Sequential Circuits

By Dr. Arun Kishor Johar

Outline

- Introduction to Sequential Circuits
- Classification of Sequential Circuits
- Memory Elements
- Types of Flip Flop
- Applications of Flip Flops
- Conversion of Flip Flops
- Counters
- Asynchronous Counter Design
- Synchronous Counter Design

Introduction to Sequential Circuits

What is Sequential Circuit?

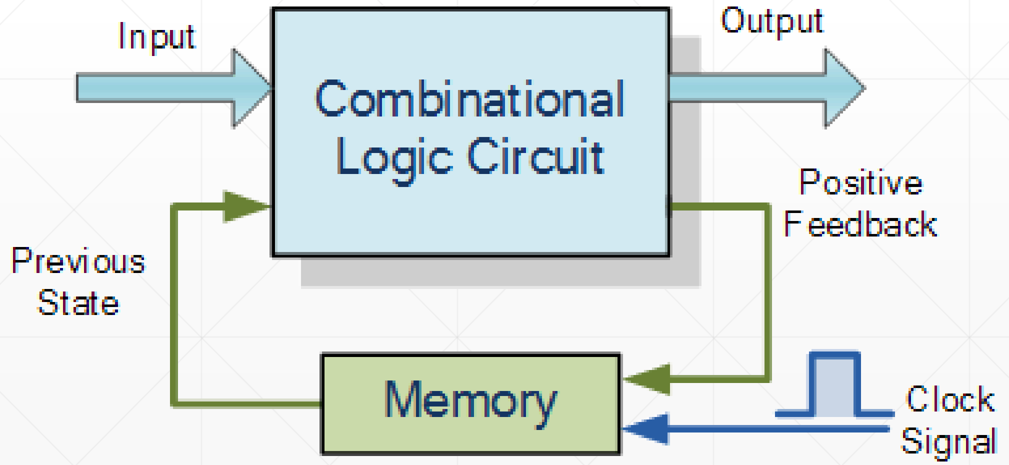
The outputs depend on the current and past input values

It uses logic gates and storage elements

Example

Vending machine

They are referred as finite state machines since they have a finite number of states



Sequential Circuits Types

Synchronous

- ❖ The circuit behavior is determined by the signals at discrete instants of time
- ❖ The memory elements are affected only at discrete instants of time
- ❖ A clock is used for synchronization
    - ❖ Memory elements are affected only with the arrival of a clock pulse
    - ❖ If memory elements use clock pulses in their inputs, the circuit is called
        - ❖ Clocked sequential circuit

Asynchronous

- ❖ The circuit behavior is determined by the signals at any instant of time
- ❖ It is also affected by the order the inputs change

Memory Elements

Flip-Flops

They are memory elements
They can store binary information
Can keep a binary state until an input signal to switch the state is received
There are different types of flip-flops depending on the number of inputs and how the inputs affect the binary state

Latches

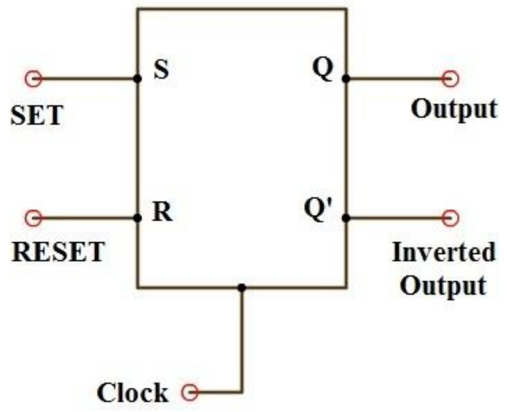
The most basic flip-flops and operated with signal levels
The flip-flops are constructed from latches
They are not useful for synchronous sequential circuits
They are useful for asynchronous sequential circuits

Types of Flip Flops

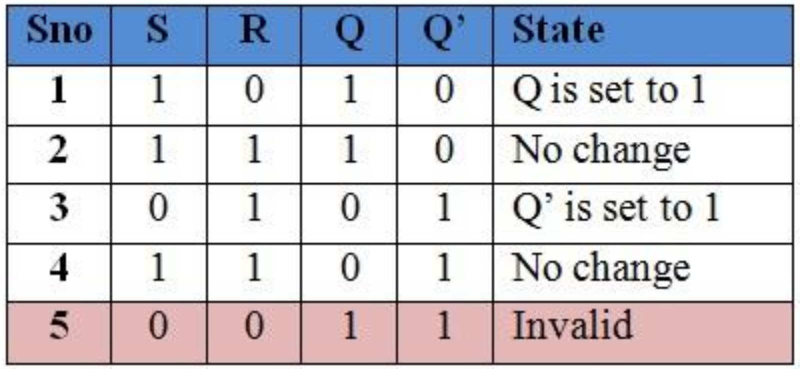
Based on their operations, flip flops are basically 4 types.

1. R-S flip flop
2. D flip flop
3. J-K flip flop
4. T flip flop

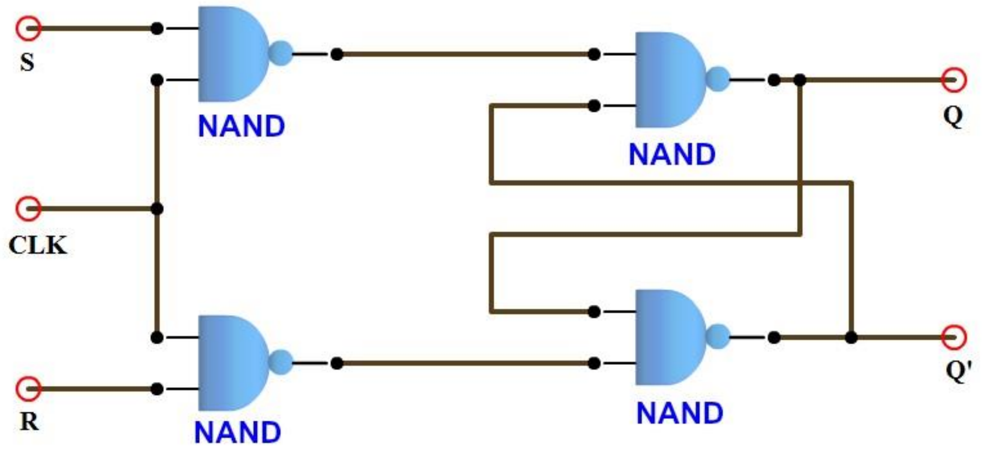
SR Flip Flop



Truth table



Sno	S	R	Q	Q'	State
1	1	0	1	0	Q is set to 1
2	1	1	1	0	No change
3	0	1	0	1	Q' is set to 1
4	1	1	0	1	No change
5	0	0	1	1	Invalid

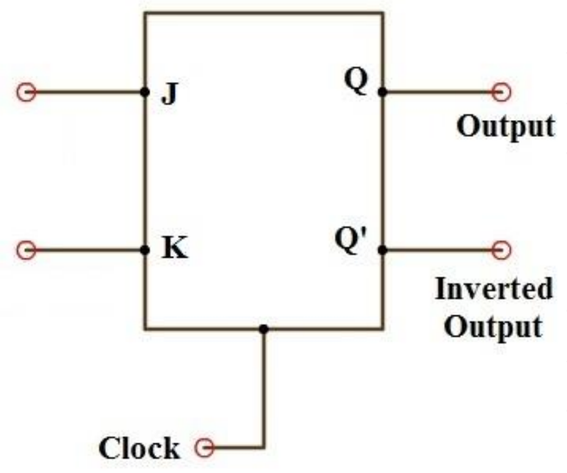


Working

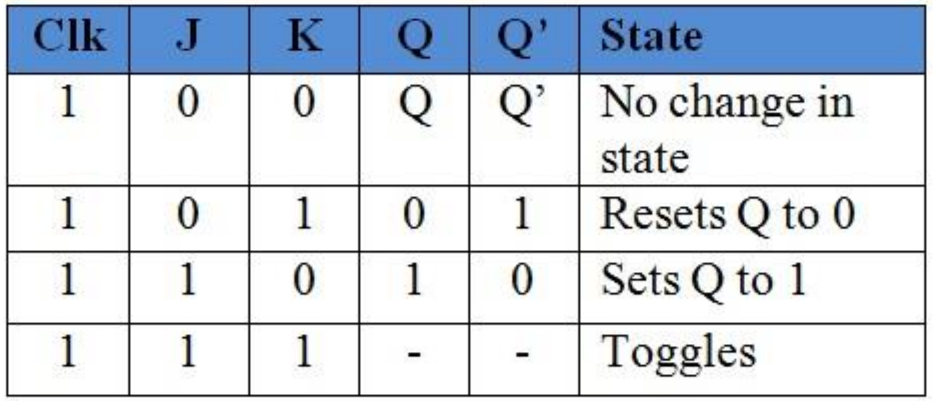
From the above truth table it is clear that SR flip flop will be set or reset for four conditions.

- For last condition it will be in invalid state.
- SR Flip-flop will be set when S=1 and R=0, if S=1 and R=1 then previous state is remembered by the flip flop.
- Flip-flop will be reset when S=0 and R=1, if S=1 and R=1, then it will remember the previous state.
- But when both the inputs are zeros, SR Flip flop will be in an uncertain state where both Q and Q' will be same. This is not same allowed..

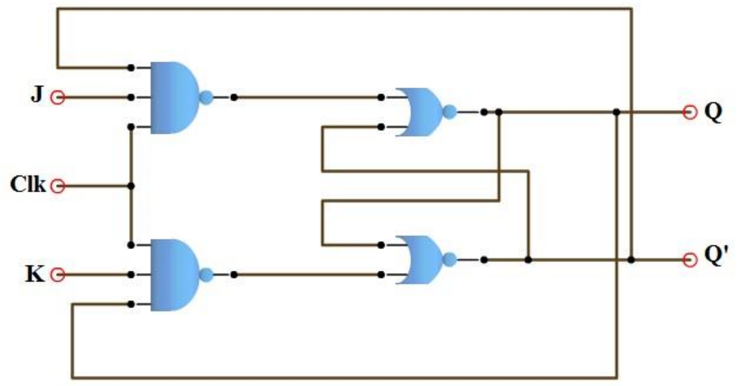
JK Flip Flop



Truth table



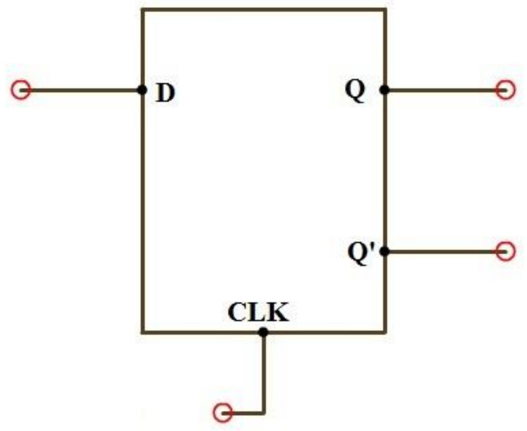
Clk	J	K	Q	Q'	State
1	0	0	Q	Q'	No change in state
1	0	1	0	1	Resets Q to 0
1	1	0	1	0	Sets Q to 1
1	1	1	-	-	Toggles



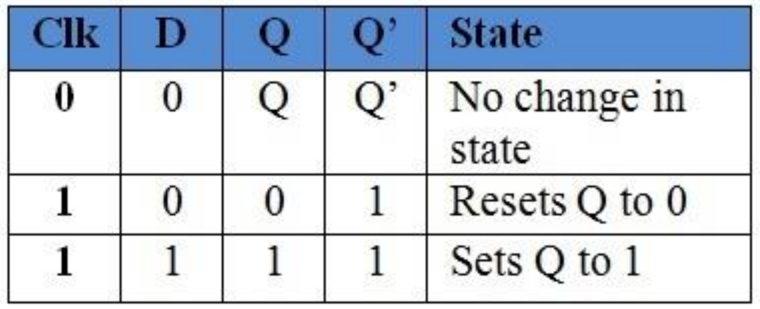
Working

- When J is low and K is low, then Q returns its previous state value i.e. it holds the current state.
- When J is low and K is high, then flip – flop will be in reset state i.e. Q = 0, Q' =1.
- When J is high and K is low then flip – flop will be in set state i.e. Q = 1, Q' =0.
- When J is high and K is high then flip – flop will be in Toggle state or flip state. This means that the output will complement to the previous state value.

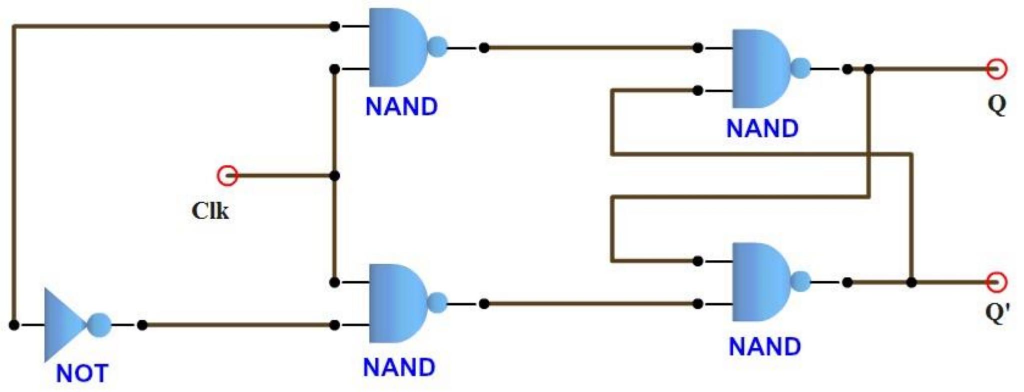
D Flip Flop



Truth table



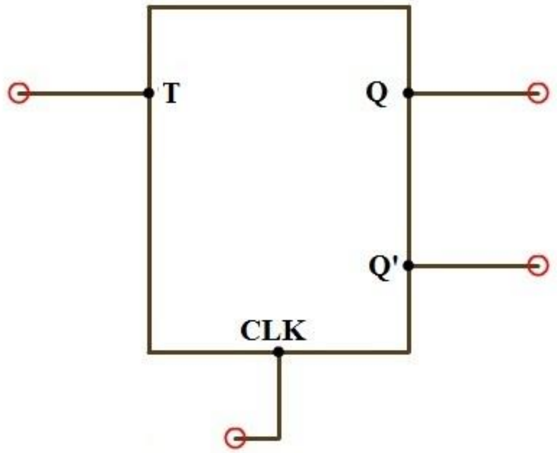
Clk	D	Q	Q'	State
0	0	Q	Q'	No change in state
1	0	0	1	Resets Q to 0
1	1	1	1	Sets Q to 1



Working

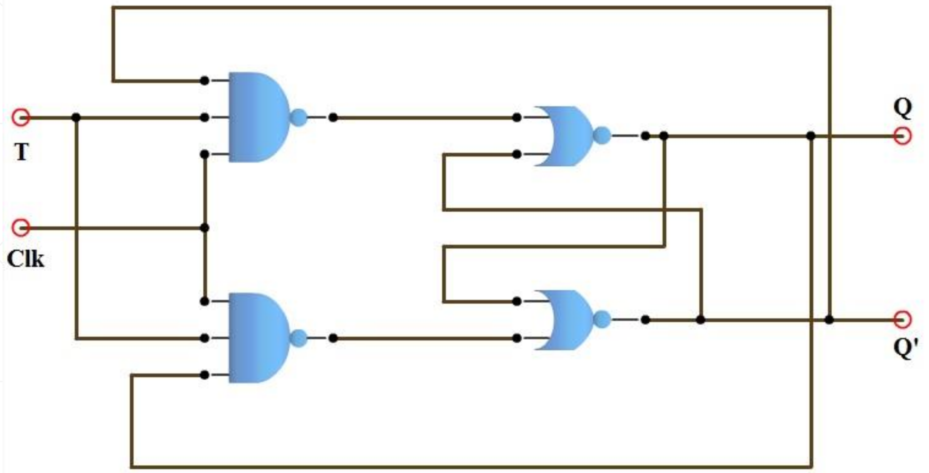
- D flip flop will work depending on the clock signal.
- When the clock is low there will be no change in the output of the flip flop i.e. it remembers the previous state.
- When the clock signal is high and if it receives any data on its data pin, it Changes the state of output.
- When data is high Q reset to 0,while Q is set to 0 if data is low.

