Micro Computer And Logic Design
Past papers

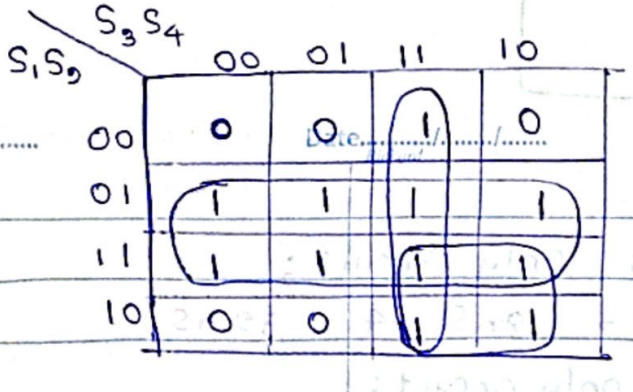
2007 | 2008.

Q1 : (a)

S_1	S_2	S_3	S_4	Output
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

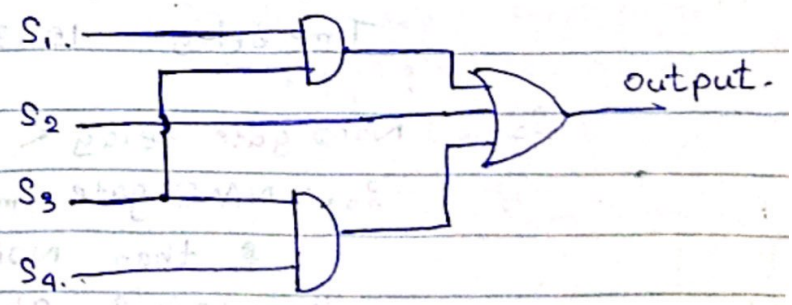
RICHARD

(b)

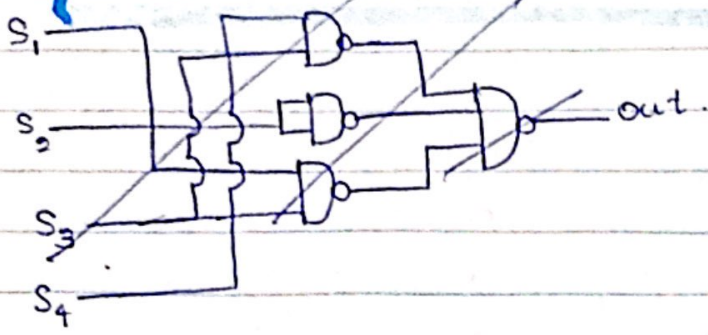


$S_2 + S_3S_4 + S_1S_3$

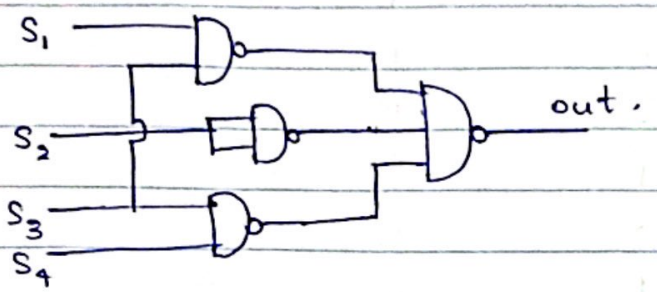
Digital circuit :



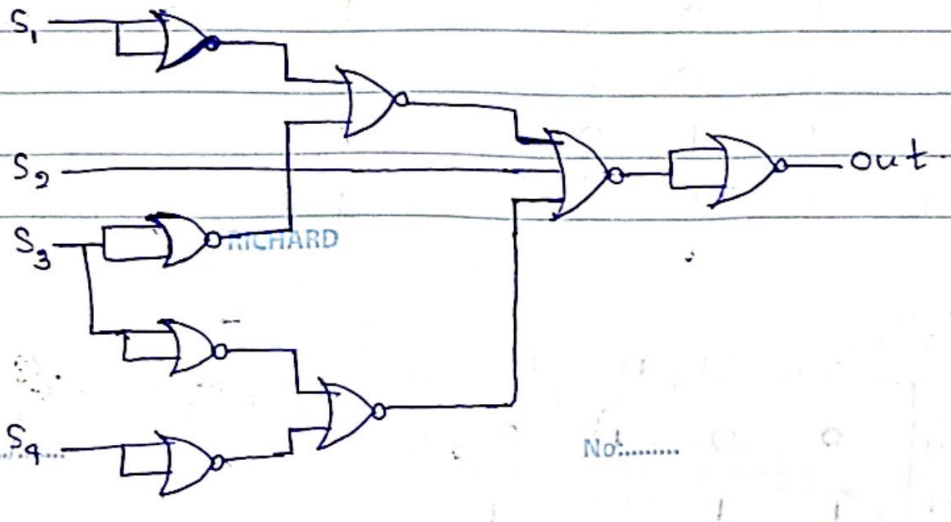
(c) (i) NAND gates Only.



(c) (ii) NAND gates only



(ii) NOR gates only.



