INDEX

Sl.No.	Date	Experiment	P. No.	Conduction & Result	Journal Submission	Sign
01	23/8/24	A] Realize 3-variable & 4-variable Boolean expressions, Simplify using k-map and implementing basic gates.				
		B] Simulate and Verify the working of above expressions using VHDL.	1-3			
02	30/8/24	A] Design and Implement Half Adder and Full Adder using Basic Gates.				
		B] Simulate and Verify the working of Half Adder & Full adder using VHDL	4-6			
03	6/9/24	A] Given a 4-variable logic expression, simplify it using Entered Variable Map and realize the simplified logic expression using 8:1 multiplexer.				
		B] Simulate and Verify the working of 8:1 multiplexer using VHDL.	7-9			
04	19/10/24	A] Design and Implement the Binary to Gray Code Converter using Basic Gates.				
		B] Simulate and Verify the working of Binary to Gray Code Converter using VHDL.	10-12			
05	26/10/24	Design and Implement the Truth Table of 3-bit parity Generator and 4-Parity checker with an Even parity bit using Basic Gates.	13-15			

INDEX & EVALUATION

Sl.No.	Date	Experiment	P. No.	Conduction & Result	Journal Submission	Sign
06	9/11/24	A] Realize a J-K Master/slave FF using NAND gates & verify its truth table.				
		B] Simulate & Verify the working of D FF with positive edge triggering using VHDL.	16-18			
07	16/11/24	A] Realize Design & Implement a MOD-n(n<8) synchronous UP Counter using J-K FF Ics.				
		B] simulate and verify the working of mod-8 up counter using VHDL.	19-21			
08.	23/11/24	A] Design & Implement an Asynchro-nous Counter Decade Counter Ic to Count up from 0 to n(n<=9) & demonstrate on 7-Segment Display (using IC7447).				
09.		B] Simulate & Verify the working of switched tail Counter using VHDL.	23-24			

Average :	
Lab IA Test :	
Final Marks :	

Signature of the Lab Incharge

Date :

Exp. No.

Exp. Title

Steps to turn of Vmware and Xilinx

Page No.

1. Turn on pc
2. Select Vmware on Desktop and double click.
3. Select windows 7 and click on power on button.
4. Slect user from login window, Switch user → other user
   username :- user, password :- KU1234 (Not required)
5. Click on ISE Design Suite on desktop

6. Steps to execute program

Vmware workstation → Poweron → ISE Design Suite
New Project → location → Make new folder (your name → save) → Next
Family → spartan 3
Device → XC3S400
Package → PQ208
Speed → 5
~~Product~~ to. Preferred lang → VHDL
Simulator → isim
Hirarchy → XC 3400-SPQ208 [Right click → new source]
→ VHDL Module → program name → Next → Define Module → next) (Given your inputs & outputs)
Type program → save
Run synthesize XST (check system) → * Select implement
Right click on XC3S400-PQ208
Isim simulator
Run Behavioral model.

3- Variable Boolean Expression

$F(A, B, C) = \Sigma m(0, 2, 4, 5, 6, 7)$

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

K-Map

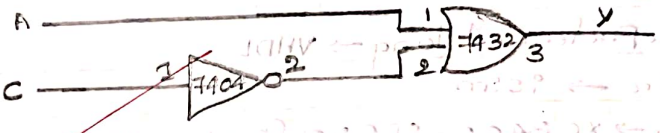
A \ BC	$\bar{B}\bar{C}$ (00)	$\bar{B}C$ (01)	BC (11)	$B\bar{C}$ (10)
$\bar{A}$ (0)	1 (0)	0 (1)	0 (3)	1 (2)
A (1)	1 (4)	1 (5)	1 (7)	1 (6)

$y = A + \bar{C}$

Truth Table

A	C	y
0	0	1
0	1	0
1	0	1
1	1	1

Circuit diagram



4- Variable Boolean Expression

$F(A, B, C, D) = \Sigma m(0, 2, 5, 7, 8, 10, 13, 15)$

$y = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + AB\bar{C}D + ABCD$

Date: 28/8/24 | Exp. Title: Realize 3-variable and 4-Variable Boolean expressions, Simplify using K-map and Implementing Basic gates. | Page No. 01

Exp. No. 01 [A]

Aim: Realize 3-variable and 4-variable Boolean expressions, Simplify using K-map and Implementing Basic Gates.

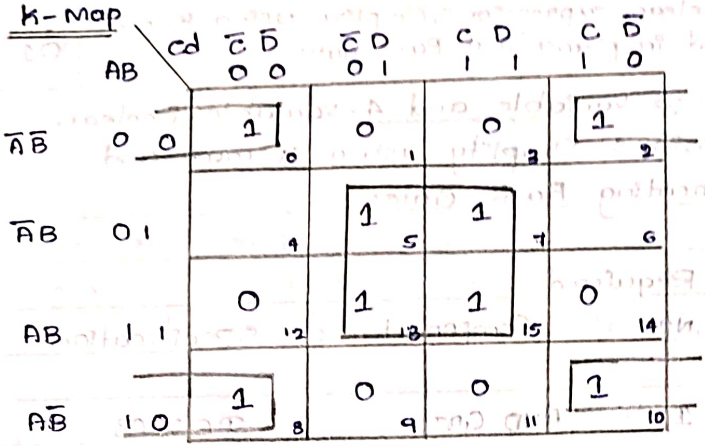
Apparatus Required

Sl.No.	Component	Specification
1	AND GATE	IC 7408
2	OR GATE	IC 7432
3	NOT GATE	IC 7404
4	IC TRAINER GATE	–
5	PATCH CORDS	–

Theory: Karnaugh maps are used to simplify real-world logic requirements so that they can be implemented using a minium number of logic gates. A sum of products expression (SOP) can always be implemented using AND gates feeding into an OR gate, and product-of sums expression (POS) leads to OR gates feeding and AND gate.

The pos expression gives a Complement of the function (if F is the function so its complement will be F). Karnaugh maps can also be used to simplify logic expressions in software design. Boolean conditions as used for example in Conditional statements, can get very complicated, which makes the codedifficult to read and to maintain. Once minimized, canonical sum-of-products and product-of-Sums expressions can be implemented directly using AND and OR logic operators.

K-Map

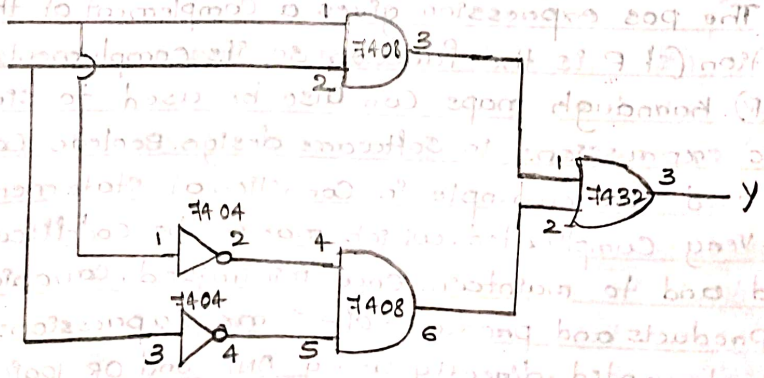


$y = B\bar{D} + BD'$

Truth table

Inputs						Output
B	D	B'	D'	B'D'	BD	B'D'+BD
0	0	1	1	1	0	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	0	1	1

Circuit diagram



Applications of Karnaugh's Map

* They are used in design and implementation of Digital Circuits.
* k-maps are useful for detecting and eliminating race condition.

