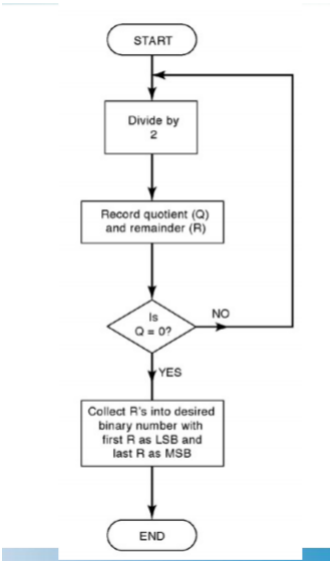
DIGITAL SYSTEM

1 Digital and Analog Systems

- **Digital System** is a combination of devices that manipulate values represented in digital form.
- **Analog System** is a combination of devices that manipulate values represented in analog form.

Analog-to-digital conversion(ADC) and digital-to-analog conversion(DAC) complicate circuitry



2 Digital Number Systems

- Number system differ in the number of symbols they use

- Decimal - 10 symbols (base 10)
- Hexadecimal - 16 symbols (base 16)
- Octal - 8 symbols (base 8)
- Binary - 2 symbols (base 2)

3 Conversion

Binary to Decimal Conversion

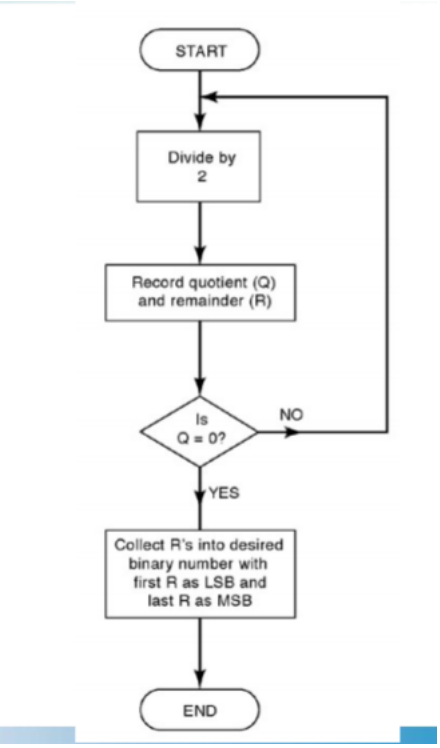
$1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1_2$
$5 \quad 4 \quad 3 \quad 2 \quad 1 \quad 0$ We consider that:
$(100101)_2 = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (37)_{10}$

Decimal to Binary Conversion

$\frac{41}{2} = 20 \quad a_0 = 1 \qquad \frac{5}{2} = 2 \quad a_3 = 1$
$\frac{20}{2} = 10 \quad a_1 = 0 \qquad \frac{2}{2} = 1 \quad a_4 = 0$
$\frac{10}{2} = 5 \quad a_2 = 0 \qquad \frac{1}{2} = 0 \quad a_5 = 1$

$(41)_{10} = (a_5 a_4 a_3 a_2 a_1 a_0)_2 = (101001)_2$

Flow chart:



Convert from Hexadecimal to Decimal

- Convert from Hexadecimal to decimal by multiplying each hex digit by its positional weight

$$\begin{aligned} 163_{16} &= 1\times(16^2)+6\times(16^1)+3\times(16^0) \\ &= 1\times 256+6\times 16+3\times 1 \\ &= 355_{10} \end{aligned}$$

Hexadecimal to Decimal Conversion

- Convert form Hexadecimal to Decimal by deviding the decimal number by 16
Same method to convertion from binary to decimal

Hexadecimal to Binary Conversion

$9F2_{16}$ = 9 F 2
1001 1111 0010 =

100111110010_2

Binary to Hexadecimal Convertion

- To convert Binary to Hexadecimal, first we should group bits in four, then we convert each group to hexadecimal.
Ex:
$1110100110_2 = 0011\ 1010\ 0110$
$= 3 \quad A \quad 6$
$= 3A6_{16}$

4 BCD

- **Binary Coded Decimal** (BCD) is a another way to present decimal numbers in binary form.

- BCD is widely uses and combines features of both decimal and binary systems.
- Each digit is converted to a binary equivalent.

Bảng 1: BCD Table

Decimal	**BCD**
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

- To convert the number 874_{10} to BCD:

  8 7 4
  1000 0111 0100 = 100001110100_{BCD}

- Each decimal digit is represented using 4 bits
- Each 4-bit group can never be greater than 9.
- Reverse the process to convert BCD to decimal.

5 Combinational Logic circuit and Sequential Logic circuit

1. Combinational Logic Circuit:

- Combinational Logic Circuit is the circuit made up from the basic **logic** gate (AND, OR, NOT) or universal gates (NAND, NOR).
- These gates are **combined** together to produced more complicated circuits.
- Output of the Combinational Circuit depend totally in the current input.

2. Sequential Digital Circuit:

- The inputs of Sequential Circuit also include the previous output from the circuit.
- In Sequential Circuit, memory elements is used to store the previous value of the last output of the circuit.
- Output depend on both circuit state and current input.

6 Question 2:

1. How can you declare a negative triggered edge in sensitive list ?

```
1                   module Edge_trigger
2                   always @(negedge clk) begin
3                           //Statement:
4                   end
```

2. In the example 6, assume that out1 = 5 and out2 = 0, in = 1. What is the value of out1 and out2 in the clock trigger? Explain your results?

- The value of out1 is 0 and out2 is 5.
- Because when initial rst = 0, when we get the out1 and out2, rst = 1. When trigger the clock trigger, out1 and out2 will equal 0 and the circuit will be performed in sensitive list, in = 1 will perform the **else if** statement the later out1 is assigned to initial out2 and the later out2 is assigned to the initial out1.

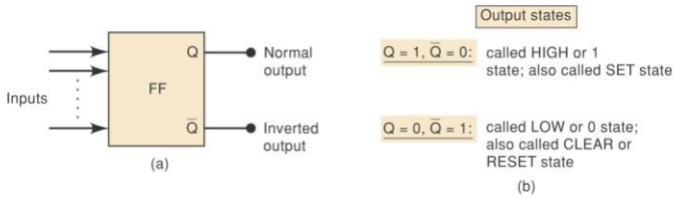
7 Flip Flop

- So far we has seen Combinational Logic that the output(s) depends only on the value of the input variables.
- Here we look at Sequential Logic circuit that the output(s) can depend on present and also past values of the input and output variables.
- Sequential circuits exist only in one of a defined number of state at any one time.

7.1 Synchronous and Asynchronous Sequence Logic:

- Synchronous:
  - The timing of all state transitions is controlled by a common clock.
  - Changes in all variables occur simultaneously.
- Asynchronous:
  - State transistions occur independently of any clock and normally dependent on the timing of transitions in the input variables.
  - Changes in more than one output do not neccessarily occur simultaneously.
- Clock:
  - A clock signal is a square wave of fixed frequency.
  - Often, transitions will occur on one of the clock pulses

7.2 General Flip Flop Symbol and definition of its two possible output states



- The Flip Flop, abbreviated FF, is a key memory element.
- The output of the Flip Flop is Q and $\bar{Q}$.
- Q is understood to be the normal output, $\bar{Q}$ is always the opposite.
- The most basic circuit can be constructed from either two NAND gates or NOR gates.

7.3 NAND gate latch

- The inputs are **set** and **clear** (reset).
- The inputs are active low, that is, the output will change when the input is pulsed low.
- When the latch is set:

$$\text{Q} = 1 \text{ and } \bar{Q} = 0$$

- When the latch is clear or reset:

$$\text{Q} = 0 \text{ and } \bar{Q} = 1$$

Summary of NAND latch

- SET = RESET = 1: Normal resting state, outputs remain in state prior to output.

- SET = 0, RESET = 1: Q will go high and remain high even if the SET input goes high.
- SET = 1, RESET = 0: Q will go low and remain low even if the SET input goes high.
- SET = RESET = 0: Output is unpredictable because the latch is being set and reset at the same time.

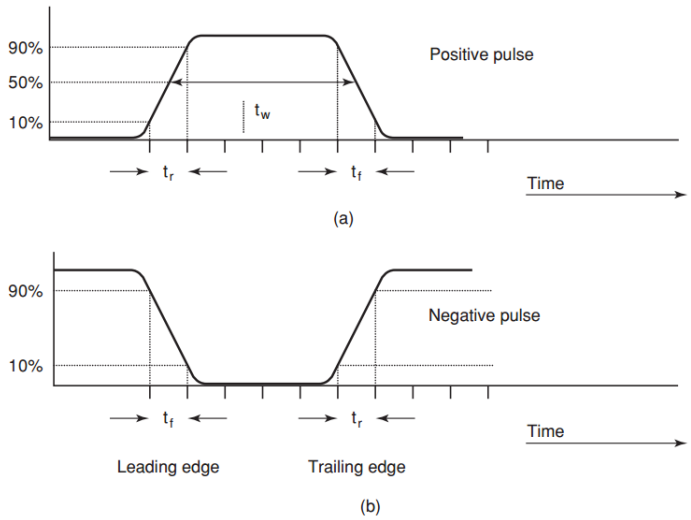
7.4 NOR Gate Latch

- The NOR Gate Latch is similar to the NAND Gate Latch, except that the output Q and $\bar{Q}$ is reserved.
- **The inputs are active high**, that is, the outputs will change only the inputs **are pulsed high**.
- In order to ensure that a FF begins operation at a known level, a pulse may be applied to SET and RESET inputs when a device is powered up.

7.5 Digital Pulses

- In digital system, when a signal switches from a normal inactive state to the oposite (active) state, thus causing something happen in the circuit. Then the signal returns to its inactive state while the effect of the recently activated signal remains in the system. These signals are called pulses.

FIGURE 5-14 (a) A positive pulse and (b) a negative pulse.



- A pulse that performs its intended function when it goes HIGH is called a positive pulse.
- A pulse that performs its intended funtion when it goes LOW is called a negative pulse.
- The transition from low to high on a positive pulse is called rise time (t_r).
- The transition from high to low on a positive pulse is called fall time (t_f).
- The transition from the beginning of the pulse is called the leading edge and the transition at the end of the pulse is called the trailing edge.

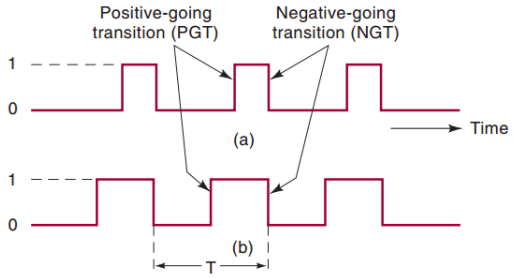
7.6 Clock Signal and CLocked Flip Flop

- Digital system can be operate either asynchronously and synchoronously:

- In asynchronous system, the outputs of circuits can change state anytime one or more of the inputs change.
- In synchronous system, the outputs can change states at the exact time by a signal called clock. This clock signal is generally a rectangular pulse train or a square wave.

Clock Signal

- Most (if not all) of the system outputs can change state only when the clock make a transition.



- A transition (also called *edges*):

- When the clock changes from 0 to 1, this is called the **positive-going transition (PGT)**.
- When the clock changes from 1 to 0, this called the **negative-going transition (NGT)**.

- Most digital system are pricipally synchronous (although there are always some asynchronous parts) because synchronous circuits are easier to design and troubleshoot.

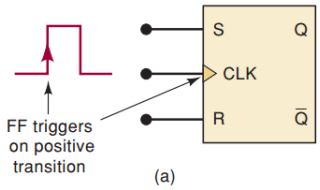
Clocked Flip Flop

- CLocked FFs can change state on one or other clock transitions. Some common characteristics:
  - CLock inputs are labeled as CLK, CK, or CP.
  - A **small triangle** at the CLK input indicates that CLK input is activated with a PGT (positive-going transition).
  - A **bubble and a triangle** at the CLK input indicates that CLK input is activated with a NGT (negative-going transition).
  - Control inputs have an effect on the output only at the active clock transition (PGT and NGT). There are also called synchronous control inputs.
  - The control inputs get the FF outputs ready to change, but the change is not triggered until the CLK edge.

Setup Time and Hold Time
- Setup time (t_s) is the minimum time interval before the active CLK transition that the control input must be kept at the proper level.
- Hold time (t_h) is the time after the active CLK transition during which the control input must be kept at the proper level.

