University Of Diyala
College Of Engineering
Computer Engineering Department

Modeling Combinational Logic in VHDL

Dr. Yasir Amer Abbas

Third stage

2017

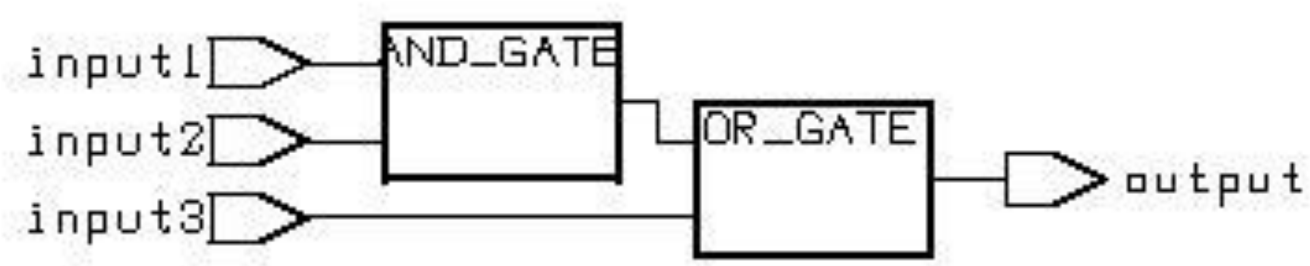
OR GATE

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
--------------------------------------
entity OR_ent is
    port(
        x: in  std_logic;
        y: in  std_logic;
        F: out std_logic );
    end OR_ent;
    -------------------------------------
Architecture OR_beh of OR_ent is
    begin
    F <= x OR y;
    end OR_beh;
```

AND GATE

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
----------------------------------------------------
Entity AND_ent is
Port
            ( x: in std_logic;
             y: in std_logic;
             F: out std_logic );
end AND_ent;
 ----------------------------------------------------
Architecture behav of AND_ent is
            Begin
                        F <= x AND y;
            end behav;
```

COMBINATIONAL LOGIC



Combinational Logic

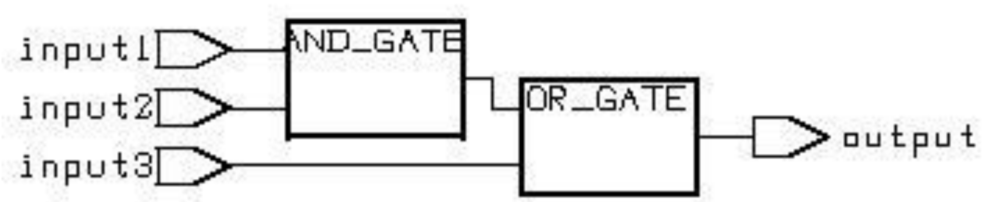
```
Library ieee;                    -- component #1
use ieee.std_logic_1164.all;
entity OR_GATE is
port(
        X: in std_logic;
        Y: in std_logic;
        F2: out std_logic );
end OR_GATE;
architecture behv of OR_GATE is
        Begin
        process(X,Y)
                begin
        F2 <= X OR Y;
        end process;
end behv;
```

Combinational Logic

```
Library ieee;                    -- component #2
use ieee.std_logic_1164.all;
entity AND_GATE is
 port(
             A: in std_logic;
             B: in std_logic;
             F1: out std_logic );
              end AND_GATE;
architecture behv of AND_GATE is
begin
             process(A,B)
                          begin
                          F1 <= A  AND B;
             end process;
end behv;
```

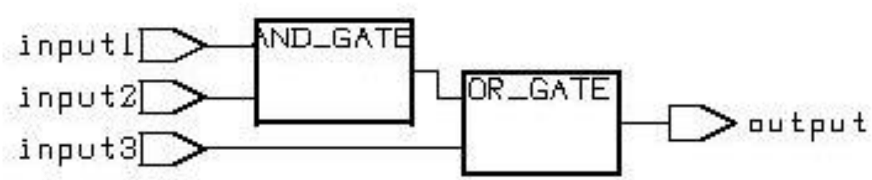
COMBINATIONAL LOGIC

```
library ieee; -- top level circuit
use ieee.std_logic_1164.all;
entity comb_ckt is
port(
        input1: in std_logic;
        input2: in std_logic;
        input3: in std_logic;
        output: out std_logic );
end comb_ckt;
architecture struct of comb_ckt is
component  AND_GATE is                          -- as entity ofAND_GATE
port(
        A: in std_logic;
        B: in std_logic;
        F1: out std_logic );
end component;
```

