DIGITAL INTEGRATED CIRCUITS ANALYSIS

ELECTRONICS & COMMUNICATION ENGINEERING

unit- 3 & 4

HAND NOTES

BY:

P.RAJESH M.TECH.,

ASSISTANT PROFFESOR

EMAIL: rajesh.crit@gmail.com

ph: 9989786119

9985786099

CRIT COLLEGE OF ENGG & TECHNOLOGY

ANANTAPURAMU

Unit - 6

The VHDL Hardware Description Language :-

HDL :- (Hardware Description Language which describes the hardware of digital systems).

VHDL :-

Normally HDL's available many. we will focus on HDL, which is IEEE (Institute of Electrical and Electronics Engineering).

VHDL — Very High Speed IC Hardware Description Language.

Library :-

The VHDL Compiler stores all information about the process and the place where it stores the information is known as VHDL library.

The IEEE standard library is

```
Library IEEE;
```

In IEEE standard 1164 Package it can be written

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

where IEEE indicates the name of Library

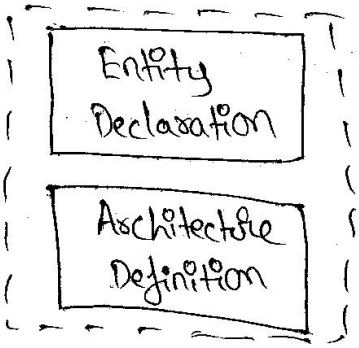
STD_LOGIC_1164. ~~ALL~~ → indicates the file name

ALL → indicates compiler to use all definations in that file.

*) VHDL Program Structure :-

The Principle of structured Programming are used in VHDL design gives VHDL Program file structure.

In the text file of VHDL Program, the Entity and Architecture are separated.



basic blocks.

*) Data Flow Design Elements of VHDL :-

The Process of representing a circuit in terms of flow of data is known as Data flow design.

The data flow design elements are

1) Concurrent signal_assignment statement
2) Conditional signal_assignment statement
3) Selected signal_assignment statement.

1) Concurrent signal_assignment statement :-

A concurrent signal assignment statement is used to assign a value to a signal in an architecture body.

Syntax :-

Signal_name <= Expression ;

we should read as "signal name gets expression".

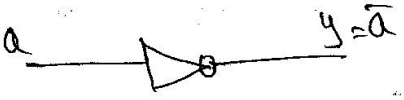
The symbol <= implies an assignment of value to a signal.

The value of expression on a right hand side is computed and assigned to signal on left hand side called a target signal.

Eg:- VHDL Program for Designing NOT gate in data flow style using Concurrent signal assignment.

i/p	o/p
a	$y = \bar{a}$
0	1
1	0

Sol



Program :-

```
Library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity not_gate is
  Port (a : in STD_LOGIC;
        y : out STD_LOGIC);
  end not_gate;
architecture data flow of NOT_gate is
begin
  y <= Not a;
  end data flow;
```

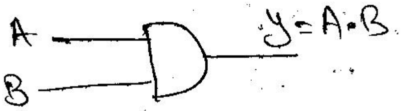
2) Conditional Signal Assignment Statement :-

" " " selects different values for target signal based on specified conditions.

Syntax :-

```
Signal_name <= Expression when boolean Expression Else,
               "          "     "         "         " ;
               - - - - - - - - - - - - - - - -
               - - - - - - - - - - - - - - - -
               Expression when boolean Expression Else;
         Expression ;
```

Eg:- VHDL Program for AND gate data flow using Conditional signal statement :-



A	B	y
0	0	0
0	1	0
1	0	0
1	1	1

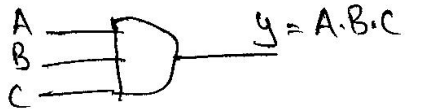
Program :-

```
Library IEEE ;
Use IEEE . STD _ LOGIC _ 1164 . ALL;
Use IEEE . STD _ LOGIC _ ARITH . ALL;
Use IEEE . STD _ LOGIC _ UNSIGNED . ALL;
Entity and_gate is
Port ( a : in STD_LOGIC ;
       b : in STD_LOGIC;
       y : out STD_LOGIC);
end and_gate;
architecture dataflow of AND_gate is
begin   y<=1 when a='1' and b='1' else '0'
     end data flow;
```

architecture of data flow MUX 4x1 is
with 'S' select
y <= 'd(0)' when "00";
'd(1)' when "01";
'd(2)' when "10";
'd(3)' when "11";
end data flow;

Q) write a VHDL Program for 3-ip AND gate in data flow style?