T Flip Flop



Truth table

T	Q	Q'
0	0	0
1	0	1
0	1	0
1	1	0



Working

- When the T input is low, then the next sate of the T flip – flop is same as the present state i.e. it holds the current state.
- T = 0 and present state = 0 then the next state = 0.
- T = 0 and present state = 1 then the next state = 1.
- When the T input is high, then the next sate of the T flip – flop is toggled i.e. it is same as the complement of present state on clock transition.
- T = 1 and present state = 0 then the next state = 1.
- T = 1 and present state = 1 then the next state = 0.

Applications of Flip Flops

Flip flops are widely used in:

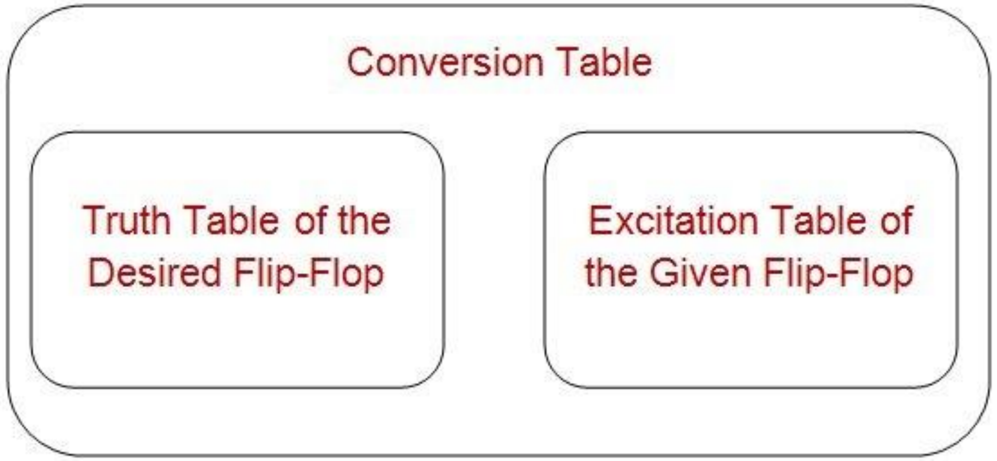
Registers: As the flip flops have two stable states, we use them in memory elements like registers, for data storage. Generally we use registers in electronic devices like computers.

Counters: The groups of interconnected flip flops are uses as counters, to count the increment or decrement of an event occurrence.

Frequency division: Flip flops are used as frequency division circuits, which divide the input frequency to exactly to its half. Frequency division circuits are used to regularize the frequency of electronic circuits.

Data transfer: We use shift registers (A special-type of registers) to transfer the data from one flip flop to another, which are connected in a specific order.

Conversion of Flip Flops


Conversion Table
Truth Table of the Desired Flip-Flop
Excitation Table of the Given Flip-Flop

Step 1: Write the Truth Table of the Desired Flip-Flop

Step 2: Obtain the Excitation Table for the given Flip-Flop from its Truth Table

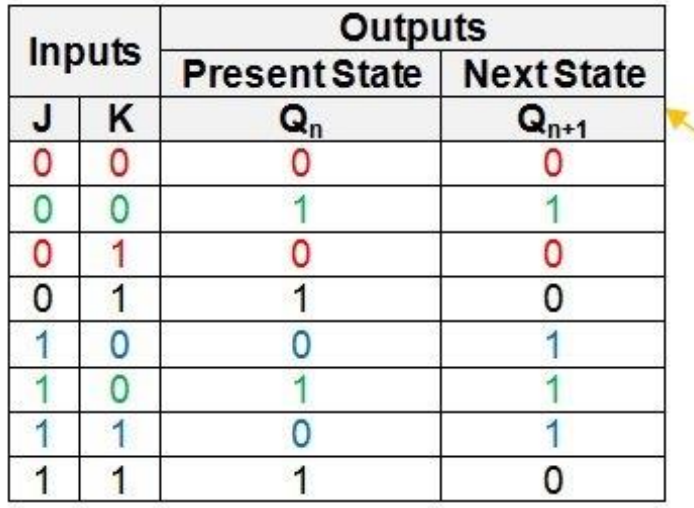
Step 3: Append the Excitation Table of the given Flip-Flop to the Truth Table of the Desired Flip-Flop Appropriately to obtain Conversion Table

Step 4: Simplify the Expressions for the Inputs of the given Flip-Flop

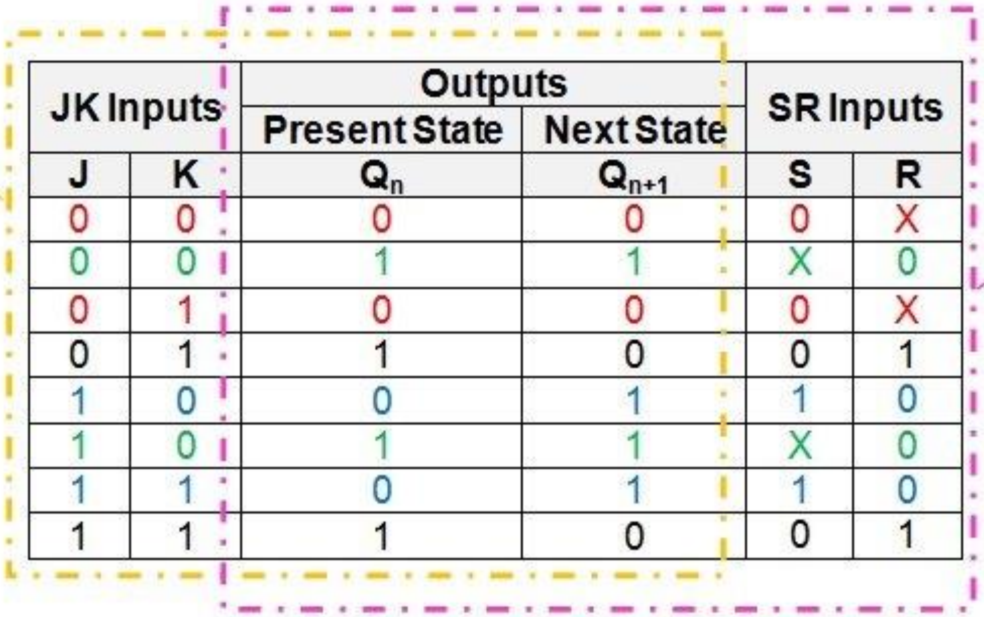
Step 5: Design the Necessary Circuit and make the Connections accordingly

Conversion of SR to JK Flip Flop

TruthTable of JK Flip-flop



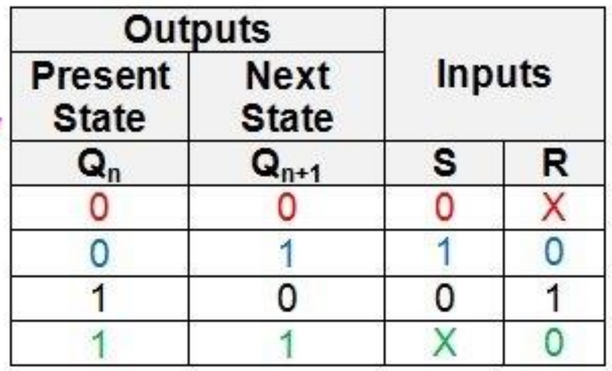
Inputs		Outputs	
		Present State	Next State
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



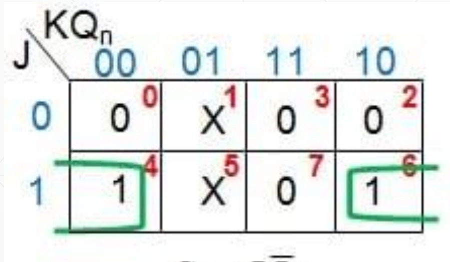
JK Inputs		Outputs		SR Inputs	
		Present State	Next State		
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

SR to JK Conversion Table

Excitation Table of SR Flip-flop

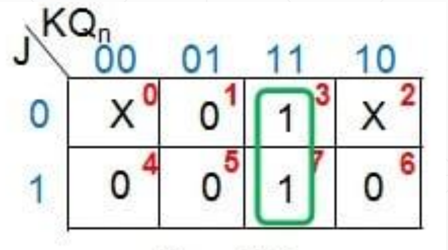


Outputs		Inputs	
Present State	Next State		
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0



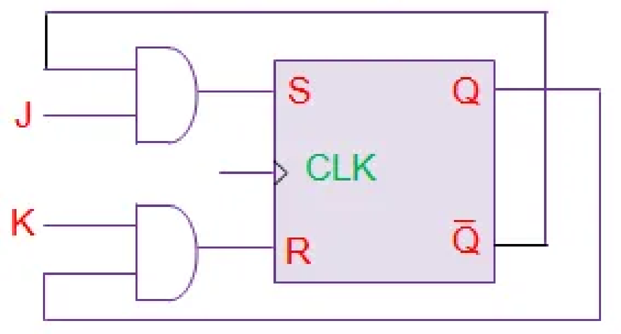
J \ KQ_n	00	01	11	10
0	0^{0}	X^{1}	0^{3}	0^{2}
1	1^{4}	X^{5}	0^{7}	1^{6}

$S = J\bar{Q}_n$



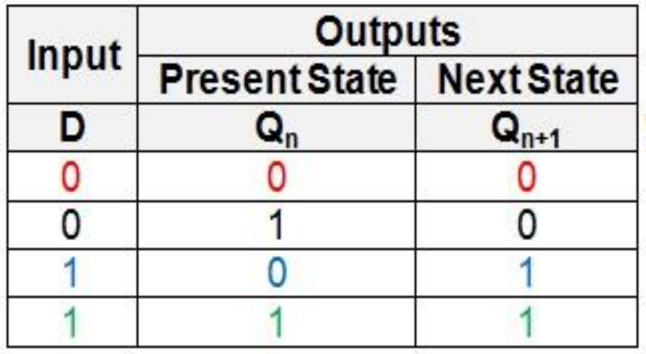
J \ KQ_n	00	01	11	10
0	X^{0}	0^{1}	1^{3}	X^{2}
1	0^{4}	0^{5}	1^{7}	0^{6}

$R = KQ_n$

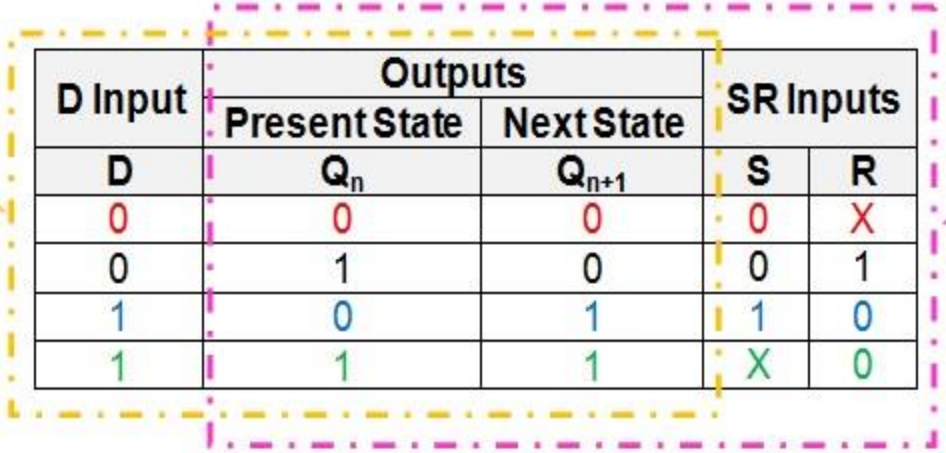


Conversion of SR to D Flip Flop

Truth Table of D Flip-flop



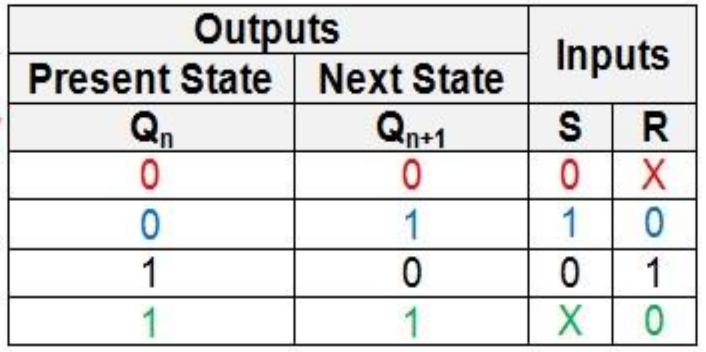
Input	Outputs	
	Present State	Next State
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1



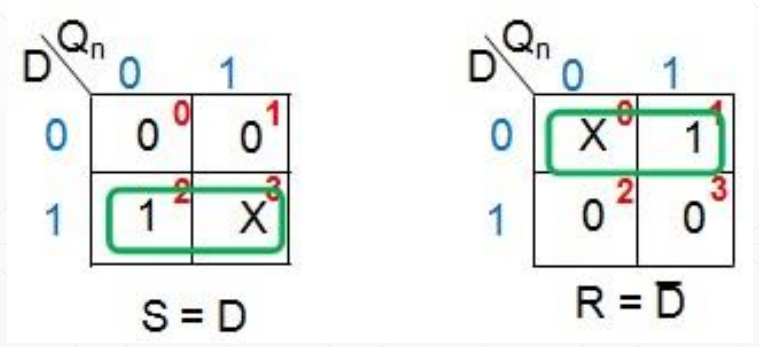
D Input	Outputs		SR Inputs	
	Present State	Next State		
D	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	0	0	1
1	0	1	1	0
1	1	1	X	0

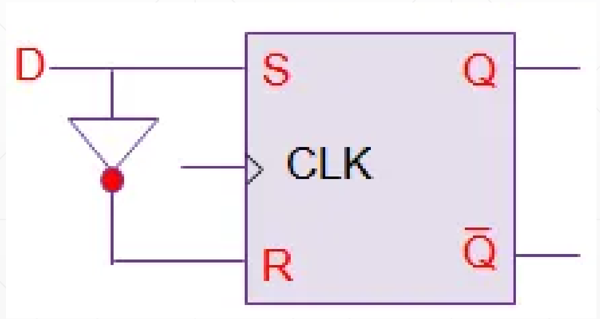
SR to D Conversion Table

Excitation Table of SR Flip-flop

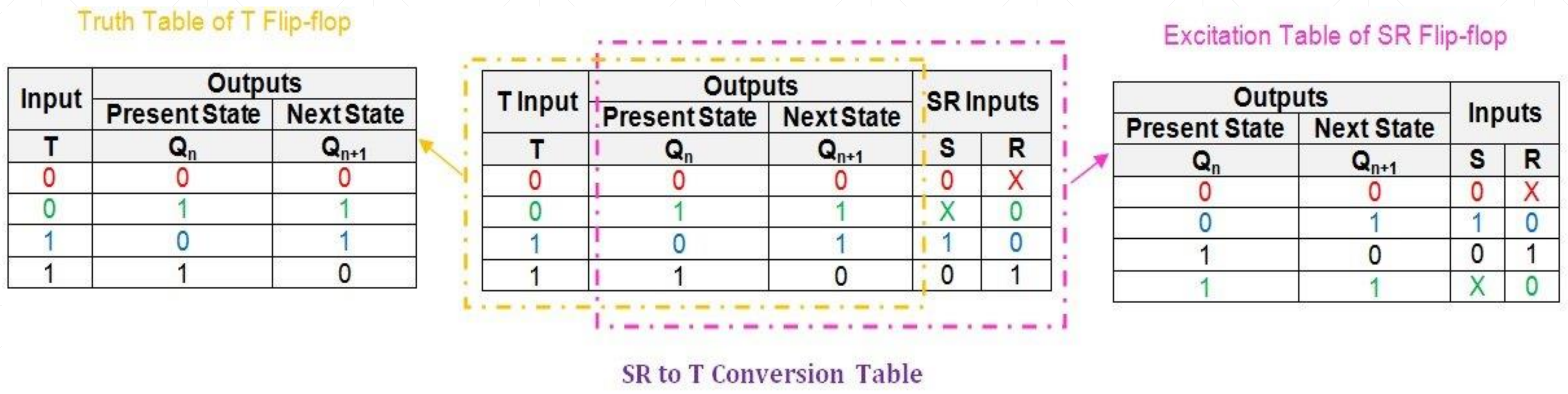


Outputs		Inputs	
Present State	Next State		
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0





