

**FACULTY OF ENGINEERING AND TECHNOLOGY**

**ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT**

**ADVANCED DIGITAL DESIGN ENCS3310**

**COURSE PROJECT**

**Prepared by:**

**Shorooq Najjar ID:1192415**

**Instructor: Dr. Abdallatif Abuissa**

**Section:1**

**Table of contents:**

**1.Introduction & background ………………………………………………………………………………………………….1**

**2.Design Philosophy…………………………………………………………………………………………………………………2**

**2.1 Stage1………………………………………………………………………………………………………………………....2**

**2.2 Stage2…………………………………………………………………………………………………………………….……3**

**3.Results………………………………………………………………………………………………………………………………….5**

**3.1 Stage 1…………………………………………………………………………………………………………………………..5**

**3.2 Stage 2…………………………………………………………………………………………………………………………..6**

**4.Conclusion & Future Work…………………………………………………………………………………………………….6**

**5.Appendix……………………………………………………………………………………………………………………………...7**

**5.1. D Flip Flop code ……………………………………………………………………………………………………….7**

**5.2. Flip Flop Register code…………………………………………………………………………………………..…8**

**5.3.Carry Look Ahead Adder code…………………………………………………………………………………...9**

**5.4.Comparator Stage 1 code………………………………………………………………………………………....12**

**5.5.Test-bench Stage 1 code………………………………………………………………………………………..….14**

**5.6. Magnitude Comparator code ……………………………………………………………………………….….19**

**5.7.Comparator Stage 2 code………………………………………………………………………………………..…21**

**5.8. Test-bench Stage 2 code………………………………………………………………………………………..….23**

**Table of Figures**

**Figure 2.1:** **Stage 1 Comparator……………………………………………………………………………………………………3**

**Figure 2.2** **Stage 2 Comparator ......................................................................................................4**

**Figure 3.1: Stage 1 simulation. ......................................................................................................5**

**Figure 3.1.2: Stage 1 simulation with error.....................................................................................5**

**Figure 3.2: Stage 2 simulation.........................................................................................................5**

**Figure 3.2.1: Stage 2 simulation with error….. ................................................................................6**

**1.Introduction & background:**

The concept of comparisons are those circles that compare two inputs, for example the comparison between the two inputs A & B and give the results in form (a is greater than b, a is less than b, a is equal to b). There are many different ways to compare the two inputs. Mostly circuits which are used for the comparison are magnitude comparator and adder/ subtractor circuits.

There are two types of comparators one is signed comparator and other is unsigned comparator. In unsigned comparators inputs are taken as unsigned binary values whereas in signed comparators inputs are taken as 2’s complement representation of numbers. Task of this project is to design the signed number comparators which compare the 2’scomplement representation of numbers.

The first stage in this project is to build the comparison using the adder circuit , In this stage there are two inputs of adder which is A and B. A input is connected with the adder as it is but B input is connected with the adder with invert values and carry input of adder is ‘1’. This whole connection makes the adder as subtractorand do the subtraction of “A-B”. After that by using the result of subtraction two inputs can be compare.

The second stage in this project is to design the comparator by using the magnitude comparator. Magnitude comparator check that which input is greater or lessor or both inputs are equal. By using the magnitude comparator outputs and inputs MSB bits which represents the sign of the input, signed comparators are design.

**Page1**

**2.Design Philosophy**

**2.1.Stage 1**

In this design first a adder of 8 bit designed which is carry look ahead adder. Than adder is used in the comparator circuit so that it can perform the subtraction “A-B”. If the result of subtraction is “0” than its mean both inputs values are equal so that output “AEQB” will be ‘1’ and others output will be ‘0’. If output of the subtraction result “A-B” not zero than MSB bits of inputs A and B which represent the sign and subtraction output MSB bit passes through some logic gates which determine that A is greater than B or A is less than B. Below Boolean equations determine the outputs of the comparators. In the Boolean equations A(7) represent the MSB bit of A input, B(7) represent the MSB bit of B input and R(7) represent the MSB bit of “A-B” subtraction result.