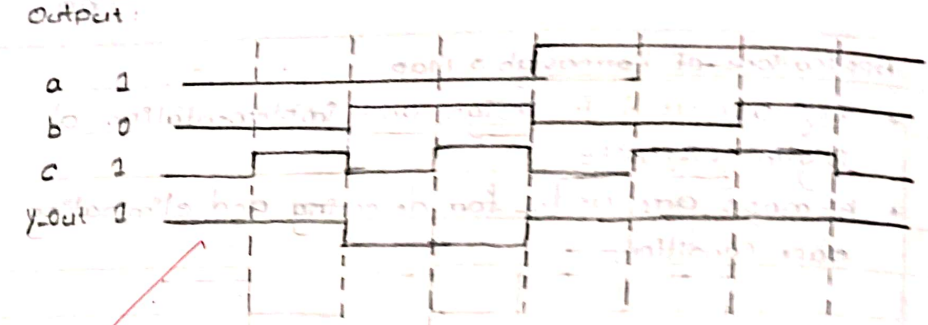
Procedure

1. Make sure that the given components are working properly before connecting to the trainer circuit.
2. Place the given IC's in the sockets provided on the trainer circuit.
3. Make the connections as per the circuit diagram. Now on the power supply.
4. For the different combinations of the inputs given in the truth table, and verify the outputs in trainer kit.
5. Switch off the power supply and remove the connections.

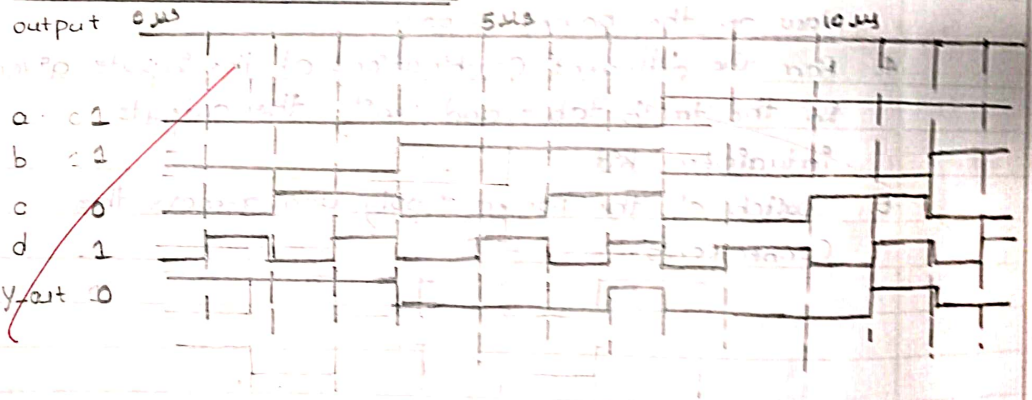
Result:

3 variable and 4-variable Boolean expressions are simplified and verified using Basic gates

3- Variable Boolean Expression



4 - Variable Boolean Expression



Date :

Exp. No. 01 [B]

Exp. Title Simulate and verify the working of above expressions using VHDL.

Page No. 03

3- variable Boolean Expression

$Y = A + \bar{C}$

```
entity three_variable_booleanExp
  is port (a,b,c : in STD_LOGIC; y_out: out STD_LOGIC);
end three_variable_booleanExp;
architecture Behavioral of three_variable_booleanExp is
begin
    y_out <=(a or (not(c)));
end Behavioral;
```

4- variabe Boolean Expression

$Y = \bar{B}\bar{D} + BD$

```
entity Four_variableBool_Exp is
  Port (a,b,c,d : in STD_LOGIC; y_out: out STD_LOGIC);
end Four_variableBool_Exp;
architecture Behavioral of Four_variableBool_Exp is begin
  y_out<=(((not(b)) and (not(d))) or (b and d));
    end Behavioral;
```

Result :

Simulation and verification of the working of above expressions using VHDL is verified.

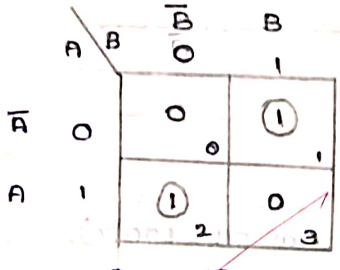
Half Adder

Truth Table

Cell No.	Input		Output	
	A	B	Sum	Carry
0	0	0	0	0
1	0	1	1	0
2	1	0	1	0
3	1	1	0	1

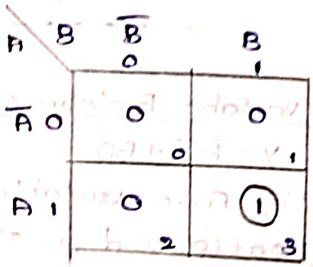
K-Map

Sum



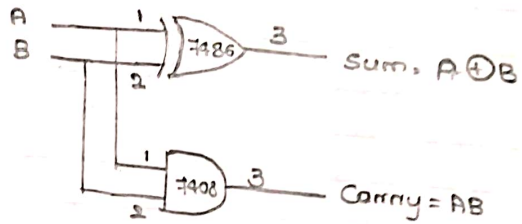
$Sum = \overline{A}B + A\overline{B}$

Carry



$Carry = AB$

Circuit diagram



Date: 30/8/24

Exp. No. 02 [A]

Exp. Title: Design and Implement Half Adder and Full Adder using Basic Gates

Page No. 04

Aim: Design and Implement Half Adder and Full Adder using Basic Gates.

Apparatus Required

Sl.No.	Component	Specification
1	AND GATE	IC 7408
2	OR GATE	IC 7432
3	NOT GATE	IC 7404
4	IC TRAINER KIT	
5	PATCH CORDS	

Theory

Half Adder A half adder has two inputs, for the two bits to be added and two outputs one from the Sum 'S' and other from the Carry 'C' into the higher adder position. Above circuit is called as a Carry signal from the addition of the less significant bits Sum from the XOR Gate the Carry out from the AND gate.

Full Adder A full adder is a combinational circuit that froms the arithmetic sum of inputs it consists of three inputs and two outputs. A full adder is useful to add three bits at a time but a half adder cannot do so. In full adder sum output will be taken from XOR Gate, Carry output will be taken from OR Gate.

Applications

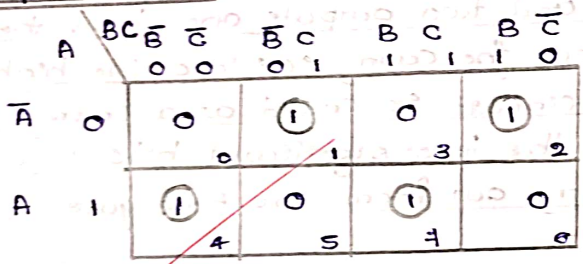
* They are used computeers, Calculatoors and various digital measuring instruments.
* They are used in digital processing devices.

Full Adder

Truth Table

Cell No.	Inputs			Outputs	
	A	B	C	Sum	Carry
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

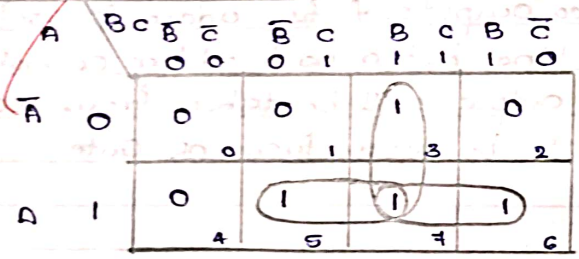
K-Map for Sum:



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

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

K-Map for Carry:



$Carry = BC + AC + AB$

Procedure

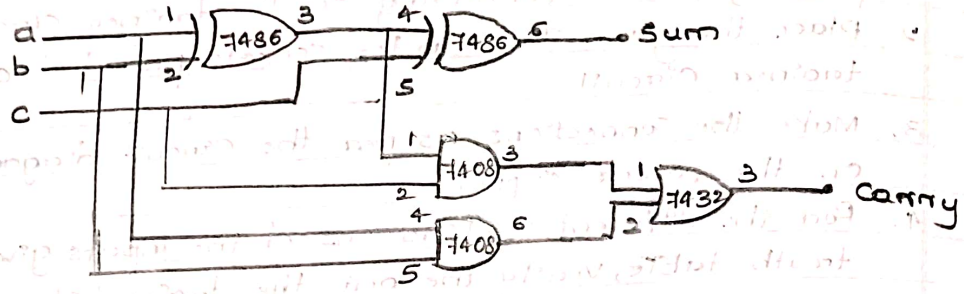
1. Make sure that the given components are working properly before connecting to the trainer circuit.
2. Place the given IC's in the sockets provided on the trainer circuit.
3. Make the connections as per the circuit diagram. Now on the power supply.
4. For the different combinations of the inputs given in the truth table, verify the out the trainerkit

Switch off the power supply and remove the connections.