Sol



A	B	C	y = A.B.C
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Program:-

Library IEEE;
use IEEE.Std_logic_1164.ALL;
use IEEE.Std_logic_ARITH.ALL;
use IEEE.Std_logic_UNSIGNED.ALL;
entity AND_gate is
Port (A : in Std_logic;
B : in Std_logic;
C : in Std_logic;
y : out Std_logic);
end AND_gate;
architecture data flow of AND_gate is
begin
y <= A and B and C;
end data flow;

*) Libraries :-

1) Library ieee;
2) use ieee.Std_logic_1164.All;
3) use ieee.Std_logic_arith.all;
4) use ieee.Std_logic_signed.all;
5) use ieee.Std_logic_unsigned.all;

*) Data types :-

1) bit values '0', '1'
2) boolean values : TRUE, FALSE
3) integer values -231 to +(231-1)
4) Std_logic_vector (n downto 0);
5) Std_logic_vector (0 up to n);

*) VHDL features :-

1) Case insensitive :-
   ↳ INPUTA & inputA, inputa are refer to same variables

2) Comments :-
   ↳ "--" untill end of the line

3) Statements are terminated by ';'

4) signal assignment :
   "<="

5) User defined names :
   ↳ letters, numbers, underscores("_")
   ↳ start with a letter.

*) Syntax of architecture :-

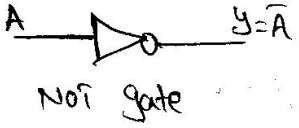
```
architecture  architecture_name of entity_name is
    type declaration
    signal declaration
    Constant declaration
    function declaration
    Procedure declaration
    Component declaration
begin
    Concurrent - statement
    . . .
    Concurrent - statement
end architecture_name;
```

*) Definations of VHDL std_logic type :-

type std_ULOGIC is ("U" — unintialized
'X' — forcing unknown
'0' — forcing 0
'1' — forcing 1
'Z' — High Impedance
'W' — weak unknown
'L' — weak 0
'H' — weak 1
'-' → Dont care
);

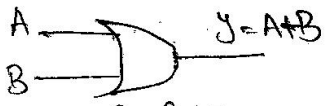
Write a VHDL Program for Logic gates using vector Statements?

4-6 (5)



