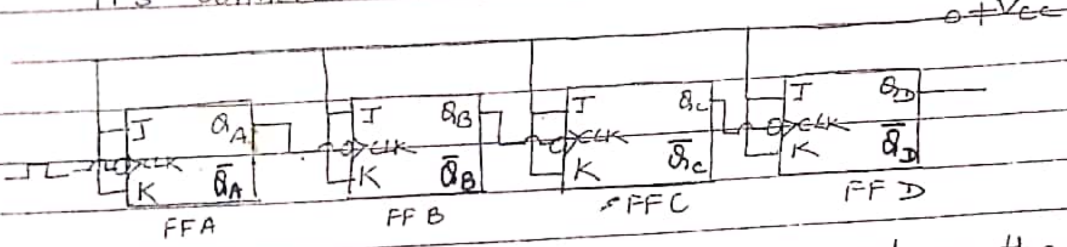
COUNTERS & REGISTERS

Counter :

This is used to count the ~~no~~ number of clock pulses arriving at its input. Counters are of 2 types: Ripple / series / Asynchronous and Parallel / Synchronous.

4 bit binary counter or mod-16 counter :

For this counter, we make use of 4 FFs connected in the manner shown below.



When the o/p of a FF is used as the clock input for the next FF, the counter is called a ripple counter.

A clock signal is fed at the clock input of A FF. The output of A drives B FF, the output of B drives C FF, the output of C drives D FF. All the J, K inputs are tied to +Vcc. The FFs are -vely edge triggered ones. Hence each FF will toggle with a -ve transition at its clock input.

Suppose that initially all the FFs are reset ie $Q_D Q_C Q_B Q_C = 0000$, which is just before the arrival of the 1st clk pulse. When the 1st clock pulse arrives and makes a 1 to 0 transition A FF is triggered. ie Q_A changes from 0 to 1.

This being a +ve going change, B FF is not triggered. Hence FFs C and D also do not change states. The output condition is $Q_D Q_C Q_B Q_A = 0001$.

On the arrival of the 2nd clock pulse, Q_A goes from 1 to 0, which triggers B FF driving its output to 1 from 0. This being a +ve going change, C FF is not triggered. So no change in the output of D FF also.

$\therefore Q_D Q_C Q_B Q_A = 0010$

This continues and at the end of the 15th clock pulse, we have the output condition $Q_D Q_C Q_B Q_A = 1111$.

When the 16th clock pulse arrives & makes a 1 to 0 transition, Q_A changes to 0 from 1. This triggers B, making its output 0. The -ve going change triggers C which changes from 1 to 0. FF D $\therefore$ changes from 1 to 0. Hence at the end of the 16th clock pulse, output condition is $Q_D Q_C Q_B Q_A = 0000$.

Truth table:

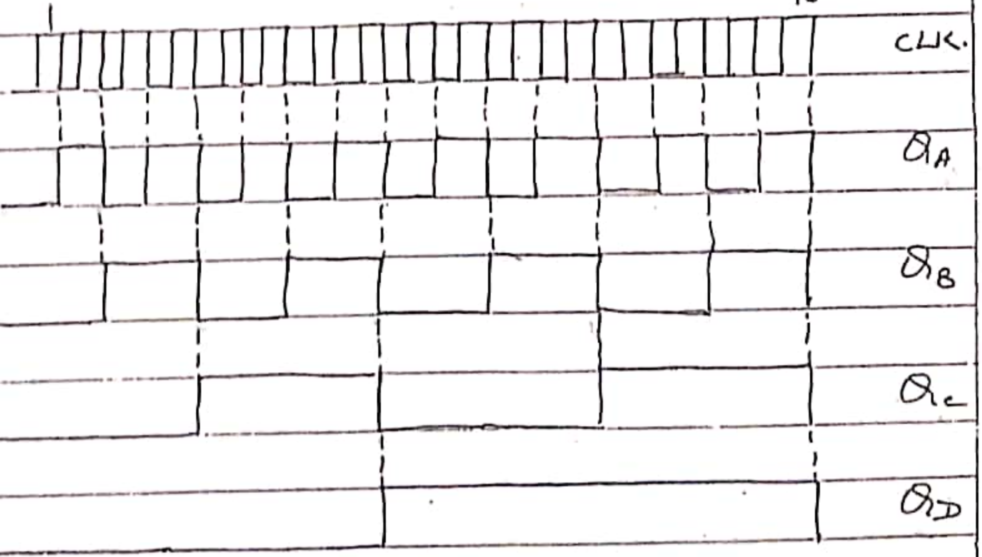
Q_D	Q_C	Q_B	Q_A	Clock pulse
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

Q_D	Q_C	Q_B	Q_A	Clock pulse.
1	0	0	0	8.
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

From the above t. table it can be seen that the output condition of the FFs is a binary no: equivalent to the no: of ~~ve~~ negative clock transitions that have occured.

Modulus of a counter is the total no: of states through which the counter can progress. This 4 FF counter progresses through 16 different states ∴ referred to as a mod-16 counter. $(2^4 = 16)$.

