

EE2000 Logic Circuit Design

Registers and Counters

Outline

1 Registers

- Registers

 - Parallel load, Bidirectional shift

 - Serial Adder, Accumulator

2 Counters

- Asynchronous Counters

 - Ripple counters

- Synchronous Counters

 - Up-down counters,

 - Modulo counters,

 - Ring counters, Twist Ring counters

Applications of Flip-Flops

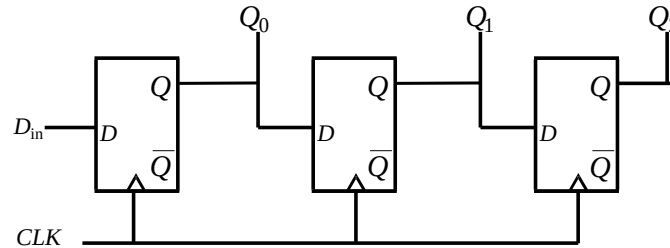
- Flip-Flops are basic element of Static Random Access Memory (SRAM) in CPUs, ALUs, microcontrollers.
- Shift register is an example of this kind of memories. This includes:
 - Parallel loading shift register : can be used for parallel to serial conversion in serial data communication
 - Serial in, parallel out shift register: can be used for serial to parallel conversion in a serial data communication system.

Registers

- To store n -bit data, require n flip-flops
- n -bit register
 - Composed of a set of n edge-triggered flip-flops
 - The clock pulse is the enable signal
 - Data will be latched to the flip-flops during the clock active period (pos. edge or neg. edge)
- Two designs
 - Parallel, or Serial

Shift Register

- e.g. 3bit shift register by D-type FFs

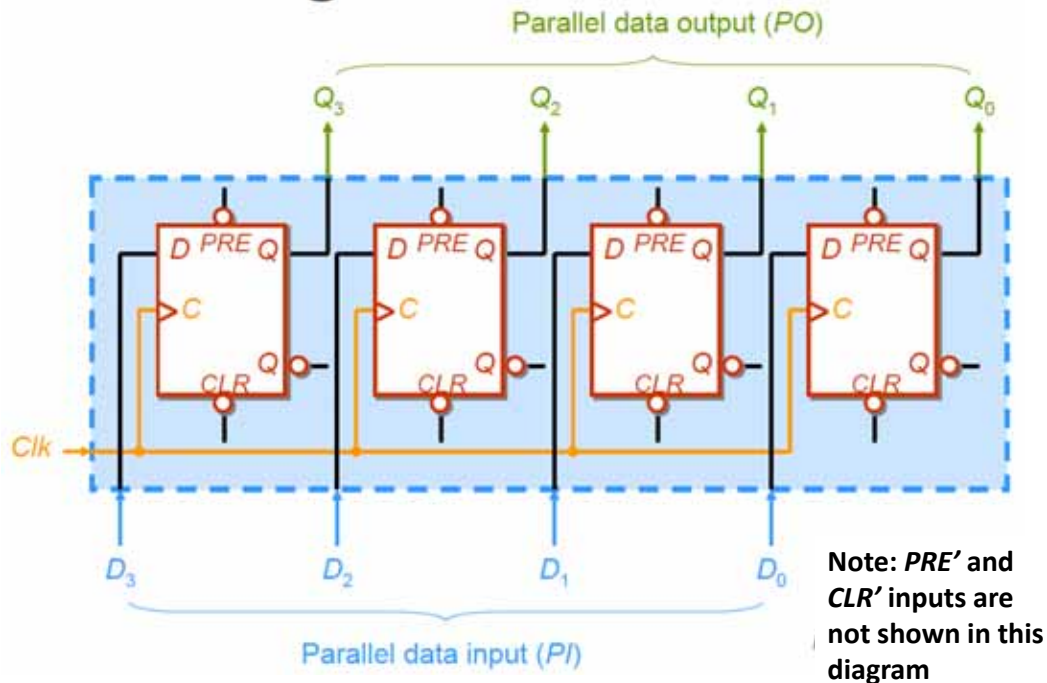


- serial input, D_{in} and parallel output Q_0 , Q_1 and Q_2 .
- data moves one position to the right on each clock edge.

