CBCS Scheme

USN 1 C R 1 6 E E 0 7 4 　　　　　　　　　　　　　　　　15EE35

Third Semester B.E. Degree Examination, Dec.2017/Jan.2018

Digital System Design

Time: 3 hrs. 　　　　　　　　　　　　　　　　Max. Marks: 80

Note: Answer any FIVE full questions, choosing one full question from each module.

Module-1

1 a. With basic block diagram, explain the combinational logic circuit. **(04 Marks)**

b. Reduce the following function using K-map technique and implement using basic gates

i) $f(P, Q, R, S) = \sum m(0, 1, 4, 8, 9, 10) + d(2, 11)$

ii) $f(A, B, C, D) = \pi M(0, 2, 4, 10, 11, 14, 15)$ **(12 Marks)**

OR

2 a. Simplify using the Quine-Mcclusky minimization technique.

$Y = f(a, b, c, d) = \sum m(0, 2, 8, 10)$ **(08 Marks)**

b. Simplify the given function using MEV technique.

$f(a, b, c, d) = \sum(2, 3, 4, 5, 13, 15) + \sum d(8, 9, 10, 11)$. **(08 Marks)**

Module-2

3 a. With the aid of general structure, clearly distinguish between a decoder and encoder. **(05 Marks)**

b. Implement following multiple output function using one 74LS138 and external gates.

$F_1(A, B, C) = \sum m(1, 4, 5, 7)$

$F_2(A, B, C) = \pi M(2, 3, 6, 7)$ **(06 Marks)**

c. Draw the interfacing diagram of ten keypad interface to a digital system using decimal to BCD encoder (IC 74LS147 Decimal to BCD priority encoder). **(05 Marks)**

OR

4 a. Design a full adder by constructing the truth table and simplify the output equations. **(06 Marks)**

b. Write a truth table for two-bit magnitude comparator. Write the Karnaugh map for each output of two bit magnitude comparator and the resulting equation. **(10 Marks)**

Module-3

5 a. What is the difference between a flip-flop and a latch? With logic diagram and truth table, explain the operation of gated SR latch. **(08 Marks)**

b. Explain the operation of Master slave JK Flip-flop along with its circuit diagram. **(08 Marks)**

OR

6 a. Explain the working principle of four bit binary ripple counter, with the help of a logic diagram, timing diagram and counting sequence. **(10 Marks)**

b. With logic diagram and counting sequence explain Mod – 4 ring counters. **(06 Marks)**

Important Note : 1. On completing your answers, compulsorily draw diagonal cross lines on the remaining blank pages.
2. Any revealing of identification, appeal to evaluator and /or equations written eg, 42+8 = 50, will be treated as malpractice.

Module-4

7 a. Distinguish between Moore and Mealy model with necessary block diagrams. **(08 Marks)**

b. Give output function, transition table and state diagram by analyzing the sequential circuit shown in Fig. Q7(b). **(08 Marks)**

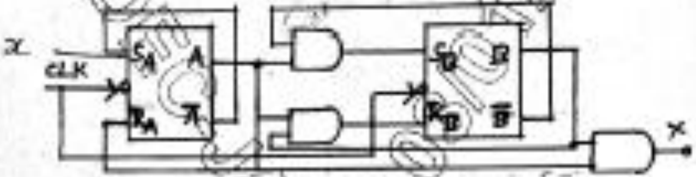


Fig. Q7(b)

OR

8 a. Write the basic recommended steps for the design of a clocked synchronous sequential circuit. **(06 Marks)**

b. Design a synchronous counter using J-K flip flops to count the sequence 0, 1, 2, 4, 5, 6, 0, 1, 2. Use state diagram and state table. **(10 Marks)**

Module-5

9 a. Explain brief history of HDL and structure of HDL module. **(06 Marks)**

b. List the classification of VHDL data types. Compare the VHDL data types and Verilog data types. **(10 Marks)**

OR

10 a. Explain signal declaration and signal assignment statements with relevant example. **(06 Marks)**

b. Write a data flow description VHDL for a system that has three 1-bit inputs a (1), a(2) and a(3) one 1-bit output b. The least significant bit is a(1) ; and b is 1, only when (a(1) a(2) a(3)) = 1, 3, 6 or 7 (all in decimal) otherwise b is 0. Derive a minimized Boolean function of the system and write the data flow description. **(10 Marks)**

* * * * *

1 a.

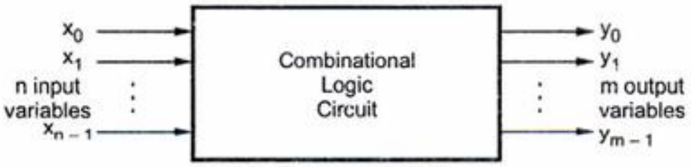
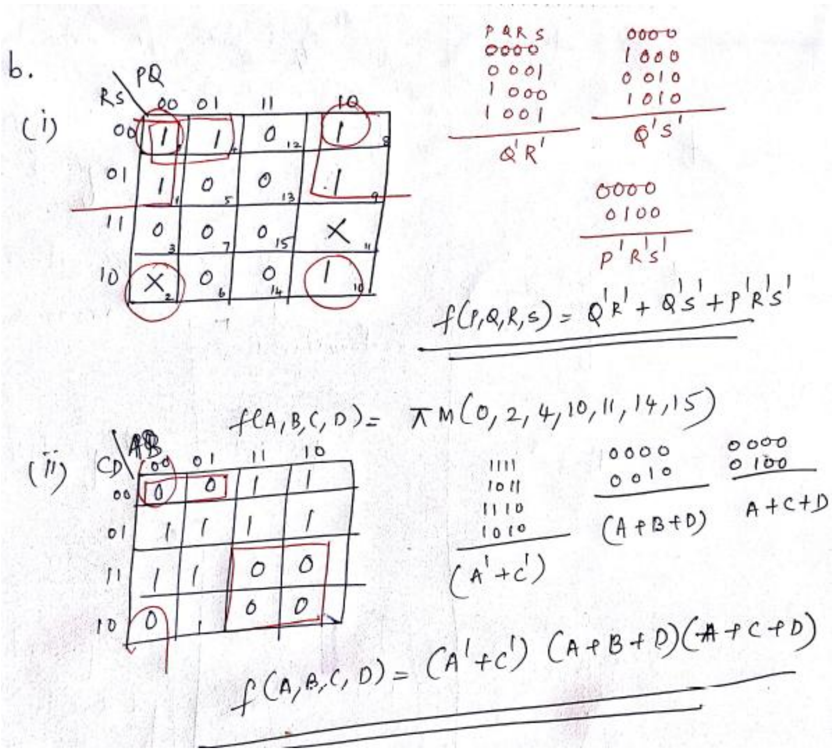


Fig. 3.1 Block diagram of a combinational circuit

In the combinational circuit, let X be the set of n input variables $\{x_0, x_1, \dots, x_{n-1}\}$, and Y be the set of m output variables $\{y_0, y_1, \dots, y_{m-1}\}$. The combinational function F, operate on the set X, to produce the output variable set, Y. The output is thus related to input as

$$Y = F(X)$$



b.

