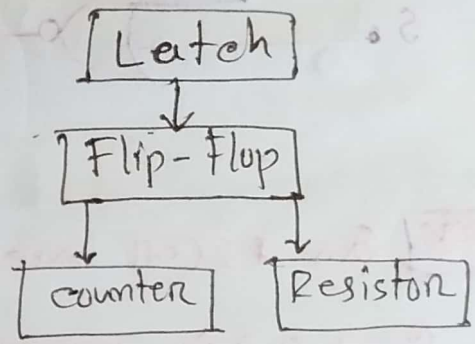
Latches (DLD)

Latch: Latches are the basic building blocks of any flip flop and they are capable of holding 1 bit unit necessary.

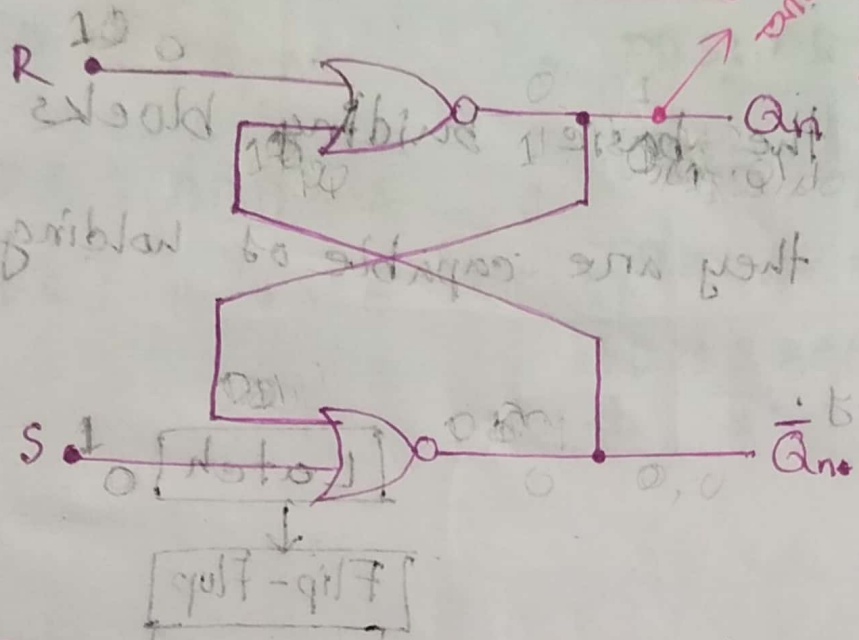


NOR Latch & NAND ~~gate~~ Latch:

The SR Latch is a circuit with two cross-coupled NOR gates / two cross-coupled NAND gates and two inputs labeled S for 'Set' and R for 'Reset'. Outputs Q_n and $\overline{Q_n}$ are the complement of each other.

** A Flip-Flop is a binary storage device capable of storing one bit of information. In a stable state the output of a flip-flop is either 0 or 1 (it is called bi-state multi-vibrator).

NOR:

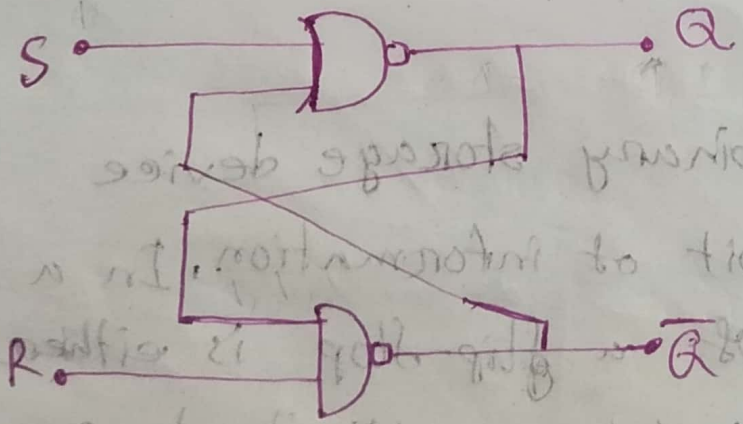


এটাই input আবার এটাই output

S	R	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

(**) Q_n হলো memory এর মতো যেটা input হিসাবে দিবো এটাই আবার output হিসাবে বের হয়ে যাবে। $\bar{Q}_n$, Q_n এর complement. যখনই Q_n and $\bar{Q}_n$ same result output হিসাবে দিবে তখন বুঝাতে হবে circuit টাকে বন্ধ করতে হবে R and S control key হিসাবে কাজ করে।

(*)



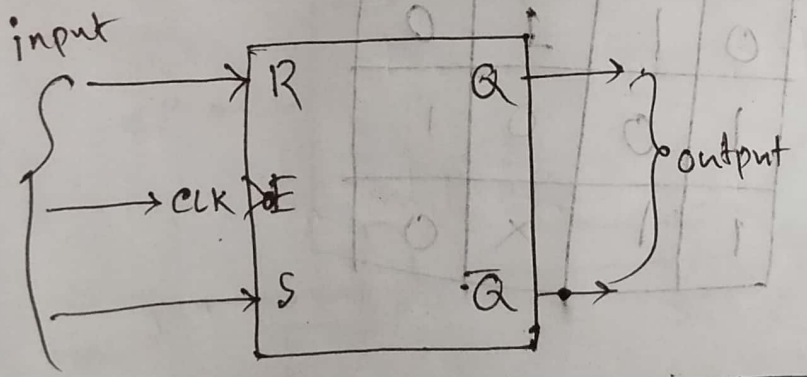
S	R	Q_n	Q_{n+1}
0	0	0	X
0	0	1	X
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

SR - flip-flop

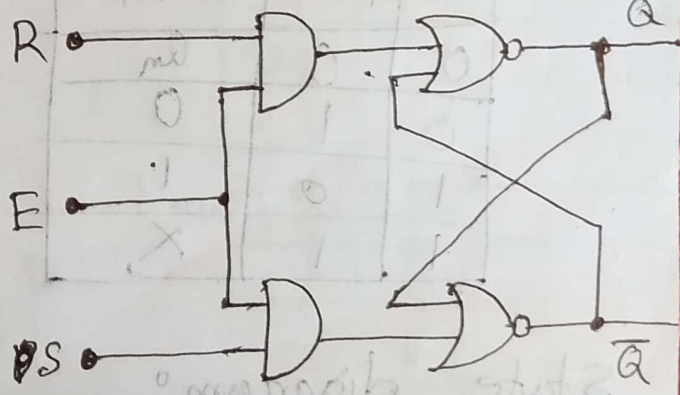
SR flip-flop: It is a flip-flop with two inputs, one is S (set) and another is R (Reset). Set basically indicates set the flip flop which means output 1 and reset do the complement of set (output = 0).

Here a clock pulse is supplied to operate the flip flop.

Block diagram:



Circuit diagram



Truth Table:

	S	R	Q_n	Q_{n+1}
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	X
7	1	1	1	X

	sr	s'r'	s'r	sr	sr'
q		00	01	11	10
q'	00	0	2	6	4
q	01	1	3	7	5

characteristics equation:

$$Q_{n+1} = S + r'Q_n$$

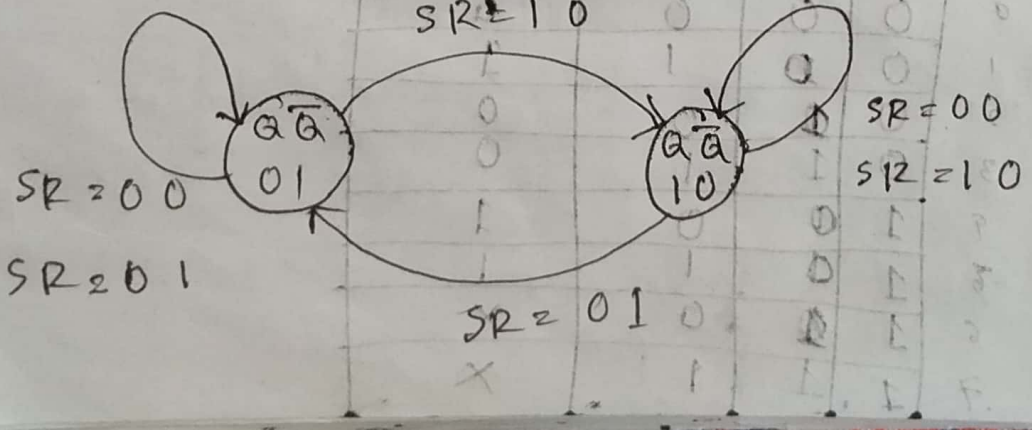
Functional Table

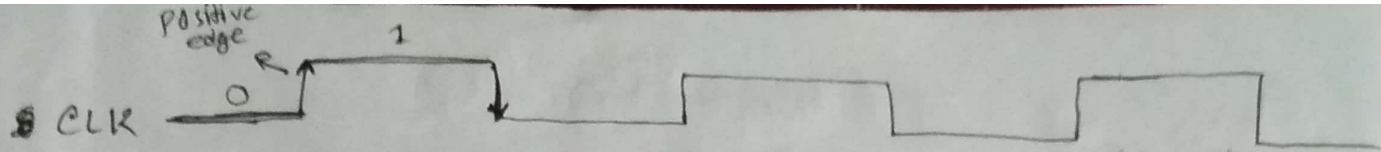
S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	X

Excitation table

Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

State diagram:





S

R

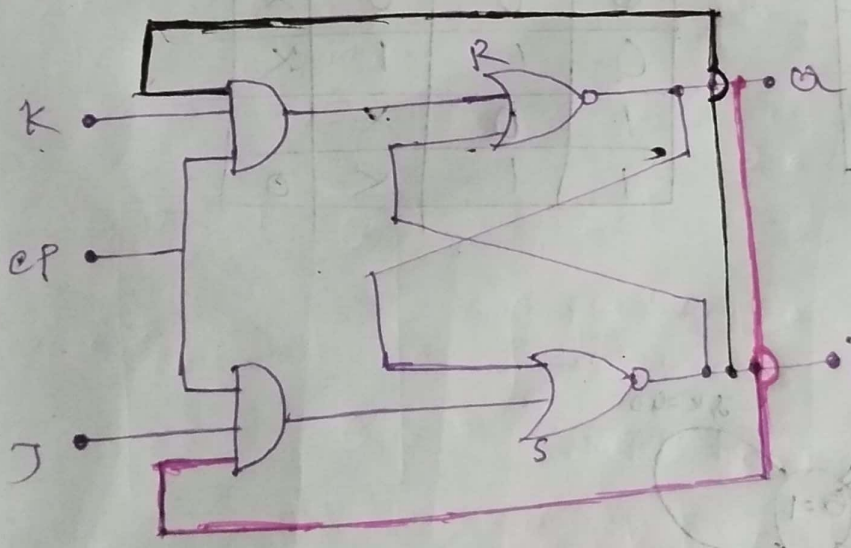
JK – flip-flop.

to solve the invalid input state $S = R = 1$ of SR flip flop we use JK flip-flop.

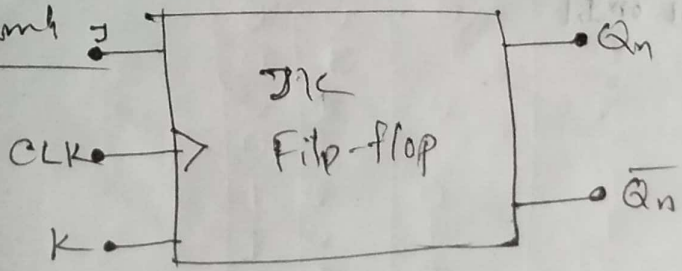
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

0 এলে $Q_{n+1} = 1$

1 এলে $Q_{n+1} = 0$



Block diagram:



K-Map

	JK	J'K'	J'K	JK	JK'
q		00	01	11	10
q'	00	0	0	1	1
q	01	1	0	0	1

characteristics Eqn:

$$Q_{n+1} = \bar{J}Q\bar{n} + \bar{K}Q_n$$

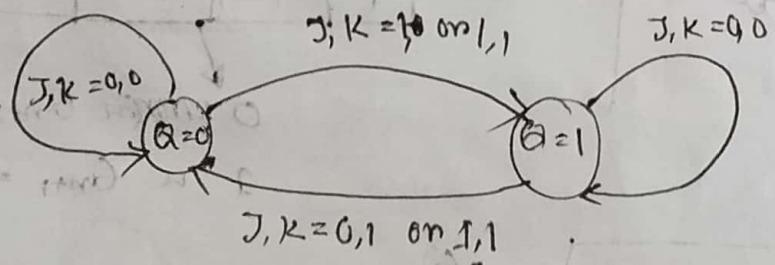
Functional table:

J	k	$Q\bar{n}_{+1}$
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

Excitation table:

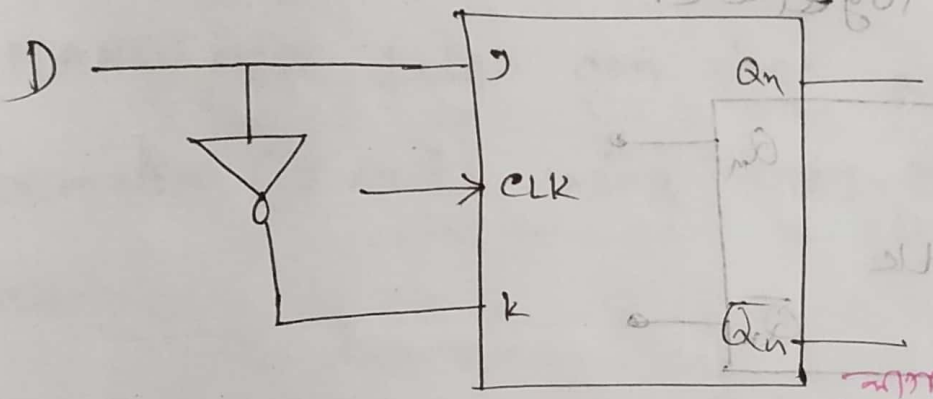
Q_n	Q_{n+1}	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

State diagram:



D flip-flop

The D (Data/Delay) flip-flop, tracks the input at D and produces the same value as output.



Qn আউটপুট বেওয়া লাগবে না, D এ যা input দিব Qn+1 এ তাই output আসবে।

D

J	K	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

J-K flip-flop এ

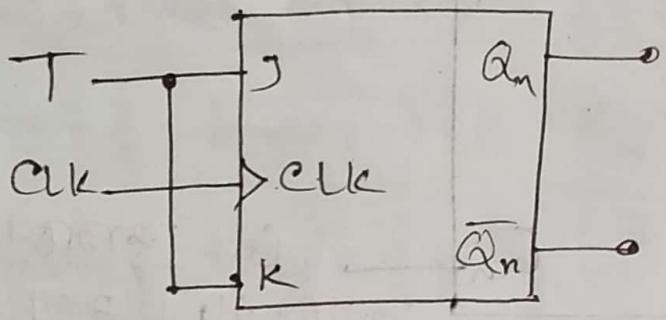
J and K এর same input যাবে

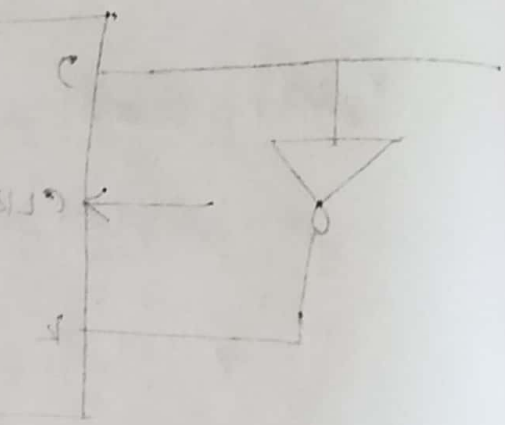
D	Qn	Qn+1
0	0	0
0	1	0
1	0	1
1	1	1

এরপর বাকি input গুলোর জন্যই J এর জন্য D বসিয়ে দিব।

T (toggle) flip flop

The T (toggle) flip-flop is a complement of flip-flop and can be obtained from J-K flip-flop when inputs J and K are ~~same~~ tied together.





J	K	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

J and K এর different input গুলো বাদ গেল. J এর ক্ষেত্রে T ধরে নিলে.

T	Qn	Qn+1
0	0	0
0	1	1
1	0	1
1	1	0

$$Q_{n+1} = T \oplus Q_n$$

1. Explain how a NAND/NOR latch store a bit. What is their limitation? (20-21, 19-20) [3]

Answer:

A NAND/NOR latch can store a single bit of information (0 or 1) using cross-coupled logic gates. In those latch the inputs are indicated by S(set) and R(reset) and outputs are indicated by Q_n and $|Q_n$.

The process of storing bit of NAND Latch:

1. When S = R = 1, the latch maintains the current state.
2. When S = 0 and R = 1, the output = 1 means set the latch.
3. When S = 1 and R = 0, the output = 0 means reset the latch.

S (set)	R (reset)	Q_n	Q_{n+1}
0	0		invalid
0	0		invalid
0	1		1
0	1		1
1	0		0
1	0		0
1	1		unchanged
1	1		unchanged

Limitations:

1. When S = R = 0, the output = invalid and latch throws error.
2. When $Q_n = |Q_n$ = same bit of value (0 or 1), the latch level-triggered and it goes to race condition.

The process of storing bit of NOR Latch:

1. When S = R = 0, the latch maintains the current state.
2. When S = 0 and R = 1, the output = 0 means reset the latch.
3. When S = 1 and R = 0, the output = 1 means set the latch.

S (set)	R (reset)	Q_n	Q_{n+1}
0	0	0	unchanged
0	0	1	unchanged
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	invalid
1	1	1	invalid

Limitations:

1. When S = R = 1, the output = invalid and latch throws error.
2. When $Q_n = |Q_n$ = same bit of value (0 or 1), the latch level-triggered and it goes to race condition.

2. What are the limitations of S-R flip flop? How can be resolved these? (20-21, 18-19) [4]

The limitations and resolutions of S-R flip-flop are given below:

Indeterminate State:

- Limitation: When both inputs (S and R) = 1, the output becomes unstable.
- Solution: Use a J-K flip-flop, which toggles instead of becoming undefined when both inputs are high.

Lack of Clock Control:

- Limitation: Basic S-R flip-flops are level-triggered and may not reliably capture input changes.
- Solution: Use a Clocked S-R flip-flop or a D flip-flop for stable, edge-triggered responses.

Glitches from Asynchronous Inputs:

- Limitation: Small timing differences can cause unstable output.
- Solution: Use a Clocked or Master-Slave flip-flop to synchronize inputs and prevent glitches.

Limited in Complex Circuits:

- Limitation: The S-R flip-flop's simplicity restricts its use in advanced circuits.
- Solution: Use more versatile flip-flops like D or J-K flip-flops in complex designs.