**AGTB = AEQB’ A(7)’ R(7) + AEQB’ A(7)’ B(7) + AEQB’ B(7) R(7)’**

**AEQB = [R(7) + R(6) + R(5) + R(4) + R(3) + R(2) + R(1) + R(0)]’**

**ALTB = AEQB’ B(7)’ R(7) + AEQB’ A(7) B(7)’ + AEQB’ A(7) R(7)**

At the inputs A and B two registers are connected which values are updated when the clock positive edge comes and at the output D flip flops are connected which values are also updated when clock positive edge comes. Result of comparison comes after one clock cycle when inputs are updated in the registers. By using the reset registers and flip flops values will be reset to ‘0’. Critical Path Delay of stage 1 comparator = 54 ns . So clock of 60 ns period is best for the registers and flip flops.

**Clock needed = 𝟏 / 𝟓𝟒 𝒏𝒔 = 18.52 MHz**

**Below is the block diagram of comparator by using carry look ahead adder:**

**Page2**

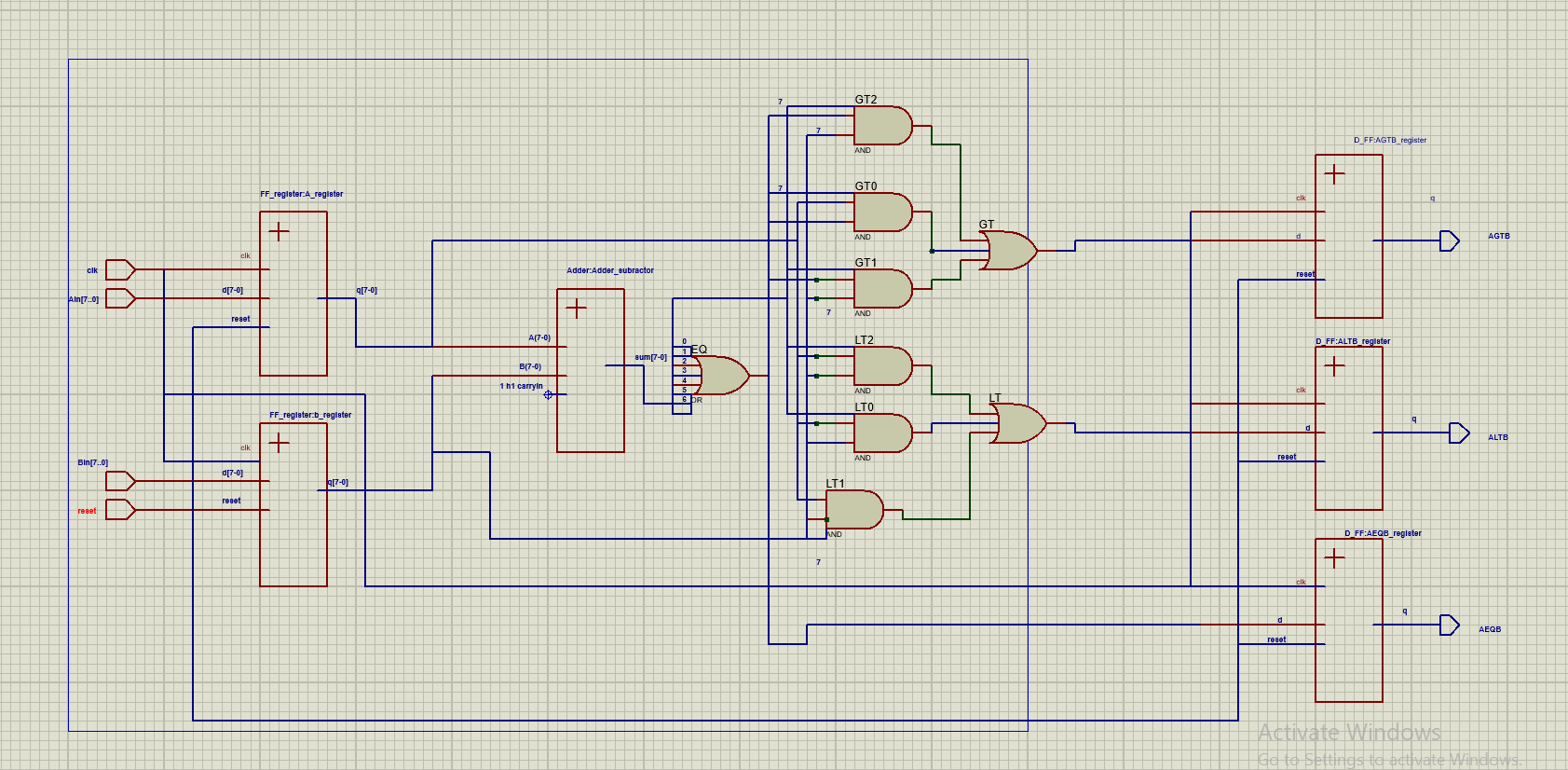


Figure 2.1 Stage 1 Comparator

**Stage 2:**

Same as the stage 1 this stage comparators input is also connected with the registers and outputs