(i)

RS \ PQ	00	01	11	10
00	1_{0}	1_{4}	0_{12}	1_{8}
01	1_{1}	0_{5}	0_{13}	1_{9}
11	0_{3}	0_{7}	0_{15}	X_{11}
10	X_{2}	0_{6}	0_{14}	1_{10}

```
P Q R S
0 0 0 0
0 0 0 1
1 0 0 0
1 0 0 1
-------
 Q'R'
```

```
0 0 0 0
1 0 0 0
0 0 1 0
1 0 1 0
-------
 Q'S'
```

```
0 0 0 0
0 1 0 0
-------
 P'R'S'
```

$$f(P,Q,R,S) = Q'R' + Q'S' + P'R'S'$$

$$f(A,B,C,D) = \pi M(0,2,4,10,11,14,15)$$

(ii)

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

```
1111
1011
1110
1010
----
(A'+C')
```

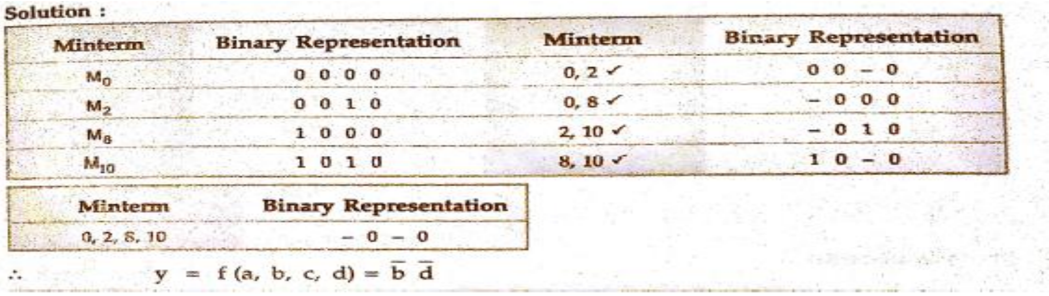
```
0000
0010
----
(A+B+D)
```

```
0000
0100
----
A+C+D
```

$$f(A,B,C,D) = (A'+C')\,(A+B+D)(A+C+D)$$

b.

2 a.



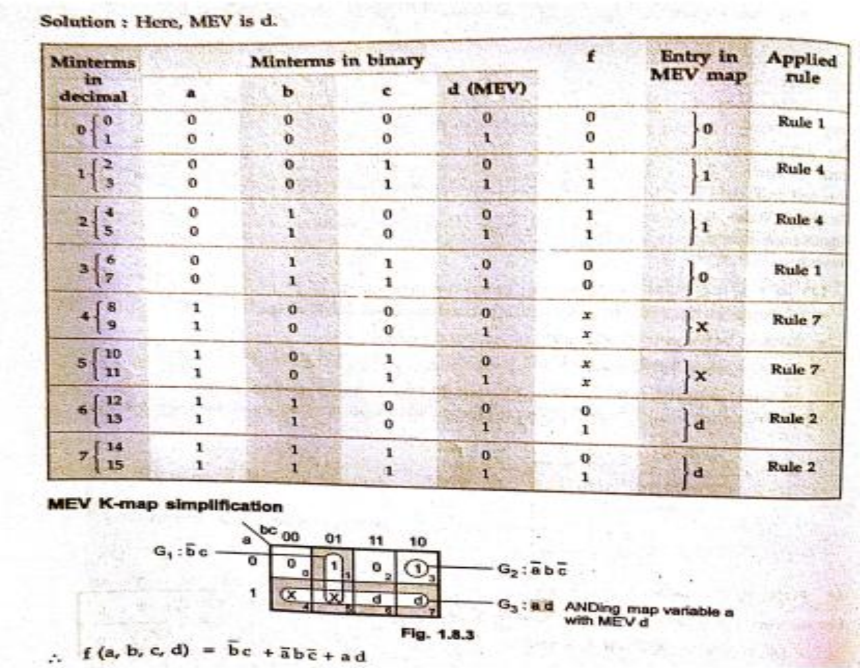
Solution :

Minterm	Binary Representation	Minterm	Binary Representation
M_0	0 0 0 0	0, 2 ✓	0 0 – 0
M_2	0 0 1 0	0, 8 ✓	– 0 0 0
M_8	1 0 0 0	2, 10 ✓	– 0 1 0
M_{10}	1 0 1 0	8, 10 ✓	1 0 – 0

Minterm	Binary Representation
0, 2, 8, 10	– 0 – 0

$\therefore \quad y = f(a, b, c, d) = \bar{b}\,\bar{d}$

b.



Solution : Here, MEV is d.

Minterms in decimal	Minterms in binary				f	Entry in MEV map	Applied rule
	a	b	c	d (MEV)			
0 {0 1	0 0	0 0	0 0	0 1	0 0	} 0	Rule 1
1 {2 3	0 0	0 0	1 1	0 1	1 1	} 1	Rule 4
2 {4 5	0 0	1 1	0 0	0 1	1 1	} 1	Rule 4
3 {6 7	0 0	1 1	1 1	0 1	0 0	} 0	Rule 1
4 {8 9	1 1	0 0	0 0	0 1	x x	} X	Rule 7
5 {10 11	1 1	0 0	1 1	0 1	x x	} X	Rule 7
6 {12 13	1 1	1 1	0 0	0 1	0 1	} d	Rule 2
7 {14 15	1 1	1 1	1 1	0 1	0 1	} d	Rule 2