Conversion of SR to T Flip Flop



Truth Table of T Flip-flop

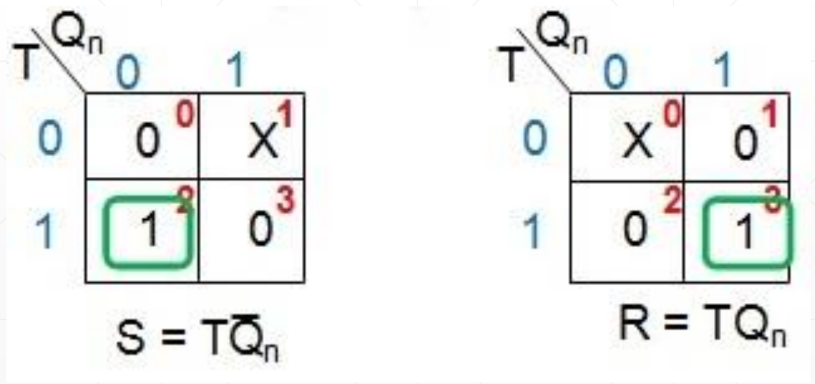
Input	Outputs	
	Present State	Next State
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

T Input	Outputs		SR Inputs	
	Present State	Next State		
T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1

SR to T Conversion Table

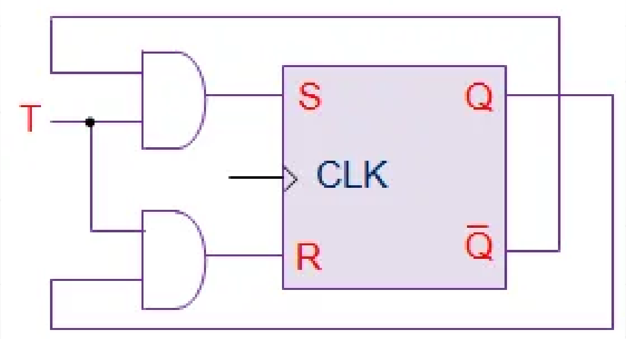
Excitation Table of SR Flip-flop

Outputs		Inputs	
Present State	Next State		
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0



$S = T\overline{Q}_n$

$R = TQ_n$



Conversion of D to JK Flip Flop

1. Truth Table for JK flip-flop

Inputs		Outputs	
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

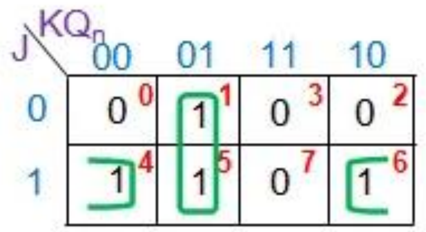
2. Excitation Table for D flip-flop

Outputs		Input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

3. Conversion Table

J	K	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0

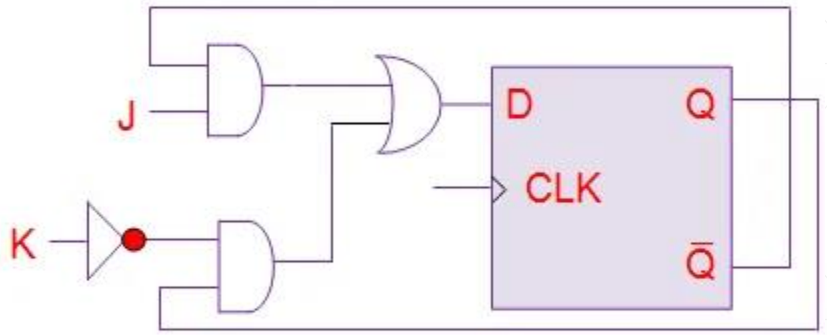
4. K-map Simplification



J \ KQ_n	00	01	11	10
0	0 [0]	1 [1]	0 [3]	0 [2]
1	1 [4]	1 [5]	0 [7]	1 [6]

$$D = J\bar{Q}_n + \bar{K}Q_n$$

5. Circuit Design



Conversion of D to SR Flip Flop

1. Truth Table for SR flip-flop

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	invalid	
1	1	invalid	

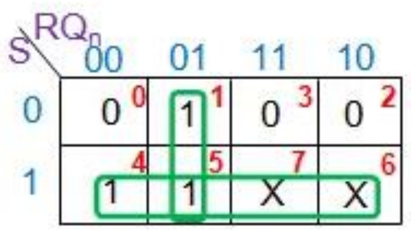
2. Excitation Table for D flip-flop

Outputs		Input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

3. Conversion Table

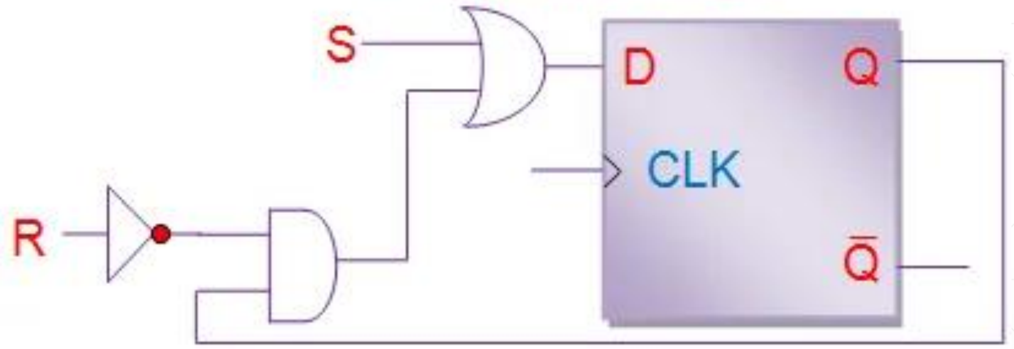
S	R	Q_n	Q_{n+1}	D
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	invalid		X
1	1	invalid		X

4. K-map Simplification



$$D = S + \bar{R}Q_n$$

5. Circuit Design



Conversion of D to T Flip Flop

1. Truth Table for T Flip Flop

Input	Outputs	
T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

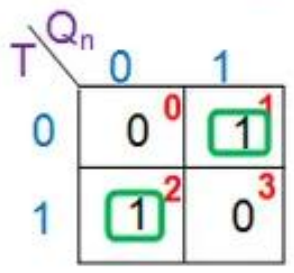
2. Excitation Table for D Flip Flop

Outputs		Input
Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

3. Conversion Table

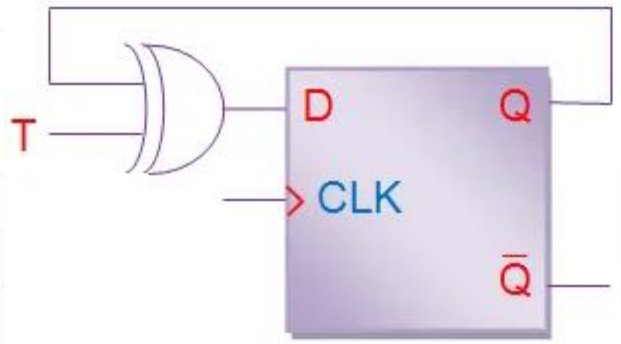
T	Q_n	Q_{n+1}	D
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

4. K-map Simplification



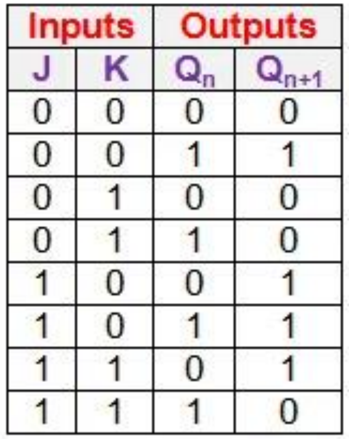
$$D = T\overline{Q}_n + \overline{T}Q_n$$
$$= T \oplus Q_n$$

5. Circuit Design



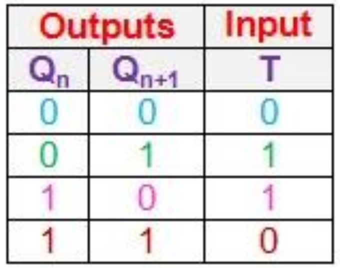
Conversion of T to JK Flip Flop

1. Truth Table for JK Flip Flop



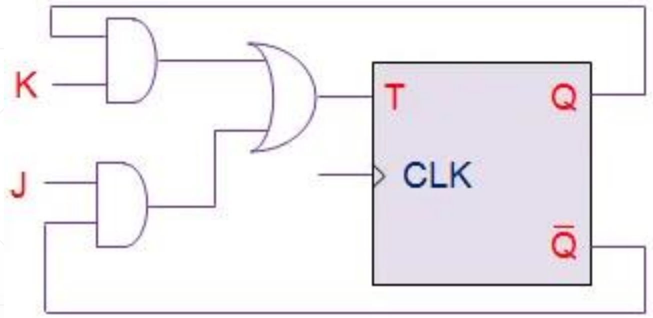
Inputs		Outputs	
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

2. Excitation Table for T Flip Flop

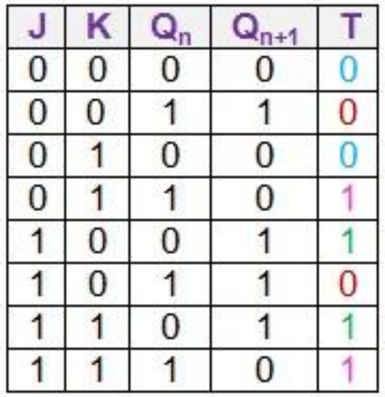


Outputs		Input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

5. Circuit Design

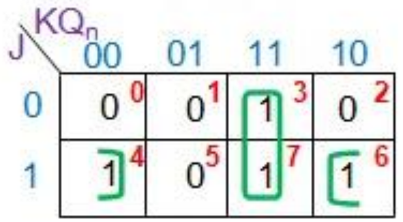


3. Conversion Table



J	K	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	1

4. K-map Simplification



J \ KQ_n	00	01	11	10
0	0 [0]	0 [1]	1 [3]	0 [2]
1	1 [4]	0 [5]	1 [7]	1 [6]

$$T = J\bar{Q}_n + KQ_n$$

Conversion of T to SR Flip Flop

1. Truth Table for SR Flip Flop

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	invalid	
1	1	invalid	

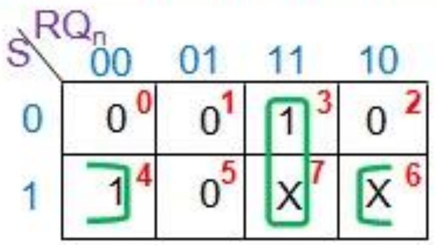
2. Excitation Table for T Flip Flop

Outputs		Input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

3. Conversion Table

S	R	Q_n	Q_{n+1}	T
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	invalid		X
1	1	invalid		X

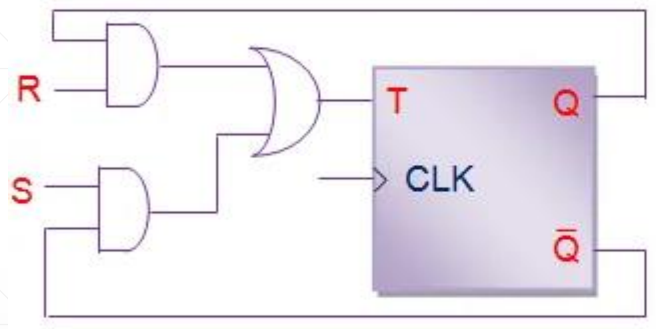
4. K-map Simplification



S \ RQ_n	00	01	11	10
0	0 [0]	0 [1]	1 [3]	0 [2]
1	1 [4]	0 [5]	X [7]	X [6]

$$T = S\bar{Q}_n + RQ_n$$

5. Circuit Design



Conversion of T to SR Flip Flop

1. Truth Table for D Flip Flop

Input	Outputs	
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

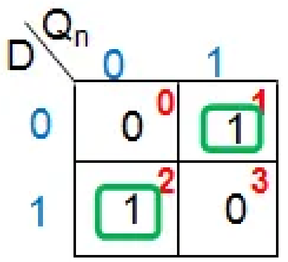
2. Excitation Table for T Flip Flop

Outputs		Input
Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

3. Conversion Table

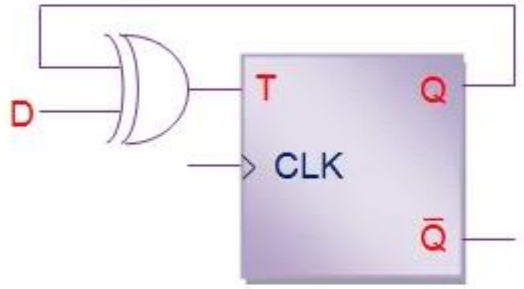
D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

4. K-map Simplification



$$D = D\bar{Q}_n + \bar{D}Q_n$$
$$= D \oplus Q_n$$

5. Circuit Design



Counters

By Dr. Arun Kishor Johar

Introduction

- Counter is a digital device and the output of the counter includes a predefined state based on the clock pulse applications.
- The output of the counter can be used to count the number of pulses.
- Two types of counter
  - Synchronous counter (e.g. parallel)
  - Asynchronous counter (e.g. ripple)
- Ripple counter let some flip-flop output to be used as clock signal source for other flip-flop
- Synchronous counter use the same clock signal for all flip-flop

Asynchronous Counters

Asynchronous Counters

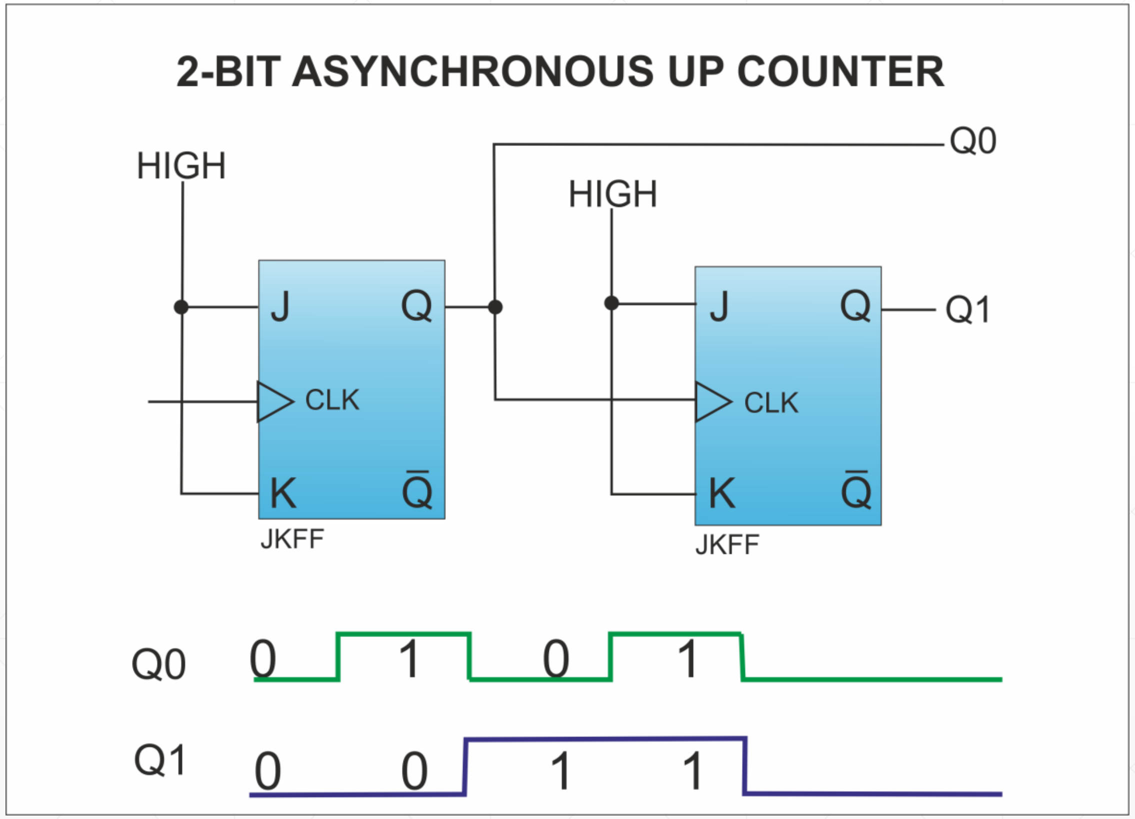
- Only the first flip-flop is clocked by an external clock. All subsequent flip-flops are clocked by the output of the preceding flip-flop.