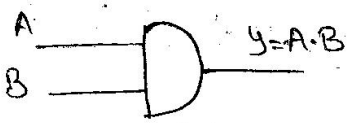
NOT gate

A	y
0	1
1	0



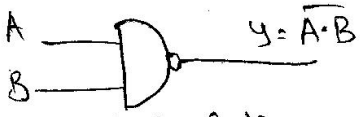
OR gate

A	B	y=A+B
0	0	0
0	1	1
1	0	1
1	1	1



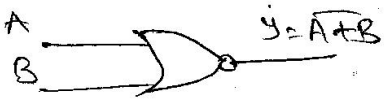
AND gate

A	B	y=A·B
0	0	0
0	1	0
1	0	0
1	1	1



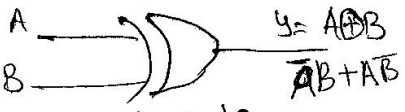
NAND gate

A	B	y=Ā·B
0	0	1
0	1	1
1	0	1
1	1	0



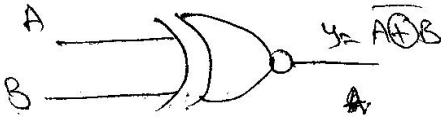
NOR gate

A	B	y=Ā+B
0	0	1
0	1	0
1	0	0
1	1	0



XOR gate

A	B	y=A⊕B
0	0	0
0	1	1
1	0	1
1	1	0



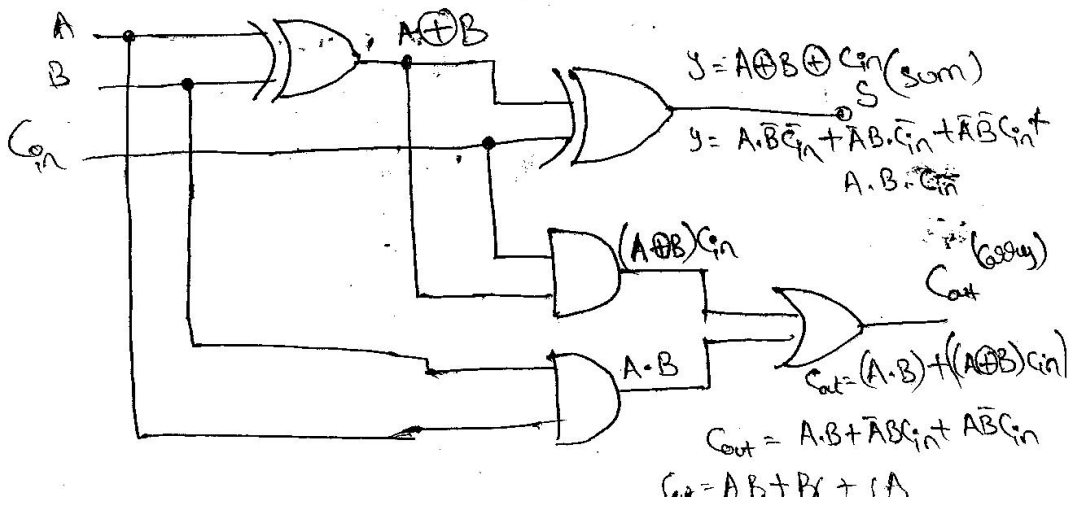
Ex-NOR.

A	B	y=A⊕B
0	0	1
0	1	0
1	0	0
1	1	1

VHDL Program:-

```
Library IEEE;
USE IEEE.STD_Logic_1164.ALL;
USE IEEE.STD_Logic_ARITH.ALL;
USE IEEE.STD_Logic_UNSIGNED.ALL;
entity Logic_gates is
Port (a, b : in Std_logic;
      y : out Std_logic_vector (6 downto 0));
end logic_gates;
architecture data flow of logic_gates is
begin
    y(0) <= not a;
    y(1) <= a or b;
    y(2) <= a and b;
    y(3) <= a nand b;
    y(4) <= a nor b;
    y(5) <= a xor b;
    y(6) <= a xnor b;
end data flow;;
```

*) Write a Program for full adder circuit using data flow?



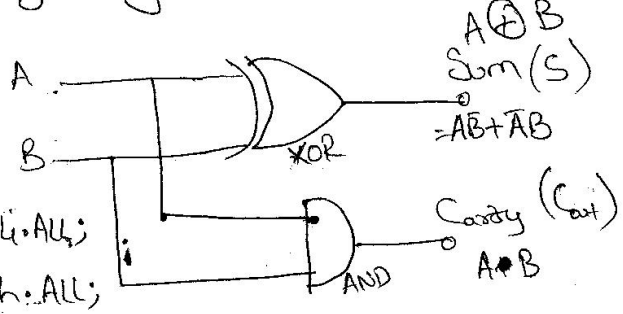
VHDL Program:-

```
Library Ieee;
use IEEE.Std_logic_1164.ALL;
use IEEE.Std_logic_ARITH.ALL;
use IEEE.Std_logic_UNSIGNED.ALL;

entity full_adder is
Port ( Cin, A, B : in Std_logic;
       S, Cout   : out Std_logic);

   end full_adder;
architecture dataflow of full_adder is
 begin
 Sum <= A xor B xor C;
 Cout <= (A and B) or (B and C) or (C and A);
end dataflow;
```

*) Write VHDL Program for Half Adder using data flow?



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

Program:-

```
Library IEEE;
use IEEE.Std_logic_1164.ALL;
use IEEE.Std_logic_Arith.ALL;
use IEEE.Std_logic_UNSIGNED.ALL;
entity Half_Adder is
Port ( A : in Std_logic;
       B : in Std_logic;
       S, Cout : out Std_logic);

End Half_Adder;
architecture dataflow of Half_Adder is
begin  Sum <= A xor B;
       A and B;
```

* Write a VHDL Program for function

$f_1 = \bar{a}b + de$

$f_2 = a + b + c$

Program:-

```
Library IEEE;
USE IEEE.Std_logic_1164.ALL;
USE IEEE.Std_logic_ARITH.ALL;
USE IEEE.std_logic_UNSIGNED.ALL;
Entity function is;
Port ( a,b,c,d,e : in Std_logic;
       f1,f2     : out Std_logic);
end function;
Architecture data flow of function is
begin
f1 <= (not a and b) & (d and e);
f2 <= a & b & c;
end data flow;
```

* Write a VHDL Program for Prime number detector.

