-- AND Gate

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Saugat\_AND\_Gate IS

PORT (

A : IN STD\_LOGIC;

B : IN STD\_LOGIC;

Y : OUT STD\_LOGIC

);

END Saugat\_AND\_Gate;

ARCHITECTURE Behavior OF Saugat\_AND\_Gate IS

BEGIN

Y <= A AND B;

END Behavior;

-- OR Gate

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Saugat\_OR\_Gate IS

PORT (

A : IN STD\_LOGIC;

B : IN STD\_LOGIC;

Y : OUT STD\_LOGIC

);

END Saugat\_OR\_Gate;

ARCHITECTURE Behavior OF Saugat\_OR\_Gate IS

BEGIN

Y <= A OR B;

END Behavior;

-- NOT Gate

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Saugat \_NOT\_Gate IS

PORT (

A : IN STD\_LOGIC;

Y : OUT STD\_LOGIC

);

END Saugat\_NOT\_Gate;

ARCHITECTURE Behavior OF Saugat\_NOT\_Gate IS

BEGIN

Y <= NOT A;

END Behavior;

-- NOR Gate

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Saugat\_NOR\_Gate IS

PORT (

A : IN STD\_LOGIC;

B : IN STD\_LOGIC;

Y : OUT STD\_LOGIC

);

END Saugat\_NOR\_Gate;

ARCHITECTURE Behavior OF Saugat\_NOR\_Gate IS

BEGIN

Y <= NOT (A OR B);

END Behavior;

-- NAND Gate

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Saugat\_NAND\_Gate IS

PORT (

A : IN STD\_LOGIC;

B : IN STD\_LOGIC;

Y : OUT STD\_LOGIC

);

END Saugat\_NAND\_Gate;

ARCHITECTURE Behavior OF Saugat\_NAND\_Gate IS

BEGIN

Y <= NOT (A AND B);

END Behavior;

-- XOR Gate

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Saugat\_XOR\_Gate IS

PORT (

A : IN STD\_LOGIC;

B : IN STD\_LOGIC;

Y : OUT STD\_LOGIC

);

END Saugat\_XOR\_Gate;

ARCHITECTURE Behavior OF Saugat\_XOR\_Gate IS

BEGIN

Y <= A XOR B;

END Behavior;

-- XNOR Gate

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Saugat\_XNOR\_Gate IS

PORT (

A : IN STD\_LOGIC;

B : IN STD\_LOGIC;

Y : OUT STD\_LOGIC

);

END Saugat\_XNOR\_Gate;

ARCHITECTURE Behavior OF Saugat\_XNOR\_Gate IS

BEGIN

Y <= A XNOR B;

END Behavior;

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

ENTITY Saugat\_All\_Gates IS

PORT (

A : IN STD\_LOGIC;

B : IN STD\_LOGIC;

D : OUT STD\_LOGIC; -- AND Gate Output

E : OUT STD\_LOGIC; -- OR Gate Output

F : OUT STD\_LOGIC; -- NOR Gate Output

G : OUT STD\_LOGIC; -- NAND Gate Output

H : OUT STD\_LOGIC; -- XOR Gate Output

I : OUT STD\_LOGIC; -- XNOR Gate Output

J : OUT STD\_LOGIC -- NOT Gate Output (only uses A)

);

END Saugat\_All\_Gates;

ARCHITECTURE Behavior OF Saugat\_All\_Gates IS

BEGIN

D <= A AND B; -- AND Gate

E <= A OR B; -- OR Gate

F <= NOT (A OR B); -- NOR Gate

G <= NOT (A AND B); -- NAND Gate

H <= A XOR B; -- XOR Gate

I <= A XNOR B; -- XNOR Gate

J <= NOT A; -- NOT Gate (only uses input A)

END Behavior;

--Half Adder

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_Half\_Adder is

port (

A : in std\_logic;

B : in std\_logic;

Sum : out std\_logic;

Carry: out std\_logic

);

end Saugat\_Half\_Adder;

architecture Behavioral of Saugat\_Half\_Adder is

begin

process (A, B)

begin

Sum <= A xor B;