(d) Using NAND gates Only Circuit;

Time delay = 8nS x 4 = 32 nS.

NOR gates only circuit;

Time delay = 12nS x 8 = 96 nS

So; Nand gate delay < NOR gate Delay.

∴ NAND gate implementation is faster than NOR gate implemation.

Ans: 100% 96 - 32 = 64nS//

) $A + AB$

$= A(1+B)$

$= A \times B$

$=$

(e) Distributive Law;

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

$A + AB = A(1+B)$

Identity Law;

$A \cdot 1 = A$.

$A + AB = A \cdot 1$

Absorption Law;

$A + AB = A$

Expression 1: $A + AB = A$.

Expression 2: $A + B$

Since $A + AB = A$ and $A + B$ is also A. we can conclude that $A + AB$ is indeed equal to $A + B$

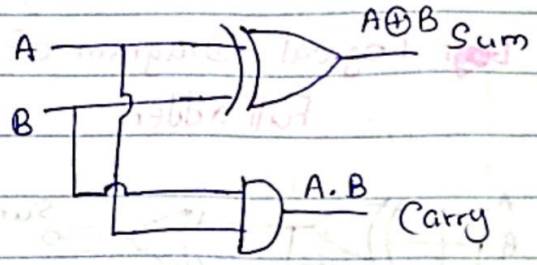
Truth Table.

Lets see how its work.

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

When both a inputs are 1 Then there is a carry bit. So Carry get Output as 1.

Logic diagram of Half Adder.



Q2 (a) Operation of a half Adder.

A Half adder is a basic logic circuit that performs addition on two binary digits and generates two outputs.

That out puts called as Sum and Carry.

It can only handle the addition of two single bits and does not handle any carry from previous. Because why this Call half adder.

(b). Operation of a Full Adder.

Full adder is used for adding two binary digits along with a carry input from a previous addition.

So there are three inputs. first input bit, second input bit and Carry bit.

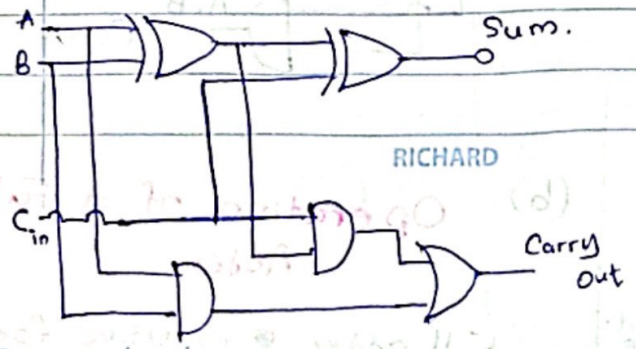
Outputs are Sum and. Carry output.

Truth Table for Full Adder.

A	B	C_{in}	Sum	Carry.
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

If 3 inputs are 1, Then There is one Sum and one for Carry.

Logical Diagram of Full adder.



Date...../...../.....

How Full adder Construct with two-half adders?

- Construct the First Half adder to generate the initial Sum and Carry outputs
- Construct the second Half adder to combine the initial Sum with Carry input to produce the final Sum and carry out.
- Use an OR gate to combine the Carry outputs from the first Half adder and the second Half adder to produce the overall Carry out of the Full adder.

Why need a Full adder.

There is no provide Carry-input from the previous circuit when adding together multiple data bits in Half adder.

Q3:

(a) Operation of XOR gate.

The XOR gate produce High (1) output if the ~~number~~ two inputs are not equal.
If the inputs are Same. then XOR gate produce Low (0) output.

No:..... Date...../...../.....

Symbol:

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Q3: (b) Method 01.

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

A	$\bar{A}$	B	$\bar{B}$	AND	NAND	OR	NOR	XOR
0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	1
1	0	0	1	0	1	1	0	1
1	0	1	0	1	0	1	0	0

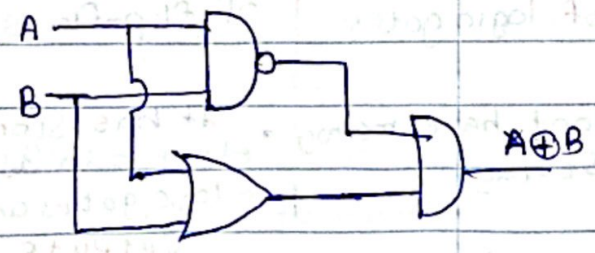
NAND • OR = XOR

Method 01.

$A \oplus B = \overline{AB} \cdot (A+B)$ (De' Morgan)

$= (\bar{A}+\bar{B}) \cdot (A+B)$

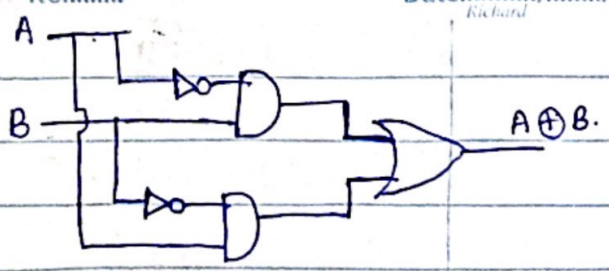
Circuit:



Method 02.