Waveforms:

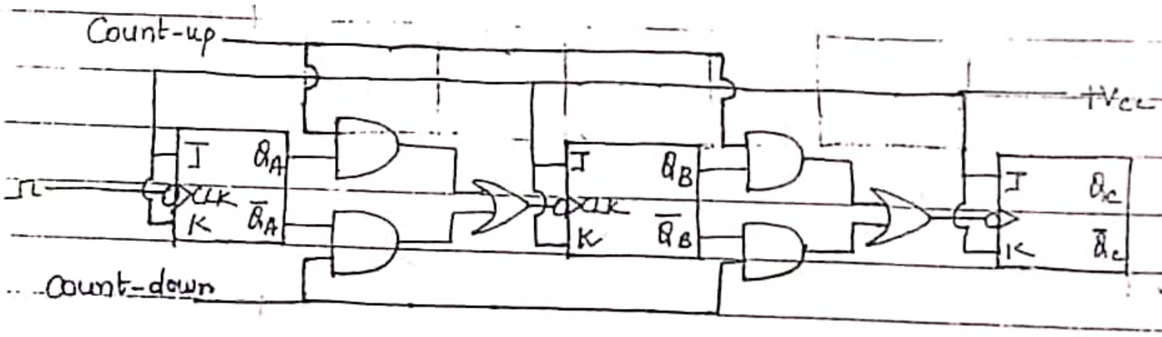


UP/DOWN COUNTERS

UP counter — When the Q output of a FF, serves as the clock input to the next FF, the counter counts from 0000 to 1111 ie. UP counting.

DOWN counter – When the $\bar{Q}$ output of a FF serves as the clock input to the next FF, the counter counts from 1111 to 0000 ie DOWN counting.

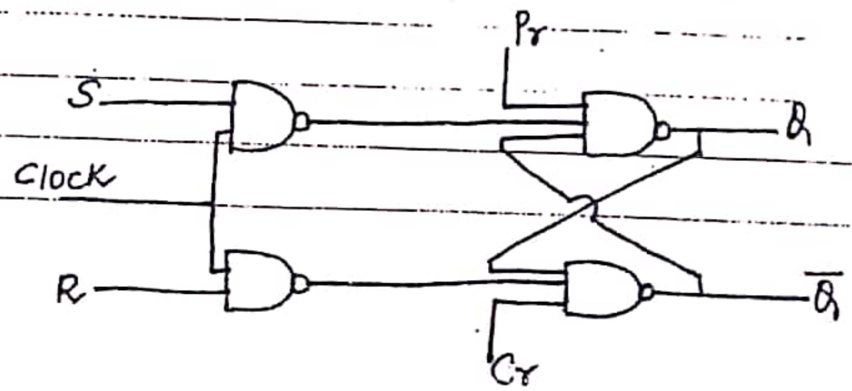
3-bit binary up-down counter :



For this counter to progress through a count-up sequence, it is necessary to trigger each flipflop with the Q output of the previous FF. If the count-down control line is low and the count-up control line high, this will be the case and the counter progresses from count 000 to 111

On the other hand, if count-down is high and count-up is low, each FF will be triggered from the complement side ($\bar{Q}$) of the previous FF. The counter will then be in a count-down mode and will progress from count 111 to 000.

Preset and Clear terminals in an clocked RS FF:-



Preset and Clear action :- In most FFs, ope the clear or reset operation may be done by programming the clear input of an FF to 0 and the preset input to 1. ie CLR = 0, PR = 1.

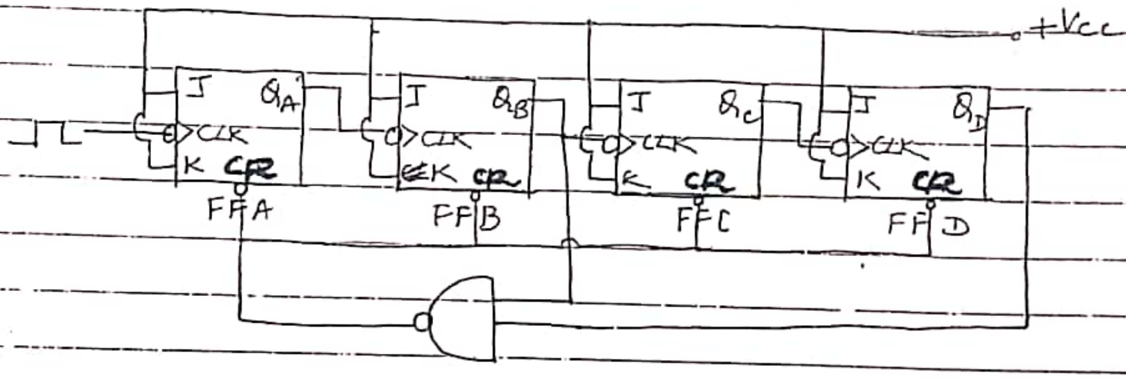
Similarly, if it is required to preset the FF into the 1 state, it is necessary to choose PR = 0, CLR = 1. The preset and clear inputs are also called direct or asynchronous inputs ie they are not in sychronism with the clock, but may be applied at any time in between clock pulses. Once the state of the FF is established asynchronously, the direct inputs must be maintained at PR = 1, CLR = 1, before the next pulse arrives, in order to trigger the FF.

Mod-N counters : In order to construct a counter whose modulus 'N' cannot be expressed as a power of 2, we find the smallest value of 'n' for which $2^n > N$. We then construct the counter starting with a ripple chain of n FFs. A NAND gate is added, whose o/p feeds all the clear inputs of the FFs. Each input to the NAND gate is a FF ~~#~~ output Q which becomes 1 at the count N.

