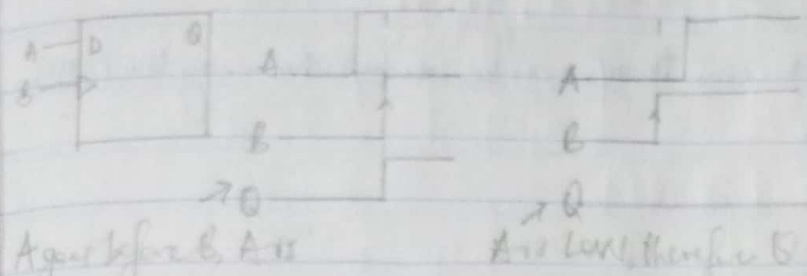
② Detecting an Input Sequence

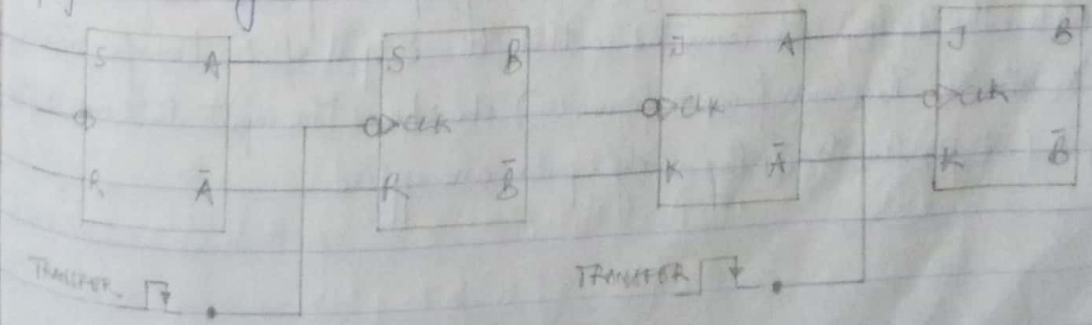
In many situations, an output is to be activated only when the inputs are activated in a certain sequence. This cannot be accomplished using pure combinational logic but requires the storage characteristics of flip flops

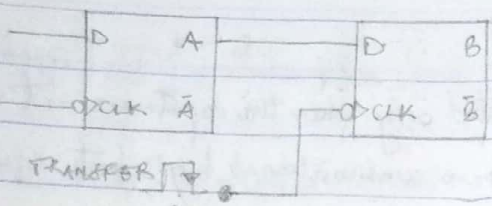


For example, an AND gate can be used to determine when two inputs A and B are both HIGH, but its output will respond the same regardless of which input goes HIGH first. But suppose that we want to generate a HIGH output only if A goes HIGH and then B goes HIGH some time later.

DATA STORAGE AND TRANSFER

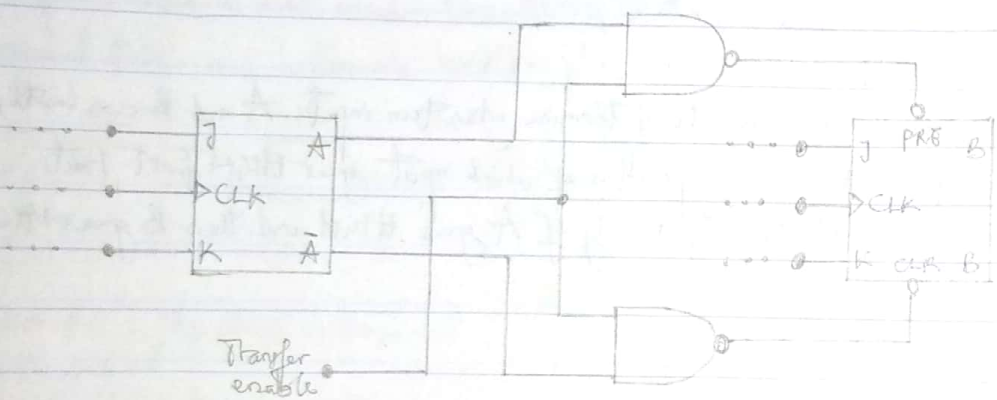
By far the most common use of flip flops is for the storage of data or information. The data may represent numerical values (e.g. binary numbers, BCD-coded decimal numbers) or any of a wide variety of types of data that have been encoded in binary. These data are generally stored in groups of flip flops called registers. The operation most often performed on data that are stored in a flip flop or a register is the data transfer operation. This involves the transfer of data from one flip flop or register to another. The transfer operations below are examples of synchronous transfer because the synchronous control and Clk inputs are used to perform the transfer.





