SYLLABUS

EE6301 DIGITAL LOGIC CIRCUITS LT P C 3 1 0 4

UNIT I NUMBER SYSTEMS AND DIGITAL LOGIC FAMILIES

Review of number systems, binary codes, error detection and correction codes (Parity and Hamming code0- Digital Logic Families ,comparison of RTL, DTL, TTL, ECL and MOS families -operation, characteristics of digital logic family

UNIT II COMBINATIONAL CIRCUITS

Combinational logic - representation of logic functions-SOP and POS forms, K-map representations- minimization using K maps - simplification and implementation of combinational logic - multiplexers and demultiplexers - code converters, adders, subtractors.

UNIT III SYNCHRONOUS SEQUENTIAL CIRCUITS

Sequential logic- SR, JK, D and T flip flops - level triggering and edge triggering - counters - asynchronous and synchronous type - Modulo counters - Shift registers - design of synchronous sequential circuits – Moore and Melay models- Counters, state diagram; state reduction; state assignment.

UNIT IV ASYNCHRONOUS SEQUENTIAL CIRCUITS AND PROGRAMMABLE LOGIC DEVICES

Asynchronous sequential logic circuits-Transition table, flow table-race conditions, hazards &errors in digital circuits; analysis of asynchronous sequential logic circuits-introduction to Programmable Logic Devices: PROM – PLA –PAL.

UNIT V VHDL

RTL Design – combinational logic – Sequential circuit – Operators – Introduction to Packages –Subprograms – Test bench. (Simulation /Tutorial Examples: adders, counters, flipflops, FSM, Multiplexers /Demultiplexers).

TOTAL (L: 45+T: 15): 60 PERIODS

TEXT BOOKS:

1. Raj Kamal, ' Digital systems-Principles and Design', Pearson Education 2nd edition, 2007.
2. M. Morris Mano, 'Digital Design with an introduction to the VHDL', Pearson Education, 2013.
3. Comer "Digital Logic & State Machine Design, Oxford, 2012.

REFERENCES:

1. Mandal "Digital Electronics Principles & Application, McGraw Hill Edu,2013.
2. William Keitz, Digital Electronics-A Practical Approach with VHDL,Pearson,2013.
3. Floyd and Jain, 'Digital Fundamentals', 8th edition, Pearson Education, 2003.
4. Anand Kumar, Fundamentals of Digital Circuits,PHI,2013.
5. Charles H.Roth,Jr,Lizy Lizy Kurian John, 'Digital System Design using VHDL, Cengage, 2013.
6. John M.Yarbrough, 'Digital Logic, Application & Design', Thomson, 2002.

7. Gaganpreet Kaur, VHDL Basics to Programming, Pearson, 2013.
8. Botros, HDL Programming Fundamental, VHDL& Verilog, Cengage, 2013.

WEB RESOURCES:

- http://www.site.uottawa.ca/~petriu/Digital-Logic.pdf
- http://jjackson.eng.ua.edu/courses/ece380/lectures/

AIM &OBJECTIVE OF THE SUBJECT

Aim:

To understand and analyse, linear and digital electronic circuits.

Objective:

- To study various number systems , simplify the logical expressions using Boolean functions
- To study implementation of combinational circuits
- To design various synchronous and asynchronous circuits.
- To introduce asynchronous sequential circuits and PLCs
- To introduce digital simulation for development of application oriented logic circuits.

EE6301 DIGITAL LOGIC CIRCUITS

DETAILED COURSE PLAN

TEXT BOOKS:

1. Raj Kamal, ' Digital systems-Principles and Design', Pearson Education 2nd edition, 2007.
2. M. Morris Mano, 'Digital Design with an introduction to the VHDL', Pearson Education, 2013.
3. Comer "Digital Logic & State Machine Design, Oxford, 2012.

REFERENCES:

1. Mandal "Digital Electronics Principles & Application, McGraw Hill Edu,2013.
2. William Keitz, Digital Electronics-A Practical Approach with VHDL,Pearson,2013.
3. Floyd and Jain, 'Digital Fundamentals', 8th edition, Pearson Education, 2003.
4. Anand Kumar, Fundamentals of Digital Circuits,PHI,2013.
5. Charles H.Roth,Jr,Lizy Lizy Kurian John, 'Digital System Design using VHDL, Cengage, 2013.
6. John M.Yarbrough, 'Digital Logic, Application & Design', Thomson, 2002.
7. Gaganpreet Kaur, VHDL Basics to Programming, Pearson, 2013.
8. Botros, HDL Programming Fundamental, VHDL& Verilog, Cengage, 2013.

Sl. No.	UNIT	Topics	No of Hours	Cumulative Hours	Book No.
UNIT I NUMBER SYSTEMS AND DIGITAL LOGIC FAMILIES					
1	UNIT 1	Review of number systems	1	1	T1
2		Binary codes	1	2	T2
3		*Practice hours*	2	4	T2
4		Error detection and correction codes (Parity and Hamming code)	1	5	T2
5		Digital Logic Families	1	6	T2
6		Comparison of RTL, DTL, TTL, ECL and MOS families	2	8	T2
7		Operation, characteristics of digital logic family	2	10	T2
UNIT II COMBINATIONAL CIRCUITS					
8	UNIT 2	Combinational logic	1	11	T2
9		Representation of logic functions	1	12	T1

Sl. No.	UNIT	Topics	No of Hours	Cumulative Hours	Book No.
10		SOP and POS forms	1	13	T1
11		*Practice hours*	2	15	T1
12		K-map representations	1	16	T2
13		Minimization using K maps	1	17	T1
14		*Practical hours*	1	18	T1
15		Simplification and implementation of combinational logic	1	19	T1
16		Multiplexers and demultiplexers	1	20	T1
17		Code converters	1	21	T1
18		Adders, subtractors	1	22	T2
UNIT III SYNCHRONOUS SEQUENTIAL CIRCUITS					
19	UNIT 3	Sequential logic- SR, JK	1	23	T1
20		D and T flip flops	1	24	T1
21		Level triggering and edge triggering	1	25	T1
22		*Practice hours*	2	27	T1
23		Counters	1	28	T2
24		*Practical hours*	1	29	T2
25		Asynchronous and synchronous type	1	30	T1
26		Modulo counters	1	31	T2
27		*Practice hours*	2	33	T2
28		Shift registers	1	34	T2
29		Design of synchronous sequential circuits	1	35	T2
30		Moore and Melay models	1	36	T2
31	UNIT 3	Counters	1	37	T2
32		State diagram and state reduction	1	38	T2
33		State assignment.	1	39	T2
34		*Practical hours*	2	41	T2
UNIT IV ASYNCHRONOUS SEQUENTIAL CIRCUITS AND PROGRAMMABLE LOGIC DEVICES					
35		Asynchronous sequential logic circuits	1	42	T2

Sl. No.	UNIT	Topics	No of Hours	Cumulative Hours	Book No.
36	UNIT 4	Transition table, flow table-race conditions	1	43	T2
37		hazards &errors in digital circuits	1	44	T2
38		analysis of asynchronous sequential logic circuits	1	45	T2
39		*Practice hours*	2	47	T2
40		introduction to Programmable Logic Devices	1	48	T1
41		Digital logic families :	1	49	T1
42		PROM, PLA and PAL	1	50	T1
43		*Practical hours*	1	51	T1
UNIT V VHDL					
44	UNIT 5	RTL Design	1	52	T2
45		Combinational logic	1	53	R5
46		Sequential circuit	1	54	T2
47		Operators	1	55	T2, R5
48		Introduction to Packages	1	56	T1
49		Sub programs and test bench	1	57	T2, R5
50		Examples: adders, counters	1	58	R5
51		Flip-flops,FSM	1	59	T1
52		Multiplexers / Demultiplexers	1	60	T2

INDEX

UNIT-I NUMBER SYSTEMS AND DIGITAL LOGIC FAMILIES
TWO MARKS

1. Define weight and non weighted codes (MAY 2014)

Each digit position of the number represents a specific weight. Example: 8421, 2421.

Non weighted codes are not assigned with any weight to each digit position.

Examples: Excess 3 and gray codes

2. What are error correcting codes? (Nov2011)

Due to presence of noise, when the digital information in the binary form is transmitted from one circuit or system to another circuit or system an error may occur. This means a signal corresponding to 0 may change to 1 or vice-versa. To maintain the data integrity between transmitter and receiver, extra bit or more than one bit is added in the data. These extra bits allow the detection and sometimes correction of error in the data. Codes which allow only error detection are called error detecting codes.

3. Mention the important characteristics of digital IC's? (MAY 2014, MAY2011)

Fan out, Power dissipation, Propagation Delay, Noise Margin, Fan In, Operating temperature.

4. Convert the $(101010.1010)_2$ numbers with the indicated base to decimal? (May 2015)

$(101010.1010)_2 =1*2^5+0*2^4+1*2^3+0*2^2+1*2^1+0*2^0+1*2^{-3}+0*2^{-2}+1*2^{-1}+0*2^0=(42.625)_{10}$

5. Define Unit distance code. (Dec 2015)

The Gray **code** is a single-step **code** (i.e. a **unit-distance code**). It's often used in analog/ digital conversion devices. Adjacent **code** patterns of Gray **code** differ in just only one bit to avoid ambiguity, i.e. consecutive **code** elements have a hamming **distance** of one.

6. Define Fan-in and Fan-out? (DEC 2015)

Fan in is the number of inputs connected to the gate without any degradation in the voltage level.

Fan out specifies the number of standard loads that the output of the gate can drive without impairment St of its normal operation.

7. What is propagation delay?(MAY 2015)

Propagation delay is the average transition delay time for the signal to propagate from input to output when the signals change in value. It is expressed in ns.

8. Comparison between Totem pole & Open collector. (Nov 2014)

Totem pole	Open collector
Output stage consists of pull-up transistor(Q3), diode resistor and pull-down transistor(Q4).	Output stage consists of pull-down transistor.
External pull-up resistor is not required.	External pull-up resistor is required for proper operation of gate.
Output of two gates cannot be tied together	Output of two gates can be tied together using wired AND technique.
Operating speed is high	Operating speed is low

9. Write the advantages of CMOS family. (May 2013)

1. Consumes less power
2. Can be operated at high voltages, resulting in improved noise immunity
3. Fan-out is more and better noise margin.

10. State the advantages and disadvantages of totem-pole output. (Nov 2011)

Advantages: 1.External pull up resistor is not required and Operating speed is high

Disadvantages: 1. Output of two gates cannot be tied together.

1(i). Explain in detail about Resistor transistor logic.

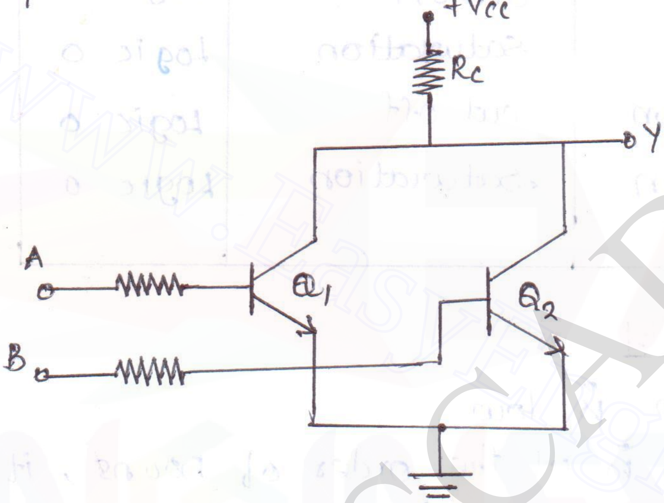
RESISTOR TRANSISTOR LOGIC (RTL)

* RTL circuit consist of resistors and transistors of both input and output stage circuits in a NOR logic gate.

* The emitters of both the transistors are connected to a common ground and collectors of both transistors are tied through a common collector resistor R_c to the supply voltage V_{cc}.

* The resistor R_c is commonly known as passive pull up resistor.

2 input RTL NOR Gate circuit diagram.



Circuit Operation.

* RTL gate input voltage corresponding to low level is required to be low enough for the corresponding transistor to be cut-off.

* when the input voltage corresponding to high level should be high enough to drive the corresponding transistor to saturation

* when both the inputs are low, transistor Q_1 and Q_2 are cut-off and the output is high.

* When the both inputs are high, transistor Q_1 and Q_2 are saturation and the output is low.

* The saturation voltage, $V_{CE(sat)}$ for transistor is approximately 0.2V. So RTL gates low level output voltage is 0.2.

*A high level output voltage is depends on the number of gates connected to the output.

* the number of gates connected to the output increases output voltage decreases. This ~~is~~ is deciding factor for the fanout of the gates.

* The number of gates connected to the output also affects the propagation delay time.

V_A	V_B	Transistor Q_1	Transistor Q_2	V_y
Logic 0	Logic 0	cut-off	cut-off	Logic 1
Logic 0	Logic 1	cut-off	Saturation	Logic 0
Logic 1	Logic 0	saturation	cut-off	Logic 0
Logic 1	Logic 1	saturation	Saturation	Logic 0

Characteristics of RTL family

* The speed of operation is low
(The propagation delay is of the order of 500 ns, it can't operate at speeds above 4 MHz.)

* Fanout is 4 or 5 with a switching delay of 50ns, and fanin is 4.

* Poor noise immunity

* Elimination of base resistors in RTL will reduce the power dissipation.

* sensitive to temperature

* The noise margin from zero to the threshold voltage is about 0.5V, and from one to the threshold voltage is only 0.2V.

Disadvantage

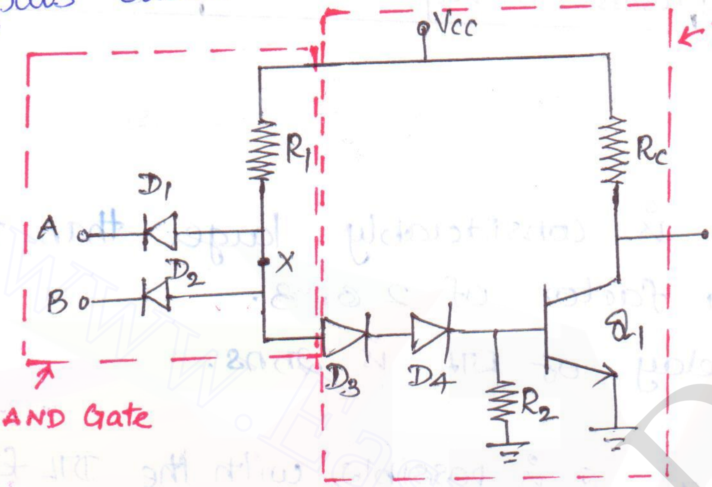
* Low speed
* poor noise immunity

1(ii) Explain in detail about diode transistor logic.

DIODE TRANSISTOR LOGIC (DTL)

*The diode transistor logic is somewhat more complex than RTL. but it have greater fan-out and improved noise margins.

*The circuit consist of diodes and transistors of a two inputs diode transistor logic NAND gate.



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

circuit operation.

* when both inputs are low, diode D_1 and D_2 conduct resulting 0.7 volts at point x. This 0.7 voltage at point is not sufficient to drive transistor Q_1.

∴ Q_1 is cut-off giving output voltage $V_0 = V_{cc}$ so logic 1.

*when both inputs are high, D_1 and D_2 are reversed biased. This causes the base current of transistor Q_1 to flow through R_1, D_3, D_4 and the base of the transistor Q_1

∴ The Transistor Q_1 is saturation giving output voltage

$V_{CE(sat)} = 0.2 = \text{Logic } 0$

* The Diode D_3, D_4 we need increased voltage level to drive transistor in saturation. This improve the noise margin for DTL gate

* when any one input is high (or) Low. The transistor Q_1 is cut-off giving output voltage $V_0 = V_{cc}$. and Logic 1.

Inputs		Diodes		Transistor	output
A	B	D_1	D_2	T	Y
Logic 0	Logic 0	Forward biased	Forward biased	Cut-off	Logic 1
Logic 0	Logic 1	Forward biased	Reverse biased	Cut-off	Logic 1
Logic 1	Logic 0	Reverse biased	Forward biased	Cut-off	Logic 1
Logic 1	Logic 1	Reverse biased	Reverse biased	Saturation	Logic 0

DTL family characteristics

Propagation delay

* The turn off delay is considerably larger than the turn on delay, often by a factor of 2 or 3.

* The propagation delay of DTL is 25ns.

Fan out

* A fan-out as high as 8 is possible with the DTL family because of the high input impedance of the subsequent gates in the logic 1 state.

Fan in

* It has a fan in of 8.

Noise immunity

* The noise margin is high due to the additional diode D_4 connected in series with D_1.

Advantages.

* Fan out is high
* Power dissipation is 8-12mW
* Noise immunity is good

Disadvantages

* More elements are required
* Propagation delay is more
* Speed of operation is less

2. Explain in detail about Emitter coupled logic.

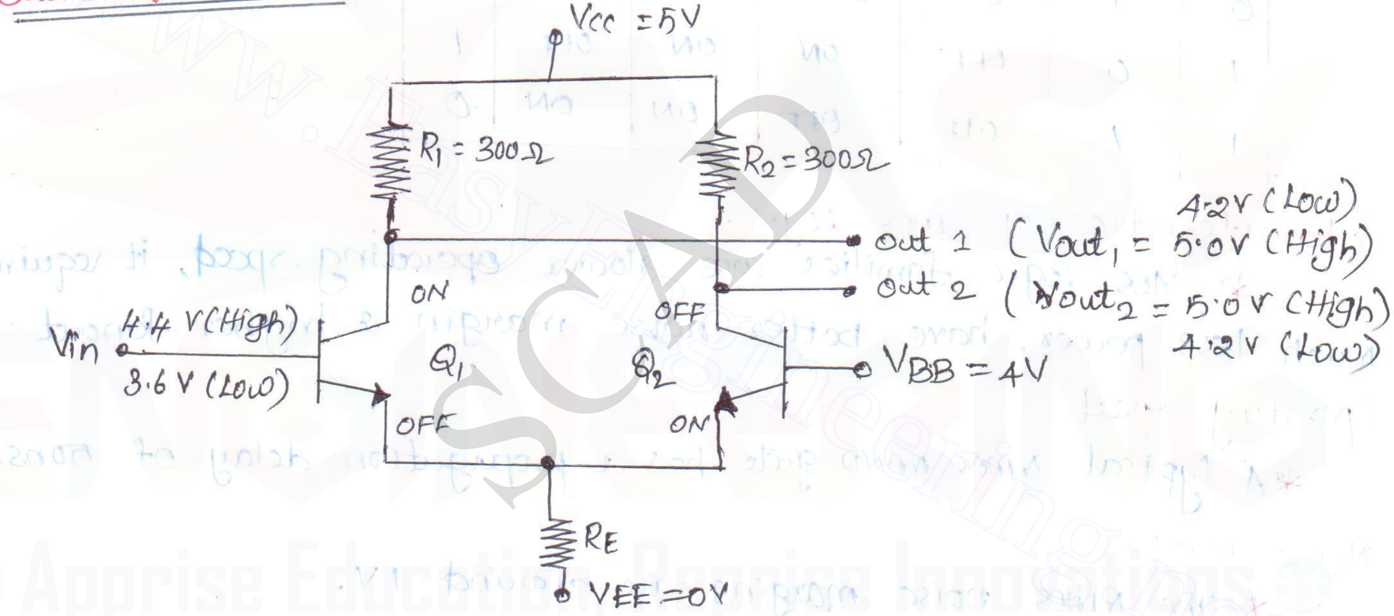
Emitter coupled Logic (ECL) [NOV/DEC 2012], [MAY/JUNE 2013], [MAY 2014] [APRIL/MAY 2015]

* TTL family uses transistors operating in the saturation mode. as a result their switching speed is limited by the storage delay time associated with a transistor is driven into saturation.

* Another logic family has been developed that prevents transistor saturation, there by increasing overall switching speed by using radially different structure is called "current mode logic (CML). This family is also known as "Emitter coupled logic".

* ECL does not produce a large voltage swing between the low and high levels. It internally switches current between two possible path depending on the output state.

Basic ECL circuit



Circuit operation

* when input voltage Vin is ~~low~~ high (4.4V), transistor Q1 is ON, but not saturated and transistor Q2 is OFF. Vout2 is pulled to 5.0V (High) through R2 and drop across R1 is 0.8V. so that Vout1 is 4.2V (low)

* when input voltage Vin is low (3.6V), transistor Q2 is ON, but not saturated and transistor Q1 is OFF. Thus Vout1 is pulled to 5.0V (High) through R1 and drop across R2 is 0.8V so that Vout2 is 4.2V (Low).

Circuit operation of 2 input CMOS NOR Gate.

* When both inputs are low, Q_1 and Q_2 are ON, Q_3 and Q_4 are OFF and the output is high (V_{DD}).

* When any one of the inputs is low (OV or -ve), then the corresponding MOSFET Q_1 or Q_2 is ON, Q_3 or Q_4 is ON and the output is low.

* When inputs are high, Q_1 and Q_2 are OFF, Q_3 and Q_4 are ON, and the output is low.

A	B	Q_1	Q_2	Q_3	Q_4	V_o
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	0
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

Characteristics OF MOS Logic.

* MOS Logic families are slower operating speed, it require much less power, have better noise margin & higher fanout.

Operating speed

* A typical NMOS NAND gate has a propagation delay of 50ns.

Noise margin

* The NMOS noise margins are around 1V.

Fan-out

* The fan-out capabilities of MOS logic would be virtually unlimited because of the extremely high input resistance at each MOSFET input. It easily operate at a fan-out value of 50

Power dissipation

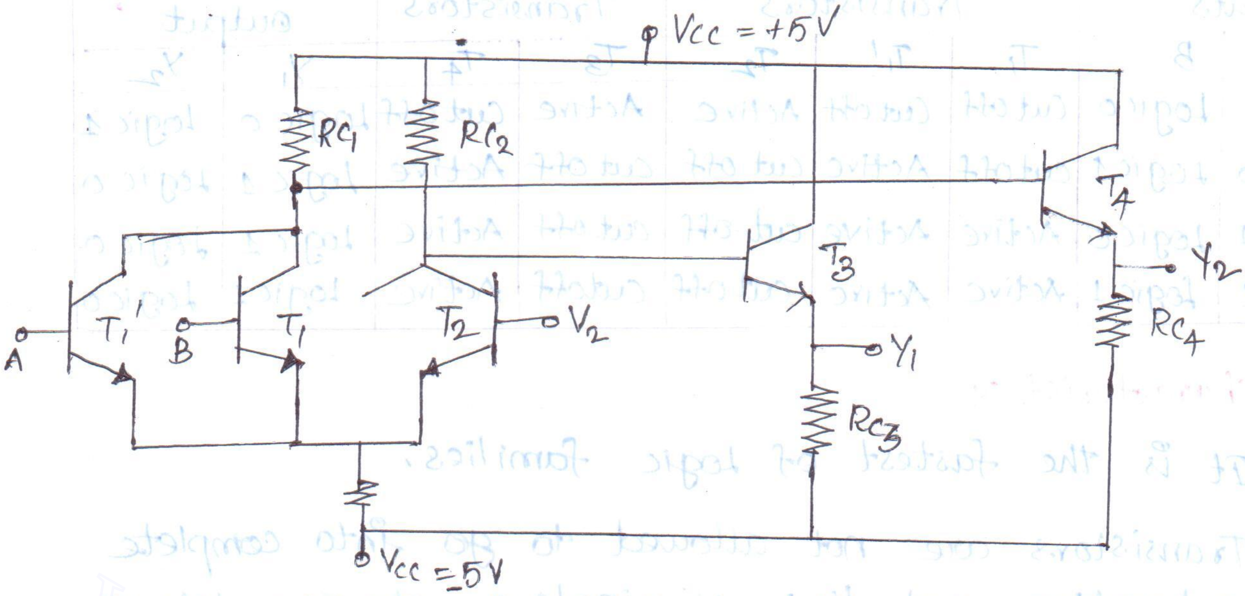
* The power dissipation of a CMOS IC is very low as long as it is in a dc condition.

* Unfortunately power dissipation of CMOS IC increases in propagation to the frequency at which the circuits are switching states.

Propagation delay:

* The propagation delay in CMOS is the sum of delay due to internal capacitance & due to the load capacitance

Emitter coupled logic OR/ NOR gate



* The circuit consist of emitter followers are used at the output of difference amplifier to shift the DC level.

* The circuit has two outputs Y_1 and Y_2, which are complementary. Y_1 corresponds to OR logic and Y_2 corresponds to NOR logic.

Circuit Operation.

* When both inputs are Logic '0'. T_1 and T_1' operate in cut-off and T_2 operates in active region, voltage V_{01} is high T_3 is ON and the output at Y_2 is logic '1', Voltage V_{02} is low. T_4 operates in cut off & output Y_1 is logic '0'.

* When any one of the input is logic '1', the transistors T_1 and T_1' are operated in active region and T_2 operates in cut-off, Voltage V_{01} is low, T_3 operates in cutoff & Y_2 is logic 0, voltage V_{02} is high, T_4 operates in active region and Y_1 is logic '1'.

*When both inputs are logic 1 state T_1 and T_1' operate in active region and T_2 operates in cutoff, voltage V_{01} is low, T_3 operates in cut-off and Y_2 is logic '0', voltage V_{02} is high, T_4 operates in active region and Y_1 is logic 1.

Inputs		Transistors			Transistors		output	
A	B	T_1	T_1'	T_2	T_3	T_4	Y_1	Y_2
Logic 0	Logic 0	Cut off	Cut off	Active	Active	Cut off	Logic 0	Logic 1
Logic 0	Logic 1	cutoff	Active	cut off	cut off	Active	Logic 1	Logic 0
Logic 1	Logic 0	Active	Active	cut-off	cut off	Active	Logic 1	Logic 0
Logic 1	Logic 1	Active	Active	cut off	cutoff	Active	Logic 1	Logic 0

ECL Characteristics.

- * It is the fastest of logic families.
- * Transistors are not allowed to go into complete saturation, and thus eliminating storage delays
- * To prevent transistors from going into complete saturation, logic levels are kept close to each other.
- * Noise margin is reduced and it is difficult to achieve good noise immunity
- * Another disadvantage of this approach is that power consumption is more because transistors are not completely saturated.
- * Switching transients are less because power supply current is more suitable stable than in TTL and cmos circuits.

Advantages.

- * Fastest logic family
- * Good noise immunity

Disadvantage:

- * Power consumption is more

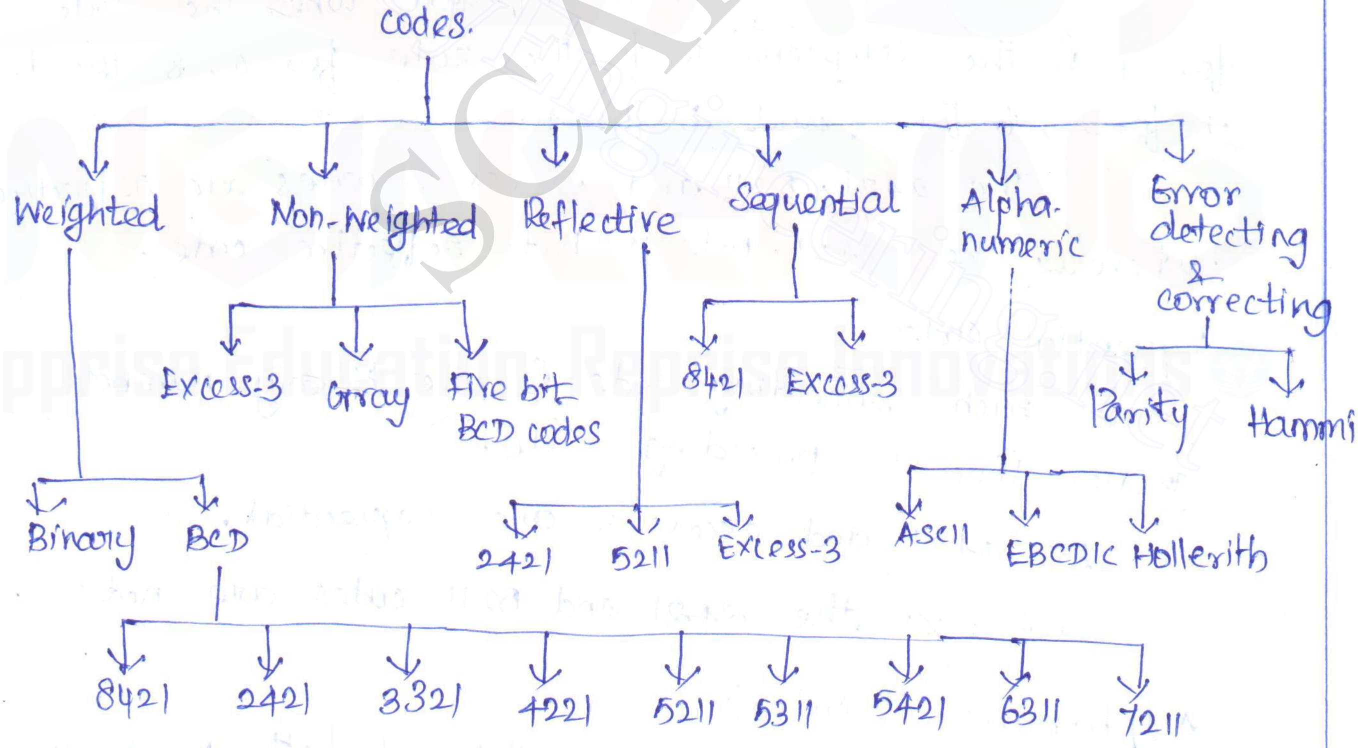
3. Explain in detail about binary codes.

Binary codes [NOV/DEC 2014]

The digital data represented, stored and transmitted as groups of binary digits. The group of bit also known as binary codes.

classification

1. Weighted codes
2. Non weighted codes
3. Reflective codes
4. Sequential codes
5. Alphanumeric codes
6. Error detecting and correcting codes.



Weighted codes :-

Each digit position of the number represent a specific weight.

Number 567 then weight of 5 is 100, weight of 6 is 10, and weight of 7 is 1.

Examples: 8421, 2421 and 5211

Non weighted codes:

Non weighted codes are not assigned with any weight to each digit position.

Example : Excess 3 and gray codes

Reflective codes :-

A code is said to be reflective when the code for 9 is the complement for the code for 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4.

The 2421, 5211 and excess-3 codes are reflective, whereas the 8421 code is not reflective code.

Sequential codes :-

Each succeeding code is one binary number greater than its preceding code.

Ex: 8421 and excess-3 are sequential.

whereas the 2421 and 5211 codes are not.

Alphanumeric codes:-

The codes which consist of both numbers and alphabetic characters are called alphanumeric codes.

ASCII character code (American Standard code for information interchange)

Digital computer require the handling not only of numbers, but also the other characters or symbols.

				$b_7 b_6 b_5$							
b_4	b_3	b_2	b_1	000	001	010	011	100	101	110	111
0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	STX	DC2	"	2	B	R	b	r
0	0	1	1	ETX	DC3	#	3	C	S	c	s
0	1	0	0	EOT	DC4	$	4	D	T	d	t
0	1	0	1	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	ACK	SYN	&	6	F	V	f	v
0	1	1	1	BEL	ETB	'	7	G	W	g	w
1	0	0	0	BS	CAN	(	8	H	X	h	x
1	0	0	1	HT	EM	)	9	I	Y	i	y
1	0	1	0	LF	SUB	*	:	J	Z	j	z
1	0	1	1	VT	ESC	+	;	K	[	k	{
1	1	0	0	FF	FS	,	<	L	\	l	|
1	1	0	1	CR	GS	-	=	M	]	m	}
1	1	1	0	SO	RS	.	>	N	^	n	~
1	1	1	1	SI	US	/	?	O	_	o	DEL

NUL – Null

SOH – Start of heading

STX – Start of text

EIX – End of text

EOT – End of transmission

ENQ – Enquiry

ACK – Acknowledge

Bel – Bell

DLE – Data link escape.

BS – Back space

HT – Horizontal tab

LF – Line feed

VT – Vertical tab

FF – Form feed

CR – Carriage return

SO – Shift out

SI – Shift in

SP – Space.

Error detecting and correcting codes

When the digital information in the binary form is transmitted from one circuit or system to another circuit (or) system, an error may be occur.

A signal corresponding to 0 may change to 1. due to presence of noise.

Codes which allow only error detection are called error detecting codes and codes which allow error detection and correction are called error detecting and correcting codes.

OTHER CODES.

Decimal	8421	5421	2421	5211
0	0000	0000	0000	0000
1	0001	0001	0001	0001
2	0010	0010	0010	0011
3	0011	0011	0011	0101
4	0100	0100	0100	0111
5	0101	1000	1011	1000
6	0110	1001	1100	1001
7	0111	1010	1101	1011
8	1000	1011	1110	1110
9	1001	1100	1111	1111

DC1 – Device contol 1

DC2 – Device control 2

DC3 – Device control 3

DC4 – Device control 4

NAK – Negative acknowledge

SYN – Synchronous idle

ETB – End of transmission block

CAN – cancel

EM – End of medium.

SUB – Substitute

ESC – Escape.

FS – File separator

GS – Group separator

RS – Record separator

US – Unit separator

DEL – Delete.

4 (i) Error Correcting codes

Hamming code: [NOV / DEC 2014] [NOV / DEC 2015]

The system provides a methodical way to add one or more parity bits to a data character in order to ~~prox~~ detect and correct errors.

Hamming distance between two code words is defined as the number of bits changed from one code word to another.