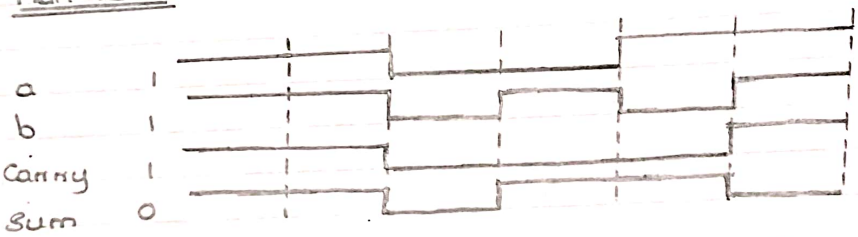
Result

Half adder and full adder truth tables are verified.

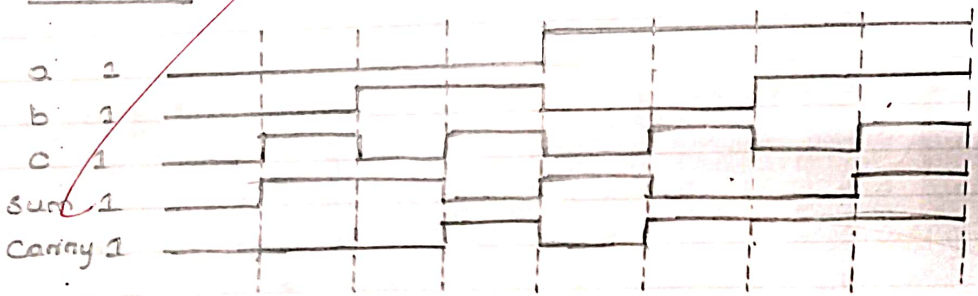
Circuit diagram for Full adder



Half Adder



Full Adder



Date :

Exp. No 02 [B]

Exp. Title Simulate and Verify the working of above expressions using VHDL.

Page No. 06

Half Adder

```
entity half_Adder is
    Port (a, b: in STD_LOGIC; Carry, Sum: out STD_LOGIC);
end half_Adder;
architecture Behavioural of half_Adder is begin
    Sum <= (not (a) and b) or (a and (not(b)));
    Carry <= a and b;
end Behavioural;
```

Full Adder

```
entity FullAdder is
    Port (a, b, c : in STD_LOGIC ; Sum, Carry: out STD_LOGIC);
end Full Adder;
architecture Behavioural of Full Adder is begin
    Sum <= (not(C) and (a xor b)) or (C and (a xnor b));
    Carry <= (a and b) or (a and c) or (b and c);
end Behavioral;
```

Result

Simulation and verification of working of above expression using VHDL is verified.

Truth Table

	A	B	C	D	y	Implementation Table		
0	0	0	0	0	1	1	D_0	4
	0	0	0	1	1			
1	0	0	1	0	0	0	D_1	3
	0	0	1	1	0			
2	0	1	0	0	1	$\overline{D}$	D_2	2
	0	1	0	1	0			
3	0	1	1	0	1	$\overline{D}$	D_3	1
	0	1	1	1	0			
4	1	0	0	0	0	D	D_4	15
	1	0	0	1	1			
5	1	0	1	0	1	$\overline{D}$	D_5	14
	1	0	1	1	0			
6	1	1	0	0	1	$\overline{D}$	D_6	13
	1	1	0	1	0			
7	1	1	1	0	1	$\overline{D}$	D_7	12
	1	1	1	1	0			

Date: 06/09/24 | Exp. Title: A 4-variable logic expression, simplify it using Entered Variable Map and realize the simplified logic exp using 8:1 MUX. | Page No. 07

Exp. No. 03 [A]

Aim: Given a 4-variable logic expression, simplify it using Entered Variable Map and realize the simplified logic expression using 8:1 multiplexer.

Apparatus Required

Sl.No	Component	Specification
1	8:1 MUX	IC 74151
2	NOT GATE	IC 7404
3	IC TRAINER KIT	-
4	PATCH CORDS	-

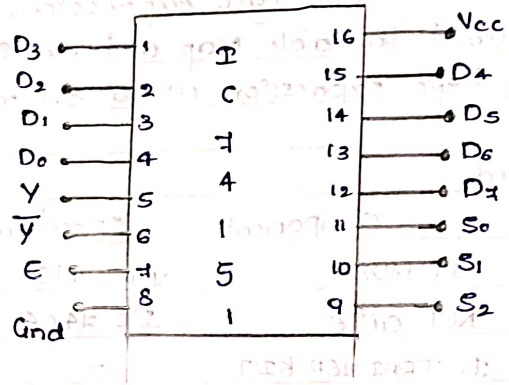
Theory

Entered Variable Map K-Map is the best manual technique to solve Boolean equations, but it becomes difficult to manage when number of variables exceed 5 or 6. So, a technique called variable Entrant Map (VEM) is used to increase the effective size of k-map. It allows a smaller map to handle large number of variables. This is done by writing output in terms of input.

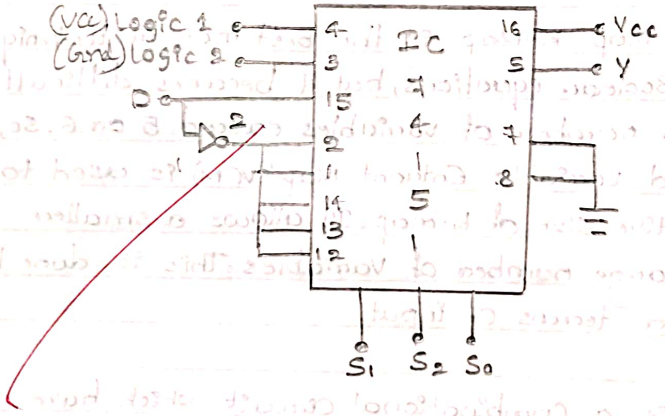
Multiplexers It is a combinational circuit which have many data inputs and single output depending on control or select inputs. For N inputs lines, log n (base 2) selection lines, or we can say that for 2n input lines, n selection lines are required. Multiplexers are also known as "Data n selector, Parallel to serial Convertor, many to one circuit, universal logic circuit". Multiplexers are mainly used to increase amount of the data that can be sent over the network within certain amount of time and bandwidth.

4-variable function using IC 74151

8:1 Multiplexer using IC 74151



$f(a,b,c,d) = m(0,1,4,6,9,10,12,14)$



Date : Exp. No.	Exp. Title	Page No. 08

Application

Multiplexers

* Communication system
* Telephone Network
* Computer Memory

Entered Variable Map

* It is commonly used to solve problems involving multiplexers.

