COUNTERS

A counter driven by clock can be used to count the number of pulses. Since clock pulses occur at known intervals, the counter can be used as an instrument for measuring time and therefore period or frequency.

There are basically two different types of counters

1. Asynchronous counters
2. Synchronous counters

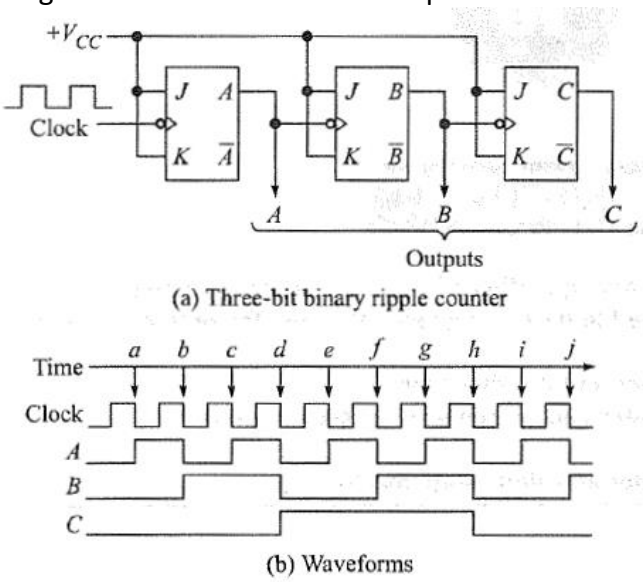
The ripple counter is simple and straightforward in operation and construction usually requires a minimum hardware. It does, however it has speed limitations. Each flip is triggered by the previous flip flop, and thus the counter has a cumulative settling time. Counters such as these are called serial, or asynchronous.

An increase in speed of operation can be achieved by use of a parallel or synchronous counter. Here, every flip-flop is triggered by the clock (in synchronism), and thus settling time is simply equal to the delay time of a single flip-flop. The increase in speed is usually obtained at the price of increased hardware.

Serial and parallel counters are used in combination to compromise between speed of operation and hardware count. Serial, parallel, or combination counters can be designed such that each clock transition advances the contents of the counter by one; it is then operating in a count-up mode. The opposite is also possible; the counter then operates in the count-down mode.

ASYNCHRONOUS COUNTERS (Ripple Counters)

A binary ripple counter can be constructed using clocked *JK* flip-flops. Figure 1show three negative edge-triggered, *JK* flip-flops connected in cascade. The system clock, a square wave, drives flip-flop *A.* The output of *A* drives *B,* and the output of *B* drives flip-flop C. All the] andK inputs are tied to $+V_{CC}$. This means that each flip-flop will change state (toggle) with a negative transition at its clock input.



(a) Three-bit binary ripple counter

(b) Waveforms

Negative clock transitions	C	B	A	State or count
---	0	0	0	0
a	0	0	1	1
b	0	1	0	2
c	0	1	1	3
d	1	0	0	4
e	1	0	1	5
f	1	1	0	6
g	1	1	1	7
h	0	0	0	0

(c) Truth table

Fig.1: Asynchronous Counter [Up mode]

When the output of a flip-flop is used as the clock input for the next flip-flop, we call the counter a *ripple counter,* or *asynchronous counter.* The *A* flip-flop must change state before it can trigger the *B* flip-flop, and the *B* flip-flop has to change state before it can

trigger the C flip-flop. The triggers move through the flip-flops like a ripple in water. Because of this, the overall propagation delay time is the sum of the individual delays. For instance, if each flip-flop in this three-flip-flop counter has a propagation delay time of 10 ns, the overall propagation delay time for the counter is 30 ns.

The waveforms given in Fig. 1(b) show the action of the counter as the clock runs. Let's assume that the flip-flops are all initially reset to produce O outputs. If we consider *A* to be the least-significant bit(LSB) and C the most-significant bit (MSB), we can say the contents of the counter is *CEA* = 000.Every time there is a clock NT, flip-flop *A* will change state. This is indicated by the small arrows ( ) on the time line. Thus at point *a* on the time line, *A* goes high, at point *b* it goes back low, at *c* it goes back high, and so on. The waveform at the output of flip-flop *A* is one-half the clock frequency.

Since *A* acts as the clock for *B,* each time the waveform at *A* goes low, flip-flop *B* will toggle. Thus at point *b* on the time line, *B* goes high; it then goes low at point *d* and toggles back high again at point *f* . The waveform at the output of flip-flop *B* is one-half the frequency of *A* and one-fourth the clock frequency.

Since *B* acts as the clock for C, each time the waveform at *B* goes low, flip-flop *C* will toggle. Thus *C* goes high at point don the time line and goes back low again at point *h.* The frequency of the waveform at *C* is one-half that at *B,* but it is only one-eighth the clock frequency.