$(\bar{A} \cdot B) + (A \cdot \bar{B}) = A \oplus B.$

Circuit:



(c)

$2 \mid 15$

$2 \mid 7 - 1$

$2 \mid 3 - 1$

$1 - 1$

$= 1111$

0.758_{10}

$0.758 \times 2 = 0.516 \quad - 1$

$0.516 \times 2 = 0.032 \quad - 1$

$0.032 \times 2 = 0.064 \quad - 0$

$\therefore 0.758_{10} = 1111.110_2$

(d)

$44 - 57$

$44 \rightarrow 00101100$

$-57 \rightarrow 00111001$

$\downarrow$

11000110

$+ \quad 1$

11000111

00101100

$+ 11000111$

11110011

Q5:- (a) Explain the Operation of a JK-flip-flop.

J	K	CLK	Q	
0	0	↑	Q_0	No change
0	1	↑	1	Reset
1	0	↑	0	Set
1	1	↑	Q_0	Toggles.

J, K both inputs low then there are no changes.

J, K both inputs high then toggle from one state to other.

J, K are different inputs then the output Q takes the value of J at the next clock edge. (Set / Reset)

Difference between JK flipflop & SR flip-flop.

JK	SR
• Inputs: J, K, Clock	• Inputs: S (set) R (Reset) clock.
• Can be used for; toggling, setting or resetting	• Can be used for; setting and resetting.
• doesn't have an invalid state.	• has invalid state.

Q5 (b) ?

Q6: (a) ?

(b) ?

(c). Different between Combinational and Sequential

Combinational logic circuit	Sequential Logic Circuit
• outputs are determined from present inputs.	• Outputs are determined from both present and Past inputs.
• Faster and better than sequential.	• Slower and Low performences
• Inter Connection of logic gates	• Inter Connection of flip-flops.
• dont have memory Storage	• It has Storage element in addition to logic gates and Outputs

(d)

2018/2019.

01) a) **Boolean expression** is a mathematical or logical expression that uses Boolean operators to represent and manipulate binary values.

Truth table is a tabular representation used to show the possible combinations of input values for a Boolean expression and the resulting output values.

b) **Advantages of Boolean expression Simplification.**

- Reduced complexity
- Improved performance.
- Easier debugging
- Minimized cost
- Optimize memory usage
- Optimized design process.

c) ii) $\overline{a \cdot b}((\overline{a} \cdot b) \cdot (\overline{a} \cdot c))$

$= ((\overline{a \cdot b}) \cdot (\overline{a \cdot c}))$

$= (\overline{\overline{a}} + \overline{b}) \cdot (\overline{\overline{a}} + \overline{c})$

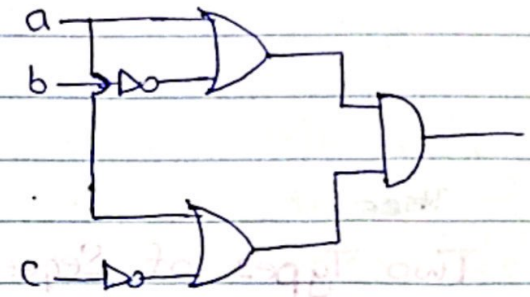
$= ((a + \overline{b}) \cdot (a + \overline{c}))$

$= (a + \overline{b}) \cdot (a + \overline{c})$

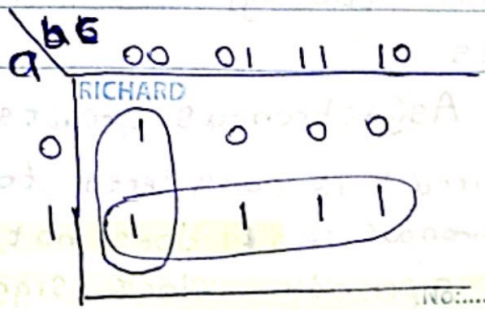
i. Truth Table

a	b	c	$\overline{b}$	$\overline{c}$	$(a+\overline{b}).(a+\overline{c})$
0	0	0	1	1	1
0	0	1	1	0	0
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	1	1	1
1	0	1	1	0	1
1	1	0	0	1	1
1	1	1	0	0	1

(iii)

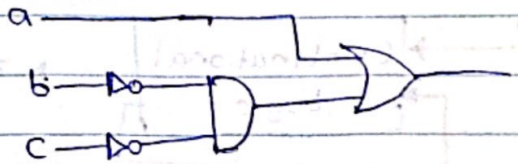


(iv)



$\overline{b}\,\overline{c} + a$

(v)



02. a. i. A circuit is a closed path or loop through which electric current can flow.

ii.

