CHAPTER 7: REGISTERS AND COUNTERS

ER.HOM NATH TIWARI

PASHCHIMANCHAL CAMPUS,

LAMACHAUR, POKHARA

HOMNATH@WRC.EDU.NP

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

COURSE STRUCTURE

- 7.1. REGISTER TYPES
- 7.2. SISO, SIPO, PISO AND PIPO REGISTORS
- 7.3. DATA TRANSFER TIMING DIAGRAMS
- *7A.* ASYNCHRONOUS COUNTER DESIGN
- 7.5. ASYNCHRONOUS UP, DOWN, UP/DOWN AND MOD-COUNTERS
- 7.6. DECADE BCD COUNTERS
- 7.7. SYNCHRONOUS COUNTER DESIGN
- 7.8. SYNCHRONOUS UP, DOWN, UP/DOWN AND MOD-COUNTERS
- 7.9. COUNTER APPLICATIONS

REGISTERS

A register is a group of flip-flops that can be used to store a binary number. An n-bit register, has n flip-flops and is capable of holding n-bits of information. In addition to flip-flops a register can have a combinational part that performs data-processing tasks.

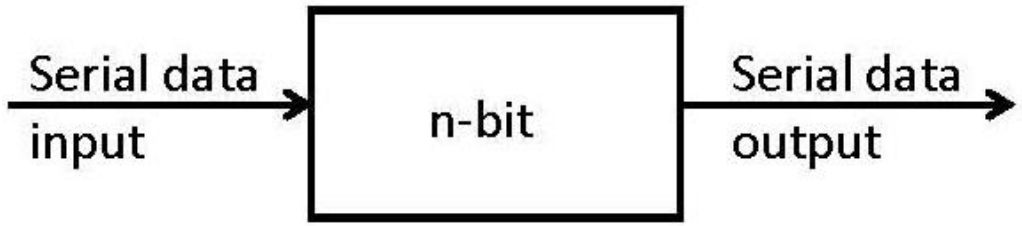
7.1 Types of Registers

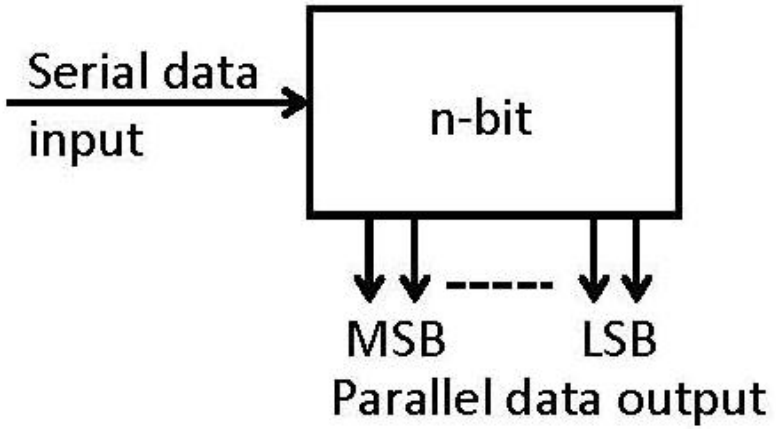
The flip-flops must be connected such that the binary number can be entered and shifted into the register and possibly shifted out. *A register capable of shifting its binary contents either to the left or to the right is called a shift register*. The shift register permits the stored data to move from a particular location to some other location within the register. Registers can be designed using discrete flip-flops(S-R, J-K, and D-type).

The data in a shift register can be shifted in two possible ways: (*a*) serial shifting and (*b*) parallel shifting. The serial shifting method shifts one bit at a time for each clock pulse in a serial manner, beginning with either LSB or MSB. On the other hand, in parallel shifting operation, all the data (input or output) gets shifted simultaneously during a single clock pulse. Hence, we may say that parallel shifting operation is much faster than serial shifting operation.

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. All of the four configurations are commercially available as TTL MSI/LSI circuits. They are:

1. Serial in/Serial out (SISO)
2. Serial in/Parallel out (SIPO)
3. Parallel in/Serial out (PISO)
4. Parallel in/Parallel out (PIPO)





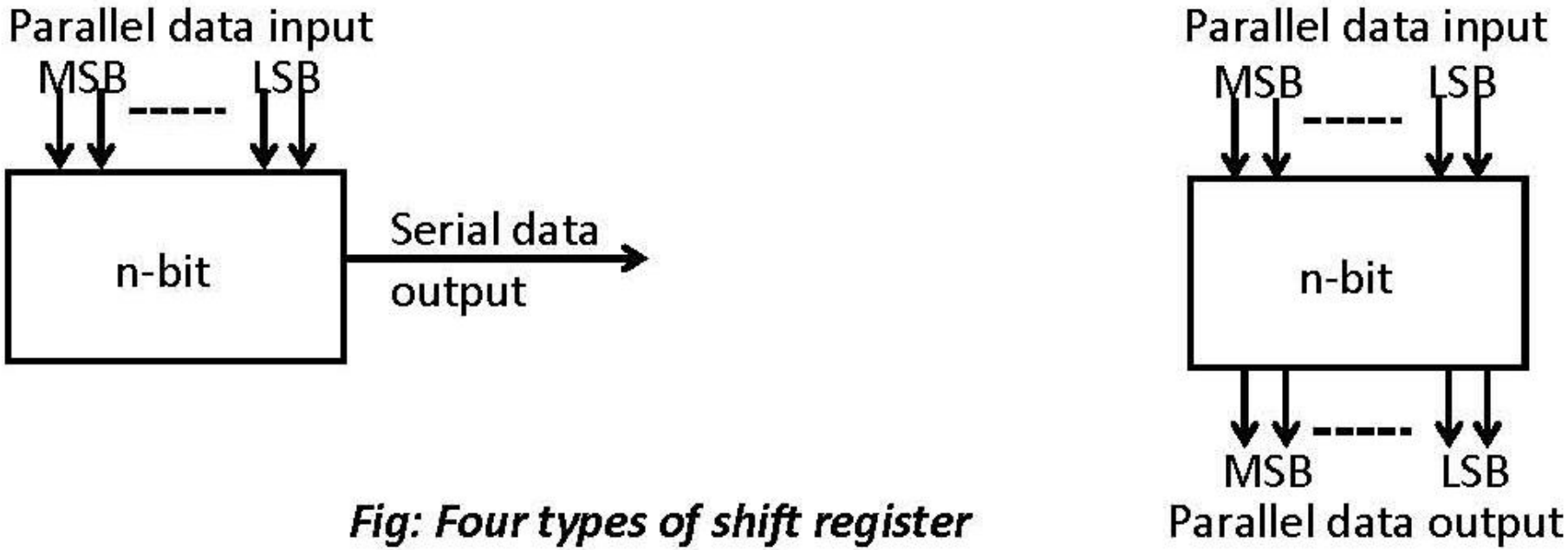


Fig: Four types of shift register

There are two ways for shifting. a) Shift-right & b) Shift-left.
Here we only talk about *Shift-right* register.

7.2 Serial in/Serial out (SISO)

The serial in serial out shift register accepts data serially that is one bit at a time on a single line. It produces the stored information on its output also in serial form.

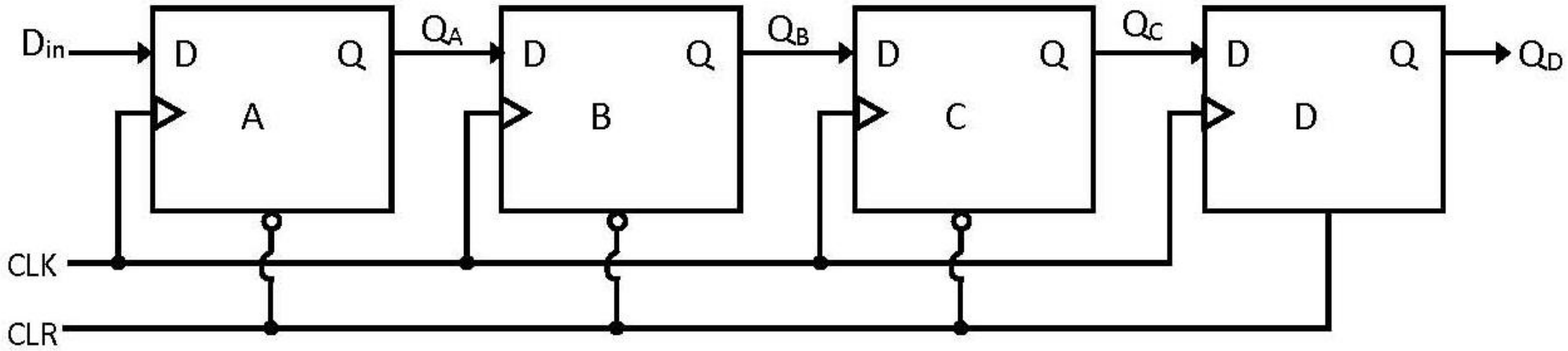


Fig: 4-bit serial in serial out circuit

The clock pulse is applied to all the flip-flops simultaneously. When the clock pulse is applied, each flip-flop is either set or reset according to the data available at that point of time at the respective inputs of the individual flip-flops. Hence the input data bit at the serial input line is entered into flip-flop A by the first clock pulse. At the same time, the data of stage A is shifted into stage B and so on to the following stages. For each clock pulse, data stored in the register is shifted to the right by one stage. New data is entered into stage A, whereas the data present in stage D are shifted out (to the right).

For example, consider that all the stages are reset and a logical input 1011 is applied at the serial input(LSB to MSB) line connected to stage A. The data after four clock pulses is shown in table.

Timing pulse	Q_A	Q_B	Q_C	Q_D	*Serial output at Q_D*
Initial value	0	0	0	0	0
After 1st clock pulse	1	0	0	0	0
After 2nd clock pulse	1	1	0	0	0
After 3rd clock pulse	0	1	1	0	0
After 4th clock pulse	1	0	1	1	1

Let us now illustrate the entry of the 4-bit number 1011 into the register, beginning with the right-most bit. A 1 is applied at the serial input line, making D = 1. As the first clock pulse is applied, flip-flop A is SET, thus storing the 1. Next, a 1 is applied to the serial input, making D = 1 for flip-flop A and D = 1 for flip-flop B also, because the input of flip-flop B is connected to the Q_A output.

When the second clock pulse occurs, the 1 on the data input is “shifted” to the flip-flop A and the 1 in the flip-flop A is “shifted” to flip-flop B. The 0 in the binary number is now applied at the serial input line, and the third clock pulse is now applied. This 0 is entered in flip-flop A and the 1 stored in flip-flop A is now “shifted” to flip-flop B and the 1 stored in flip-flop B is now “shifted” to flip-flop C. The last bit in the binary number that is the 1 is now applied at the serial input line and the fourth clock pulse is now applied. This 1 now enters the flip-flop A and the 0 stored in flip-flop A is now “shifted” to flip-flop B and the 1 stored in flip-flop B is now “shifted” to flip-flop C and the 1 stored in flip-flop C is now “shifted” to flip-flop D. Thus the entry of the 4-bit binary number in the shift-right register is now completed.

From the third column of the above table we can get the serial output of the data that is being entered in the register. We find that after the first, second, and the third clock pulses the

output at the serial output line *i.e.*, Q_D is 0. After the fourth clock pulse the output at the serial output line is 1. If we want to get the total data that we have entered in the register in a serial manner from Q_D, then we have to apply another three clock pulses. After the fifth clock pulse we will gate another 1 at Q_D. After the sixth clock pulse the output at Q_D will be 0 and after the seventh clock pulse the output at Q_D will be 1. In this process of the fifth, sixth, and the seventh clock pulses if no data is being supplied at the serial input line then the A, B, and C flip-flops will again be RESET with output 0.

Timing Diagram

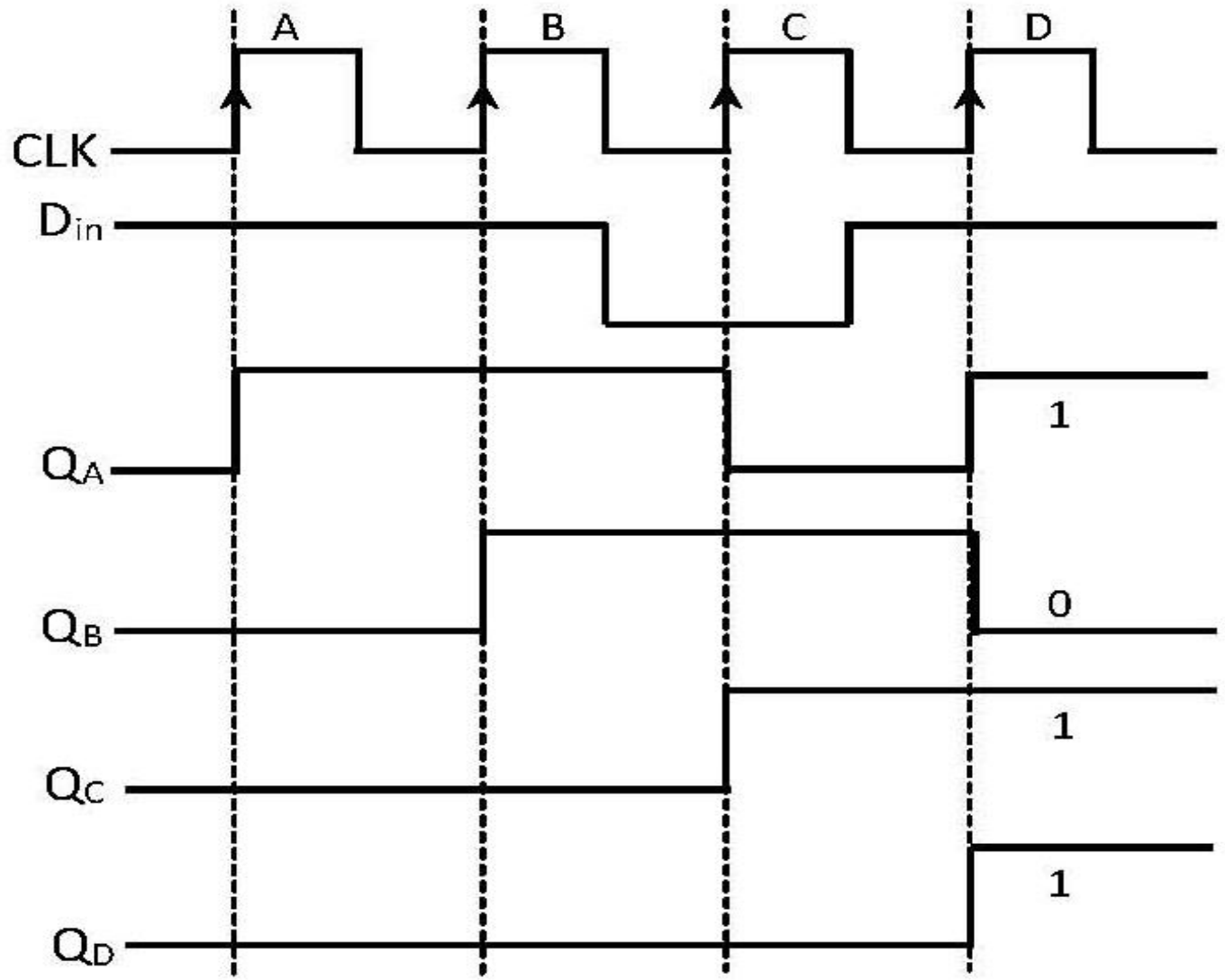


Fig: waveforms of 4-bit serial input shift register

7.3 Serial in/Parallel out (SIPO)

In this type of register, the data is shifted in serially, but shifted out in parallel. To obtain the output data in parallel, it is required that all the output bits are available at the same time. This can be accomplished by connecting the output of each flip-flop to an output pin. Once the data is stored in the flip-flop the bits are available simultaneously.