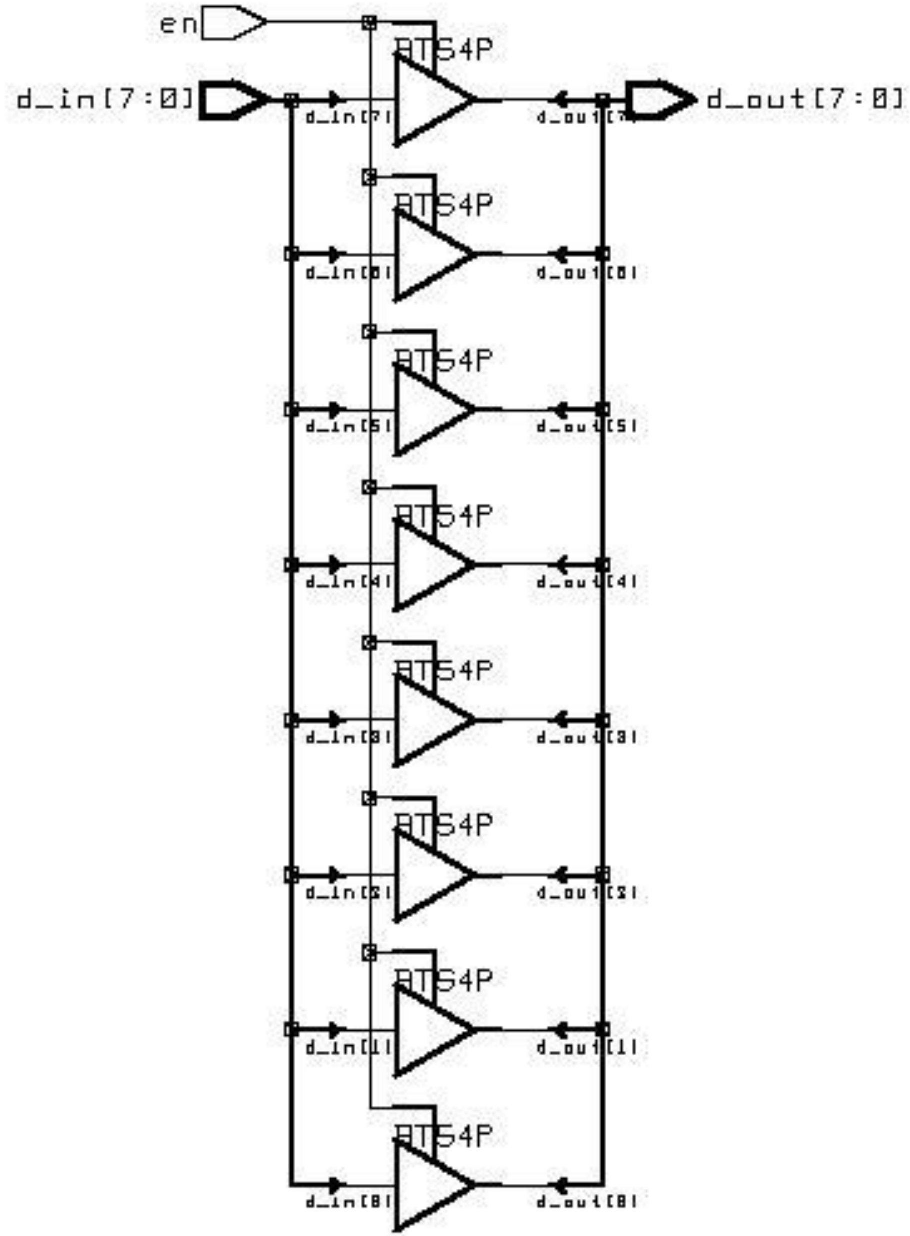


Combinational Logic

```
component OR_GATE is                    -- as entity of OR_GATE
port(
        X: in std_logic;
        Y: in std_logic;
        F2: out std_logic );
end component;
        signal wire: std_logic;    -- signal just like wire
        begin
        -- use sign "=>" to clarify the pin mapping
        Gate1: AND_GATE port map (A=>input1, B=>input2, F1=>wire);
        Gate2: OR_GATE port map (X=>wire, Y=>input3, F2=>output);
end struct;
```



