Digital System Design

Hajar Falahati

hfalahati@ipm.ir
hfalahati@ce.sharif.edu

Outline

- Logic Design Recall
- Combinational Logic
- Sequential Logic

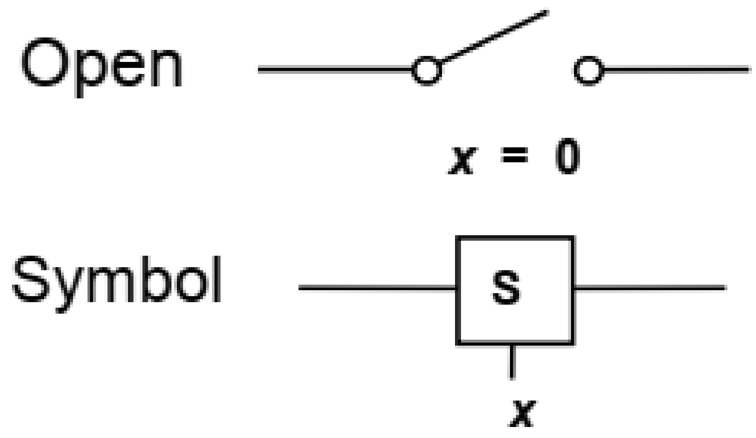
Logic Design Recall

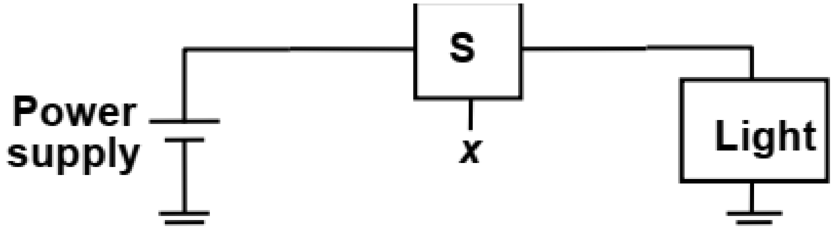
Binary Logic

- Binary variables
  - 1/0
- Logical operators
  - AND : (a.b)
  - OR : (a+b)
  - NOT : ($\bar{a}$), (a‘), (~a)
- Logical gates
  - Implement logic functions
- Boolean Algebra
  - Specify and transform logic functions

Switches

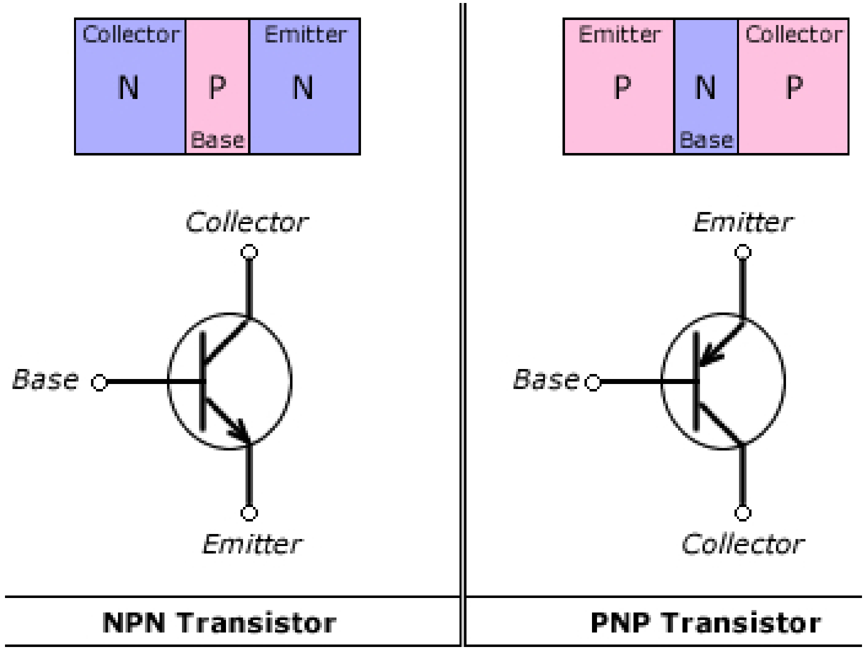
- A switch has two states
  - Closed / On
  - Open / OFF

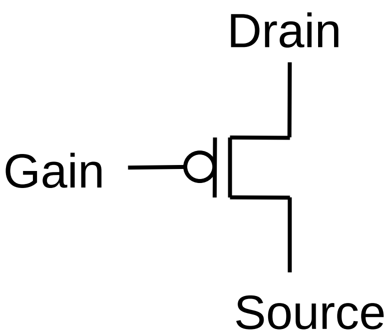




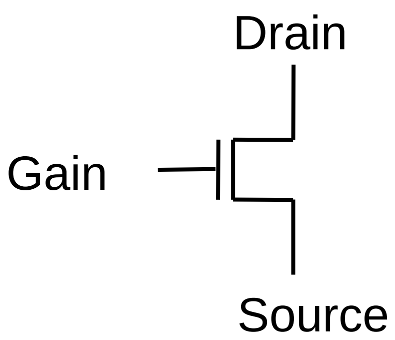
Transistor

- BJT
- Mosfet



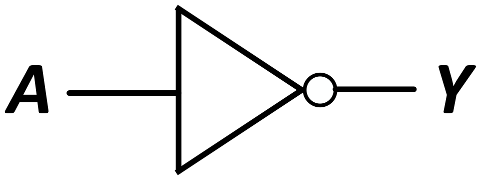


PMOS Transistor

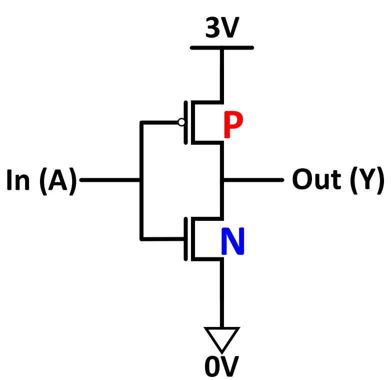


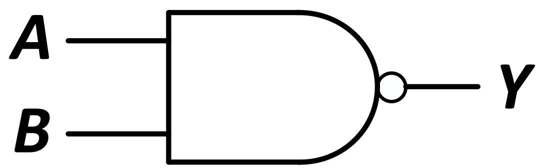
NMOS Transistor

Logic Gates

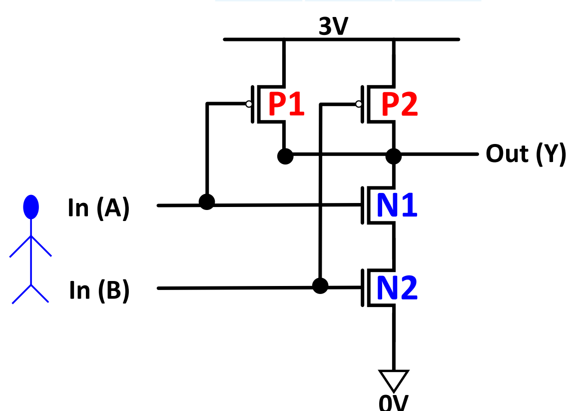


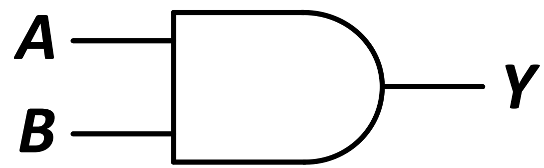
A	Y
0	1
1	0



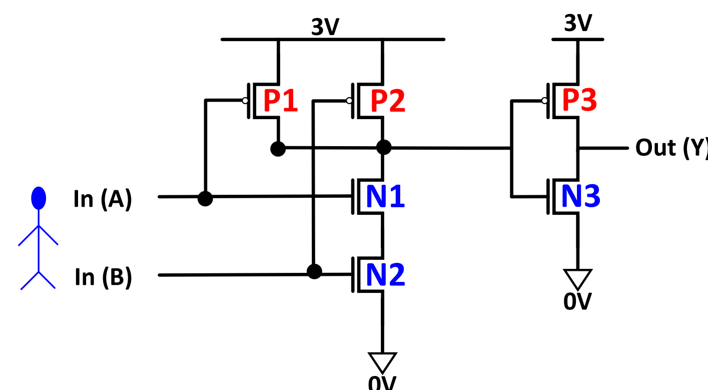


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

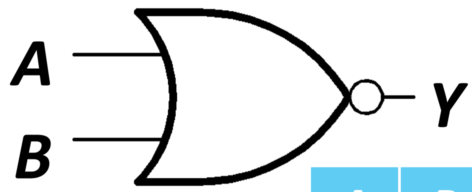




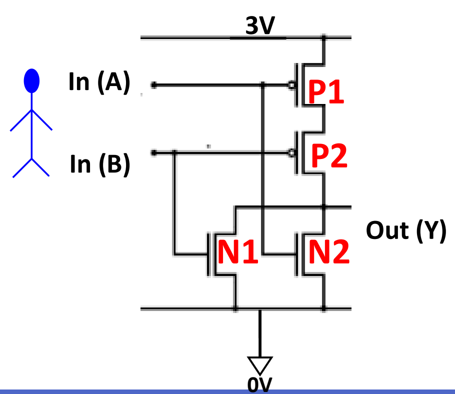
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

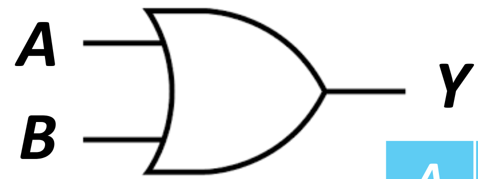


Logic Gates (cont'd)

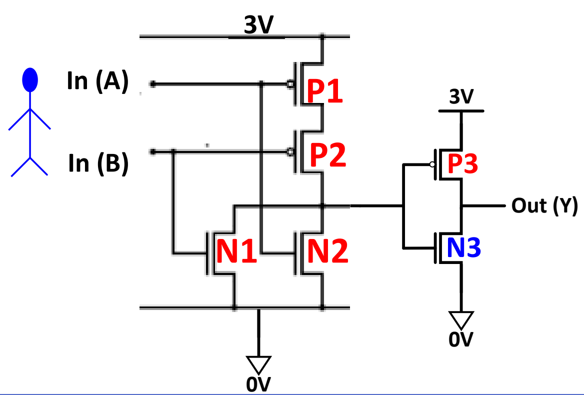


A	*B*	*Y*
0	0	1
0	1	0
1	0	0
1	1	0





A	*B*	*Y*
0	0	0
0	1	1
1	0	1
1	1	1



Some Definition

- **Literal**
  - A variable, complemented or uncomplemented
  - $\boldsymbol{A}$, $\boldsymbol{\overline{A}}$
- **Product term**
  - A literal or literals ANDed together
  - $(\boldsymbol{A \cdot B \cdot \overline{C}})$, $(\boldsymbol{\overline{A} \cdot C})$, $(\boldsymbol{B \cdot \overline{C}})$
- **Sum term**
  - A literal or literals ORed together.
  - $(\boldsymbol{A + B + \overline{C}})$, $(\boldsymbol{\overline{A} + C})$, $(\boldsymbol{B + \overline{C}})$
- **Minterm**
  - A product that includes all the variables
  - $(\boldsymbol{A \cdot B \cdot \overline{C}})$, $(\boldsymbol{\overline{A} \cdot \overline{B} \cdot C})$, $(\boldsymbol{\overline{A} \cdot B \cdot \overline{C}})$
- **Maxterm**
  - A sum that includes all the variables
  - $(\boldsymbol{A + B + \overline{C}})$, $(\boldsymbol{\overline{A} + \overline{B} + C})$, $(\boldsymbol{\overline{A} + B + \overline{C}})$

Canonical Form: SOP

- **Sum of products (SOP)**
  - ORed minterms

$$Y = \bar{A}B\bar{C} + \bar{A}BC + AB\bar{C} + ABC$$

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Canonical Form: POS

- **Product of sum (POS)**
  - ANDed maxterms

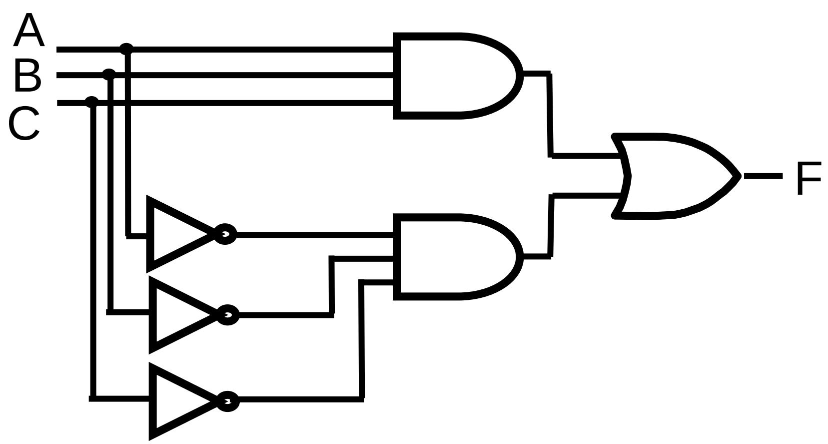
$$Y = (A + B + C).(A + B + \bar{C}).(\bar{A} + B + C).(\bar{A} + B + \bar{C})$$

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

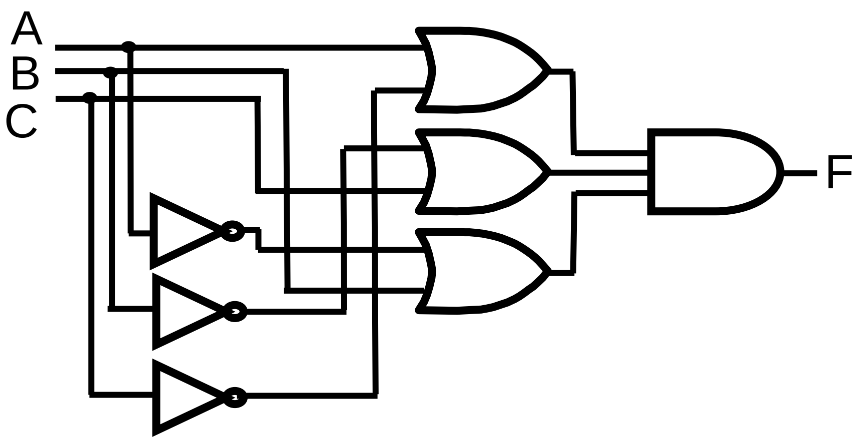
Two-level Logics

- SOP and POS lead to two-level logic

F = A B C + A’ B’ C’

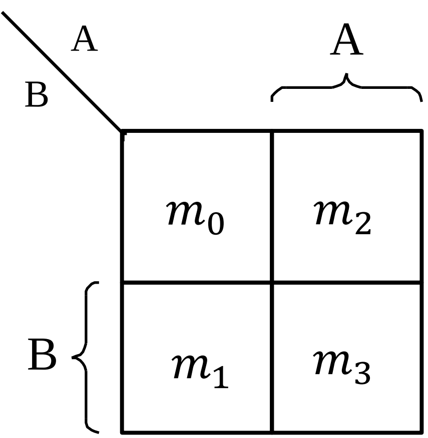


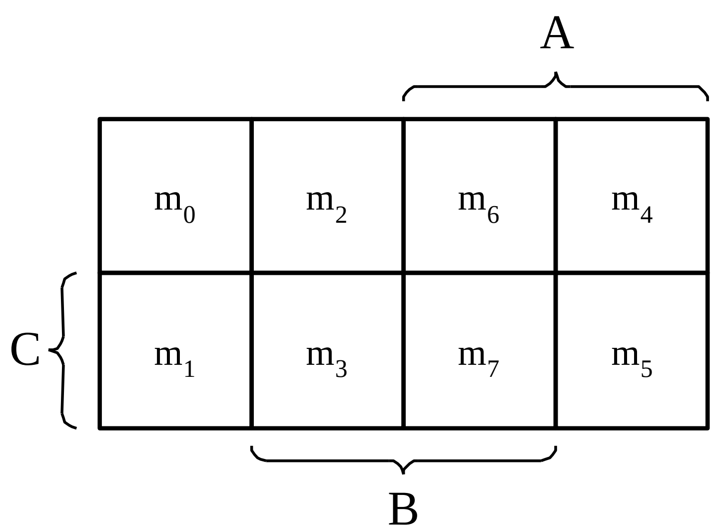
F = (A + C’)(B’ + C)(A’ + B)

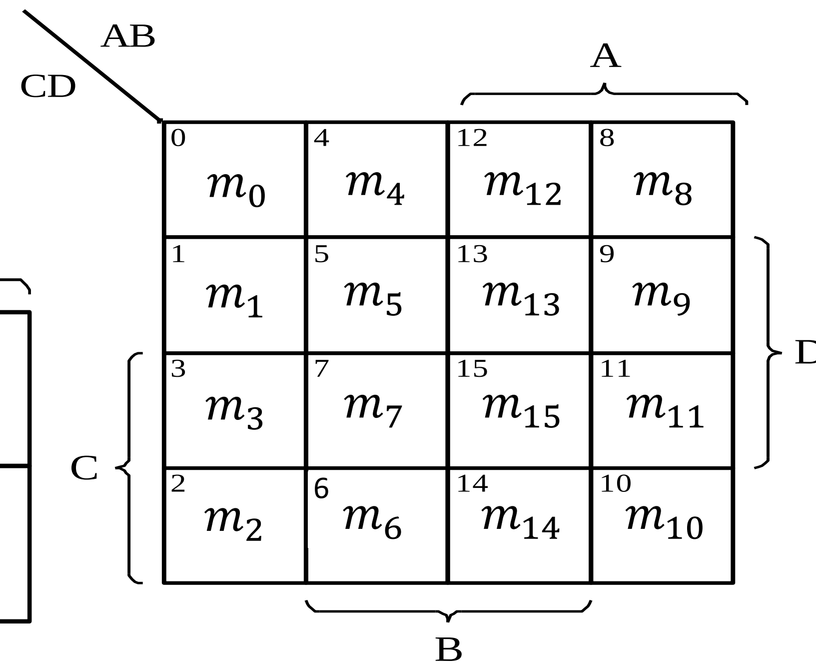


Karnaugh Map (K-map)

- An alternative method of representing the **truth table**
- Visualizes adjacencies in up to 6 dimensions
  - Physical adjacency ↔ Logical adjacency