are connected with the flip flops. Outputs are updated after the one clock cycle when inputs are

inserted in the registers. In this design a magnitude comparator of 8 bits is used which compare

the inputs and check that which input magnitude is greater. Magnitude comparator gives the output

in the form of A greater than B (A\_GT\_B), A equal to B (A\_EQ\_B) and A less than B (A\_LT\_B).

By using these three outputs of magnitude comparator and sign bits of input A and B which are

A(7) and B(7) signed comparator outputs are determine. Below Boolean equations uses these

inputs and compare the signed numbers and gives the result of comparison of A and B signed 2’s

complement representation numbers.

**AGTB = A(7)’ A\_GT\_B + B(7) A\_GT\_B + A(7)’ B(7)**

**EQ = A\_EQ\_B**

**ALTB = B(7)’ A\_LT\_B + A(7) A\_LT\_B + A(7) B(7)’**

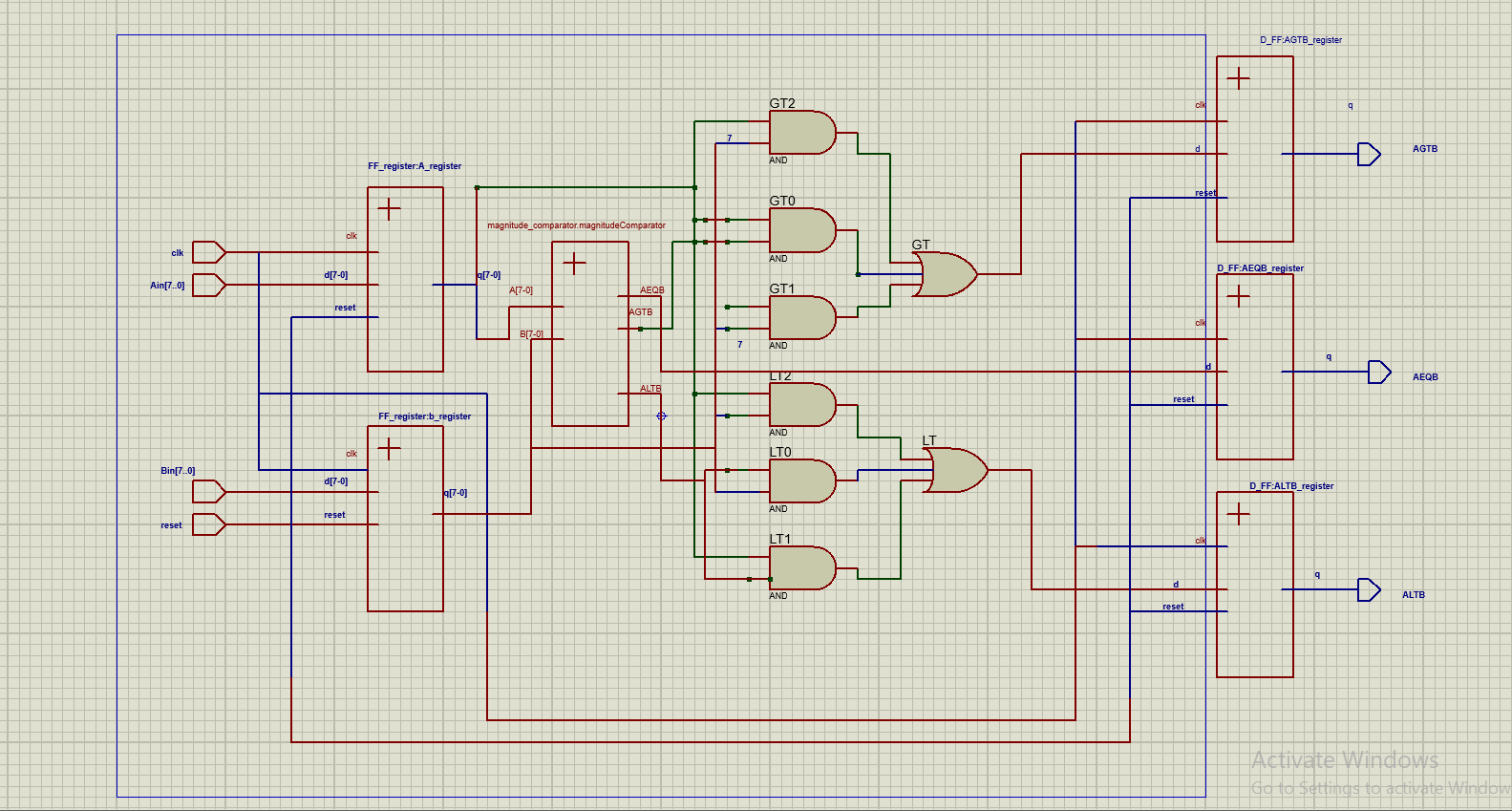
**Page3**

Critical Path Delay of stage 1 comparator = 48 ns

So clock of 50 ns period is best for the registers and flip flops

**Clock needed = 𝟏/𝟒𝟖 𝒏𝒔= 20.83 MHz**

**Below is the block diagram of comparator by using magnitude comparator.**

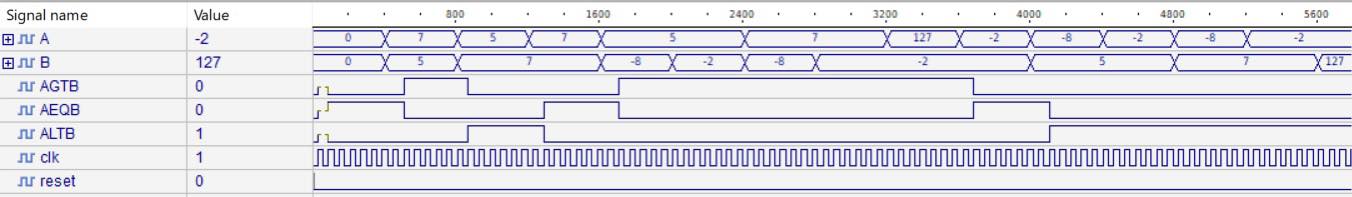
****

**Figure2.2 Stage 2 Comparator**

**Page4**

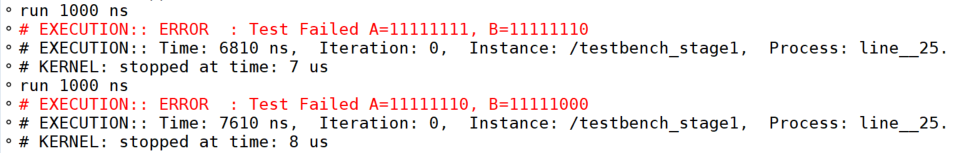
**Results**

**3.1 Stage 1**

**Correct simulation: **

**Figure 3.1 Stage 1 Simulation**

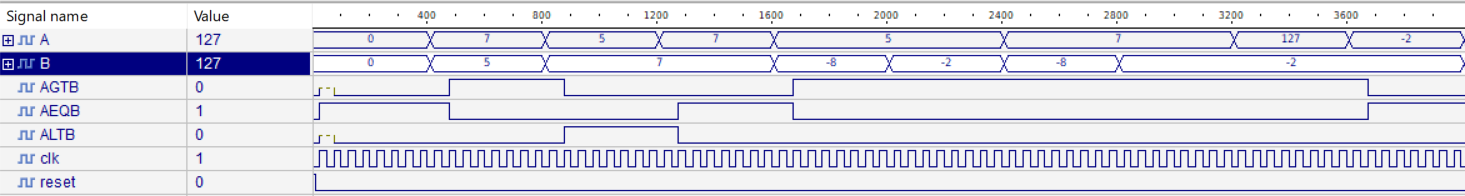
