COUNTERS

Counters

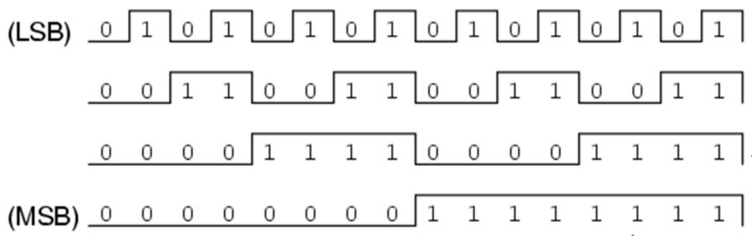
Counters can be classified into two broad categories according to the way they are clocked:

- Asynchronous (Ripple) Counters - the first flip-flop is clocked by the external clock pulse, and then each successive flip-flop is clocked by the Q or Q' output of the previous flip-flop.
- Synchronous Counters - all memory elements are simultaneously triggered by the same clock.

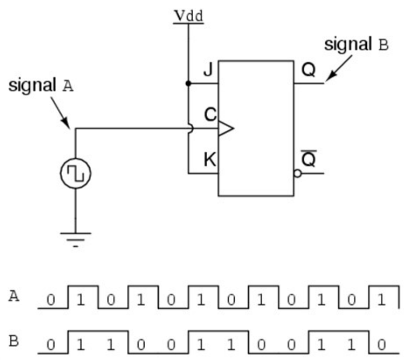
Binary count sequence

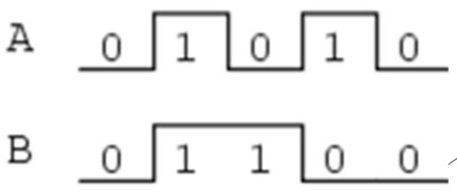
000
001
010
011
100
101
110
111

If we wanted to design a digital circuit to "count" in four-bit binary, all we would have to do is design a series of frequency divider circuits, each circuit dividing the frequency of a square-wave pulse by a factor of 2:


(LSB) 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
(MSB) 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1

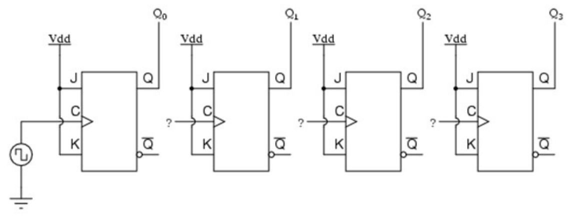
- J-K flip-flops are ideally suited for this task, because they have the ability to "toggle" their output state at the command of a clock pulse when both J and K inputs are made "high" (1):

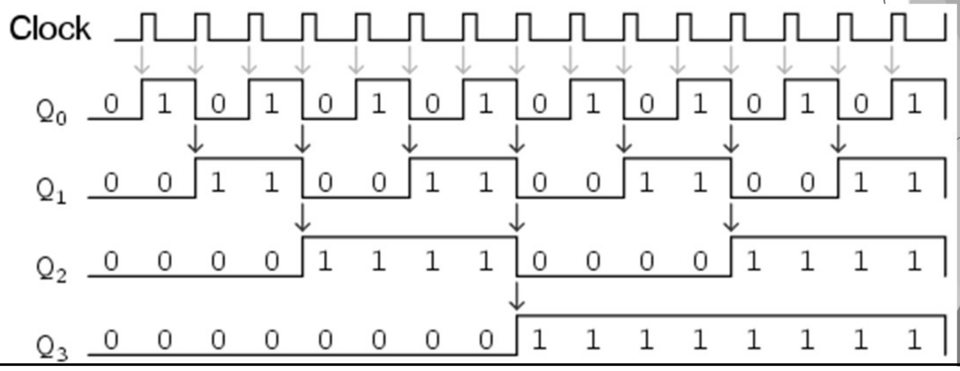


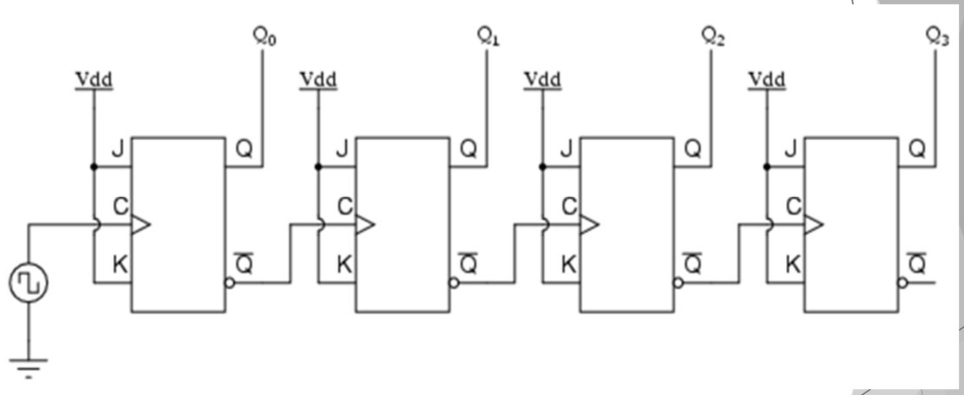


Asynchronous (Ripple) Counters

- In this counter all the flip-fl ops are not driven by the same clock pulse. Here, the clock pulse is applied to the first flip-flop; i.e., the least significant bit state of the counter, and the successive flip-fl op is triggered by the output of the previous flip-fl op.