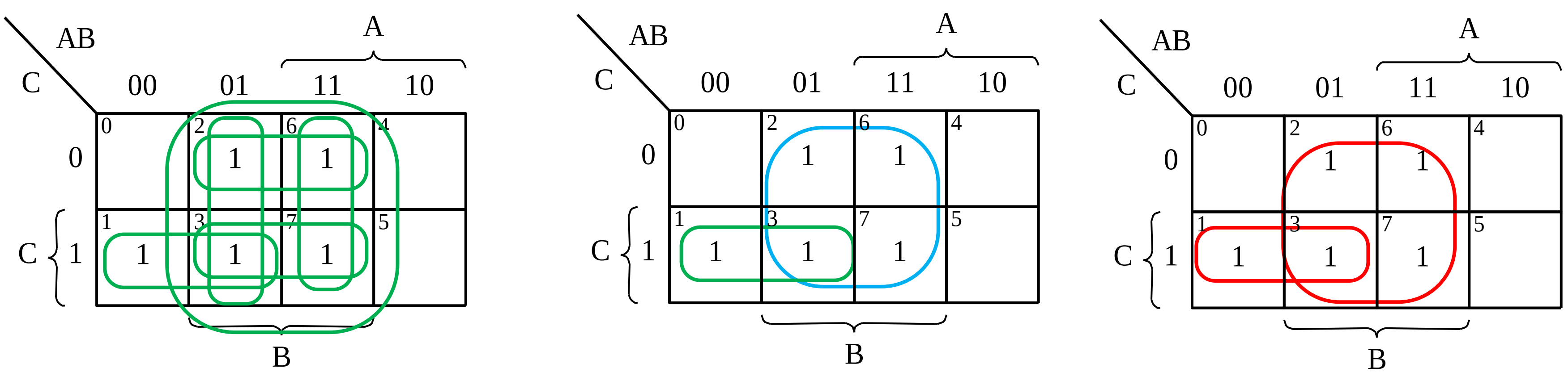






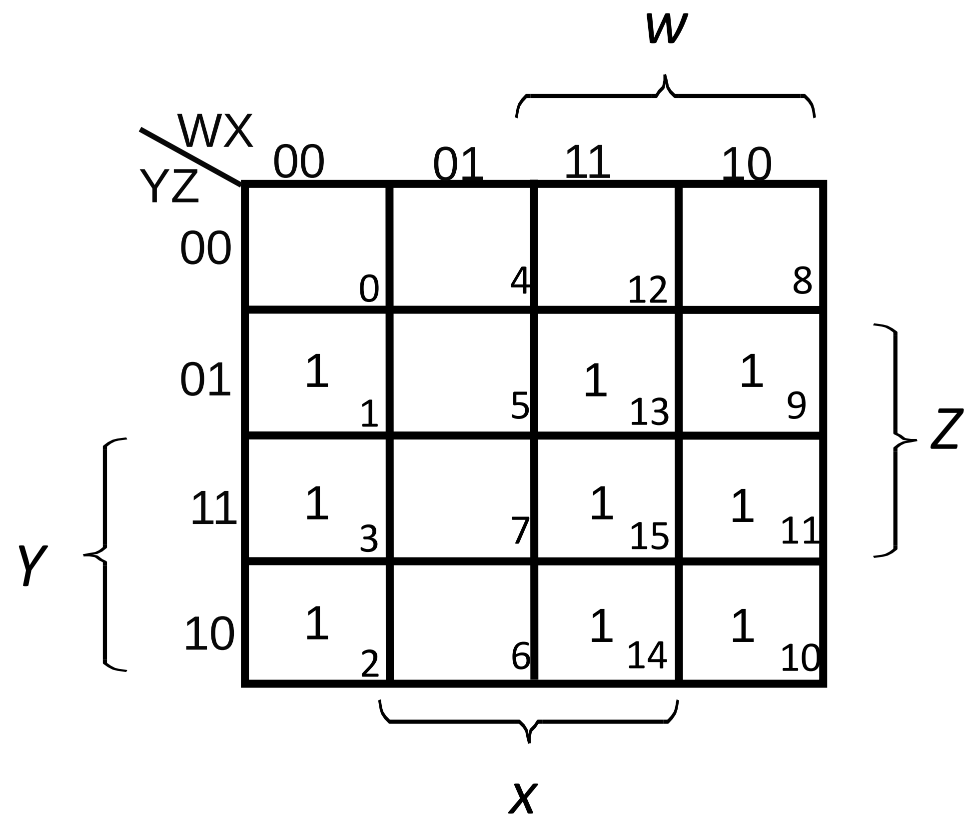
K-Map: Terms

- Implicant
  - A product term that can cover minterms of a function.
- Prime implicant
  - A product term that is not covered by another **implicant** of the function.
  - Combining the **maximum possible number of adjacent squares**
- Essential prime implicant
  - A **prime implicant** that covers **at least one minterm** that is not covered by any other **prime implicant**.



K-Map: 4-Variable Function

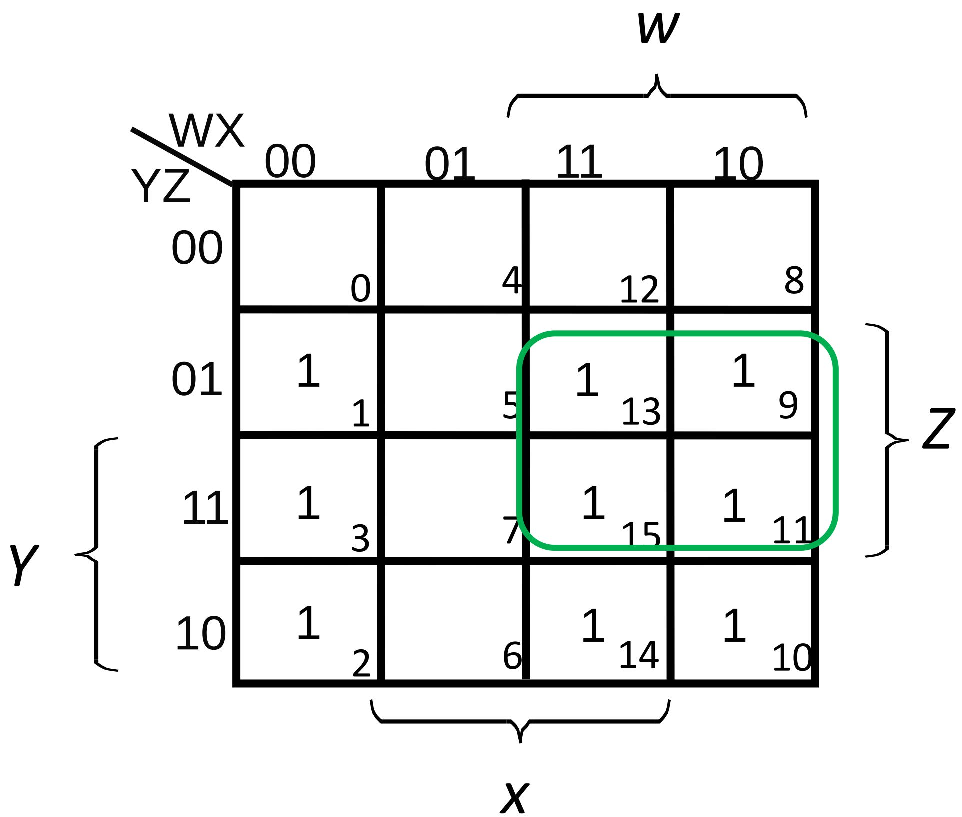
$F(W,X,Y,Z) = \sum m(1, 2, 3, 9, 10, 11, 13, 14, 15)$



K-Map: 4-Variable Function

$F(W,X,Y,Z) = \sum m(1, 2, 3, 9, 10, 11, 13, 14, 15)$

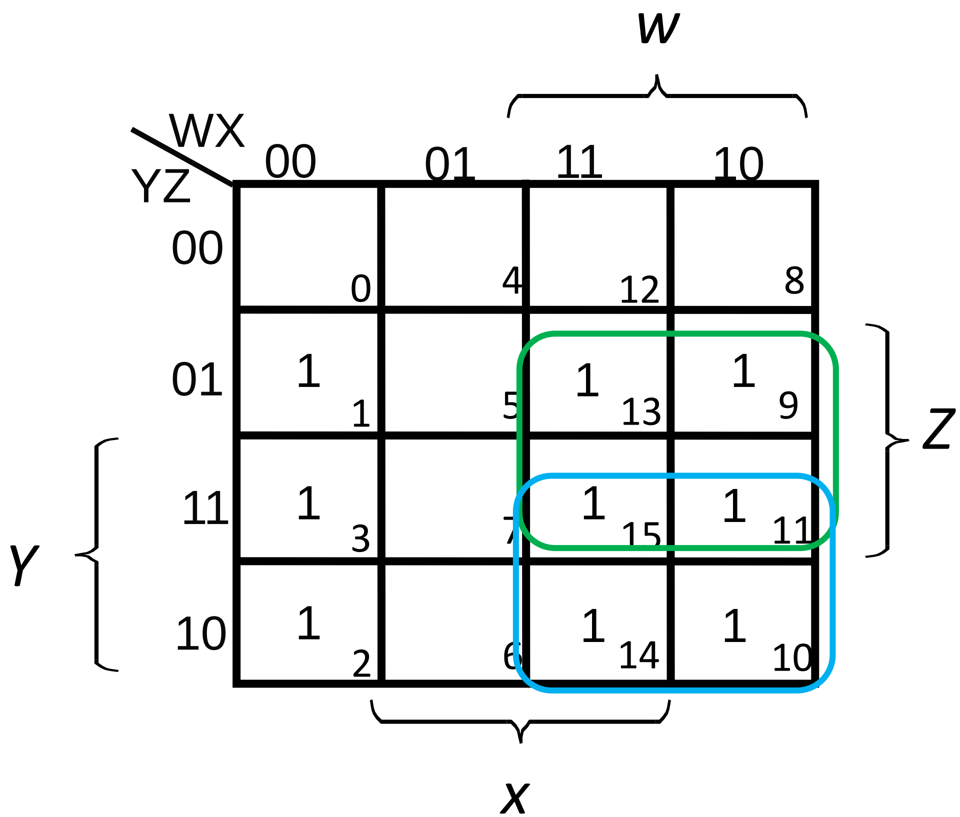
$F(A,B,C,D,E) = WZ$



K-Map: 4-Variable Function

$F(W,X,Y,Z) = \sum m(1, 2, 3, 9, 10, 11, 13, 14, 15)$

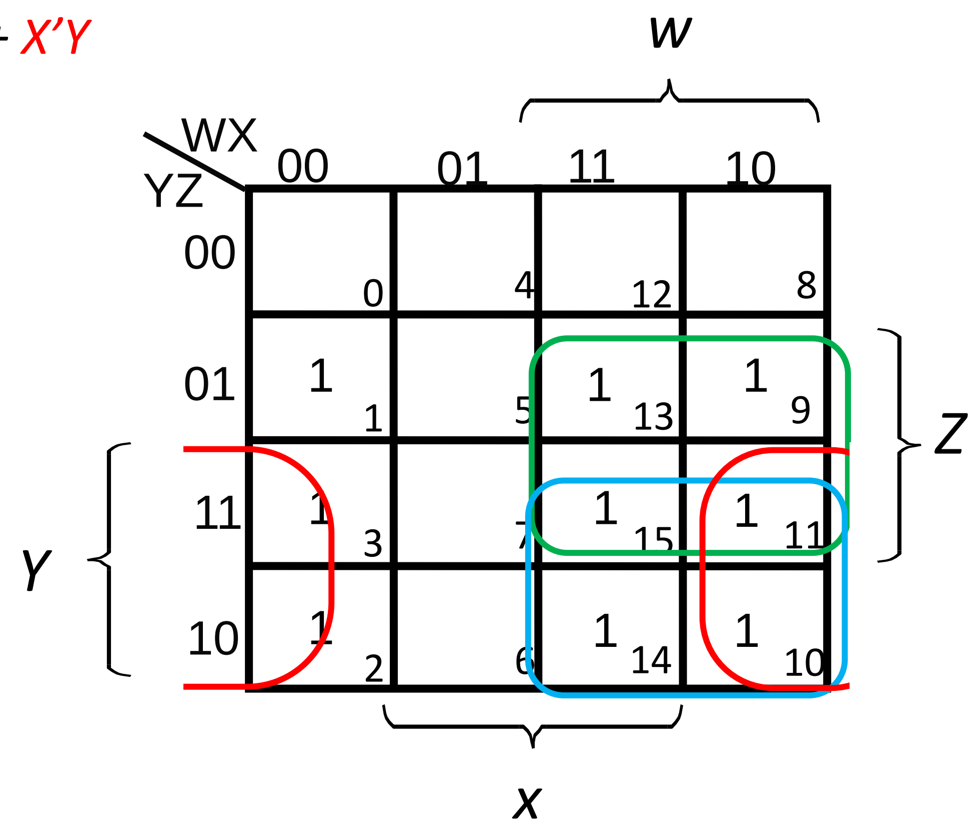
$F(A,B,C,D,E) = WZ + WY$



K-Map: 4-Variable Function

$F(W,X,Y,Z) = \sum m(1, 2, 3, 9, 10, 11, 13, 14, 15)$

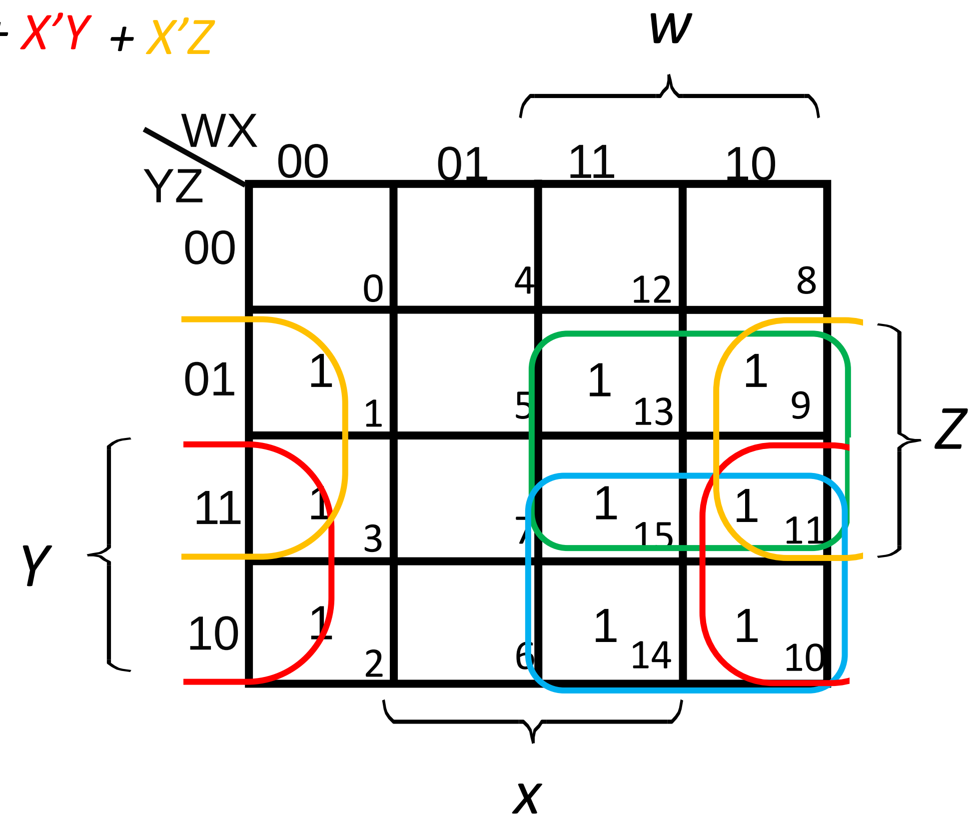
$F(A,B,C,D,E) = WZ + WY + X'Y$



K-Map: 4-Variable Function

$F(W,X,Y,Z) = \sum m(1, 2, 3, 9, 10, 11, 13, 14, 15)$

$F(A,B,C,D,E) = WZ + WY + X'Y + X'Z$



K-Map: 5-Variable Function

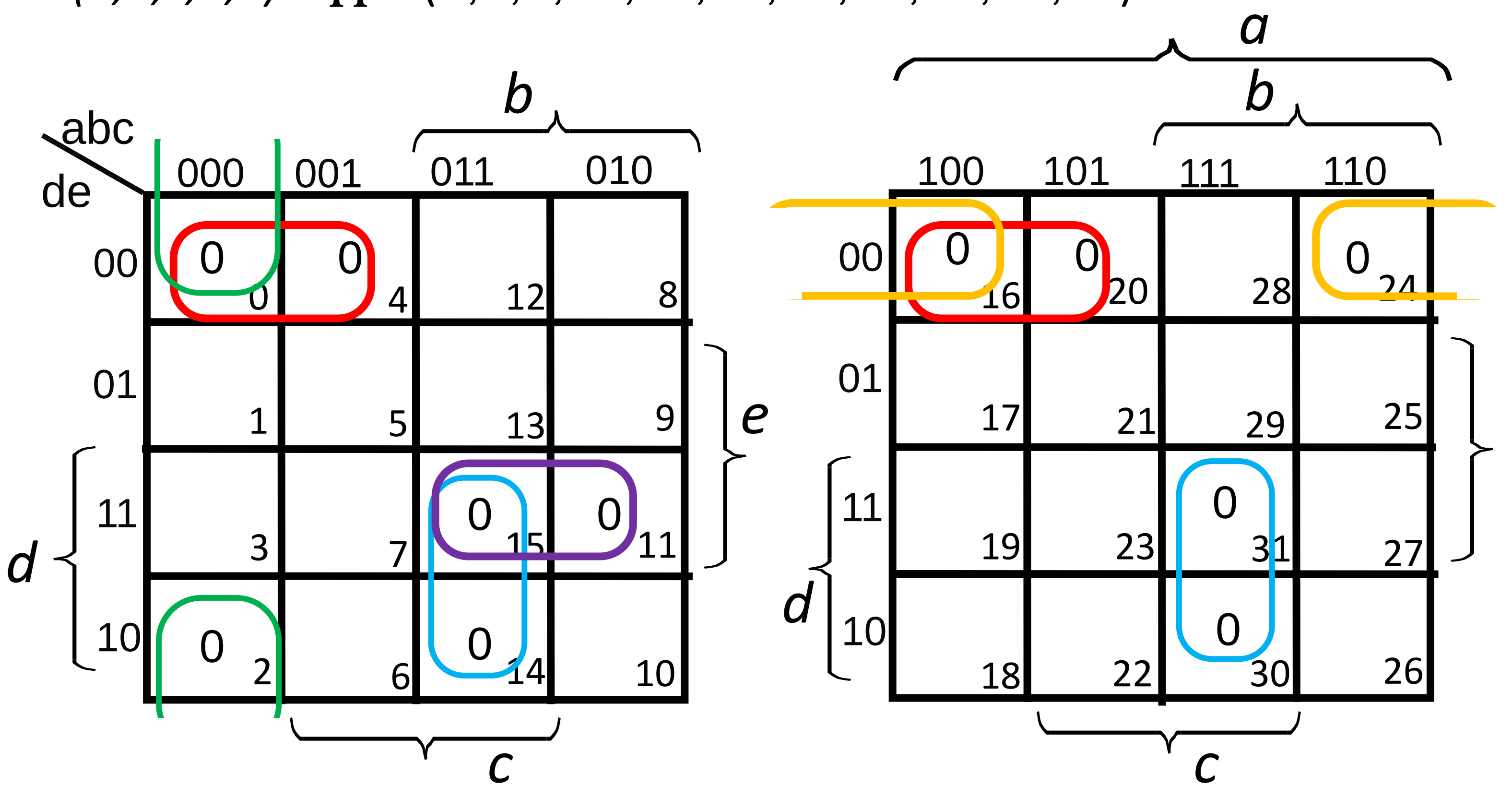
F(a,b,c,d,e) = $\prod M(0,2,4,11,14,15,16,20,24,30,31)$

de \ abc	000	001	011	010
00	0 $_{0}$	0 $_{4}$	$_{12}$	$_{8}$
01	$_{1}$	$_{5}$	$_{13}$	$_{9}$
11	$_{3}$	$_{7}$	0 $_{15}$	0 $_{11}$
10	0 $_{2}$	$_{6}$	0 $_{14}$	$_{10}$

de \ abc	100	101	111	110
00	0 $_{16}$	0 $_{20}$	$_{28}$	0 $_{24}$
01	$_{17}$	$_{21}$	$_{29}$	$_{25}$
11	$_{19}$	$_{23}$	0 $_{31}$	$_{27}$
10	$_{18}$	$_{22}$	0 $_{30}$	$_{26}$

