MODULE II

Logic Circuits: Logic gates, Bistables, R-S Bistables, D-type Bistables, J-K Bistables.

Data representation, Data types, Data storage, A microcontroller system.

Realization using basic gates and truth table of Half Adder and Full Adder, Multiplexer and Decoder.

Shift registers, Register type: Operation and truth table, Counters and asynchronous counters.

2.1 Logic Gates

- Logic gates are electronic circuits designed to produce the basic logic functions such as AND, OR, etc.

2.1.1 Buffers

- Buffers do not affect the logical state of a digital signal.
- A logic 1 input results in a logic 1 output & a logic 0 input results in a logic 0 output.
- Buffers are used to provide extra current drive at the output.
- The Boolean expression for the output: **Y = X**

2.1.2 Inverters

- Inverters are used to complement the logical state.
- A logic 1 input results in a logic 0 output and vice versa.
- The Boolean expression for the output Y, of an Inverter with an input, X, is: $\mathbf{Y = \overline{X}}$

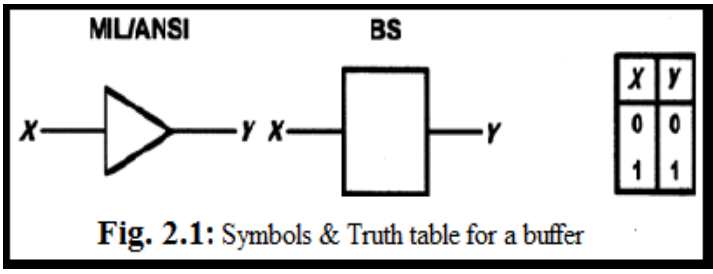


Fig. 2.1: Symbols & Truth table for a buffer

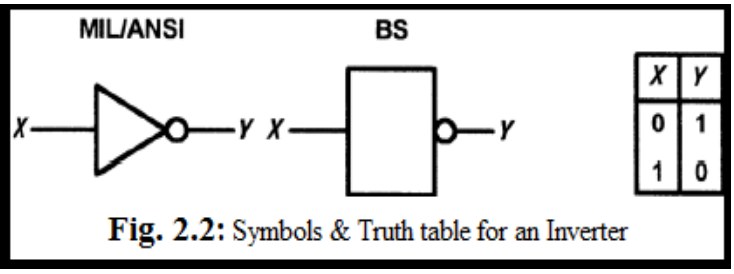


Fig. 2.2: Symbols & Truth table for an Inverter

2.1.3 AND gates

- AND gates will produce logic 1 output only when all inputs are simultaneously at logic 1.
- Any other input combination results in a logic 0 output.
- The Boolean expression for the output: **Y = A · B**

2.1.3 OR gates

- OR gates will produce logic 1 output whenever any one, or more inputs are at logic 1.
- OR gate will produce a logic 0 output only whenever all of its inputs are at logic 0.
- The Boolean expression for the output: **Y = A + B**

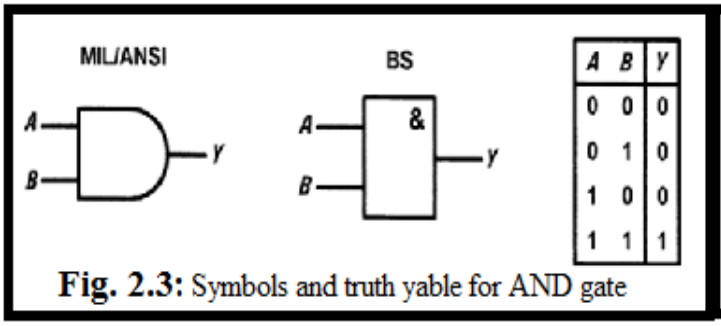


Fig. 2.3: Symbols and truth yable for AND gate

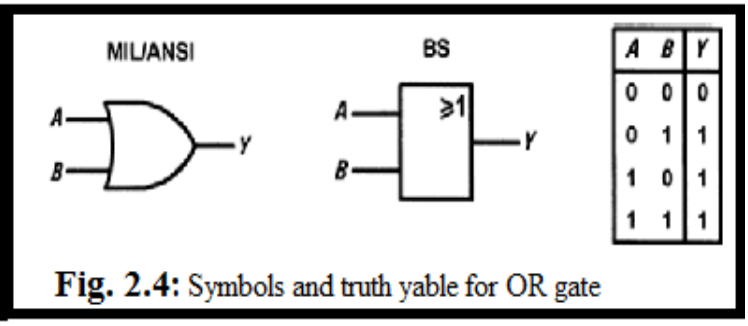


Fig. 2.4: Symbols and truth yable for OR gate

2.1.3 NAND gates

- NAND gate will produce logic 0 output only when all inputs are simultaneously at logic 1
- Other input combination will produce a logic 1 output.
- A NAND gate, is an AND gate with its output inverted (NOT-AND). The circle shown at the output denotes this inversion.
- The Boolean expression for the output: $\mathbf{Y = \overline{A\,B}}$

2.1.3 NOR gates

- NOR gates will produce logic 1 output only when all inputs are simultaneously at logic 0
- Other input combination will produce a logic 0 output.
- A NOR gate, is an OR gate with its output inverted (NOT-OR).
- The Boolean expression for the output: $\mathbf{Y = \overline{A + B}}$