Transistors.	Active.
Diodes.	
Op-Amps.	
Integrated Circuits	
Resistors	Passive.
Capacitors	
Connectors.	
Transformers	

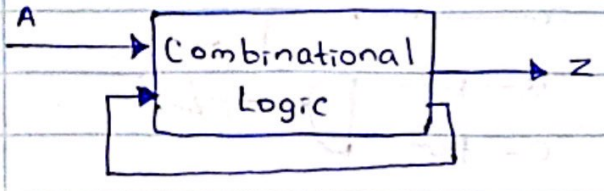
b. i. ~~There~~

Two Types of Sequential Circuits.

There are two types of sequential circuits.

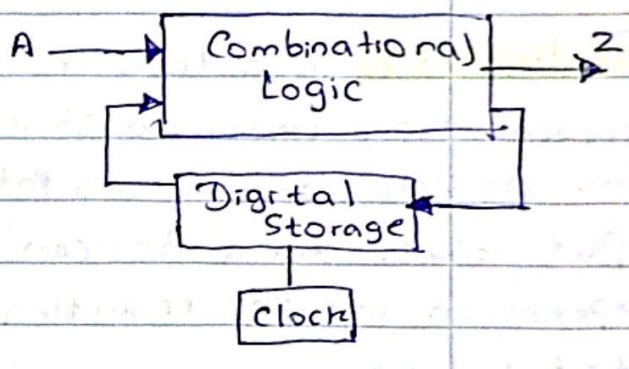
01. Asynchronous Circuits.

The circuit is considered to be asynchronous if it does not employ a periodic clock signal to synchronize its interal changes of state.



02. Synchronous circuits

The input are pulses with certain restrictions on pulse width and circuit propagation delay. Use same clock pulse.



ii. The draw back of the SR flip flop, namely the race coditron, is overcome by using the D-flipflop, which has a sigle data input and updates its output only on clock transitions. The ensuse more predictable and synchronized behavior in sequential circuits.

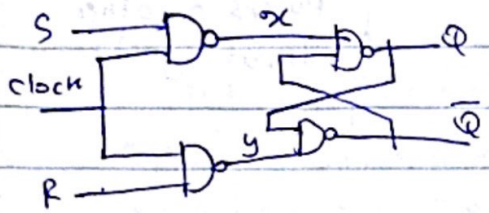
iii. Memory States of SR flip flop by providing Examples.

SR flipflop has Three inputs. They are S, R and clock. This flipflop made with Nand gates. So lets se how the truth table ... Ne can

R	C	x	y	Q	$\bar{Q}$	
1	0	1	1			memory state
0	1	1	1	0	1	
1	0	1	0	1	0	
0	0	1	1	0		mem
1	1	1	0	1	0	

If x ,y got Output=1

Then it is a memory state.



03. a. Microprocessor.

The processor and control unit part of the single-chip computer.

Three types of information microprocessor can carry

- Data
- Instructions
- Control Signals.

b. Six types of instructions which can be supported by 8086 architecture.

- Data transfer Instructions
- Arithmetic Instructions
- Logical Instructions
- Control transfer Instructions
- Comparison Instructions.
- String Instructions.

2016/2017.

01) a) Positive Logic System

In a positive logic system, logical operations are based on the assumption that high voltage levels or logical "1" values represent true/active conditions, low voltage level or logical "0" represent false/inactive condition.

Negative Logic System.

Low voltage levels or logical "0" values represent true or active conditions, while high voltage levels or logical "1" values represent false or inactive conditions

b) $xy + x(wz + wz')$

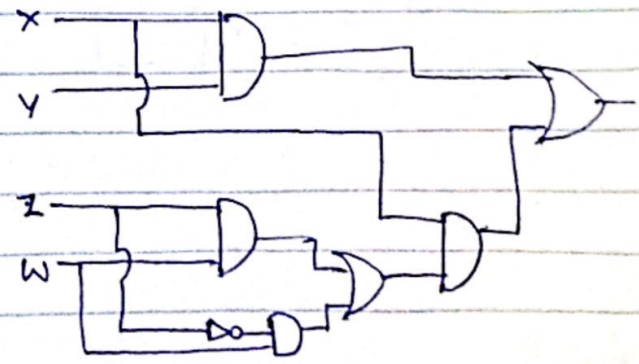
$= xy + x(w(z+z'))$

$= xy + x(w \times 1)$

$= xy + xw$

$= x(y+w)$

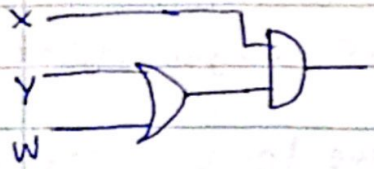
c). original.



02. a. i

Simplyfy

$x(y+w)$



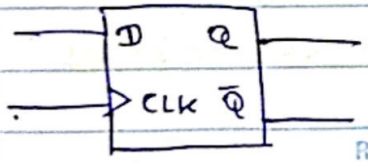
*d. $F = xy + x'z$

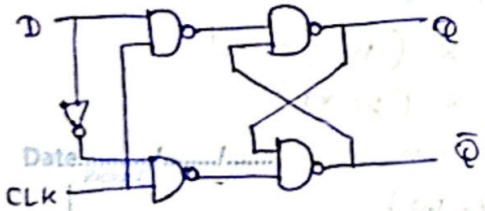
$= (x+y)(x'+z)$

2. a. What is sequential circuit?