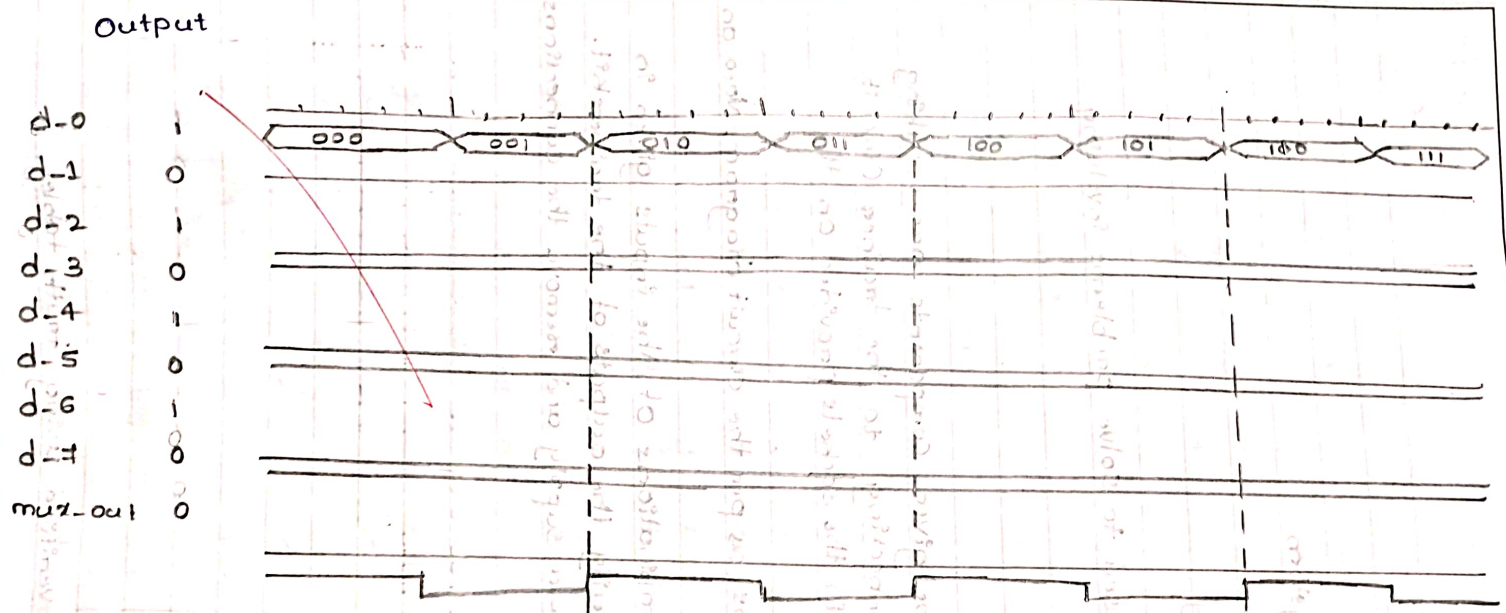
Procedure

1. Make sure that the given components are working properly before connecting to the trainer Circuit.
2. Place the given IC's in the sockets provided on the trainer circuit.
3. Make the connections as per the circuit diagram. Now on the power supply.
4. For the different combinations of the inputs given in the truth table, Verify the outputs of the trainer kit. Switch off the power supply and remove the connections

Result

Given function is verified using truth table.

Output


d-0 1
d-1 0
d-2 1
d-3 0
d-4 1
d-5 0
d-6 1
d-7 0
mux_out 0
000
001
010
011
100
101
110
111

Date :

Exp. No. 03 [B]

Exp. Title Simulate and Verify the working of 8:1 multiplexer using VHDL.

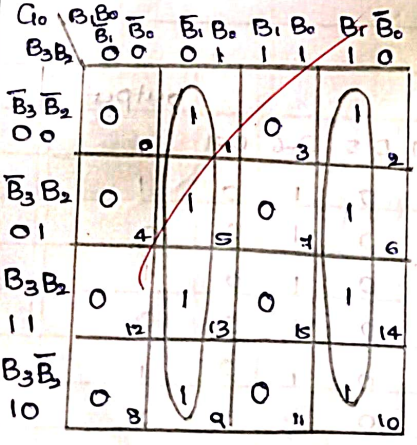

```
entity Multiplexer is
  Port(Sel: in STD_LOGIC_VECTOR (2 downto 0));
  d_0,d_1,d_2,d_3,d_4,d_5,d_6,d_7 : in STD_LOGIC;
  mux_out:out STD_LOGIC);
end Multiplexer;
architecture Behavioral of Multiplexer is begin
  Process(sel,d_0,d_1,d_2,d_3,d_4,d_5,d_6,d_7)
  begin
    Case sel is
      when "000" => mux_out<=d_0;
      when "001" => mux_out<=d_1;
      when "010" => mux_out<=d_2;
      when "011" => mux_out<=d_3;
      when "100" => mux_out<=d_4;
      when "101" => mux_out<=d_5;
      when "110" => mux_out<=d_6;
      when "111" => mux_out<=d_7;
      when others=>null;
    end Case;
  end process;
  end Behavioral;
```

Select lines			Inputs								output
A	B	C	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	
0	0	0	1	0	1	0	1	0	1	0	1
0	0	1	1	0	1	0	1	0	1	0	0
0	1	0	1	0	1	0	1	0	1	0	1
0	1	1	1	0	1	0	1	0	1	0	0
1	0	0	1	0	1	0	1	0	1	0	1
1	0	1	1	0	1	0	1	0	1	0	0
1	1	0	1	0	1	0	1	0	1	0	1
1	1	1	1	0	1	0	1	0	1	0	0

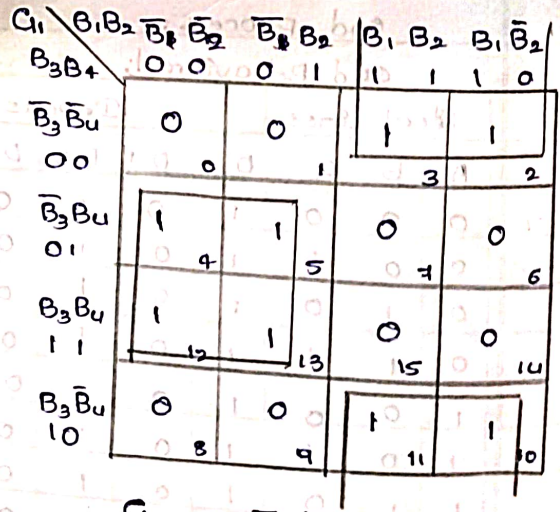
Truth Table

Cell No.	Binary code B_3	B_2	B_1	B_0	Grey code G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

K-Map



$G_0 = B_0\bar{B}_1 + \bar{B}_0B_1$



$G_1 = B_2\bar{B}_1 + \bar{B}_2B_1$

0/24 [A]	Exp. Title Design and Implement the Binary to grey Code Converter using Basic gates.	Page No. 10

Aim: Design and Implement the Binary to grey code Converter using Basic gates.

Apparatus Required

Sl. No.	Component	Specification
1	AND GATE	IC 7408
2	OR GATE	IC 7432
3	NOT GATE	IC 7404
4	IC TRAINER KIT	
5	PATCH CORDS	

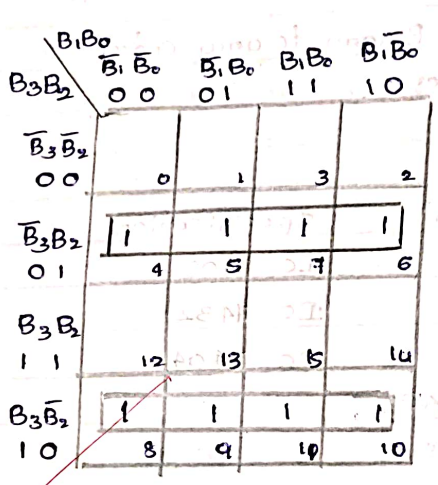
Theory : Binary Number is the default way to store numbers but in many applications, binary numbers are difficult to use and a variety of binary numbers is needed. This is where Gray codes are very useful.

Gray code has a property that two successive numbers differ in only one bit because of this property gray code does the cycling through various states with minimal effort.

Applications

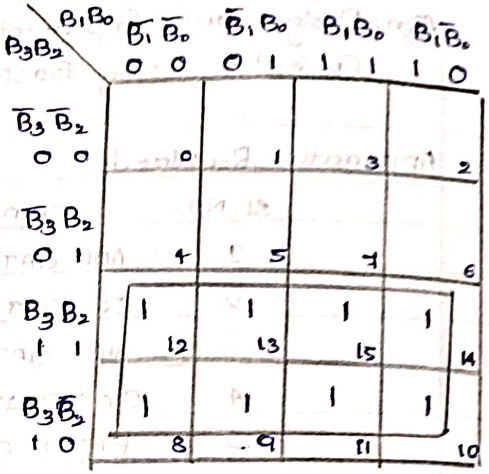
* Analog to digital Converters.
* Used for error Correction in digital Communications.
* Used in transmission of digital signals.