3. Define race-around condition in J-K flip-flop? How can you overcome the problem? Explain with appropriate figure and waveforms. (20-21, 19-20) [5]

Answer:

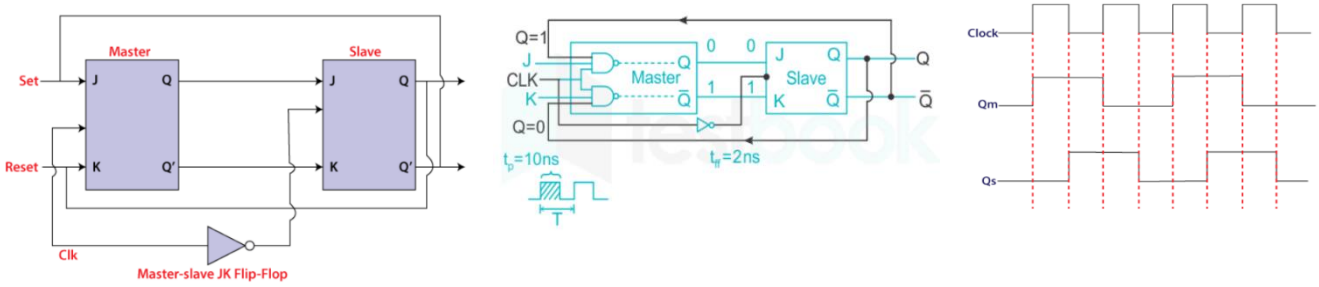
In JK Flip-flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as CLK is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flip-flop.

The truth table of the J-K flip-flop is given below:

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

The invalid at J = K = 1 is known as the race-around condition.

To overcome this condition, a Master-slave JK filp-flop is used.



Explaination:

1. The master-slave flip flop is constructed by combining two JK flip flops.
2. These flip-flops are connected in a series configuration. In these two flip flops, the 1st flip flop work as a "master", called the master flip flop, and the 2nd work as a "slave", called the slave flip flop.
3. When the clock pulse is true, the slave flip flop will be in the isolated state, and the system's state may be affected by the J and K inputs. The "slave" remains isolated until the CP is 1. When the CP is set to 0, the master flip-flop passes the information to the slave flip-flop to obtain the output.

4. How a NAND/NOR latch store a bit? (19-20) [2]

Answer:

NAND Latch

1. Setup: Made of two cross-connected NAND gates.
2. Inputs: S(Set) and R(Reset).
3. Operation:
   - Set: S = 0 and R = 1, makes output Q = 1.
   - Reset: S = 1, R = 0, makes output Q = 0.
   - Hold: S = 1, R = 1, keeps the last value of Q, storing a bit.

NOR Latch:

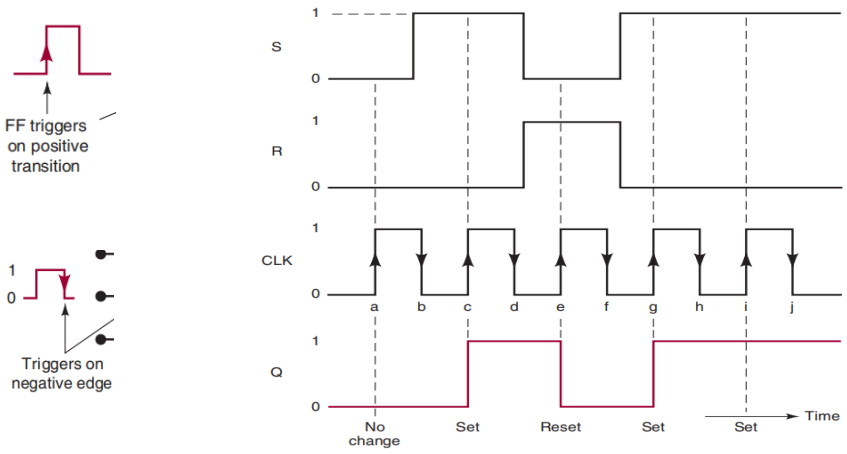
1. Setup: Made of two cross-connected NOR gates.
2. Inputs: S(Set) and R(Reset).
3. Operation:
    - Set: S = 0 and R = 1, makes output Q = 0.
    - Reset: S = 1, R = 0, makes output Q = 1.
    - Hold: S = R = 0, keeps the last value of Q, storing a bit.

5. Explain the synchronous/clocked S-R filp-flop with waveforms. (19-20) [4]

Answer:

A **synchronous/clocked S-R flip-flop** is a type of S-R (Set-Reset) flip-flop that includes a clock signal to control when the inputs S (Set) and R (Reset) can affect the output. The flip-flop changes its state only during a specific clock edge (usually the rising edge) if the clock signal is active, making it synchronous with the clock.

Waveform:



The waveforms in Figure show how a **clocked S-R flip-flop** operates:

S	R	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

1. **Initial State**: All inputs are 0, and Q is assumed to be 0.
2. **First Clock Pulse (Point a)**: With both S and R at 0, the flip-flop remains in its current state (Q = 0).
3. **Second Clock Pulse (Point c)**: S = 0 and R = 0, so the flip-flop sets Q to 1.
4. **Third Clock Pulse (Point e)**: S = 0 and R=1, so the flip-flop clears Q to 0.
5. **Fourth Clock Pulse (Point g)**: S=1 and R=0, setting Q to 1 again.
6. **Fifth Clock Pulse**: S=1 and R=0 again, so Q remains at 1 since it's already high.
7. **Invalid Condition**: When S=R=1, this creates an ambiguous state and should be avoided.

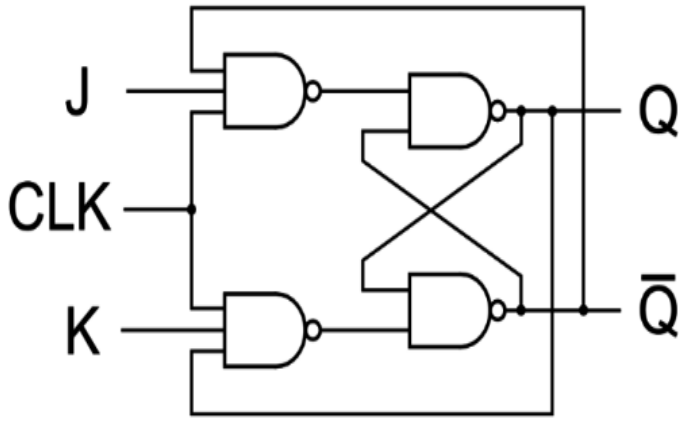
6. What is a flip flop? Explain the function of a JK flip flop with suitable circuit diagram. (18-19) [4]

Answer:

A flip-flop is a fundamental memory element in digital electronics that can store a single bit (0 or 1) of data.

J-K Flip-Flop

The **J-K flip-flop** is an improvement over the S-R flip-flop, as it eliminates the indeterminate (invalid) state that occurs when both Set (S) and Reset (R) inputs are 1. Instead, the J-K flip-flop includes feedback that allows it to toggle its output when both inputs are high.



Truth Table

J	K	Q_N	Q_{N+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Function and Operation

The J-K flip-flop's operation depends on the values of J, K, and the clock:

When Clock is Low: The flip-flop maintains its previous state; inputs J and K have no effect.

When Clock is High (or on the rising edge, in edge-triggered designs):

- **J = 0, K = 0**: The flip-flop holds its previous state.
- **J = 0, K = 1**: The output Q is reset to 0,
- **J = 1, K = 0**: The output Q is set to 1.
- **J = 1, K = 1**: The output Q toggles (switches between 0 and 1) with each clock pulse.

Chapter 6 (Basic)

Binary Addition, Subtraction, Multiplication & division:

addition:

$$\begin{array}{r} 1 \\ +1 \\ \hline \boxed{1}\,0 \end{array} \qquad \begin{array}{r} 1 \\ +0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ +0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ +1 \\ \hline \boxed{1}\,0 \end{array}$$

carry carry

Lets Example:

$$\begin{array}{r} 1100 \\ +\ 1110 \\ \hline \boxed{1}\,1010 \end{array}$$

carry

$$\begin{array}{r} 1101.011 \\ +\ 1100.110 \\ \hline \boxed{1}\,1010.001 \end{array}$$

carry

Subtraction:

$$\begin{array}{r} 1 \\ -1 \\ \hline 0 \end{array} \qquad \begin{array}{r} 1 \\ -0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 0 \\ -0 \\ \hline 0 \end{array}$$

$0-1 = 1$ (borrow 1)

$0-10 = 0$ (borrow 1)

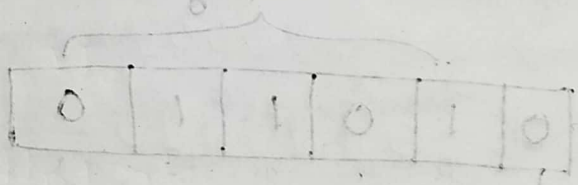
$1-10 = 1$ (borrow 1)

$$\begin{array}{r} 11001 \\ -\ 10101 \\ \hline 00100 \end{array} \qquad \begin{array}{r} 11000 \\ -\ 00111 \\ \hline 10001 \end{array}$$

Multiplication:

$$\begin{array}{r} 10101 \\ \times\ \ \ 101 \\ \hline 10101 \\ 000000 \\ +\ 1010100 \\ \hline 1101001 \end{array}$$

স্বাভাবিক
সহজ পদ্ধতি



Division:

$$\begin{array}{r} 11 \overline{)\ 1001\ (} 11 \\ -\ 11 \\ \hline 0011 \\ \ \ \ 11 \\ \hline 0 \end{array}$$

স্বাভাবিক

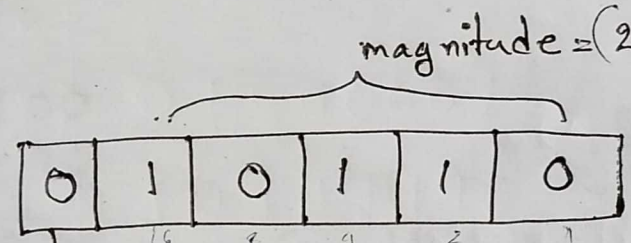
$$\begin{array}{r} 100 \overline{)\ 11101\ (} 111 \\ -\ 100 \\ \hline 0110 \\ \ \ \ 100 \\ \hline 010 \\ \ \ \ 100 \\ \hline 001 \end{array} \qquad = 111 \qquad \text{Reminder} = 001$$

2's complement: (addition and subtraction)

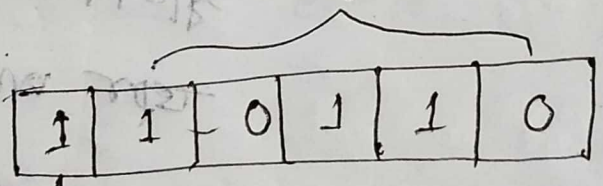
Representing signed numbers:

In general the common convention is that a 0 in the sign bit represents a positive number and a 1 in the sign bit repres a negative number.

