Sequential Circuit Design

Sequential Logic Design

Contents

- Why sequential logic?
- Flip-flop criteria table
- Sequential circuit analysis
- Sequential circuit design

Why Sequential Logic?

- Sequential circuit has **additional dimension** which is **time**
- Combinational logic only depends on current input
- Sequential circuit output depends on previous input other than current input
- **More powerful** than combination logic
- Able to model condition which can't be modeled by combinational logic

Sequential Circuit Analysis

- Given sequential circuit diagram, behavioral analysis from **state table** and also **state diagram**
- Need state equations to get **flip-flop input** and **output functions** for circuit output other than flip-flop (if any)
- We use A(t) and A(t+1) to represent current condition and the next condition for flip-flop represented by A.
- Other method, we can use A and A^+ to represent current condition and the following condition

Types of tables in sequential circuit

- Characteristic table
- Criteria Table
- State Table
- Excitation table

Q(t)	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	indeterminate

J	K	Q(t+1)	Comments
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q(t)'	Toggle

Present State		Input	Next State		Output
A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

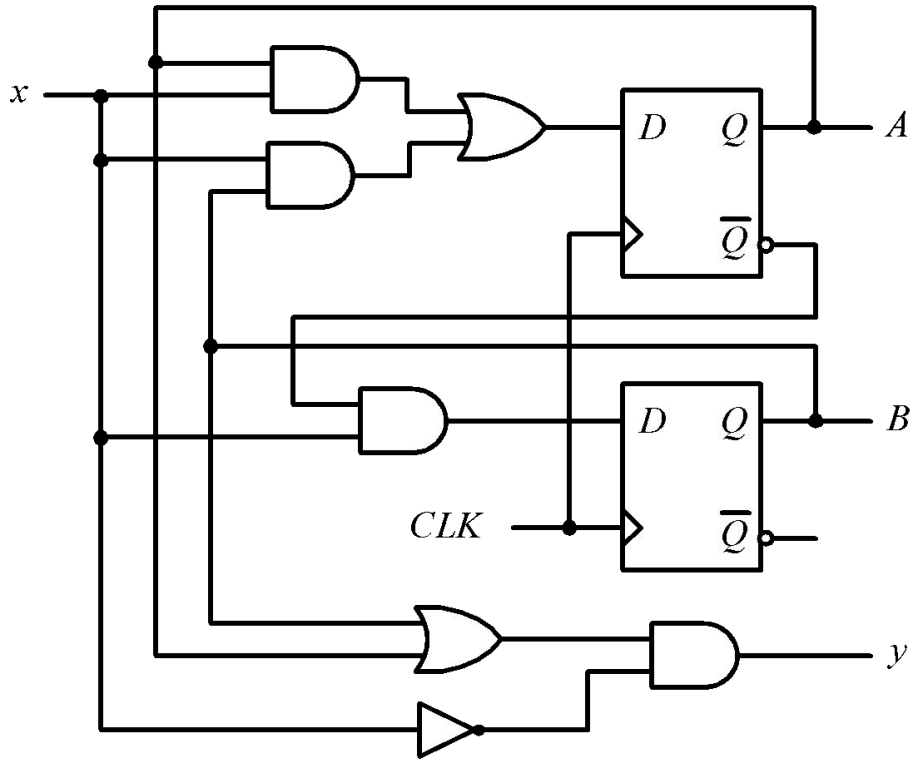
Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Flip-flop Excitation Tables: Terminology (1)

- **Analysis**: Start from circuit diagram**, build state table** or state diagram
- **Design**: Start from specification set (i.e. in state equation form, state table or state diagram) **build logic circuit**.
- Criteria table is used in analysis
- Excitation tables is used in design

Sequential Circuit Analysis: Equations

Example 1: Find the input equation for flip-flops (using D flip-flop) and output equation



Sequential Circuit Analysis: Equations

- Example 1 (using D flip-flop)

Flip-flop input

External input

State equation

Next state

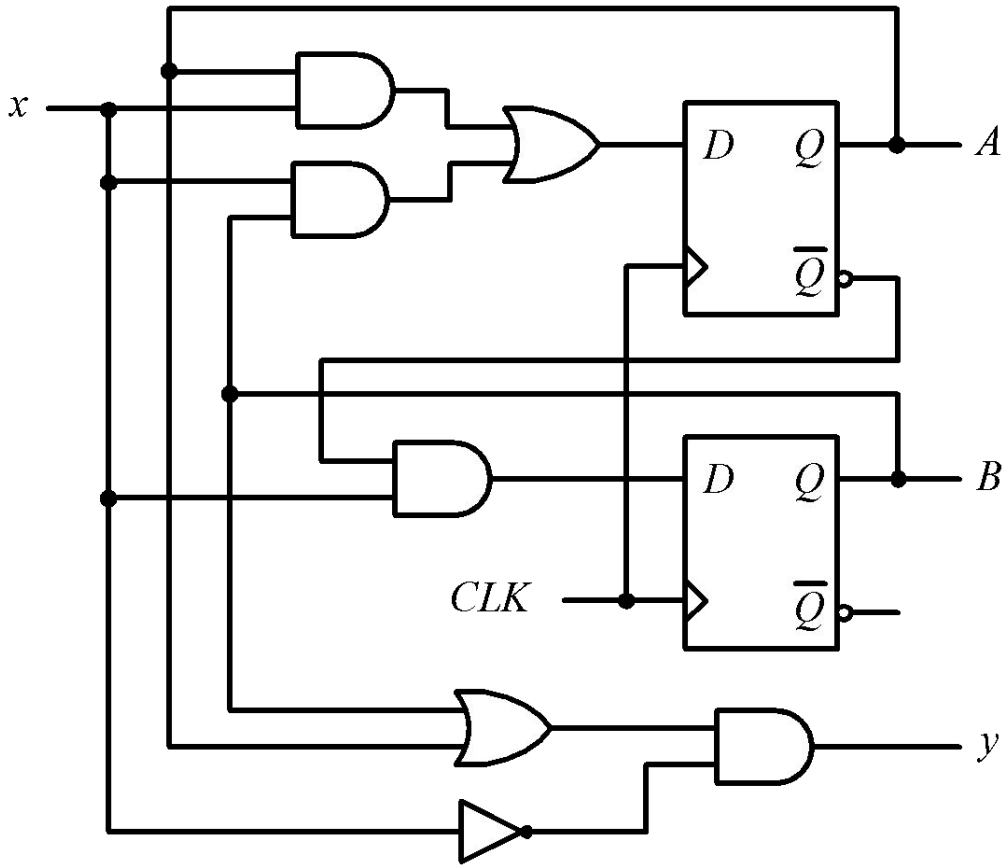
$DA = A.x + B.x$

$DB = A'.x$

Present state

Output Function

$y = (A + B).x'$



Note:
A+ or A(t+1): both refer to next state of fipflop

Sequential Circuit Analysis: State Table

- From the state equations and output function, we can derive state table which contains all combined binary available for current condition and input
- **State table**
  - The same as Truth Table
  - Input and condition pad on the left
  - Output and next condition on the right
  - combined binary available for current condition and input
- **M flip-flop and n input => 2^{m+n} line**

Sequential Circuit Analysis

State equation

$A^+ = A.x + B.x$

$B^+ = A'.x$

Output function

$y = (A + B).x'$

State table for circuit in Example 1

Present State		Input	Next State		Output	FF Inputs	
						DA	DB
A	B	x	A^+	B^+	y		
0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	1
0	1	0	0	0	1	0	0
0	1	1	1	1	0	1	1
1	0	0	0	0	1	0	0
1	0	1	1	0	0	1	0
1	1	0	0	0	1	0	0
1	1	1	1	0	0	1	0

Sequential Circuit Analysis

Present State		Input	Next State		Output
A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Other method

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
AB	A^+B^+	A^+B^+	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

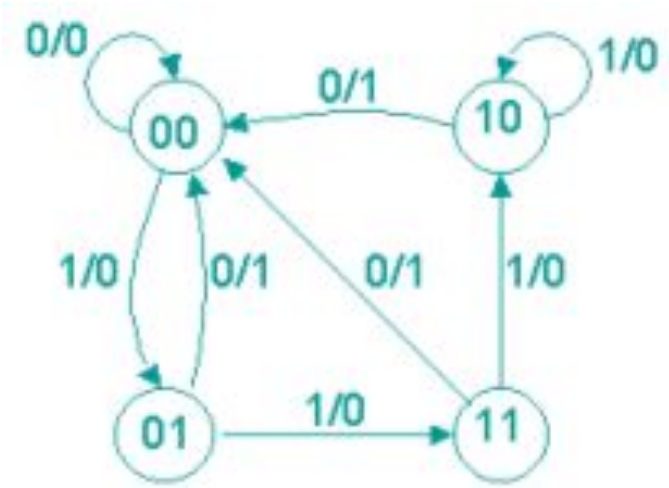
Sequential Circuit Analysis

- From the truth table, we can draw state diagram
- State diagram
  - Each state is represented by circle
  - Each arrow (between two circle) represent transfer for sequential logic (i.e. line transition in truth table)
  - a/b label for each arrow where a represent inputs and b represent output for circuit in transition
- Each flip-flop value combination represent state. Therefore, **m flip-flop**=> until **2^m state**.

