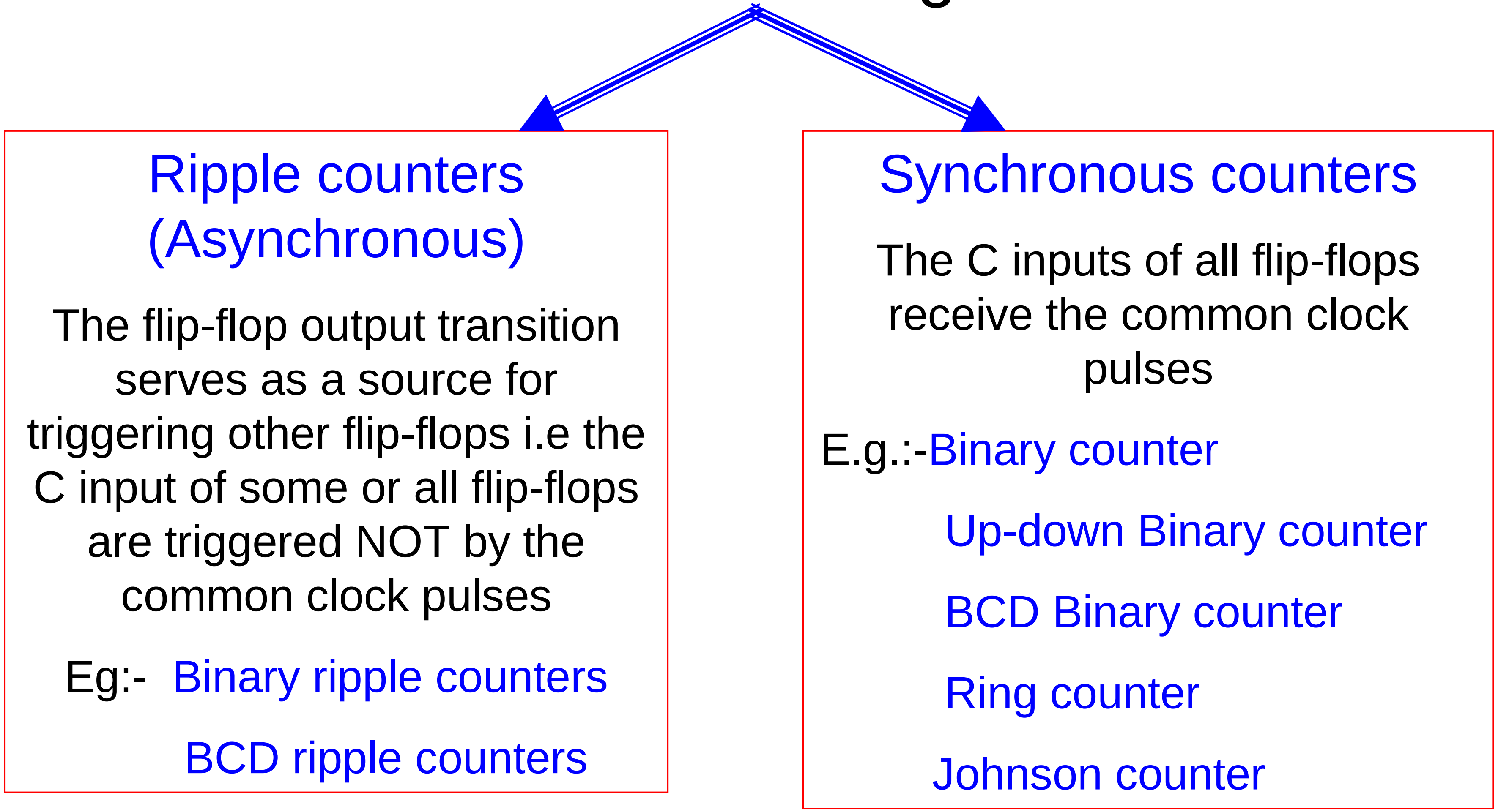
COUNTERS

- Counters are sequential circuits that cycle through some states.
- They can be implemented using flip-flops.
- Implementation is simple: using *T* flip-flops (with toggle output) or with any other flip-flops that can be connected to give the required function

COUNTERS

- Are available in two categories



Ripple counters (Asynchronous)

The flip-flop output transition serves as a source for triggering other flip-flops i.e the C input of some or all flip-flops are triggered NOT by the common clock pulses

Eg:- Binary ripple counters

BCD ripple counters

Synchronous counters

The C inputs of all flip-flops receive the common clock pulses

E.g.:-Binary counter

Up-down Binary counter

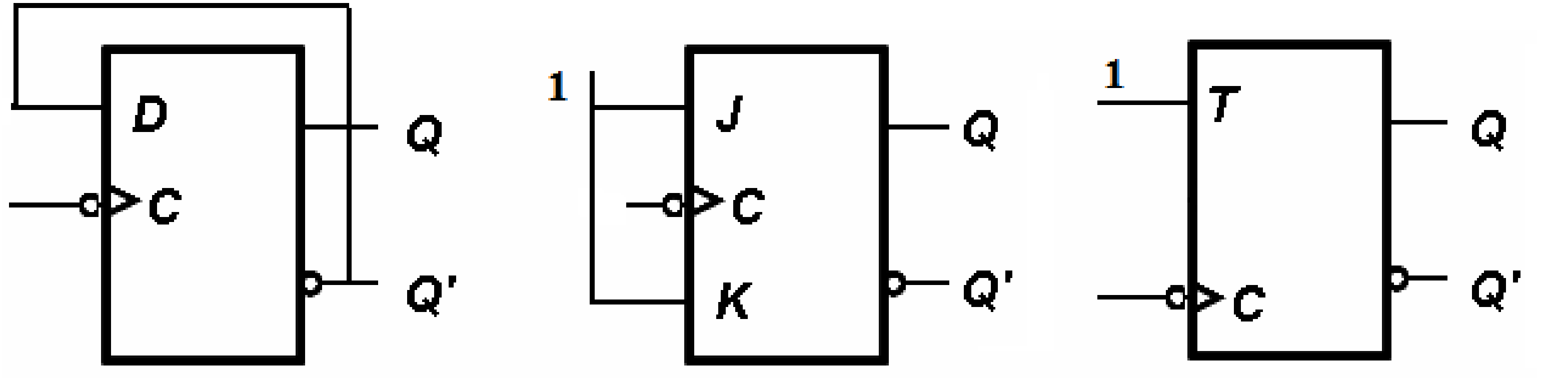
BCD Binary counter

Ring counter

Johnson counter

Ripple counters

- use complemented flip flop

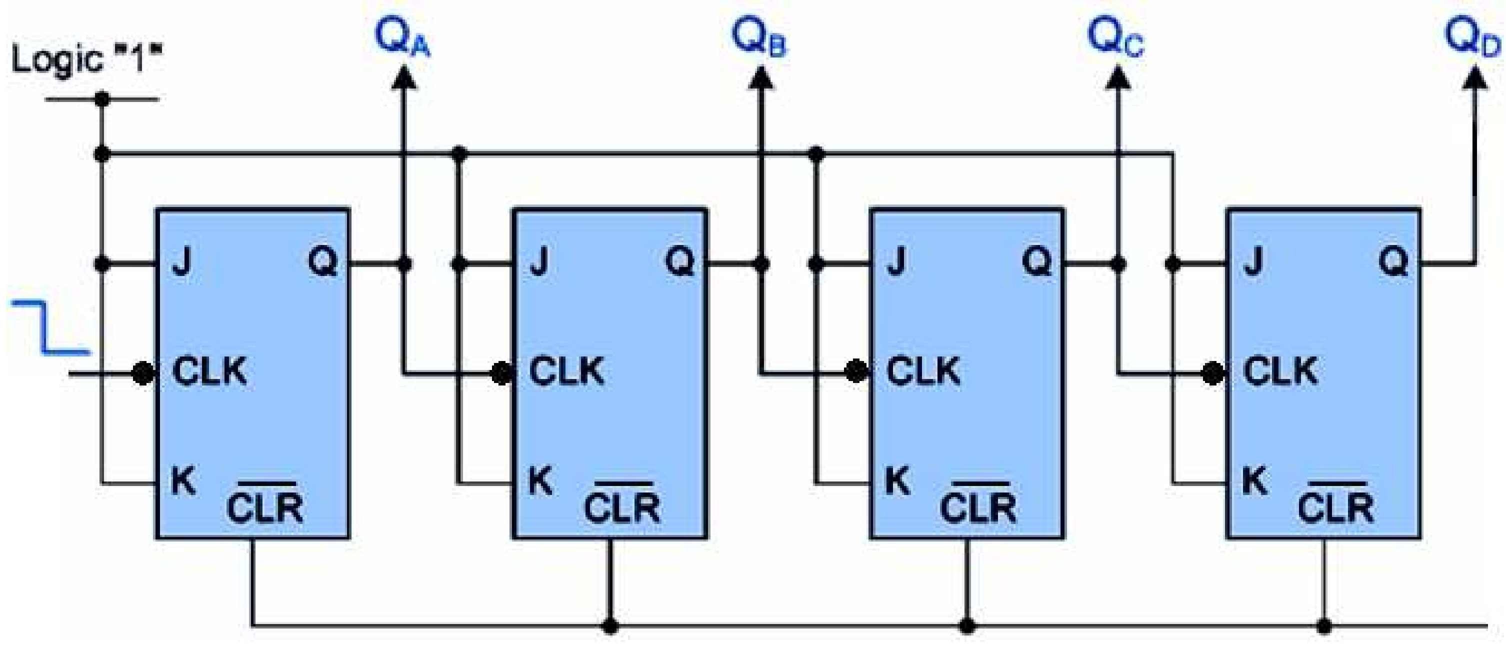


Negative edge-triggered flip-flops

Q(t+1)=Q`(t)

Binary Ripple up counter

- Consist of a series of connection of negative edge triggering complementing flip-flops with the output of each flip-flop connected to the C input of the next high order flip flop



•The flip flop holding the LSB receives the input pulses.