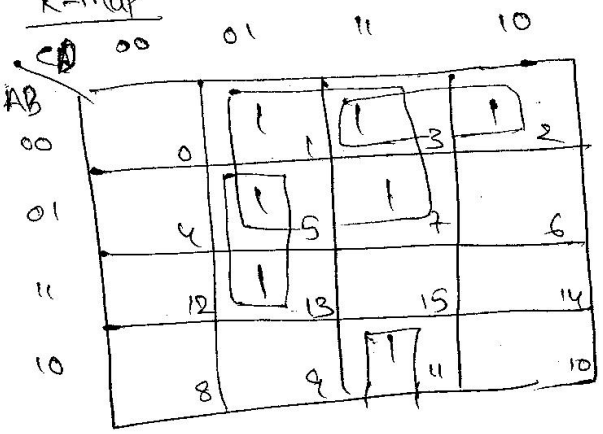
Sol

Decimal	A	B	C	D	Y
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1
12	1	1	0	0	0

1, 2, 3, 5, 7, 11, 13

4-b (+)

K-map



A=1
$\bar{A}$=0

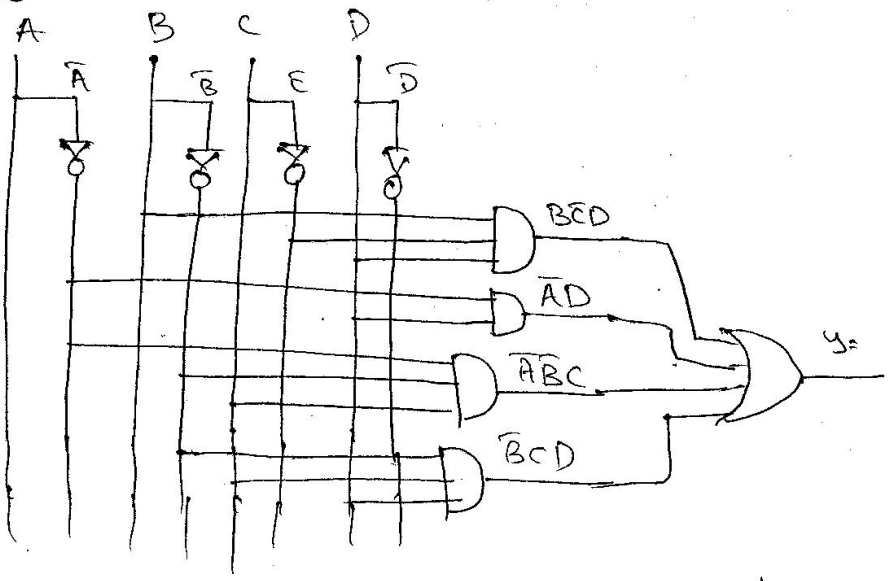
ABCD
0001
0011
0101
0111
$\bar{A}$ D

0101
1101
$B\bar{C}D$

0011
0010
$A\bar{B}C\bar{D}$

$\bar{A}BCD$

$y = \bar{B}CD + \bar{A}D + \bar{A}\bar{B}C + B\bar{C}D$



VHDL Program:- Concurrent assignment statement

```
Library IEEE;
use IEEE.Std_logic_1164.ALL;
use IEEE.Std_logic_ARITH.ALL;
use IEEE.Std_logic_UNSIGNED.ALL;
entity Prime number detector is
Port (A,B,C,D : in Std_logic;
      y : out Std_logic);
end Prime number detector;
Architecture dataflow of Prime number detector is
begin.
```

```
signal S1, S2, S3, S4 : std_logic;
begin
S1 <= not A and D;
S2 <= not A and not B and C;
S3 <= B and not C and D;
S4 <= not B and C and D;
y <= S1 & S2 & S3 & S4;
end data flow;
```

*) write VHDL Program for Prime detector using Conditional signal assignment statement?

Program :-

```
library IEEE;
use IEEE.STD_logic_1164.ALL;
use IEEE.std_logic_ARITH.ALL;
use IEEE.std_logic_UNSIGNED.ALL;
entity Prime_number_detector. is;
Port (A, B, C, D : in std_logic;
      y : out std_logic);
end Prime number detector;
Architecture dataflow of Prime detector is
  signal S1, S2, S3, S4 : std_logic;
begin.
S1 <= '1' when A='0' and D='1' else '0';
S2 <= '1' when B='0' and A='0' and C='1' else '0';
S3 <= '1' when B='1' and C='0' and D='1' else '0';
S4 <= '1' when B='0' and C='1' and D='1' else '0';
y <= S1 & S2 & S3 & S4;
end data flow;
```

A=0, D=1 } S1

*) Write a VHDL Program for Prime detector using selected signal assignment statements?