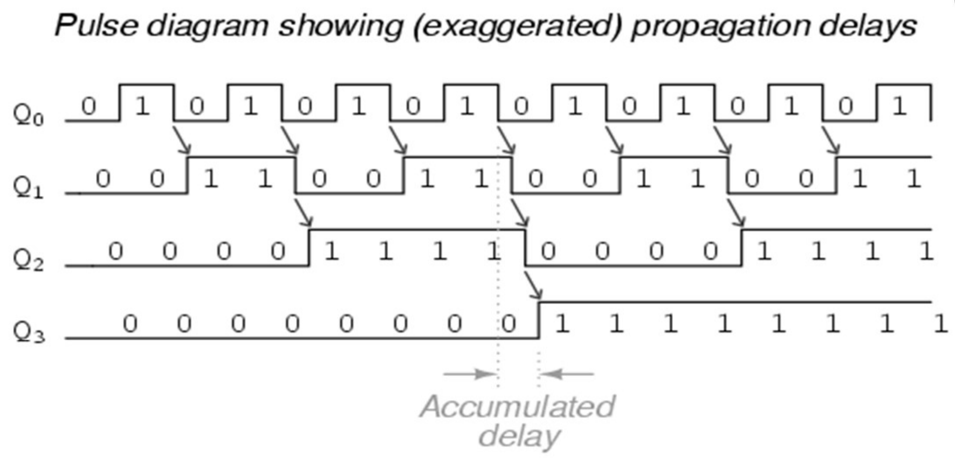




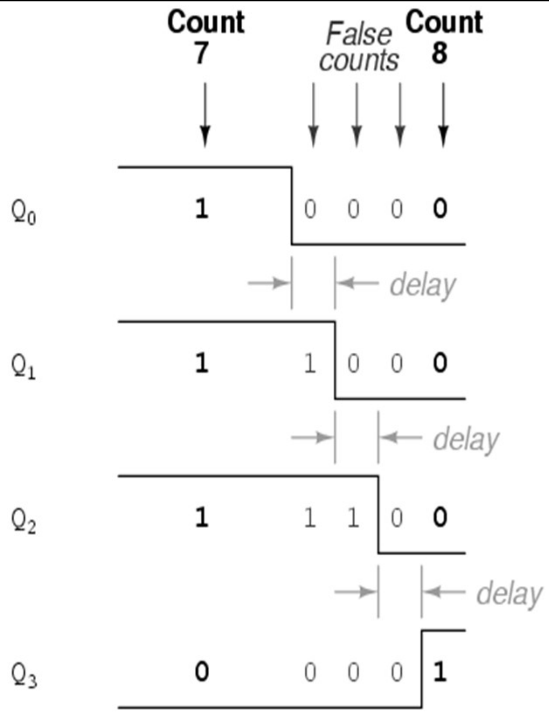


Problem

- ▶Unfortunately, all of the counter circuits shown thus far share a common problem: the *ripple* effect. This effect is seen in certain types of binary adder and data conversion circuits, and is due to accumulative propagation delays between cascaded gates.



This Accumulative Delay of an Asynchronous Counter limits the rate at whicg a counter can be clocked and can also create decoding problems.

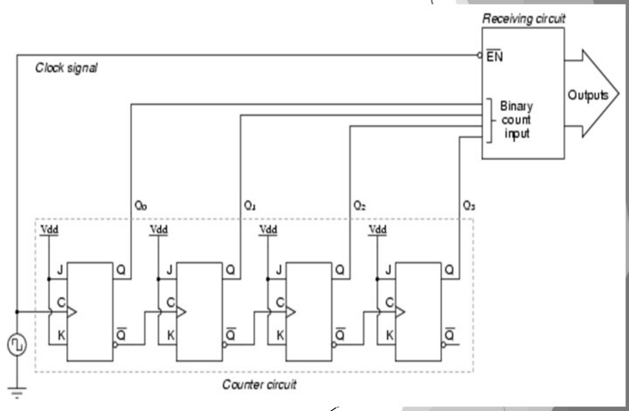


- These false states of the output are known as G**LITCHES.**

N = NUMBER OF FLIP-FLOPS
dt = PROP. DELAY OF EACH FLIPFLOP
TOTAL DELAY = **N. dt**

Solution

- *Strobing* is a technique applied to circuits receiving the output of an asynchronous (ripple) counter, so that the false counts generated during the ripple time will have no ill effect.

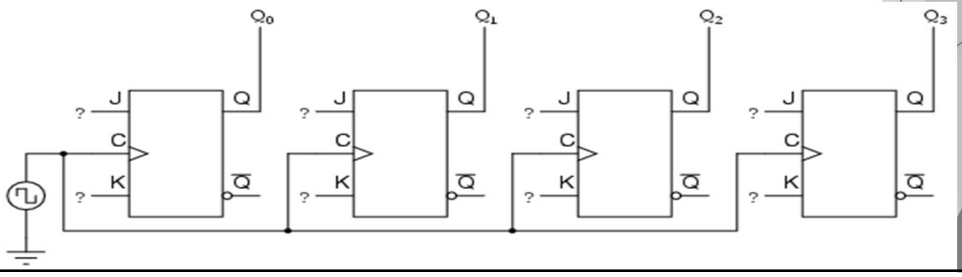


Synchronous Counters

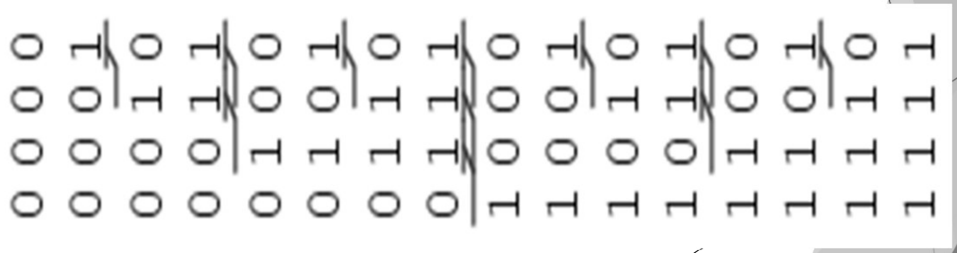
- all the flip-flops are clocked synchronously.
- Synchronous counters can be designed for any count sequence
- Solves the problem of high frequency limitation and glitches occurring at the output of decoding gates

Synchronous counters

- ▶we can build such a counter circuit from J-K flip-flops by connecting all the clock inputs together, so that each and every flip-flop receives the exact same clock pulse at the exact same time:

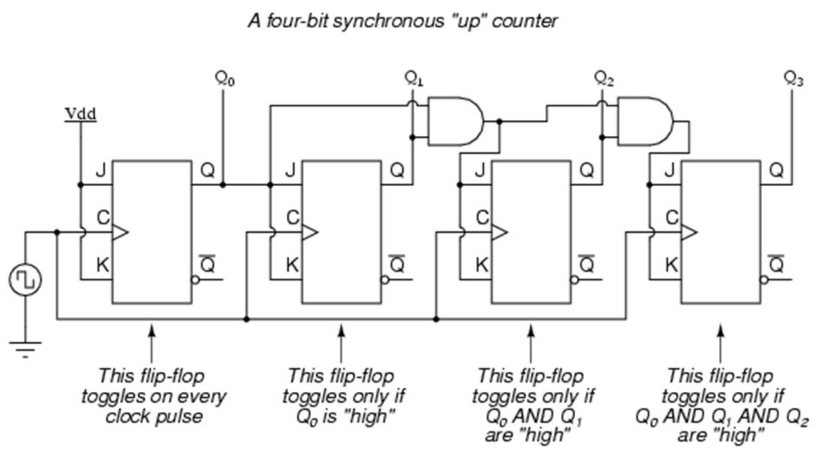


- ▶Examining the four-bit binary count sequence, another predictive pattern can be seen. Notice that just before a bit toggles, all preceding bits are "high:"

