-- Combined basic logic gates

library ieee;

use ieee.std\_logic\_1164.all;

entity ayush\_basicgates is

port(a,b: in std\_logic;

c,d,e,f,g,h,i: out std\_logic);

end ayush\_basicgates;

architecture dataflow of ayush\_basicgates is

begin

c<= a and b;

d<= a or b;

e<= not b;

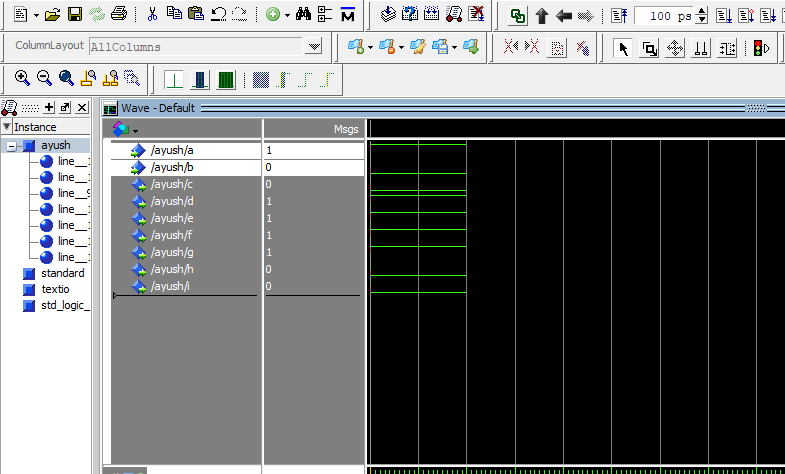
f<= a xor b;

g<= a nand b;

h<= not(a xor b);

i<= a nor b;

end dataflow;



-- half adder

library ieee;

use ieee.std\_logic\_1164.all;

entity ayush\_halfadder is

port(a: in std\_logic; b: in std\_logic;

s: out std\_logic,cy: out std\_logic);

end ayush\_halfadder;

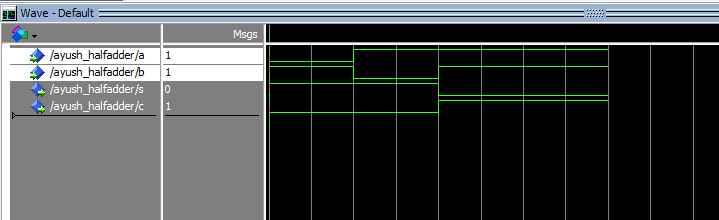
architecture dataflow of ayush\_halfadder is

begin

s<= a xor b;

c<= a and b;

end dataflow;



-- Full adder

library ieee;

use ieee.std\_logic\_1164.all;

entity ayush\_fulladder is

port(a,b,c: in std\_logic;

s,cy: out std\_logic);

end ayush\_fulladder;

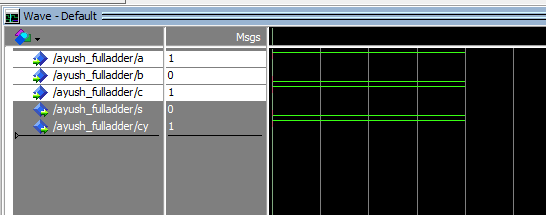
architecture dataflow of ayush\_fulladder is

begin

s<= (a xor b) xor c;

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

end dataflow;



-- Binary Adder Subtractor

LIBRARY ieee;

USE ieee.STD\_LOGIC\_1164.ALL;

USE ieee.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY ayush\_bas 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 ayush\_bas;

ARCHITECTURE BEHAVIORAL OF ayush\_bas IS

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

BEGIN

B\_twos\_COMPLIMENT <= (B XOR (MODE & MODE & MODE & MODE ))+MODE;

PROCESS(A,B\_twos\_COMPLIMENT,MODE)

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

BEGIN

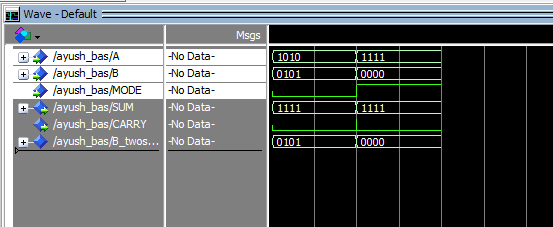
TEMP :=('0' & A)+('0' & B\_twos\_COMPLIMENT);

SUM<=TEMP(3 DOWNTO 0);

CARRY<= TEMP(4);

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 ayush\_overflow is

Port (a : in std\_logic\_vector (3 downto 0);

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

overflow : out STD\_LOGIC);

end entity ayush\_overflow;

architecture Behavioral of ayush\_overflow is

begin

process(a,b)

variable sum: std\_logic\_vector(4 downto 0);

begin

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

if sum(sum'high)='1'

then overflow <='1';

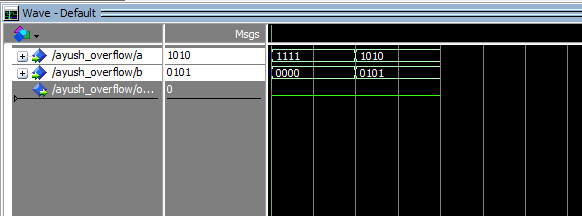
else

overflow <='0';

end if;

end process;

end architecture Behavioral;



-- ALU

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity ayush\_alu 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 ayush\_alu;

architecture Behavioral of ayush\_alu 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';