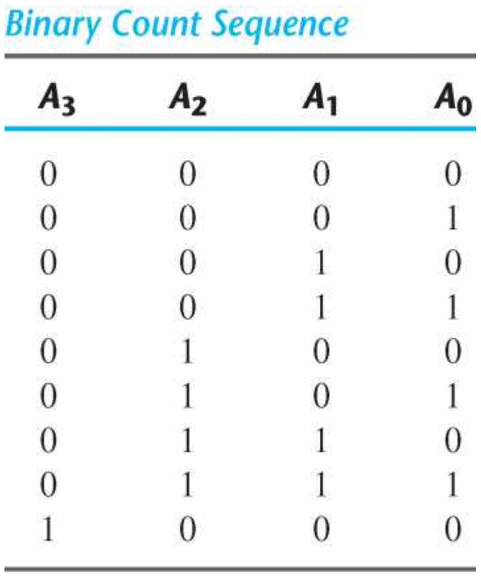
•The count starts with binary 0 and increments by one with each count pulse input

•After the count 15 the counter goes back to binary 0 to repeat the count

Binary ripple up counter

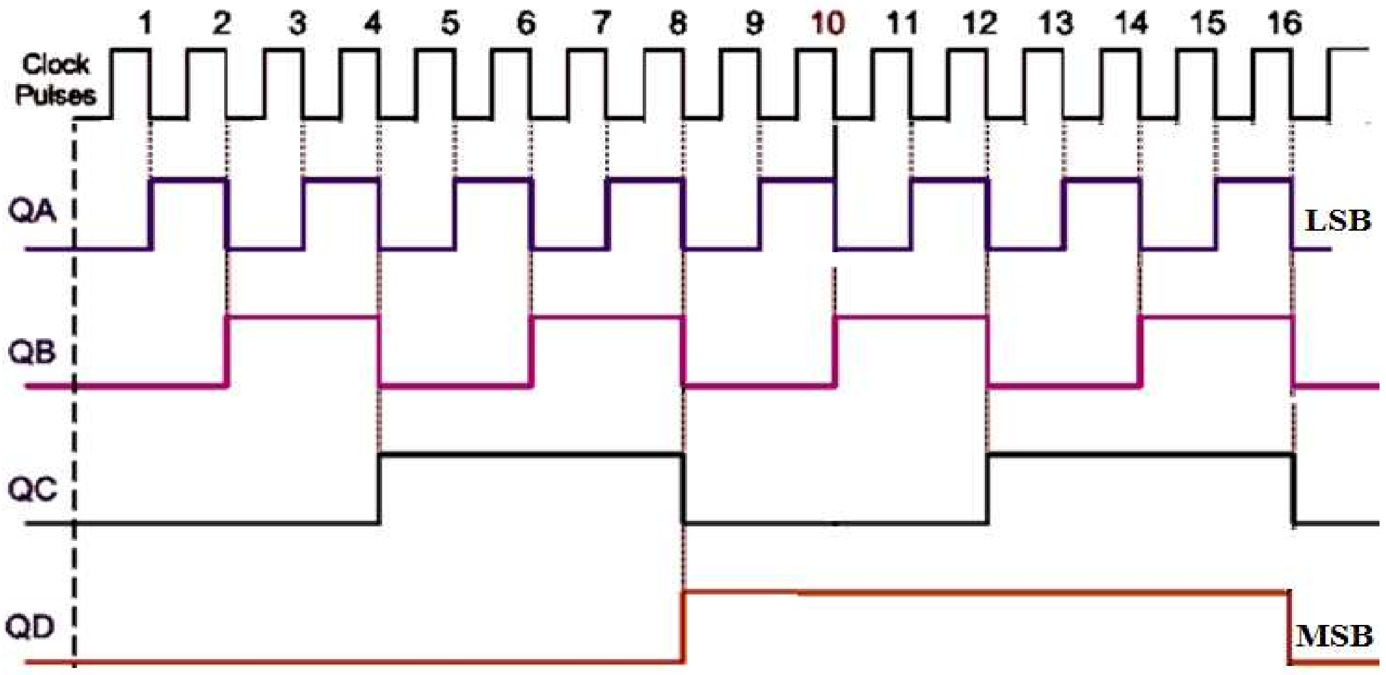
- The LSB A is complemented with each count pulse input. Transition of A from 1 to 0

  complement B and so on



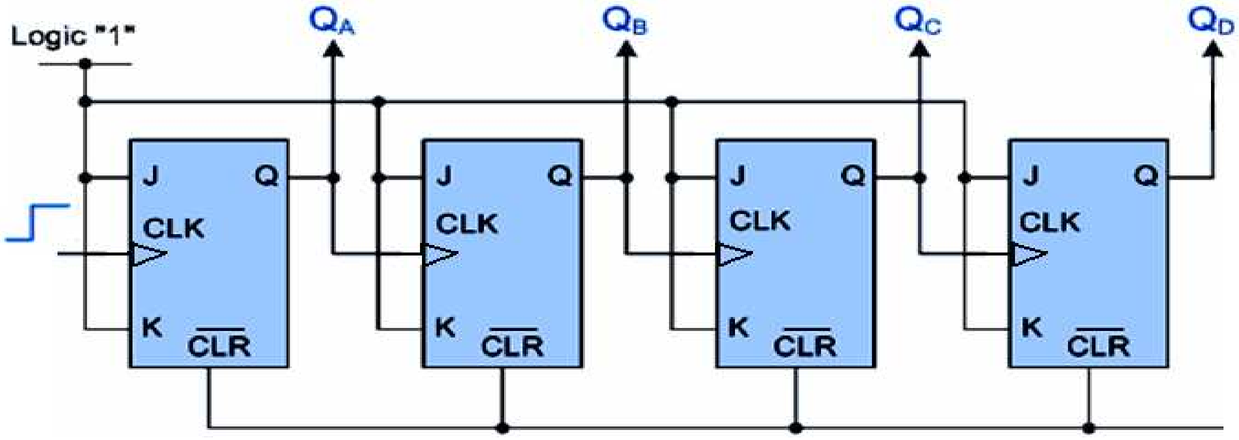
Binary Count Sequence

A_3	A_2	A_1	A_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

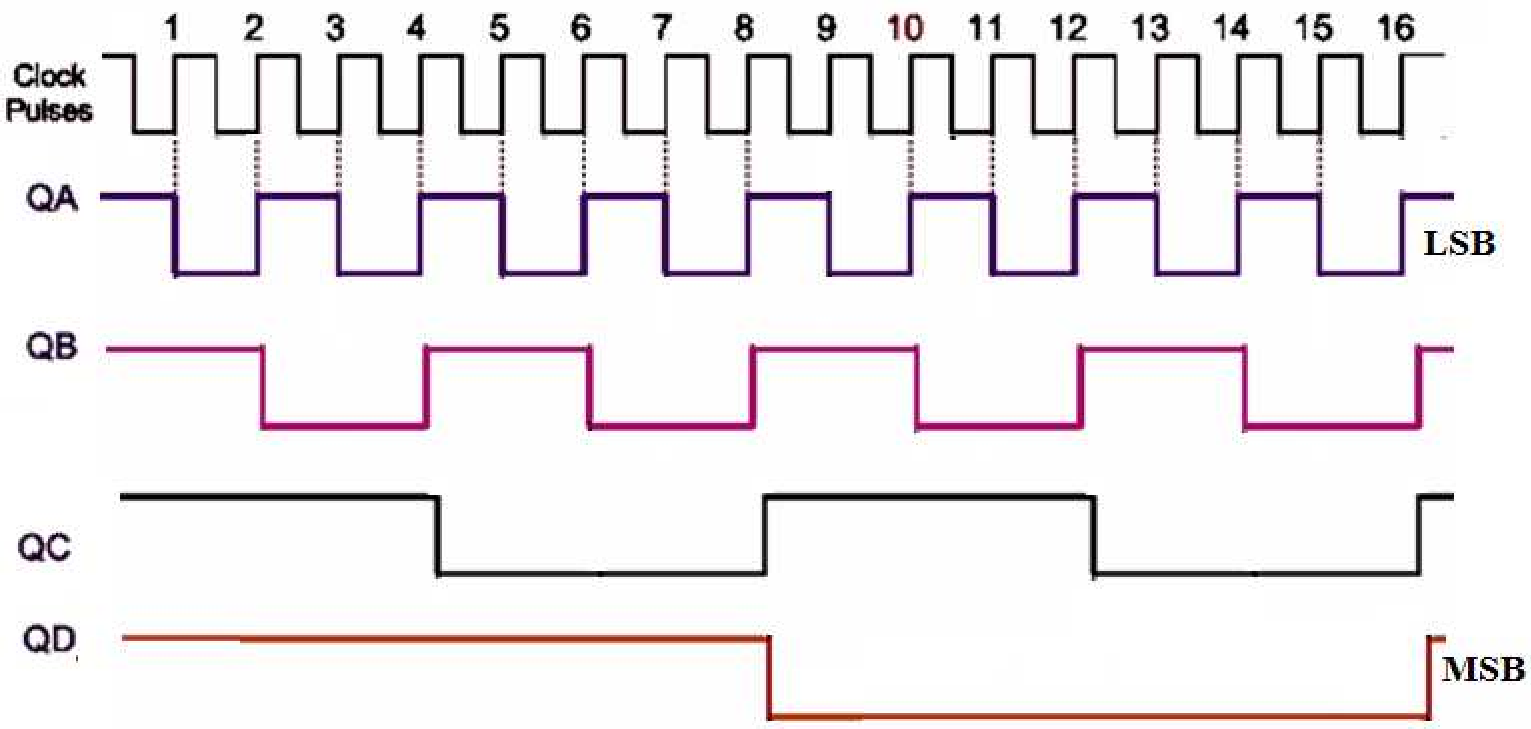


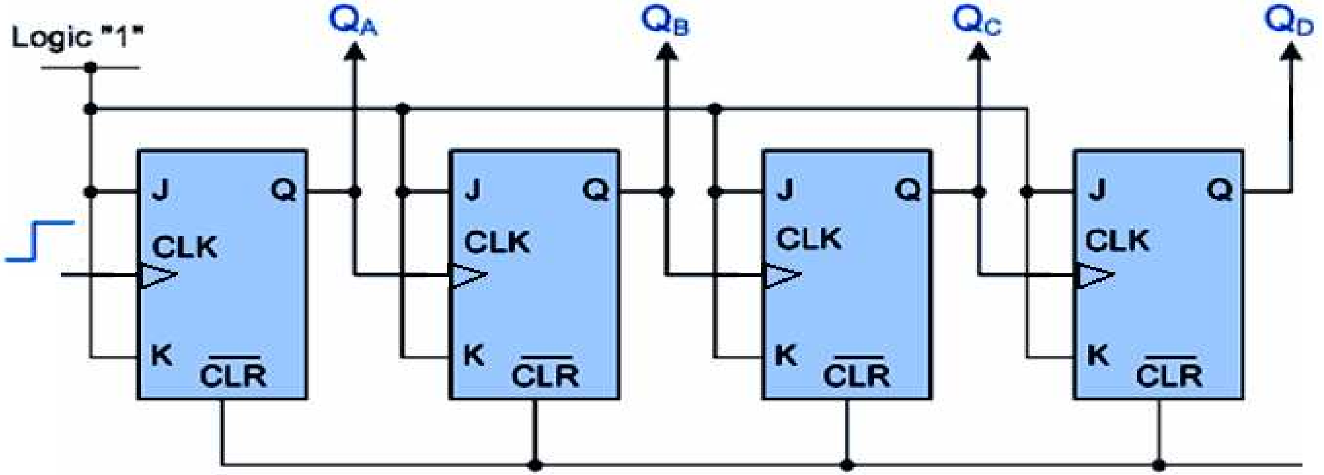
Binary Ripple down counter

- For positive edge triggered flip-flops the counter count down:
- e.g start from15 to 14 to 13 to…….
- The diagram is same as the count up binary counter except that the flip-flop trigger on the positive edge of the clock.
- If negative edge triggered flip-flops are used then the C input of each flip-flop must be connected to the complement output of the previous flip-flop. So, when the true output goes from 0 to 1, the complement will go from 1 to o and complement the next flip flop as required