Shift Register

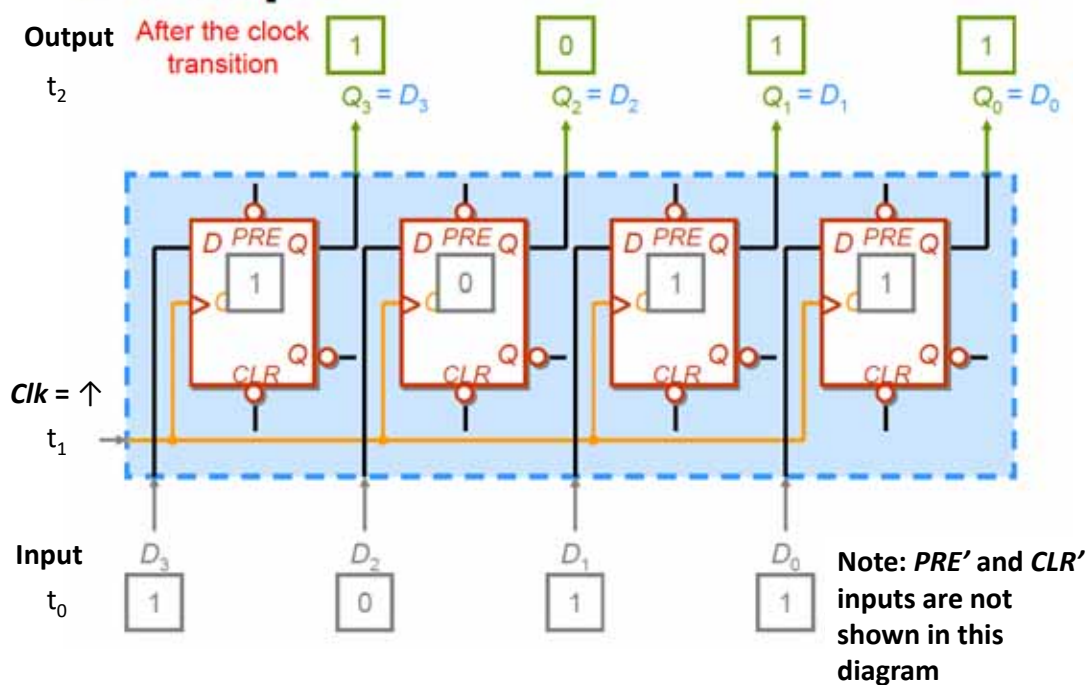
- Preset and Clear inputs on the FFs can be utilised to provide a parallel data input feature
- Data can then be clocked out through Q_2 in a serial fashion, i.e., we now have a parallel in, serial out arrangement
- This along with the previous serial in, parallel out shift register arrangement can be used as the basis for a serial data link

