4.VHDL PROGRAMMING

VHDL Programming Introduction to VHDL Features and Capabilities, entity, architecture and levels of abstraction using case study of RS Flip-Flop. Modeling of combinational logic circuits with example of multiplexer, decoder. Sequential logic design with example of shift resister, Up Down counter

INTRODUCTION VHDL:-

- VHDL Stands for VHSIC HDL (Very High Speed Integrated Circuit Hardware Description Language)
- VHDL was developed in 1980 by US defense department
- IEEE (Institute Of Electrical And Electronics Engineering) sponsored development in 1987
- The basic purpose of VHDL is fabrication of IC
- In VHDL there are three level of abstraction
  i) Behavioral ii) Structural iii) Structural & Behavioral
- *Entity* is basic building Block of VHDL
- One of the advantages of VHDL it is uses both concurrent and sequential statements
- It is one of the most important standard language for i) specifying ii)verifying iii)designing iv) modeling v) Debugging vi) physical Realization of digital System
- *MUX* is a basic element in VHDL

Features Of VHDL (VHDL vs. Conventional Language)

i) VHDL is inherently parallel
Commands are executed concurrently and executed as soon as input is arrival

ii) Mimics the behavior of a physical an usually digital system

iii) Allows a incorporation of timing specification (Gate delays) as well as to descried the system as interconnection of different component.

iv) VHDL can be used for simulation, synthesis, verification.

v) In VHDL system can be described four different points of view : Behavioral, Structural functional, physical properties

vi) Conventional languages are mainly used for computation, data manipulation and for execution on a specific hardware model.

vii) VHDL supports different development methods a) top down b)bottom up c)mix

Level of Abstraction

Level of Abstraction is used for managing the complex system description.
Types of Level of Abstraction

i) Behavioral Abstraction
ii) Structural Abstraction
iii) Behavioral And Structural Abstraction

1) Behavioral Abstraction :-

- This is algorithm based Design
- Highest level of Abstraction
- Description of the circuit based on how it behaves?
- Specifies the relationship between input and output signals

e.g S<=A XOR B;

- Behavioral Abstraction Of the two Types
  i) Data flow :- Design Specified on the Movement of Data
  ii) Sequential :- Sequence of Execution of event

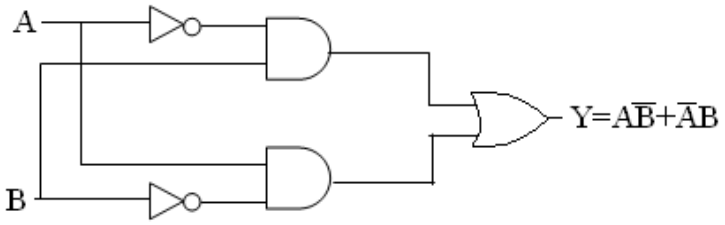
2) Structural Abstraction :-

- This one of the most optimized Design

- Describe the system as collection of gates and components interconnected to perform a designed function.
- It is a schematic of interconnected logic gates.
- This Abstraction is closer to physical realization of the system. i.e. closer to the hardware tool insensitive design.

The example of structural abstraction is as follow

This is structure of XOR gate.



3) Behavioral and Structural Abstraction

- This is mixed mode design
- It uses sequential statements e.g. Shift register

Entity:-

- Basic building Block of VHDL
- It gives component of the design whose behavior to be designed and simulated
- It consist of three things

i) Name :- it is case insensitive e.g.:- half_adder,rs_ff etc

ii) Input/output/in out:- Interface of the entity with environment to be describes the port list.

iii) Behavior:- Description of the entity convert input to output

Ports are interfaces through which an entity can communicate with its environment. Each port must have a name, direction and a type. An entity may have no port declaration also. The direction will be input, output or inout.