- Asynchronous counters are slower than synchronous counters because of the delay in the transmission of the pulses from flip-flop to flip-flop.

- Asynchronous counters are also called *ripple-counters* because of the way the clock pulse ripples it way through the flip-flops.

Asynchronous Counters

- Example: 2-bit ripple counter
- Output from one flip-flop is connected to clock input for the next flip-flop MSB



States / Modulus / Flip-Flops

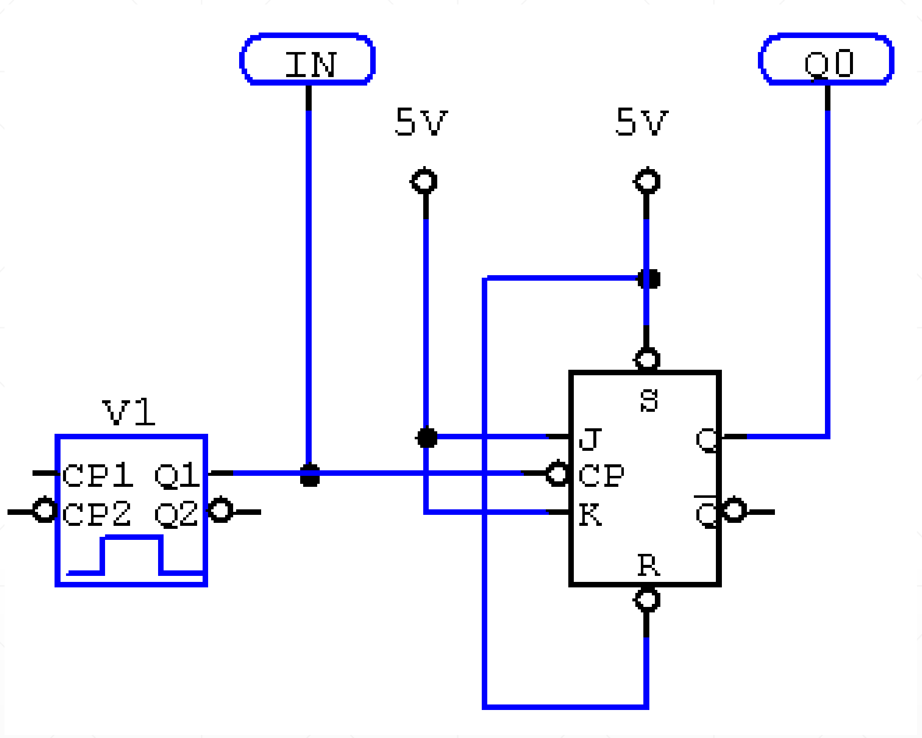
- The number of flip-flops determines the count limit or number of states.

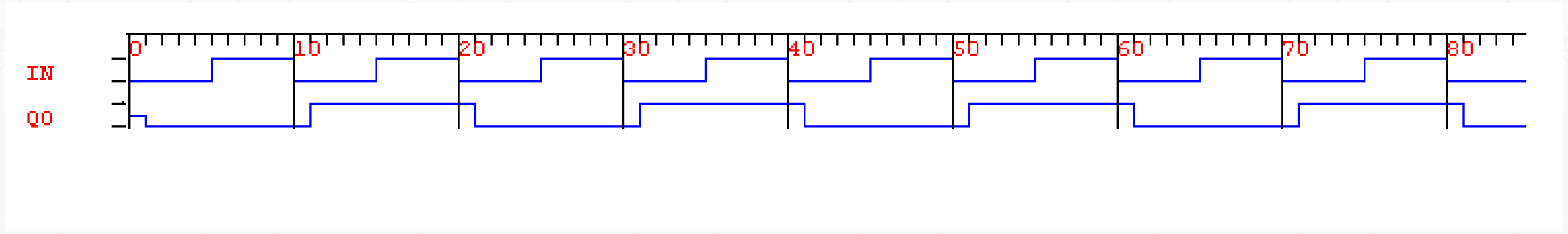
  (STATES = $2^{\textit{\# of flip flops}}$)

- The number of states used is called the *MODULUS.*

- For example, a Modulus-12 counter would count from 0 (0000) to 11 (1011) and requires four flip-flops (16 states - 12 used).

1 Bit Asynchronous-Counter / Modulus 2





Asynchronous Counters

Advantages

- Asynchronous counters can be easily designed by T flip flop or D flip flop.
- These are also called as Ripple counters, and are used in low speed circuits.
- They are used as Divide by- n counters, which divide the input by n, where n is an integer.
- Asynchronous counters are also used as Truncated counters. These can be used to design any mod number counters, i.e. even Mod (ex: mod 4) or odd Mod (ex: mod3).

Disadvantages

- Sometimes extra flip flop may be required for “Re synchronization”.
- To count the sequence of truncated counters (mod is not equal to 2n), we need additional feedback logic.
- While counting large number of bits, the propagation delay of asynchronous counters is very large.
- For high clock frequencies, counting errors may occur, due to propagation delay.

Asynchronous Counters

Applications of Asynchronous Counters

- Asynchronous counters are used as frequency dividers, as divide by N counters.
- These are used for low power applications and low noise emission.
- These are used in designing asynchronous decade counter.
- Also used in Ring counter and Johnson counter.
- Asynchronous counters are used in Mod N ripple counters. EX: Mod 3, Mod 4, Mod 8, Mod 14, Mod 10 etc.

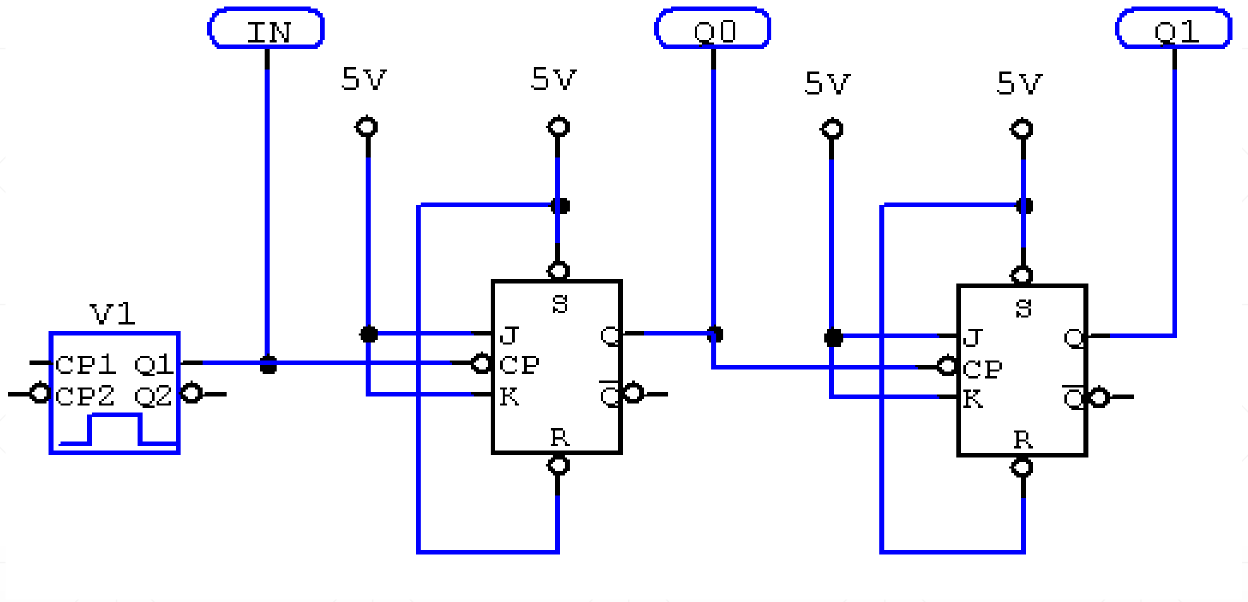
Asynchronous Counter Design Steps

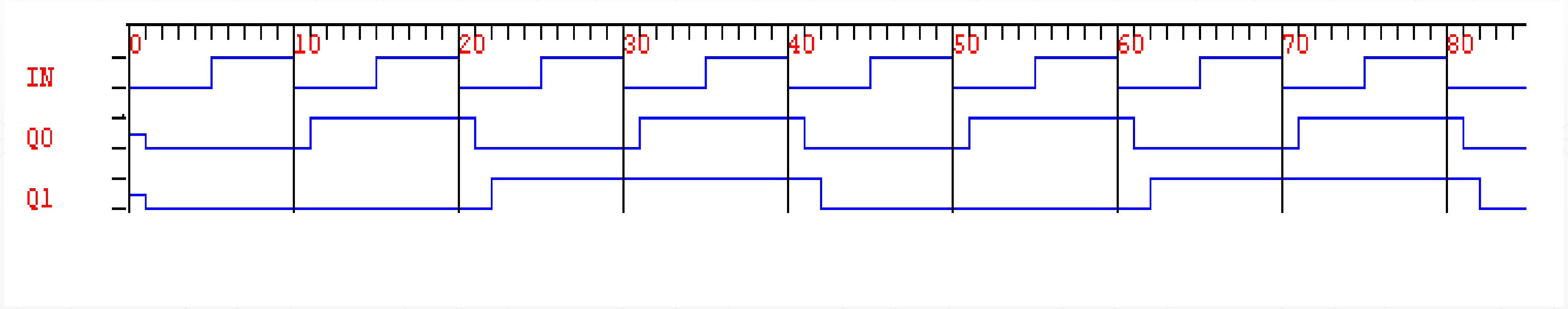
1. Select Type
   - Up or Down
   - Modules
2. Select Flip-Flop Type
   - J-K or D
   - Positive Edge Trigger (PET) or Negative Edge Trigger (NET)
3. Determine Number of Flip-Flops
   - ($2^{\#\ Flip\text{-}Flop} \geq$ Modules)
4. Design Basic Counters
   - Same polarity for down counters: $Q \rightarrow \text{PET or } \overline{Q} \rightarrow \text{NET}$
   - Opposite polarity for up counters: $Q \rightarrow \text{NET or } \overline{Q} \rightarrow \text{PET}$
5. Design Limits Logic
   - Input to logic is count that is one past the end of sequence.

Design Example

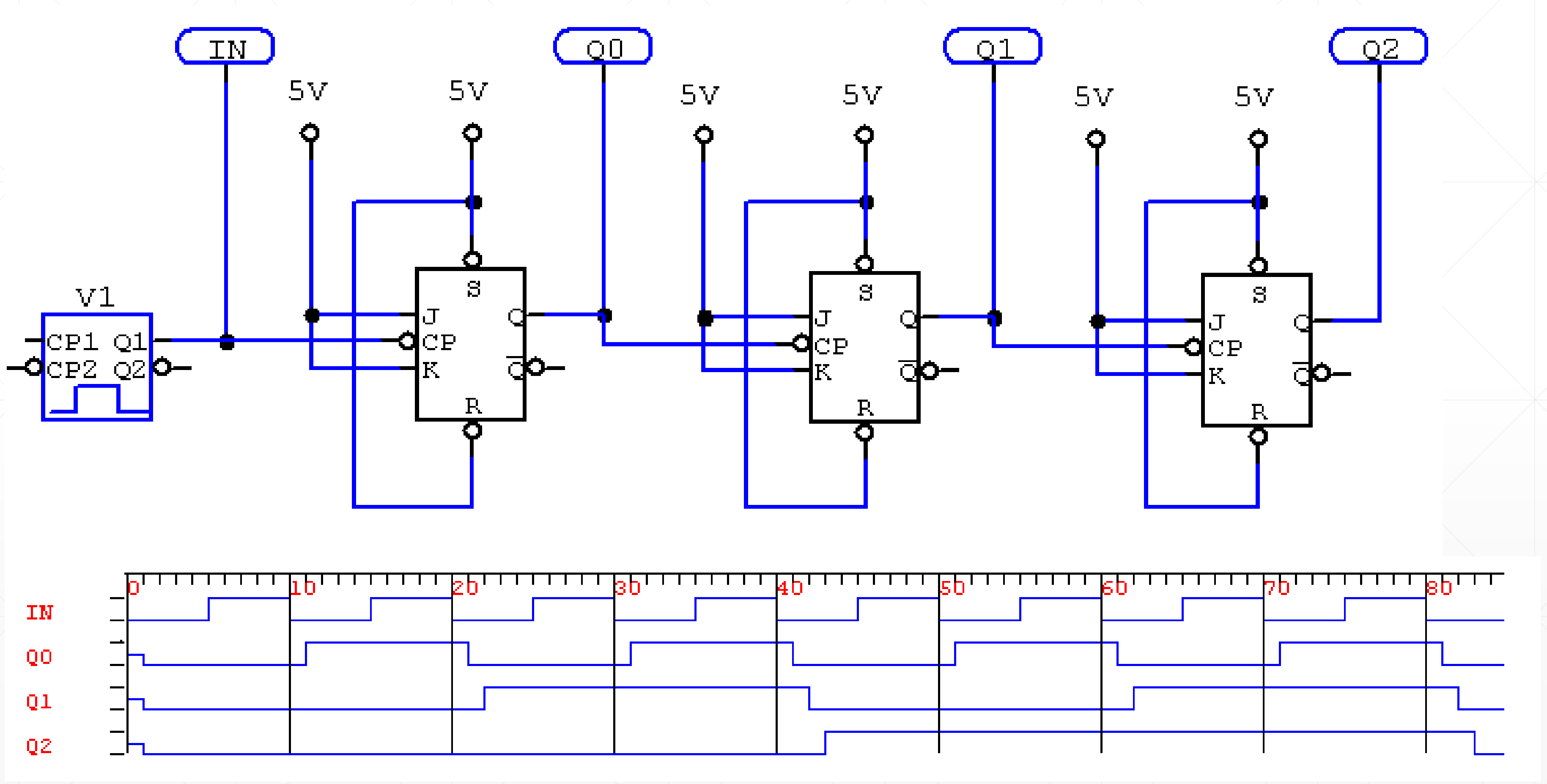
1. Select Type
   - $\boxed{\text{Up}}$ or Down
   - Modules $\boxed{\text{MOD – 14 (0..13)}}$
2. Select Flip-Flop Type
   - J-K or $\boxed{\text{D}}$
   - $\boxed{\text{Positive Edge Trigger (PET)}}$ or Negative Edge Trigger (NET)
3. Determine Number of Flip-Flops
   - $(2^{\#\,\textit{Flip-Flop}} \geq \text{Modules})$ $\boxed{2^{4\ \text{Flip-Flop}} \geq 16}$
4. Design Basic Counters
   - Same polarity for down counters: $Q \to \text{PET}$ or $\overline{Q} \to \text{NET}$
   - Opposite polarity for up counters: $Q \to \text{NET}$ or $\boxed{\overline{Q} \to \text{PET}}$
5. Design Limits Logic
   - Input to logic is count that is one past the end of sequence $\boxed{\text{Limit } 13+1 = 14\ (1110)}$