7.7 Clocked S-R Flip Flop

- In Clocked S-R Flip Flop system, FF only change states when when a signal applied to its clock input makes a transition from 0 to 1 or from 1 to 0 (PGT and NGT).
- The up arrow (↑) indicates that a PGT is required at CLK; the label Q_0 indicates the level at Q prior to the PGT.



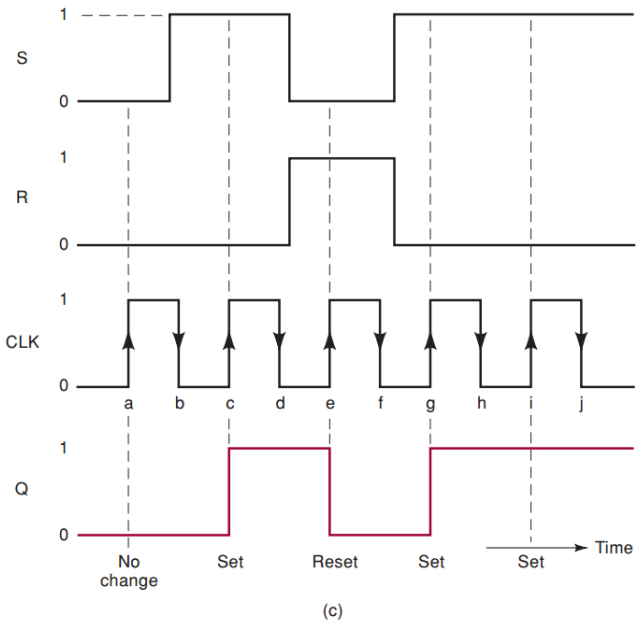
(a)

Inputs			Output
S	R	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguous

Q_0 is output level prior to ↑ of CLK.
↓ of CLK produces no change in Q.

(b)

- We can analyze these waveforms as follows:



1. Initiallly all inputs are 0 and the Q output is assumed to be 0; that is $Q_0 = 0$
2. When the PGT of the first clock pulse occurs (point a), the S and R inputs are both 0, so the FF is not effected and remains in the Q = 0 state (Q = Q_0)
3. When the PGT of the second clock pulse (point c), the S = 1 and R = 0. Thus the FF set to 1 state at the rising edge of this clock pulse.
4. When the third clock pulse makes its positive transition (point e) S = 0 and R = 1. Thus it causes the FF set to 0 state.
5. At the fourh pulse set the FF once again to the Q = 1 state (point g) because S = 1 and R = 0 when PGT occurs
6. The fifth pulse also find that S = 1 and R = 0 at PGT occurs so the FF remains state.
7. The case that S = R = 1 should not be used because it results in an ambiguous condition.

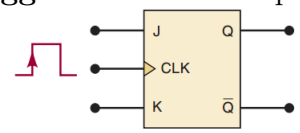
8. It should be noted that those waveforms that the FF is **not effected** by the NGT of the clock pulses. Also note that the S and R levels have no effects on the FF, except upon the occurence of the PGT **in PGT transitions**

9. For the NGT transition the FF operate at the same manner at PGT transition, except the output can change states only on the falling edge of the clock pulse

- As a general rule for stable flip flop triggering, the clock pulse rise and fall time must be very short, around typically 2 - 5 ns.

7.8 Clocked J-K Flip Flop

- The J and K inputs control the state of the FF in the same way as the S and R inputs do for the clocked S-R flip flop, except 1 difference that when J = K = 1 condition **does not result** in ambiguous output. In this condition, the FF will always go to its opposite side of the clock signal, this is called **toggle mode** of the operation.



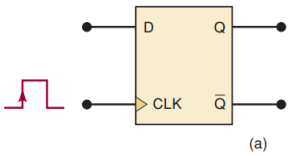
(a)

J	K	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	$\overline{Q_0}$ (toggles)

- The operation of the J-K Clocked Flip Flop is the same as the S-R Clocked Flip Flop , except when J = K = 1 at the PGT transition the output will change to the opposite state.
- Also in NGT, the operation is still same as PGT when clock is triggered in NGT.
- In edge-triggered J-K Flip Flop, CLK* must be high for FF to change states. This condition only occurs at the edge of a CLK transition

7.9 Clocked D Flip FLop

- Unlike the S-R and J-K Clocked FLip Flop, this flip flop has only one synchronous control input, *D*, which stands for data.
- The operation of the D CLocked Flip Flop is very simple: Q will go as the same state on the D input when a PGT occurs at CLK.
- A negative-edge-triggered D flip flop operates in the same way and Q will take on the value of D when a NGT occurs at CLK.



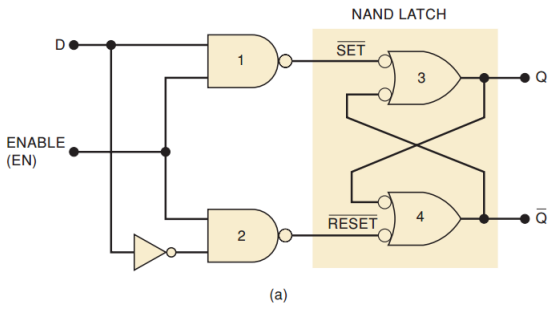
(a)

D	CLK	Q
0	↑	0
1	↑	1

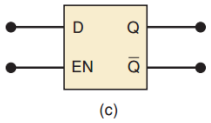
7.10 D Latch (Transparent Latch)

- The edge trigggered D Flip Flop uses an edge-detector circuit to ensure that the output will repond to the D input only when the active transition of the clock occurs. The latch that its edge-detector is not used called the D latch.

- One data input.
- The clock has been replaced by an ENABLE (EN).
- The device is not edge-triggered.
- When EN = 1; the output follows the change of input.
- When EN = 0; wheither D changes or not Q will stay at their current level (no change).

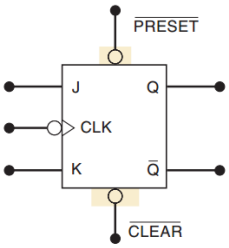


- Logic symbol of D latch:



7.11 Asynchronous Inputs

- For the clocked flip flop, the S-R, J-K and D inputs have been reffered as **synchronous inputs**
- Most clocked FF also have one or more asynchronous inputs that operate independently of the synchronous inputs.
- These asynchronous inputs can be used to set FF to the 1 state of clear (reset) and FF to 0 state at **any time, regardless conditions of inputs**.
- State in another way, the asynchronous inputs are **overide inputs**



J	K	Clk	PRE	CLR	Q
0	0	↓	1	1	Q (no change)
0	1	↓	1	1	0 (Synch reset)
1	0	↓	1	1	1 (Synch set)
1	1	↓	1	1	Q̄ (Synch toggle)
x	x	x	1	1	Q (no change)
x	x	x	1	0	0 (asynch clear)
x	x	x	0	1	1 (asynch preset)
x	x	x	0	0	(Invalid)

- $\overline{PRESET} = \overline{CLEAR} = 1$: The asynchronous inputs are inactive and the output depend only on CLK transitions.
- $\overline{PRESET} = 0; \overline{CLEAR} = 1$: The $\overline{PRESET}$ is activated and Q = 1 (no matter what conditions). The CLK inputs cannot affect the FF while $\overline{PRESET} = 0$.
- $\overline{PRESET} = 1; \overline{CLEAR} = 0$: The $\overline{CLEAR}$ is activated and Q = 0 (no matter what conditions). The CLK inputs cannot affect the FF while $\overline{CLEAR} = 0$.
- $\overline{PRESET} = \overline{CLEAR} = 0$: Ambiguous response.

8 Flip Flop Timing Consideration

8.1 Setup and Hold Times

- Setup time and Hold time represent for reliable FF triggering. In IC's datasheet will always spectify the minimum values of t_s and t_h

8.2 Propagation Delays

- Whenever a signal is to change the state of a FF's output, there is delay from the time the signal is applied to the time when the output makes its change.
- The delay is measured between 50 percent points on the input and output waveform. - Modern IC flip-flops have propagation delay that ranges from a few ns to 100 ns.

8.3 Maximum Clocking Frequency (f_{Max})

- This is the highest frequency that may be applied to the CLK input of a FF and still have it triggerreliably.

8.4 CLock Pusle HIGH and LOW time

- Clock Pulse HIGH is the time when the CLK is kept at HIGH before goes LOW.
- CLock Pulse LOW is the time when the CLK is kept at LOW before goes HIGH.

8.5 Asynchronous active pulse width

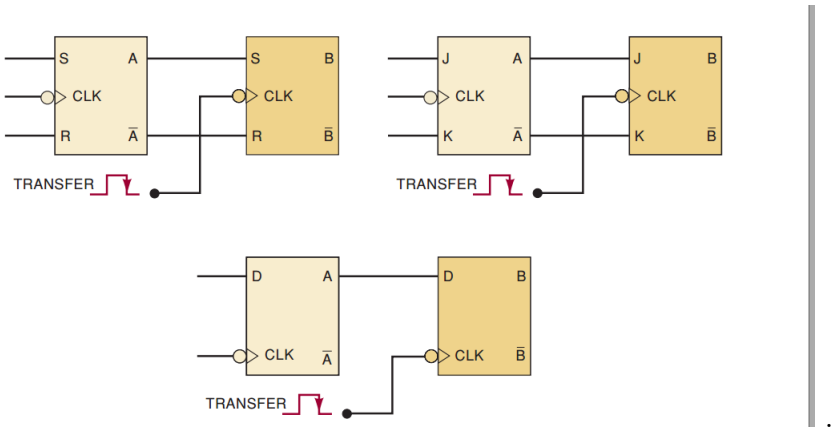
- The minimum time that PRESET or CLEAR must be held for the FF to set of clear reliably.

8.6 Clock Transition Time

- For reliable triggering, the clock waveform transition times (rise and fall time) should be kept be very short. If the clock signal takes too long to make the transition from 1 level to another, the FF will trigger eratically. - The transition time should be $\leqslant$ 50 ns for TTL devices and $\leqslant$ 200 ns for CMOS

9 Potential Timing Problem in FF circuit

- When the output of one FF is connected to the input of another FF and both devices are triggered by the same clock, there is a potential timing problem.
- Propagation time delay may cause unpredictable outputs. - The problem can be reduced due to low hold time.



10 Flip Flop Synchronization

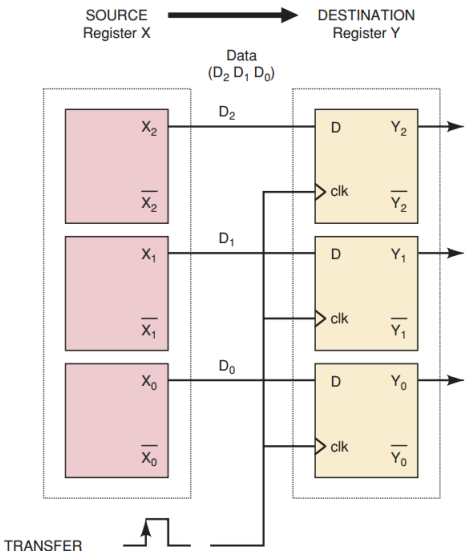
- Most system are primarily synchronous in their operation most of the signals will change states in synchronism with the clock transition.
- The randomness by asynchronous can produce unpredictable result.

11 Data Storage and Transfer

- Most common of the flip flop is for storage and information.
- The data is stored in groups is called **register**.
- The operation most often performed in data that is stored in a FF or a register is the **data transfer** operation, the operation involves the transfer from one FF or register to another.

- For Asynchronous Data Transfer, in the figure, the inputs respond to LOW level. When the **Transfer enable** is LOW, the two NAND outputs is HIGH.
- When the **Transfer enable** is HIGH, one of two NAND outputs is LOW, depends on the states of A and $\overline{A}$. This LOW value (SET or RESET) will make FF B as same state with FF A.
- Asynchronous is also called as **jammed transfer**.

11.1 Paralled Date Transder



- The transfer from X register into Y register is a synchronous transfer.

- The transfer is called **parallel transfer** due to the contents of X_2, X_1 and X_0 and is transfer simultaneously into Y_2, Y_1 and Y_0.
- If a **serial data** transfer were performed, the content of X register would be transfered to Y register one bit at a time.