Entity Declaration

```
entity  name_of_entity is [genric]
  port( signal_names : mode type;
              signal_names : mode type;
                --------
                --------------------
        signal_names : mode type);

  end name_of_entity;
```

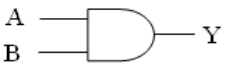
Here mode= Input/output/inout

Type= Single bit or multiple bit

;= This symbol is called as delimiters it gives the direction to the compiler

Example of entity

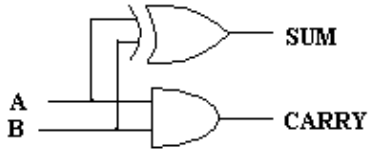
1) Entity for AND gate




```
entity  and_1 is
              port(A,B: in std_logic;
                        Y: out std_logic);
          end  and_1;
```

2) Entity for half_adder

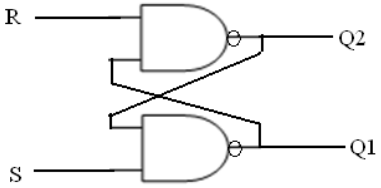
```
entity  half_adder is
  port(A,B: in std_logic;
                C,S: out std_logic);
end  half_adder;
```



3) Entity For full adder

```
entity  full_adder is
  port(A,B,C: in std_logic;
                CARRY,SUM: out std_logic);
end  full_adder;
```

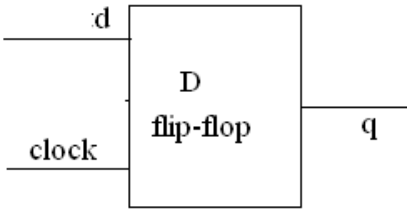
4) Entity for RS F/F



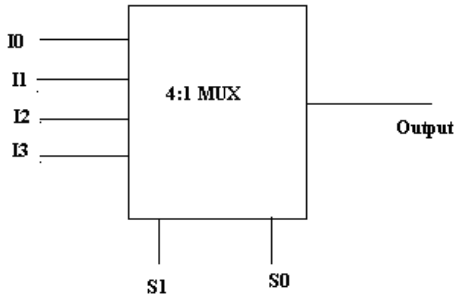

```
entity rsff is
      port ( r,s : in  STD_LOGIC;
            q1,q2 : inout  STD_LOGIC);
end rsff;
```

5) Entity for D F/F

```
entity dff is
      port ( d, clock : in  STD_LOGIC;
            q: out  STD_LOGIC);
end dff;
```



6) Entity For 4:1 Multiplexer



```
entity mux_4 is
                port ( i : in  STD_LOGIC_VECTOR (3 downto 0);
                       s : in  STD_LOGIC_VECTOR (1 downto 0);
                      output : out  STD_LOGIC);
            end mux_4;
```

ARCHITECTURE

It describes the internal description of design or it tells what is there inside design. Each entity has at least one or many architecture. Architecture can be described using structural, dataflow, behavioral or mixed style. Architecture Specifies operation and implementation of the circuit.

DECLARATION OF ARCHITECTURE

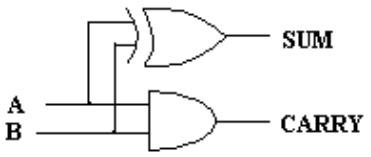
```
architecture name_of_architecture of name_of_entity is
        --declaration
        --component declaration
        --signal declaration
        --constant declaration
        --function declaration
        --procedure declaration
        --type declaration
begin
        --statement 1;
        ----------
        --statement n;
end name_of_architecture;
```

1)HALF ADDER

Half adder is used to implement addition of the two binary digtits.It is not complete adder because it is not able to add generated carry.