Sequential Circuit Analysis: State Diagram

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
AB	A^+B^+	A^+B^+	y	y
00	00	01	0	0
01	00	11	1	0
10	00	10	1	0
11	00	10	1	0

State diagram for circuit in example 1



Flip-flop Input Function

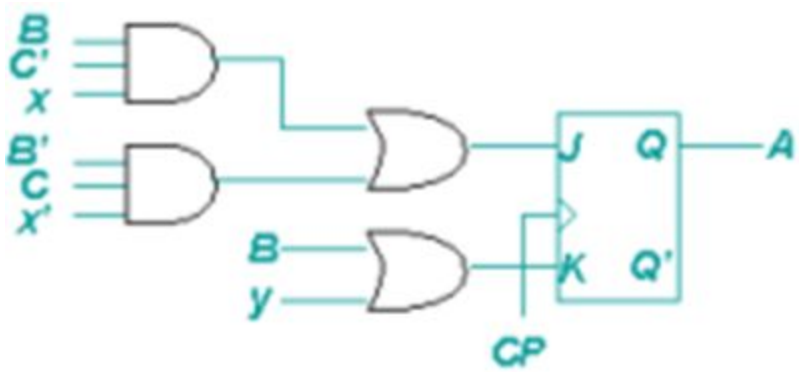
- Output of sequential circuit is function for current condition for flip-flop and input. This is explained using algebra by circuit output function
  - In example 1: y= (A+B)x'
- Circuit part that generate input to flip-flop is explained using algebra by flip-flop input functions

Flip-flop Input Function

- Flip-flop input function determine next condition
- From flip-flop input function and criteria table for flip-flop, we get next condition of the flip-flop

Flip-flop Input Function

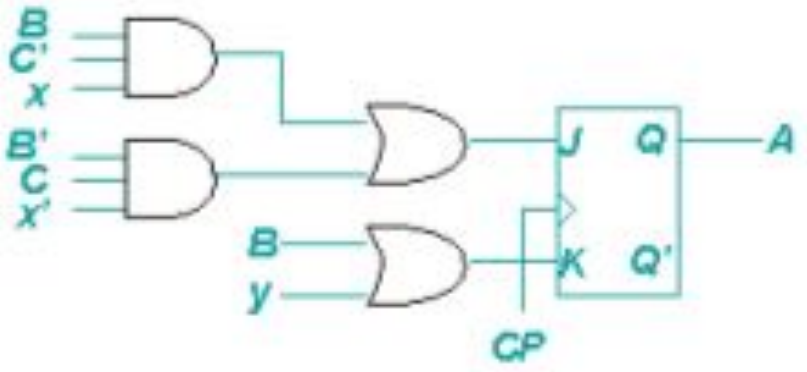
- Example 2: Find flip-flop input equation in circuit with JK flip flop



Flip-flop Input Function

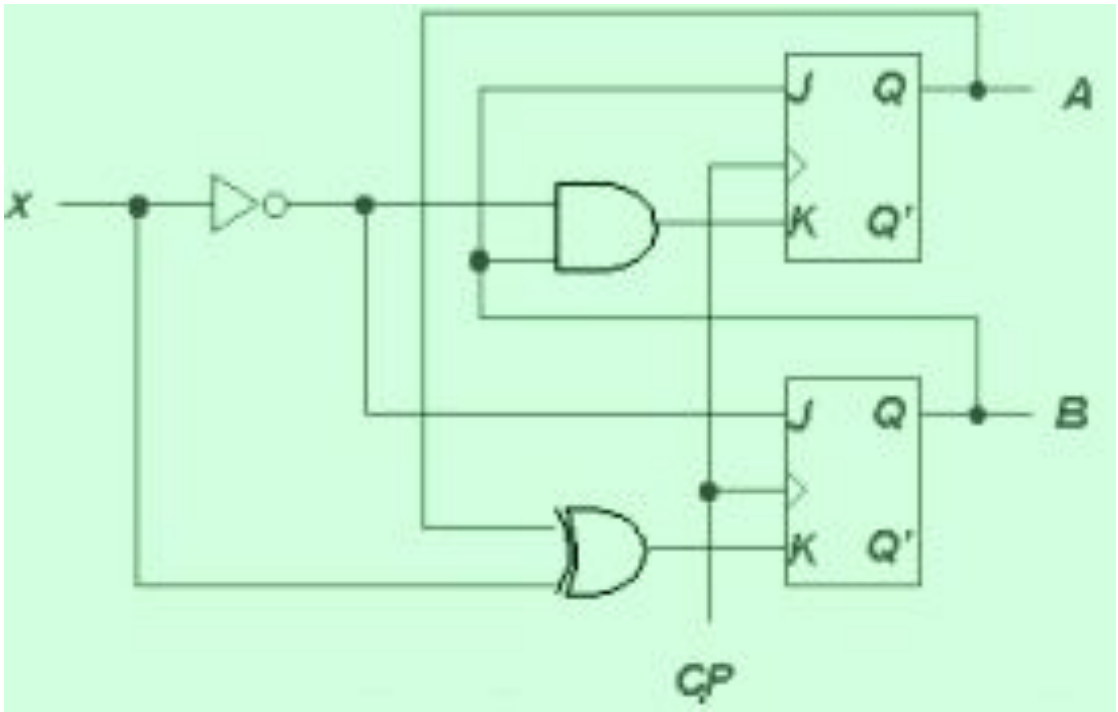
- Example 2: Find flip-flop input equation in circuit with JK flip flop
- We **use 2 character** to represent **flip-flop input**: first character represent flip-flop input (J or K for JK flip-flop, S or R for SR flip-flop, D for D flip-flop, T for T flip-flop) and second character represent name of the flip-flop

$$JA = B.C'.x + B'.C.x'$$
$$KA = B + y$$



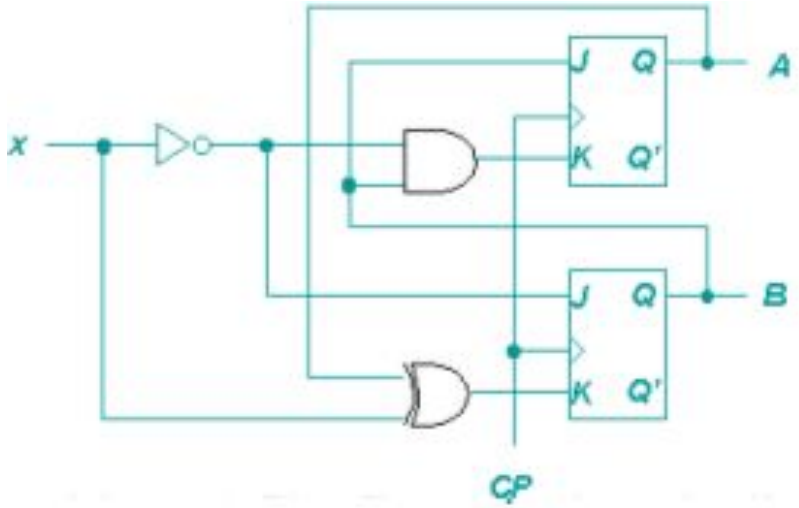
Analysis: Example 3

- Given sequential circuit with two JK flip-flop, A and B and one input x



Analysis: Example 3

- Given sequential circuit with two JK flip-flop, A and B and one input x



- **Get the input flip-flop function from the circuit**

$JA = B$ $\quad$ $JB = x'$

$KA = B.x'$ $\quad$ $KB = A'.x + A.x' = A \oplus x$

Analysis: Example 3

- Input flip-flop function

$JA = B$ $\qquad$ $JB = x'$

$KA = B.x'$ $\qquad$ $KB = A'.x + A.x' = A \oplus x$

- **Fill the state table with the above function using criteria table for used flip-flop**

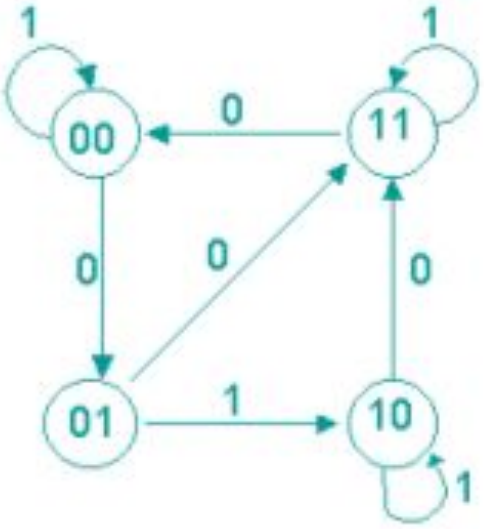
J	K	Q(t+1)	Comments
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q(t)'	Toggle

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0

Analysis: Example 3

- **Draw state diagram from the state table**

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	1
1	0	1	1	0	0	0	0	0
1	1	0	0	0	1	1	1	1
1	1	1	1	1	1	0	0	0