2 Bit Asynchronous Counter / Modulus 4

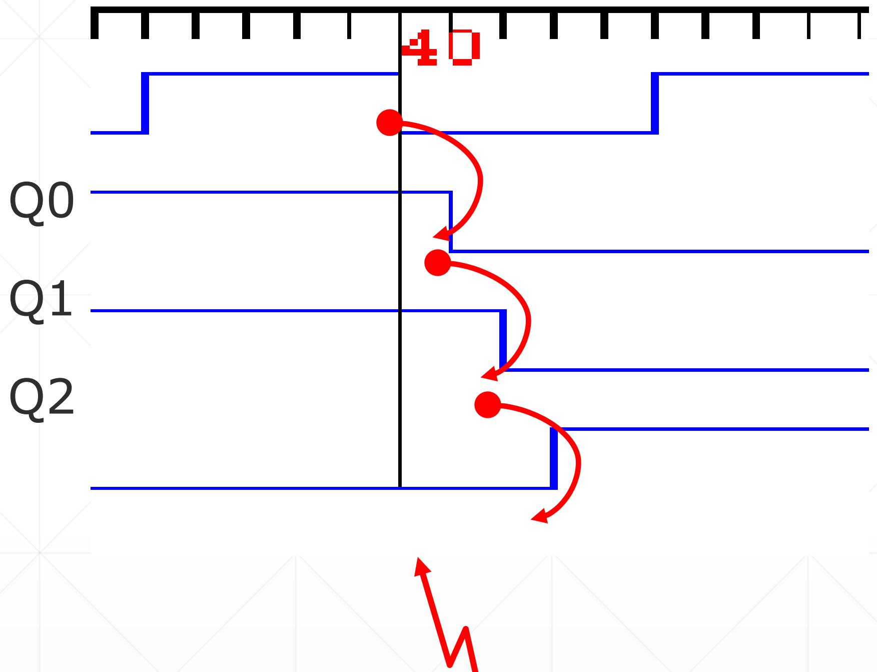


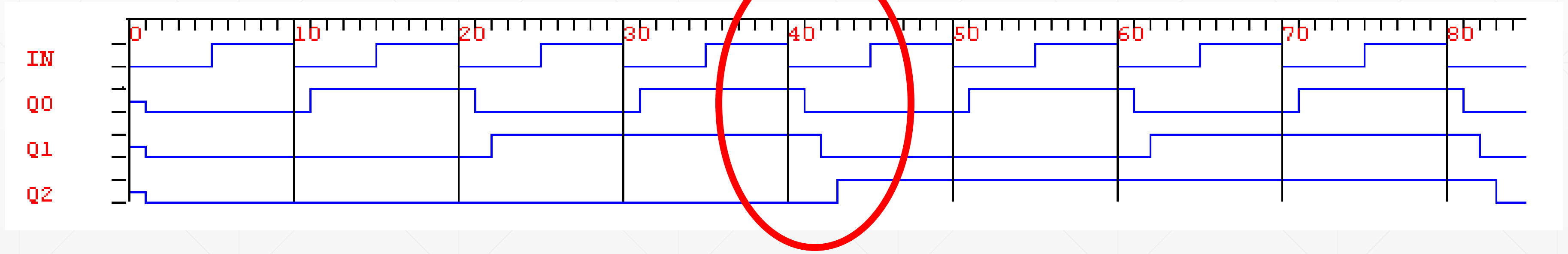


3 Bit Asynchronous Counter / Modulus 8

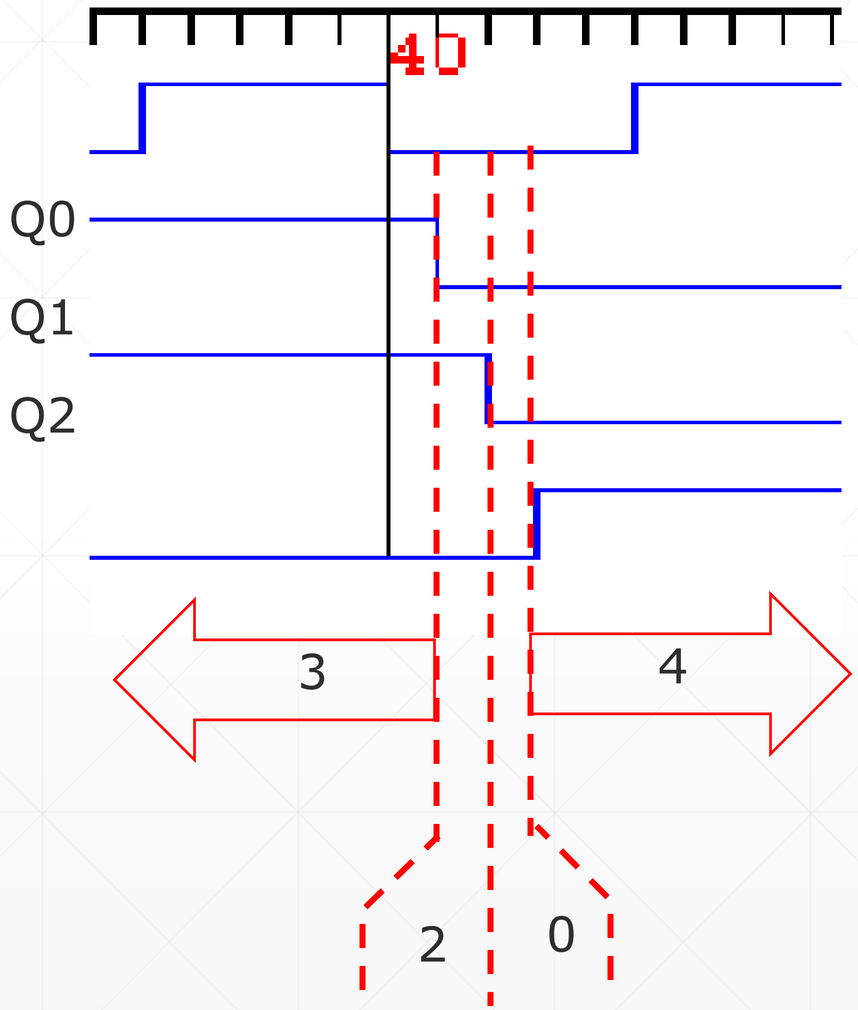


The Ripple Effect…





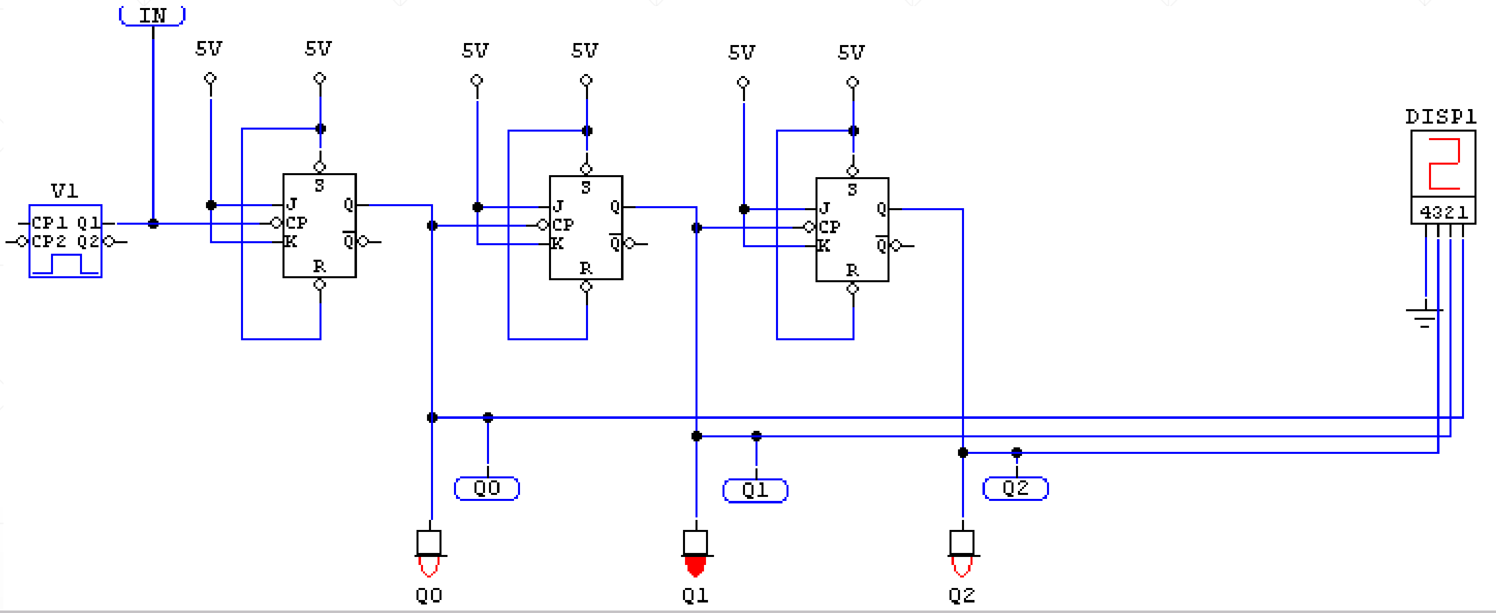
Ripple Effect…The Problem

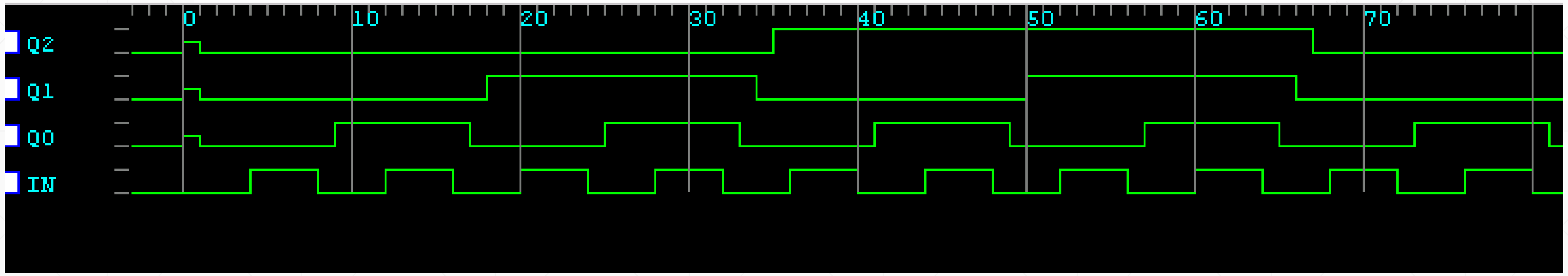


Six Examples

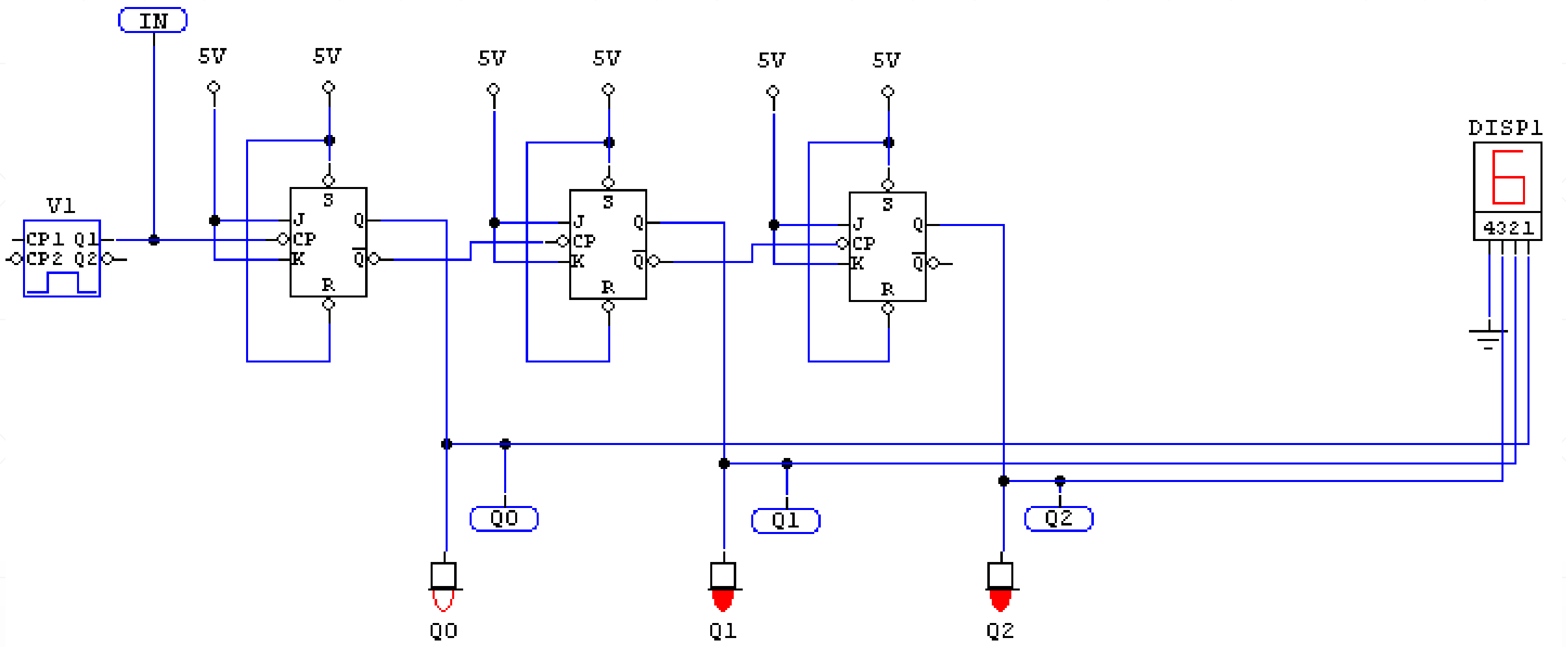
1. Modulus 4 Up Counter with Negative Edge Triggered Flip-Flops
2. Modulus 4 Down Counter with Negative Edge Triggered Flip-Flops
3. Modulus 4 Up Counter with Positive Edge Triggered Flip-Flops
4. Modulus 4 Down Counter with Positive Edge Triggered Flip-Flops
5. Truncated Counter
6. Counter Design

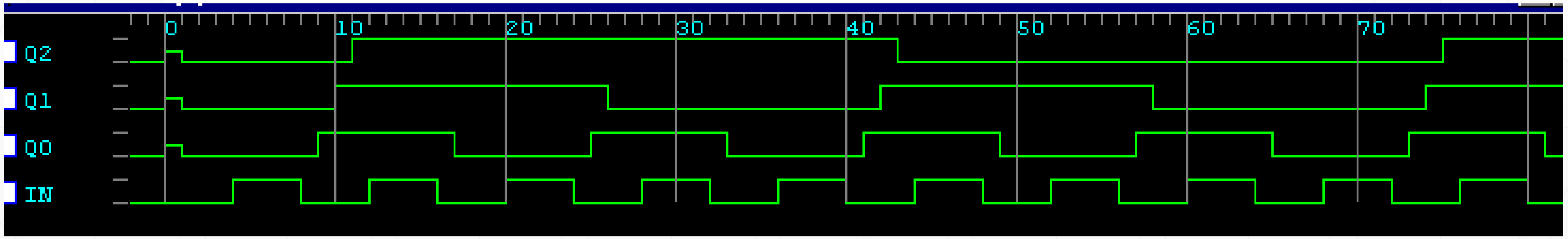
Up Counter with Negative Edge Triggered Flip-Flops