The output condition of the flip-flops is a binary number equivalent to thenumber of clock NTs that have occurred. The counter content advances one count with each clockNT in a "straight binary progression" that is summarized in the truth table in Fig. 1(c).Because each output condition shown in the truth table is the binary equivalent of the number of clock NTs, the three cascaded flip-flops in Fig. 1 comprises a 3-bit binary ripple counter. This counter cans be used to count the number of clock transitions up to a maximum of seven. The counter begins at count 000 and advances one count for each clock transition until it reaches count 111. At this point it resets back to 000 and begins the count cycle all over again. Thus this ripple counter is operating in a count-up mode.

Since a binary ripple counter counts in a straight binary sequence, it is easy to see that a counter having *n* flip-flops will have 2^noutput conditions. For instance, the three-flip-flop counter just discussed has $2^3 = 8$output conditions (000 through 111). Five flip-flops would have $2^5 = 32$ output conditions (00000 through11111), and so on.

The largest binary number that can be represented by *n* cascaded flip-flops has a decimal equivalent of $2^n - 1$. For example, the three-flip-flop counter reaches a maximum decimal number of $2^3 - 1$.The maximum decimal number for five flip-flops is $2^5 - 1 = 31$, while six flip-flops have a maximum count of 63.

A three-flip-flop counter is often referred to as a modulus-8 (or mod-8) counter since it has eight states. Similarly, a four-flip-flop counter is a mod-16 counter, and a six-flip-flop counter is a mod-64 counter. The *modulus* of a counter is the total number of states through which the counter can progress.

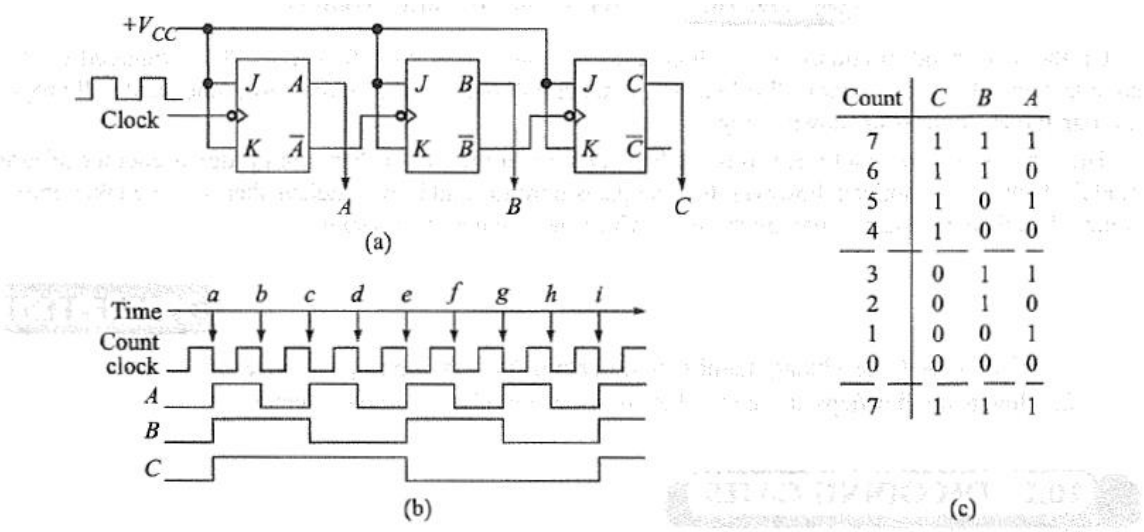
Example: What is the clock frequency in Fig. 1 if the period of the waveform at C is 24 S?
Solution Since there are eight clock cycles in one cycle of C, the period of the clock must be 24/8 = 3 µs. The clock frequency must then be $1/(3 \times 10^6) = 333$ kHz.

Example: How many flip-flops are required to construct a mod-128 counter? A mod-32? What is the largest decimal number that can be stored in a mod-64 counter?

Solution: A mod-128 counter must have seven flip-flops, since $2^7 = 128$, Five flip-flops are needed to construct amod32 counter. The largest decimal number that can be stored in a six-flip-flop counter {mod-64) is 111111 = 63.

A down counter

In down counter[fig.2], the system clock is used at the clock input to flip-flop *A,* but the complement of *A,* A, is used to drive flip flop *B,* likewise; *B* is used to drive flip-flop C.



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
7	1	1	1

Fig.2: Asynchronous Counter [Down mode]

Flip-flop *A* simply toggles with each negative clock transition as before. But flip-flop *B* will toggle each time *A* goes high! Notice that each time *A* goes high, *A* goes low, and it is this negative transition on *A* that triggers *B.* On the time line, *B* toggles at points *a, c, e, g* and i. Similarly, flip-flop *C* is triggered by *B* and so *C* will toggle each time *B* goes high. Thus *C* toggles high at point *a* on the time line, toggles back low at point *e* and goes back high again at point i.