Example:



$= +22_{10}$



$= -22_{10}$

magnitude = absolute value

1's complement:

The 1's complement of binary number means changing the bit 0 to 1 and 1 to 0.

```
1 0 1 0 1 1  ⟶ orginal number
↓ ↓ ↓ ↓ ↓ ↓
0 1 0 1 0 0  ⟶ 1's complement.
```

2's complement:

The 2's complement of a binary number is formed by taking 1's complement of the number, then a adding 1 to the least-significant-bit position.

Example:

$45_{10} = 101101_2$

101101 → original
010010 → 1's complement
+ 1 → adding 1 with 1's complement

010011 → 2's complement of 101101_2 or 45_{10}

sign bit(0) → -45_{10}

* যেকোনো সংখ্যার 2's complement বের করতে হলে তার sign bit এর sign change হবে।

positive কে negative এবং negative কে positive করবে।

* Die Decimal number (127 to -128) এর জন্য হলে 8-bit দিয়ে হিসাব করব, এর বেশি হলে 16 bit.

4. Using 2's complement system perform the following operations: (19 - 20) [2]

(i) -65 - 88 (ii) 34 + 55

128 + 16 + 8 + 1

128	64	32	16	8	4	2	1
0	1	0	0	0	0	0	1

(i) -65 - 88

$\Rightarrow (-65) + (-88)$

Now,

$65 \rightarrow 01000001$

$10111110 \rightarrow$ 1's complement

$+1$

$(-65) \rightarrow 10111111 \rightarrow$ 2's complement

sign bit (-)

$88 \rightarrow 01011000$

$10100111 \rightarrow$ 1's complement

$+1$

$(-88) \rightarrow 10101000 \rightarrow$ 2's complement

sign bit (-)

① -65-88

Given that,

= -65-88

or, (-65) + (-88)

65 → 00000000 01000001

11111111 10111110 → 1's complement

+ 1

(-65) → 11111111 10111111 → 2's complement

88 → 00000000 01011000

11111111 10100111 → 1's complement

+ 1

(-88) → 11111111 10101000 → 2's complement

-65 → 11111111 10111111

-88 → 11111111 10101000

(+)

-153 → 1 1 11111111 01100111

carry bit

sign bit

Now the result is in overflow condition. To resolved this we need to find 2's complement

of this result again.

So,

```
1 1 1 1 1111 1 0 0 1 1 0 0 1 1 1   → overflow condition
0 0 0 0 0000 0 1 0 0 1 1 0 0 0     → 1's complement
                              + 1
-------------------------------
0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0 0 1   → 2's complement
```

~~Or~~ ~~10011001 check it, is it in decimal~~

Or, 10011001

sign bit(-)

Now convent this result to decimal which is -153 as we expected.

(ii) 34+55

```
+34 → 0 0 1 0 0 0 1 0
+55 → 0 0 1 1 0 1 1 1
                  +1
-------------------------
+89 → 0 1 0 1 1 0 0 1
```

sign bit(+)

∴ The result of 34+55 in 2's complement system +89 or 01011001.

Adder

Adder
- Half adder
- Full adder

~~Adder:~~

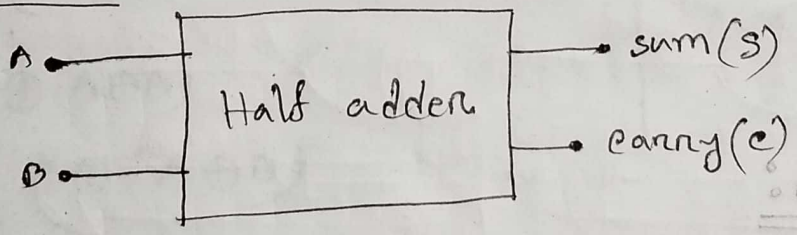
Decoder, Encoder, half adder, full adder

↳ related Q. exam এ আসবে Q2 5 step এ

বর্ণনা দিতে হবে, → (i) Defination
(ii) Block diagram
(iii) Truth Table
(iv) Boolean expression
(v) Logic gate.

(i) Half adder:

(ii) Block diagram:



(iii) Truth Table:

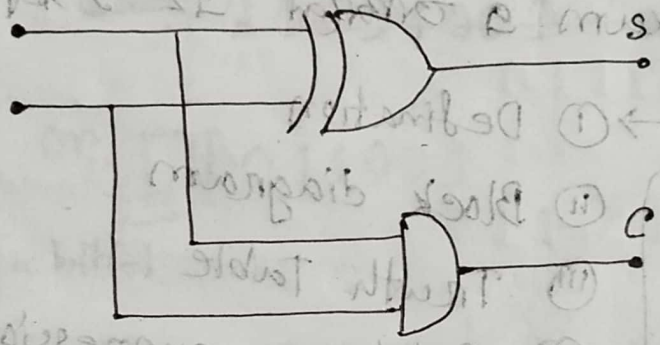
A	B	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

④ Boolean expression:

$$Sum = \bar{A}B + A\bar{B} = A \oplus B$$

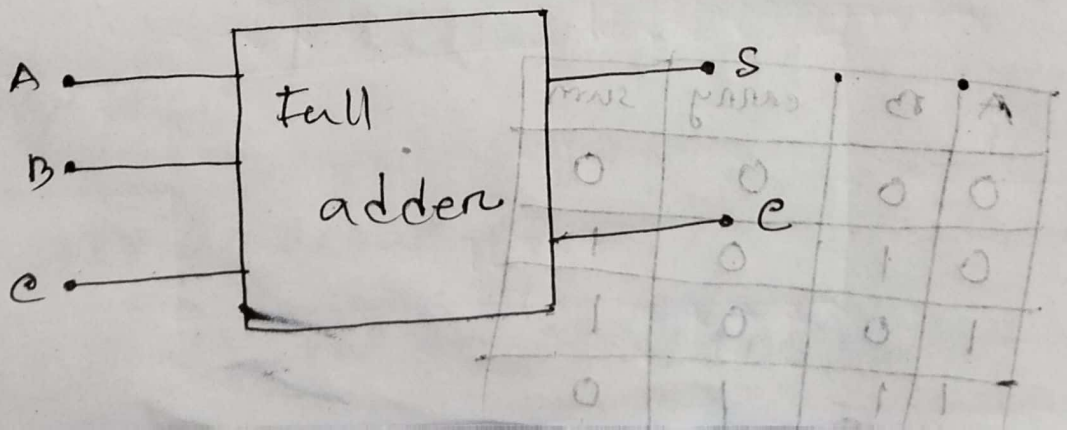
$$Carry = AB$$

⑤ Logic gate:



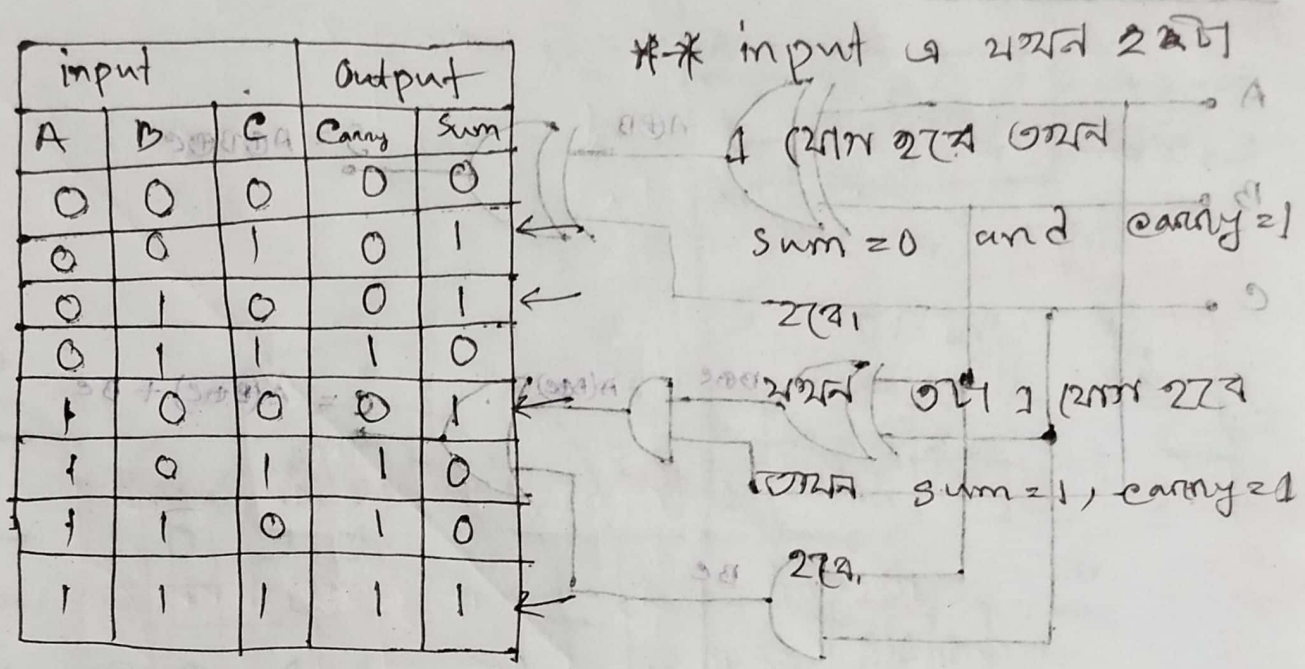
Full adder:

