 B,0=-- D,0=D,0 F,0=F,0	E,1=A,0
	A	B	C	D	E

Can't take the large 4-clique.
Can take 3-clique BDF but can't include A-B because is not fully compatible



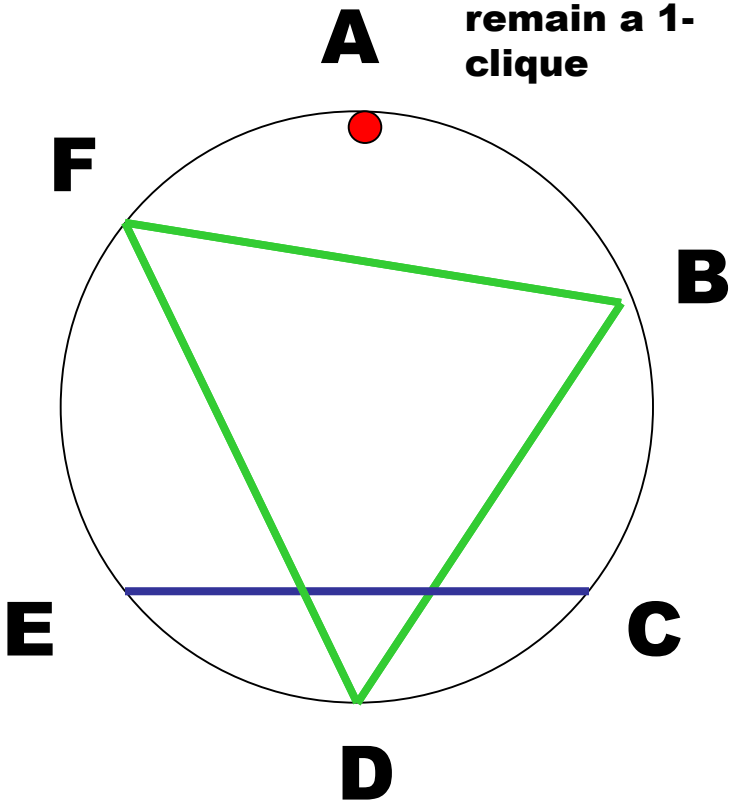
Pulse-Catching – Chord Graph

B	A,0=A,0 B,0=B,0 -- = D,0 C,1 = --				
C	A,0=E,1	A,0=E,1			
D	A,0=-- B,0=B,0 --=D,0 C,1=F,0	A,0=-- B,0=B,0 D,0=D,0 --=F,0	E,1=-- --=B,0 D,0=D,0 C,1=F,0		
E	A,0=E,1	A,0=E,1	E,1=E,1 --=B,0 D,0=-- C,1=C,1	--=E,1 B,0=B,0 D,0=-- F,0=C,1	
F	A,0=A,0 B,0=-- --=D,0 C,1=F,0	A,0=A,0 B,0=-- D,0=D,0 --=F,0	E,1=A,0	--=A,0 B,0=-- D,0=D,0 F,0=F,0	E,1=A,0
	A	B	C	D	E

This is valid

6 states to 3

State A cannot be combined and must remain a 1-clique

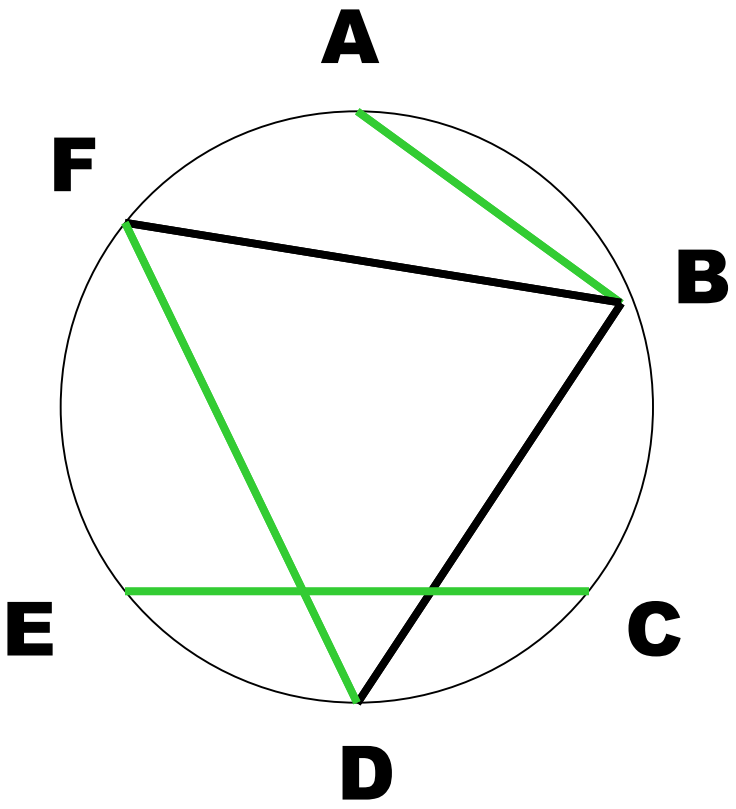


Pulse-Catching – Chord Graph

B	A,0=A,0 B,0=B,0 -- = D,0 C,1 = --				
C	A,0=E,1	A,0=E,1			
D	A,0=-- B,0=B,0 --=D,0 C,1=F,0	A,0=-- B,0=B,0 D,0=D,0 --=F,0	E,1=-- --=B,0 D,0=D,0 C,1=F,0		
E	A,0=E,1	A,0=E,1	E,1=E,1 --=B,0 D,0=-- C,1=C,1	--=E,1 B,0=B,0 D,0=-- F,0=C,1	
F	A,0=A,0 B,0=-- --=D,0 C,1=F,0	A,0=A,0 B,0=-- D,0=D,0 --=F,0	E,1=A,0	--=A,0 B,0=-- D,0=D,0 F,0=F,0	E,1=A,0
	A	B	C	D	E

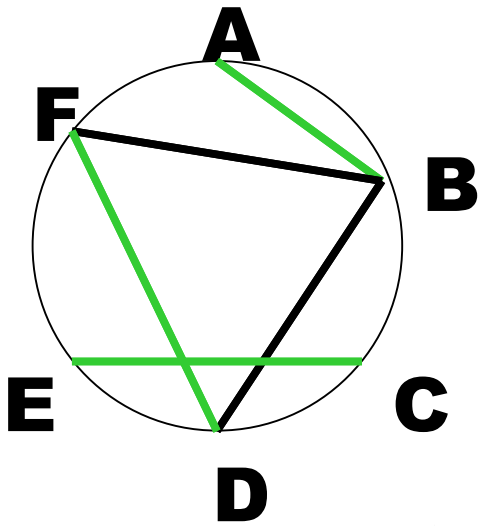
Another valid covering

6 states to 3



Pulse-Catching – Chord Graph Combining

B	A,0=A,0 B,0=B,0 -- = D,0 C,1 = --				
C	A,0=E,1	A,0=E,1			
D	A,0=-- B,0=B,0 --=D,0 C,1=F,0	A,0=-- B,0=B,0 D,0=D,0 --=F,0	E,1=-- --=B,0 D,0=D,0 C,1=F,0		
E	A,0=E,1	A,0=E,1	E,1=E,1 --=B,0 D,0=-- C,1=C,1	--=E,1 B,0=B,0 D,0=-- F,0=C,1	
F	A,0=A,0 B,0=-- --=D,0 C,1=F,0	A,0=A,0 B,0=-- D,0=D,0 --=F,0	E,1=A,0	--=A,0 B,0=-- D,0=D,0 F,0=F,0	E,1=A,0
	A	B	C	D	E



State		P	R	
	00	01	11	10
Idle = A=B	A,0	B,0	D,0	C,1
Res1 = B	A,0	B,0	D,0	- -
Pls1 = C=E	E,1	B,0	D,0	C,1
Res2 = D=F	A,0	B,0	D,0	F,0
Pls2 = E	E,1	B,0	- -	C,1
Plsn = F	A,0	- -	D,0	F,0

Removed states are grayed out

Pulse-Catching – Rename combined states

State		P	R	
	00	01	11	10
Idle = A=B	A,0	A,0	D,0	C,1
Res1 = B	A,0	B,0	D,0	- -
Pls1 = C=E	C,1	A,0	D,0	C,1
Res2 = D=F	A,0	A,0	D,0	D,0
Pls2 = E	E,1	B,0	- -	C,1
Plsn = F	A,0	- -	D,0	F,0

Active states are renamed so that only equivalent active states appear in table, namely states A, C, D.

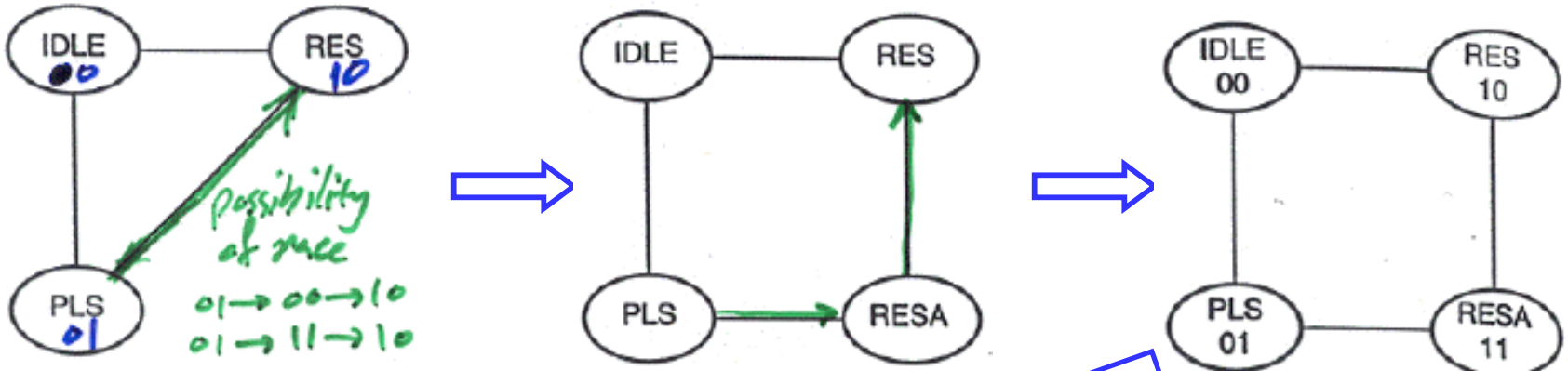
Pulse-Catching – Reduced states

State		P	R	
	00	01	11	10
Idle = A=B	A,0	A,0	D,0	C,1
Pls1 = C=E	C,1	A,0	D,0	C,1
Res2 = D=F	A,0	A,0	D,0	D,0

	P R				
S	00	01	11	10	Z
IDLE	(IDLE)	(IDLE)	RES	PLS	0
PLS	(PLS)	IDLE	RES	(PLS)	1
RES	IDLE	IDLE	(RES)	(RES)	0
	S*				

State Assignment

- No race-free assignment using 2-bits codes exist. Add one transitional state (IDLE) to have it.

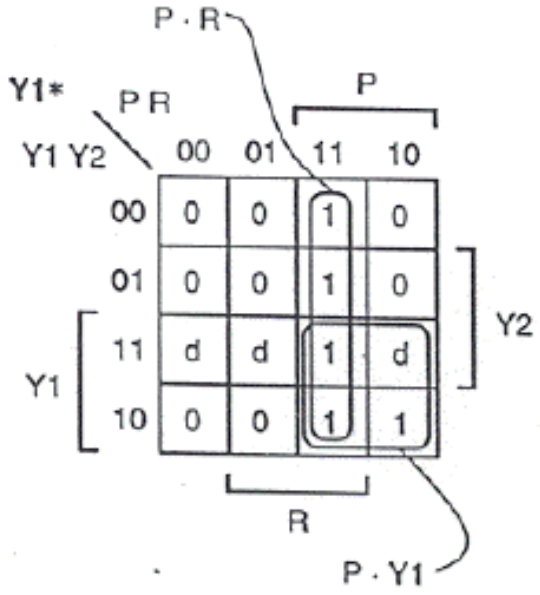


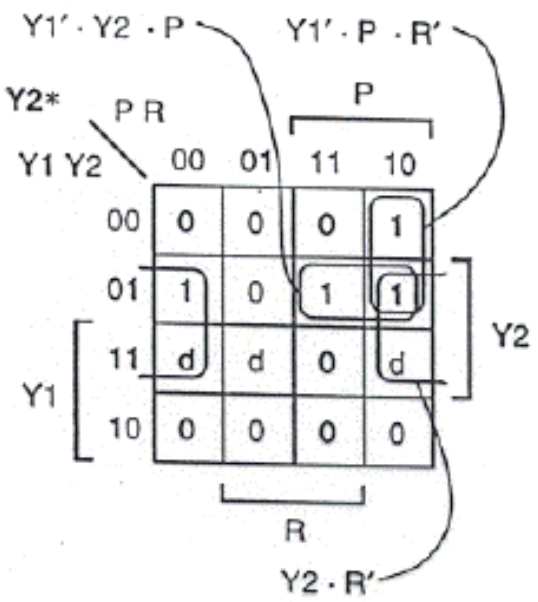
	P R				
S	00	01	11	10	Z
IDLE	(IDLE)	(IDLE)	RES	PLS	0
PLS	(PLS)	IDLE	RESA	(PLS)	1
RESA	—	—	RES	—	–
RES	IDLE	IDLE	(RES)	(RES)	0
			S*		

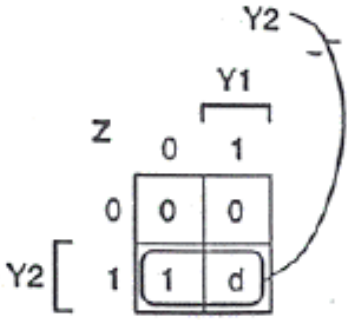
	P R				
Y1 Y2	00	01	11	10	Z
00	(00)	(00)	10	01	0
01	(01)	00	11	(01)	1
11	—	—	10	—	–
10	00	00	(10)	(10)	0
			Y1* Y2*		