Program:-

```
library IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
entity prime detector is;
Port (a,b,c,d : in std_logic;
      y : out std_logic);
end prime number detector;
architecture dataflow of prime detector is
begin
  with 'N' select
  y<='1' when '0001';
  y<=1 when '0010'|'0011'|'0101'|'0111';
  y<=1 when '1011'|'1101';
  y<=0 when others;
  end dataflow;
```

*) Behavioural Design Elements:-

VHDL is key behavioural element is the "Process". A Process is a collection of sequential statements that executes in parallel with other concurrent statements that executes in parallel with other Concurrent statements and other Process.

VHDL Process statement can be used any where that a Concurrent signal statement can be used. A Process statements is introduced by key word "Process".

Syntax:-

Process (signal_name, signal_name ..... signal_name)

Type declarations
Variable declerations
Constant declerations
function declerations
Procedure definations

begin
sequential statement
.
.
.
sequential statement
end Process;

A VHDL Process is always either running (or) suspended. The signal in the Process definition called the "sensitivity list". When the Process runs A Process initially is suspended with signal in its sensitivity list changes value the Program re-Execution, starting with its first sequential state and Continuing untill the End.

"The Process statement describes the behaviour of design using sequential statements".

Sequential statements:-

VHDL has several kinds of sequential statements as follows.

1) signal assignment statement:-

The value of expression on RHS is assigned to LHS signal. which is similar syntax as Concurrent version

(signal_name <= expression;)

2) Variable-assignment statement:-

The value of expression on RHS assigned to LHS variable

Syntax:- "variable_name := expression";

3) If Statement :-

If Condition is true, the Corresponding set of sequential statements are executed

Syntax :- if boolean - expression then sequential statement
end if ;
if boolean - expression then sequential statement
end if ;

4) else if Statement and else if syntax :-

Syntax :- if boolean - expression then sequential - statements
end if ;
if boolean - expression then sequential - statements
else sequential - statements
end if ;
if boolean - expression then sequential - statements
else if boolean - expression then sequential - statements
end if ;
if boolean - expression then sequential - statements
else if boolean - expression then " "
. . . . . " " "
else if "
Else sequential - statements
end if ;

5) Case Statements :-

This Statement Evaluates the given Expression finds the matching value in one of the choices and executes the corresponding Sequential Statements. One (or) more sequential statements can be written for each set of choices. The choices may take the form of a single value (or) multiple values seperated by vertical bares ( ).

Syntax :-

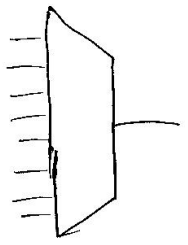
```
Case expression is
when choices => sequential - Statements
.....
When choices => sequential - Statements
end case;
```

Eg :- Write a VHDL Program for 8x1 MUX using if Statement ?

```
Library ieee;
use ieee.std_logic_1164.ALL;
Entity 8x1 Mux is
Port ( D : in std_logic_vector (7 down to 0);
       S : in std_logic_vector (2 down to 0);
       y : out std_logic);
end 8x1 Mux;
Architecture multiplexer of 8x1 Mux is
begin
  Process (D,S)
  begin
    if S="000" then y <= D(0);
  else if S="001" then y <= D(1);
  else if S="010" then y <= D(2);
  else if S="011" then y <= D(3);
  else if S="100" then y <= D(4);
  else if S="101" then y <= D(5);
```

$2^3 = 2^n$
$n = 3$

8x1 MUX



*) Write a VHDL Program for Half_Adder using behavioural

```
Library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_ARITH.ALL;
use ieee.std_logic_UNSIGNED.ALL;
Entity Half_Adder is
Port ( a, b : in std_logic;
       Sum, Carry : out std_logic);
end Half_Adder;
Architecture behavioural of Half_Adder is
begin
   Process (a, b)
   begin
      Sum <= a xor b;
      Carry <= a and b;
   end Process;
   end behavioural;
```

*) Write a VHDL Program for full adder using behavioural

```
Library ieee;
use ieee.std_logic_1164.ALL;
Entity full_Adder is
Port ( A, B, Cin : in std_logic;
       Sum, Carry : out std_logic);
end full_Adder;
Architecture behavioural model of full_Adder is
begin
   Process (a, b, c)
   begin
```