Sequential circuits are combination of combinational circuit and memory It depends on the present input as well as past output.

b. D - Flip Flop.





4. b. Differentiate a register and a counter.

Register	Counter.
• Use for data storage and manipulation	• use to keep track of events or sequences by counting and generating output patterns.
• Consists of a group of flip-flops	• based on clock. Pulses or other input signals

2016

Part - A

1. a. State the DeMorgan's theorem

The complement of the product of two or more variables is equal to the sum of their individual complements.

$\overline{A \cdot B \cdot C} = \overline{A} + \overline{B} + \overline{C}$

The complement of the sum of two or more variables is equal to the product of their individual complements.

$\overline{A+B+C} = \overline{A} \cdot \overline{B} \cdot \overline{C}$

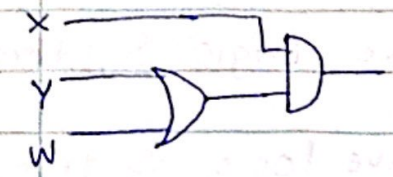
b. i. $AB + A(B+C) + B(B+C)$

~~$AB + A(B+C) + B + BC$~~

$AB + AB + AC + BB + BC$

Simplyfy

$x(y+w)$



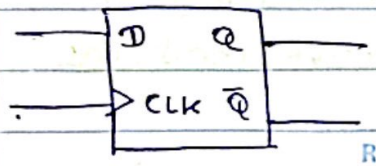
d. $F = xy + x'z$

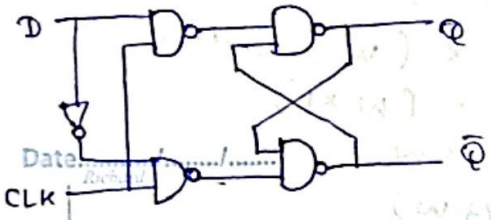
$= (x+y)(x'+z)$

2. a. What is sequential circuit?

Sequential circuits are combination of combinational circuit and memory. It depends on the present input as well as past output.

b. D - Flip Flop.





4. b. Differentiate a register and a counter.

Register	Counter.
• Use for data storage and manipulation	• Use to keep track of events or sequences by counting and generating output patterns.
• consists of a group of flip-flops	• based on clock. Pulses or other input signals

2016.

Part - A

1. a. State the De Morgan's theorem

The complement of the product of two or more variables is equal to the sum of their individual complements.

$\overline{A \cdot B \cdot C} = \overline{A} + \overline{B} + \overline{C}$

The complement of the sum of two or more variables is equal to the product of their individual complements.

$\overline{A+B+C} = \overline{A} \cdot \overline{B} \cdot \overline{C}$

b. i. $AB + A(B+C) + B(B+C)$

~~$AB + A(B+C) + B + BC$~~

$AB + AB + AC + BB + BC$

$AB + AC + B + BC$

$AB + B + AC + BC$

~~$B(A+1) + C(A+B) \quad AB+B$~~

$AB + B + C$

$B + C$

II. $A + AB + A\bar{B}C$

$A + A\bar{B}C$

$A(\bar{B}C + 1)$

$A \times 1$

A

III. $A + (A + \bar{C})(A + B)$

$A + AA + AB + \bar{C}A + \bar{C}B$

$A + A + AB + \bar{C}A + \bar{C}B$

$A + AB + \bar{C}A + \bar{C}B$

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

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

$A(1 + \bar{C}) + \bar{C}B$

$A + \bar{C}B$

C. i. $A\bar{B} + AB$

A \ B	0	1
0		
1	1	1

$= A$

ii. $\bar{A} + A\bar{B} + AB\bar{C}$

C \ AB	00	01	11	10
0	1	0	1	1
1	0	0	0	0

AC

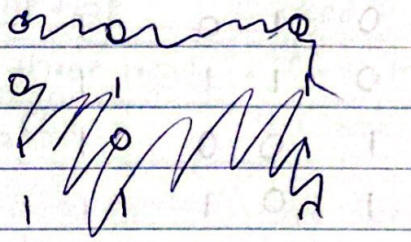
(iii) $\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

C \ AB	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$= BC + AB + AC$

(d)

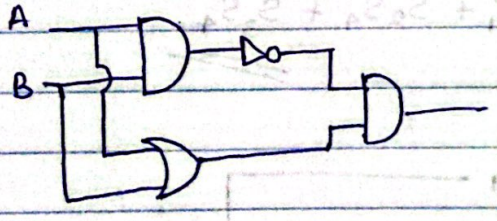
A	B	output
0	0	0
0	1	1
1	0	1
1	1	0



(d) ~~$A\bar{A}B\bar{B}$ AND OR XOR~~

A	$\bar{A}$	B	$\bar{B}$	AND	NAND	OR	NOR	XOR
0	1	0	1	0	1	0	1	0
0	1	1	0	0	1	1	0	1
1	0	0	1	0	1	1	0	1
1	0	1	0	1	0	1	0	0