The counter contents become *ABC*= 111 at point *a* on the time line, change to 110 at point *b,* and change to101 at point *c.* Notice that the counter contents are reduced by one count with each clock transition! In other words, the counter is operating in a count-down mode. The results are summarized in the truth table in Fig.2[c]. This is still a mod-8 counter, since it has eight discrete states, but it is connected as a *down counter.*

3-bit asynchronous up-down counter

A 3-bit asynchronous *up-down counter* that counts in a straight binary sequence is shown in Fig.3. It is simply a combination of the two counters up and down counter. For this counter to progress through a count-up sequence, it is necessary to trigger each flip-flop with the true side of the previous flip-flop (as opposed to the complement side.).If the count-down

control line is low and the count-up control line high, this will be the case, and the counter will have count-up waveforms such as those shown in Fig.3.

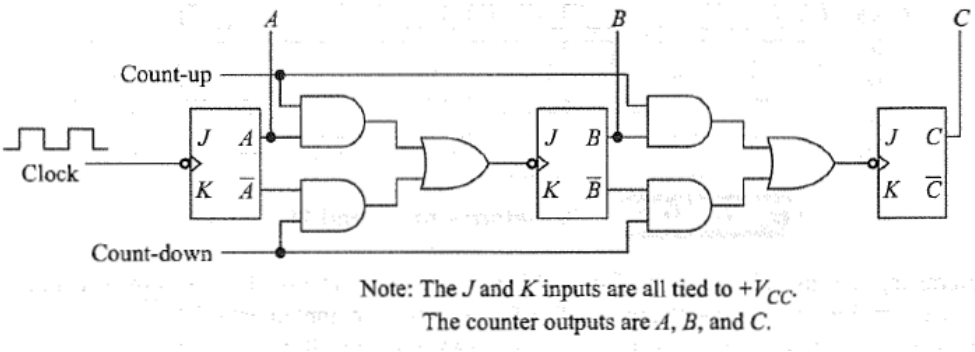
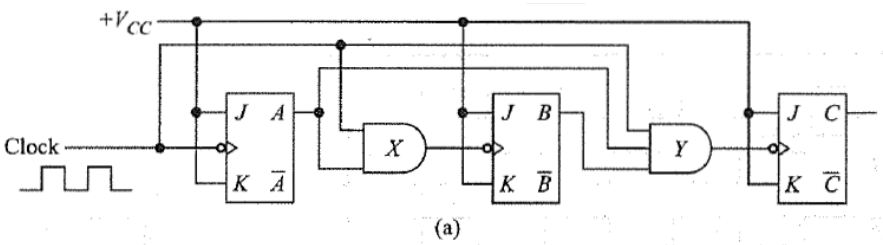


Fig.3: A 3-bit asynchronous up-down counter

On the other hand, if count-down is high and count-up is low, each flip-flop will be triggered from the complement side of the previous flip-flop. The counter will then be in a count-down mode and will progress through the waveforms as shown in Fig. 3.

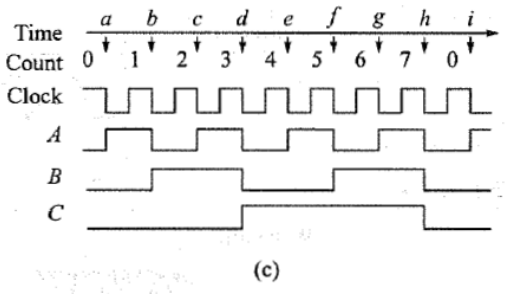
SYNCHRONOUS [PARALLEL] COUNTERS

The ripple counter is the simplest to build, but there is a limit to its highest operating frequency. Each flip-flop has a delay time. In a ripple counter these delay times are additive, and the total "settling" time for the counter is approximately the delay time times the total number of flip-flops. Furthermore, there is the possibility of glitches occurring at the output of decoding gates used with a ripple counter. The first problem fully and the second problem, to some extent can be overcome by the use of a synchronous parallel counter. The main difference here is that every flip-flop is triggered in synchronism with the clock.



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

(b)



(c)

Fig.4: A 3-bit synchronous counter

The construction of one type of parallel binary counter is shown in Fig.4, along with the truth table and the waveforms for the natural count sequence. Since each state corresponds to an equivalent binary number (or count). The *J* and *K* inputs of each flip-flop are high; therefore each flip-flop will toggle with any clock NT at its clock input. AND gates are used to gate every second clock to flip-flop *B,* every fourth clock to flip-flop C, and so on. This logic configuration is often referred to as "steering logic" since the clock pulses are gated or steered to each individual flip-flop.

The clock is applied directly to flip-flop *A.* Since the *JK* flip-flop used responds to a negative transition at the clock input and toggles when both the *J* and *K* inputs are high, flip-flop *A* will change state with each clock NT. Whenever *A* is high, AND gate *X* is enabled and a clock pulse is passed through the gate to the clock input of flip-flop *B.* Thus *B* changes state with every other clock NT at point's *b, d, f* and *h* on the time line.