TRI-STATE DRIVER

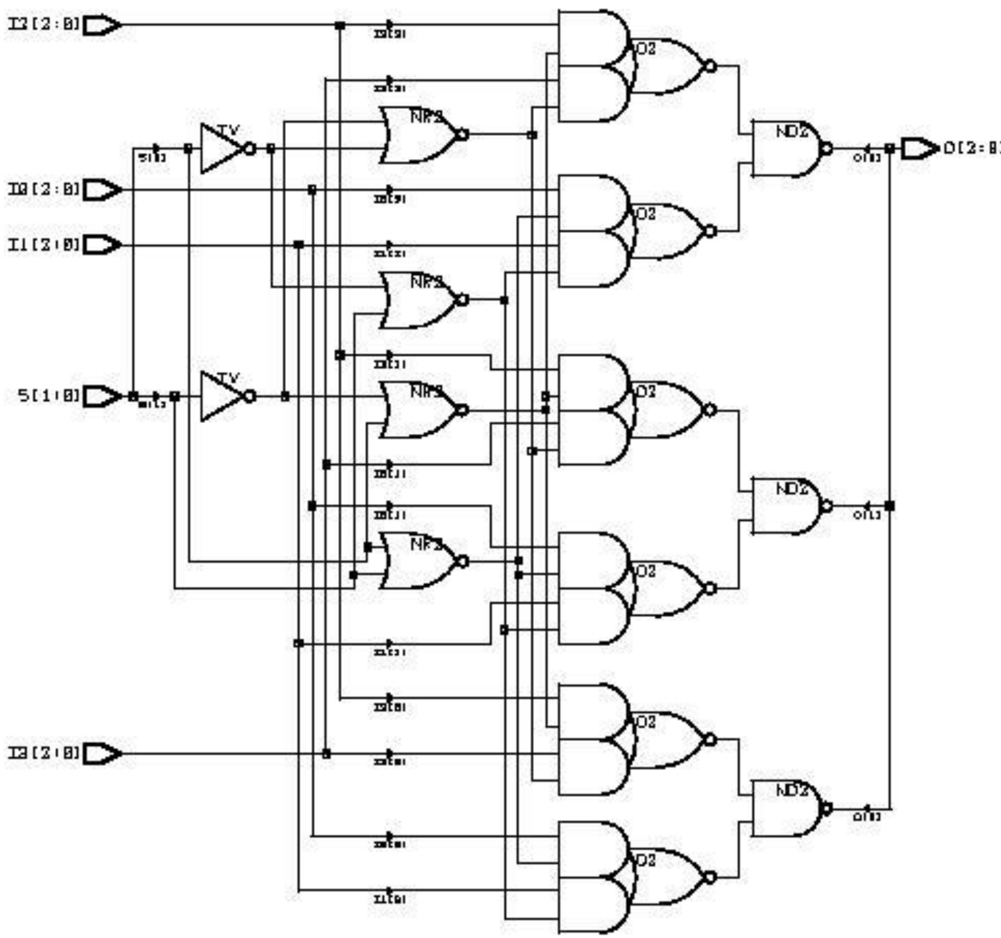


TRI-STATE DRIVER

```
library ieee;
use ieee.std_logic_1164.all;
entity tristate_dr is
port(
        d_in      : in      std_logic_vector(7 downto 0);
        en        : in      std_logic;
        d_out     : out     std_logic_vector(7 downto 0) );
        end tristate_dr;
Architecture behavior of tristate_dr is
        begin
                process(d_in, en)
                        begin
                        if en='1' then d_out <= d_in;
                        else -- array can be created simply by using vector
                        d_out <= "ZZZZZZZZ";
                        end if;
                end process;
end behavior;
```

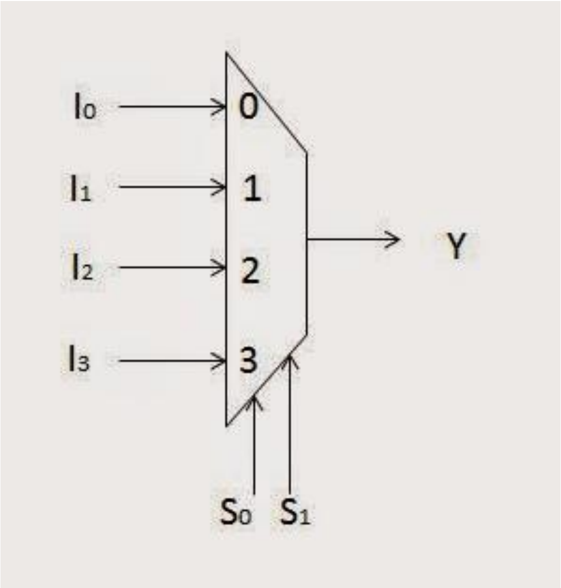
1.1 modeling with procedure statements

.