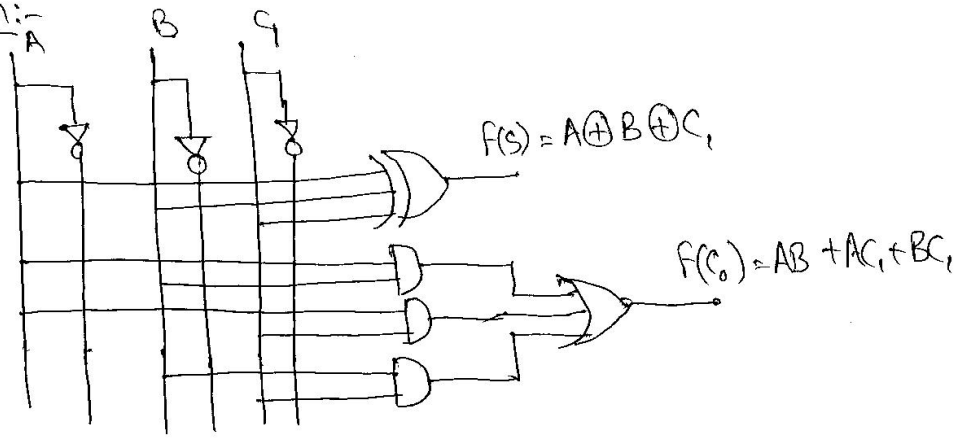
```
Sum <= a xor b xor C;
Carry <= (a and b) or (b and C) or (C and a);
end Process;
end behavioral;
```

(8) ~~Structural~~ × ~~dataflow~~ × ~~model~~

*) Write behavioral Model VHDL Program for following function with logic diagram for functions

$F(S) = A \oplus B \oplus C_1$; $F(C_0) = AB + AC_1 + BC_1$

Logic diagram:-



Program:-

```
Library ieee;
use ieee.Std_logic_1164.ALL;
entity function is
Port (A,B,C1 : in Std_logic;
      S,C0 : out Std_logic);
end function;
Architecture behavioral of function is
begin
Process (A,B,C1)
begin
S <= A xor B xor C;
C0 <= (A and B) or (A and C1) or (B and C1);
end Process; end behavioral;
```

*) Write a VHDL Program for Prime detector by using Case statement?

```
Library ieee;
use ieee.std_logic_1164.ALL;
entity Prime_detector is
Port (a : in std_logic_vector(3 down to 0);
      y: out std_logic.);
end Prime_detector;
Architecture behavioral of Prime detector is
begin
   Process(a).
   begin
      Case a is
   when 0001 then y<=1;
   when 0010|0011|0101|0111|1011|1101 then y<=1;
   when others then y<=0;
   end case;
   end Process;
   end behavioural;
```

*) write a Program for 3-i/p AND gate by using Case statement in behavioural model?

Program:-

```
Library ieee;
use ieee.std_logic_1164.ALL;
entity AND_gate
Port (a : in std_logic_vector(2 down to 0);
      y: out std_logic);
end AND_gate;
Architecture behavior of AND_gate is
begin
   Process(a)
begin case a is
when 111 then y<=1;
when others then y<=0;
end Case;
end Process; end behavioural;
```

*) Functions and Procedures :-

A VHDL function accepts a no:- of argument and returns a result similar to a function in a high level language. The type of arguments and result are to be specified in a VHDL function defination.

```
FUNCTION function_name
(
signal_names : signal_type;
signal_names : signal_type;
    :
    :
signal_names : signal_type;
)
Return return_type is

type declaration
Constant declaration
Variable declaration
function definations
Procedure definations

Begin
    sequential statement
      :
      :
    sequential statement
END function_name;
```

*) Time Dimension :-

The VHDL provides one predefined physical type : time.

The defination of type 'time' as given Package 'STANDARD' is

TYPE TIME IS RANGE < implementation dependent >

units
    fs ; ---- femto seconds
    ps = 1000 fs ; ------ Pico seconds
    ns = 1000 ps ; ---- nano seconds
    us = 1000 ns ; ------ micro seconds
    ms = 1000 us ; ------ milli seconds
    s = 1000 ms ; ------ seconds
    min = 60 s ; ------ minutes
    hr = 60 min ; ---- hours
END units ;

There are 2-key words in specify of time delay.

1. using key word "after";
2. using "wait" statement.