Since, there is an additional AND gate delay for the clock at *B* flip-flop in comparison to *A* flip-flop, it is not a parallel counter in a strict sense of the term. Since AND gate *Y* is enabled and will transmit the clock to flip-flop C only when both *A* and *B* are high, flip-flop *C* changes state with every fourth clock NT at points *d* and *h* on the time line.

Examination of the waveforms and the truth table shows that this counter progresses upward in a natural binary sequence from count 000 up to count 111, advancing one count with each clock NT; This is a mod-8parallel or synchronous binary counter operating in the count-up mode.

A parallel up-down counter

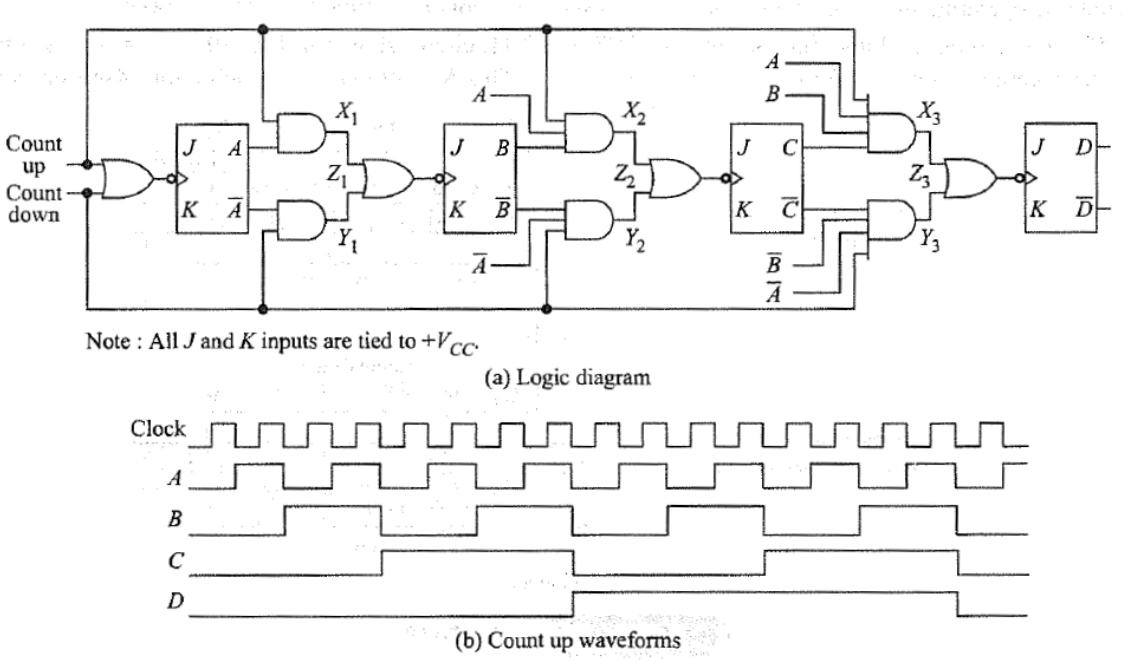


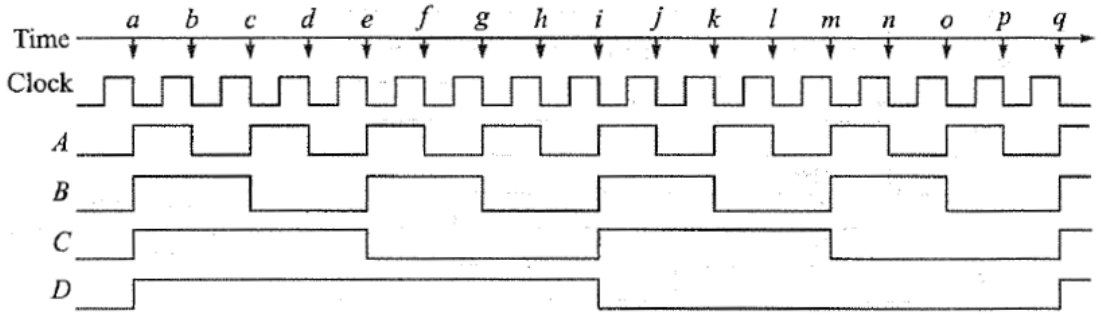
Fig.5: A 4-bit synchronous up-down counter

Figure 5 shows a 4 bit parallel up-down counter. In any parallel counter, the time at which any flip-flop changes state is determined by the states of all previous flip flops in the

counter. In the count-up mode, a flip-flop must toggle every time all previous flip-flops are in a 1state, and the clock makes a transition.