Flip-flop Excitation Tables: Terminology (2)

- **Analysis**: Start from circuit diagram, **build state table** or state diagram
- **Design**: Start from specification set (i.e. in state equation form, state table or state diagram) **build logic circuit**.
- Criteria table is used in analysis
- Excitation tables is used in design

Flip-flop Excitation Tables

- Excitation tables : it give transition characteristic between current condition and next condition to determine flip-flop input

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop

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

SR Flip-flop

Q	Q^+	D
0	0	0
0	1	1
1	0	0
1	1	1

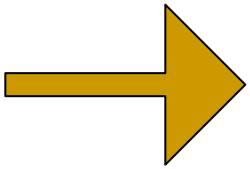
D Flip-flop

Q	Q^+	T
0	0	0
0	1	1
1	0	1
1	1	0

T Flip-flop

How to build excitation table: Example: JK Flip-Flop

Q	Q+	Actually what happens		Final Combined Result	
		J	K	J	K
0	0	0	1	0	x
		0	0		
0	1	1	0	1	x
		1	1		
1	0	0	1	x	1
		1	1		
1	1	0	0	x	0
		1	0		



Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

JK Flip-flop

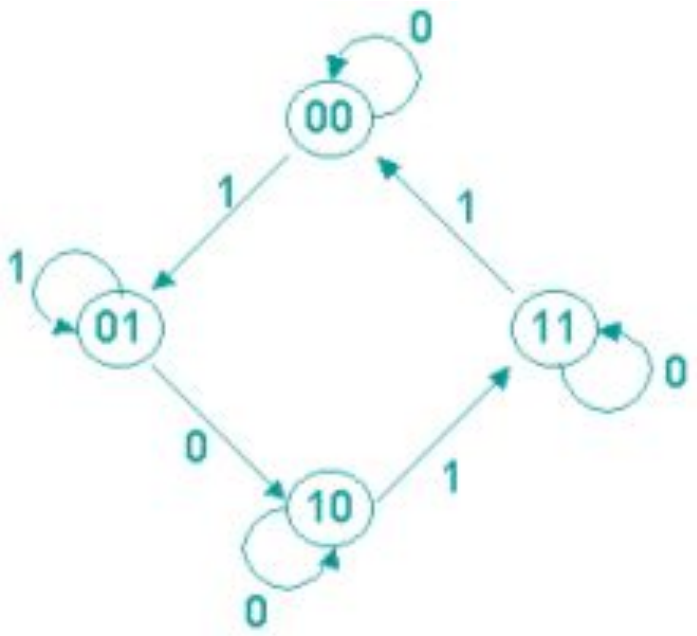
Designing Sequential Circuit

Design steps

- Step 1: Start with circuit specification – characteristic of circuit
- Step 2: Build state table
- (Ignore) Do state reduction if needed (not in syllabus)
- (Ignore) Do state assignment (not in syllabus)
- Step 3: Determine number of flip-flop which will be used
- Step 4: Build circuit excitation and output table from state table
- Step 5: Build circuit output function and flip-flop input function
- Step 6: Draw logic diagram

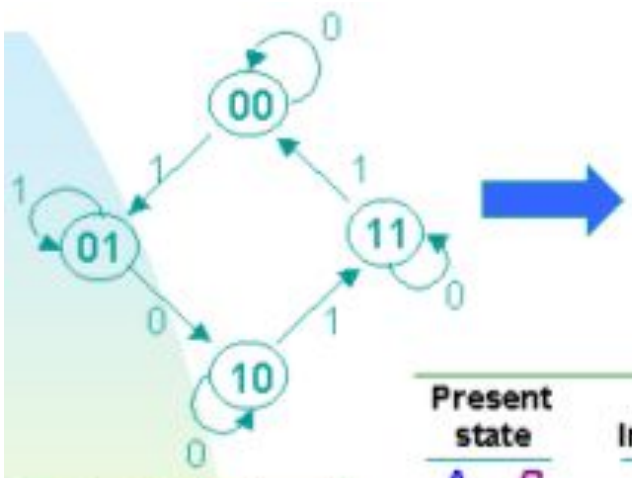
Design: Example 1

- Given state diagram as follows, get the sequential circuit using JK flip-flop



Design: Example 1

- State/excitation table using JK flip-flop



Present State	Next State x=0	Next State x=1
AB	A^+B^+	A^+B^+
00	00	01
01	10	01
10	10	11
11	11	00

Q	Q^+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

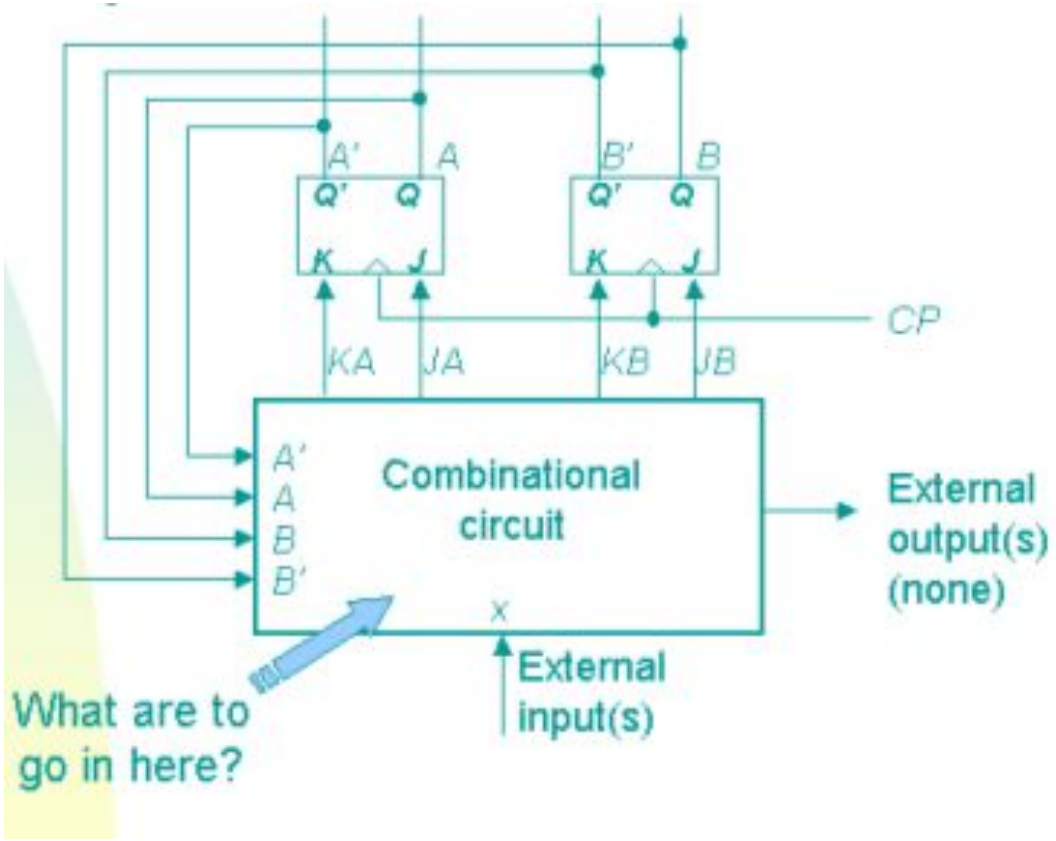
JK Flip-flop's excitation table.

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1

Note:
Step 1: Build state table
Step 2: Find no. of Flip-Flop *(i.e. 2^m states means m flip-flips)*
Step 3: Build Excitation table
Step 4: Use K-map to find input functions
Step 5: Design logic diagram

Design: Example 1

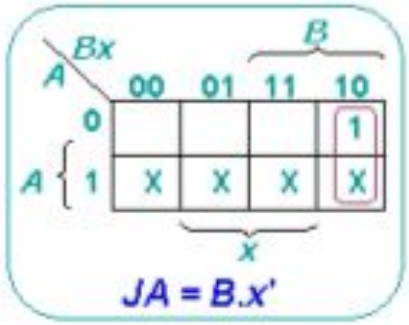
- Block diagram

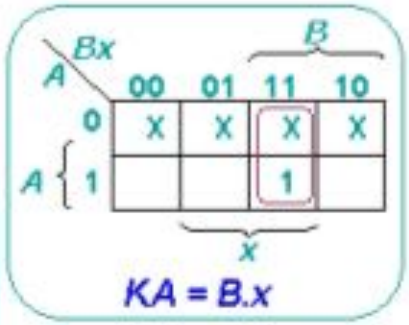


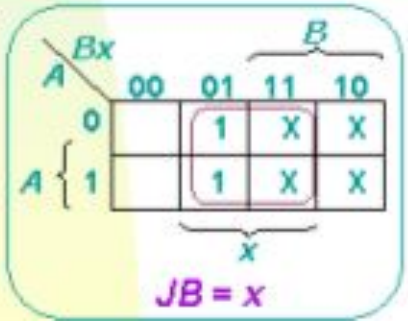
Design: Example 1

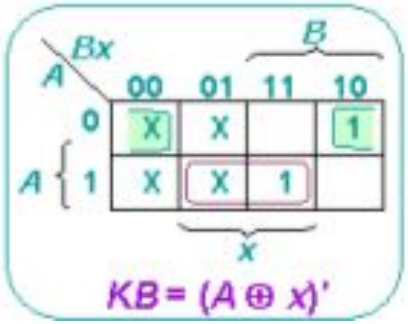
- From state table, get input flip-flop function

