CMR
INSTITUTE OF
TECHNOLOGY

USN | | | | | | | | | | |

Improvement test

Sub:	**DIGITAL SYSTEM DESIGN**							Code:	15EE35
Date:	17/ 11/ 2017	Duration:	90 mins	Max Marks:	50	Sem:	3RD	Branch:	EEE

Answer Any FIVE FULL Questions

		Marks	OBE	
			CO	RBT
1	Simplify the expression using Quine- McCluskey method(i) F(A,B,C,D)= ∑m(0,1,2,6,7,9,10,12) + ∑d(3,5)	10	CO1	L3
2	Simplify f(a,b,c,d)=∑m(1,5,7,10,11)+dc(2,3,6,13) taking least significant bit as map entered variable.	10	CO1	L3
3	Design a synchronous Mod-7 counter using T flip flop	10	CO3	L3
4	Convert SR flip flop in to JK and T flip flop and represent symbolically.	10	CO4	L2
5	Explain behavioral and data flow type of description of VHDL with the example of half adder.	10	CO5	L2
6	Compare VHDL and Verilog. Explain how data types are classified in Verilog with example	10	CO5	L2

1.

Minterm	Binary represenation	Minterm	Binary representation
m_0	0000	m_0✓	0000
m_1	0001	m_1✓	0001
m_2	0010	m_2✓	0010
m_6	0110	dm_3✓	0011
m_7	0111	dm_5✓	0101
m_9	1001	m_6✓	0110
m_{10}	1010	m_9✓	1001
m_{12}	1100	m_{10}✓	1010
dm_3	0011	m_{12}✓	1100
dm_5	0101	m_7✓	0111

Minterm	Binary representation	Minterm	Binary representation
0, 1✓	000-	0, 1, 2, 3	00--
0, 2✓	00-0	1, 3, 5, 7	0--1
1, 3✓	00-1	2, 3, 6, 7	0-1-

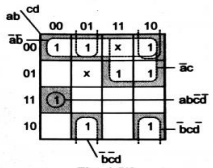


Fig. 1.5.2

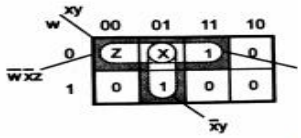
f (a, b, c, d) = $\bar{a}\bar{b} + \bar{a}\,c + \bar{b}\,\bar{c}\,d + \bar{b}\,c\,\bar{d} + a\,b\,\bar{c}\,\bar{d}$

	Minterms	Binary representation
$\bar{a}\,\bar{b}$	0, 1, 2, 3	00--
$\bar{a}\,d$	1, 3, 5, 7	0--1
$\bar{a}\,c$	2, 3, 6, 7	0-1-
$\bar{b}\,\bar{c}\,d$	1, 9	-001
$\bar{b}\,c\,\bar{d}$	2, 10	-010
$a\,b\,\bar{c}\,\bar{d}$	12	1100

Prime implicants	m_0	m_1	m_2	dm_3	dm_5	m_6	m_7	m_9	m_{10}	m_{12}
$\bar{a}\,\bar{b}$ 0, 1, 2, 3	⊙	⊙	⊙	⊙						
$\bar{a}\,d$ 1, 3, 5, 7		•		•	•		•			
$\bar{a}\,c$ 2, 3, 6, 7			⊙	⊙		⊙	⊙			
$\bar{b}\,\bar{c}\,d$ 1, 9		⊙						⊙		
$\bar{b}\,c\,\bar{d}$ 2, 10			⊙						⊙	
$a\,b\,c\,\bar{d}$ 12										⊙

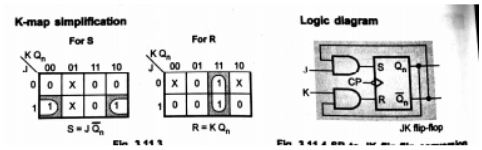
2.