Synchronous data transfer operation performed by various types of clocked flip-flops

The figure below shows how an asynchronous transfer can be accomplished using the PRESET and clear inputs of any type of flip-flops.

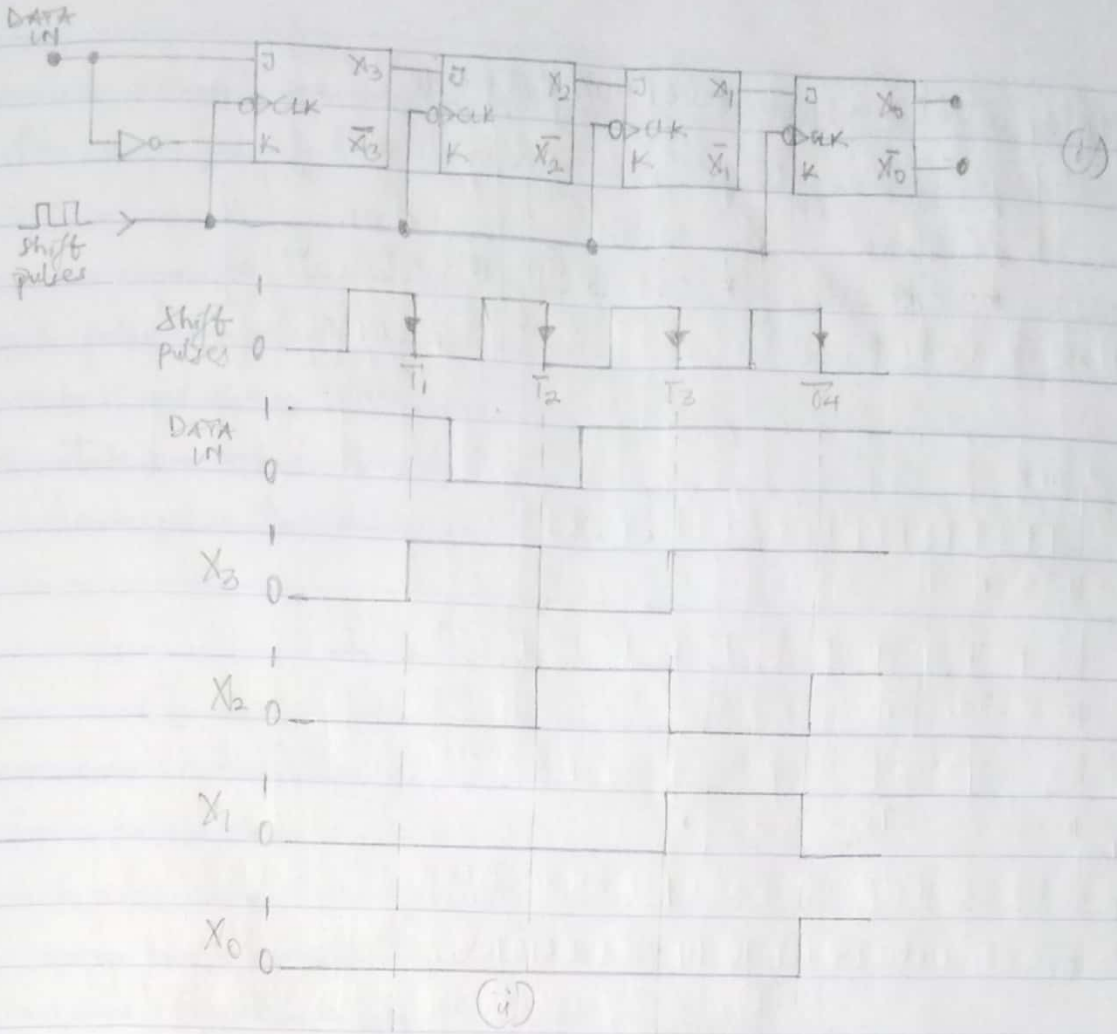


Here, the asynchronous inputs respond to LOW levels. When the TRANSFER ENABLE line is held LOW, the two NAND outputs are kept HIGH, with no effect on the flip-flop outputs. When the TRANSFER ENABLE line is made HIGH, one of the NAND outputs will go LOW, depending on the state of the A and Ā outputs.

SERIAL DATA TRANSFER: SHIFT REGISTERS

A shift register is a group of flip-flops arranged so that the binary numbers stored in the flip-flops are shifted from one flip-flop to the next for every clock pulse.