Present state		Input	Next state		Flip-flop inputs			
A	B	x	A^+	B^+	JA	KA	JB	KB
0	0	0	0	0	0	X	0	X
0	0	1	0	1	0	X	1	X
0	1	0	1	0	1	X	X	1
0	1	1	0	1	0	X	X	0
1	0	0	1	0	X	0	0	X
1	0	1	1	1	X	0	1	X
1	1	0	1	1	X	0	X	0
1	1	1	0	0	X	1	X	1









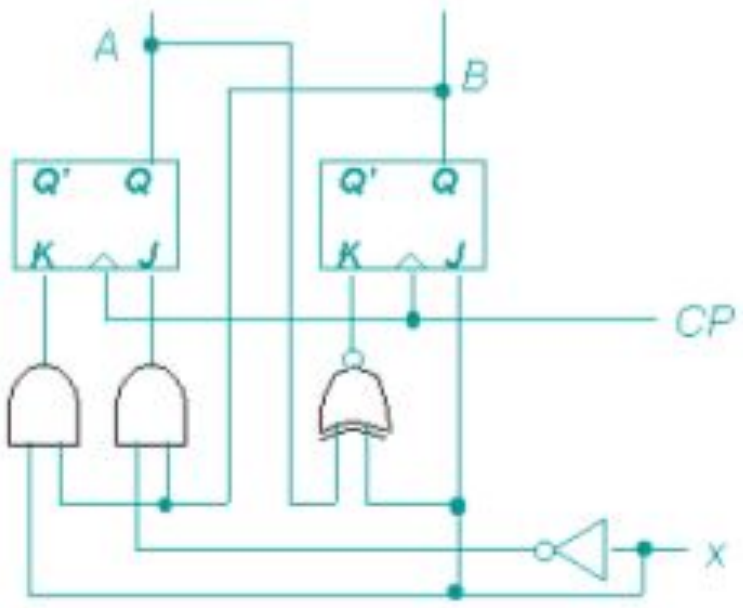
Design: Example 1

- Input flip-flop function

$JA = B.x'$ $\qquad$ $JB = x$

$KA = B.x$ $\qquad$ $KB = (A \oplus x)'$

- Logic Diagram



Design: Example 2

- Design, using D flip-flop, circuit is based on state table below. (Exercise: How if using JK flip-flop)

Present state		Input	Next state		Output
A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

Design: Example 2

- Determine input expression for flip-flop and y output

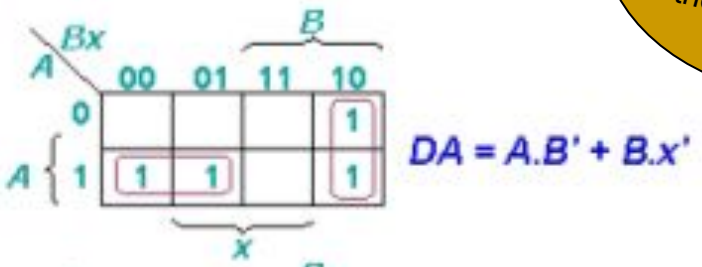
Note: D-FF is too trivial for including it in the excitation table

Present state		Input	Next state		Output
A	B	x	A^+	B^+	y
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	0
1	1	1	0	0	0

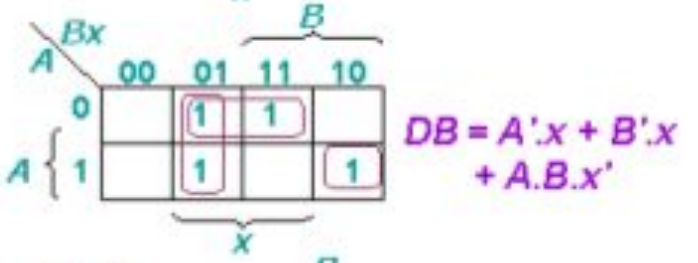
$DA(A, B, x) = \Sigma\, m(2,4,5,6)$

$DB(A, B, x) = \Sigma\, m(1,3,5,6)$

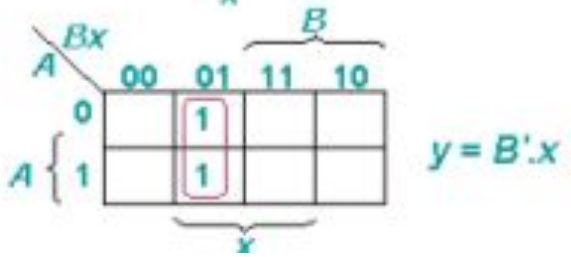
$y(A, B, x) = \Sigma\, m(1,5)$



$DA = A.B' + B.x'$



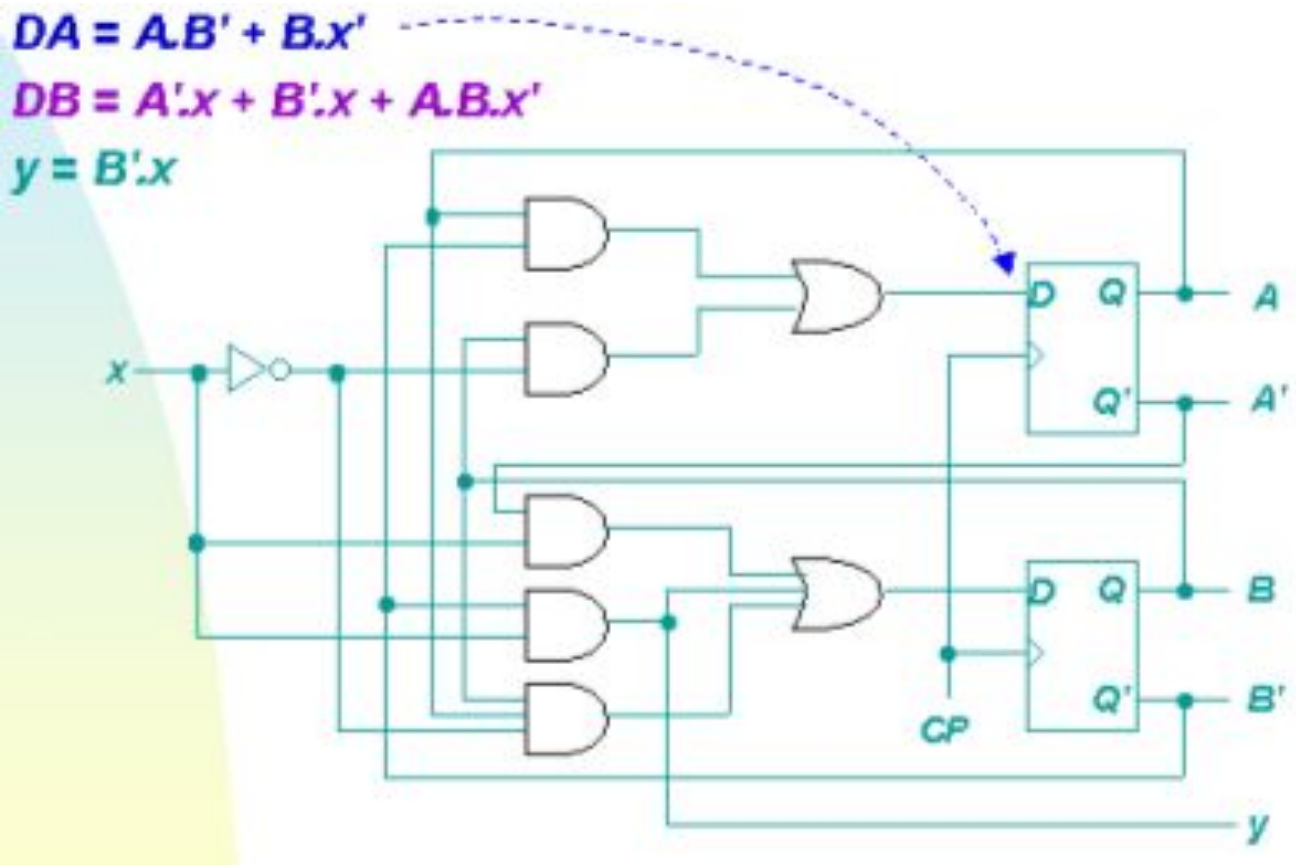
$DB = A'.x + B'.x + A.B.x'$



$y = B'.x$

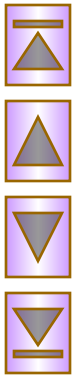
Design: Example 2

- From expression built, draw logic diagram



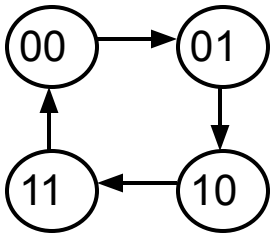
Introduction: Counters

- Counters are circuits that cycle through a specified number of states.
- Two types of counters:
  - synchronous (parallel) counters
  - asynchronous (ripple) counters
- Ripple counters allow some flip-flop outputs to be used as a source of clock for other flip-flops.
- Synchronous counters apply the same clock to all flip-flops.