A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Diagram



VHDL code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity adder1 is
   Port ( a, b : in  STD_LOGIC;
          sum, carry : out  STD_LOGIC);
end adder1;

architecture Behavioral of adder1 is

begin
        carry<=a and b;
        sum<= a xor b;
end Behavioral;
```

(Here STD_LOGIC_1164 is an IEEE standard which defines a nine-value logic type, called STD_ULOGIC.

1. '1'= forcing to logoc 1

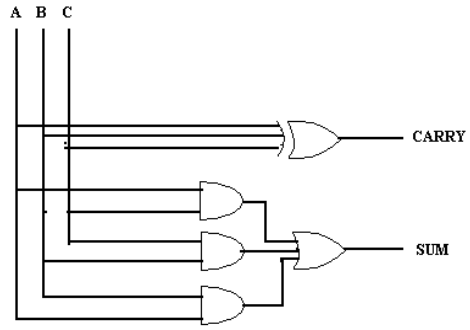
2. '0'=forcing to logic 0
3. 'Z'=high impedance state
4. 'L'= logic weak 0
5. 'H'=logic weal 1
6. 'U'=undefined
7. 'X'=unknown
8. 'W'=weak unknown
9. '-'=don't care

use is a keyword, which imports all the declarations from this package. The architecture body consists of concurrent signal assignments, which describes the functionality of the design. Whenever there is a change in RHS, the expression is evaluated and the value is assigned to LHS.)

2) FULL ADDER

Full adder is used to implement addition of the three binary digtits. It is complete adder because it is able to add generated carry.

Diagram



A	B	C	SUM	CARRY
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

VHDL code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
        entity adder1 is
   Port ( a,b,c : in  STD_LOGIC;
          sum, carry : out  STD_LOGIC);
end adder1;

architecture Behavioral of adder1 is

begin
        carry<=a xor b xor c;
        sum<= (a and b)or(b and c)or(c and a);
end Behavioral;
```

Important Statement

1. **if statement**

This statement used to execute different conditional statement. The syntax of if statement as follow,

Syntax

```
If(condition) then
                statement1;
                statement 2;
                ----------
                statement n;
end if;
```

2. **Process:-**

This is a keyword used to add the sensitivity list.

syntax

```
process(sensitivity port list)
        begin
                statement1;
                statement 2;
                -----------
                statement n;
 end process;
```

3. **Case:-**

This is a keyword used to select different choice.

syntax

```
        case(expression) is

                when choice 0=>output<=selected signal ;
                when choice 1=>output<=selected signal ;
                ------------------------------------------------;
                when choice n=>output<=selected signal ;
                when others=>output<=selected signal ;

        end case;
```

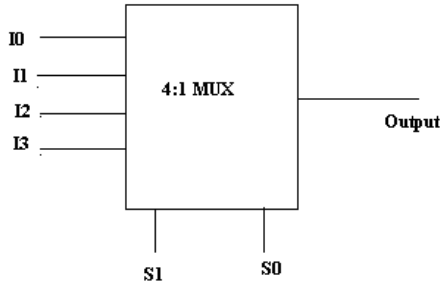
3) 4- BIT MUX

A multiplexer means it has many input and one output. Multiplexer has N input and M select lines one output. M select lines depends on N input which is given by

$N=2^M$

4:1 mux has 4 input line 2 select lines and one output line.

Diagram



S1	S0	output
0	0	I0
0	1	I1
1	0	I2
1	1	I3

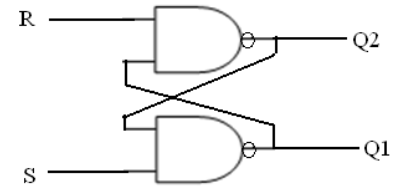
VHDL code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
      entity mux_1 is
   Port ( i : in  STD_LOGIC_VECTOR (3 downto 0);
          s : in  STD_LOGIC_VECTOR (1 downto 0);
          output : out  STD_LOGIC);
end mux_1;
architecture Behavioral of mux_1 is
begin
          process(i)
            begin
                    case s is
```