**Simulation with Error:**

****

**Figure 3.1.2 Stage 1 Simulation with error**

**3.2 Stage2**

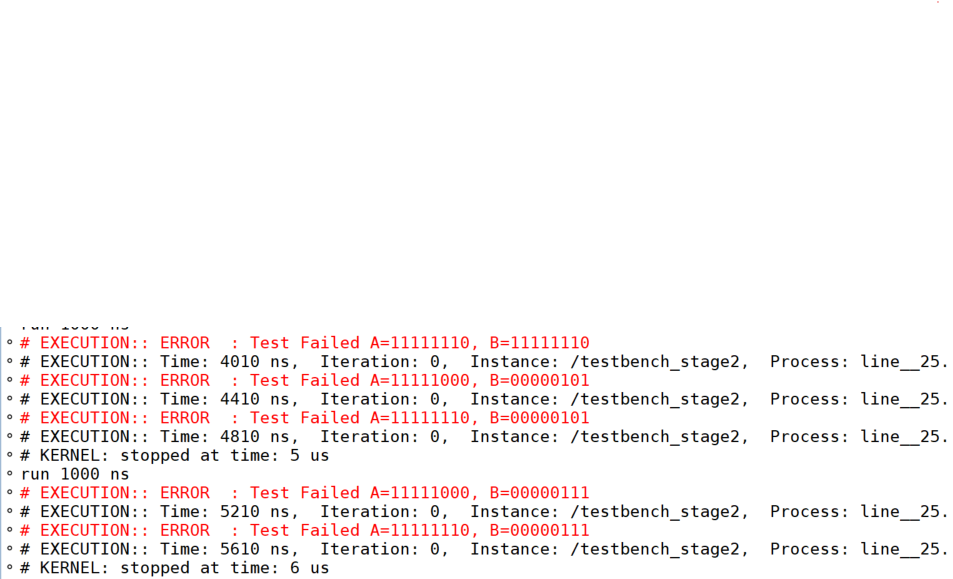
**Correct Simulation:**

****

**Figure 3.2 Stage 2 Simulation**

**Page5**

**Simulation with Error:**

****

**Figure 3.2.1 Stage 2 Simulation with error**

**4.Conclusion & Future Work**

This task is to design the signed 2’s complement representation numbers comparators by using two different design one is by using adder circuit and one is by using magnitude comparator circuit. This is designing task and it is very interesting for the learning of the digital design circuits. Also through this project we learned that how to calculate the critical path of the digital circuits. For the future work we can reduce the critical path by some logics if possible so that comparator gives the output fastly . We can also increases the inputs bits of the comparator for the future work so that we can compare bigger values by using signed comparators.

**Page6**

**5.Appendix**

**5.1 D Flip Flop**

library ieee;

use ieee.std\_logic\_1164.all;

entity D\_FF is

port (d : in std\_logic;

clk : in std\_logic;

reset : in std\_logic;

q : out std\_logic);

end D\_FF;

architecture behavior of D\_FF is

begin

process(clk,reset)

begin

if reset='1' then

q<='0';

elsif rising\_edge(clk) then

q <= d;