Carry <= A and B;

end process;

end Behavioral;

--Half Subtractor

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_Half\_Subtractor is

port (

A : in std\_logic;

B : in std\_logic;

Diff : out std\_logic;

Borrow : out std\_logic

);

end Saugat\_Half\_Subtractor;

architecture Behavioral of Saugat\_Half\_Subtractor is

begin

process (A, B)

begin

Diff <= A xor B;

Borrow <= (not A) and B;

end process;

end Behavioral;

--FULL ADDER

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_fulladder is

port (

a, b, c : in std\_logic;

s, cy : out std\_logic

);

end Saugat\_fulladder;

architecture beh of Saugat\_fulladder is

begin

s <= (a xor b) xor c;

cy <= (a and b) or (b and c) or (c and a);

end beh;

--FULL Subtractor

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_Full\_Subtractor is

port (

A : in std\_logic;

B : in std\_logic;

Bin : in std\_logic;

Diff : out std\_logic;

Bout : out std\_logic

);

end Saugat\_Full\_Subtractor;

architecture Behavioral of Saugat\_Full\_Subtractor is

begin

process (A, B, Bin)

begin

-- Difference = A - B - Bin

Diff <= A xor B xor Bin;

-- Borrow out logic

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

end process;

end Behavioral;

--OVERFLOW

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY Saugat\_over\_flow IS

PORT (

a : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

b : IN STD\_LOGIC\_VECTOR(3 DOWNTO 0);

overflow : OUT STD\_LOGIC

);

END ENTITY Saugat\_over\_flow;

ARCHITECTURE Behavioral OF Saugat\_over\_flow IS

BEGIN

PROCESS (a, b)

VARIABLE sum : STD\_LOGIC\_VECTOR(4 DOWNTO 0);

BEGIN

sum := ('0' & a) + ('0' & b);

IF sum(4) = '1' THEN

overflow <= '1';

ELSE

overflow <= '0';

END IF;

END PROCESS;

END ARCHITECTURE Behavioral;

--BAS

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_unsigned.ALL;

ENTITY Saugat\_B\_A\_S IS

PORT (

A : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

B : IN STD\_LOGIC\_VECTOR (3 DOWNTO 0);

Mode : IN STD\_LOGIC;

Sum : OUT STD\_LOGIC\_VECTOR (3 DOWNTO 0);

Carry : OUT STD\_LOGIC

);

END Saugat\_B\_A\_S;

ARCHITECTURE Behavioral OF Saugat\_B\_A\_S IS

SIGNAL B\_twos\_complement : STD\_LOGIC\_VECTOR (3 DOWNTO 0);

BEGIN

B\_twos\_complement <= (B XOR (Mode & Mode & Mode & Mode)) + Mode;

PROCESS (A, B\_twos\_complement, Mode)

VARIABLE Temp : STD\_LOGIC\_VECTOR (4 DOWNTO 0);

BEGIN

Temp := ('0' & A) + ('0' & B\_twos\_complement);

Sum <= Temp(3 DOWNTO 0);

Carry <= Temp(4);

END PROCESS;

END Behavioral;

--ALU

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity Saugat\_A\_L\_U is

port (

A, B : in std\_logic\_vector(3 downto 0);

Op : in std\_logic\_vector(2 downto 0);

Result : out std\_logic\_vector(3 downto 0);

Zero : out std\_logic ); end Saugat\_A\_L\_U;

architecture Behavioral of Saugat\_A\_L\_U is

begin process (A, B, Op) variable temp\_result : std\_logic\_vector(3 downto 0);

begin case Op is when "000" =>

temp\_result := std\_logic\_vector(unsigned(A) + unsigned(B));

when "001" =>

temp\_result := std\_logic\_vector(unsigned(A) - unsigned(B));

when "010" =>

temp\_result := A and B;

when "011" =>

temp\_result := A or B;

when "100" =>

temp\_result := A xor B;

when others => temp\_result := (others => '0'); end case;