Minterms in decimal	Minterms in binary				f
	w	x	y	z(MEV)	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	X
3	0	0	1	1	X
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	X
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	X
14	1	1	1	0	0
15	1	1	1	1	0



4.

Inputs		Present state	Next state	Flip-flop inputs	
J	K	Q_n	Q_{n+1}	S	R
0	0	0	0	0	X
0	0	1	1	X	0
0	1	0	0	0	X
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	1	X	0
1	1	0	1	1	0
1	1	1	0	0	1

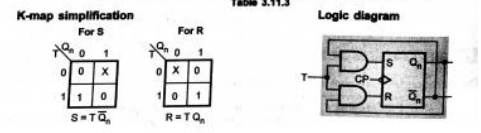


3.11.3 **SR Flip-Flop to T Flip-Flop**

The excitation table for above conversion is as shown in the Table 3.11.3.

Input	Present state	Next state	Flip-flop inputs	
T	Q_n	Q_{n+1}	S	R
0	0	0	0	X
0	1	1	X	0
1	0	1	1	0
1	1	0	0	1

Table 3.11.3



5.

Behavioral Descriptions

A behavioral description models the system as to how the outputs behave with the inputs. The definition of behavioral description is one where the architecture (VHDL) or the module (Verilog) includes the predefined word `process` (VHDL) or `always` (Verilog) or `initial` (Verilog). The description is considered pure behavioral if it does not contain any other features from other styles or descriptions (

Data-Flow Descriptions

Data flow describes how the system's signals flow from the inputs to the outputs. Usually, the description is done by writing the Boolean function of the outputs. The data-flow statements are concurrent; their execution is controlled by events.

VHDL Behavioral

```
entity half_add
    port (I1, I2   in bit; O1, O2 : out bit);
end half_add;
architecture          of half_add is
--The archi        consists of a process construct
begin
process (I1, I2)
--The above          is process statement

   begin
      O1 <= I1     I2 after 10 ns;
      O2 <= I1     I2 after 10 ns;
   end process;
   behave
```

VHDL Data-Flow Description

```
entity halfadder is
port (
a : in bit;
b : in bit;
s : out bit;
c : out bit);
end halfadder;

architecture HA_DtFl of halfadder is
--The architecture has no process, component, cmos,
--tranif0, tran, or tranif0

begin
    s <= a xor b;
    c <= a and b;
end HA_DtFl;
```

6.

- **Data Types**

  VHDL: Definitely a type-oriented language, VHDL types are built in, or the user can create and define them. User-defined types give the user a tool to write the code effectively; these types also support flexible coding. VHDL can handle objects with multidimensional array types. Another data type that VHDL supports is the physical type; the physical type supports more synthesizable or targeted design code.

  Verilog: Compared to VHDL, Verilog data types are very simple and easy to use. All types are defined by the language. There are no user-defined types. Some beginners may consider these simple data types as an advantage over VHDL. Verilog, however, cannot handle objects with multidimensional array types.

- **Ease of Learning**

  VHDL: For beginners, VHDL may seem hard to learn because of its rigid type requirements. Advanced users, however, may find these rigid type requirements easier to handle.

  Verilog: Easy to learn, Verilog users just write the module without worrying about what Library or package should be attached. Many of the statements in the language are very similar to those in C language.

- **Libraries and Packages**

  VHDL: Libraries and packages can be attached to the standard VHDL package. Packages can include procedures and functions, and the package can be made available to any module that needs to use it. Packages are used to target a certain design. For example, if the system modeled/designed includes arithmetic functions, a package can be used that includes those functions.

  Verilog: There is no concept of Libraries or packages in Verilog.

- **Operators**

  VHDL: An extensive set of operators is available in VHDL, but it does not have predefined unary operators.

  Verilog: An extensive set of operators is also available in Verilog. It also has predefined unary operators (see Section 1.4).

- **Procedures and Tasks:** Procedures (VHDL) and tasks (Verilog) are implemented to simplify the writing of HDL code for complex systems.

Data types in Verilog