The 7 bit hamming (7,4) code word $h_1 h_2 h_3 h_4 h_5 h_6 h_7$ associated with 4-bit binary number $b_3 b_2 b_1 b_0$.

$h_1 = b_3 \oplus b_2 \oplus b_0$ $\qquad h_3 = b_3$

$h_2 = b_3 \oplus b_1 \oplus b_0$ $\qquad h_5 = b_2$

$h_4 = b_2 \oplus b_1 \oplus b_0$ $\qquad h_6 = b_1$

$h_7 = b_0$

To decode a hamming code, one must check for odd parity over the bit fields in which even parity was previously ~~establmat~~ established.

$c_1 = h_1 \oplus h_3 \oplus h_5 \oplus h_7$

$c_2 = h_2 \oplus h_3 \oplus h_6 \oplus h_7$

$c_4 = h_4 \oplus h_5 \oplus h_6 \oplus h_7$

Encode data bits 0101 into a 7-bit even parity hamming code

Soln: $b_3\ b_2\ b_1\ b_0 = 0\ 1\ 0\ 1$

$h_1 = b_3 \oplus b_2 \oplus b_0 = 0 \oplus 1 \oplus 1 = 0$ $\qquad h_3 = b_3 = 0$

$h_2 = b_3 \oplus b_1 \oplus b_0 = 0 \oplus 0 \oplus 1 = 1$ $\qquad h_5 = b_2 = 1$

$h_4 = b_2 \oplus b_1 \oplus b_0 = 1 \oplus 0 \oplus 1 = 0$ $\qquad h_6 = b_1 = 0$

$h_7 = b_0 = 1$

h_1	h_2	h_3	h_4	h_5	h_6	h_7
0	1	0	0	1	0	1

A 7 bit hamming code is received as 0101101. What is the correct code.

h_1	h_2	h_3	h_4	h_5	h_6	h_7
0	1	0	1	1	0	1

Find error

$$c_1 = h_1 \oplus h_3 \oplus h_5 \oplus h_7 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$$

$$c_2 = h_2 \oplus h_3 \oplus h_6 \oplus h_7 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$$

$$c_3 = h_4 \oplus h_5 \oplus h_6 \oplus h_7 = 1 \oplus 1 \oplus 0 \oplus 1 = 1$$

$$c_4\, c_2\, c_1 = 100$$

∴ bit 4 is in error & the corrected code word can be obtained by complementing the fourth bit in the received code word as 0100101.

check sums

The parity bit method can detect only a single error with in a word and not double errors.

∴ the double error will not change the parity of the bits ∴ the parity checker will not indicate the error.

Check sum method is used to detect the double errors and pinpoint erroneous bits.

EBCDIC:

The frequently encounted is called the Extended Binary code Decimal Interchange code. In which decimal digits are represented by the BCD code

Hollerith code:

The code used in the system to represent alphanumeric information is known as Hollerith code.

Encode the binary word 1011 into seven bit even parity Hamming code. [APRIL/MAY 2015]

Soln:

step 1 : Find the number of parity bits required. Let P=3,

$$2^P = 2^3 = 8 \qquad \boxed{2^P \geq x+P+1}$$

$$x + P + 1 = 4+3+1 = 8 \qquad 8 \geq 7$$

Three parity bits are sufficient.

∴ Total code bits = 4+3 = 7

step 2 : Construct a bit location table

Bit designation	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	7	6	5	4	3	2	1
Binary location number	111	110	101	100	011	010	001
Information bits	1	0	1		1		
				0		0	1

step 3 : Determine the parity bits

For P_1 : Bit locations 3, 5 and 7 have three 1s and therefore to have an even parity P_1 must be 1.

For P_2 : Bit locations 3, 6 and 7 have two 1's and therefore to have an even parity P_2 must be 0.

For P_4 : Bit locations 5, 6 and 7 two 1's and therefore to have an even parity P_4 must be 0.

step 4 : Enter the parity bits into the table to form a seven bit Hamming code = 1010101

'The Hamming distance between two code words is defined as the number of bits changed from one code word to another.'

Given that a frame with bit sequence 1101011011 is transmitted, it has been received as 1101011010. Determine the method of detecting the error using any one error detecting code. [NOV/DEC 2014]

Soln:

Step 1: construct the bit location table.

Bit designation	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
Bit location	10	9	8	7	6	5	4	3	2	1
Binary location number	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Received code	1	1	0	1	0	1	1	0	1	0

Step 2: check for parity bits

For P_1 : P_1 check locations 1, 3, 5, 7 and 9

There are three 1's in the group.

∴ parity check for ODD parity is correct → 0

For P_2 : P_2 check locations 2, 3, 6, 7 and 10

There are three 1's in the group

∴ Parity check for odd parity is correct → 0

For P_4 : P_4 check location 4, 5, 6 and 7

There are three 1's in the group

∴ Parity check for ODD parity is correct → 0

For P_8 : P_8 check location 8, 9 and 10

there are two 1's in the group

∴ Parity check for ODD parity is wrong → 1

The resultant word is 0001. This says that the bit in the number 1 location is in error. It is 0 and should be a '1'. Therefore, the correct code is 1101011011, which agrees with the transmitted code.

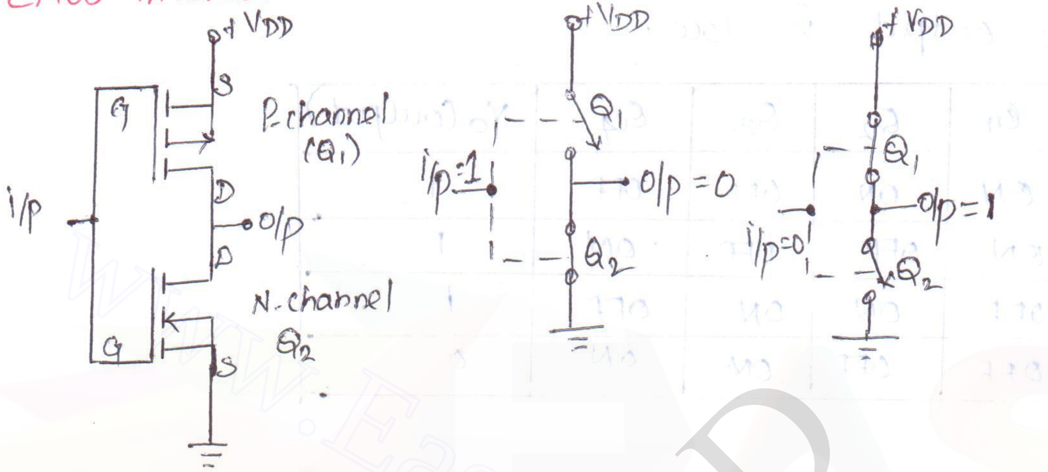
4(ii) Draw & Explain the CMOS NAND & CMOR NOR gates ?

CMOS LOGIC [MAY/JUNE 2014] [NOV/DEC 2015] [NOV/DEC 2014]

* Complementary Metal Oxide Semiconductor (CMOS) circuits contain both NMOS and PMOS devices to speed the switching of capacitive loads.

* It consumes low power and can be operated at high voltages, resulting in improved noise immunity.

CMOS Inverter



A	Q1	Q2	O/P
0	ON	OFF	1
1	OFF	ON	0

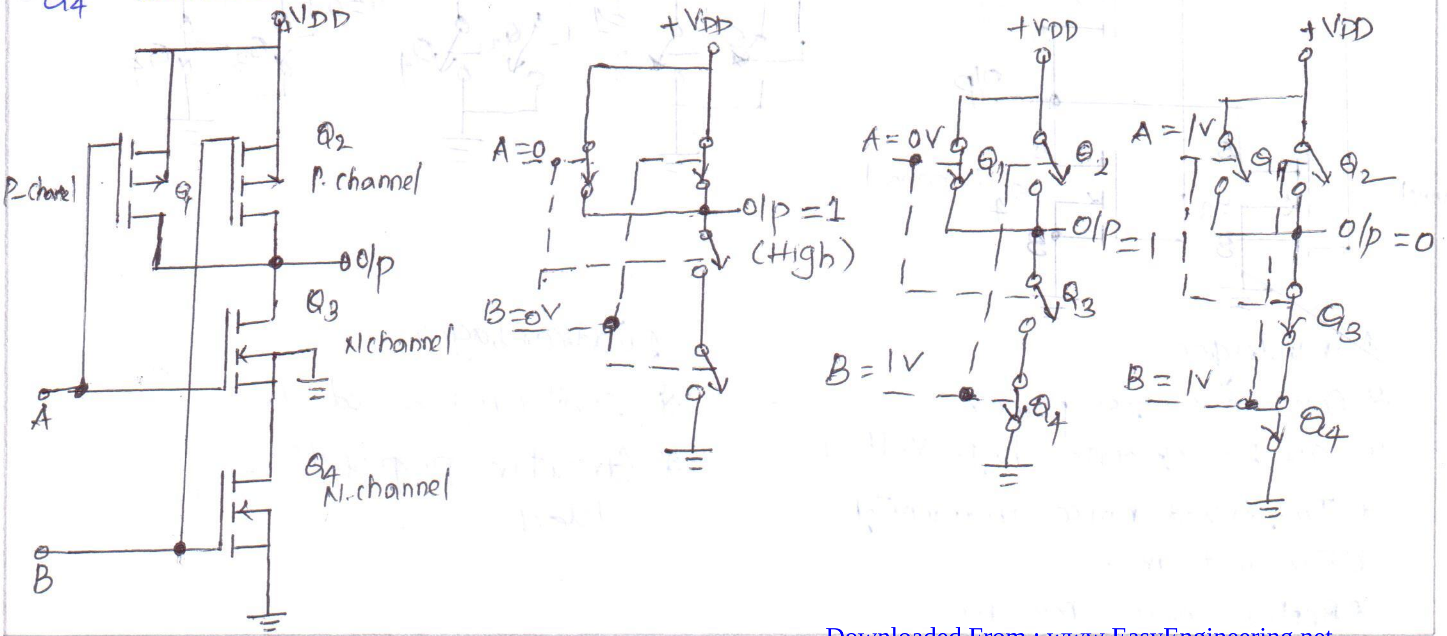
Circuit operation

* When the input is low, Q1 is ON and Q2 is OFF, output is high.

* When the input is high, Q1 is OFF and Q2 is ON, output is low.

CMOS NAND GATE

* CMOS 2-input NAND gate consist of two P-channel MOSFETs, Q1 and Q2, connected in Parallel and two N-channel MOSFET Q3 and Q4 connected in series.



Circuit operation of 2 input CMOS NAND Gate.

* when the inputs are low, Q_1 and Q_2 are ON, Q_3 and Q_4 are OFF, and the output is high (VDD)

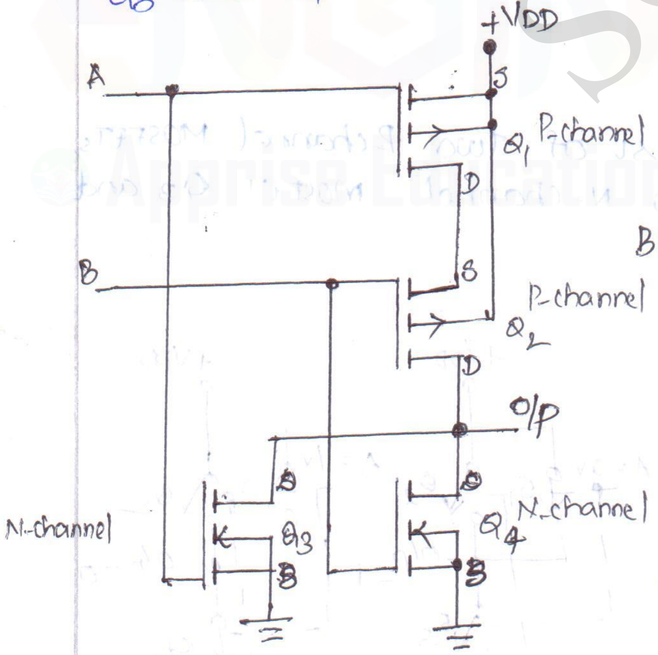
* when any one of the inputs is low (0V or -Ve), then the corresponding MOSFET Q_1 or Q_2 is ON, Q_3 or Q_4 is ON and the output is high.

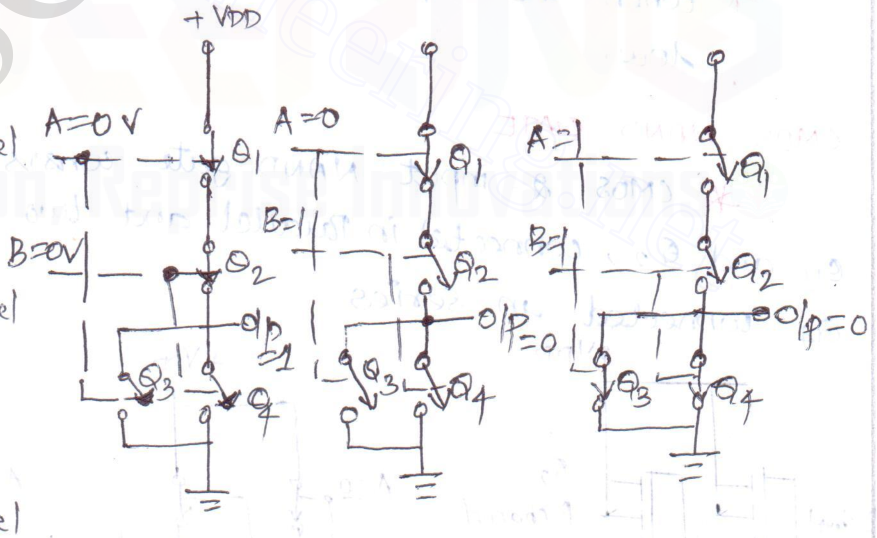
* when the inputs are high, Q_1 and Q_2 are OFF, Q_3 and Q_4 are ON and the output is low.

A	B	Q_1	Q_2	Q_3	Q_4	V_0 (output)
0	0	ON	ON	OFF	OFF	1
0	1	ON	OFF	OFF	ON	1
1	0	OFF	ON	ON	OFF	1
1	1	OFF	OFF	ON	ON	0

CMOS NOR GATE

* CMOS 2 input NOR gate consist of two P-channel MOSFET Q_1 and Q_2 are connected in series and N-channel MOSFETs Q_3 and Q_4 are connected in parallel.





Advantages:

* Consumes less power
* can be operate high Voltages
* Improved noise immunity
* Fan out more
* Better noise margin

Disadvantages.

* switching speed low
* Greater propagation delay

5. Design a TTL logic circuit for a 2 input NAND gate.

TRANSISTOR TRANSISTOR LOGIC (TTL) [APRIL/MAY 2015], [NOV/DEC 2014]

* Transistor transistor logic, TTL is named for its dependance on transistors alone to perform basic logic operation.

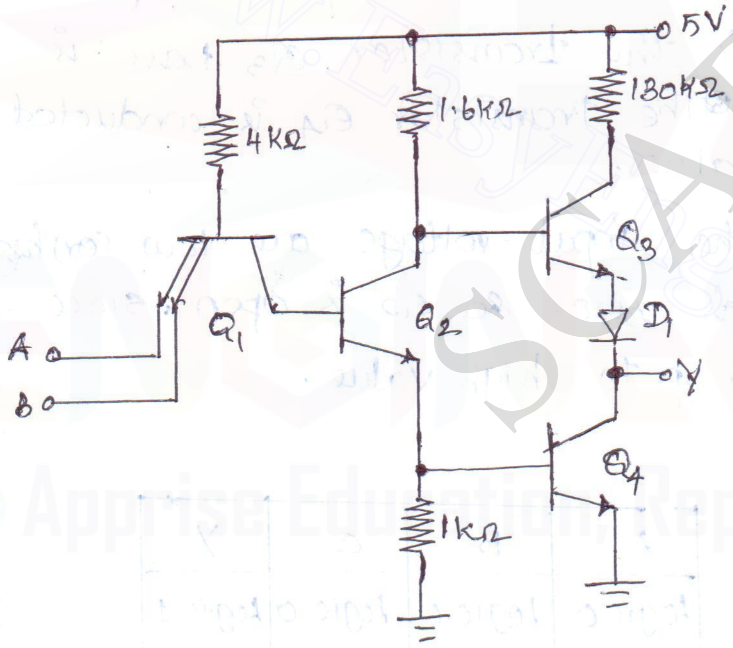
* The first version, which is now known as standard TTL

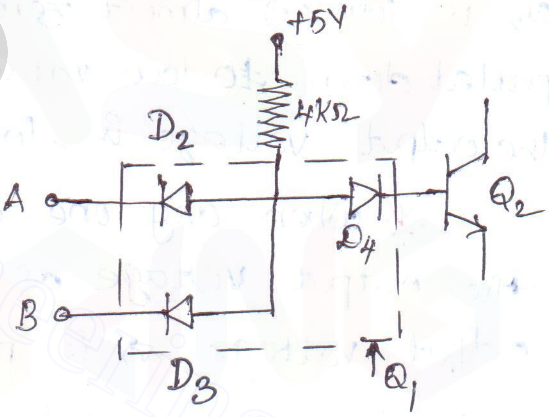
2-input TTL NAND Gates.

* The 2 input TTL NAND gate input structure consists of multiple emitter transistor and output structure consist of totem pole output.

* The transistor Q_1 having two emitters, one for each input to the gate.

* Diodes D_2 and D_3 represent the two emitter base junction of Q_1 and D_4 is the collector-base junction of Q_1.





Input		output
A	B	Y
Logic 0	Logic 0	Logic 1
Logic 0	Logic 1	Logic 1
Logic 1	Logic 0	Logic 1
Logic 1	Logic 1	Logic 0.

Operation:

* When both input voltage A and B are low. The transistor Q_1 is ON and the output voltage of Q_1 is almost zero. Therefore Q_2 is cutoff.

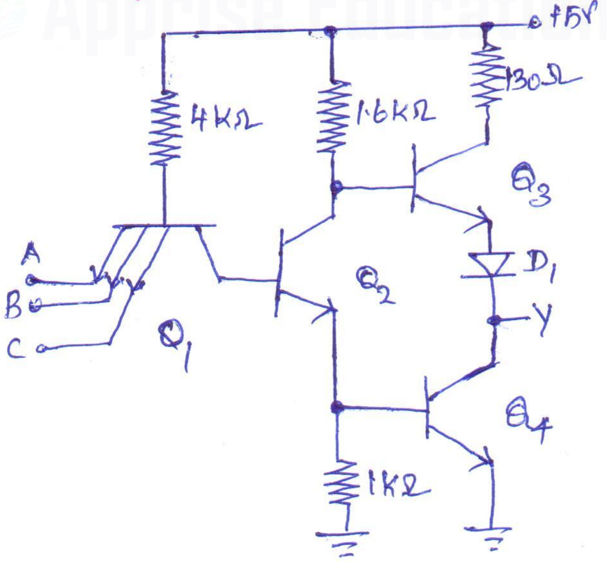
* When Q_2 is open (or) cutoff the output voltage of Q_2 is high. So the transistor Q_3 base is pulled high. Since Q_3 act as an emitter follower. the output Y is pulled up to a high voltage.

* When both input voltages are ~~low~~ high. The transistor Q_1 is OFF (or) cutoff and output voltage of Q_1 is high. Therefore Q_2 is saturated. (or) ON.

* When the transistor Q_2 is ON the output voltage of Q_2 is low (or) almost zero. So the transistor Q_3 base is pulled down ~~to~~ low value. The transistor Q_4 is conducted. the output voltage is low value.

* When any one of the input voltage are low (or) high the output voltage of Q_1 is zero. So Q_2 is open. Since. output voltage Y is pulled up to high value.

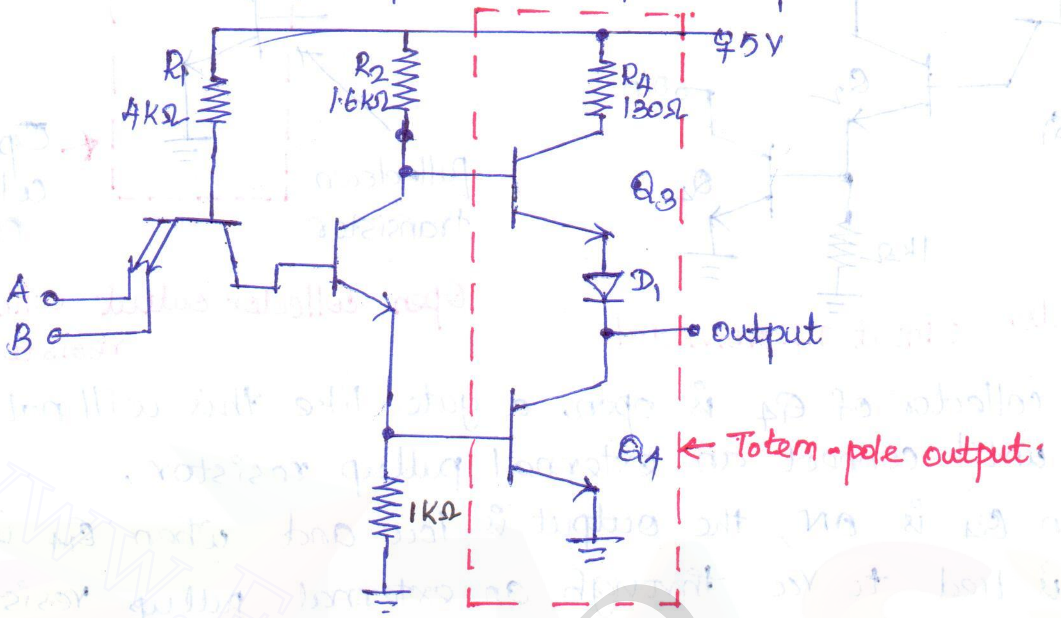
3-input TTL NAND gates.



A	B	C	Y
Logic 0	Logic 0	Logic 0	Logic 1
Logic 0	Logic 0	Logic 1	Logic 1
Logic 0	Logic 1	Logic 0	Logic 1
Logic 0	Logic 1	Logic 1	Logic 1
Logic 1	Logic 0	Logic 0	Logic 1
Logic 1	Logic 0	Logic 1	Logic 1
Logic 1	Logic 1	Logic 0	Logic 1
Logic 1	Logic 1	Logic 1	Logic 0

TOTEM POLE OUTPUT

* Transistor Q_3 and Q_4 form a totem-pole. such a configuration is known as active pull-up or totem pole output.



* Totem-pole transistors produce a low output impedance.

* Either Q_3 acts as an emitter follower (high output) or Q_4 is saturated (Low output).

* When Q_3 is conducting, the output impedance is approximately 70Ω.

* when Q_4 is saturated, the output impedance is only 12Ω the output impedance value is low. this means the output voltage can change quickly from one state to the other because any stray output capacitance is rapidly charged or discharged through the low output impedance.

*The propagation delay is low in totem pole TTL logic.

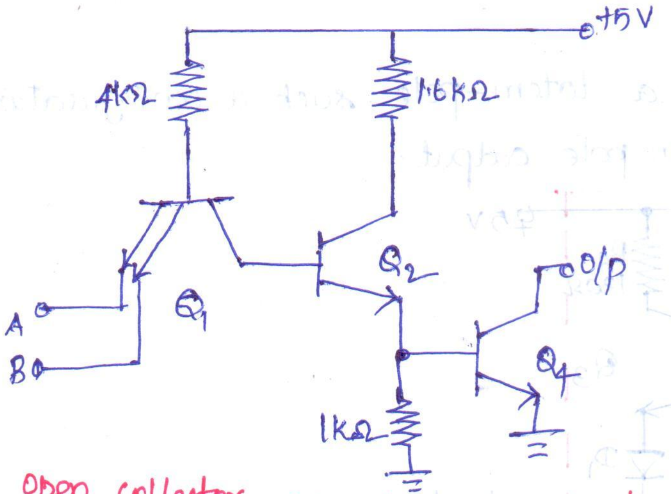
Open-Collector output.

* TTL devices provide another type of output called open collector output.

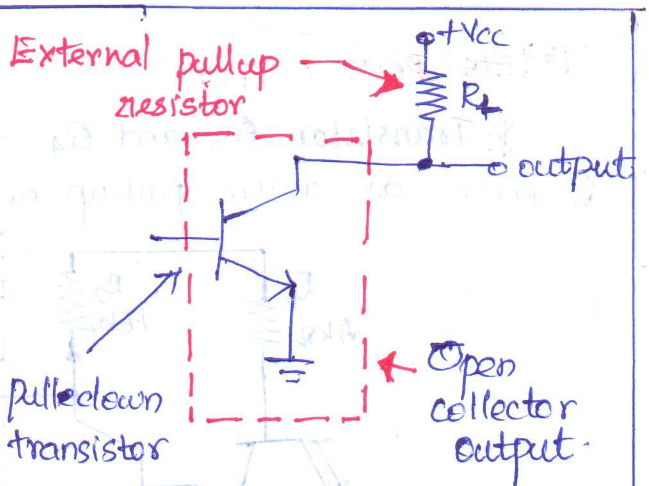
*The output of two different gates with open collector output can be tied together. This is known as wired Logic.

*A 2 input NAND gate with an open collector output eliminates the pull up transistor Q_3, D_1 and R_4.

*The output is taken from the open collector terminal of transistor Q_4.



open collector 2 input TTL NAND gate



External pullup resistor

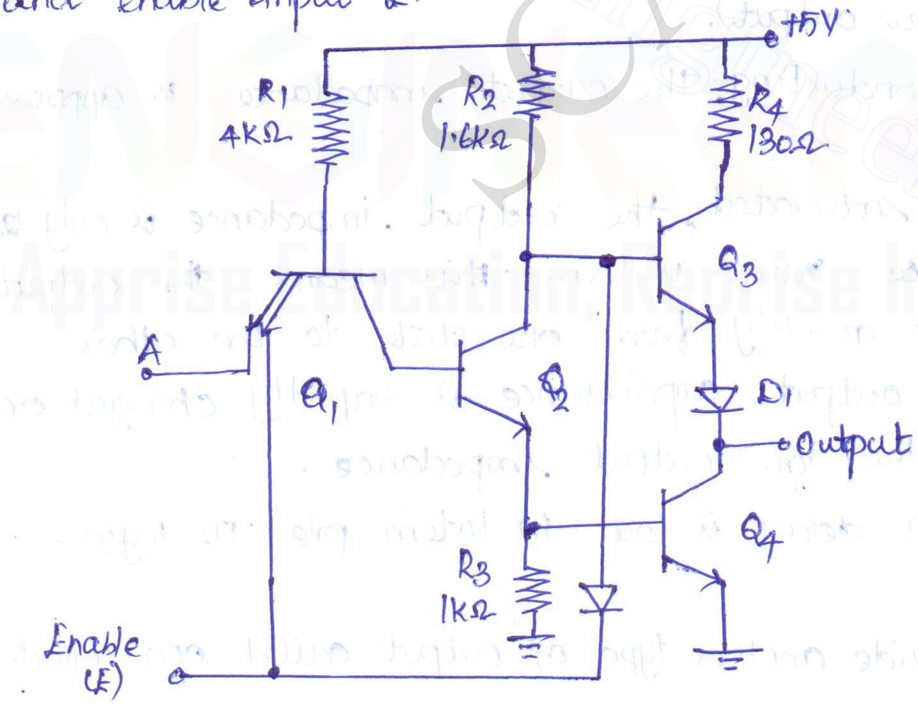
Open collector output with pullup resistor.

*The collector of Q4 is open, a gate like this will not work properly until connect an external pullup resistor.

*when Q4 is ON, the output is low and when Q4 is OFF output is tied to Vcc through an external pullup resistor.

Tri state TTL inverter.

*The tristate TTL inverter has two inputs - normal input A and enable input E.



*It utilizes the high speed operation of the totem-pole arrangement while permitting outputs to be connected together. It is called tristate TTL, because it allows three possible output stage High, Low and high impedance.

* when the transistor Q_3 is ON when output is high. and transistor Q_4 is ON output voltage of transistor Q_4 is low.

* In high impedance state both transistor Q_3 and Q_4 in the totem pole arrangement are turned off.

* The result of output is open or floating. neither low or high

* 'A' is the normal logic input whereas E is an ENABLE input.

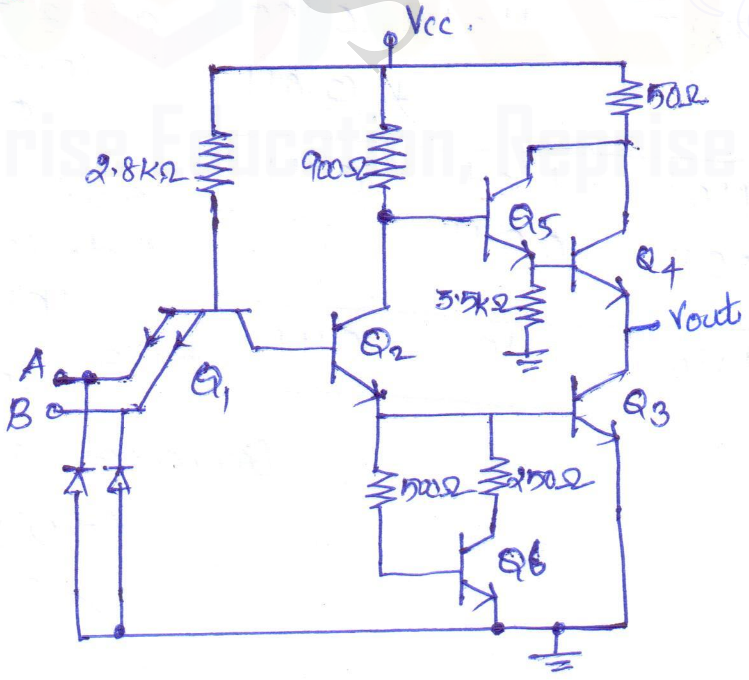
* when ENABLE input is high the circuit works as a normal inverter.

* when E is high the state of transistor Q_1 depends on logic input A

* when ENABLE input is low, regardless of the state of logic input A, the base emitter junction of Q_1 is forward biased and result it turn ON.

* when ENABLE input is low, both transistor Q_3, Q_4 is OFF and output is at high impedance state.

Schottky TTL Gate:



Advantages of TTL:

* High speed
* Propagation delay 10ns
* Moderate power dissipation.
* Low cost.
* Moderate packaging density

Disadvantages of TTL:

* Higher power dissipation than CMOS.
* Lower noise immunity than CMOS
* Less fanout than CMOS.

6. Perform the following addition using BCD and Excess-3 addition (205+569) [APRIL/MAY 2015]

Soln:

BCD addition

```
  205      0010    0000    0101    - BCD for 205
+ 569      0101    0110    1001    - BCD for 569
 ----     ----------------------
  774      0111    0110    1110     - 1110 > 9 So
                         + 0110       add 6
          ----------------------
                           0100    - corrected sum
           0111    0111              (774)
          ----------------------
```

Exless-3 addition.

```
  205      0101    0011    1000    - Ex-3 for 205
+ 569      1000    1001    1100      Ex-3 for 569
          ----------------------
           1101    1101    0100      add 3 to
                         + 0011        correct
          ----------------------       0100
           1101    1101    0111     - Subtract 3
           0011    0011                to correct
          ----------------------        1101
           1010    1010    0111     Ex-3 for
          ----------------------
                                   Corrected Sum
                                      (774)
```

Code conversion

Binary to Decimal conversion

$$(10011011)_2 = 1\times2^7+0\times2^6+0\times2^5+1\times2^4+1\times2^3+0\times2^2+1\times2^1+1\times2^0$$

$$= 128+0+0+16+8+0+2+1$$

$$(10011011)_2 = (155)_{10}$$

(or)

$$(10011011)_2 = \begin{array}{cccccccc} 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}$$

$$= 128+0+0+16+8+0+2+1$$

$$=(155)_{10}$$

Decimal to Binary conversion

$$(156)_{10} = \begin{array}{r|l l} 2 & 156 & \\ 2 & 78 & -0 \\ 2 & 39 & -0 \\ 2 & 19 & -1 \\ 2 & 9 & -1 \\ 2 & 4 & -1 \\ 2 & 2 & -0 \\ & 1 & -0 \end{array}$$

$$(156)_{10} = (10011100)_2$$

Binary to octal conversion

$$(0011000 1100)_2 = \underline{\boxed{0}00}\ \underline{110}\,\underline{001}\,\underline{100}$$

$$= \quad 0 \quad 6 \quad 1 \quad 4$$

$$= (614)_8$$

Binary to Hexadecimal conversion

(i) $(0100100101)_2 = \underset{1}{\underline{0001}}\,\underset{2}{\underline{0010}}\,\underset{5}{\underline{0101}}$

$$=(125)_{16}$$

(ii) $(1010110111100010)_2 = \underline{1010}\ \underline{1101}\ \underline{1110}\ \underline{0010}$

$= A\ \ D\ \ E\ \ 2$

$= (ADE2)_{16}$

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Octal	0	1	2	3	4	5	6	7	10	11	12	13	14	15	16	17

Octal to Binary conversion.

$(536)_8 = (101\ 011\ 110)_2$

Hexadecimal to Binary conversion

$(5ABC)_{16} = 0101\ 1010\ 1011\ 1100$

$= (0101101010111100)_2$

Binary addition

Rules of binary addition

$0+0 = 0$
$0+1 = 1$
$1+0 = 1$
$1+1 = 0$, and carry 1 to the next more significant bit

(i) $00011010 + 0000\ 1100$

$= 0011010$
$\ \ 00001100$

128 64 32 16 8 4 2 1
$00011010 = 26_{10}$
$00001100 = 12_{10}$

$00100110 = (38)_{10}$

(ii) $101 + 110$

101 — 5
110 — 6

Carry ← 1 011 = 11

Rules of binary subtraction.