In the count-down mode, flip-flop toggles must occur when all prior flip-flops are in a O state. The counter in Fig.5 is a synchronous 4-bit up-down counter. To operate in the count-up mode, the system clock is applied at the count-up input, while the count-down input is held low. To operate in the countdown mode, the system clock is applied at the count-down input while holding the count-up input low.

Holding the count-down input low (at ground) will disable AND gates Y1, *Y2,* and *Y3*.The clock applied at count-up will then go directly into flip-flop *A* and will be steered into the other flip-flops by AND gates *X1,X2,* and *X3.* This counter will then function exactly as parallel counter. The only difference here is that this is a mod-16counter that advances one count with each clock NT, beginning with 0000 and ending with 1111. The correct waveforms are shown in Fig. 5[b].



(c) Count down waveforms

If the count-up line is held low, the upper AND gates X1, *X2,* and X3 are disabled. The clock applied at input count-down will go directly into flip-flop *A* and be steered into the following flip-flops by AND gates*Y1, Y2,* and *Y3.*Flip-flop *A* will toggle each time there is a clock NT as shown in Fig. 5[c]. Each time *A* is high, AND gate Y1 will be enabled and the clock NT will toggle flip-flop Bat points *a, c, e, g,* and so on. Whenever both A and *B* are high, AND gate *Y2* is enabled, and thus a clock will be steered into flip-flop *Cat* points *a, e, i, m,* and *q.* Similarly, AND gate *Y3* will steer a clock into flip-flop *D* only when *A, B,* and *C* are all high. Thus flip-flop *D* will toggle at points *a* and *i* on the time line. The waveforms in Fig. 5[c] clearly show that the counter is operating in a count-down mode, progressing one count at a time from 1111 to 0000.

Count up

Count down

D	*C*	*B*	*A*	Count
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
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
0	0	0	0	0

Table: Truth table of up-down counter

A Mod-5 Counter

Mod-5 Counter

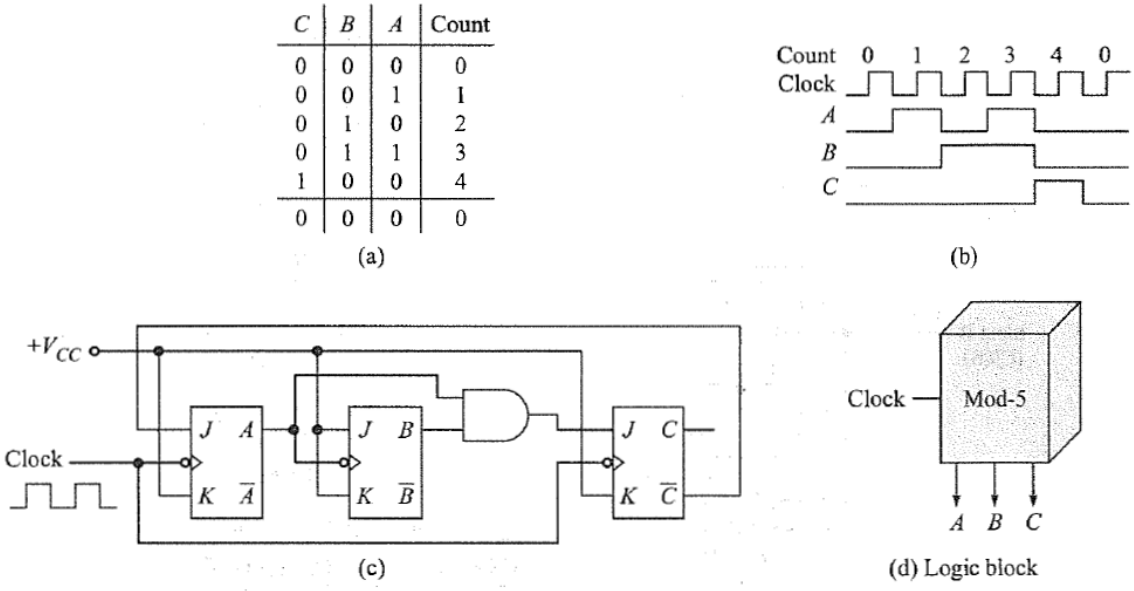


Fig.6: Mod 5 Counter

The three-flip-flop counter shown in Fig.6 has a natural count of 8, but it is connected in such a way that it will skip over three counts. It will, in fact, advance one count at a time, through a strict binary sequence, beginning with 000 and ending with 100; therefore, it is a mod-5 counter.

The waveforms show that flip-flop *A* changes state each time the clock goes negative, except during the transition from count 4 to count 0. Thus, flip-flop *A* should be triggered by the clock and must have an inhibit during count 4-that is, some signal must be provided during the transition from count 4 to count 0. Notice that *C* is high during all counts except count 4. If *C* is connected to the *J* input of flip-flop *A,* we will have the desired inhibit signal. This is true since the *J* and *K* inputs to flip-flop *A*are both true for all counts except count 4; thus the flip-flop triggers each time the clock goes negative. However, during count 4, the *J* side is low and the next time the clock goes negative the flip-flop will be prevented from being set. The connections which cause flip-flop *A* to progress through the desired sequence are shown in Fig. 6[a].