Synchronous (Parallel) Counters

- Synchronous (parallel) counters: the flip-flops are clocked at the same time by a common clock pulse.
- We can design these counters using the sequential logic design process.
- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).



Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

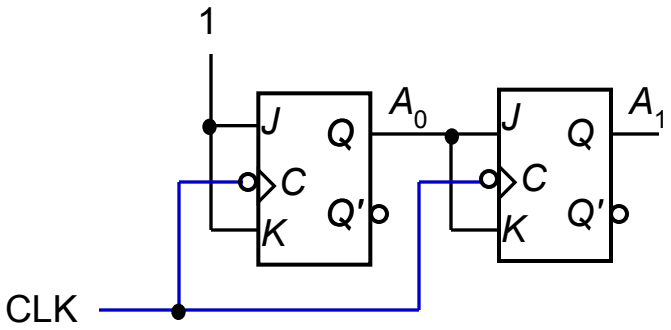
Synchronous (Parallel) Counters

- Example: 2-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J,K inputs).

Present state		Next state		Flip-flop inputs	
A_1	A_0	A_1^+	A_0^+	TA_1	TA_0
0	0	0	1	0	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	0	0	1	1

$TA_1 = A_0$

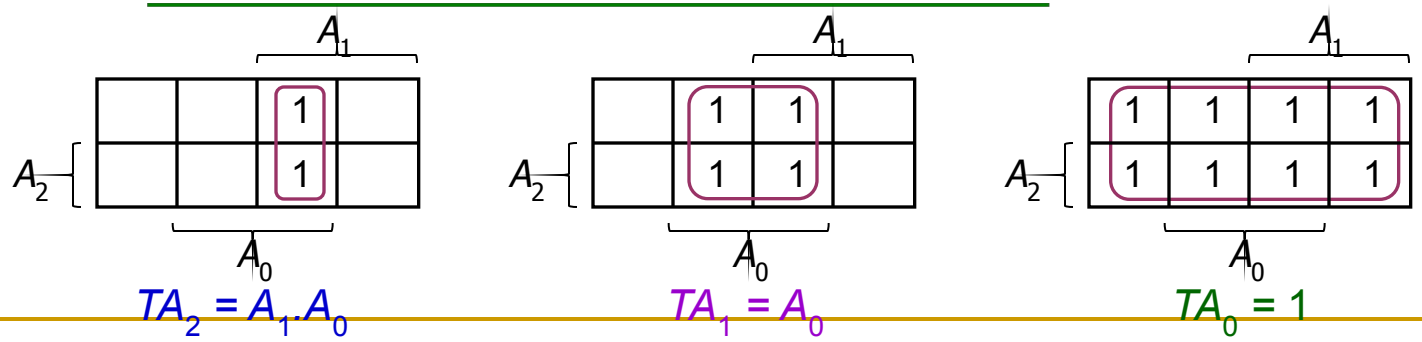
$TA_0 = 1$



Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (using T flip-flops, or JK flip-flops with identical J, K inputs).

Present state			Next state			Flip-flop inputs		
A_2	A_1	A_0	A_2^+	A_1^+	A_0^+	TA_2	TA_1	TA_0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	0	1
0	1	1	1	0	0	1	1	1
1	0	0	1	0	1	0	0	1
1	0	1	1	1	0	0	1	1
1	1	0	1	1	1	0	0	1
1	1	1	0	0	0	1	1	1

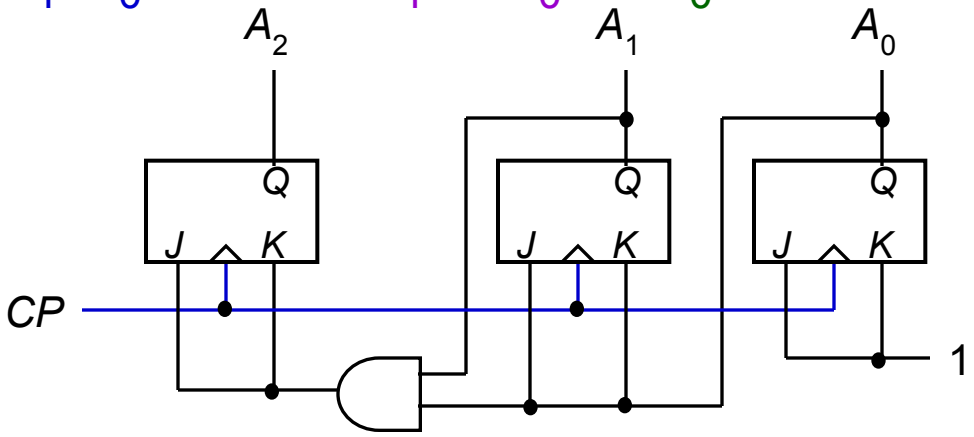


$TA_2 = A_1.A_0$ $TA_1 = A_0$ $TA_0 = 1$

Synchronous (Parallel) Counters

- Example: 3-bit synchronous binary counter (cont'd).

$TA_2 = A_1.A_0$ $TA_1 = A_0$ $TA_0 = 1$



Synchronous (Parallel) Counters

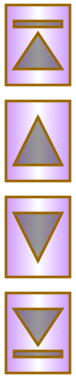
- Note that in a binary counter, the n^{th} bit (shown underlined) is always complemented whenever

  $\underline{0}11\ldots11 \rightarrow \underline{1}00\ldots00$

  or $\underline{1}11\ldots11 \rightarrow \underline{0}00\ldots00$

- Hence, X_n is complemented whenever $X_{n-1}X_{n-2} \ldots X_1X_0 = 11\ldots11$.
- As a result, if T flip-flops are used, then

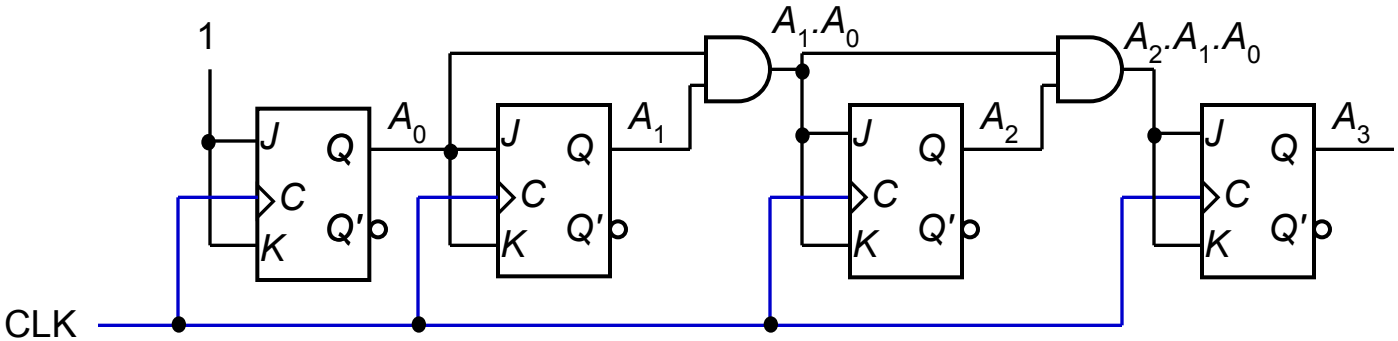
  $$TX_n = X_{n-1} \cdot X_{n-2} \cdot \ldots \cdot X_1 \cdot X_0$$



Synchronous (Parallel) Counters

- Example: 4-bit synchronous binary counter.

$$TA_3 = A_2 \cdot A_1 \cdot A_0$$
$$TA_2 = A_1 \cdot A_0$$
$$TA_1 = A_0$$
$$TA_0 = 1$$



Synchronous (Parallel) Counters

SELF STUDY

- Example: Synchronous decade/BCD counter.