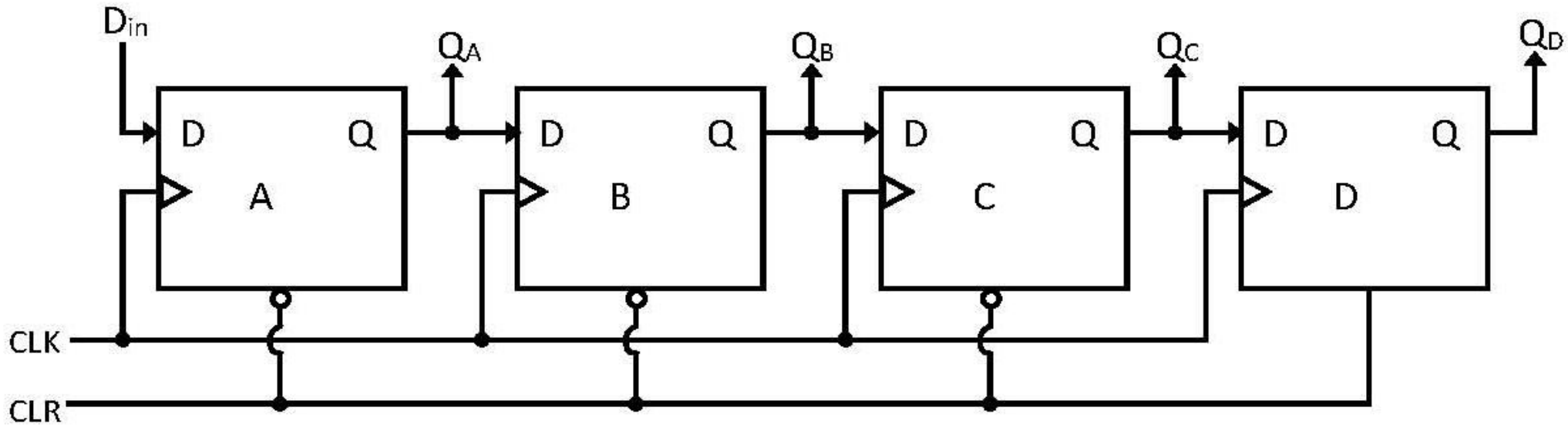


Fig: 4-bit serial in parallel out circuit

The above details of the serial-in/parallel-out shift register are fairly simple. It looks like a serial-in/serial-out shift register with taps added to each stage output. Serial data shifts in at D_{in} (Serial Input). After a number of clocks equal to the number of stages, the first data bit in appears at Q_D

in the above figure. In general, there is no serial out pin. The last stage (Q_D above) serves as serial out and is cascaded to the next package if it exists.

If a serial-in/parallel-out shift register is so similar to a serial-in/ serial-out shift register, why do manufacturers bother to offer both types? Why not just offer the serial-in/parallel-out shift register? They actually only offer the serial-in/parallel-out shift register, as long as it has no more than 8-bits. Note that serial-in/serial-out shift registers come in bigger than 8-bit lengths of 18 to 64-bits. It is not practical to offer a 64-bit serial-in/parallel-out shift register requiring that many output pins.

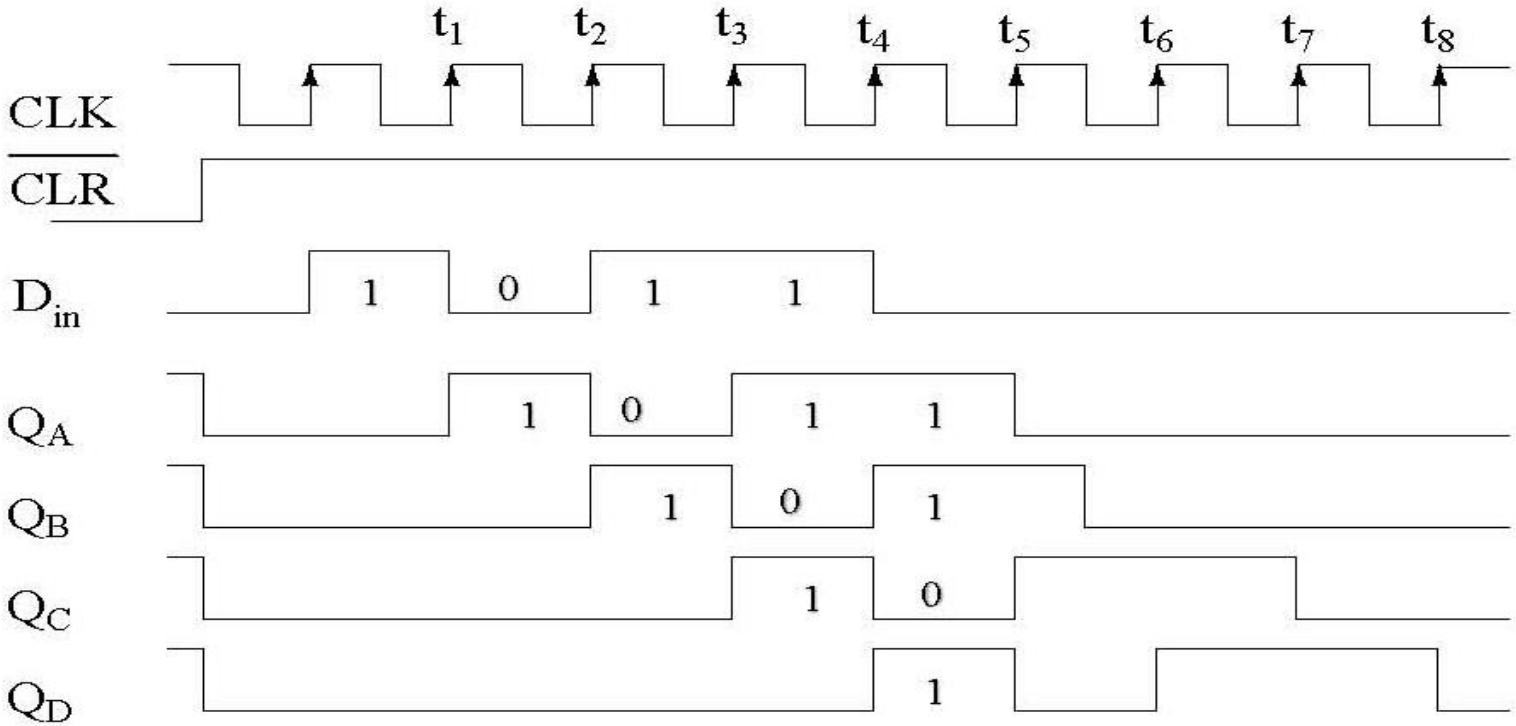


Fig: waveforms of 4-bit Serial in/Parallel out shift register

The shift register has been cleared prior to any data by *CLR'*, an active low signal, which clears all type D Flip-Flops within the shift register. Note the serial data *1011* pattern presented at the D_{in} input. This data is synchronized with the clock *CLK*. This would be the case if it is being shifted in from something like another shift register, for example, a parallel-in/serial-out shift register (not shown here). On the first clock at t_1, the data *1* at D_{in} is shifted from *D* to *Q* of the first shift register stage. After t_2 this first data bit is at Q_B. After t_3 it is at Q_C. After t_4 it is at Q_D. Four clock pulses have shifted the first data bit all the way to the last stage Q_D. The second data bit a 0 is at Q_C after the 4th clock. The third data bit a 1 is at Q_B. The fourth data bit another 1 is at Q_A. Thus, the serial data input pattern 1011 is contained in (Q_D Q_C Q_B Q_A).

It is now available on the four outputs. It will available on the four outputs from just after clock t_4 to just before t_5. This parallel data must be used or stored between these two times, or it will be lost due to shifting out the Q_D stage on clocks t_5 to t_8 as shown above.

7.4 Parallel in/Serial out (PISO)

Parallel-in/ serial-out shift registers do everything that the previous serial-in/serial-out shift registers do plus input data to all stages simultaneously. The parallel-in/ serial-out shift register stores data, shifts it on a clock by clock basis, and delays it by the number of stages times the clock period. In addition, parallel-in/ serial-out really means that we can load data in parallel into all stages before any shifting ever begins. This is a way to convert data from a *parallel* format to a *serial* format. By parallel format we mean that the data bits are present simultaneously on individual wires, one for each data bit as shown below. By serial format we mean that the data bits are presented sequentially in time on a single wire or circuit as in the case of the "data out" on the diagram below.

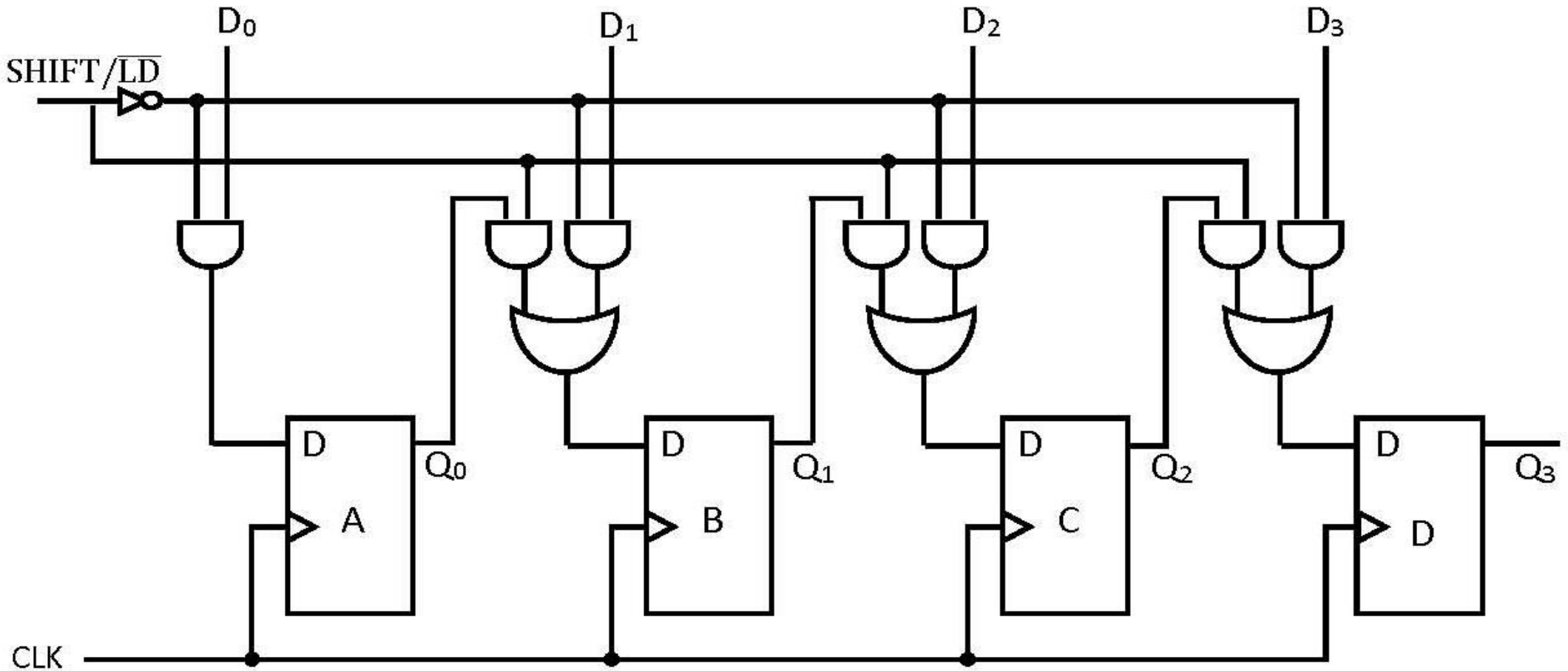


Fig: 4-bit parallel in serial out circuit

The circuit shows 4-bits PISO shift register D_0, D_1, D_2 and D_3 are parallel inputs where, D_0 is MSB and D_3 is LSB. To write data in, the mode control line is taken to LOW and the data is clocked in. The data can be shifted when the mode control line is HIGH as SHIFT is active high.

Timing waveform for 4-bit PISO shift register when data bits $D_0D_1D_2D_3 = 0101$ is entered. Assume D input remains a 1.