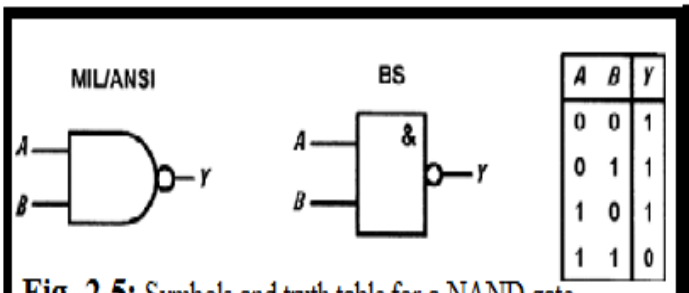


Fig. 2.5: Symbols and truth table for a NAND gate

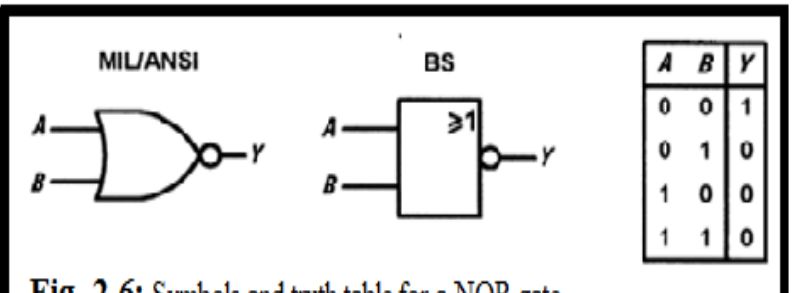


Fig. 2.6: Symbols and truth table for a NOR gate

2.1.7 Exclusive-OR gate

- A two input Exclusive-OR gates will produce a logic 1 output, whenever either one of the inputs is at logic 1 and the other is at logic 0.
- Exclusive-OR gates produce a logic 0 output whenever both inputs have the same logical state (i.e. when both are at logic 0 or when both are at logic 1).
- Boolean expression for the output: $\mathbf{Y = \overline{A}\,B + A\,\overline{B}}$

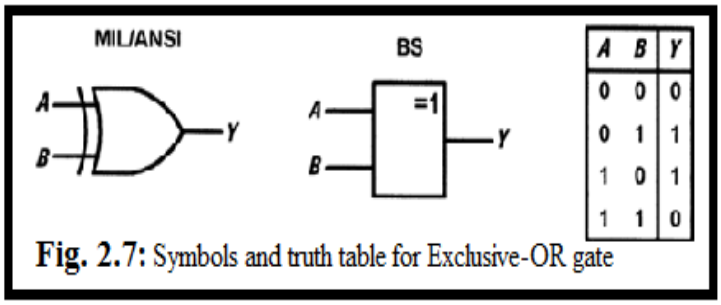


Fig. 2.7: Symbols and truth table for Exclusive-OR gate

2.2 Combinational Logic Circuits

- By using a standard range of logic levels, logic circuits can be combined in order to solve complex logic functions.

Example 1: Construct a logic circuit that will produce a logic 1 output whenever two or more of its three inputs are at logic 1.

- The truth table for the given condition is given in fig. 2.8(a).

A	B	C	Y	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	← B·C
1	0	0	0	
1	0	1	1	← A·C
1	1	0	1	← A·B
1	1	1	1	← A·B·C

$Y = (B \cdot C) + (A \cdot C) + (A \cdot B) + A \cdot B \cdot C$

$= (A \cdot C) + (A \cdot B) + (B \cdot C)$

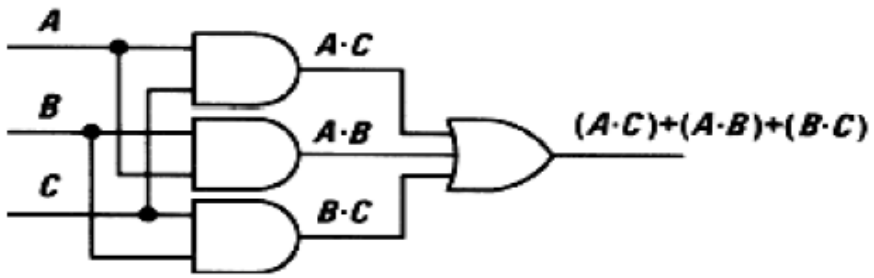


Fig. 2.8: (a) Truth Table (b) Logic Circuit

- It is observed from truth table that Y=1 for rows which have two 1's otherwise it is 0. Therefore its Boolean function is Y = A B + B C + C A
- The logic circuit for above equation is shown in fig. 2.8(b).

Example 2: Construct an exclusive or gate using only (AND, OR and NOT) basic gates.

- The Boolean expression for EXOR gate is: $Y = \overline{A}B + A\overline{B}$ This can be implemented using basic gates as shown in Fig. 2.9.

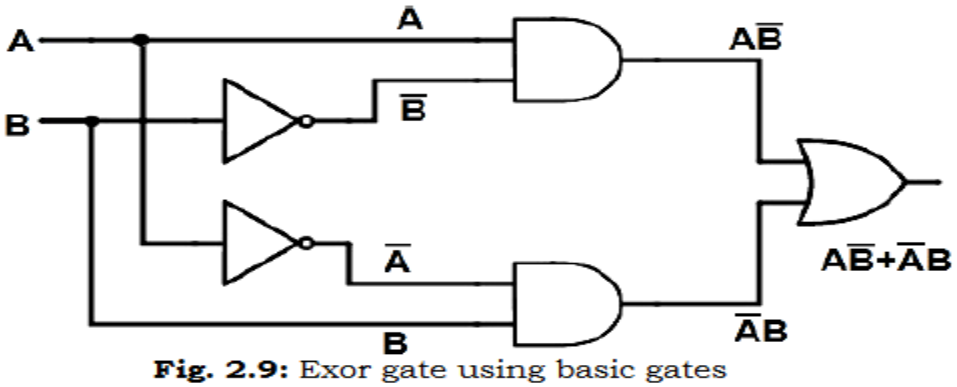


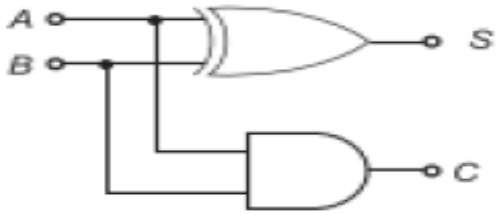
Fig. 2.9: Exor gate using basic gates

2.2.1 Half Adder

- An adder is a digital logic circuit that is used for the addition of numbers.
- The combinational circuit that performs addition of two bits is called Half Adder.
- Half adder adds two 1-bit binary inputs and provides two binary outputs: Sum & Carry.