Eg:- Key word of 'After' :-

After specifys time delay in signal assignment statement such as Concurrent, Selected, Conditional statements.

```
Library ieee ;
use ieee.std_logic_1164.all ;
Entity flip_flop is
port ( D, clock : in std_logic ;
       Q : out std_logic ) ;
End flip_flop ;
```

```
Architecture behavior of flip_flop is
begin
  Process (clock)
  begin
    if clock'Event AND clock = '1' then
      Q <= D After 3ns;
    End if;
  End Process;
  End behavior;
```

Key word of "wait" Statement

Wait statement causes suspension of process for the time period obtained by evaluating the time expression

different types of wait statements :-

1) wait for time expression;-
   Eg:- wait for 10 ns;
2) wait ON signal; → Eg:- wait on clk, reset;
3) wait UNTIL <Condition>; → Eg:- wait until (clk'Event AND clk = '1')
4) wait.

```
Eg:- Library ieee;
use ieee.std_logic_1164.ALL;
Entity flip_flop is
Port (D, clock : In std_logic;
      Q : out std_logic);
End flip_flop;
Architecture Behavior of flip_flop is
begin
  Process.
```

Program:-

```
Library ieee;
use ieee.std_logic_1164.ALL;
entity function is
Port (a,b,c,d : in std_logic;
      F(x) : out std_logic);
end function;
Architecture data flow of function is
begin
F(x) <= (not D) or (not B) or (not A and B and not C) or
        (not A and not B and C);
end data flow;
```

*) Design the logic circuit and write a data flow style VHDL Program for following function

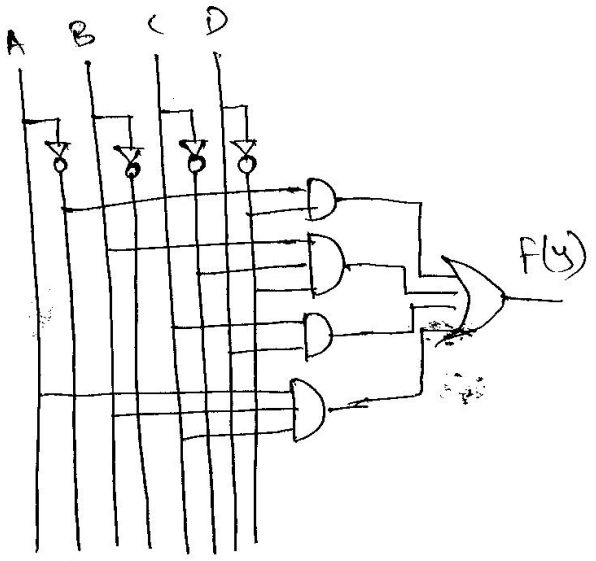
$$F(y) = \Sigma_{A,B,C,D}(1,4,5,7,12,14,15) + d(3,11)$$

Sol

K-map

AB \ CD	00	01	11	10
00	0	1	X 3	2
01	1 4	1 5	7	6
11	1 12	13	1 15	1 14
10	8	9	X 11	10

$$F(y) = \bar{A}D + B\bar{C}\bar{D} + CD + ABC$$



Program:-

```
Library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_ARITH.ALL;
use ieee.std_logic_UNSIGNED.ALL;
entity function is
Port (A,B,C,D : in std_logic;
       F      : out std_logic);
end function;
architectural data flow of function is
F <= (not A and D) or (C and D) or (B and not C and not D)
                   or (A and B and C);

end data flow;
```

*) Write VHDL Program for data flow style function

$f(y) = \sum A,B,C,D\ (1,4,5,7,9,11,12,13,15)$

written as $\Sigma A,B,C,D\ (0,2,3,6,8,10,14)$

*) Structural Design model :-

The most basic of VHDL Concurrent statement is Component statement.

Syntax :-

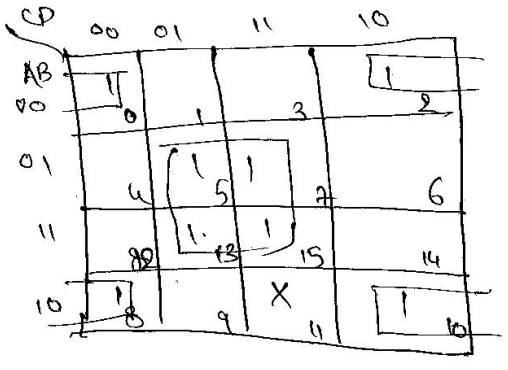
```
Component_name Port map (signal 1, signal 2, .... signal n);
Component_name Port map( Port 1 => signal 1 , Port 2 => signal 2 .... );
```