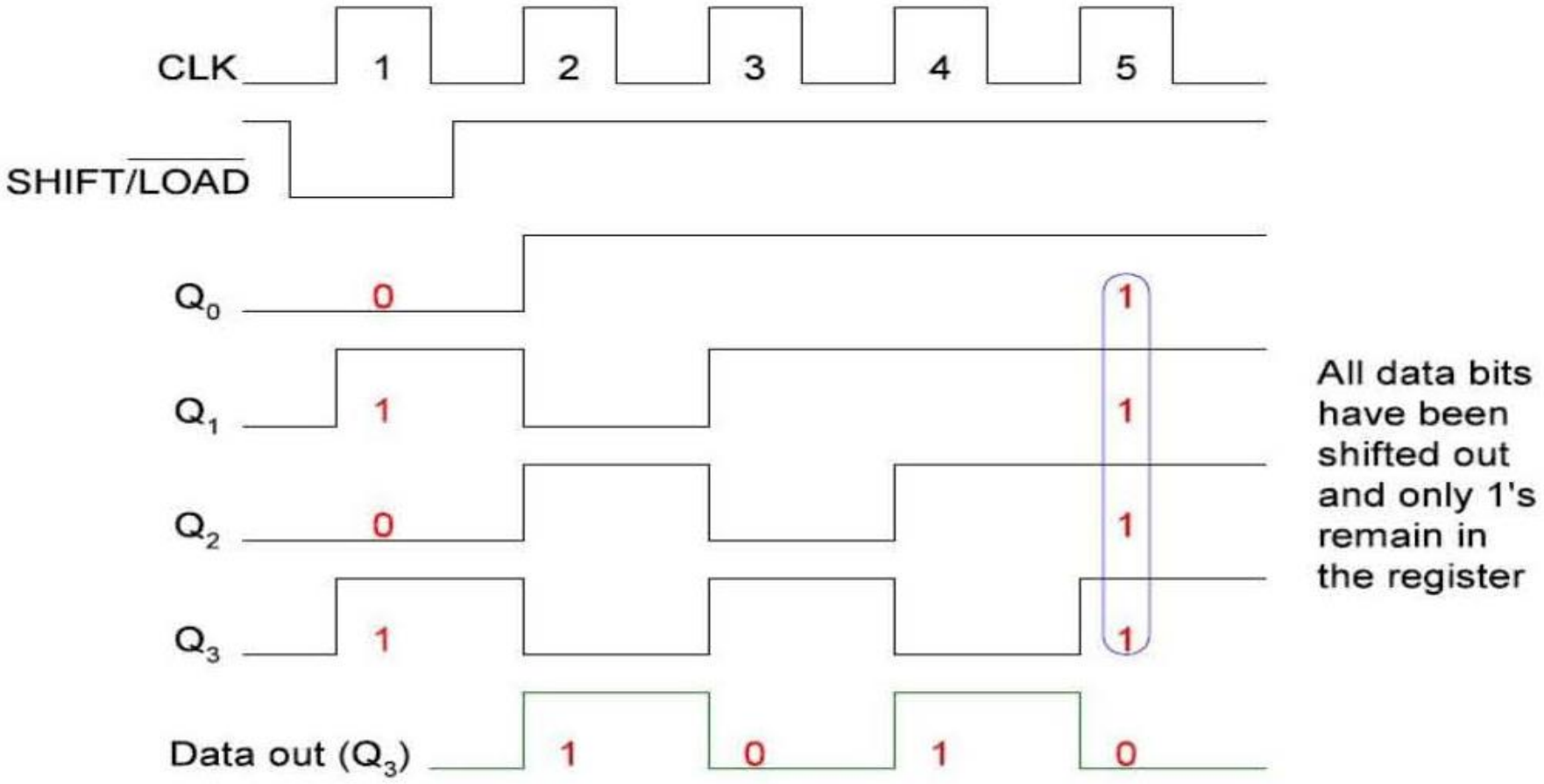


Fig: waveforms of PISO

Why provide serial in and serial out pins on a shift register? These connections allow us to cascade shift register stages to provide large shifters than available in a single IC (Integrated Circuit) package. They also allow serial connections to and from other ICs like microprocessors.

7.5 Parallel in/Parallel out (PIPO)

In parallel-in/parallel-out registers the data will appear immediately at the output after a single clock pulse and there is no clock delay. This PIPO register are also known as general register because of no shifting operation.

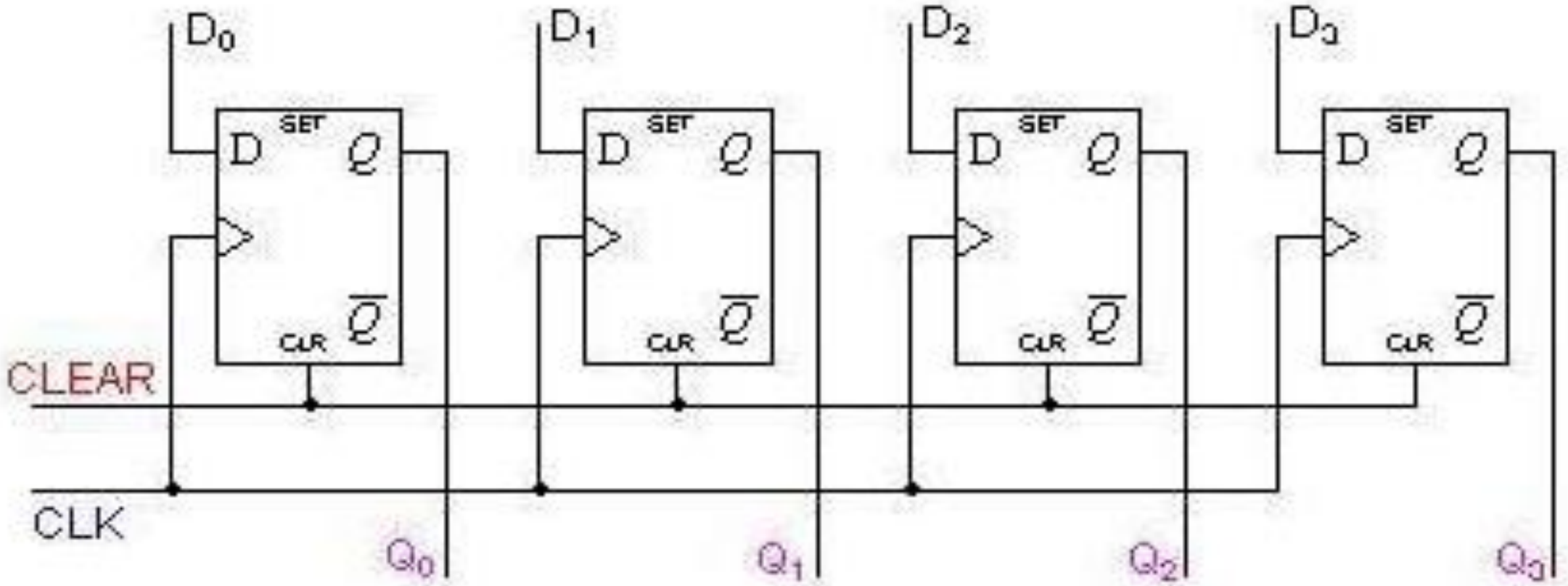


Fig: parallel in parallel out circuit

The D's are the parallel inputs and the Q's are the parallel outputs. Once the register is clocked, all the data at the D inputs appear at the corresponding Q outputs simultaneously. The above circuit is a four-bit parallel-in/parallel-out shift register constructed by D flip-flop switch $D_0=1$, $D_1=0$, $D_2=1$ and $D_3=0$.

The clock waveform can be drawn as follows where the inserted data bits are shifted out in parallel.

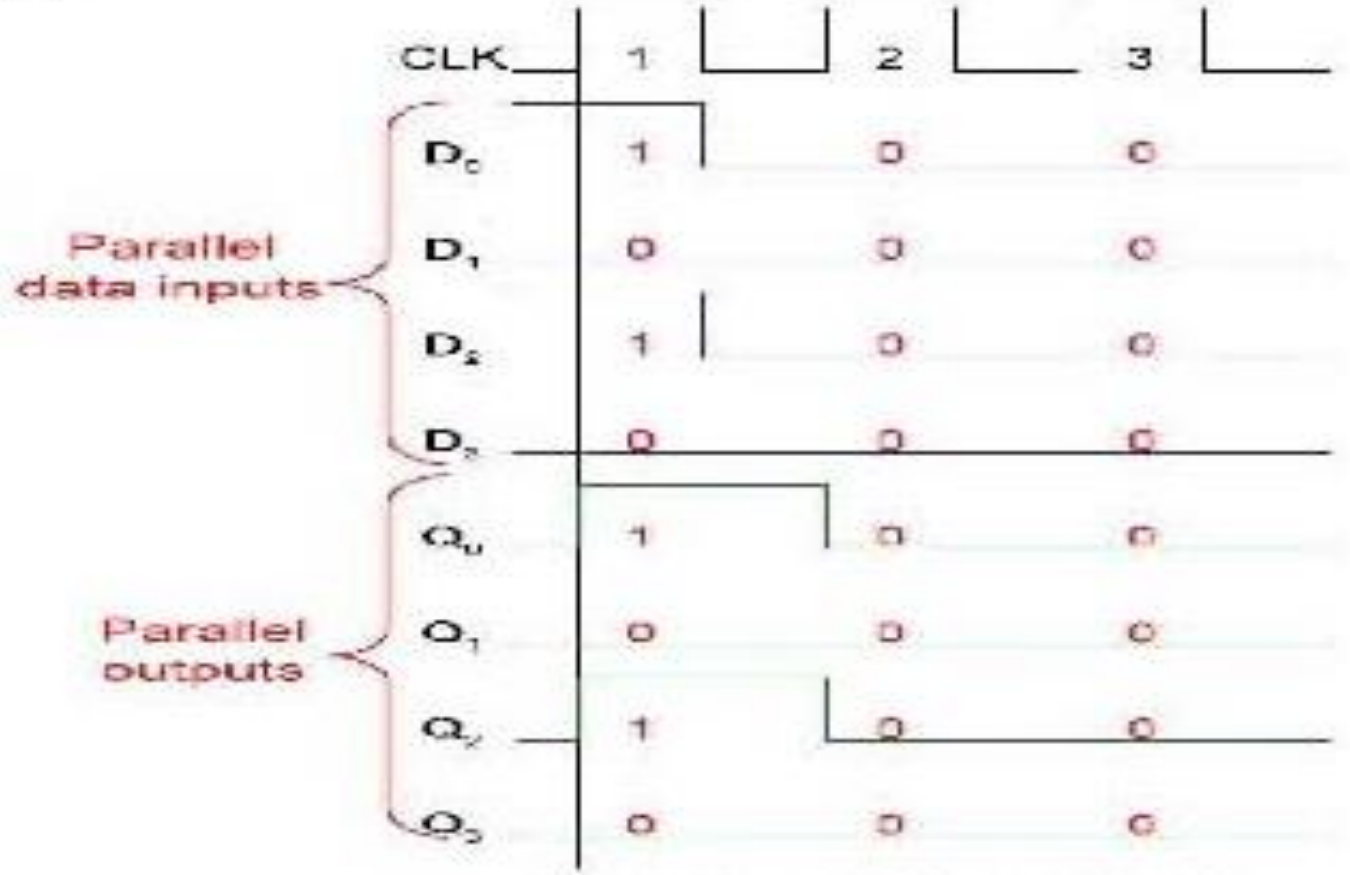


Fig: clock waveforms of PIPO

COUNTERS

A counter is essentially a register that goes through a predetermined sequence of states upon the application of input pulse. The gate in a counter are connected in such a way as to produce a prescribed sequence of binary state in a register. A counter driven by clock can be used to count the number of clock cycles. It can be used as an important for measuring time and therefore period and frequency. There are two different types of counter they are:

1) Synchronous counter
2) Asynchronous counter

8.1 Asynchronous counter:

In asynchronous counter each flip flop is triggered by previous flip flop. Ripple counter is one of the example. It is also called as the serial counter. It require minimum hardware

❖ Ripple counter:

In ripple counter the flip flop output transition serves as a source of triggering other flip flop. In other words, the clock pulse input of all flip flop except the first are triggered not by the incoming pulse, but rather by the transition that occurs in other flip flop. A binary ripple counter can be constructed by using clocked JK flip flop.

Below figure is a three negative edge triggered JK flip flop connected in cascade.

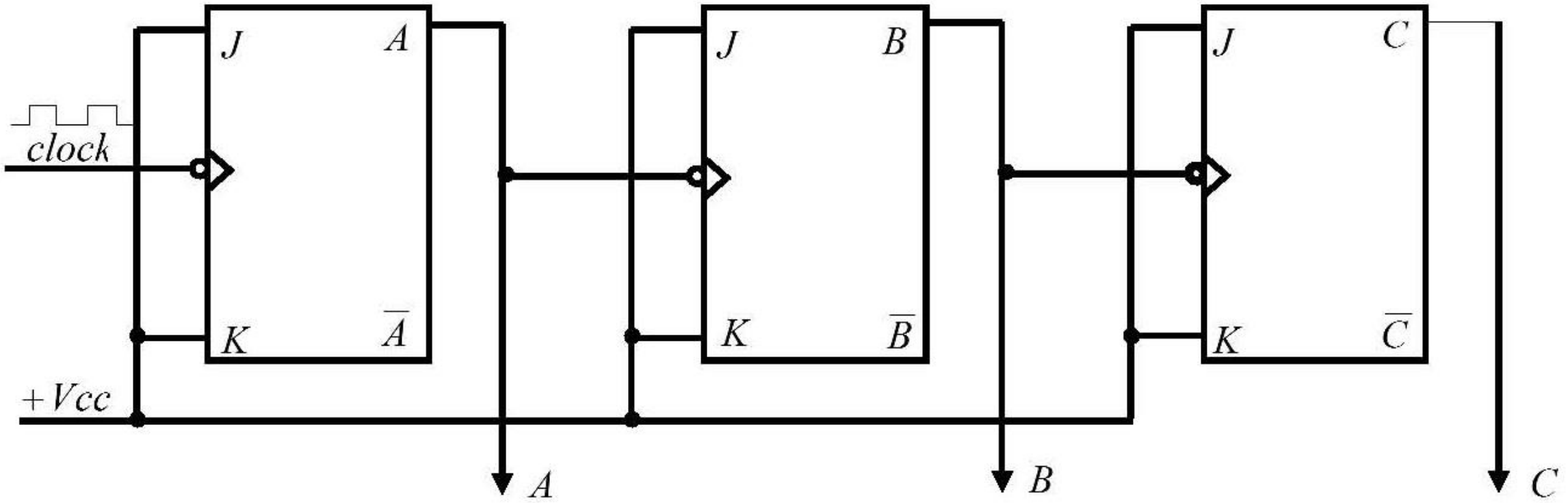


Fig: three bit binary ripple

Here in figure, the A flip flop must change state before it trigger the B flip flop and B flip flop has to change the state before it can trigger the C flip flop. The trigger moves through the flip flop like ripple of the water. Because of this the overall propagation delay time is the sum off all individual delays.