4-bit Register



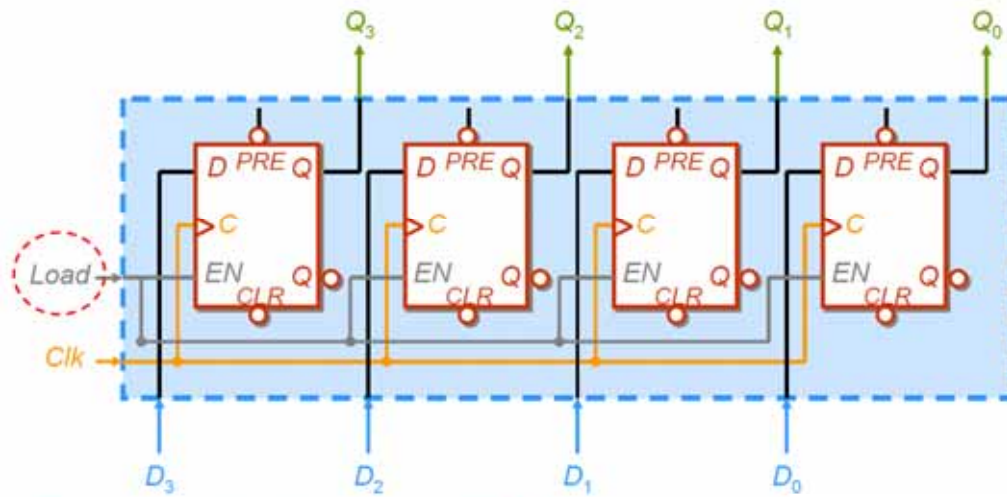
7

Example



8

Register with Parallel Load

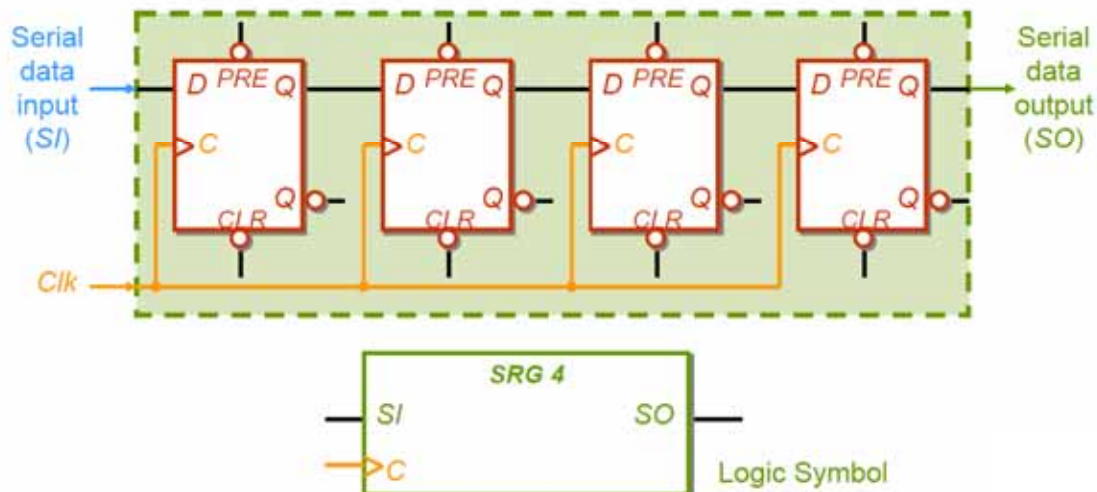


The register will load the inputs into the flip-flops only when Clk is at the rising edge AND Load = 1

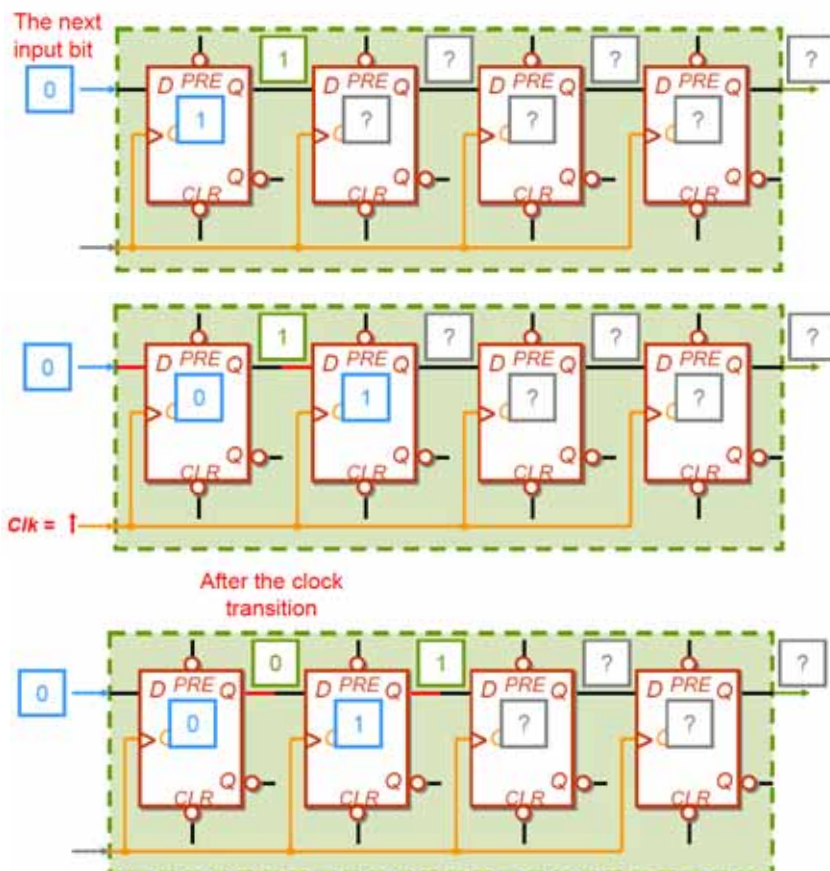
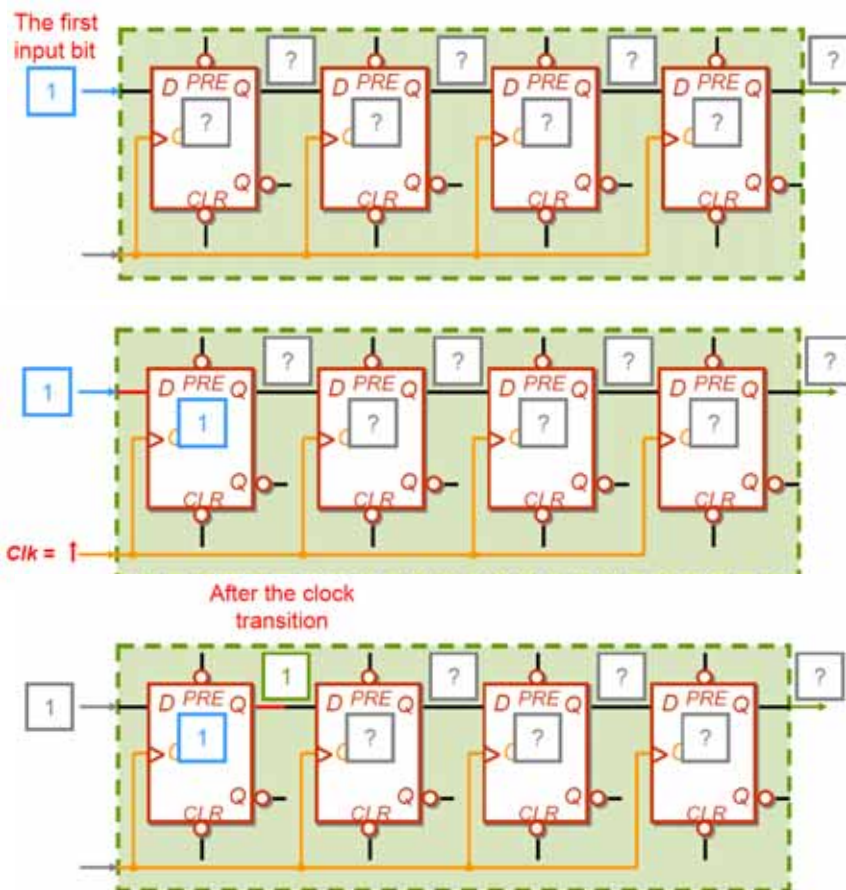
9