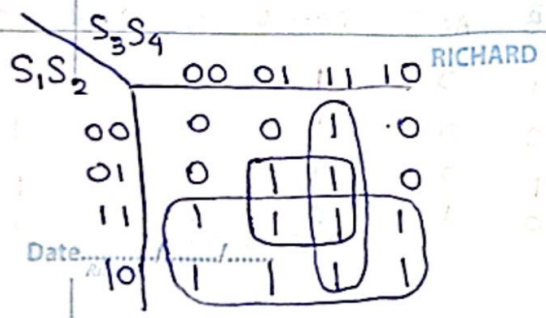
So;



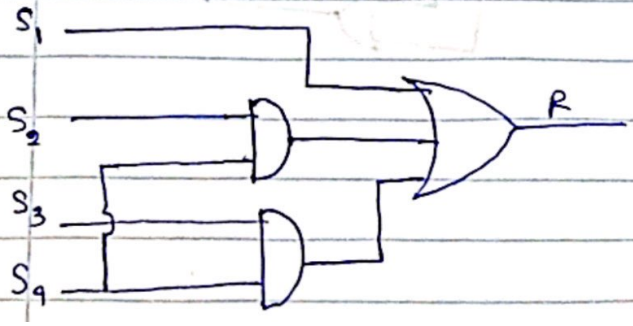
02. a. i.

S_1	S_2	S_3	S_4	R
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

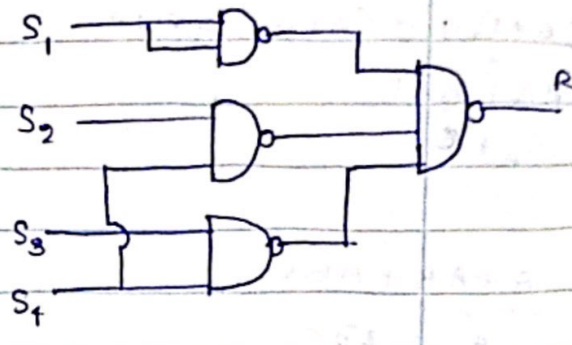
ii.



$S_1 + S_2S_4 + S_3S_4$



iv.



3.

b. i.

Sequential	Combinational

Answer - 2007/2008 Paper

ii.
- Asynchronous
- Synchronous.

c. Importance of Synchronization.
- Data integrity and Consistency
- Orderly collaboration
- Security and Authentication.
- Distributed computing

Q3. a. i. Latch.

A latch is a circuit that has two stable states and can be use to store state information.

ii. Difference between active High & active-low. in SR Latch.

active - high	active - low.
• ~~Both inputs are normally tied to ground (Low)~~	• ~~Both inputs are normally High~~
• When S input is active high it sets the latch to the "set" state	• When S input is active low it sets the latch to the "set" state.
• When R input is active high it resets the latch to the "reset"	• When R input is active low it reset the latch to "reset" state.
• Both inputs are high latch is undefined.	• Both inputs is active low latch is undefined.

D Latch	SR Latch.
• has a single data input (D) and a control input.	• has two inputs set and reset.
• Commonly used for temporary data storage and Synchronization	• basic memory used for elements and simple control functions.
• edge-Sensitive.	• level Sensitive

b. i.

J-K flip flop	clocked D flip flop
• has three inputs (set) (reset) (clock) J, K, CLK.	• Two inputs D (data) and CLK
• Used in applications where toggling functionality is required, such as counters, frequency divides ---	• Used for simple data storage, synchronization and state storage applications

RICHARD

(iii)

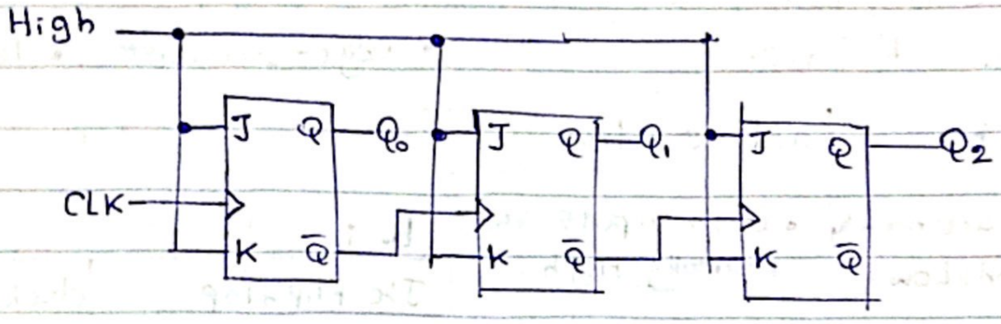
(iv) Difference between SR Latch and D Latch.

→

Q4. a. i. Ripple counters.

A ripple counter is an asynchronous counter wh- only the first flip-flop is clocked by an external clock.