0 - 0 = 0
0 - 1 = 1, and borrow 1 from the next more significant bit
1 - 0 = 1
1 - 1 = 0

(i) 00100101 - 00010001 = 00010100

```
 128 64 32 16 8 4 2 1
  00100101   = 37
  00010001   = 17
 ----------   -----
  00010100   = 20
```

$(00010100)_2 = (20)_{10}$

Rules of multiplication

0 × 0 = 0
0 × 1 = 0
1 × 0 = 0
1 × 1 = 1, and no borrow bits

(i) 00101001 × 00000110

```
        00101001        41
        00000110         6
       ----------
        00000000
       00101001
      00101001
     ------------      ----
     0011110110         246
```

Binary division.

00101010 ÷ 00000110

= 111

```
          111
      ---------
  110 ) 00101010
          110
       --------
          1001
           110
         ------
           0110
            110
           -----
              0
```

2's complement addition.

5 + (-3) = 2

```
   101 - 5          3 → 011
   011 - 3            → 100 → 1's complement
                           1
                        101 - 2's complement
   101
   101
[1] 010 - 2
```

2's complement subtraction

7-12 = (-5)

```
7 → 111  ,  12 → 1100
                 0011  → 1's
                    1
                 0100  → 2's
  0111
  0100
1| 011
 0 1 0 0
       1
   101
```

UNIT-II COMBINATIONAL CIRCUITS
TWO MARKS

1. State De Morgan's theorem. (May 2014, Nov 2012)

De Morgan suggested two theorems that form important part of Boolean algebra. The Complement of a product is equal to the sum of the complements.

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

The complement of a sum term is equal to the product of the complements.

$(A + B)' = A'B'$

2. Reduce A'B'C' + A'BC' + A'BC (May2015, May 2011)

$A'B'C' + A'BC' + A'BC = A'C'(B' + B) + A'BC$

$= A'C' + A'BC \ [A + A' = 1]$

$= A'(C' + BC) = A'((C'+B)(C'+C))$ Since $C'+C=1$

$= A'(C' + B) \ [A + A'B = A + B]$

3. What is a karnaugh map? And it limitations

A karnaugh map or k map is a pictorial form of truth table, in which the map diagram is made up of squares, with each squares representing one minterm of the function.

A limited to six variable map(i.e.) more than six variables involving expression are not Boolean expression represented in standard form.

4. Convert the given expression in canonical SOP form (Dec 2015, May 2015)

Y = AC + AB + BC

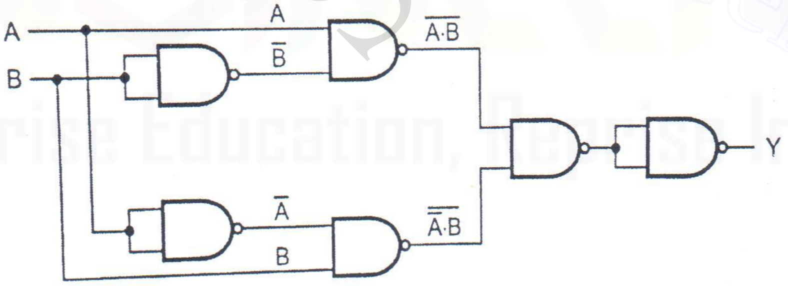
$Y = AC + AB + BC = AC\ (B + B') + AB\ (C + C') + (A + A')\ BC$

$= ABC + ABC' + AB'C + AB'C' + ABC + ABC' + ABC$

$= ABC + ABC' + AB'C + AB'C'\ [A + A = 1]$

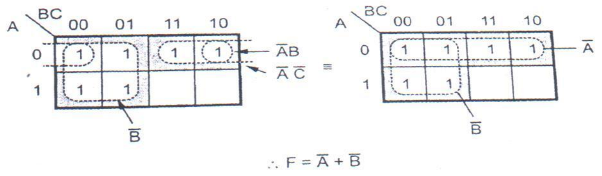
5. Implement EX-NOR gate using only NAND gate. (Nov 2015)

Y= (A+B)'



A
B
A
B̄
$\overline{A \cdot \overline{B}}$
Y
Ā
B
$\overline{\overline{A} \cdot B}$

6. Identify the Redundant term for F=B'+A'B+A'C' using K-map. (Nov 2014)



The redundant term in given expression is $\overline{A}\ \overline{C}$.

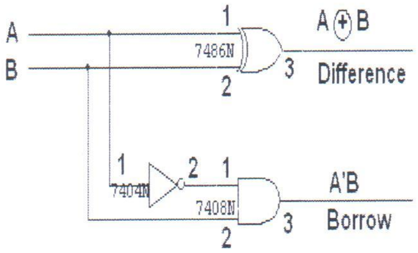
7. Define multiplexer? And its application. (Dec 2014, May 2013)

Multiplexer is a digital switch. If allows digital information from several sources to be routed onto a single output line.

Application: It used in route data with in a computer, function generator, used in data communication for several computers.

8. Draw the truth table and logic gates for Half Subtractor. (Dec 2012)

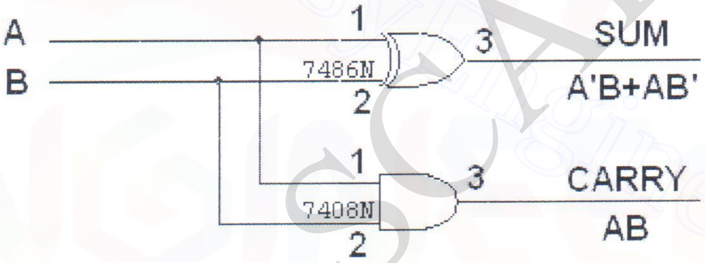
A	B	BORROW	DIFFERENCE
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0



9. Define half adder and full adder (Nov 2011)

The logic circuit that performs the addition of two bits is a half adder. The circuit that performs the addition of three bits is a full adder.

10. Draw the logic gates for Half Adder. (Dec 2012)



11. Differentiate Encoder and Decoder (Nov 2011)

Encoder	Decoder
The output lines generate the binary code, corresponding to the input value.	The output lines is activated corresponding to the binary input.
Input of the encoder is a decoded information presented as 2^n inputs producing n possible outputs	Input of the decoder is an encoded information presented as n inputs producing 2^n possible outputs
The input code generally has more bits than the output code	The input code generally has fewer bits than the output code

12. Write the application of Decoder. (Nov 2014)

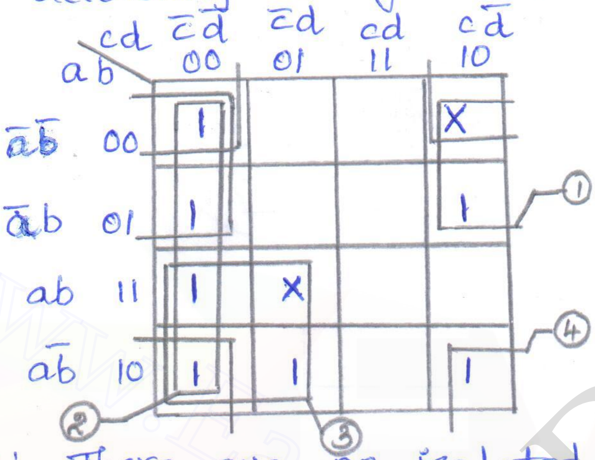
- Code converters
- Implementation of combinational circuits
- Address decoding
- BCD to 7-segment decoder

UNIT-2

1 (i) Minimize the function $F(a,b,c,d) = \Sigma(0,4,6,8,9,10,12)$ with $d = \Sigma(2,13)$. Implement the function using only NOR gates. [NOV /DEC 2014]

Soln

Step 1: The k-map for four variables and it is plotted according to given minterms



ab \ cd	$\bar c\bar d$ 00	$\bar c d$ 01	cd 11	$c\bar d$ 10
$\bar a\bar b$ 00	1			X
$\bar a b$ 01	1			1
ab 11	1	X		
$a\bar b$ 10	1	1		1

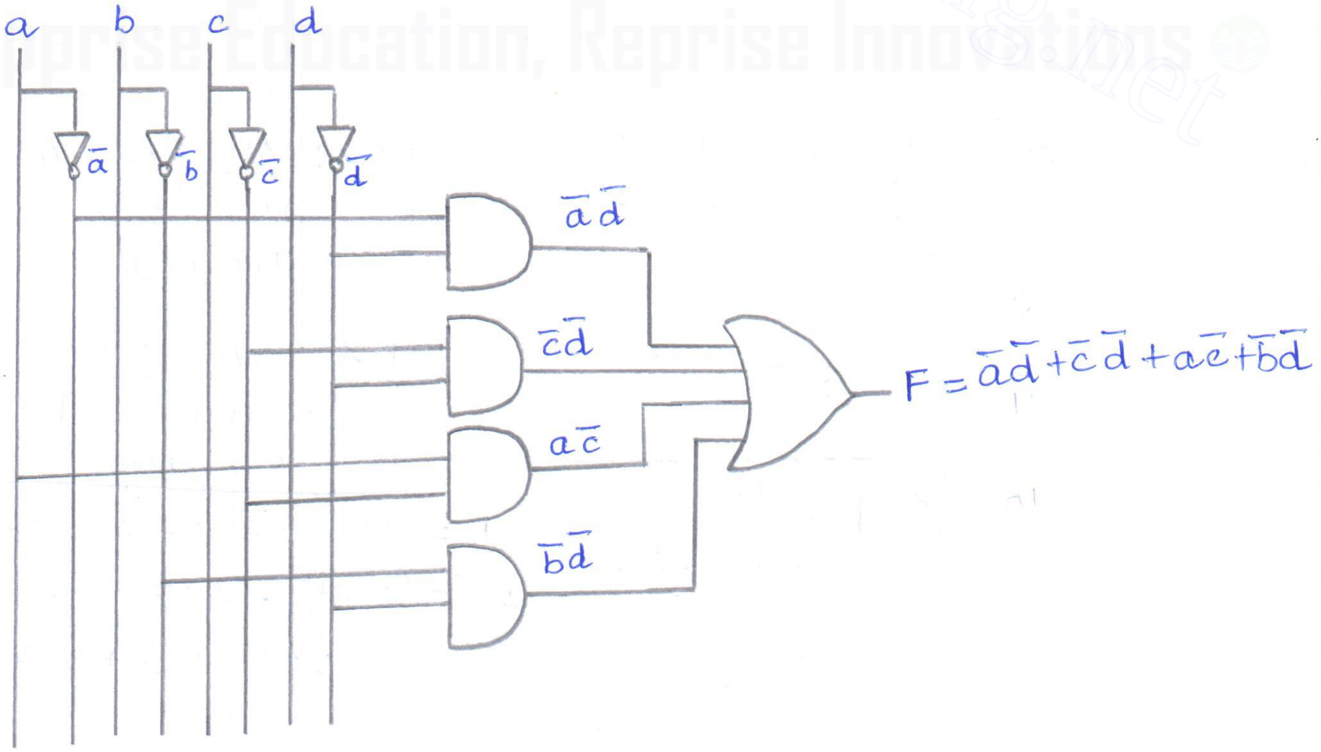
Step 2: There are no isolated 1's.

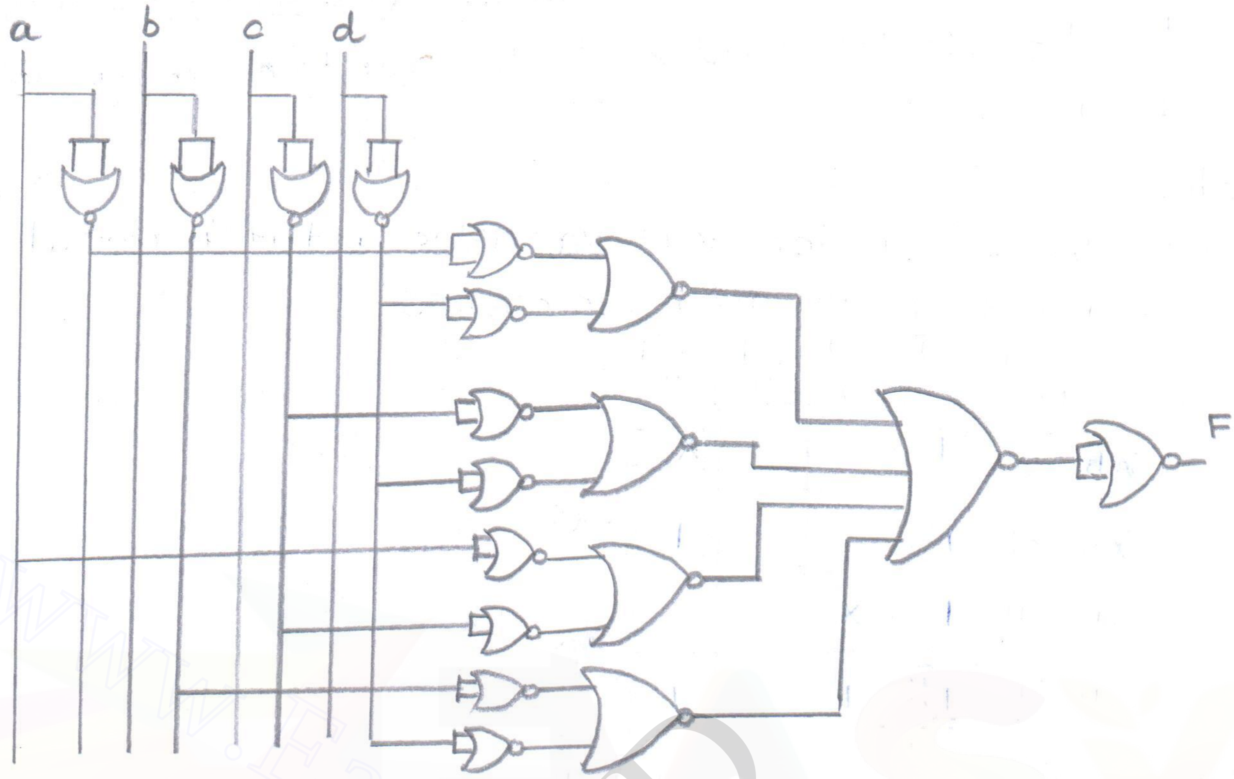
Step 3: Dont care considered as 1's and all the 1's have been grouped.

Step 4: We get simplified equation as,

$$F = \bar a\bar d + \bar c\bar d + a\bar c + \bar b\bar d$$

step 5: Draw the logic diagram





1 (ii) Reduce the following function using k-map and implement the function using NAND gates.

Soln: $F(A,B,C,D) = \Pi M (0, 2, 3, 8, 9, 12, 13, 15)$

[APRIL/MAY 2015]

Step 1 : The k-map for four variables and it is plotted according to given maxterms

Step 2 : There are no isolated 0's

AB \ CD	00	01	11	10
00	0		0	0
01				
11	0	0	0	
10	0	0		

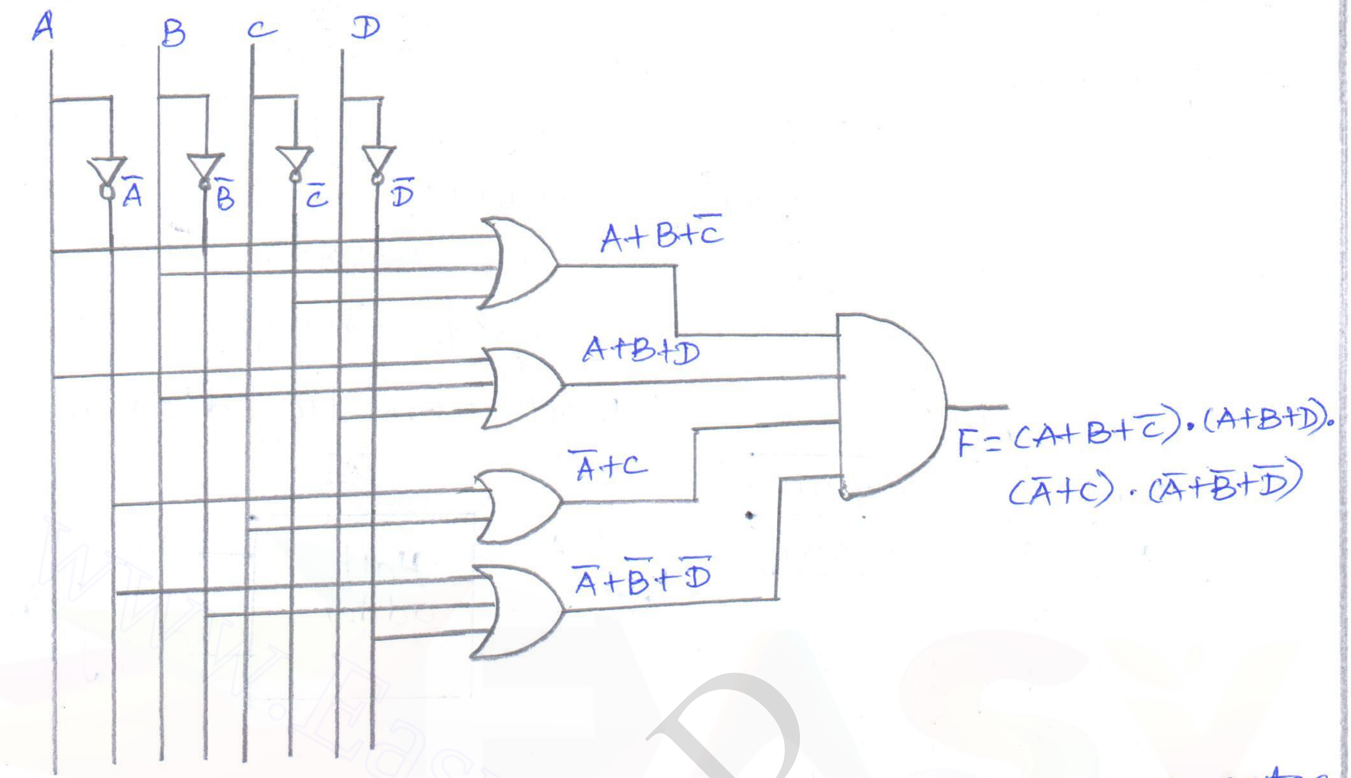
Step 3 : All the 0's have been grouped

Step 4 : We get simplified equation.

$F = \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{D} + A\bar{C} + ABD$

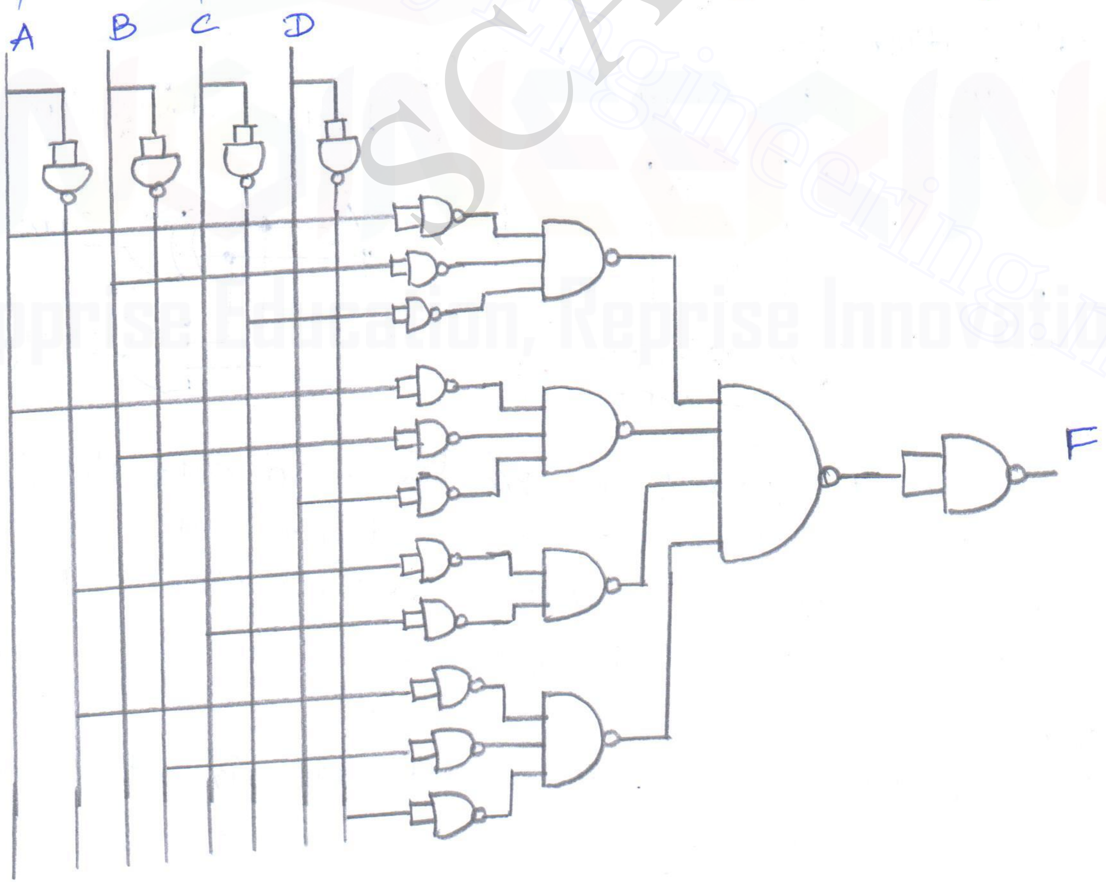
$F = (A+B+\bar{C}) \cdot (A+B+D) \cdot (\bar{A}+C) \cdot (\bar{A}+\bar{B}+\bar{D})$

Step 5: Draw the logic diagram.



$F=(A+B+\bar{C}).(A+B+D).(\bar{A}+C).(\bar{A}+\bar{B}+\bar{D})$

Step 6: Implement the logic diagram using NAND gates.



2. Design halfadder, full adder, half subtractor and Full Subtractor. [NOV/DEC 2015]

Half adder.

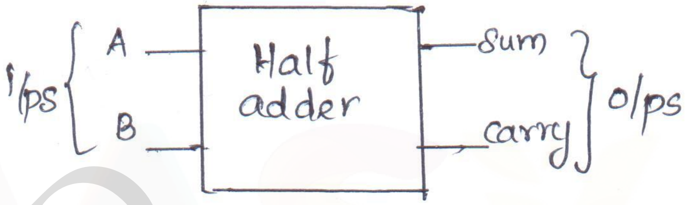
The circuit perform the addition of two binary digit is called half adder.

The truth table gives the relation between the input and output variables for half-adder operation.

Truth table

Inputs		Outputs	
A	B	Sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Block schematic diagram



K-map simplification for sum and carry

For carry

A \ B	0	1
0	0	0
1	0	1

Carry = AB

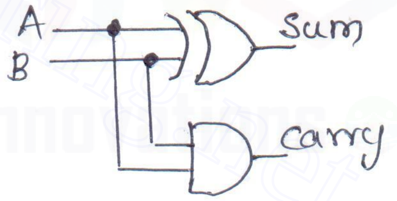
For Sum

A \ B	0	1
0	0	1
1	1	0

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

$= A \oplus B$

Logic diagram.



Limitations:

In multidigit addition we have to add two bits along with the carry of previous digit addition. Effectively such addition requires addition of 3-bits. This not possible with halfadder.

Full adder.

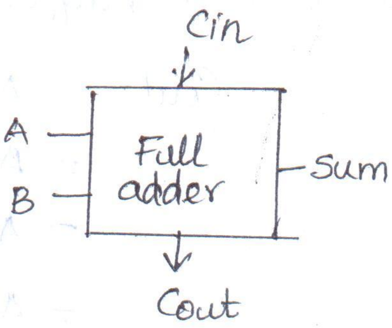
A full adder is a combinational circuit that forms the arithmetic sum of three input bits.

It consists of three inputs and two outputs.

Truth table

Inputs			Outputs	
A	B	Cin	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

Block diagram



K map simplification

A \ BCin	00	01	11	10
0	0	1	0	1
1	1	0	1	0

For Sum

A \ BCin	00	01	11	10
0	0	0	1	0
1	0	1	1	1

For carry

$Sum = \bar{A}\bar{B}Cin + \bar{A}B\overline{Cin} + A\bar{B}\overline{Cin} + ABCin$

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

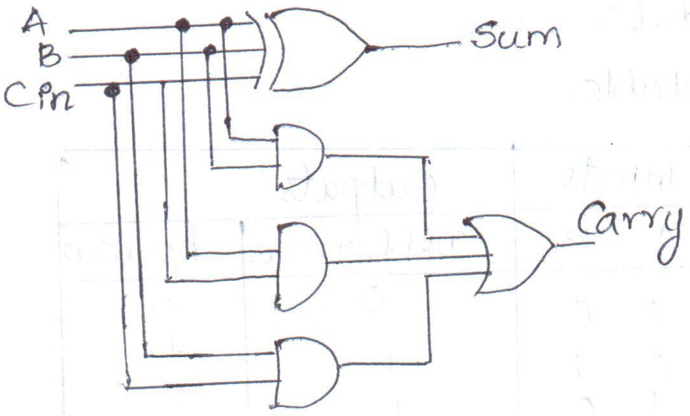
$= Cin(A \odot B) + \overline{Cin}(A \oplus B)$

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

$Sum = A \oplus B \oplus Cin$

$Cout = AB + ACin + BCin$

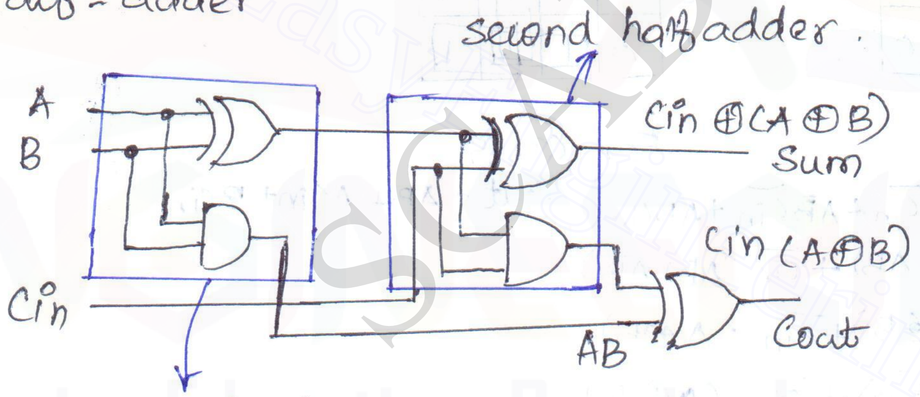
Implementation of full-adder



A full adder can also be implemented with with two halfadders

$$C_{out} = AB + ACin + BCin$$
$$= AB + ACin(B+\overline{B}) + BCin(A+\overline{A})$$
$$= AB + ABCin + A\overline{B}Cin + ABCin + \overline{A}BCin$$
$$= AB(1+Cin+Cin) + A\overline{B}Cin + \overline{A}BCin$$
$$= AB + A\overline{B}Cin + \overline{A}BCin$$
$$= AB + cin(A\overline{B}+B\overline{A})$$
$$= AB + Cin(A\oplus B)$$

Implementation of a fulladder with two half-adder



First half-adder.

Half subtractor

A half subtractor is a combinational circuit that subtracts two bits and produces their difference

Truth table

inputs		Outputs	
A	B	Differenie	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

K-map simplification for half subtractor.

A \ B	0	1
0	0	1
1	0	0

For borrow,

Borrow = $\bar{A}B$

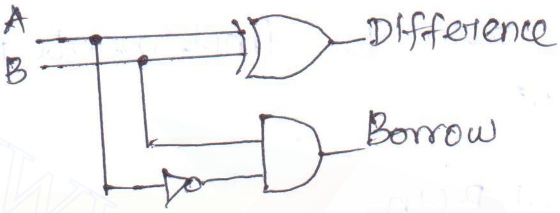
A \ B	0	1
0	0	1
1	1	0

For difference

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

$= A \oplus B$.

Logic diagram:



Limitations

In multidigit subtraction, we have to subtract two bits along with the borrow of previous digit subtraction.

In Effective subtraction requires three bits.

Full subtractor.

A full subtractor is a combinational circuit that performs a subtraction between three bits.

Truth table.

Inputs			Outputs	
A	B	Cin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-map simplification.

A \ BCin	00	01	11	10
0	0	1	0	1
1	1	0	1	0

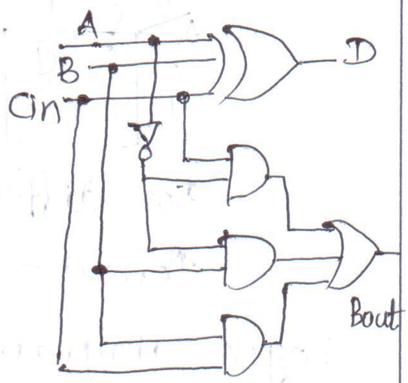
$D = \bar{A}\bar{B}\,Cin + \bar{A}B\,\overline{Cin} + A\bar{B}\,\overline{Cin} + AB\,Cin$

A \ BCin	00	01	11	10
0	0	1	1	1
1	0	0	1	0

$Bout = \bar{A}Cin + \bar{A}B + BCin.$

$$D = \bar{A}\bar{B}Cin + \bar{A}B\overline{Cin} + A\bar{B}\overline{Cin} + ABCin$$
$$= Cin\ (\bar{A}\bar{B}+AB) + \overline{Cin}\ (\bar{A}B+A\bar{B})$$
$$= Cin\ (A \odot B) + \overline{Cin}\ (A \oplus B).$$
$$= Cin\ (\overline{A \oplus B}) + \overline{Cin}\ (A \oplus B)$$
$$= Bin \oplus (A \oplus B)$$

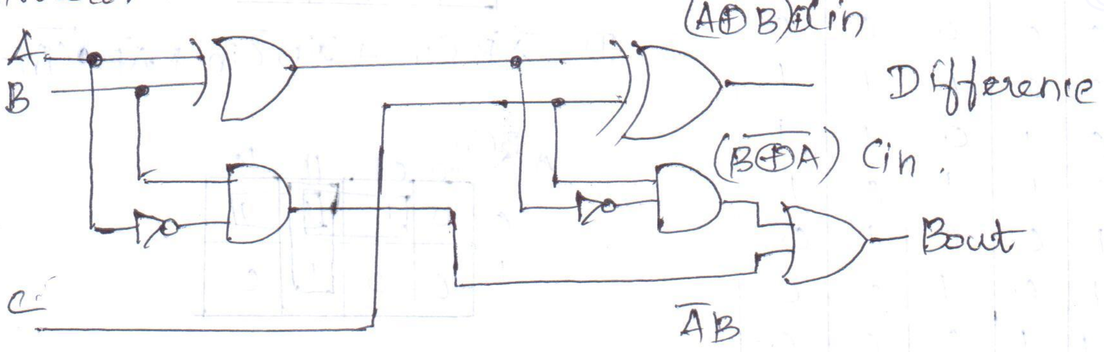
Logic diagram



A full subtractor can also be implemented with two half subtractors

$$Bout = \bar{A}\ Cin + \bar{A}B + BCin$$
$$= \bar{A}\ Cin\ (B+\bar{B}) + \bar{A}B + BCin\ (A+\bar{A})$$
$$= \bar{A}B\ Cin + \bar{A}\bar{B}\ Cin + \bar{A}B + ABCin + \bar{A}BCin$$
$$= \bar{A}B\ (Cin + 1 + Cin) + \bar{A}\bar{B}\ Cin + ABCin$$
$$= \bar{A}B + \bar{A}\bar{B}Cin + ABCin.$$
$$= \bar{A}B + Cin\ (\bar{A}\bar{B} + AB)$$
$$= \bar{A}B + Bin\ (\overline{A \oplus B})$$

Implementation of full subtractor with two half Subtractor



3. Binary to gray code converter [NOV /DEC 2014]

The gray code is often used in digital systems because it has the advantage that only one bit in the numerical representation changes between successive numbers.

Step 1: Truth table

Binary code				Gray code			
D	C	B	A	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

Step 2: Using K-map simplification

DC \ BA	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$G_0 = \bar{B}A + B\bar{A}$

$= B \oplus A$

For G_0

DC \ BA	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$G_1 = \bar{C}B + C\bar{B}$

$= C \oplus B$

For G_1

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$G_2 = \bar{D}C + D\bar{C}$

$= D \oplus C$

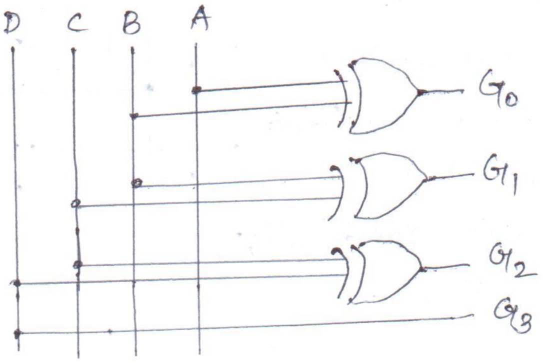
For G_2

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$G_3 = D$

For G_3

Step 3: Draw the logic diagram.