Clock pulse	Q_3	Q_2	Q_1	Q_0
Initially	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10 (recycle)	0	0	0	0

$T_0 = 1$

$T_1 = Q_3'.Q_0$

$T_2 = Q_1.Q_0$

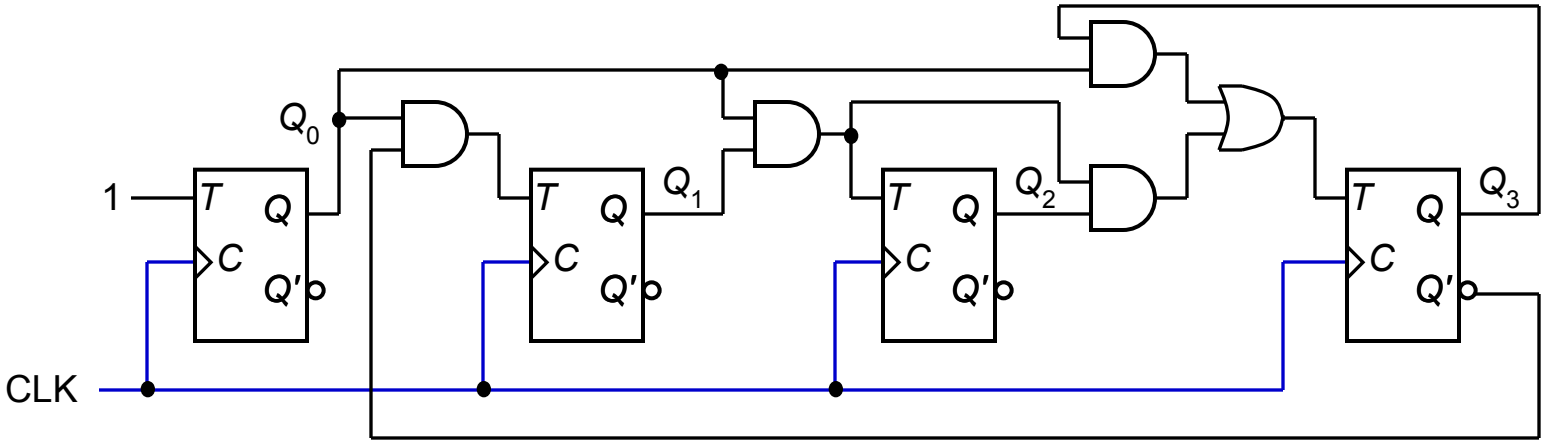
$T_3 = Q_2.Q_1.Q_0 + Q_3.Q_0$

https://www.youtube.com/watch?v=fKVZpupyP_o&list=PLBInK6fEyqRjMH3mWf6kwqiTbT798eAOm&index=186

Synchronous (Parallel) Counters

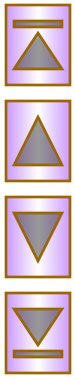
- Example: Synchronous decade/BCD counter (cont'd).

$T_0 = 1$
$T_1 = Q_3'.Q_0$
$T_2 = Q_1.Q_0$
$T_3 = Q_2.Q_1.Q_0 + Q_3.Q_0$



Up/Down Synchronous Counters

- Up/down synchronous counter: a *bidirectional* counter that is capable of counting either up or down.
- An input (control) line *Up*/*Down* (or simply *Up*) specifies the direction of counting.
  - *Up*/$\overline{Down}$ = 1 → Count upward
  - *Up*/$\overline{Down}$ = 0 → Count downward



https://www.youtube.com/watch?v=svFUEJkoeVY&list=PLBlnK6fEyqRjMH3mWf6kwqiTbT798eAOm&index=189

Up/Down Synchronous Counters

- Example: A 3-bit up/down synchronous binary counter.

Clock pulse	*Up*	Q_2	Q_1	Q_0	*Down*
0		0	0	0	
1		0	0	1	
2		0	1	0	
3		0	1	1	
4		1	0	0	
5		1	0	1	
6		1	1	0	
7		1	1	1	

$TQ_0 = 1$

$TQ_1 = (Q_0.Up) + (Q_0'.Up')$

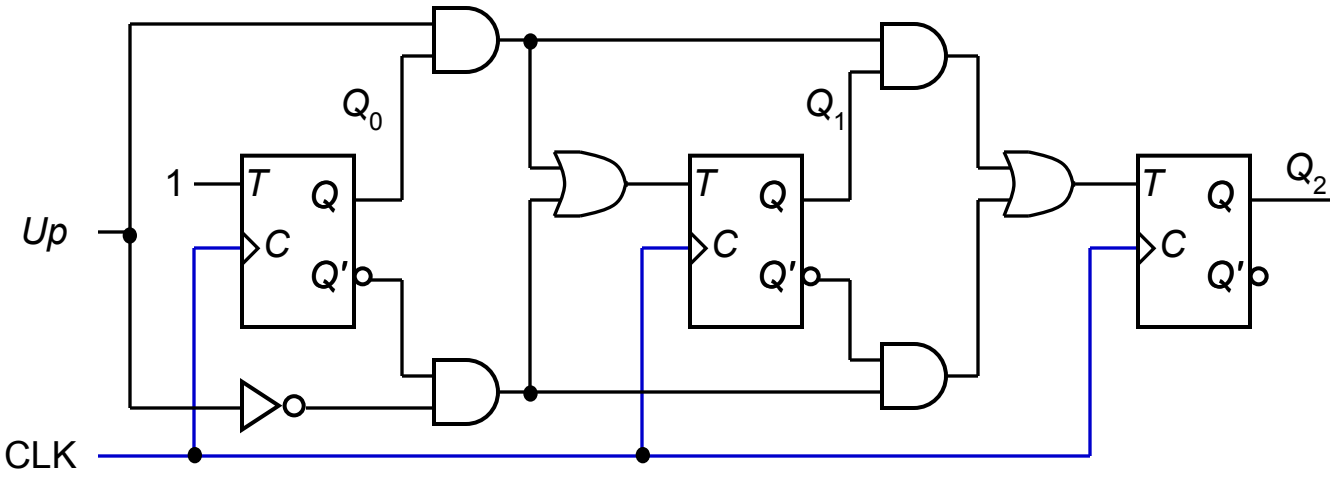
$TQ_2 = (Q_0.Q_1.Up) + (Q_0'.Q_1'.Up')$

Up counter	Down counter
$TQ_0 = 1$	$TQ_0 = 1$
$TQ_1 = Q_0$	$TQ_1 = Q_0'$
$TQ_2 = Q_0.Q_1$	$TQ_2 = Q_0'.Q_1'$

Up/Down Synchronous Counters

- Example: A 3-bit up/down synchronous binary counter (cont'd).

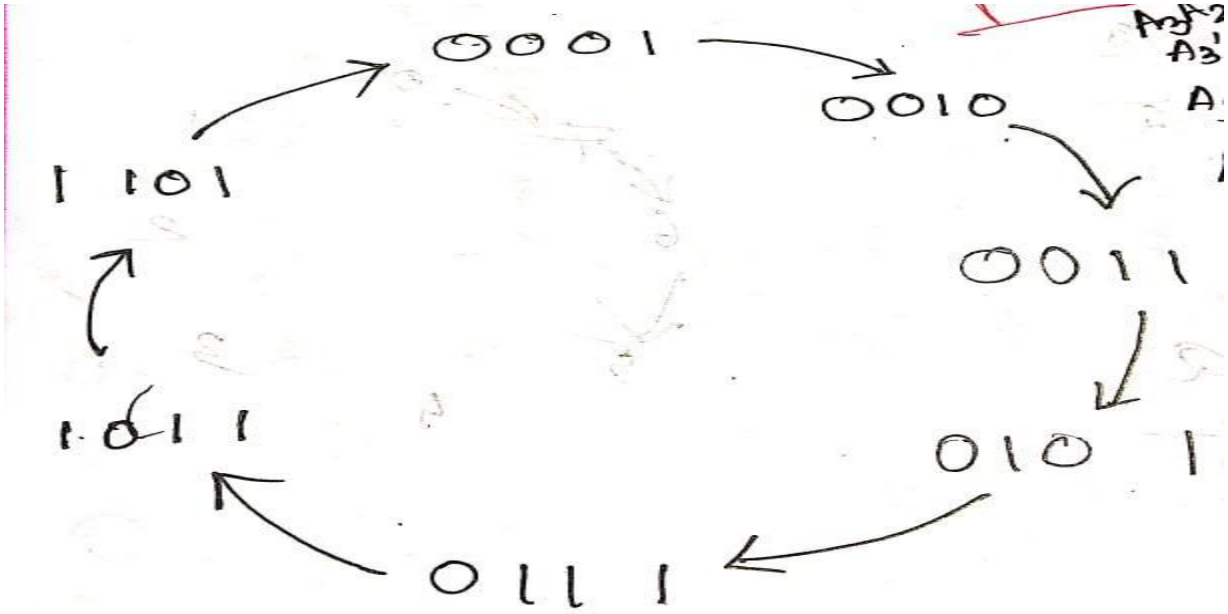
$$TQ_0 = 1$$
$$TQ_1 = (Q_0.Up) + (Q_0'.Up')$$
$$TQ_2 = (\ Q_0.Q_1.Up\ ) + (Q_0'.\ Q_1'.\ Up')$$