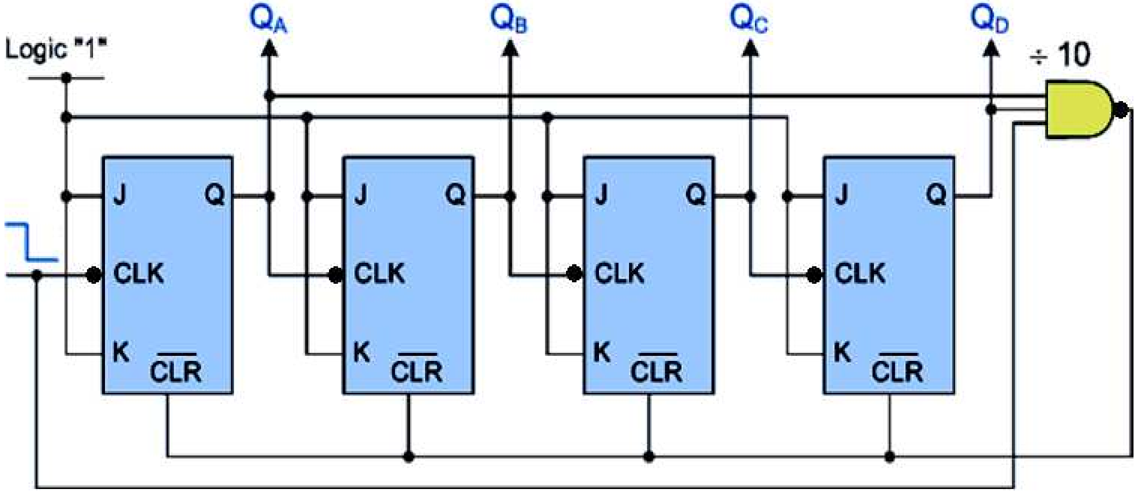
Ripple down counter

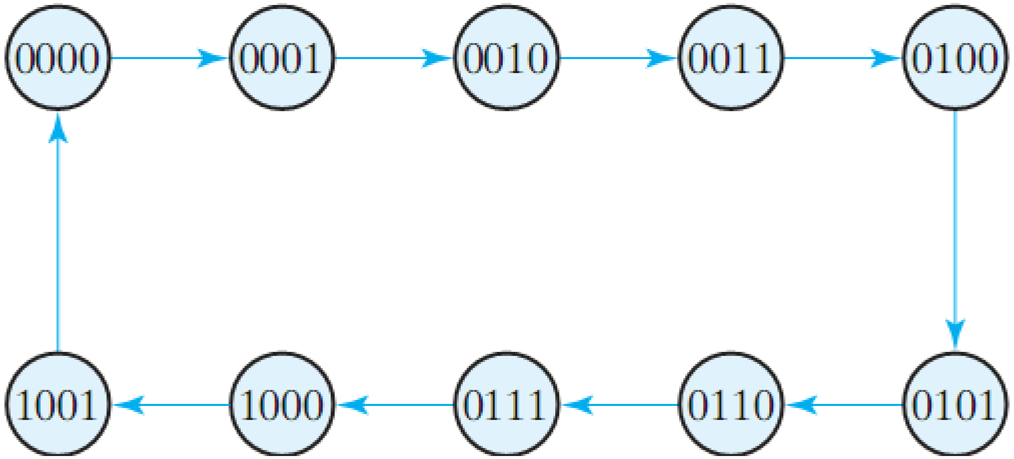




BCD Ripple Counter, Decade counter

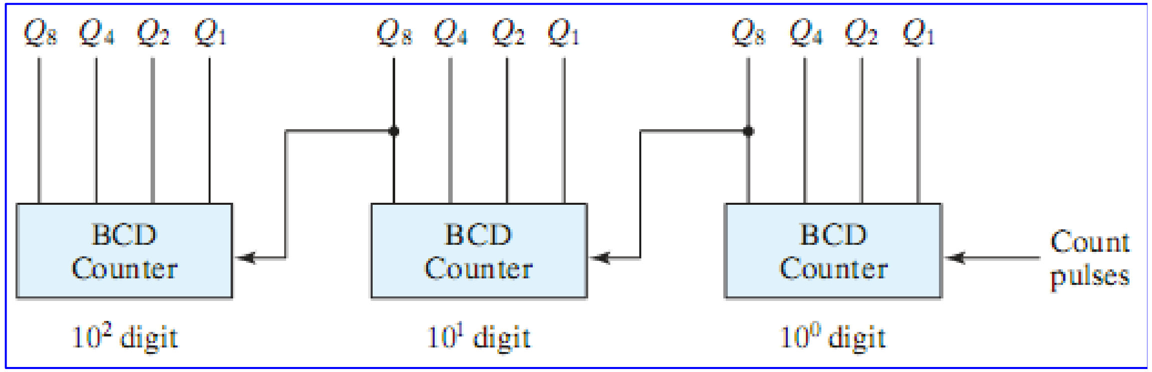
This counter counts upwards on each negative edge of the input clock signal starting from "0000" until it reaches an output "1001“. Both outputs Q_A and Q_D are now equal to logic "1" and the output from the NAND gate changes state from logic "1" to a logic "0" level when the clock goes to level one and whose output is also connected to the CLEAR (CLR) inputs of all the J-K Flip-flops

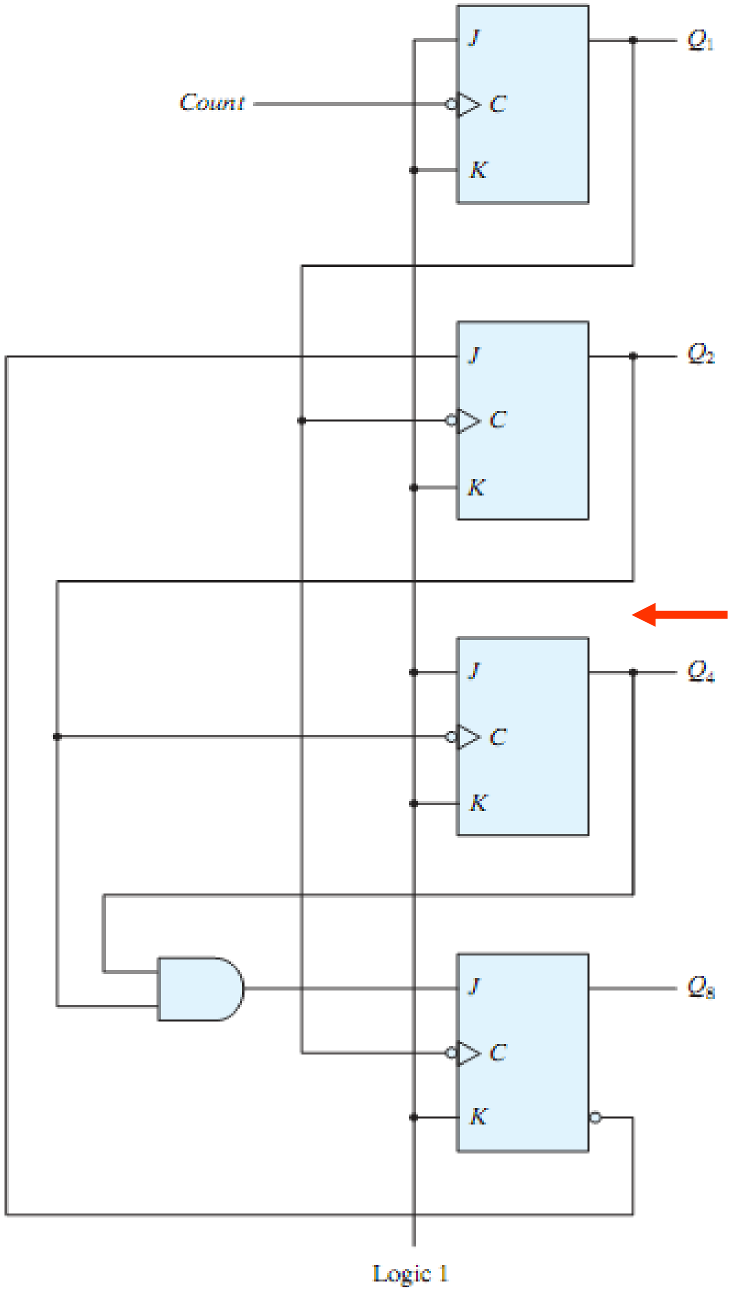




BCD Ripple Counter,

BCD is called decade counter (0-9). To count from 0-99 2-decade counters are needed, and to count up to 999 3-decade counters are need and connected as shown below





BCD counters can also be constructed as shown.