Let us assume all flip flop are at reset to produce 0 output. If we consider A to be LSB and C to be MSB. The content of the counter initially be 000. When there is negative edge triggered then the flip flop A will change state. This is indicated by downward arrow. At the point a in the time line, A goes high, at the point b it goes back low, at the point c it again goes back high, at the point d it goes low and so on. The wave form at the output of flip flop A is one half the clock frequency.

Since A acts as the clock for B, each time the wave form at A goes low, flip flop B will toggle thus at the point b in time line B goes high; it then goes low at point d and toggle

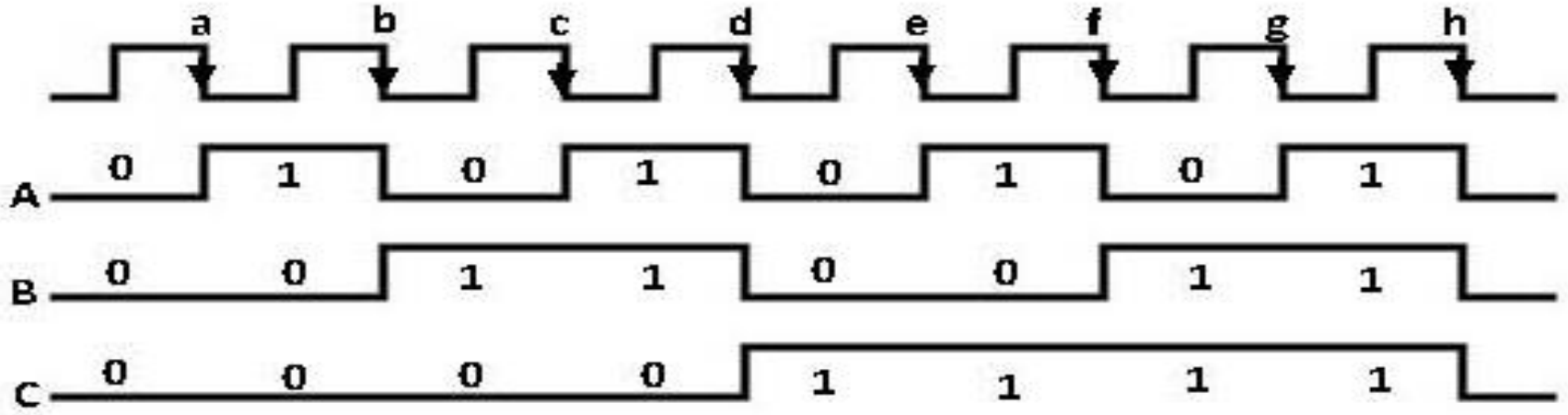
back high at again at point f. The wave form at the output of flip flop B is one fourth of the clock frequency.

Since B acts as the clock for C, each time the wave form at B goes low, flip flop C will toggle thus C goes high at point d and goes low at point h. The waveform at the output of flip flop C is one eighth of the clock frequency.

Truth table:

Count	C	B	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

Wave form:



❖ Ripple down counter:

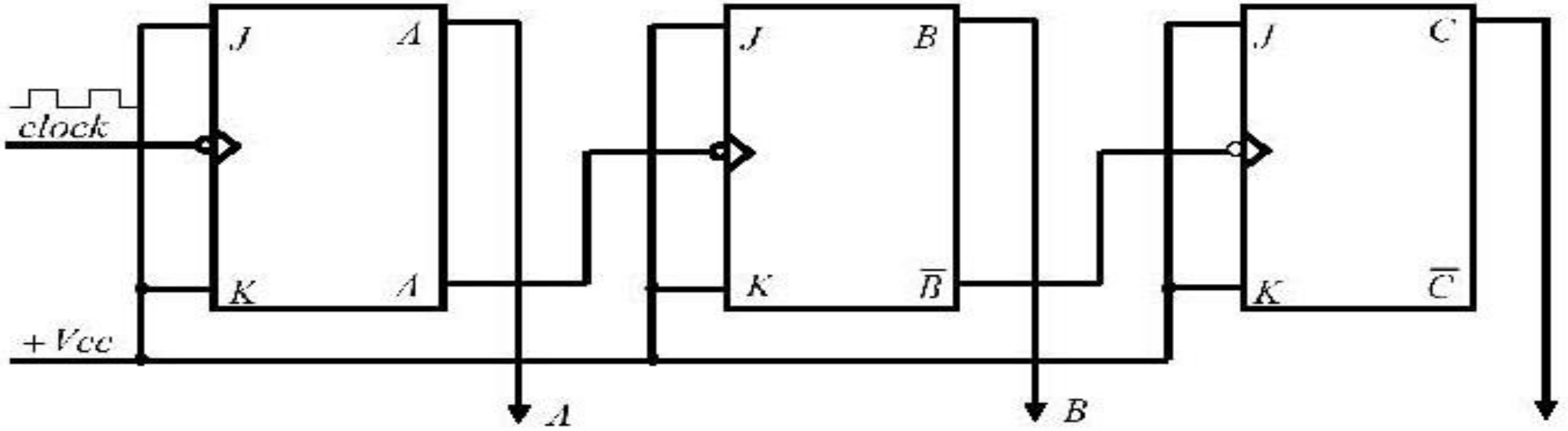


Fig: ripple down counter

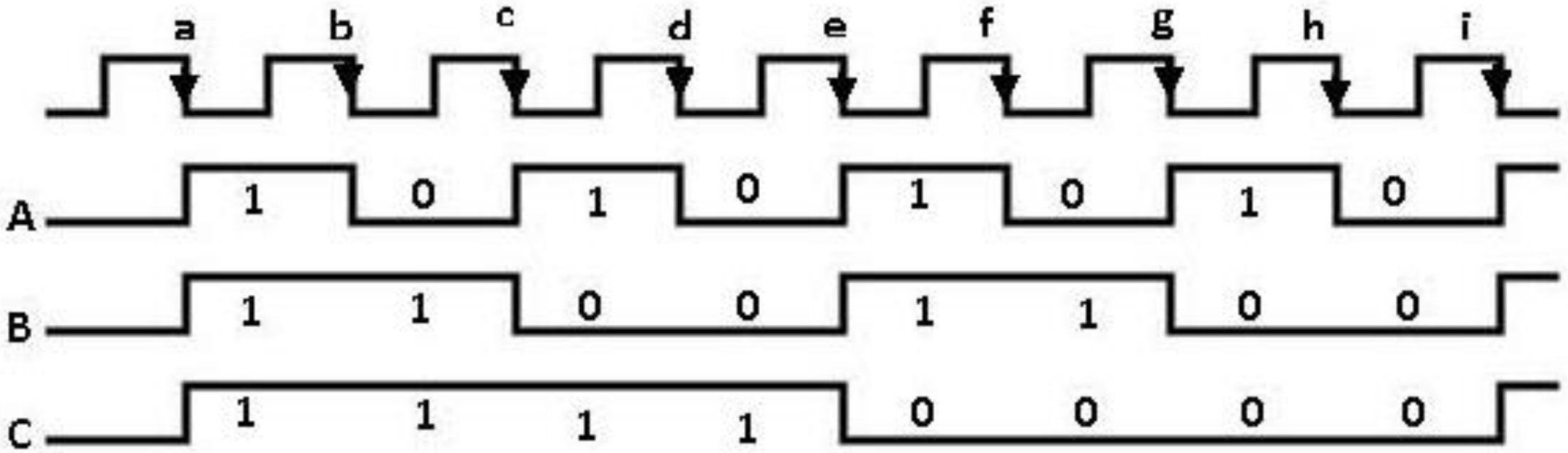
In the ripple down counter the system clock is used at the clock input of flip flop A but the complement i.e. $\overline{A}$ is used to drive flip flop B likewise $\overline{B}$ is used to drive flip flop C and so on.

Truth table:

Count	C	B	A
7	1	1	1
6	1	1	0
5	1	0	1

4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0

Wave form:



❖ Three bit binary up-down counter:

It is the combination of two counter i.e. up counter and down counter. Here if the count-down control line is low and count-up control line is high then the counter will have count-up wave form.

On the other hand, if the count-down is high and count-up is low, each flip flop will be triggered from the complement side of previous flip flop. The counter will then be in count down mode.

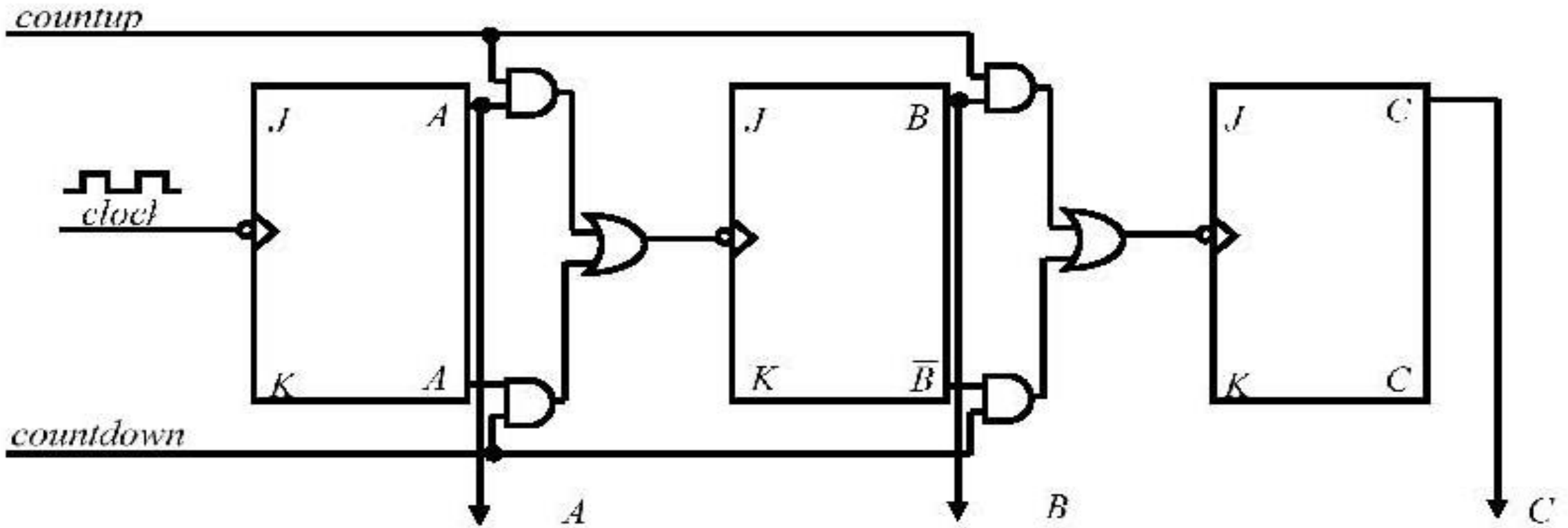


Fig: ripple up-down counter

Disadvantage of Ripple counter:

a) It has a limit to its highest operating frequency.
b) Each flip flop has a delay time, in a ripple counter these delay time are additive and the total "settling" time for a counter is approximately the delay time times the total number of flip flop.
c) There is a possibility of glitches i.e. an undesired positive or negative pulse appearing at the outside of logic gate with a ripple counter.

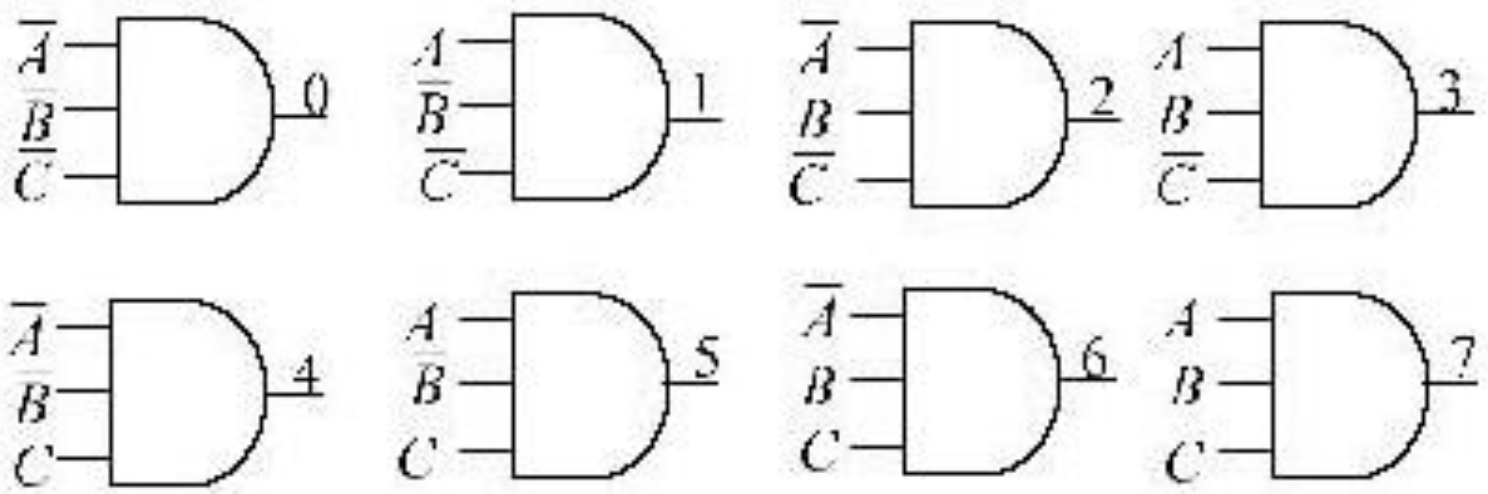
Q) Define ripple counter? Explain the operation of MOD-10 ripple counter with timing diagram. [BE IOE2069 Chaitra]

8.2 Decoding gates:

A decoding gates can be connected to the output of the counter in such a way that the output of the gate will be high or low only when the counter content are equal to a given state.