12 Serial data Transfer: Shift Registers

13 Arithmetic Operation

13.1 Binary Addition

- $0 + 0 = 0$
- $0 + 1 = 1$
- $1 + 1 = 0 + \text{carry } 1$
- $1 + 1 + 1 = 1 + \text{carry } 1$

- The carry 1 will be added to the next position, which is add to the left.

011 (3)	1001 (9)	11.011 (3.375)
+ 110 (6)	+ 1111 (15)	+ 10.110 (2.750)
1001 (9)	11000 (24)	110.001 (6.125)

- The carry 1 from the operation will be added to next position.

13.2 Represent Signed number

- By adding 2 numbers 0 and 1, we can represent the **sign** of the number which is called **sign bits**

- To represent the positive number, we add **0** before the MSB of the binary number.
- To represent the negative number, we add **1** before the MSB of the binary number.

- But that is the way to show the sign of the binary number. In computer do not normally use it because the circuit implementation is complex than

other systems.
Ex: $52_{10} = 0110100_2$, we cannot assume that $-52_{10} = 1110100_2$. The result is different.
- In this case, **2's complement system** is used to represent the signed number.

1. 1's complement:
   - First, we start with the 1's complement form:

   - The 1's complement of the binary number is obtained by changing each 0 to 1 and each 1 to 0.

     1 0 1 1 0 1 original binary number
     ↓↓↓↓↓↓
     0 1 0 0 1 0 complement each bit to form 1's complement
   -

2. 2's Complement Form:
   - The 2's complement form is formed by taking 1's complement and then adding 1 to the LSB of the binary number.

	1 0 1 1 0 1	binary equivalent of 45
	0 1 0 0 1 0	complement each bit to form 1's complement
+	1	add 1 to form 2's complement
	0 1 0 0 1 1	2's complement of original binary number

3. Represent the signed number by 2's Complement form:
   - This method can also use in unsigned number.

   - Take 1's complement of the binary number.
   - Add to LSB.

13.3 Sign Extension

- When extend the binary number to another more bits, we duplicate the sign bit.
Ex: We have 4 bits of 12_{10} is 1100, add **sign bit** 0 to MSB then we append 0 until we get the possible bits we want to extend.

13.4 Negation

- **Negation** is the operation of converting a positive number to its negative equivalent or a negative to its positive equivalent.

13.5 Special Cases in 2's Complement Representation

- Signed Number contains 1 as sign bits and 0 as its magnitudes.
- The complete range of signed number have N magnitude bits is:

$$-2^N \text{ to } +(2^N - 1)$$

→ There are a total 2^{N+1} different values, including zero.
Ex: Some Example:

What is the range of *unsigned* decimal values that can be represented in a byte?

Solution

Recall that a byte is eight bits. We are interested in unsigned numbers here, so there is no sign bit, and all of the eight bits are used for the magnitude. Therefore, the values will range from

$$00000000_2 = 0_{10}$$

to

$$11111111_2 = 255_{10}$$

This is a total of 256 different values, which we could have predicted because $2^8 = 256$.

What is the range of *signed* decimal values that can be represented in a byte?

Solution

Because the MSB is to be used as the sign bit, there are seven bits for the magnitude. The largest negative value is

$$10000000_2 = -2^7 = -128_{10}$$

The largest positive value is

$$01111111_2 = +2^7 - 1 = +127_{10}$$

Thus, the range is −128 to +127; this is a total of 256 different values, including zero. Alternatively, because there are seven magnitude bits ($N = 7$), then there are $2^{N+1} = 2^8 = 256$ different values.

13.6 Addition in the 2's Complemetn System

1. Two Positve Number:

```
+9 → 0 1001    (augend)
+4 → 0 0100    (addend)
     0 1101    (sum = +13)
     ↑
     └ sign bits
```

2. Positve and Negative smaller number:

```
       ┌ sign bits
       ↓
+9 → 0 1001    (augend)
−4 → 1 1100    (addend)
   1 0 0101
   ↑
   └ This carry is disregarded; the result is 00101 (sum = +5).
```

3. Positive and Negative larger number:

```
−9 →  10111
+4 →  00100
      11011    (sum = −5)
      ↑
      └ negative sign bit
```

4. Two Negative Number:

$-9 \rightarrow \quad 10111$
$-4 \rightarrow \quad 11100$
$\overline{\quad 1 \quad 10011}$

sign bit

This carry is disregarded; the result is 10011 (sum = −13).

13.7 Subtraction in 2's Complement System

- When subtracting one binary number (the subtrahend) from another binary number (the minuend) :

- Negate the subtrahend: This will change the subtrahend to its equivalent value of opposite sign.
- Add the result to minuhend

13.8 Arithmetic Overflow

- Arithmetic Overflow occurs when we add 2 positive or negative binary number which the result require more than bits and therefore **overflow** into the sign-bit position - When we perform the subtraction, the **overflow** occurs when the subtrahend is negated and become the addition berween 2 positive numbers.

13.9 Multiplication of Binary Number

1. Multiplication between 2 binary number: - To perform the multiplication between 2 binary number we perform the multiplication between multiplicand and the LSB of multiplier, the result is written down as the first partial product.
   - Then we come with the second number from thw left of the equation, the second result is written down as second partial product, the second

result should be **shifted** 1 bit to the left.
- Then we try with the third and fourth ... after that, perform the addtion with partial products to get the final result.

```
   1001      ← multiplicand = 9_10
   1011      ← multiplier = 11_10
  ------
   1001   }
  1001    }
 0000     }   partial products
1001      }
---------
1100011   }   final product = 99_10
```

2. Multiplication between 2's Complement Form:
   - When 2 numbers is negative in multiplication, we negate both 2 number then perform the equation.
   - When one number is positive and the other is negatve, we negate the negative one then perform the equation and the final result should be negated.

13.10 Binary Devision

```
   0011
11|1001
  011
  0011
   11
   0
```

- The figure below shows the division between 1001_2 and 11_2 and the quotient of the division is 0011_2.
- To perform the binary devision, follow the step:

- As we have the devisor is 11, first take the first number in the left to divide for 11. Consider 1001, we take 1 / 11, because 1 is $<$ 11 then the result is 0, write 0 as the result. Then continue with 10, 100 and 1001, if the dividend is larger than the devisor the result will come to 1.
- If the result is 1, perform the multiplication between 11 to 1 is 11, write 11 below 100 (dividend) and the perform the binary substraction

to get the remainder, after that get the last 1 from the dividend to the remainder, repeate the step

13.11 BCD Addition

1. Sum equals 9 or less:

```
  5      0101   ← BCD for 5
 +4    + 0100   ← BCD for 4
  9      1001   ← BCD for 9
```

```
  45      0100 0101   ← BCD for 45
 +33    + 0011 0011   ← BCD for 33
  78      0111 1000   ← BCD for 78
```

2. Sum greater than 9:
   - When peform the addition between BCDs that produce the sum that is greater than 9, we should add 6 bits to the sum ($6_{10} = 0110_2$) to get the result in 4 bits.

```
          0110   ← BCD for 6
        + 0111   ← BCD for 7
          1101   ← invalid sum
          0110   ← add 6 for correction
        ------
  0001    0011   ← BCD for 13
   1       3
```

   - Another example:

```
  47     0100    0111   ← BCD for 47
 +35   + 0011    0101   ← BCD for 35
  82     0111    1100   ← invalid sum in first digit
            1←┐  0110   ← add 6 to correct
         1000  └0010   ← correct BCD sum
           8      2
```

14 Hexedecimal Arthimetic

14.1 Hex Addition

- To perform the Hexadecimal Addition, follow the procedure:

- Add the 2 hex digit in decimal, mentally inserting the decimal equivalent for those digits larger than 9.
- If the sum is 15 or less, it can be directly expressed in hex digits(A, B, C, D, F).
- If the sum is larger than or equals 16, substract 16 and get carry 1 to the next position.

14.2 Hex Substraction

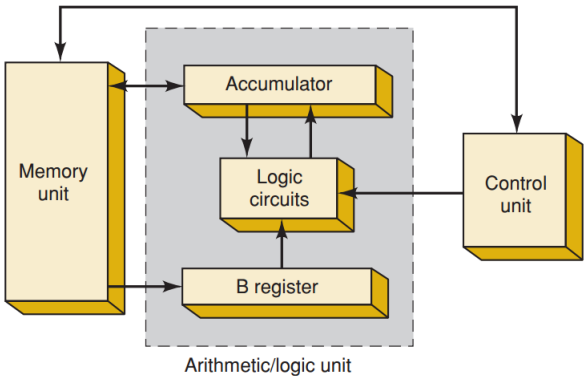
- The procedure to perform the hex substraction is similar to the binary substraction.
- To change the hex digit to its 2's complement form, we take FFF substracts to the hex number then add 1.
- After that take the addtion with the hex number in 2's complement form to get the result.

15 Arithmetic/Logic Unit (ALU)

- All arithmetic operation take place in the **Arthimetic/ Logic Unit** (ALU) of a computer.
- The arithmetic/logic unit contains at least 2 flip-flop register: the B register and the **accumulator register**

- The operation follow:

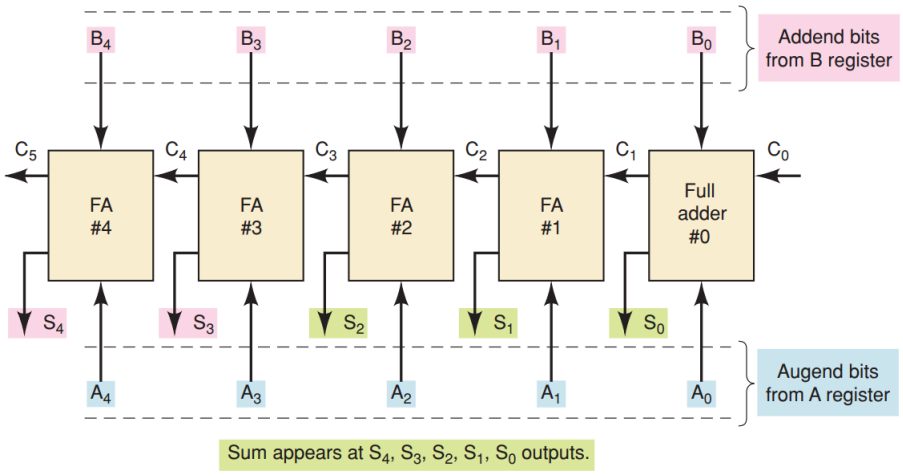
- The control units receives an instruction (from the memory unit) specify that a number stored in a particular memory location (address) is to be added to the number represently stored in the **accumulator register**.
- The number to be added is transfer from memory to the B register.
- The number in B register and the number in the **accumulator register** are added together in the logic circuit. The result is sent to the accumulator to be stored.
- The new number in the accumulator can remain there so that another number can be added to it or if the particular arithmetic process is finished, it can be transfer to memory for storage.



15.1 Parallel Binary Adder

- To perform the addition between two binary digits we perform the addition of each digit, the carry after the addition will be added to the next position.
- So that we can calculate the sum **parallel** with add carry to the next position. This is called **Parallel Binary Adder**.

- The whole process is performed in logic circuit called **full adder**.