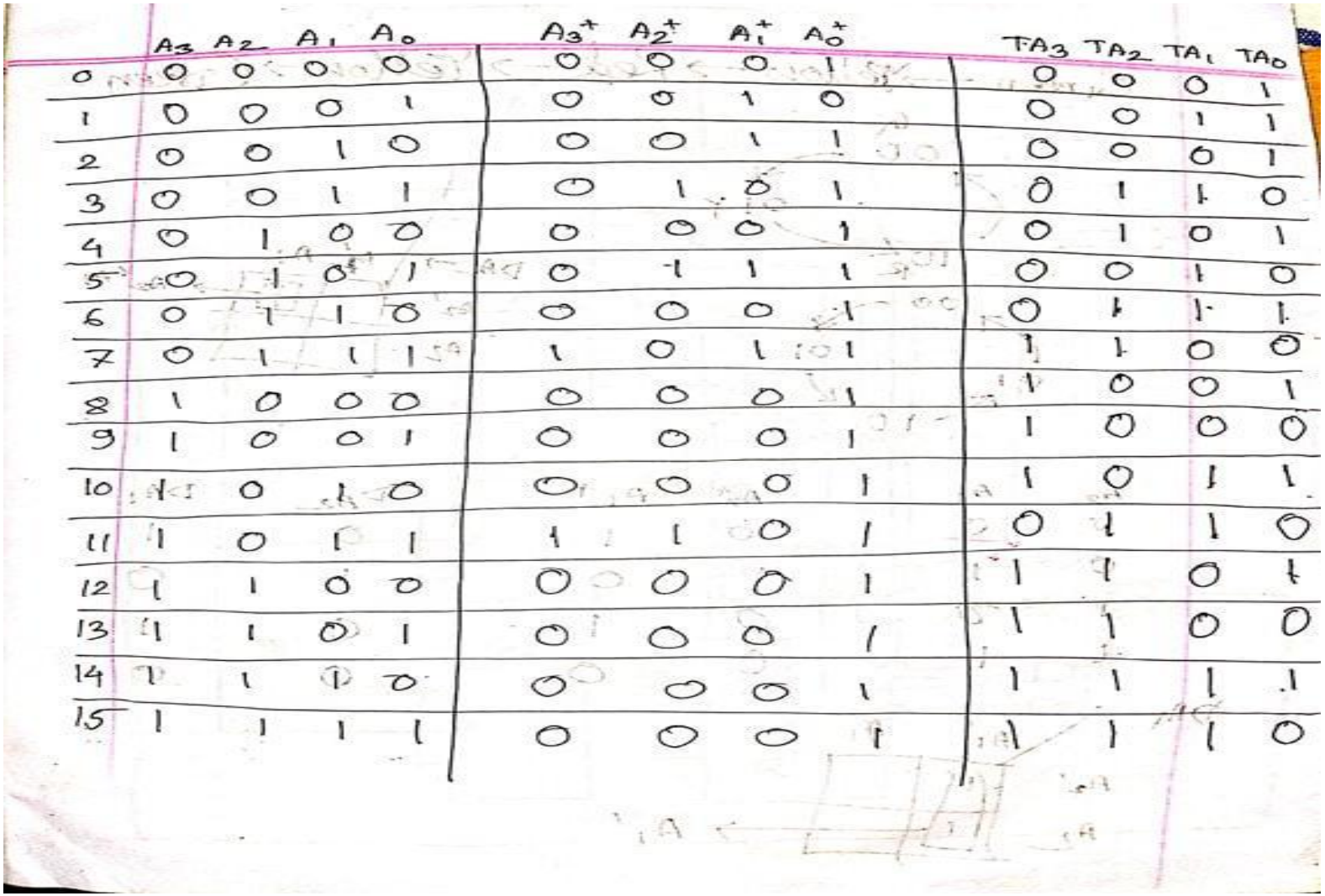
PRACTICE PROBLEM

- Implement the following counter using T FF

  1->2->3->5->7->11->13->1

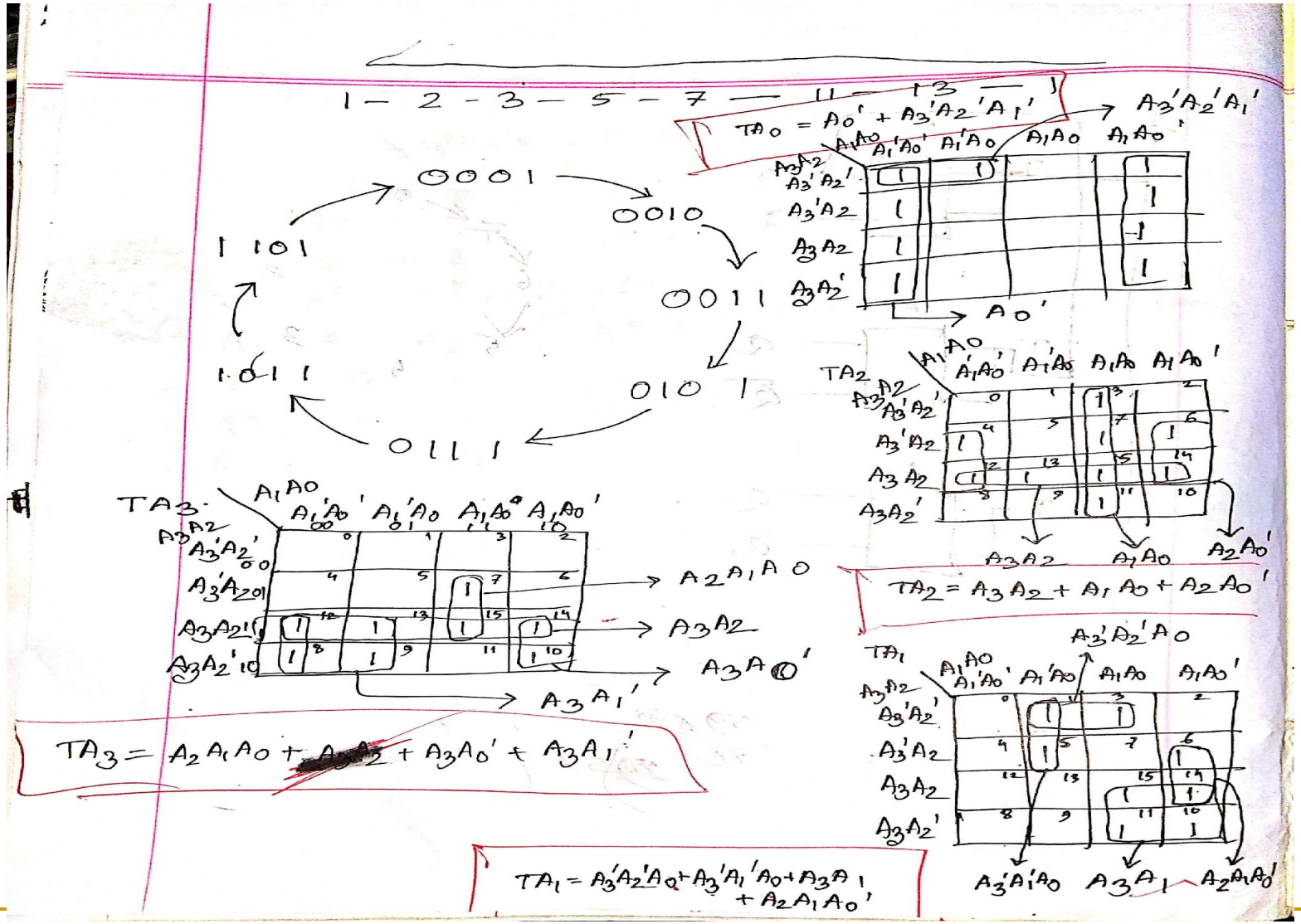


PRACTICE PROBLEM



	A_3	A_2	A_1	A_0	A_3^+	A_2^+	A_1^+	A_0^+	TA_3	TA_2	TA_1	TA_0
0	0	0	0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	0	0	1	0	0	0	1	1
2	0	0	1	0	0	0	1	1	0	0	0	1
3	0	0	1	1	0	1	0	1	0	1	1	0
4	0	1	0	0	0	0	0	1	0	1	0	1
5	0	1	0	1	0	1	1	1	0	0	1	0
6	0	1	1	0	0	0	0	1	0	1	1	1
7	0	1	1	1	1	0	1	1	1	1	0	0
8	1	0	0	0	0	0	0	1	1	0	0	1
9	1	0	0	1	0	0	0	1	1	0	0	0
10	1	0	1	0	0	0	0	1	1	0	1	1
11	1	0	1	1	1	1	0	1	0	1	1	0
12	1	1	0	0	0	0	0	1	1	1	0	1
13	1	1	0	1	0	0	0	1	1	1	0	0
14	1	1	1	0	0	0	0	1	1	1	1	1
15	1	1	1	1	0	0	0	1	1	1	1	0