K-Map Optimization

Y1 Y2 \ P R	00	01	11	10	Z
00	(00)	(00)	10	01	0
01	(01)	00	11	(01)	1
11	—	—	10	—	–
10	00	00	(10)	(10)	0
	Y1* Y2*				







$$Y1^* = P \cdot R + P \cdot Y1$$

$$Y2^* = Y2 \cdot R' + Y1' \cdot Y2 \cdot P + Y1' \cdot P \cdot R'$$

$$Z = Y2$$

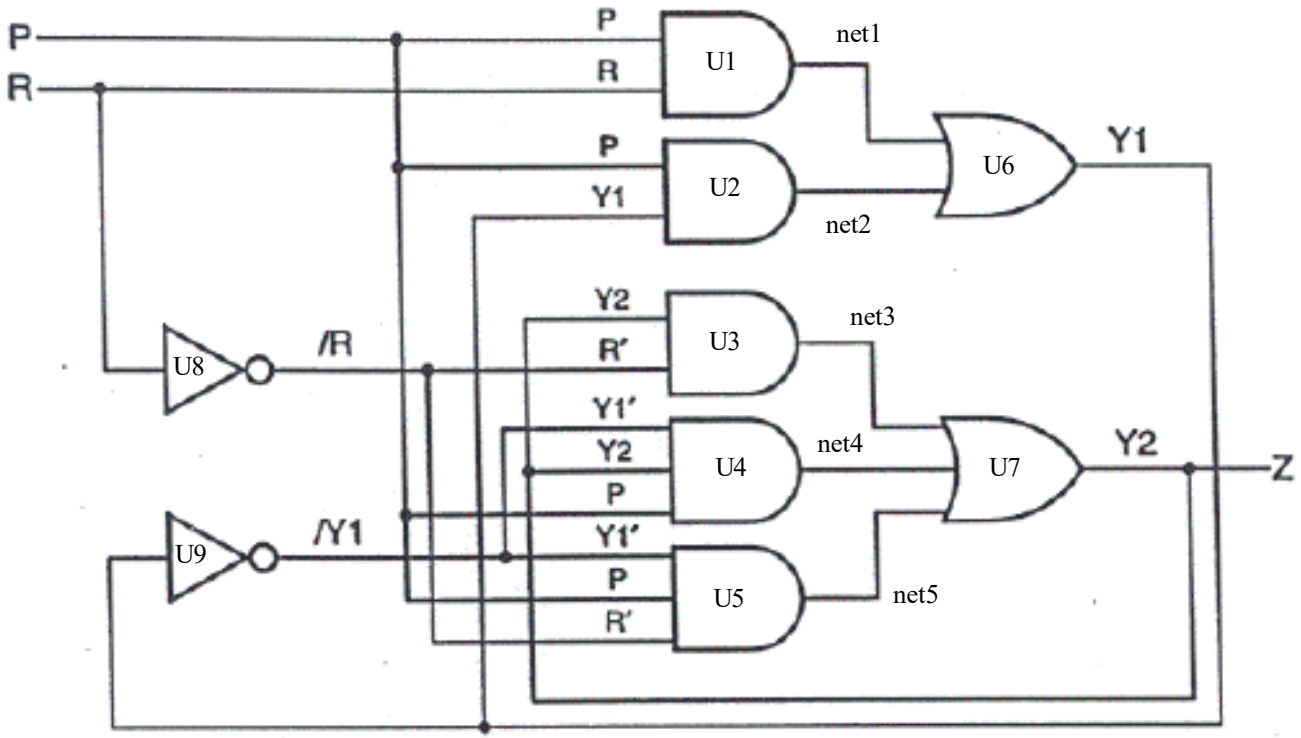
Circuit Implementation

- All functions are static hazard-free

$$Y1^* = P \cdot R + P \cdot Y1$$

$$Y2^* = Y2 \cdot R' + Y1' \cdot Y2 \cdot P + Y1' \cdot P \cdot R'$$

$$Z = Y2$$



Essential Hazards

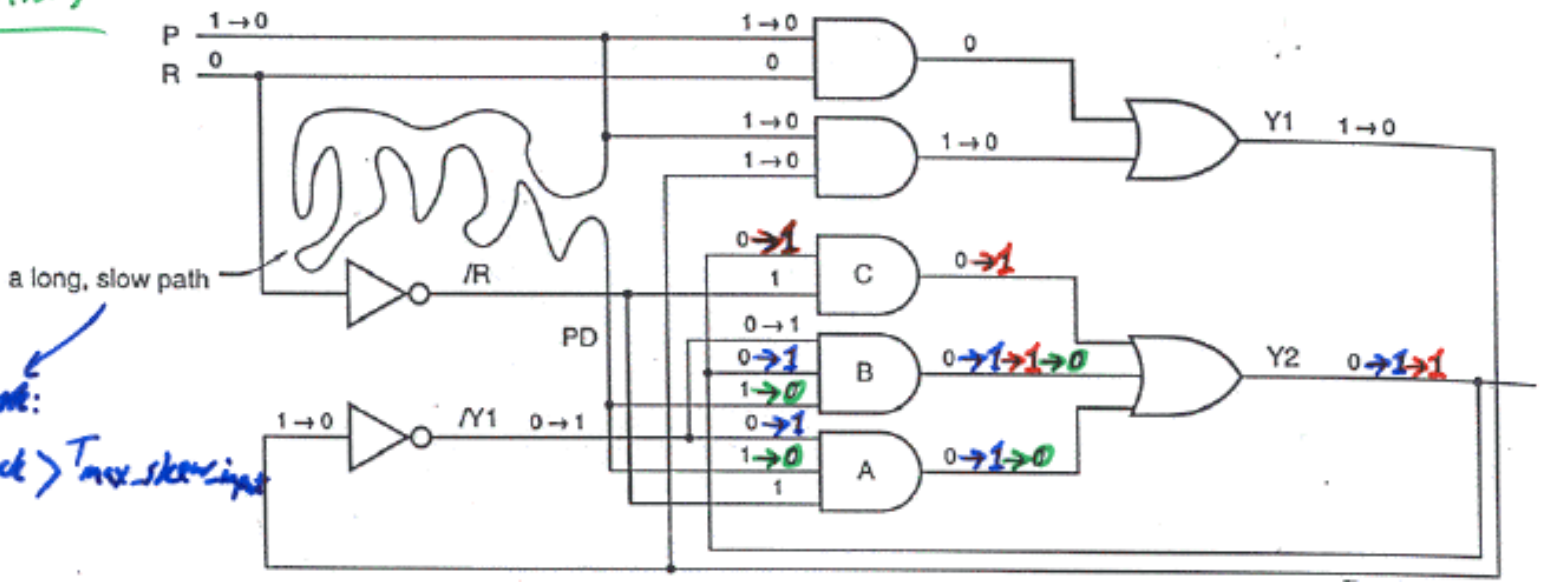
- Essential hazard occurs if the input change is not seen by all of the excitation logic before the resulting state transitions propagate back to the inputs.

- Essential hazards exists in a flow table if there is a stable state from which three consecutive changes in a single input variable will take the circuit to a different state than the first change alone.

Check for Essential Hazards

Order of transitions

① →
② →
③ →
④ →



For proper work:

$T_{min\ flop} + T_{feedback} > T_{max\ skew\ in\ gate}$

PR : 10 → 00

{ 1 transition (without timing skew): →
{ 3 transitions : 10 → 00 → 10 → 00 : →

{ P: ⊓_ (without skew)
{ P: ⊓⊓_ (with skew)

Y1 Y2	P R = 00	01	11	10	Z
00	(00)	(00)	10	01	0
01	(01)	00	11	(01)	1
11	—	—	10	—	–
10	00	00	(10)	(10)	0
	Y1* Y2*				

Avoiding Essential Hazards

- Essential hazards are called "essential" because they are inherent in the flow table for a particular sequential function, and will appear in any circuit realization of that function.

- Essential hazards can be masked only by controlling the delays in the circuit. Compare with static hazard in combinational logic, where we could eliminate hazards by adding consensus terms to a logic expression.

- In previous example, the only way to avoid the erroneous behavior is to ensure that changes in P arrive at the inputs of all the excitation logic before any changes in state variables do. That is $T_{min\_prop} + T_{feedback} > T_{max\_skew\_input}$

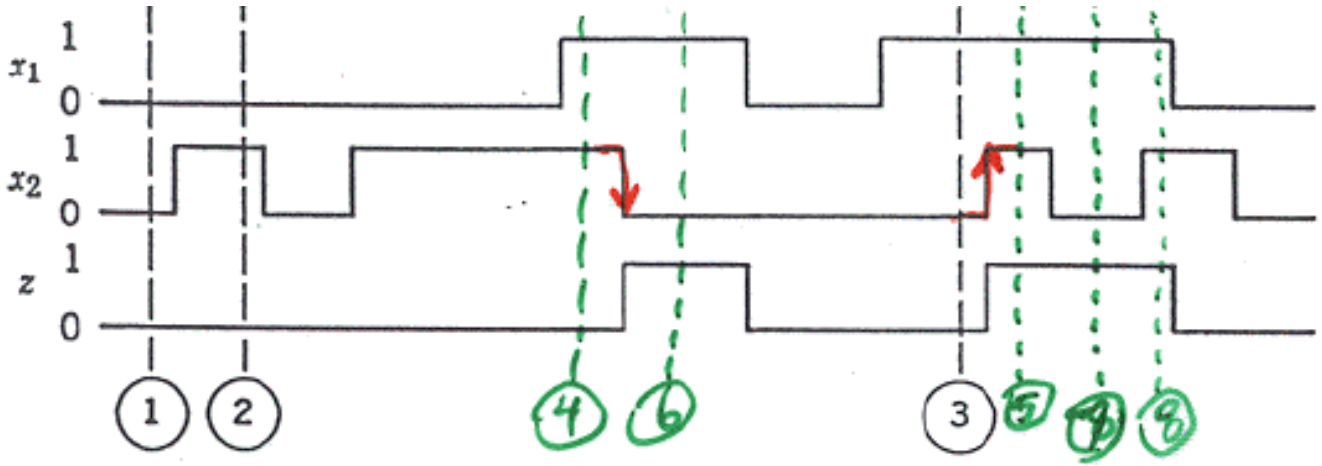
Design Procedure -
Double-Edge Pulse Catching Example

Review of Basic Design Steps

1. Construct the primitive flow table from the circuit's word description.
2. Minimize the number of states in the flow table.
3. Find a race-free assignment for the coded states, adding auxiliary states or splitting the states as required.
4. Construct the transition table.
5. Construct the excitation maps and obtain a hazard-free realization of the equations.
6. Draw the logic diagram.
7. Check for essential hazards. Modify the circuit if needed.

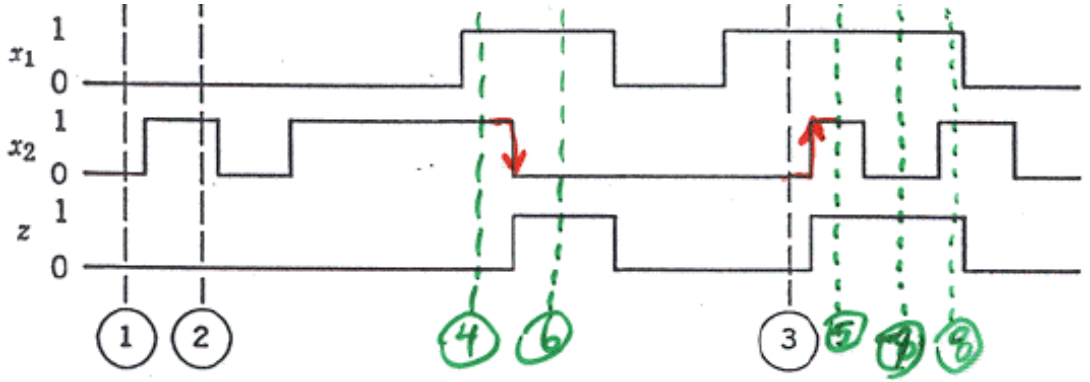
Double-Edge Pulse Catching Example

- Design a fundamental mode circuit with two inputs x_1 and x_2 and a single output z. The circuit output is to be z=0 whenever x_1=0. The first change (0→1 or 1→ 0) in the input x_2 occurring while x_1=1, will cause the circuit output to go to 1. The output is then to remain 1 until x_1 returns to 0. Changes in inputs will always be separated sufficiently in time to permit the circuit to stabilize before a second input transition takes place.
- Typical Behavior



Double-Edge PC Example (cont.)

- Typical Behavior



- Flow Table

Blanks are don't cares

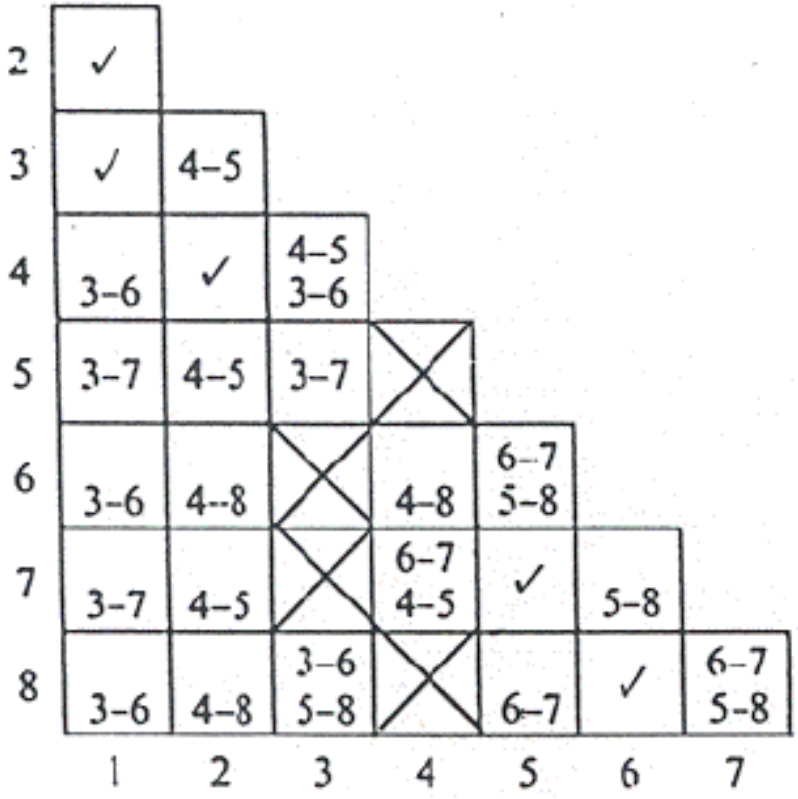
x_2x_1	00	01	11	10
①	3	–	2	
1	–	4	②	
1	③	5	–	
–	6	④	2	
–	7	⑤	2	
1	⑥	8	–	
1	⑦	5	–	
–	6	⑧	2	

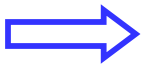
x_2x_1	00	01	11	10
0				
				0
	0			
			0	
			1	
	1			
	1			
			1	

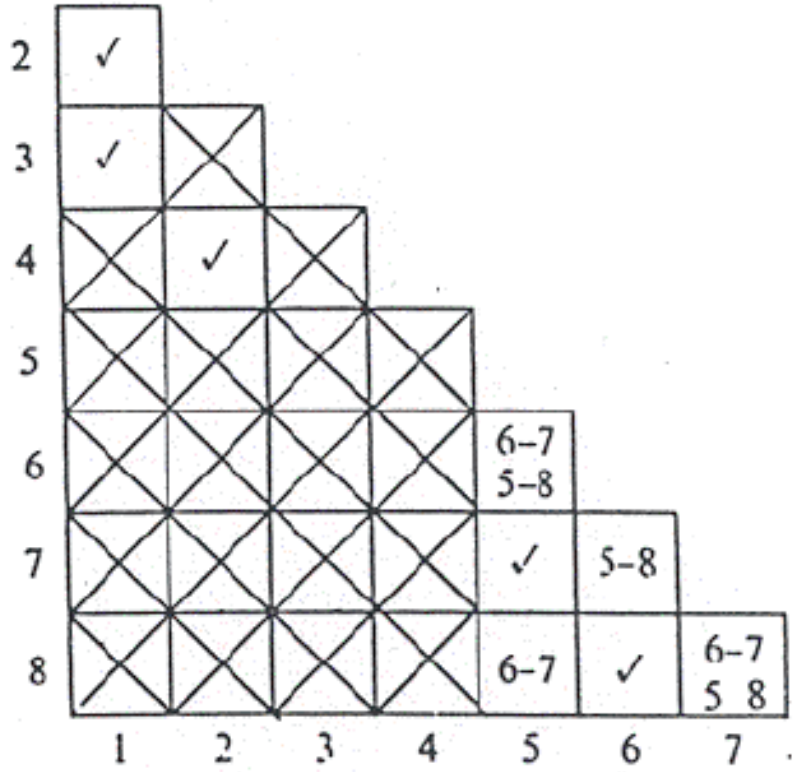
q^v	x_2x_1 00	01	11	10	x_2x_1 00	01	11	10
①	①	3	–	2	0			
②	1	–	4	②				0
③	1	③	5	–		0		
④	–	6	④	2			0	
⑤	–	7	⑤	2			1	
⑥	1	⑥	8	–		1		
⑦	1	⑦	5	–		1		
⑧	–	6	⑧	2			1	
	q^{v+1}				z			

State Minimization

- Implication Chart (Initial and Final)