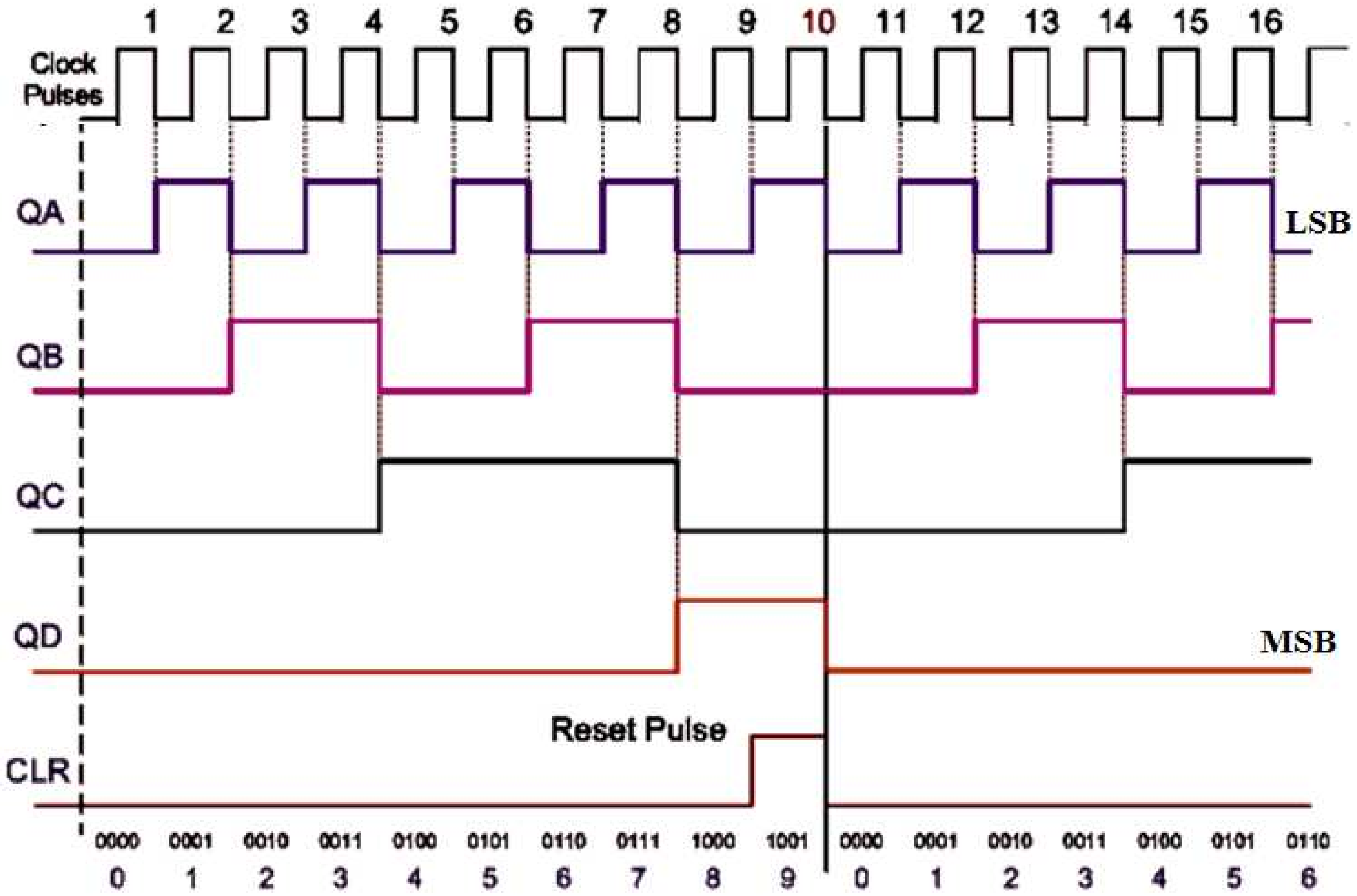
Q_1 change state after each clock pulse

Q_2 complement every time Q1 goes 1-0 as long as Q_8=0. When Q_8=1 Q_2 remains at 0

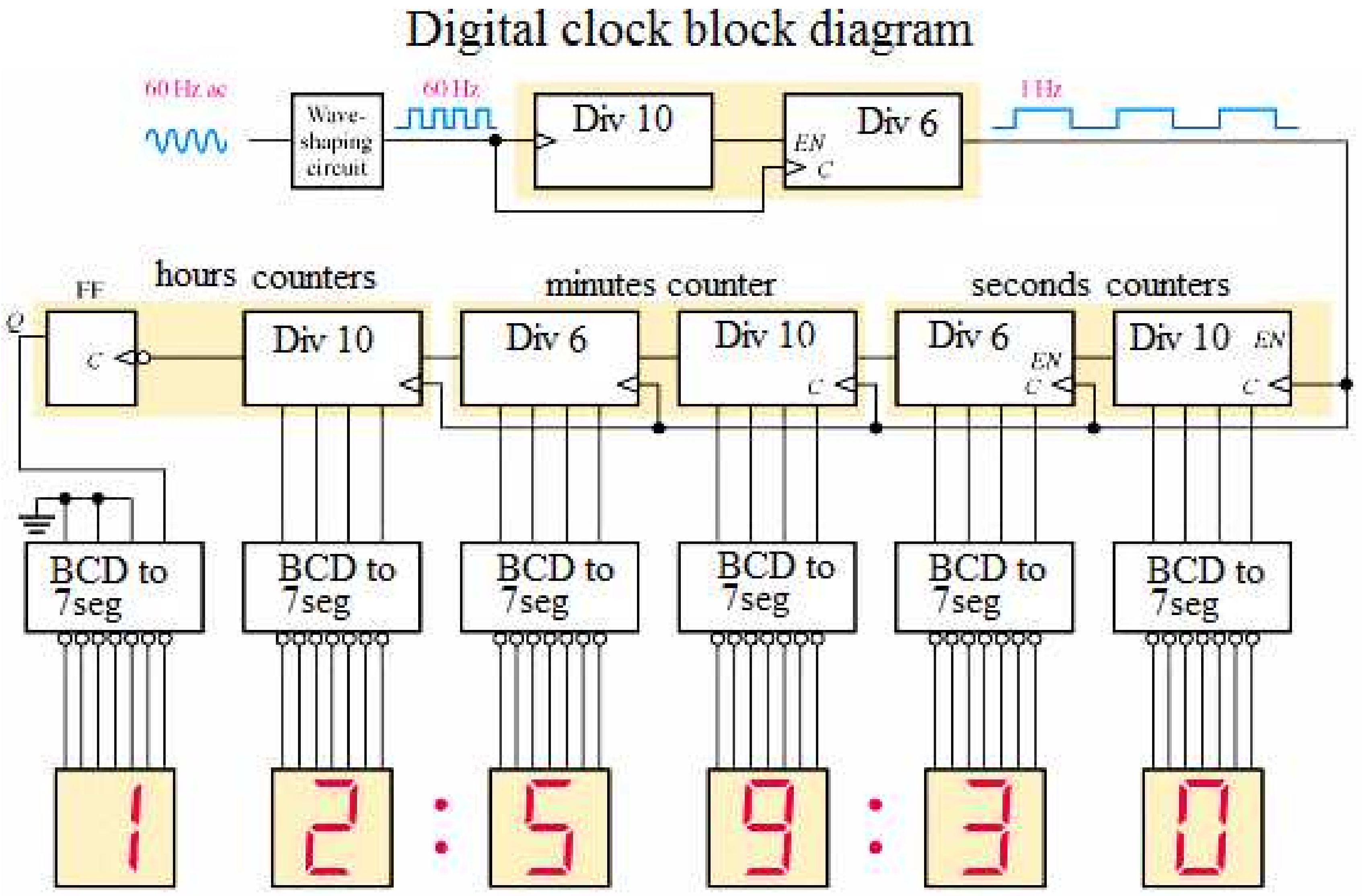
Q_4 complements every time Q_2 goes from 1 to 0

Q_8 remains at 0 as long as Q_2 or Q_4 is 0

Decade Counter Timing Diagram



Decade counters applications

- 

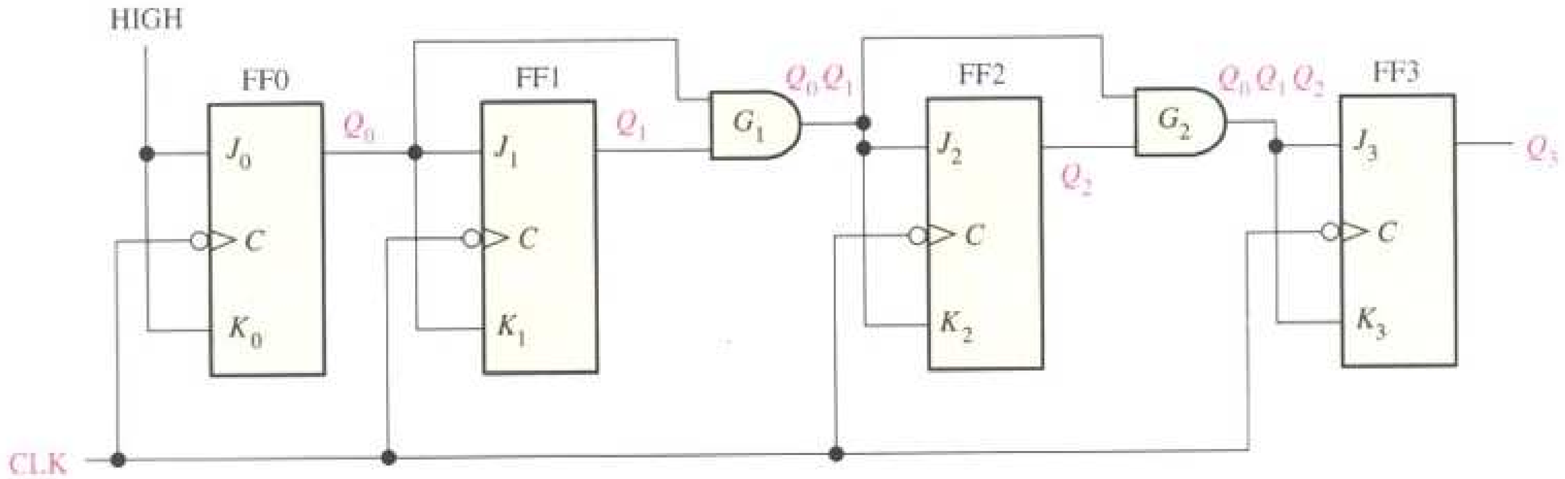
Synchronous counters

- Binary Counters
- Up-Down Binary Counter
- BCD Counter
- Binary counter with Parallel Load
- Ring and Johnson counters

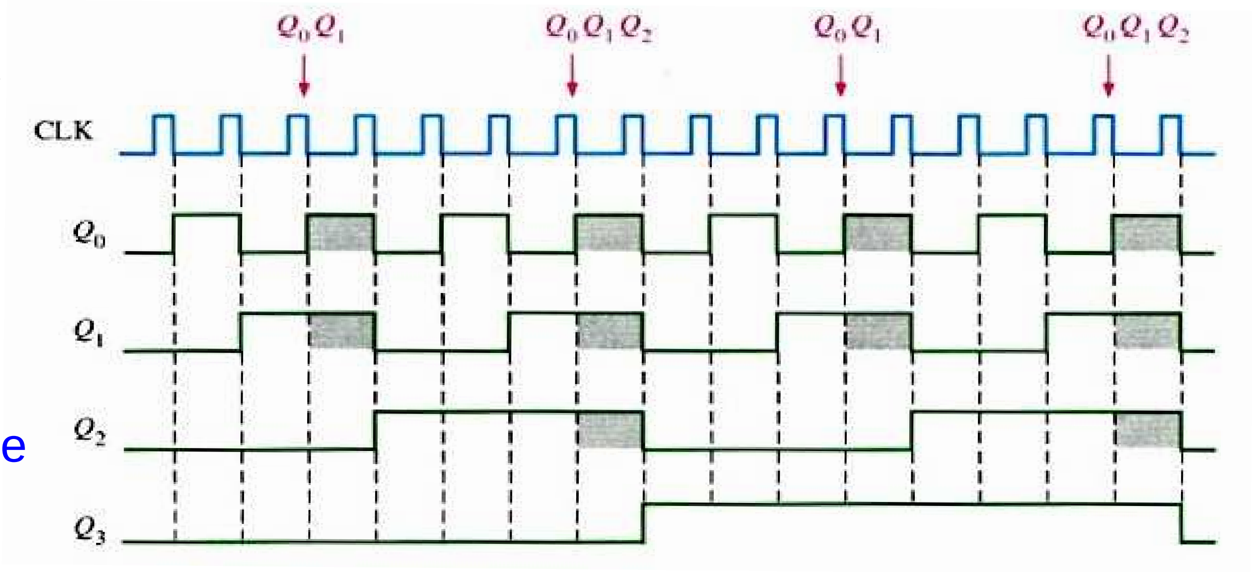
they are used to generate time signals necessary for digital operations