```
                    when "00"=>output<= i(0);
                    when "01"=>output<= i(1);
                    when "10"=>output<= i(2);
                    when "11"=>output<= i(3);
                    when others=>output<='Z';
                end case;
        end process;
end Behavioral;
```

4) RS FLIP FLOP

Diagram



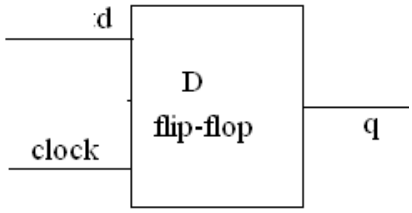
VHDL code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity rsff is
   Port ( R,S : in  STD_LOGIC;
          Q1,Q2 : inout  STD_LOGIC);
end rsff;
architecture Behavioral of rsff is
begin
        Q1<=not(Q2 and S);
        Q2<=not(Q1 and R);
end Behavioral;
```

5) CLOCKED RS FLIP FLOP

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 entity rs_clk is
   Port ( r,s,clock : in  STD_LOGIC;
          q1,q2 : inout  STD_LOGIC);
end rs_clk;

architecture Behavioral of rs_clk is

begin
        process(clock)
        begin
                if(clock='1' and clock' event)then
                        q1<=not(q2 and s);
                        q2<=not(q1 and r);
                end if;
        end process;
end Behavioral;
```

6) CLOCKED D FLIP FLOP

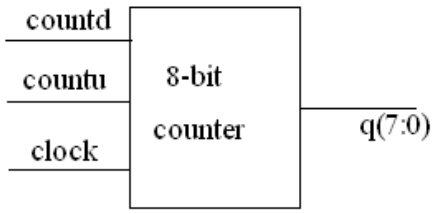



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 entity dff is
   Port ( d,clock : in  STD_LOGIC;
                  q : inout  STD_LOGIC);
end dff;

architecture Behavioral of dff is

begin
          process(clock)
          begin
                  if(clock='1' and clock' event)then
                          q<=d;
                  end if;
          end process;
end Behavioral;
```

7) UP-DOWN COUNTER

Digital counters consist of a number of flip-flops. Their function to count electrical pulses.

Diagram

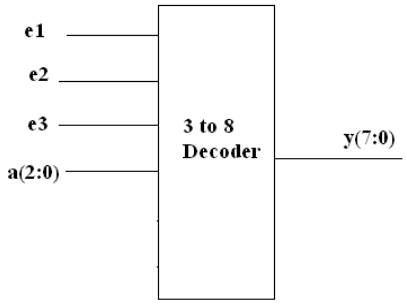


VHDL CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 entity counter_8 is
   Port ( clock,countu,countd : in  STD_LOGIC;
        q : out  STD_LOGIC_VECTOR (7 downto 0));
end counter_8;

architecture Behavioral of counter_8 is

          signal Q1: std_logic_vector(7 downto 0);

begin
```

```
process(clock, countd, countu)
   begin
          if (clock='1' and clock'event) then
          if countd = '1' then
             Q 1<= Q1 - 1;
                    if countu = '1' then
                  Q1 <= Q1 + 1;
                   end if;
          end if;
          end if;
   end process;
q<=Q;
end behavioral;
```

8) 3:8 DECODER(74138)



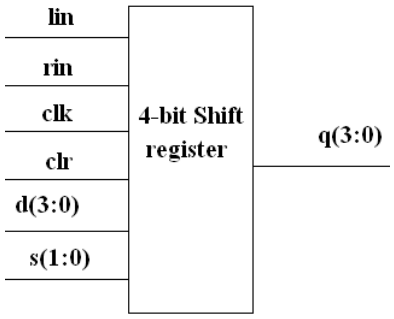
VHDL CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity decoder is
   Port ( a : in  STD_LOGIC_VECTOR (2 downto 0);
        e1,e2,e3 : in  STD_LOGIC;
        y : out  STD_LOGIC_VECTOR (7 downto 0));
end decoder;

architecture Behavioral of decoder is
begin
          process(a)
          begin
                  if(e1='1' and e2='0' and e3='0') then
                                  case conv_integer(a) is
                            -- conv_integer(signal)conversion of binary value to the decimal form
                                  when 0=>y<="11111110";
                                  when 1=>y<="11111101";
                                  when 2=>y<="11111011";
                                  when 3=>y<="11110111";
                                  when 4=>y<="11101111";
                                  when 5=>y<="11011111";
                                  when 6=>y<="10111111";
                                  when 7=>y<="01111111";
                                  when others=>y<="11111111";
                          end case;
                  end if;
          end process;
```