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) Truth table:
S = Sum; C = Carry



(b) Circuit using XOR

Fig. 2.8: Half Adder

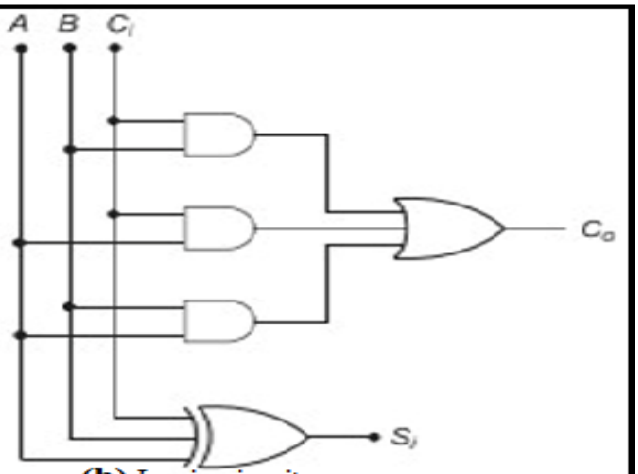
- From truth table, we can write: $S = \overline{A}B + A\overline{B} = A \oplus B$, $C = AB$

2.2.2 Full Adder

- A Full Adder adds three inputs and produces two outputs.
- The first two inputs are A and B and the third input is an input carry, C_i and the output carry is given as C_o.

A	B	C_i	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a) Truth Table



(b) Logic circuit

Fig. 2.9: Full Adder

- It is observed from truth table that C_0=1 for rows which have two 1's otherwise it is 0. Therefore its Boolean function is C_0= A B + B C_i + C_i A
- Again in truth table, S=1 for rows with one 1 and three 1's., i.e. odd number of 1's. Hence it can be implemented by a three input XOR.
- The logic circuit is shown in Fig. 2.9 (b).
- A full adder can be implemented using 2 half adders and OR gate as shown in fig. 2.9(c).

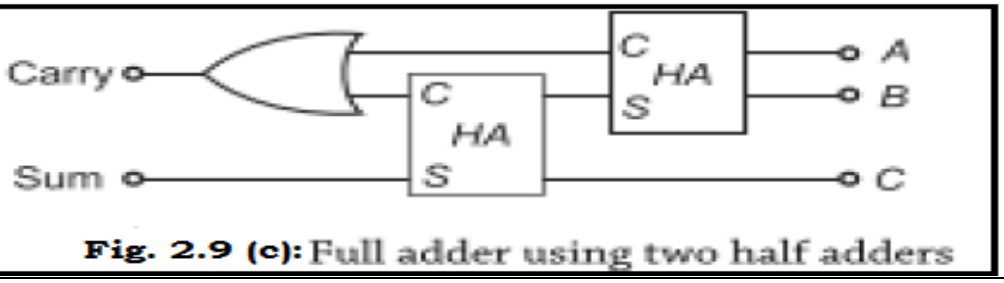
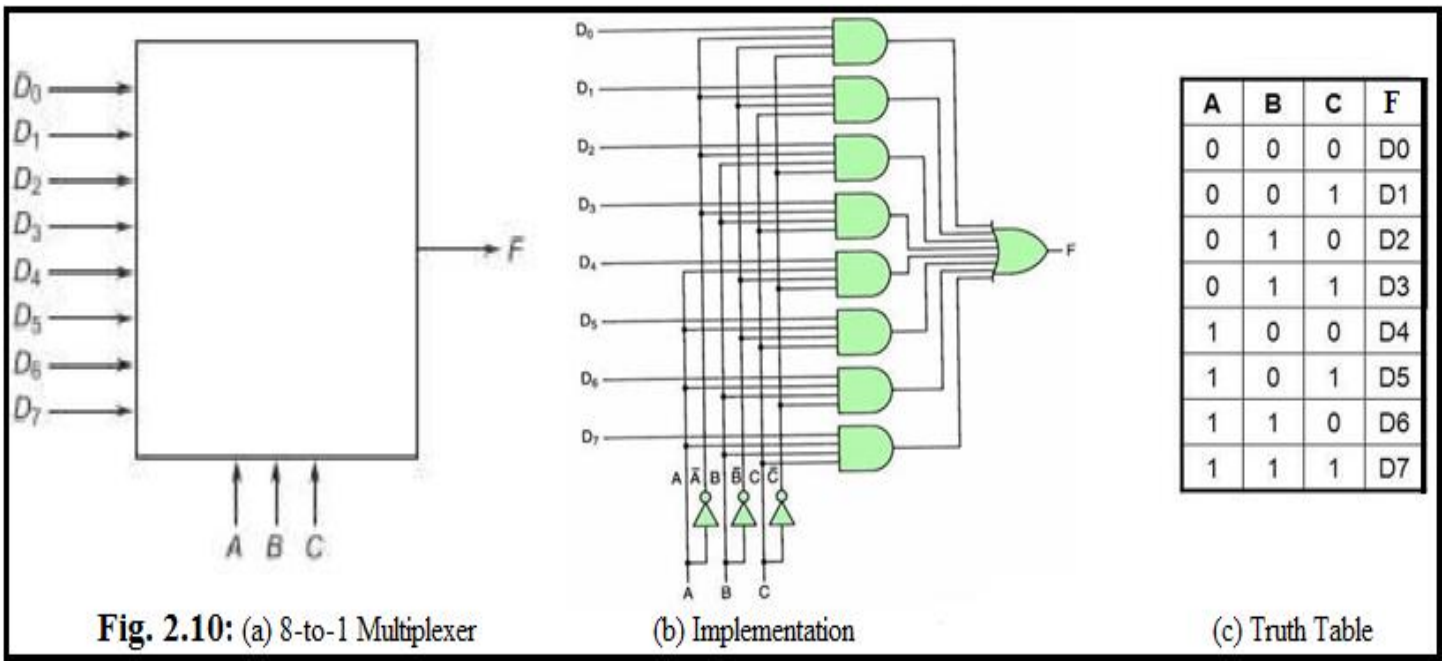


Fig. 2.9 (c): Full adder using two half adders

2.2.3 Multiplexers

- Multiplexer is logic circuit that selects one signal from a group of 2^n inputs, to be an output on a single output line.
- Select lines are available to choose one of the input lines to be an output on output line.
- For example, 8–to-1 multiplexer shown in fig. 2.10 (a), lines D_0……D_7 are the data input lines and F is the output line. Lines A, B and C are called the select lines, which is used to choose one of the D lines to be output on the line F.
- A multiplexer can be designed with a regular pattern of AND and OR gates, as shown in fig. 2.10 (b) and the corresponding truth table is given in fig. 2.10 (c).