Block diagram:



Truth table:

input			Output	
A	B	C	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Boolean expression:

$$S = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$$

$$= \bar{A}B\bar{C} + A\bar{B}\bar{C} + \bar{A}\bar{B}C + ABC$$

$$= \bar{C}(\bar{A}B + A\bar{B}) + C(AB + \bar{A}\bar{B})$$

$$= \bar{C}(A \oplus B) + C(\overline{A \oplus B})$$

$$= C \oplus (A \oplus B)$$

$\therefore$ Sum $= C \oplus (A \oplus B)$

যদি, $x \oplus y = \bar{x}y + x\bar{y}$

$x = C$

$y = A \oplus B$

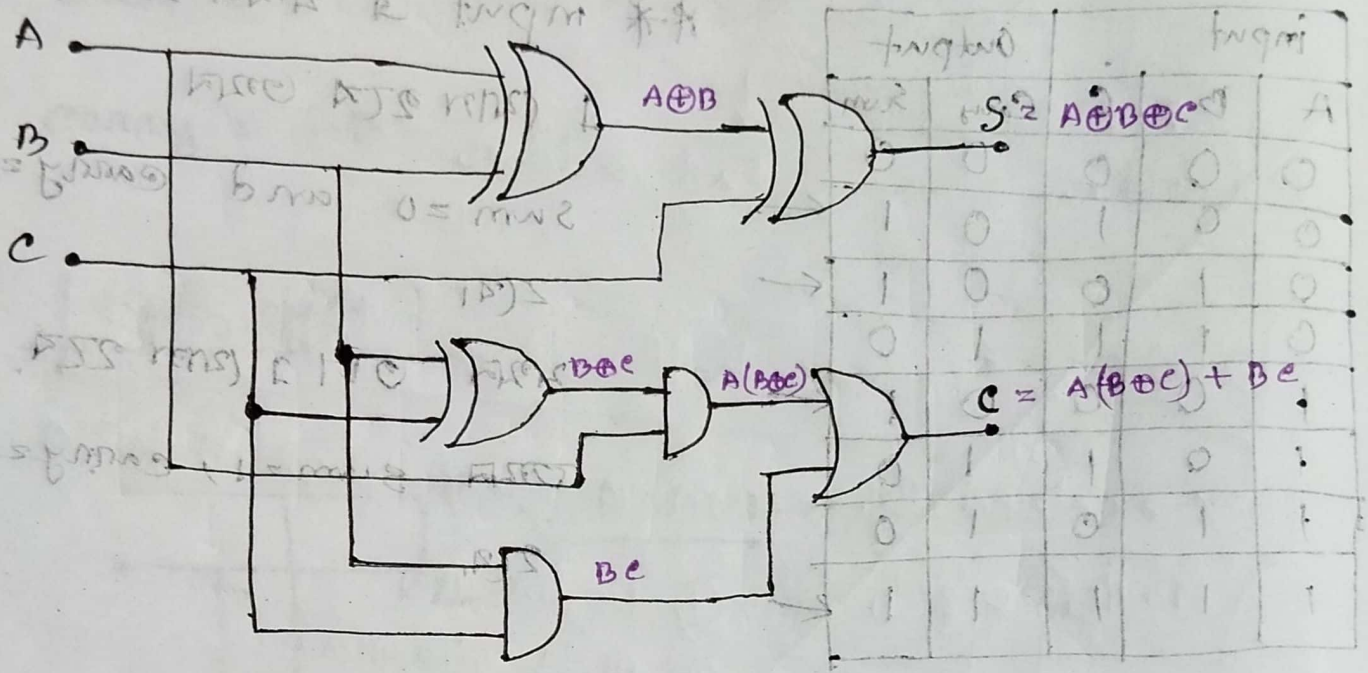
$$\text{Carry} = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$$

$$= A\bar{B}C + AB\bar{C} + \bar{A}BC + ABC$$

$$= A(\bar{B}C + B\bar{C}) + BC(\bar{A} + A)$$

$$= A(B \oplus C) + BC$$

Logic gate:

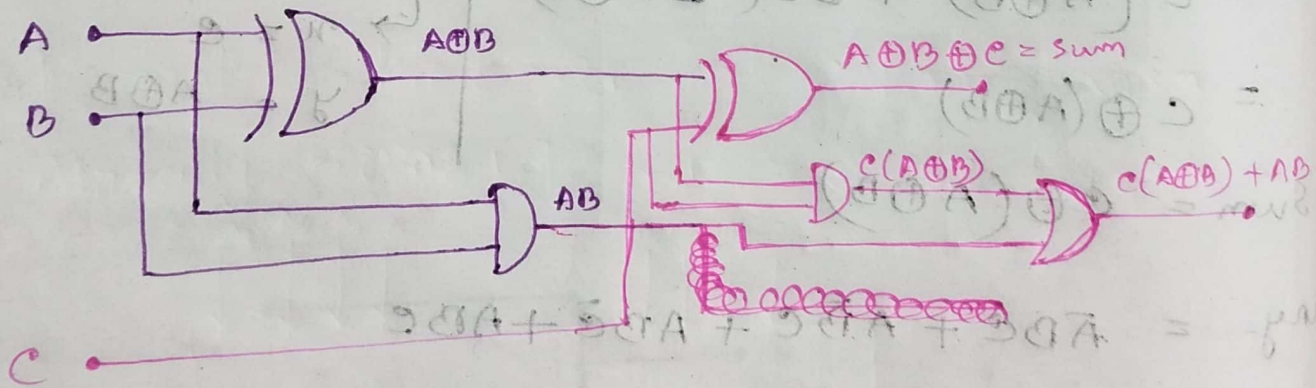


A	B	C	S = A⊕B⊕C
0	0	0	0
1	0	0	1
1	0	1	0
0	1	0	0
1	1	0	0
0	1	1	1
1	1	1	1

Logic gate of full adder.

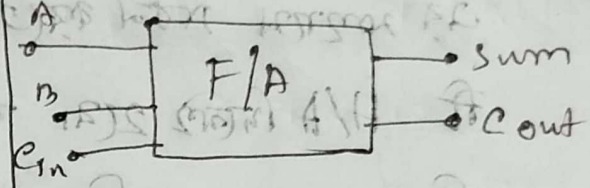
Half adder to full adder:

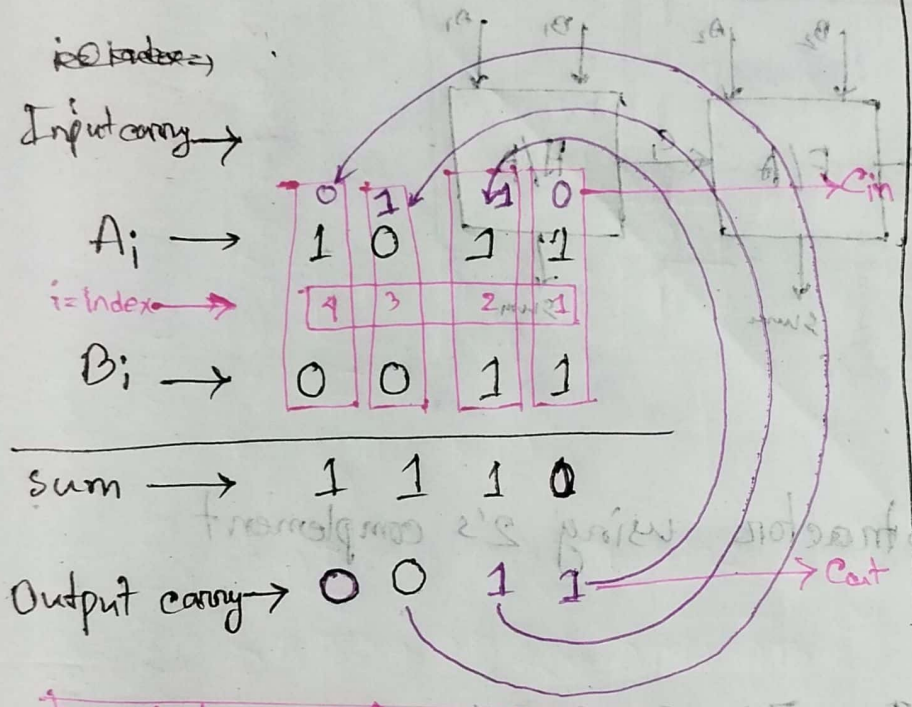
Full adder = 2 × Half adder + 1X OR gate