The figure(i) below shows one way to arrange J-K flip-flops to operate as a four-bit shift register. Note: The flip-flops are connected so that the output of X_3 transfers into X_2, X_2 into X_1 and X_1 into X_0. The waveforms in figure (ii) show how the input data are shifted from left to right from flip-flop to flip-flop as shift pulses are applied.

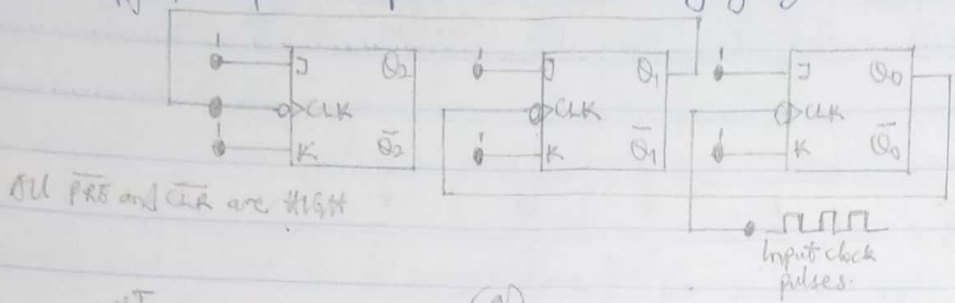


FREQUENCY DIVISION AND COUNTING

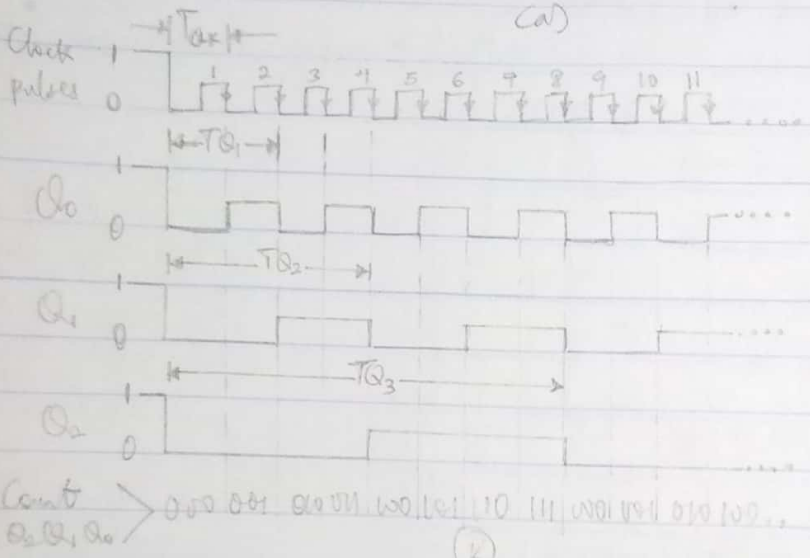
In the figure below, each flip-flop has its J and K inputs at the 1 level, so that it will change states (toggle) whenever the signal on its CLK input goes from HIGH to LOW. The clock pulses are applied only to the CLK input of flip-flop Q_0. Output Q_0 is connected to the CLK input of flip-flop Q_1 and output Q_1 is connected to the CLK input of flip-flop Q_2. The waveforms in figure (b) show how the flip-flops change states as the pulses are applied. The following important points should be noted:

1) Flip-flop Q_0 toggles on the negative-going transition of each input clock pulse. Thus, the Q_0 output waveform has a frequency that is exactly one-half of the clock pulse frequency.
2) Flip-flop Q_1 toggles each time the Q_0 output goes from HIGH to LOW. The Q_1 waveform has a frequency equal to exactly one-half the frequency of the Q_0 output and therefore one-fourth of the clock frequency.
3) Flip-flop Q_2 toggles each time the Q_1 output goes from HIGH to LOW. Thus, the Q_2 waveform has one-half the frequency of Q_1 and therefore one-eighth of the clock frequency.

4) Each flip flop output is a square wave (50% duty cycle).



