

**MICROProgramming And architecture LABORATORY**

# **NAME- SAHIN NAYAK**

# **COURSE- MCA**

# **STREAM- MCA**

# **SEMESTER- 1**

# **SECTION- B**

# **ClASS ROLL- 53**

# **ENROLLMENT NO.- 12023006015086**

# **REGISTRATION NO.-**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Assignment No** | **Program Name** | **Date** | **Page No** | **Signature** |
| 1 | VHDL code for AND Gate |  | 1 |  |
| 2 | VHDL code for OR Gate |  | 2 |  |
| 3 | VHDL code for NOT Gate |  | 3 |  |
| 4 | VHDL code for NAND Gate |  | 4 |  |
| 5 | VHDL code for NOR Gate |  | 5 |  |
| 6 | VHDL code for XOR Gate |  | 6 |  |
| 7 | VHDL code for XNOR Gate |  | 7 |  |
| 8 | VHDL code for HALF ADDER |  | 8 |  |
| 9 | VHDL code for FULL ADDER |  | 9 |  |
| 10 | VHDL code for HALF SUBTRACTOR |  | 10 |  |
| 11 | VHDL code for FULL SUBTRACTOR |  | 11-12 |  |
| 12 | VHDL code for 4x1 MULTIPLEXER |  | 13-14 |  |
| 13 | VHDL code for 1x4 DEMULTIPLEXER |  | 15-16 |  |
| 14 | VHDL code for 4 to 2 ENCODER |  | 17-18 |  |
| 15 | VHDL code for 2 to 4 DECODER |  | 19-20 |  |

**Assignment 1 : VHDL code for AND Gate:**

Library IEEE;

use IEEE.std\_logic\_1164.all;

entity AND\_gate is

port(A : in std\_logic;

B : in std\_logic;

Y : out std\_logic);

end AND\_gate;

architecture andLogic of AND\_gate is

begin

Y <= A AND B;

end andLogic;

|  |  |
| --- | --- |
| **Truth Table** | **Output** |
| |  |  |  | | --- | --- | --- | | A | B | Y= A.B | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | |  |

**Assignment 2 : VHDL code for OR Gate:**

Library IEEE;

use IEEE.std\_logic\_1164.all;

entity OR\_gate is

port(A : in std\_logic;

B : in std\_logic;

Y : out std\_logic);

end OR\_gate;

architecture orLogic of OR\_gate is

begin

Y <= A OR B;

end orLogic;

|  |  |
| --- | --- |
| **Truth Table** | **Output** |
| |  |  |  | | --- | --- | --- | | A | B | Y= A + B | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 | |  |

**Assignment 3 : VHDL code for NOT Gate:**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity not\_gate is

port(A : in std\_logic;

Y : out std\_logic);

end not\_gate;

architecture notLogic of not\_gate is

begin

Y <= not(A) ;

end notLogic;

|  |  |
| --- | --- |
| **Truth Table** | **Output** |
| |  |  | | --- | --- | | A | Y=Ā | | 0 | 1 | | 1 | 0 | | C:\Users\nayak\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20231114_125343.png |

**Assignment 4 : VHDL code for NAND Gate:**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity nand\_gate is

port(A: in std\_logic;

B: in std\_logic;

Y: out std\_logic);

end nand\_gate;

architecture nandLogic of nand\_gate is

begin

Y <= not (A and B);

end nandLogic;

|  |
| --- |
| **Truth Table** |
| |  |  |  | | --- | --- | --- | | A | B | Y=(AB)c | | 0 | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | |
| **Output** | |
|  | |

**Assignment 5 : VHDL code for NOR Gate:**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity nor\_gate is

port(A: in std\_logic;

B: in std\_logic;

Y: out std\_logic);

end nor\_gate;

architecture norLogic of nor\_gate is

begin

Y <= not(A OR B);

end norLogic;

|  |  |
| --- | --- |
| **Truth Table** | **Output** |
| |  |  |  | | --- | --- | --- | | A | B | Y=(A+B)c | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 0 | |  |

**Assignment 6 : VHDL code for XOR Gate:**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity xor\_gate is

port(A: in std\_logic;

B: in std\_logic;

Y: out std\_logic);

end xor\_gate;

architecture xorLogic of xor\_gate is

begin

Y <= A xor B;

end xorLogic;

|  |  |
| --- | --- |
| **Truth Table** | **Output** |
| |  |  |  | | --- | --- | --- | | A | B | Y=  A ⊕B | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 0 | |  |

**Assignment 7 : VHDL code for XNOR Gate:**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity xnor\_gate is

port(A : in std\_logic;

B : in std\_logic;

Y : out std\_logic);

end xnor\_gate;

architecture xnorLogic of xnor\_gate is

begin

Y <= not(A xor B);

end xnorLogic;

|  |
| --- |
| **Truth Table** |
| |  |  |  | | --- | --- | --- | | A | B | Y=(A ⊕B)c | | 0 | 0 | 1 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | |

|  |
| --- |
| **Output** |
|  |

**Assignment 8 : VHDL code for HALF ADDER:**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity half\_adder is

port(A,B: in std\_logic;

sum,carryout: out std\_logic

);

end half\_adder;

architecture flow1 of half\_adder is

begin

sum<= A xor B;

carryout<=A and B;

end flow1;

|  |
| --- |
| **Truth Table** |
| |  |  |  |  | | --- | --- | --- | --- | | **Input** | | **Output** | | | **A** | **B** | **Sum** | **Carry** | | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 0 | | 1 | 0 | 1 | 0 | | 1 | 1 | 0 | 1 | |
| **Output** | |
| C:\Users\nayak\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20231114_133326.png | |

**Assignment 9 : VHDL code for 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;

|  |  |
| --- | --- |
| **Truth Table** | **Output** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Input** | |  | **Output** | | | **A** | **B** | **Cin** | **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 | |  |