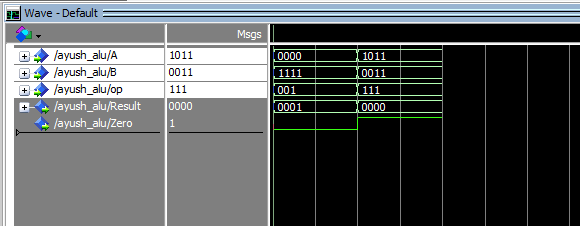
else

Zero <='0';

end if;

end process;

end Behavioral;



-- Control unit

library ieee;

use ieee.std\_logic\_1164.all;

entity ayush\_cu is

port(

input1 : in std\_logic;

input2 : in std\_logic;

ctrl : in std\_logic;

output1 : out std\_logic;

output2 : out std\_logic

);

end ayush\_cu;

architecture behavioral of ayush\_cu is

begin

process (input1, input2,ctrl)

begin

case ctrl is

when '0' =>

if input1 = '1' then

output1 <= '1';

else

output1 <= '0';

end if;

if input2 ='1' then

output2 <= '1';

else

output2 <= '0';

end if;

when '1' =>

output1 <= input1 and input2;

output2 <= input1 or input2;

when others =>

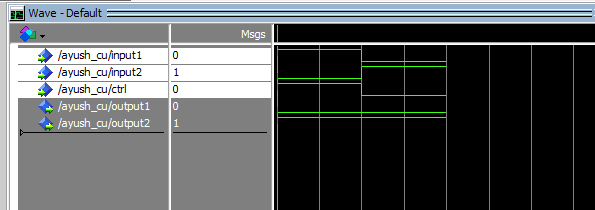
output1 <= '0';

output2 <= '0';

end case;

end process;

end behavioral;



-- Mapping

library ieee;

use ieee.std\_logic\_1164.all;

entity ayush\_mapping 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 ayush\_mapping;

architecture behavioral of ayush\_mapping 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 ;

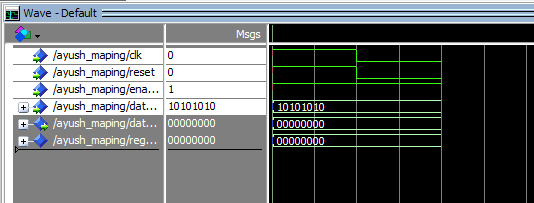
end if;

end if;

end process;

data\_out <= reg\_data;

end architecture behavioral;



-- Parity generator

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity ayush\_pg is

port(

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

parity\_type : in std\_logic;

parity\_bit : out std\_logic

);

end ayush\_pg;

architecture behavioral of ayush\_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

parity\_bit <= not temp;

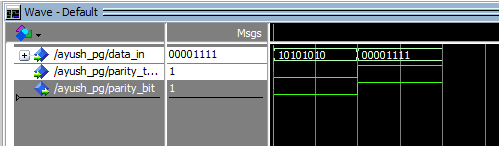
else

parity\_bit <= temp;

end if;

end process;

end behavioral;



-- Parity checker

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity ayush\_pc is

port(

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

parity\_type : in std\_logic;

received\_parity : in std\_logic;

error\_flag : out std\_logic

);

end ayush\_pc;

architecture behavioral of ayush\_pc is

begin

process (data\_in,received\_parity, parity\_type)

variable temp\_check : std\_logic;

begin

temp\_check:= '0';

for i in data\_in'range loop

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

end loop;

temp\_check := temp\_check xor received\_parity;

if (parity\_type = '1' and temp\_check = '0') or (parity\_type = '0' and temp\_check = '1') then

error\_flag <= '1';

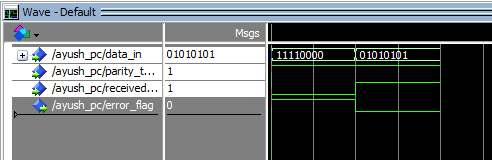
else

error\_flag <= '0';

end if;

end process;

end behavioral;



-- Encoder

library ieee;

use ieee.std\_logic\_1164.all;

entity ayush\_en is

port (

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

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

);

end ayush\_en;

architecture behavioral of ayush\_en is

begin

process(p)

begin

case p is

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

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

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

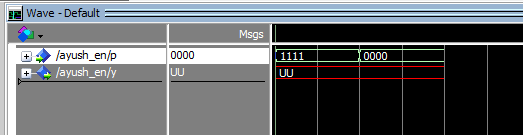
when "1000" => y <= "11";

when others => y <= "UU";

end case;

end process;

end behavioral;



-- Decoder

library ieee;

use ieee.std\_logic\_1164.all;

entity ayush\_de is

port (

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

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

);

end ayush\_de;

architecture behavioral of ayush\_de is

begin

with A select

D <= "0001" when "00",

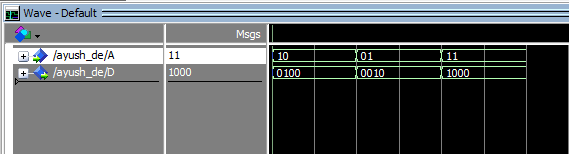
"0010" when "01",

"0100" when "10",

"1000" when "11",

"0000" when others;

end behavioral;



- Multiplexer

library ieee;

use ieee.std\_logic\_1164.all;

entity ayush\_mux is

port(a,b:in std\_logic;

s:in std\_logic;

y:out std\_logic);

end ayush\_mux;

architecture beh of ayush\_mux 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 behavioral;