Shift Registers

■ A 4-bit Serial-In-Serial-Out Register



10



Counters

- Many different counters can be designed based on applications.
- We shall focus on *ripple* counters and *synchronous* counters

Counters

- Register are useful for storing and manipulating information
- What is a counter?
 - A special type of register
 - Goes through a predetermined sequence of states
 - Used to sequence and control operations in a circuit

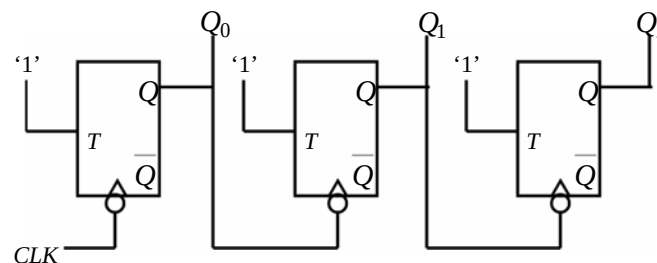
Counters

- Two classes of counters
 - Asynchronous (binary) counters
 - Synchronous (binary) counters
- Asynchronous counters
 - e.g. ripple counters: the flip-flops are triggered by the **transition of other flip-flops**
- Synchronous Counters
 - All flip-flops are triggered by the **common clock**

15

Ripple Counters

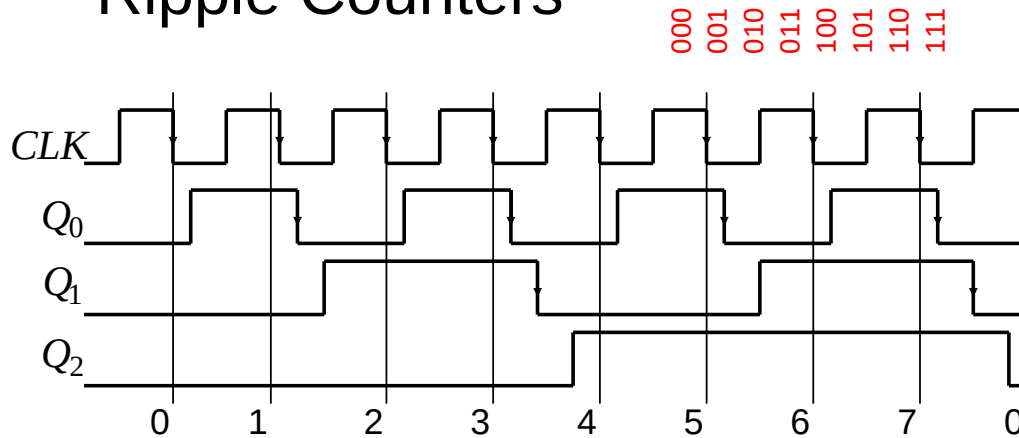
- A ripple counter can be made by cascading together negative edge triggered T-type FFs operating in 'toggle' mode, i.e., $T=1$