Note : If the modulus of the counter to be constructed can be expressed as a power of 2 ie for eg, in order to construct a mod-8 counter; 8 is expressed as 2^3, then start with a ripple chain of 3 FFs.

Decade counter or Mod-10 counter.

In order to construct such a counter we make use of 4 FFs since the smallest value of 'n' for which $2^n > 10$ is 4. The decimal no: 10 is the binary no: 1010 and hence $Q_D = 1$, $Q_C = 0$, $Q_B = 1$, $Q_A = 0$. The inputs to the NAND gate are : Q_D and Q_B and the complete circuit is shown below.



A clock signal is fed at the clock input of A FF. O/p of A drives B FF, O/p of B drives C FF, O/p of C drives D FF. All the FFs are negatively edge triggered FFs. J and K inputs of all the FFs are maintained high so that the FF simply toggles at every negative transition at its clock input.

Initially all the FFs are reset ie $Q_D Q_C Q_B Q_A = 0000$. When the 1st clock pulse makes a 1 to 0 transition, Q_A changes from 0 to 1. This being a positive going change, B is not triggered. Hence C and D also remain in their initial states. Output condition is $Q_D Q_C Q_B Q_A = 0001$

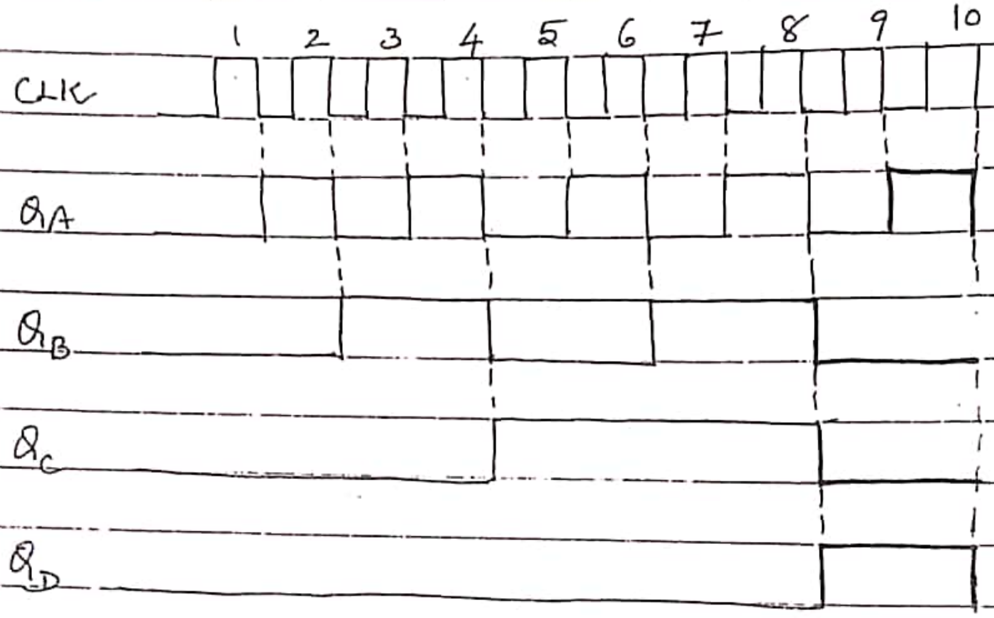
At the end of the 2nd clock pulse, Q_A becomes 0, Q_B changes from 0 to 1. FFs C and D were not triggered so that output condition is $Q_D Q_C Q_B Q_A = 0010$. This continues and at the end of the 9th clock pulse, we have the output condition $Q_D Q_C Q_B Q_A = 1001$

When the 10th clock pulse arrives, Q_A changes from 1 to 0. This being a negative going change, Q_B goes high. C and D FFs do not change states. With both Q_B & Q_D equal to 1, o/p of NAND gate is 0 and all the FFs are cleared. Hence at the end of the 10th clock pulse, the output condition is $Q_D Q_C Q_B Q_A = 0000$.

Truthtable

Q_D	Q_C	Q_B	Q_A	Clock
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	10

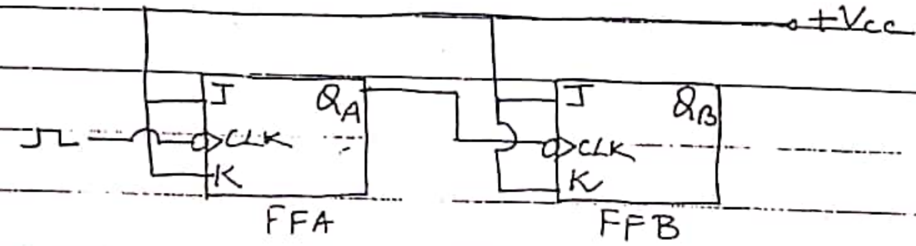
The waveforms showing the action of the counter is shown below.