end if;

end process;

end behavior;

**Page7**

**5.2 Flip Flop Register**

library ieee;

use ieee.std\_logic\_1164.all;

entity FF\_register is

port (d : in std\_logic\_vector (7 downto 0);

clk : in std\_logic;

reset : in std\_logic;

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

end FF\_register;

architecture behavior of FF\_register is

begin

process(clk,reset)

begin

if reset='1' then

q<="00000000";

elsif rising\_edge(clk) then

q <= d;

end if;

end process;

end behavior;

**Page8**

**5.3 Carry Look Ahead Adder**

library IEEE;

use IEEE.std\_logic\_1164.all;

entity Adder is

port (A : in std\_logic\_vector (7 downto 0);

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

CarryIn : in std\_logic;

CarryOut : out std\_logic;

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

end Adder;

architecture arch of Adder is

signal P,G : std\_logic\_vector (7 downto 0);

signal C : std\_logic\_vector (8 downto 0);

signal

c1,c20,c21,c30,c31,c32,c40,c41,c42,c43,c50,c51,c52,c53,c54,c60,c61,c62,c63,c64,c65,c70,c71,c72,c73,c74,c75,c76

,c80,c81,c82,c83,c84,c85,c86,c87 : std\_logic;

begin

P <= A xor B after 12 ns;

G <= A and B after 7 ns;

C(0) <= CarryIn;

c1 <= (P(0) and CarryIn) after 7 ns;

C(1) <= G(0) or c1 after 7 ns;

**Page9**

c20 <= (P(1) and G(0)) after 7 ns;

c21 <= (P(1) and P(0) and CarryIn) after 7 ns;