A	B	C	F
0	0	0	D0
0	0	1	D1
0	1	0	D2
0	1	1	D3
1	0	0	D4
1	0	1	D5
1	1	0	D6
1	1	1	D7

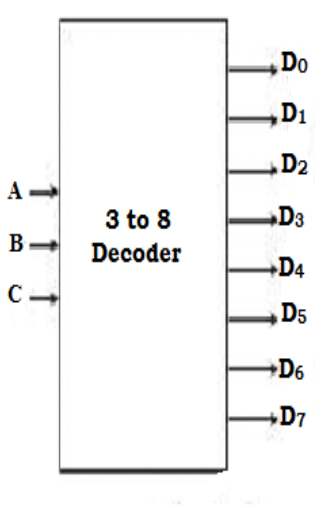
Fig. 2.10: (a) 8-to-1 Multiplexer (b) Implementation (c) Truth Table

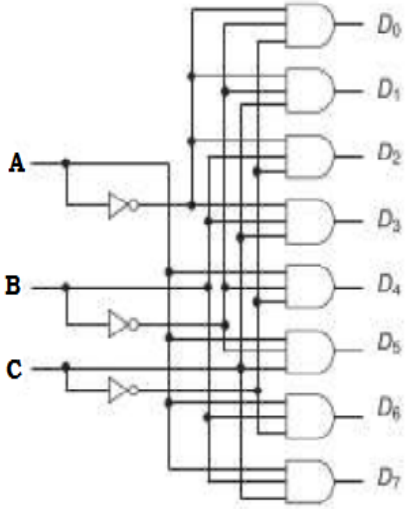
- Suppose if we need the output F = D_5, then three inputs to the corresponding AND gate should be 1s. Then, if D_5=0, F=0 & if D_5=1, F=1. Thus, select line inputs are ABC= 101.
- **Applications:** Multiplexers are used in telephone network, computer memory, ALU etc.

2.2.4 Decoders

- A decoder is a logic circuit that decodes an n-bit binary number, producing a signal on one of 2^n output lines.
- **3 to 8 decoder:** A 3 to 8 decoder has three inputs (A, B, C) and eight outputs (D_0 to D_7). Based on the 3 inputs one of the eight (2^3 = 8) outputs is selected.

  The block diagram of a 3 to 8 decoder is shown in fig. 2.11(a). It can be implemented using basic logic gates as shown fig. 2.11(b). Truth table is shown in fig. 2.11(c).
- **Applications:** Data multiplexing, Memory address decoding, 7 segment display.





INPUTS			OUTPUTS							
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

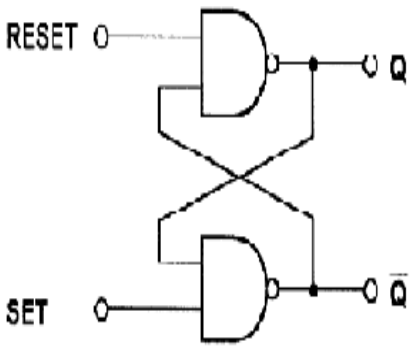
Fig. 2.11: (a) 3 to 8 Decoder (b) Implementation (c) Truth Table

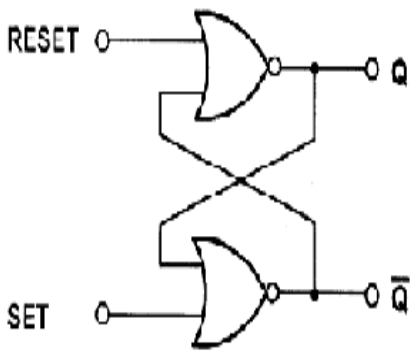
2.3 Bistables

- All memory elements consist of a basic bistable element.
- The bistable element has two stable conditions or states (logic 0 or logic 1).
- Types of Bistables: R-S bistables, D type bistables, J-K bistables.

2.3.1 R-S Bistables

- The simplest form of bistable is the R-S bistable.
- This device has two inputs, SET and RESET, and complementary outputs, Q and $\bar{Q}$.
- A logic 1 applied to the SET input will cause the Q output to become logic 1 while a logic 1 applied to the RESET input will cause the Q output to become logic 0.
- In either case, the bistable will remain in its SET or RESET state until an input is applied in such a sense as to change the state.
- Simple form of R-S bistable based on cross coupled logic gates are shown in Fig. 2.12.





INPUTS		OUTPUTS		COMMENTS
S	R	Q_{N+1}	$\overline{Q_{N+1}}$	
0	0	Q	$\bar{Q}$	Memory
0	1	0	1	Reset
1	0	1	0	Set
1	1	?	?	Forbidden

Fig. 2.12: R-S bistables using cross-coupled NAND and NOR gates

2.3.1 D type Bistables

- D-type bistable has two inputs: D ('data' or 'delay') and CLOCK (CLK).

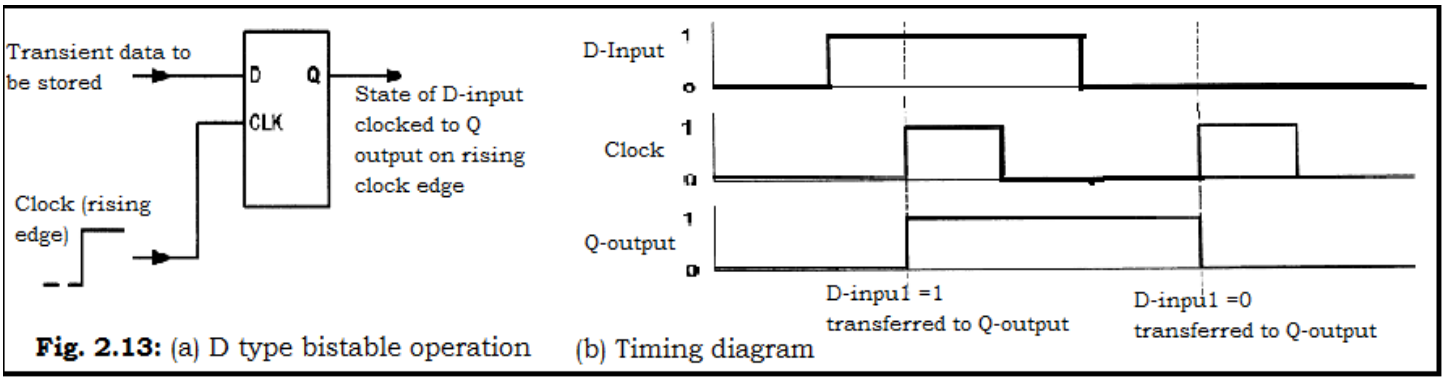


Fig. 2.13: (a) D type bistable operation (b) Timing diagram

- The data input (logic 0 or logic 1) is clocked into the bistable such that the output state only changes when the clock changes state.
- Operation is thus said to be synchronous.
- D-type bistables are used both as latches (a form of memory) and as binary dividers.
- Fig. 2.13 illustrates the operation of D-type bistables with timing diagram shown in (b).