Types of Counters :—

Ripple / Asynchronous / series counter :—

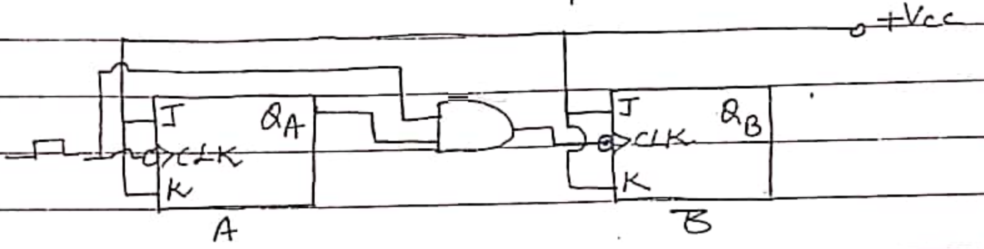
This is a counter in which each FF is triggered by the o/p of the previous FF as shown below, for a mod-4 ripple counter



The amount of time a FF takes to change state after being triggered is called its delay time. Since A FF must change state before it can trigger the B FF, the overall delay time is the sum of the individual delay's ie the counter takes a time equal to the sum of the individual delay times to progress through a count. ∴ this counter has a speed limitation which is its disadvantage. However, this counter is simple and straightforward in operation and construction and usually requires a minimum of hardware.

Parallel / synchronous Counters :-

This is a counter in which all FFs are triggered by the same clock, as shown below for a mod-4 parallel counter.



Truth table :-

Q_B	Q_A	CLK
0	0	0
0	1	1
1	0	2
1	1	3
0	0	4

The clock is applied directly to A FF. Whenever Q_A is high, AND gate is enabled and a clock pulse is passed through the gate to the clock input of B FF. Since both FFs are triggered in synchronism with the clock, the overall delay time is simply equal to the delay time of a single FF. ie an increase in speed of operation is achieved by using this counter. The hardware required is more than that of a ripple counter.

Series and parallel counters are used in combinations to compromise between speed of operation and hardware count.

Problems:

(1) Determine the no: of FFs that would be required to build the foll. counters.

3 (a) Mod 6 (b) Mod 11 4(c) Mod 15

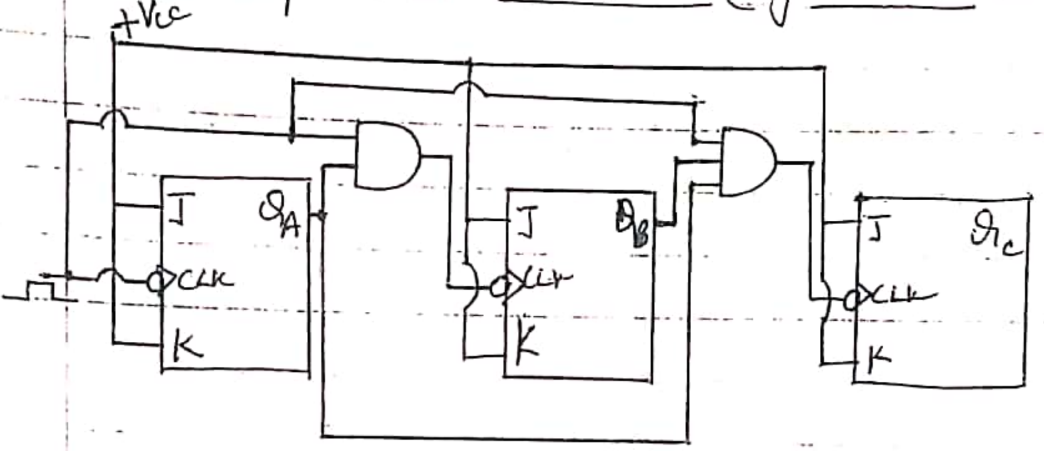
(d) Mod 19 (e) Mod 31 (f) Mod 128.

(2) Draw the block diagrams of mod-14 and mod-5 ripple counters and give their t.tables. Also show the waveform at the output of each FF in the case of a mod-5 counter.

(3) An 8MHz square wave drives a 5 bit binary counter. What is the frequency of the waveform at the output of the last FF of the counter.

(4) Construct a mod 8 parallel counter.

Mod-8 parallel counter
or
3 bit parallel counter (synchronous)



All the JK FFs used are negatively edge triggered ones, with J and K inputs maintained high.
All the three FFs are triggered by the same clock. FF A is triggered whenever the clock makes a negative transition. FF B is triggered for a negative or falling edge of the clock only if $Q_A = 1$. Similarly FF C is triggered for a falling clock edge, provided Q_A and Q_B are both equal to 1.