C(2) <= G(1) or c20 or c21 after 7 ns;

c30 <= (P(2) and G(1)) after 7 ns;

c31 <= (P(2) and P(1) and G(0)) after 7 ns;

c32 <= (P(2) and P(1) and P(0) and CarryIn) after 7 ns;

C(3) <= G(2) or c30 or c31 or c32 after 7 ns;

c40 <= (P(3) and G(2)) after 7 ns;

c41 <= (P(3) and P(2) and G(1)) after 7 ns;

c42 <= (P(3) and P(2) and P(1) and G(0)) after 7 ns;

c43 <= (P(3) and P(2) and P(1) and P(0) and CarryIn) after 7 ns;

C(4) <= G(3) or c40 or c41 or c42 or c43 after 7 ns;

c50 <= (P(4) and G(3)) after 7 ns;

c51 <= (P(4) and P(3) and G(2)) after 7 ns;

c52 <= (P(4) and P(3) and P(2) and G(1)) after 7 ns;

c53 <= (P(4) and P(3) and P(2) and P(1) and G(0)) after 7 ns;

c54 <= (P(4) and P(3) and P(2) and P(1) and P(0) and CarryIn) after 7 ns;

C(5) <= G(4) or c50 or c51 or c52 or c53 or c54 after 7 ns;

c60 <= (P(5) and G(4)) after 7 ns;

c61 <= (P(5) and P(4) and G(3)) after 7 ns;

c62 <= (P(5) and P(4) and P(3) and G(2)) after 7 ns;

c63 <= (P(5) and P(4) and P(3) and P(2) and G(1)) after 7 ns;

c64 <= (P(5) and P(4) and P(3) and P(2) and P(1) and G(0)) after 7 ns;

**Page10**

c65 <= (P(5) and P(4) and P(3) and P(2) and P(1) and P(0) and CarryIn) after 7 ns;

C(6) <= G(5) or c60 or c61 or c62 or c63 or c64 or c65 after 7 ns;

c70 <= (P(6) and G(5)) after 7 ns;

c71 <= (P(6) and P(5) and G(4)) after 7 ns;

c72 <= (P(6) and P(5) and P(4) and G(3));

c73 <= (P(6) and P(5) and P(4) and P(3) and G(2)) after 7 ns;

c74 <= (P(6) and P(5) and P(4) and P(3) and P(2) and G(1)) after 7 ns;

c75 <= (P(6) and P(5) and P(4) and P(3) and P(2) and P(1) and G(0)) after 7 ns;

c76 <= (P(6) and P(5) and P(4) and P(3) and P(2) and P(1) and P(0) and CarryIn) after 7 ns;

C(7) <= G(6) or c70 or c71 or c72 or c73 or c74 or c75 or c76 after 7 ns;

c80 <= (P(7) and G(6)) after 7 ns;

c81 <= (P(7) and P(6) and G(5)) after 7 ns;

c82 <= (P(7) and P(6) and P(5) and G(4)) after 7 ns;

c83 <= (P(7) and P(6) and P(5) and P(4) and G(3)) after 7 ns;

c84 <= (P(7) and P(6) and P(5) and P(4) and P(3) and G(2)) after 7 ns;

c85 <= (P(7) and P(6) and P(5) and P(4) and P(3) and P(2) and G(1)) after 7 ns;

c86 <= (P(7) and P(6) and P(5) and P(4) and P(3) and P(2) and P(1) and G(0)) after 7 ns;

c87 <= (P(7) and P(6) and P(5) and P(4) and P(3) and P(2) and P(1) and P(0) and CarryIn) after 7 ns;

C(8) <= G(7) or c80 or c81 or c82 or c83 or c84 or c85 or c86 or c87 after 7 ns;

Sum <= P xor C(7 downto 0) after 12 ns;

CarryOut <= C(8);

end arch;

**Page11**

**5.4 Comparator Stage 1**

library ieee;

use ieee.std\_logic\_1164.all;

entity comparator\_stage1 is

port (Ain : in std\_logic\_vector (7 downto 0);

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

clk : in std\_logic;

reset : in std\_logic;

AGTB : out std\_logic;

AEQB : out std\_logic;

ALTB : out std\_logic);

end comparator\_stage1;

architecture behavior of comparator\_stage1 is

component Adder

port (A : in std\_logic\_vector (7 downto 0);

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

CarryIn : in std\_logic;

CarryOut : out std\_logic;

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

end component;

component D\_FF

port (d : in std\_logic;

clk : in std\_logic;