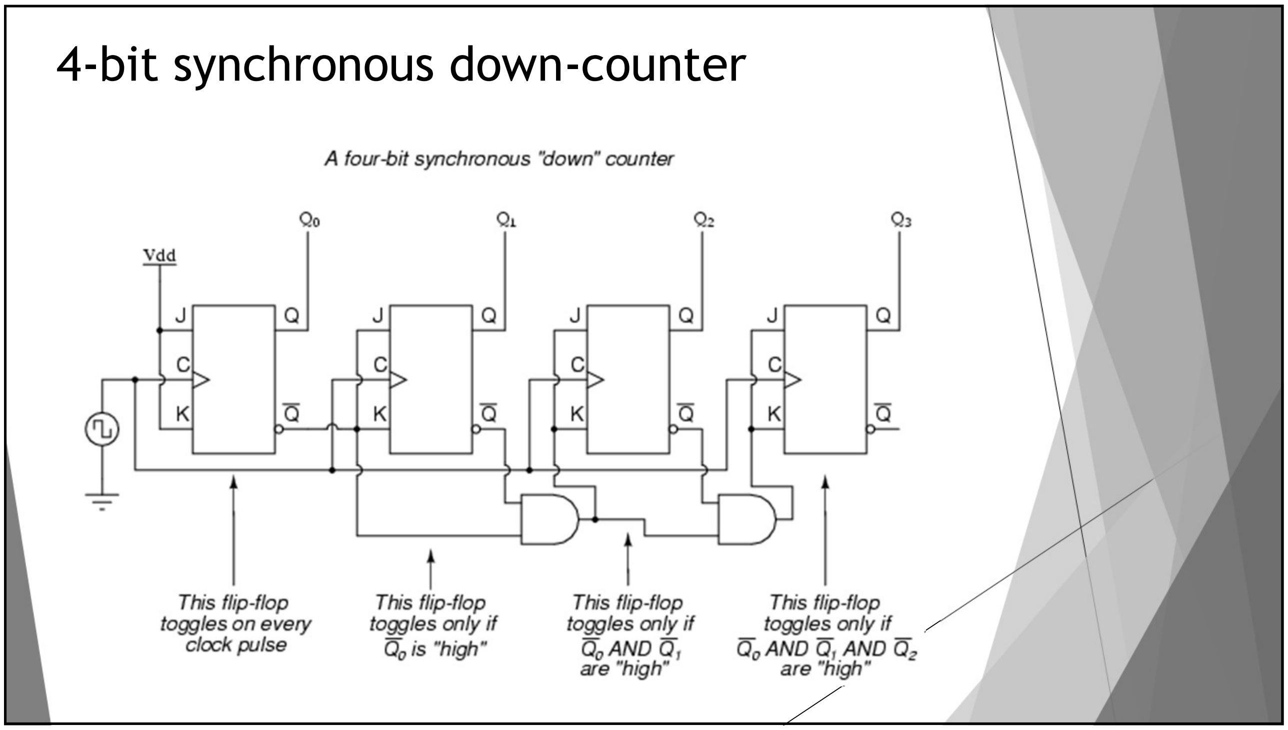


0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

4-bit synchronous up-counter


A four-bit synchronous "up" counter
Vdd
Q_0
Q_1
Q_2
Q_3
J
C
K
Q
$\overline{Q}$
This flip-flop toggles on every clock pulse
This flip-flop toggles only if Q_0 is "high"
This flip-flop toggles only if Q_0 AND Q_1 are "high"
This flip-flop toggles only if Q_0 AND Q_1 AND Q_2 are "high"

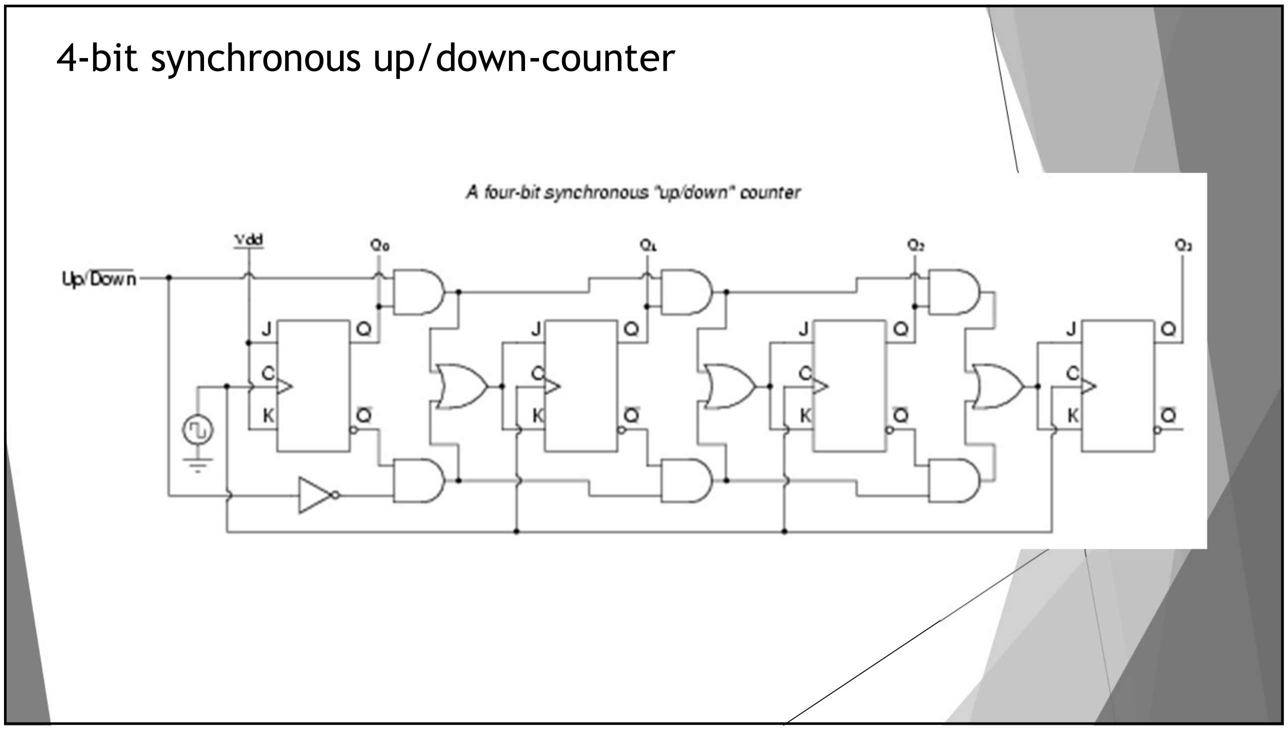
- To make a synchronous "down" counter, we need to build the circuit to recognize the appropriate bit patterns predicting each toggle state while counting down.
- When we examine the four-bit binary count sequence, we see that all preceding bits are "low" prior to a toggle (following the sequence from bottom to top):