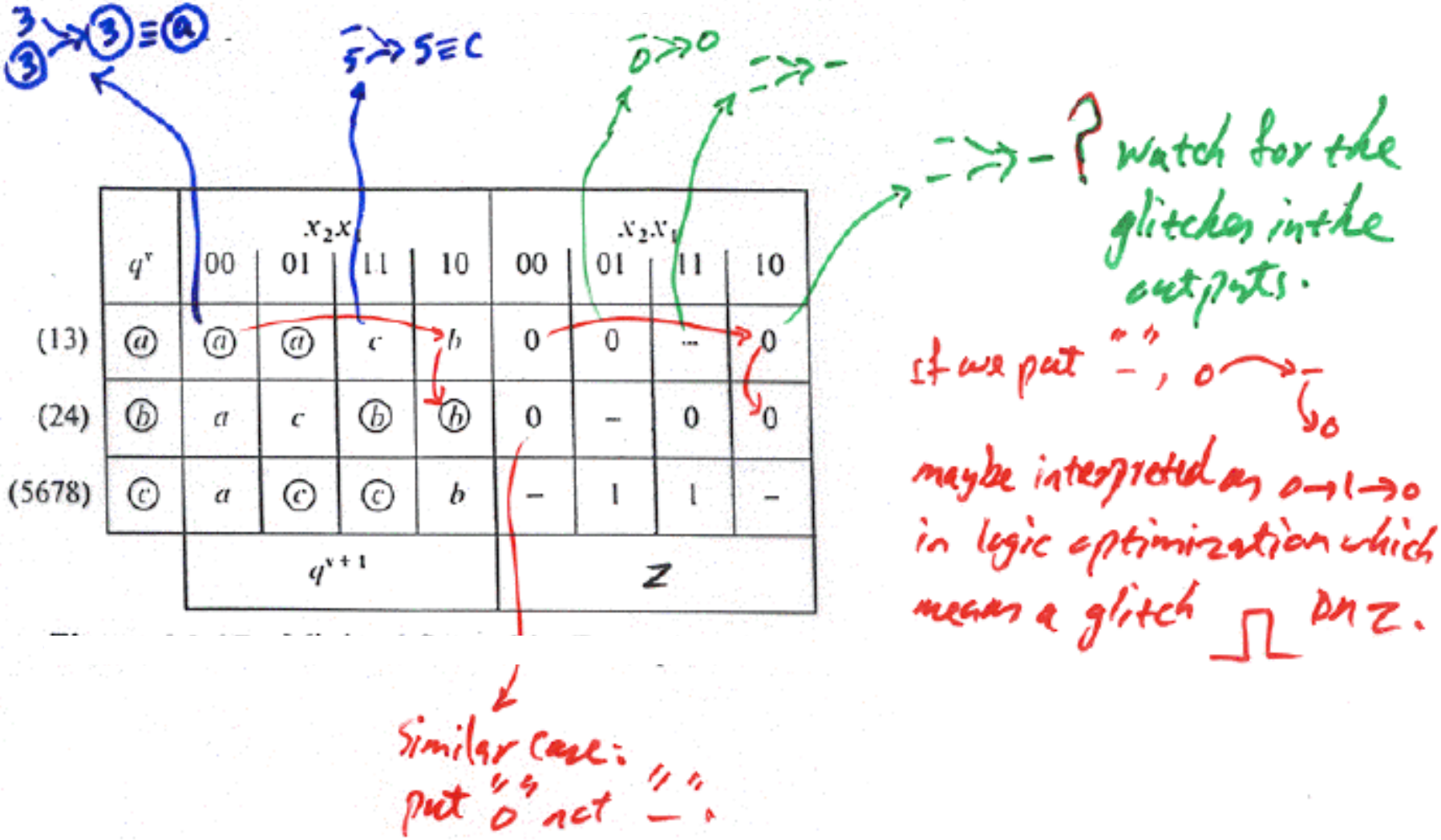






- All compatibles (e.g. cliques in graph representation): (12)(13)(24)(5678)
- Smallest collection (e.g. covering table): (13)(24)(5678)

Minimal Flow Table

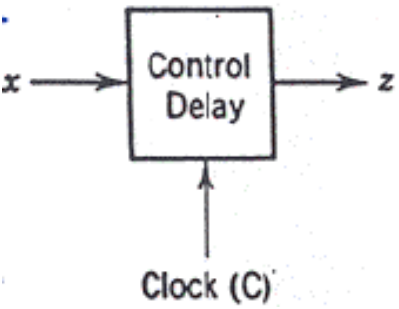


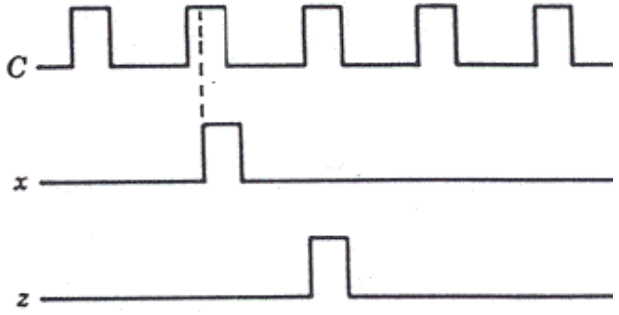
	q^y	x_2x_1 = 00	01	11	10	x_2x_1 = 00	01	11	10
(13)	ⓐ	ⓐ	ⓐ	c	b	0	0	–	0
(24)	ⓑ	a	c	ⓑ	ⓑ	0	–	0	0
(5678)	ⓒ	a	ⓒ	ⓒ	b	–	1	1	–
		q^{y+1}				Z			

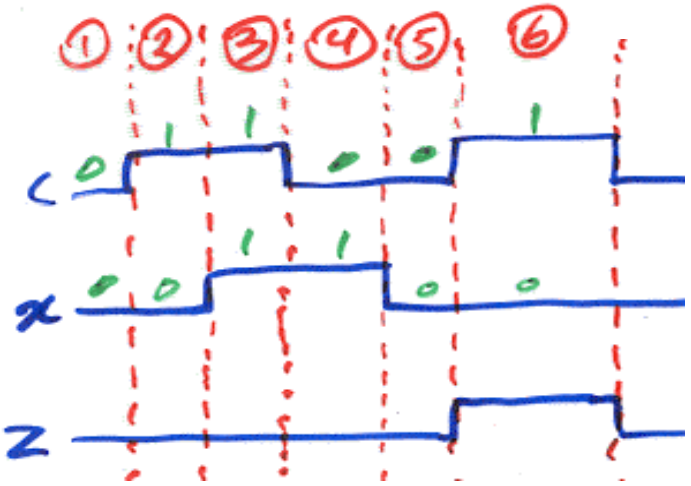
Design Procedure - Control Delay Example

Control Delay Example

- Design a circuit that has a control pulse input x, a clock input C and a control pulse output z. Pulses on line x will always be separated by several clock periods. Whenever a pulse occurs on line x, it will overlap a clock pulse and be of approximately the same width as a clock pulse. That is line z will only go to 1 after the clock has gone to 1 and will return to 0 only after the clock has returned to 0. For each input pulse, there is to be an output pulse on line z coinciding with the next clock pulse following the x pulse. Thus each x pulse results in a z pulse delayed by approximately one clock period.

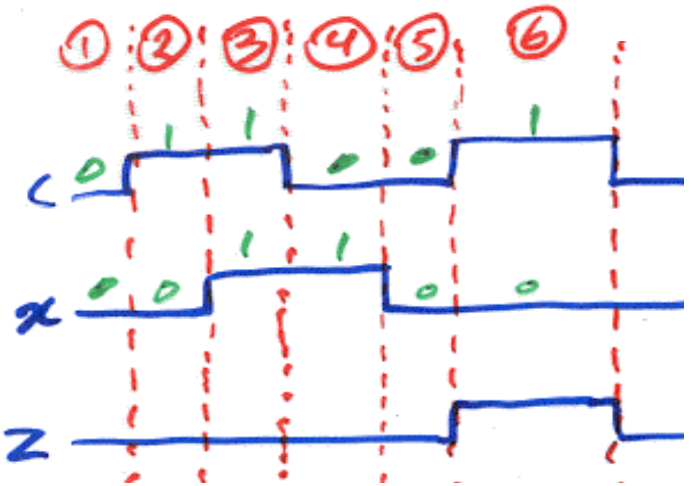






Control Delay Example (cont.)

- Typical Behavior



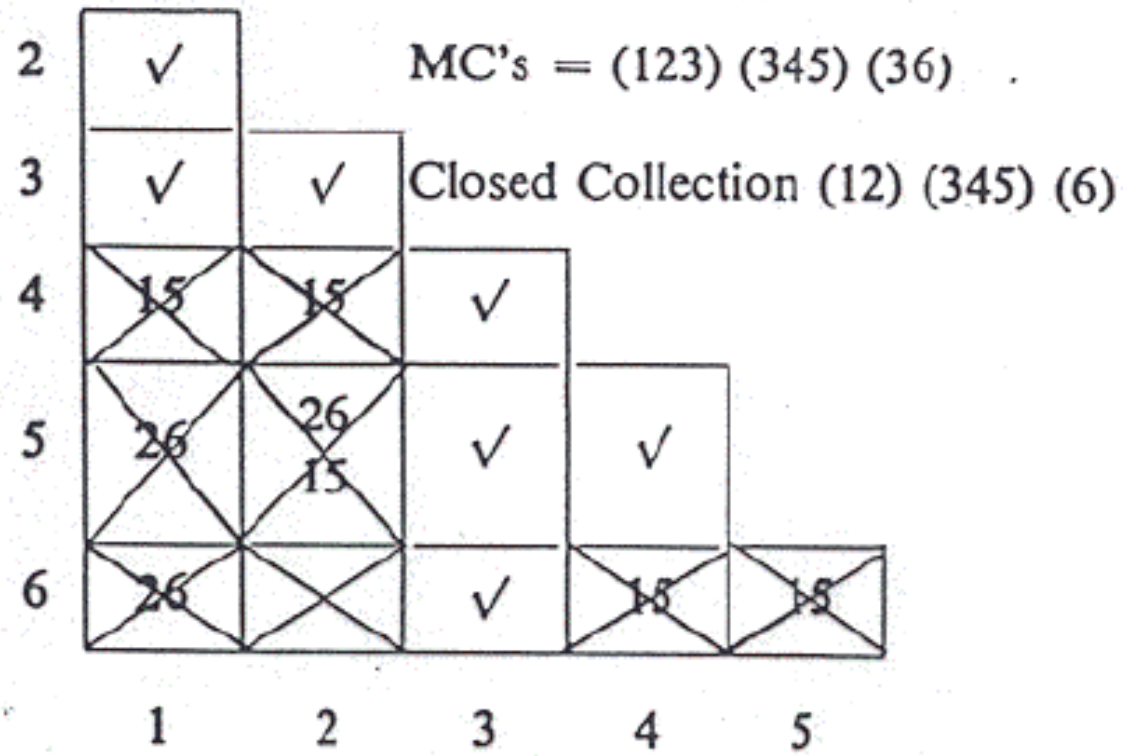
- Flow Table

Blanks are don't cares

State	xC = 00	01	11	10	00	01	11	10
1	①	2	–	–	0			
2	1	②	3	–		0		
3	–	–	③	4			0	
4	5	–	–	④				0
5	⑤	6	–	–	0			
6	1	⑥	–	–		1		
					z			

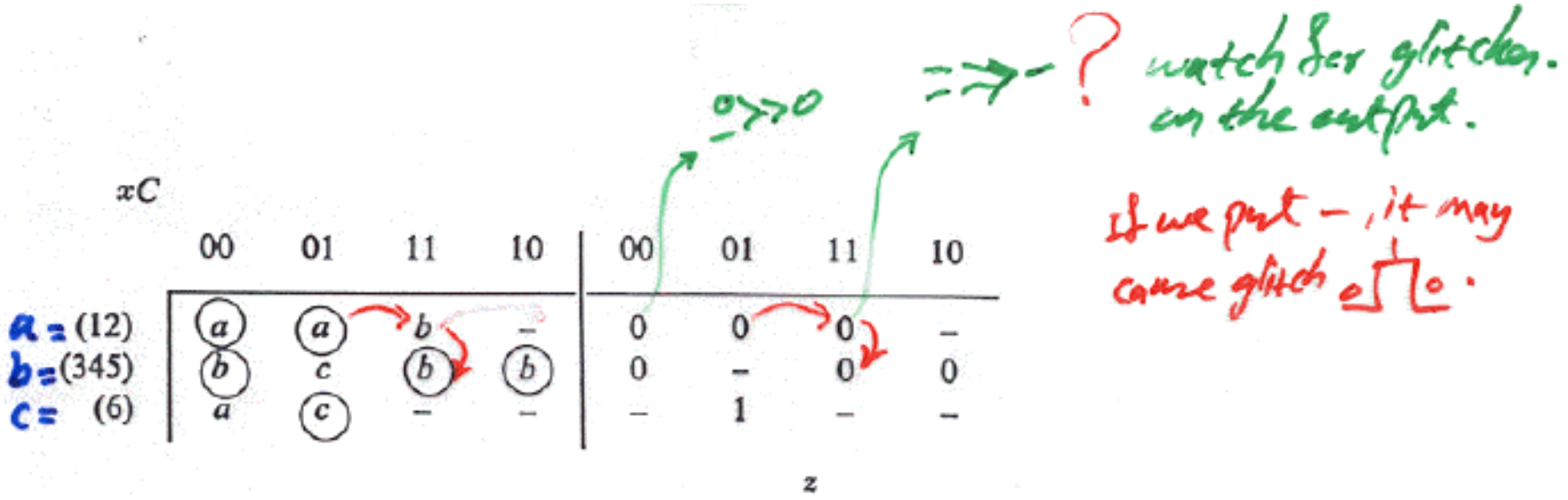
State Minimization

- Implication Chart



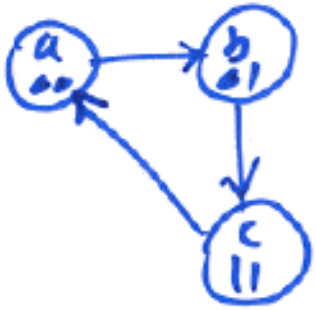
Minimal Flow Table

- Replace don't cares carefully to avoid glitches



State Assignment – Race Exists

- This is not a race-free assignment

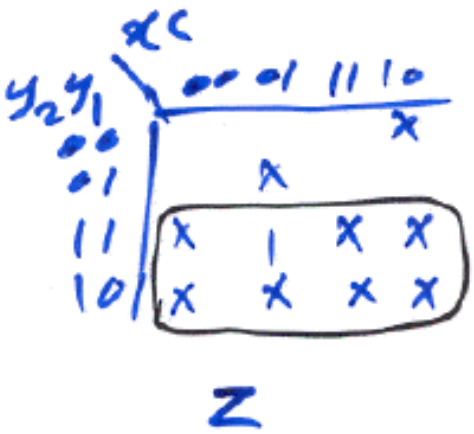


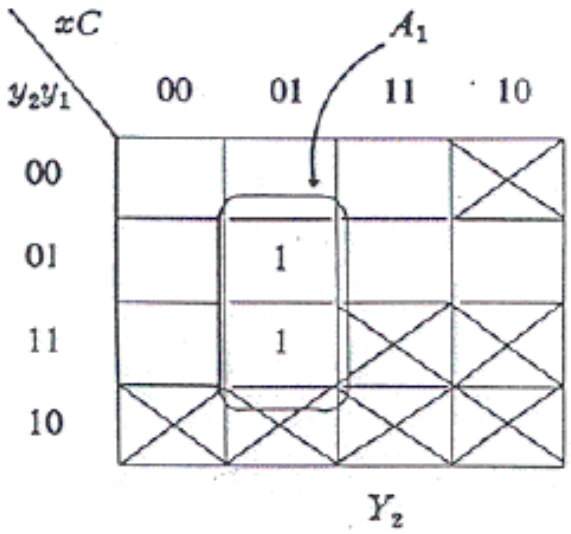
- State Assignment (Y_2Y_1)
- Critical race exists for
  - —11→01→00 or
  - —11→10→00

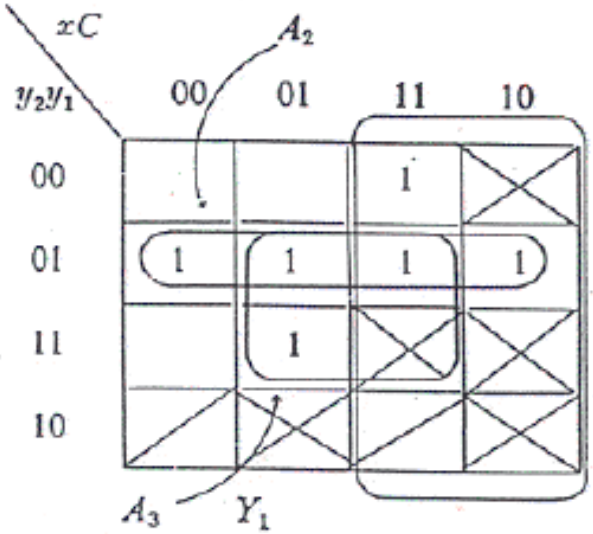
y_2y_1 \ xC	00	01	11	10
00	(00)	(00)	01	
01	(01)	11	(01)	(01)
11	00	(11)		
10				

State Assignment – Race Exists (cont.)

