VHDL Codes

by

Prof. Sujata Wakchaure

VHDL Code For Half Adder By Data Flow Modelling

```
library ieee;
use ieee.std_logic_1164.all;

entity half_adder is
port(a, b: in bit;
        s, c: out bit);
end half_adder;

architecture half_adder of half_adder is
 begin
   s <= (a xor b);
   c <= (a and b);
end half_adder;
```

VHDL Code For Full Adder By Data Flow Modelling

```
library ieee;
use ieee.std_logic_1164.all;

entity full_adder is
  port(a, b, c: in bit;
          sum, carry: out bit);
end full_adder;

architecture full_adder of full_adder is
  begin
    sum <= ((a xor b) xor c);
    carry <= ((a and b) or (b and c) or (c and a));
end full_adder;
```

Full Adder using Behavioral Modeling

```
entity full_adder is
    Port ( a : in std_logic;
           b : in std_logic;
           cin : in std_logic;
           sum : out std_logic;
           carry : out std_logic);
end full_adder;

architecture Behavioral of full_adder is

begin
process(a,b,cin)
  begin
     if(a='0' and b='0' and cin='0')then
       sum<='0';
       carry<='0';
```

Full Adder using Behavioral Modeling (Contd..)

```
elseif( a='0' and b='0' and cin='1')then
   sum<='1';
   carry<='0';

elseif( a='0' and b='1' and cin='0')then
   sum<='1';
   carry<='0';

elseif( a='0' and b='1' and cin='1')then
   sum<='0';
   carry<='1';

elseif( a='1' and b='0' and cin='0')then
   sum<='1';
  carry<='0';
```

Full Adder using Behavioral Modeling (Contd..)

```
elseif( a='1' and b='0' and cin='1')then
   sum<='0';
   carry<='1';

elseif( a='1' and b='1' and cin='0')then
  sum<='0';
  carry<='1';

else
  sum<='1';
  carry<='1';
end if;
end process;

end Behavioral;
```

Full Adder using Structural Modeling

```
entity full_adder is
    Port ( a, b, cin : in  STD_LOGIC;
           sum, cout : out  STD_LOGIC);
end full_adder;


architecture full_adder_str of full_adder
is

component xor2
  port(d1,d2:in std_logic;
       dz:out std_logic);
end component;

component or2
  port(n1,n2:std_logic;
       z:out std_logic);
end component;
```

OR

```
entity full_adder is
    Port ( a, b, c: in STD_LOGIC;
sum ,carry: out STD_LOGIC);
end full_adder;

architecture  Structural_FA of full_adder is


signal c1, c2, c3: in std_logic;

component xor_2 is
 port (k, d, e: in std_logic;
       f: out std_logic);
end component;

component and_1 is
Port ( x, y : in STD_LOGIC;
       z : out STD_LOGIC);
end component;
```

Full Adder using Structural Modeling (Contd..)

```
component and2
     port(a1,a2:in std_logic;
               u:out std_logic);
end component;

signal s1,s2,s3,s4,s5:std_logic;

begin
   x1:xor2 port map(a,b,s1);
   x2:xor2 port map(s1,cin,sum);

   r1:and2 port map(a,b,s2);
   r2:and2 port map(b,cin,s3);
   r3:and2 port map(a,cin,s4);

   o1:or2 port map(s2,s3,s5);
   o2:or2 port map(s4,s5,cout);
end full_adder_str;
```

OR

```
component or_1 is
Port ( g, h, i : in STD_LOGIC;
z : out STD_LOGIC);
end component;

begin
X0: xor_2 port map (a, b, c, sum);

X2: and_1 port map (a,b,c1);
X3: and_1 port map (a,b,c2);
X4: and_1 port map (a,b,c3);

x5: or_1 port map (c1, c2, c3,
carry);
end Structural_FA;
```

4:1 Mux using Dataflow modeling

```
library ieee;
use ieee.std_logic_1164.all;

entity mux is
  port(s: in bit_vector(1 downto 0);
       d: in bit_vector(3 downto 0);
       y: out bit);
end mux;

architecture dataflow of mux is
  begin
    y<=d(0) when s="00" else
       d(1) when s="01" else
       d(2) when s="10" else
       d(3);
end dataflow;
```

VHDL Code For 4:1 Multiplexer Using IF-ELSE Statements (Behavioral Modelling)

```
library ieee;
use ieee.std_logic_1164.all;

entity mux_4_1 is
 port(s: in bit_vector(0 to 1);
        d: in bit_vector(0 to 3);
        y: out bit);
end mux_4_1;

architecture mux_4_1_behavioural_if_else of mux_4_1 is
 begin
   process(s,d)
      begin
         if s="00" then y<=d(0);
         elseif s="01" then y<=d(1);
         elseif s="10" then y<=d(2);
         else  y<=d(3);
        end if;
    end process;
end mux_4_1_behavioural_if_else;
```

VHDL Code For 4×1 Multiplexer Using CASE Statements (Behavioral Modelling)

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

entity multiplexer4_1 is
port (
      i0 : in std_logic;
      i1 : in std_logic;
      i2 : in std_logic;
      i3 : in std_logic;
     sel : in std_logic_vector(1 downto 0);
     bitout : out std_logic
     );
end multiplexer4_1;
```

VHDL Code For 4×1 Multiplexer Using CASE Statements (Contd..)