4-bit synchronous down-counter

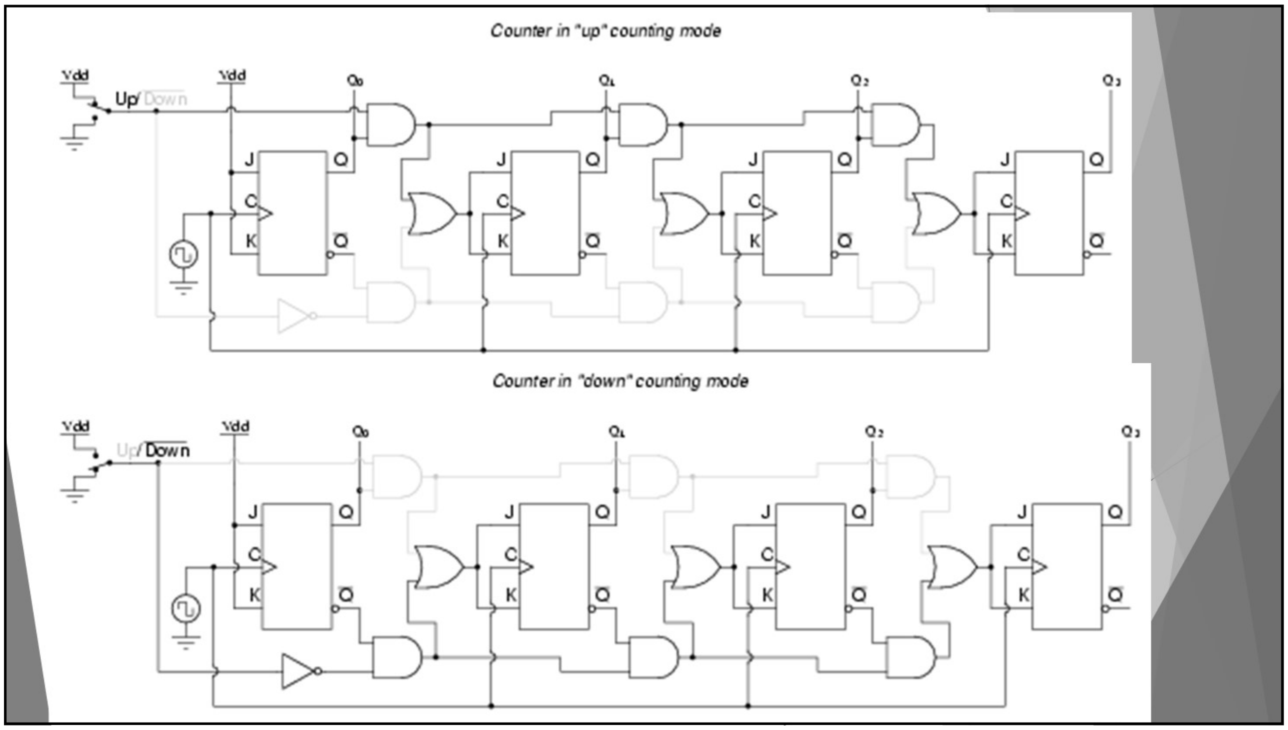
A four-bit synchronous "down" counter

Vdd
Q_0
Q_1
Q_2
Q_3
J
C
K
Q
$\overline{Q}$
This flip-flop toggles on every clock pulse
This flip-flop toggles only if $\overline{Q}_0$ is "high"
This flip-flop toggles only if $\overline{Q}_0$ AND $\overline{Q}_1$ are "high"
This flip-flop toggles only if $\overline{Q}_0$ AND $\overline{Q}_1$ AND $\overline{Q}_2$ are "high"



4-bit synchronous up/down-counter

A four-bit synchronous "up/down" counter

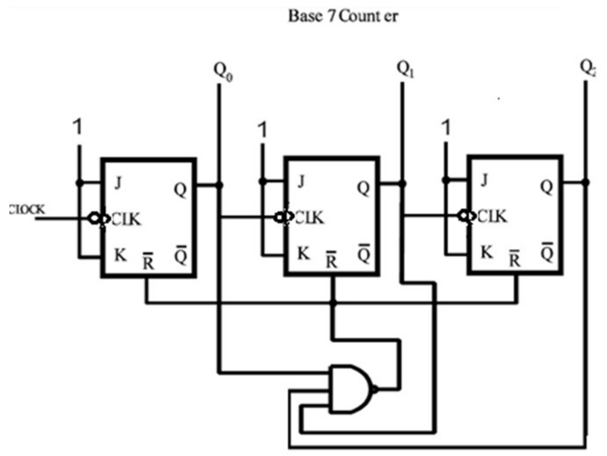

Counter in "up" counting mode
Vdd
Up/Down
Q0
Q1
Q2
Q3
J
C
K
Q
Q̄
Counter in "down" counting mode
Up/Down

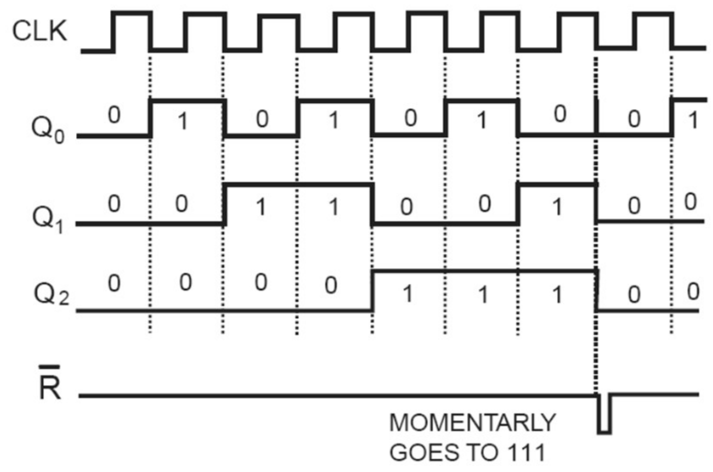
MOD Number

- The MOD number is equal to the number of states that the counter goes through in each complete cycle before it recycles back to its starting state.
- MOD number=2^N, where N is the number of flip-flops
- Frequency division
- Problem: How to convert a 60Hz signal to a 1Hz signal using frequency division?