MEV K-map simplification

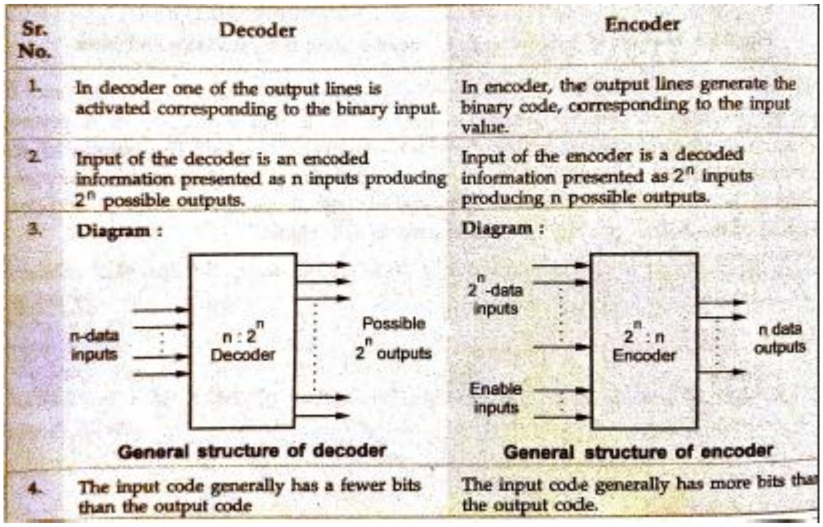
a \ bc	00	01	11	10
0	0 (0)	1 (1)	0 (2)	1 (3)
1	X (4)	X (5)	d (6)	d (7)

$G_1 : \bar{b}\,c$
$G_2 : \bar{a}\,b\,\bar{c}$
$G_3 : a\,d$ ANDing map variable a with MEV d

Fig. 1.8.3

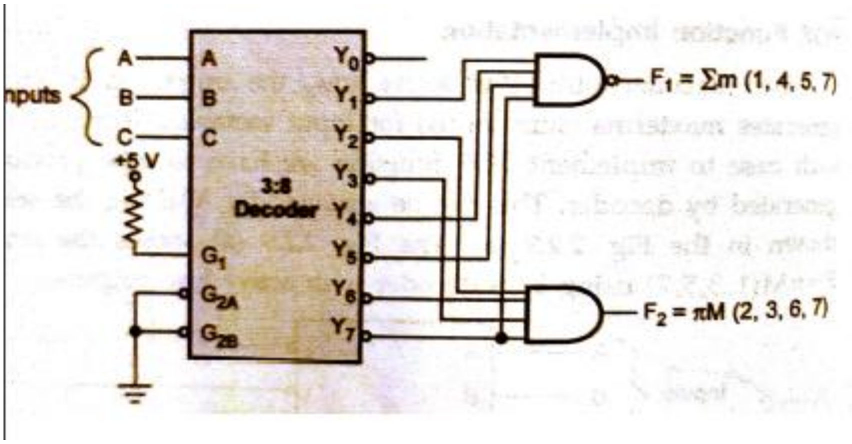
$\therefore \quad f(a, b, c, d) = \bar{b}\,c + \bar{a}\,b\,\bar{c} + a\,d$

3 a.

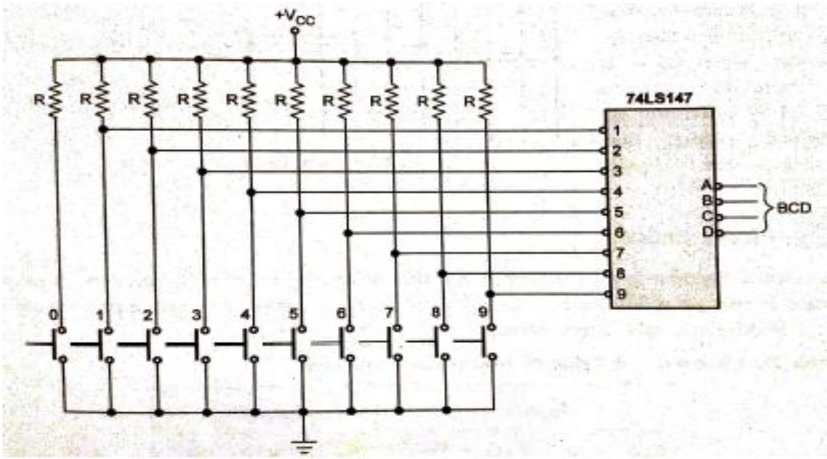


Sr. No.	Decoder	Encoder
1.	In decoder one of the output lines is activated corresponding to the binary input.	In encoder, the output lines generate the binary code, corresponding to the input value.
2.	Input of the decoder is an encoded information presented as n inputs producing 2^n possible outputs.	Input of the encoder is a decoded information presented as 2^n inputs producing n possible outputs.
3.	**Diagram :** **General structure of decoder**	**Diagram :** **General structure of encoder**
4.	The input code generally has a fewer bits than the output code	The input code generally has more bits tha the output code.

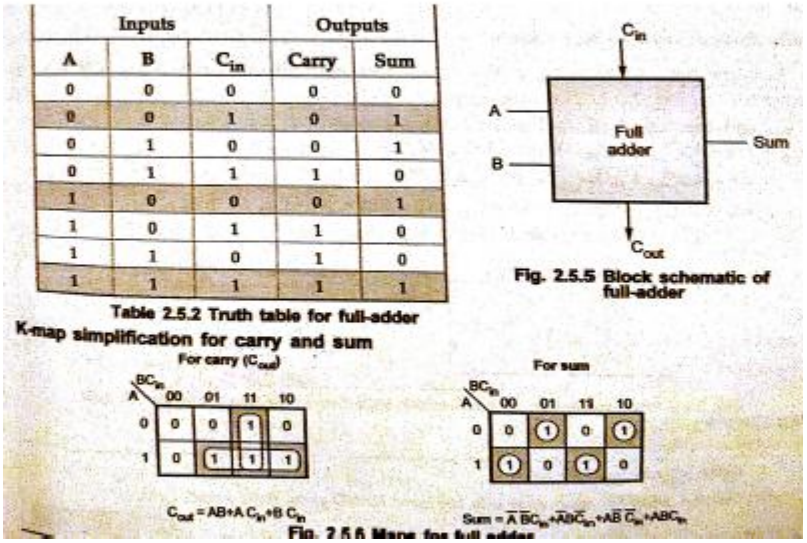
b.



c.



4 a.



Inputs			Outputs	
A	B	C_{in}	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 2.5.2 Truth table for full-adder

C$_{in}$
A
B
Full
adder
Sum
C$_{out}$

Fig. 2.5.5 Block schematic of full-adder

K-map simplification for carry and sum

For carry (C_{out})

A \ BC_{in}	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_{out} = AB + A\,C_{in} + B\,C_{in}$

For sum

A \ BC_{in}	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$Sum = \overline{A}\,\overline{B}C_{in} + \overline{A}B\overline{C}_{in} + A\overline{B}\,\overline{C}_{in} + ABC_{in}$

Fig. 2.5.6 Maps for full adder

b.