**Page12**

reset : in std\_logic;

q : out std\_logic);

end component;

component FF\_register

port (d : in std\_logic\_vector (7 downto 0);

clk : in std\_logic;

reset : in std\_logic;

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

end component;

signal Bn,R,A,B : std\_logic\_vector (7 downto 0);

signal one : std\_logic :='1';

signal cout : std\_logic;

signal EQ,EQn,GT0,GT1,GT2,An,LT0,LT1,LT2,Rn,GT,LT : std\_logic;

begin

A\_register : FF\_register port map(d => Ain, clk => clk, reset => reset, q => A);

B\_register : FF\_register port map(d => Bin, clk => clk, reset => reset, q => B);

AGTB\_register : D\_FF port map(d => GT, clk => clk, reset => reset, q => AGTB);

AEQB\_register : D\_FF port map(d => EQ, clk => clk, reset => reset, q => AEQB);

ALTB\_register : D\_FF port map(d => LT, clk => clk, reset => reset, q => ALTB);

Bn <= not B after 2 ns;

Adder\_Subtractor : Adder port map(A => A,B => Bn,CarryIn => one,CarryOut => cout,Sum => R);

EQ <= not(R(7) or R(6) or R(5) or R(4) or R(3) or R(2) or R(1) or R(0)) after 5 ns;

EQn <= not EQ after 2 ns;

**Page13**

An <= not A(7) after 2 ns;

Rn <= not R(7);

GT0 <= EQn and An and Rn after 7 ns;

GT1 <= EQn and An and B(7) after 7 ns;

GT2 <= EQn and B(7) and Rn after 7 ns;

GT <= GT0 or GT1 or GT2 after 7 ns;

LT0 <= EQn and Bn(7) and R(7) after 7 ns;

LT1 <= EQn and A(7) and Bn(7) after 7 ns;

LT2 <= EQn and A(7) and R(7) after 7 ns;

LT <= LT0 or LT1 or LT2 after 7 ns;

end behavior;

**5.5 Test-bench Stage 1**

library ieee;

use ieee.std\_logic\_1164.all;

entity testbench\_stage1 is

end testbench\_stage1;

architecture behavior of testbench\_stage1 is

component comparator\_stage1

port (Ain : in std\_logic\_vector (7 downto 0);

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

clk : in std\_logic;

reset : in std\_logic;

AGTB : out std\_logic;

**Page14**

AEQB : out std\_logic;

ALTB : out std\_logic);

end component;

signal A,B : std\_logic\_vector (7 downto 0):="00000000";

signal clk,reset : std\_logic :='0';

signal AGTB,AEQB,ALTB : std\_logic;

begin

uut: comparator\_stage1 port map(A,B,clk,reset,AGTB,AEQB,ALTB);

process

begin

wait for 30 ns;

clk <= not clk;

end process;

process

begin

reset<='1';

wait for 10 ns;

reset<='0';

wait for 400 ns;

A <="00000111"; B<="00000101";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000111, B=00000101" severity error;

**Page15**

A <="00000101"; B<="00000111";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=00000101, B=00000111" severity error;

A <="00000111"; B<="00000111";

wait for 400 ns;

assert (AGTB='0' and AEQB='1' and ALTB='0') report "Test Failed A=00000111, B=00000111" severity error;

A <="00000101"; B<="11111000";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000101, B=11111000" severity error;

A <="00000101"; B<="11111110";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000101, B=11111110" severity error;

A <="00000111"; B<="11111000";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000111, B=11111000" severity error;

A <="00000111"; B<="11111110";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000111, B=11111110" severity error;

A <="01111111"; B<="11111110";

**Page16**

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=01111111, B=11111110" severity error;

A <="11111110"; B<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='1' and ALTB='0') report "Test Failed A=11111110, B=11111110" severity error;

B <="00000101"; A<="11111000";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111000, B=00000101" severity error;

B <="00000101"; A<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111110, B=00000101" severity error;

B <="00000111"; A<="11111000";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111000, B=00000111" severity error;

B <="00000111"; A<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111110, B=00000111" severity error;

B <="01111111"; A<="11111110";

wait for 400 ns;

**Page17**

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111110, B=01111111" severity error;

B <="01111111"; A<="01111111";

wait for 400 ns;

assert (AGTB='0' and AEQB='1' and ALTB='0') report "Test Failed A=01111111, B=01111111" severity error;