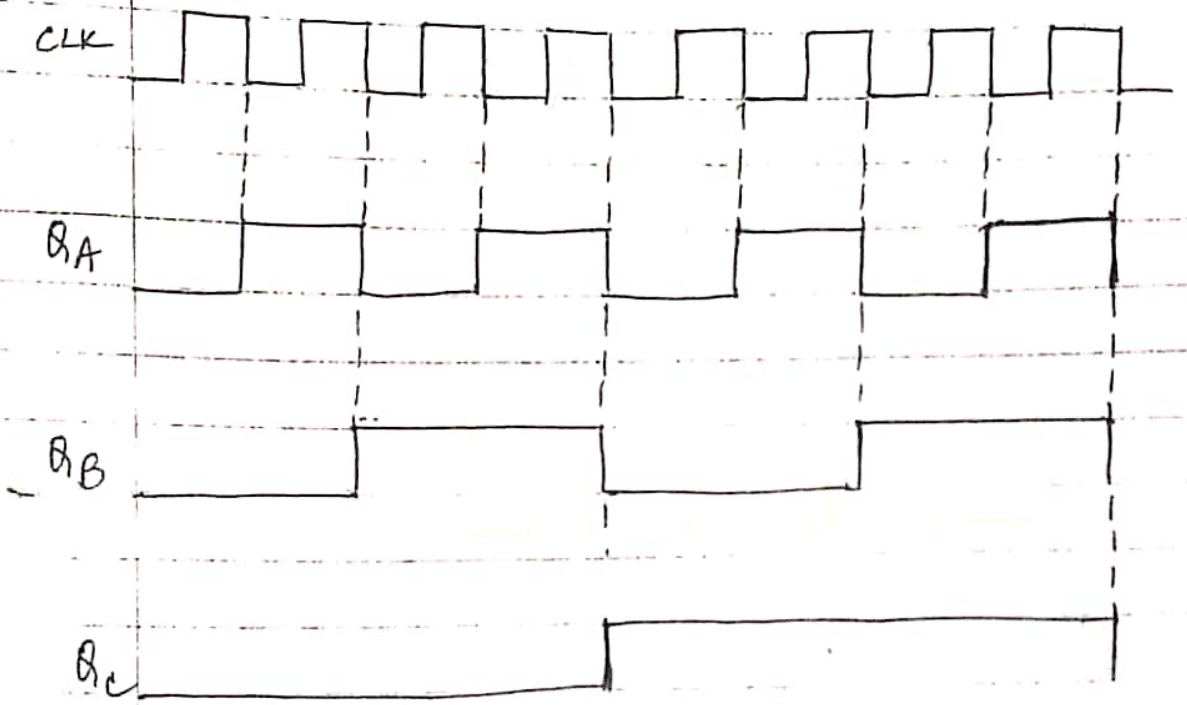
Initially, suppose all the FFs are reset ie $Q_C Q_B Q_A = 000$. Since Q_A and Q_B are both equal to 0, for the first negative clock edge, only FF A is triggered ie FF A toggles and $Q_A = 1$. $\therefore Q_C Q_B Q_A = 001$
Now with $Q_A = 1$, $Q_B = 0$, for the

second clock pulse, FF A and FF B toggles while FF C is not triggered. ie $Q_A = 0$, $Q_B = 1$ and $Q_C = 0$.
This continues and at the end of the third clock pulse $Q_A = 1$, $Q_B = 1$, $Q_C = 0$.
Now for the fourth clock pulse, A FF C FF and B FF triggers giving $Q_A = 0$, $Q_B = 0$ and $Q_C = 1$. In this way, at the end of the seventh clock pulse, $Q_A = Q_B = Q_C = 1$. For the eig'th clock pulse, all the FFs toggle to the reset state. ie counter is reset.

Clock pulse	Q_C	Q_B	Q_A
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
8	0	0	0

DATE: / /

Waveforms.

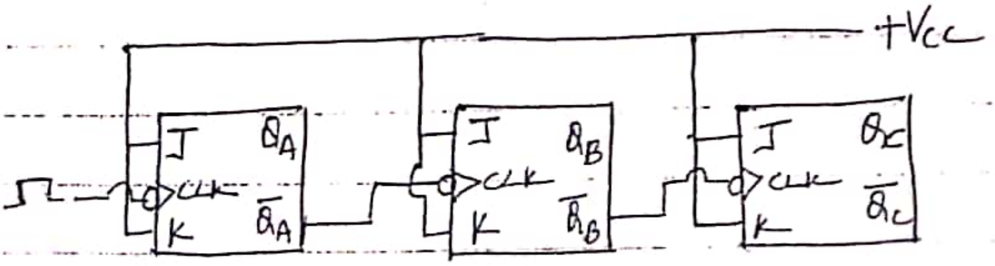


UP COUNTER

NOTE : Explain 3 bit ripple counter or 4 bit ripple counter as given previously, for up or ascending counter.

DOWN COUNTER

NOTE :- Explain 3 bit or 4 bit ripple counter, but with $\overline{Q}$ going as clock input to the next FF in the chain, for down or descending counter.



3 bit down counter

CLK	Q_C	Q_B	Q_A
0	0	0	0
1	1	1	1
2	1	1	0
3	1	0	1
4	1	0	0
5	0	1	1
6	0	1	0
7	0	0	1
8	0	0	0

Working :- Initially, the counter is reset When the first clock pulse arrives A FF toggles and its $\overline{Q}$ output goes from 1 to 0. This negative transition triggers B and its $\overline{Q}$ output goes from 1 to 0. This in turn triggers C FF and its $\overline{Q}$ output goes from 1 to 0. ie $Q_A=Q_B=Q_C=1$ When the second clock pulse arrives, ^ FF toggles and $\overline{Q}_A$ goes from 0 to 1. This positive transition does not trigger B FF and hence C FF also doesnot change state. ∴ $Q_A = 0$, $Q_B = Q_C = 1$.

This continues and at the end of the seventh clock pulse $Q_C = 0$, $Q_B = 0$, $Q_A = 1$ For the eighth clock pulse, A toggles and its output $\overline{Q}_A$ goes from 0 to 1. This does not trigger B and C FFs and hence $Q_B = Q_C = 0$ and $Q_A = 0$.

REGISTERS

A register is simply a group of flipflops that can be used to store a binary number. There must be one flipflop for each bit in the binary number. For instance, a register used to store an 8-bit binary number must have eight flipflops. Naturally the flipflops must be connected such that the binary number can be entered (shifted) into the register and possibly shifted out. A group of flipflops connected to provide either or both of these functions is called a shift register.

The bits in a binary number (called data) can be moved from one place to another in either of two ways. The first method involves shifting the data 1 bit at a time in a serial fashion, beginning with either the MSB or the LSB. This technique is referred to as serial shifting. The second method involves shifting all the data bits simultaneously and is referred to as parallel shifting.