Inputs				Outputs		
A_1	A_0	B_1	B_0	A > B	A = B	A < B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

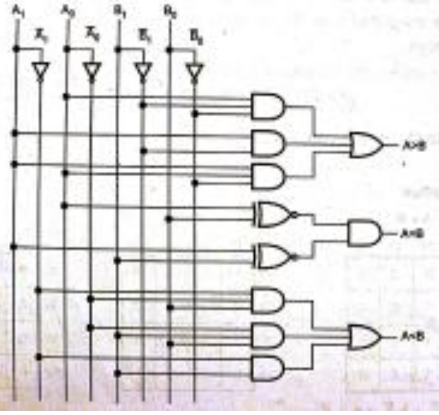
Table 2.12.1

$$(A = B) = \overline{A}_1 \overline{A}_0 \overline{B}_1\overline{B}_0 + \overline{A}_1 A_0 \overline{B}_1 B_0 + A_1 A_0 B_1 B_0 + A_1 \overline{A}_0 B_1 \overline{B}_0$$

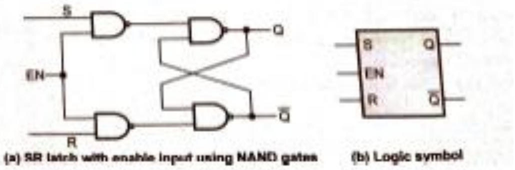
$$= \overline{A}_1 \overline{B}_1 (\overline{A}_0 \overline{B}_0 + A_0 B_0) + A_1 B_1 (A_0 B_0 + \overline{A}_0 \overline{B}_0) = (A_0 \odot B_0)(A_1 \odot B_1)$$

$$(A < B) = \overline{A}_1 \overline{A}_0 B_0 + \overline{A}_0 B_1 B_0 + \overline{A}_1 B_1$$

Logic diagram



5 a.



(a) SR latch with enable input using NAND gates (b) Logic symbol

b.

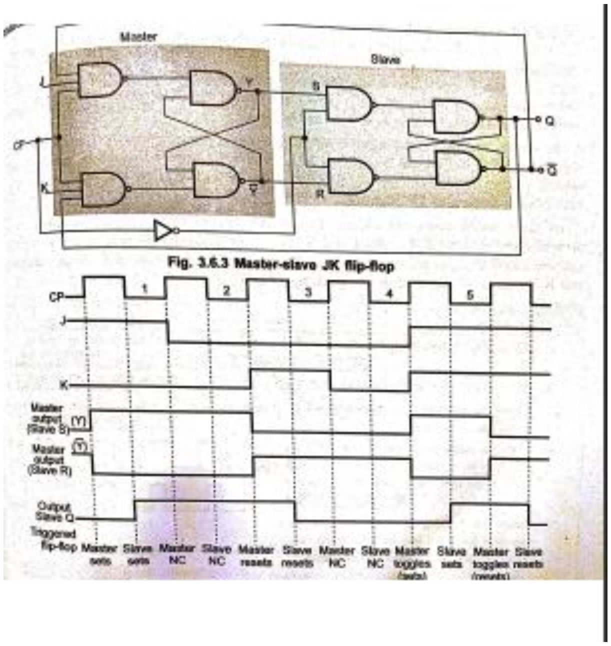


Fig. 3.6.3 Master-slave JK flip-flop

6 a.

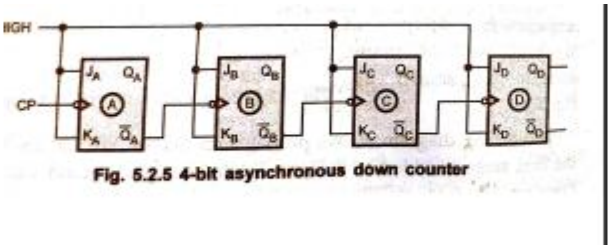


Fig. 5.2.5 4-bit asynchronous down counter

b.

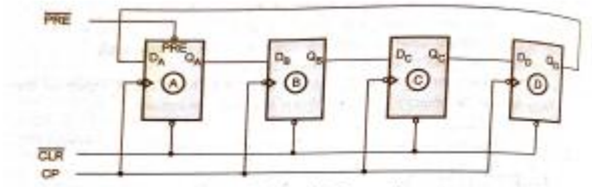
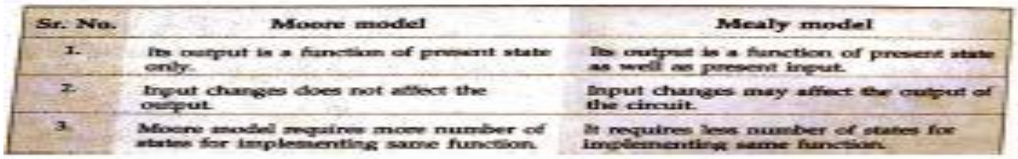