- FFs in the counter are clocked at the same time by a **common clock pulse**.
- The design procedure is the same as that of sequential circuit

Binary Synchronous Counter



- The FF in the LSB is complemented with every pulse. A flip flop in other position is complemented when all the bits in the lower significant positions are 1
- Synchronous counter have a regular pattern and can be constructed with complementing flip flops and gates



The polarity of the clock is not important, so can be triggered with either the positive or the negative clock edge

Up-Down Binary Counters

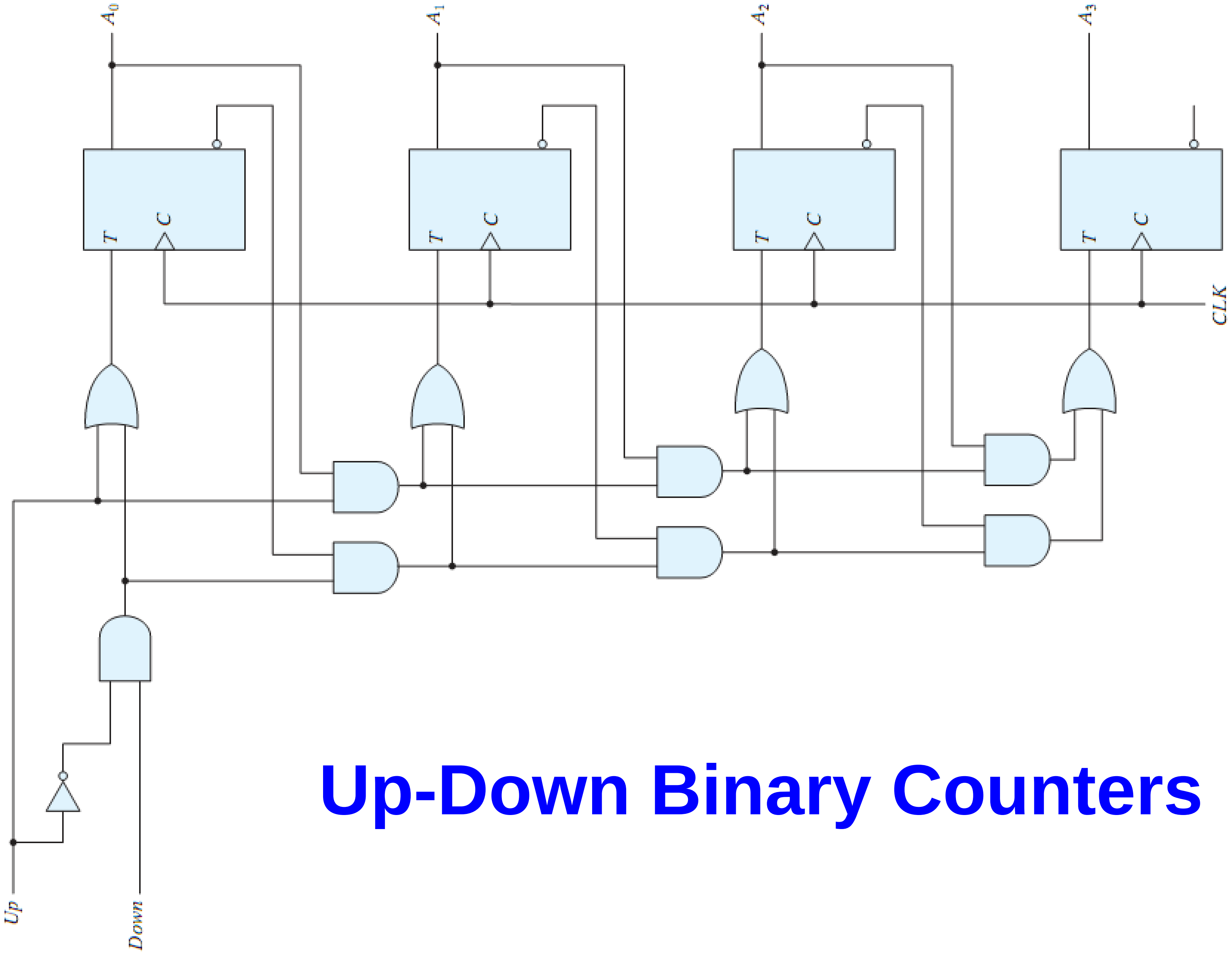
- It can progress in either direction (up or down)

0 1 2 3 4 5 4 3 2 3 4 5 6 7 6 5 etc...

up dn up dn

The count down counter can be constructed as follows, the inputs to the AND gates must come from the complement outputs instead of the normal outputs of the previous flip flops.

The Up and down counters can be combined in one circuit to form a counter capable of counting either up or down.



Up-Down Binary Counters

Design Problem

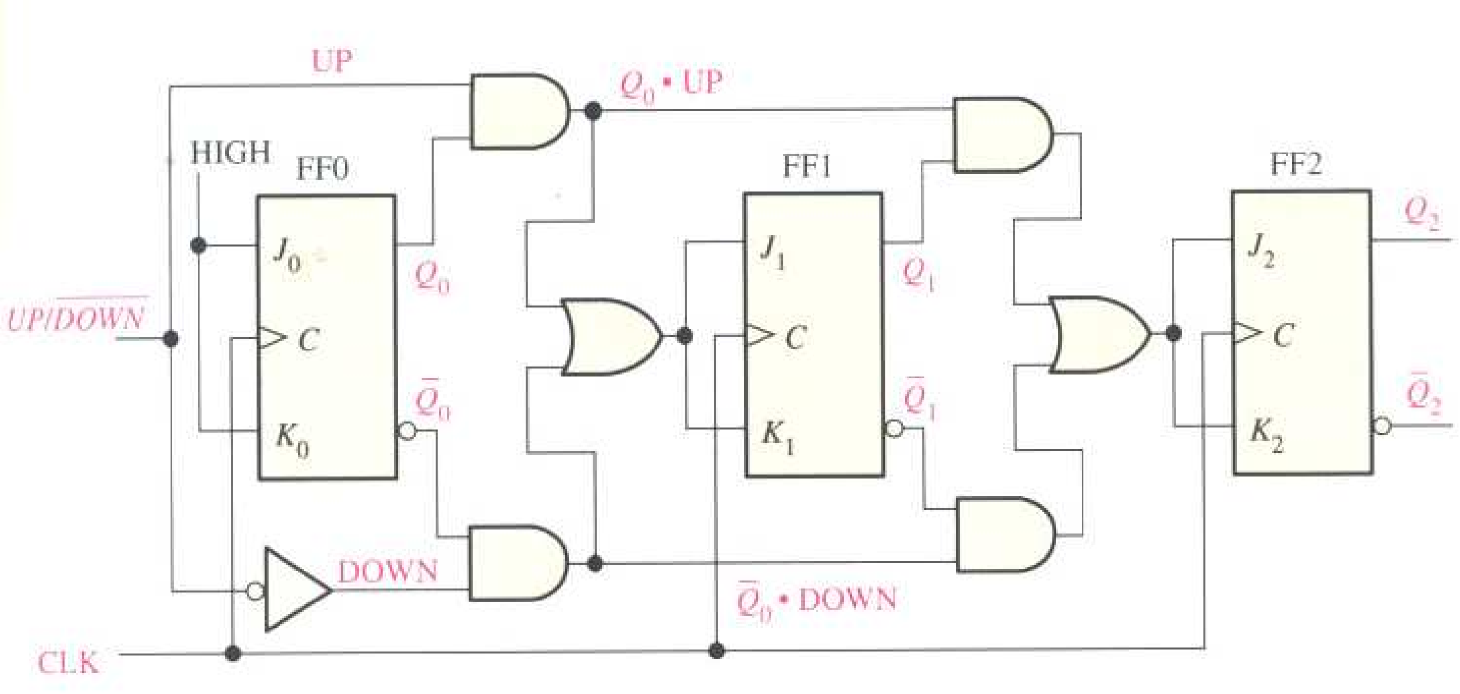
1-Design 2-bit up-down synchronous counters with T flip flops