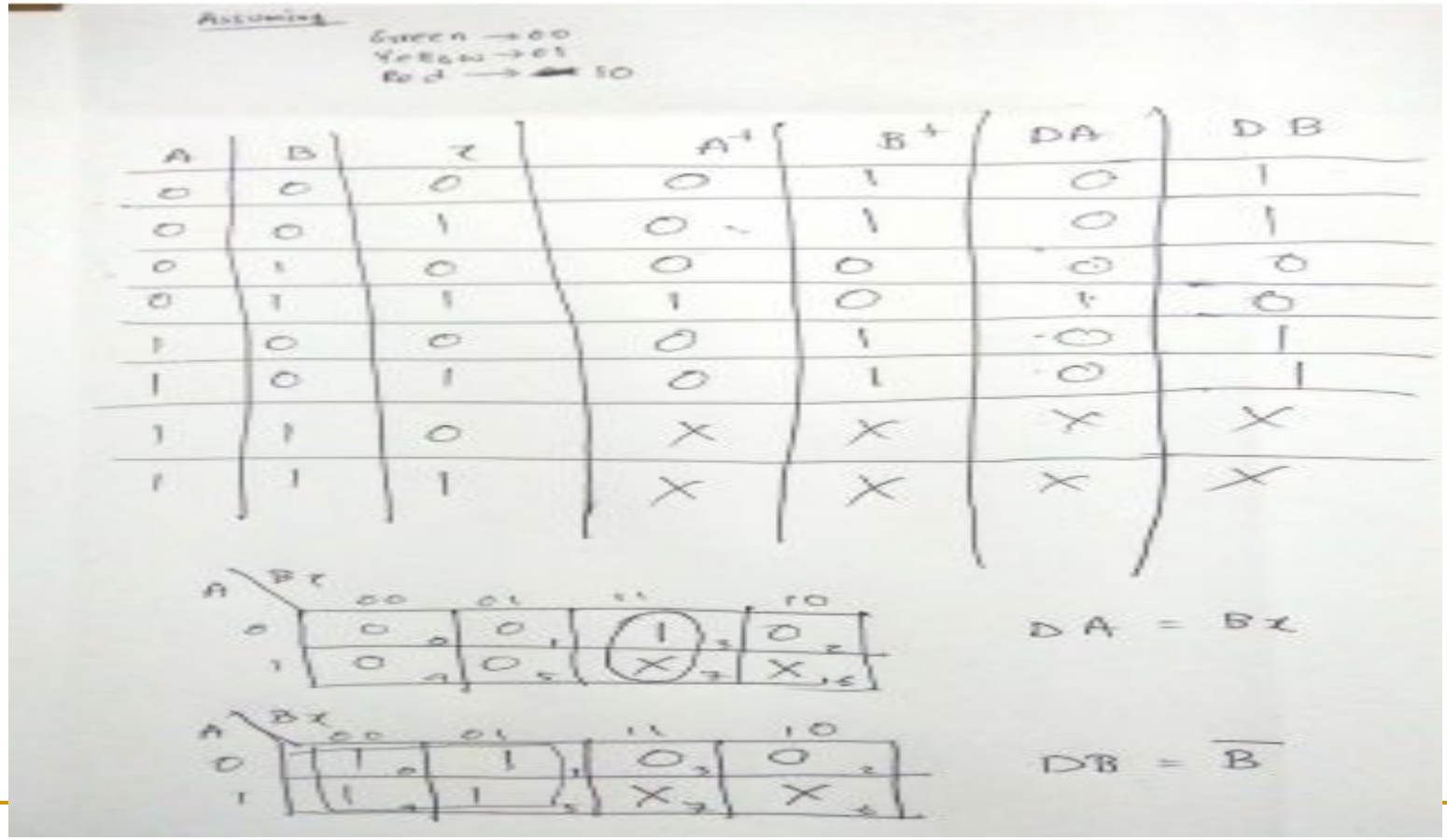
PRACTICE PROBLEM



PRACTICE PROBLEM

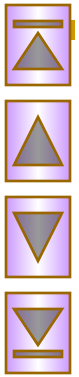
- Implement the following counter using D FF

  Green->Yellow->Red->Yellow->Green



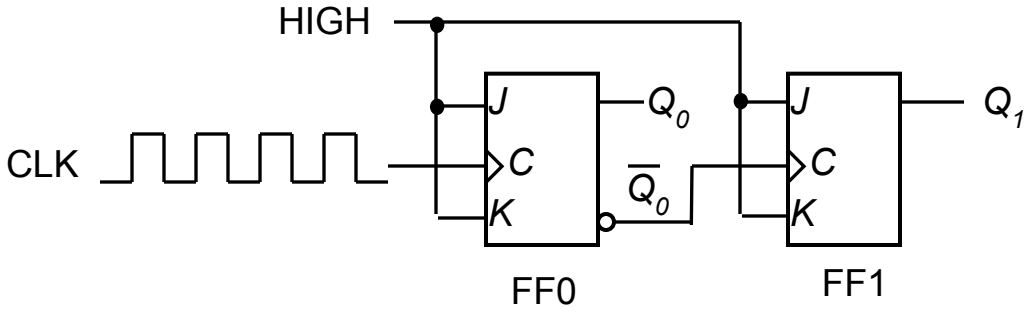
Asynchronous (Ripple) Counters

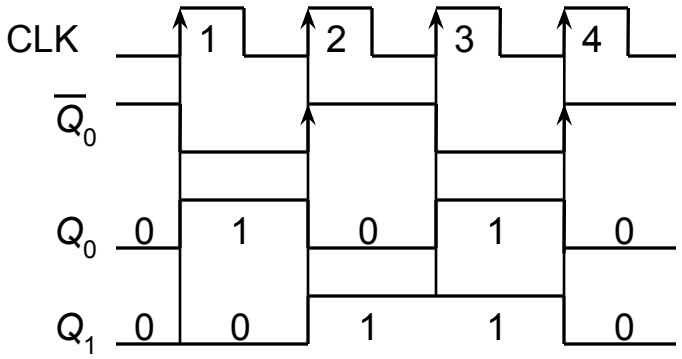
- Asynchronous counters: the flip-flops do not change states at exactly the same time as they do not have a common clock pulse.
- Also known as ripple counters, as the input clock pulse "ripples" through the counter – cumulative delay is a drawback.
- This counter is also a *frequency divider*.



Asynchronous (Ripple) Counters

- Example: 2-bit ripple binary counter.
- Output of one flip-flop is connected to the clock input of the next more-significant flip-flop.



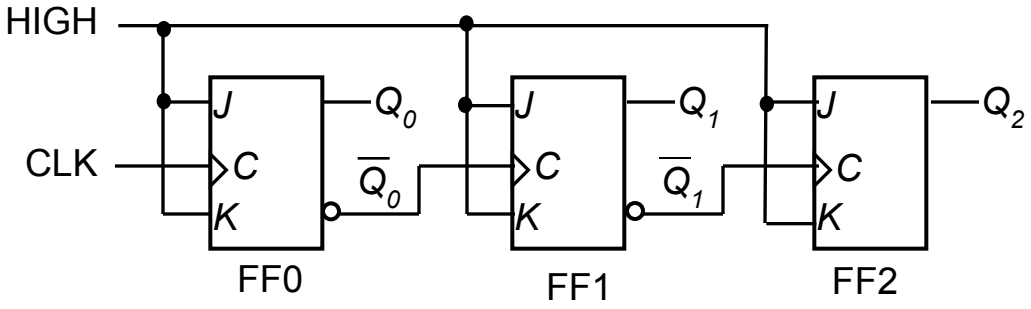


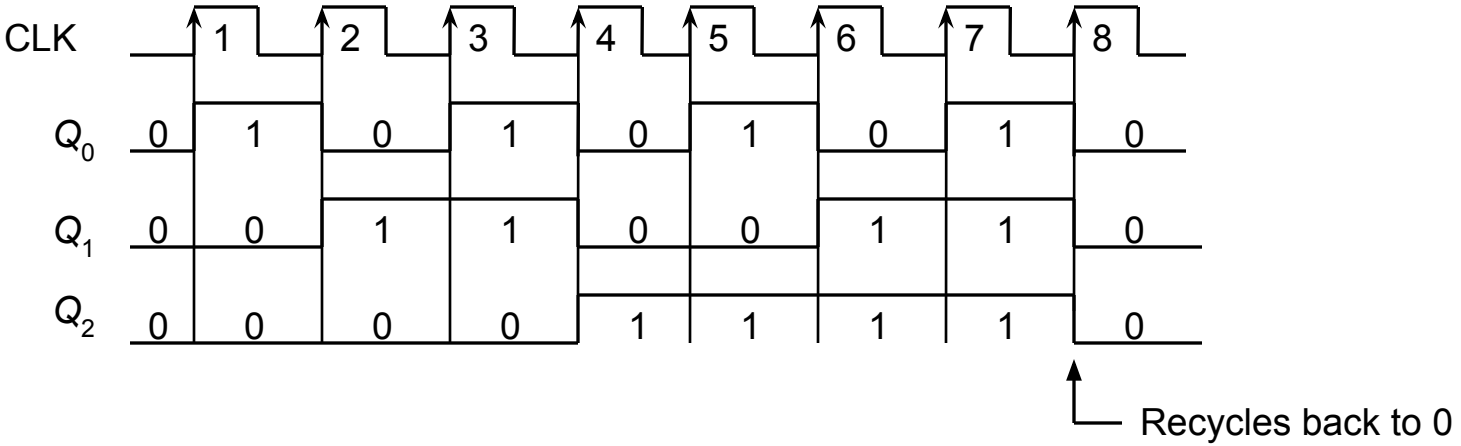
Timing diagram

00 → 01 → 10 → 11 → 00 ...

Asynchronous (Ripple) Counters

- Example: 3-bit ripple binary counter.





Asynchronous (Ripple) Counters

- Example: 4-bit ripple binary counter (negative-edge triggered).