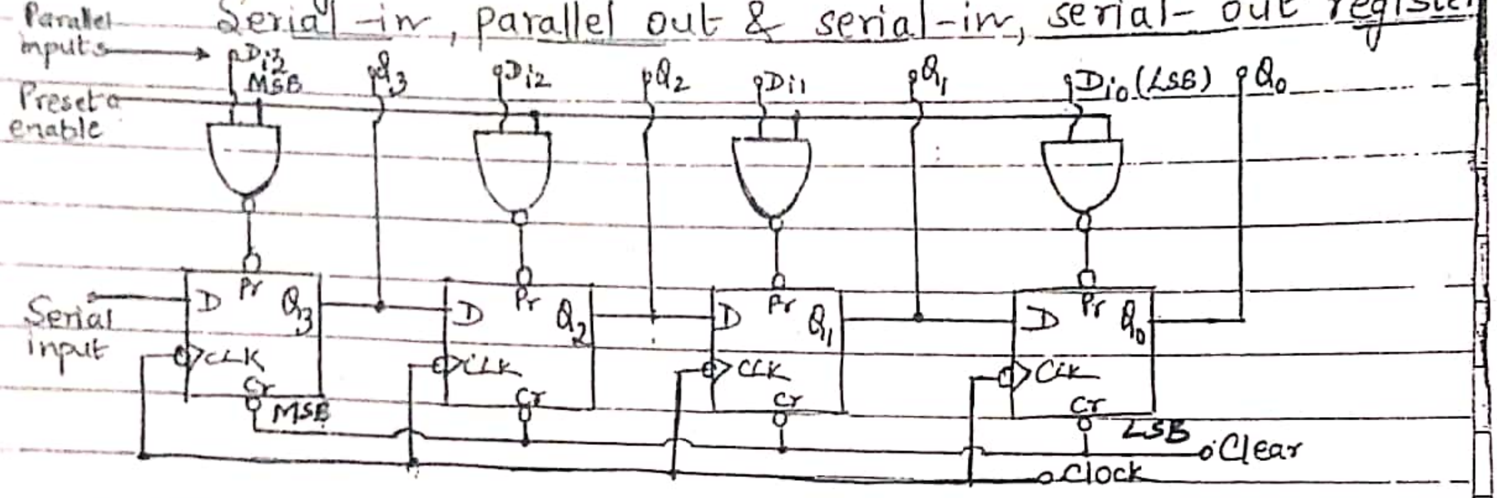
There are two ways to shift data into a register (serial or parallel) and similarly two ways to shift the data out of the register. This leads to the construction of four basic register types :- (i) serial-in, serial-out, (ii) serial-in, parallel-out (iii) parallel-in, serial out (iv) parallel-in, parallel-out

In serial shifting, bits are shifted into the FF's with the occurence of clock pulses either in the right direction (right shift register) or in the left direction (left shift register). In the bidirectional shift register, data can be shifted from left to right as well as in the reverse direction, using the mode control

Shift Register

A 4-bit shift register using 4 D FF's is shown below. This circuit can be used in any of the four modes. ~~The~~ Suppose the binary number 0110 (MSB … LSB) is to be stored in the register.

Serial-in, parallel out & serial-in, serial-out register



For serial input, preset enable is held at 0 so that Pr for every flipflop is 1. Using the clear line, all the flipflops are cleared or reset first. The process of entering the digital word starts with the data input corresponding to the least-significant bit (0) at the serial input and first clock pulse. At the falling edge of the first clock pulse, the output of FF 3 (Q_3) will be 0 and the outputs of all the other FFs are 0 since their inputs are 0. Next, the input corresponding to next bit (1) is applied and at the falling edge of the second clock pulse, the FF outputs will be

$Q_3 = 1$

$Q_2 = Q_1 = Q_0 = 0.$

Similarly, the input corresponding to each bit is applied till the MSB and the bits go on shifting from left to right at the falling edge of each clock pulse

Clock pulse	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0
1	0	0	0	0
2	1	0	0	0
3	1	1	0	0
4	0	1	1	0

At the end of the fourth clock pulse, the outputs of the FFs are $Q_3=0$, $Q_2=1$, $Q_1=1$, $Q_0=0$ which is the same as the number to be stored. The number of clock pulses required for entering the data is the same as the number of bits. The process of entering the data is also referred to as writing into the register.

The data stored can be retrieved (also referred to as reading) in two ways: serial-out and parallel-out. ~~The~~

For parallel out, the contents of the register ($Q_3 Q_2 Q_1 Q_0$) is read simultaneously. ie data is available at $Q_3 Q_2 Q_1 Q_0$ and clock is not required for reading. Register contents can be read any number of times, without destroying or emptying the register.

For serial out, the data in serial form is obtained at Q_0 when clock pulses are applied. The number of clock pulses required will be same as the number of bits (four in this case). In serial output, after the n^{th} clock pulse, for a n-bit word, each FF output is 0. This means that once the data is retrieved, the register is empty.