The desired waveforms (Fig.6 b) show that flip-flop *B* must change state each time *A* goes negative. Thus the clock input of flip-flop *B* will be driven by *A* (Fig. 6c).If flip-flop *C* is triggered by the clock while the *J* input is held low and the *K* input high, every clock pulse will reset it. Now, if the *J* input is high only during count 3, *C* will be high during count 4 and low during all other counts. The necessary levels for the *J* input can be obtained by ANDing flip-flops *A* and *B.* Since *A* and *B* are both high only during count 3, the *J* input to flip-flop *C* is high only during count 3. Thus, when the clock goes negative during the transition from

count 3 to count 4, flip-flop *C* will be set. At all other times, the J input to flip-flop *C* is low and is held in the reset state.

The 7490 [Decade Counter]

The 54/7490A is a TTL MSI decade counter. Its logic diagram, truth table, and pin out are given in Fig.7.A careful examination will shows that flip-flops *QB, Qc,* and *QD* form a mod-5 counter. The flip-flop *QD* in the '90A is an *RS* flip-flop that has a direct connection from its *Q* output back to its *R* input. The net result in this case is that *QD* behaves exactly like a *JK* flip-flop.

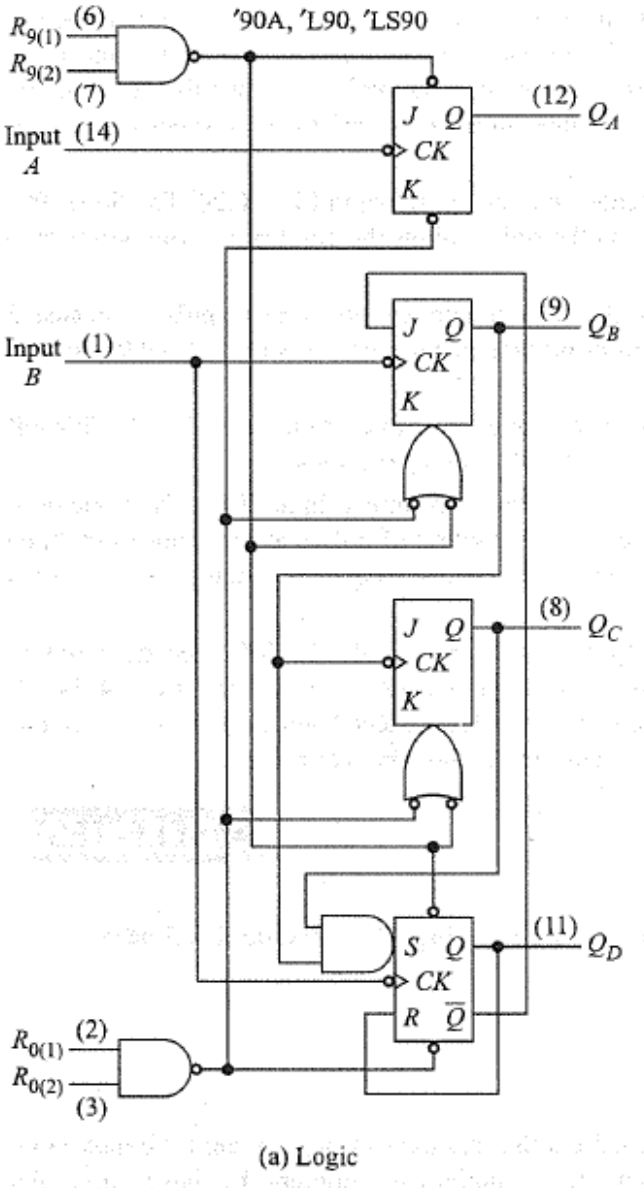


Fig.7: IC **54/7490A [Decade Counter]**

If the system clock is applied at input *A* and *QA* is connected to input *B,* we have a true binary decade counter. On the other hand, if the system clock is applied at input *B* and *QD* isconnected to input *A,* we have the biquinary counter.Fig7(b) shows truth table and fig.7(c) shows pin diagram of IC 7490

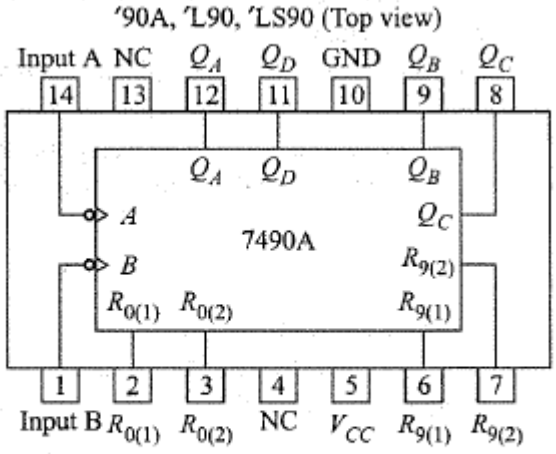
'90A, 'L90, 'LS90
BCD count sequence
(See note A)