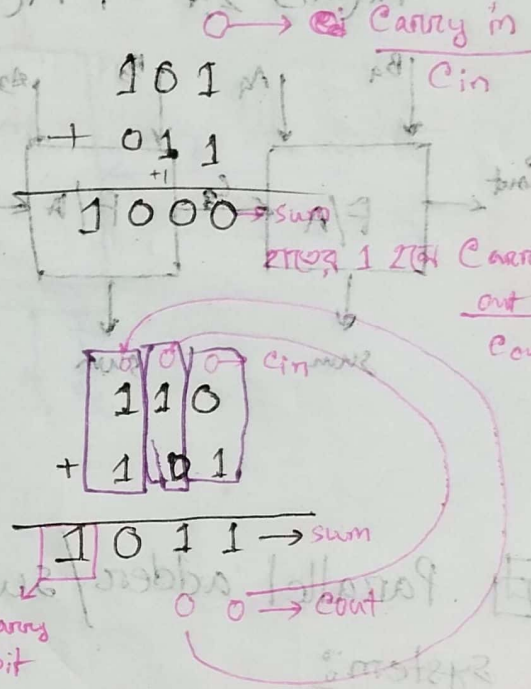


Parallel adder:

Basic





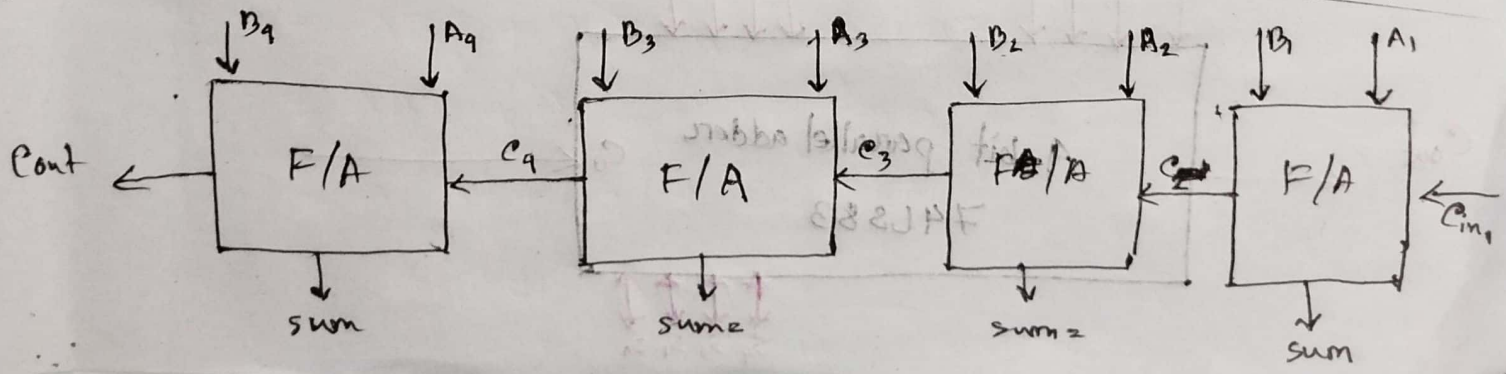


Parallel adder Mechanism.

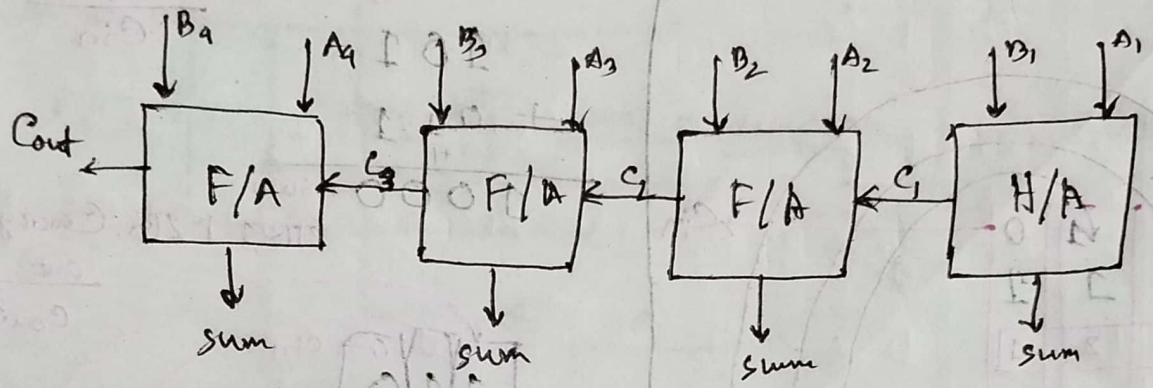
* n-bit ~~para~~ binary parallel adder এর জন্য n-bit full adder প্রয়োজন।

দুটি 2 bit এর যোগ করলে কিন্তু Finally তা 3 bit এ হয়ে। এবং প্রত্যেকটা Parallel. তাই একে Parallel ~~adder~~ বলে addition.

Block diagram: (4 bit parallel adder)



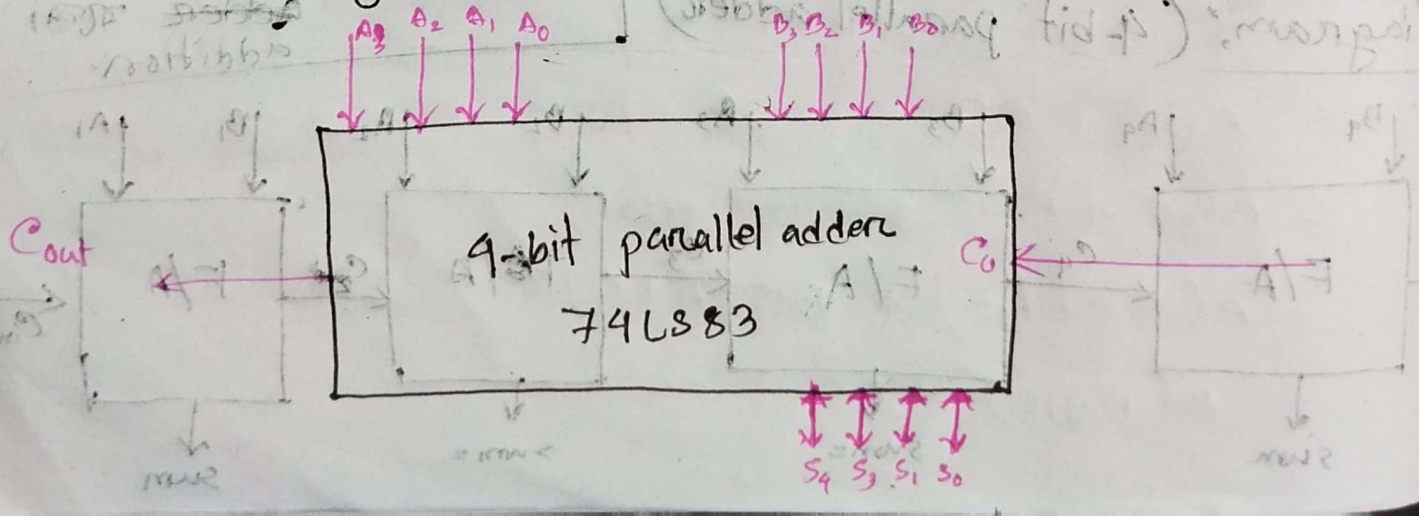
** যদি n-bit binary parallel adder H/A and F/A এর সাহায্যে গঠন করতে বলে, তাহলে মুটি প্রথম adder টি H/A দিলেই হবে। কারন প্রথমে $C_{in} = 0$ থাকে, তা দিলেও হয় না দিলেও হয়।



Parallel adder/subtractor using 2's complement system:

Understand the Ic 74LS83. It is a 4-bit parallel adder.

4-bit parallel adder এর internal mechanism Already দেখেছি। IC এর ক্ষেত্রে একটু ভিন্ন ভাবে দেখা

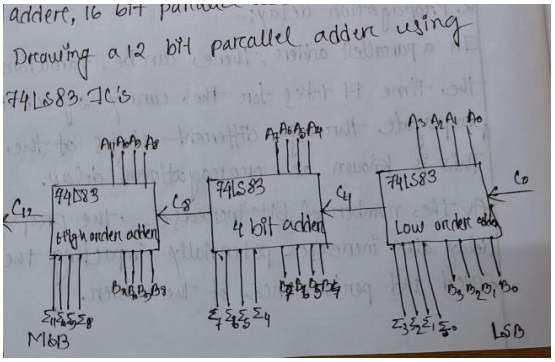


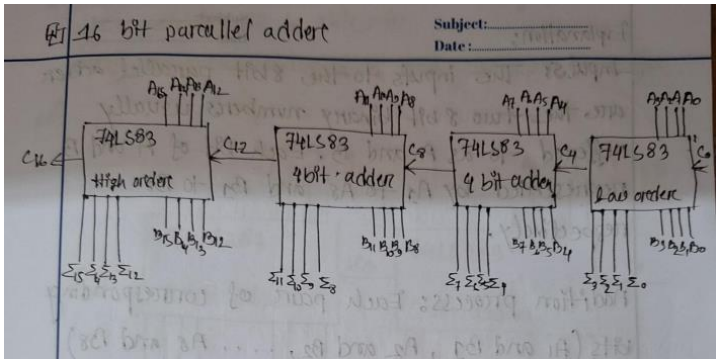
7. What is the limitation of a parallel adder? Using 74LS83 ICs draw a 16/12 bit parallel adder. (20-21, 18-19) [3]

Answer:

The main limitations of a parallel adder are:

1. **Carry Propagation Delay:** Each bit addition depends on the carry from the previous bit, which causes a delay as it moves through each stage of the adder. This delay increases with the number of bits, slowing down the adder.
2. **Complexity**: Larger adders need more gates and connections, increasing cost and size.
3. **Limited Scalability**: Due to the increased delay and complexity with each additional bit, parallel adders are not efficient for high-bit operations, such as in 32-bit or 64-bit processors.