G_2



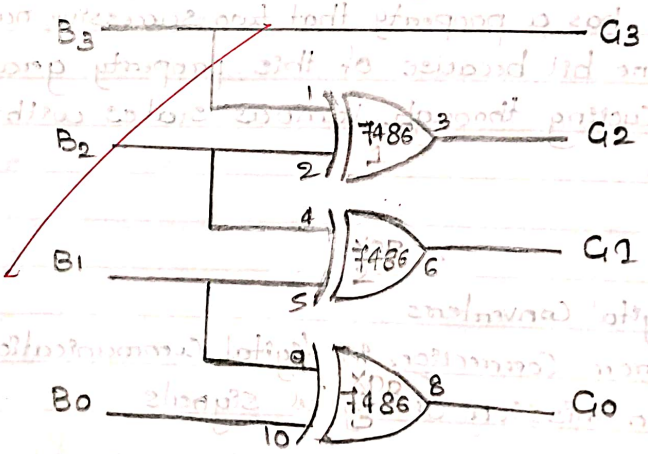
$G_2 = B_2\bar{B}_3 + \bar{B}_2 B_3$

G_3



$G_3 = B_3$

Circuit diagram



Procedure

1. Make sure that the given components are working properly before connecting to the trainer circuit.
2. Place the given IC's in the sockets provided on the trainer circuit.
3. Make the connections as per circuit diagram. Now on the power supply.
4. For the different combinations of the inputs given in the truth table, verify the outputs of trainer kit. Switch off the power supply and remove the connections.

Result

Binary to Grey code truth table is verified.

0ns
5ns
10ns
15ns
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
a[3:0]
b[3:0]

[B]	Exp. Title Simplify and verify the working of Binary to Grey Code Converter using VHDL.	Page No. 12

```
entity BinaryToGrey is
    Port (b: in std_logic_vector (3 downto 0);
    g: out std_logic_vector (3 downto 0));
end BinaryToGrey;
architecture Behavioral of BinaryToGrey is
begin
    g(3)<=b(3);
    g(2)<=b(3) XOR b(2);
    g(1)<=b(2) XOR b(1);
    g(0)<=b(1) XOR b(0);
end Behavioral;
```

Result :

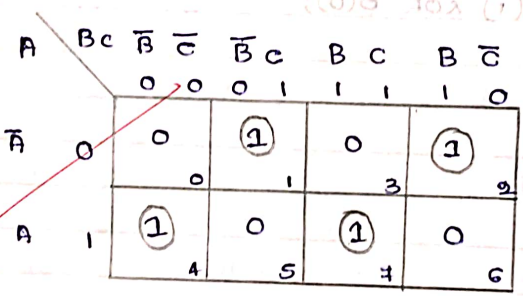
Simplication and verification of the working of Binary to Grey Code Converter using VHDL is verified.

3- Bit parity Generator

Truth Table

Cell NO.	A	B	C	Pae
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

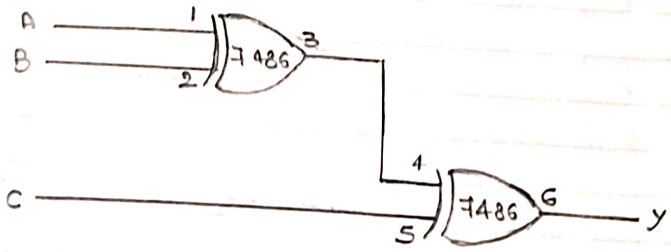
K-Map



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

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

Circuit diagram



Date: 26/10/24	Exp. Title Design and Implement the truth table of 3-Bit Parity Generator and 4-bit Checker with an even parity bit.	Page No. 13
Exp. No. 05		

Aim : Design and Implement the truth table of 3-Bit parity Generator and 4-bit checker with an Even Parity bit using basic gates.

Apparatus Required

Sl.No	Component	Specification
1	AND GATE	IC 7408
2	OR GATE	IC 7432
3	NOT GATE	IC 7404
4	IC TRAINER KIT	-
5	PATCH CORDS	-

Theory : Majority of modern communication is Digital in nature i.e., it is a combination of 1's and 0's. The digital data is transmitted either through wires (in case of wired communication) or wireless. Even in an advanced mode of communication, there will be errors while transmitting data (due to noise).

The simplest of errors is corruption of a bit i.e., a 1 may be transmitted as a 0 or vice-versa. To confirm whether the received data is the intended data or not, we should be able to detect errors at the receiver.

The parity generating technique is one of the most widely used error detection techniques for the data transmission. In digital systems, when binary data is transmitted and processed, data may be subjected to noise so that such noise can alter 0s (of data bits) to 1s and 1s to 0s.

Hence, a parity Bit is added to the word containing data in order to make number of 1s either even or odd.

4-Bit parity checker

Truth Table

Cell No.	A	B	C	D	Pch
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

The message containing the data bits along with parity bit is transmitted from transmitter to receiver.

At the receiving end, the number of 1s in the message is counted and if it doesn't match with the transmitted one, it means there is an error in the data. Thus, the parity Bit it is used to detect errors, during the transmission of binary data.

A parity Generator is a combinational logic circuit that generates the parity bit in the transmitter. On the other hand, a circuit that checks the parity in the receiver is called Parity checker. A combined circuit circuit or device of parity generators and parity checkers are commonly used in digital systems to detect the single bit errors in the transmitted data.

The sum of the data bits and parity bits can be even or odd. In even parity, the added parity bit will make the total number of 1s an even number, whereas in odd parity, the added parity bit will make the total number of 1s an odd number.

Applications

* Most widely used error detection technique for data transmission.
* Used in digital systems and many hardware applications
* Used peripheral Component Interconnect (PCI) to detect errors.

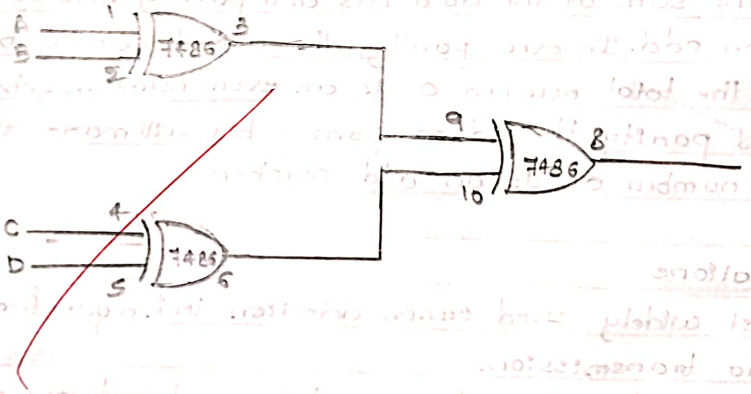
K-Map

AB \ CD	$\bar{C}\bar{D}$ (00)	$\bar{C}D$ (01)	$C D$ (11)	$C\bar{D}$ (10)
$\bar{A}\bar{B}$ (00)	0 (0)	1 (1)	0 (3)	1 (2)
$\bar{A}B$ (01)	1 (4)	0 (5)	1 (7)	0 (6)
AB (11)	0 (12)	1 (13)	0 (14)	1 (15)
$A\bar{B}$ (10)	1 (8)	0 (9)	1 (11)	0 (10)

$P_{ch} = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD$

$P_{ch} = \bar{A}\bar{B}(\bar{C}D + C\bar{D}) + \bar{A}B(\bar{C}D + \bar{C}\bar{D}) + AB(\bar{C}D + C\bar{D}) + A\bar{B}(\bar{C}\bar{D} + CD)$

Circuit diagram

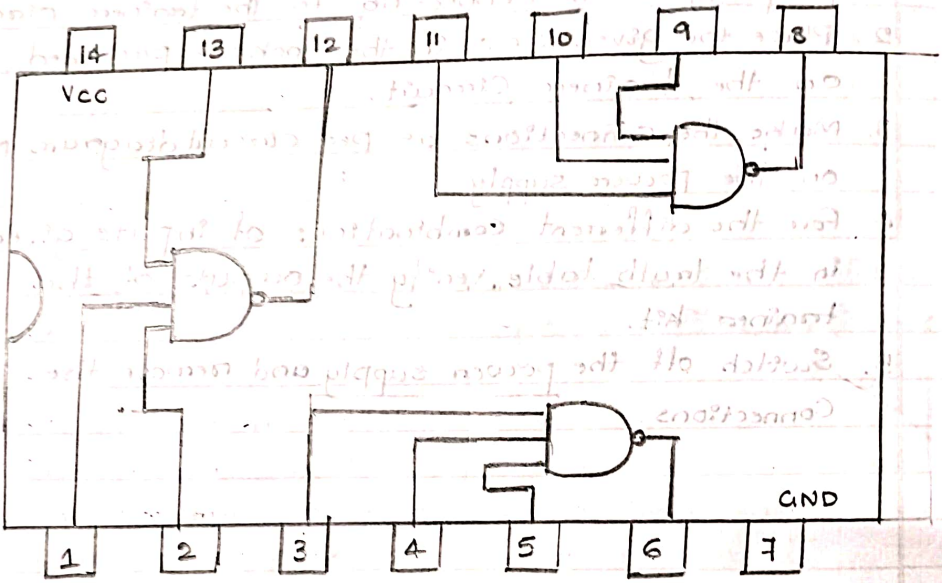


Procedure

1. Make sure that the given components are working properly before connecting to the trainer circuit.
2. Place the given IC's in the sockets provided on the trainer circuit.
3. Make the connections as per circuit diagram. Now on the power supply.
4. For the different combinations of inputs given in the truth table, verify the outputs of the trainer kit.
5. Switch off the power supply and remove the connections.