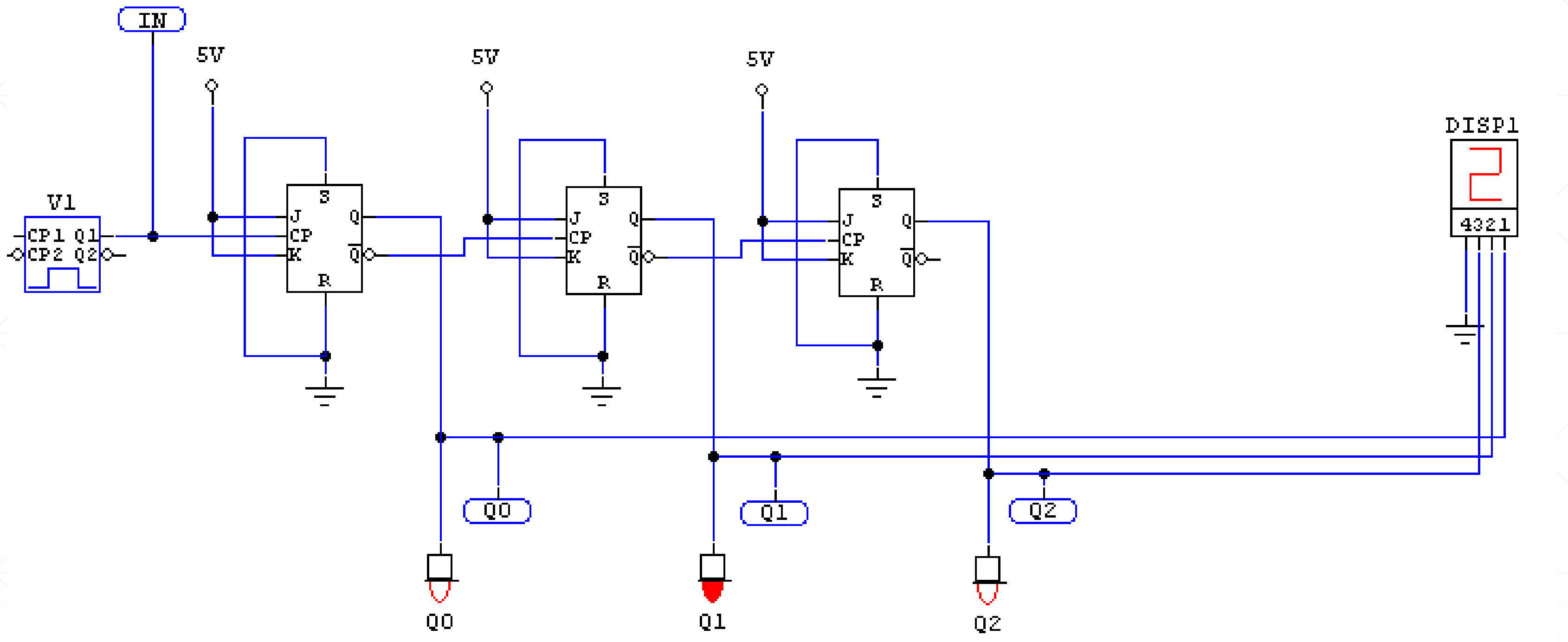


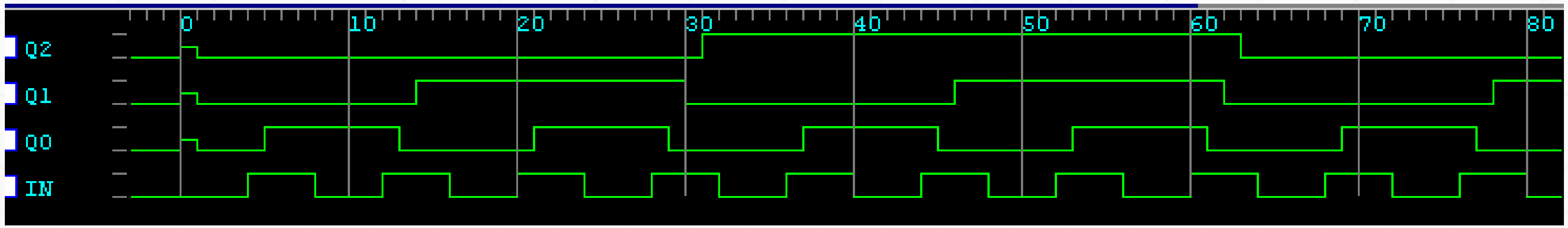
Down Counter with Negative Edge Triggered Flip-Flops



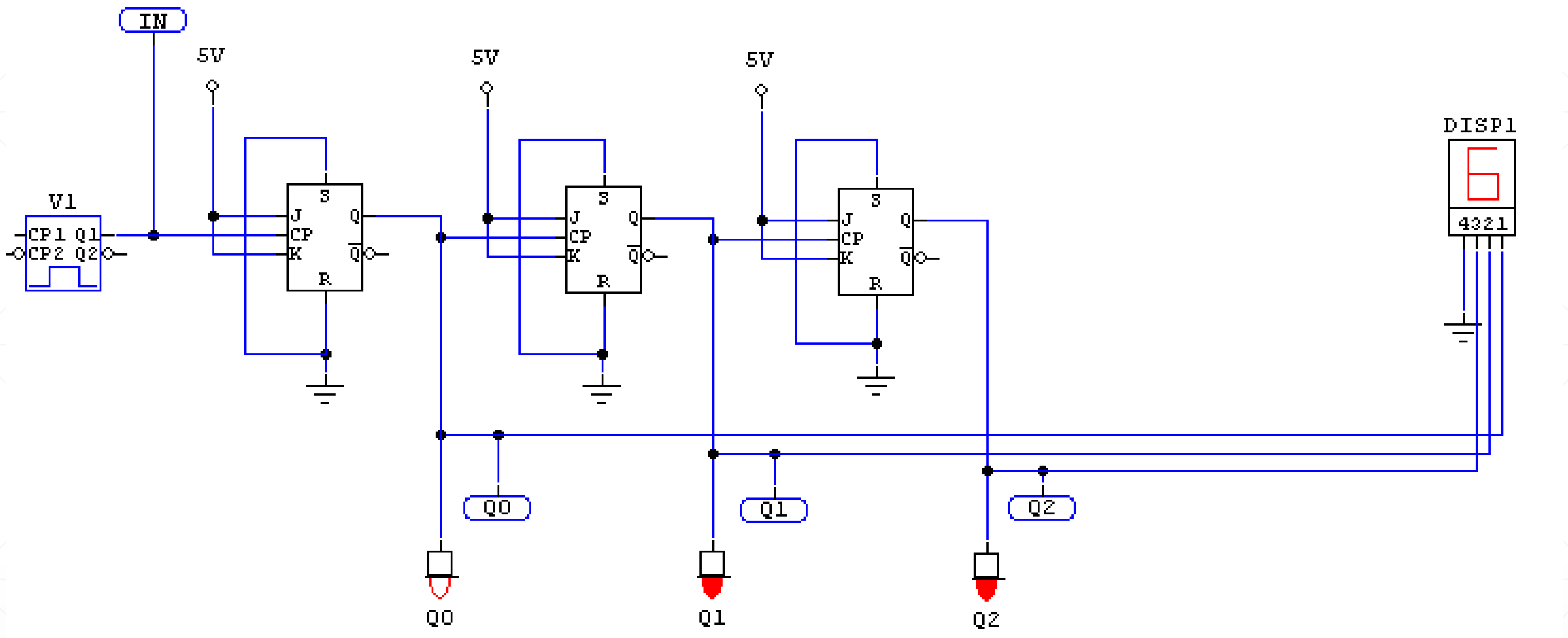


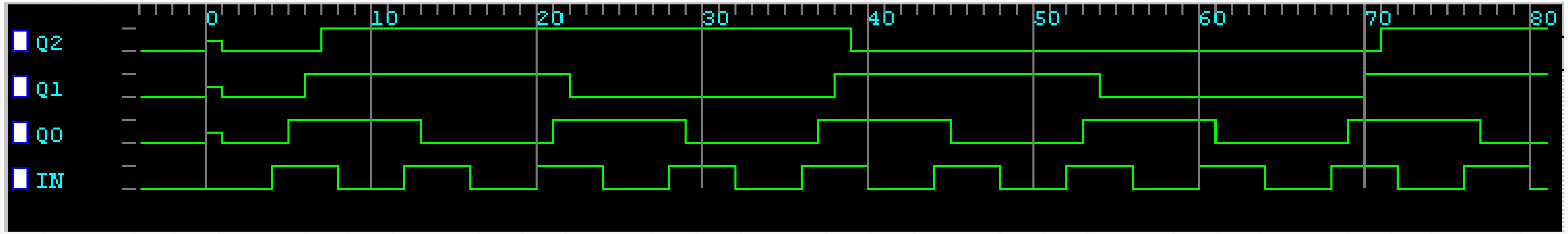
Up Counter with Positive Edge Triggered Flip-Flops



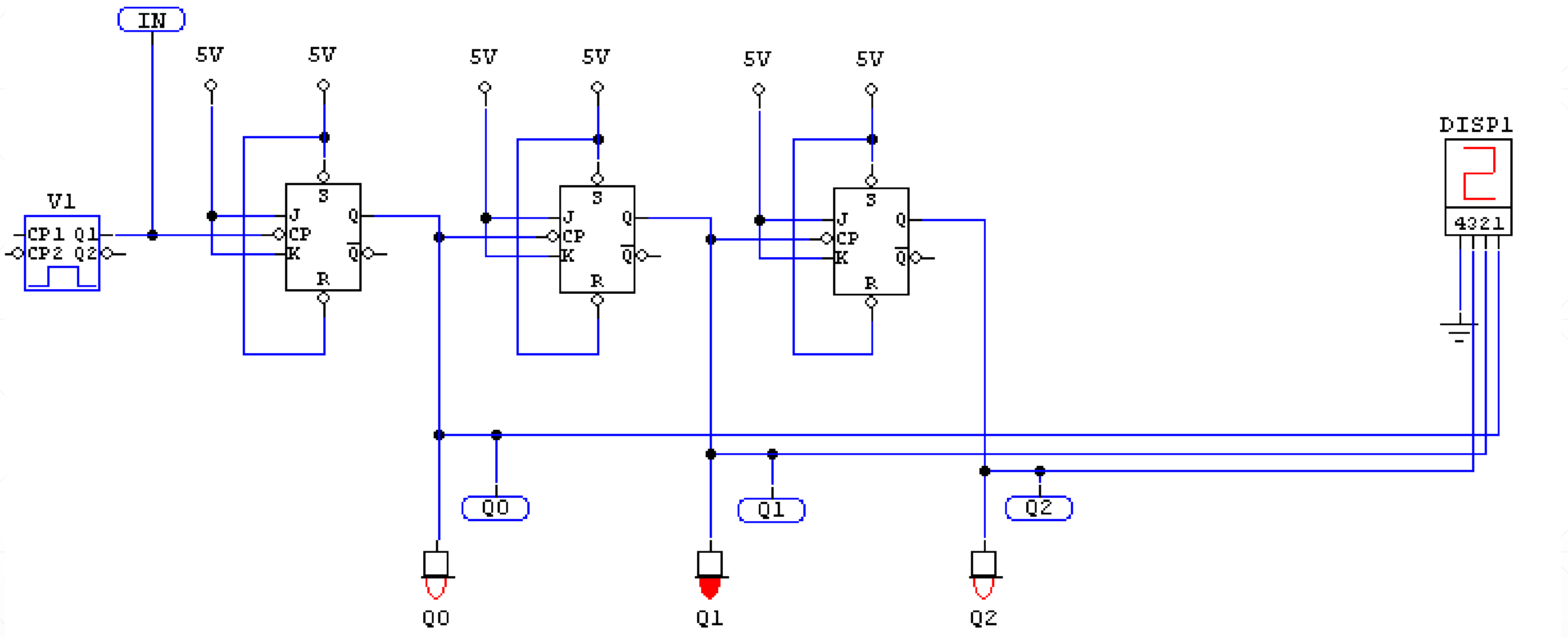


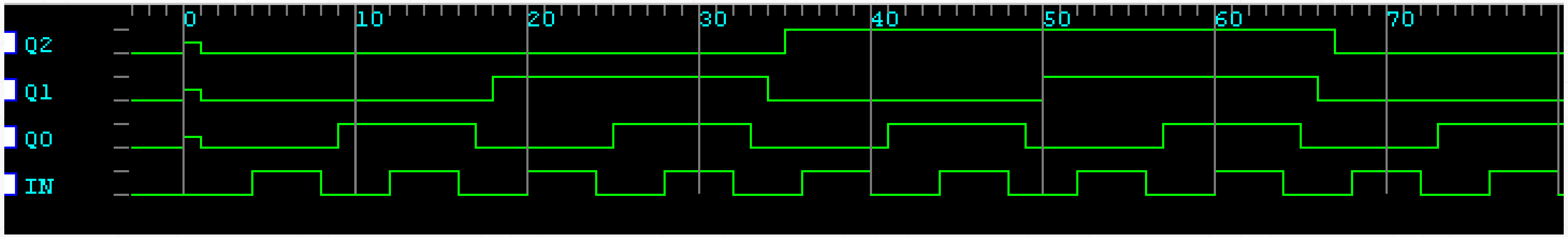
Down Counter with Positive Edge Triggered Flip-Flops