2- Design 3-bit up-down synchronous counters with JK flip flops

Answer for No.2

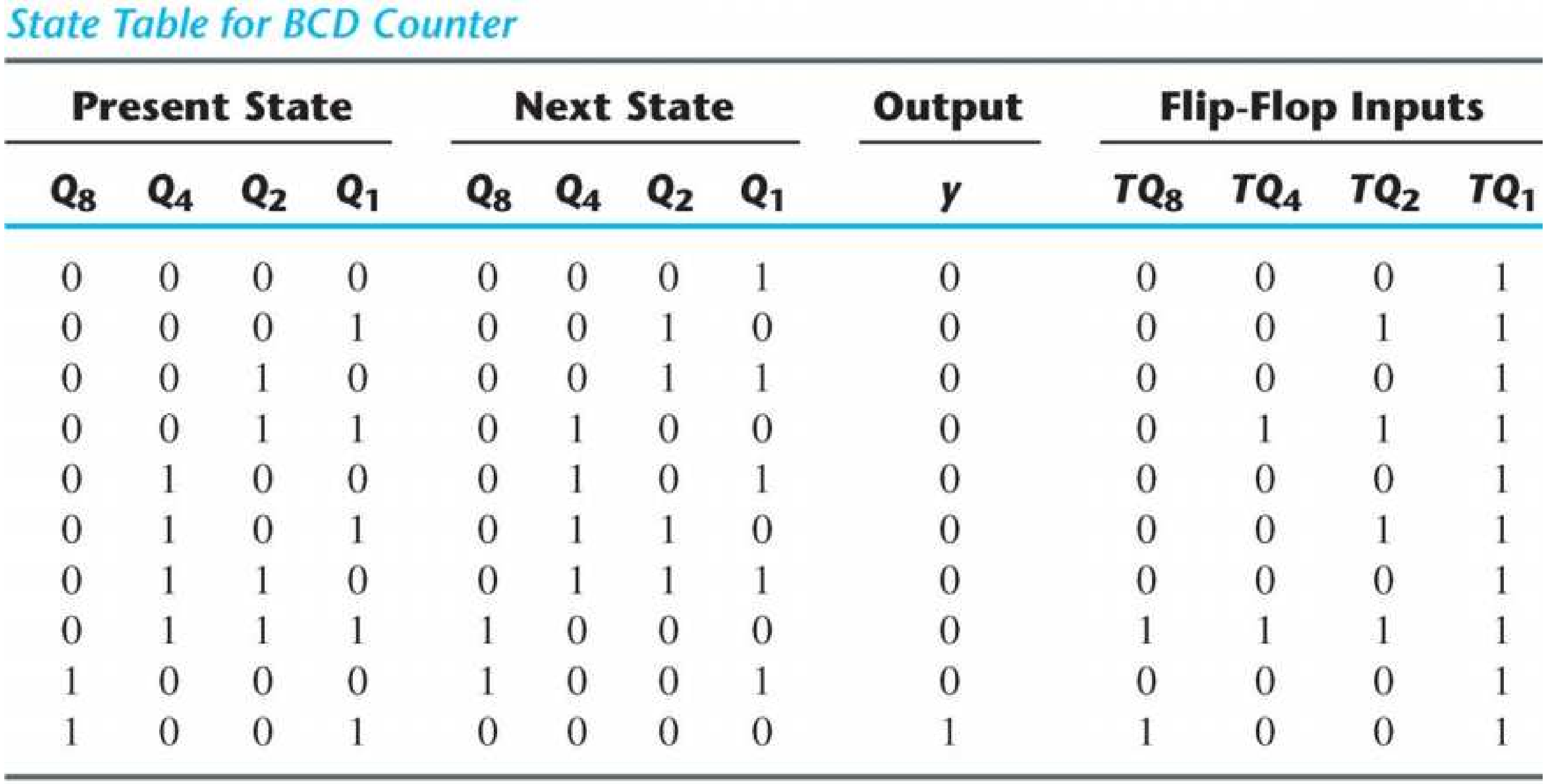
Q_0: $J_0 = K_0 = 1$ Q_1: $J_1 = K_1 = (Q_0 \cdot UP) + (\neg Q_0 \cdot DN)$

Q_2: $J_2 = K_2 = (Q_0 \cdot Q_1 \cdot UP) + (\neg Q_0 \cdot \neg Q_1 \cdot DN)$



Synchronous BCD counter

- It does not have regular pattern as in binary counter, so procedure of sequential design should be used



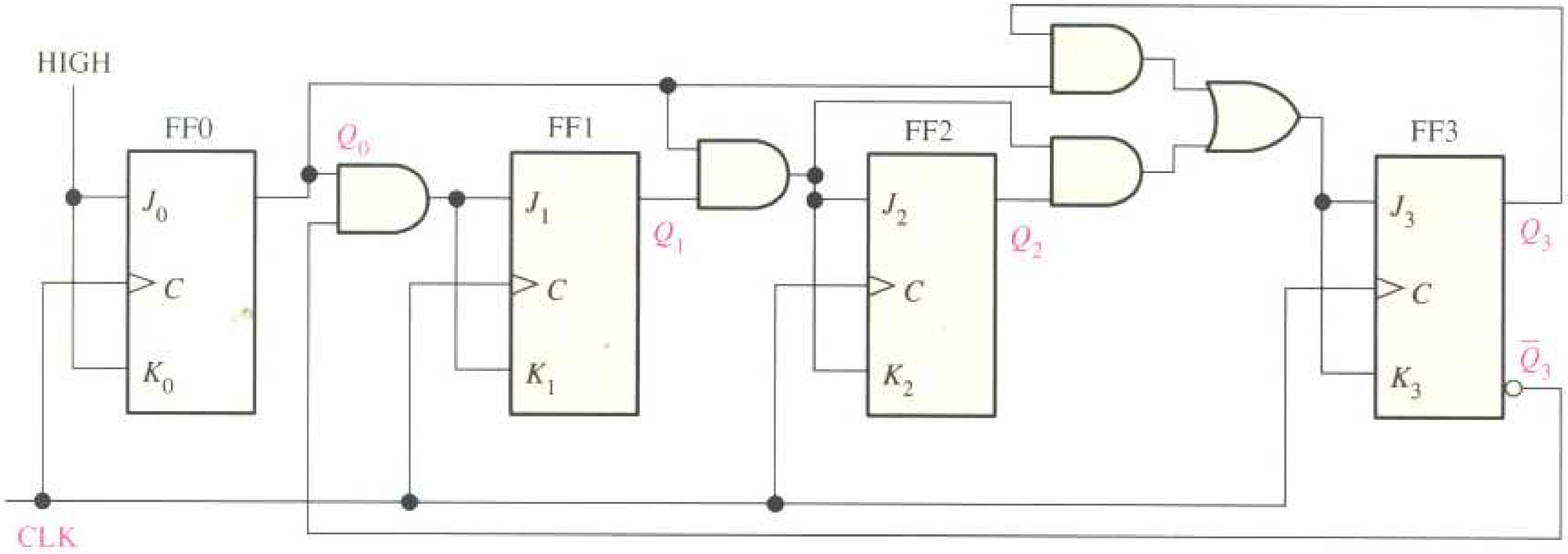
State Table for BCD Counter

Present State				Next State				Output	Flip-Flop Inputs			
Q_8	Q_4	Q_2	Q_1	Q_8	Q_4	Q_2	Q_1	y	TQ_8	TQ_4	TQ_2	TQ_1
0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	1	0	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	0	1
0	0	1	1	0	1	0	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	0	1
0	1	0	1	0	1	1	0	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	0	1
0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	0	1
1	0	0	1	0	0	0	0	1	1	0	0	1

Use k-map to find input functions to the T flipflops as:

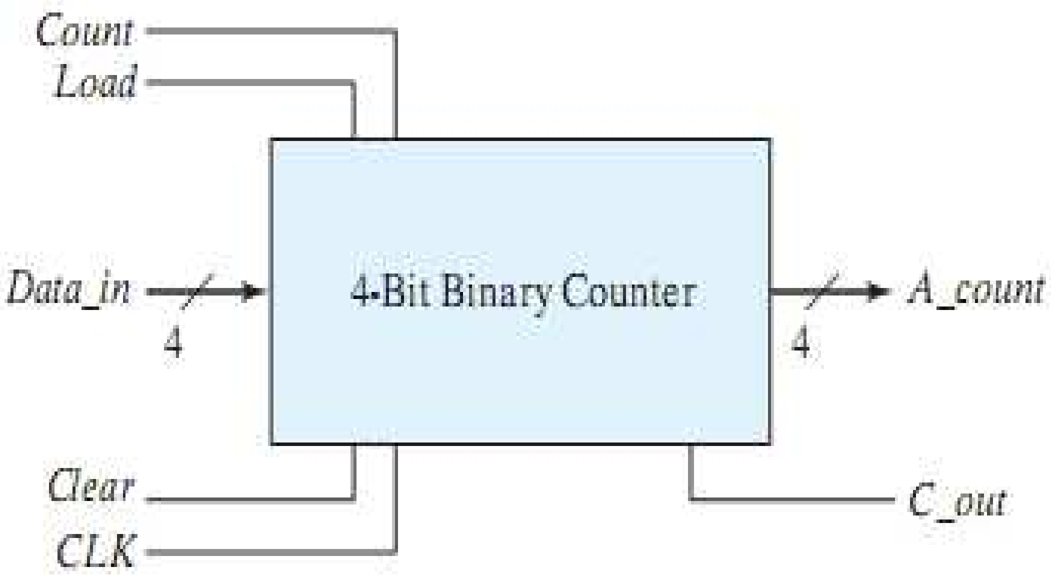
T_{Q1}=1, T_{Q2}=Q'_8Q_1 T_{Q4}= Q_2Q_1; T_{Q8}=$Q_8Q_1+Q_4Q_2Q_1$