16 Counter and Register

16.1 Asynchronous Counter

- In asynchronous Counter:

- The clock pulses are applied to the CLK input of flip-flop A.
- The output of the flip-flop A acts as the CLK input for flip-flop B, same with C and D.
- FF outputs D, C, B and A represents a 4 bits binary number, with D as MSB

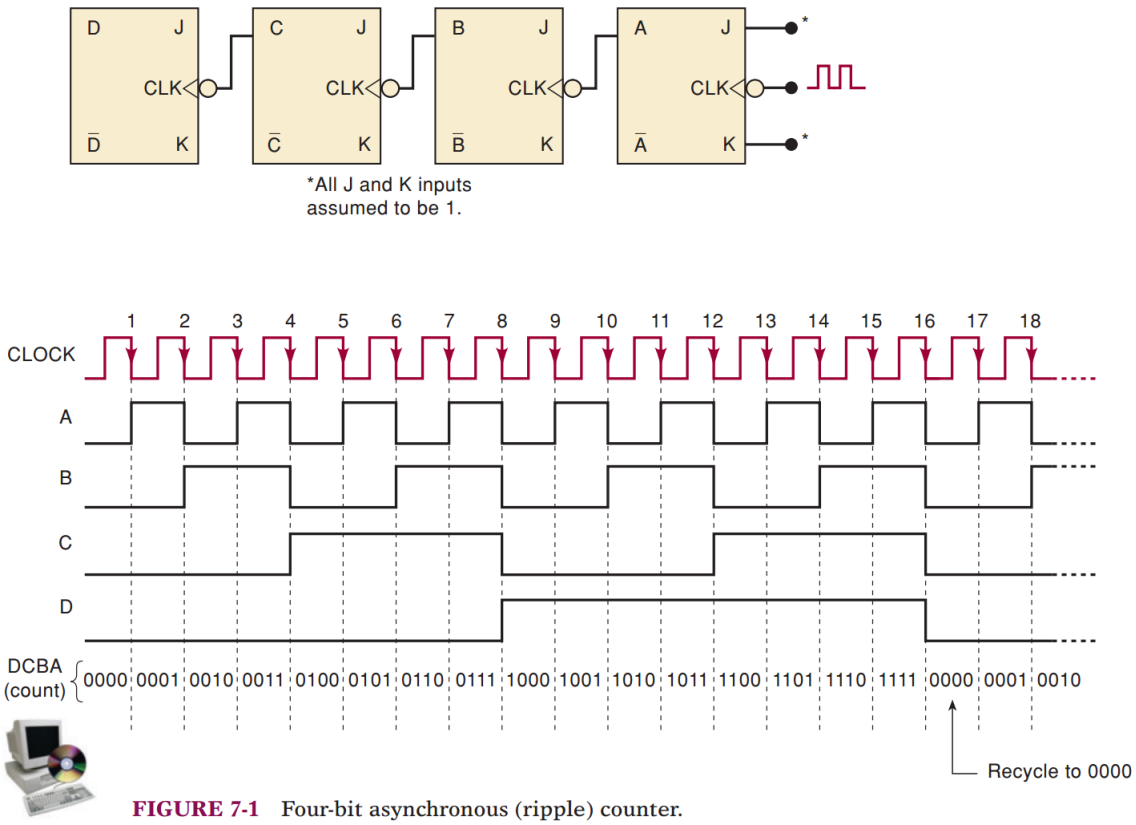


FIGURE 7-1 Four-bit asynchronous (ripple) counter.

16.2 Signal Flow

- In the counter circuit, CLK inputs of each flip flop are on the right, the ouputs are on the left, and the input clock input signal is shown comming to the right.

16.3 MOD Number

- **MOD** number is generally equal to the number of states that the counter goes though in each complete cycle before it rises back to its starting state.

$$\text{MOD number} = 2^n$$

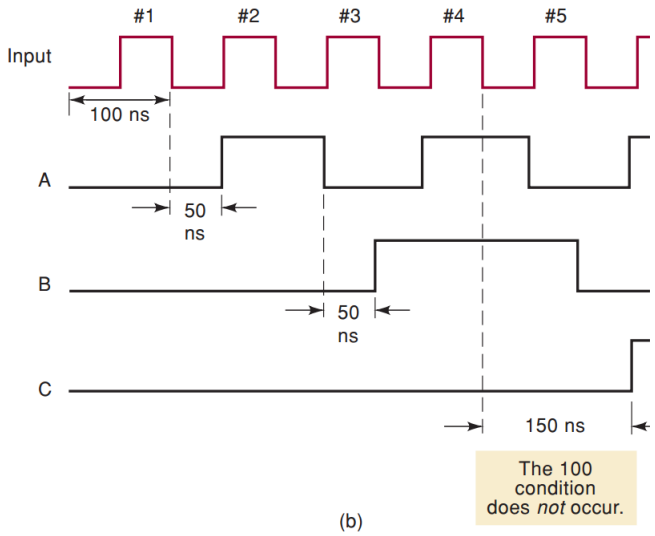
When n is the number of each FFs connected

16.4 Frequency Division

- In the basic counter, each FF provides an output waveform that is exactly the **half** frequency of the waveform at its CLK input.
- In any counter, the signal at the output of the last FF (MSB) will have the frequency equal to the input clock frequency divided by the **MOD** number of the counter.

16.5 Propagation Delay in the Ripple Counter

- Ripple counter is the simplest type of the binary counter because they require the fewest components to produce a given counting operation.
- Since each FF triggered by the transition at the output will have propagation delay time of each FF. That means the second FF will not respond until the first FF receives an active clock transition; the third FF will not respond until the 2 $\times$ propagation delay after that clock transition.



- For proper counter we need :

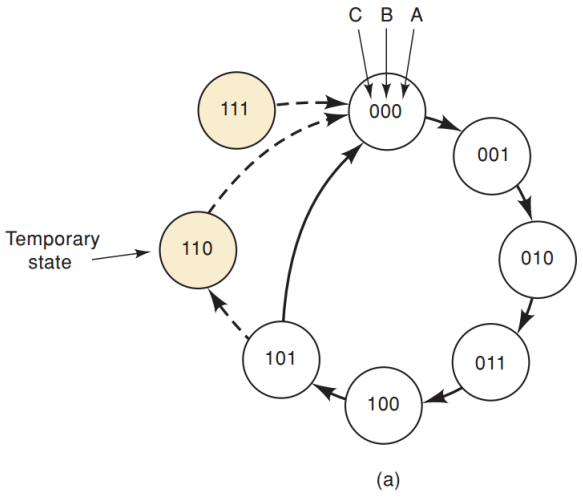
$$T_{clock} \geqslant N \times t_{pd}$$

Where N = number of FFs

- For the frequency of clock input:

$$f_max = \frac{1}{N \times t_{pd}}$$

16.6 Synchronous (PARALLEL) Counter



(a)

- Since the problem of propagation delay caused ripple counter that all FFs cannot cange simultaneously.
- Synchronous (Parallel Counter): The input pulses are applied to all the FFs.
- Some difference between Synchronous and Asynchronous Counter:

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

Circuit Operation - The A fip flop changes states at each NGT, its J and K are pernamently HIGH so that it will toggle at on each NGT of the clock input.

- The B flip flop changes states at A = 1 and C flip flop changes states at A = B = 1, so on with the flip flop D.

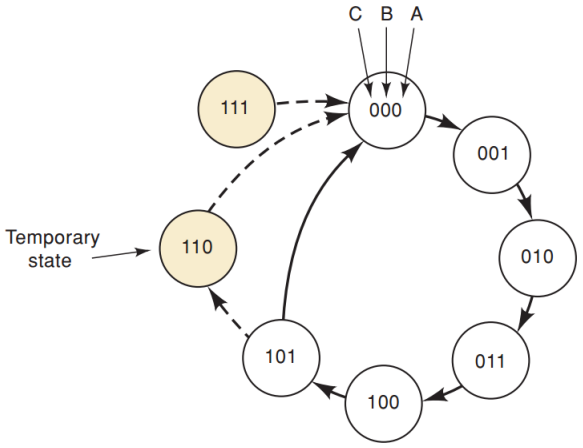
Each FF should have its J and K inputs connected so that they are HIGH only when the outputs of all lower-order FFs are in the HIGH state.

- The total delay of Synchronous Counter equals the time of one FF toggle + the time of the logic AND gate to reach the J, K inputs.

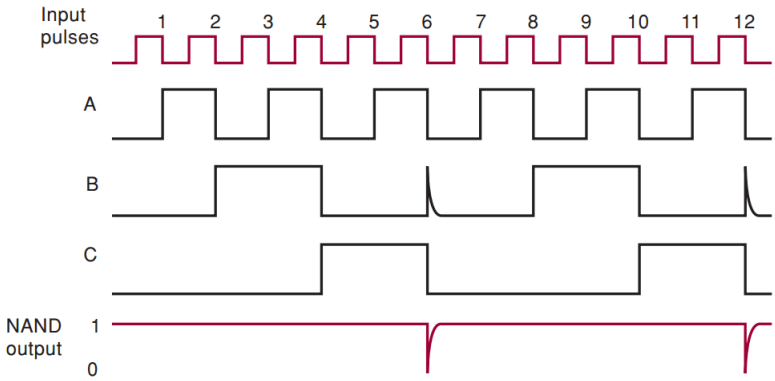
total delay = FF t_{pd} + AND gate t_{pd}

16.7 Counter with MOD number $\leqslant 2^n$

- The basic counter can be modified to produce MOD numbers less than 2^n by allowing the counter to **skip states** that are normally part of the counter sequence.



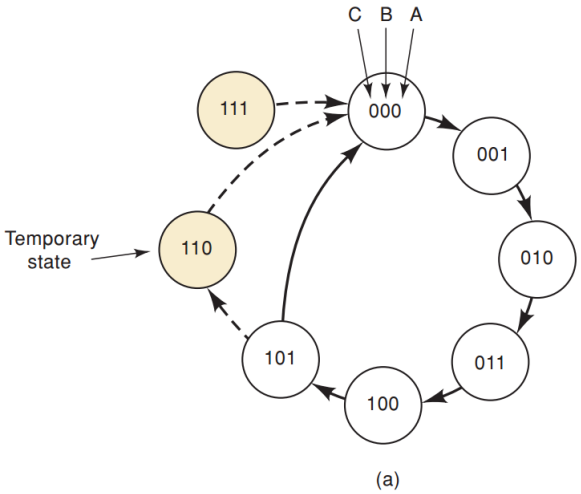
(a)



- The NAND output us connected to the asynchronous CLEAR inputs of each FF. When the NAND output is HIGH, it is no effect to the counter, but when it goes LOW it will active the CLEAR and the counter immediately goes to 0 state.
- The inputs of NAND gates are the outputs of B and C flip flops and so the NAND output will go LOW whenever the B and C equal 1.
- The counting sequence:

CBA	
000	←
001	
010	
011	
100	
101	
110	→ (temporary state needed to clear counter)

16.8 State Transition Diagram



16.9 Displaying Counter States

- We use individual indicator LEDs to have a visible display on how the counter changes for each FF output. Each FF output is connected to an INVERTER whose output provides the current path for LED.
- When the output A is HIGH, the INVERTER outputs goes LOW and the LED turns ON, the LED turning on indicates the A = 1.
- When output A is lOW, the INVERTER output is HIGH and the LED turns OFF, the led turning off indicates A = 0.

16.10 Decade Counter / BCD Counter

- A decade counter is any counter that has 10 distinct states, no matter what the sequence.
- BCD counter is the decade counter uses only 10 BCD code groups 0000,

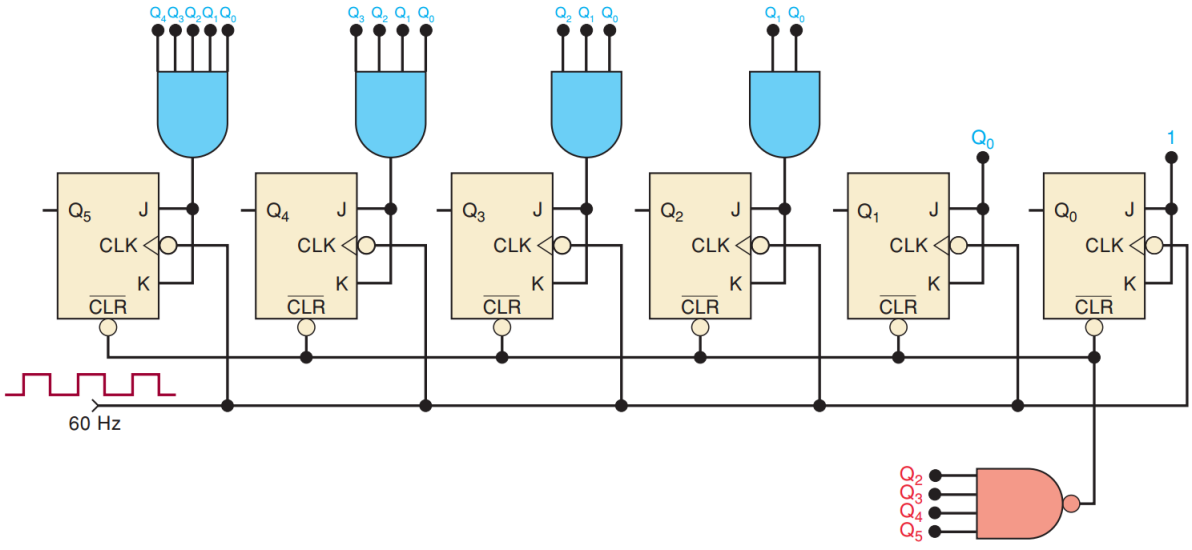
0001, ..., 1001.