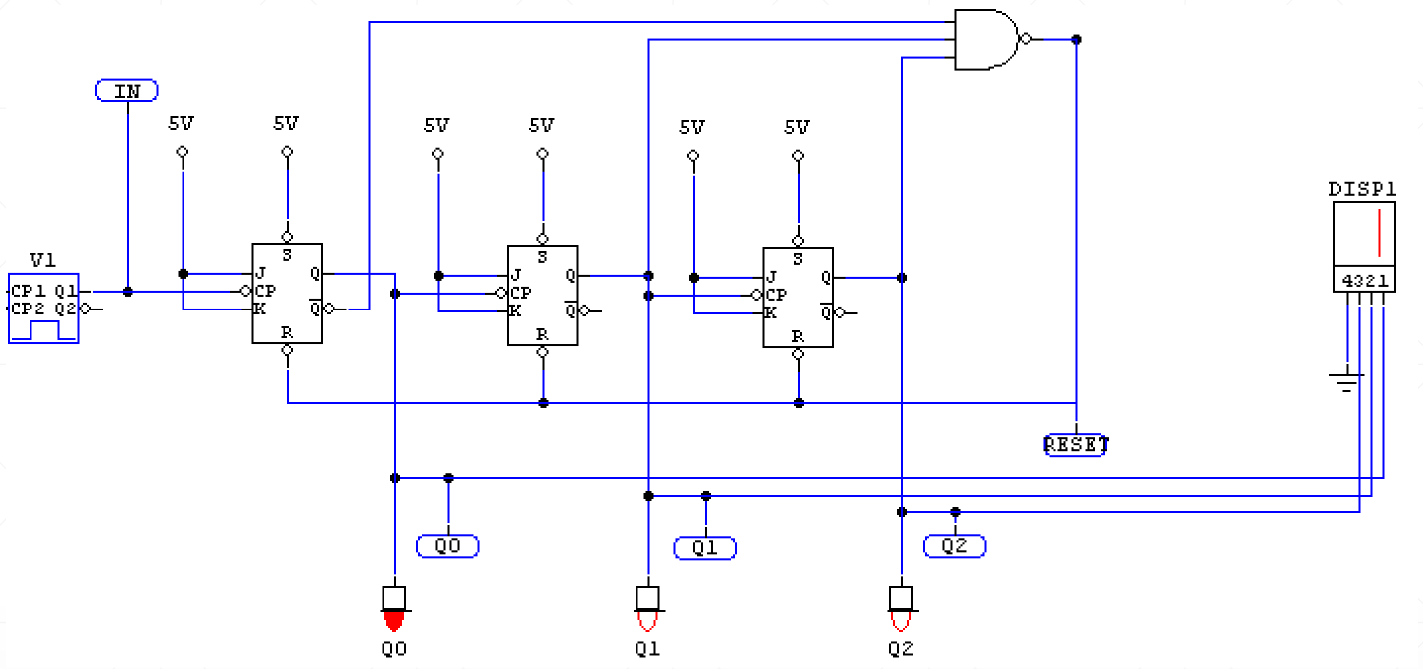


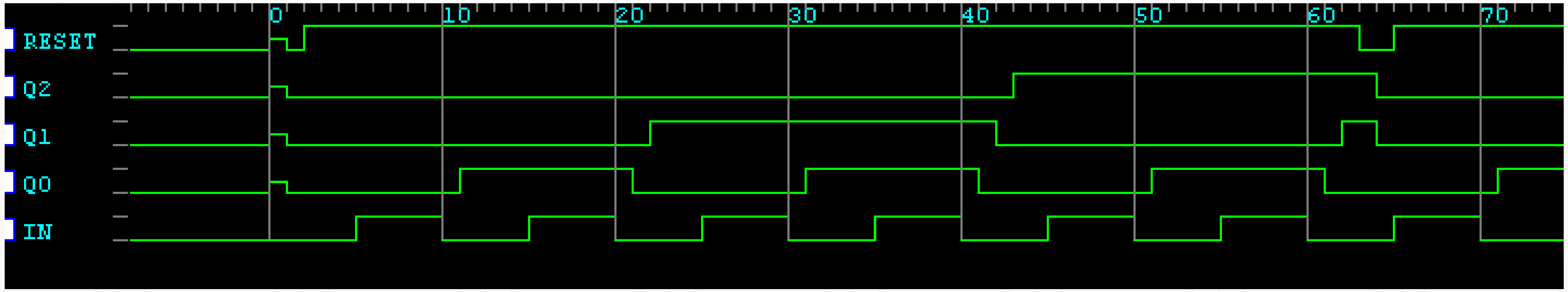
Truncating the Count… Modulus 6



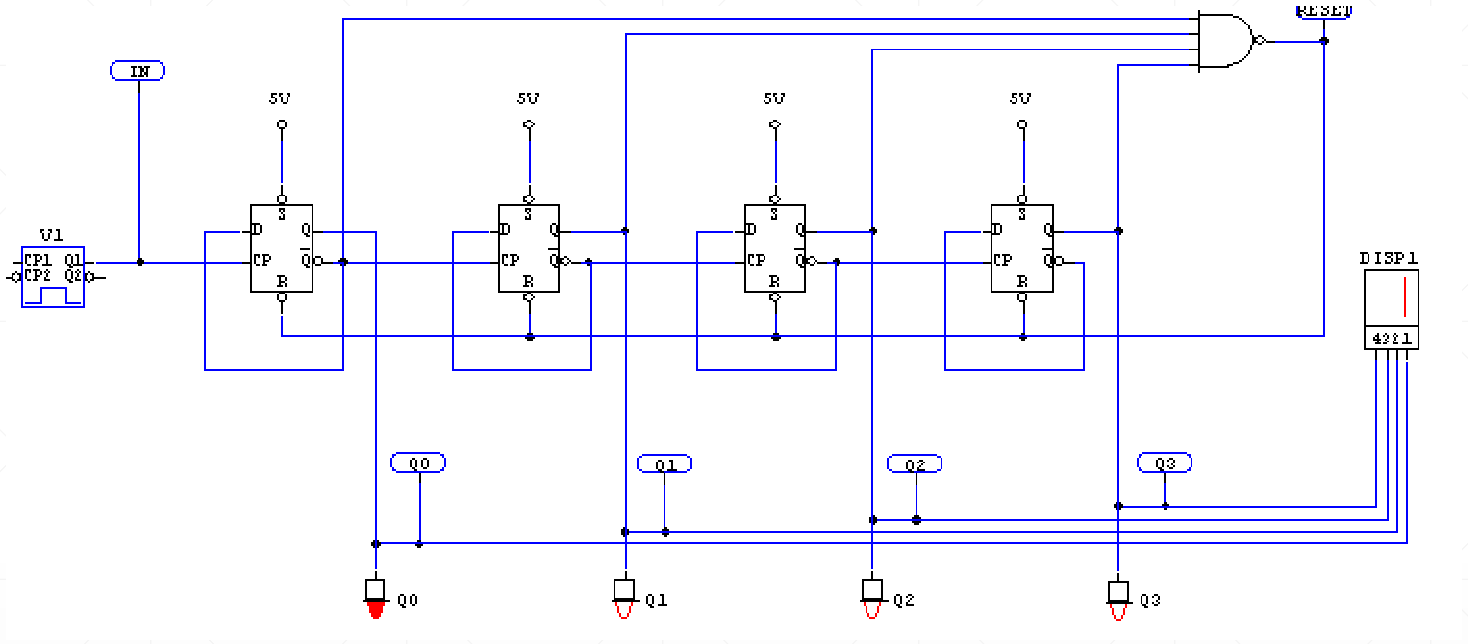


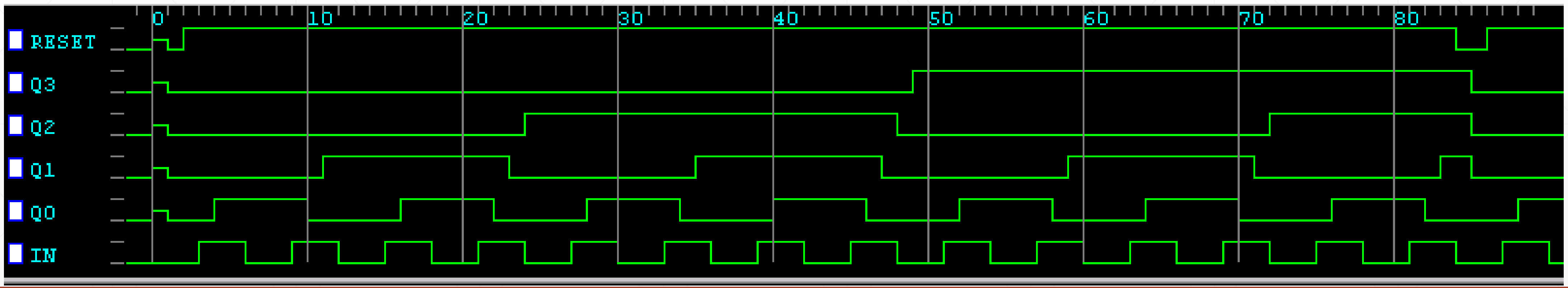
Modulus-6 Counter





Design Example…Solution





Synchronous Counters

Synchronous Counters

- All flip-flops are clocked simultaneously by an external clock.
- Synchronous counters are faster than asynchronous counters because of the simultaneous clocking.
- Synchronous counters are an example of *state machine* design because they have a set of states and a set of transition rules for moving between those states after each clocked event.

Synchronous Counters

Advantages / Disadvantages of Synchronous Counters

- Very easy to design this circuit because of the same clock pulse for all the flipflops.
- Less likely to end up in erroneous states.
- They are faster as the propagation delay are small as compared to asynchronous counters.
- There are no counting errors as compared to asynchronous counters.
- Performance is much better, liable and portable circuit.

Synchronous Counters

Applications of Synchronous Counters

- Alarm Clock, Set AC Timer, Set time in camera to take the picture, flashing light indicator in automobiles, car parking control etc.
- Counting the time allotted for special process or event by the scheduler.
- The UP/DOWN counter can be used as a self-reversing counter.
- It is also used as clock divider circuit.
- Commons used in home appliances like washing machine, microwave own, Time schedule led indicator, key board controller etc.
- They are used to generate saw-tooth waveform (Stair case voltage)
- It is also used in digital to analog converters.

Synchronous Counter (Parallel) Design Steps

1. Determine the number of FFs needed to support the counting sequence’s highest number.

$$2^n - 1 \geq \text{Highest number}$$

2. Build a State Transition Diagram. Be sure to include all states.

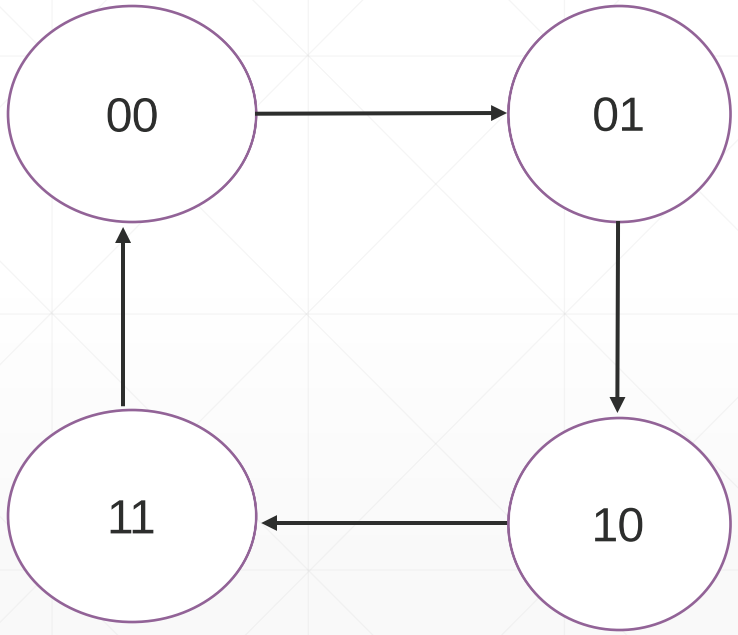
3. Build a State/Excitation Truth Table.

4. Simplify expressions for J and K inputs for each F/F on K-Maps.

5. Implement the Synchronous Counter/State Machine Circuit.

Synchronous Counter (Parallel)

- Example: 2-bit synchronous binary counter (using T flip-flop or JK)



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

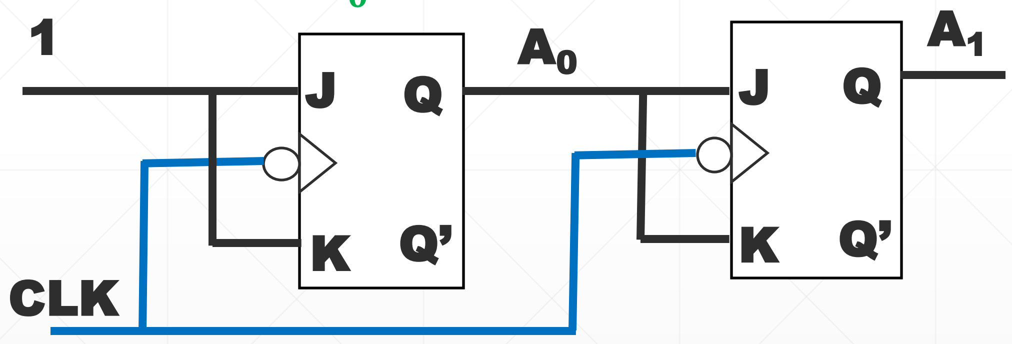
Synchronous Counter (Parallel)

- Example: 2-bit synchronous binary counter (using T flip-flop or JK) cont….

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 Counter (Parallel)

- Example: 3-bit synchronous binary counter (using T flip-flop or JK) cont….

Present State			Next State			Flip-flop Inputs		
A_2	A_1	A_0	A_2	A_1	A_0	A_2	A_1	A_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

A_2 \ $A_1 A_0$	00	01	11	10
0	0	0	1	0
1	0	0	1	0

$$TA_2 = A_1 A_0$$

A_2 \ $A_1 A_0$	00	01	11	10
0	0	1	1	0
1	0	1	1	0

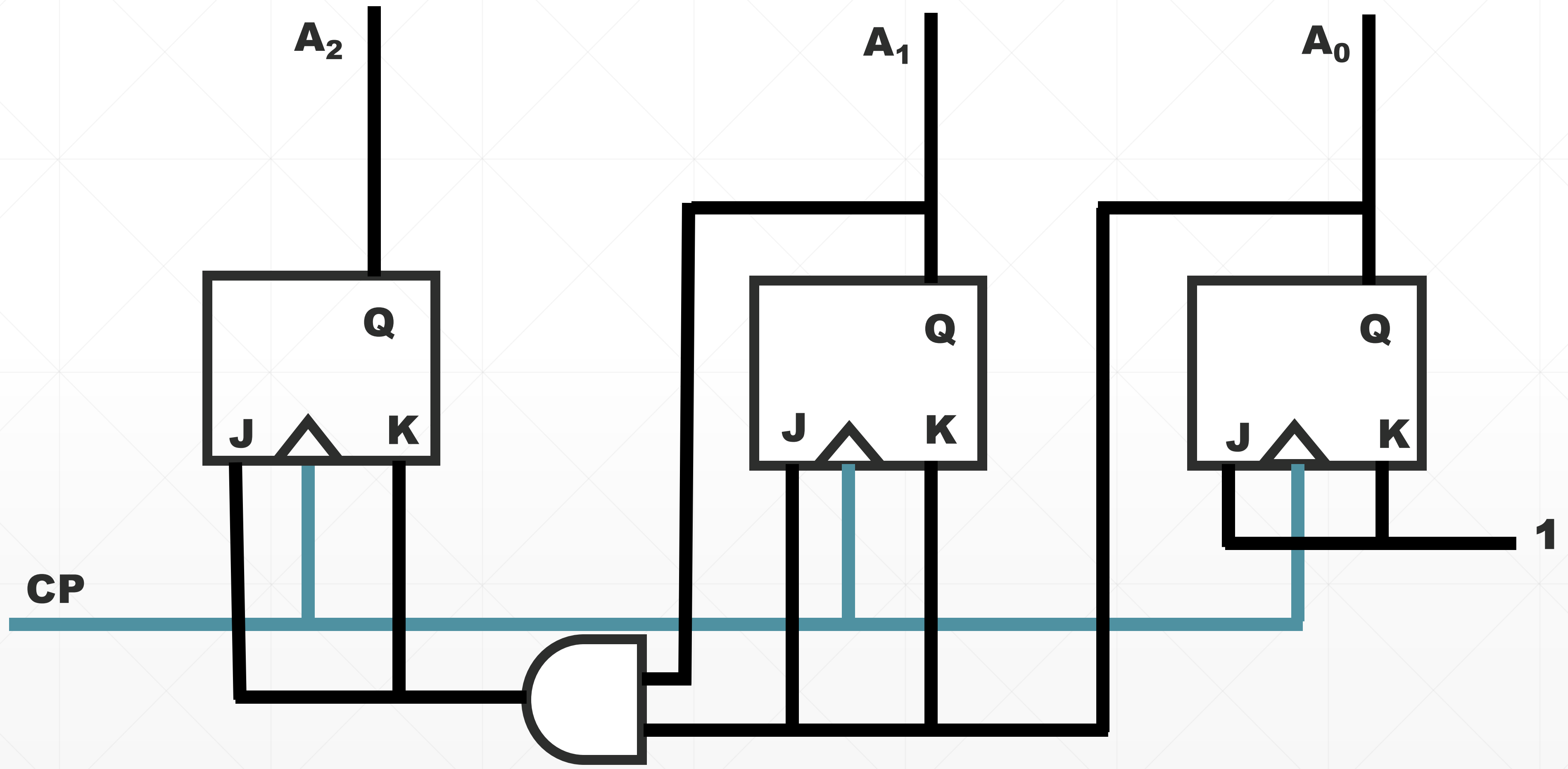
$$TA_1 = A_0$$

A_2 \ $A_1 A_0$	00	01	11	10
0	1	1	1	1
1	1	1	1	1

$$TA_0 = 1$$

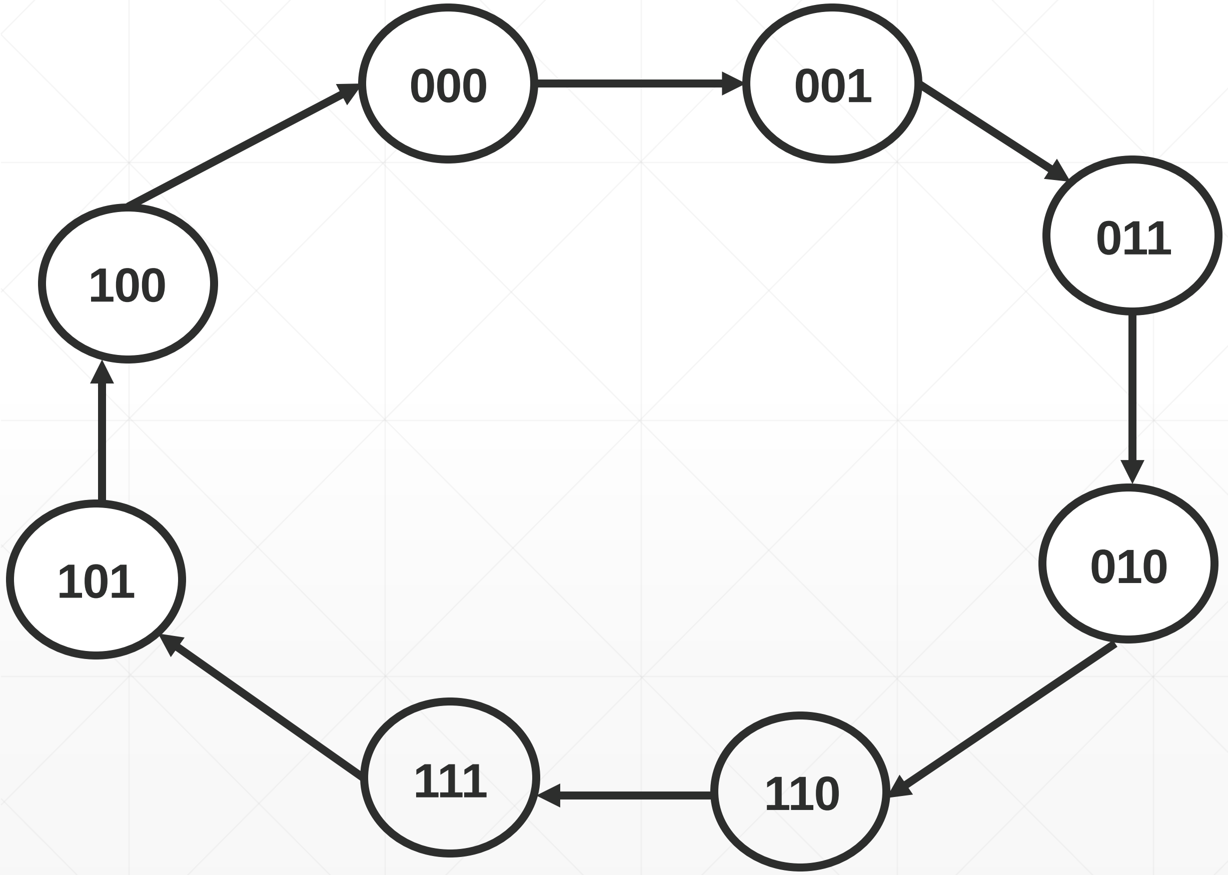
Synchronous Counter (Parallel)

- Example: 3-bit synchronous binary counter (using T flip-flop or JK) cont….