VHDL CODE FOR 4:1 MULTIPLEXOR

```
library ieee;
use ieee.std_logic_1164.all;
entity Mux is
port(
        I3: in std_logic_vector(2 downto 0);
        I2: in std_logic_vector(2 downto 0);
        I1: in std_logic_vector(2 downto 0);
        I0: in std_logic_vector(2 downto 0);
        S: in std_logic_vector(1 downto 0);
        O: out std_logic_vector(2 downto 0) );
end Mux;
architecture behv1 of Mux is
        begin
                process(I3,I2,I1,I0,S)
                begin -- use case statement
                case S is
                        when "00" => O <= I0;
                        when "01" => O <= I1;
                        when "10" => O <= I2;
                        when "11" => O <= I3;
                        when others => O <= "ZZZ";
                end case;
                end process;
end behv1;
```



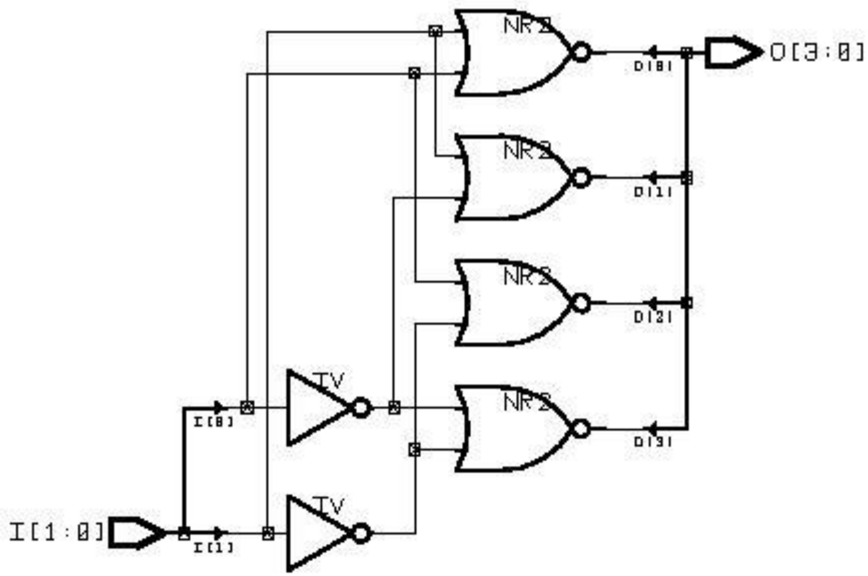
VHDL CODE FOR 4:1 MULTIPLEXOR

```
library ieee;
use ieee.std_logic_1164.all;
 ---------------------------------------------
entity Mux is
port(
                I3: in std_logic_vector(2 downto 0);
                I2: in std_logic_vector(2 downto 0);
                I1: in std_logic_vector(2 downto 0);
                I0: in std_logic_vector(2 downto 0);
                S: in std_logic_vector(1 downto 0);
                O: out std_logic_vector(2 downto 0) );
end Mux;
---------------------------------------------
Architecture
                behv2 of Mux is
                                begin -- use when.. else statement
                                                O <=        I0 when S="00" else
                                                            I1 when S="01" else
                                                            I2 when S="10" else
                                                            I3 when S="11" else
                                                                        "ZZZ";
end behv2;
```