BASE K ASYNCHRONOUS COUNTER

- **K** is **NOT** a power of 2
- We choose **N** flip-flops where $2^N > K$

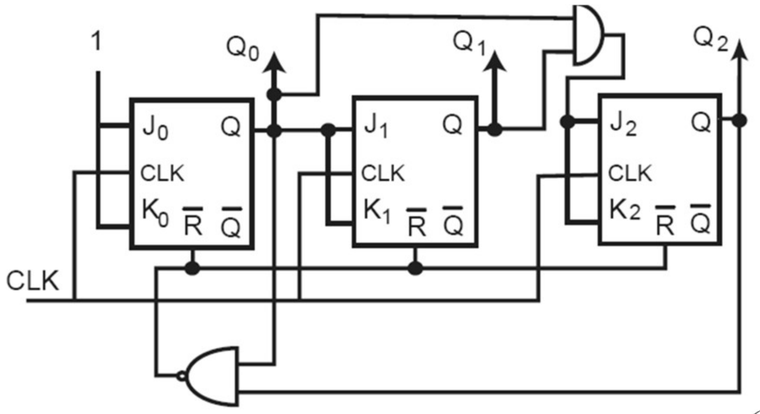




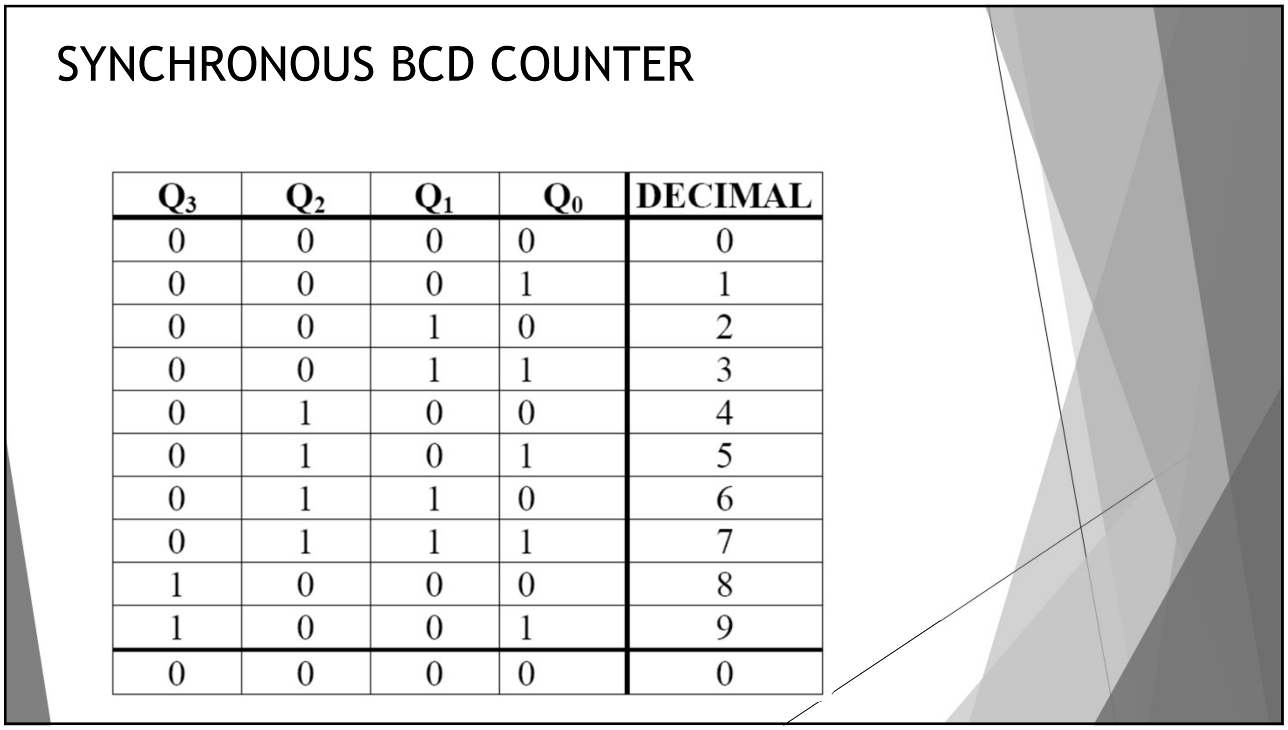
E.G. **BASE 5** SYNCHRONOUS COUNTER

STATE	Q_2	Q_1	Q_0	DECIMAL
1	0	0	0	0
2	0	0	1	1
3	0	1	0	2
4	0	1	1	3
5	1	0	0	4
6	**0**	**0**	**0**	0

FOR THIS COUNTER A GLITCH WILL BE GENERATED BEFORE THE COUNTER IS RESET.

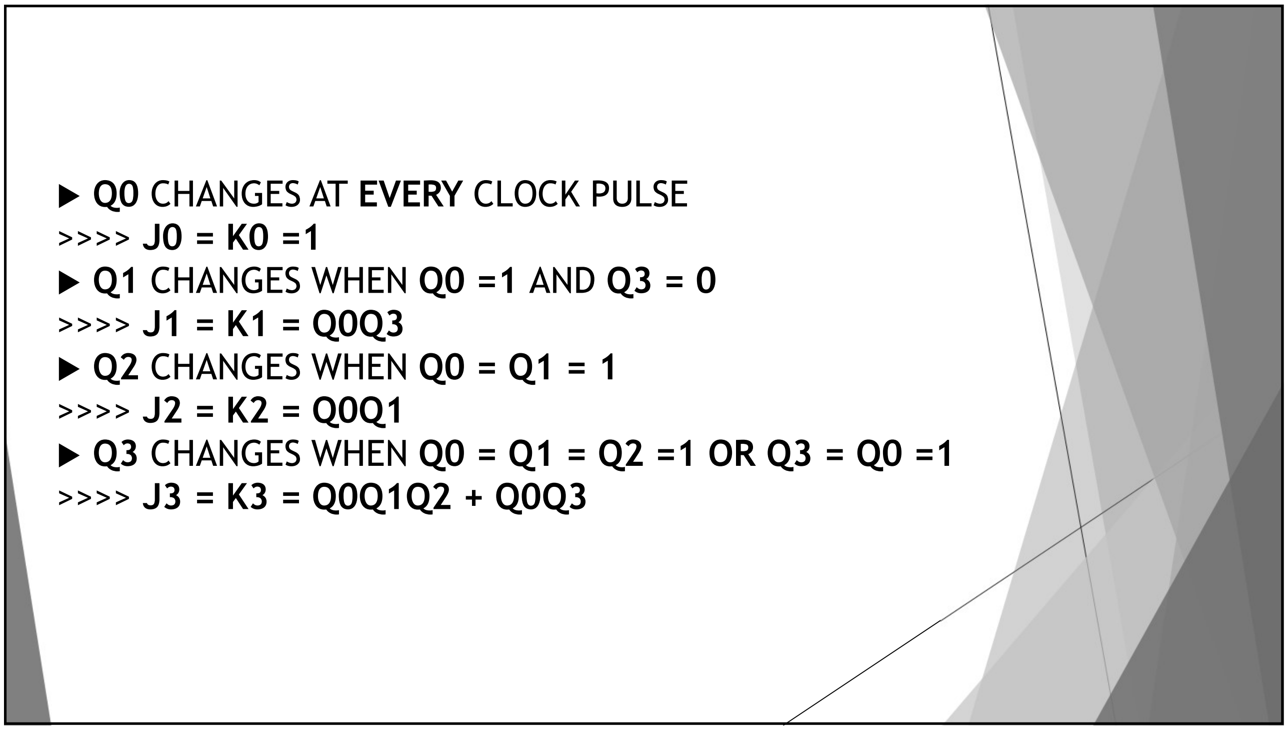


- **NOTE:** COUNTER GOES TO **‘101’** [5] **MOMENTARILY** BEFORE RESETING >>>> **GLITCH** PRESENT



SYNCHRONOUS BCD COUNTER

Q_3	Q_2	Q_1	Q_0	DECIMAL
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
0	0	0	0	0



- **Q0** CHANGES AT **EVERY** CLOCK PULSE

>>>> **J0 = K0 =1**

- **Q1** CHANGES WHEN **Q0 =1** AND **Q3 = 0**

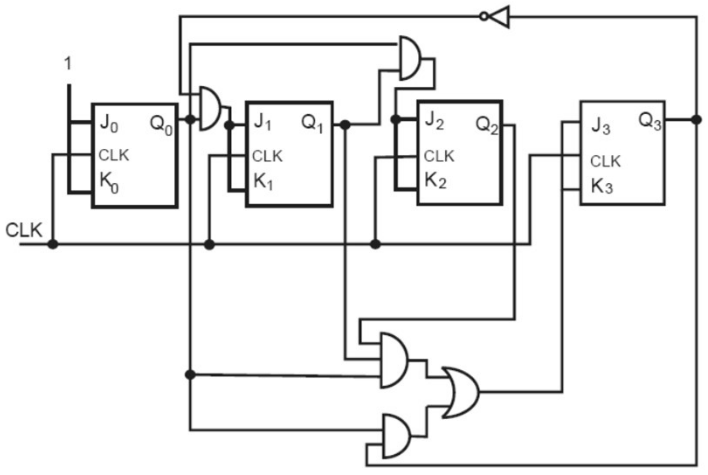
>>>> **J1 = K1 = Q0Q3**

- **Q2** CHANGES WHEN **Q0 = Q1 = 1**

>>>> **J2 = K2 = Q0Q1**

- **Q3** CHANGES WHEN **Q0 = Q1 = Q2 =1 OR Q3 = Q0 =1**

>>>> **J3 = K3 = Q0Q1Q2 + Q0Q3**



NOTE:
THE **AND - OR** GATE COMBINATION EFFECTIVELY DETECTS THE **OCCURANCE OF 1001** AND CAUSES THE COUNTER TO **RECYCLE PROPERLY** ON THE NEXT CLOCK PULSE >>>> **OVERCOME GLITCH**