end Behavioral;

8)4 BIT SHIFT REGISTER (74194)

Diagram



TRUTH TABLE

FUCTION	INPUTS		NEXT STATE			
	S1	S0	Q3*	Q2*	Q1*	Q0*
HOLD	0	0	Q3	Q2	Q1	Q0
SHIFT RIGHT	0	1	RIN	Q3	Q2	Q1
SHIFT LEFT	1	0	Q2	Q1	Q0	LIN
LOAD	1	1	D3	D2	D1	D0

RIN:- RIGHT INPUT ,LIN:- LEFT INPUT

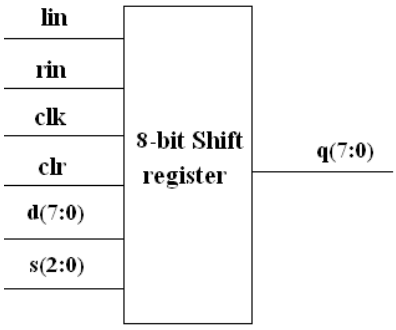
VHDL CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity shift_4 is
   Port ( clk,clr,lin,rin : in  STD_LOGIC;
          s : in  STD_LOGIC_VECTOR (1 downto 0);
          d : in  STD_LOGIC_VECTOR (3 downto 0);
          q : out  STD_LOGIC_VECTOR (3 downto 0));
end shift_4;

architecture Behavioral of shift_4 is
          signal iq: STD_LOGIC_VECTOR (3 downto 0);
begin

          process(clk,clr,iq)

          begin
                    if(clr='1')then iq<=(others=>'0');
                              if(clk' event and clk='1') then
                              case conv_integer(s)is
                                        when 0=>null;
                                        when 1=> iq<=rin & iq(3 downto 1);
                                        when 2=> iq<=iq(2 downto 0)& lin;
                                        when 3=> iq<=d;
                                        when others=> null;

                              end case;
                              end if;
                    end if;
          end process;
```