(a)

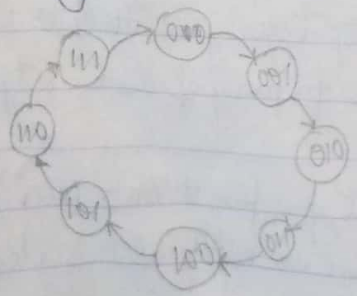


(b)

As described above, each flip flop divides the frequency of its input by 2. Thus, if we were to add a fourth flip flop to the chain, it would have a frequency equal to one-sixteenth of the clock frequency and so on. Using the appropriate number of flip flops, this circuit could divide a frequency by any power of 2. Specifically using N flip flops would produce an output frequency from the last flip flop, which is equal to $1/2^N$ of the input frequency. This application of flip flops is referred to as frequency division

STATE TRANSITION DIAGRAM

Another way to show how the states of the flip flops change with each applied clock pulse is to use a state transition diagram.



Note: Each arrow represents the occurrence of a clock pulse.

Each circle represents one possible state, as indicated by the binary number inside the circle. For example, the circle containing the number 100 represents the 100 state (i.e., $Q_2 = 1, Q_1 = Q_0 = 0$).

The arrow connecting one circle to another show how one state changes to another as a clock pulse is applied. By looking at a particular state circle, we can see which state precedes it and which state follows it. For example, looking at the 000 state, we see that this state is reached whenever the counter is in the 111 state and a clock pulse is applied. Likewise, we see that the 000 state is always followed by the 001 state.

Counters And Registers

Asynchronous (Ripple) Counters

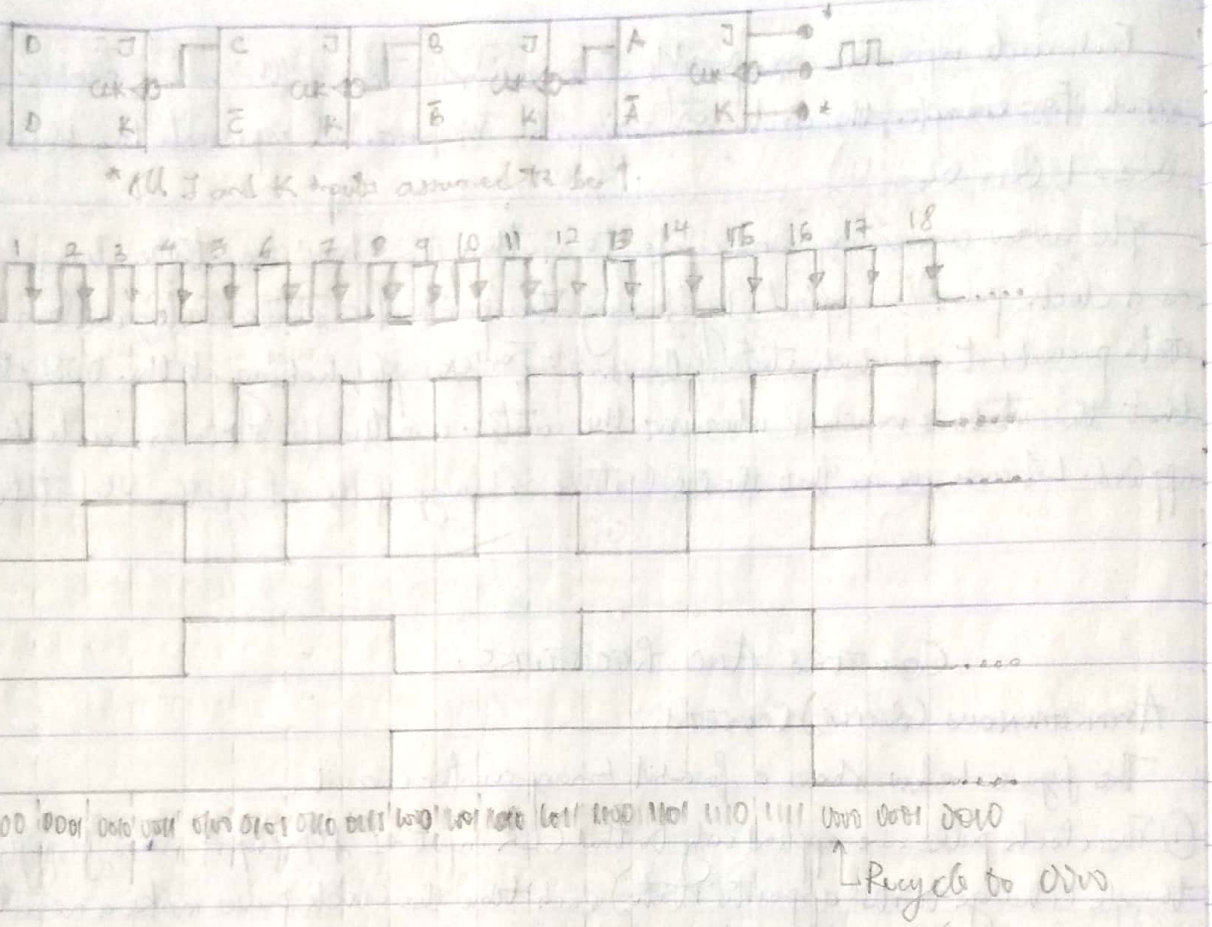
The figure below shows a four-bit binary counter circuit