Gray code to Binary code converter

Soln:

step 1 : Truth table

Gray code				Binary code			
G_3	G_2	G_1	G_0	D	C	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
1	0	0	0	1	1	1	1
1	0	0	1	1	1	1	0
1	D	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	0	1	0	1	1
1	1	1	1	1	0	1	0

step 2 : Using k-map:

G_3G_2 \ G_1G_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

For A

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

For B

G_3G_3 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

For C

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

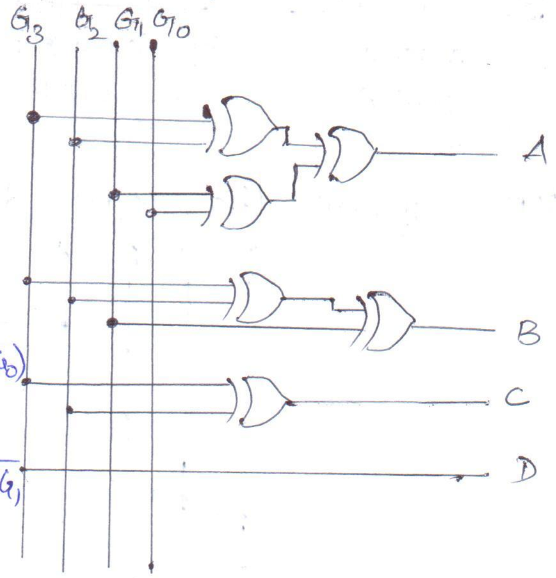
For D

$$C = \bar{G}_3 G_2 + G_3 \bar{G}_2$$
$$= G_3 \oplus G_2$$

$$D = G_3$$

$$A = (\bar{G}_3 G_2 + G_3 \bar{G}_2)\bar{G}_1 \bar{G}_0 + (\bar{G}_3 \bar{G}_2 + G_3 G_2)\bar{G}_1 G_0$$
$$+ (\bar{G}_3 G_2 + G_3 \bar{G}_2) G_1 G_0 + (\bar{G}_3 \bar{G}_2 + G_3 G_2) G_1 \bar{G}_0$$
$$= (G_3 \oplus G_2)\bar{G}_1 \bar{G}_0 + (G_3 \odot G_2)\bar{G}_1 G_0$$
$$+ (G_3 \oplus G_2) G_1 G_0 + (G_3 \odot G_2) G_1 \bar{G}_0$$
$$= (G_3 \oplus G_2)(\bar{G}_1 \bar{G}_0 + G_1 G_0) +$$
$$(G_3 \odot G_2)(\bar{G}_1 G_0 + G_1 \bar{G}_0)$$
$$= (G_3 \oplus G_2)(\overline{G_1 \oplus G_0}) + \overline{(G_3 \odot G_2)} \cdot (G_1 \oplus G_0)$$
$$= (G_3 \oplus G_2) \oplus (G_1 \oplus G_0)$$

$$B = (\bar{G}_3 \bar{G}_2 + G_3 G_2) G_1 + (\bar{G}_3 G_2 + G_3 \bar{G}_2)\bar{G}_1$$
$$= (G_3 \odot G_2) G_1 + (G_3 \oplus G_2) \cdot \bar{G}_1$$
$$= G_3 \oplus G_2 \oplus G_1$$



Design a 4-bit BCD to Binary code converter

Step 1: The truth table for BCD to binary converter

B_4	B_3	B_2	B_1	B_0	E	D	C	B	A
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0
0	0	0	1	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0	0
0	0	1	0	1	0	0	1	0	1
0	0	1	1	0	0	0	1	1	0
0	0	1	1	1	0	0	1	1	1
0	1	0	0	0	0	1	0	0	0
0	1	0	0	1	0	1	0	0	1
1	0	0	0	0	0	1	0	1	0
1	0	0	0	1	0	1	0	1	1
1	0	0	1	0	0	1	1	0	0
1	0	0	1	1	0	1	1	0	1
1	0	1	0	0	0	1	1	1	0
1	0	1	0	1	0	1	1	1	1
1	0	1	1	0	1	0	0	0	0
1	0	1	1	1	1	0	0	0	1
1	1	0	0	0	1	0	0	1	0
1	1	0	0	1	1	0	0	1	1

Step 2: Using k-map simplification.

$B_4 = 0$

$B_3B_2 \backslash B_1B_0$	$\bar{B_1}\bar{B_0}$	$\bar{B_1}B_0$	B_1B_0	$B_1\bar{B_0}$
$\bar{B_3}\bar{B_2}$	0	1	1	0
$\bar{B_3}B_2$	0	1	1	0
B_3B_2	X	X	X	X
$B_3\bar{B_2}$	0	1	X	X

$B_4 = 1$

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	X	X	X	X
10	0	1	X	X

$B_4 = 0$

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	X	X	X	X
10	0	0	X	X

$B_4 = 1$

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	X	X	X	X
10	1	1	X	X

For A

$A = B_0$

For B

$B = B_1\bar{B_4} + \bar{B_1}B_4.$

$= B_1 \oplus B_4.$

$B_4=0$

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	X	X	X	X
10	0	0	X	X

$B_4=1$

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

For C

$$C = \bar{B}_4 B_2 + B_2 \bar{B}_1 + B_4 \bar{B}_2 B_1$$

$B_4=0$

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	1	1	X	X

$B_4=1$

B_3B_2 \ B_1B_0	00	01	11	10
00	1	1	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

For D

$$D = \bar{B}_4 B_3 + B_4 \bar{B}_3 \bar{B}_2 + B_4 \bar{B}_3 B_1$$

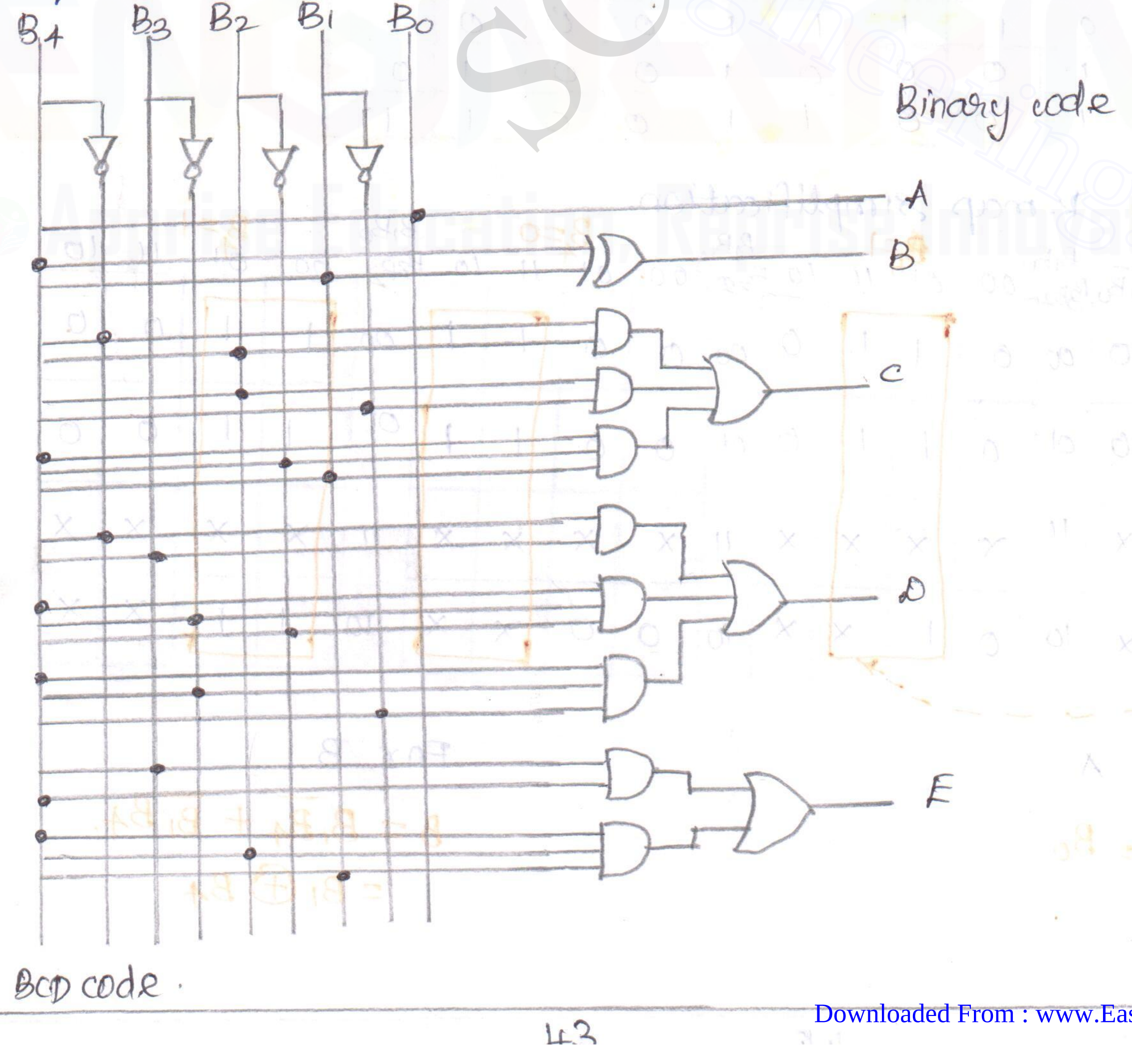
B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	X	X	X	X
10	1	1	X	X

For E

$$E = B_4 B_3 + B_4 B_2 B_1$$

step3: Draw the logic diagram.

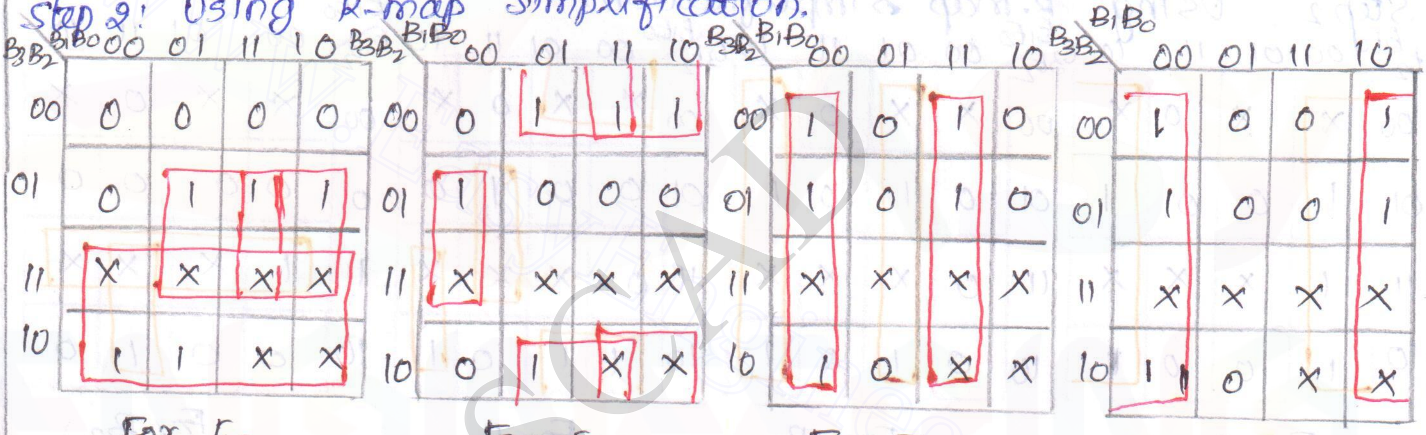


Design a 4-bit BCD to Excess 3 converter. [NOV/DEC 2015], [APRIL/MAY 2015]

Step 1 : Truthtable.

Decimal	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Step 2 : Using k-map simplification.



For E_3

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

For E_2

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

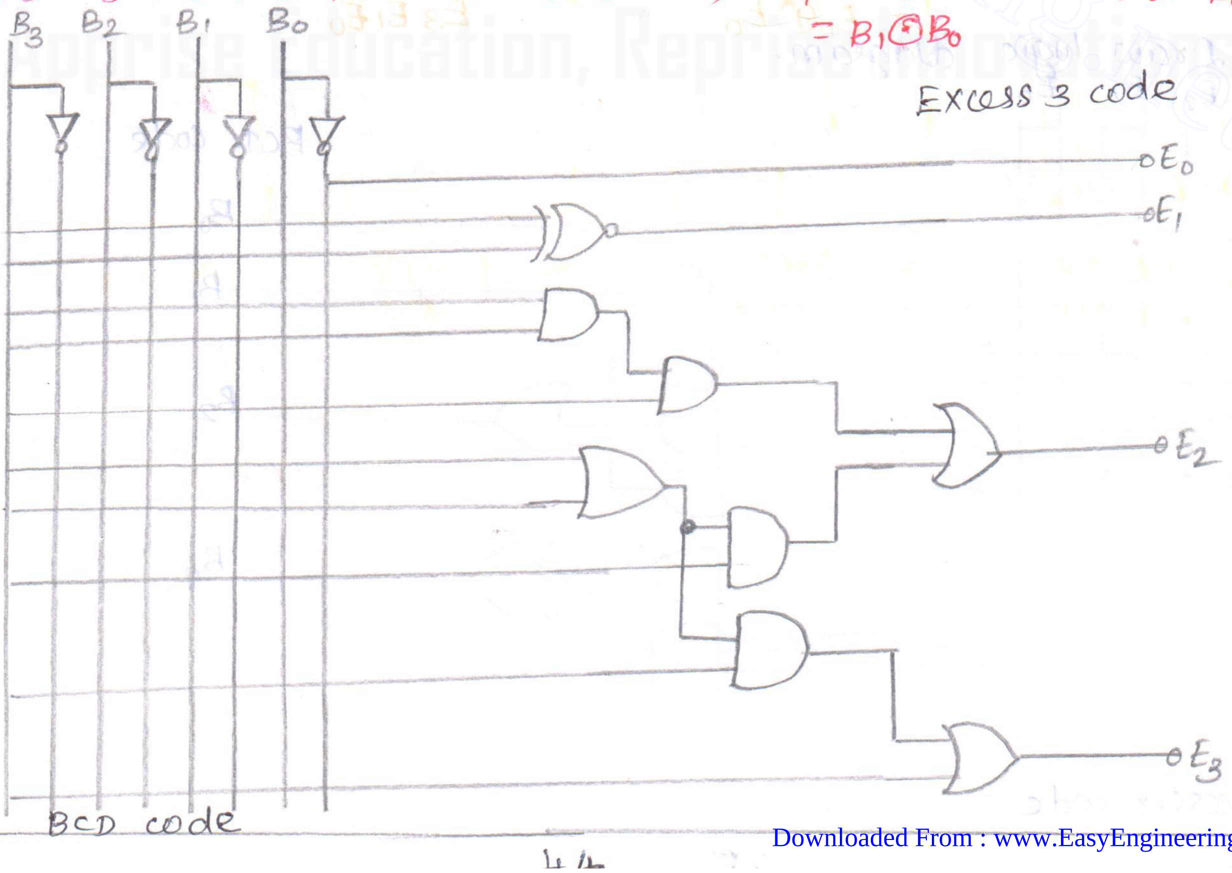
For E_1

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

For E_0

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$E_3 = B_3 + B_2B_0 + B_2B_1$ $E_2 = B_2\bar{B}_1\bar{B}_0 + \bar{B}_2(B_0+B_1)$ $E_1 = \bar{B}_1\bar{B}_0 + B_1B_0$ $= B_1 \odot B_0$ $E_0 = \bar{B_0}$



4. Design a 4-bit Excess-3 code to BCD code converter.

Step 1: Truth table.

E_3	E_2	E_1	E_0	B_3	B_2	B_1	B_0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

Step 2: Using K-map simplification.

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	1	0	0	1
11	1	X	X	X
10	1	0	0	1

For B_0

$B_0 = \overline{E_0}$

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	0	1	0	1
11	0	X	X	X
10	0	1	0	1

For B_1

$B_1 = \overline{E_1}E_0 + E_1\overline{E_0}$

$= E_1 \oplus E_0$

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	0	0	1	0
11	0	X	X	X
10	1	1	0	1

For B_2

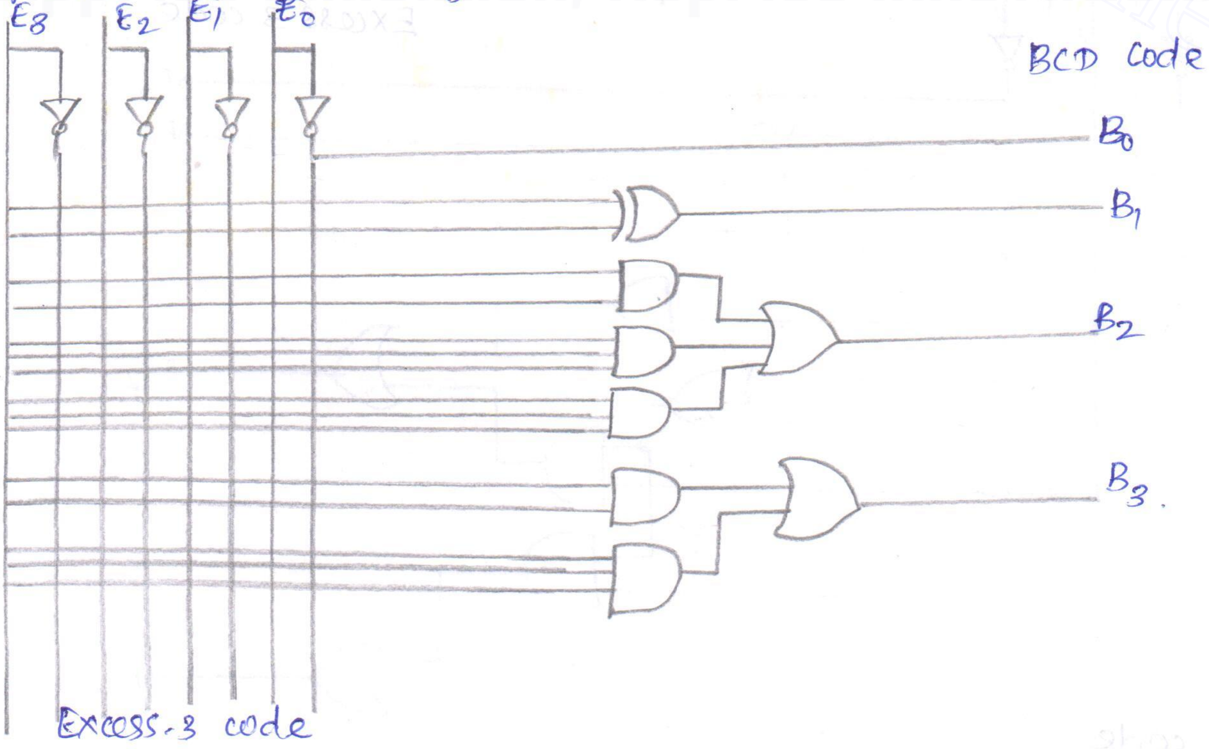
$B_2 = \overline{E_2}\,\overline{E_1} + E_2E_1E_0 + E_3E_1\overline{E_0}$

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	0	0	0	0
11	1	X	X	X
10	0	0	1	0

For B_3

$B_3 = E_3E_2 + E_3E_1E_0$

Step 3: Draw logic diagram.

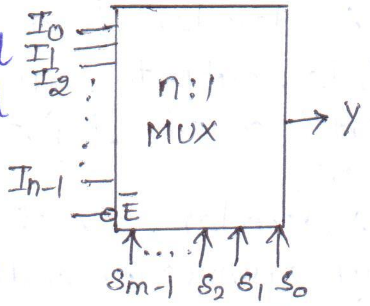


5. (i) Multiplexer

The multiplexer is a combinational logic circuit with multiple inputs and a single output.

Selectors select one input at a time and send it to the output line.

m select inputs are required, where $n = 2^m$. A multiplexer is also known as data selector.



Block diagram

Applications :

- ✱ Simplification of logic expression is not required
- ✱ It minimizes the IC count. logic design is simplified

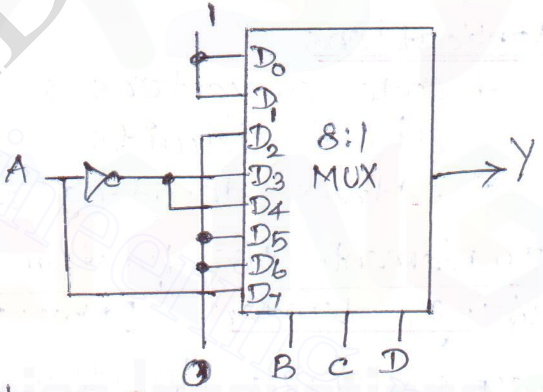
Implement the following Boolean function using 8:1 mux

$F(A,B,C,D) = \sum m(0,1,3,4,8,9,15)$ [APRIL/MAY 2015]

Soln:

Implementation table

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	⓪	①	2	③	④	5	6	7
A	⑧	⑨	10	11	12	13	14	⑮
	1	1	0	A	$\bar{A}$	0	0	A



Implement the following Boolean function with 8×1 mux and external gates.

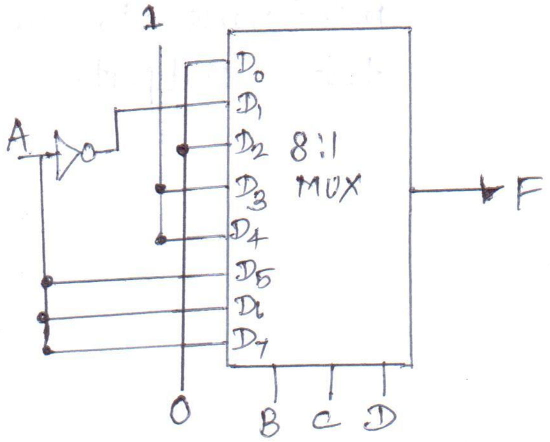
$F(A,B,C,D) = \sum m(1,3,4,11,12,13,14,15)$

[NOV/DEC 2015]

Soln:

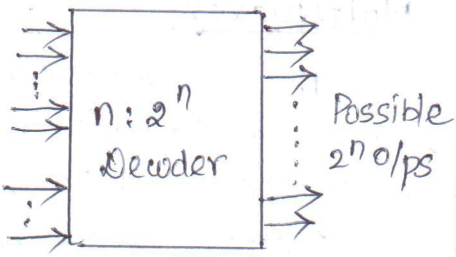
Implementation table.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	①	2	③	④	5	6	7
A	8	9	10	⑪	⑫	⑬	⑭	⑮
	0	$\bar{A}$	0	1	1	A	A	A



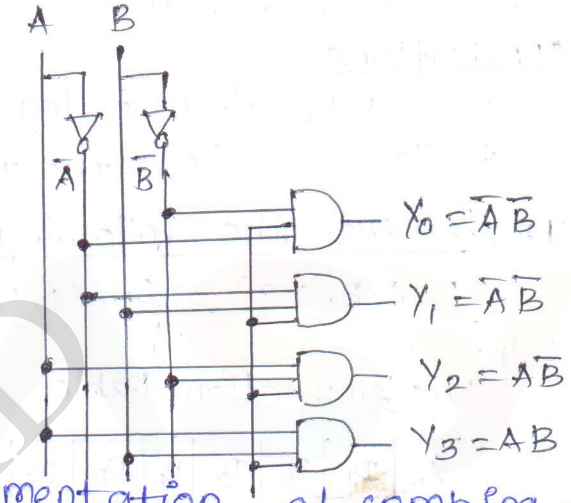
Decoders

A decoder is a multiple-input, multiple-output logic circuit which converts coded input into coded outputs.



A decoder which has an n-bit binary number (or) code and a one activated output out of 2^n output code is called binary decoder.

Input			Outputs			
EN	A	B	Y_3	Y_2	Y_1	Y_0
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0



Applications.

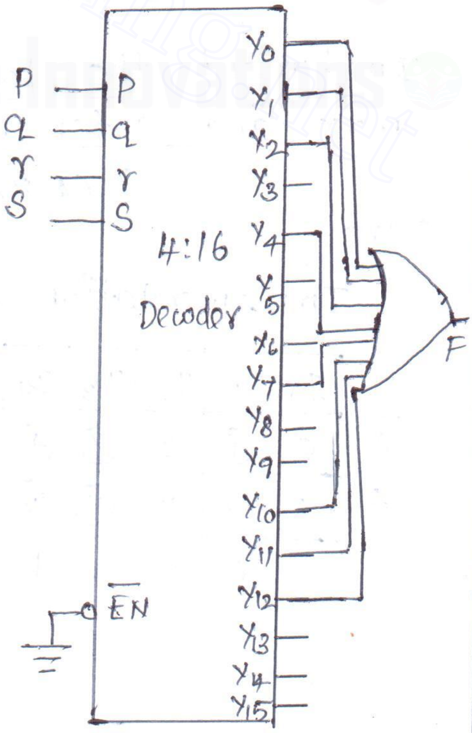
* code converters & Implementation of combinational circuits

* Address decoding & BCD to 7-segment decoder.

5) (ii) Implement the function $F(P,q,r,s) = \Sigma(0,1,2,4,7,10,11,12)$ using decoder. [NOV/DEC 2014]

Step 1: connect function variables as inputs to the decoder

Step 2: Logically OR the outputs correspond to present minterms to obtain the output.



UNIT-III SYNCHRONOUS SEQUENTIAL CIRCUITS
TWO MARKS

1. What is the operation of RS flip-flop?& Disadvantages (May 2014)

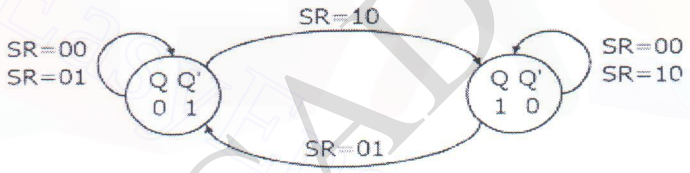
When R input is low and S input is high the Q output of flip-flop is set. When R input is high and S input is low the Q output of flip-flop is reset. When both the inputs R and S are low the output does not change. When both the inputs R and S are high the output is unpredictable.

The **disadvantage** of the **SR flip-flop** is that both inputs shouldn't be HIGH when the clock is triggered. This is considered an invalid input condition, and the resulting output isn't predictable if this condition occurs

2. What is edge-triggered flip-flop?(Nov 2015)

The problem of race around condition can solved by edge triggering flip flop. The term edge triggering means that the flip-flop changes state either at the positive edge or negative edge of the clock pulse and it is sensitive to its inputs only at this transition of the clock.

3. Draw the state diagram and truth table of SR flip flop. (Nov 2015)

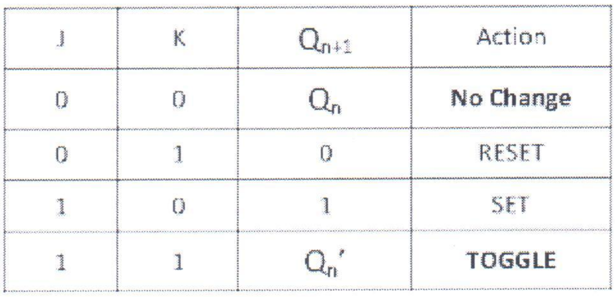


CP	S	R	Q_n	Q_{n+1}	State
	0	0	0	0	No Change (NC)
	0	0	1	1	
	0	1	0	0	Reset
	0	1	1	0	
	1	0	0	1	Set
	1	0	1	1	
	1	1	0	X	Indeterminate
	1	1	1	X	
0	X	X	0	0	No Change (NC)
0	X	X	1	1	

4. Draw the excitable for the conversion of T to D flip flop. (Nov 2014 & May 2015)

Input	Present state	Next state	Flip-flop input
D	Q_n	Q_{n+1}	T
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

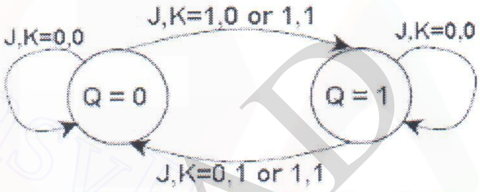
5. Draw the truth table for JK flip flop (May 2013)



J	K	Q_{n+1}	Action
0	0	Q_n	No Change
0	1	0	RESET
1	0	1	SET
1	1	Q_n'	TOGGLE

6. Give the characteristic equation and state diagram of JK flip-flop. (May 2012)

Characteristic equation Q_{n+1}=J Q'_n + K' Q_n



7. The JK flip-flop is an universal flip-flop. Justify. (Nov 2012)

The JK flip-flop is called an universal flip-flop because it can be easily configured to work as any other flip-flops such as T flip-flop, D flip-flop and SR flip-flop.

8. Define sequential circuit?(May 2014)

In sequential circuits the output variables dependent not only on the present input variables but they also depend up on the past history of these input variables.

9. Give the comparison between combinational circuits and sequential circuits.

Combinational circuit	Sequential circuit
When the logic gates connected together to produce specified output for the specified in put variables.	The output is not only depend upon the input variables and also depend upon past history of the input variables.
Combinational circuits are not have storage elements	Sequential circuit have storage elements
There is no clock pulse	It have clock pulse

10. Difference between Moore and Mealy Model.(Dec 2014)

oore Model	Mealy Model
Its output is a function of present only	It output is a function of present state as well as present input.
Input changes does not affect the output	Input changes may affect the output of the circuit
It requires more number of states for implementing same function	It requires less number of states for implementing same function

11. Give the comparison between synchronous & Asynchronous counters. (May 2012, 2011)

Asynchronous counters	Synchronous counters
In this type of counter flip-flops are Connected in such a way that output of 1st Flip-flop drives the clock for the next flipflop	In this type there is no connection between output of first flip-flop and clock input of the next flip – flop
All the flip-flops are not clocked Simultaneously	All the flip-flops are clocked simultaneously

12. Define race around condition

In JK flip-flop output is fed back to the input. Therefore change in the output results change in the input. Due to this in the positive half of the clock pulse if both J and K are high then output toggles continuously. This condition is called race around condition.

13. Define Dead lock. How it is avoided? (May 2012) (Nov 2014)

In a counter if the next state of some unused state is again an unused state and if by chance the counter happens to find itself in the unused states and never arrived at a used state then the counter is called dead lock conditions or lock out conditions.

To avoid lockout, the counter should be provided with an additional logic circuitry which will force the counter from an unused state to the next state as initial state.

14. How many flip-flops are required to design mod 25 counters? (May 2013)

$2^n \geq 25 \quad n=5$

Thus, 5 flip flop are required to design mod 25 counter.

15. Write the rules followed by state assignment (May 2015)

1. States having the same Next states for a given input condition should have assignments which can be grouped into logically adjacent cell in a K-map.

2. State that are the NEXT states of a single state should have assignment which can be grouped into logically adjacent cells in a K-map.

1. Explain the circuit of a SR flipflop and explain its operation. [NOV/DEC 2014]

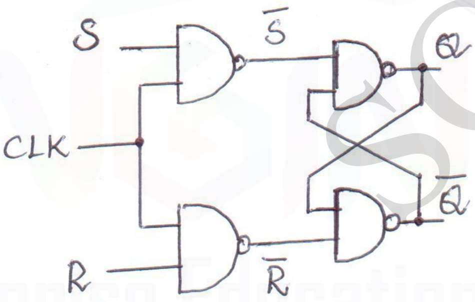
Flipflop is a one bit memory cell that has only two states either logic '0' (or) logic '1'.

Types of flipflop :

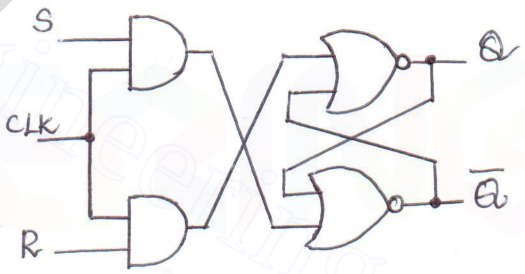
- ✱ Set - Reset flipflop (SR)
- ✱ Data flip flop (D)
- ✱ Jump and kick flipflop (JK)
- ✱ Toggle flipflop (T)

SR Flipflop

The circuit output responds to the S and R inputs only at the positive (or) negative edges of the clock pulse. At any other instants of time, the SR flipflop will not respond to the changes in input.



SR flipflop using NAND gates



SR flipflop using NOR gates

Case 1 : If S=R=0 and the clock pulse is applied, the output do not change, ie. $Q_{n+1} = Q_n$. This is indicated in the first row of the truth table.

case 2 : If S=0, R=1 and the clock pulse is applied, $Q_{n+1} = 0$. This is indicated in the second row of the truth table.

case 3 : If S=1, R=0 and the clock pulse is applied, $Q_{n+1} = 1$. This is indicated in the third row of the truth table.

Case 4 : If S=R=1 and the clock pulse is applied, the state of the flip flop is undefined and therefore is indicated as Indeterminate in the fourth row of the truth table.

Truth table

CLK	S	R	Q_n	Q_{n+1}	State
1	0	0	0	0	No change
1	0	0	1	1	
1	0	1	0	0	Reset
1	0	1	1	0	
1	1	0	0	1	Set
1	1	0	1	1	
1	1	1	0	x	Indeterminate
1	1	1	1	x	
0	x	x	0	0	No change
0	x	x	1	1	

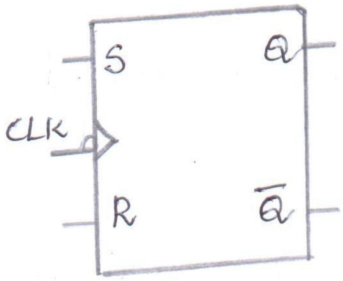
S	R	Q_n	Q_{n+1}
0	0	Q_n	Q_n
0	1	Q_n	0
1	0	Q_n	1
1	1	Q_n	x

charateristic equation.