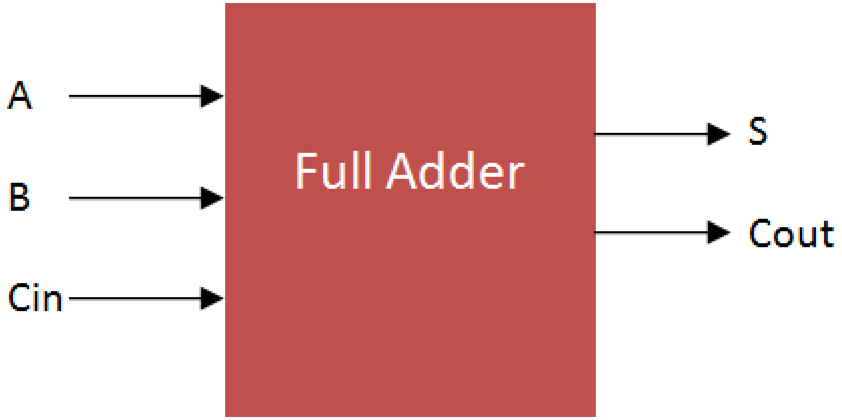
DECODER

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

entity DECODER is
port(
    I: in std_logic_vector(1 downto 0);
    O: out std_logic_vector(3 downto 0) );
    end DECODER;
    ----------------------------------------------
    architecture behv of DECODER is
        begin -- process statement
    process (I)
    begin -- use case statement
    case I is
        when "00" => O <= "0001";
        when "01" => O <= "0010";
        when "10" => O <= "0100";
        when "11" => O <= "1000";
        when others => O <= "XXXX";
        end case; end process;
    end behv;
```

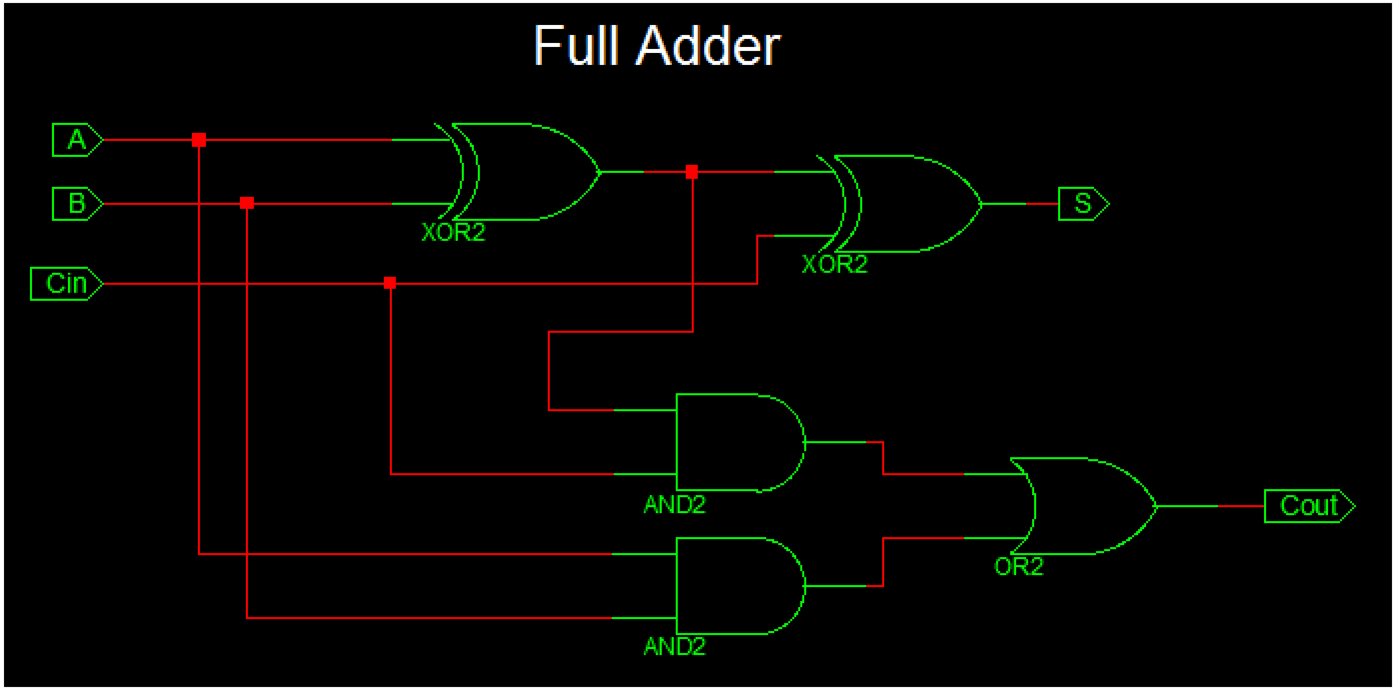


Full Adder



Cin	B	A	S	Cout
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

Full Adder



Full Adder

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;

Entity full_adder_vhdl_code is
 Port (
        A : in STD_LOGIC;
        B : in STD_LOGIC;
        Cin : in STD_LOGIC;
        S : out STD_LOGIC;
        Cout : out STD_LOGIC);
end full_adder_vhdl_code;

Architecture gate_level of full_adder_vhdl_code is
 begin

         S <= A XOR B XOR Cin ;
        Cout <= (A AND B) OR (Cin AND A) OR (Cin AND B) ;

end gate_level;
```

H.W

- Design Full Adder using Component by VHDL
- Design 8 bit adder using VHDL
- Design simple comparator has two n-bit inputs and three 1-bit outputs, using VHDL