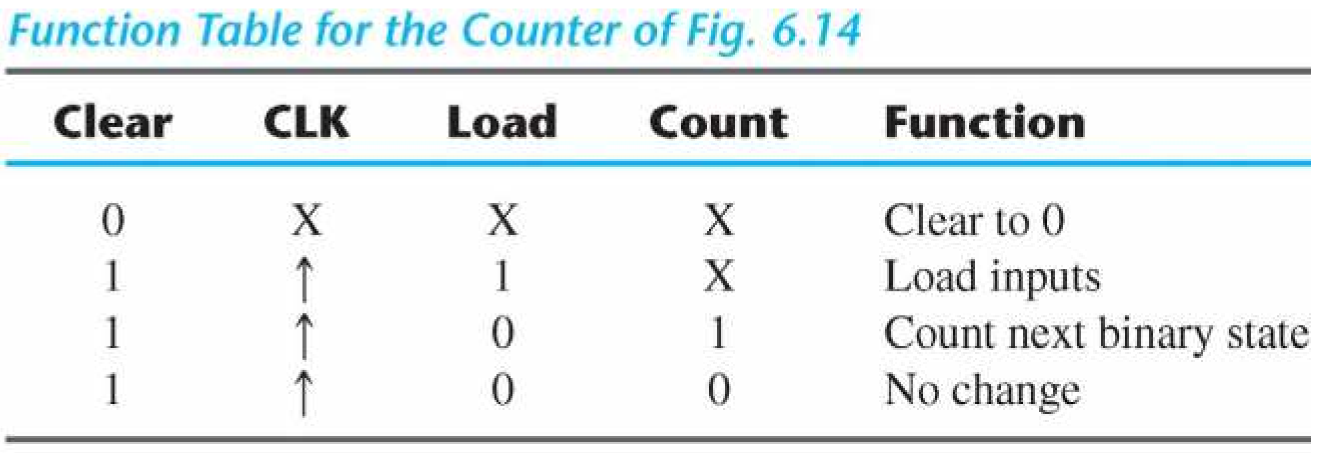
JK Synchronous Decade Counter



Binary Counter with Parallel Load

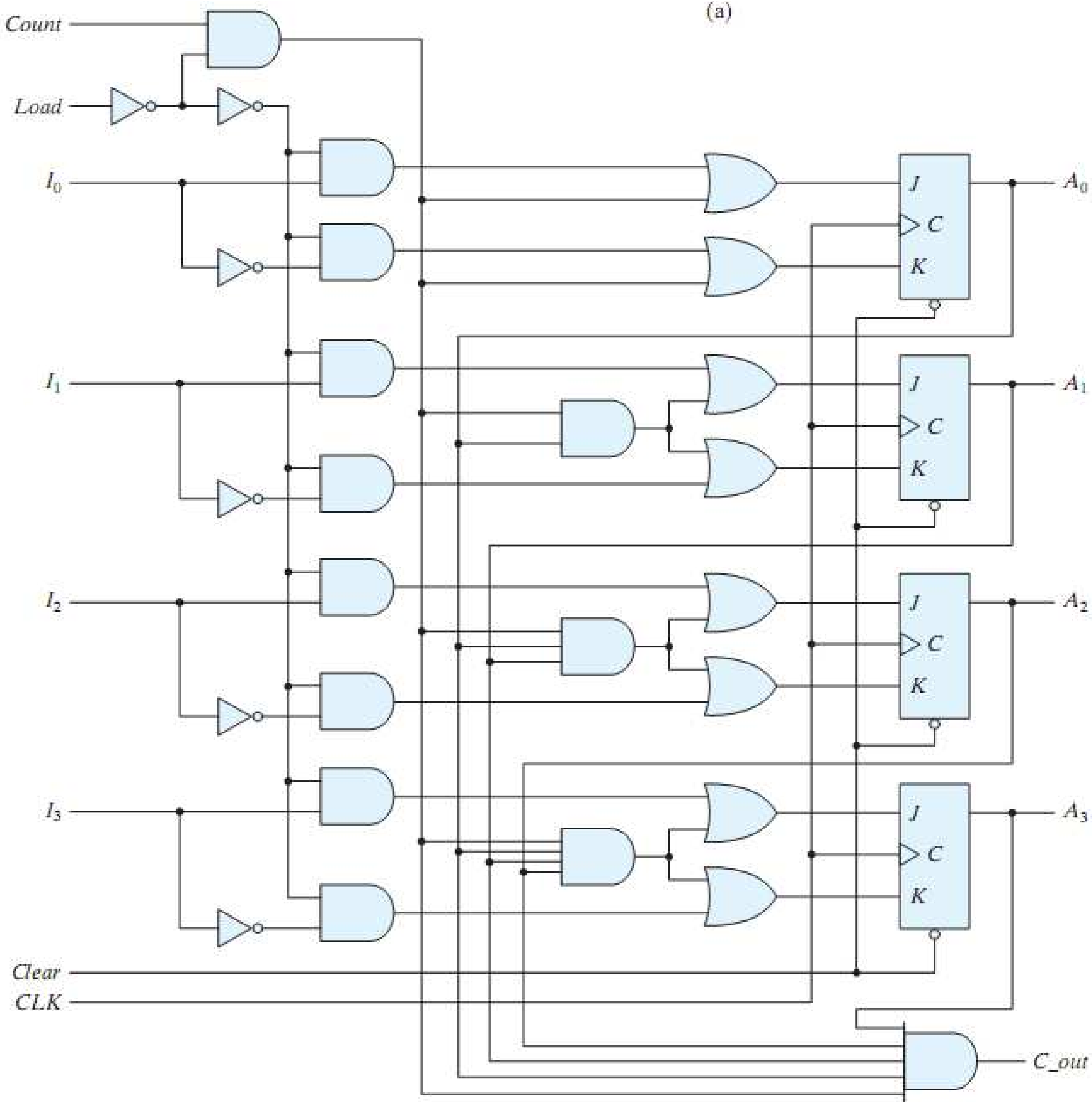
- It can be loaded with initial value to start counting




Function Table for the Counter of Fig. 6.14

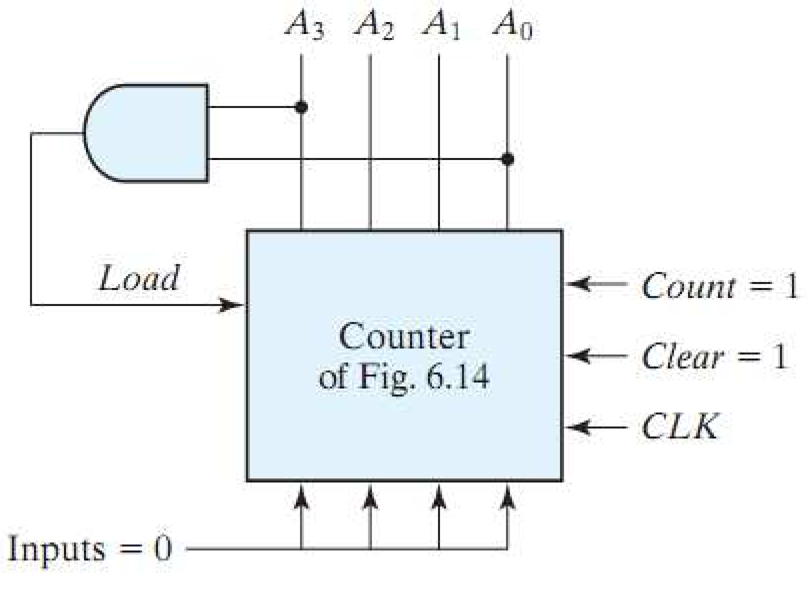
Clear	CLK	Load	Count	Function
0	X	X	X	Clear to 0
1	↑	1	X	Load inputs
1	↑	0	1	Count next binary state
1	↑	0	0	No change

(a)

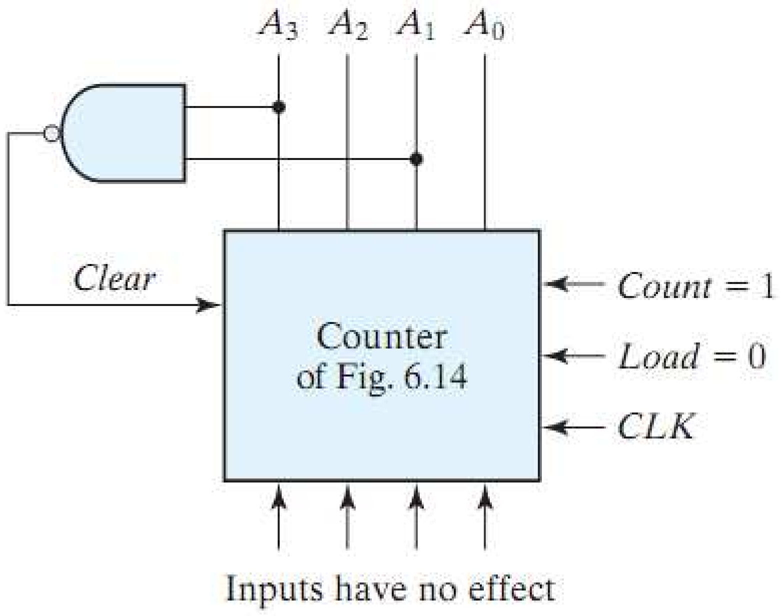


Binary Counter with Parallel Load

- A counter with parallel load can be used to generate any count sequence. Figures below show two ways in which a counter with parallel load is used to generate the BCD count



(a) Using the load input



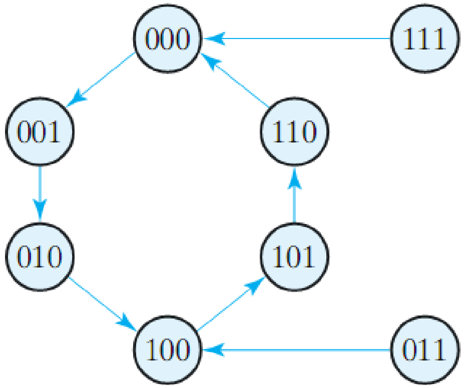
(b) Using the clear input

When 1001 is detected, the counter is initially cleared to zero, then the clear and count inputs are set to 1

The NAND gate detects the count 1010 but as soon as this count occur, the register is cleared, the count 1010 has no chance to stay on for any appreciable time because the register goes immediately to 0.

- **Problem**

  Design synchronous counters that goes through the shown states (use JK flip flops). Assume that state 111 and 011 are unused state. Test the unused state to not block the system



Sol.

State Table for Counter

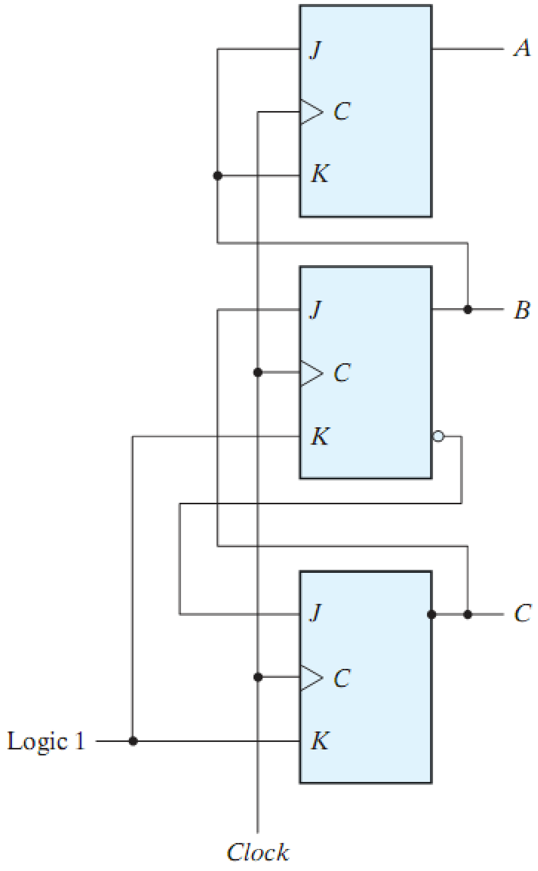
Present State			Next State			Flip-Flop Inputs					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	0	0	0	X	1	X	1	0	X