REGISTERS

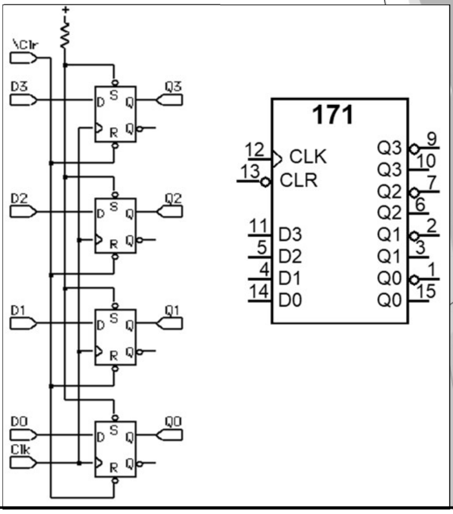
Registers

- **Groups of flip-flop arranged to provide data storage for several bits**
- Types of Registers:
  - Data/Storage Registers
  - Shift Registers

Storage Register

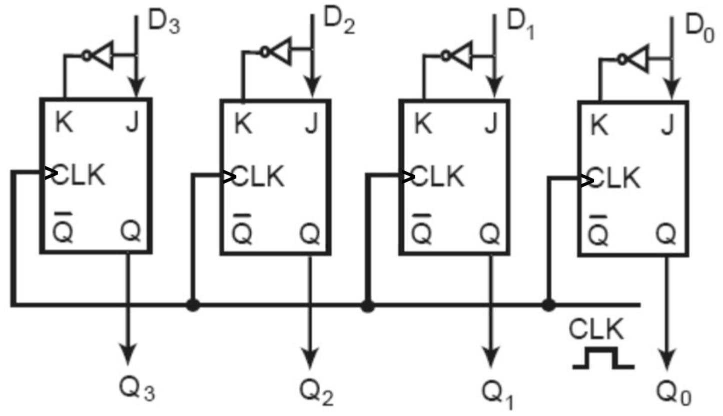
- Group of storage elements read/written as a unit

- 4-bit register constructed from 4 D FFs
- Shared clock and clear lines

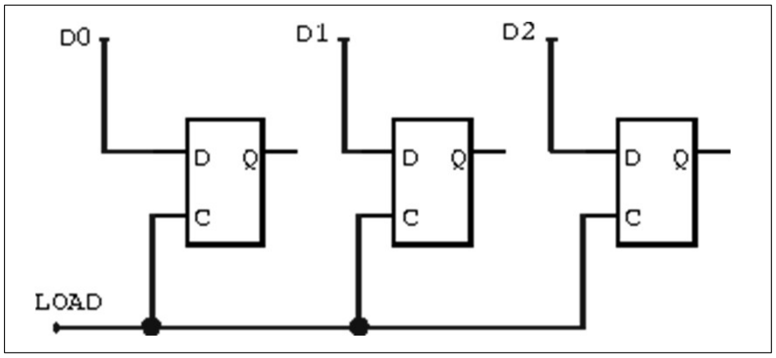


- **TTL 74171 Quad D-type FF with Clear**

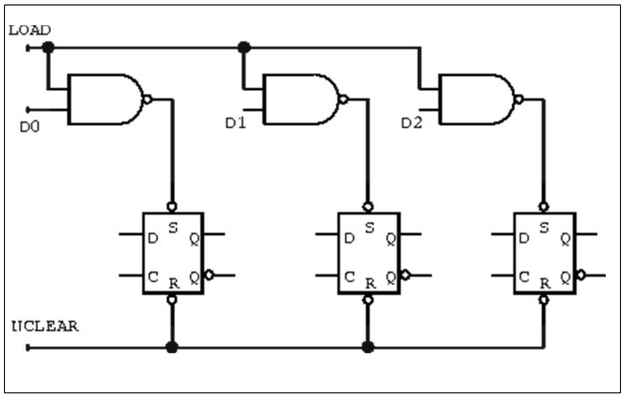
Data Register using J-K FF



A data register using the clocked inputs to D-type flip-flops.



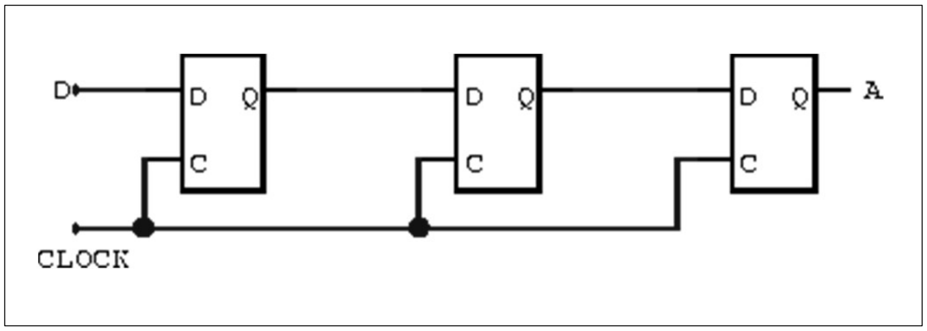
- A more complicated data-loading technique leaves the clocked inputs free but requires a clear-load pulse sequence.

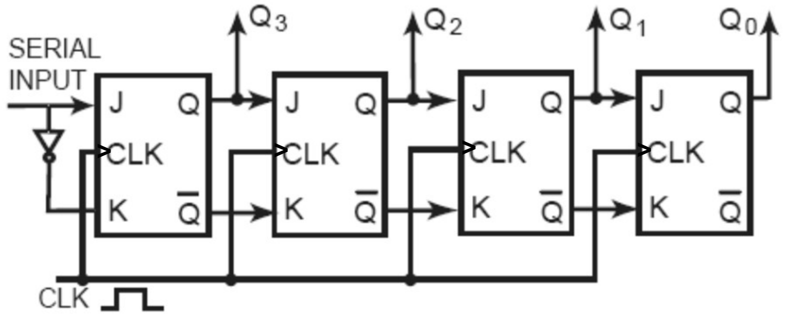


Shift Register

- A shift register is a register in which the contents may be shifted one or more places to the left or right.
- This type of register is capable of performing a variety of functions. It may be used for serial-to-parallel conversion and for scaling binary numbers.

A 3-bit shift register constructed with D flip-flops.





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

SHIFT REGISTER OPERATION:

- **E.G.** Store the word: **0 1 0 1**

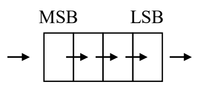
CLK	SERIAL INPUT	Q_3	Q_2	Q_1	Q_0
1	1	1	0	0	0
2	0	0	1	0	0
3	1	1	0	1	0
4	0	**0**	**1**	**0**	**1**

- Assume the stored word initially is **0 0 0 0**.

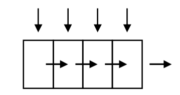
Register Classifications

Where bits come in & go out:

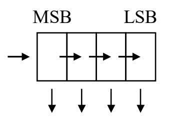
- Serial in-serial out.