000
001
010
011
100
101
110
111

- Note that the FFs are not clocked using the same clock, i.e., this is **not** a synchronous design.

Ripple Counters



Problems:

- Outputs do not change at the same time or synchronously. So hard to know when count output is actually valid.
- Propagation delay builds up from stage to stage, limiting maximum clock speed before miscounting occurs.

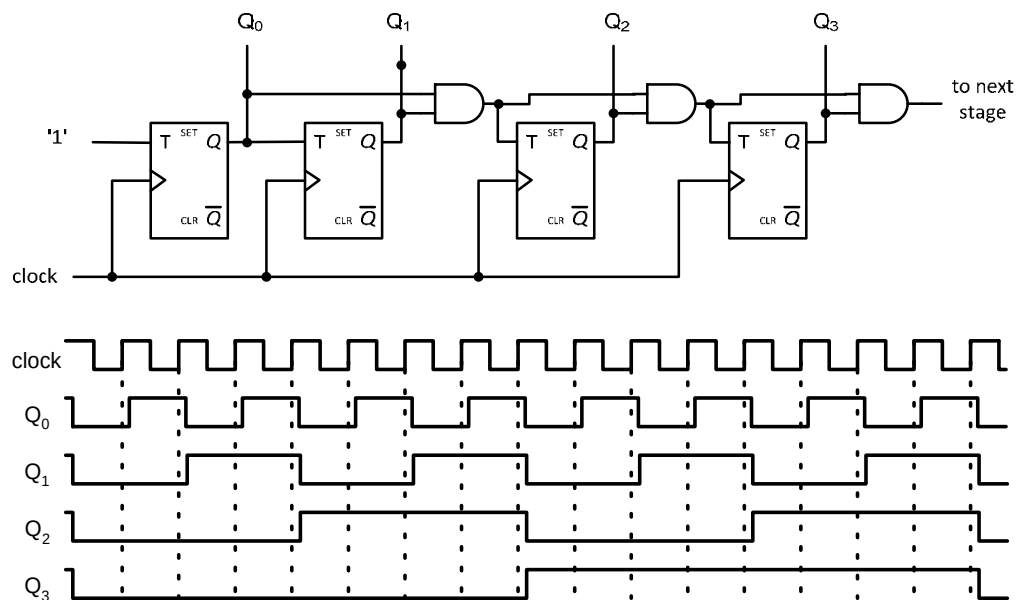
Ripple Counters

- If you observe the frequency of the counter output signals you will note that each has half the frequency, i.e., double the repetition period of the previous one. This is why counters are often known as dividers
- Often we wish to have a count which is not a power of 2, e.g., for a BCD counter (0 to 9). To do this:
 - use FFs having a Reset/Clear input
 - Use an AND gate to detect the count of 10 and use its output to Reset the FFs

Synchronous Counters

- *Synchronous* counter overcomes the problems identified with ripple counters.
- In a synchronous design
 - all the FF clock inputs are directly connected to the clock signal and so all FF outputs change at the same time, i.e., *synchronously*
 - more complex combinational logic is now needed to generate the appropriate FF input signals (which will be different depending upon the type of FF chosen)

Synchronous Counter



The delays of all FFs are the propagation delays of their own.

It can be used for **frequency division**.

Synchronous Counter Design procedures

- Work out the state transition table
- Determine the FF excitation (easy for D-types)
- Determine the combinational logic for generating the required FF excitation from the current states;
 - **Note:** remember to take into account any unused counts since these can be used as don't care states when determining the combinational logic circuits

Synchronous Counter Example

Use D-type FFs to design a synchronous counter counting 0 to 7.