```
        q<=iq;
end Behavioral;
```

9) 8 BIT SHIFT REGISTER (74194)

Diagram



Truth Table

FUCTION	INPUTS			NEXT STATE							
	S2	S1	S0	Q7*	Q6*	Q5*	Q4*	Q3*	Q2*	Q1*	Q0*
HOLD	0	0	0	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
LOAD	0	0	1	D7	D6	D5	D4	D3	D2	D1	D0
SHIFT RIGHT	0	1	0	RIN	Q7	Q6	Q5	Q4	Q3	Q2	Q1
SHIFT LEFT	0	1	1	Q6	Q5	Q4	Q3	Q2	Q1	Q0	LIN
SHIFT CIRCULAR RIGHT	1	0	0	Q0	Q7	Q6	Q5	Q4	Q3	Q2	Q1
SHIFT CIRCULAR LEFT	1	0	1	Q6	Q5	Q4	Q3	Q2	Q1	Q0	Q7
SHIFT ARITHMETIC RIGHT	1	1	0	Q7	Q7	Q6	Q5	Q4	Q3	Q2	Q1
SHIFT ARITHMETIC LEFT	1	1	1	Q6	Q5	Q4	Q3	Q2	Q1	Q0	0

RIN:- RIGHT INPUT ,LIN:- LEFT INPUT

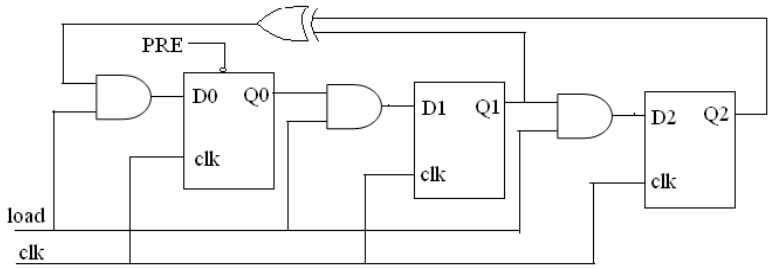
VHDL CODE

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

entity shift_8 is
   Port ( clk,clr,lin,rin : in  STD_LOGIC;
          s : in  STD_LOGIC_VECTOR (2 downto 0);
          d : in  STD_LOGIC_VECTOR (7 downto 0);
          q : out  STD_LOGIC_VECTOR (7 downto 0));
end shift_8;

architecture Behavioral of shift_8 is

          signal iq: Std_logic_vector(7 downto 0);
begin
          process(clk,clr,iq)
          begin
                    if(clr='1') then iq<=(others=>'0');
                    if(clk' event and clk='1')then
                              case conv_integer(s) is
                                        when 0 =>null;
                                        when 1=> iq<=d;
                                        when 2=> iq<=rin & iq(7 downto 1);
                                        when 3=> iq<=iq(6 downto 0)& lin;
                                        when 4=> iq<=iq(0)& iq(7 downto 1);
```

```
                                when 5=> iq<=iq(6 downto 0) & iq(7);
                                when 6=> iq<=iq(7) & iq(7 downto 1);
                                when 7=> iq<=iq(6 downto 0) & '0';
                                when others=> null;
                        end case;
                end if;
                end if;
        end process;
        q<=iq;
end Behavioral;
```

10)LFSR (Linear feedback Shift Register)

Diagram



TRUTH TABLE

CLOCK	Q2	Q1	Q0
0	1	0	1
1	0	1	1
2	1	1	0
3	1	1	0
4	1	0	0
5	0	0	1
6	0	1	0
7	1	0	1

VHDL CODE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity lfsr_1 is
   Port ( d : inout  STD_LOGIC_VECTOR (2 downto 0);
        clk,load : in  STD_LOGIC;
        q : inout  STD_LOGIC_VECTOR (2 downto 0));
end lfsr_1;

architecture Behavioral of lfsr_1 is
        signal y : STD_LOGIC_VECTOR (2 downto 0);
begin
        process(clk,load)
        begin
                if(pre<='0') then
                        y<="001";
                if(load='1')then
                        d(0)<=y(1)xor y(2)and load;
                        d(1)<=y(0)and load;
                        d(2)<=y(1)and load;
                if(clk='1' and clk' event)then
                        y(0)<=y(1)xor y(2)and load;
                        y(1)<=y(0)and load;
                        y(2)<=y(1)and load;
                end if;
                end if;
                end if;
        end process;
        q<=y;
end Behavioral;
```

Question Bank

1) Write a short note on VHDL.
2) Explain features of VHDL or Difference Between VHDL Vs Conventional Language.
3) What are level of abstraction in VHDL? Explain in detail.
4) Explain Entity and its declaration in VHDL With examples.
5) Explain architecture and its declaration in VHDL With examples.
6) Write entity and architecture for
   i) half adder
   ii) full adder
   iii) 4 –bit mux
   iv) Rs flip flop clocked and without clocked
   v) Up down counter
   vi) Decoder
   vii) Shift register (4 & 8 bit) or Sequential circuit
   viii) LFSR
7) VHDL stands for----------
8) ---------- is a basic building block in VHDL
9) ---------- is basic element in VHDL
10) --------- is a delimiter in VHDL