K-Map: 5-Variable Function

$F(a,b,c,d,e) = \prod M(0,2,4,11,14,15,16,20,24,30,31)$



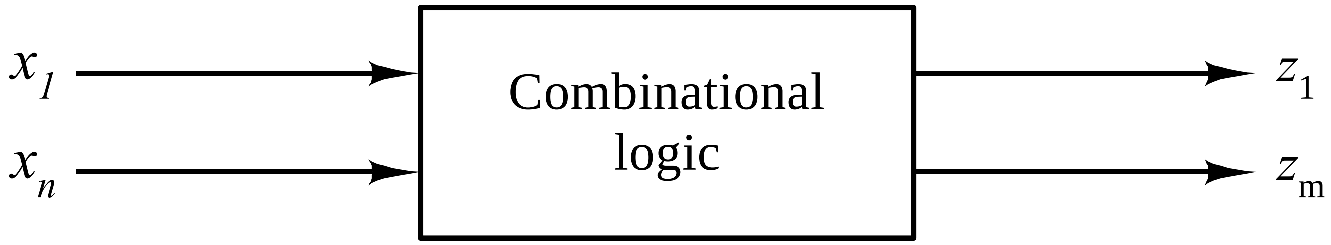
F(a,b,c,d,e) = (b’ + c’ + d’) (b + d + e) (a+ b + c + e) (a’+ c + d + e) (a+ b’ + d’ + e’)

Combinational Logics

Combinational Logics

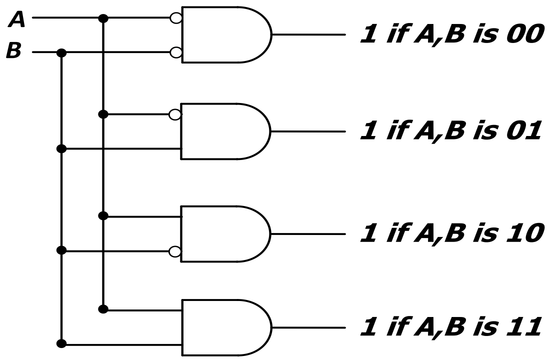
- Always Current inputs produce the output
- Output is independent of
  - Sequence of inputs
  - Time of applying inputs

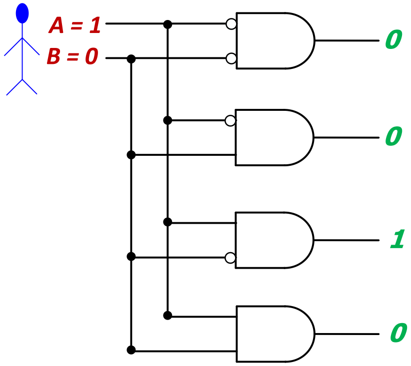
- => Combinational logics are memory less
  - Memory-less circuits do not contain any feedback lines



Decoder

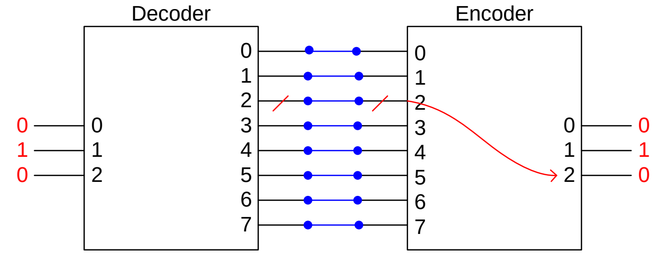
- n inputs and 2^n outputs
- Exactly one of the outputs is 1 and all the rest are 0s
- The one output that is logically 1 is the output corresponding to the input pattern that the logic circuit is expected to detect

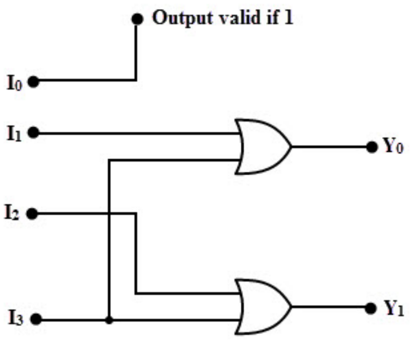




Encoder

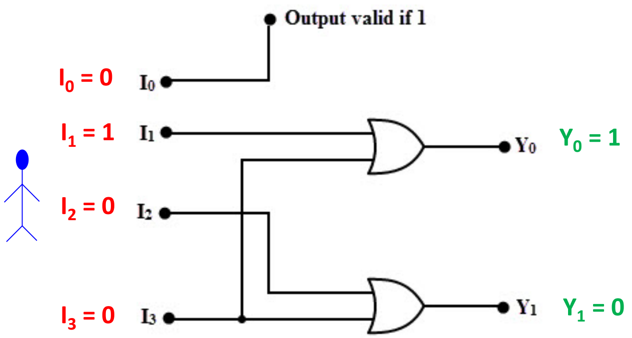
- 2^n inputs and n outputs
- At each time only one input can be active
- Generates the binary code corresponding to the input values





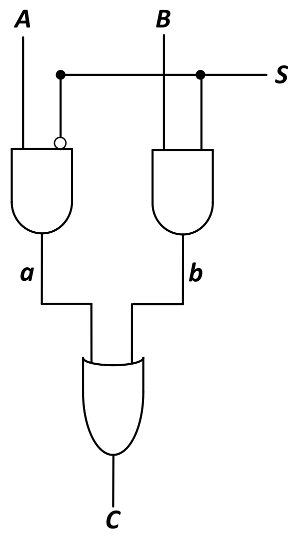
$Y_0 = 1$ *if* $I_1 + I_3$

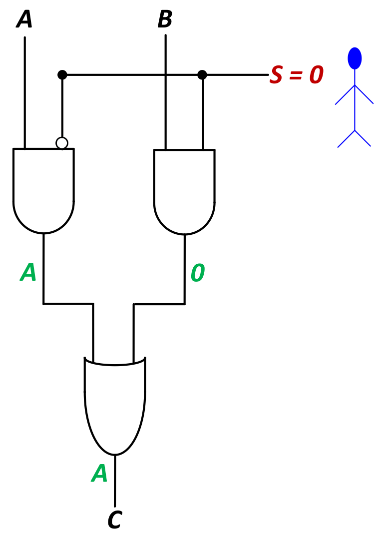
$Y_1 = 1$ *if* $I_2 + I_3$



Multiplexer (MUX)

- Selects one of the N inputs to connect it to the output
- Needs $\log_2 N$-bit control input
- 2:1 MUX

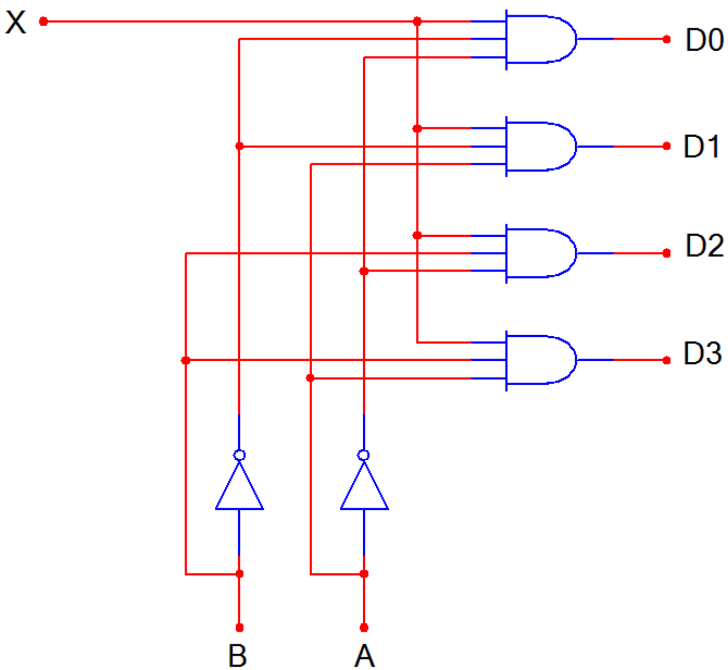


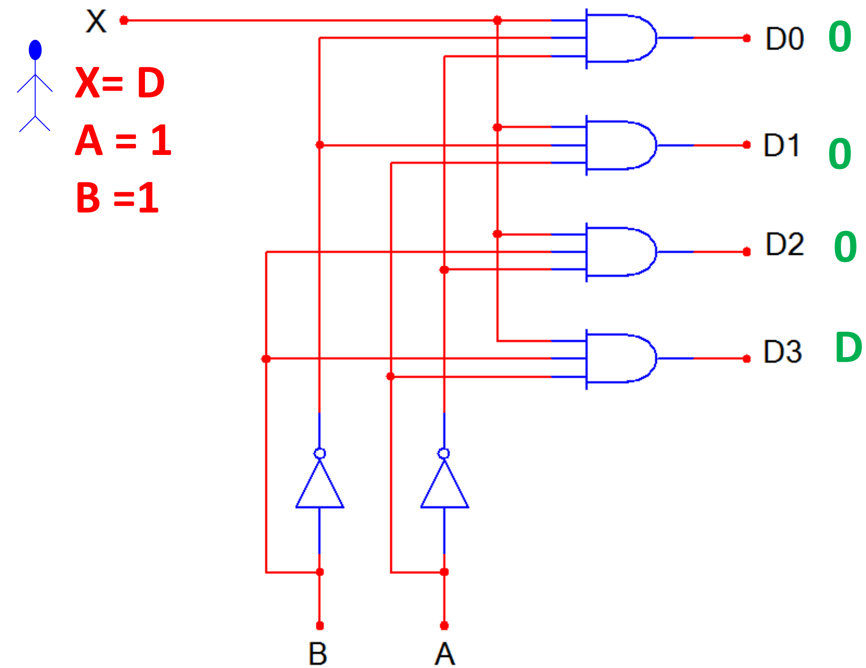


- How is it useful?

DeMultiplexer (DeMUX)

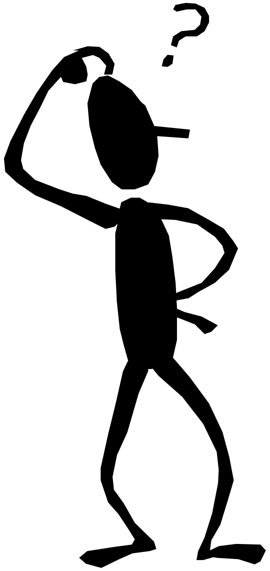
- Selects one of the N outputs and send the data
- Needs $\log_2 N$-bit control input
- 1 input and 2^n output lines





Design a 4-bit Adder

- Design a 1-bit half adder
- Design a 1-bit full adder
- Design a 4-bit full adder

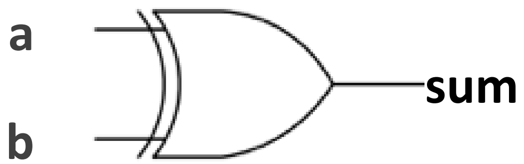


Step1: 1-bit Half Adder: Truth Table

a	b	C_{out}	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

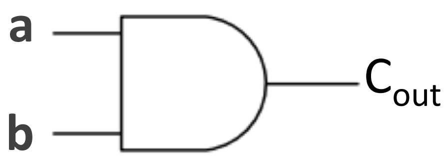
Step1: 1-bit Half Adder: minterm for sum

a	b	C_{out}	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

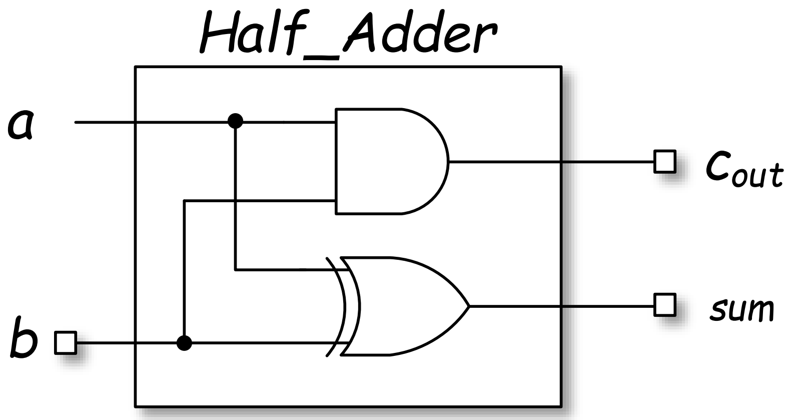


Step1: 1-bit Half Adder: minterm for C_{out}

a	b	C_{out}	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Step1: Design a 1-bit Half Adder



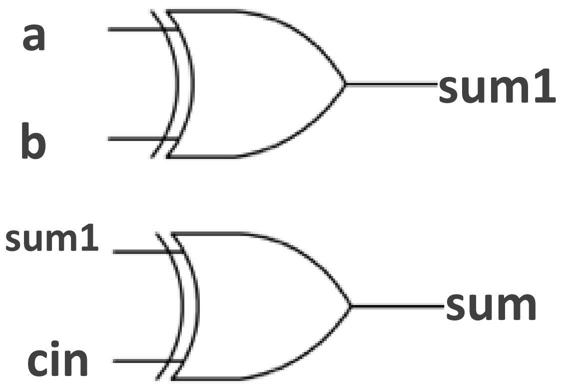
a	b	C_{out}	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

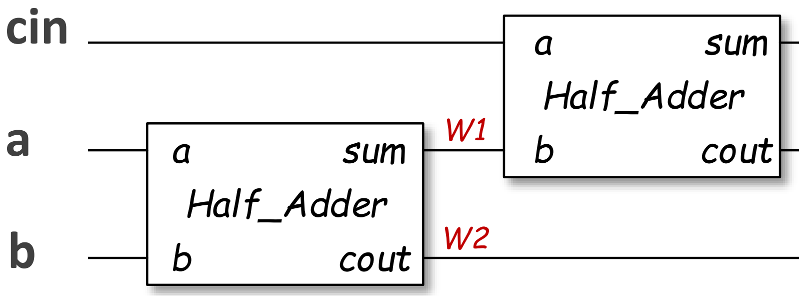
Step2: 1-bit Full Adder: Truth Table

a	b	cin	C_{out}	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Step2: 1-bit Full Adder: minterms of sum

a	b	cin	C_{out}	sum	sum1
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	1	1	0





Step2: 1-bit Full Adder: minterm for C_{out}

a	b	C_{in}	C_{out}	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

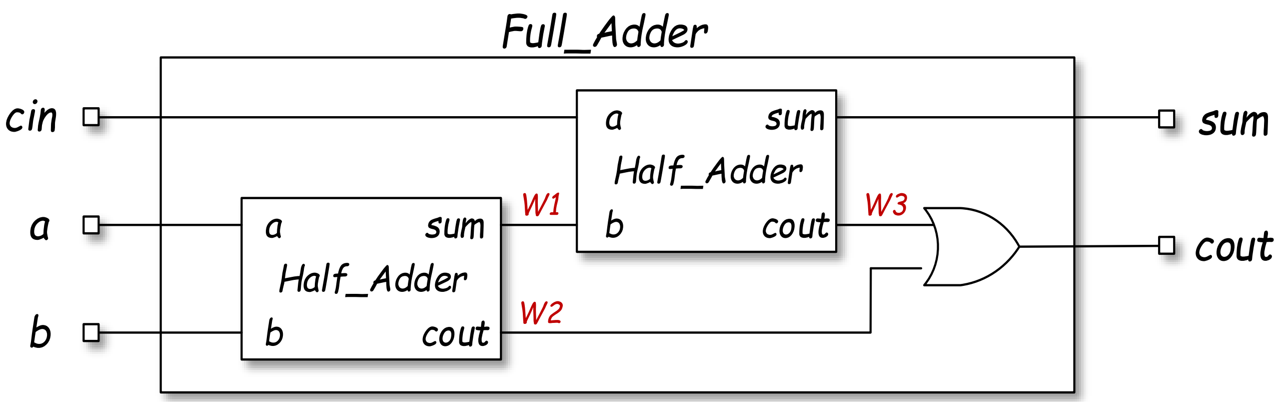
$$C_{out} = abC_{in} + ab\overline{C_{in}} + a\bar{b}\,C_{in} + \bar{a}\,b\,C_{in}$$