Suppose the decoding gates connected to the three bit ripple counter will decode 2 (CBA = 010). Thus the gate output will be high only when A=0, B=1 and C=0. The Boolean expression for this gate can be written 2=CBA. A comparison with the truth table for this counter will reveal that the condition CBA=010. Only true for the state 2.

Other seven state of the counter can be decoded in a similar manner. Suppose decode state 7, the truth table reveal that CBA=111 is a unique state. Similarly suppose decode state 5, the truth table reveal that CBA=101 is the unique state. All the decoding gates of three bit counter are below.



8.3 Synchronous counter:

In a synchronous counter pulse are applied to the input of all flip flop. The common pulse trigger all the flip flop simultaneously, rather than one at a time in a succession as in a ripple counter. The decision whether a flip flop is to be complemented or not is determined from the values of J and K inputs at the time of the pulse. If J = K = 0, the flip flop remains unchanged. If J = K = 1 the flip flop complements.

❖ Three bit synchronous up counter:

Up counter are those which counts upward or in the forward direction by one LSB every time it is clocked. Figure below figure is three bit synchronous up counter. Here clock pulse is passed to all of the flip flop. But the A output of the first stage is used to derive the J and K input of second stage. B output of second stage is used to derive the J and K of third stage.

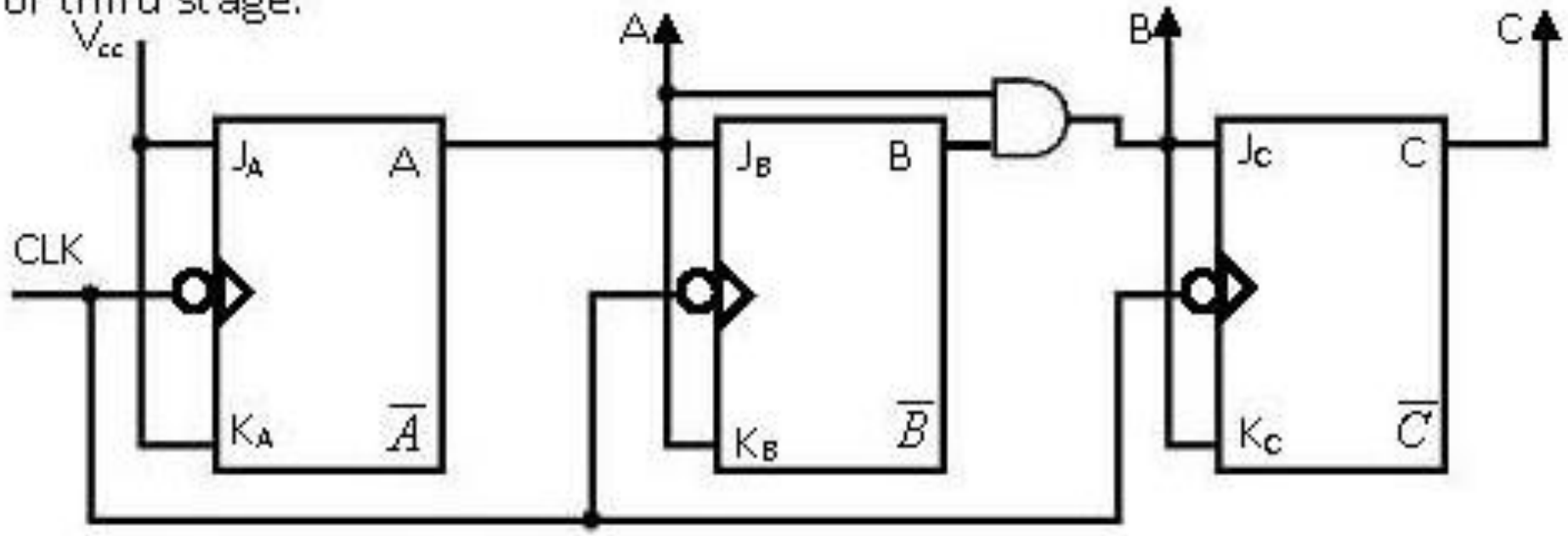
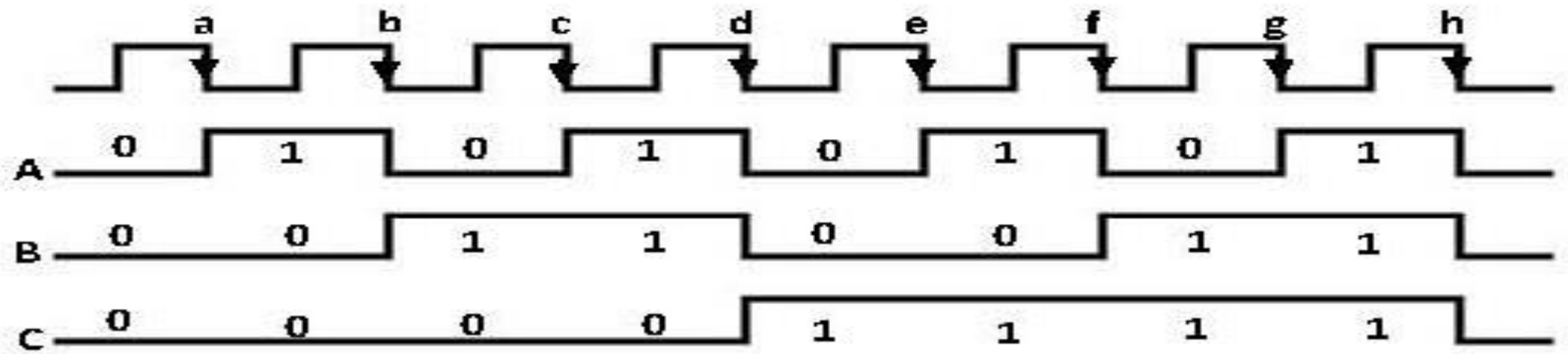


Fig: three bit synchronous up counter

Truth table:

Count	C	B	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

Wave form:



❖ Three bit synchronous down counter:

A down counter counts in a reverse direction or downward by the LSB every time it is clocked. Figure below is a three bit synchronous down counter. Here the clock pulse is passed to all of the flip flop but the complement $\overline{A}$ is used to derive the J and K input of second stage. $\overline{B}$ output of the second stage is used to derive the J and K of the third stage.

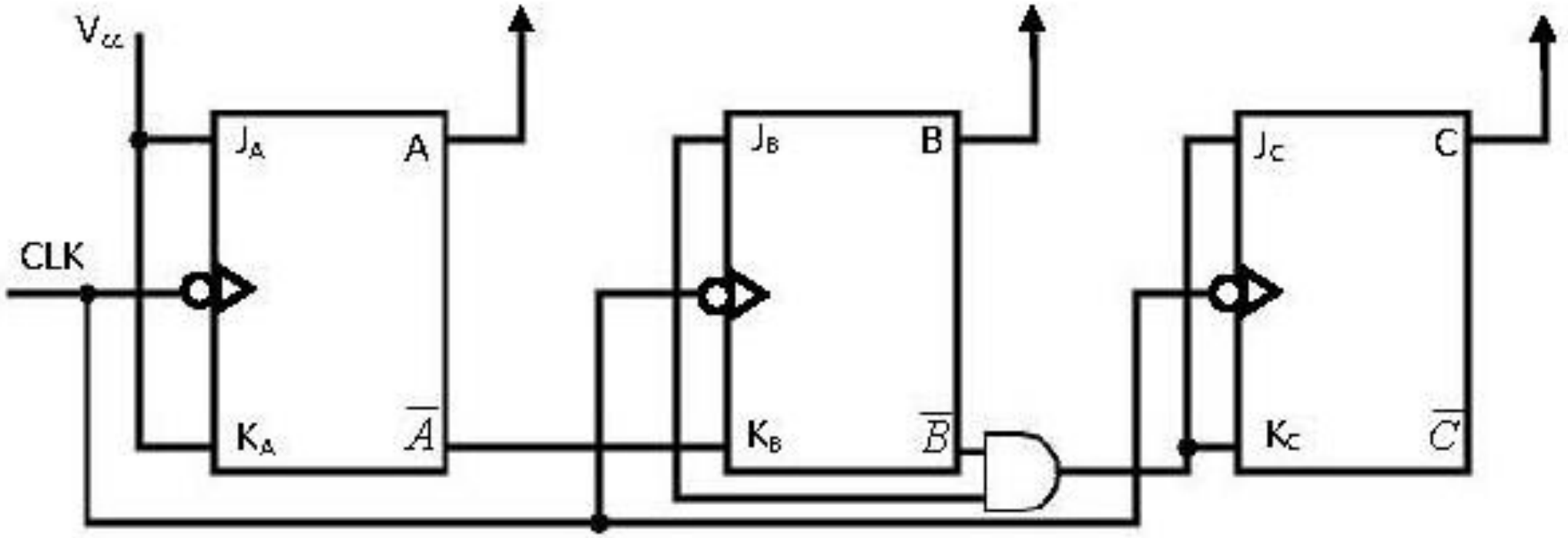
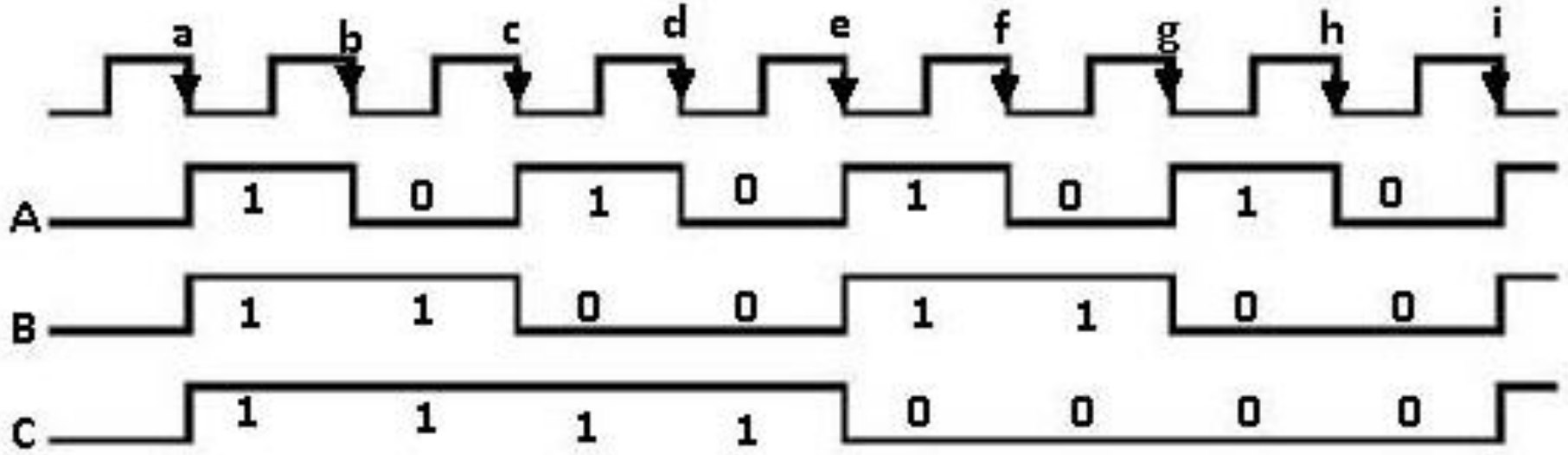


Fig: three bit synchronous down counter

Truth table:

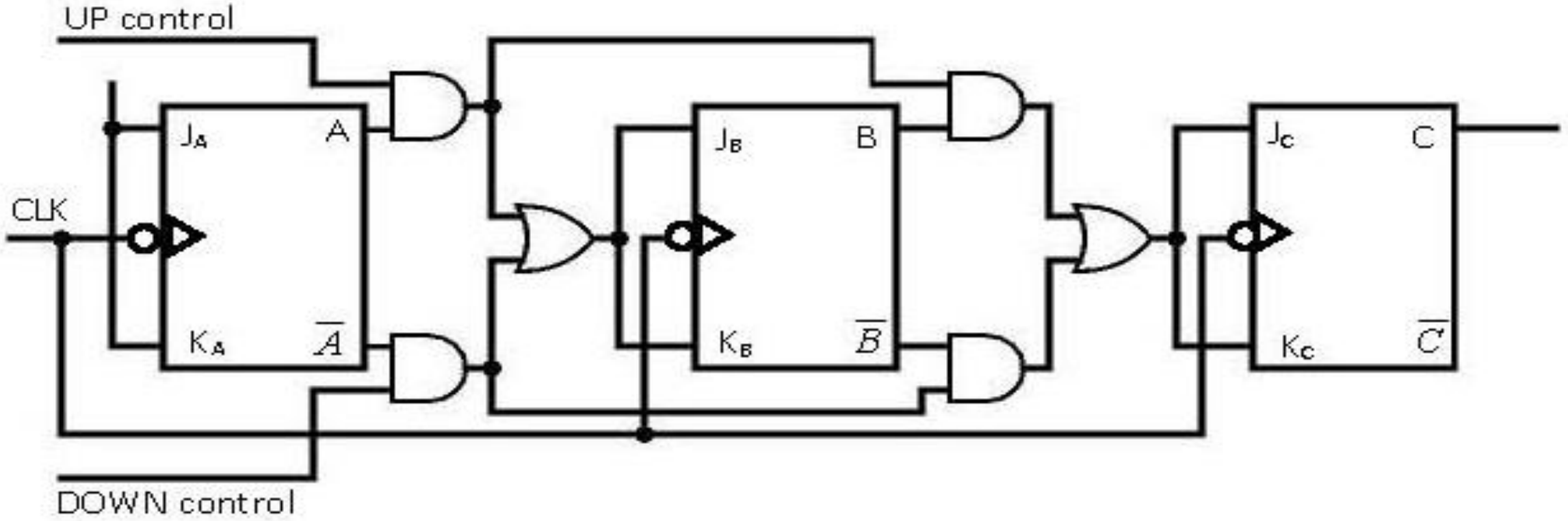
Count	C	B	A
7	1	1	1
6	1	1	0
5	1	0	1
4	1	0	0
3	0	1	1
2	0	1	0
1	0	0	1
0	0	0	0

Wave form:



- Three bit up - down counter:

  Counter are also available in integrated circuit form as up/down counter which can be made to operate as either up or down counter. The counter counts up when UP control is logic 1 and DOWN control is logic 0. Similarly the counter counts downward when UP control input is logic 0 and DOWN control input is logic 1.