```
architecture Behavioral of multiplexer4_1 is
begin

process(i0,i1,i2,i3,sel)
begin
case sel is
  when "00" => bitout <= i0;
  when "01" => bitout <= i1;
  when "10" => bitout <= i2;
  when others => bitout <= i3;
end case;
end process;

end Behavioral;
```

The testbench code used for testing the code is given below:

```
LIBRARY ieee;
  USE ieee.std_logic_1164.ALL;

  ENTITY testbench IS
  END testbench;

  ARCHITECTURE behavior OF testbench IS
         SIGNAL i0,i1,i2,i3,bitout :  std_logic:='0';
         SIGNAL sel :  std_logic_vector(1 downto 0):="00";
  BEGIN
    UUT : entity work.multiplexer4_1 port map(i0,i1,i2,i3,sel,bitout);
```

The testbench code used for testing the code (Contd..)

- ```
  tb : PROCESS
    BEGIN
          i0<='1';
          i1<='0';
          i2<='1';
          i3<='0';
          sel <="00";
          wait for 2 ns;
          sel <="01";
          wait for 2 ns;
          sel <="10";
           wait for 2 ns;
           sel <="11";
            wait for 2 ns;
          --more input combinations can be given here.
    END PROCESS tb;

  END;
  ```
  ```

VHDL Code for D FF

```
1.  LIBRARY IEEE;
2.  USE IEEE.std_logic_1164.ALL;
3.  ENTITY dff IS
4.          PORT( d, clk : IN std_logic;
5.          PORT( q : OUT std_logic);
6.  END dff;
7.  ARCHITECTURE dff OF dff IS
8.  BEGIN
9.          PROCESS(clk)
10.         BEGIN
11.             IF ( clk = '1') AND ( clk'EVENT ) THEN
12.                 q <= d;
13.             END IF;
14.         END PROCESS;
15. END dff;
```

VHDL Code for D FF

```
library ieee;
use ieee.std_logic_1164.all;
entity DFF is
•     -- Signals are initialized to 0 by default.
•     -- To make QN a 1, it has to be initialized
  port ( D, CLK : in std_logic;
           Q : out std_logic;
           QN : out std_logic := '1');
end DFF;

architecture data_flip of DFF is
  begin
      process ( CLK )
        begin
          if (CLK = '1' and CLK'event ) then
             Q <= D after 10ns;
             QN <= not D after 10ns;
          end if;
       end process;
end data_flip;
```

D FF with preset and clear

```
1.  ENTITY dtype IS
2.      PORT(clk, d, clr, pre : IN std_logic;
3.      q, n_q : OUT std_logic);
4.  END dtype;
5.  ARCHITECTURE behav OF dtype IS
6.  SIGNAL temp_q : std_logic; -- internal signal
7.  BEGIN
8.      PROCESS (clk, clr, pre)
9.  BEGIN
10.     IF clr = '1' THEN -- clear operation
11.     temp_q <= '0';
12.     ELSIF pre = '1' THEN -- preset operation
13.     temp_q <= '1';
14.     ELSIF clk'EVENT AND clk = '1' THEN -- clock
15.     temp_q <= d;
16.     END IF;
17.     END PROCESS;
18.     q <= temp_q;
19.     n_q <= NOT temp_q;
20. END behav;
```

VHDL Code For 3 Bit Counter By Behavioral Modelling

```
library ieee;
use ieee.numeric_std.all;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter_3_bit is
   port(clock, reset: in bit;
            z: out unsigned(0 to 2));
end counter_3_bit;
```

VHDL Code For 3 Bit Counter By Behavioral Modelling (Contd..)

```
architecture behavioural of counter_3_bit is
 begin
   process(clock, reset)
    variable temp: unsigned(0 to 2):=(others=>'0');
   begin
   if reset='1' then temp:=(others=>'0');
    elseif clock='0' and clock'event then
    temp:=temp+1;
   end if;
  z<=temp;
end process;
end behavioural;
```

3 Bit Counter using D Flip Flop

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

entity Counter_3bit is
    Port ( CLK : in  STD_LOGIC;
           Count : out  STD_LOGIC_VECTOR (2 downto 0));
end Counter_3bit;

architecture Behavioral of Counter_3bit is
    signal cin : std_logic_vector(2 downto 0) :="000";
begin
    process(CLK)
    begin
        if(rising_edge(CLK)) then
            if(cin = "111") then
                cin <= "000";
            else
                cin <= cin + 1;
            end if;
        end if;

        Count <= cin;
    end process;
end Behavioral;
```

Full adder in Structural modelling using two Half adders and a n Or gate as components

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

entity fa_str is
Port ( a, b, cin : in STD_LOGIC;
          s, cout : out STD_LOGIC);
end fa_str;
```

Full adder in Structural modelling using two Half adders and a n Or gate as components (Contd..)

```
architecture fa_struct of fa_str is

component ha
port (a, b:in std_logic;
        s, c: out std_logic);
end component;

component or21
port (a, b: in std_logic;
        y: out std_logic);
end component;

signal t1,t2,t3 : std_logic;

begin
h1: ha port map (a,b,t2,t1);
h2: ha port map (t2,cin,s,t3);
h3: or21 port map (t1,t3,cout);
end fa_struct;
```

VHDL Code For Binary to Gray Code Converter By Data Flow Modelling

```
library ieee;
use ieee.std_logic_1164.all;

entity binary_to_gray is
  port(b0,b1,b2,b3: in bit;g0,g1,g2,g3: out bit);
end binary_to_gray;

architecture b_t_g of binary_to_gray is
  begin
    g3<=b3;
    g2<=b3 xor b2;
    g1<=b2 xor b1;
    g0<=b1 xor b0;
end b_t_g;
```