$$C_{out} = ab + a\bar{b}\,C_{in} + \bar{a}\,b\,C_{in}$$

$$C_{out} = ab + C_{in}(a \wedge b)$$

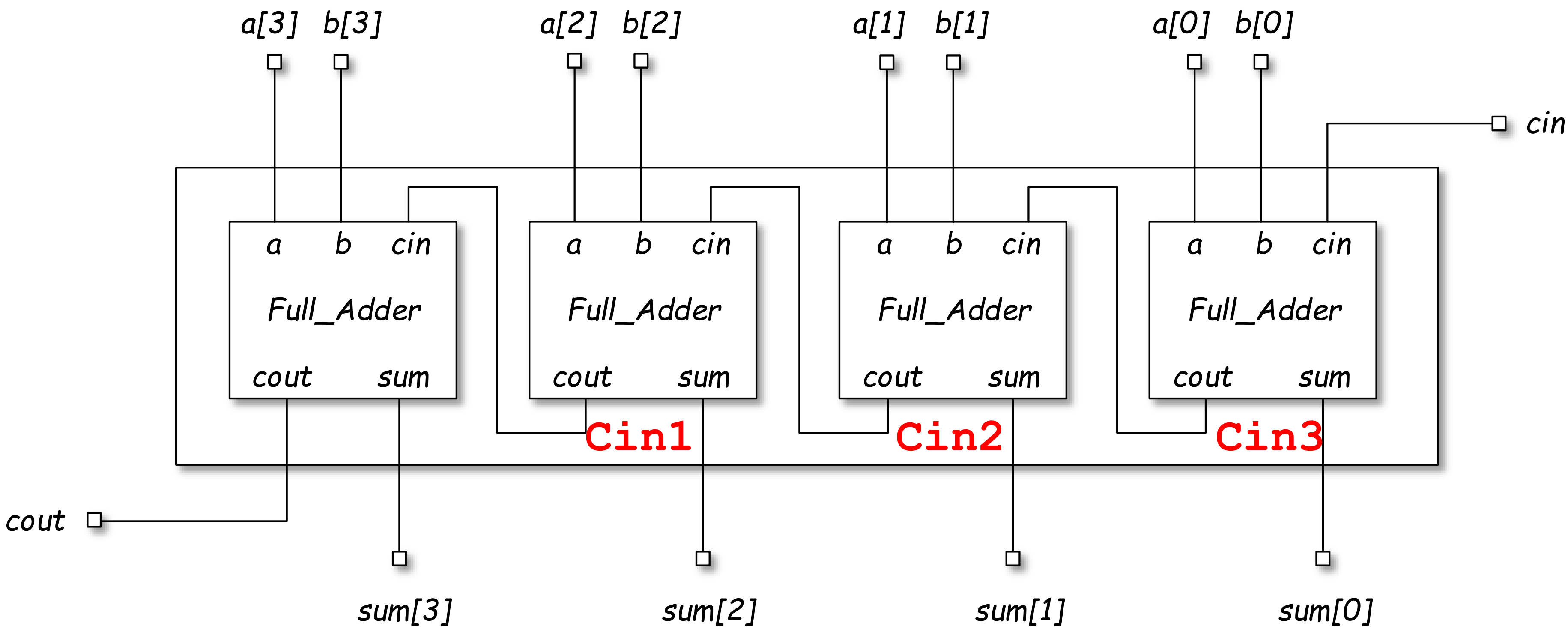
$$C_{out} = ab + C_{in}(a + b)$$

Step2: Design a 1-bit Full Adder

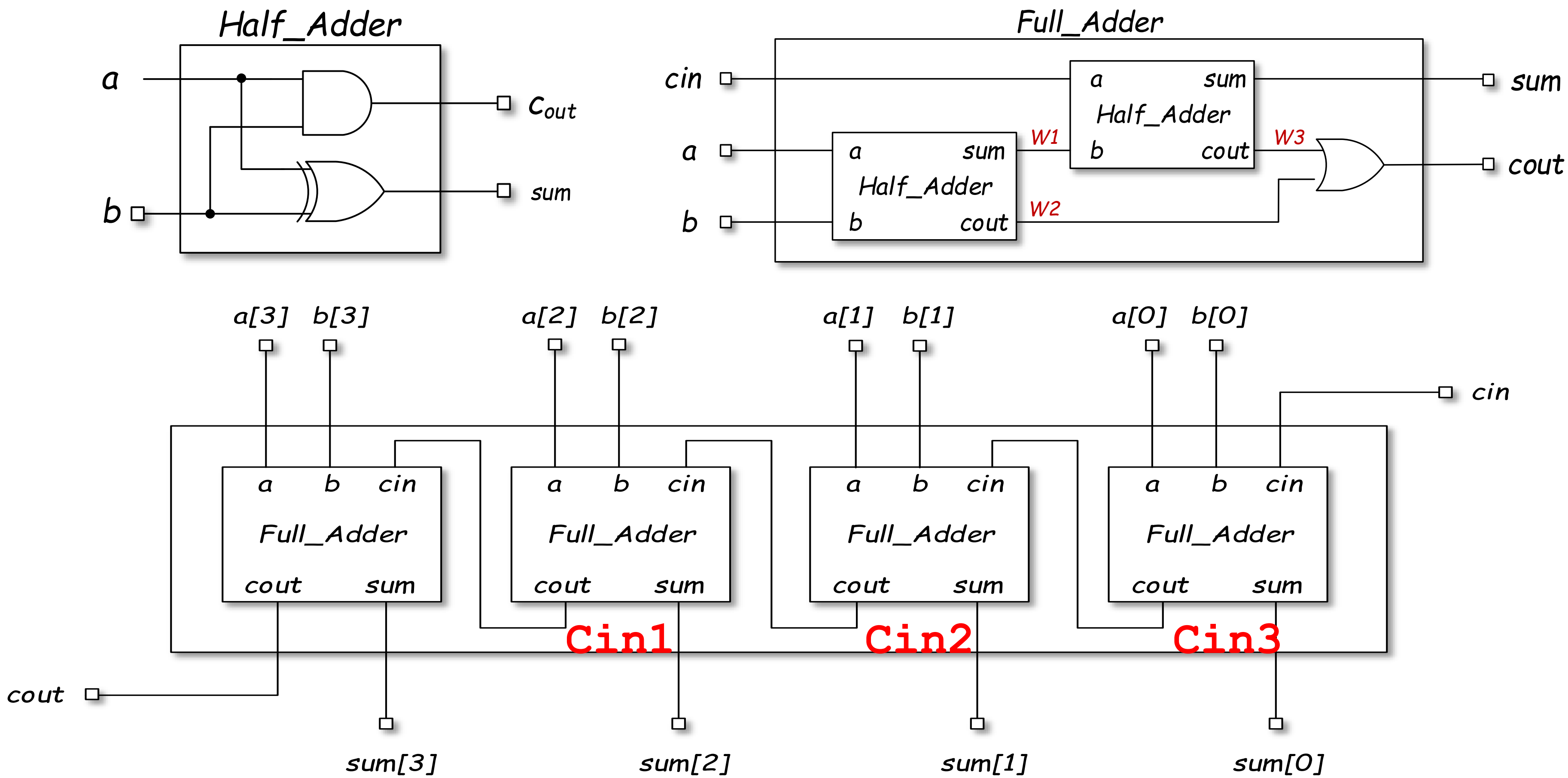


a	b	cin	C_{out}	sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	1	1

Step3: Design a 4-bit Full Adder



A 4-bit Full Adder



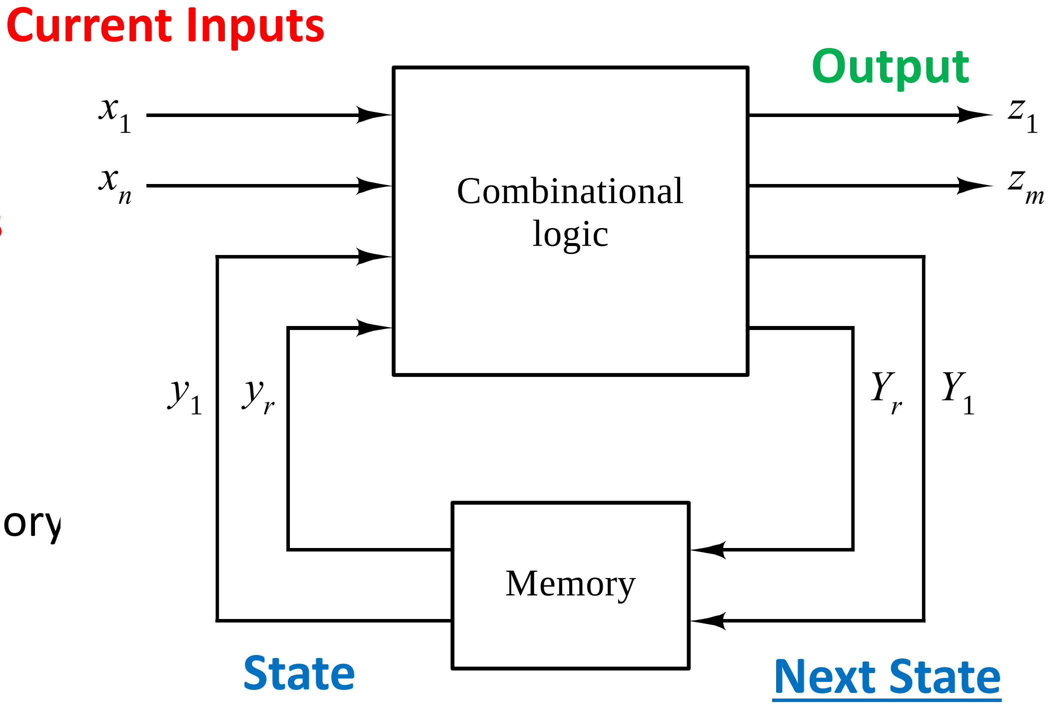
Sequential Logics

Combinational Logics + Memory

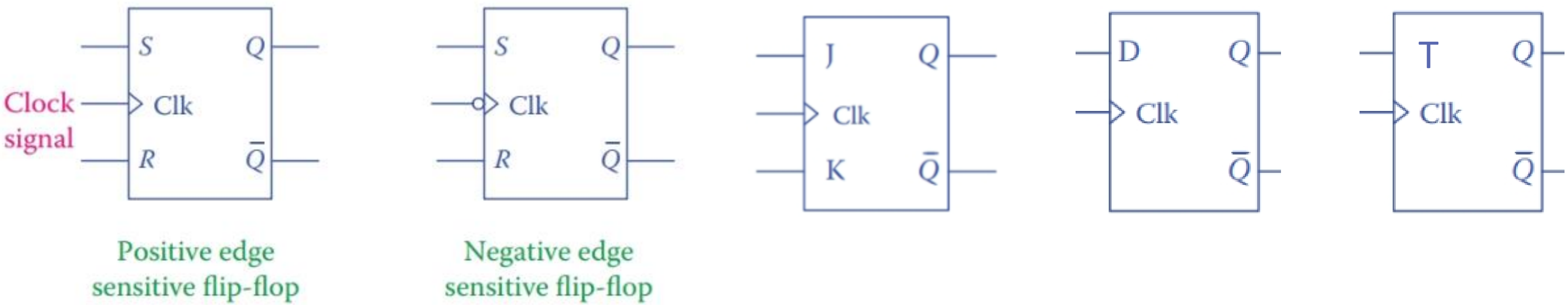
- **Output** of some logics **changes** by
  - **Sequence of inputs**
  - **Time of applying inputs**

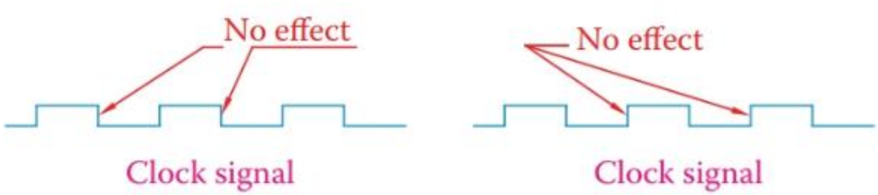
- => They **have memory**
- They **contain feedback lines**

- State
  - Produced by previous inputs
  - Information stored in a memory



Flip Flops





	R	S	Q	$\bar{Q}$
Hold the value	0	0	Q	$\bar{Q}$
Set the value	0	1	1	0
Reset the value	1	0	0	1
Invalid	1	1	Invalid	

J	K	Q	$\bar{Q}$	
0	0	Q	$\bar{Q}$	Hold the value
0	1	0	1	Reset the value
1	0	1	0	Set the value
1	1	$\bar{Q}$	Q	Toggle the value

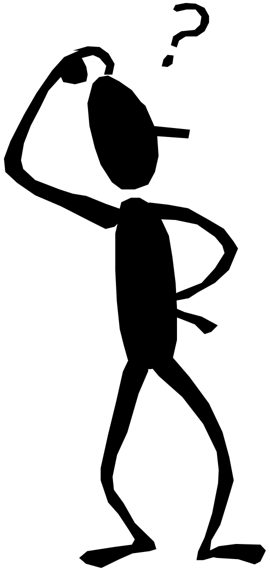
D	Q	$\bar{Q}$
0	0	1
1	1	0

Delay FF

T	Q	$\bar{Q}$	
0	Q	$\bar{Q}$	Hold the value
1	$\bar{Q}$	Q	Toggle the value

Design a Counter

- Design a 3-bit synchronous bidirectional counter
  - Input $Up\ /\ \overline{Down}\ \ (Up)$
  - Up =1 => Counting upward
  - Up =0 => Counting downward



Design a Counter: Step1

Present State			Next State $Up\,/\,\overline{Down}$ =1			Next State $Up\,/\,\overline{Down}$ =0			Flip Flops $Up\,/\,\overline{Down}$ =1			Flip Flops $Up\,/\,\overline{Down}$ =0		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0	T_2	T_1	T_0
0	0	0	0	0	1	1	1	1			1			1
0	0	1	0	1	0	0	0	0			1			1
0	1	0	0	1	1	0	0	1			1			1
0	1	1	1	0	0	0	1	0			1			1
1	0	0	1	0	1	0	1	1			1			1
1	0	1	1	1	0	1	0	0			1			1
1	1	0	1	1	1	1	0	1			1			1
1	1	1	0	0	0	1	1	0			1			1

Design a Counter: Step1

Present State			Next State $Up\,/\,\overline{Down}$ =1			Next State $Up\,/\,\overline{Down}$ =0			Flip Flops $Up\,/\,\overline{Down}$ =1			Flip Flops $Up\,/\,\overline{Down}$ =0		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0	T_2	T_1	T_0
0	0	0	0	0	1	1	1	1		0	1		1	1
0	0	1	0	1	0	0	0	0		1	1		0	1
0	1	0	0	1	1	0	0	1		0	1		1	1
0	1	1	1	0	0	0	1	0		1	1		0	1
1	0	0	1	0	1	0	1	1		0	1		1	1
1	0	1	1	1	0	1	0	0		1	1		0	1
1	1	0	1	1	1	1	0	1		0	1		1	1
1	1	1	0	0	0	1	1	0		1	1		0	1

Design a Counter: Step1

Present State			Next State $Up\,/\,\overline{Down}$ =1			Next State $Up\,/\,\overline{Down}$ =0			Flip Flops $Up\,/\,\overline{Down}$ =1			Flip Flops $Up\,/\,\overline{Down}$ =0		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	T_2	T_1	T_0	T_2	T_1	T_0
0	0	0	0	0	1	1	1	1	0	0	1	1	1	1
0	0	1	0	1	0	0	0	0	0	1	1	0	0	1
0	1	0	0	1	1	0	0	1	0	0	1	0	1	1
0	1	1	1	0	0	0	1	0	1	1	1	0	0	1
1	0	0	1	0	1	0	1	1	0	0	1	1	1	1
1	0	1	1	1	0	1	0	0	0	1	1	0	0	1
1	1	0	1	1	1	1	0	1	0	0	1	0	1	1
1	1	1	0	0	0	1	1	0	1	1	1	0	0	1

$T_0 = 1$

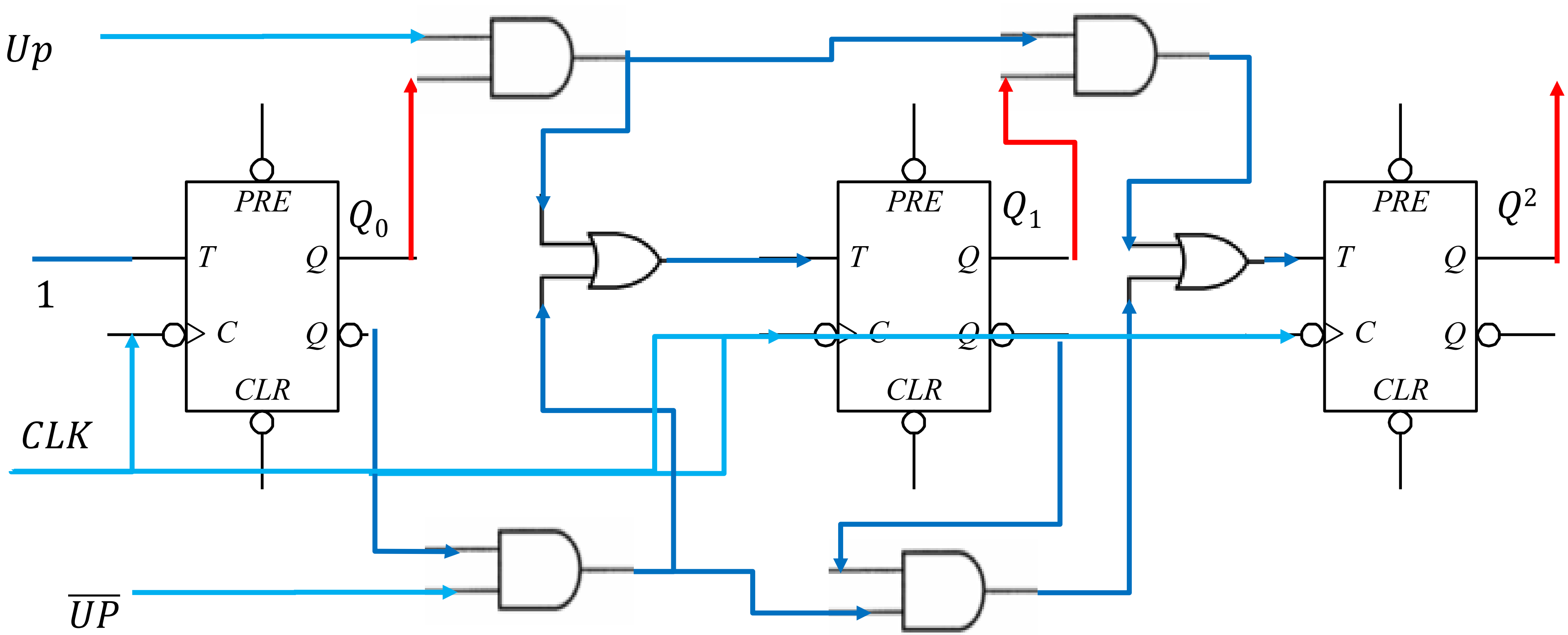
$T_1 = (Q_0 Up) + \overline{Q_0} \,.\, \overline{UP}$

$T_2 = (Q_0 Q_1 Up) + \overline{Q_0} \,.\, \overline{Q_1} . \overline{UP}$

Q_2

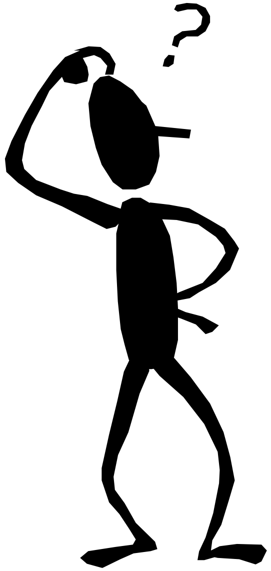
Design a Counter: Step1

$T_0 = 1$ $\quad$ $T_1 = (Q_0 Up) + \overline{Q_0} . \overline{UP}$ $\quad$ $T_2 = (Q_0 Q_1 Up) + \overline{Q_0} . \overline{Q_1} . \overline{UP}$



Design an Asynchronous Counter

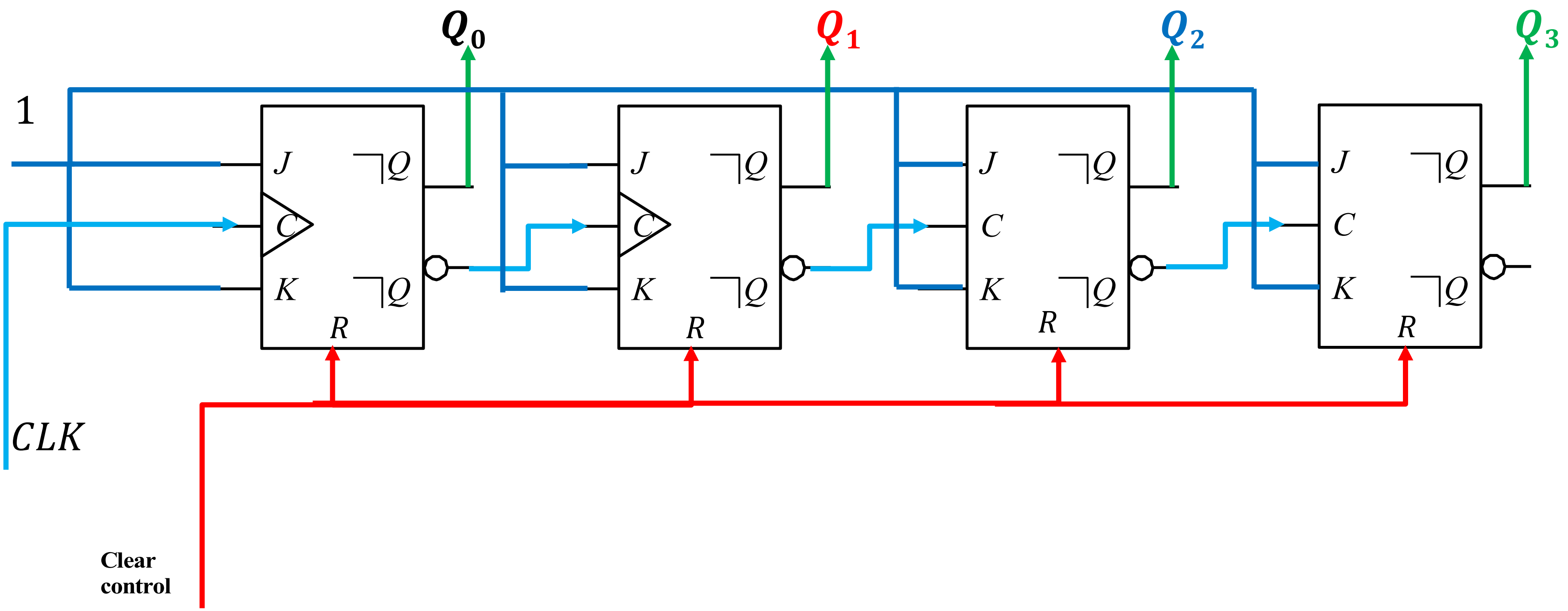
- Design a 4-bit asynchronous counter
  - Counting upward