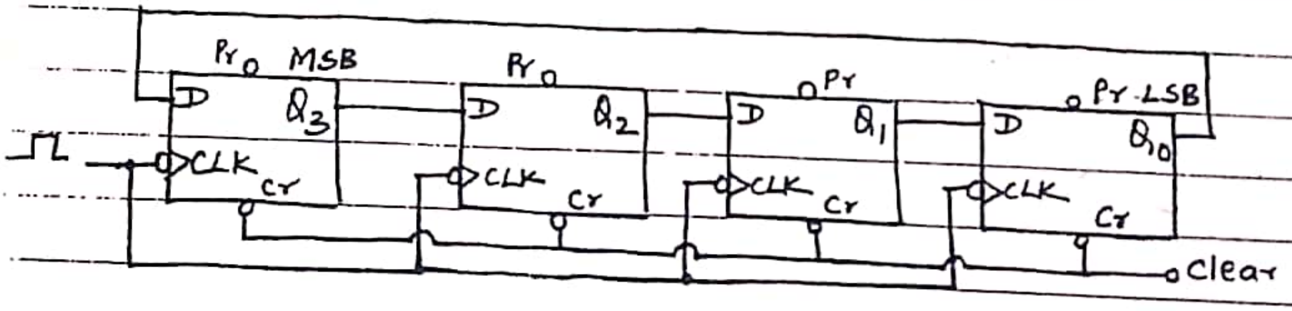
Parallel-in, parallel-out and Parallel-in, serial out register

Refer the above diagram. Data can be entered in the parallel form making use of the preset inputs. After clearing the flipflops, if the data say 0110 is given to the data lines $D_{i3}, D_{i2}, D_{i1}, D_{i0}$ and a '1' is applied to the preset enable input, the data is written into the register. This is referred to as asynchronous loading.

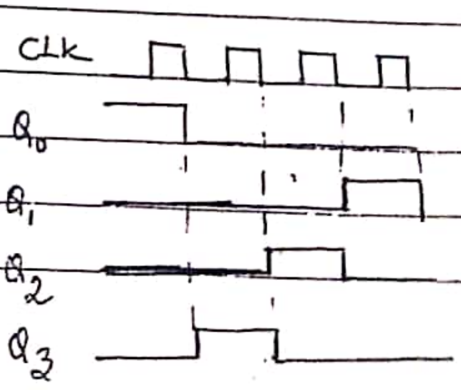
As explained before, the stored word may be read in the serial form at Q_0 by applying 4 clock pulses (serial out) or in the parallel form at Q outputs (parallel out)

Ring Counter

If the serial output Q_0 of the shift register is connected back to the serial input, then an injected pulse will keep cirulating. This circuit is referred to as a ring counter. The pulse is injected by entering 0001 in the parallel form after clearing the flipflops When the clock pulses are applied, this 1 cirulates around the circuit



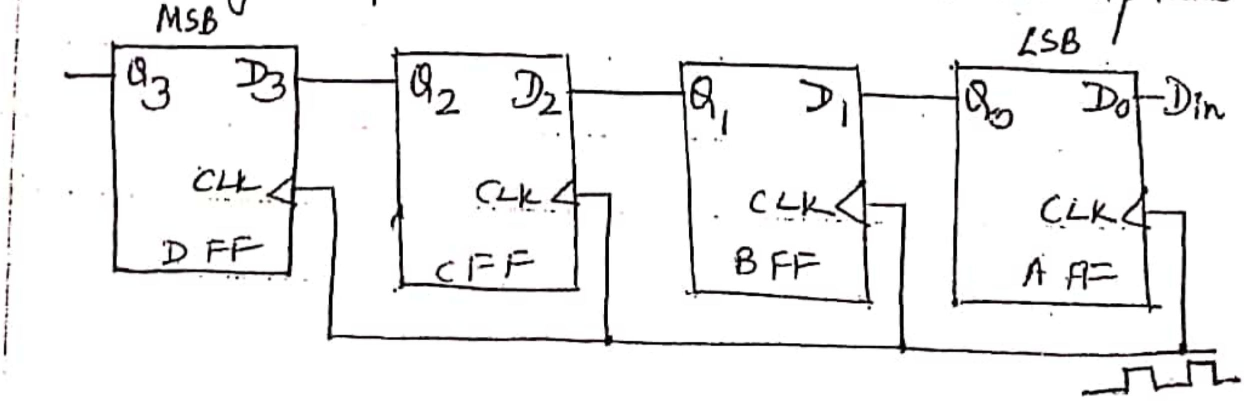
Clock pulse	Q_3	Q_2	Q_1	Q_0
0	0	0	0	1
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1



Suppose initially the register content is $Q_3 Q_2 Q_1 Q_0 = 0001$.
When the first negative clock transition occurs, Q_3 becomes high, while Q_2, Q_1, Q_0 remain low. ie $Q_3 Q_2 Q_1 Q_0 = 1000$
For the second negative clock - transition, only Q_2 goes high and hence $Q_3 Q_2 Q_1 Q_0 = 0100$.
When the third negative clock transition takes place, '1' is shifted to the next FF. ie $Q_3 Q_2 Q_1 Q_0 = 0010$.
For the fourth negative clock edge we again have $Q_3 Q_2 Q_1 Q_0 = 0001$
The stored '1' bit follows a circular path, moving through the flipflops until the final flipflop sends it back to the first FF. This is why the circuit is called a ring counter.