S \ RQ_n	00	01	11	10
0	0	1	0	0
1	1	1	X	X

$$Q_{n+1} = S + \bar{R} Q_n$$

Logic symbol

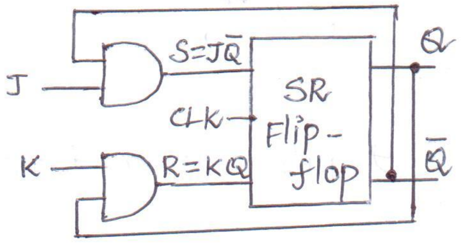


Excitation table

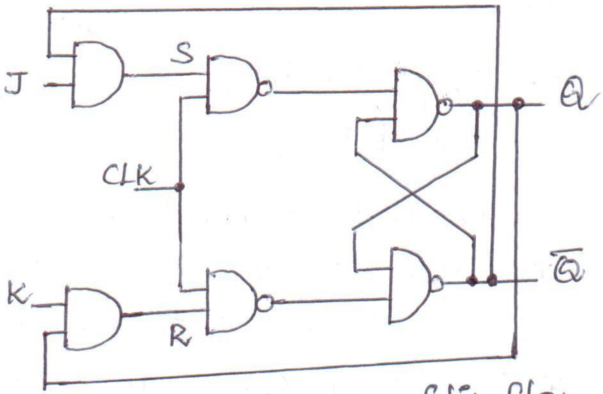
Q_n	Q_{n+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

JK flipflop [APRIL/MAY 2015]

The uncertainty in the state of an SR flip-flop when $S=R=1$ can be eliminated by converting it into a JK flipflop:



JK flipflop using SR flipflop

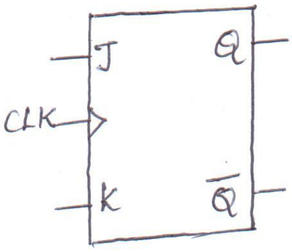


Clocked JK flipflop

case 1: $J=K=0$, output does not change.
when $J=K=0$, $S=R=0$ and according to truth table of SR flip-flop there is no change in the output

case 2: $J=1$ and $K=0$ ie set state
when $J=1$, $K=0$. and according to truth table of SR flip-flop it is set state and the output Q will be 1.

case 3: $J=0$ and $K=1$ ie reset state
when $J=0$, $K=1$ and according to truth table of SR flip-flop it is reset state and the output Q will be 0.



Q_n	J	K	Q_{n+1}	State
0	0	0	0	No change
0	0	1	0	
0	1	0	1	Set
0	1	1	1	
1	0	0	0	Reset
1	0	1	0	
1	1	0	1	Toggle
1	1	1	0	

⇒

J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	$\bar{Q}_n$

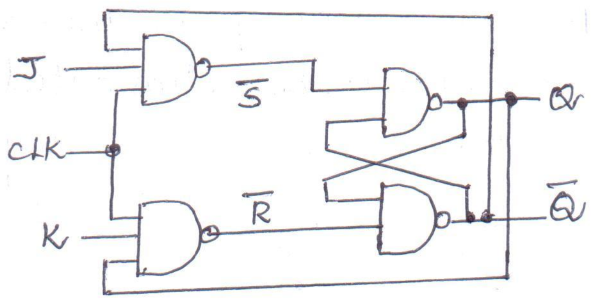
case 4: $J=K=1$, toggles the flipflop output.

characteristic equation.

Qn \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

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

JK flipflop using NAND gates



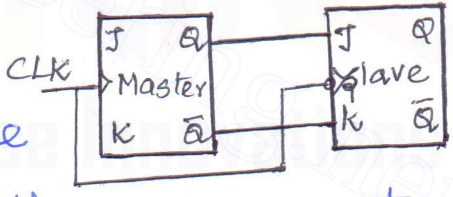
Race - around condition

* when J=K=1, the output goes to toggle either from 0 to 1 or from 1 to 0.

* Initially Q=0 and J=K=1 after a time interval Δt equal to the propagation delay, the output will change to 1. and after another time interval of Δt the output will change back to 0 [∴ Q=0]. This toggling will continue until the flipflop is enabled. At the end of clock pulse the flipflop is disabled Q is uncertain. this situation is referred to race around condition.

Master - slave JK flip-flop

* It consists of clocked JK flipflop as a master and clocked JK flip-flop as a slave.



*The output of the master flipflop is fed as an input to the slave flip-flop.

*clock signal is connected directly to the master flip-flop, but it is connected through inverter to slave flipflop.

* when J=K=0, the output of master remains same at the +ve clock and the output of slave also remain same at negative clock pulse.

* when J=K=1, master toggles on the +ve clock and slave the copies the output of master at -ve clock.

* when J=1 and K=0, Master set +ve clock. Slave sets copying the action of master at -ve clock.

* when J=0 and K=1, Master reset on +ve clock. slave reset again copying the action of master at negative (-ve) clock pulse.

2.(i) Design and implement a synchronous decade counter (or) MOD-10 counter using T flipflop. [NOV/DEC 2015]

Soln:

Step 1 : Determine the number of flip-flops needed.

$$2^n \geq N$$

Since $N=10$, $n=4$ ie. 4 flipflops are required.

Step 2 : T flip-flops to be used.

Step 3 : Determine the excitation table for counter.

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Present state				Next state				Flipflop Inputs			
D	C	B	A	D+	C+	B+	A+	T_D	T_C	T_B	T_A
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	0	0	1
0	0	1	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	0	1	0	0	0	1
0	1	0	1	0	1	1	0	0	0	1	1
0	1	1	0	0	1	1	1	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1	1
1	0	0	0	1	0	0	1	0	0	0	1
1	0	0	1	0	0	0	0	1	0	0	1
1	0	1	0	X	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X

Step 4 : K-map simplification.

For T_D

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	X	X	X	X
10	0	1	X	X

$T_D = AD + ABC$

For T_C

DC \ BA	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	X	X	X	X
10	0	0	X	X

$T_C = AB$

For T_B

DC \ BA	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	X	X	X	X
10	0	0	X	X

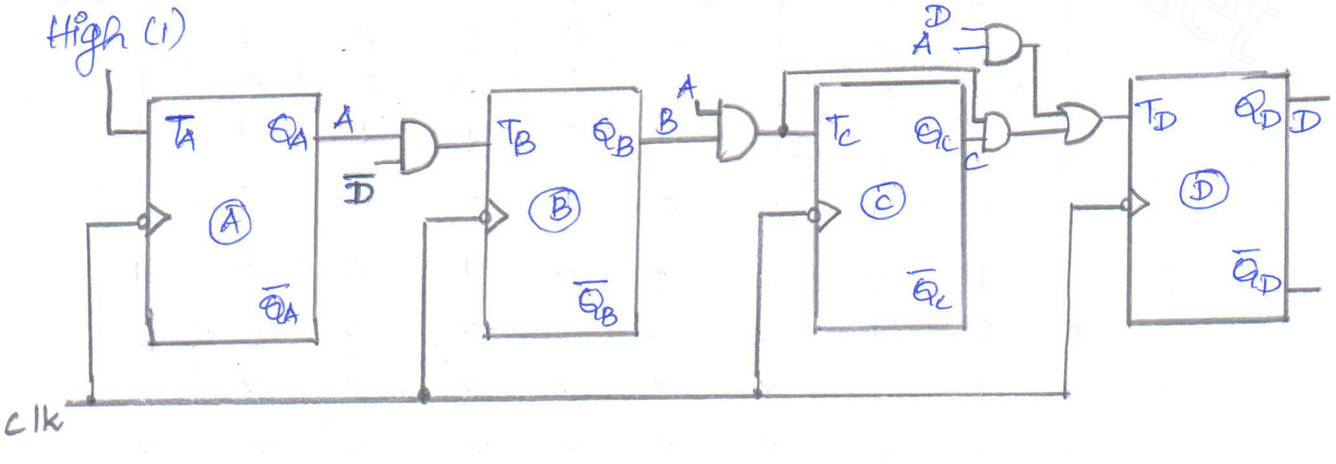
$T_B = A\bar{D}$

For T_A

DC \ BA	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$T_A = 1$

Step 5 : Logic diagram



(ii) Design and implement a asynchronous decade counter (or) BCD ripple counter using T flipflop.

Soln:

Step 1 : Determine the number of flipflops needed.

$2^n \geq N$

$2^4 \geq 10$ ∴ We need 4 flipflops.

Step 2 : T flipflops to be used.

Step 3 : Write the truthtable for the counter.

CLK	A	B	C	D	Output of reset logic Y
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	1
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	1
—	1	0	1	0	0
—	1	0	1	1	0
—	1	1	0	0	0
—	1	1	0	1	0
—	1	1	1	0	0
—	1	1	1	1	0

Step 4: Derive reset logic

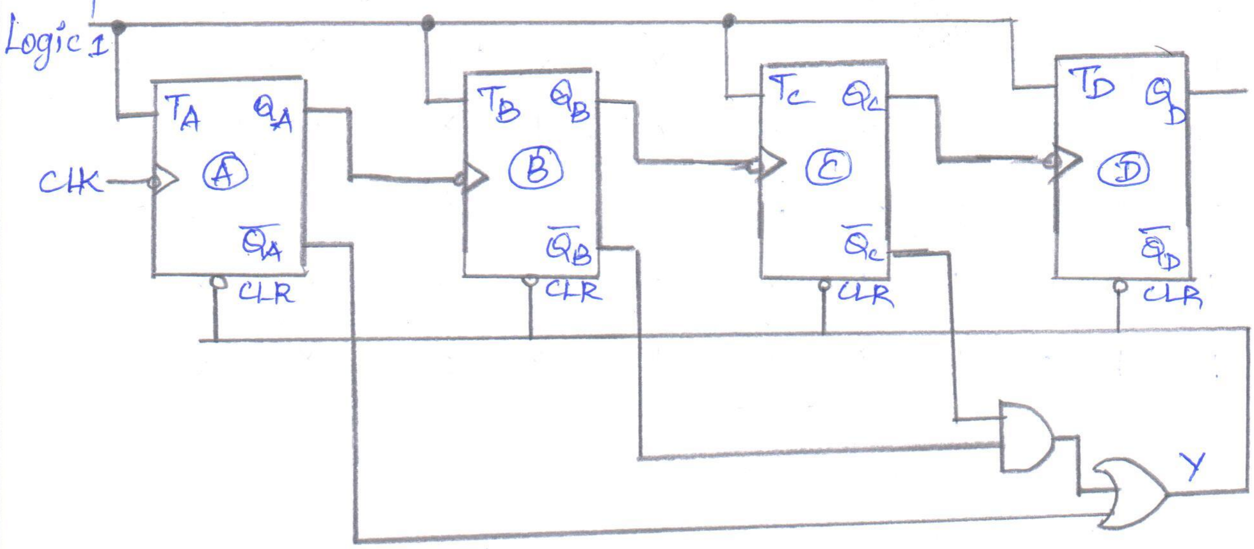
AB \ CD	00	01	11	10
00	1	1	1	1
01	1	1	1	1
11	0	0	0	0
10	1	1	0	0

→ $\bar{A}$

$\bar{B}\bar{C}$

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

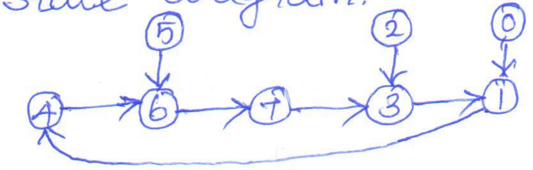
Step 5: Draw logic diagram.



3(i). Design a synchronous counter for 4, 6, 7, 3, 1, 4 . . .
Avoid lockout condition. Use JK type design.

Soln:

Step 1: State diagram.



Step 2: Determine the number of flipflops needed.
Maximum count 7, We need 3 flip flops.

Step 3: JK type flipflops to be used.

Step 4: Determine the excitation table for the counter.

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

⇐ JK flipflop excitation table.

Present states			Next states			Flipflop inputs					
A	B	C	A+	B+	C+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	1	1	X	1	X	0	X	0

Step 5: K-map simplification

For J_A

A \ BC	00	01	11	10
0	0	1	0	0
1	X	X	X	X

$J_A = \bar{B}C$

For K_A

A \ BC	00	01	11	10
0	X	X	X	X
1	0	0	1	0

$K_A = BC$

For J_B

A \ BC	00	01	11	10
0	0	0	X	X
1	1	1	X	X

$J_B = A$

For K_B

A \ BC	00	01	11	10
0	X	X	1	0
1	X	X	0	0

$K_B = \bar{A}C$

For J_c

A \ BC	00	01	11	10
0	1	X	X	1
1	0	X	X	1

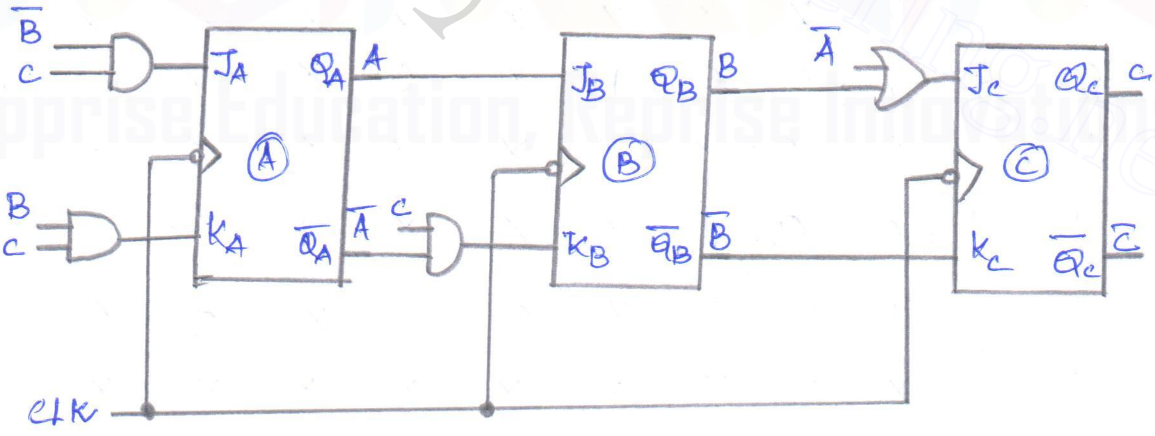
$J_c = \bar{A} + B$

For K_c

A \ BC	00	01	11	10
0	X	1	0	X
1	X	1	0	X

$K_c = \bar{B}$

Step 6: Logic diagram.



3(ii) Design a synchronous counter for 4, 6, 7, 3, 1, 4, ... using JK flipflop. [NOV/DEC 2013]

Soln:

Step 1: Determine the number of flipflops needed. Maximum count 7, we need 3 flipflops.

Step 2: JK type flipflops to be used.

Step 3: Determine the excitation table for the counter.

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

Present State			Next State			Flipflop inputs					
A	B	C	A^+	B^+	C^+	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	X	X	X	X	X	X	X	X	X
0	0	1	1	0	0	1	X	0	X	X	1
0	1	0	X	X	X	X	X	X	X	X	X
0	1	1	0	0	1	0	X	X	1	X	0
1	0	0	1	1	0	X	0	1	X	0	X
1	0	1	X	X	X	X	X	X	X	X	X
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	1	1	X	1	X	0	X	0

Step 4 : K-map simplification

For J_A

A \ BC	00	01	11	10
0	X	1	0	X
1	X	X	X	X

$J_A = \overline{B}$

For K_A

A \ BC	00	01	11	10
0	X	X	X	X
1	0	X	1	0

$K_A = C$

For J_B

A \ BC	00	01	11	10
0	X	0	X	X
1	1	X	X	X

$J_B = A$

For K_B

A \ BC	00	01	11	10
0	X	X	1	X
1	X	X	0	0

$K_B = \overline{A}$

For J_C

A \ BC	00	01	11	10
0	X	X	X	X
1	0	X	X	1

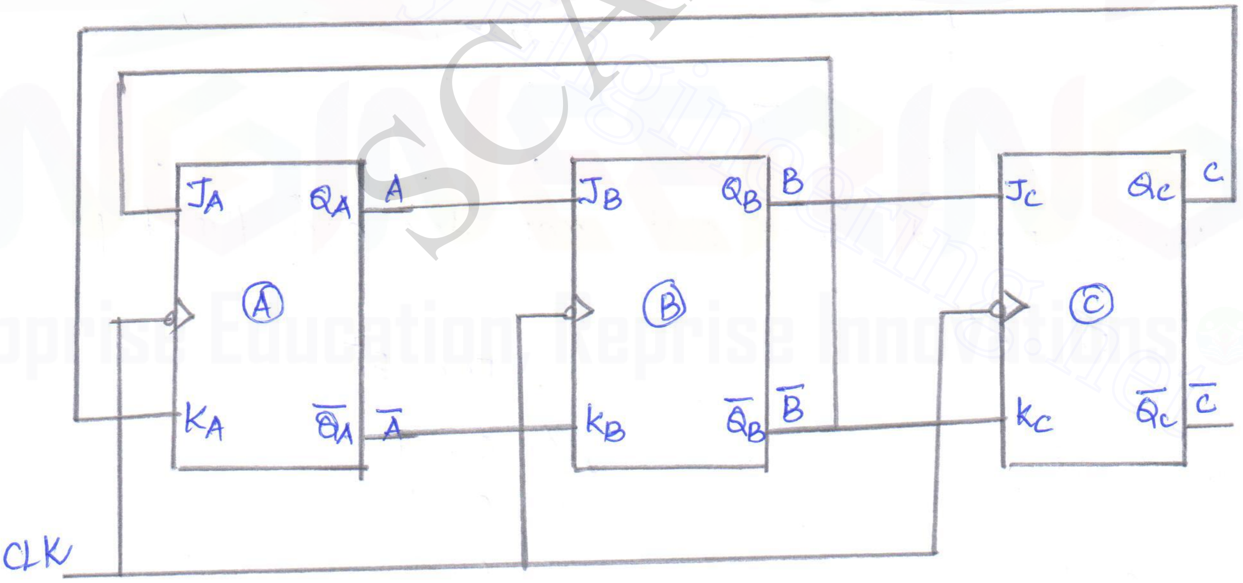
$J_C = B$

For K_C

A \ BC	00	01	11	10
0	X	1	0	X
1	X	X	0	X

$K_C = \overline{B}$

Step 5 : Logic diagram.



4(i). A sequential circuit with 2D FFs A and B and input X and output Y is specified by the following next state and output equations.

$$A(t+1) = AX + BX$$

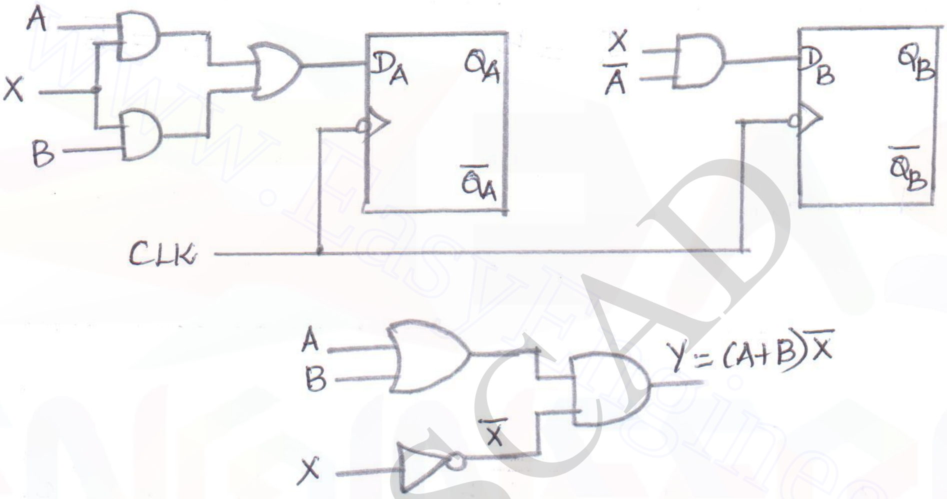
$$B(t+1) = A'X$$

$$Y = (A+B)X'$$

Draw the logic diagram, derive the state table and derive the state diagram. [NOV/DEC 2015] [MAY/JUNE 2012]

Soln:

Step 1 : Logic diagram



Step 2 : Plot the next state map for each flip-flop

For A^+

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

$A^+ = AX + BX$

For B^+

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

$B^+ = \overline{A}X$

Step 3: Plot the transition table

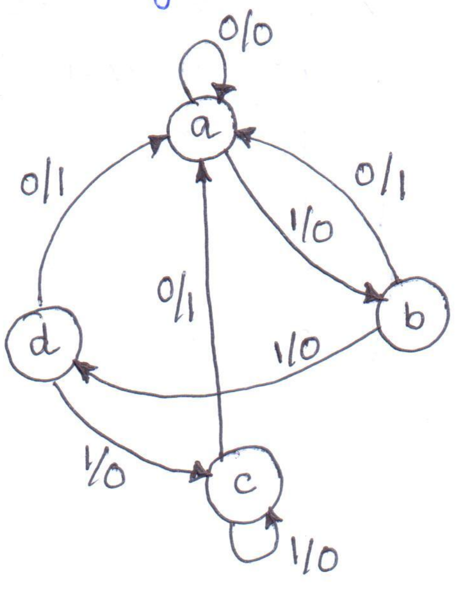
Present state		Next state				Output	
		X=0		X=1		X=0	X=1
A	B	A^+	B^+	A^+	B^+	$Y=(A+B)\overline{X}$	
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

Step 4: Draw the state table

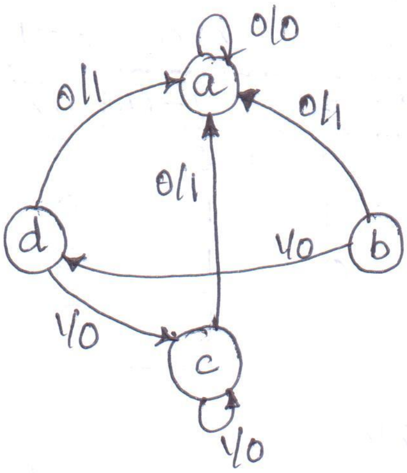
By assigning a=00, b=01, c=10 and d=11 write state table from the transition table

Present state	Next state		Output Y	
	X=0	X=1	X=0	X=1
	A^+ B^+	A^+ B^+		
a	a	b	0	0
b	a	d	1	0
c	a	c	1	0
d	a	c	1	0

Step 5: State diagram



4) (ii) Design a synchronous sequential circuit specified by a state diagram using D flipflop.



Soln:

Step 1 : Plot state table

Present state	Next state		Output Y	
	x=0	x=1	x=0	x=1
	$A^+ B^+$	$A^+ B^+$		
a	a	b	0	0
b	a	d	1	0
c	a	c	1	0
d	a	c	1	0

Step 2 : Plot the transition table

Present state		Next state				Output Y	
		x=0		x=1		x=0	x=1
A	B	A^+	B^+	A^+	B^+	$Y=(A+B)\overline{X}$	
0	0	0	0	0	1	0	0
0	1	0	0	1	1	1	0
1	0	0	0	1	0	1	0
1	1	0	0	1	0	1	0

step3 : Excitation table

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Present state		i/p	Next state		Flipflop		Output
A	B	X	A^+	B^+	D_A	D_B	Y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	1
1	1	1	1	0	1	0	0

step 4 : K-map simplification.

For D_{A^+}

A \ BX	00	01	11	10
0	0	0	1	0
1	0	1	1	0

$D_A = AX + BX$

For D_{B^+}

A \ BX	00	01	11	10
0	0	1	1	0
1	0	0	0	0

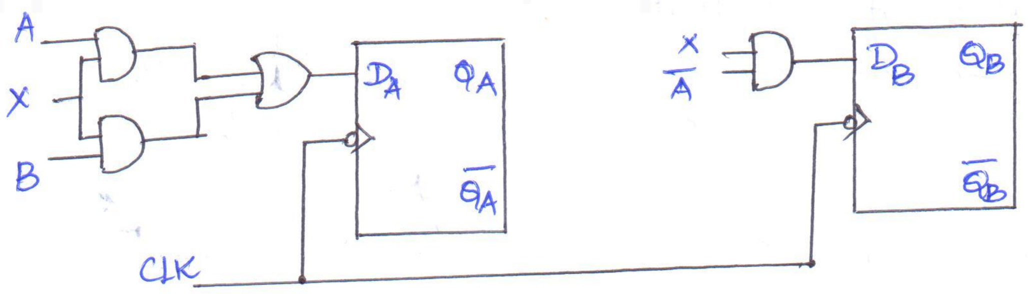
$D_B = \bar{A}X$

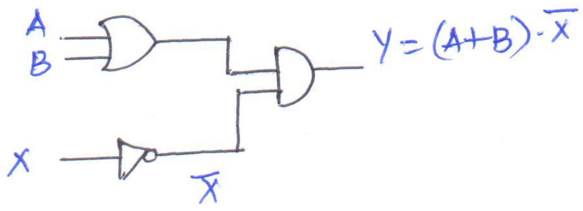
For Y

A \ BX	00	01	11	10
0	0	0	0	1
1	1	0	0	1

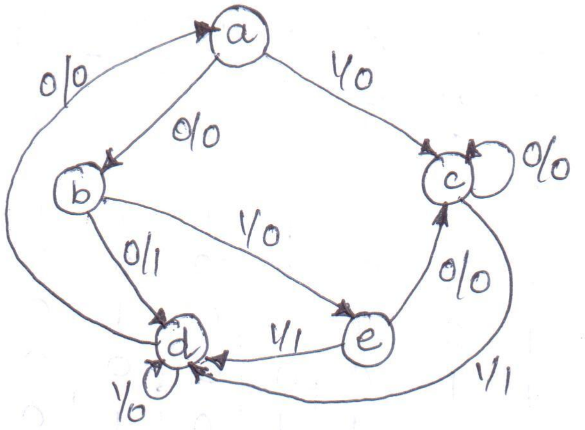
$Y = A\bar{X} + B\bar{X} = (A+B)\bar{X}$

step 5 : Logic diagram.





5. To reduce the number of state in the following state diagram. [MAY/JUNE 2013]



Soln:

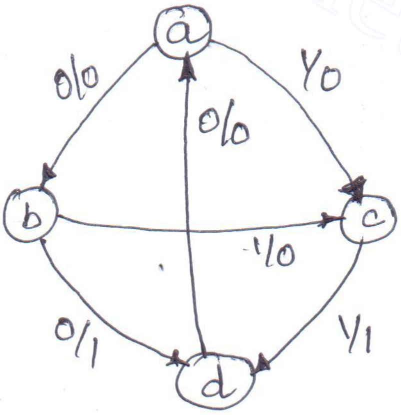
step 1 : state table

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	b	c	0	0
b	d	e	1	0
c	c	d	0	1
d	a	d	0	0
× e	c	d	0	1

Step 2: Reduced state table

Present state	Next state		Output	
	X=0	X=1	X=0	X=1
a	b	c	0	0
b	d	c	1	0
c	c	d	0	1
d	a	d	0	0

step 3: Reduced State diagram



6. Explain in detail about shift register and it types.

SHIFT REGISTERS [APRIL/MAY 2015]

* A register capable of shifting the binary information held in each cell to its neighboring cell, in a selected direction is called shift register.

* The logical configuration of a shift register consist of a chain of flip-flops in cascade, with the output of one flip flop connected to the input of the next flipflop.

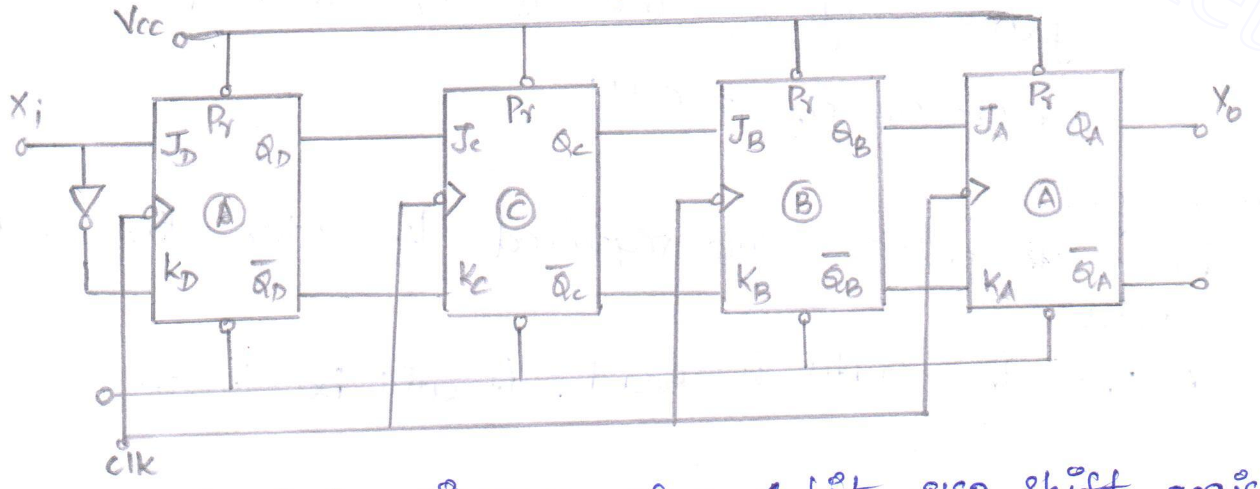
* All flip-flops receive common clock pulses, which activate the shift of data from one stage to next.

* The shift registers are classified as.

1. Serial In Serial Out (SISO)
2. Serial In Parallel Out (SIPO)
3. Parallel In Serial Out (PISO)
4. Parallel In Parallel Out (PIPO)

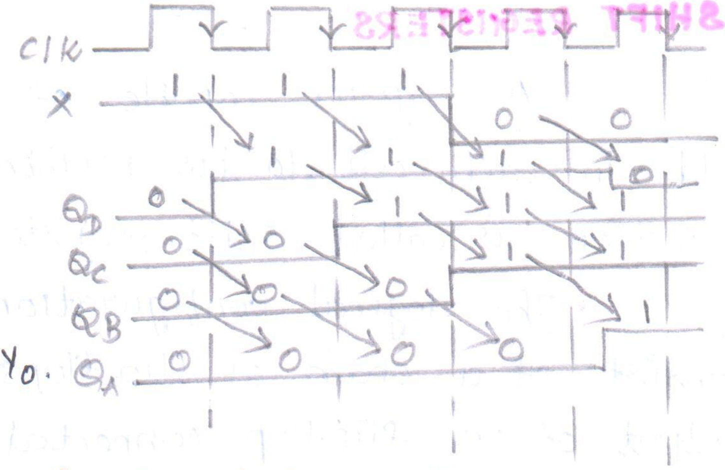
SISO shift Register.

* In a SISO shift register data is entered and retrieved in serial fashion with clock.



* The logic diagram of a 4-bit SISO shift register using JK flip-flop. Xi is input and Yo is output of the shift register.

Clk	X	Q_D	Q_C	Q_B	Q_A
0	1	0	0	0	0
1	1	1	0	0	0
2	1	1	1	0	0
3	0	1	1	1	0
4		0	1	1	1 → Y_0.



Operation:

* The inputs of flip-flops D, C, B and A are X_i, Q_D, Q_C, Q_B and Q_A. The flipflop input 0111.

Initially the shift register is cleared.

* $Q_D\ Q_C\ Q_B\ Q_A$ = 0000 and input $X = 1$

The negative edge of the first clock pulse, the input data is entered into the flip-flop and the end of the clock pulse

* $Q_D\ Q_C\ Q_B\ Q_A$ = 1000 and $X = 1$.

The negative edge of the second clock pulse, the input entered and at the end of second clock pulse.

* $Q_D\ Q_C\ Q_B\ Q_A$ = 1100 and input $X = 1$

* The negative edge of the third clock pulse

$Q_D\ Q_C\ Q_B\ Q_A$ = 1110 input $X = 0$.

* The negative edge of the fourth clock pulse.

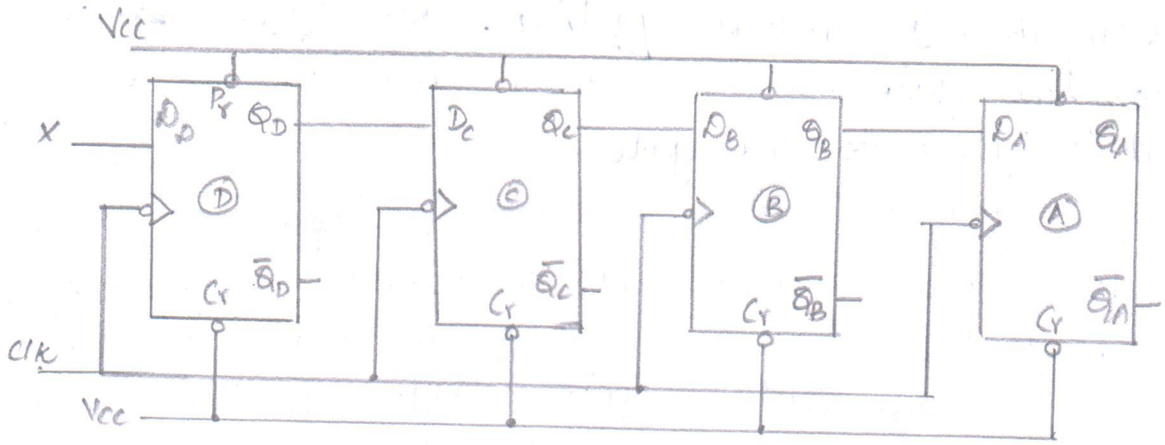
$Q_D\ Q_C\ Q_B\ Q_A$ = 0111

Disadvantages:

* n clock pulses are required to enter the n-bit data.

* Once the data is read, it will be lost.