Asynchronous 4-bit Counter

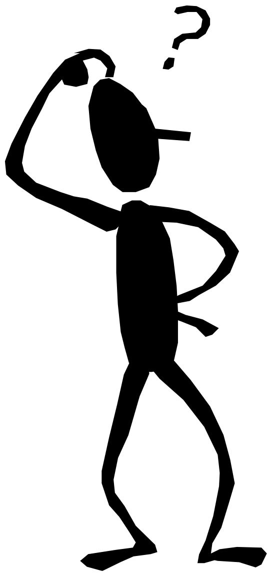
Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

Asynchronous 4-bit Counter

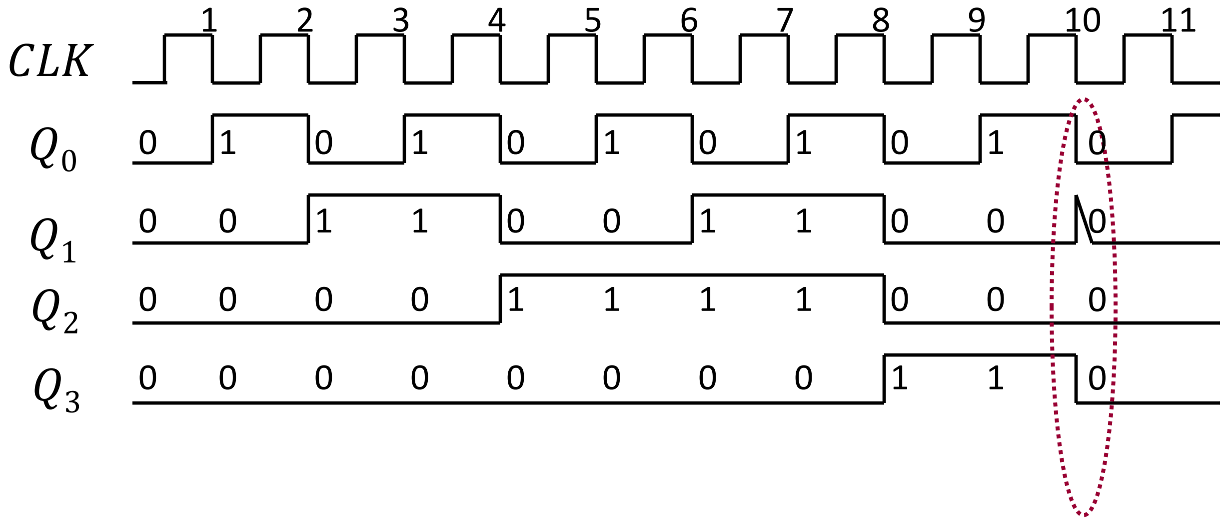


Design an Asynchronous Counter

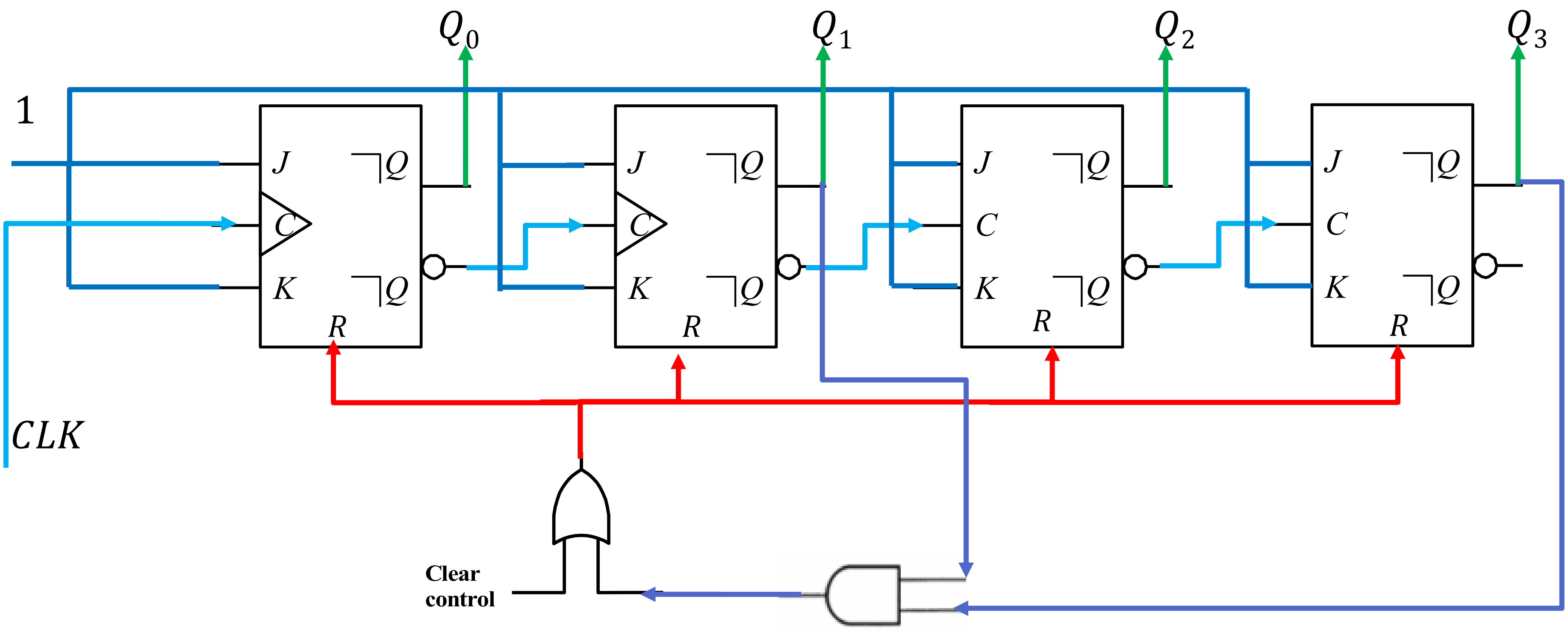
- Design a 4-bit asynchronous counter with mode 10



Asynchronous 4-bit Counter: Timing Diagram



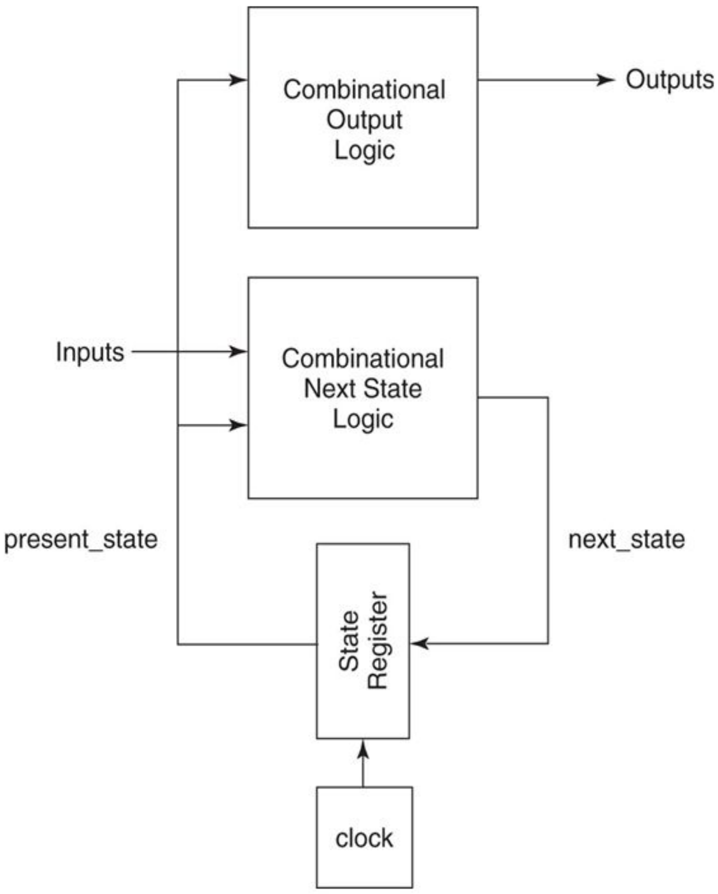
Asynchronous 4-bit Counter: Realization



Simplification

- Design a **simplified** sequential circuit
  - Reduce the number of states
  - <u>**Without changing the functionality**</u>

- Advantages of reducing the states
  - Less memory elements
  - => decreases cost
  - => decreases complexity
  - => Aids failure analysis



How to Simplify?

- Find equivalent states
  - E.g., S0, S1, S2
- Keep one of these equivalent states
  - E.g., S0
- Remove other equivalent states in each group ones
  - E.g., S1, S2

Equivalent States

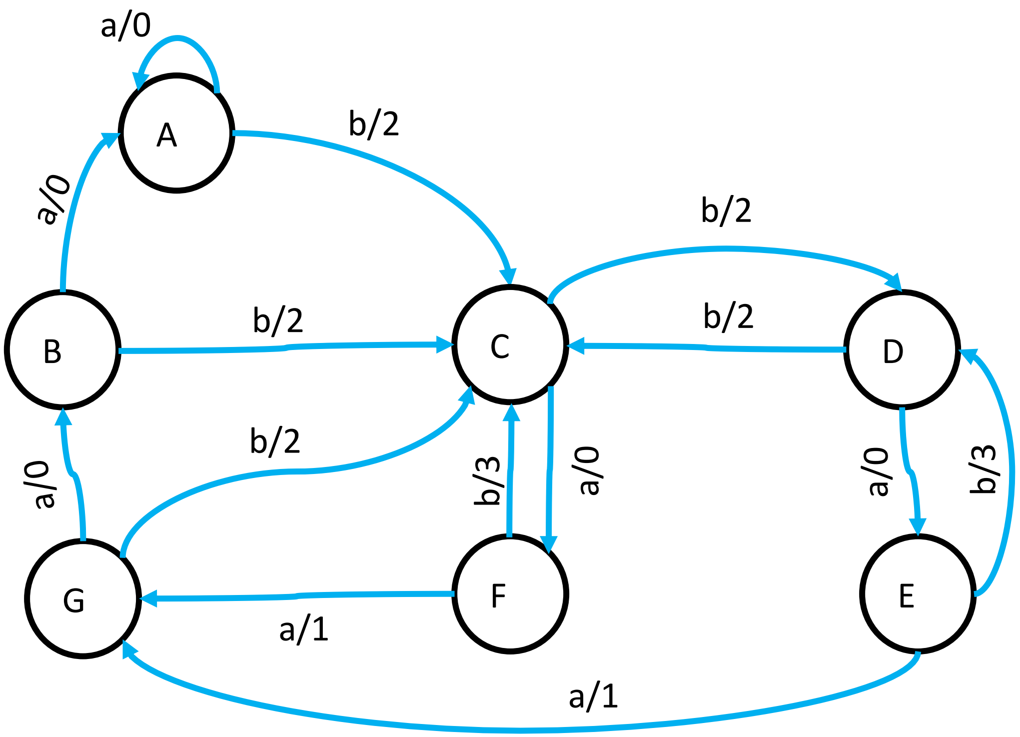
- S_1, S_2, …, S_j are **equivalent** *If and ony if*
  - For **every possible input sequence**
  - **Same output sequence** is produced
    - Same **output**
    - Same **next state**

- S_1, S_2, …, S_j are **conditional equivalent** *If and ony if*
  - For **every possible input sequence**
    - Same **output**
    - Different **next state**
  - **Next states should** be **equivalent**

Implication Table

- A procedure for finding all the equivalent states

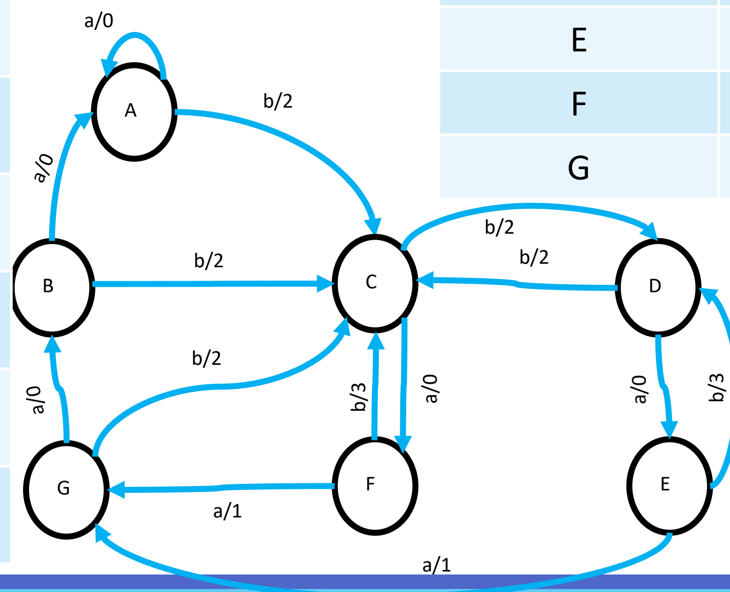
Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2



Implication Table: Big Picture

A							
B							
C							
D							
E							
F							
G							
	A	B	C	D	E	F	G

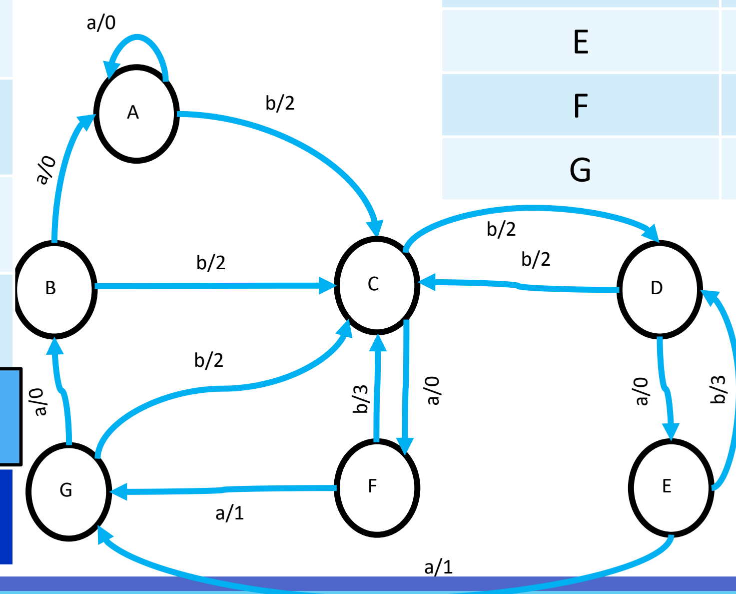
Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2



Do We Need All the Cubes?

A							
B							
C							
D							
E							
F							
G							
	A	B	C	D	E	F	G

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

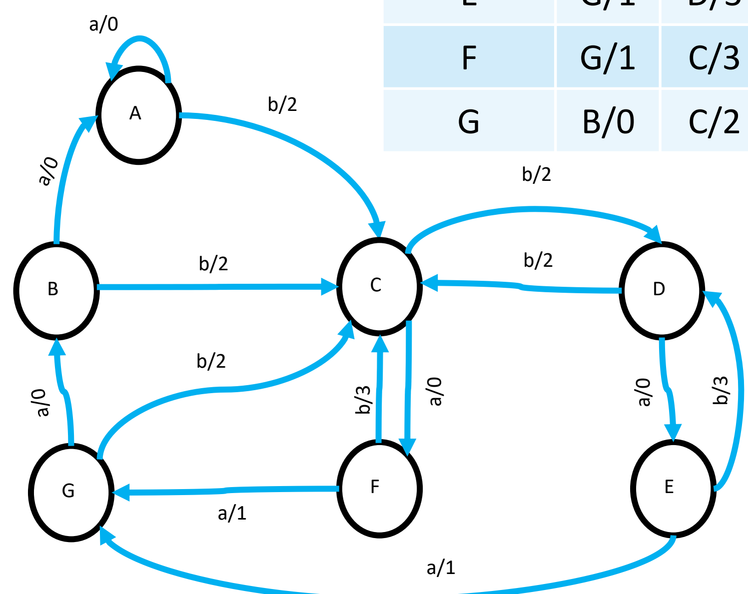


Implication Table: Step 1

- Draw a chart that has a square for each pair of states

B						
C						
D						
E						
F						
G						
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

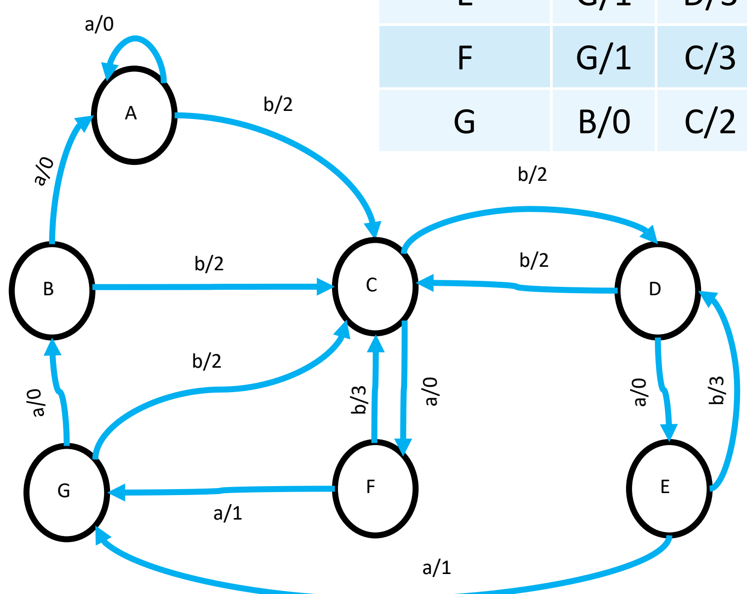


Implication Table: Step 2

- Find incompatible States
- Put X in the corresponding square

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

B						
C						
D						
E						
F						
G						
	A	B	C	D	E	F

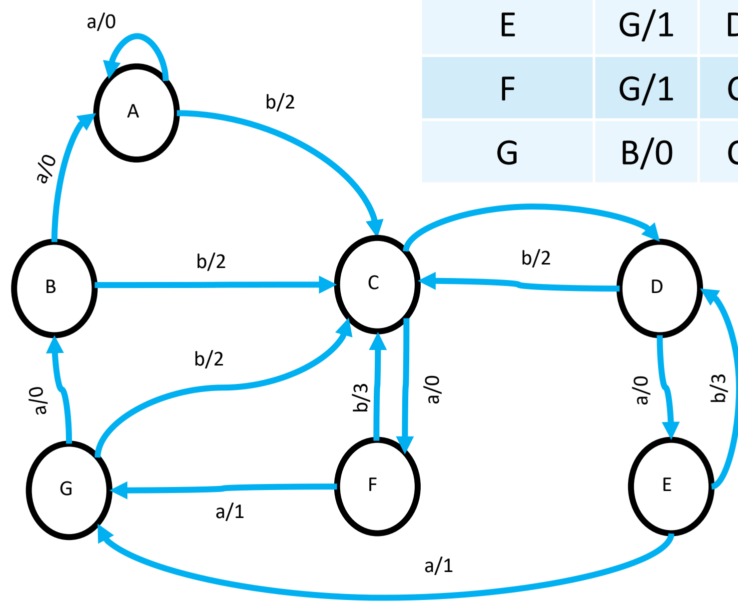


Implication Table: Step 2 (cont’d)