Shift-left Register :

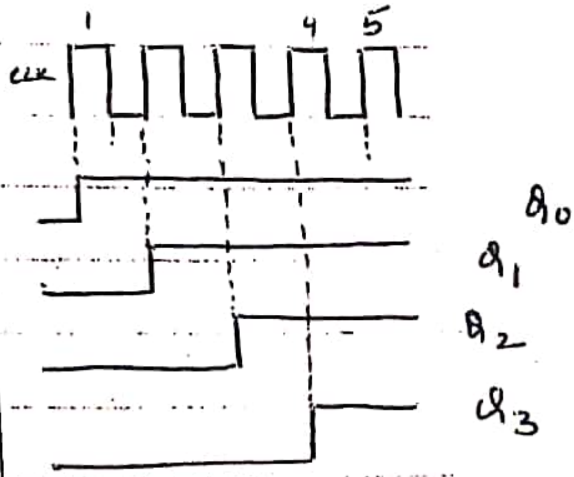
A shift register moves the stored bits left or right. This bit shifting is essential for certain arithmetic and logic operations used in microcomputers



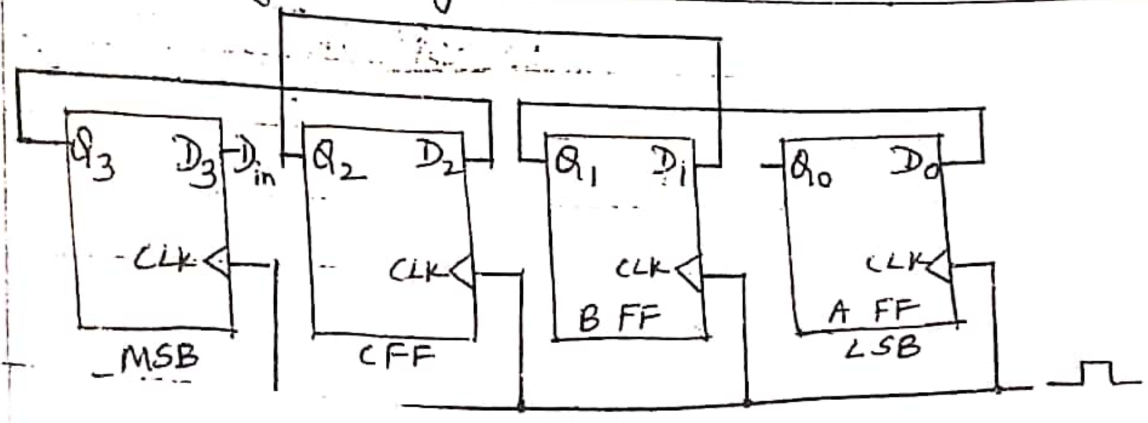
A register where in data movement is towards the left ie from LSB to MSB FF, for the application of clock pulses is called a shift-left register. A shift-left operation is equivalent to multiplying by 2 action.

Suppose Din=1 and the register is initially reset. ie $Q_3 Q_2 Q_1 Q_0$ = 0000. On the arrival of the first ~~clock~~ rising clock edge, A FF sets and the FF outputs read $Q_3 Q_2 Q_1 Q_0$ = 0001. This means D_1 now equals 1 as well as Do. When the next positive clock edge arrives, BFF sets and the register contents become $Q_3 Q_2 Q_1 Q_0$ = 0011. The third positive clock edge results in $Q_3 Q_2 Q_1 Q_0$ = 0111 and the fourth rising clock edge gives $Q_3 Q_2 Q_1 Q_0$ = 1111. Hereafter, as long as Din=1, this stored word remains unchanged.

CLOCK PULSE	Din	Q_3	Q_2	Q_1	Q_0
0	1	0	0	0	0
1	1	0	0	0	1
2	1	0	0	1	1
3	1	0	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1



Shift-Right Register



A register wherein data movement is towards the right ie from MSB to LSB FF, for the application of clock pulses is called a shift-right register.

A shift right operation is equivalent to divide by 2 action.

As shown, each Q output sets up the D input of the preceding flipflop. When the rising clock edge arrives, the stored bits move one position to the right.

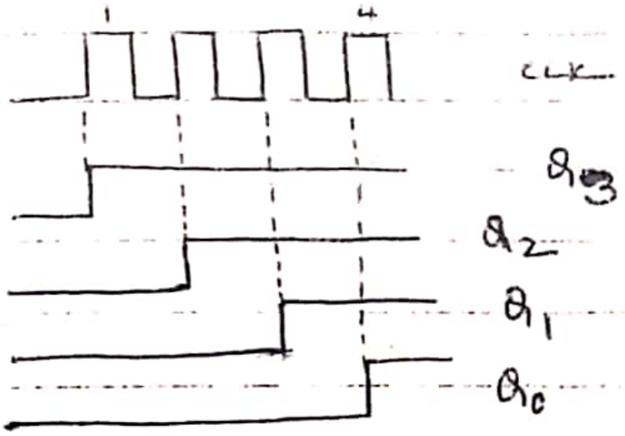
With Din = 1 and register cleared initially, on the arrival of the first positive clock edge, D FF sets and $Q_3 = 1$. $Q_2 = Q_1 = Q_0 = 0$.

On the arrival of the 2nd clock pulse, with Din = 1, $D_2 = 1$, both Q_3 and Q_2 equal 1. $Q_1 = Q_0 = 0$.

For the third positive clock edge, B FF sets giving $Q_1 = 1$ ∴ $Q_3Q_2Q_1Q_0 = 1110$

And, for the fourth positive edge $Q_3Q_2Q_1Q_0 = 1111$

CLOCK PULSE	Din	Q_3	Q_2	Q_1	Q_0
0	1	0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
3	1	1	1	1	0
4	1	1	1	1	1



— X —