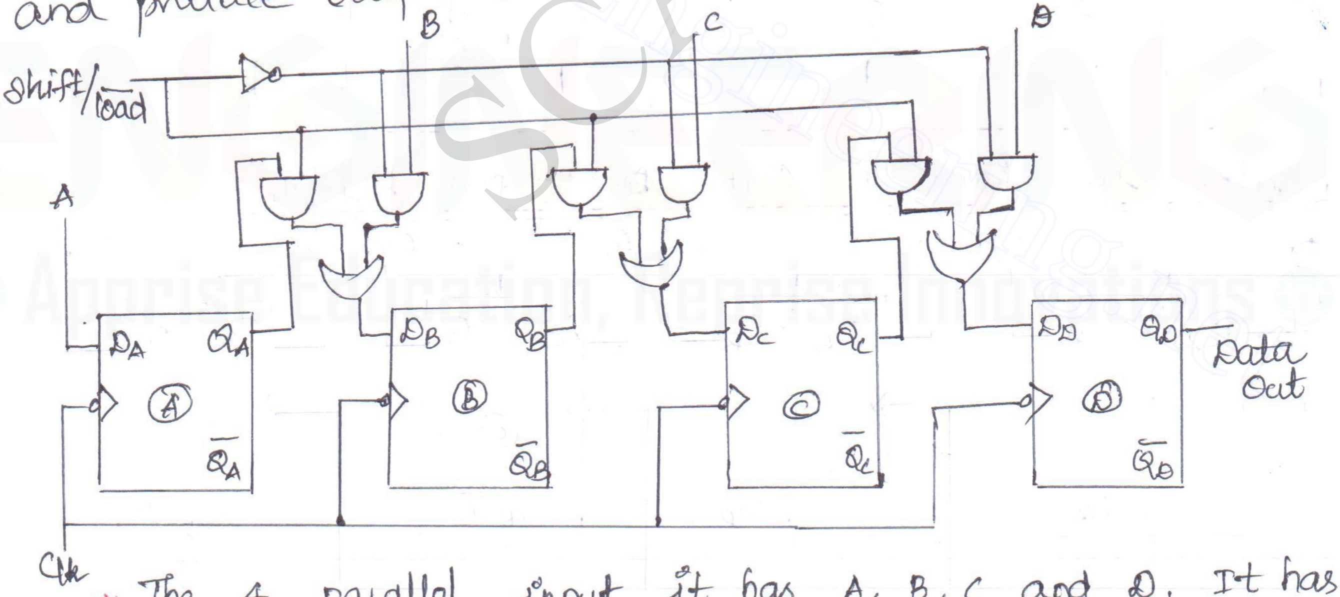
Serial In Parallel Out (SIPO)



* In a SIPO shift register, data is entered into the register in serial fashion as in SISO shift register and read from the shift register in parallel fashion.

Parallel in Serial Out register. (PISO)

* Data are entered simultaneously into the flipflop and produce output in serial fashion.



* The 4 parallel input it has A, B, C and D. It has a control input called shift load time.

* When shift/$\overline{load}$ is high the AND gate G_1, G_2 and G_3 are enabled.

* When shift/$\overline{Load}$ is high the work as a serial in serial out.

Operation.

* Assume the input 1101.

* First input is '1' clock is applied. Now $Q_A=1$ This is applied to AND gate G4, The other input of G4 is 1 G4 output is given to OR gate.

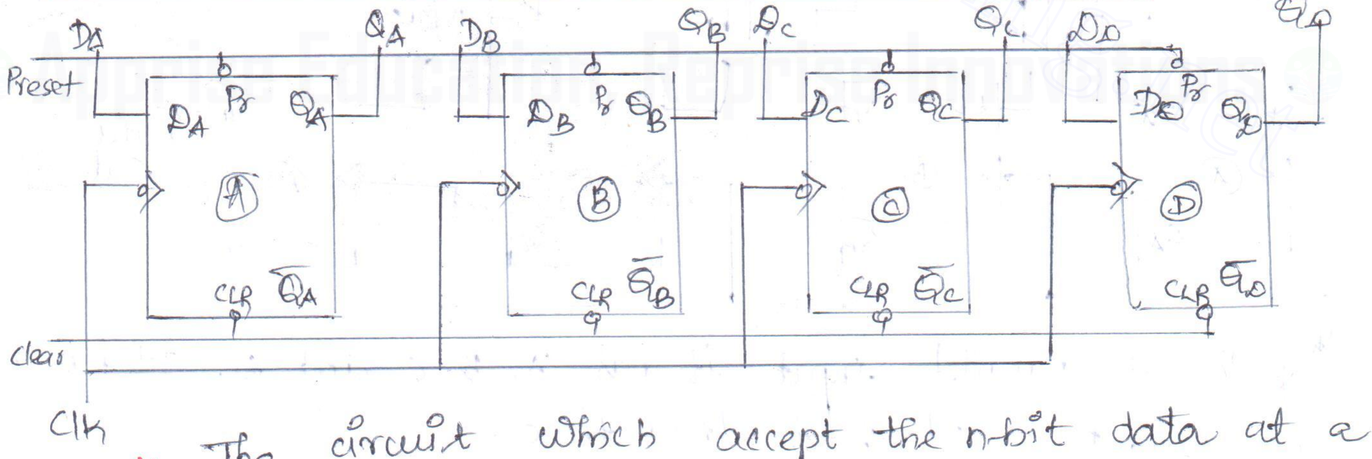
* OR gate output is given to next flipflop. In this same fashion remaining circuit constructed. now $D_A=1$

* Now next input 0 is given to in serial fashion. After each clock one bit is given after all the input is given the output is obtained at Q_A Q_B Q_C Q_D

Let as assume example =1011.

clk	shift	serial input	Q_D	Q_C	Q_B	Q_A
0	0	0	0	0	0	0
1	1	1	1	0	0	0
2	2	1	1	1	0	0
3	3	0	0	1	1	0
4	4	1	1	0	1	1

Parallel in Parallel Out (PI PO) Register.



* The circuit which accept the n-bit data at a time and produce the same (or) different data at time based on the type of flip flop

* Once the register gets the data. when clock pulse is applied. The same data for D flip flop (or) different for data other flipflop will produce in the output.

7(i). Realize T flipflop using JK flipflop [NOV / DEC 2015]

step1: The excitation table for JK flipflop.

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

step 2: JK flip-flop to T flipflop conversion

Input T	Present state Q_n	Next state Q_{n+1}	Flip flop i/p	
			J	K
0	0	0	0	X
0	1	1	X	0
1	0	1	1	X
1	1	0	X	1

step 3: K-map Simplification

For J_A

T \ Q_n	0	1
0	0	X
1	1	X

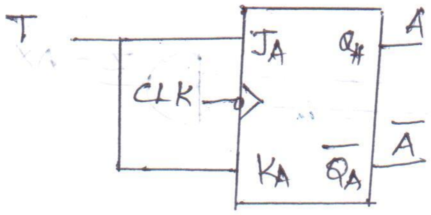
$J_A = T$

For K_A

T \ Q_n	0	1
0	X	0
1	X	1

$K_A = T$

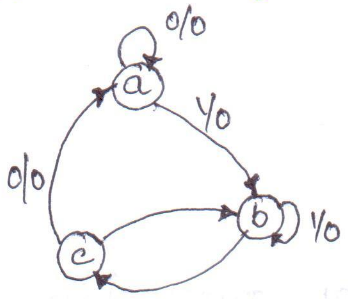
step 4: Logic diagram.



7(ii) Design a sequence detector to detect the sequence 101 using D flip-flop. [APRIL/MAY 2015] [NOV/DEC 2015]

The specified input sequence can be detected using a sequential machine called sequence detector.

Step 1: State diagram.



Step 2: State table

Present State	Next State		Output	
	x=0	x=1	x=0	x=1
a	a	b	0	0
b	c	b	0	0
c	a	b	0	1

Step 3: State assignment

Assign state

a=00, b=01, c=10

Present State		Next state		Output	
		$A^+ B^+$		Z	
A	B	X=0	X=1	X=0	X=1
0	0	00	01	0	0
0	1	10	01	0	0
1	0	00	01	0	1

Step 4: Simplify using K-map

For A^*

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

$A^+ = B\bar{X}$

For B^+

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

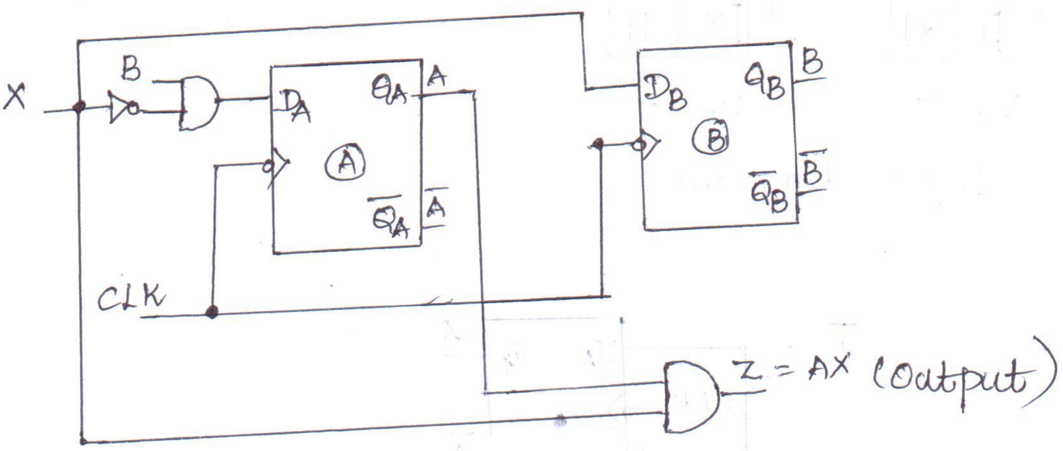
$B^+ = X$

For Z

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

$Z = AX$

Step 5: Logic diagram



UNIT-IV ASYNCHRONOUS SEQUENTIAL CIRCUITS & PROGRAMMABLE LOGIC DEVICES

TWO MARKS

1. Draw the block diagram of asynchronous sequential circuit. (May 2011)

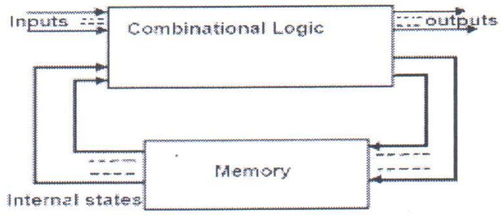


Figure 1: Asynchronous Sequential Circuit

2. When does race condition occur? (May 2013, Dec 2012)

Two or more binary state variables change their value in response to the change in i/p variable

3. What is hazard? And cause for essential hazards, types of hazards. (Nov 2011)

Hazard-unwanted switching transients

Essential hazard- unequal delays along 2 or more path from same input

Types- 1.Static hazard 2.Dynamic hazard

4. Define primitive flow table. (May 2013, May 2012, May 2011)

It is defined as a flow table which has exactly one stable state for each row in the table. The design process begins with the construction of primitive flow table.

5. Define PROM. (Nov 2015, May 2015)

PROM is Programmable Read Only Memory. It consists of a set of fixed AND gates Connected to a decoder and a programmable OR array.

6. What is the drawback of asynchronous sequential machines. (May 2014)

Asynchronous circuit responds to all the transient values and problems like oscillations, critical race and hazards. So asynchronous circuits are difficult to design.

7. What is the difference between flow table and transition table (May 2013)

The difference between flow table and transition table is that the internal states in flow table are symbolized with letters whereas internal states in transition table are represented by binary numbers.

8. How does the operation of an asynchronous input differ from that of a synchronous input? (May 2012)

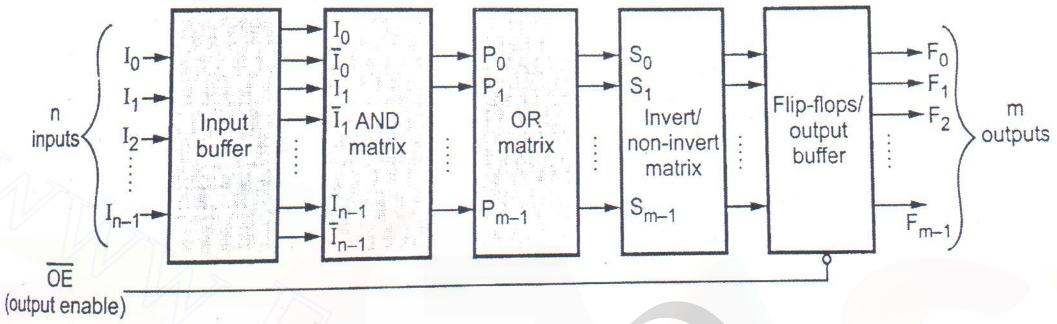
In synchronous sequential circuits, memory elements are clocked flip-flops. Hence input signals can affect memory element at any instant of time.

In asynchronous sequential circuits, memory elements are either un-clocked flip flops or time delay elements. Therefore in asynchronous sequential circuits change in input signals can affect memory element at any instant of time.

9. Write the advantages of PLA. (Nov 2012)

- It's costliest and complex than PAL and PROMs.
- Both AND and OR arrays are programmable.
- AND array can be programmed to get desired minterms.

10. Draw the block diagram of PLA. (Nov 2014)



11. Compare Pulse mode and fundamental mode. (Nov 2015)

ndamental mode	lse mode
ly one input variable can change at a given time	ses should not occur simultaneously on two or more input lines.
uts are levels and not pulses	e input variables are pulses instead of levels

12. What is turing machine. (May 2014)

A turing machine is a general example of CPU that controls all data manipulation done by a computer, with the canonical machine using sequential memory to store data.

A finite state machine is the mathematical model of computation used to design both computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states.

13.What are the significance of state assignment?(Nov 2015)

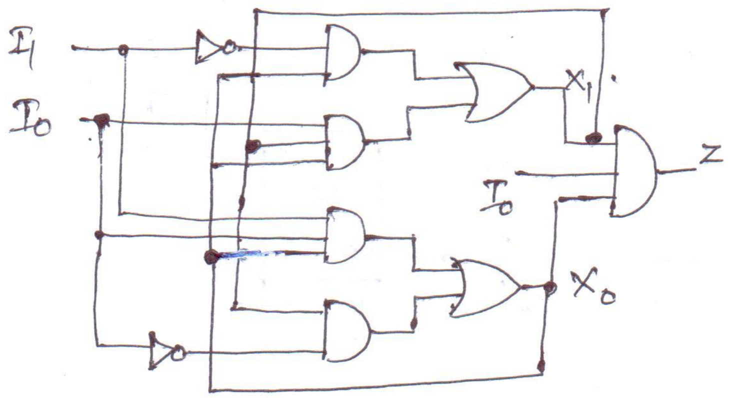
In synchronous circuits state assignments are made with the objective of circuit reduction. In asynchronous circuits objective is to avoid critical races.

Techniques:

1. Shared row state assignment
2. One hot state assignment.

1. Fundmental model problem.

1. Analyze the fundamental mode asynchronous sequential circuit



Soln:

Step 1 : Determine next secondary state and output equations.

$X_1^+ = X_0\bar{I_1} + X_0 I_0 X_1$; $X_0^+ = X_0 I_0 I_1 + X_1 \bar{I_0}$; $Z = X_0 X_1 I_0$

Step 2: : construct state table

Present total state				Next total state				Stable Total state Yes / No	Z
Secondary state		Inputs		Next secondary state		Inputs.			
X_1	X_0	I_1	I_0	X_1	X_0	I_1	I_0		
0	0	0	0	0	0	0	0	Yes	0
0	0	0	1	0	0	0	1	Yes	0
0	0	1	0	0	0	1	0	Yes	0
0	0	1	1	0	0	1	1	Yes	0
0	1	0	0	1	0	0	0	No	0
0	1	0	1	1	0	0	1	No	0
0	1	1	0	0	0	1	0	No	0
0	1	1	1	0	1	1	1	Yes	0
1	0	0	0	0	1	0	0	No	0
1	0	0	1	0	0	0	1	No	0
1	0	1	0	0	1	1	0	No	0
1	0	1	1	0	0	1	1	No	0
1	1	0	0	1	1	0	0	yes	0
1	1	0	1	1	0	0	1	No	1
1	1	1	0	0	1	1	0	No	0
1	1	1	1	1	1	1	1	yes	1

Step 3:

I_1I_0 / X_1X_0	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	1	0
10	0	0	0	0

X_0

I_1I_0 / X_1X_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	0	1	1
10	1	0	0	1

X_1

I_1I_0 / X_1X_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	0
10	0	0	0	0

Z

* circle represent stable state

Step :- 4

I_1I_0 / X_1X_0	00	01	11	10
00	00	00	00	00
01	10	10	01	00
11	11	10	11	01
10	01	00	00	01

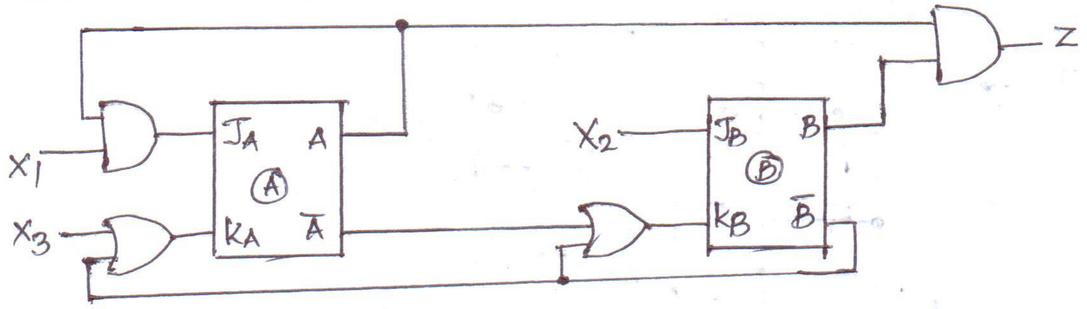
Unstable state

	00	01	11	10
00	0	0	0	0
01	—	—	0	—
11	0	—	1	—
10	—	—	—	—

Z

Pulse mode problem. [NOV/DEC - 2013]

③ Analyze the pulse mode circuit shown in fig. Determine flow table and state diagram.



slon:

step 1: $J_A = x_1 A$; $K_A = x_3 + \bar{B}$; $J_B = x_2$; $k_B = \bar{A}+\bar{B}$; $Z = AB$

Step 2: JK flipflop characteristic equation.

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

$$Q_{A+1} = J_A\bar{Q_A} + \bar{K_A} Q_A = x_1 A \bar{A} + (x_3+\bar{B})A \Rightarrow (x_3+\bar{B})A$$

$$Q_{B+1} = J_B\bar{Q_B} + \bar{k_B} Q_B = x_2\bar{B} + (\bar{A}+\bar{B})B \Rightarrow x_2\bar{B} + \bar{A}B$$

Step 3: construct the state variable transition table.

Pulse input variables

AB	x_1	x_2	x_3
00	$\frac{00}{0}$	$\frac{01}{0}$	$\frac{00}{0}$
01	$\frac{01}{0}$	$\frac{01}{0}$	$\frac{01}{0}$
11	$\frac{00}{1}$	$\frac{00}{1}$	$\frac{10}{1}$
10	$\frac{10}{0}$	$\frac{11}{0}$	$\frac{10}{0}$

$(x_3+\bar{B})A = Q_{A+1}$

$x_2\bar{B} + \bar{A}B = Q_{B+1}$

$Z = AB.$

step 4: Derive the flow table and state diagram.

00 → S_0

01 → S_1

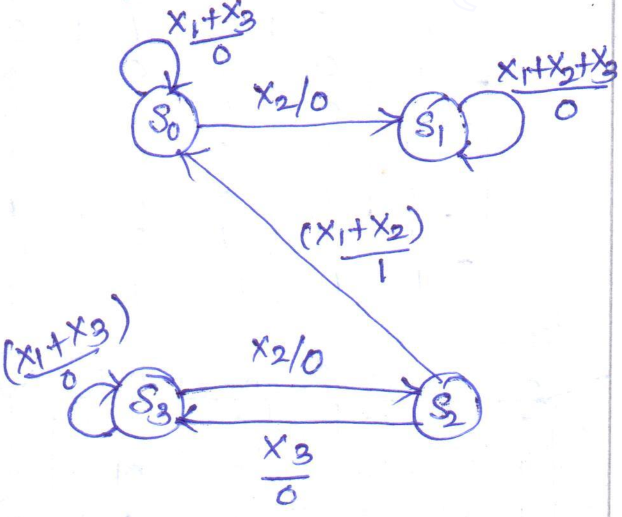
11 → S_2

10 → S_3

Pulse input

AB	x_1	x_2	x_3
S_0	$\frac{S_0}{0}$	$\frac{S_1}{0}$	$\frac{S_0}{0}$
S_1	$\frac{S_1}{0}$	$\frac{S_1}{0}$	$\frac{S_1}{0}$
S_2	$\frac{S_0}{1}$	$\frac{S_0}{1}$	$\frac{S_3}{1}$
S_3	$\frac{S_3}{0}$	$\frac{S_2}{0}$	$\frac{S_3}{0}$

State table.

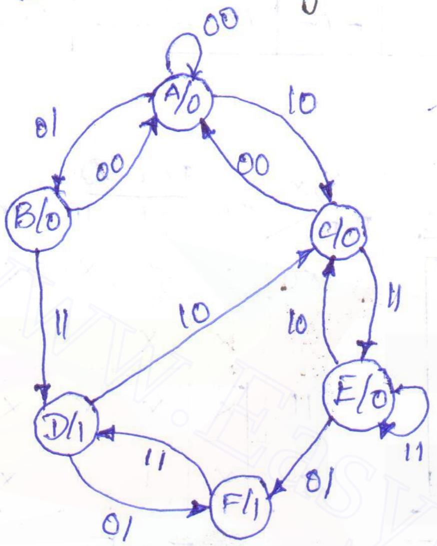


state diagram

2. Design an asynchronous sequential circuit that has two input x_1 and x_2 and one output z. When $x_1=0$, then output z is 0. The first change in x_2 that occurs while x_1 is 1 will cause output z to be '1'. The output z will remain 1 until x_1 return to 0.

[NOV/DEC-2015] [APRIL/MAY 2015]

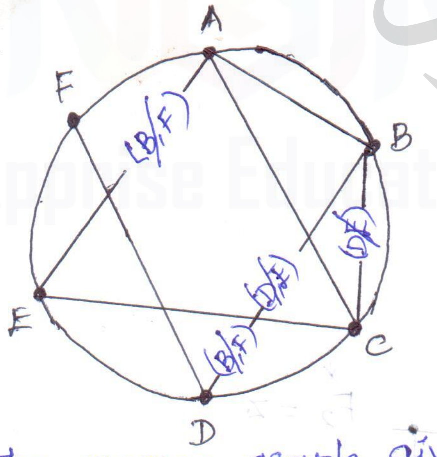
Step 1: State diagram.



Step 2: Primitive flow table

Present State	Next state/output z for x_2 x_1 inputs			
	00	01	11	10
A	Ⓐ,0	B,–	–,–	C,–
B	A,–	Ⓑ,0	D,–	–,–
C	A,–	–,–	E,–	Ⓒ,0
D	–,–	F,–	Ⓓ,1	C,–
E	–,–	F,–	Ⓔ,0	C,–
F	A,–	Ⓔ,1	D,–	–,–

Step 3: State reduction using Merger graph



The merger graph gives two compatible pairs as a set of maximal compatibles

$(A,B) \rightarrow S_0$

$(C,E) \rightarrow S_1$

$(D,F) \rightarrow S_2$

Step 4: Reduced flow table

Present State	next state/output z for x_2, x_1 inputs			
	00	01	11	10
S_0	$\textcircled{S_0}$,0	$\textcircled{S_0}$,0	S_2,–	S_1,–
S_1	S_0,–	S_2,–	$\textcircled{S_1}$,0	$\textcircled{S_1}$,0
S_2	S_0,–	$\textcircled{S_2}$,1	$\textcircled{S_2}$,1	S_1,–

Step 5: State Assignment

Now if we assign $S_0 \rightarrow 00$, then $S_1 \rightarrow 01$ and $S_2 \rightarrow 10$ then. one more state $S_3 \rightarrow 11$ to prevent critical race during transition of $S_1 \rightarrow S_2$ (or) $S_2 \rightarrow S_1$. By introducing S_3 the transition $S_1 \rightarrow S_2$ (or) $S_2 \rightarrow S_1$ are routed through S_3.

step 5 continuation.

Present State F_2 F_1	Next state/output z for x_2 x_1 inputs			
	00	01	11	10
$S_0 \rightarrow 00$	S_0, 0	S_0, 0	S_2, –	S_1, –
$S_1 \rightarrow 01$	S_0, –	S_3, –	S_1, 0	S_1, 0
$S_2 \rightarrow 10$	S_0, –	S_2, 1	S_2, 1	S_3, –
$S_3 \rightarrow 11$	–, –	S_2, –	–, –	S_1, –

step 6: Transition table

Present State F_2 F_1	Next State (F_2* F_1*)/ output z for x_1 x_2 i/ps			
	00	0·1	11	10
0 0	00, 0	00, 0	10, –	01, –
0 1	00, –	11, –	01, 0	01, 0
1 0	00, –	10, 1	10, 1	11, –
1 1	–, –	10, –	–, –	01, –

Step 7: simplify using k-map

For F_2*

F_2F_1 \ x_2x_1	00	01	11	10
00	0	0	1	0
01	0	1	0	0
11	x	1	X	0
10	0	1	1	1

$F_2^* = F_2 x_1 + F_1 \overline{x_2} x_1 + F_2 \overline{F_1} x_2 + \overline{F_1} x_2 x_1$

For F_1*

F_2F_1 \ x_2x_1	00	01	11	10
00	0	0	0	1
01	0	1	1	1
11	X	0	X	1
10	0	0	0	1

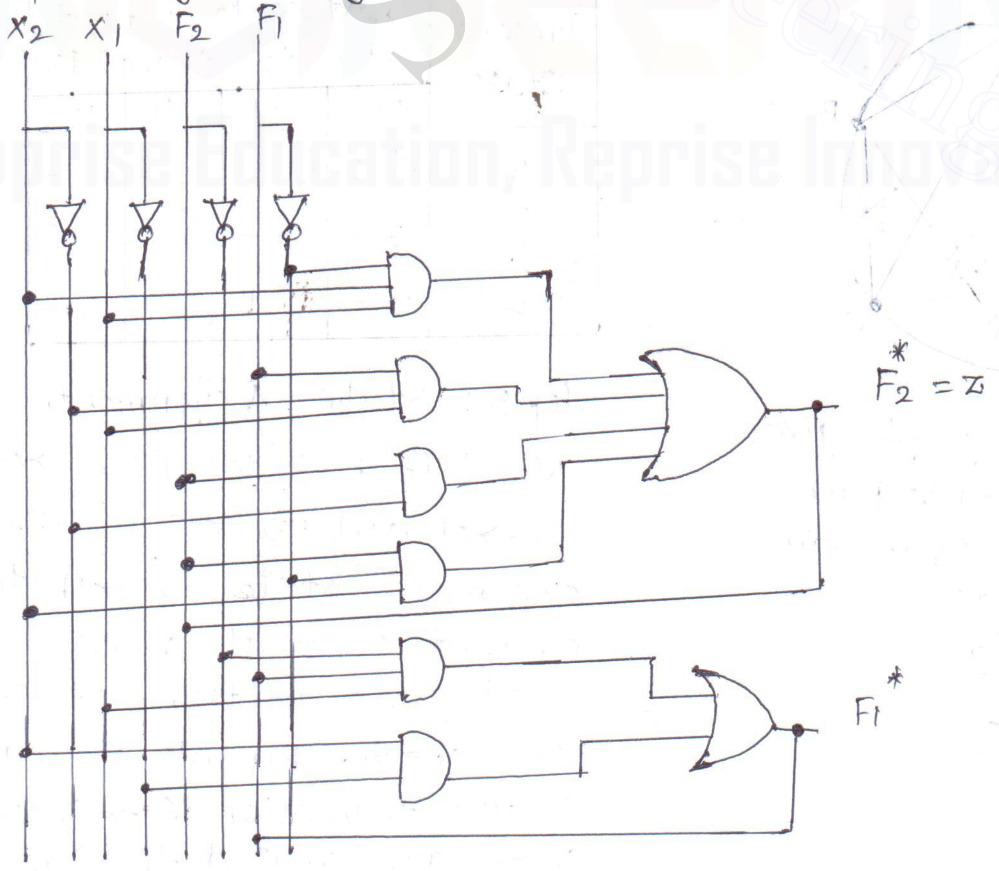
$F_1^* = \overline{F_2} F_1 x_1 + x_2 \overline{x_1}$

For z

F_2F_1 \ x_2x_1	00	01	11	10
00	0	0	X	X
01	x	X	0	0
11	X	X	X	X
10	X	1	1	X

$z = F_2$

Step 7: Logic diagram



ASYNCHRONOUS DESIGN PROBLEM.

3. Problem : 1

Develop the state diagram and primitive flow table for a logic system that has two inputs, x and Y, and a single output z, Initially both inputs and outputs are equal to 0. whenever x=1 and Y=0, the z becomes 1 and whenever x=0 and Y=1, z becomes 0. when inputs are zero, x=Y=0 (or) inputs are one x=Y=1, the output z does not change; it remains in the previous state. [NOV/DEC 2015]

Soln:

step: 1 ~~x~~ ~~x~~

xy = 00 ; z = 0 (Initial state)
xy = 01 ; z = 0
xy = 10 ; z = 1
xy = 11 ; z = 1 (Previous state)

STATE A : xy = 00 ; z = 0 → (A)
xy = 01 ; z = 0 → (B)
xy = 10 ; z = 1 → (C)

STATE B xy = 01 ; z = 0 → (B)
xy = 11 ; z = 0 → (D)
xy = 00 ; z = 0 → (A)

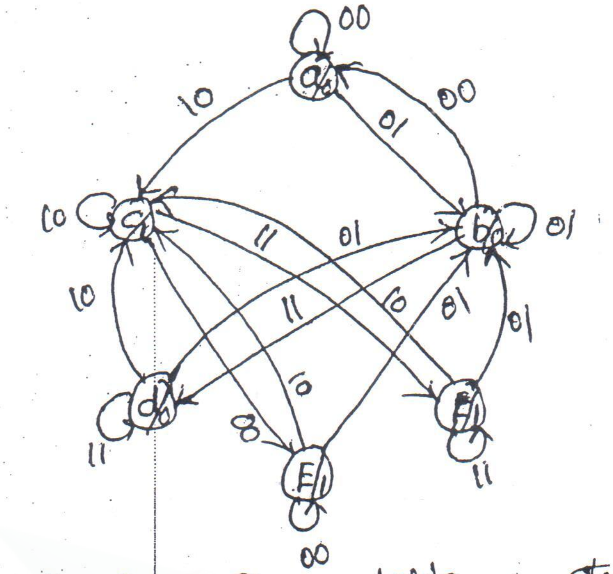
STATE C xy = 10 ; z = 1 → (C)
xy = 11 ; z = 1 → (F)
xy = 00 ; z = 1 → (E)

STATE D xy = 11 ; z = 0 → (D)
x,y = 01 ; z = 0 → (B)
xy = 00 ; z = 1 → (A)

STATE E xy = 00 ; z = 1 → (E)
xy = 01 ; z = 0 → (B)
xy = 10 ; z = 1 → (C)

STATE F xy = 11 ; z = 1 → (F)
xy = 01 ; z = 0 → (B)
xy = 10 ; z = 1 → (C)

step 2: Draw state diagram.



step 3: Primitive flow table

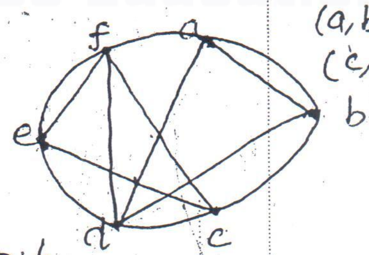
	00	01	11	10
a	ⓐ,0	b,-	-,-	c,1
b	a,-	ⓑ,0	d,-	-,-
c	e,-	-,-	F,-	ⓒ,1
d	-,-	b,-	ⓓ,0	c,-1
e	ⓔ,1	b,-	-,-	c,-
f	-,-	b,-	ⓕ,1	c,-

step: 4.

b	✓				
c	a-e	a-e d-F			
d	✓	✓	F-d		
e	a-e	a-e	✓	d-f	
f	✓	d-f	✓	✓	✓
	a	b	c	d	e

compatable pairs
(a,b), (a,d), (a,f), (b,d), (c,e), (c,f)
(d,f), (e,f)

step 5: MERGER DIAGRAM.



(a,b,d) —a
(c,e,f) —b

W

W \ xy	00	01	11	10
0	0	0	0	1
1	1	1	0	1

Z

W \ xy	00	01	11	10
0	0	0	0	X
1	1	X	1	1

a=0, b=1

$W = w\bar{x} + w\bar{y}$

$Z = W$

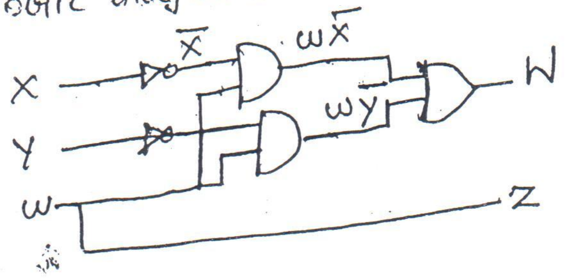
step: 6

	00	01	11	10
a,b,d	a,0	b,0	d,0	c,-
c,e,f	e,1	b,-	f,1	c,1

⇓

	00	01	11	10
a	ⓐ,0	ⓐ,0	ⓐ,0	b,-
b	ⓑ,1	ⓐ,-	ⓑ,1	ⓑ,1

logic diagram.