Result

Parity generator and checker truth tables are verified.



7410 Triple 3 Input NAND

Date : 09/11/24 | Exp. Title Realize a J-K Master/slave Flip Flop using NAND gates and Verify its Truth Table. | Page No. 16

Exp. No. 06 [A]

Aim : Realise a J-K Master/slave FlipFlop using NAND gates and verify its truth table.

Apparatus Required

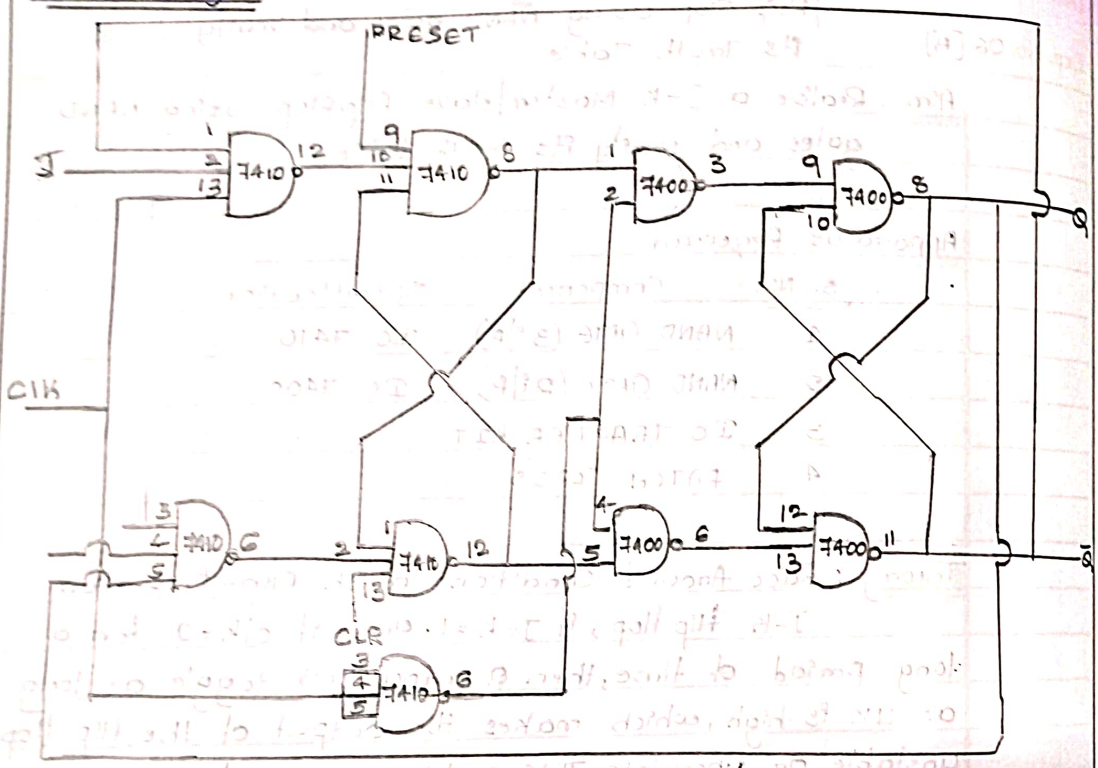
SL.NO.	Component	Specification
1	NAND GATE (3 i/p)	IC 7410
2	NAND GATE (2 i/p)	IC 7400
3	IC TRAINER KIT	
4	PATCH CORDS	

Theory : Race Around Condition in JK Flip-flop - For J-K flip flop, if J=K=1, and if clk=1 for a long period of time, then Q output will toggle as long as clk is high, which makes the output of the flip-flop unstable or uncertain. This problem is called race around condition in J-K flipflop. This problem (Race Around Condition) can be avoided by ensuring that the clock input is at logic "1" only for a every short time. This introduced the concept of Master slave JK flip flop.

The Master slave flip flop is basically a combination of two JK flip-flops connected together in a series Configuration. Out of these, one acts as the "master" and the other as a "slave". The output from the master flipflop is connected to the two inputs of the slave flip flop whose output is fed back to inputs of the masterflip flop.

In addition to these two flip-flops, the circuit also includes an inverter. The inverter is connected to clock pulse in such a way that the inverted clock pulse is given to the slave flip-flop. In other words

Circuit diagram



Truth Table

clock	J	K	Q_{n+1}	$\overline{Q_{n+1}}$	Comments
0	X	X	Q_n	$\overline{Q_n}$	No change
Pulse	0	0	Q_n	$\overline{Q_n}$	No change
Pulse	0	↑	0	1	Reset
Pulse	1	0	1	0	Set
Pulse	1	1	$\overline{Q_n}$	Q_n	Toggle

If CP=0 for a master flip-flop, then CP=1 for a slave flip-flop and if CP=1 for master flip flop then it becomes 0 for slave flip flop.

Applications

* Registers.
* Counters.
* Event detectors.

Procedure

1. Make sure that the given components are working properly before connecting to the trainer circuit.
2. Place the given IC's in the sockets provided on the trainer circuit.
3. Make the connections as per the circuit diagram. Now on the power supply.
4. For the different combinations of the inputs given in the truth table, verify the outputs of the trainer kit.
5. Switch of the power supply and remove the connections.

Result

JK master slave flip flop truth table is verified.

ame	value	kind	mode
clk	1	signal	In
d	1	signal	In
q	1	signal	out

Date :	Exp. Title Simulate and verify the working of D Flip Flop with positive edge triggering using VHDL.	Page No. 18
Exp. No. 06 [B]		

```
entity D_FlipFlop is
    Port (clk,d: in STD_LOGIC;
          q :out STD_LOGIC);
    end D_FlipFlop;
architecture Behavioral of D_FlipFlop is
begin
process(clk)
begin
    if (rising_edge(clk)) then
    q<=d;
    end if;
end process;
end Behavioral;
```

Result

Simulation and verification of the working of D FlipFlop with positive edge triggering using VHDL is verified.

Excitation Table

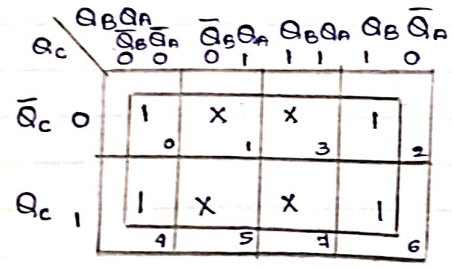
Present state (Q_n)	Next State (Q_{n+1})	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0
X	X	X	X

State Table

Present State			Next state			Flip Flop Inputs					
Q_C	Q_B	Q_A	Q_{C+1}	Q_{B+1}	Q_{A+1}	J_C	K_C	J_B	K_B	J_A	K_A
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	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	0	0	X	1	X	1	X	1

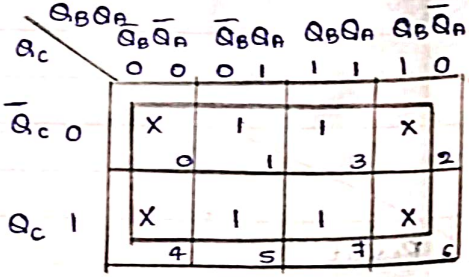
K-Map

J_A



$J_A = 1$

K_A



$K_A = 1$

Date : 16/11/24 | Exp. Title Design and implement a MOD-n(<8) synchronous UP Counter using J-K Flip Flop. | Page No. 19

Exp. No. 07 [A]

Aim : Design and implement a MOD-n(<8) synchronous Up Counter using J-K Flip Flop.