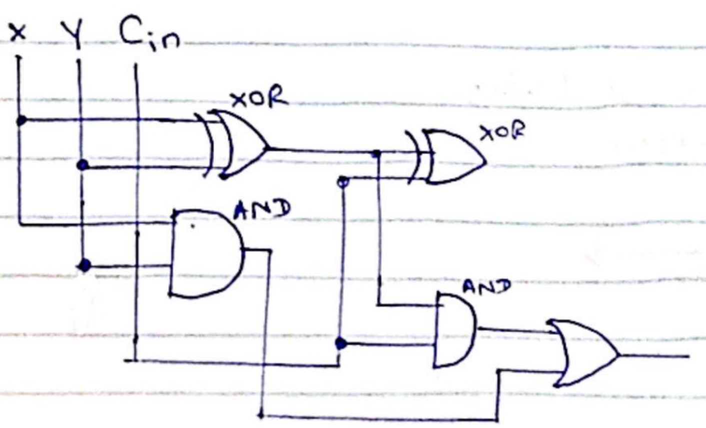
ii. Three bit asynchronous ripple counter using J-K flip-flops.



c. i. Why we need full adder and half adder.

- While half adders can perform basic single-bit addition, they lack the ability to handle carry inputs from previous stages.
- Full adders, on the other hand, can handle both the current addition and carry inputs, ~~making~~.

ii. Full adder using two half adders



Steps

- 1. x y දෙකම AND 1ක් සහ XOR එකක් යවයි.
  2. එම XOR එකේ output 1 සහ Cin භාවිතයෙන් නව XOR 1ක් යවයි.
  3. එම XOR (x, y දෙකම ඇති) output 1ක් සහ Cin භාවිතයෙන් AND එකක් යවයි.
  4. එම AND සහ පෙර x, y දෙකේ AND, OR gate එකක් හරහා යවයි.

Multiplexers	Demultiplexers.
• selects one of many input lines and forwards its data to a single output line.	• takes a single input and forwards it to one of several output lines
• multiple data inputs and One Output	• One input and multiple outputs.

2015.

1. a. i. 10101.101

$10101 = 1\times1 + 4\times1 + 16\times1 = 21$

$.101 = 2^{-1}\times1 + 2^{-3}\times1$

$= 0.5 + 0.125 = 0.625$

∴ Answer = 21.625_{10}.

(ii) 11.375

$2 \mid 11$
$2 \mid 5 - 1$
$2 \mid 2 - 1$
$1 - 0$

$0.375 \times 2 = 0.750 \quad — 0$
$0.75 \times 2 = 1.50 \quad — 1$
$0.5 \times 2 = 1.0 \quad — 1$

$= 1011.011_2$

b. i. $x - y$

$= 01101 - 01001 = 01101 + (-01001)$

$x = 00001101$
$y = 00001001$
$= 11110110$
$+ \quad 1$
11110111

00001101
$+ \ 11110111$
$\times 1 \ 00000100$

Answer = 00000100_2

c. 568

binary numer of 5 = 0101

binary number of 6 = 0110

binary number of 8 = 1000

So BCD cod of 568 = 0101 0110 1000 //

d. $9 = 1001_2$

leftmost bit = 1

1 XOR 0 = 1

0 XOR 0 = 0

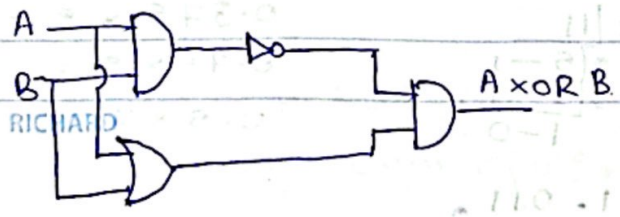
0 XOR 1 = 1

Answer = 1101 //

2. a. i.

A	B	$\bar{A}$	$\bar{B}$	AND	NAND	OR	NOR	XOR
0	0	1	1	0	1	0	1	0
0	1	1	0	0	1	1	0	1
1	0	0	1	0	1	1	0	1
1	1	0	0	1	0	1	0	0

So



ii. $(\overline{A \cdot B}) \cdot (A+B)$

b. i. $C + (BC)'$

$(B'+C)(C'+C)$ Distributive Law

$= B'+C \cdot 1$ Complementary Law.

$= B'+C$

(ii) $(AB)'(A'+B)(B'+B)$

$(AB)'(A'+B)$ 1 Complem La

$(\bar{A}+\bar{B}) \cdot (\bar{A}+B)$ (Distributive

$\bar{A}\bar{A} + \bar{A}B + \bar{A}\bar{B} + \bar{B}B$ (complemen..

$0 + \bar{A}B + \bar{A}\bar{B} + 0$

$\bar{A}B + \bar{A}\bar{B}$

(iii) $(A+C)(AD + AD') + AC + C$

$= (A+C)\,A(D+D') + C(A+1)$ (complementary)

$= (A+1)(A \cdot 1) + C(A+1)$ (union Law)

$= (A+C)(A \cdot 1) + C \cdot 1$ (Intersection Law)

$(A+C)A + C$ → Distributive $AA + AC + C = A + C(A+1)$ (union)

a. i. SR Flip-Flop, JK Flip-Flop, D Flip-flop, T Flip-Flop

ii Functionality of flipflops.