① The clock pulse are applied only to the CLK input of flip-flop A. Thus, flip-flop A will toggle (change to its opposite state) each time the clock pulse make a negative (HIGH-to-LOW) transition. Note that J = K = 1 for all flip-flops.

② The normal output of flip-flop A acts as the CLK input for flip-flop B, and so flip-flop B will toggle each time the A output goes from 1 to 0. Similarly, flip-flop C will toggle when B goes from 1 to 0 and flip-flop D will toggle when C goes from 1 to 0.

③ Flip-flop outputs D, C, B and A represent a four-bit binary number with D as the MSB. Assuming that all flip-flops have been cleared to the 0 state (CLEAR inputs are not shown). The waveforms in the figure (a) shows that a binary counting sequence from 0000 to 1111 is followed as clock pulses are continuously applied.

④ After the NGT of the fifteenth clocks pulse has occured, the counter flip-flops are in the 1111 condition. On the sixteenth NGT, flip-flop A goes from 1 to 0, which causes flip-flop B to go from 1 to 0 and so on until the counter is in the 0000 state. In other words, the counter has gone through one complete cycle (0000 through 1111) and has recycled back to 0000. From this point, it will begin a new counting cycle as subsequent clock pulses are applied.



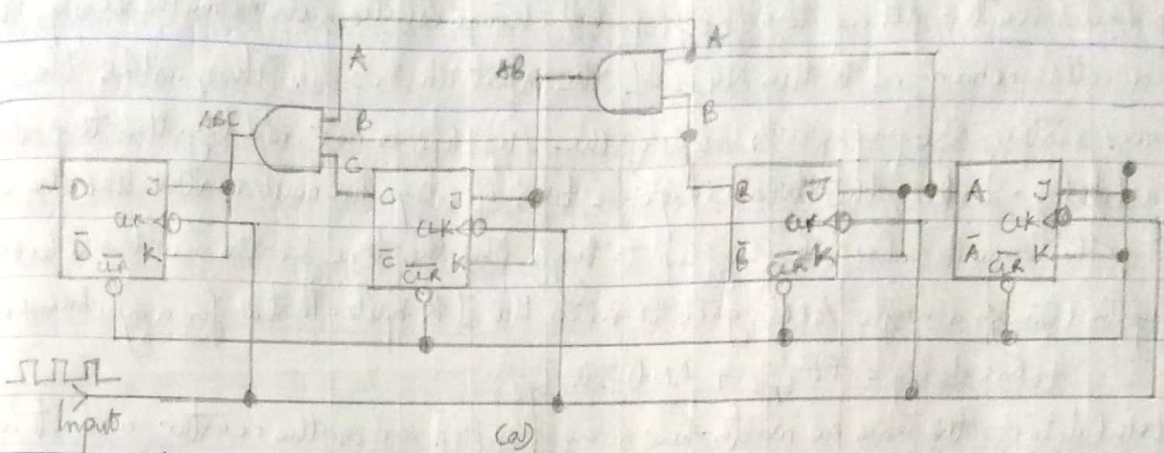
Fig*

PROPAGATION DELAY IN RIPPLE COUNTERS

Ripple counters are the simplest type of binary counters because they require the fewest components to produce a given counting operation. They do, however, have one major drawback, which is caused by their basic principle of operation: each flip-flop is triggered by the transition at the output of the preceding flip-flop.

SYNCHRONOUS (PARALLEL) COUNTERS

The problems encountered with ripple counters are caused by the accumulated flip-flop propagation delays; stated another way, the flip-flops do not all change states simultaneously in synchronism with the input pulses. These limitations can be overcome with the use of synchronous or parallel counters in which all of the flip-flops are triggered simultaneously (in parallel) by the clock input pulses.