Result <= temp\_result; if temp\_result = "0000" then

Zero <= '1'; else

Zero <= '0'; end if;

end process;

end Behavioral;

--CU

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_control\_unit is

port (

input1 : in std\_logic;

input2 : in std\_logic;

ctrl : in std\_logic;

output1 : out std\_logic;

output2 : out std\_logic );

end Saugat\_control\_unit;

architecture behavioral of Saugat\_control\_unit is begin

process (input1, input2, ctrl) begin

case ctrl is when '0' =>

output1 <= '0'; -- No need for redundant if-else

if input2 = '1' then

output2 <= '1';

else output2 <= '0';

end if;

when '1' =>

output1 <= input1 and input2;

output2 <= input1 or input2;

when others =>

output1 <= '0';

output2 <= '0';

end case;

end process;

end behavioral;

--Mapping

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_mping is

port (

clk : in std\_logic;

reset : in std\_logic;

enable : in std\_logic;

data\_in : in std\_logic\_vector(7 downto 0);

data\_out : out std\_logic\_vector(7 downto 0) );

end entity Saugat\_mping;

architecture Behavioral of Saugat\_mping is

signal reg\_data : std\_logic\_vector(7 downto 0);

begin

process (clk, reset)

begin

if reset = '1' then

reg\_data <= (others => '0');

elsif rising\_edge(clk) then

if enable = '1' then

reg\_data <= data\_in;

end if;

end if;

end process;

data\_out <= reg\_data;

end architecture Behavioral;

--PG

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity Saugat\_PG is

port(

data\_in : in std\_logic\_vector(7 downto 0);

parity\_type : in std\_logic;

parity\_bit : out std\_logic

);

end Saugat\_PG;

architecture behavioral of Saugat\_PG is

begin

process (data\_in, parity\_type)

variable temp : std\_logic;

begin

temp := '0';

for i in data\_in'range loop

temp := temp xor data\_in(i);

end loop;

if parity\_type = '1' then -- Even parity

parity\_bit <= not temp;

else -- Odd parity

parity\_bit <= temp;

end if;

end process;

end behavioral;

--PC

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_PC is

port(

data\_in : in std\_logic\_vector(7 downto 0);

parity\_bit : in std\_logic;

parity\_type : in std\_logic; -- '1' for even, '0' for odd

error : out std\_logic

);

end Saugat\_PC;

architecture behavioral of Saugat\_PC is

begin

process(data\_in, parity\_bit, parity\_type)

variable temp : std\_logic;

begin

temp := parity\_bit;

for i in data\_in'range loop

temp := temp xor data\_in(i);

end loop;

if temp = parity\_type then

error <= '0'; -- No error

else

error <= '1'; -- Error detected

end if;

end process;

end behavioral;

--Encoder

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_encoder42 is

port (p: in std\_logic\_vector (3 downto 0);

y : out std\_logic\_vector ( 1 downto 0 ) );

end Saugat\_encoder42;

architecture behavioral of Saugat\_encoder42 is

begin

process (p)

begin

case p is

when "0001" =>y<="00";

when "0010" =>y<="01";

when "0100" =>y<="10";

when "1000" =>y<="11";

when others => y <= "UU";

end case;

end process;

end behavioral;

--Decoder

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_decoder2to4 is

port (

A : in std\_logic\_vector(1 downto 0);

D : out std\_logic\_vector(3 downto 0)

);

end Saugat\_decoder2to4;

architecture beh of Saugat\_decoder2to4 is

begin

with A select

D <= "0001" when "00",

"0010" when "01",

"0100" when "10",

"1000" when "11",

"0000" when others;

end beh;

--MUX

library ieee;

use ieee.std\_logic\_1164.all;

entity Saugat\_mux21 is

port (

a, b : in std\_logic;

s : in std\_logic;

y : out std\_logic

);

end Saugat\_mux21;

architecture beh of Saugat\_mux21 is

begin

process (a, b, s)

begin

case s is

when '0' => y <= a;

when '1' => y <= b;

when others => y <= 'U';

end case;

end process;

end beh;