Current state	Next state	FF inputs
$Q_2 Q_1 Q_0$	$Q_2' Q_1' Q_0'$	$D_2 D_1 D_0$
0 0 0	0 0 1	0 0 1
0 0 1	0 1 0	0 1 0
0 1 0	0 1 1	0 1 1
0 1 1	1 0 0	1 0 0
1 0 0	1 0 1	1 0 1
1 0 1	1 1 0	1 1 0
1 1 0	1 1 1	1 1 1
1 1 1	0 0 0	0 0 0

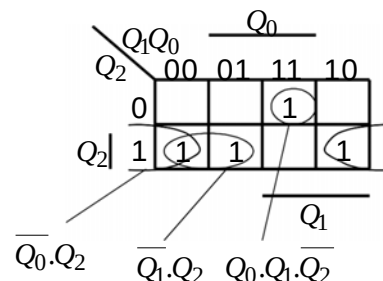
By inspection,

$$D_0 = \overline{Q_0}$$

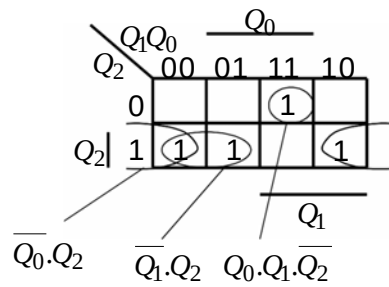
Note: FF₀ is toggling Also,

$$D_1 = Q_0 \oplus Q_1$$

Use a K-map for D_2 ,



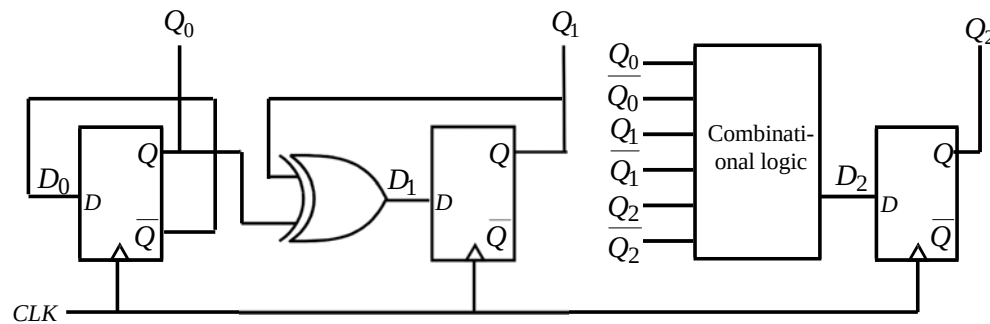
Synchronous Counter Example



So,

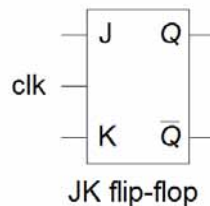
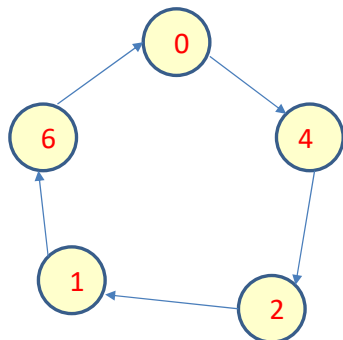
$$D_2 = \overline{Q_0} \cdot Q_2 + \overline{Q_1} \cdot Q_2 + Q_0 \cdot Q_1 \cdot \overline{Q_2}$$

$$D_2 = Q_2 \cdot (\overline{Q_0} + \overline{Q_1}) + Q_0 \cdot Q_1 \cdot \overline{Q_2}$$