8. Explain how 2's compliment system can facilitate arithmetic operation in digital computing. (20-21, 18-19) [3]

Answer:

The **2's complement system** is widely used in digital computing to represent negative numbers and simplify arithmetic operations, particularly subtraction. Here's how it facilitates these operations:

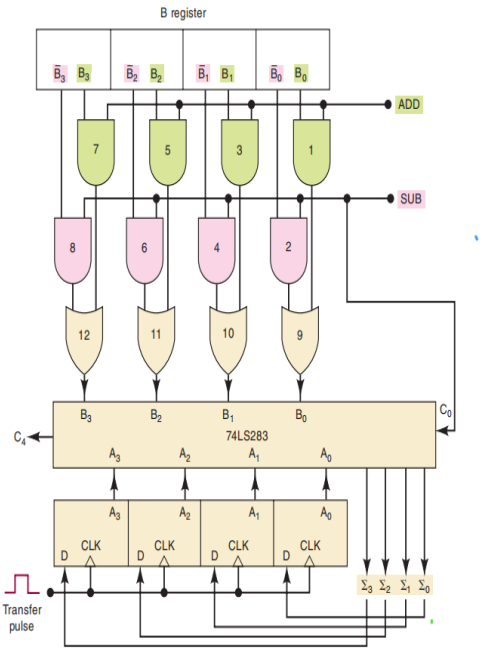
1. **Unified Addition and Subtraction**:
   - In 2's complement, subtraction can be performed as addition, simplifying the circuit design.
   - To subtract a number, simply add its 2's complement (invert all bits and add 1).
   - This means a single adder circuit can handle both addition and subtraction, making it efficient for hardware design.
2. **Single Representation of Zero**:
   - Unlike other signed number systems, 2's complement has only one representation for zero, reducing ambiguity in calculations.
3. **No Special Circuit Needed for Sign**:
   - Positive and negative numbers can be added directly without separate circuits to handle the sign, simplifying the hardware.
   - Overflow conditions are handled automatically when the sum exceeds the responsible range.
4. **Efficient Use of Bits**:
   - In a fixed number of bits, the 2's complement system maximizes the range of representable values, allowing both positive and negative values in a simple format.

9. Draw a parallel adder/subtractor using 2's complement system and explain its operation. (20-21) [6]

Answer:

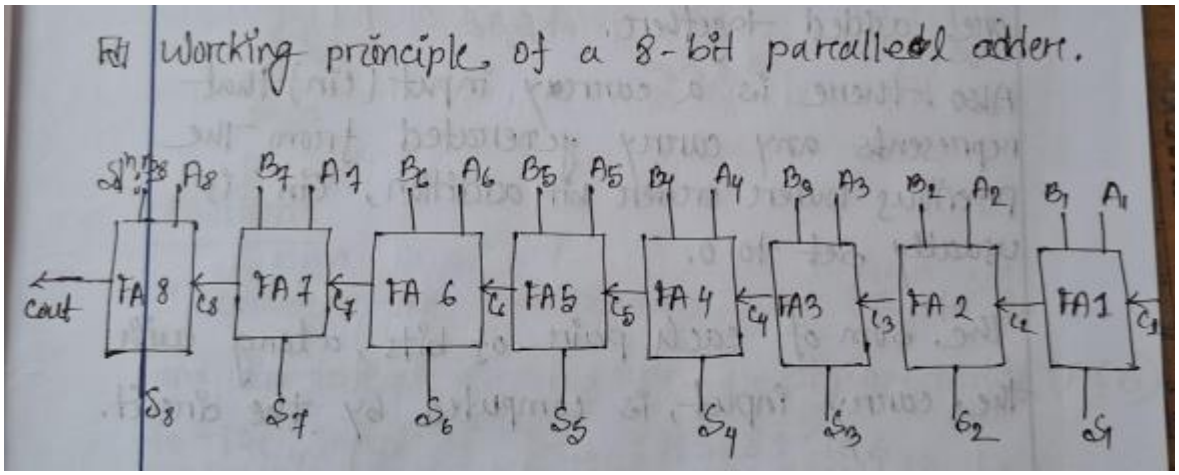
Here's a simplified description of the operation of the parallel adder/subtractor circuit:

1. **Addition Mode (ADD = 1, SUB = 0)**:
   - **Control Signal**: SUB = 0 disables certain AND gates (2, 4, 6, 8) to hold their outputs at 0, while ADD = 1 enables AND gates (1, 3, 5, 7) to pass the B_0 to B_3 values.
   - **Operation**: The B register values B_0 to B_3 pass through OR gates to the adder, where they are added to A_0 to A_3.
   - **Carry-In**: $C_0 = 0$, so no additional carry is added.
   - **Output**: The sum appears at outputs S_0 and S_3.
2. **Subtraction Mode (ADD = 0, SUB = 1)**:
   - **Control Signal**: ADD = 0 disables certain AND gates (1, 3, 5, 7), while SUB = 1 enables the other AND gates (2, 4, 6, 8) to pass B_0 and B_3 values.
   - **Operation**: The B register values B_0 to B_3 pass through OR gates to the adder, but with $C_0 = 1$, effectively converting B to its 2's complement.
   - **Output**: The adder performs A-B and the difference appears at outputs S_0 and S_3.



10. Design and explain the working principle of an 8-bit parallel adder. (19-20) [6]

Answer:



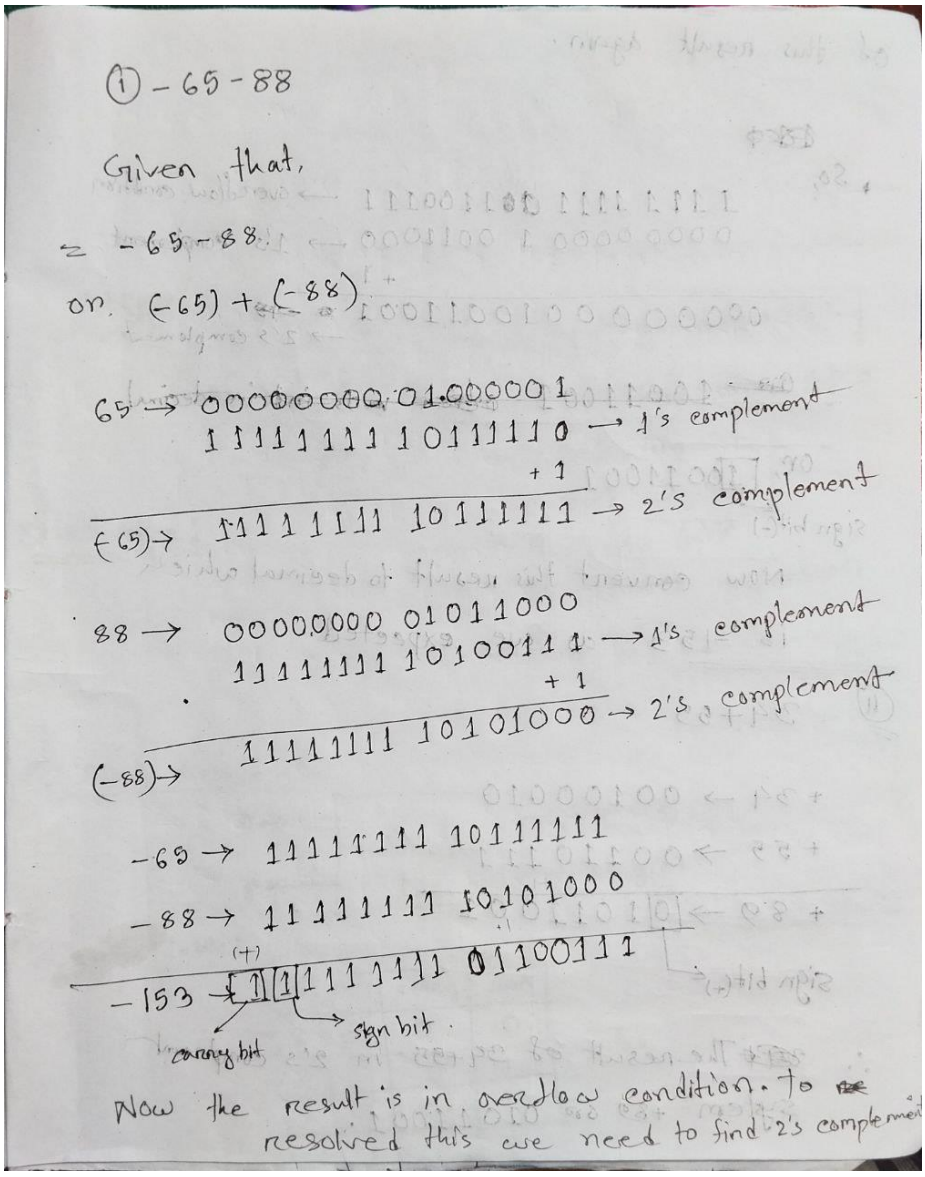
An 8-bit parallel adder is a digital circuit that performs the addition of two 8-bit binary numbers in parallel. It adds each pair of corresponding bits from the two operands simultaneously, producing an 8-bit sum output.