(a)

Count	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1
0	0	0	0	0

Because the input pulses are applied to all flip-flops, some means must be used to control when a flip-flop is to toggle and when it is to remain unaffected by a clock pulse. This is accomplished by using the J and K inputs and is illustrated in the figure above for a four-bit, mod-16 synchronous counter. If we compare the circuit arrangement for this synchronous counter with its asynchronous counterpart in Fig *, we can see the following notable differences:

- The CLK inputs of all the flip-flops are connected together so that the input clock signal is applied to each flip-flop simultaneously.
- Only flip-flop A, the LSB, has its J and K inputs permanently at the HIGH level. The J,K inputs of the other flip-flops are driven by some combination of flip-flop outputs.
- The synchronous counter requires more circuitry than does the asynchronous counter.

Advantage Of Synchronous Counters Over Asynchronous

In a parallel counter, all of the flip flops will change states ~~either~~ simultaneously; that is, they are all synchronized to the N&Ts of the input clock pulses. Thus, unlike the asynchronous counters, the propagation delays of the flip flops do not add together to produce the overall delay. Instead, the total response time of a synchronous counter like the one in fig (a) is the time it takes one flip flop to toggle plus the time for the new logic levels to propagate through a single AND gate to reach the J/K inputs. That is, for a synchronous counter,

$$\text{total delay} = FF_{tpd} + \text{AND gate } t_{pd}$$

This total delay is the same no matter how many flip flops are in the counter, and it will generally be much lower than with an asynchronous counter with the same number of flip flops. Thus, a synchronous counter can operate at a much higher input frequency. Of course, the circuitry of the synchronous counter is more complex than that of the asynchronous counter.

Decoding A Counter

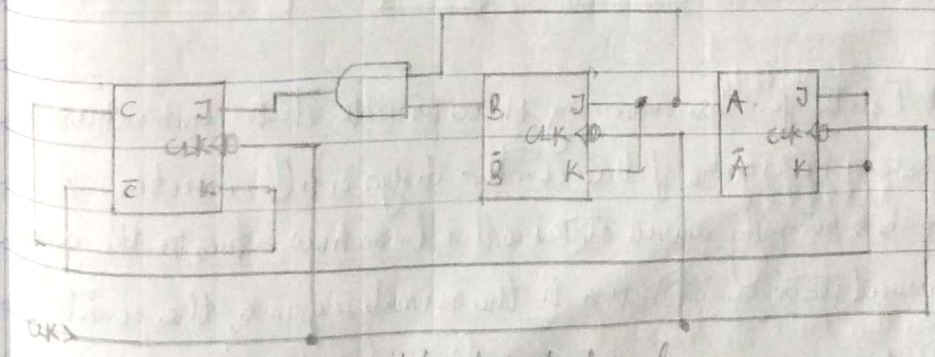
Digital counters are often used in applications where the count represented by the states of the flip flops must somehow be determined or displayed. One of the simplest means for displaying the contents of a counter involves just connecting the output of each flip flop to a small indicator LED. In this way the states of the flip flops are visibly represented by the LEDs (on = 1, off = 0), and the count can be mentally determined by decoding the binary state of the LEDs. For instance, suppose that this method is used for a BCD counter and the states of the LEDs are off-on-on-off, respectively. This would represent 0110, which we would mentally decode as decimal 6. Other combinations of LED states would represent the other possible counts.

The indicator LED method becomes inconvenient as the size (number of bits) of the counter increases because it is much harder to decode the displayed results mentally. For this reason, it is preferable to develop a means for electronically decoding the contents of a counter and displaying the results in a form that is immediately recognizable and requires no mental operations.

ANALYZING SYNCHRONOUS COUNTERS

Synchronous counter circuits can be custom-designed to generate any desired count sequence. We can use just the synchronous inputs that are applied to the individual flip flops to produce the counter's sequence. By not using asynchronous flip flop controls, such as the clears, to change the counter's sequence, we will never have to deal with transient states and possible glitches in output waveforms. The process of designing completely synchronous counters will be investigated here. First, let's see how to analyze a counter design of this type by predicting the flip flop control inputs for each state of the counter. A PRESENT state/NEXT state is a very useful tool in this analysis process. The first step is to write the logic expressions for each flip flop control input. Next assume a PRESENT state for the counter and apply that combination of bits to the control logic expressions. The outputs from the control expressions will allow us to predict the commands to each flip flop and the resulting NEXT state for the counter after clocking. The process should be repeated until the entire count sequence is determined. The control input expressions for this counter are: $J_C = A \cdot B$, $K_C = C$, $J_B = K_B = A$, $J_A = K_A = C$.