4. Explain in detail about races and how to minimize races?

Race free state assignment [Nov/DEC-2014]

* The state assignment step in asynchronous circuits is essentially the same as it is for synchronous circuits, exept for one difference.

* In synchronous circuits, the state assignments are made with the objective of circuit reduction.

* In asynchronous circuits the objective of state assignment is to avoid critical races.

Races and cycles.

* When two or more binary state variables change their value in response to a change in an input variable, race condition occurs in an asynchronous sequential circuit.

* In case of unequal delays, a race condition may cause the state variables to change in an unpredictable manner.

* For example if there is a change in two state variables such that both change due to change in input variable from 00 to 11.

* In this situation the difference in delays may cause the first variable to change faster than the second resulting the state variable to change in sequence from 00 to 10 then to 11.

* On the other hand, if the second variable changes faster than the first, the state variable change from 00 to 01and then to 11.

* If the final ~~statble~~ state that the circuit reaches does not depend on the order in which the state variable changes, the race condition is not harmful and it is called a noncritical race.

* If the final stable state depends on the order in which the state variable changes, the race condition is harmful and it is called a critical race. such critical races must be avoided for proper operation.

Non Critical race

* The transition tables in which X' is a input variable and $y_1 y_2$ are the state variables.

* consider a circuit is in a stable state $y_1 y_2 x = 000$ and there is a change in input from 0 to 1.

* They can either change simultaneously from 00 to 11, or they may change in sequence from 00 to 01 and then to 11, or they may change in sequence from 00 to 10 and then to 11.

EX:

y_1y_2 \ x	0	1
00	00	11
01		11
11		11
10		11

Possible transitions

00 → 11

00 → 01 → 11

00 → 10 → 11

y_1y_2 \ x	0	1
00	00	11
01		11
11		10
10		10

Possible transition

00 → 11 → 10

00 → 10

00 → 01 → 11 → 10

Critical Races.

*consider a circuit is in a stable state $y_1 y_2 x = 000$ and there is a change in input from 0 to 1.

*If state variable change simultaneously, the final stable state is $y_1 y_2 x = 111$.

*If y_2 changes to 1 before y_1 because of unequal propagation delay, then the circuit goes to the stable state 011.

*On the other hand Y_1 changes faster than Y_2, then the circuit goes to stable state 101.

*The race is critical because the circuit goes to different stable states depending on the order in which the state variable change.

y_1y_2 \ x	0	1
00	00	11
01		01
11		11
10		10

Possible transitions

00 → 01

00 → 10

00 → 11

Cycles.

* A cycle occurs when an asynchronous circuit makes a transition through a series of unstable states.

When a state assignment is made so that it introduces cycles, care must be taken to ensure that each cycles terminates on a stable state.

* A cycle does not contain a stable state, the circuit will go from one unstable state to another, until the inputs are changed. Such a situation must always be avoided when designing asynchronous circuits.

* Two techniques are commonly used for making a critical race free state assignment-

1. Shared row state assignment
2. One hot state assignment.

Shared row state assignment.

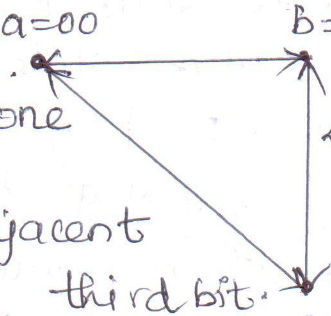
* Races can be avoided by making a proper binary assignment to the state variables.

* The state variables are assigned with binary numbers in such a way that only one state variables can change at any one time when a state transition occurs.

* It is necessary that states between which transition occur be given adjacent assignments.

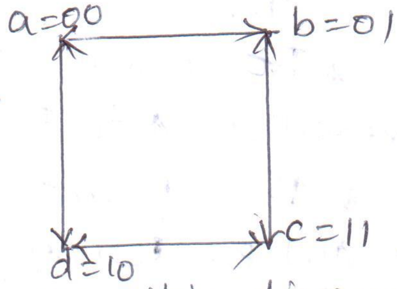
* Two binary values are said to be adjacent if they differ in only one variable.

* For example 110 and 111 are adjacent because they differ only in the third bit.



← Transiti diagra

* This assignment will causes a critical race during the transition from a to c. because there is two changes of binary state variables.



Transition diagram wit race free assignment

* A race free assignment can be obtained by introducing addition binary state say 'd' with binary value '10' with the adjacent to both a and c.

00 → 10 → 11.

*which satisfy the condition that only binary variable changes during each transition. thus avoiding critical race this technique is called as shared raw state assignment

One hot state assignment.

* The one hot state assignment is an another method for finding a race free assignment.

* This method only variable is active or hot for each row in the original flow table.

* Additional rows are introduced to provide single variable changes between internal state transitions.

Ex:

State Variables				State	input x_1x_2			
F_4	F_3	F_2	F_1		00	01	11	10
0	0	0	1	A	A	B	C	C
0	0	1	0	B	A	B	C	D
0	1	0	0	C	A	B	C	C
1	0	0	0	D	D	B	C	D

flow table.

original table

add rows

state variables				State	input x_1, x_2			
F_4	F_3	F_2	F_1		00	01	11	10
0	0	0	1	A	A	~~B~~ E	~~C~~ F	~~C~~ F
0	0	1	0	B	~~A~~ E	B	C	D
0	1	0	0	C	~~A~~ F	~~B~~ G	C	C
1	0	0	0	D	D	~~B~~ H	~~C~~ I	D
0	0	1	1	E	A	B	–	–
0	1	0	1	F	A	–	C	C
0	1	1	0	G	–	B	C	C
1	0	1	0	H	–	B	–	D
1	1	0	0	I	–	–	C	–

one hot state flow table.

* A transition from state A to state B requires two state variable changes, F_1 from 1 to 0 and F_2 from 0 to 1.

* By directing the transition A to B through a new row E which contains 1s where both states A and B have 1s.

* ~~Ite~~ require only one state variable change from transition A to E and then from transition E to B.

* This permit the race free ~~[illegible]~~ transition between A and B. Remaining state variable similar the same.

5. Implement the following function using PLA [APRIL/MAY 2015]

$A(x,y,z) = \Sigma(1,2,4,6)$

$B(x,y,z) = \Sigma(0,1,6,7)$

$C(x,y,z) = \Sigma(2,6)$

Soln: To determine the truth table

step1:

X	Y	Z	A	B	C
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	0

step 2: k map simplification.

x \ yz	00	01	11	10
0	0	1	0	1
1	1	0	0	1

$A = \bar{x}\bar{y}z + x\bar{z} + y\bar{z}$

x \ yz	00	01	11	10
	1	1	0	0
	0	0	1	1

$B = \bar{x}\bar{y} + xy$

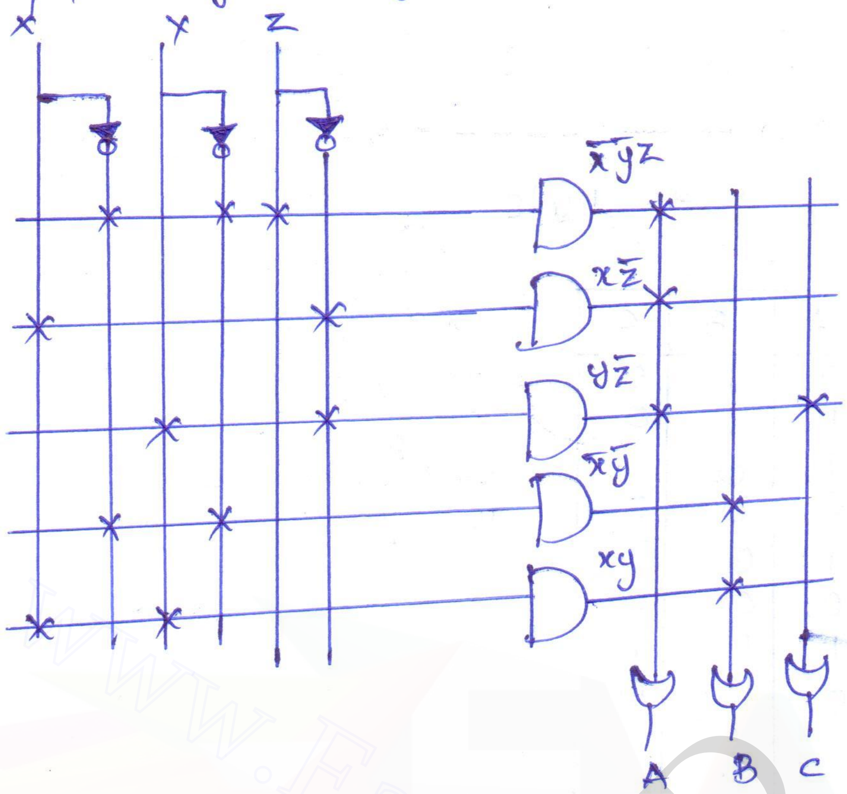
x \ yz	00	01	11	10
0	0	0	0	1
1	0	0	0	1

$C = y\bar{z}$

step3. Program table

	Product term	Inputs			Outputs		
		x	y	z	A	B	C
$\bar{x}\bar{y}z$	1	0	0	1	1	—	—
$x\bar{z}$	2	1	—	0	1	—	—
$y\bar{z}$	3	—	1	0	1	—	1
$\bar{x}\bar{y}$	4	0	0	—	—	1	—
xy	5	1	1	—	—	1	—

step 4: logic diagram



Implement the following function using PROM.

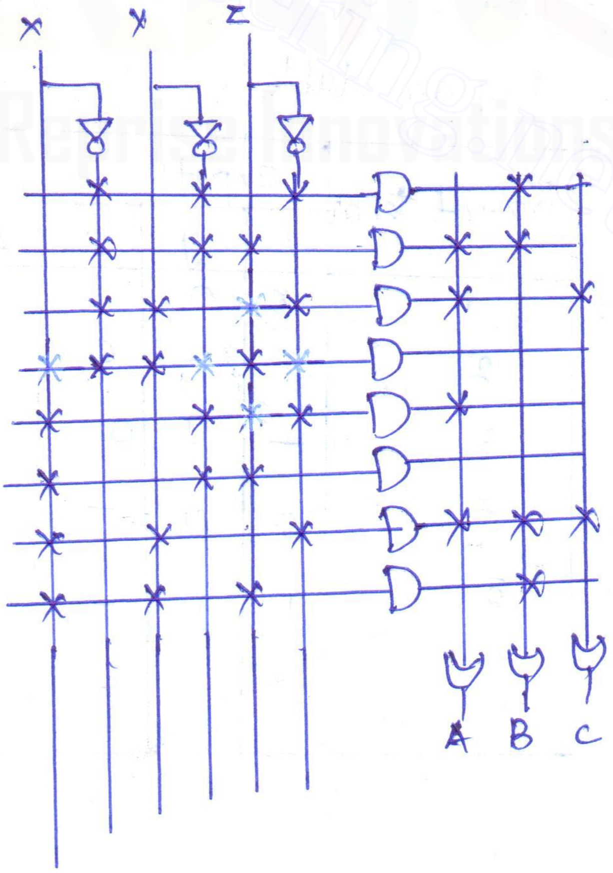
$A(x,y,z) = \Sigma m(1,2,4,6)$

$B(x,y,z) = \Sigma m(0,1,6,7)$

$C(x,y,z) = \Sigma m(2,6)$

Soln:

x	y	z	A	B	C
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	1	0



PAL problem.

6. Design BCD to Excess-3 converter using PAL

Step 1 : Derive the truth table of BCD to Excess-3 converter.

Decimal	BCD code				Excess-3 code			
	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

Step 2: Simplify the boolean functions for Excess 3 code outputs.

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$E_3 = B_3 + B_2B_0 + B_2B_1$

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	1	1
01	1	0	0	0
11	X	X	X	X
10	0	1	X	X

$E_2 = B_2\overline{B_1}\,\overline{B_0} + \overline{B_2}B_0 + \overline{B_2}B_1$

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

$E_1 = \overline{B_1}\,\overline{B_0} + B_1B_0$

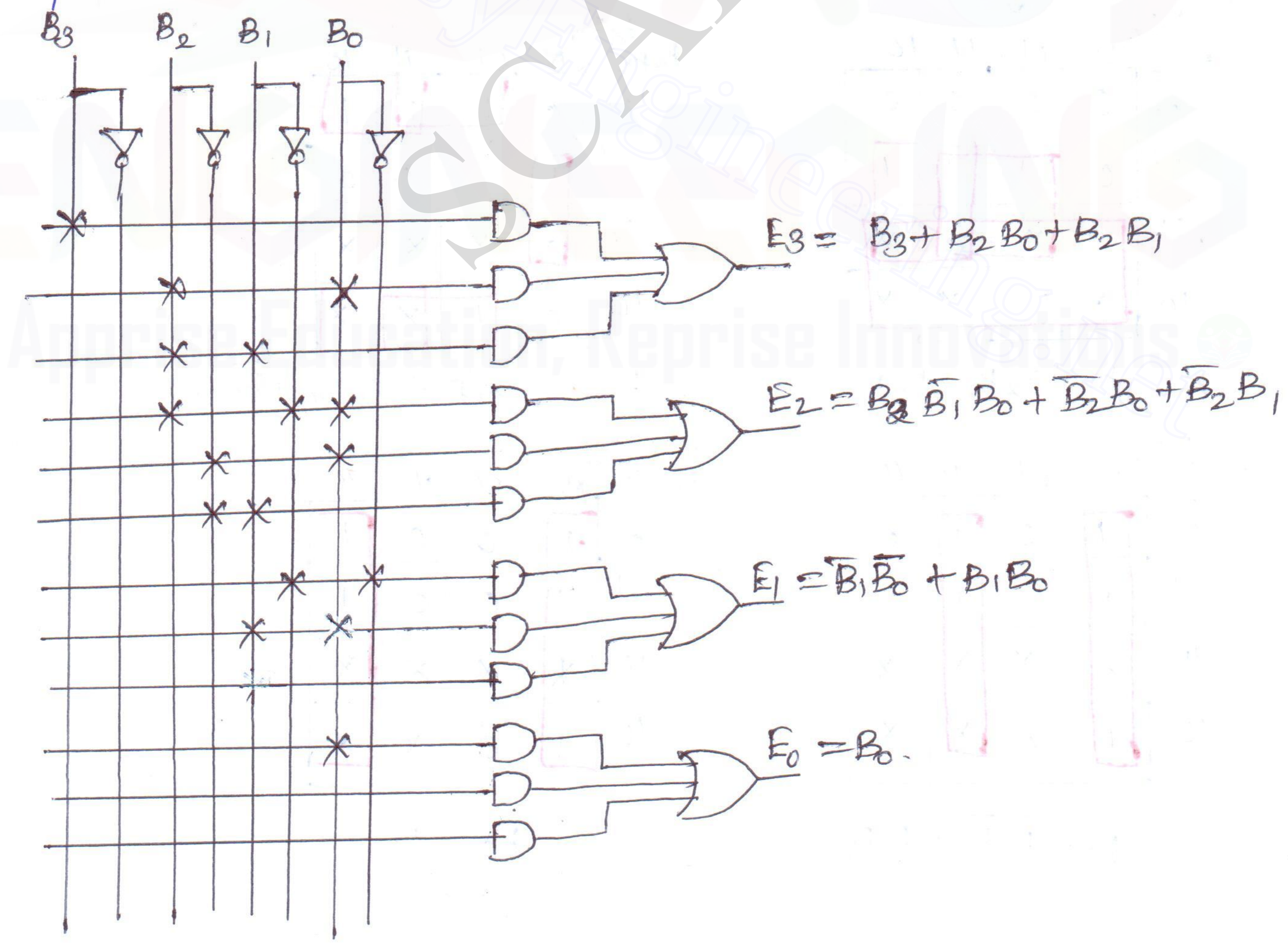
B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$E_0 = B_0$.

Step 3: Implementation

	Product terms	Inputs				Outputs
		B_3	B_2	B_1	B_0	
B_3	1	1	—	—	—	$E_3 = B_3 + B_2 B_0 + B_2 B_1$
$B_2 B_0$	2	—	1	—	1	
$B_2 B_1$	3	—	1	1	—	
	4	—	1	0	0	$E_2 = B_2 \bar{B_1} \bar{B_0} + \bar{B_2} B_0 + \overline{B_2} B_1$
	5	—	0	—	1	
	6	—	0	1	—	
	7	—	—	0	0	$E_1 = \bar{B_1} \bar{B_0} + B_1 B_0$
	8	—	—	1	1	
	9	—	—	—	—	
	10	—	—	—	1	$E_0 = B_0$
	11	—	—	—	—	
	12	—	—	—	—	

step 4: Logic diagram.



B_3
B_2
B_1
B_0
$E_3 = B_3 + B_2 B_0 + B_2 B_1$
$E_2 = B_2 \bar{B_1} B_0 + \bar{B_2} B_0 + \bar{B_2} B_1$
$E_1 = \bar{B_1} \bar{B_0} + B_1 B_0$
$E_0 = B_0$.

7. Hazard. [NOV / DEC - 2014]

*The unwanted swftching transients that may appear at the output of a circuit are called Hazards.

Types!

- * Static hazard
  - * Static '0' hazard
  - * Static 1 hazard
- * Dynamic hazard

Static - 1 hazard

*A combinatinal circuit of output goes momentarily 0 when it should remain a '1' ~~are~~ the hazard is static 1 hazard.

Static 0 hazard

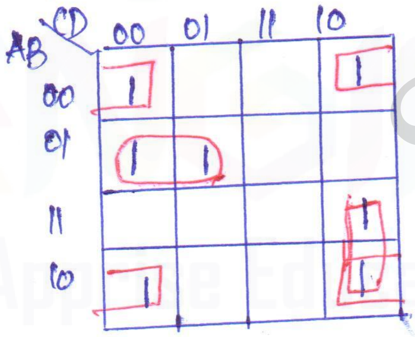
*A combinational circuit of output goes momentarily 1 when it should remain a '0' the hazard is static 0 hazard.

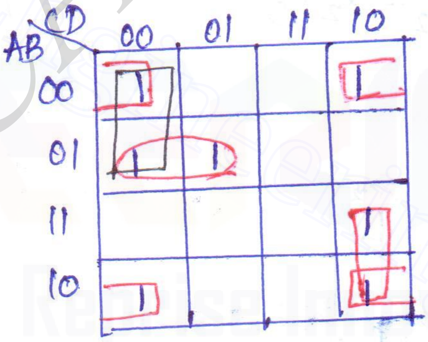
Hazard free problem.

Give hazard free realisation for the following boolean function

$f = \Sigma(0, 2, 4, 5, 8, 10, 14)$ [NOV 2014], [APRIL / MAY 2015]

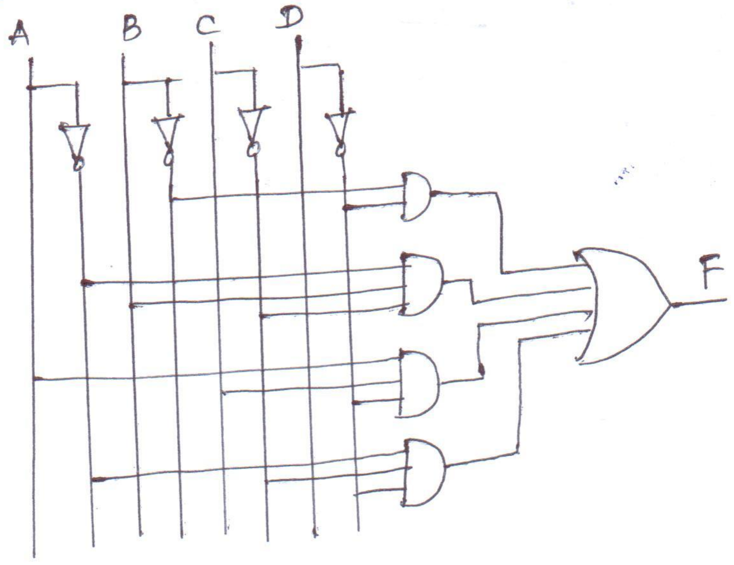
Soln:





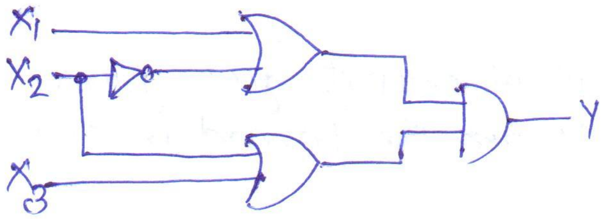
$F = \bar{B}\bar{D} + \bar{A}B\bar{C} + AC\bar{D}$

$F = \bar{B}\bar{D} + \bar{A}B\bar{C} + AC\bar{D} + \bar{A}\bar{C}\bar{D}$



Draw the Logic diagram for the product of sums expression given by $Y = (x_1+\bar{x}_2)(x_2+x_3)$. Show that there is a static-0 hazard when x_1 and x_3 are equal to 0 and x_2 goes from 0 to 1. Find a way to remove the hazard by adding one more OR gate.

Soln: step 1



$\Leftarrow \quad Y = (x_1+\bar{x}_2)(x_2+x_3)$

Step 2:

Eliminating a hazard.

$x_1 \backslash x_2x_3$	00	01	11	10
0	0	1	0	0
1	0	1	1	1

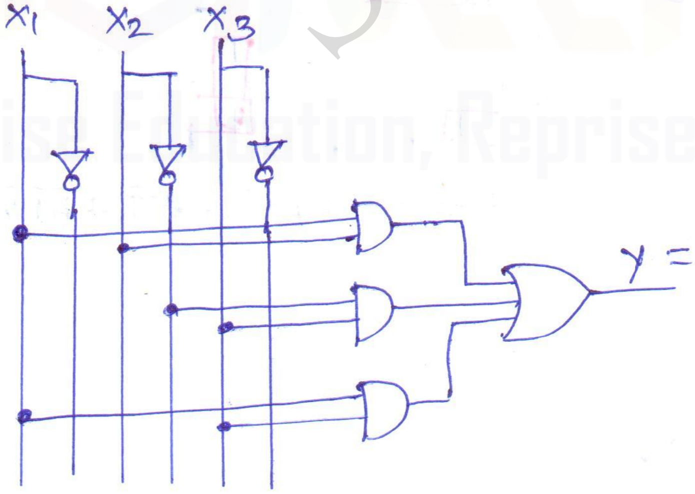
$Y = x_1x_2 + \bar{x}_2x_3$

$x_1 \backslash x_2x_3$	00	01	11	10
0	0	1	0	0
1	0	1	1	1

$Y = x_1x_2 + \bar{x}_2x_3 + x_1x_3$

Step: 3

Logic diagram.



$Y = x_1x_2 + \bar{x}_2x_3 + x_1x_3$

UNIT-V VHDL

1. What are the various modeling used in Verilog? (Dec 2012, May 2012)

Gate level modeling
Data flow modeling
Structural modeling

2. What is test bench?(Nov 2014)

A test bench is a model which is used to exercise and verify the correctness of a hardware model. Example : Half Adder, Full Adder etc.
Example: Half adder ,Full adder,etc

3. What are the various operators in VHDL ? (Dec 2015, Dec 2012)

1.Logical operators
2.Relational operators
3.Shift operators
4.Adding operators
5.Multiplying operators
6.Miscellaneous operators

4. What is the need of package declaration? (Nov2014)

There are some declaration which are common across many design units. A package is a convenient mechanism to store and share such declarations. A set of declarations contained in a package declaration may be shared by many design units.

5. What are the advantages of hardware language? (May 2014)

- HDLs are used to describe hardware for the purpose of simulation, modeling, testing, design and documentation.
- HDL makes it easy to exchange the ideas between and designers.
- The HDL represents digital systems in the form of documentation which can be understood by human as well as computers.

6. Define Packages. (May 2015, May 2011)

A packages is a convenient mechanism to store and share such declarations. A set of declarations contained in a package declaration may b shared by many design units. It defines items that can be made visible to other design units.

7. What are the needs of VHDL (May 2013)

The most prominent modern HDLs in industry are Verilog and VHDL. Verilog is one of the two major hardware description languages (HDLs) used by hardware designers in industry and academic. VHDL is the other one.

8. When can RTL be used to represent digital systems? (May 2011)

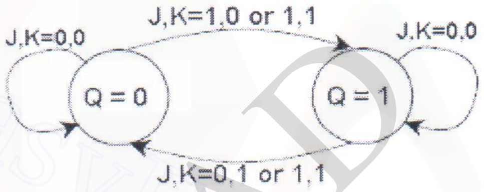
When digital systems are composed of registers and combinational function blocks. The RTL can be used to represent digital systems.

5. Draw the truth table for JK flip flop (May 2013)

J	K	Q_{n+1}	Action
0	0	Q_n	**No Change**
0	1	0	RESET
1	0	1	SET
1	1	Q_n'	**TOGGLE**

6. Give the characteristic equation and state diagram of JK flip-flop. (May 2012)

Characteristic equation $Q_{n+1}=J\,Q'_n + K'\,Q_n$



7. The JK flip-flop is an universal flip-flop. Justify. (Nov 2012)

The JK flip-flop is called an universal flip-flop because it can be easily configured to work as any other flip-flops such as T flip-flop, D flip-flop and SR flip-flop.

8. Define sequential circuit?(May 2014)

In sequential circuits the output variables dependent not only on the present input variables but they also depend up on the past history of these input variables.

9. Give the comparison between combinational circuits and sequential circuits.

Combinational circuit	Sequential circuit
When the logic gates connected together to produce specified output for the specified in put variables.	The output is not only depend upon the input variables and also depend upon past history of the input variables.
Combinational circuits are not have storage elements	Sequential circuit have storage elements
There is no clock pulse	It have clock pulse

1. Explain Register Transfer level design in detail with suitable example.

RTL Design (Register transfer level design) [NOV/DEC 2015], [MAY 2012]

* Register transfer level design lies between a purely behavioral description of the desired circuit and a purely structural one.

* RTL description a circuit register and the sequence of transfer between these register but does not describe the hardward used to carry out these operations.

* RTL design steps are.

* Determine the number and sizes of registers needed to hold the data used by the device.

* Determine the logic and arithmetic operations that need to be performed on these register contents, and.

* Design a sequential circuit whose output control how the register contents are updated inorder to obtain the desired results.

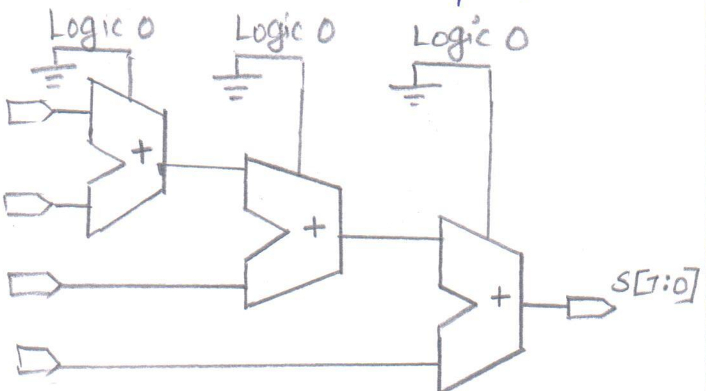
* An RTL design is similar to writing a computer program in a conventional programming language.

* Choosing registers is the same as choosing variables

* Designing the flow of data in the "datapath" is analogous to writting expressions involving the variables and operators.

* Designing the controller sequential circuit is similar to deciding on the flow control within the program.

S<= ((a+b)+c)+d;



* This particular description is simple enough that it can be synthesized.

* The resulting circuit will be a fairly large combinational circuit comprising three adder circuits.

* A behavioral description, not being concerned with implementation details, would be complete at this point.

```
s=0;
s = s+a;
s = s+b;
s = s+c;
s = s+d;
```

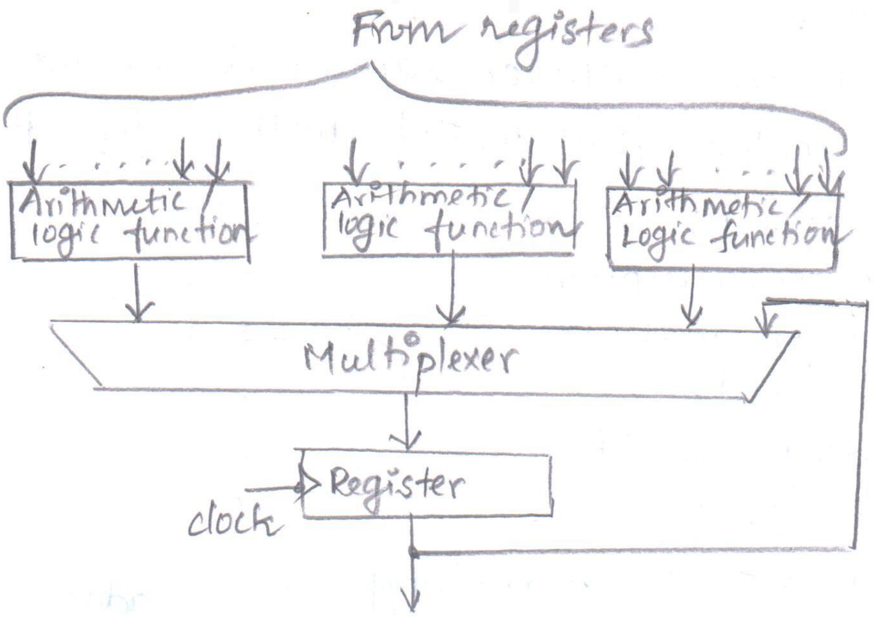
where each operation is executed sequentially.

* The logic required is now one adder, a register to hold the value of s in-between operations, a multiplexer to select the input to be added, and a circuit to clear 's' at the start of the computation.

* The process requires more steps and will take longer time. circuits that divide up a computation into a sequence of arithmetic and logic operations are quite common and this type of design is called RTL (or) 'data flow' design.

RTL design composed of

* 'Registers and combinational function blocks called data path. The controller that controls the transfer of data through the function blocks and between the registers.'


From registers
Arithmetic / logic function
Arithmetic / logic function
Arithmetic / Logic function
Multiplexer
Register
clock

* RTL design, the gate level design and optimization of the datapath (register, multiplexer, and combinational functions) is done by the synthesizer.

* The designer must design the sequential circuit and decide which register transfers are performed in which state.

* RTL designer can trade off datapath complexity against speed (using more adder)

* RTL design is well suited for the design of CPUs and special purpose processors. such as disk drive controller, video ~~pager~~ display cards, network adapter cards, etc.

* The width of registers, types of combinational functions and their input will be determined by the application.