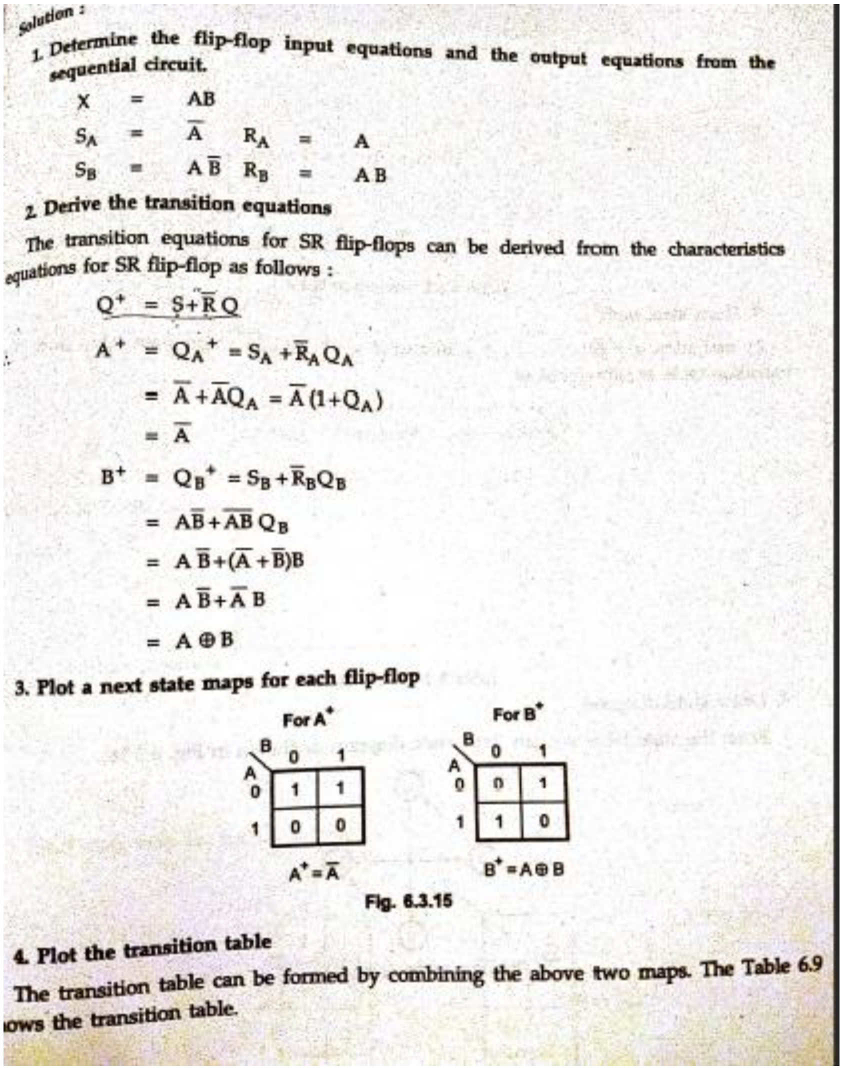
Fig. 5.4.1 Four-bit ring counter

7 a.



Sr. No.	Moore model	Mealy model
1.	Its output is a function of present state only.	Its output is a function of present state as well as present input.
2.	Input changes does not affect the output.	Input changes may affect the output of the circuit.
3.	Moore model requires more number of states for implementing same function.	It requires less number of states for implementing same function.

b.



Solution :

1. **Determine the flip-flop input equations and the output equations from the sequential circuit.**

$$X = AB$$

$$S_A = \overline{A} \quad R_A = A$$

$$S_B = A\overline{B} \quad R_B = AB$$

2. **Derive the transition equations**

The transition equations for SR flip-flops can be derived from the characteristics equations for SR flip-flop as follows :

$$Q^+ = S + \overline{R}Q$$

$$\begin{aligned} A^+ &= Q_A{}^+ = S_A + \overline{R}_A Q_A \\ &= \overline{A} + \overline{A}Q_A = \overline{A}(1+Q_A) \\ &= \overline{A} \end{aligned}$$

$$\begin{aligned} B^+ &= Q_B{}^+ = S_B + \overline{R}_B Q_B \\ &= A\overline{B} + \overline{AB}\,Q_B \\ &= A\,\overline{B} + (\overline{A} + \overline{B})B \\ &= A\,\overline{B} + \overline{A}\,B \\ &= A \oplus B \end{aligned}$$

3. **Plot a next state maps for each flip-flop**

For A^+

A \ B	0	1
0	1	1
1	0	0

$A^+ = \overline{A}$

For B^+

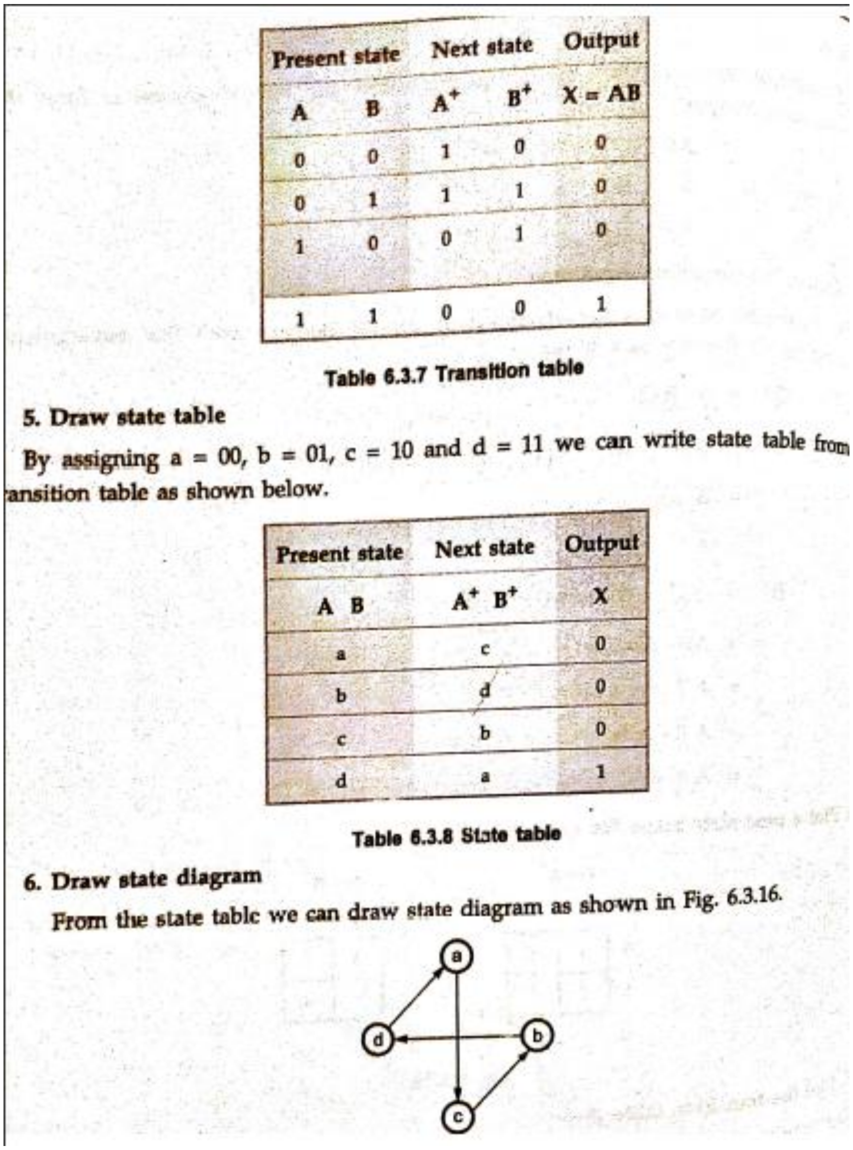
A \ B	0	1
0	0	1
1	1	0

$B^+ = A \oplus B$

Fig. 6.3.15

4. **Plot the transition table**

The transition table can be formed by combining the above two maps. The Table 6.9 ows the transition table.



Present state		Next state		Output
A	B	A^+	B^+	X = AB
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	0	1

Table 6.3.7 Transition table

5. Draw state table

By assigning a = 00, b = 01, c = 10 and d = 11 we can write state table from ansition table as shown below.