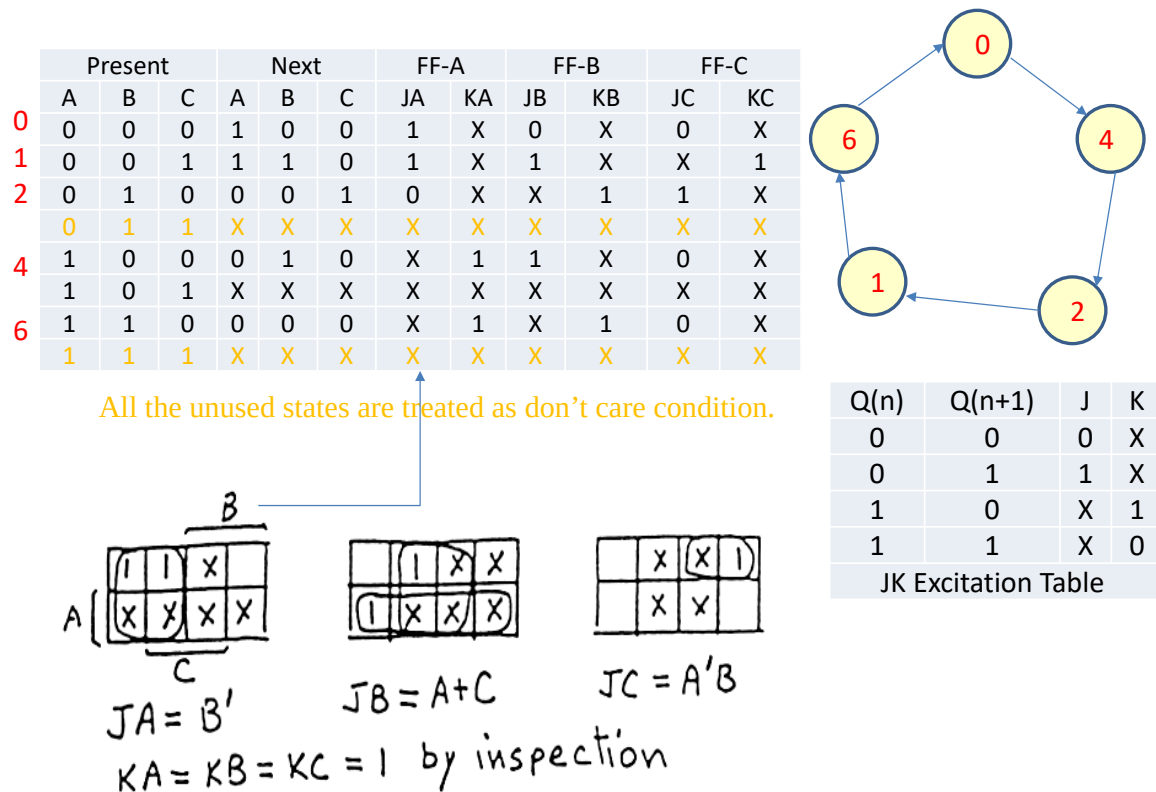
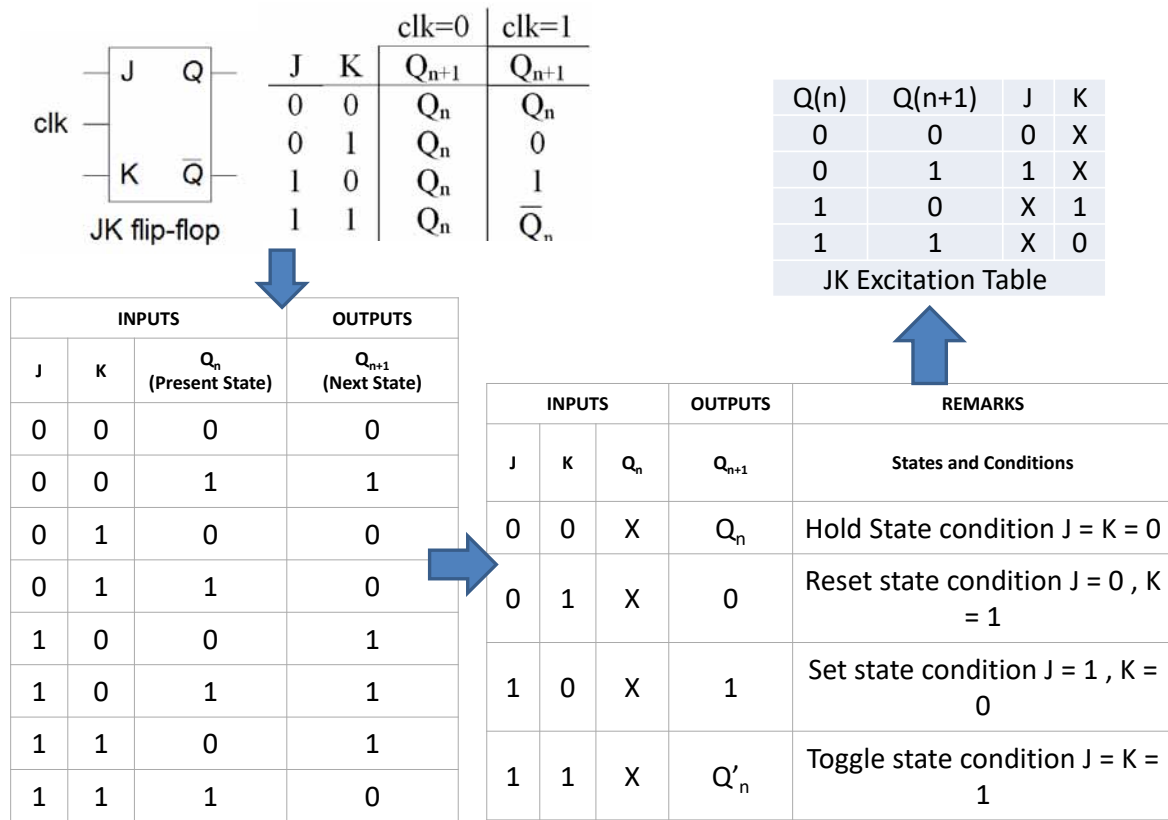