Examples:

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use work.summer.all;
use work.summer_components.all;
entity summer is
port (a,b,c,d : in num ; sum : out num;
                update, clk : in std_logic);
end summer;
architecture rtl of summer is
 signal sel : std_logic_vector (1 downto 0);
 signal load, clear : std_logic;
begin   d1: datapath port map (a,b,c,d, sum, sel, load,
                          clear, clk);
        c1: controller port map (update, sel, load, clear,
                                        clk);
end rtl;
```

2. Briefly explain the use of 'Packages' in VHDL with suitable example.

Package declaration. [NOV/ DEC 2012]

* A package is a convenient mechanism to store and share shach declarations.

* It is an optional design unit..

* A set of declarations contained in a package declaration may be shared by many design units. It defines items that can be made visible to other design units.

* A package is represented by

* Package declaration

* Package body.

Package declaration

* It defines the interface to the package.

Syntax:

```
PACKAGE package-name IS
  type        declarations
  Subtype     declarations
  constant    declarations
  signal      declarations
  variable    declarations
  Subprogram  declarations
  file        declarations
  alise       declarations
  component   declarations
  attribute   declarations
  attribute   specifications
  disconnection specifications
  use clauses
END package-name;
```

* The items declare in a package declaration can be accessed by other design units by using the ~~[illegible]~~ 'library' and 'use' clauses.

Package Body

* It contains the details of a package, that is the behavior of the subprograms and the values of the deferred constants which are declared in a package declaration.

* The package body may contain other. declarations.

Syntax:

```
PACKAGE BODY package-name IS
  subprogram bodies
  complete constant declarations
  subprogram declarations.
  type and subtype declarations
  file and alias declarations
  use clauses
  END package_name;
```

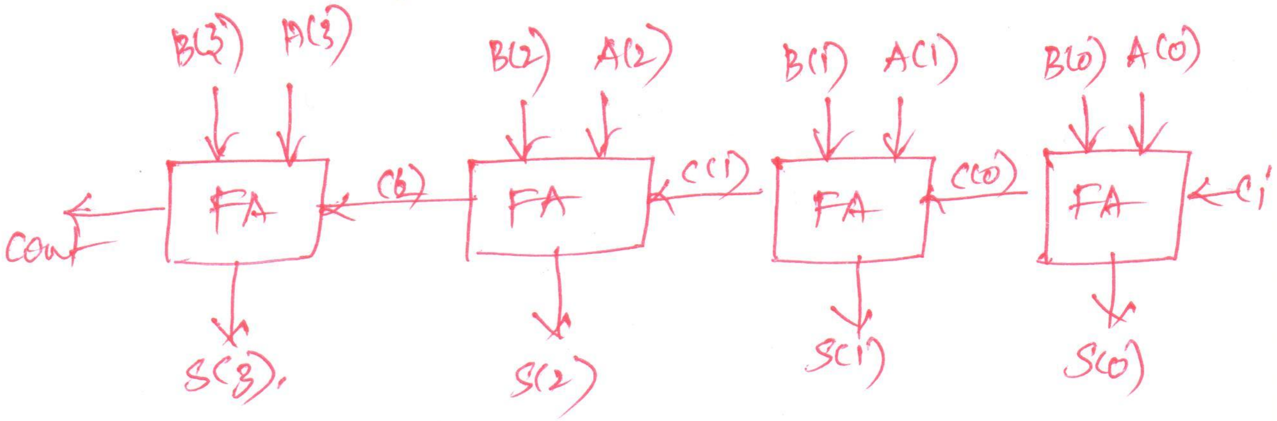
* The name of the package must be same as the name of its corresponding package declaration.

* The package declaration does not have any subprogram (or) deferred constant declarations, a package body is not necessary.

Example:

The 4 bit fulladder circuit include Package declaration.

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.All;
ENTITY fulladder IS
 PORT (x,y,cin : in bit; Cout, Sum: out bit);
 END fulladder;
 ARCHITECTURE equation OF fulladder IS
  BEGIN
      sum <= x xor y xor cin;
      cout <= (x and y) or (x and cin) or (y and cin);
  END equation;
 ENTITY adder4 IS
 PORT (a,b : in bit_vector(3 downto 0); Ci : in bit;
      S : out bit_vector(3 downto 0); Co : out bit);
 END adder4;
 ARCHTECTURE structure OF adder4 IS
  COMPONENT fulladder
   PORT (x, y, cin : in bit; sum, cout : out bit);
   END component;
   SIGNAL C: bit_vector(3 down to 1);
    BEGIN
    FA0: fulladder port map (a(0), b(0), Ci, C(1), S(0));
    FA1: fulladder port map (a(1), b(1), C(1), C(2), S(1));
   FA2: fulladder port map (a(2), b(2), C(2), C(3), S(2));
    FA3: fulladder port map (a(3), b(3), C(3), Co, S(3));
     END structure;
```

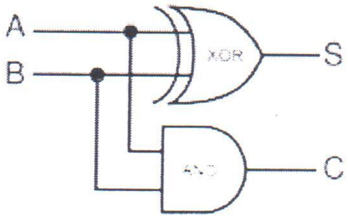


4-bit adder

①

3(i) Write HDL for half adder

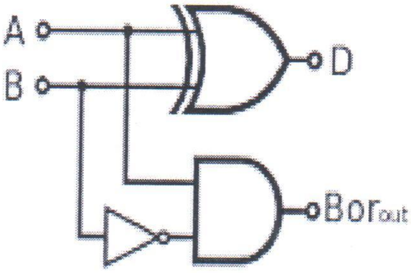
Half adder



```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity halfadder is
    port(
        a,b: in STD_LOGIC;
       S ,  C: out STD_LOGIC );
end halfadder;

architecture half_add_arc of
halfadder  is
begin

   S<= a xor b;
   C <= a and b;

end  half_add_arc;
```

write HDL for half subtractor.

Half subtractor



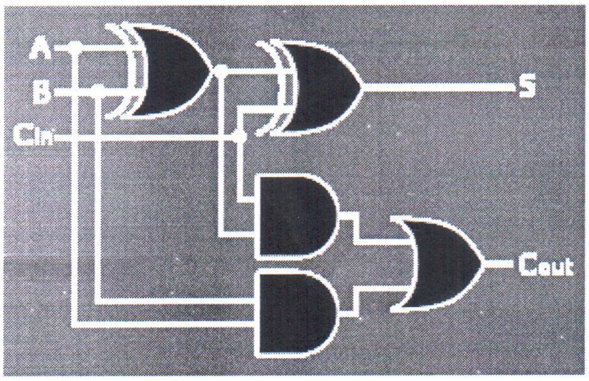
```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity half_subtractor is
    port(
        a,b: in STD_LOGIC;
        diff ,  borrow: out STD_LOGIC
);
end half_subtractor;

architecture half_subtractor_arc of
half_subtractor is
begin

   diff <= a xor b;
   borrow <= (not a) and b;

end half_subtractor_arc;
```

3(ii) write HDL for full adder

[NOV/DEC 2015]
[APRIL/MAY 2015]

FULL ADDER

VHDL CODE FOR FULL ADDER

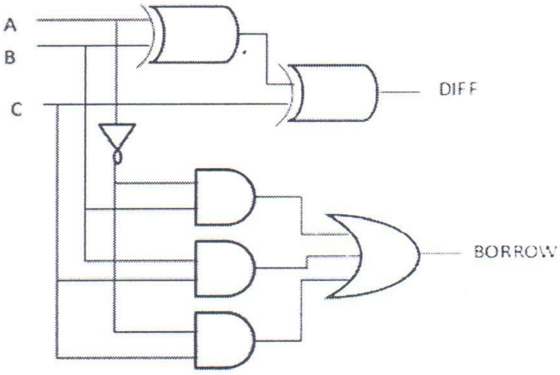


VHDL Code for Full Adder:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fulladd is
 Port ( A : in STD_LOGIC;
 B : in STD_LOGIC;
 Cin : in STD_LOGIC;
 S : out STD_LOGIC;
 Cout : out STD_LOGIC);
end fulladd;

architecture Behavioral of  fulladd is
begin

 S <= A XOR B XOR Cin ;
 Cout <= (A AND B)
                OR
            (Cin AND A)
                OR
            (Cin AND B) ;

end Behavioral;
```

write HDL for full subtractor

FULL SUBTRACTOR



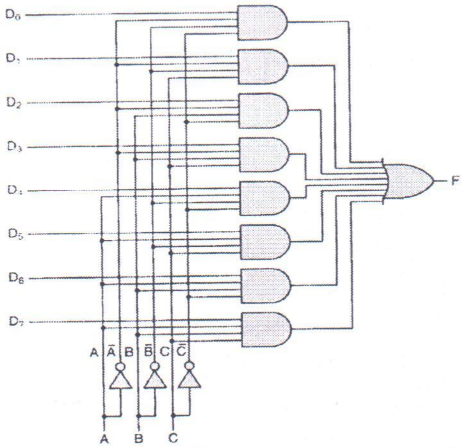
DIFF = A ⊕ B ⊕ C

BORROW = A'.B + B.C + A'.C

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity full_subtractor is
    port(
        a,b,c : in STD_LOGIC;

 difference, borrow : out
STD_LOGIC );
end full_subtractor;

architecture fullsub of full_subtractor
is
begin

   difference <= a xor b xor c;
   borrow <= ((not a) and b)
                   or
                 (b and c)
                   or
               (c and (not a));

end fullsub;
```

4 write HDL program for 8:1 multiplexer [MAY/JUNE 2014]

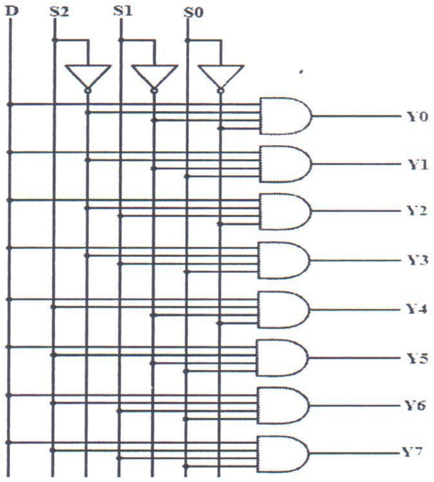
8 :1 multiplexer



```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity mux8_1 is
    port(
        d0,d1,d2,d3,d4,d5,d6,d7 : in bit;
        s0,s1,s2: in  bit;
        F: out bit );
end mux8_1;

architecture mux8_1arc of  mux8_1 is

signal x0,x1,x2,x3,x4,x5,x6,x7 : bit;
begin
x0 <= d0 and(not s0) and (not s1) and (not s2);

x1 <= d1 and (not s0)  and (not s1) and  s2;

x2 <= d2 and (not s0)  and s1  and (not s2);

x3 <= d3 and  (not s0)    and  s1 and s2;

x4 <= d4 and s0 and (not s1) and (not s2);

x5 <= d5 and  s0  and (not s1) and s2;

x6 <= d6 and s0  and s1  and (not s2);

x7 <= d7 and s0 and s1 and s2;

F<= x0 or x1 or x2 or x3 or x4 or x5 or x6
or x7;

end demux1_8arc;
```

write HDL program for 1:8 demultiplexer. [NOV/DEC 2015]

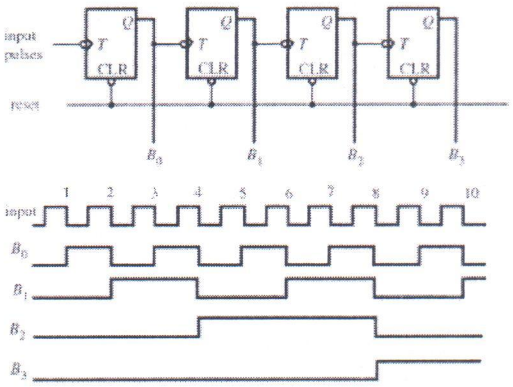
1:8 Demultiplexer



```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity demux1_8 is
    port(
        din : in bit;
        s0,s1,s2: in  bit;
        d0,d1,d2,d3,d4,d5,d6,d7 : out bit;
        );
end demux1_8;

architecture demux1_8arc of demux1_8 is
begin
d0 <= din and(not s0) and (not s1) and (not s2

d1 <= din and (not s0)  and (not s1) and  s2;

d2 <= din and (not s0)  and s1  and (not s2);

d3 <= din and  (not s0)    and  s1 and s2;

d4 <= din and s0 and (not s1) and (not s2);

d5 <= din and  s0  and (not s1) and s2;

d6 <= din and s0  and s1  and (not s2);

d7 <= din and s0 and s1 and s2;

end demux1_8arc;
```

5. write VHDL program for 4-Bit Up counter.

4-Bit Binary Up Counter



VHDL Code for 4-bit binary counter

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter is
port(C, CLR : in std_logic;
Q : out std_logic_vector(3 downto
0));
end counter;

architecture bhv _ counter is
signal tmp: std_logic_vector(3
downto 0);
begin
process (C, CLR)
begin
if (CLR='1') then
tmp <= "0000";
elsif (C'event and C='1') then
tmp <= tmp + 1;
end if;
end process;
Q <= tmp;
end bhv _ counter;
```

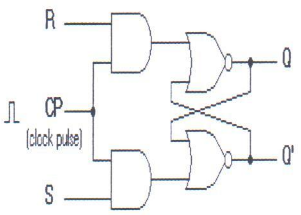
write VHDL program for MOD-6 synchronous counter.

MOD-6 synchronous counter [MAY / JUNE 2012]

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity mod6_counter is
    port( clk, reset: in STD_LOGIC;
dout : out
STD_LOGIC_VECTOR(2 downto
0) );
end mod6_counter;

architecture mod6 of mod6_counter
is
begin

   counter : process (clk,reset) is
   variable m : integer range 0 to 7
:= 0;
   begin
      if (reset='1') then
         m := 0;
      elsif (rising_edge (clk)) then
         m := m + 1;
      end if;
      if (m=6) then
         m := 0;
      end if;
      dout <=
conv_std_logic_vector (m,3);
   end process counter;

end mod6;
```

6. Write VHDL Program for Flipflops.

SR FlipFlop

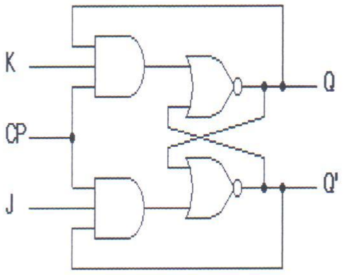


Q S R	Q(t+1)
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	indeterminate
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	indeterminate

VHDL Code for SR FlipFlop

```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;
 entity SR_FF is
PORT( S,R,CLOCK: in std_logic;
Q, QBAR: out std_logic);
end SR_FF;
 Architecture behavioral of SR_FF is
begin
PROCESS(CLOCK)
variable tmp: std_logic;
begin
if(CLOCK='1' and CLOCK'event) then
if(S='0' and R='0')then
tmp:=tmp;
elsif(S='1' and R='1')then
tmp:='Z';
elsif(S='0' and R='1')then
tmp:='0';
else
tmp:='1';
end if;
end if;
Q <= tmp;
QBAR <= not tmp;
end PROCESS;
end behavioral;
```

JK-FlipFlop

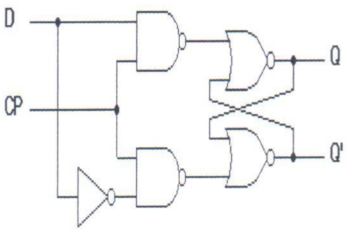


Q J K	Q(t+1)
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	1
1 0 0	1
1 0 1	0
1 1 0	1
1 1 1	0

VHDL Code for JK FlipFlop

```
library ieee;
 ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;
 entity JK_FF is
PORT( J,K,CLOCK: in std_logic;
Q, QB: out std_logic);
end JK_FF;
 Architecture behavioral of JK_FF is
begin
PROCESS(CLOCK)
variable TMP: std_logic;
begin
if(CLOCK='1' and CLOCK'EVENT) the
if(J='0' and K='0')then
TMP:=TMP;
elsif(J='1' and K='1')then
TMP:= not TMP;
elsif(J='0' and K='1')then
TMP:='0';
else
TMP:='1';
end if;
end if;
Q<=TMP;
QB<=not TMP;
end PROCESS;
end behavioral;
```

D FlipFlop

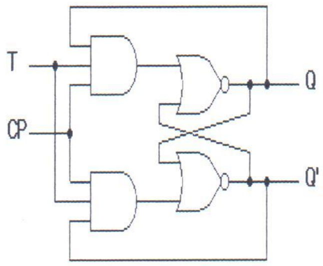


Q D	Q(t+1)
0 0	0
0 1	1
1 0	0
1 1	1

VHDL Code for D FlipFlop

```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;
entity D_FF is
PORT( D,CLOCK: in std_logic;
Q: out std_logic);
end D_FF;
architecture behavioral of D_FF is
begin
process(CLOCK)
begin
if(CLOCK='1' and CLOCK'EVENT) then
Q<=D;
end if;
end process;
end behavioral;
```

T FlipFlop



Q T	Q(t+1)
0 0	0
0 1	1
1 0	1
1 1	0

VHDL Code for T FlipFlop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 entity T_FF is
port( T: in std_logic;
Clock: in std_logic;
Q: out std_logic);
end T_FF;
 architecture Behavioral of T_FF is
signal tmp: std_logic;
begin
process (Clock)
begin
if Clock'event and Clock='1' then
 if T='0' then
tmp <= tmp;
          elsif T='1' then
          tmp <= not (tmp);
          end if;
          end if;
          end process;
          Q <= tmp;
          end Behavioral;
```

Reg. No. 9 5 0 7 1 3 1 0 5 3 1 7

Question Paper Code : 27206

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2015.

Third Semester

Electrical and Electronics Engineering

EE 6301 — DIGITAL LOGIC CIRCUITS

(Common to Electronics and Instrumentation Engineering and Instrumentation and Control Engineering)

(Regulations 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. What is an unit distance code? Give an example.. Pg. No: 1
2. Define Fan-out. Pg. No: 1
3. Convert the given expression in canonical SOP form $Y = AB + A'C + BC'$. Pg. No: 30
4. Draw the logical diagram of EX-OR gate using NAND gates. Pg. No: 30
5. Draw the truth table and state diagram of SR flip-flop. Pg. No: 48
6. What is edge triggered flip flops? Pg. No: 48
7. What is PROM? Pg. No: 73
8. Compare pulsed mode and fundamental mode asynchronous circuit. Pg. No: 74
9. Write the behavioral model of D flip flop.. Pg. No: 92
10. List out the operators present in VHDL P. No. 91

PART B — (5 × 16 = 80 marks)

11. (a) (i) Draw the CMOS logic circuit for NOR gate and explain its operation. Pg. No: 18 (8)

(ii) Perform the following operation $(756)_8 - (437)_8 + (725)_{16}$.. Express the answer in octal form. (8)

Or

(b) (i) A 12 bit Hamming code word containing 8 bits of data and 4 parity bits is read from memory. What was the original 8 bit data word that was written into memory if the 12 bit word read out is as (1) 101110010100 and (2) 111111110100. Pg. No: 17 (12)

(ii) Briefly discuss weighted Binary code. Pg. No: 10 (4)

12. (a) (i) Simplify the boolean function using K-map and implement using only NAND gates.

$F(A,B,C,D) = \sum m(0,8,11,12,15) + \sum d(1,2,4,7,10,14)$. Pg. No: 32

Mark the essential and non-essential prime implicants. (8)

(ii) Design a full subtractor and implement using logic gates. Pg. No: 35 (8)

Or

(b) (i) Design a 4 bit BCD to excess 3 code converter and implement using logic gates. Pg. No: 42 (8)

(ii) What is a multiplexer? Implement the following Boolean function with 8 × 1 MUX and external gates

$F(A,B,C,D) = \sum m(1,3,4,11,12,13,14,15)$. Pg. No: 46 (8)

13. (a) (i) A sequential circuit with two D flip flops A and B, input X and output Y is specified by the following next state and output equations

$A(t+1) = AX + BX$,

$B(t+1) = A'X$

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

Pg. No: 62

Draw the logic diagram, derive state table and state diagram. (12)

(ii) Realize T flip-flop using JK flip-flop. Pg. No: 71 (4)

Or

(b) (i) Design a synchronous decade counter using T flip flop and construct the timing diagram Pg. No: 55. (8)

(ii) Design a mealy model of sequence detector to detect the pattern 1001. Pg. No: 72 (8)

14. (a) Design an asynchronous sequential circuit (with detailed steps involved) that has 2 inputs x_1 and x_2 and one output z. The circuit is required to give an output z = 1 when $x_1 = 1$, $x_2 = 1$ and $x_1 = 1$ being first. (16)

Pg. No: 79

Or

(b) Show how to program the fusible links to get a 4 bit Gray code from the binary inputs using PLA and PAL and compare the design requirements with PROM. Pg. No: 87 (16)

15. (a) (i) Write a VHDL program for 1 to 4 Demux using dataflow modelling. Pg. No: 101 (8)

(ii) Write a VHDL program for Full adder using structural modelling. (8) Pg. No: 100

Or

(b) Explain in detail the RTL design procedure. Pg. No: 93 (16)

Reg. No. : 9 | 5 | 0 | 7 | 1 | 3 | 1 | 0 | 5 | 3 | 1 | 7

Question Paper Code : 77123

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2015.

Third Semester

Electrical and Electronics Engineering

EE 6301 — DIGITAL LOGIC CIRCUITS

(Common to Electronics and Instrumentation Engineering and Instrumentation and Control Engineering)

(Regulation 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Convert : Pg. No: 1
   (a) $(475.25)_8$ to its decimal equivalent
   (b) $(549.B4)_{16}$ to its binary equivalent.

2. Define propagation delay. Pg. No: 1

3. Convert the given expression in canonical SOP form Pg. No: 30
   $Y = AC + AB + BC$.

4. Simplify the expression $Z = AB + A\overline{B}.(\overline{\overline{A}.\overline{C}})$. Pg. No: 30

5. Convert T Flip Flop to D Flip Flop. Pg. No: 48

6. State the rules for state assignment. Pg. No: 50

7. State the difference between static 0 and static 1 hazard. Pg. No: 3

8. What is a PROM? Pg. No: 73

9. What is a package in VHDL? Pg. No: 91

10. Write the behavioral modeling code for a D Flip Flop. Pg. No: 92

107

PART B — (5 × 16 = 80 marks)

11. (a) (i) Perform the following addition using BCD and Excess-3 addition (205+569). Pg. No: 25 (8)

(ii) Encode the binary word 1011 into seven bit even parity Hamming code. Pg. No:16 (8)

Or

(b) (i) With circuit schematic, explain the operation of a two input TTL NAND gate with totem-pole output. Pg. No: 20 (10)

(ii) Compare totem pole and open collector outputs. Pg. No: 01 (6)

12. (a) (i) Reduce the following function using K-map.

$f(A,B,C,D) = \pi M(0,2,3,8,9,12,13,15)$. Pg. No: 33 (8)

(ii) Design a full adder using two half-adders and an OR gate. (8) Pg. No: 35

Or

(b) (i) Design a BCD to Excess 3 code converter. Pg. No: 42 (8)

(ii) Implement the following Boolean function using 8:1 Mux :

$F(A,B,C,D) = \Sigma m(0,1,3,4,8,9,15)$. Pg. No: 46 (8)

13. (a) (i) Explain the operation of a master slave JK flip flop. Pg. No: 53 (8)

(ii) Design a 3-bit bidirectional shift register. Pg. No: 67 (8)

Or

(b) (i) Design a MOD-5 synchronous counter using JK flip-flops. P. No: 53 (8)

(ii) Design a sequence detector to detect the sequence 101 using JK flip flop. Pg. No: 72 (8)

14. (a) Design an asynchronous sequential circuit that has two inputs X_2 and X_1 and one output Z. When $X_1 = 0$, the output Z is 0. The first change in X_2 that occurs while X_1 is 1 will cause output Z to be 1. The output Z will remain 1 until X_1 returns to 0. Pg. No: 77

Or

(b) ~~(i)~~ Implement the following function using PLA :

$F(x,y,z) = \Sigma m(1,2,4,6)$ Pg. No: 85

(ii) For the given boolean function, obtain the hazard-free circuit

$F(A,B,C,D) = \Sigma m(1,3,6,7,13,15)$. P.No. 89

15. (a) Write the VHDL code to realize a full adder using Pg. No: 100

(i) Behavioral modeling

(ii) Structural modeling. (8 + 8)

Or

(b) Write the VHDL code to realize a 3-bit Gray code counter using case statement. Pg. No: 102 (16)

Reg. No. : 9 5 2 8 1 3 1 0 5 3 1 7

Question Paper Code : 97064

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2014.

Third Semester

Electrical and Electronics Engineering

EE 6301 — DIGITAL LOGIC CIRCUITS

(Common to Electronics and Instrumentation Engineering and Instrumentation and Control Engineering)

(Regulation 2013)

Time : Three hours Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Determine $(377)_{10}$ in Octal and Hexa-Decimal equivalent. Pg. No: 1

2. Compare the totem-pole output with open-collector output? Pg. No: 1

3. Given F = B' + A' B + A' C': Identify the redundant term using K-Map. Pg. No: 30

4. Give one application each for Multiplexer and Decoder. Pg. No: 30

5. Show how the JK flip flop can be modified into a D flip flop or a T flipflop. Pg. No: 48

6. Differentiate between Mealy and Moore models. Pg. No: 50

7. What is a deadlock condition? Pg. No: 50

8. Draw the block diagram of PLA. Pg. No: 74

9. Write a VHDL code for 2×1 MUX. Pg. No: 92

10. State the advantage of package declaration over component declaration. Pg. No. 91

PART B — (5 × 16 = 80 marks)

11. (a) (i) Given that a frame with bit sequence 1101011011 is transmitted, it has been received as 1101011010. Determine the method of detecting the error using any one error detecting code. Pg.No:17 (8)

(ii) Draw the MOS logic circuit for NOT gate and explain its operation. Pg.No:18 (8)

Or

(b) (i) Explain Hamming code with an example. State its advantages over parity codes. Pg.No:10 (8)

(ii) Design a TTL logic circuit for a 3-input NAND gate. Pg.No:20 (8)

12. (a) (i) Minimize the function $F(a, b, c, d) = \Sigma(0, 4, 6, 8, 9, 10, 12)$ with $d = \Sigma(2, 13)$. Implement the function using only NOR gates. (8) Pg.No:32

(ii) Design a Full Subtractor and implement it using logic gates. (8) Pg.No:35

Or

(b) (i) Implement the function $F(p, q, r, s) = \Sigma(0, 1, 2, 4, 7, 10, 11, 12)$ using Decoder. Pg.No:47 (8) (4)

(ii) Design a 4-bit Binary to gray code converter and implement it using logic gates. Pg.No:40 (8)

13. (a) (i) Design an asynchronous Modulo-8 Down counter using JK flipflops. (8)

(ii) Explain the circuit of a SR flip-flop and explain its operation. (8) Pg.No:51

Or

(b) (i) Design synchronous sequential circuit that goes through the count sequence 1,3,4,5 repeatedly. Use T flip-flops for your design. (8) Pg.No:58

(ii) Explain the various types of triggering with suitable diagrams. Compare their merits and demerits. (8)

14. (a) Explain the various types of hazards in sequential circuit design and the methods to eliminate them. Give suitable examples. Pg.No:89 (16)

Or

(b) Describe with reasons, the effect of races in asynchronous sequential circuit design. Explain its types with illustrations. Show the method of race-free state assignments with examples. Pg.No:81 (16)

15. (a) (i) Explain the digital system design flow sequence with the help of a flow chart. (8)

(ii) Write a VHDL code for a 4-bit universal shift register. (8)

Or

(b) Explain the concept of Behavioural modeling and Structural modeling in VHDL. Take the example of Full Adder design for both and write the coding. (16)

Pg. No: 100

$2^3 = 8 \rightarrow 0-7$

-8

$0-7$ 8

2^4 -

8

Reg. No. : | 9 | 5 | 0 | 7 | 1 | 5 | 1 | 0 | 5 | 0 | 2 | 2 |

Question Paper Code : 80366

B.E./B.Tech. DEGREE EXAMINATION, NOVEMBER/DECEMBER 2016.

Third Semester

Electrical and Electronics Engineering

EE 6301 — DIGITAL LOGIC CIRCUITS

(Common to Electronics and Instrumentation Engineering and Instrumentation and Control Engineering)

(Regulations 2013)

Time : Three hours　　　　Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Construct OR gate and AND gate using NAND gates.

2. Convert the following Excess - 3 numbers into decimal numbers.

   (a) 1011

   (b) 1001 0011 0111

3. Convert the given expression in canonical SOP form

   Y = AB + A'C + BC'

4. Draw the truth table of 2 :1 MUX.

5. Differentiate Mealy and Moore model.

6. Draw the state diagram of JK flip flop.

7. What is static hazard and dynamic hazard?

8. Define races in asynchronous sequential circuits.

9. Write VHDL behavioral model for D flip flop.

10. Write the VHDL code for a logical gate which gives high output only when both the inputs are high.

PART B — (5 × 13 = 65 marks)

11. (a) (i) Explain with an aid of circuit diagram the operation of 2 input CMOS NAND gate and list out its advantages over other logic families. (10)

(ii) Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction $Y - X$ by using 2's complements. (3)

Or

(b) (i) Explain in detail the usage of Hamming codes for error detection and error correction with an example considering the data bits as 0101. (10)

(ii) Convert 23.625_{10} to octal (base 8). (3)

12. (a) Simplify the logical expression using K-map in SOP and POS form $F(A,B,C,D) = \Sigma m(0, 2, 3, 6, 7) + d(8,10,11,15)$. (13)

Or

(b) Design a full subtractor and realise using logic gates. Also, implement the same using half subtractors (13)

13. (a) Design a sequence detector that produces an output '1' whenever the non-overlapping sequence 101101 is detected. (13)

Or

(b) (i) Explain the realization of JK flip flop from T flip flop. (7)

(ii) Write short notes on SIPO and draw the output waveforms. (6)

14. (a) Design an asynchronous circuit that has two inputs $x1$ and $x2$ and one output z. The circuit is required to give an output whenever the input sequence (0,0), (0,1) and (1, 1) received but only in that order (13)

Or

(b) (i) Design a PLA structure using AND and OR logic for the following functions. (10)

$F1 = \Sigma m(0, 1, 2, 3, 4, 7, 8, 11, 12, 15)$

$F2 = \Sigma m(2, 3, 6, 7, 8, 9, 12, 13)$

$F3 = \Sigma m(1, 3, 7, 8, 11, 12, 15)$

$F4 = \Sigma m(0, 1, 4, 8, 11, 12, 15)$

(ii) Compare PLA and PAL circuits. (3)

15. (a) Explain in detail the concept of structural modeling in VHDL with an example of full adder. (13)

Or

(b) (i) Write short notes on built- in operators used in VHDL programming. (6)

(ii) Write VHDL coding for 4 × 1 Multiplexer. (7)

PART C — (1 × 15 = 15 marks)

16. (a) Assume that there is a parking area in a shop whose capacity is 10. No more than 10 cars are allowed inside the parking area and the gate is closed as soon as the capacity is reached. There is a gate sensor to detect the entry of car which is to be synchronized with the clock pulse. Design and implement a suitable counter using JK flip flops. Also, determine the number of flip flops to be used if the capacity is increased to 50. (15)

Or

(b) Design a 4 bit code converter which converts given binary code into a code in which the adjacent number differs by only 1 by the preceding number. Also, develop VHDL coding for the above mentioned code converter. (15)

Reg. No. : □□□□□□□□□□□□

Question Paper Code : 71764

B.E./B.Tech. DEGREE EXAMINATION, APRIL/MAY 2017.

Third Semester

Electrical and Electronics Engineering

EE 6301 — DIGITAL LOGIC CIRCUITS

(Common to Electronics and Instrumentation Engineering, Instrumentation and Control Engineering)

(Regulations 2013)

Time : Three hours   Maximum : 100 marks

Answer ALL questions.

PART A — (10 × 2 = 20 marks)

1. Reduce $a(b+bc')+ab'$.

2. Convert 143_{10} into its binary and binary coded decimal equivalent.

3. Write the POS form of the SOP expression $f(x,y,z) = x'yz + xyz' + xy'z$.

4. Design a Half Subtractor.

5. Give the characteristic equation and characteristic table of a T Flip Flop.

6. State the differences between Moore and Melay state machines.

7. What is a flow table? Give example.

8. State the difference between PROM, PAL and PLA.

9. Give the syntax for package declaration and package body in VHDL.

10. Write the VHDL code for a 2 × 1 multiplexer using behavioral modeling.

PART B — (5 × 13 = 65 marks)

11. (a) (i) Design a odd-parity hamming code generator and detector for 4-bit data and explain their logic.

(ii) Convert $FACE_{16}$ into its binary, octal and decimal equivalent.

Or

(b) (i) With circuit schematic explain the working of a two-input TTL NAND gate.

(ii) Compare Totem Pole and open collector outputs.

12. (a) (i) Reduce the following minterms using Karnaugh – Map $f(w,x,y,z) = \sum m\,(0,1,3,5,6,7,8,12,14) + \sum d(9,15)$. (7)

(ii) Implement the following function using a suitable multiplexer $f(a,b,c) = \sum m\,(3,7,4,5)$. (6)

Or

(b) (i) Design a 3 × 8 decoder and explain its operation as a minterm generator. (7)

(ii) Design a full adder using only NOR gates. (6)

13. (a) (i) Draw and explain the operation of a Master – Slave JK Flip Flop. (7)

(ii) Design a 5-bit ring counter and mention its applications. (6)

Or

(b) (i) Design a 4-bit parallel-in serial-out shift register using D Flip Flops. (7)

(ii) Using partitioning minimization procedure reduce the following state table : (6)

Present state	Next state		Output
	$w=0$	$w=1$	Z
A	B	C	1
B	D	F	1
C	F	E	0
D	B	G	1
E	F	C	0
F	E	D	0
G	F	G	0

14. (a) A control mechanism for a vending machine accepts nickels and dimes. It dispense merchandise when 20 cents is deposited ; it does not give change if 25 cents is deposited. Design the FSM that implements the required control, using as few states as possible. Find a suitable assignment and derive next-state and output expressions. (13)

Or

(b) (i) Implement the following logic and analyse for the pressure of any hazard $f = x_1x_2 + \bar{x}_1x_3$. If hazard is present briefly explain the type of hazard and design a hazard-free circuit. (7)

(ii) Implement the following functions using programmable logic array :

$f_1(x, y, z) = \sum m(0, 1, 3, 5, 7)$

$f_2(x, y, z) = \sum m(2, 4, 6)$. (6)

15. (a) Design a 3 –bit magnitude comparator and write the VHDL code to realize it using structural modeling. (13)

Or

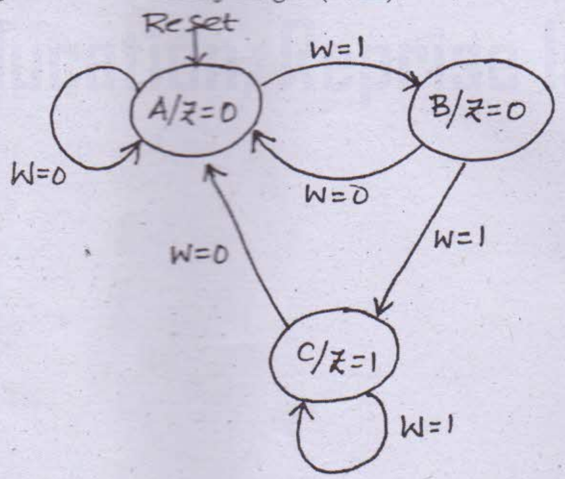
(b) Design a 4 × 4 array multiplier and write the VHDL code to realize it using structural modeling. (13)

PART C — (1 × 15 = 15 marks)

16. (a) Design a CMOS inverter and explain its operation. Comment on its characteristics such as Fan-in, Fan-out power dissipation, propagation delay and noise margin. Compare its advantages over other logic families. (15)

Or

(b) Write the VHDL code for the given state diagram, using behavioral modeling. Design it using one-hot state assignment and implement it using Programmable Array Logic (PAL). (15)


Reset
W=1
A/z=0
B/z=0
W=0
W=0
W=0
W=1
C/z=1
W=1