Let's assume that the PRESENT state for the counter is CBA = 000. Applying this combination to the control expressions above will yield $J_C K_C = 00$, $J_B K_B = 00$ and $J_A K_A = 11$. These control inputs will tell flip flops C and B to hold and flip flop A to toggle on the next ↓CLK on CLK. Our predicted NEXT state is 001 for CBA. This information has been entered in the first line of the PRESENT state/NEXT state table shown in the figure below.

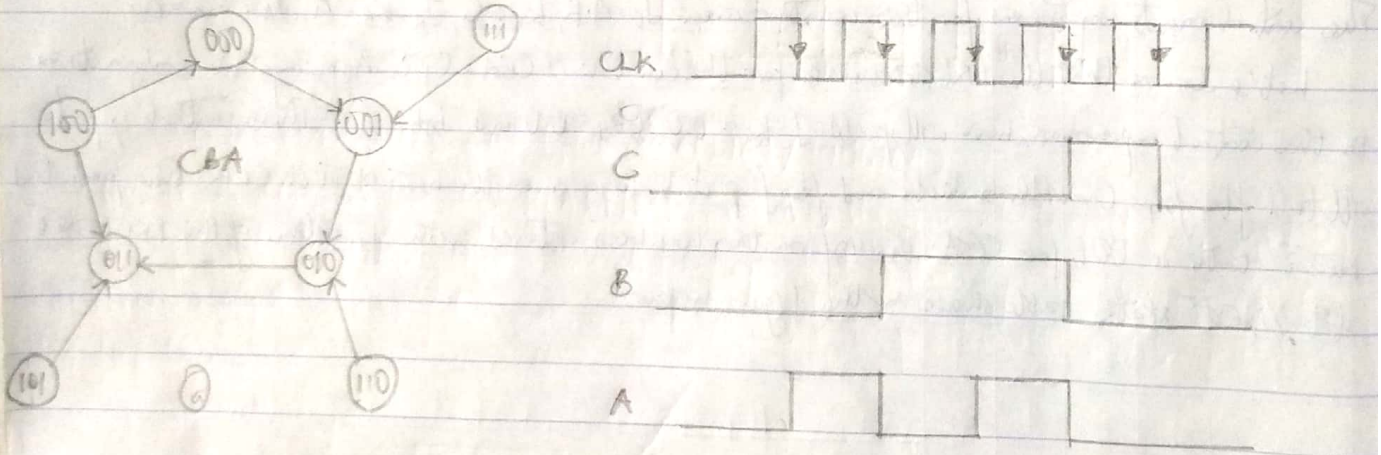


Synchronous counter with different control inputs

Next, we can use the state 001 as our PRESENT state. Analyzing the control expressions with this new combination will now yield $J_C K_C = 00$, $J_B K_B = 11$, and $J_A K_A = 11$ giving us a hold command for flip flop C and toggle commands for flip flops B and A. This will produce a NEXT state of 010 for CBA, which is listed on the second line of the table below.

PRESENT state			Control Inputs						NEXT State		
C	B	A	J_C	K_C	J_B	K_B	J_A	K_A	C	B	A
0	0	0	0	0	0	0	1	1	0	0	1
0	0	1	0	0	1	1	1	1	0	1	0
0	1	0	0	0	0	0	1	1	0	1	1
0	1	1	1	0	1	1	1	1	1	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	1	0	1	1	1	0	0	0	1	1
1	1	0	0	1	0	0	0	0	0	1	0
1	1	1	1	1	1	1	0	0	0	0	1

Continuing with this process will result in a recycling count sequence of 000, 001, 010, 011, 100, 000. This would be a MOD-5 count sequence.



We can also predict the NEXT states for the remaining three possible state combinations in the same way. By doing so, we can determine if the counter design is self-correcting. A self-correcting counter is one in which normally unused states will all somehow return to the normal count sequence. If any of these unused states cannot return to the normal sequence, the counter is said to be not self-correcting.