Advantages of synchronous counter over a ripple counter:

a) It has no limit to its highest operating frequency. Faster in operation if high frequency is applied.
b) There is not possibility of glitches i.e. an undesired positive or negative pulse appearing at the outside of logic gate with a ripple counter.
c) Here settling time is equal to the delay time of a single flip flop.

8.4 Changing the counter modulus:

Modulus is defined as the number of states through which a counter can progress. The counters which progress 1 count at a time in a strict binary progression, and they all have a modulus given by 2^n, where n=number of flip-flops. Such counters are said to have a "naturalcount" of 2^n. For example,

→ mod2 counter consists of 1 flip-flop, and it counts two discrete states (0->1)

→ mod4 counter consists of 2 flip-flops, and it counts through 4 discrete states (00->01->10 ->11)

→ mod8 counter consists of 3 flip-flops, and it counts through 8 discrete states (000->001 ->010->011->100->101->110->111)

It is often desirable to construct counters having a modulus other than 2, 4, 8 and so on. Forexample, a counter having a modulus of 3 or 7 would be useful. A small modulus counter can always be constructed from a larger modulus counter by skipping states. Such counters are said to have a *modified-count*.
Firstly, it is necessary to determine the number of flip-flops required. The correct number of flip flops is determined choosing the lowest natural count that is greater than the desired modified count. For example, a mod-7 counter requires 3 flip-flops, since 8 is the lowest natural count greaterthan the desired modified count of 7.

8.5 Decade and BCD counter:

A **decading counter** is one that goes through 10 unique output combinations and then reset as the clock proceed further. Since it is a MOD-10 counter, it can be constructed with a minimum of four flip flop. A four flip flop counter would have a 16 states. By skipping any of the six states, by using some kind of feedback or some kind of additional logic, we can convert a normal four bit counter in to decade counter. A decade counter does not necessary count from 0000 to 1001. It could even count as 0000, 0001, 0010, 0101, 0110, 1001, 1010, 1100, 1101, 1111, 0000,..... . in this count sequence we have skipped 0011, 0100, 0111, 1000,1011 and 1110.

A **BCD counter** is a special case of decade counter in which the counter counts from 0000 to 1001 and then reset. Different counter state in this counter are binary equivalent of the decimal number 0 to 9.

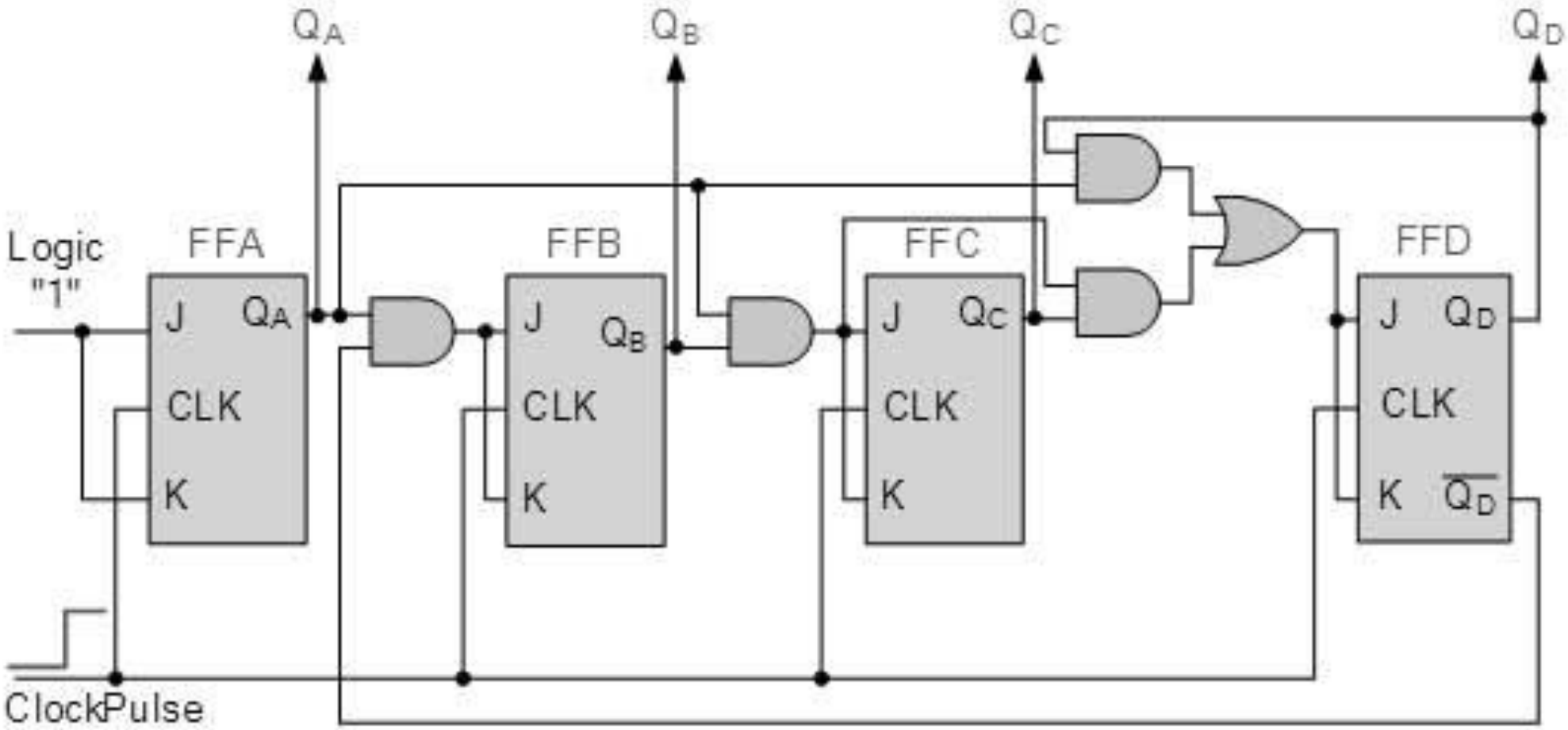


Fig: decade four bit synchronous counter

8.6 Presettable counter:

Presettable counter are those that can be preset to any starting count either asynchronous (independent of clock signal) or synchronous (with the active transition of the clock signal). The presettable operation is achieved with the help of PRESET and CLEAR input available in the flip flop. The presettable operation is known as the 'preloading' or simply 'loading' operation. known. The diagram below shows a 3-bit asynchronously presettable synchronous up counter.

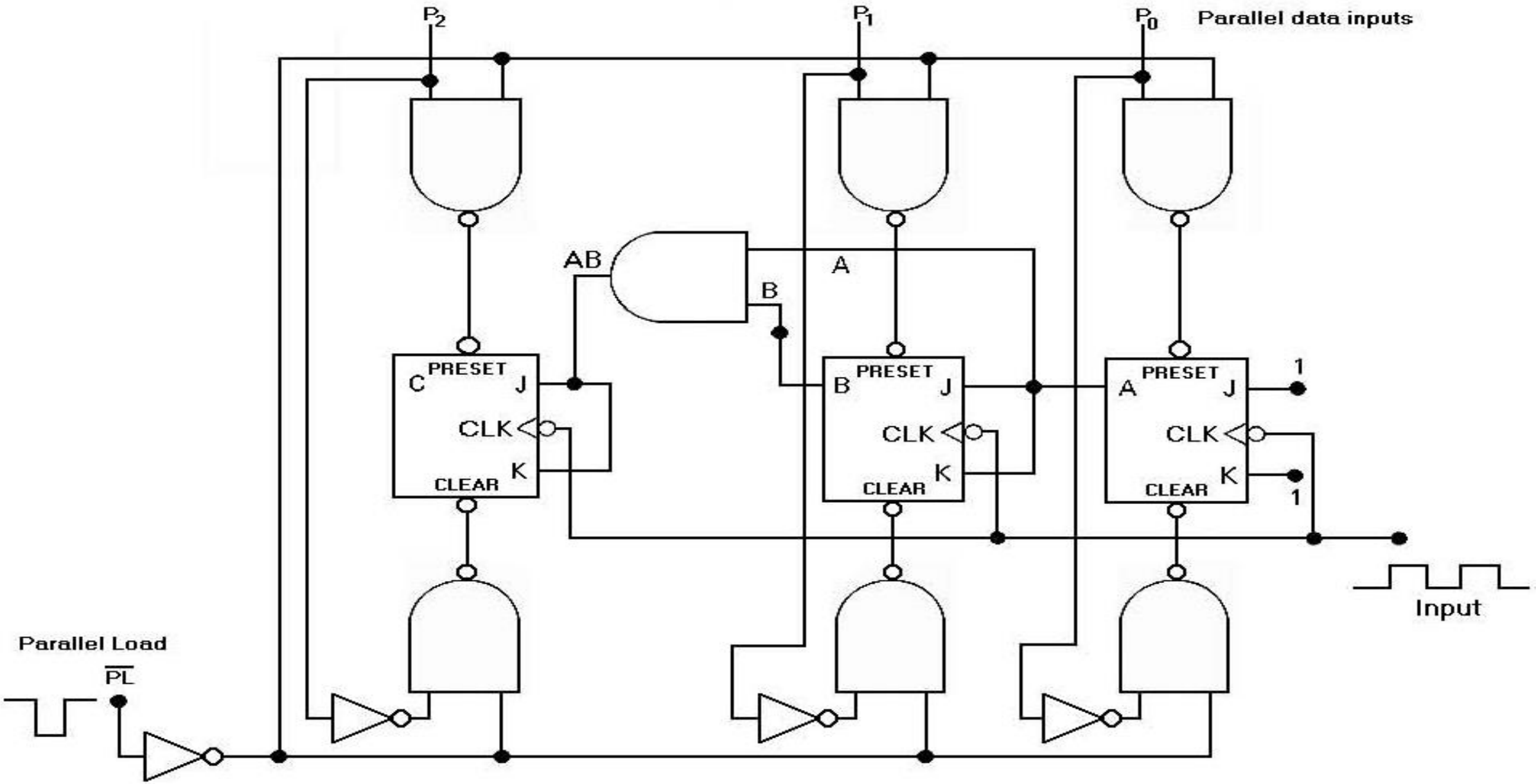


Fig: 3-bit Synchronous Binary Presettable Counter

In the diagram above, the J, K and CLK inputs are wired the same way as a synchronous up counter. The asynchronous PRESET and CLEAR inputs are used to perform the asynchronous presetting. The counter is loaded by applying the desired binary number to the inputs P_2, P_1 and P_0 and a LOW pulse is applied to the PARALLEL LOAD input, not(PL). This will asynchronously transfer P_2, P_1 and P_0 into the flip-flops. This transfer occurs independently of the J, K, and CLK inputs. As long as not(PL) remains in the LOW state, the CLK input has no effect on the flip-flop. After not(PL) returns to high, the counter resumes counting, starting from the number that was loaded into the counter.

For the example above, say that $P_2 = 1$, $P_1 = 0$, and $P_0 = 1$. When not(PL) is high, these inputs have no effect. The counter will perform normal count-up operations if there are clock pulses. Now let's say that not(PL) goes low at $Q_2 = 0$, $Q_1 = 1$ and $Q_0 = 0$. This will produce LOW states at the CLEAR input of Q_1, and the PRESET inputs of Q_2 and Q_0. This will make the counter go to state 101 regardless of what is occuring at the CLK input. The counter will remain at state 101 until not(PL) goes back to HIGH. The counter will then continue counting from 101.

8.7 Counter design as a synthesis problem:

MOD – 5 counter:

Suppose we are going to design a MOD – 5 counter. While designing counter we can use either JK - flip flop or T - flip flop. It is easier to design a counter using T – flip flop then JK flip flop.

To design a modulo – 5 counter firstly we have to draw a state transition diagram. It is shown below in figure 1. We need four flip flop for this and we use T-flip flop named A, B, C as a memory element of design.

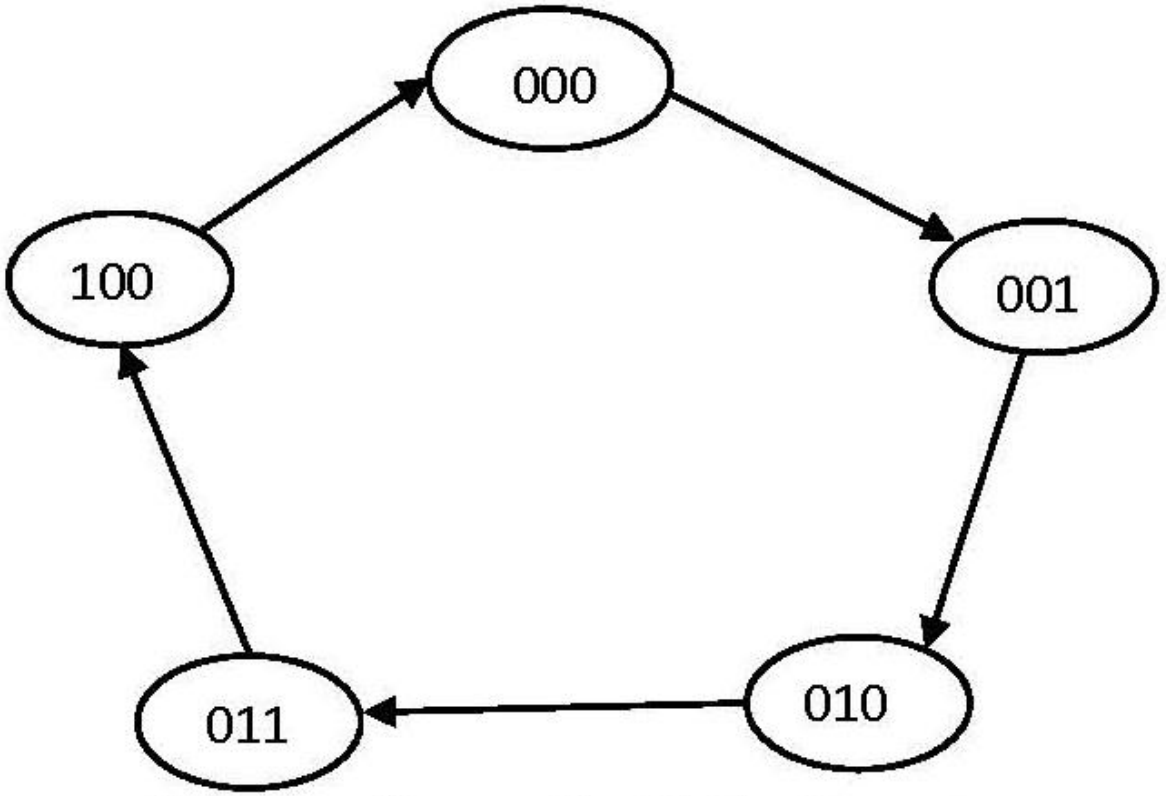


Fig: state diagram for M0D – 5 counter

With three flip flop 8 different state are possible but in our design 101, 110, 111 are not used in a counting sequence.