Any MOD 10 counter is decade counters and any decade counters that count from 0000 to 1001 are BCD counter

In Example 7-3, a MOD-60 counter was needed to divide the 60-Hz line frequency down to 1 Hz. Construct an appropriate MOD-60 counter.

Solution

$2^5 = 32$ and $2^6 = 64$, and so we need six FFs, as shown in Figure 7-9. The counter is to be cleared when it reaches the count of 60 (111100). Thus, the outputs of flip-flops Q_5, Q_4, Q_3, and Q_2 must be connected to the NAND gate. The output of flip-flop Q_5 will have a frequency of 1 Hz.

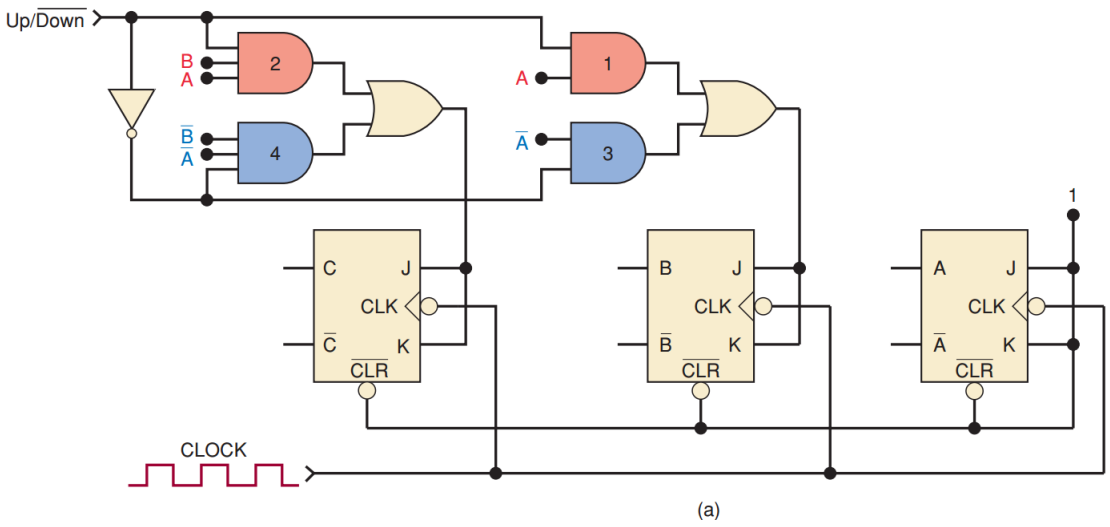


16.11 Synchornous Down and Up/Down Counter

- A synchronous up counter has the output of the FFs comes from the positive value.

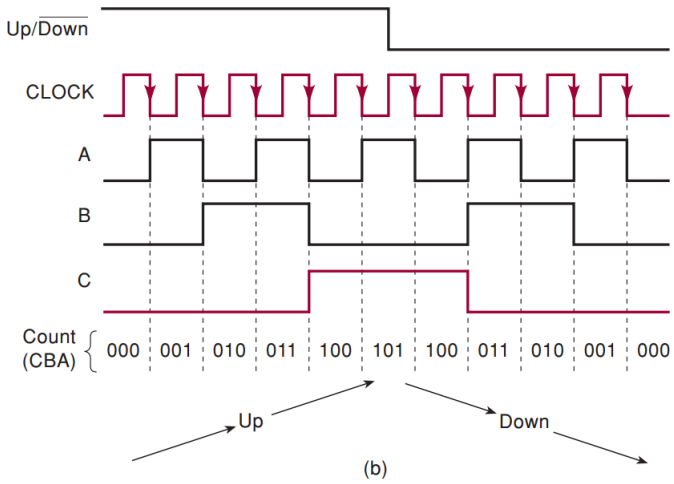
- A synchronous down counter has the output of the FFs comes from the negative value.

Up/Down Counter



(a)

- When Up/Down is held HIGH, AND gates 1 and 2 are enabled while AND gates 3 and 4 ar disabled. This allows the A and B outputs through gates 1 and 2 to the J, K inputs of FFs of B and C.
- When Up/Down is held LOW, AND gates 1 and 2 are diabled while AND gates 3 and 4 are enabled. This allows the inverted outputs of A and B through gates 3 and 4 into the J, K of FFs B and C.
- Up/Down = 1 and the counter counts up, Up/Down = 0 and the counter counts down.



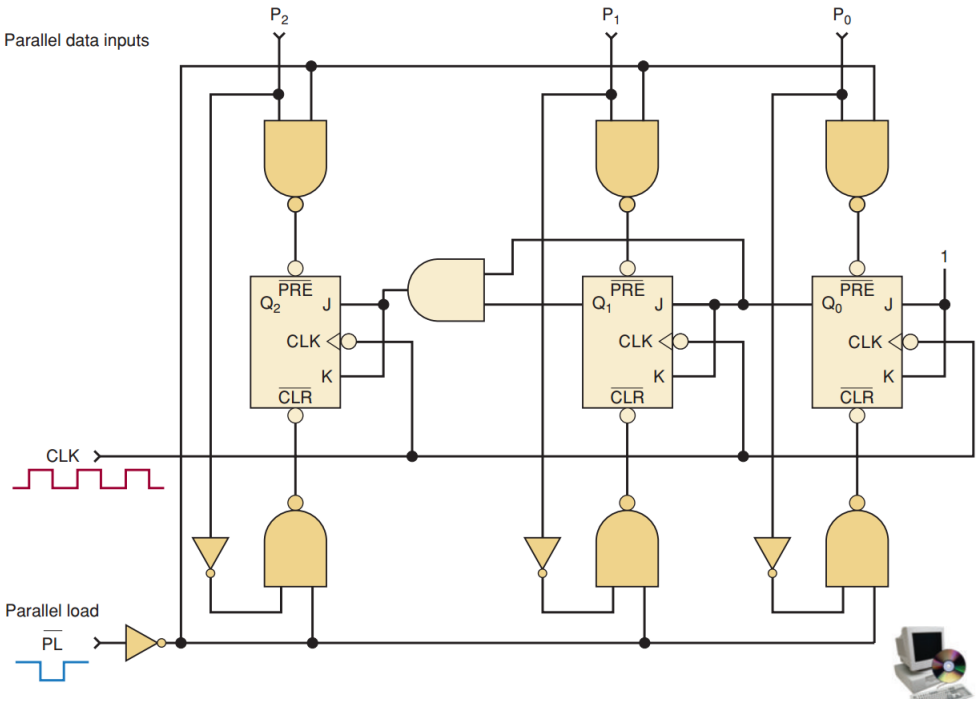
What problems might be caused if the Up/$\overline{\text{Down}}$ signal changes levels on the NGT of the clock?

Solution

The FFs might operate unpredictably because some of them would have their J and K inputs changing at about the same time that a NGT occurs at their *CLK* input. However, the effects of the change in the control signal must propagate through two gates before reaching the J, K inputs, so it is more likely that the FFs will respond predictably to the levels that are at J, K prior to the NGT of *CLK*.

16.12 Presetable Counter

- Presetable Counter is the counter that can be preset to any desired starting count either asynchronously or synchronously.
- This presetting operation is also reffered to as **parallel loading**.



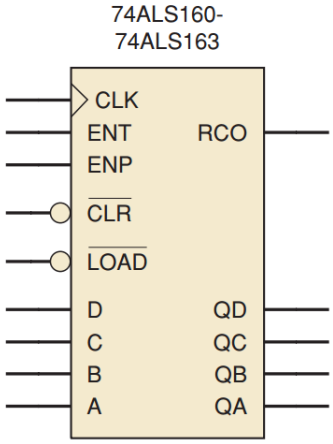
- Apply the desired count to the parallel data inputs (P_2, P_1, P_0).
- Apply a LOW pulse to the PARALLEL LOAD input, $\bar{PL}$
- This procedure will perform the asynchronous transfer to P_2, P_1, P_0 levels into flip-flops P_2, P_1, P_0. This **tram transfer** occurs independently of the J, K and CLK inputs.
- The effect of the CLK input will be disabled when $\bar{PL}$ is LOW because each FF will have one of its asynchronous inputs activated while $\bar{PL}$ = 0.
- Once $\bar{PL}$ returns HIGH, the FFs can respond to their CLK inputs and can resume the counting-up operation.
- This asynchronous presetting is used by several IC counters (the TTL 74ALS190, 74ALS191, 74ALS192, and 74ALS193 and the CMOS equivalents, 74HC190, 74HC191, 74HC192, and 74HC193.)

16.13 Synchronous Presetting

- Many IC parallel counters use synchronous presetting whereby the counter is preset on the active transition of the same clock signal that is used for counting.The logic level on the parallel load control input determines if the counter is preset with the applied input data at the next active clock transition.
- Examples of IC counters that use synchronous presetting include the TTL 74ALS160, 74ALS161, 74ALS162, and 74ALS163 and their CMOS equivalents, 74HC160, 74HC161, 74HC162, and 74HC163.

16.14 IC Synchronous Counters

The 74ALS160-163 / 74HC160-163 Series



(a)

Part Number	Modulus
74ALS160	10
74ALS161	16
74ALS162	10
74ALS163	16

(b)

74ALS160-74ALS163 Function Table

$\overline{CLR}$	$\overline{LOAD}$	ENP	ENT	CLK	Function	Part Numbers
L	X	X	X	X	Asynch. Clear	74ALS160 & 74ALS161
L	X	X	X	↑	Synchr. Clear	74ALS162 & 74ALS163
H	L	X	X	↑	Synchr. Load	All
H	H	H	H	↑	Count up	All
H	H	L	X	X	No change	All
H	H	X	L	X	No change	All

(c)

- In the 74ALS160-163 series:

- Two of counters are the MOD-10 counters (74ALS160 and 74ALS162), while the other two is MOD-16 counter (74ALS161 and 74ALS163).
- The operation of the CLEAR function:
  - The 74ALS160 and 74ALS161 has asynchronous clear input.
  - THE 74ALS162 and 74ALS163 has synchronous clear input.
- About the parallel loading of data into the counter's FFs:
  - To preset a data value, make the CLEAR input inactive (HIGH), apply the desired four-bit value to the data input pins D, C, B, A (A is LSB and D is MSB), apply LOW to the $\overline{LOAD}$ input control, and the CLK with PGT.
  - The LOAD function is synchronous and has priority over counting, so it doesn't matter the what logic levels are applied to ENT or ENP.
- To enable counting, the CLEAR and LOAD musu be at inactice states, ENT and ENP are both HIGH and if one of them are LOW then the counter will hold the current state (no change).
- When counting, the decade counters (74ALS160 and 74ALS162) will automatically recycle to 0000 after state 1001 (9) and the binary counters (74ALS161 and 74ALS163) will automatically recycle after 1111 (15).

- This series of IC counter chips has one more output pin, RCO. This function of active-HIGH output is to detect (decode) the last state of the counter.
  - ENT must be HIGH for the counter to indicate with the RCO output that it has reached its terminal state.