SR Flip-Flop – The SR flip-flop can be in one of two stable states: SET ($Q=1$) or RESET ($Q=0$)

JK Flip-Flop – The JK Flip-Flop can be in one of two stable states: SET ($Q=1$) or RESET ($Q=0$). It has the additional feature of toggling its output.

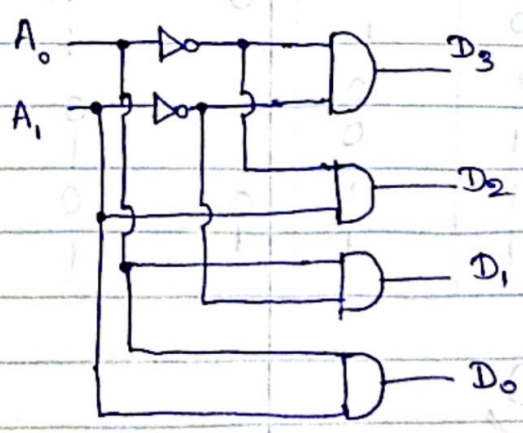
D Flip-Flop – The D-Flipflop stores the value of the data input (D) at the clock edge.

T Flip-Flop – The T flip-flop toggles its output between its complement and it self on each clock edge when the T input is active.

b- i.

A_1	A_0	D_3	D_2	D_1	D_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

$$\begin{aligned} D_3 &= \overline{A_1} \cdot \overline{A_0} \\ D_2 &= A_1 \cdot \overline{A_0} \\ D_1 &= \overline{A_1} \cdot A_0 \\ D_0 &= A_1 \cdot A_0 \end{aligned}$$



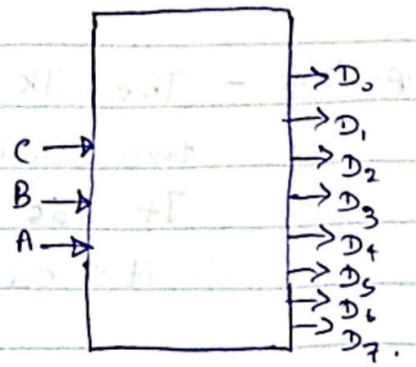
2010

01) I. Combinational Circuits.

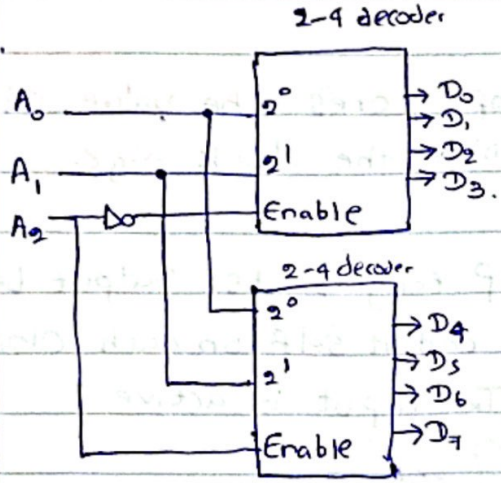
Combinational logic refers to circuits whose output is a function of the present value of the inputs only.

II Multiplexer.
Divider.
Encoder
Adders.

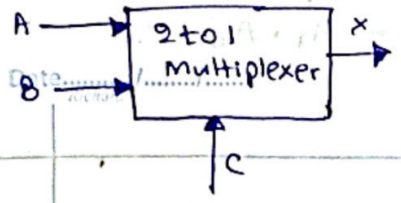
III.



III.



IV i. Truth Table for 2 to 1 Multiplexer.



~~(A·C)+(B·C)~~

~~X = (A·C)+(B·C)~~

C	A	B	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

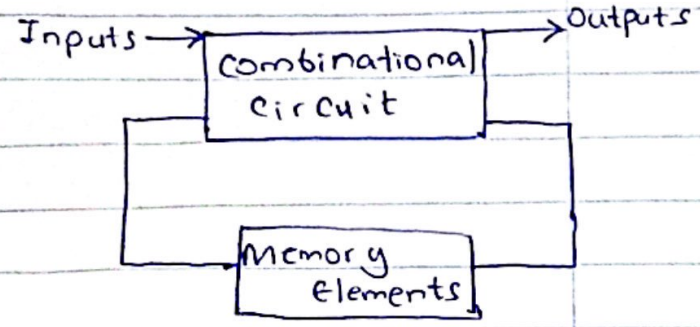
(ii) $X = (C' \cdot B) + (A \cdot C)$

$X = (C'.B) + (C.A)$

$X = B(C'+C) + C.A$

$= B + CA$ //

02) I. Block Diagram of Sequential Circuit.



II. Different between Latch and FlipFlop.

Latch	FlipFlop
• Asynchronous • Outputs can change as soon as the inputs do.	• edge-triggered and only changes state when a control signal Signal goes from high to low or low to high.

VI. 2007/2008 - Qs

* Latches are transparent because their output follows the input in real-time, while flip flops are not transparent as they only respond to clock edges to capture and store data.

RICHARD

No:.....