- K-Maps







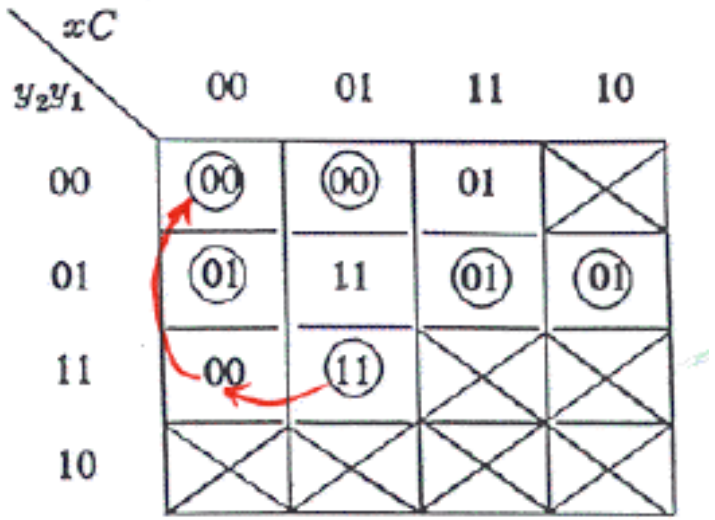
$$Y_2 = \bar{x}Cy_1$$

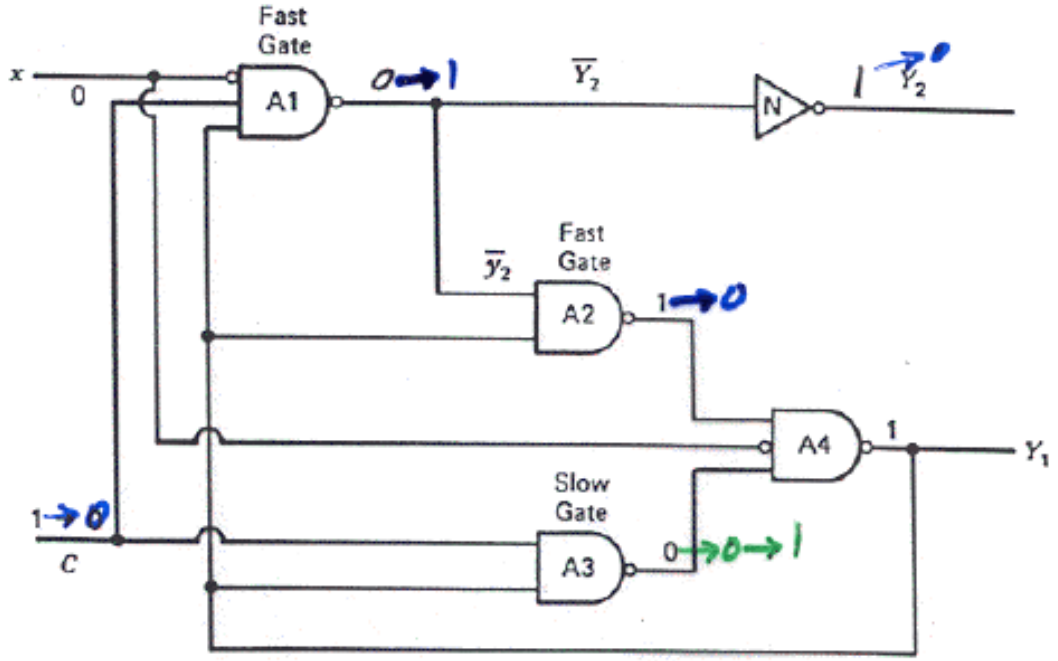
$$Y_1 = x + \bar{y}_2 y_1 + Cy_1$$

$$z = y_2$$

State Assignment – Race Exists (cont.)

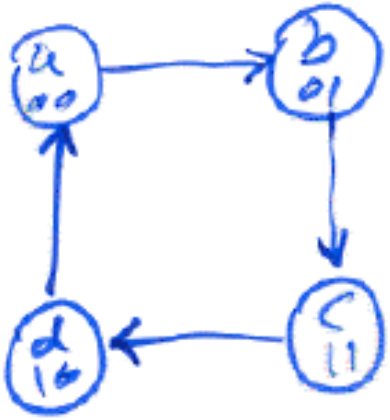
- If the gates have unequal delays, the change in C will first drive the output of A1 to 1, which will in turn drive A2 to 0 before the output of A3 goes to 1. As a result, Y_1 will not change and the circuit will make an erroneous transition to Y_2Y_1=01 (i.e. state 01 instead of 00 in the first column).





Race-Free Assignment

- This is a race-free assignment

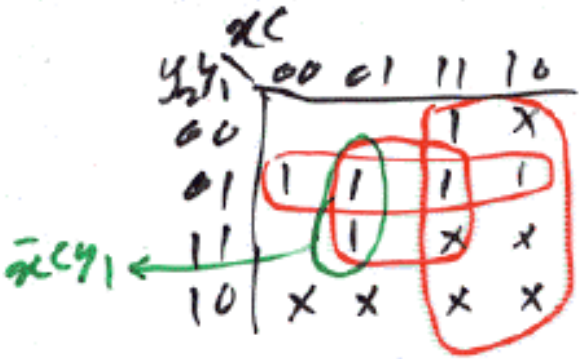


- State Assignment (Y_2Y_1)

y_2y_1 \ xC	00	01	11	10
00	(00)	(00)	01	
01	(01)	11	(01)	(01)
11	10	(11)		
10	00			

Race-Free Assignment (cont.)

- K-Maps

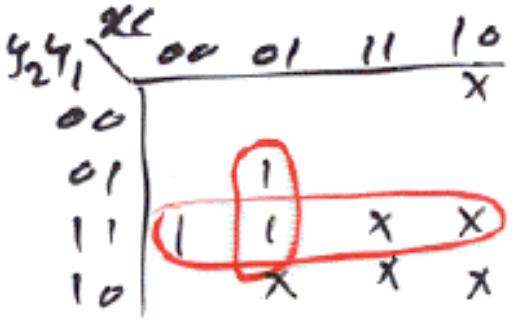


$Y_1 = x + \bar{y}_2 y_1 + Cy_1$

$= x + \bar{y}_2 y_1 + \bar{x}Cy_1$

$Y_1 = x + \bar{y}_2 y_1 + \bar{x}Cy_1$

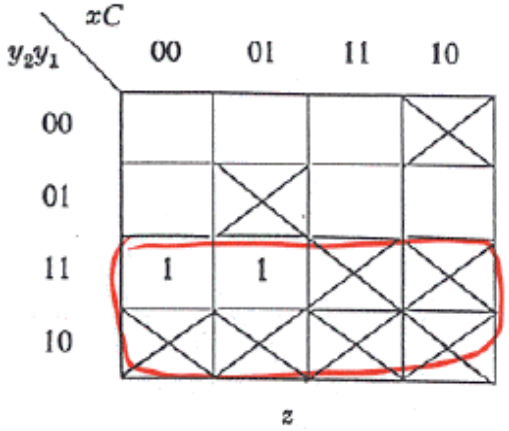
$= \overline{x \cdot \overline{\bar{y}_2 y_1} \cdot \overline{\bar{x}Cy_1}}$



$Y_2 = y_2y_1 + \bar{x}Cy_1$

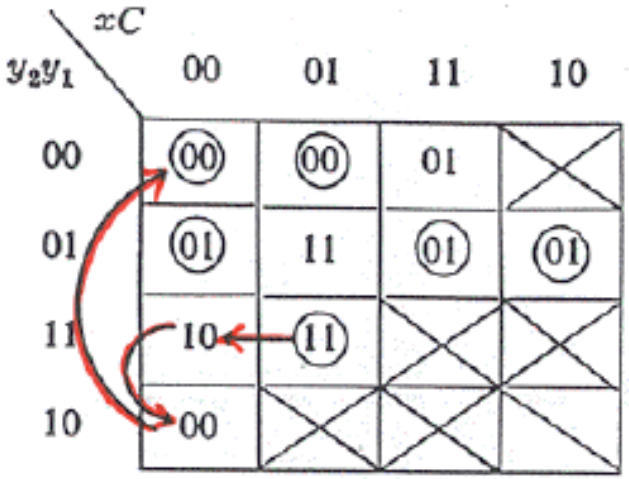
$= \overline{\overline{y_2y_1} \cdot \overline{\bar{x}Cy_1}}$

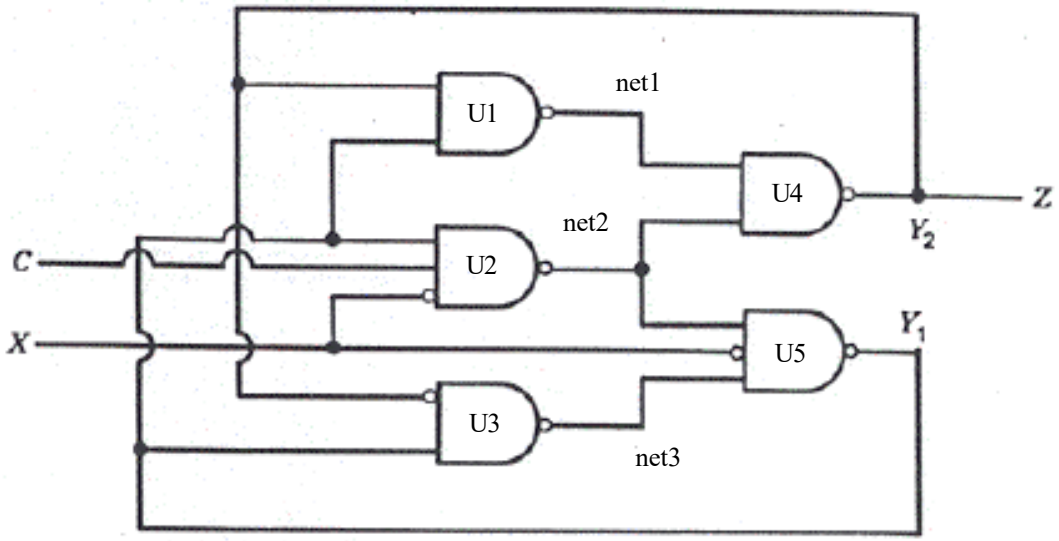
Z=Y_2



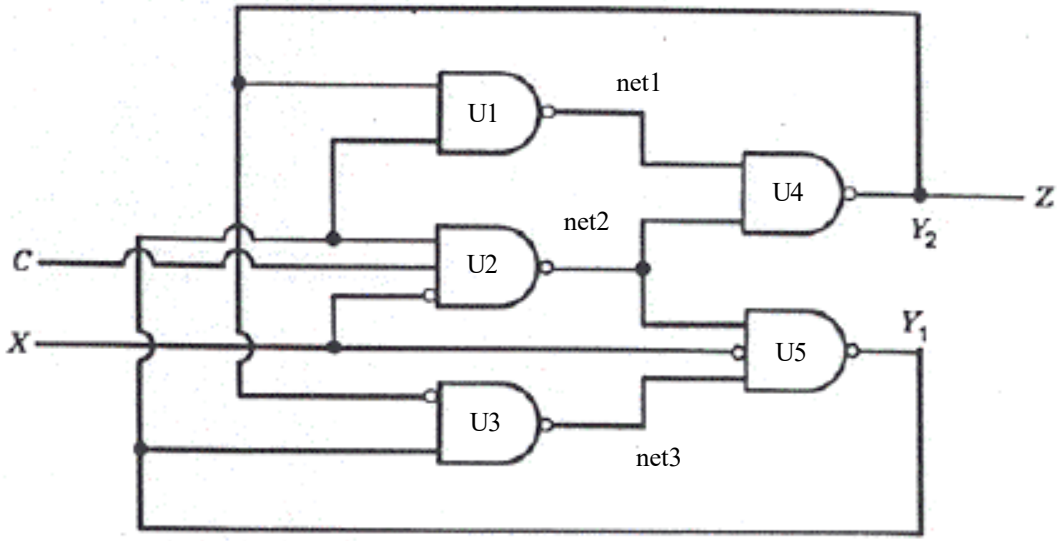
Race-Free Assignment (Cont.)

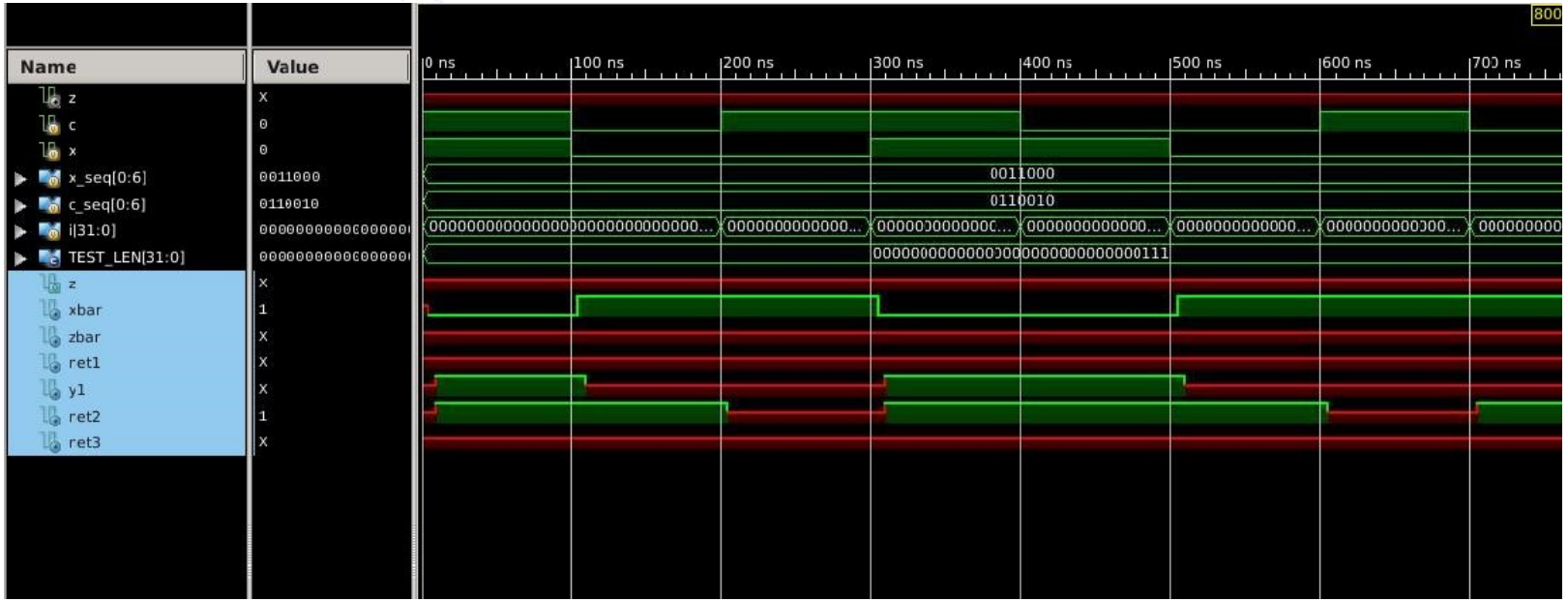
- This circuit will be race-free provided the input pulses satisfy the fundamental mode constraints.