THe 74ALS190-191 / 74HC190-191 Series

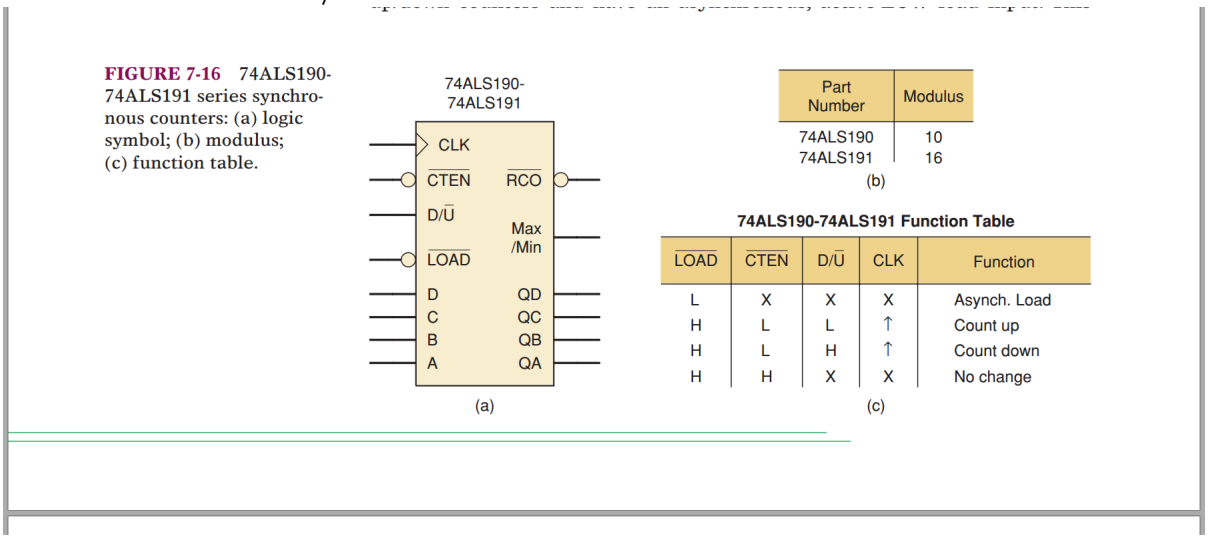
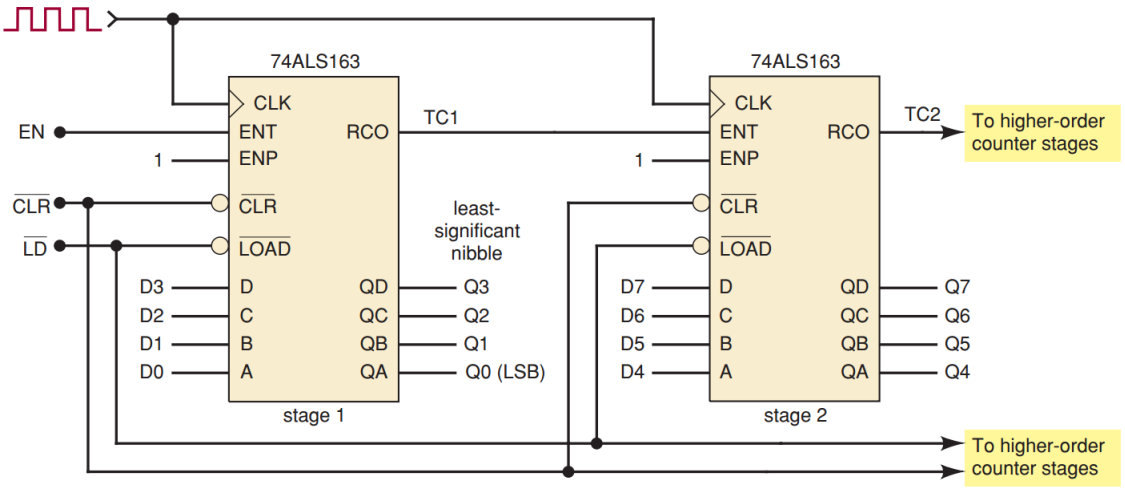


FIGURE 7-16 74ALS190-74ALS191 series synchronous counters: (a) logic symbol; (b) modulus; (c) function table.

16.15 Multistage Arrangment



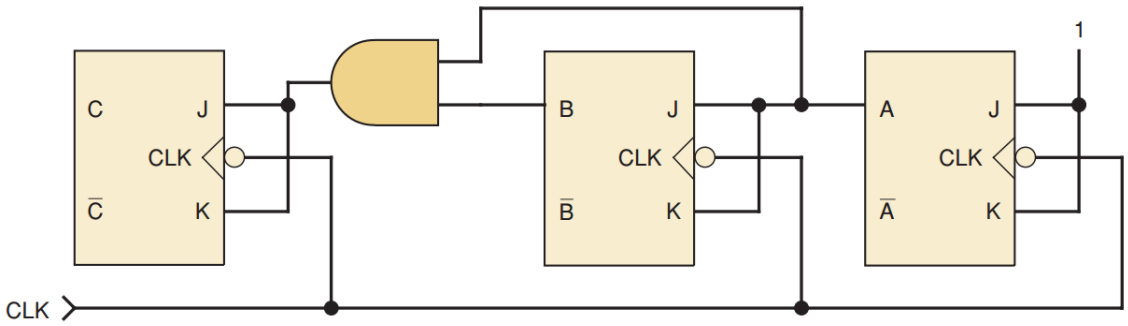
- The procedure of the Multistage Arrangment:

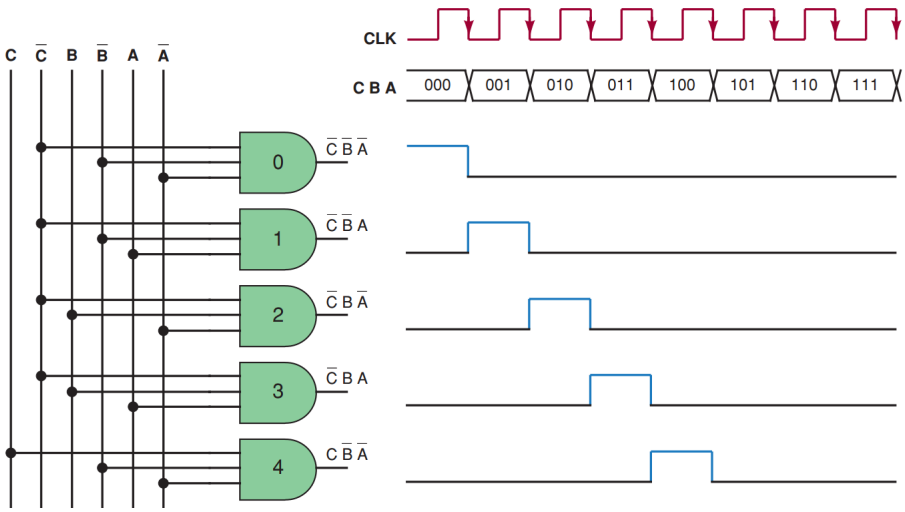
- EN, the enable for the eight-bit counter, is connected to ENT output on stage 1 since ENT control the RCO output.
- Both counter blocks are clocked together synchronously, but the block on the stage 2 is disabled until the last-significant output has reached its terminal stage, incicated as TC1.
- When Q3, Q2, Q1, reaches 1111 and if EN is HIGH, then TC1 will output a HIGH. This will allow both counter stages to count up one with the next PGT on the clock.
- Stage 1 will back to 0000 and stage 2 will increment from its previous output state. TC1 is LOW and stage 1 is no longer to at it terminal stage.

16.16 Decoding the Counter

- Decode is the conversion of the binary output to a decimal value.

1. **ACTIVE-HIGH Decoding:**
   - A active-HIGH decoding produces HIGH outputs to indicate detection.





   - The decoder consists of eight three-bit input AND gates. Each AND gate produces a HIGH output for one particular state of the counter.

2. **ACTIVE-LOW Decoding:**
   - If we replace AND gates by NAND gates, the decoder outputs produce a normal HIGH signal, which goes LOW only when the number being decoded occurs.

16.17 BCD Counter Decoding

- A BCD counter has 10 states that can be decoded and provides 10 outputs corresponding to the decimal digits 0 thorough 9 and represented by the states of the counter FFs.

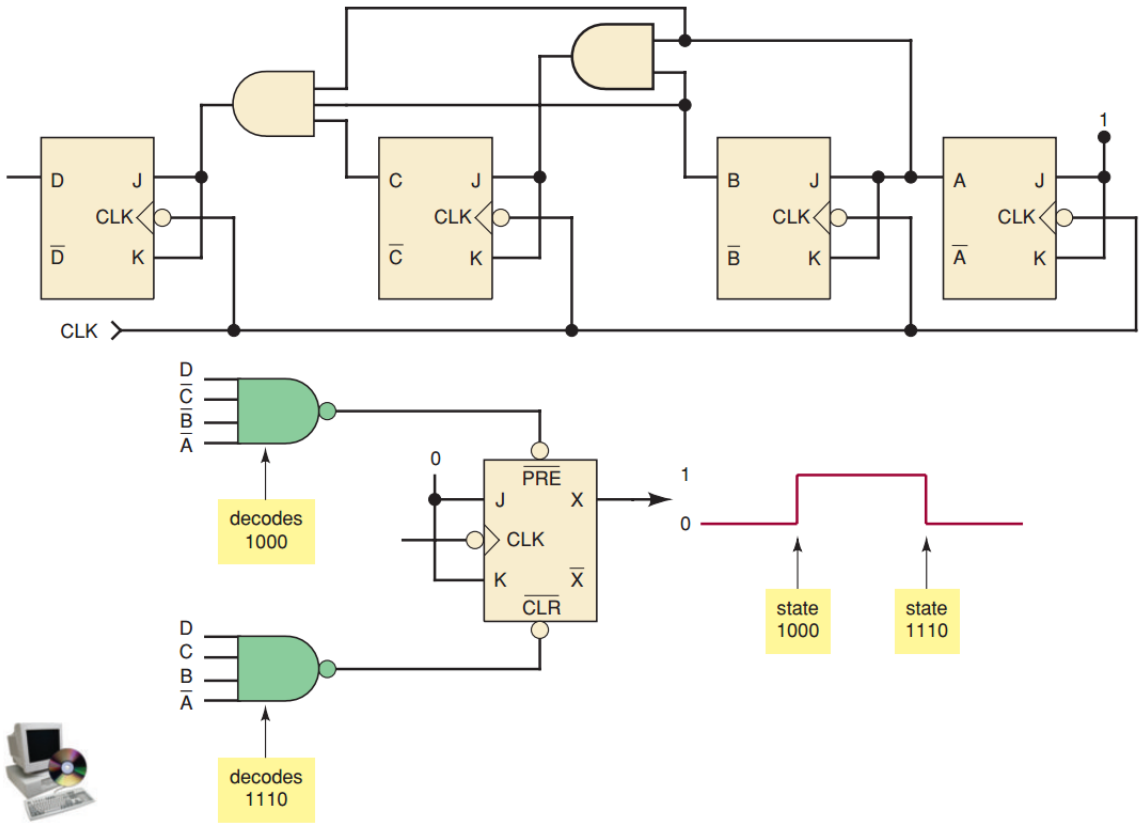
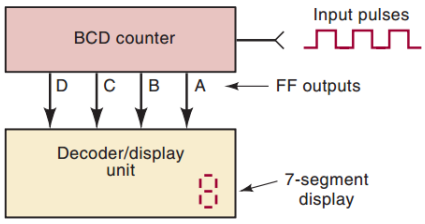


FIGURE 7-21 Example 7-15.

- These 10 outputs can be used to control invidual indicator LEDs for a visual display and use a single display device instead of 10 separate LEDS to display the decimal number from 0 to 9.



16.18 Analyzing Synchronous Counter:

- To analyze the Synchronous Counter we use a **PRESENT/NEXT** stage table:

- Write the logic expression for each Ff control input.
- Assume the **PRESENT** state for the counter and apply the combination of bits to the control logic expressions. The outputs from the control expression will allow us to predict **NEXT** for the counter after clocking.

16.19 Synchronous Counter Design:

1. J-K Excitation Table:

Transition at FF Output	PRESENT State Q_n	NEXT State Q_{n+1}	J	K
0 → 0	0	0	0	x
0 → 1	0	1	1	x
1 → 0	1	0	x	1
1 → 1	1	1	x	0

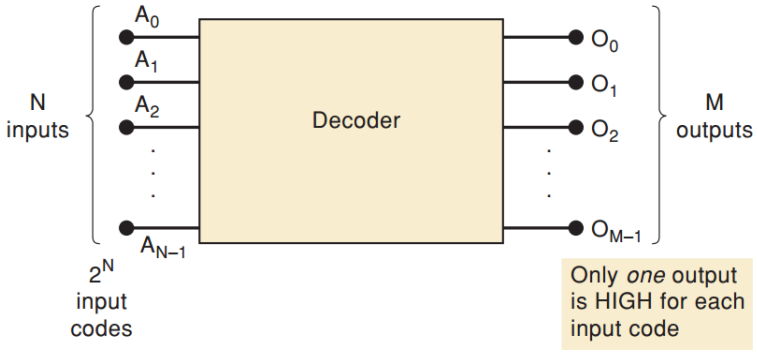
2. D Exitation Table:
   - For D FF D will have the value of the **NEXT** stage:

PRESENT	NEXT	D
0	0	0
0	1	1
1	0	0

17 MSI (Medium Scale Itegration) Logic Circuit

17.1 Decoders

- A **Decoder** is a logic circuit that accepts a set of inputs that represents a binary number and activates only the outputs that corresponds to that input number.
- A **decoder circuit** looks at the inputs, determines which binary number represent there, and activates the outputs of that number, the other outputs will remain inactive.