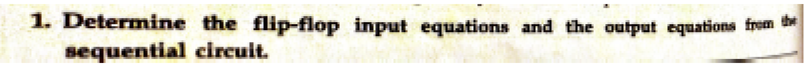
Present state	Next state	Output
A B	A^+ B^+	X
a	c	0
b	d	0
c	b	0
d	a	1

Table 6.3.8 State table

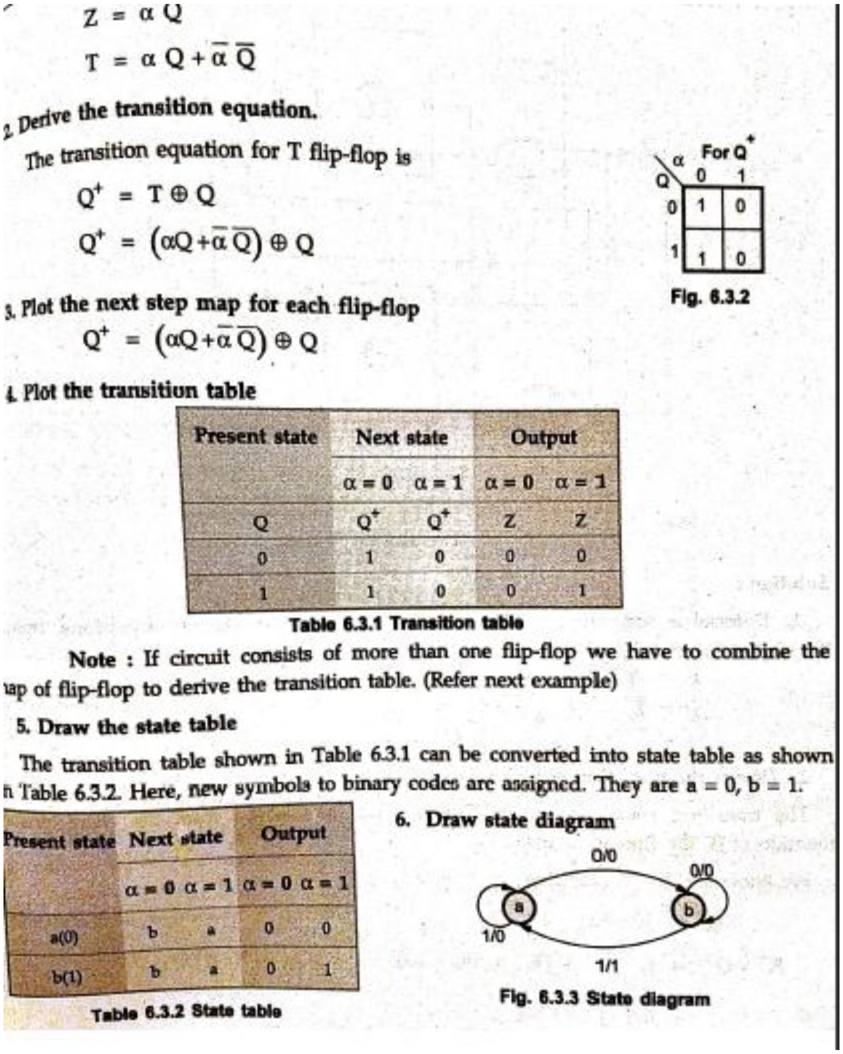
6. Draw state diagram

From the state table we can draw state diagram as shown in Fig. 6.3.16.

8 a.



1. Determine the flip-flop input equations and the output equations from the sequential circuit.


α
Q
For Q^+
0
1
0
1
1
0
1
1
0
0/0
0/0
a
b
1/0
1/1

$$Z = \alpha Q$$

$$T = \alpha Q + \bar{\alpha}\,\bar{Q}$$

2. Derive the transition equation.

The transition equation for T flip-flop is

$$Q^+ = T \oplus Q$$

$$Q^+ = \left(\alpha Q + \bar{\alpha}\,\bar{Q}\right) \oplus Q$$

Fig. 6.3.2

3. Plot the next step map for each flip-flop

$$Q^+ = \left(\alpha Q + \bar{\alpha}\,\bar{Q}\right) \oplus Q$$

4. Plot the transition table

Present state	Next state		Output	
	$\alpha = 0$	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$
Q	Q^+	Q^+	Z	Z
0	1	0	0	0
1	1	0	0	1

Table 6.3.1 Transition table

Note : If circuit consists of more than one flip-flop we have to combine the ap of flip-flop to derive the transition table. (Refer next example)

5. Draw the state table

The transition table shown in Table 6.3.1 can be converted into state table as shown n Table 6.3.2. Here, new symbols to binary codes are assigned. They are a = 0, b = 1.

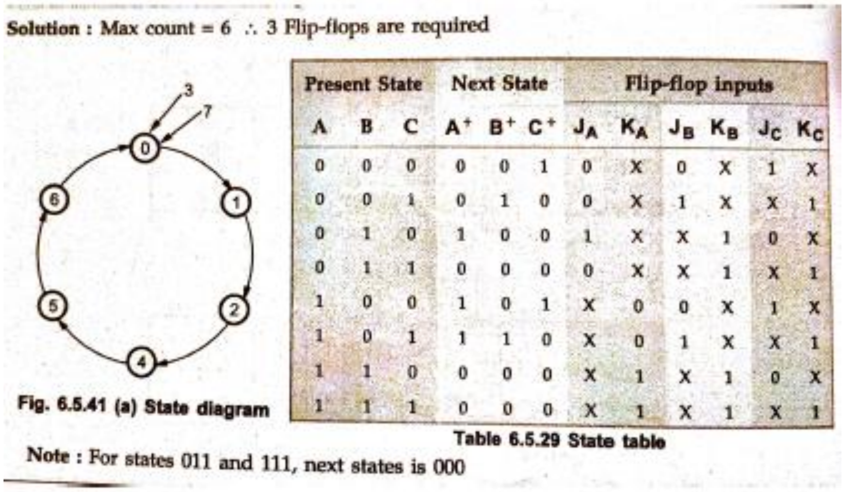
Present state	Next state		Output	
	$\alpha = 0$	$\alpha = 1$	$\alpha = 0$	$\alpha = 1$
a(0)	b	a	0	0
b(1)	b	a	0	1

Table 6.3.2 State table

6. Draw state diagram

Fig. 6.3.3 State diagram

b.