2.3.3 J-K bistables

- J_K bistables have two clocked inputs (J and K), two direct inputs (PRESET and CLEAR), a CLOCK (CK) input, and outputs (Q and $\overline{Q}$).
- Similarly, the PRESET and CLEAR inputs are invariably both active low.
- A 0 on the PRESET input will set the Q output to 1 whereas a 0 on the CLEAR input will set the Q output to 0.
- Tables 2.1 and 2.2 summarize the operation of a J-K bistable, respectively, for the PRESET and CLEAR inputs and for clocked operation.

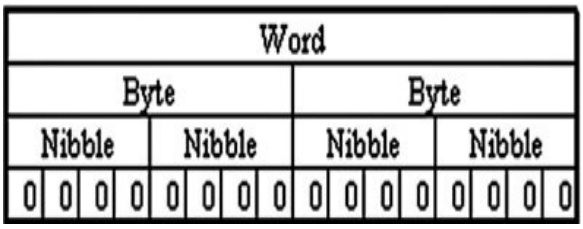
Inputs		Output	Comments
PRESET	CLEAR	Q_{N+1}	
0	0	?	Indeterminate
0	1	0	Q output changes to 0 (i.e. Q is reset) regardless of the clock state
1	0	1	Q output changes to 1 (i.e. Q is set) regardless of the clock state
1	1	See Right	Enables clocked operation (refer to Table 2.2)

Table 2.1: Input and output states for a J-K bistable (PRESET and CLEAR inputs)

Inputs		Output	Comments
J	K	Q_{N+1}	
0	0	Q_N	No change in state of the Q output on the next clock transition
0	1	0	Q output changes to 0 (i.e. Q is reset) on the next clock transition
1	0	1	Q output changes to 1 (i.e. Q is set) on the next clock transition
1	1	Q_N	Q output changes to the opposite state on the next clock transition

Table 2.2: Input and output states for a J-K bistable (clocked operation)

2.4 Data representation



- Data's are represented using binary numbers.
- Large binary numbers are inconvenient to handle & hence are converted to hexadecimal.
- **Nibble:** A group of 4 bits or single hex character.
- **Byte:** A group of 8 bits, represented by two hex characters.
- **Word:** A group of 16 bits, represented by four hex characters.
- **Double Word:** A group of 32 bits, represented by eight hex characters.
- The value of a byte expressed in binary can be easily converted to hex by arranging the bits in groups of four and converting each nibble into hexadecimal using Table 2.3.
- Each hexadecimal number is represented using $ before the number or add H to the end of the number.
- **Example 1:** Convert hexadecimal A3 into binary.
- From Table 2.3, A = 1010 and 3 = 0011. Thus A3 in hexadecimal is equivalent to 10100011 in binary.
- **Example 2:** Convert binary 11101000 binary to hex.
- From Table 2.3, 1110 = E and 1000 = 8. Thus 11101000 in binary is equivalent to E8 in hexadecimal.

Binary	Decimal	Hex
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F

Table 2.3: Binary, Decimal & Hexa decimal

2.4.1 Data types

- A byte of data can be stored at each address within the total memory space of a microprocessor system.
- Hence one byte can be stored at each of the 65,536 memory locations within a microprocessor system having a 16-bit address bus.
- The range of integer data values that can be represented as bytes, words and long words are shown in Table 2.4.

2.4.2 Data storage

- Semiconductor **ROM** within a microprocessor system provides storage for program code as well as permanent data that requires storage.
- All these data are referred to as non-volatile because they remain intact when the power supply is disconnected.

Data type	Bits	Range of values
Unsigned byte	8	0 to 255
Signed byte	8	–128 to +127
Unsigned word	16	0 to 65,535
Signed word	16	–32,768 to + 32,767

Table 2.4: Data types

- Semiconductor **RAM** within a microprocessor system provides storage for the transient data and variables that are used by programs.
- Part of the RAM is also used by the microprocessor as a temporary store for data while carrying out its normal processing tasks.
- In **CMOS RAM**, data is kept alive using a small battery. This battery backed memory is used to retain important data such as time and date.
- The data stored in the memory device is represented in kilobytes (KB). 1 Kilobyte = 1024 bytes. (Nearest power of 2 i.e. 2^{10} =1024)
- Capacity of a semiconductor ROM is specified in terms of an address range and number of bits stored at each address. For e.g., 2Kx8 bits (capacity 2 Kbytes), 4Kx8 bits & so on.

2.5 Microcontroller system

- Microcontroller is a programmable device consisting of a central processing unit (CPU), memory, peripherals, and support circuitry on a single chip. Fig. 2.14 shows the arrangement of a typical microcontroller system.
- **Central Processing Unit (CPU):** CPU is the main component of the processor that contains Arithmetic Logic Unit (ALU) and Control Unit (CU). It is capable of performing simple arithmetic, logical and timing operations. It also communicates with devices like memory, input ports, output ports and clock unit.