- In the picture:

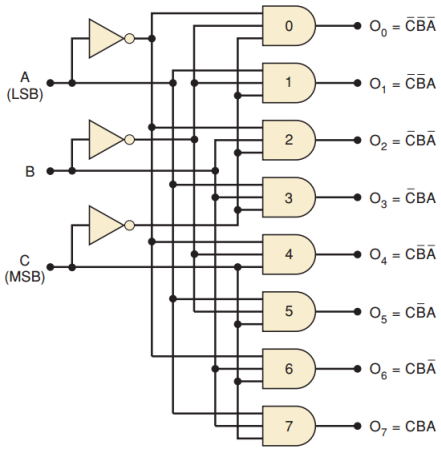
- there are N-inputs can be 0 or 1 so that there are 2^N inputs combination

or codes.

- For each of these input combinations, only one of the M outputs will be stay HIGH; all the other outputs will stay LOW.
- Many decoder are designed to produce te active-LOW outputs while all the other will stay HIGH.
- Not all the decoders ultilize all of 2^N possible inputs but only the certai n ones.

ENABLE Input
- Some decoders have one or more than ENABLE inputs that are used to control the operation of the decoder.
- With the ENABLE line is HIGH, the decoder will function normally and the A, B, C will determine which output is HIGH.
- When the ENABLE line is LOW, **all** of the output will be forced to the LOW state regardless of the levels at the A, B, C inputs.

3-Line to 8-Line Decoder:

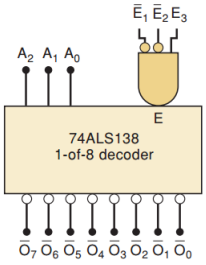


C	B	A	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

74ALS138 1-of-8 Decoder:

$\bar{E}_1$	$\bar{E}_2$	E_3	Outputs
0	0	1	Respond to input code $A_2A_1A_0$
1	X	X	Disabled – all HIGH
X	1	X	Disabled – all HIGH
X	X	0	Disabled – all HIGH

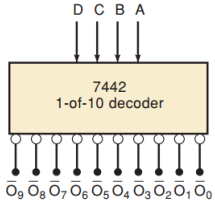
(b)



(c)

- The input code is applied at A_2, A_1 and A_0, where A_2 is the MSB.
- Inputs $\bar{E}_1$, $\bar{E}_2$ and E_3 are separately enable inputs that are combined in AND gate.
- To enable the output NAND gates to respond to the input code at $A_2A_1A_0$ this AND gate must be HIGH.

BCD-to-Decimal Decoder:



(b)

Inputs				
D	C	B	A	Active Output
L	L	L	L	$\overline{O}_0$
L	L	L	H	$\overline{O}_1$
L	L	H	L	$\overline{O}_2$
L	L	H	H	$\overline{O}_3$
L	H	L	L	$\overline{O}_4$
L	H	L	H	$\overline{O}_5$
L	H	H	L	$\overline{O}_6$
L	H	H	H	$\overline{O}_7$
H	L	L	L	$\overline{O}_8$
H	L	L	H	$\overline{O}_9$
H	L	H	L	None
H	L	H	H	None
H	H	L	L	None
H	H	L	H	None
H	H	H	L	None
H	H	H	H	None

H = HIGH Voltage Level
L = LOW Voltage Level

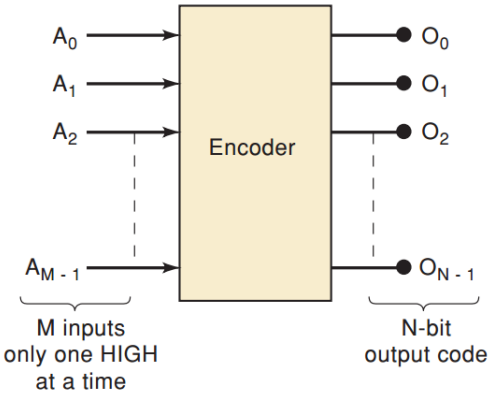
Decoder Application

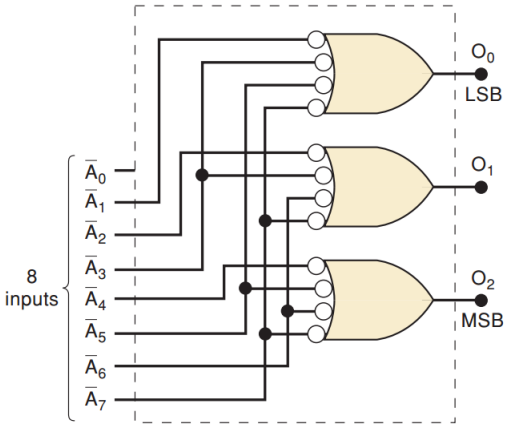
- Decoders are used whenever an output or group of the outputs is to be activated only on the occurence of a specified combination.
- These input levels are often provided by the outputs of a counter or a register. When the decoder inputs are connected with a counter that is being continually pulsed, the decoder outputs will be activated sequentially and they can be used as timing or sequencing signals to turn devices on or off at specific time.
- Decoder are wirely used in the memory system where they respond to the address code generated by the general processor to activated the particular memory location:

  - Each memory IC consists of many register that can store binary numbers (data).
  - Each register has it own unique address to distinguish with the other register.
  - A decoder is built into the memory IC's circuitry and allows a particular storage register to be activated when the combination of inputs are activated.

17.2 Encoder

- Most decoders accept an input code and produce a HIGH (or LOW) at one output line. The opposite of decode process is **encoding** and is performed by a logic circuit called an **encoder**.
- An encoder has a number of input lines, only one of which is activated at a given time and produces an N-bit output code, depending on which input is activated.





Inputs								Outputs		
$\overline{A}_0$	$\overline{A}_1$	$\overline{A}_2$	$\overline{A}_3$	$\overline{A}_4$	$\overline{A}_5$	$\overline{A}_6$	$\overline{A}_7$	O_2	O_1	O_0
X	1	1	1	1	1	1	1	0	0	0
X	0	1	1	1	1	1	1	0	0	1
X	1	0	1	1	1	1	1	0	1	0
X	1	1	0	1	1	1	1	0	1	1
X	1	1	1	0	1	1	1	1	0	0
X	1	1	1	1	0	1	1	1	0	1
X	1	1	1	1	1	0	1	1	1	0
X	1	1	1	1	1	1	0	1	1	1

FIGURE 9-13 Logic circuit for an octal-to-binary (8-line-to-3-line) encoder. For proper operation, only one input should be active at one time.

- A LOW at any single input will produce the output binary code corresponding to that input.
- A_0 is not connected to the logic gates because the encoder outputs will normally be at 000.

17.3 Prority Encoder

- One drawback of the simple encoder is when more than one input are activated at one time.
- So we will have a necessary logic gate to ensure that when we have more than one inputs, the output will correspond to the highest-numbered input.

17.4 74147 Decimal to BCD Priority Encoder

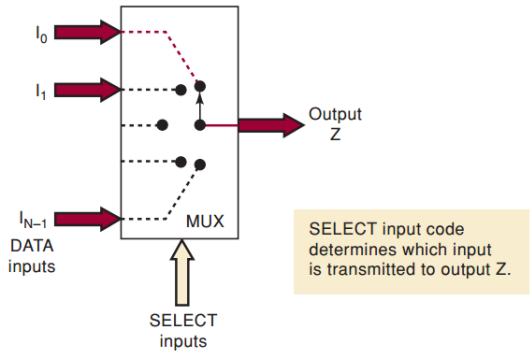
(sẽ thêm vào sau)

17.5 Switch Encoder

- The switch acts as open type, when a digit is depressed the circuit will produce the BCD code for that digit.
- The switch is used whenever the BCD data must be entered manually to the digital system.
- When the first key is depressed, the BCD code for that digits is sent to a four-bit FF register, when the second key depressed, the BCD code for that digits is sent to **another** four-bit FF register.
- Each four-bit register drives a decoder/driver and a numerial number display so that the number can be displayed.

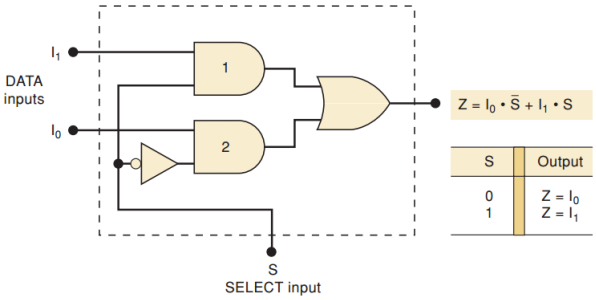
17.6 Multiplexer (Data Collector)

- A digital multiplexer or data selector is a logic circuit that accepts several digital data inputs and selects one of them at any given time to pass on to the output.
- The routing of the desired data input to the output is controlled by SELECT inputs:
- The multiplexer acts like a digitally contrlled multiposition switch where the digital code appied to the SELECT input controls which data inputs is switched into the output.



Two-Input Multiplexer

- For a two-input multiplexer with data inputs I_0 and I_1 and the SELECT input.



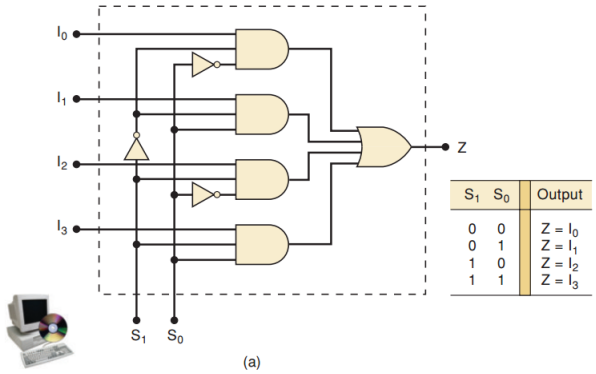
- The boolean expression for the output Z is:

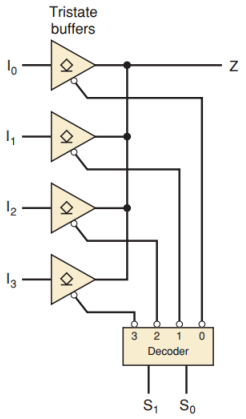
$$Z = I_0\bar{S} + I_1 S$$

- When S = 0, this expression becomes:

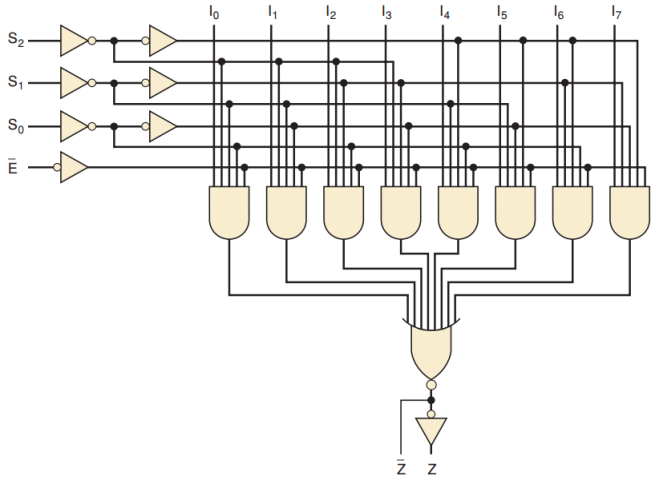
$$\begin{aligned} Z &= I_0 \cdot 1 + I_1 \cdot 0 \quad (1) \\ &= I_0 \quad (2) \end{aligned}$$