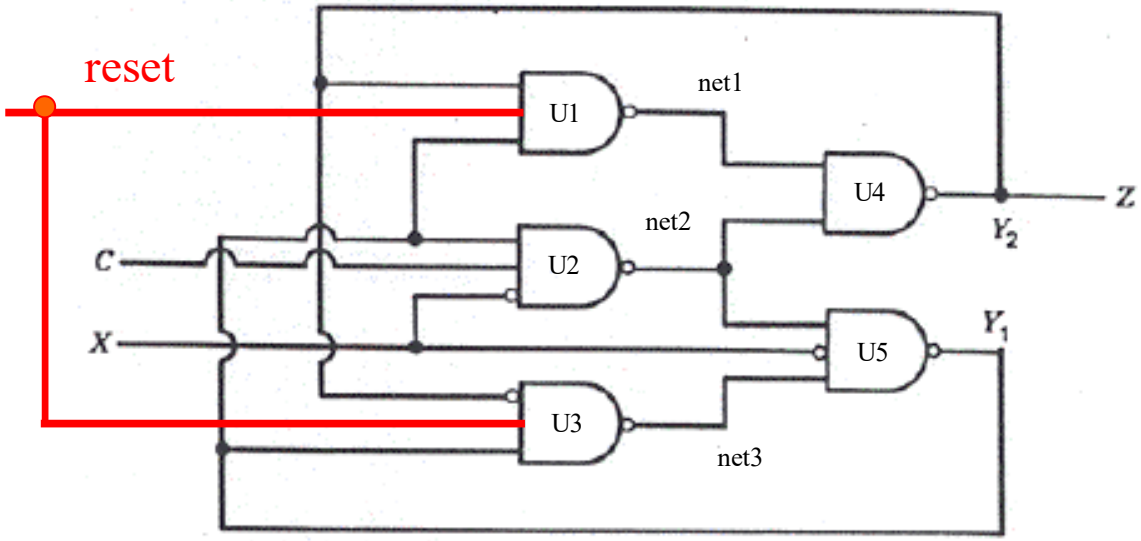


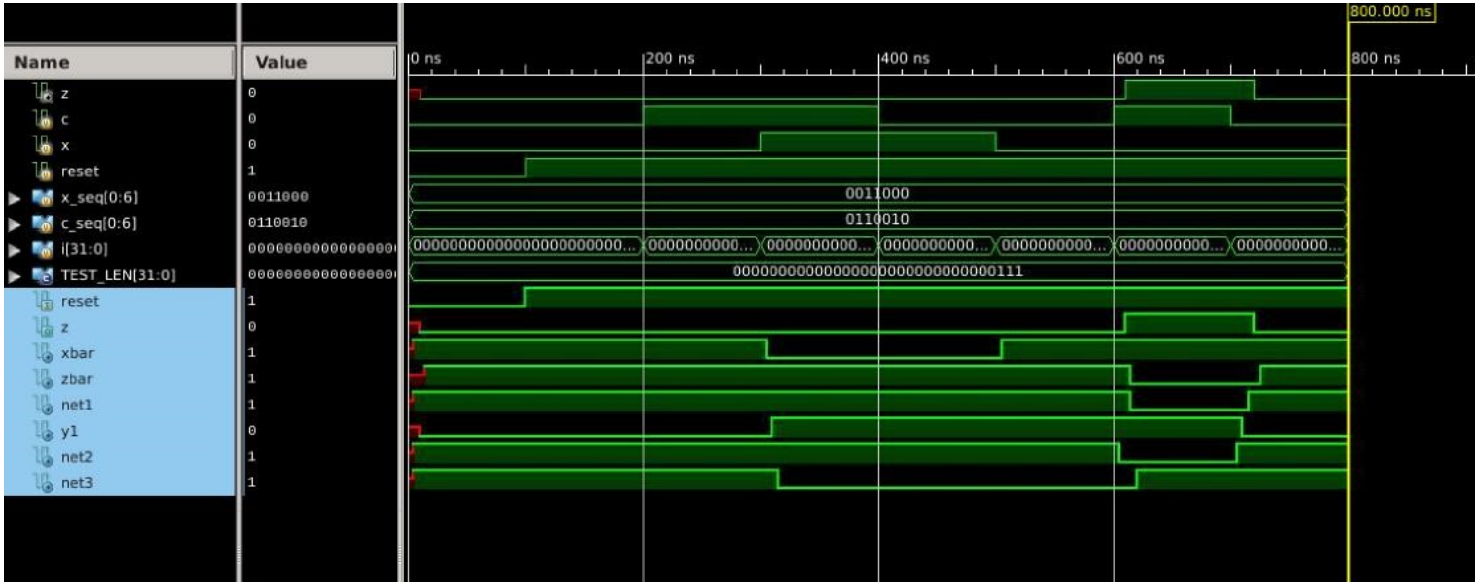
Simulation


U1
net1
U4
Z
net2
U2
C
Y_2
X
Y_1
U5
U3
net3


Name
Value
z
c
x
x_seq[0:6]
c_seq[0:6]
i[31:0]
TEST_LEN[31:0]
z
xbar
zbar
ret1
y1
ret2
ret3
0 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
0011000
0110010

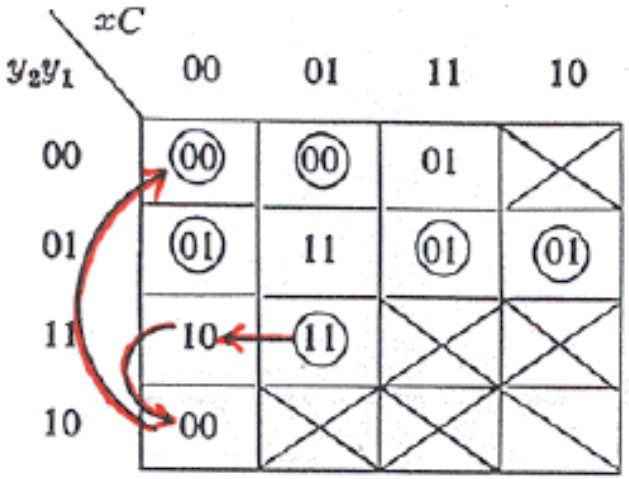
Simulation

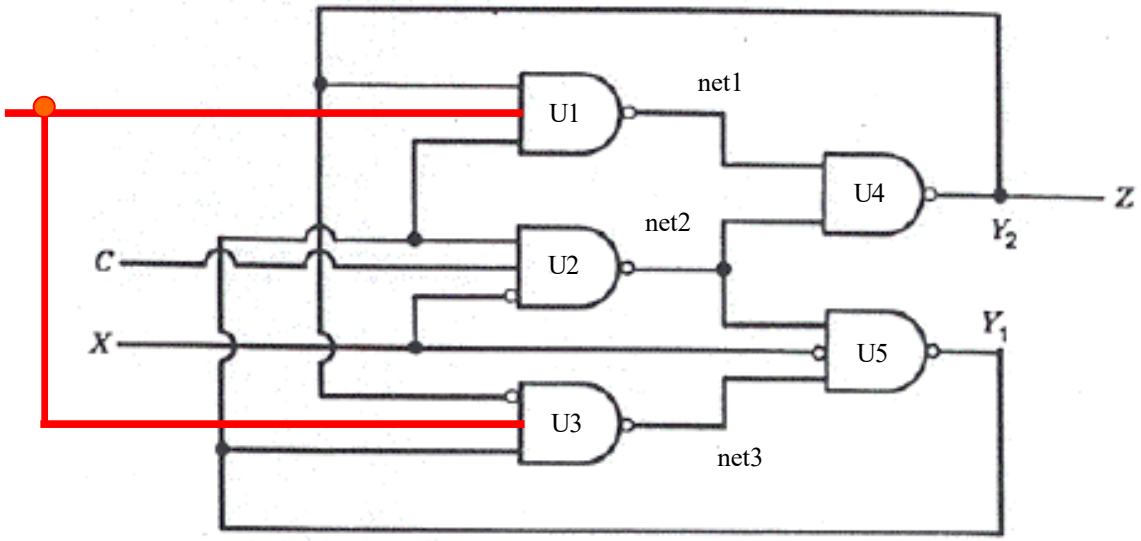

reset
U1
net1
U4
Z
net2
C
U2
Y_2
X
U5
Y_1
U3
net3


Name
Value
z
c
x
reset
x_seq[0:6]
c_seq[0:6]
i[31:0]
TEST_LEN[31:0]
reset
z
xbar
zbar
net1
y1
net2
net3
0 ns
200 ns
400 ns
600 ns
800 ns
800.000 ns
0011000
0110010

Simulation Results

- This circuit will be race-free provided the input pulses satisfy the fundamental mode constraints.

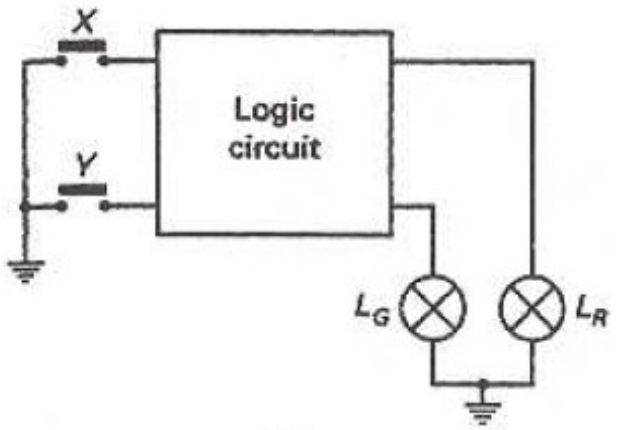




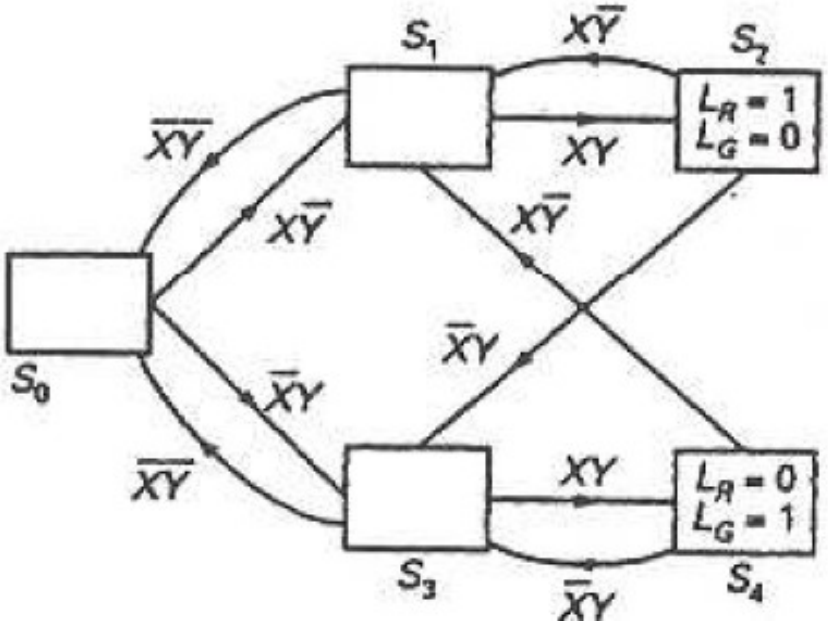
Design Procedure - Lamp Switching Example

Lamp Switching Example

- Design a switching circuit by which an operator can control two lights. If switch X is made followed by switch Y, then a red lamp (L_R) is to be turned on indicating that an incorrect switching procedure has been followed. If switch Y is made followed by switch X, then a green lamp (L_G) is to be turned on indicating that a correct procedure has been adopted.



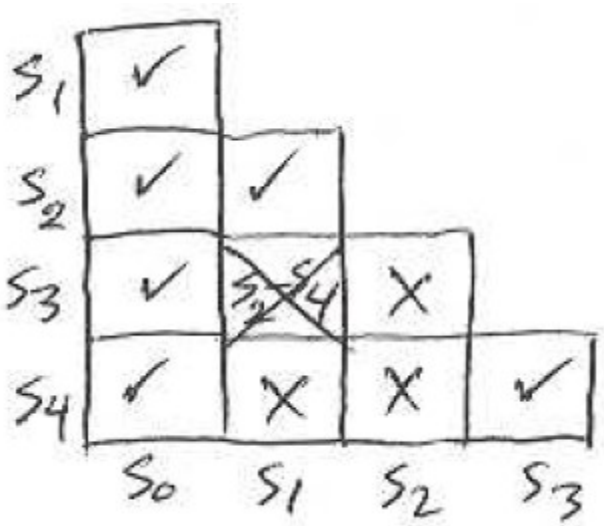
State Diagram and State Table



XY	00	01	11	10	
S_0	(S_0) $L_R=0$ $L_G=0$	S_3 $L_R=0$ $L_G=0$	–	S_1 $L_R=0$ $L_G=0$	3 states merge to form S_{012}
S_1	S_0 $L_R=0$ $L_G=0$	–	S_2 $L_R=-$ $L_G=0$	(S_1) $L_R=0$ $L_G=0$	
S_2	–	S_3 $L_R=-$ $L_G=0$	(S_2) $L_R=1$ $L_G=0$	S_1 $L_R=-$ $L_G=0$	
S_3	S_0 $L_R=0$ $L_G=0$	(S_3) $L_R=0$ $L_G=0$	S_4 $L_R=0$ $L_G=-$	–	2 states merge to form S_{34}
S_4	–	S_3 $L_R=0$ $L_G=-$	(S_4) $L_R=0$ $L_G=1$	S_1 $L_R=0$ $L_G=-$	

(c)