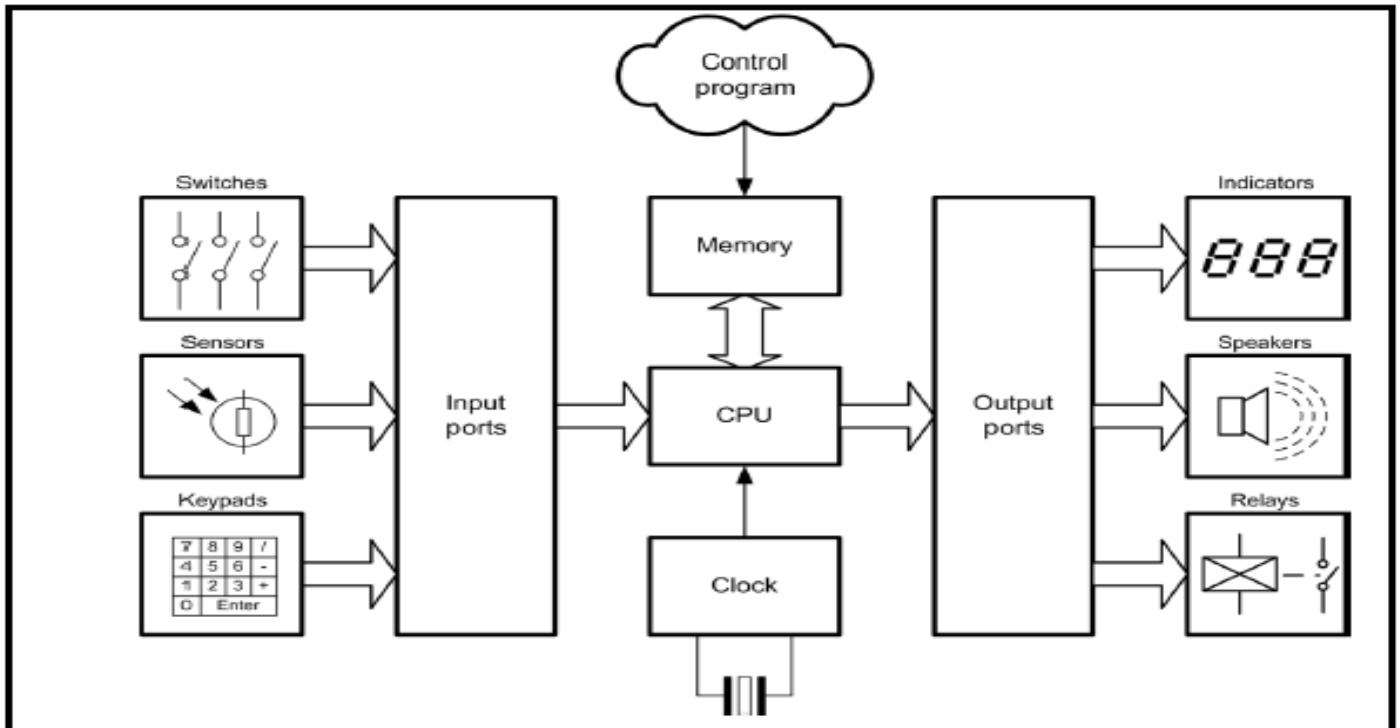


Fig. 2.14: A microcontroller system with typical inputs and outputs

- **Control Program:** The operation of the microcontroller is controlled by a sequence of software instruction known as control program. Control program operates continuously, examining inputs from input devices and output signals sent to controlled devices.
- **Input devices:** Input ports are mainly used to communicate with various input devices such as switches, sensors and keypads. Sensors convert physical quantities (temperature, position, etc.) into corresponding electrical signals. The microcontroller also accepts inputs from the user through keypads and switches.

  Some devices may sense analog quantities (such as velocity, temperature), but microcontroller input system receives digital input signals as, a 0 V = a logic 0 signal and a 5 V = a logic 1 signal. To represent analog quantities in terms of digital value Analog to Digital Converter (ADC) will be used as shown in fig. 2.15.
- **Output devices:** Output ports are mainly used to communicate with various output devices such as LED indicators, printers, speakers, relays, motors, etc. Output devices are used to convey information to the outside world.

  When analog quantities are required at the output of microcontroller rather than digital on/off operation, a digital-to-analog converter (DAC) will be used as shown in fig. 2.16.
- **Interface circuits:** When input and output signals are not logic compatible or outside the range of signals some interface circuitry (relays) may be required in order to shift the voltage levels or to provide higher current levels.

  For example, relays are used to operate a high power (AC 240V) loads like motors, heating systems, etc., with the help of low control signals (5V) from microcontroller.

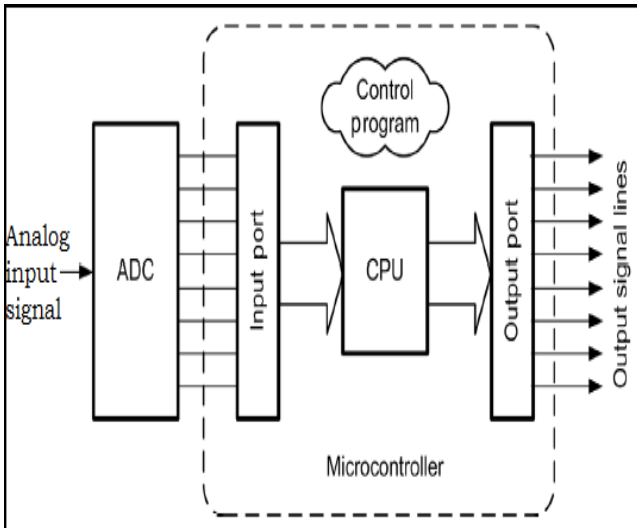


Fig. 2.15: An analogue input signal can be connected to a microcontroller input port via an analogue-to-digital converter (ADC)

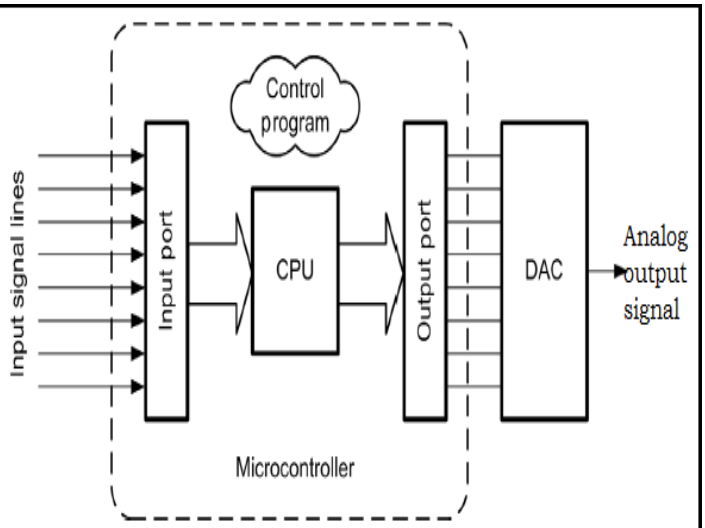


Fig. 2.16: An analogue output signal can be produced by connecting a digital-to-analogue converter (DAC) to a microcontroller output port