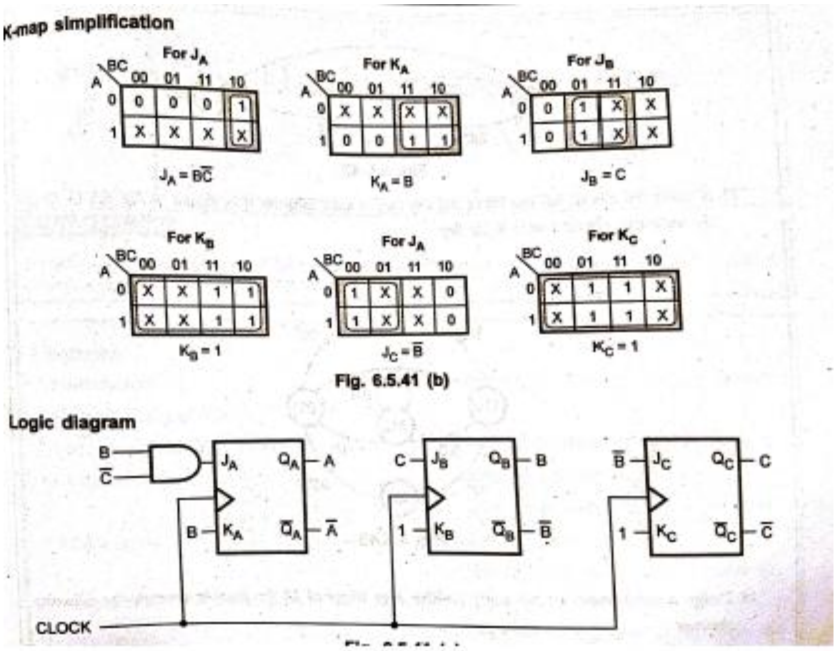
Solution : Max count = 6 ∴ 3 Flip-flops are required

Fig. 6.5.41 (a) State diagram

Present State			Next State			Flip-flop 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	0	1	0	0	X	1	X	X	1
0	1	0	1	0	0	1	X	X	1	0	X
0	1	1	0	0	0	0	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	0	0	0	X	1	X	1	0	X
1	1	1	0	0	0	X	1	X	1	X	1

Table 6.5.29 State table

Note : For states 011 and 111, next states is 000



K-map simplification

For J_A

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

$J_A = B\overline{C}$

For K_A

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

$K_A = B$

For J_B

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

$J_B = C$

For K_B

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

$K_B = 1$

For J_A

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

$J_C = \overline{B}$

For K_C

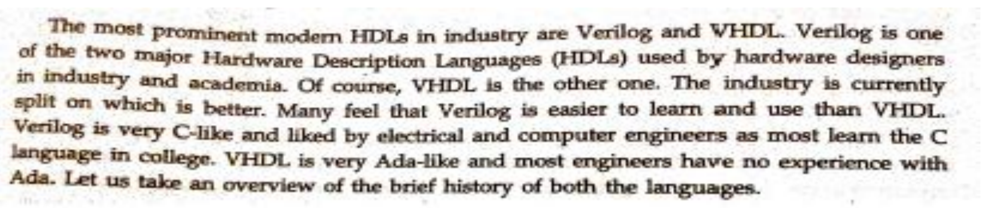
A \ BC	00	01	11	10
0	X	1	1	X
1	X	1	1	X

$K_C = 1$

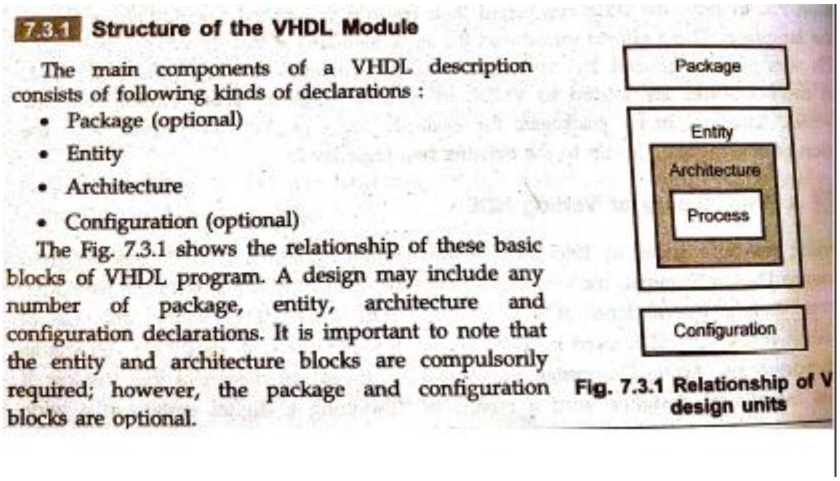
Fig. 6.5.41 (b)

Logic diagram

9 a.



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 academia. Of course, VHDL is the other one. The industry is currently split on which is better. Many feel that Verilog is easier to learn and use than VHDL. Verilog is very C-like and liked by electrical and computer engineers as most learn the C language in college. VHDL is very Ada-like and most engineers have no experience with Ada. Let us take an overview of the brief history of both the languages.



7.3.1 Structure of the VHDL Module

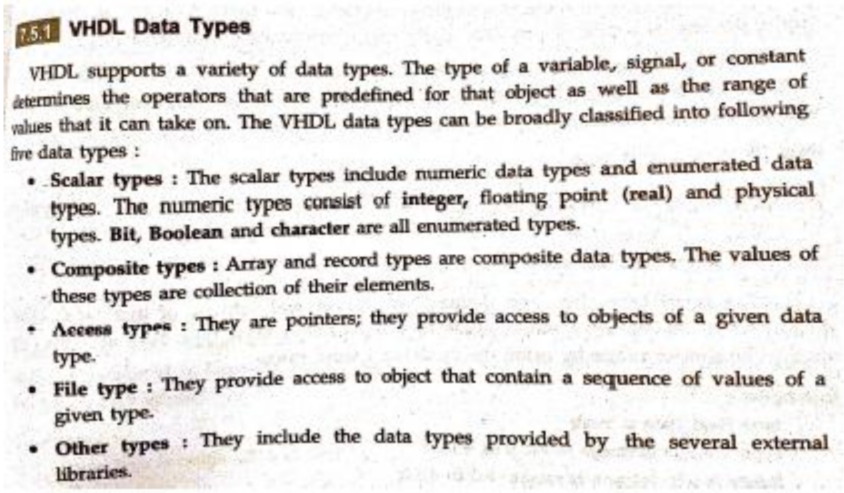
The main components of a VHDL description consists of following kinds of declarations :

- Package (optional)
- Entity
- Architecture
- Configuration (optional)

The Fig. 7.3.1 shows the relationship of these basic blocks of VHDL program. A design may include any number of package, entity, architecture and configuration declarations. It is important to note that the entity and architecture blocks are compulsorily required; however, the package and configuration blocks are optional.