Apparatus Required

Sl. No.	Component	Specification
1	J-K Flip Flop	IC 7476
4	IC TRAINER KIT	-
5	PATCH CORDS	-

Theory The Counters which use clock signal to change their transition are Called "synchronous Counters". This means the synchronous Counters depends on their Clock input to change state values. In synchronous Counters, all flip flops are Connected to the same clock signal and all flip flops will trigger at the same time.

The result of this synchronization is that all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

Applications

* Alaram clock, set AC Timer, Set time in Camera to take the picture, flashing light indicator in automobiles, Car parking control etc.
* Mostly used in digital clocks and multiplexing Circuits.
* It is also used in digital to analog Converters.

J_B

$\bar{Q}_A\bar{Q}_B$	$\bar{Q}_A Q_B$	$Q_A Q_B$	$Q_A\bar{Q}_B$
0 0	0 1	1 1	1 0
$\bar{Q}_C$ 0	0 (0)	1 (1)	X (3)	X (2)
Q_C 1	0 (4)	1 (5)	X (7)	X (6)

$J_B = Q_A$

K_A

$\bar{Q}_A\bar{Q}_B$	$\bar{Q}_A Q_B$	$Q_A Q_B$	$Q_A\bar{Q}_B$
0 0	0 1	1 1	1 0
$\bar{Q}_C$ 0	X (0)	X (1)	1 (3)	0 (2)
Q_C 1	X (4)	X (5)	1 (7)	0 (6)

$K_B = Q_A$

J_C

$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_B Q_A$	$Q_B Q_A$	$Q_B\bar{Q}_A$
0 0	0 1	1 1	1 0
0	0 (0)	0 (1)	1 (3)	0 (2)
$\bar{Q}_C$	X (4)	X (5)	X (7)	X (6)

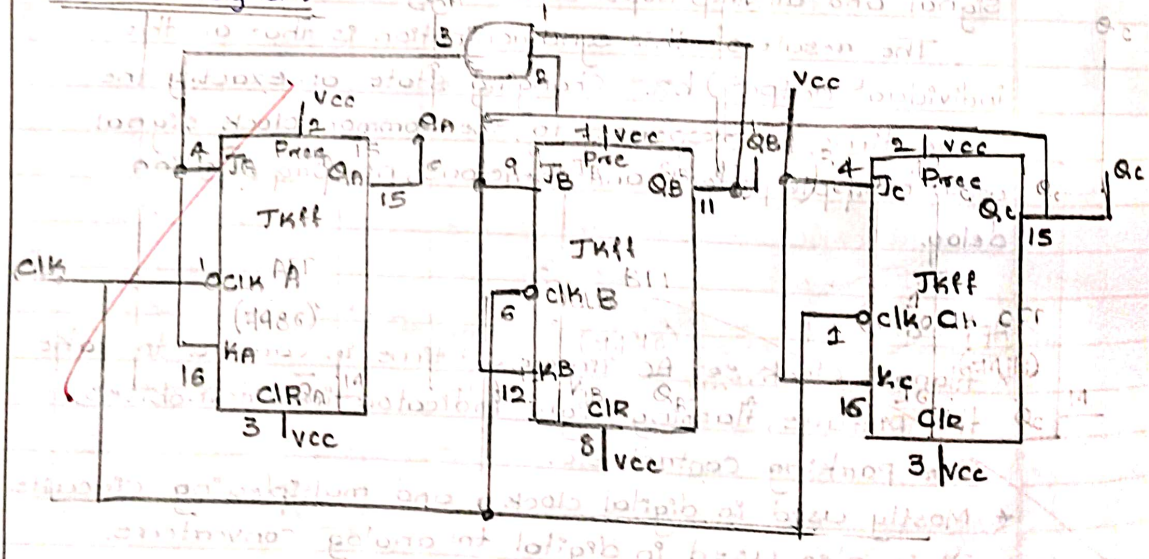
$J_C = Q_B Q_A$

K_C

$\bar{Q}_B\bar{Q}_A$	$\bar{Q}_B Q_A$	$Q_B Q_A$	$Q_B\bar{Q}_A$
0 0	0 1	1 1	1 0
Q_C 0	X (0)	X (1)	X (3)	X (2)
Q_C 1	0 (4)	0 (5)	1 (7)	0 (6)

$K_C = Q_B Q_A$

Circuit diagram

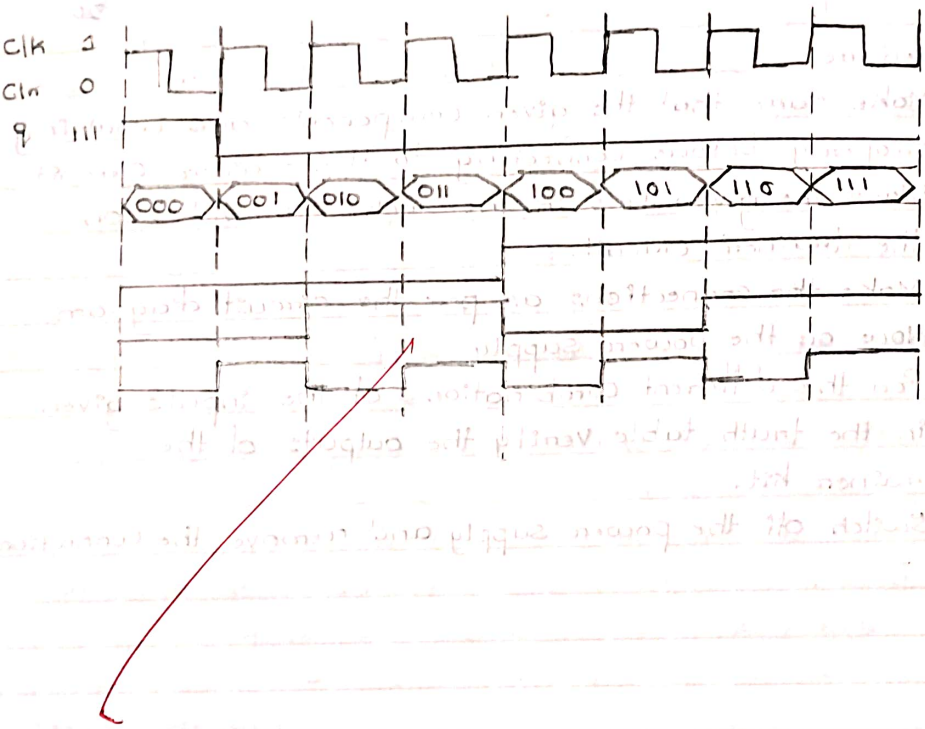


Procedure

1. Make sure that the given components are working properly before connecting to the trainer circuit.
2. Place the given IC's in the sockets provided on the trainer circuit.
3. Make the connections as per the circuit diagram. Now on the power supply.
4. For the different combinations of the inputs given in the truth table, verify the outputs of the trainer kit.
5. Switch off the power supply and remove the connections

Result

Synchronous counter truth table is verified.