2.6 Shift Registers

- A set of N flip-flops (bistable devices) is called register. Each flip-flop stores one bit binary value (1 or 0).
- Two basic functions of registers are: Data storage and Data shift.
- A Shift Register is a sequential circuit used to store and transfer of binary data. It receives the data from its inputs and store, then shifts it to its output for every clock pulse, hence the name shift register.
- Shift Register is made of the number of individual Flip Flops.
- Shift registers operate in one of four different modes with the basic movement of data.
  - i) Serial in Serial out (SISO) Shift Register
  - ii) Serial in parallel out (SIPO) Shift Register
  - iii) Parallel in Serial out (PISO) Shift Register
  - iv) Parallel in Parallel out (PIPO) Shift Register

2.6.1 Serial in Serial out (SISO) Shift Register

- A 4 bit shift registers using JK flip-flop (bistables) is shown in figure 2.17.
- Pr (Preset) and CLR (Clear) are set and reset active low inputs.
- The flip-flops can be cleared by applying low input to CLR (CLR =0).
- During normal operation, CLR is made high and Pr is made low (CLR =1, Pr =1).
- In FF3, J and K inputs are connected through an inverter so that it acts as a D flip-flop through which the data inputs are entered.
- All the flip-flops are triggered by common clock pulses using CK input.

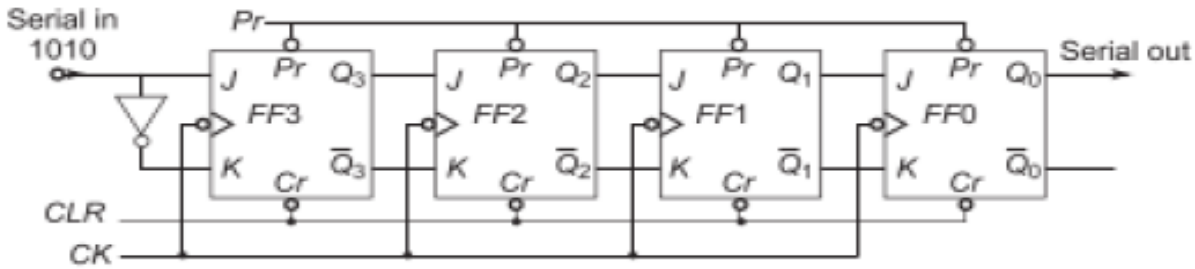


Fig. 2.17: 4-bit shift register

- Initially all the flip flops are cleared ($Q_3Q_2Q_1Q_0 = 0000$).
- During each clock pulse, one bit of input data is entered through FF3 and shifted from left to right (FF3 to FF0).
- The operation of SISO shift register for the input 1010 is shown in below table.

Table 2.5: Operation of Shift Register

Clock pulse	Serial in	Q_3	Q_2	Q_1	Q_0 (serial out)	
0	0	0	0	0	0	
1	1	1	0	0	0	
2	0	0	1	0	0	
3	1	1	0	1	0	Data entered
4	0	0	1	0	1	
5	0	0	0	1	0	
6	0	0	0	0	1	
7	0	0	0	0	0	1 Register cleared

2.6.2 All four type Shift register

- A 4-bit shift register which can operate in all the four modes can be constructed using JK flip-flop as shown in figure 2.18.

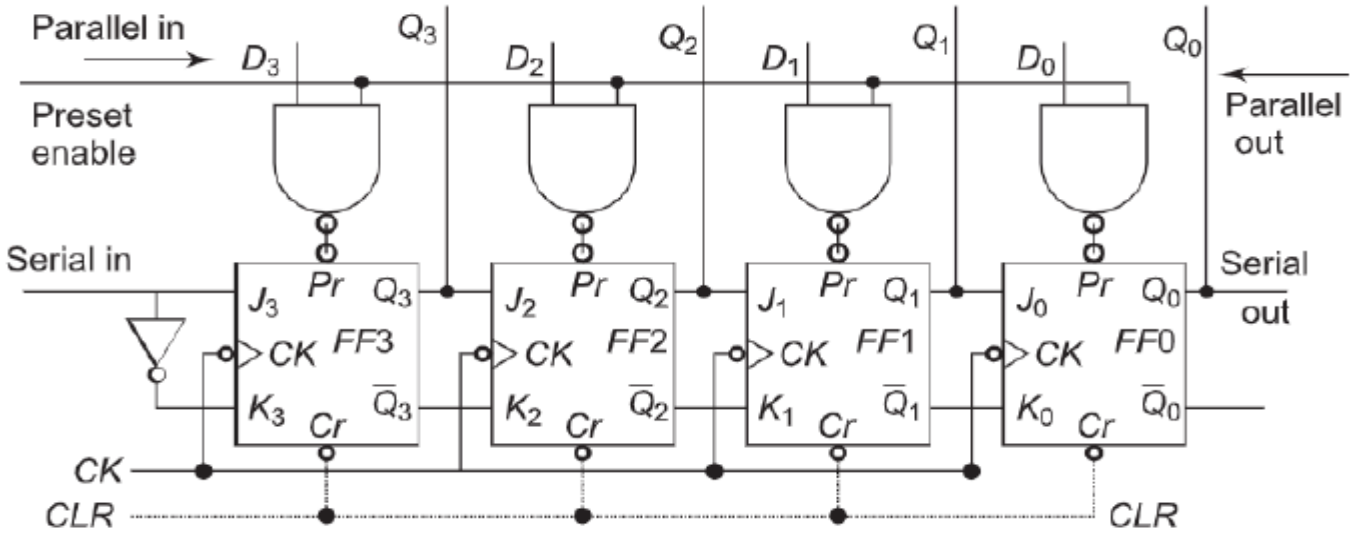


Fig. 2.18: 4-bit shift register

- **Serial in Serial out (SISO) Shift Register:** The serial input datas are entered into FF3 through the Serial in input. The data is shifted to right upon the application of clock pulses through CK input and the output is taken from Q_0 terminal of FF0.
- **Serial in Parallel out (SIPO) Shift Register:** Upon each clock pulse, the data is entered serially through FF3 and the parallel outputs are directly collected from terminals $Q_3Q_2Q_1Q_0$ of flip-flops FF3-FF0 as shown in figure 2.18.