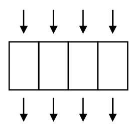
- Serial in-parallel out.



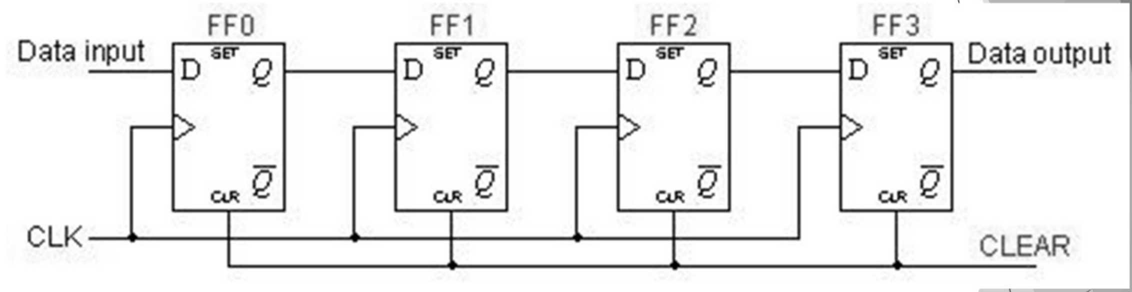
- Parallel in-serial out.



- Parallel in-parallel out.



Serial In - Serial Out Shift Registers

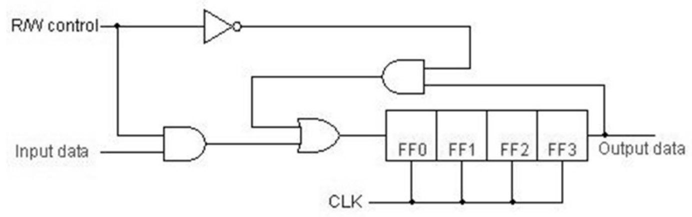


	FF0	FF1	FF2	FF3
CLEAR	0	0	0	0

- In order to get the data out of the register, they must be shifted out serially. This can be done destructively or non-destructively. For destructive readout, the original data is lost and at the end of the read cycle, all flip-flops are reset to zero.

	FF0	FF1	FF2	FF3	
0000	1	0	0	1	0000

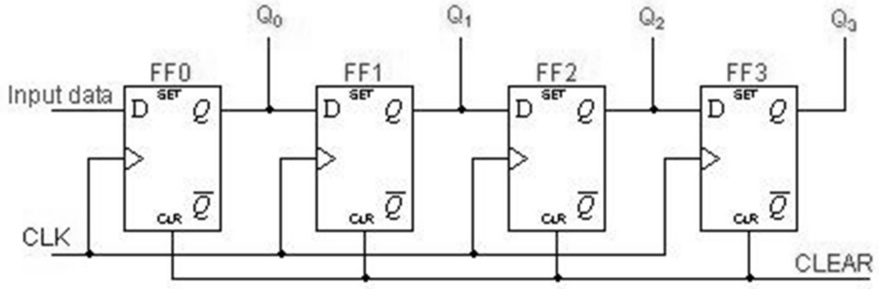
- To avoid the loss of data, an arrangement for a non-destructive reading can be done by adding two AND gates, an OR gate and an inverter to the system. The construction of this circuit is shown below.



WRITE
1001

FF0	FF1	FF2	FF3
0	0	0	0

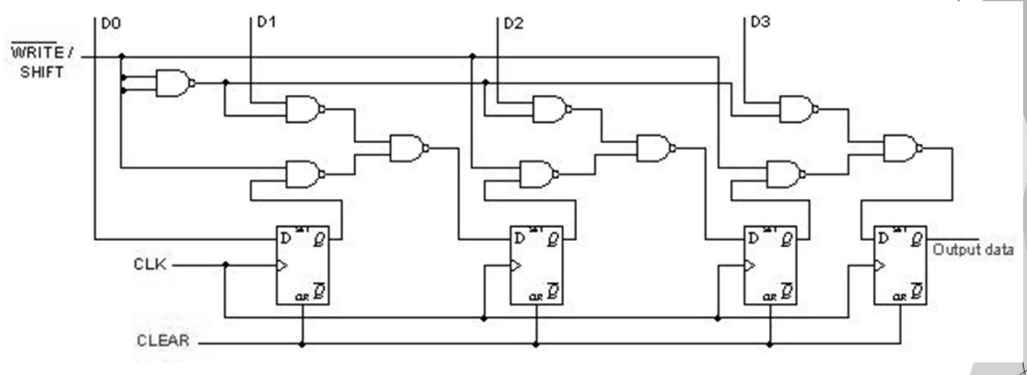
Serial In - Parallel Out Shift Registers



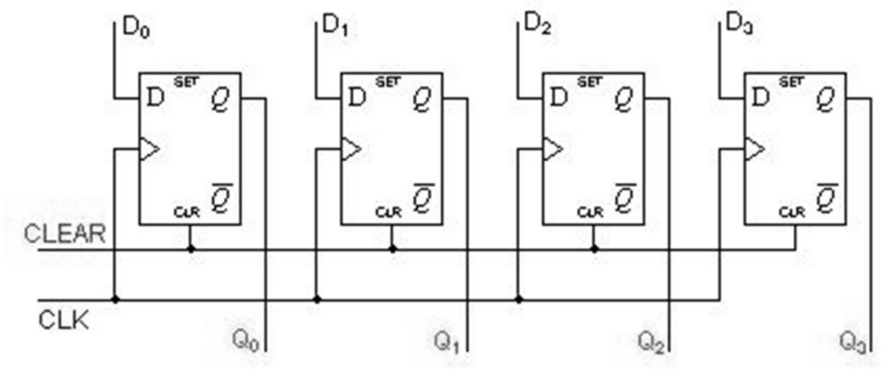
CLEAR
1001

Q0	Q1	Q2	Q3
0	0	0	0

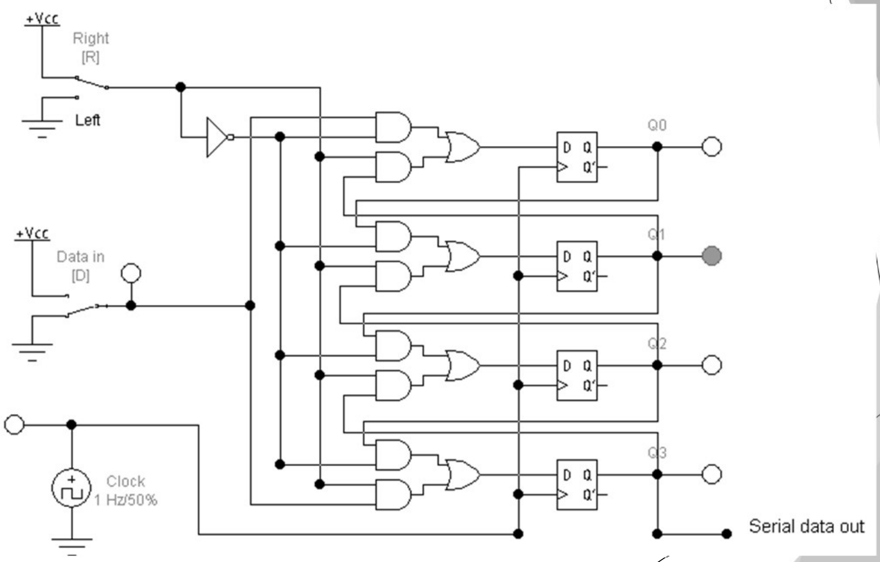
Parallel In - Serial Out Shift Registers