State Minimization



S0=S1

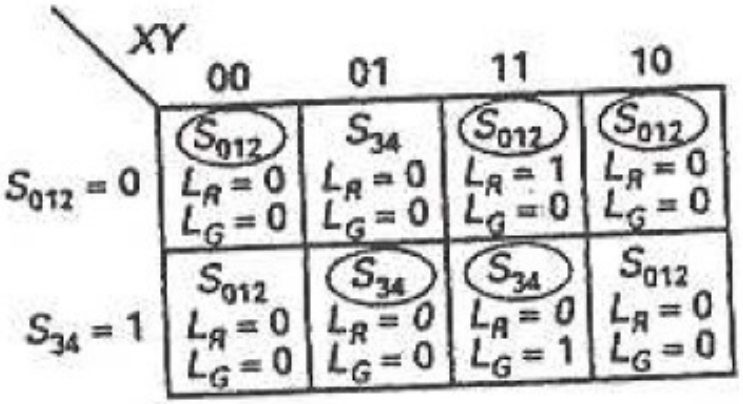
S0=S2

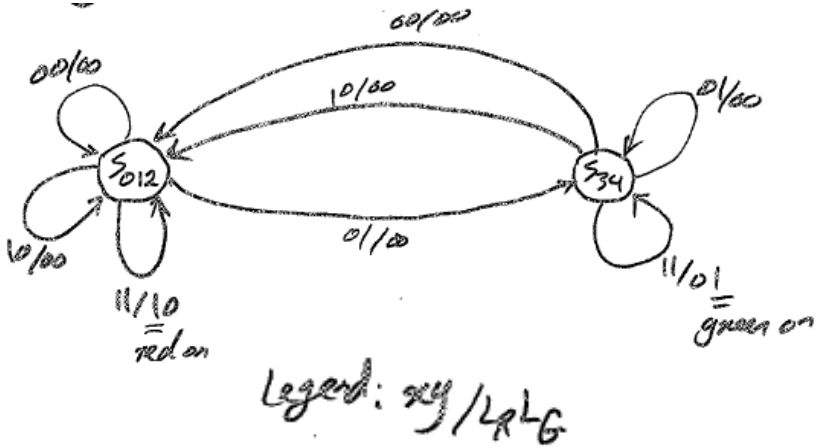
S0=S3

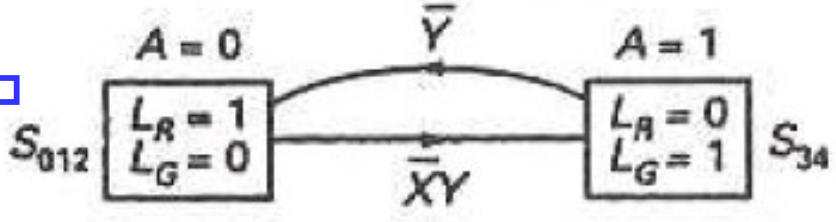
S0=S4

S1=S2

S3=S4



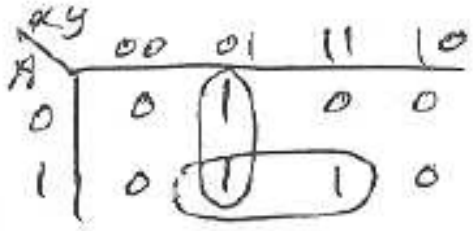




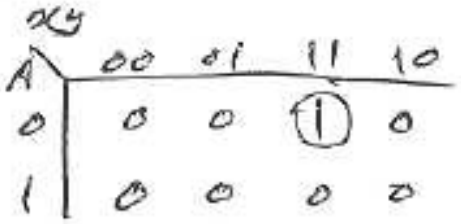
Compact Representation

Implementation

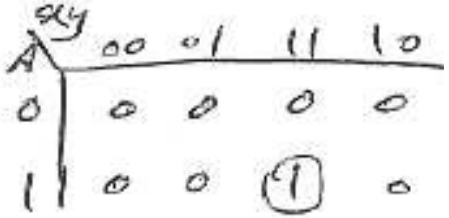
- K-Maps



$A^+ = \bar{x}y + A\bar{y}$

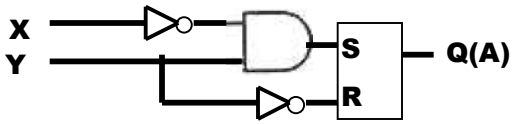


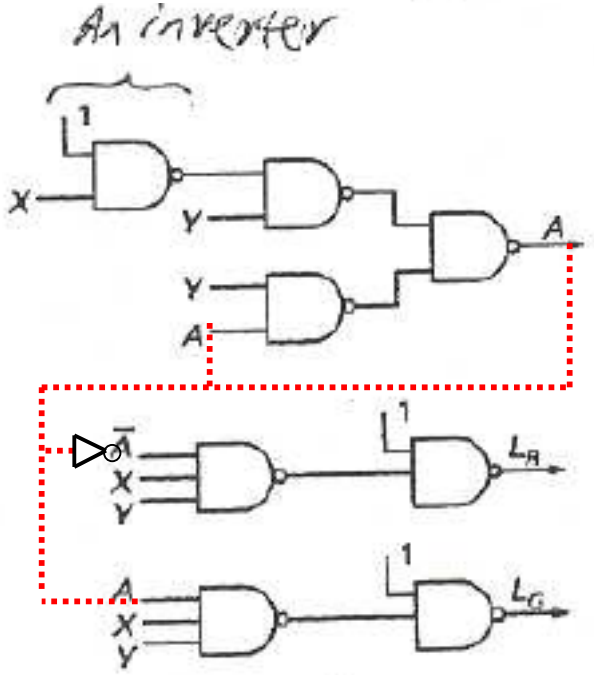
$L_R = xy\bar{A}$



$L_G = xyA$

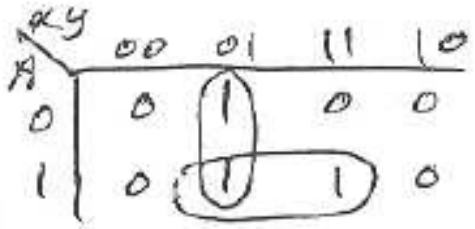
- SR latch can be used to provide feedbacks
  - SR latch: $Q^* = S + R' Q$
  - Here: $A^* = X' Y + Y A$
  - So, S=X' Y and R=Y'





Implementation (SR-Latch Mapping)

- More formal mapping of Next state function to SR latch can be done similar to synchronous design approach



$$A^{+} = \bar{x}y + A\bar{y}$$

Q	$Q^{t+\delta t}$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

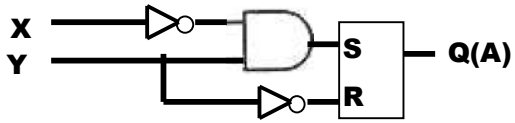
SR-latch transition table

A \ xy	00	01	11	10
0	0	1	0	0
1	0	x	x	0

S=X'Y

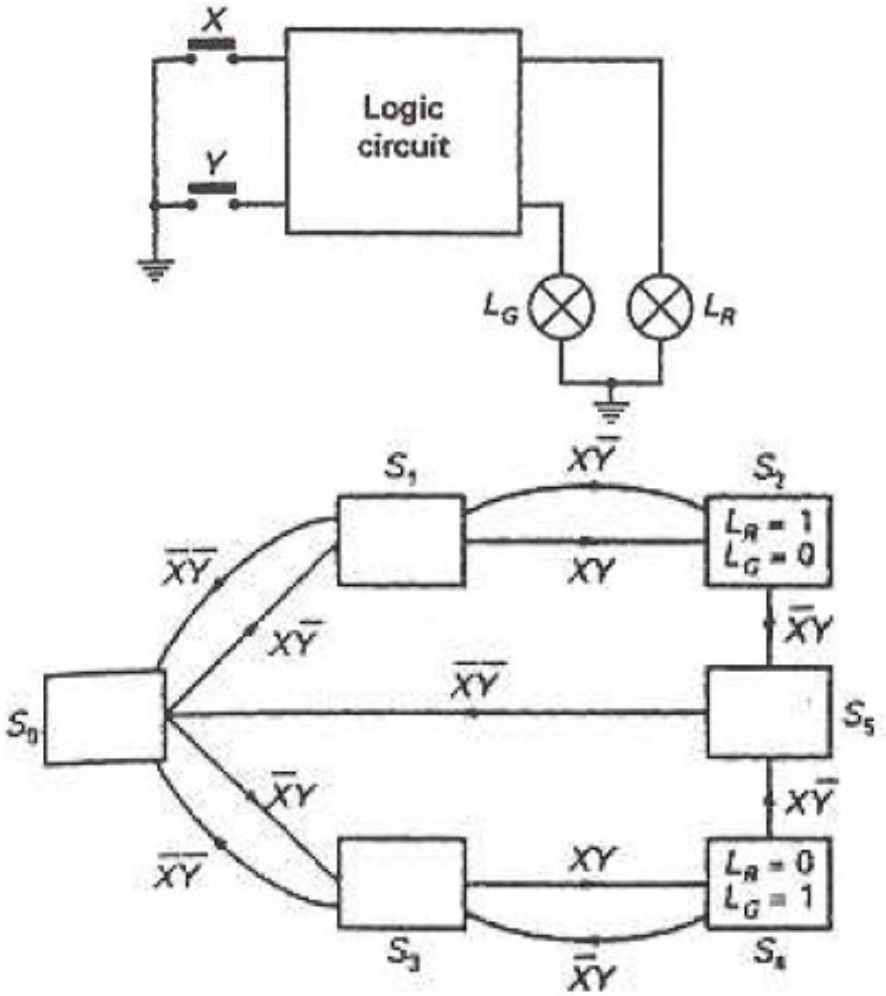
A \ xy	00	01	11	10
0	x	0	x	x
1	1	0	0	1

R=Y'



Modified Lamp Switching Example

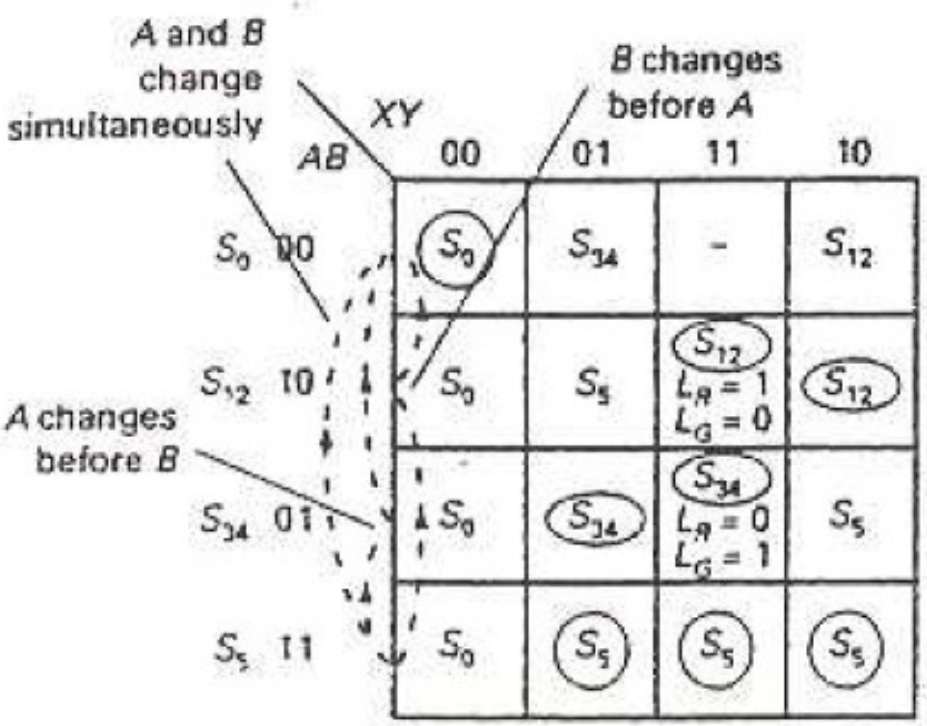
- Same as before only add an extra state to go back to quiescent state S0 after a light is turned on.

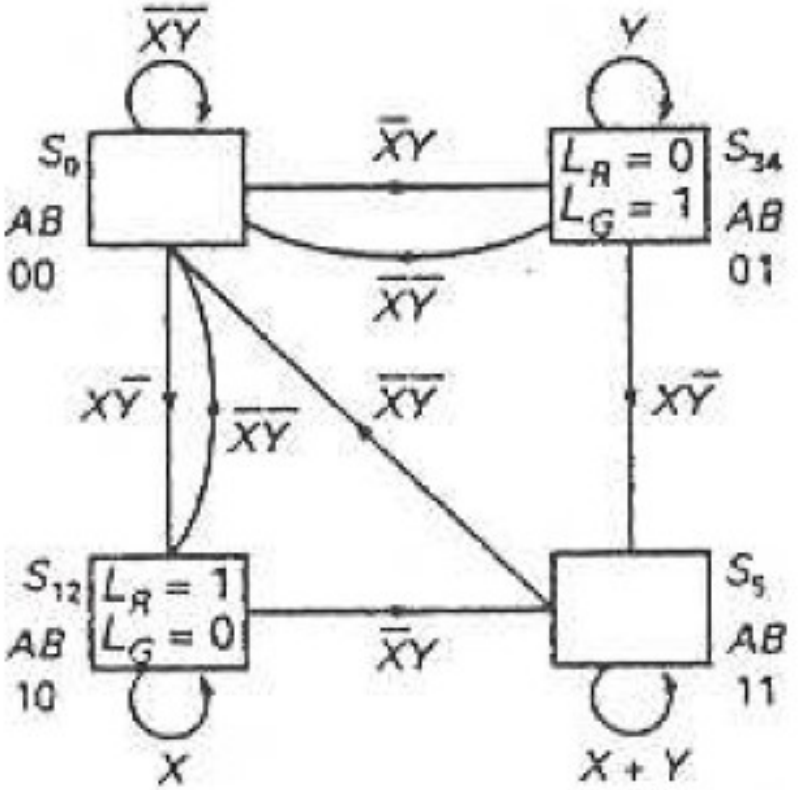


XY	00	01	11	10	
S_0	S_0 $L_R=0$ $L_G=0$	S_3 $L_R=0$ $L_G=0$	–	S_1 $L_R=0$ $L_G=0$	
S_1	S_0 $L_R=0$ $L_G=0$	–	S_2 $L_R=-$ $L_G=0$	S_1 $L_R=0$ $L_G=0$	2 states merge to form S_{12}
S_2	–	S_5 $L_R=-$ $L_G=0$	S_2 $L_R=1$ $L_G=0$	S_1 $L_R=-$ $L_G=0$	
S_3	S_0 $L_R=0$ $L_G=0$	S_3 $L_R=0$ $L_G=0$	S_4 $L_R=0$ $L_G=-$	–	2 states merge to form S_{34}
S_4	–	S_3 $L_R=0$ $L_G=-$	S_4 $L_R=0$ $L_G=1$	S_5 $L_R=0$ $L_G=-$	
S_5	S_0 $L_R=0$ $L_G=0$	S_5 $L_R=0$ $L_G=0$	S_5 $L_R=0$ $L_G=0$	S_5 $L_R=0$ $L_G=0$	

State Minimization

- No critical race exists.





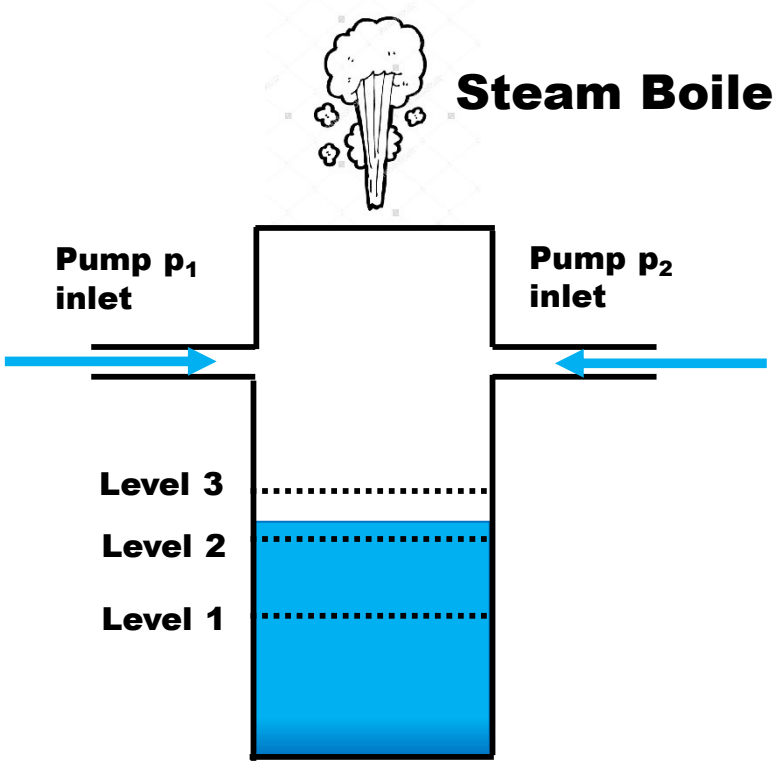
Design Procedure - Pump Control Example

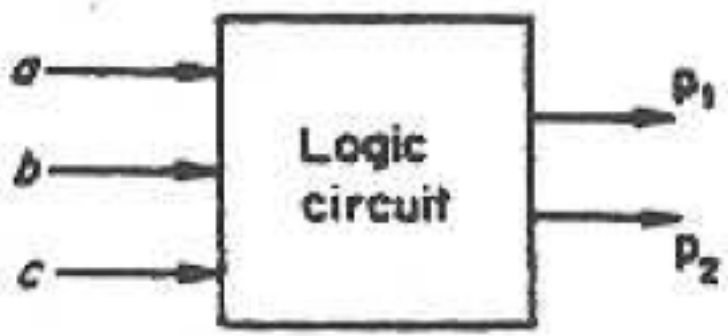
Pump Control Example - Problem Definition

- Water is pumped into a water tank by two pumps p_1 and p_2. Both pumps are to turn on when the water goes below level 1 and they are to remain on until water reaches level 2, when pump p_1 turns off and remains off until water is below level 1 again. Pump p_2 remains on until level 3 is reached when it also turns off and remains off until water falls below level 1 again.
- Sensors
  - a=1 when water is at or above level 1, otherwise a=0
  - b=1 when water is at or above level 2, otherwise b=0
  - c=1 when water is at or above level 3, otherwise c=0