- Find incompatible States
- Put X in the corresponding square

B						
C						
D						
E	X					
F	X					
G						
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

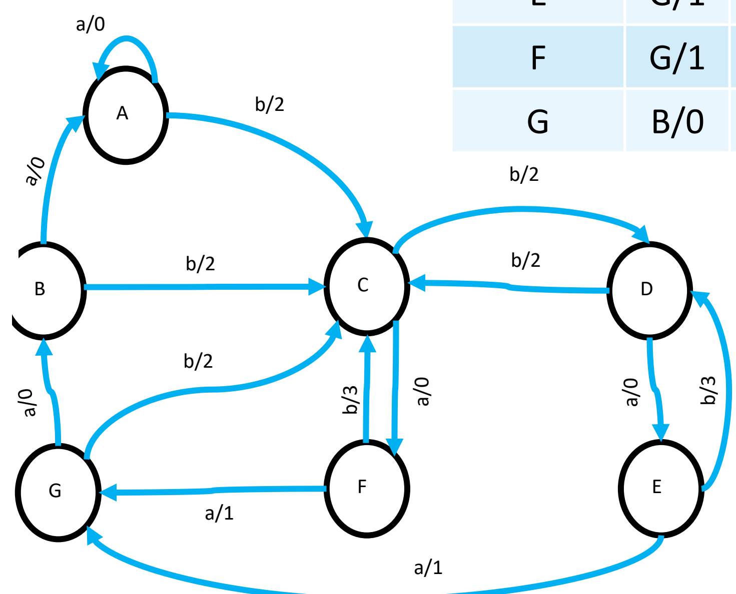


Implication Table: Step 2 (cont'd)

- Find incompatible States
- Put X in the corresponding square

Present State	Next State a	Next State b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

B						
C						
D						
E	X	X				
F	X	X				
G						
	A	B	C	D	E	F

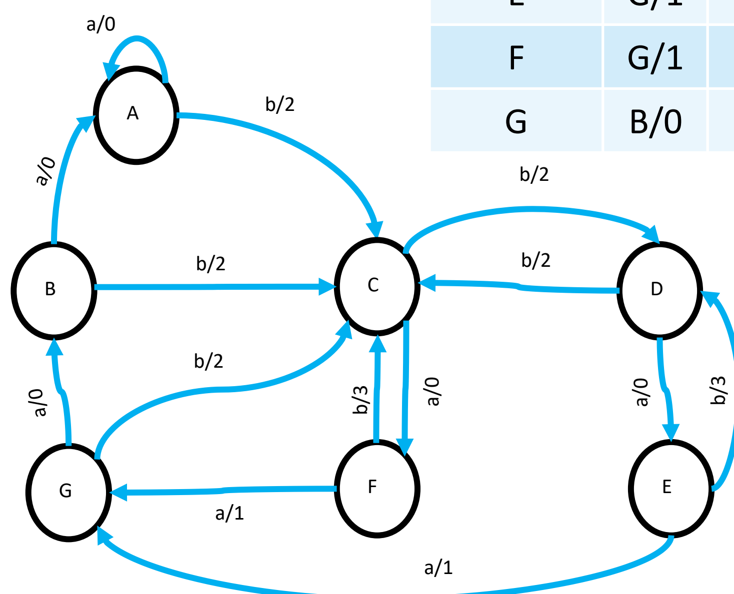


Implication Table: Step 2 (cont’d)

- Find incompatible States
- Put X in the corresponding square

B						
C						
D						
E	X	X	X			
F	X	X	X			
G						
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

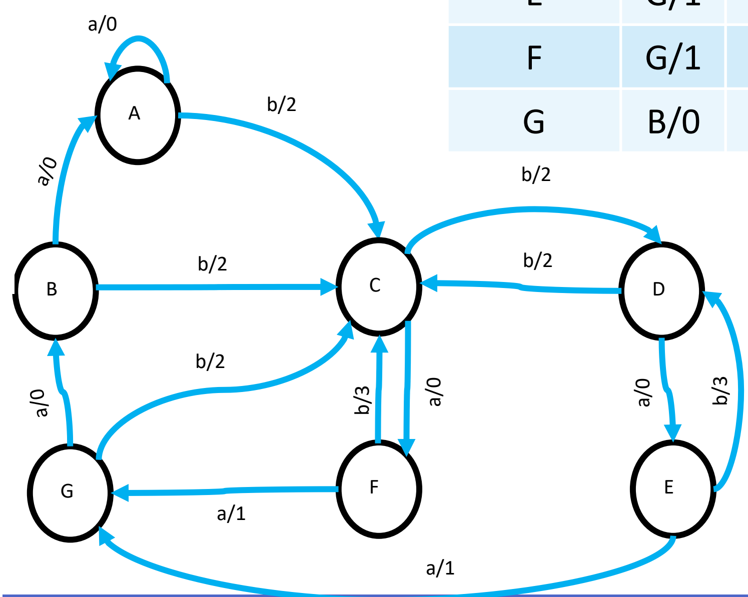


Implication Table: Step 2 (cont'd)

- Find incompatible States
- Put X in the corresponding square

B						
C						
D						
E	X	X	X	X		
F	X	X	X	X		
G						
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

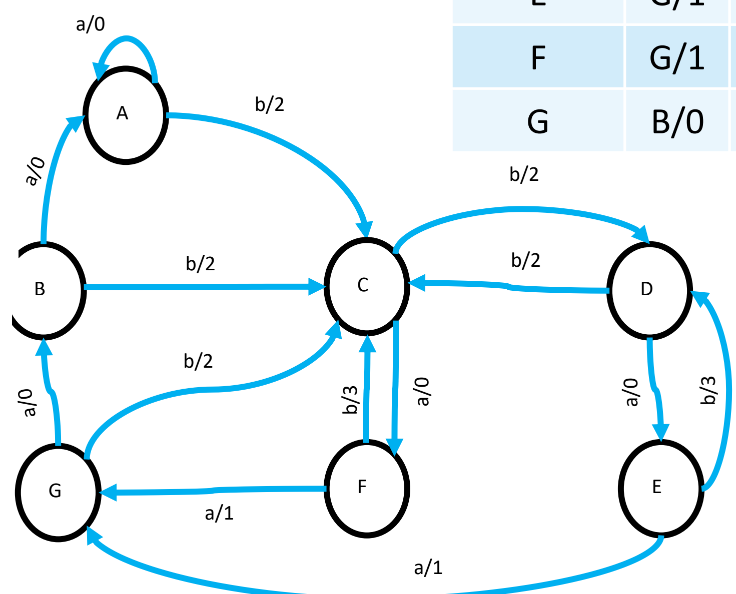


Implication Table: Step 2 (cont'd)

- Find incompatible States
- Put X in the corresponding square

Present State	Next State a	Next State b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

B						
C						
D						
E	X	X	X	X		
F	X	X	X	X		
G					X	X
	A	B	C	D	E	F

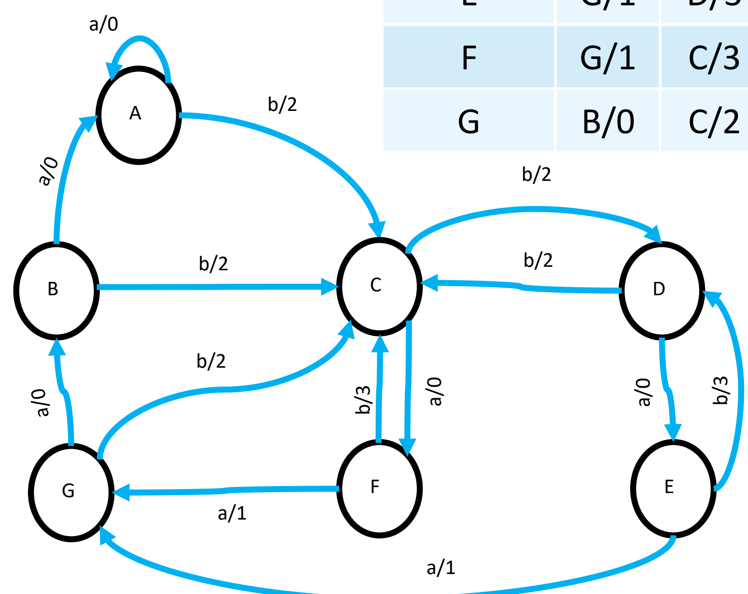


Implication Table: Step 3

- Enter implied pair in non square
  - Put ✓ for equivalent states
  - Write conditional states for conditional equivalent states

B						
C						
D						
E	X	X	X	X		
F	X	X	X	X		
G					X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

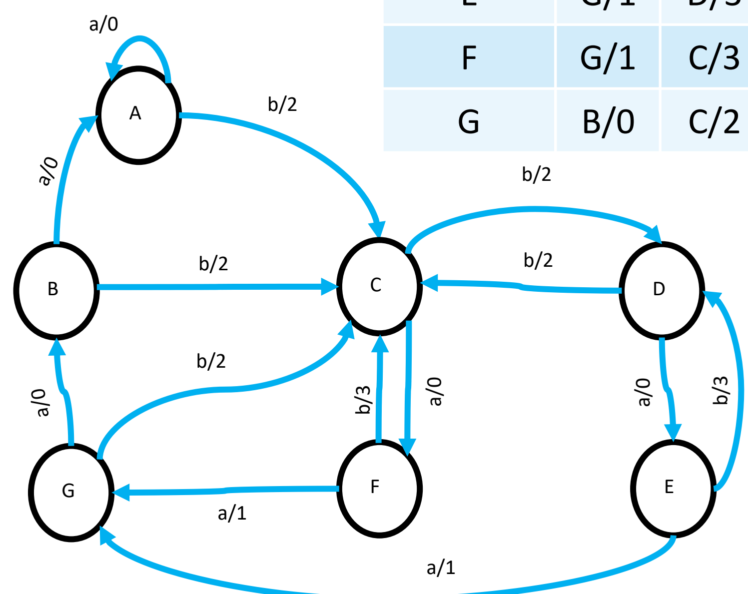


Implication Table: Step 3 (cont'd)

- Enter implied pair in non square
  - Put *V* for equivalent states
  - Write conditional states for conditional equivalent states

B	V					
C						
D						
E	X	X	X	X		
F	X	X	X	X		
G					X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

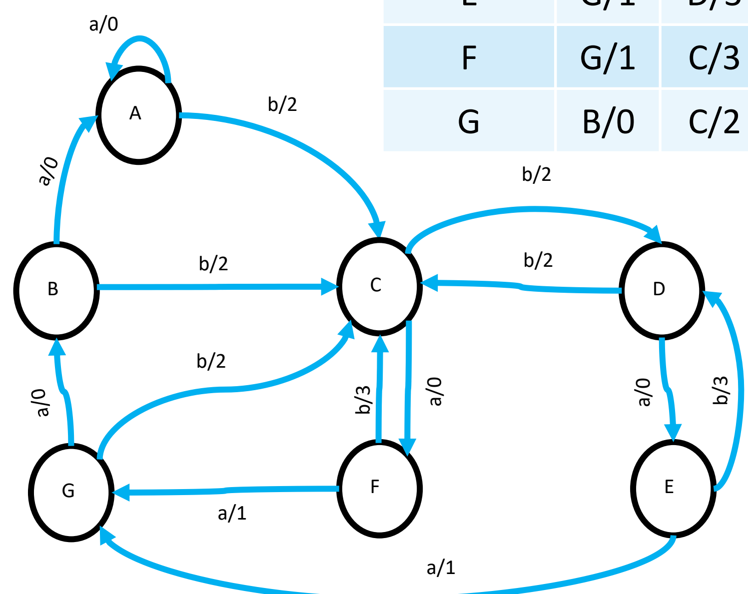


Implication Table: Step 3 (cont’d)

- Enter implied pair in non square
  - Put *V* for equivalent states
  - Write conditional states for conditional equivalent states

	A	B	C	D	E	F
B	*V*					
C	A=F C=D					
D	A=E					
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B				X	X

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

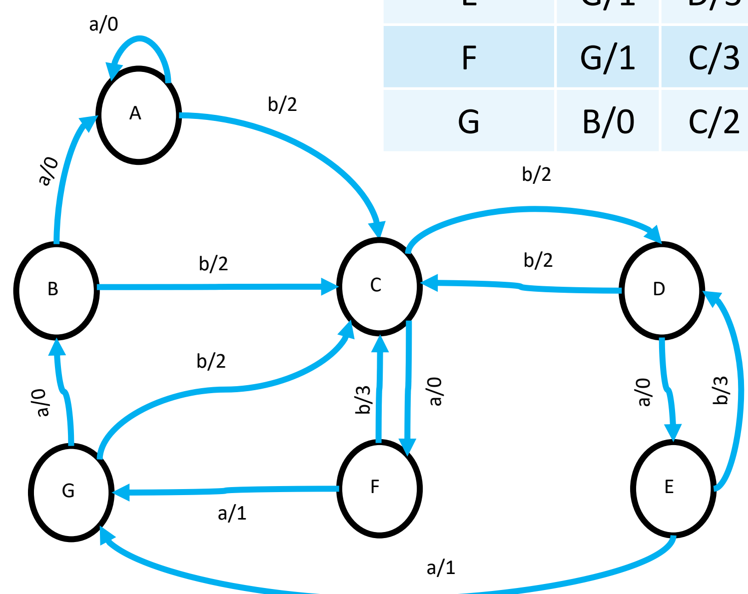


Implication Table: Step 3 (cont’d)

- Enter implied pair in non square
  - Put *V* for equivalent states
  - Write conditional states for conditional equivalent states

B	*V*					
C	A=F C=D	A=F C=D				
D	A=E	A=E				
E	X	X	X	X		
F	X	X	X	X		
G	A=B	A=B			X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

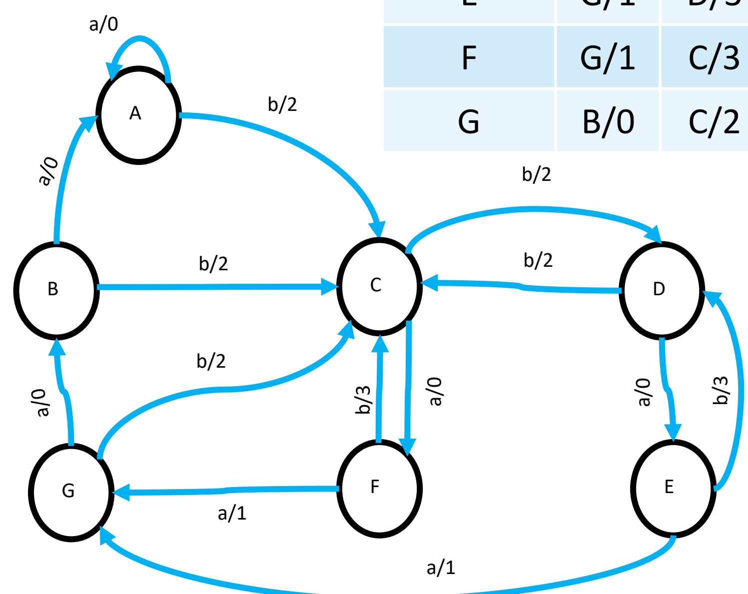


Implication Table: Step 3 (cont'd)

- Enter implied pair in non square
  - Put *V* for equivalent states
  - Write conditional states for conditional equivalent states

B	*V*					
C	A=F C=D	A=F C=D				
D	A=E	A=E	C=D E=F			
E	X	X	X	X		
F	X	X	X	X		
G	A=B	A=B	C=D B=F		X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

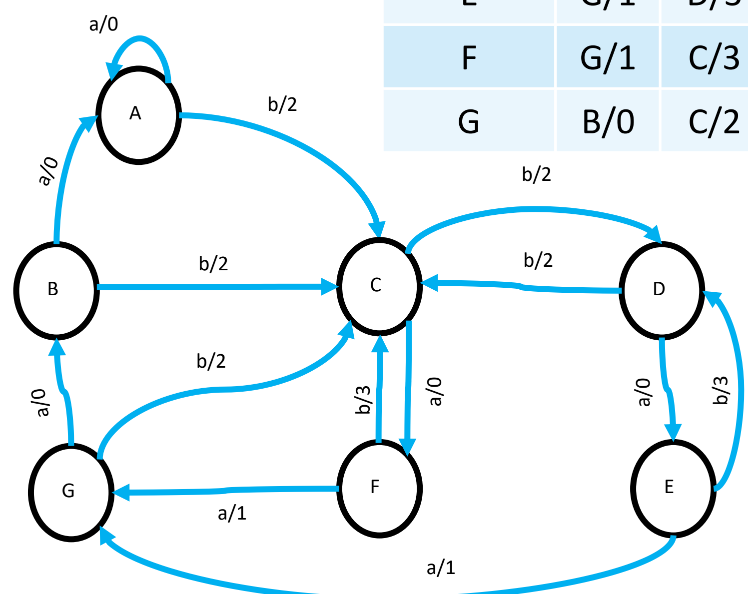


Implication Table: Step 3 (cont’d)

- Enter implied pair in non square
  - Put *V* for equivalent states
  - Write conditional states for conditional equivalent states

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

B	*V*					
C	A=F C=D	A=F C=D				
D	A=E	A=E	C=D E=F			
E	X	X	X	X		
F	X	X	X	X		
G	A=B	A=B	C=D B=F	B=E	X	X
	A	B	C	D	E	F

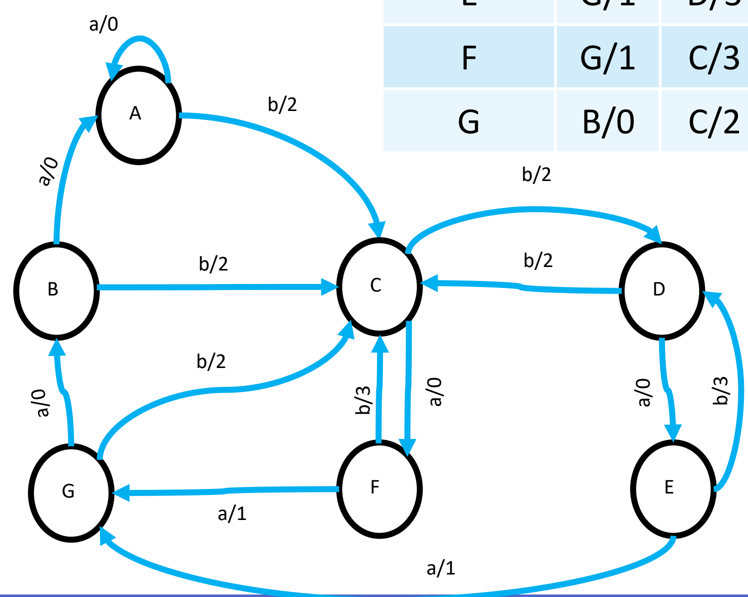


Implication Table: Step 3 (cont'd)