Designing Synchronous Counter

- Example: 3-bit Gray Code Counter (using JK flip-flop)



Present State			Next State			Flip-flop Inputs					
Q_2	Q_1	Q_0	Q_{2+1}	Q_{1+1}	Q_{0+1}	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	0
0	1	0	0	1	1	1	X	X	0	0	X
0	1	1	1	0	0	0	X	X	0	X	1
1	0	0	1	0	1	X	1	0	X	0	X
1	0	1	1	1	0	X	0	0	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	0	X	1	X	0

Designing Synchronous Counter

- 3-bit Gray Code Counter: flip-flop input

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	0	0	0	1
1	X	X	X	X

$$J_2 = Q_1 . \overline{Q_0}$$

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	0	1	X	X
1	0	0	X	X

$$J_1 = \overline{Q_2} . Q_0$$

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	1	X	X	0
1	0	X	X	1

$$J_0 = \overline{Q_2} . \overline{Q_1} + Q_2 . Q_1$$

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	X	X	X	X
1	1	0	0	0

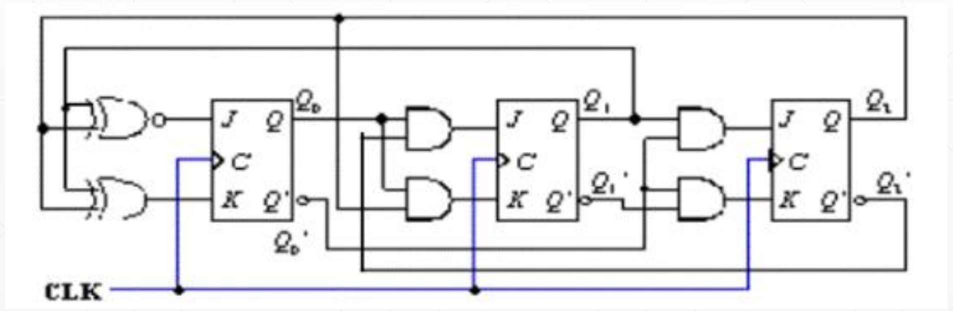
$$K_2 = \overline{Q_1} . \overline{Q_0}$$

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	X	X	0	0
1	X	X	1	0

$$K_1 = Q_2 . Q_0$$

Q_2 \ $Q_1 Q_0$	00	01	11	10
0	X	0	1	X
1	X	1	0	X

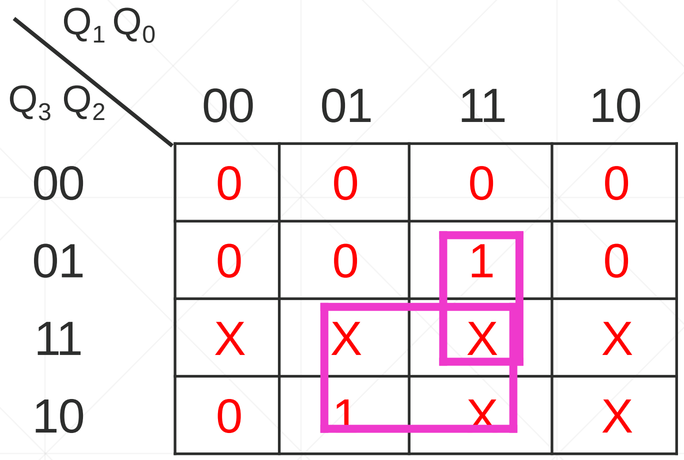
$$K_0 = Q_2 . \overline{Q_1} + \overline{Q_2} . Q_1$$



BCD Synchronous Counter (Parallel)

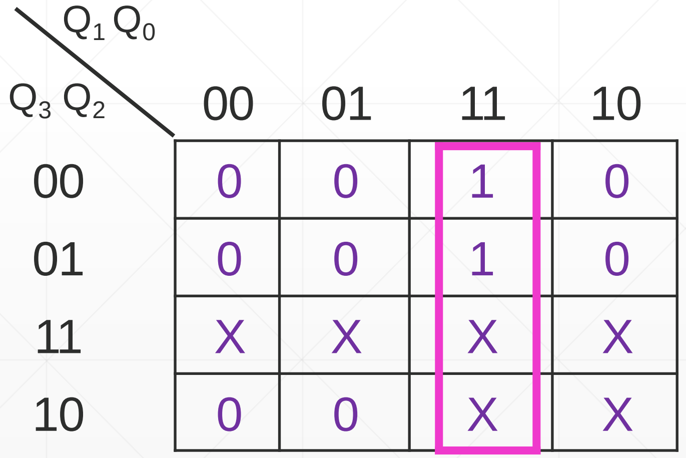
- Example: BCD Synchronous Counter

Clock Pulse	Present State				Next State				Flip Flops Input			
	Q_3	Q_2	Q_1	Q_0	Q_{3+1}	Q_{2+1}	Q_{1+1}	Q_{0+1}	T_3	T_2	T_1	T_0
Initially	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	0	0	1	1	1
4	0	1	0	0	0	1	0	1	0	0	0	1
5	0	1	0	1	0	1	1	0	0	0	1	1
6	0	1	1	0	0	1	1	1	0	0	0	1
7	0	1	1	1	1	0	0	0	1	1	1	1
8	1	0	0	0	1	0	0	1	0	0	0	1
9	1	0	0	1	0	0	0	0	1	0	0	1
10 (recycle)	0	0	0	0	0	0	0	1	0	0	0	1



Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	X	X	X	X
10	0	1	X	X

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

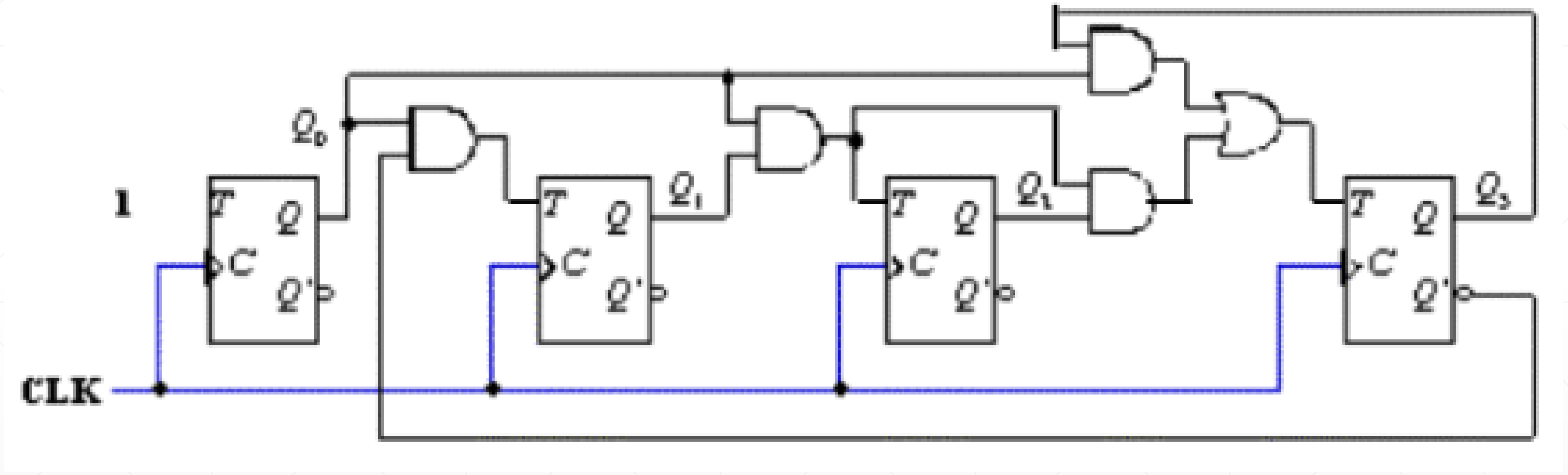


Q_3Q_2 \ Q_1Q_0	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	X	X	X	X
10	0	0	X	X

$$T_2 = Q_1 . Q_0$$

BCD Synchronous Counter (Parallel)

- Example: BCD Synchronous Counter



$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	X	X	X	X
10	0	0	X	X

$$T_1 = Q_3'.Q_0$$

$Q_3Q_2 \backslash Q_1Q_0$	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$$T_0 = 1$$

Up/Down Synchronous Counter (Parallel)

- Up/Down Synchronous Counter: two way counter which able to count up or down

- Up/Down control input line which determine the counter
  - Up/Down = 1 (count up)
  - Up/Down = 0 (count down)

Up/Down Synchronous Counter (Parallel)

- Example: 3-bit Up/Down Synchronous 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	

$$\mathrm{T}Q_0 = 1$$

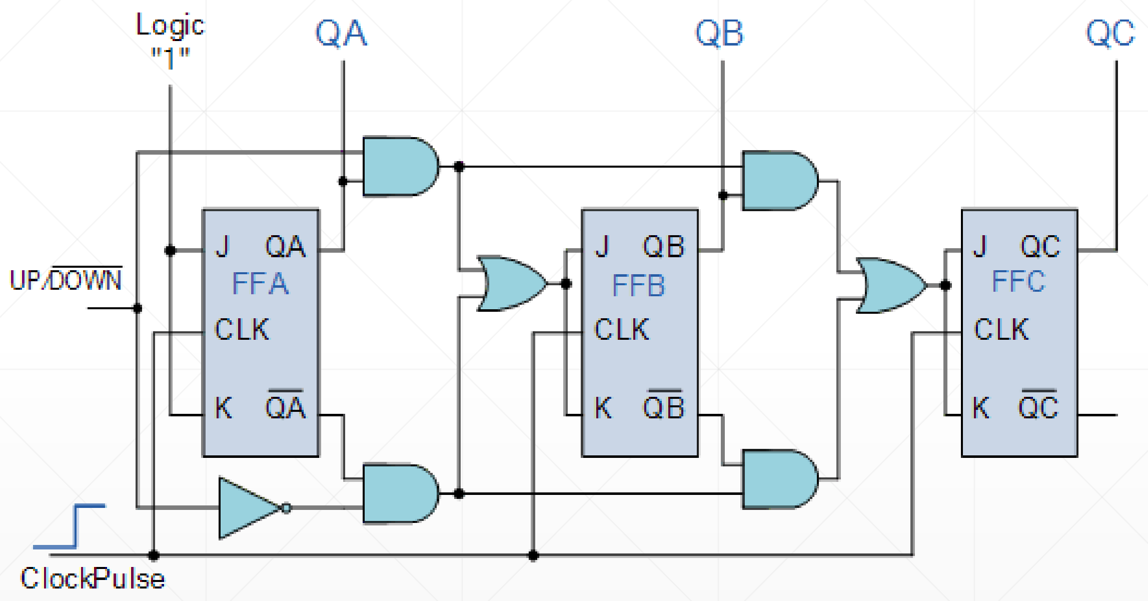
$$TQ_1 = (Q_0 . UP) + (Q_0' . UP')$$

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

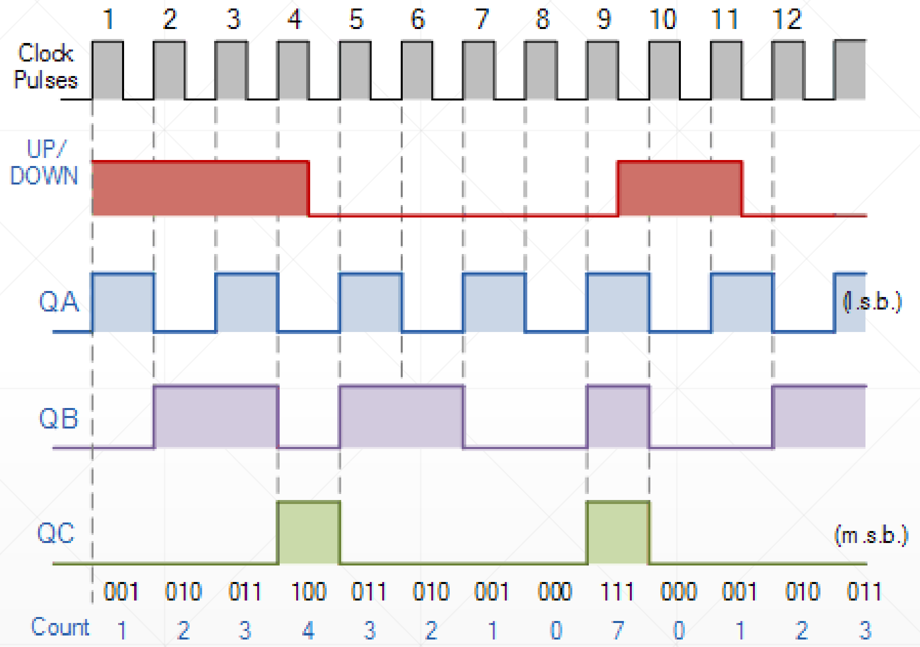
UP Counter	Down Counter
$\mathrm{T}Q_0 = 1$	$\mathrm{T}Q_0 = 1$
$TQ_1 = Q_0$	$TQ_1 = Q_1'$
$TQ_2 = Q_1 . Q_0$	$TQ_2 = Q_0' . Q_1'$

Up/Down Synchronous Counter (Parallel)

- Example: 3-bit Up/Down Synchronous Counter (cont)



Logic "1"
QA
QB
QC
UP/$\overline{\text{DOWN}}$
J QA
FFA
CLK
K $\overline{\text{QA}}$
J QB
FFB
CLK
K $\overline{\text{QB}}$
J QC
FFC
CLK
K $\overline{\text{QC}}$
Clock Pulse


Clock Pulses
UP/ DOWN
QA
(l.s.b.)
QB
QC
(m.s.b.)
Count 1 2 3 4 3 2 1 0 7 0 1 2 3