- **Parallel in Parallel Out (PIPO) & Parallel in Serial out (SIPO) Shift Register:** The register is cleared by applying CLR=1 and Preset enable is made HIGH. This results in all the NAND gates to act as an inverter.
  - Any value on D_i input causes below results:
    If $D_i = 0$ then Pr = 1, implies that $Q_i = 0$.
    If $D_i = 1$ then Pr = 0, implies that $Q_i = 1$.
  - For PIPO operation, the data is entered into register through Pr inputs using $D_3D_2D_1D_0$ inputs and outputs are available directly on the terminals $Q_3Q_2Q_1Q_0$.
  - For PISO operation, the data is entered into register through Pr inputs using $D_3D_2D_1D_0$ inputs and datas can be shifted by applying the clock pulses and the output is taken from Q_0 terminal of FF0.

2.6.3 Left Shift- Shift Register

- Input data can be shifted from left (FF0) to right (FF3) by one bit on the occurrence of clock pulse by feeding output of FF0 (Q0) to input of FF1 (D1), and FF1 to FF2, FF2 to FF3. This logic is shown in figure 2.19.

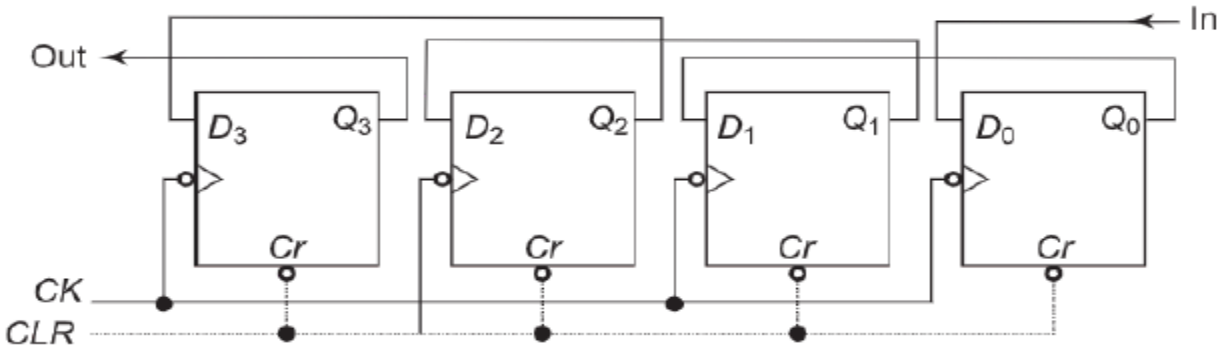


Fig. 2.19: Register shift left

2.7 Counters

- A counter is a sequential circuit that counts the number of occurrences of an input pulses and are primarily constituted of flip-flops.
- Each count is called as state of the counter. Thus, a counter counting in terms of n bits has 2^n different states.
- The number of different states of a counter is known as modulus of the counter. Thus, an n bit counter is called as a modulo 2^n counter.
- Counters are used in a lot of applications such as frequency counters, digital clocks, ADC, frequency divider circuits and timers.

- Depending on the manner in which the flip-flops are triggered, counters can be divided into two major categories:
  - **Asynchronous (ripple) counters:** In this type, all the flip flops are not clocked simultaneously. Only the first flip-flop is externally clocked using clock pulse while the remaining flip-flops are clocked by output of previous flip-flops.
  - **Synchronous counters:** In this type, all the flip flops are clocked simultaneously using an external clock pulse.

2.7.1 Asynchronous (ripple) counters

- Figure 2.20 shows a 3-bit (modulo 8) asynchronous counter using JK flip-flops.
- The J and K inputs of all the flip-flops are tied together and logic 1 signal is applied, to achieve toggling at the negative transition of the clock input.
- The external clock input is applied to first flip-flop FF0 only while the remaining flip-flops are clocked by Q output of previous flip-flops.

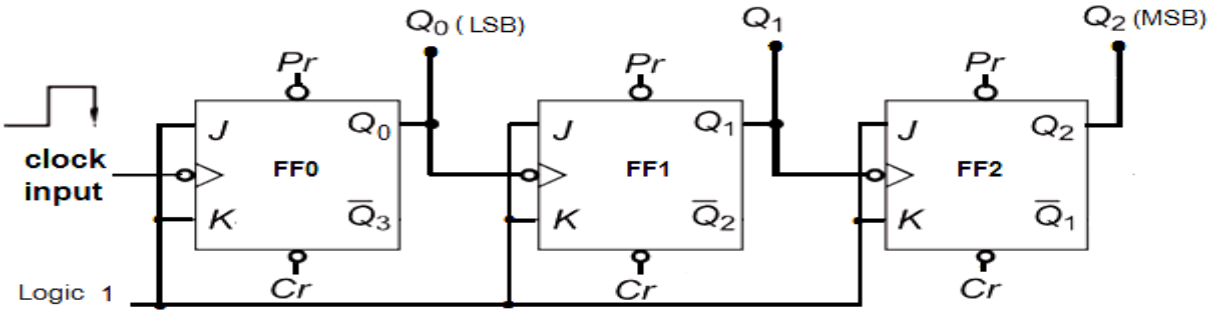


Fig. 2.20: 3-bit Asynchronous Counter

- **Operation**:
  - All the flip flops are initially cleared.
  - The clock input is then applied.
  - With clock pulse negative transition (1 to 0), Q_0 toggles.
  - For each negative transition of Q_0, Q_1 toggles.
  - Similarly, for each negative transition of Q_1, Q_2 toggles.
  - The transition and count are shown in table 2.6.
  - Thus the counter counts from 0 to 7 (000 – 111) and the cycle repeats.
  - As the pulses get applied to flip-flops in sequence, it is also called a Ripple counter.
- **Down counter:** Qs^{-} outputs are connected to the next flip-flop. As Q^{-} toggles from 1 to 0, the corresponding Q toggles 1 to 0, which is count down.
- **Up-down counter:** CK is connected to Q for up-down and Q through AND-OR logic.

Table 2.6: Operation of Asynchronous counter

Input (Clock)	Q_2	Q_1	Q_0
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