- Enter implied pair in non square
  - Put *V* for equivalent states
  - Write conditional states for conditional equivalent states

	A	B	C	D	E	F
B	*V*					
C	A=F C=D	A=F C=D				
D	A=E	A=E	C=D E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	B=E	X	X

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

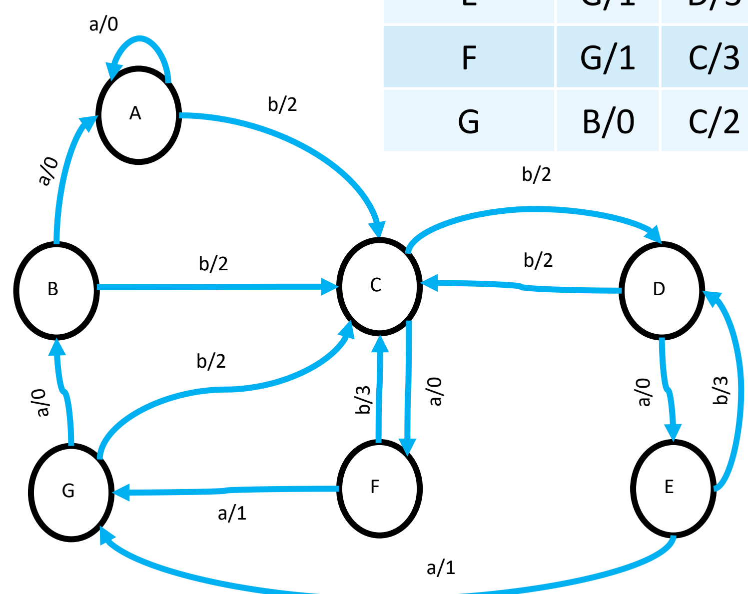


Implication Table: Step 4

- Remove self redundant pairs
  - C-D

B	V					
C	A=F C=D	A=F C=D				
D	A=E	A=E	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	B=E	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

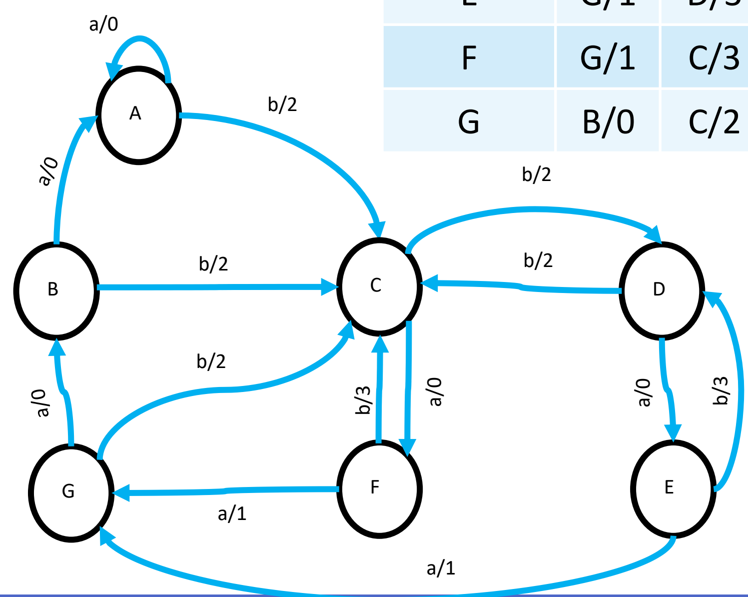


Implication Table: Step 5

- Find squares with implied pairs that are not equivalent
- Put x
  - A-E are not compatible
  - => Each square that has A-E is incompatible too

B	*V*					
C	A=F C=D	A=F C=D				
D	A=E	A=E	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	B=E	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

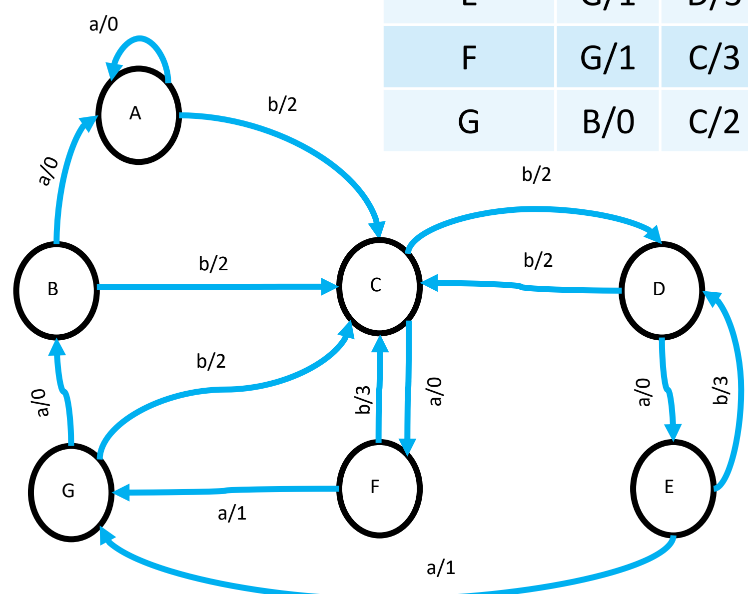


Implication Table: Step 5 (cont'd)

- Find squares with implied pairs that are not equivalent
- Put x
  - A-E are not compatible
  - => Each square that has A-E is incompatible too

B	V					
C	A=F C=D	A=F C=D				
D	~~A=E~~	A=E	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	B=E	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

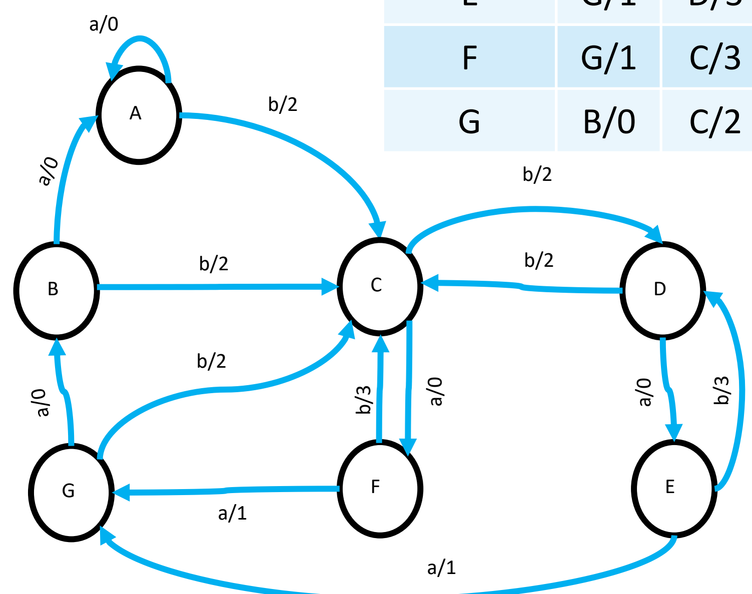


Implication Table: Step 5 (cont'd)

- Find squares with implied pairs that are not equivalent
- Put x
  - A-E are not compatible
  - => Each square that has A-E is incompatible too

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

B	*V*					
C	A=F C=D	A=F C=D				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	B=E	X	X
	A	B	C	D	E	F

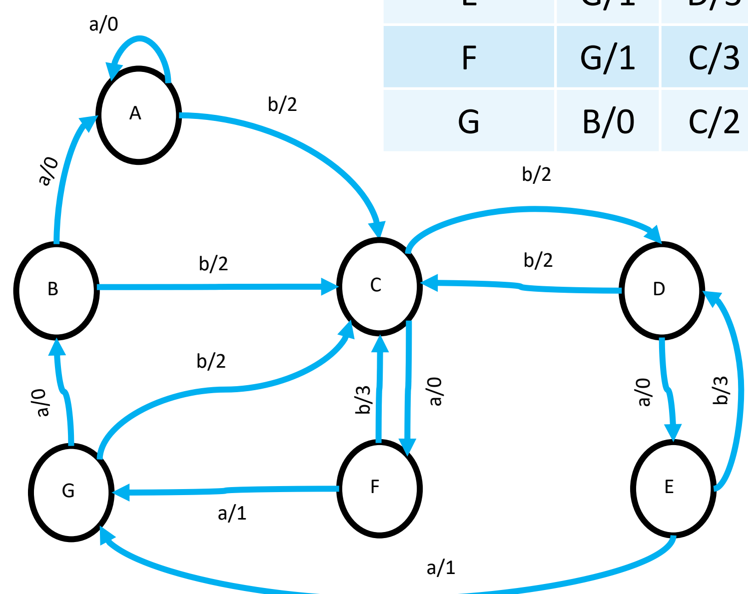


Implication Table: Step 5 (cont'd)

- Find squares with implied pairs that are not equivalent
- Put x
  - A-F are not compatible
  - => Each square that has A-F is incompatible too

B	V					
C	A=F C=D	A=F C=D				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	B=E	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

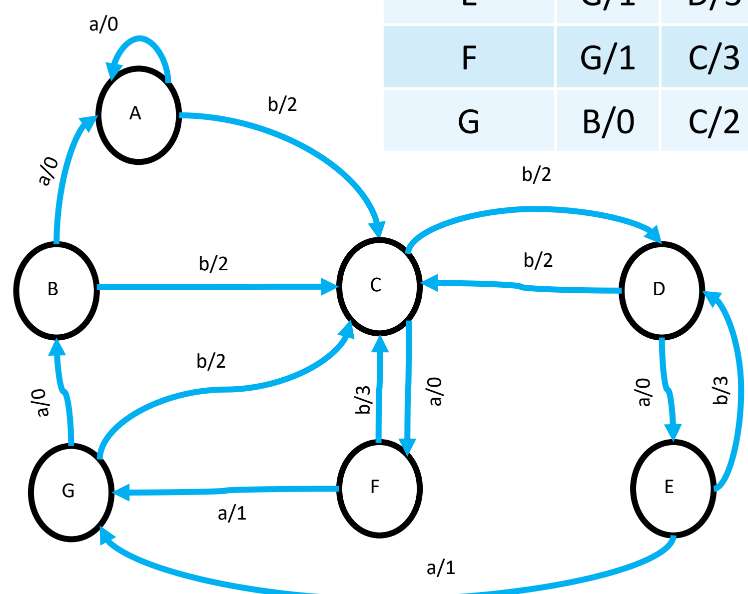


Implication Table: Step 5 (cont’d)

- Find squares with implied pairs that are not equivalent
- Put x
  - A-F are not compatible
  - => Each square that has A-F is incompatible too

	A	B	C	D	E	F
B	*V*					
C	~~A=F~~ ~~C=D~~	A=F C=D				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	B=E	X	X

Present State	Next State a	Next State b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

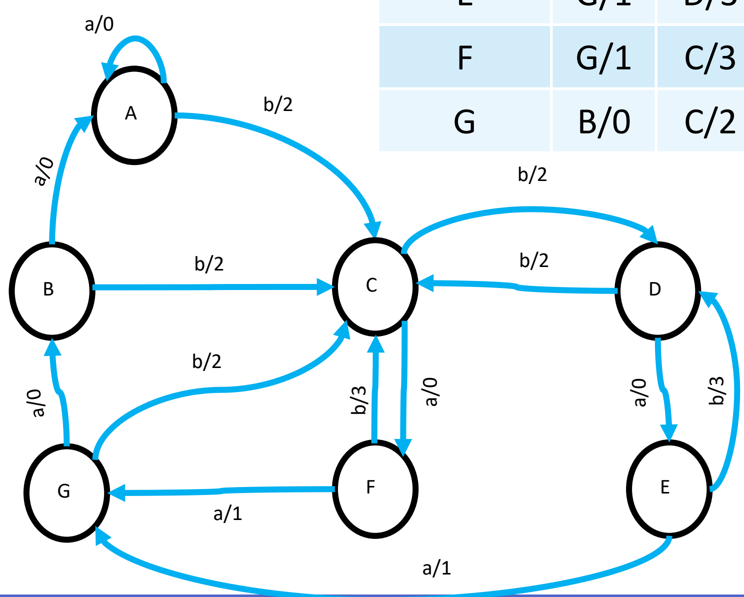


Implication Table: Step 5 (cont’d)

- Find squares with implied pairs that are not equivalent
- Put x
  - B-E are not compatible
  - => Each square that has B-E is incompatible too

B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	B=E	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

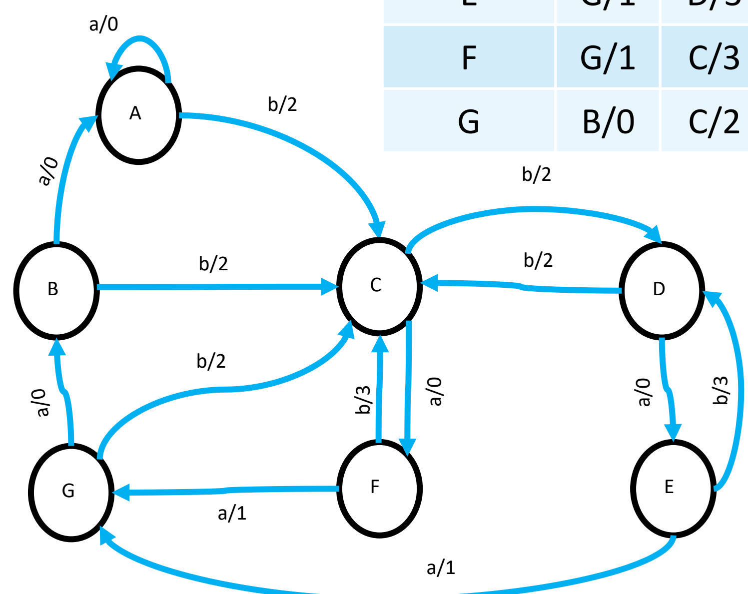


Implication Table: Step 5 (cont'd)

- Find squares with implied pairs that are not equivalent
- Put x
  - B-E are not compatible
  - => Each square that has B-E is incompatible too

B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	~~B=E~~	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

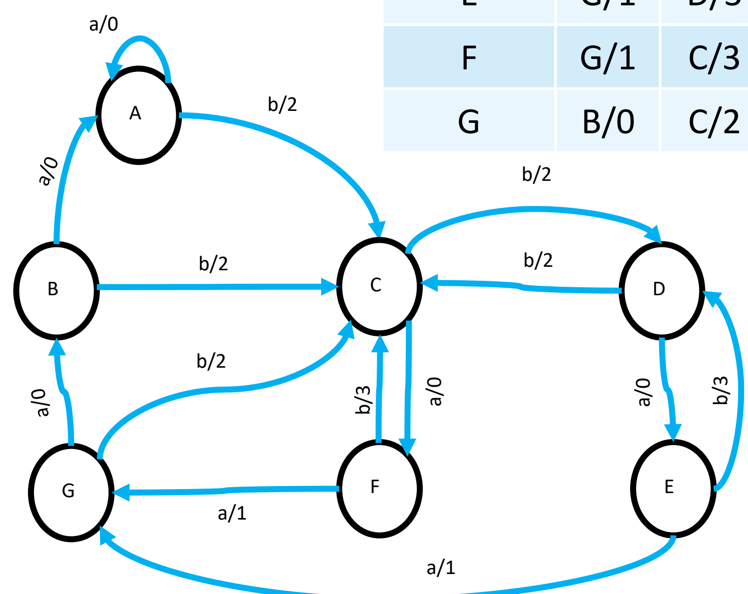


Implication Table: Step 5 (cont'd)

- Find squares with implied pairs that are not equivalent
- Put x
  - B-F are not compatible
  - => Each square that has B-F is incompatible too

B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	C=D B=F	~~B=E~~	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

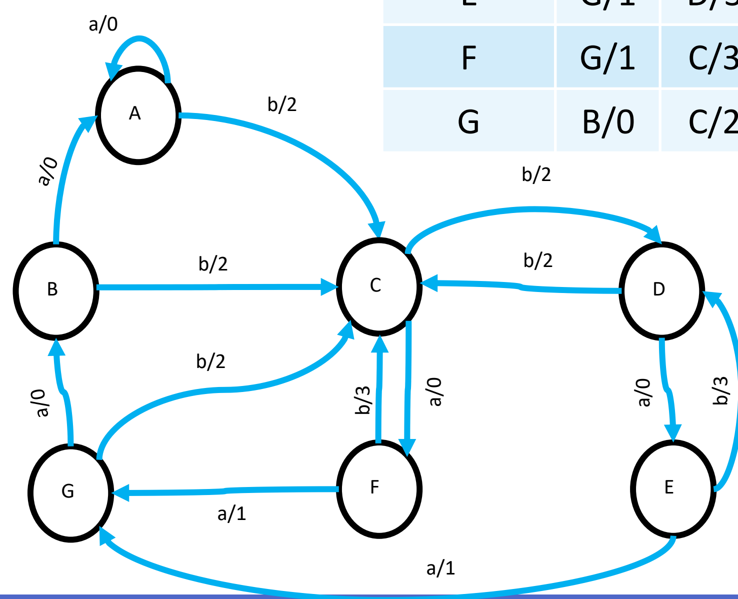


Implication Table: Step 6

- Find squares with implied pairs that are equivalent
- Put *V*
  - A-B are not compatible
  - => Each square that has A-B is incompatible too

B	*V*					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B	A=B	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

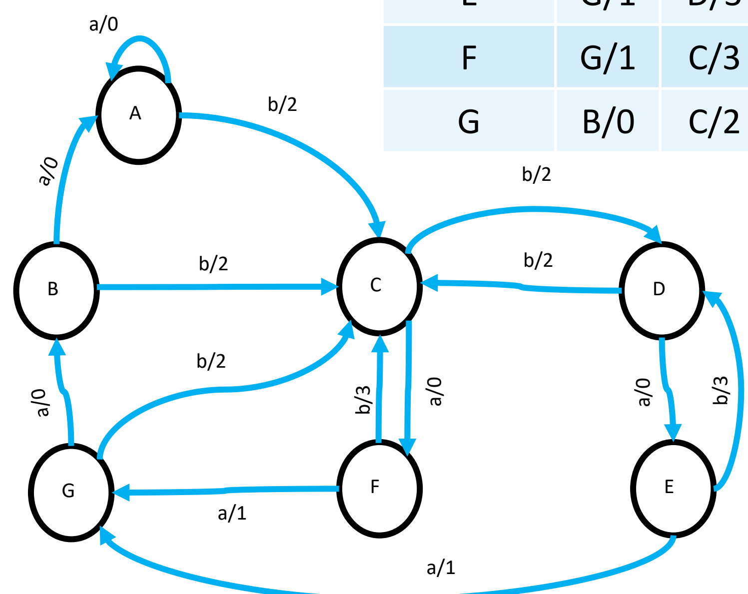


Implication Table: Step 6 (cont'd)