Clk
Clr
Q
000
001
010
011
100
101
110
111

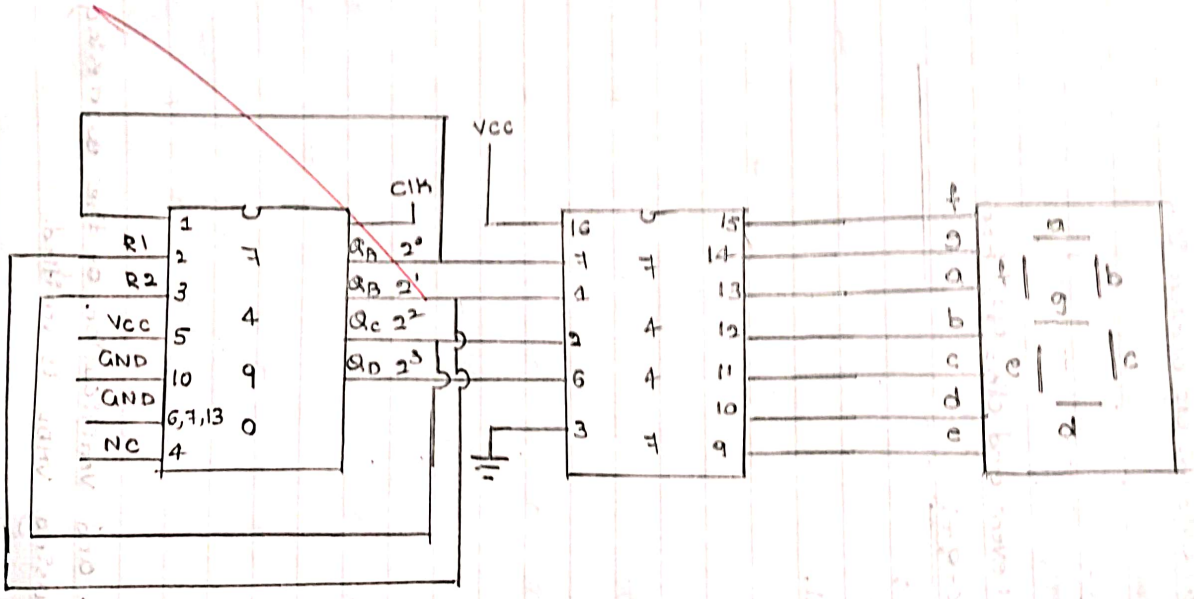
Date :	Exp. Title Simulate and Verify the working of mod-8 up Counter using VHDL.	Page No. 21
Exp. No. 07 [B]		

```
entity upCounter is
    Port (clk : in STD_LOGIC;
          q : buffer STD_LOGIC_VECTOR (2 downto 0) := "000");
end up counter;
architecture Behavioral upcounter is
begin
Process (clk)
begin
    if (clk'event and clk='1') then
        q <= q+1;
    end if;
end process;
end Behavioral;
```

Result

Simulation and verification of the working of mod-8 up Counter using VHDL is verified.

Mod-6


Vcc
CLK
R1
R2
Vcc
GND
GND
NC
1
2
3
5
10
6,7,13
4
14
7
0
9
4
Q_A 2^0
Q_B 2^1
Q_C 2^2
Q_D 2^3
16
7
1
2
6
3
15
14
13
12
11
10
9
a
b
c
d
e
f
g

Date : 23/11/24

Exp. No. 08 [A]

Exp. Title Design and Implement an Asynchronous Counter using Decade Counter IC to countup from 0 to n (n<=9) & Demonstrate on 7-Segment display

Aim : Design and Implement an Asynchronous Counter using Decade Counter IC to Countup from 0 to n(n<=9) and Demonstrate on 7-Segment Display (using IC7447).

Apparatus Required

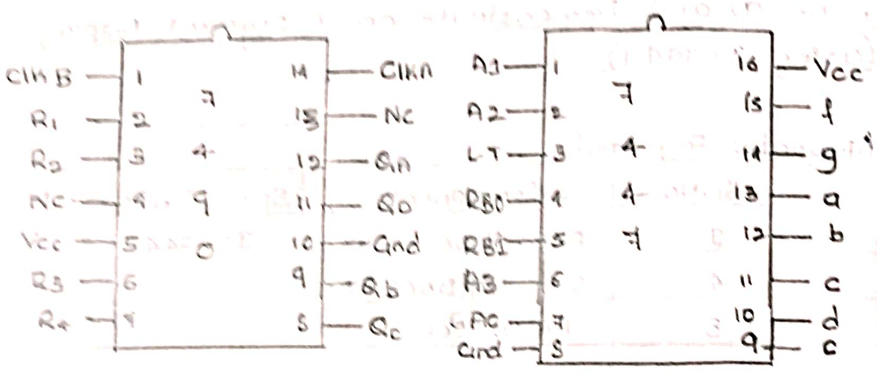
Sl.No	Component	Specification
1	BCD-7 SEGMENT	IC 7447
4	IC TRADNER KIT	
5	PATCH CORDS	

Theory : Asynchronous Stands for the absence of Synchronization. Something that is not existing or occurring at the same time. In asynchronous ripple Counter output of the first flip-flop is provided as the clock to the second flip-flop i.e., flip flop (FF) are not clocked simultaneously. Circuit is simpler, but speed is slow.

An Asynchronous counter can Count using Asynchronous clock input. Counters can be easily made using flipflops. As the Count depends on the clocks signal, in case of an Asynchronous Counter, Changing State bits are provided as the clock signal to the subsequent flip-flops. Those flip-flops are serially connected together, and the clock pulse ripples through the counter. Due to the ripple Counter. An Asynchronous Counter Can Count $2^n - 1$ possible Counting States.

Truth Table

An configuration



Truth Table

Decimal No.	A	B	C	D	E	F	G	Seven Segment Display
0	1	1	1	1	1	1	0	
1	0	1	1	0	0	0	0	
2	1	1	0	1	1	0	1	
3	1	1	1	1	0	0	1	
4	0	1	1	0	0	1	1	
5	1	0	1	1	0	1	1	

Applications

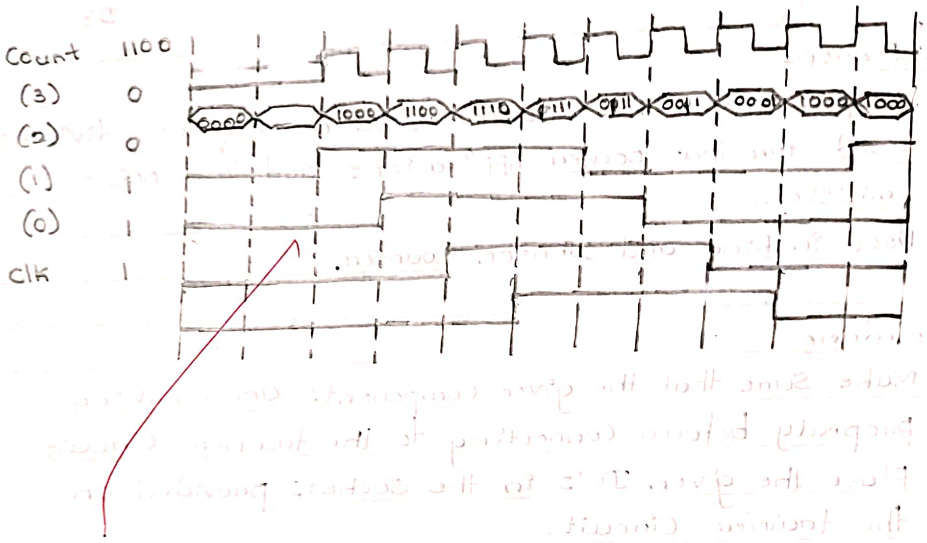
* Asynchronous Counters are used as frequency dividers.
* Used for low power Applications and low noise emission.
* Used in Ring and Johnson Counters.

Procedure

1. Make sure that the given Components are working properly before Connecting to the trainer circuit.
2. Place the given Ic's in the sockets provided on the trainer circuit.
3. Make the Connections as per the circuit diagram. Now on the power supply.
4. For the different combinations of the inputs given in the truth table, verify the outputs of the trainer kit.
5. Switch off the power supply and remove the Connections.

Result

Asynchronous Counter truth table is verified.


Count 1100
(3) 0
(2) 0
(1) 1
(0) 1
CLK 1
0000
1000
1100
1110
1111
0111
0011
0001
0000
1000
1100

Date :

Exp. No. 08 [B]

Exp. Title Simulate and Verify the working of Switched tail Counter using VHDL.


```
entity JohnsonCounter is
    Port (clk : in STD_LOGIC;
          Count : buffer STD_LOGIC_VECTOR (0 to 3) := "0000");
end JohnsonCounter;
architecture Behavioral of JohnsonCounter
is begin
Process (clk)
begin
if (clk'event and clk='1') then
    Count(0) <= not (Count(3));
      for i in 0 to 2 loop
          Count(i+1) <= Count(i);
      end loop;
end if;
end Process;
end Behavioral;
```

Result

Simulation and Verification of the working of Switched tail Counter using VHDL is verified.