Syntax for Component declaration :-

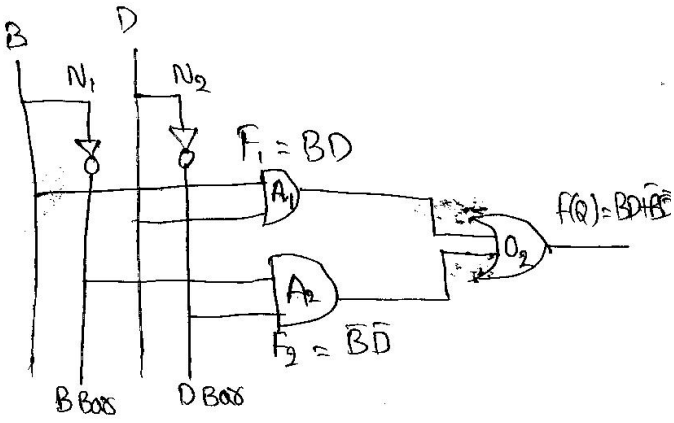
```
Component  Component-name
  Port (signal_names : mode signal_type;
        signal_names : mode signal_type;
        .....
        signal_names : mode signal_type);
  end Component;
```

*) Design the logic circuit and write a structural style VHDL Program for the following function.

$$F(Q) = \Sigma_{A,B,C,D}(0,2,5,7,8,10,13,15) + d(11)$$



$$F(Q) = BD + \bar{B}\bar{D}$$



Program:-

```
Library ieee;
use ieee.std_logic_1164.ALL;
use ieee.std_logic_ARITH.ALL;
use ieee.std_logic_UNSIGNED.ALL;

entity function is
Port ( B, D : in std_logic;
       F(Q) : out std_logic);

end function;
architecture structural of function is
signal F1, F2, B Bar, D Bar : in std_logic;

Component NOT 1
   Port ( B : in std_logic;
          y : out std_logic);
     end Component;

Component NOT 2
   Port ( D : in std_logic;
          y : out std_logic);
   end Component;
Component AND 1
   Port ( B, D : in std_logic;
          F1 : out std_logic);

   end Component;

Component AND 2
   Port (B Bar, D Bar : in std_logic;
         F2 : out std_logic);

   end Component;

Component OR 2
   Port ( F1, F2 : in std_logic;
          F(Q) : out std_logic);

      end Component;
```

```
begin
N1 : NOT 1 Port map (BBar, B);
N2 : NOT 2 Portmap (D, DBar);
A1 : AND 1 Port map (B, D, F1);
A2 : AND 2 Port map (BBar, DBar, F2);
O2 : OR 2  Port map (F1, F2, f(Q));
end structural;
```

NOT 1 Component source Code

```
Library ieee;
use ieee.std_logic_1164.ALL;
entity NOT 1 is
Port (B: in std_logic;
      y: out std_logic);
end not1;
architecture data flow of NOT 1 is
begin
      y <= NOT B;
   end data flow;
```

NOT 2 Component source Code

```
Library ieee;
use ieee.std_logic_1164.ALL;
entity NOT 2 is
Port (D : in std_logic;
      y: out std_logic);
end NOT 2;
architecture data flow of NOT 2 is
begin
      y <= NOT D;
   end data flow;
```

AND 1 Component data flow

```
Port ( B, D : in Std_logic ;
       F1   : out Std_logic);
end AND 1 ;
```

AND 2 Component data flow

```
Port ( N1, N2 : in Std_logic ;
       F2     : out Std_logic);
```

AND 1 Component source Code

```
library ieee ;
use ieee. std_logic_1164. ALL;
entity AND 1 is
Port ( B , D : in Std_logic ;
       F1 : out Std_logic);
end AND 1 ;
Architecture data flow of AND 1 is
begin   y <= B and D ;
end data flow ;
```

AND 2 Component source Code

```
Library ieee ;
use ieee. Std_logic_1164. ALL;
entity AND 2 is
Port ( N1 , N2 : in Std_logic ;
       F2 : out Std_logic);
end AND 2 ;
Architecture data flow of AND 2 is
begin   y <= N1 and N2 ; end data flow ;
```

OR 2 Component data flow

```
library ieee;
use ieee.std_logic_1164.ALL;
Entity OR2 is
Port ( F1, F2 : in std_logic;
       F(0) : out std_logic);
end OR2;
Architecture data flow of OR2
begin
   y <= F1 or F2;
end dataflow;
```