Pump Control Example - Schematic

- System schematic and input/output





Pump Control – Inputs

In this problem, there are 3 sensors:

a, b, c

But there really is another input and that is whether furnace is on

f

Pump Control – Flow Table

	A	B
fabc	0000	1000
P_1	0	1
P_2	0	1

State		f	a	b	c
	0000	1000	1100	1110	1111
A	A,00	B,11			
B		B,11			

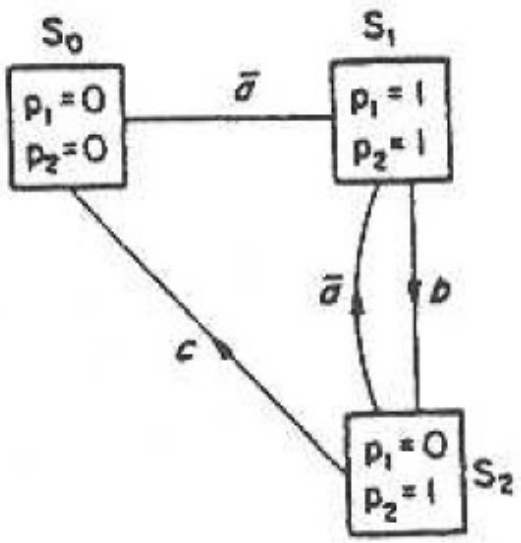
Pump Control – Flow Table

	B	C	
fabc	1000	1100	???
P_1	1	0	
P_2	1	1	

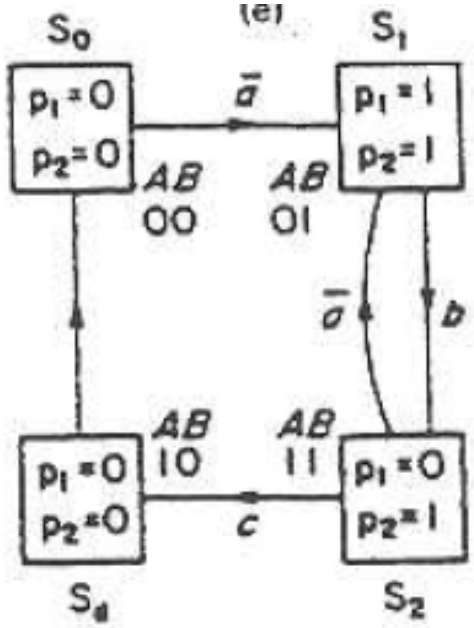
State		f	a	b	c
	0000	1000	1100	1110	1111
A	A,00	B,11			
B	A,00	B,11	C,01		
C			C,01		

State Diagram

- Original vs. Modified state diagram to get race-free assignment

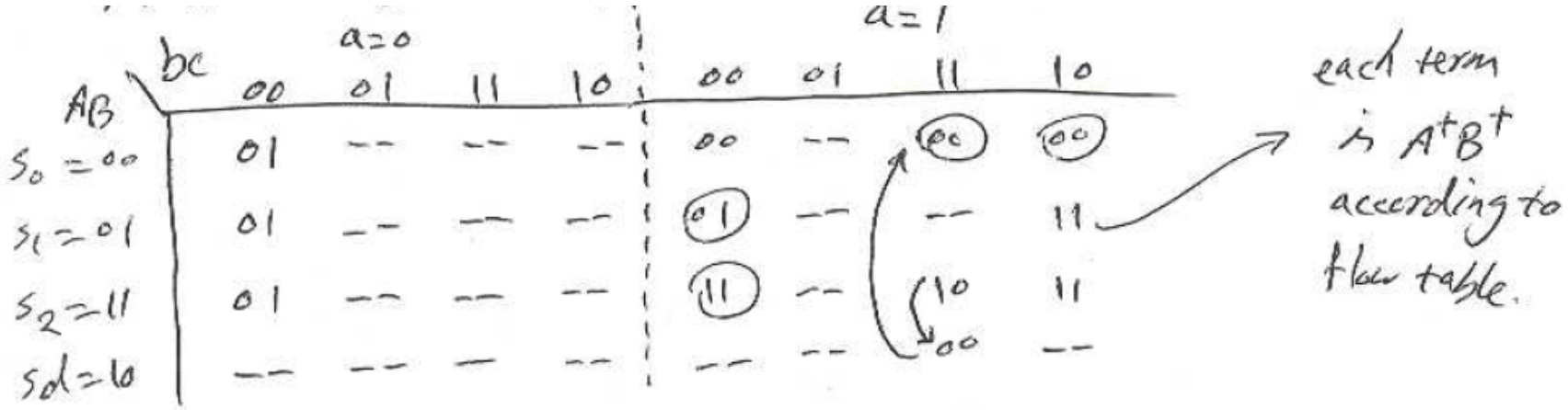


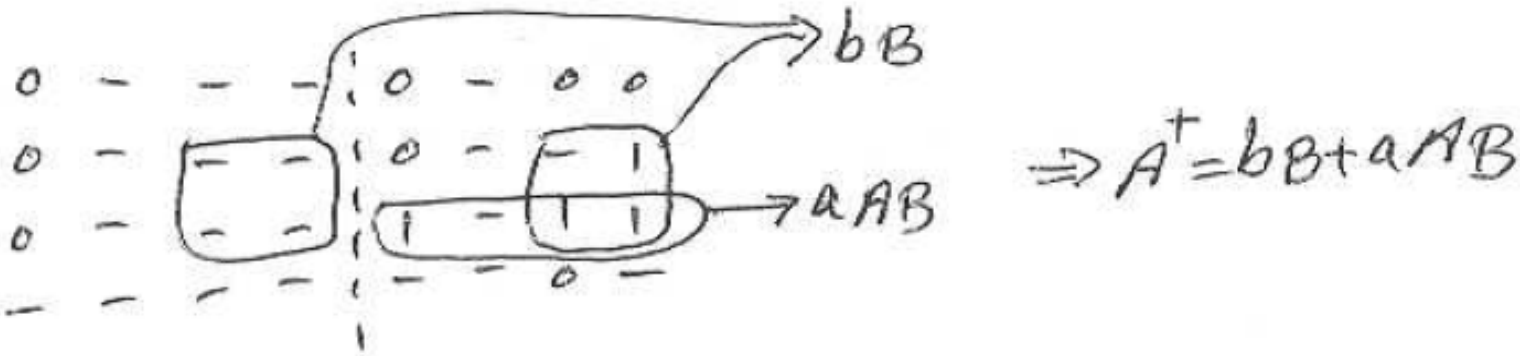
abc	000	100	110	111
S_0	S_1 $p_1=1$ $p_2=1$	(S0) $p_1=0$ $p_2=0$	(S0) $p_1=0$ $p_2=0$	(S0) $p_1=0$ $p_2=0$
S_1	(S1) $p_1=1$ $p_2=1$	(S1) $p_1=1$ $p_2=1$	S_2 $p_1=0$ $p_2=1$	
S_2	S_1 $p_1=1$ $p_2=1$	(S2) $p_1=0$ $p_2=1$	(S2) $p_1=0$ $p_2=1$	(S0) $p_1=0$ $p_2=0$



Implementation

- 5-variable K-Map needed. For example for A^* :

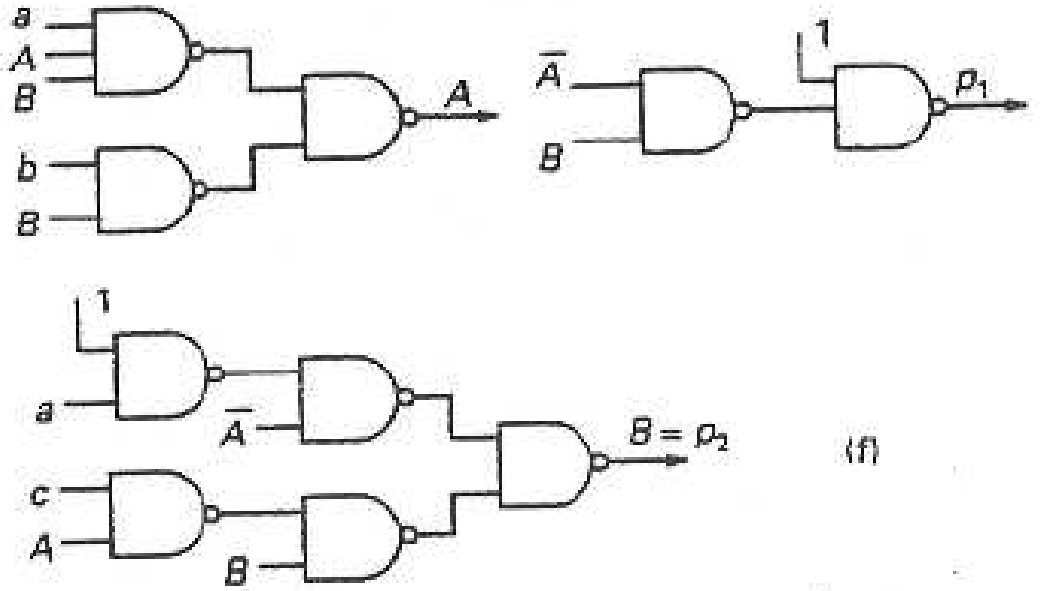




Implementation (cont.)

- One typical implementation (direct or using SR latch)

$$
\begin{aligned}
A^{t+\delta t} &= [bB+\overline{(\bar{B}+\bar{a})}A]^t \\
&= [bB+aAB]^t \\
B^{t+\delta t} &= [\bar{a}\bar{A}+(\overline{cA})B]^t \\
&= [\bar{a}\bar{A}+(\bar{c}+\bar{A})B]^t \\
p_1 &= \bar{A}B \\
p_2 &= \bar{A}B+AB=B
\end{aligned}
$$

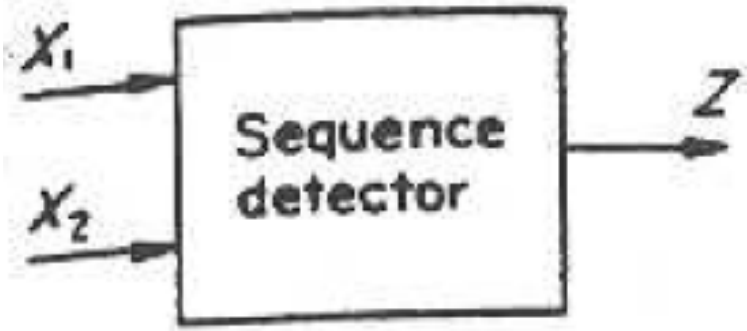


Design Procedure - Sequence Detector Example

Sequence Detector - Problem Definition

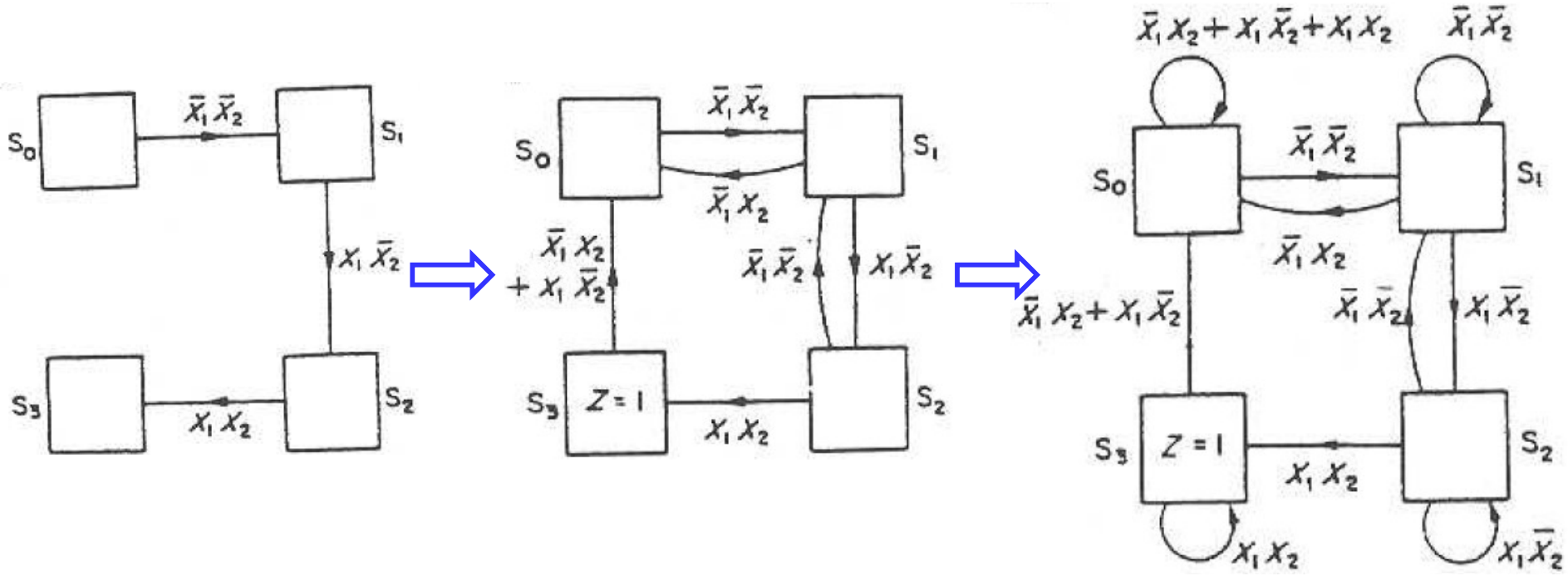
- Design a sequence detector which has two inputs X_1 and X_2, and one output Z. The circuit is required to give an output Z=1 when the sequence of input signals X_1X_2=00, 10, 11 has occurred.