4-input Multiplexer



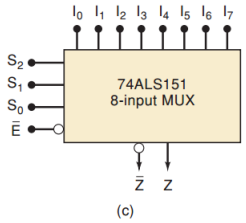


8-input Multiplexer



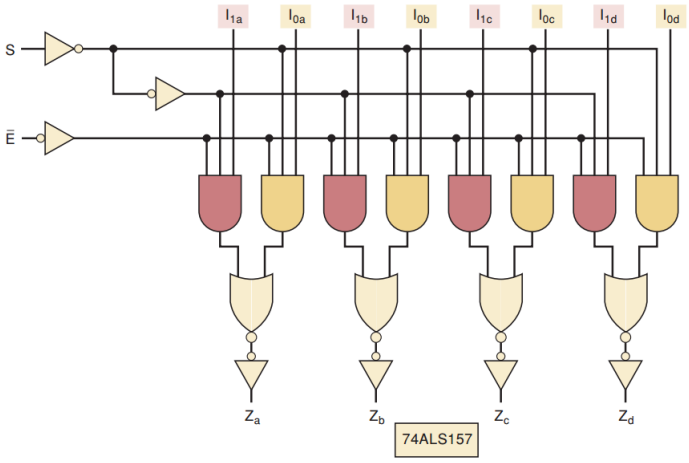
Inputs				Outputs	
$\bar{E}$	S_2	S_1	S_0	$\bar{Z}$	Z
H	X	X	X	H	L
L	L	L	L	$\bar{I}_0$	I_0
L	L	L	H	$\bar{I}_1$	I_1
L	L	H	L	$\bar{I}_2$	I_2
L	L	H	H	$\bar{I}_3$	I_3
L	H	L	L	$\bar{I}_4$	I_4
L	H	L	H	$\bar{I}_5$	I_5
L	H	H	L	$\bar{I}_6$	I_6
L	H	H	H	$\bar{I}_7$	I_7

(b)



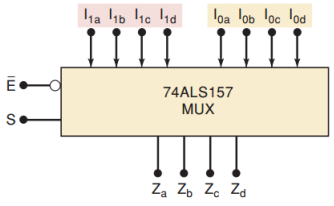
(c)

17.7 Quad 2-Input MUX



(a)

(74ALS157)



(b)

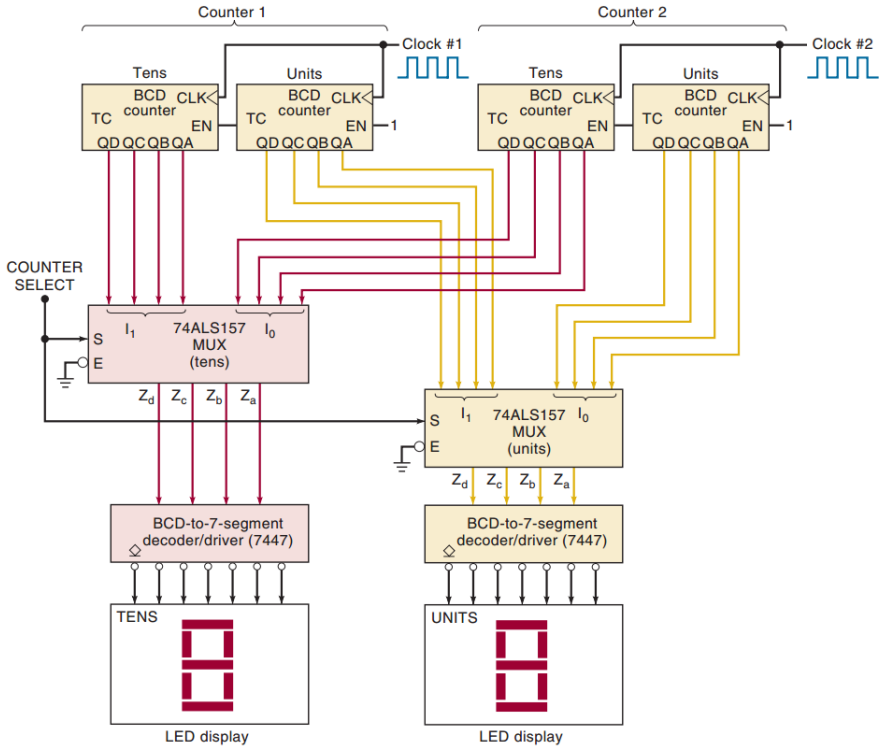
Ē	S	Z_a	Z_b	Z_c	Z_d
H	X	L	L	L	L
L	L	I_{0a}	I_{0b}	I_{0c}	I_{0d}
L	H	I_{1a}	I_{1b}	I_{1c}	I_{1d}

(c)

17.8 MUX Application

Data Routing

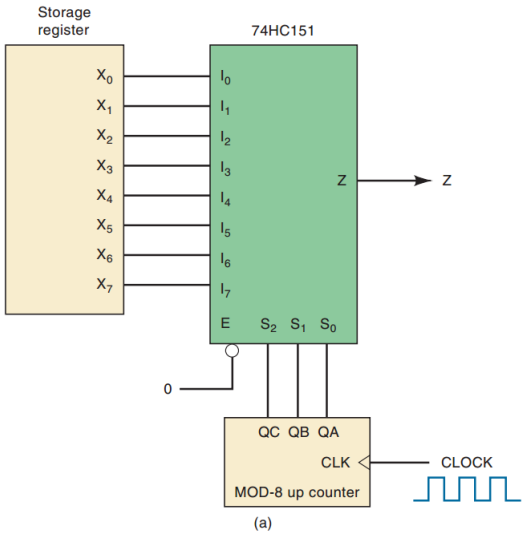
- Multiplexer can route data from one of several sources to one destination. The circuit below shows the system for displaying two multidigit BCD counters and choose to display one LED only at time.



1. Each counter consists of either 2 BCD counters and each one is driven by its own CLK.

2. When the **COUNTER SELECT** line is HIGH, the ouputs of counter 1 will be will be allowed to pass through the multiplexer to the decoder (7447) and display on lED. When the **COUNTER SELECT** = 0, the output of counter 2 will be allowed to pass through the multiplexer to the decoder and display on LED.

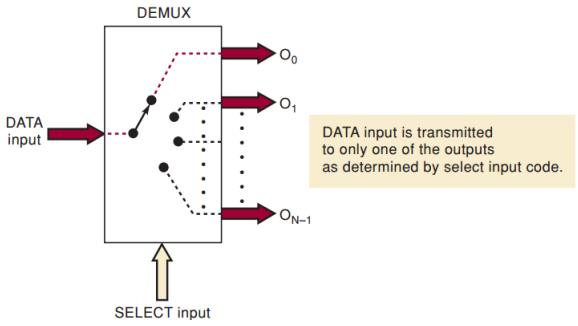
Parallel to Serial Conversion



- The data is present in parallel form at the outputs of X register and are fed to the 8-input MUX.
- A 3-bit MOD3 counter is used to provide the select code bits $S_2S_1S_0$ so they cycle through from 000 to 111 as CLK pulses are applied.
Ex: We have the signal where $X_7X_6X_5X_4X_3X_2X_1X_0 = 10110101$.
Solution: This conversion process takes a total 8-CLK cycles. Note that X_0 is the LSB and X_7 is MSB.

17.9 Demultiplexer (Data Distributor:)

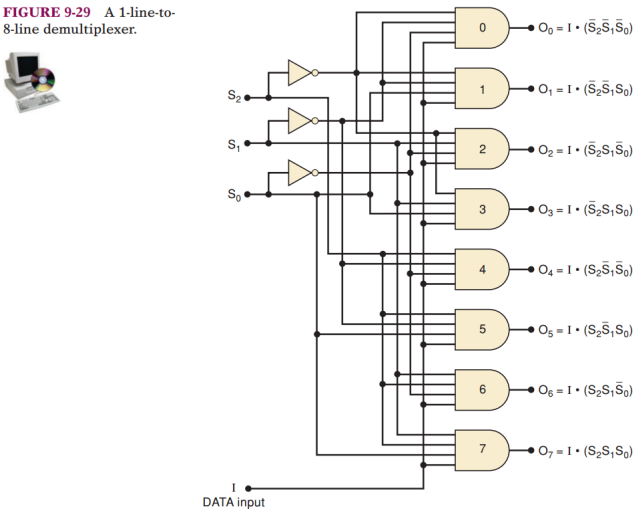
- A demultiplexer does reversed operation to the multiplexer: it takes a single input and distributes several outputs.



- In demultiplexer, the select input code determines to which output the DATA input will be transmitted.
- In other words, the demultiplexer takes ont input data source and selectively distributes it to 1 of N output channels just like a multiposition switch.

1-Line to 8-Line Demultiplexer:

FIGURE 9-29 A 1-line-to-8-line demultiplexer.



- The single data input line I is connected to all 8 AND gates, but only one of these gates will be enabled by the SELECT input lines. For each SELECT code the demultiplexer will decides which output will be distributed.

SELECT code			OUTPUTS							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	I
0	0	1	0	0	0	0	0	0	I	0
0	1	0	0	0	0	0	0	I	0	0
0	1	1	0	0	0	0	I	0	0	0
1	0	0	0	0	0	I	0	0	0	0
1	0	1	0	0	I	0	0	0	0	0
1	1	0	0	I	0	0	0	0	0	0
1	1	1	I	0	0	0	0	0	0	0

Note: I is the data input

18 Fuzzy Logic

- Fuzzy Logic (FL) is a method of reasoning that resemble the human reasoning.
- Unlike Combinational Logic which only takes precise input and produce a definite output as 0 and 1.
- Fuzzy Logic provide the wide range of the output that surround the value of 0 and 1.

- To simlify the the result is 1, we can give the output of around 0.75 or 0.9 when the result is 0, the output can be a 0.34 or 0.2.
- The Fuzzy Logic works on the levels of possiblities of inputs to achieve the definite output.
- Fuzzy Logic is useful for commercial and practical product.
  - It can control machines and consumer products.
  - It may not give a accurate result, but it still a acceptable result.
  - Fuzzy logic help to deal with the uncertainly in engineering.

18.1 Fuzzy logic system

- It has 4 main parts:

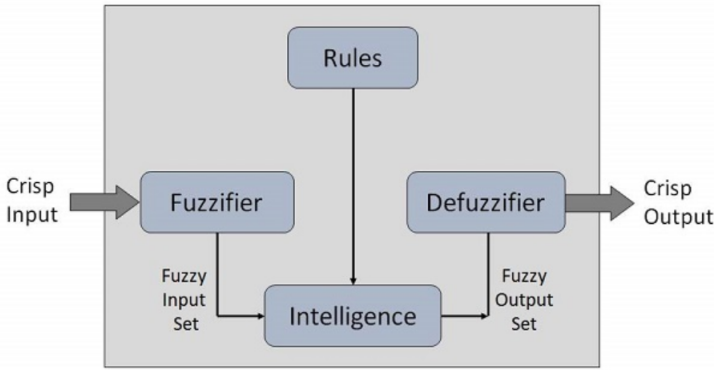
- Fuzzification Module:
  - It transforms the system inputs, from crisp number to fuzzy set.
  - Crisp number is a single point, ex: 3, 5.5, 6; when fuzzy number is a fuzzy set with different degree os closeness to a given crisp number.
  - It splits the input signal into 5 components:

LP	x is Large Positive
MP	x is Medium Positive
S	x is small
MN	x is Medium Negative
LN	x is Large Negative

- Knowledge Base:
  - It stores IF-THEN rules.

- Inference Engine:
  - it simulates the human reasoning process by making fuzzy inference on the inputs and IF-THEN.

- Defuzzification Module:
  - It transforms the fuzzy set obtained by the inference engine into crisp value.

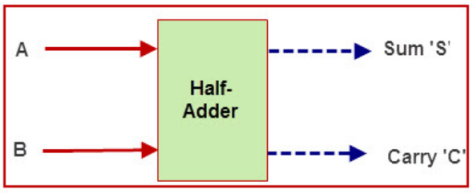


19 Half Adder and Full Adder

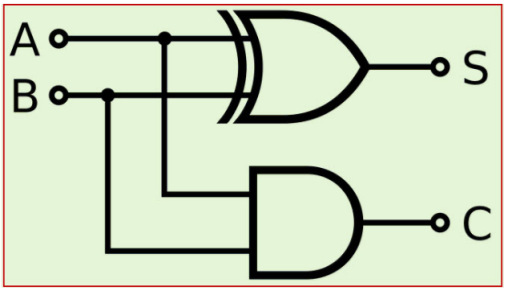
- In digital system, an adder is a **digital circuit** in electronics that implies the addition of numbers.
- It is used in the arithmetic logic units. - It also untilizes the many parts of the processors, address, ... etc.

19.1 Half Adder

- The half adder adds 2 binary inputs and produce the 2 outputs as SUM and CARRY; XOR gate is applied to both inputs to produce SUM; when AND gate is applied to produce CARRY.



Half Adder



Half Adder Logic Circuit

19.2 Full Adder

- The different between Full Adder and Half Adder is Full Adder use 3 binary inputs, which 2 of inputs can be reffered as operands and the other one is a bit carried in.