Count	Output			
	Q_D	Q_C	Q_B	Q_A
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	L	H	L	H
6	L	H	H	L
7	L	H	H	H
8	H	L	L	L
9	H	L	L	H

'90A, 'L90, 'LS90
Bi-quinary (5-2)
(See note B)

Count	Output			
	Q_A	Q_D	Q_C	Q_B
0	L	L	L	L
1	L	L	L	H
2	L	L	H	L
3	L	L	H	H
4	L	H	L	L
5	H	L	L	L
6	H	L	L	H
7	H	L	H	L
8	H	L	H	H
9	H	H	L	L

(b) Truth table



Positive logic: See function tables

Note: A output Q_A connected to input B
B output Q_D connected to input A

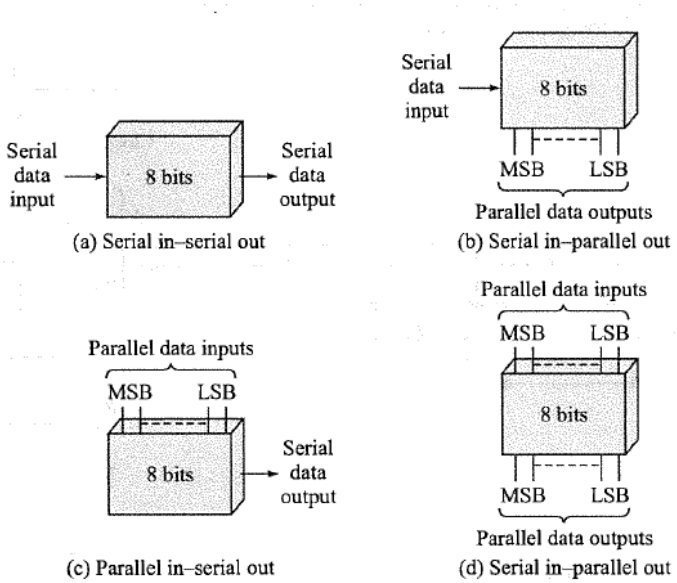
(c) Pinout

REGISTERS

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

The bits in a binary number (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 most significant bit (MSB) or the least significant bit (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 as shown in Fig. 1-serial in-serial out, serial in-parallel out, parallel in-serial out, and parallel in-parallel out

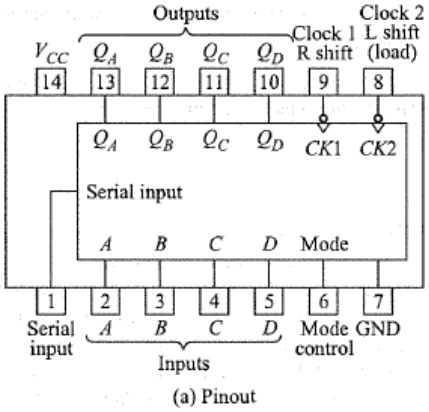


Shift register types

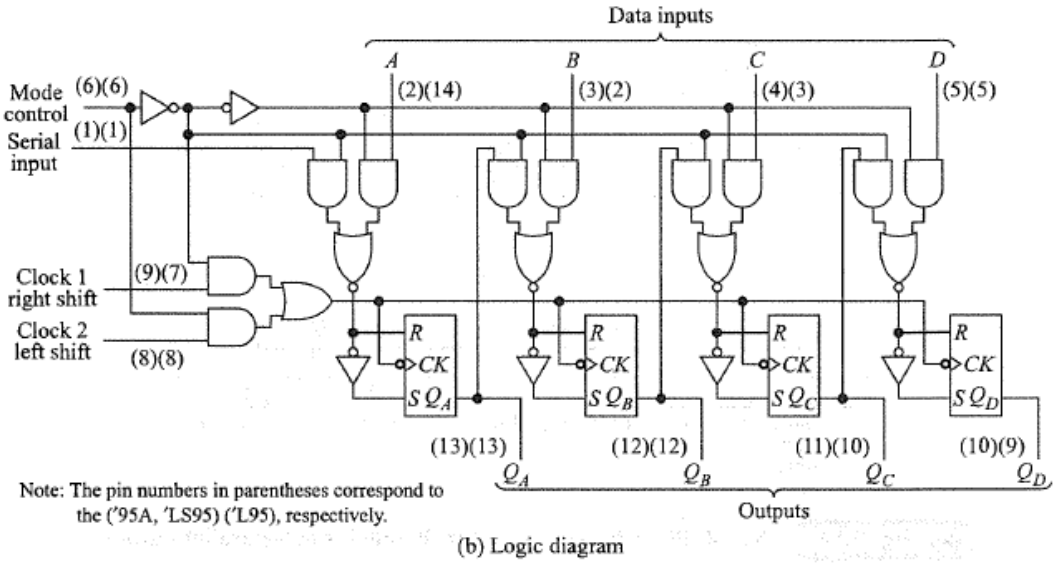
All of these configurations are commercially available as TTL MSI/LSI circuits. For instance:

Serial in-serial out-54/74LS91, 8 bits
Serial in-parallel out-54/74164, 8 bits
Parallel in-serial out-54/74165, 8 bits
Parallel in-parallel out-54/74198, 8 bits

IC 54/7495A [Universal shift register]



(a) Pinout



(b) Logic diagram

54/7495A

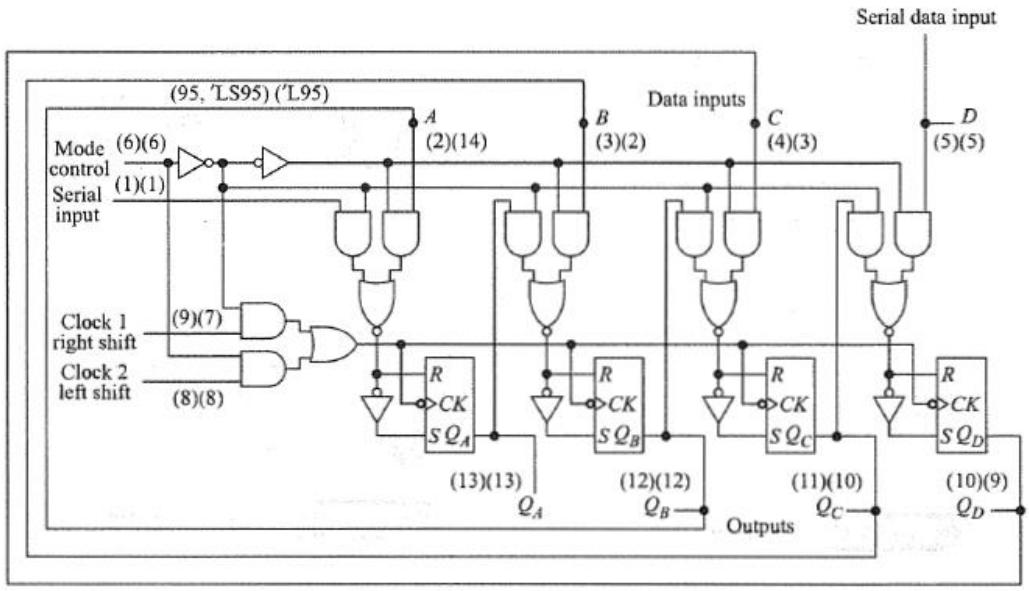
Fig.2: IC 54/7495

The data sheet for the 5417495A describes it as a 4-bit parallel-access shift register. It also has serial data input and can be used to shift data to the right (from *QA*toward QB) and in the opposite direction-to the left. he DIP pin out and logic diagram are given in Fig2. The parallel data outputs are simply the *Q* sides of each of the four flip-flops in the register. In fact, note that the output ***QD*** could be used as a serial output when data is shifted from left to right through the register(right shift).

When the mode control line is held high, the AND gate on the right input to each NOR gate is enabled while the left AND gate is disabled. The data at inputs, *A, B, C* and *D* will then be loaded into the register on a negative transition of the clock-this is parallel data input.

When the mode control line is low, the AND gate on the right input to each NOR gate is disabled whilethe left AND gate is enabled. The data input toflip-flop *QA* is now at serial input; the data input to $Q_B$is $Q_A$and so on down the line. On each clock NT, a data bit is entered serially into the register at the first flip-flopQ_A, and each stored data bit is shifted one flip-flop to the right (toward the last flip-flop Q_D). This is the serialinput of data (at serial input), and also the right-shift operation.

In order to effect a shift-left operation, the input data must be connected to the *D* data input as shown inFig. 3 below. It is also necessary to connect $Q_D$to C, $Q_C$to *B,* and $Q_B$to *A* as shown in Fig. 3. Now,when the mode control line is held high, data bit will be entered into flip-flop *QD,* and each stored data bitwill be shifted one flip-flop to the left on each clock NT. This is also serial input of data (but at input D) andis the left-shift operation.



7495A wired for shift left

There are two clock inputs--clock 1 and clock 2. This is to accommodate requirements where the clock used to shift data to the right is separate from the clock used to shift data to the left. If such a requirement is unnecessary, simply connect clock 1 and clock 2 together. The clock signal will then pass through the AND/OR gate combination non inverted, and the flip-flops will respond to clock NTs.