Use K-map

J_A=B, K_A=B

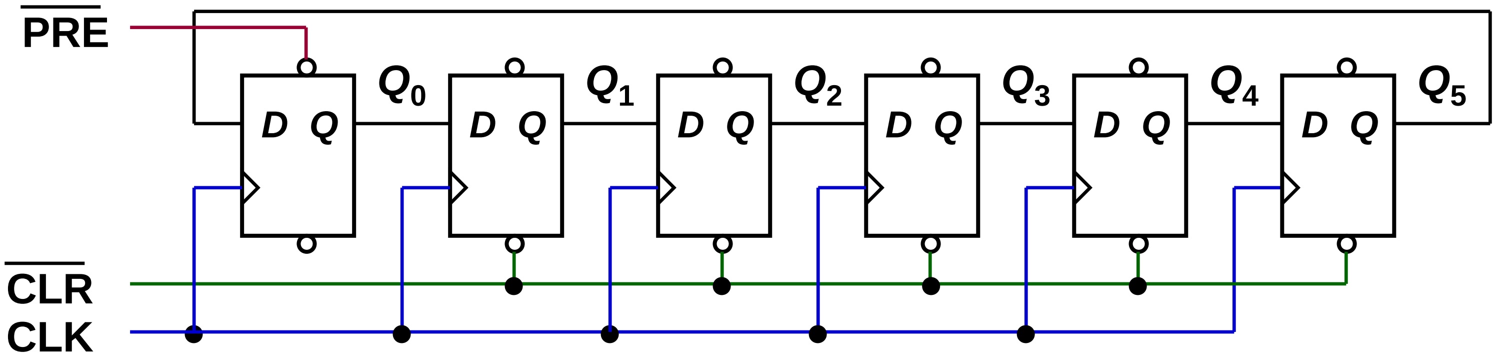
J_B=C, K_B=1

J_C=B', K_C=1

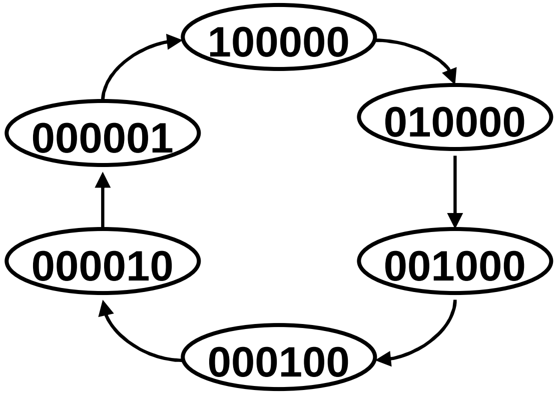


Ring counters

- An *n-bit* ring counter cycles through *n* states. The single bit is shifted from one flip flop to the next in order to generate unique timing signals



Clock	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1

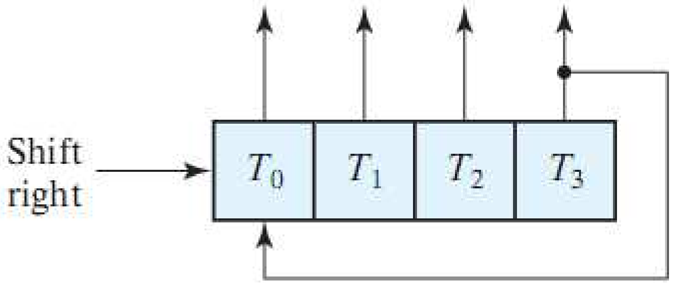


Only one flip-flop is set and all others are cleared

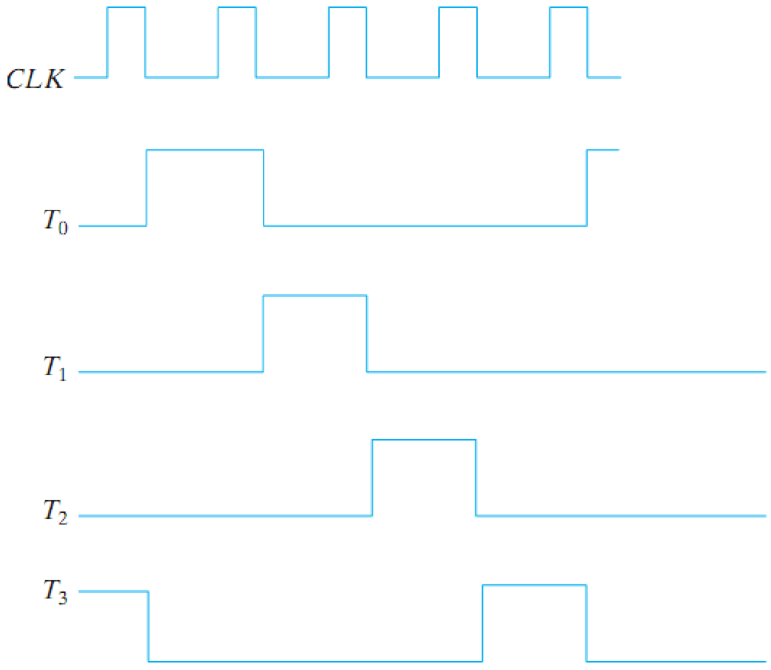
Ring counters

Figure show a 4 bit register connected as a Ring counter, the initial value is set to 1000.

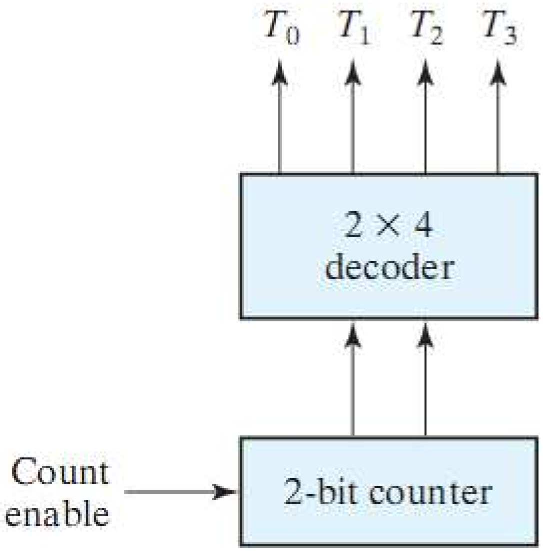
As alternative design, the timing signals can be generated by a two-bit counter that goes through four distinct states. The decoder shown in the figure decodes the four states of the counter and generates the required sequence of timing signals. To generate 2^n timing signals we require either a shift register with 2^n FFs or n bit binary counter with an n-to 2^n line decoder



(a) Ring-counter (initial value = 1000)



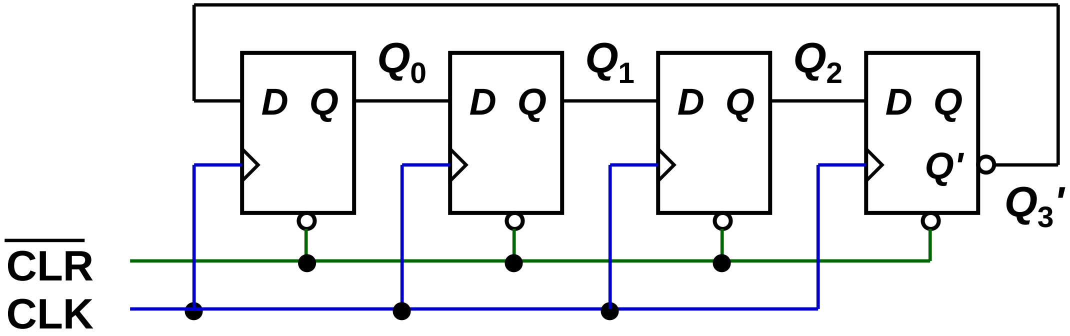
(b) Sequence of four timing signals



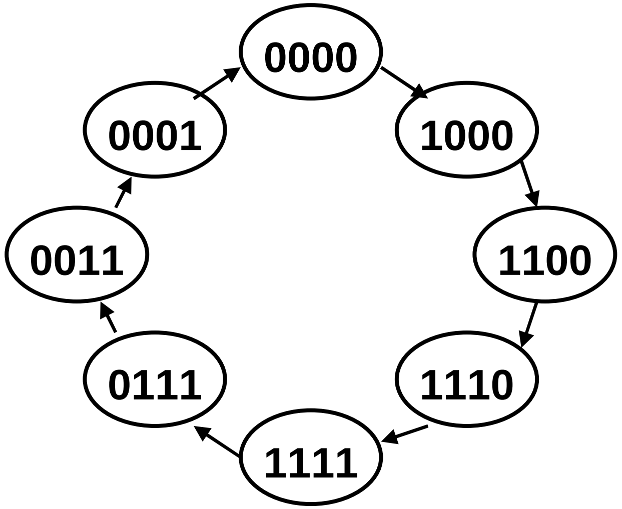
(c) Counter and decoder

Johnson Counters

- An *n-bit* Johnson counter (also called switch tail ring counter) cycles through 2*n* states.
- Example: A 4-bit John counter (also called mod-8 Johnson counter)



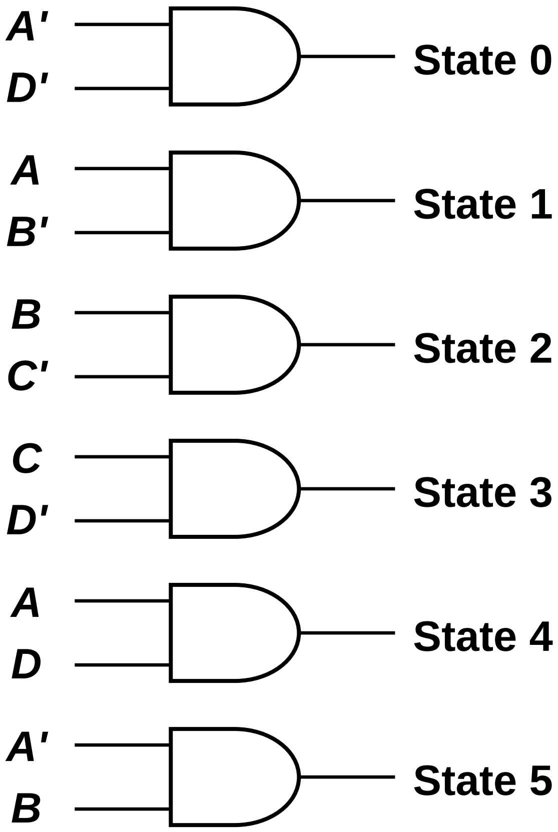
Clock	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

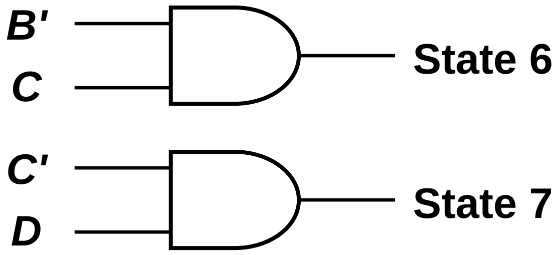


Johnson Counters

- Requires decoding logic for the states.
- Example: Decoding logic for a 4-bit Johnson counter.

Clock	*A*	*B*	*C*	*D*	Decoding
0	0	0	0	0	*A'.D'*
1	1	0	0	0	*A.B'*
2	1	1	0	0	*B.C'*
3	1	1	1	0	*C.D'*
4	1	1	1	1	*A.D*
5	0	1	1	1	*A'.B*
6	0	0	1	1	*B'.C*
7	0	0	0	1	*C'.D*


A'
D'
State 0
A
B'
State 1
B
C'
State 2
C
D'
State 3
A
D
State 4
A'
B
State 5


B'
C
State 6
C'
D
State 7