**Assignment 10 : VHDL code for HALF SUBTRACTOR:**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity half\_sub is

port(A,B: in std\_logic;

diff,borrow : out std\_logic

);

end half\_sub;

architecture flow of half\_sub is

begin

diff<= A xor B;

borrow<=not(A) and B;

end flow;

|  |
| --- |
| **Truth Table** |
| |  |  |  |  | | --- | --- | --- | --- | | **Input** | | **Output** | | | **A** | **B** | **Diff** | **Borrow** | | 0 | 0 | 0 | 0 | | 0 | 1 | 1 | 1 | | 1 | 0 | 1 | 0 | | 1 | 1 | 0 | 0 | |
| **Output** | |
|  | |

**Assignment 11 : VHDL code for FULL SUBTRACTOR:**

Library IEEE;

use IEEE.std\_logic\_1164.all;

entity full\_sub is

port(A: in std\_logic;

B: in std\_logic;

Bin: in std\_logic;

Diff: out std\_logic;

Bout: out std\_logic);

end full\_sub;

architecture fullsubLogic of full\_sub is

begin

Diff <= ((A xor B) xor Bin);

Bout <= ((not A and B) or ((not(A xor B)) and Bin));

end fullsubLogic;

|  |
| --- |
| **Truth Table** |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Input** | |  | **Output** | | | **A** | **B** | **Borrow in(Bin)** | **Diff** | **Borrow**  **(Bout)** | | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 1 | 1 | | 0 | 1 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 1 | 0 | 0 | | 1 | 1 | 0 | 0 | 0 | | 1 | 1 | 1 | 1 | 1 | |
| **Output** |

|  |
| --- |
|  |

**Assignment 12 : VHDL code for 4x1 MULTIPLEXER:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity mux\_4to1 is

port(

A,B,C,D : in STD\_LOGIC;

S0,S1: in STD\_LOGIC;

Z: out STD\_LOGIC

);

end mux\_4to1;

architecture bhv of mux\_4to1 is

begin

process (A,B,C,D,S0,S1) is

begin

if (S0 ='0' and S1 = '0') then

Z <= A;

elsif (S0 ='1' and S1 = '0') then

Z <= B;

elsif (S0 ='0' and S1 = '1') then

Z <= C;

else

Z <= D;

end if;

end process;

end bhv;

|  |
| --- |
| **Truth Table** |
| |  |  |  | | --- | --- | --- | | **Input** | | **Output** | | **S1** | **S0** | **Z** | | 0 | 0 | A | | 0 | 1 | B | | 1 | 0 | C | | 1 | 1 | D | |
| **Output** |
|  |

**Assignment 13 : VHDL code for 1x4 DEMULTIPLEXER:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity demux\_1to4 is

port(

F : in STD\_LOGIC;

S0,S1: in STD\_LOGIC;

A,B,C,D: out STD\_LOGIC

);

end demux\_1to4;

architecture bhv of demux\_1to4 is

begin

process (F,S0,S1) is

begin

if (S0 ='0' and S1 = '0') then

A <= F;

elsif (S0 ='1' and S1 = '0') then

B <= F;

elsif (S0 ='0' and S1 = '1') then

C <= F;

else

D <= F;

end if;

end process;

end bhv;

|  |
| --- |
| **Truth Table** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Input** | **Selection Line** | | **Output** | | | | | **F** | **S0** | **S1** | **D** | **C** | **B** | **A** | | 1 | 0 | 0 | 0 | 0 | 0 | 1 | | 1 | 0 | 1 | 0 | 0 | 1 | 0 | | 1 | 1 | 0 | 0 | 1 | 0 | 0 | | 1 | 1 | 1 | 1 | 0 | 0 | 0 | | 0 | X | X | 0 | 0 | 0 | 0 | |
| **Output** |
|  |

**Assignment 14 : VHDL code for 4 to 2 ENCODER:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity encoder is

port(

a : in STD\_LOGIC\_VECTOR(3 downto 0);

b : out STD\_LOGIC\_VECTOR(1 downto 0)

);

end encoder;

architecture bhv of encoder is

begin

process(a)

begin

case a is

when "1000" => b <= "00"; when "0100" => b <= "01"; when "0010" => b <= "10"; when "0001" => b <= "11"; when others => b <= "ZZ";

end case;

end process;

end bhv;

|  |
| --- |
| **Truth Table** |
| |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Input** | | |  | **Output** | | | **A3** | **A2** | **A1** | **A0** | **B1** | **B0** | | 0 | 0 | 0 | 1 | 0 | 0 | | 0 | 0 | 1 | 0 | 0 | 1 | | 0 | 1 | 0 | 0 | 1 | 0 | | 1 | 0 | 0 | 0 | 1 | 1 | |
|  | **Output** |
|  |

**Assignment 15 : VHDL code for 2 to 4 DECODER:**

library IEEE;

use IEEE.STD\_LOGIC\_1164.all;

entity decoder is

port(

a : in STD\_LOGIC\_VECTOR(1 downto 0);

b : out STD\_LOGIC\_VECTOR(3 downto 0)

);

end decoder;

architecture bhv of decoder is

begin

process(a)

begin

case a is

when "00" => b <= "0001"; when "01" => b <= "0010"; when "10" => b <= "0100"; when "11" => b <= "1000";

end case;

end process;

end bhv;

|  |
| --- |
| **Truth Table** |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Input** | | |  |  | **Output** | | | **A1** | **A0** | **B3** | | **B2** | **B1** | **B0** | | 0 | 0 | 0 | | 0 | 0 | 1 | | 0 | 1 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | | 0 | 0 | 0 | |

|  |
| --- |
| **Output** |
|  |