Excitation table for JK flip flop:

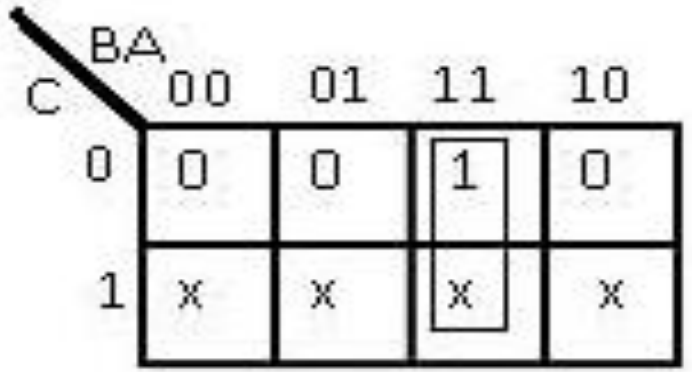
Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

State table:

Present state			Next state								
C	B	A	C	B	A	J_C	K_C	J_B	K_B	J_A	K_A
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	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	0	0	0	x	1	0	x	0	x

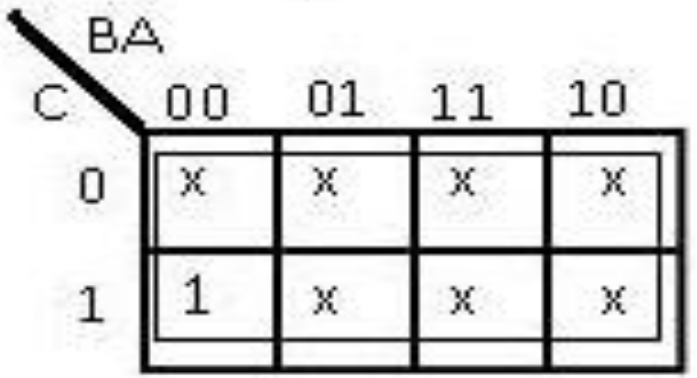
In k maps:

For J_c



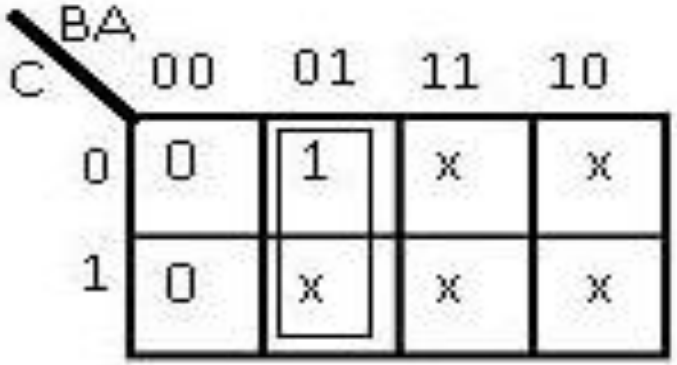
$J_c=BA$

For K_c



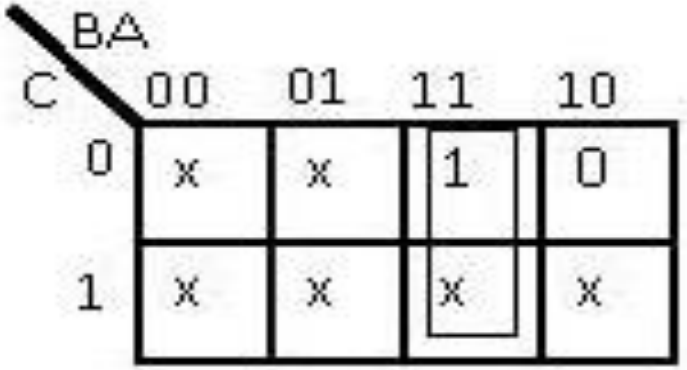
$K_c=1$

For J_B



$J_B=B'A$

For K_B



$K_B=BA$

For J_A

C \ BA	00	01	11	10
0	1	x	x	1
1	0	x	x	x

$J_A = C'$

For K_A

C \ BA	00	01	11	10
0	x	1	1	x
1	x	x	x	x

$K_A = 1$

Design:

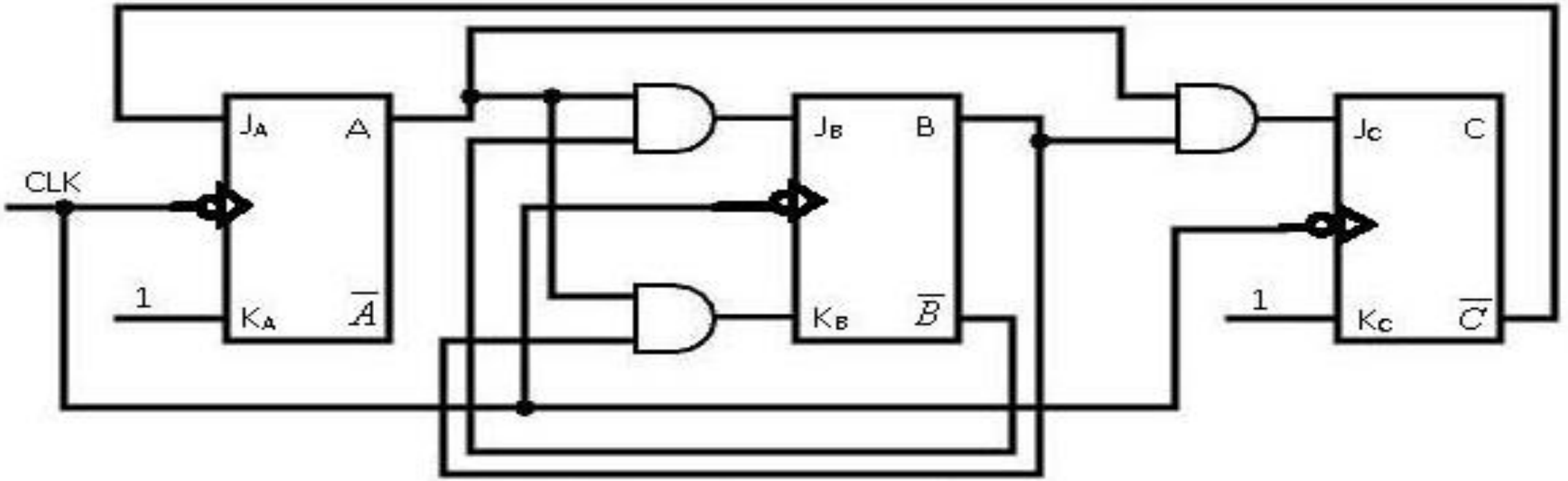
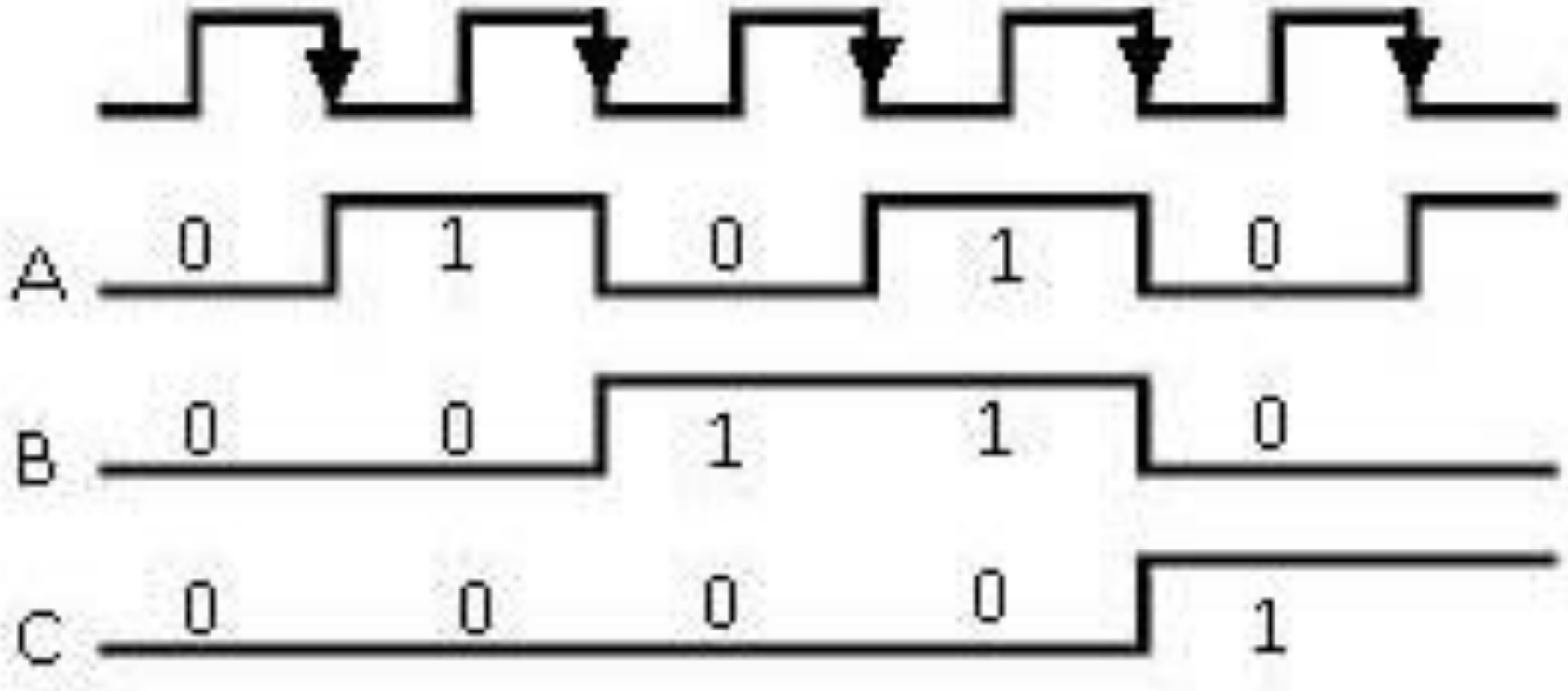


Fig: MOD-5 synchronous counter

Wave form:

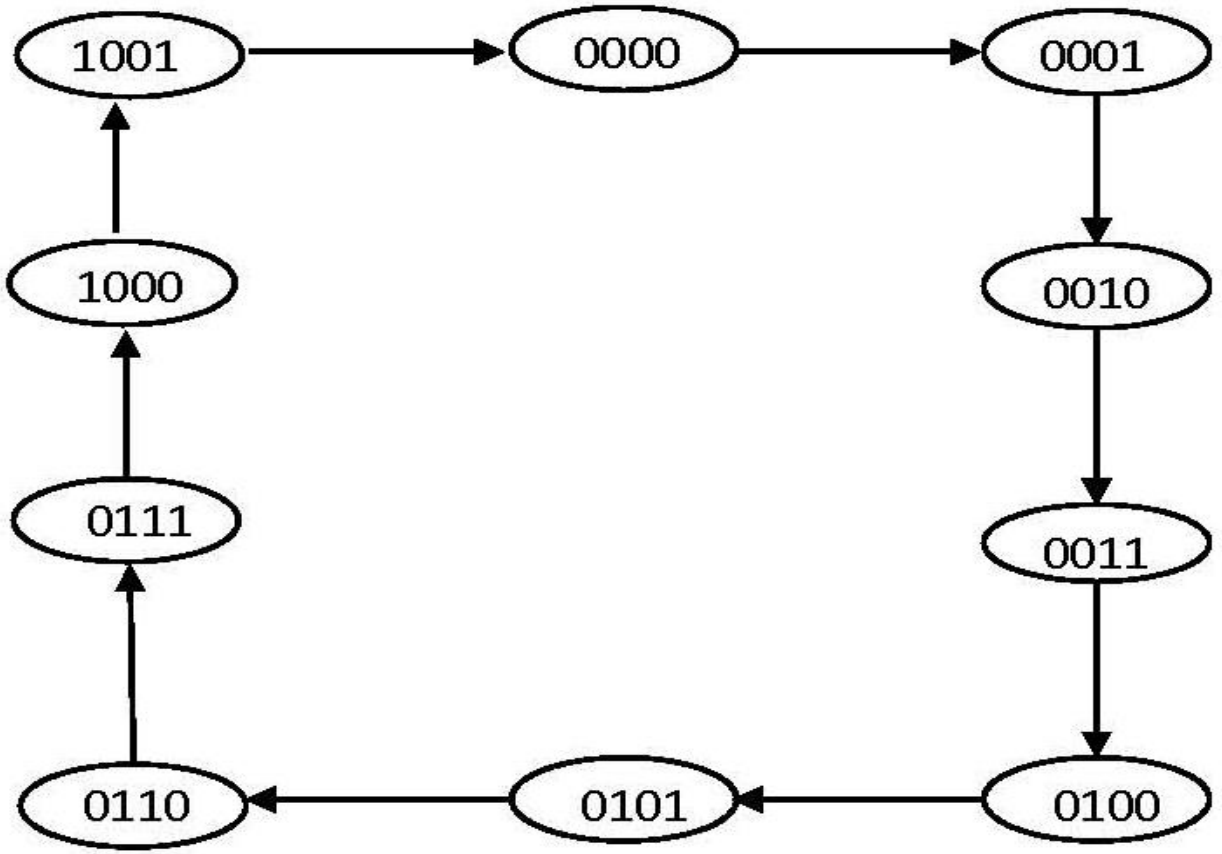


Q) What is counter? Design a MOD – 6 synchronous counter. Draw its timing diagram. [BE IOE 2068 baisakh]