- Find squares with implied pairs that are equivalent
- Put *V*
  - A-B are not compatible
  - => Each square that has A-B is incompatible too

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

B	*V*					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B *V*	A=B *V*	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

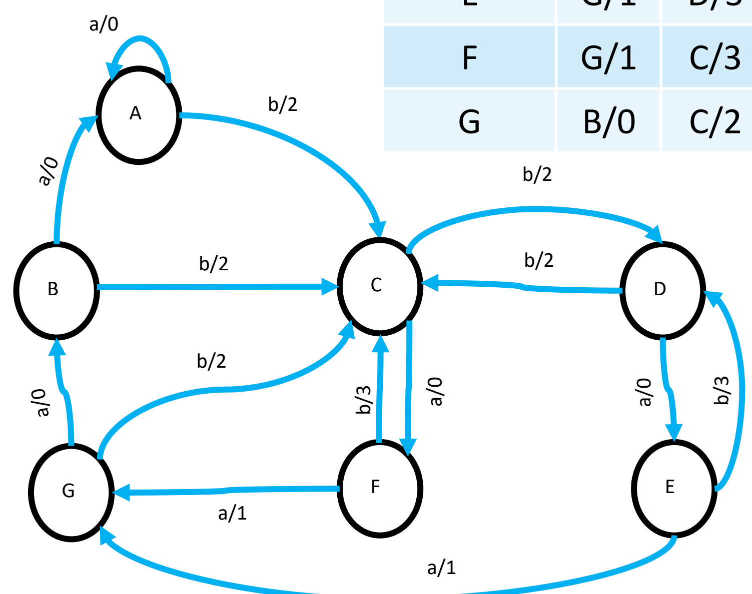


Implication Table: Step 6 (cont'd)

- Find squares with implied pairs that are equivalent
- Put *V*
  - C-D is equivalent if E=F be equivalent
  - E-F is equivalent if C-D be equivalent
  - => They are equal

B	*V*					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F			
E	X	X	X	X		
F	X	X	X	X	C=D	
G	A=B *V*	A=B *V*	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

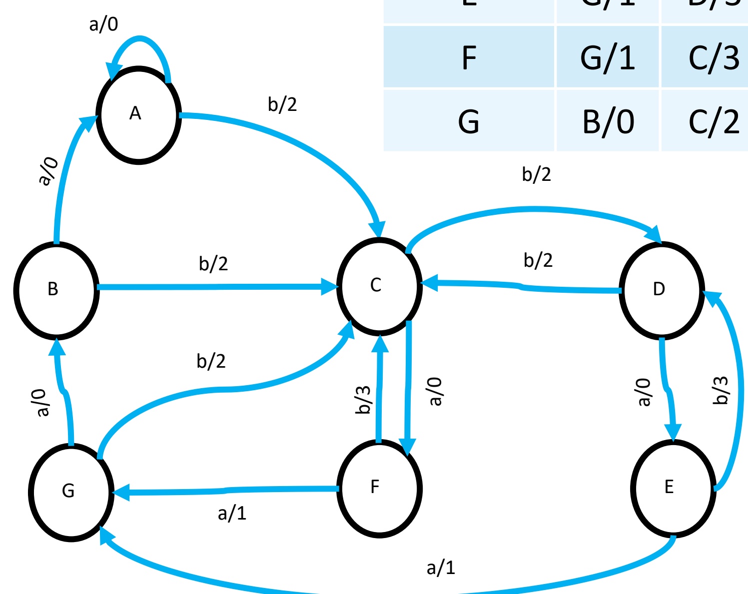


Implication Table: Step 6 (cont'd)

- Find squares with implied pairs that are equivalent
- Put *V*
  - C-D is equivalent if E=F be equivalent
  - E-F is equivalent if C-D be equivalent
  - => They are equal

B	*V*					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F *V*			
E	X	X	X	X		
F	X	X	X	X	C=D *V*	
G	A=B *V*	A=B *V*	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

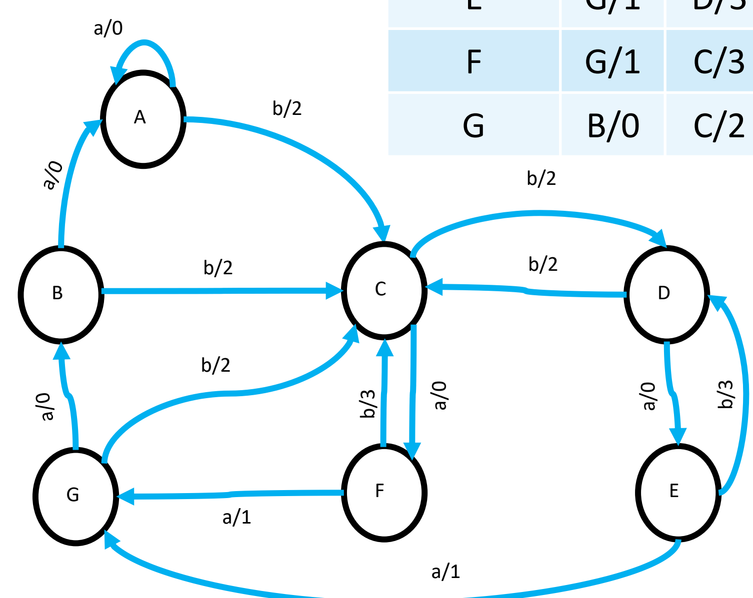


Implication Table: Step 7

- Reduce transition table
  - Remove equivalent states

B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F V			
E	X	X	X	X		
F	X	X	X	X	C=D V	
G	A=B V	A=B V	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

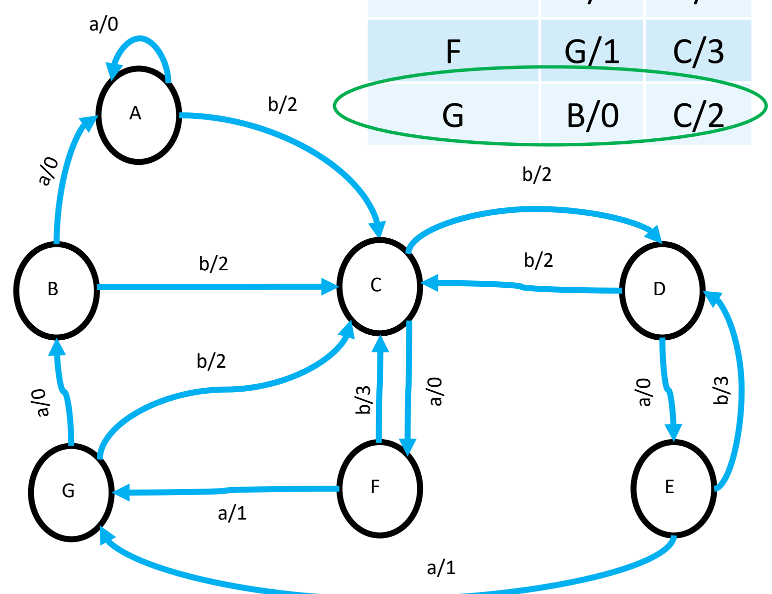


Implication Table: Step 7 (cont'd)

- Reduce transition table
  - Remove equivalent states

B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F V			
E	X	X	X	X		
F	X	X	X	X	C=D V	
G	A=B V	A=B V	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A	A/0	C/2
B	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3
G	B/0	C/2

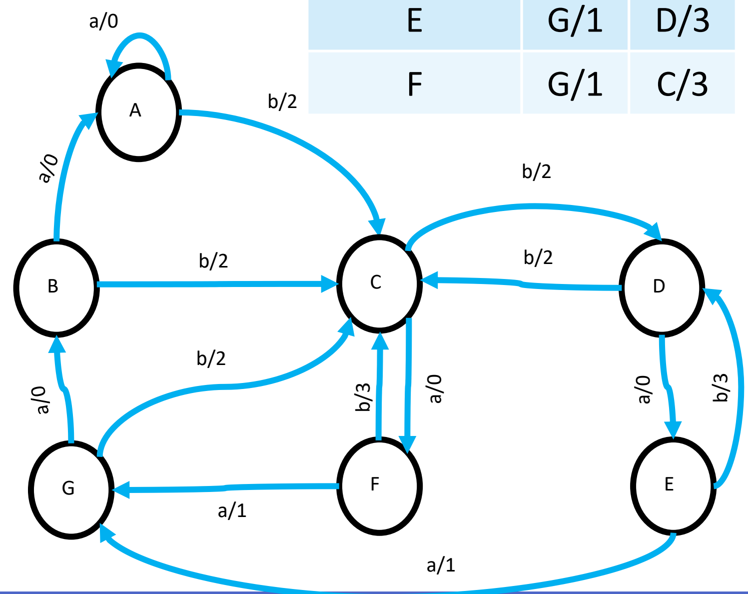


Implication Table: Step 7 (cont’d)

- Reduce transition table
  - Remove equivalent states

Present State	Next State a	Next State b
A,B,G	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3

B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F V			
E	X	X	X	X		
F	X	X	X	X	C=D V	
G	A=B V	A=B V	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

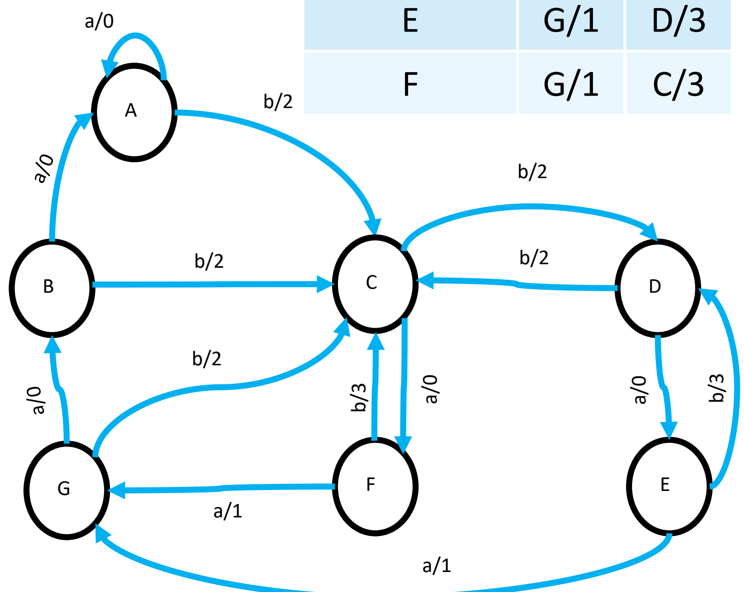


Implication Table: Step 7 (cont'd)

- Reduce transition table
  - Remove equivalent states

Present State	Next State	
	a	b
A,B,G	A/0	C/2
C	F/0	D/2
D	E/0	C/2
E	G/1	D/3
F	G/1	C/3

B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F V			
E	X	X	X	X		
F	X	X	X	X	C=D V	
G	A=B V	A=B V	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

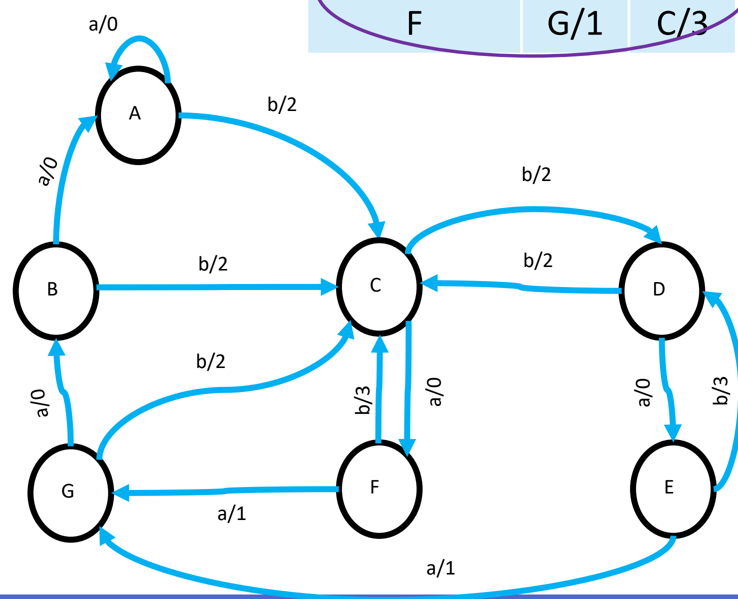


Implication Table: Step 7 (cont'd)

- Reduce transition table
  - Remove equivalent states

B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F V			
E	X	X	X	X		
F	X	X	X	X	C=D V	
G	A=B V	A=B V	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F

Present State	Next State	
	a	b
A,B,G	A/0	C/2
C, D	F/0	D/2
E	G/1	D/3
F	G/1	C/3

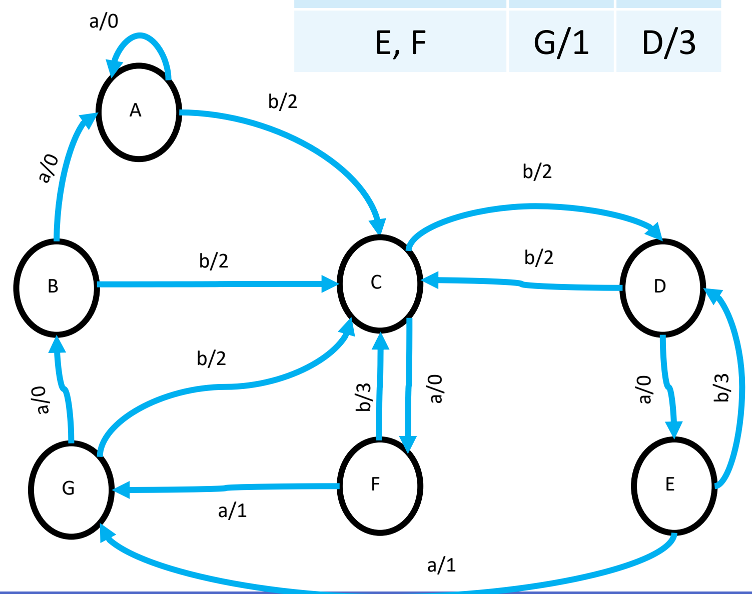


Implication Table: Step 7 (cont'd)

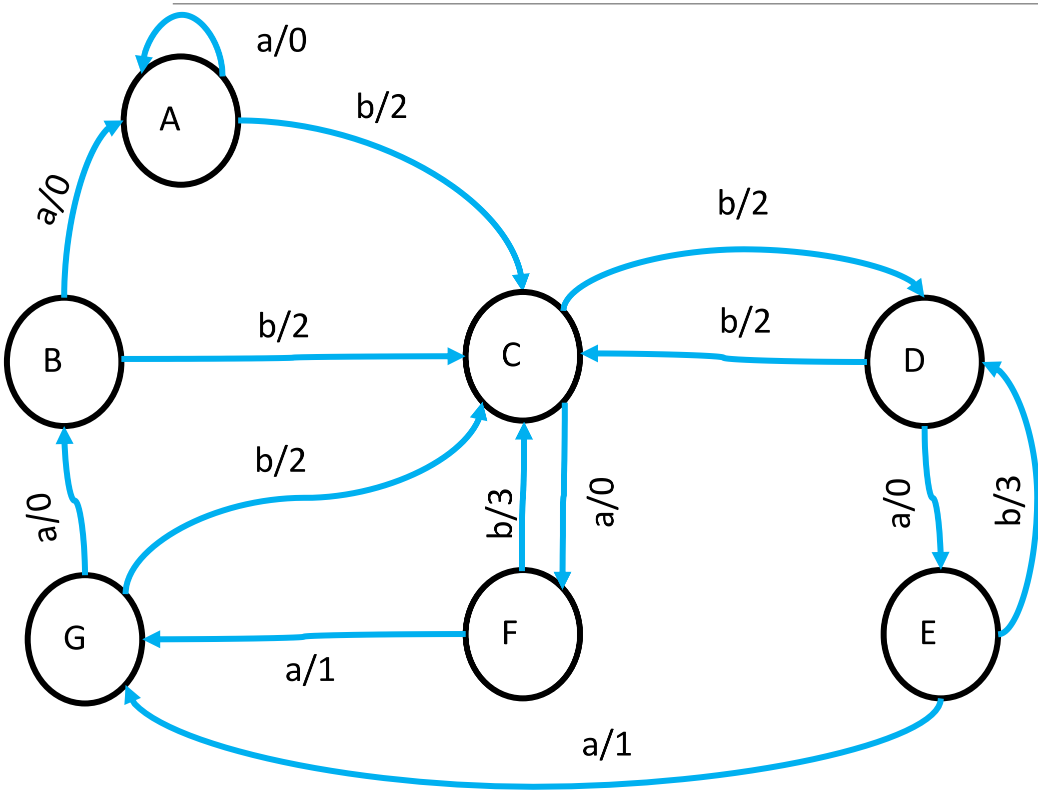
- Reduce transition table
  - Remove equivalent states

Present State	Next State a	Next State b
A, B, G	A/0	C/2
C, D	F/0	D/2
E, F	G/1	D/3

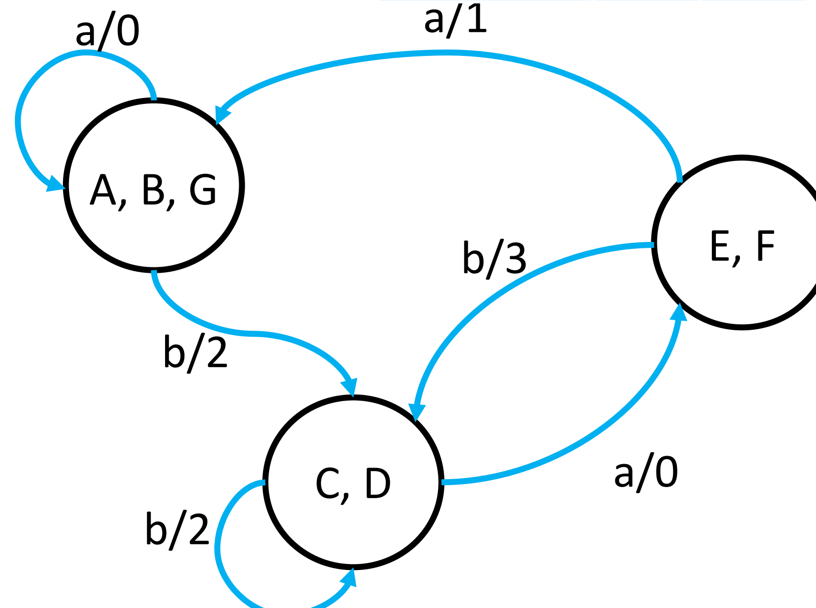
B	V					
C	~~A=F C=D~~	~~A=F C=D~~				
D	~~A=E~~	~~A=E~~	~~C=D~~ E=F V			
E	X	X	X	X		
F	X	X	X	X	C=D V	
G	A=B V	A=B V	~~C=D B=F~~	~~B=E~~	X	X
	A	B	C	D	E	F



Simplifies State Diagram



Present State	Next State	
	a	b
A, B, G	A/0	C/2
C, D	F/0	D/2
E, F	G/1	D/3



Thank You