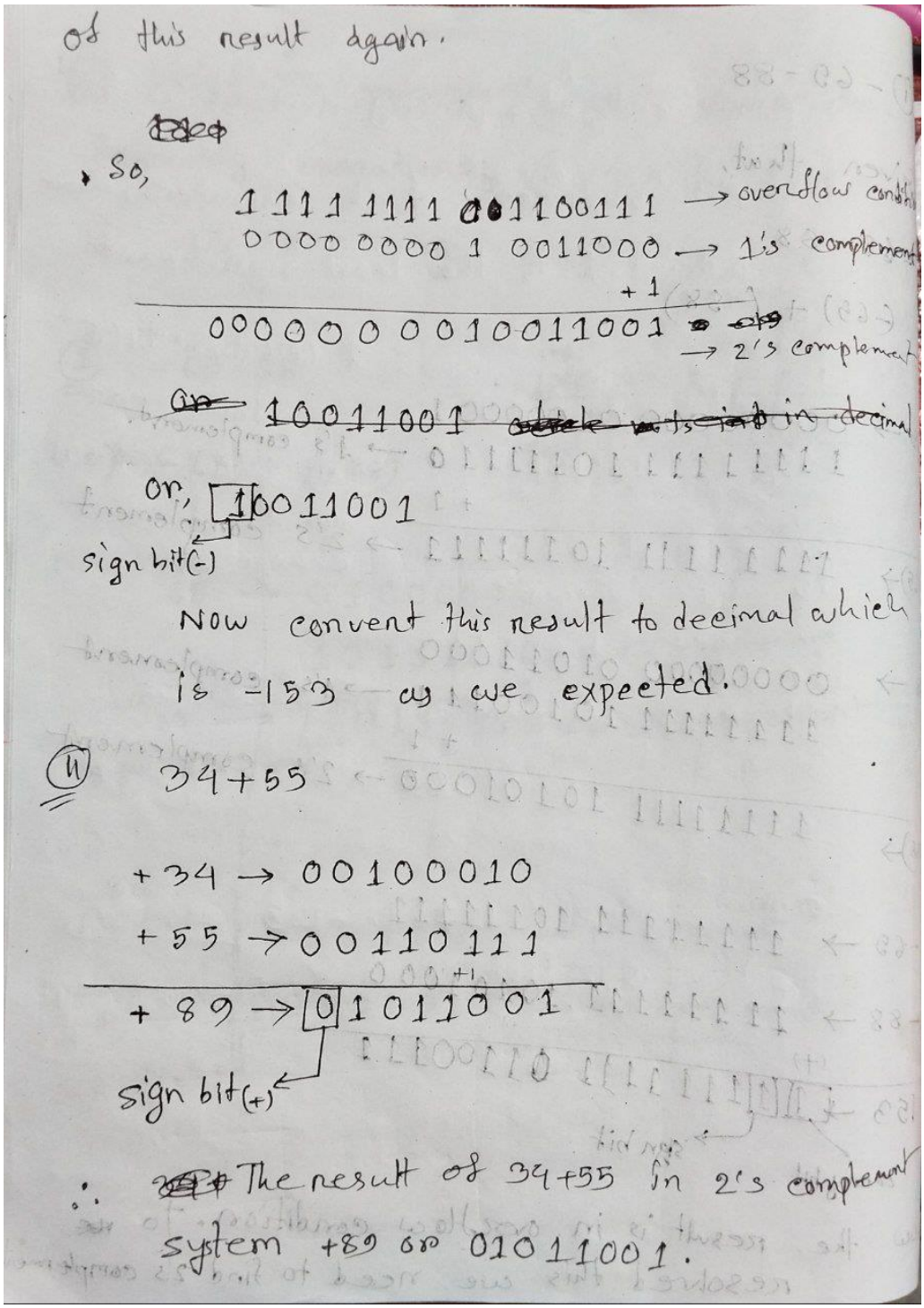
Inputs:
The inputs to the 8-bit parallel adder are two 8-bit binary numbers, typically referred to as A and B. Each bit of A and B is represented as A_0 to A_7 and B_0 to B_7, respectively.

Addition Process:
Each pair of corresponding bits (A_i and B_i) is added together. Additionally, a carry-in (C_{i-1}) from the previous bit addition is included in the calculation. The carry-in (C_{i-1}) is usually set to 0 for the least significant bit (LSB). The sum of each pair of bits, along with the carry input, is computed by the adder circuit.

Outputs:
The outputs of the 8-bit parallel adder are the 8-bit sum (S_0 to S_7), representing the result of the binary addition.

11. Using 2's complement system performs the following operations:(19-20) [2]
i) -65-88
ii) 34+55





of this result again.

So,

1 1 1 1 1111 0 1100111 → overflow carry

0000 0000 1 0011000 → 1's complement

+ 1

0000000 0010011001 → 2's complement

~~Or~~ 10011001 ~~check w.r.t. in decimal~~

Or, 10011001

sign bit(-)

Now convent this result to decimal which is -153 as we expected.

(ii) 34+55

+34 → 00100010

+55 → 00110111

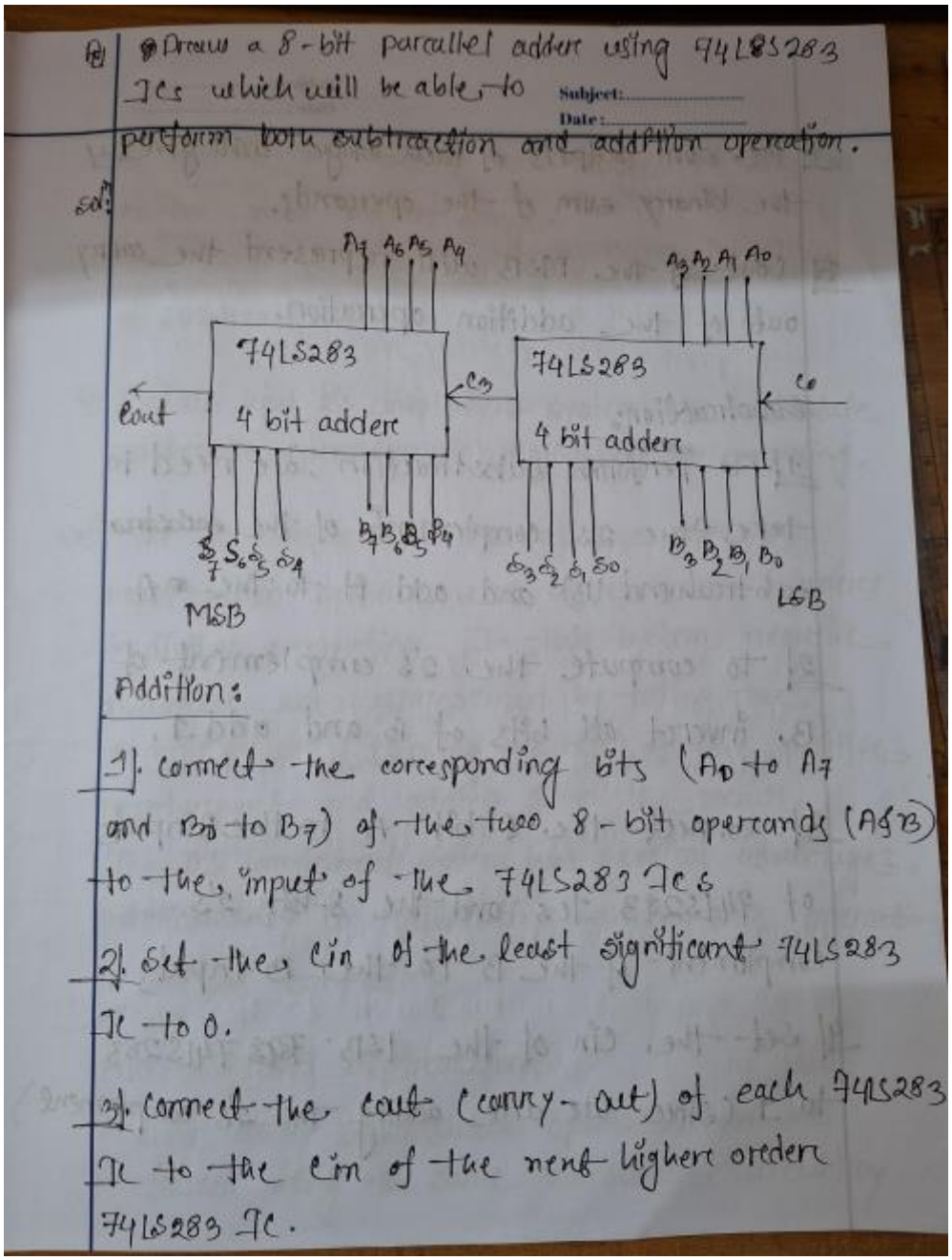
+89 → 01011001

sign bit(+)

∴ The result of 34+55 in 2's complement system +89 or 01011001.

12. Draw an 8-bit parallel adder using 74LS83 ICs, which will be able to perform both subtraction and addition operation. (19-20) [4]

Answer:



Subject:
Date:

4] The sum outputs of each stage will give us the binary sum of the operands.

5] Cout of the MSB will represent the carry out of the addition operation.

Substraction:

1] To perform substraction, we need to take the 2's complement of the ~~substra~~ subtrahend (B) and add it to the A

2] to compute the 2's complement of B, invert all bits of B and add 1.

3] connect the 8-bit A to the A inputs of 74LS283 ICs and the 8-bit 2's complement of the B to the B inputs

4] Set the Cin of the LSB ~~7~~ 74LS283 to 1 (since we are adding the 2's complement)

Subject:
Date:

5] Connect the Cout of each stage to the Cin of the next higher stage.

The sum outputs will give us the result of substraction.