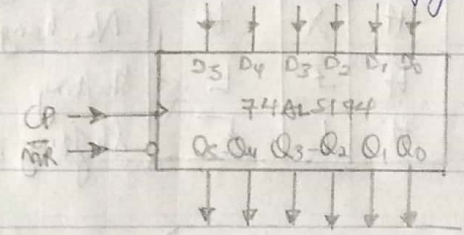
INTEGRATED CIRCUIT REGISTERS

The various types of registers can be classified according to the manner in which data can be entered into the register for storage and the manner in which data are outputted from the register. The various classifications are listed below:

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

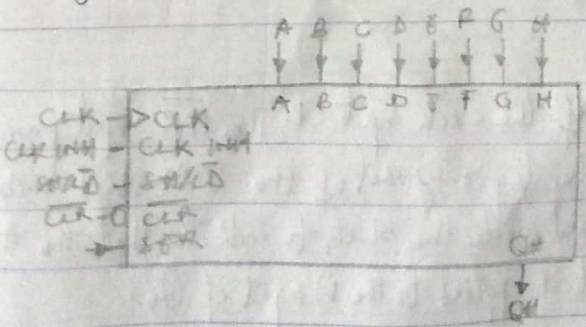
PARALLEL IN/PARALLEL OUT (THE 74ALS174/74HC174)

A group of flip-flops that can store multiple bits simultaneously and in which all bits of the stored binary value are directly available is referred to as a parallel in/parallel out register. Figure (a) shows the logic diagram for the 74ALS174 (also the 74HC174), a six-bit register that has parallel inputs D_5 through D_0 and parallel outputs Q_5 through Q_0. The logic symbol for the 74ALS174 is shown in the figure (b) below.



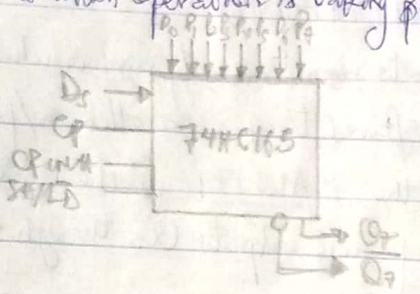
SERIAL IN/SERIAL OUT (THE 74ALS166/74HC166)

A serial in/serial out shift register will have data loaded into it one bit at a time. The data will move one bit at a time with each clock pulse through the set of flip-flops toward the other end of the register. With continued clocking, the data will then exit the register ~~the register~~ this register one bit at a time in the same order as it was originally loaded. The 74HC166 (and also the 74ALS166) can be used as a serial in/serial out register. The logic diagram and schematic symbol for the 74HC166 in the figure below



PARALLEL IN/SERIAL OUT (THE 74ALS165/74HC165)

The logic symbol for the 74HC165 is shown in the figure below. This IC is an eight-bit parallel in/serial out register. It actually has serial data entry via D_S and asynchronous parallel data entry via P_0 through P_7. The register contains eight flip-flops Q_0 through Q_7 internally connected as a shift register, but the only accessible flip-flop outputs are Q_7 and $\overline{Q}_7$. CP is the clock input used for the shifting operation. The clock input inhibit input, CP INH is used to inhibit the effect of the CP input. The shift/load input, SH/$\overline{LD}$, controls which operation is taking place - shifting or parallel loading.

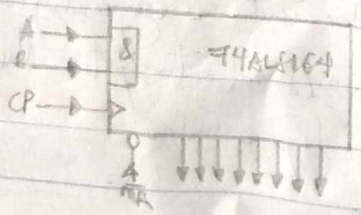


Inputs			
SH/$\overline{LD}$	CP	CP INH	Operation
L	X	X	Parallel Load
H	H	X	No change
H	X	H	No change
H	↓	L	Shifting
H	L	↓	Shifting

H = High level, L = Low level
X = Immaterial, ↓ = PGT

SERIAL IN/PARALLEL OUT (THE 74ALS164/74HC164)

The logic diagram for the 74ALS164 is shown in the figure below. It is an eight-bit serial in/parallel out shift register with each flip-flop output externally accessible. Instead of a single serial input, an AND gate combines inputs A and B to produce the serial input to flip-flop Q_0.



The shift operation occurs on the PGT of the clock input CP. The $\overline{MR}$ input provides asynchronous resetting of all flip-flops on a LOW level. The logic symbol of the 74ALS164 is shown in the figure above. Note that the & symbol is used inside the block to indicate that the A and B inputs are ANDed inside the IC and the result is applied to the D input of Q_0.

SHIFT-REGISTER COUNTERS

Shift-register counters use feedback, which means that the output of the last flip-flop in the register is connected back to the first flip-flop in some way.

RING COUNTER

The simplest shift-register counter is essentially a circulating shift register connected so that the last flip-flop shifts its value into the first flip-flop. The flip-flops are connected so that information shifts from left to right and back around from Q_0 to Q_3. In most instances, only a single 1 is in the register and it is made to circulate around the register as long as clock pulses are applied. For this reason, it is called a ring counter.

~~The waveform seq~~