MOD-10 counter:

We design MOD-10 counter using T flip flop. We need four T flip flop named A, B, C, and D

State transition diagram:



The excitation table for T flip flop:

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

State table:

Present state				Next state							
D	C	B	A	D	C	B	A	T_D	T_C	T_B	T_A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1

For T_D

AB \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	x	x	x	x
10	0	1	x	x

$T_D = AD + B\overline{C}D$

For T_C

AB \ CD	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	x	x	x	x
10	0	1	x	x

$T_C = CD$

For T_B

AB \ CD	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	x	x	x	x
10	0	0	x	x

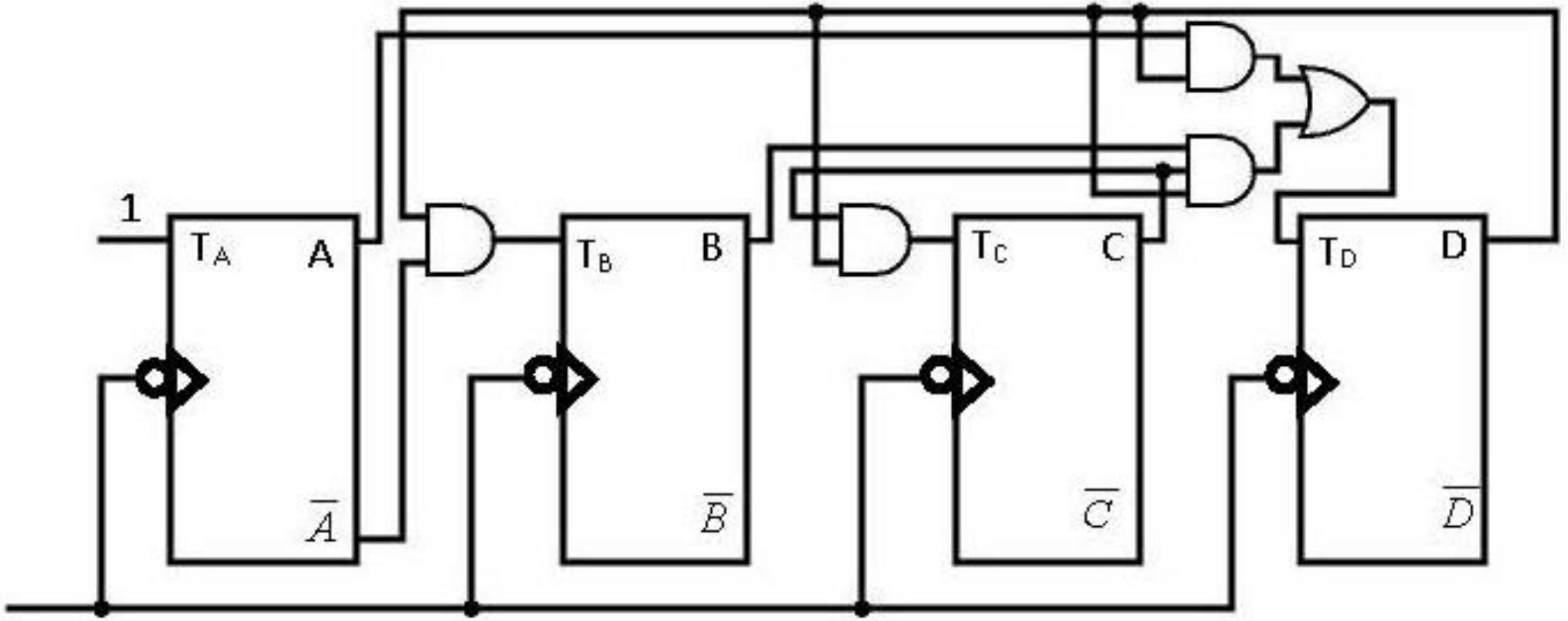
$T_B = A'D$

For T_A

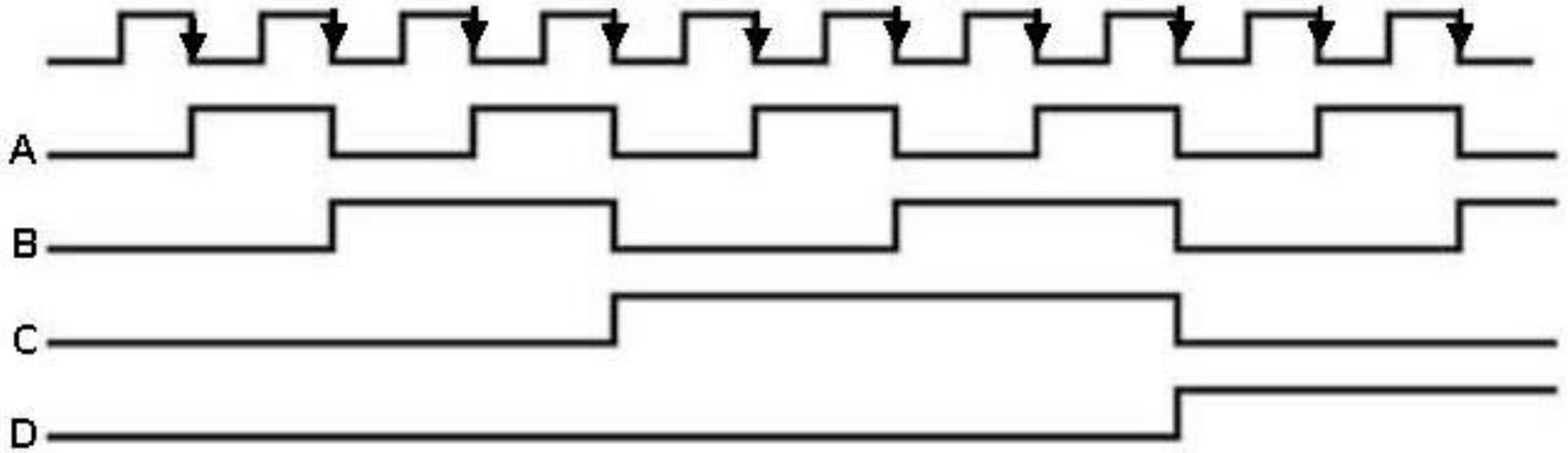
AB \ CD	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	x	x	x	x
10	1	1	x	x

$T_C = 1$

Design:



Wave form:



DIGITAL CLOCK

A digital clock, which displays the time of day in hours, minutes, and seconds, is one of the most common applications of counters. To construct an accurate digital clock, a very highly controlled basic clock frequency is required. For battery-operated digital clocks (or watches) the basic frequency can be obtained from a quartz-crystal oscillator. Digital clocks operated from the AC power line can use the 50 Hz power frequency as the basic clock frequency. In either case, the basic frequency has to be divided down to a frequency of 1 Hz or pulse of 1 second (pps). The basic block diagram for a digital clock operating from 50 Hz is shown in figure below.

The 50 Hz signal is sent through a Schmitt trigger circuit to produce square pulses at the rate of 50 pps. The 50 pps waveform is fed into a MOD-50 counter, which is used to divide the 50 pps down to 1 pps. The 1-pps signal is then fed into the SECONDS section. This section is used to count and display seconds from 0 through 59. The BCD counter advances one count per second. After 9 seconds the BCD counter recycles to 0. This triggers the MOD-6 counter and causes it to advance one count. This continues for 59 seconds. At this point, the BCD counter is at 1001 (9) count and the MOD-6 counter is at 101 (5). Hence, the display reads 59 seconds. The next pulse recycles the BCD counter to 0. This, in turn, recycles the MOD-6 counter to 0.

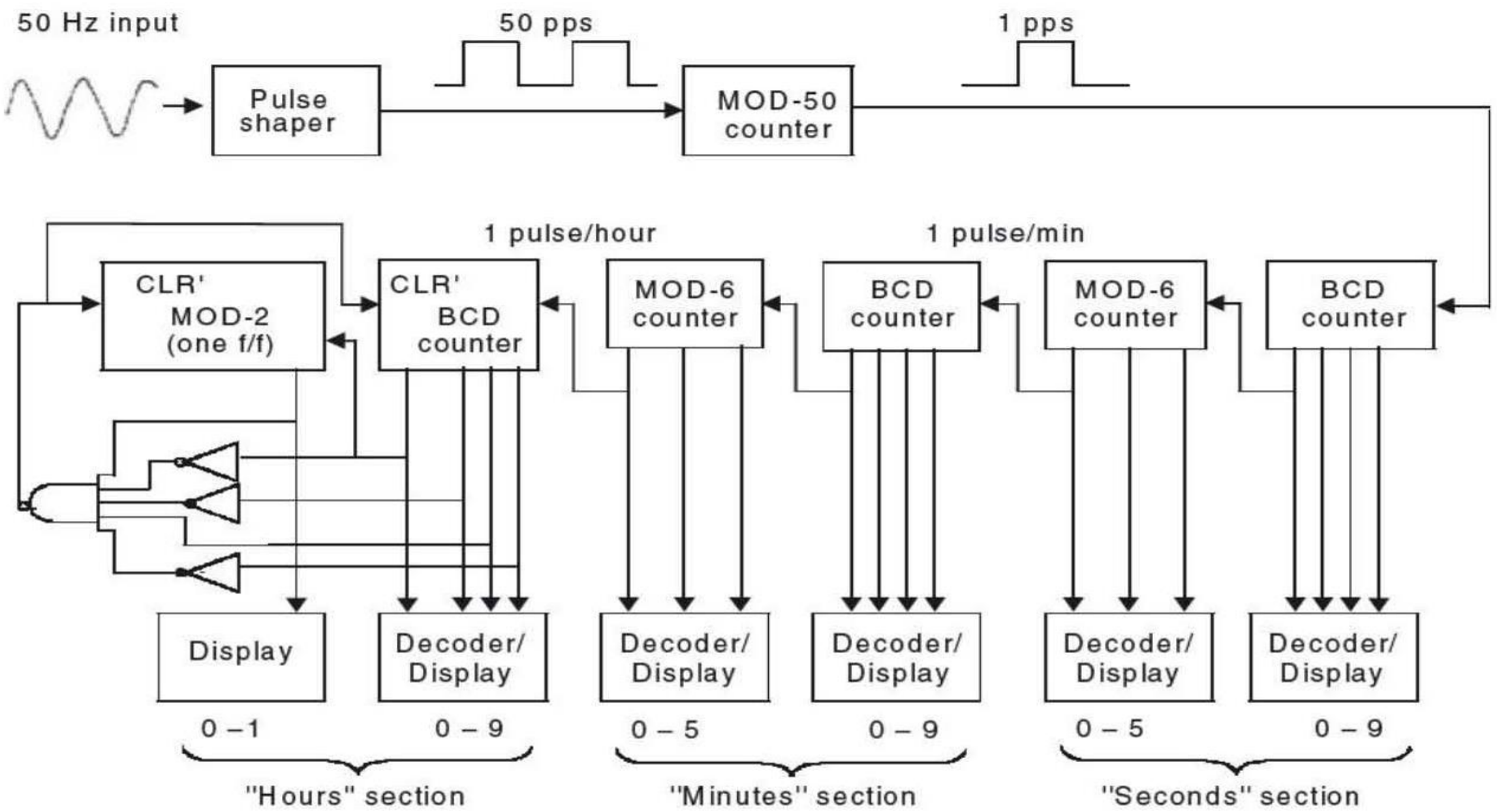


Fig: Block diagram for a digital clock

The output of the MOD-6 counter in the SECONDS section has a frequency of 1 pulse per minute. This signal is fed to the MINUTES section, which counts and displays minutes from 0 through 59. The MINUTES section is identical to the SECONDS section and operates in exactly the same manner. The output of the MOD-6 counter in the MINUTES section has a frequency of 1 pulse per hour. This signal is fed to the HOURS section, which counts and displays hours from 1 through 12. The HOURS section is different from the MINUTES and SECONDS section in that it never goes to the zero state. The circuitry in this section is different. When the hours counter reaches 12, it will be reset to zero by the NAND gate.

1) **Differentiate between synchronous and asynchronous counter.**

Asynchronous counter	Synchronous counter
- In this type of counter flip flop are connected in such a way that output of first flip flop derive the clock for next flip flop - All the flip flop are not clocked simultaneously. - Logic circuit are simpler even for more number of states. - Main drawback of this counter is their low speed as the clock is propagated through number of flip flop before it reaches last state	- In this type there is no connection between output of first flip flop and clock input of next flip flop. - All the flip flop are clocked simultaneously Design involves complex logic circuit as number of state increases - As the clock is simultaneously given to all flip flop there is no problem of propagation delay. Hence they are high speed counter and are preferred when number of flip flop increases in given design.

Some important questions:

1) Deign a MOD – 10 synchronous counter showing its state circuit diagram and output wave form. [BE IOE 2067 ashad]
2) Design a three bit down counter and show the timing diagram for 3 clock cycle, assuming that the initial reading of the counter is 0. [BE IOE 2067 magh]
3) Design a MOD – 5 binary ripple up counter. What are the advantage of synchronous counter over a ripple counter? [BE IOE 2066 bhadra]
4) Design a three bit binary synchronous up counter using the excitation table and dray its timing diagram. [BE IOE 2066 bhadra]
5) Design a synchronous decade counter and also show its timing diagram. [BE IOE 2069 chaitra]
6) Define the ripple counter. Explain the operation of MOD – 10 ripple counter with timing diagram. [BE IOE 2068 chaitra]
7) Differentiate between the synchronous an asynchronous counter. Draw and explain the operation of asynchronous decade counter with clear timing diagram.

THANK YOU