Parallel In - Parallel Out Shift Registers



Bidirectional Shift Register

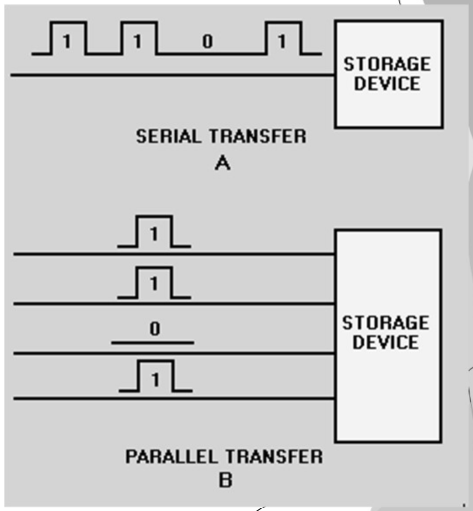


Bidirectional Shift Register

- This is a serial in serial out register than can either shift left or shift right.
- The mode of operation is determined by the the Right/Left control line.
- Clearly more economical than separate left shift and right shift devices. (But only, of course, if we need both operations.)
- The FFs act as a four-element queue, but we can do some permutation of the data by being able, in effect, to swap the head and tail of the queue.

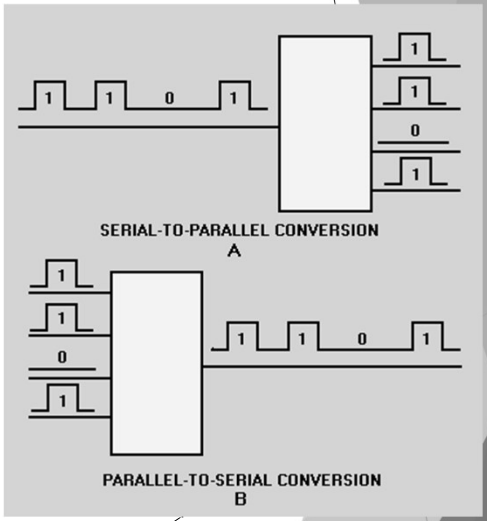
Serial and Parallel Transfers

- The four-bit word 1101 is being transferred to a storage device.
- One control pulse will cause the entire word to be stored .
- Serial transfer takes more time.



Serial and Parallel Conversion

▶ Serial-to-parallel conversion or parallel-to-serial conversion describes the manner in which data is stored in a storage device and the manner in which that data is removed from the storage device.



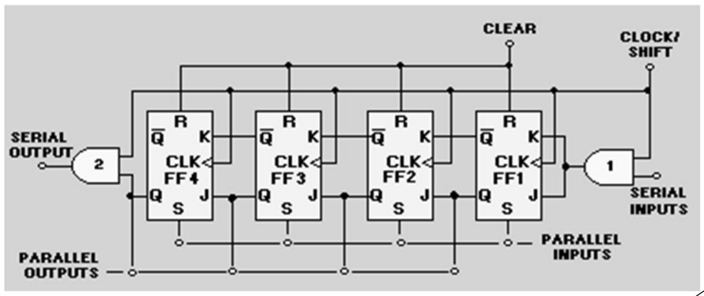
Other Register applications

Scaling

- SCALING means to change the magnitude of a number. Shifting binary numbers to the left increases their value, and shifting to the right decreases their value. The increase or decrease in value is based on powers of 2.
- A shift of one place to the left increases the value by a power of 2, which in effect is multiplying the number by 2. To demonstrate this, let's assume that the following block diagram is a 5-bit shift register containing the binary number 01100.

4-bit Shift Register

- This register is capable of left shifts only.
- Before any operation takes place, a CLEAR pulse is applied to the RESET terminal of each FF to ensure that the Q output is LOW.

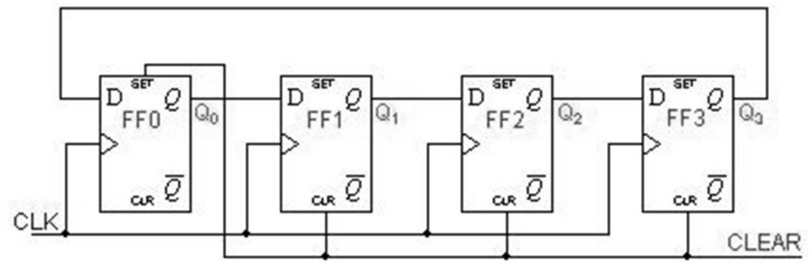


Scaling Operation

- The number to be scaled is loaded into the register either in serial or parallel form. Once the data is in the register, the scaling takes place in the same manner as that for shifting the data for serial output. A single clock pulse will cause each bit of data to shift one place to the left. Remember that each shift is the equivalent of increasing the value by a power of 2. The scaled data is read from the parallel outputs. Care must be taken not to overshift the data to the point that the MSDs are shifted out of the register.

Ring counter

- A shift register can also be used as a primitive counter, a *ring counter*.
- The shifter sequences through the states 1000, 0100, 0010, 0001 and then repeats.
- The four-element ring counter sequences through only 4 states, compared with the 16 states of the four-element binary counter.



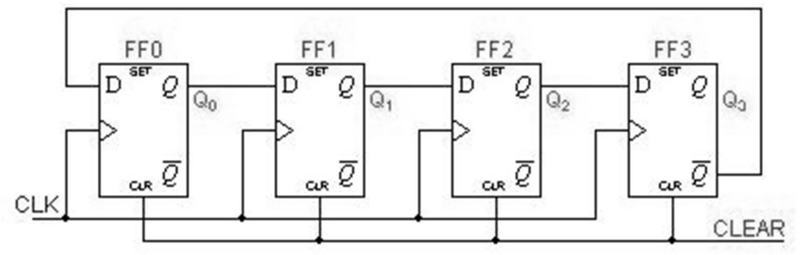
Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	1	0	0	0

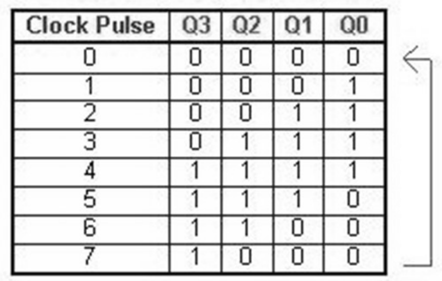
CLEAR

FF0	FF1	FF2	FF3
1	0	0	0

Johnson counter

- are a variation of standard ring counters, with the inverted output of the last stage fed back to the input of the first stage.
- They are also known as twisted ring counters.
- An *n*-stage Johnson counter yields a count sequence of length *2n*, so it may be considered to be a mod-*2n* counter.





Clock Pulse	Q3	Q2	Q1	Q0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	1	1	1
4	1	1	1	1
5	1	1	1	0
6	1	1	0	0
7	1	0	0	0

CLEAR

FF0	FF1	FF2	FF3
0	0	0	0