Fig. 7.3.1 Relationship of V design units

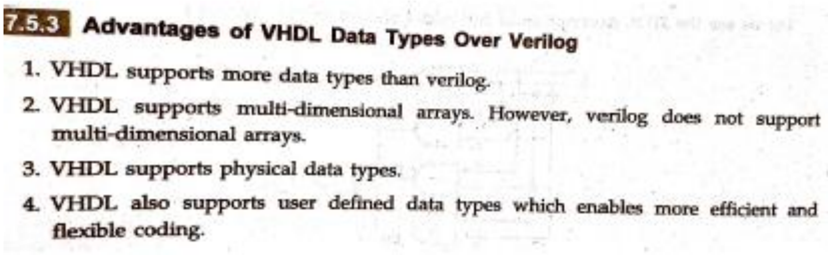
b.



7.5.1 VHDL Data Types

VHDL supports a variety of data types. The type of a variable, signal, or constant determines the operators that are predefined for that object as well as the range of values that it can take on. The VHDL data types can be broadly classified into following five data types :

- **Scalar types :** The scalar types include numeric data types and enumerated data types. The numeric types consist of **integer,** floating point **(real)** and physical types. **Bit, Boolean** and **character** are all enumerated types.
- **Composite types :** Array and record types are composite data types. The values of these types are collection of their elements.
- **Access types :** They are pointers; they provide access to objects of a given data type.
- **File type :** They provide access to object that contain a sequence of values of a given type.
- **Other types :** They include the data types provided by the several external libraries.

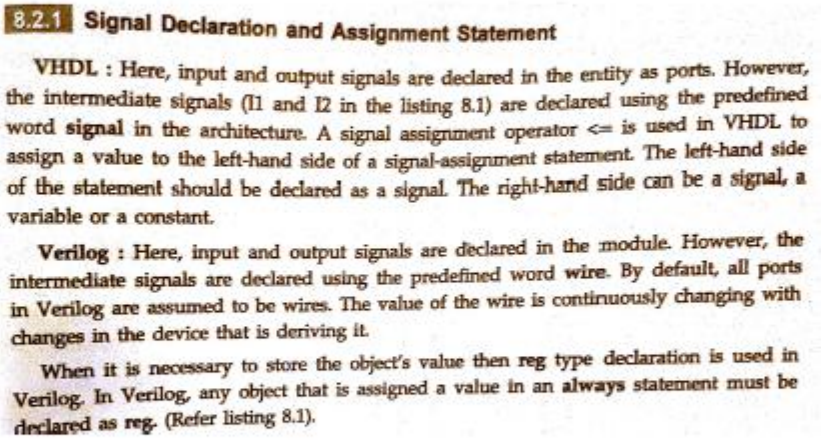


7.5.3 Advantages of VHDL Data Types Over Verilog

1. VHDL supports more data types than verilog.
2. VHDL supports multi-dimensional arrays. However, verilog does not support multi-dimensional arrays.
3. VHDL supports physical data types.
4. VHDL also supports user defined data types which enables more efficient and flexible coding.

.

10 a.



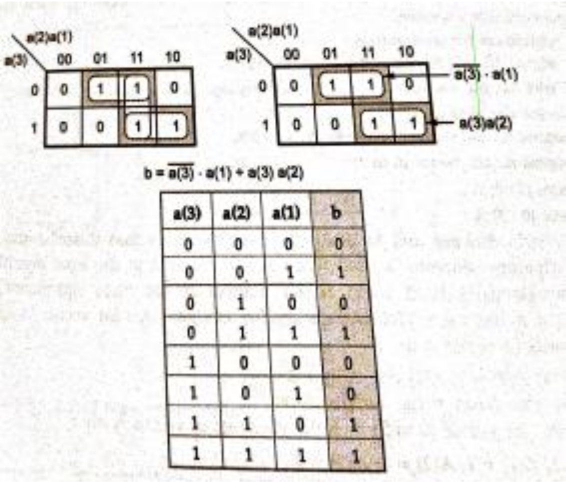
8.2.1 Signal Declaration and Assignment Statement

VHDL : Here, input and output signals are declared in the entity as ports. However, the intermediate signals (I1 and I2 in the listing 8.1) are declared using the predefined word **signal** in the architecture. A signal assignment operator <= is used in VHDL to assign a value to the left-hand side of a signal-assignment statement. The left-hand side of the statement should be declared as a signal. The right-hand side can be a signal, a variable or a constant.

Verilog : Here, input and output signals are declared in the module. However, the intermediate signals are declared using the predefined word **wire.** By default, all ports in Verilog are assumed to be wires. The value of the wire is continuously changing with changes in the device that is deriving it.

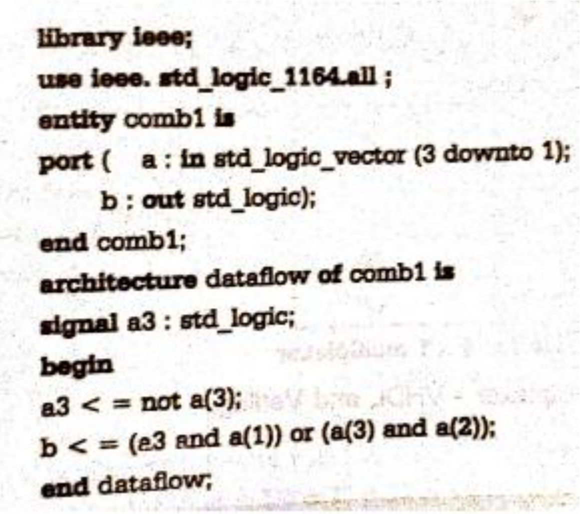
When it is necessary to store the object's value then **reg** type declaration is used in Verilog. In Verilog, any object that is assigned a value in an **always** statement must be declared as **reg.** (Refer listing 8.1).

b.



$b = \overline{a(3)} \cdot a(1) + a(3)\, a(2)$

a(3)	a(2)	a(1)	b
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



```
library ieee;
use ieee. std_logic_1164.all ;
entity comb1 is
port (   a : in std_logic_vector (3 downto 1);
        b : out std_logic);
end comb1;
architecture dataflow of comb1 is
signal a3 : std_logic;
begin
a3 < = not a(3);
b < = (a3 and a(1)) or (a(3) and a(2));
end dataflow;
```