- System schematic



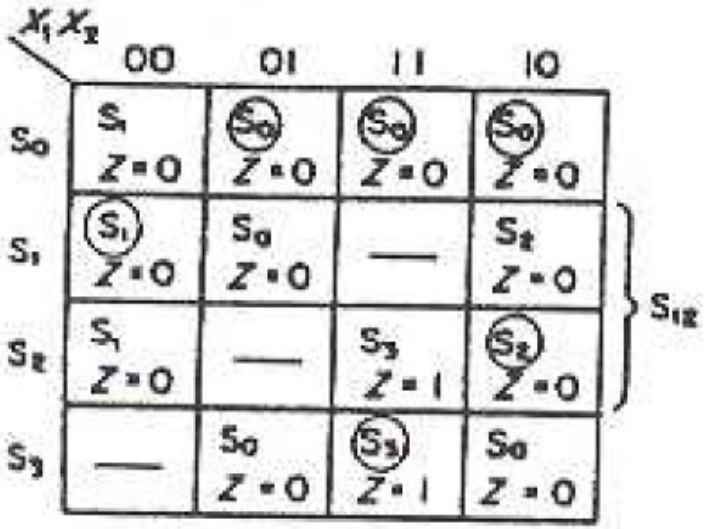
State Diagram

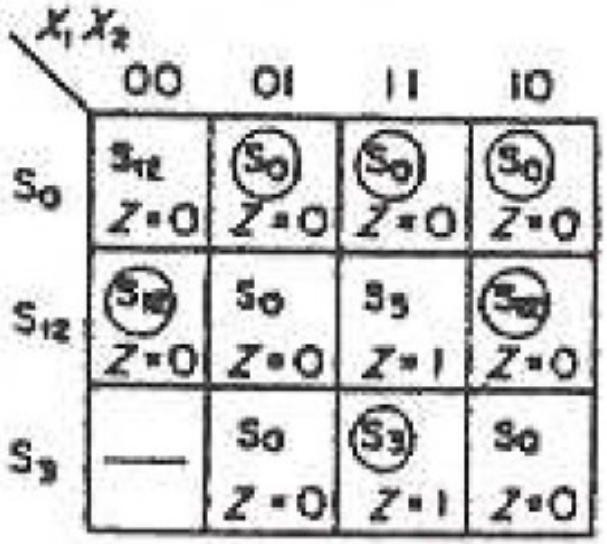
- A typical FSM construction

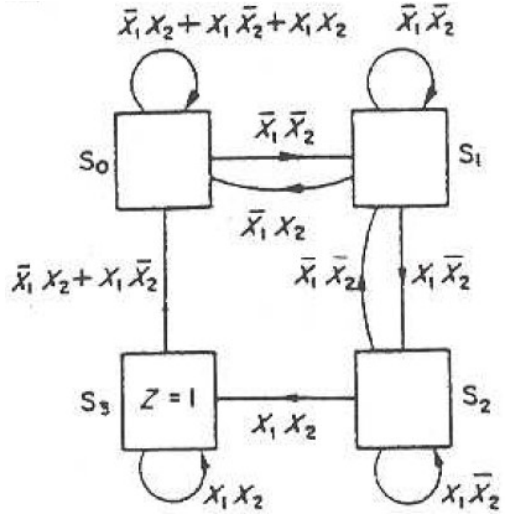


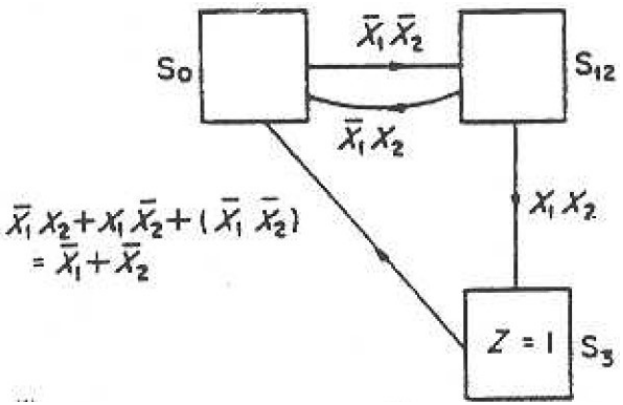
Flow Table

- Before and after state minimization



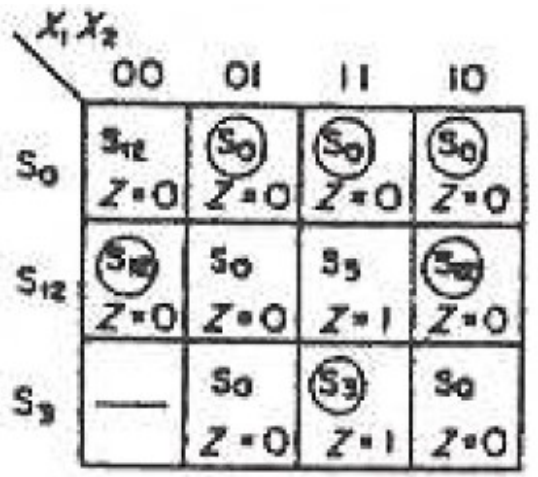


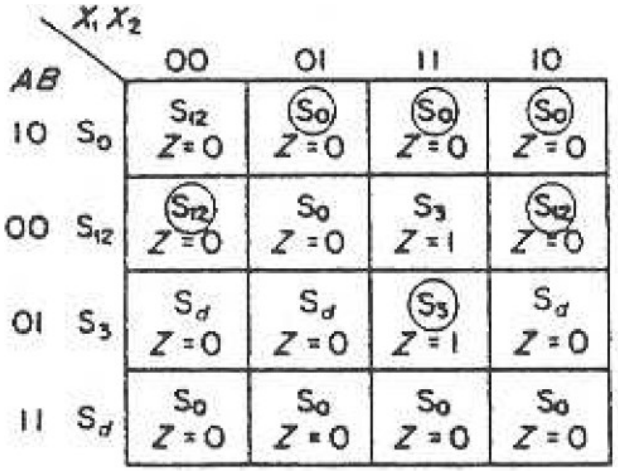




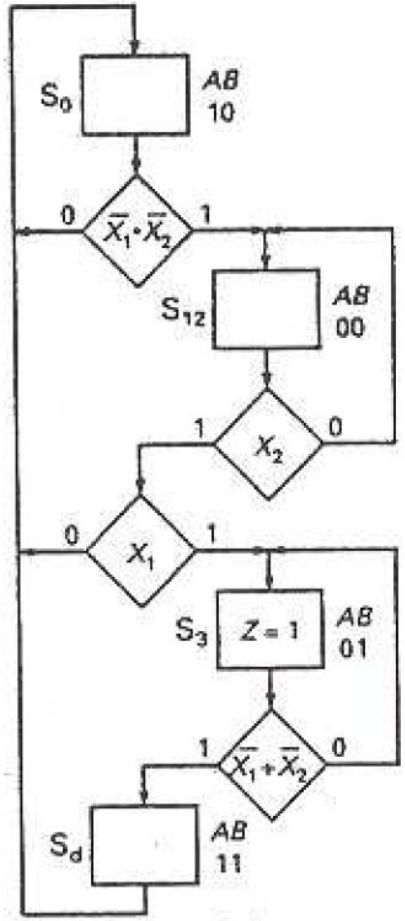
Race-Free State Code Assignment

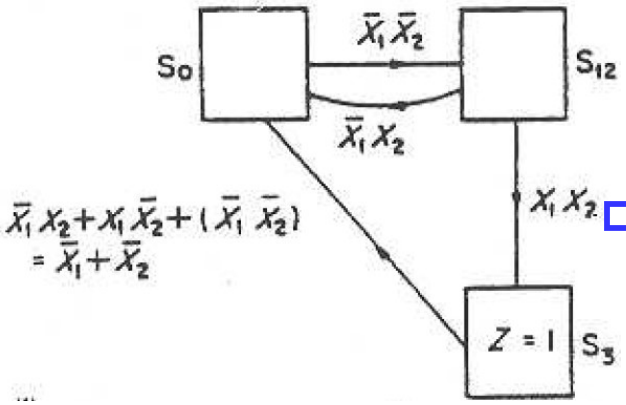
- Adding dummy state to avoid race

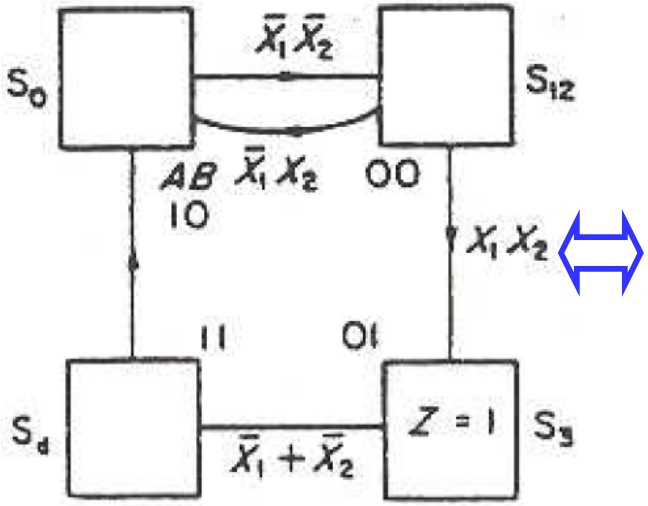




Arithmetic State Machine (ASM)







K-Maps

- Choose a state assignment

AB \ X_1X_2	00	01	11	10
10 S_0	S_{12} Z=0	(S0) Z=0	(S0) Z=0	(S0) Z=0
00 S_{12}	(S12) Z=0	S0 Z=0	S3 Z=1	(S12) Z=0
01 S_3	Sd Z=0	Sd Z=0	(S3) Z=1	Sd Z=0
11 S_d	S0 Z=0	S0 Z=0	S0 Z=0	S0 Z=0

A^tB^t \ $X_1^tX_2^t$	00	01	11	10
00	00	10	01	00
01	11	11	01	11
11	10	10	10	10
10	00	10	10	10

$A^{t+\delta t}$ $B^{t+\delta t}$

A^tB^t \ $X_1^tX_2^t$	00	01	11	10
00		1		
01	1	1		1
11	1	1	1	1
10		1	1	1

$A^{t+\delta t}$ =($X_1'X_2$+ BX'_2+AX_1)

A^tB^t \ $X_1^tX_2^t$	00	01	11	10
00			1	
01	1	1	1	1
11				
10				

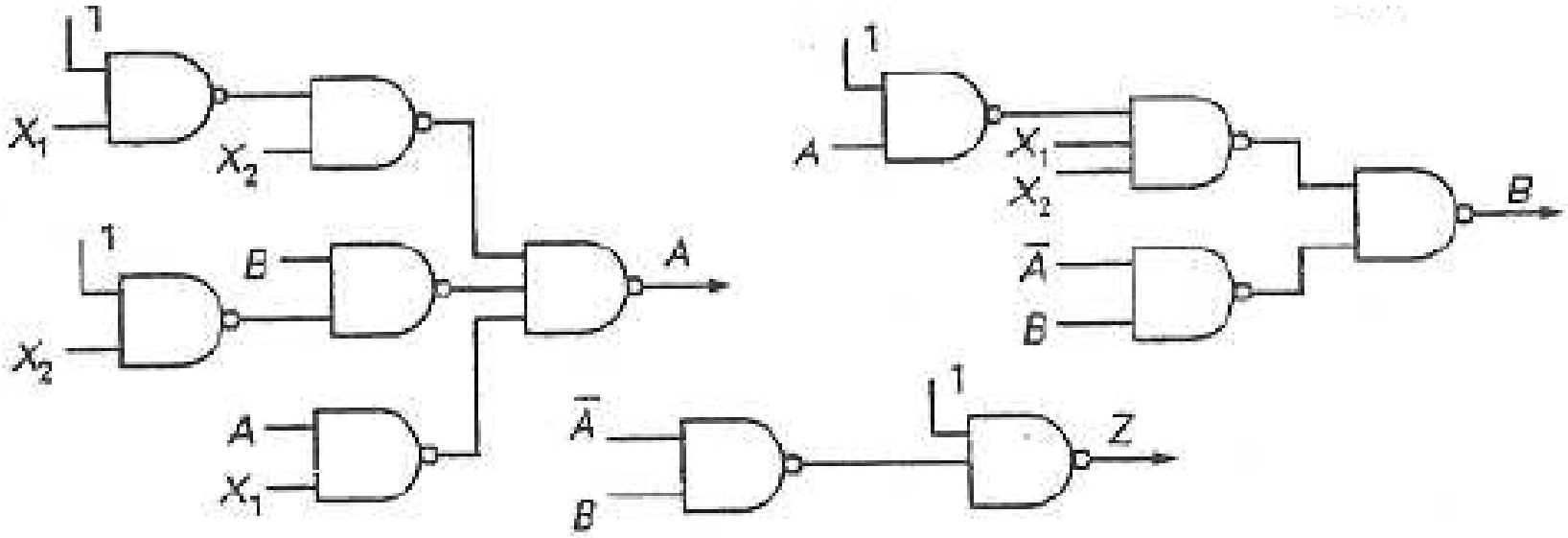
$B^{t+\delta t}$ =(A'X_1X_2+A'B)

AB \ X_1X_2	00	01	11	10
00			1	
01			1	
11				
10				

Z=A'B

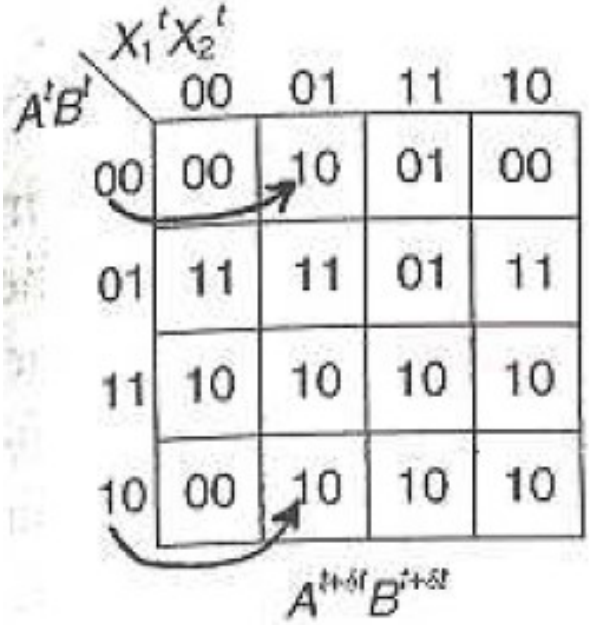
Implementation – Direct Feedbacks

- Direct feedbacks
  - $A^+ = (X_1' X_2 + B X_2' + A X_1)$
  - $B^+ = A' X_1 X_2 + A' B$
  - $Z = A' B$

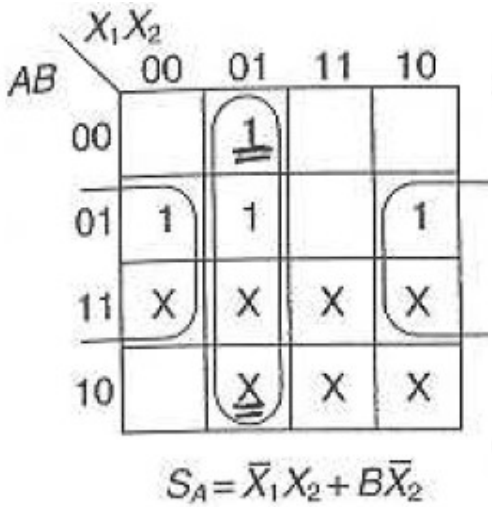


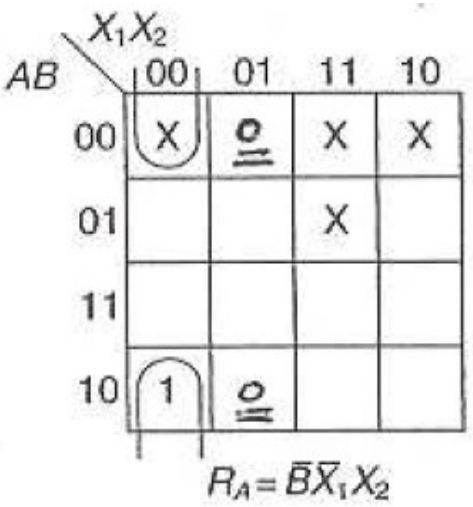
Implementation – Using SR Latch

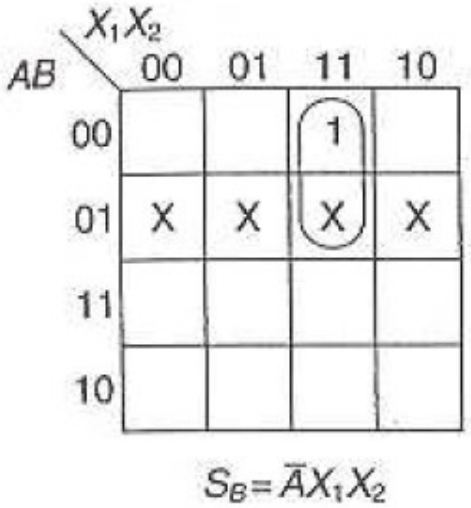
- Do re-mapping using SR-latch transition table

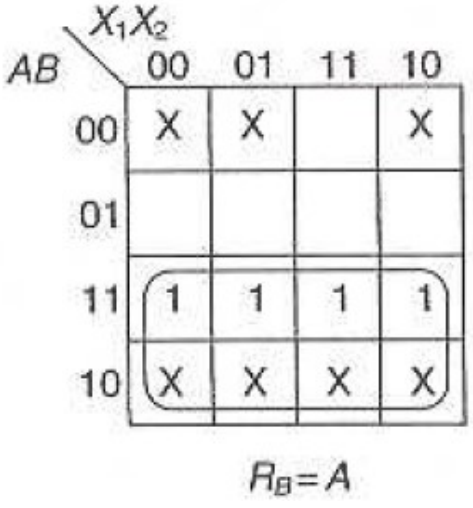


Q^t	$Q^{t+\delta t}$	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0









Implementation – Using SR Latch (cont.)

- Final circuit