Synchronous Counter Example with JK FF

- Using JK FFs and with the minimum number of states to design a counter with counting sequence: 0,4,2,1,6 and repeat.



J	K	clk=0	clk=1
		Q_{n+1}	Q_{n+1}
0	0	Q_n	Q_n
0	1	Q_n	0
1	0	Q_n	1
1	1	Q_n	$\overline{Q_n}$



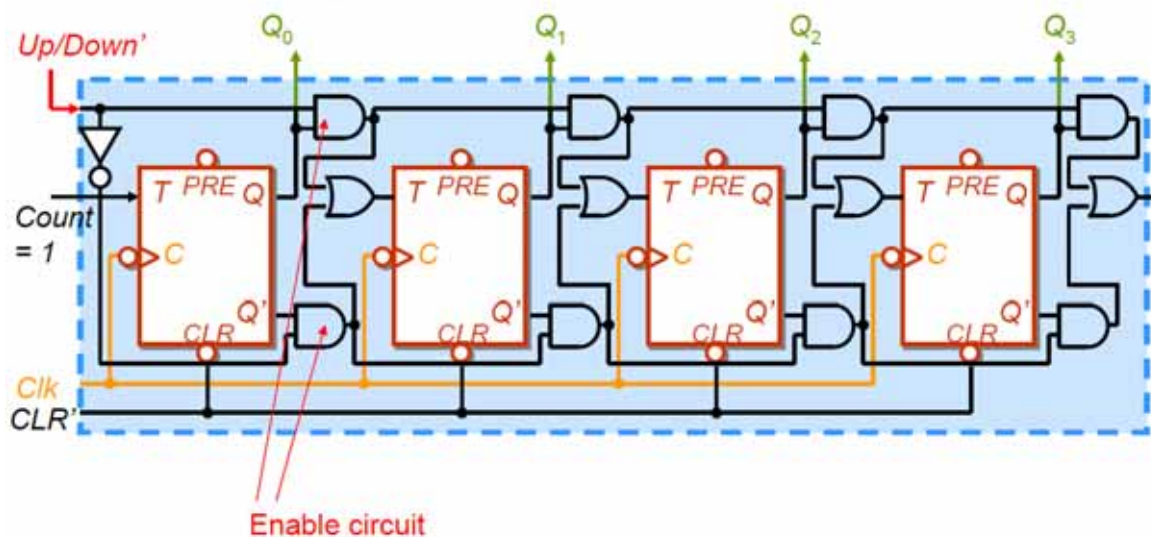
Sync. Up-Down Counter

Upward Counting Sequence			
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

Downward Counting Sequence			
Q_3	Q_2	Q_1	Q_0
1	1	1	1
1	1	1	0
1	1	0	1
1	1	0	0
1	0	1	1
1	0	1	0
1	0	0	1
1	0	0	0
0	1	1	1
0	1	1	0
0	1	0	1
0	1	0	0
0	0	1	1
0	0	1	0
0	0	0	1
0	0	0	0

27

Sync. Up-Down Counter



Up/Down' used to select feeding either Q or Q' to next T
 If *Up/Down'* = 1, upward counting (the same as previous counter). If
Up/Down' = 0, downward counting (feeding Q' to T)

28

Summary

- Synchronous Counters
 - Outputs (Q_i) appear at the same time
- Asynchronous Counters
 - Outputs (Q_i) do not appear at the same time
 - The final flip-flop has the largest propagation delay
- Counters, modulo counters, ring counters, twisted ring counters
 - Simple design to construct frequency divisor, irregular frequency divisor, time division multiplexer, period shifter respectively