A <="11111111"; B<="11111110";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=11111111, B=11111110" severity error;

A <="11111000"; B<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111000, B=11111110" severity error;

A <="11111110"; B<="11111000";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=11111110, B=11111000" severity error;

A <="11111000"; B<="11111000";

wait for 400 ns;

assert (AGTB='0' and AEQB='1' and ALTB='0') report "Test Failed A=11111000, B=11111000" severity error;

wait;

end process;

end behavior;

**Page18**

**5.6 Magnitude Comparator**

library ieee;

use ieee.std\_logic\_1164.all;

entity magnitude\_comparator is

port (A : in std\_logic\_vector (7 downto 0);

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

AGTB : out std\_logic;

AEQB : out std\_logic;

ALTB : out std\_logic);

end magnitude\_comparator;

architecture behavior of magnitude\_comparator is

signal

EQ7,EQ6,EQ5,EQ4,EQ3,EQ2,EQ1,EQ0,GT7,GT6,GT5,GT4,GT3,GT2,GT1,GT0,LT7,LT6,LT5,LT4,LT3,LT2,LT

1,LT0 : std\_logic;

signal An,Bn : std\_logic\_vector (7 downto 0);

begin

EQ7 <= not (A(7) xor B(7)) after 9 ns;

EQ6 <= not (A(6) xor B(6)) after 9 ns;

EQ5 <= not (A(5) xor B(5)) after 9 ns;

EQ4 <= not (A(4) xor B(4)) after 9 ns;

EQ3 <= not (A(3) xor B(3)) after 9 ns;

EQ2 <= not (A(2) xor B(2)) after 9 ns;

**Page19**

EQ1 <= not (A(1) xor B(1)) after 9 ns;

EQ0 <= not (A(0) xor B(0)) after 9 ns;

AEQB <= EQ7 and EQ6 and EQ5 and EQ4 and EQ3 and EQ2 and EQ1 and EQ0 after 7 ns;

An <= not A after 2 ns;

Bn <= not B after 2 ns;

GT7 <= A(7) and Bn(7) after 7 ns;

GT6 <= EQ7 and A(6) and Bn(6) after 7 ns;

GT5 <= EQ7 and EQ6 and A(5) and Bn(5) after 7 ns;

GT4 <= EQ7 and EQ6 and EQ5 and A(4) and Bn(4) after 7 ns;

GT3 <= EQ7 and EQ6 and EQ5 and EQ4 and A(3) and Bn(3) after 7 ns;

GT2 <= EQ7 and EQ6 and EQ5 and EQ4 and EQ3 and A(2) and Bn(2) after 7 ns;

GT1 <= EQ7 and EQ6 and EQ5 and EQ4 and EQ3 and EQ2 and A(1) and Bn(1) after 7 ns;

GT0 <= EQ7 and EQ6 and EQ5 and EQ4 and EQ3 and EQ2 and EQ1 and A(0) and Bn(0) after 7 ns;

AGTB <= GT7 or GT6 or GT5 or GT4 or GT3 or GT2 or GT1 or GT0 after 7 ns;

LT7 <= An(7) and B(7) after 7 ns;

LT6 <= EQ7 and An(6) and B(6) after 7 ns;

LT5 <= EQ7 and EQ6 and An(5) and B(5) after 7 ns;

LT4 <= EQ7 and EQ6 and EQ5 and An(4) and B(4) after 7 ns;

LT3 <= EQ7 and EQ6 and EQ5 and EQ4 and An(3) and B(3) after 7 ns;

LT2 <= EQ7 and EQ6 and EQ5 and EQ4 and EQ3 and An(2) and B(2) after 7 ns;

LT1 <= EQ7 and EQ6 and EQ5 and EQ4 and EQ3 and EQ2 and An(1) and B(1) after 7 ns;

LT0 <= EQ7 and EQ6 and EQ5 and EQ4 and EQ3 and EQ2 and EQ1 and An(0) and B(0) after 7 ns;

ALTB <= LT7 or LT6 or LT5 or LT4 or LT3 or LT2 or LT1 or LT0 after 7 ns;

end behavior;

**5.7 Comparator Stage 2**

library ieee;

use ieee.std\_logic\_1164.all;

entity comparator\_stage2 is

port (Ain : in std\_logic\_vector (7 downto 0);

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

clk : in std\_logic;

reset : in std\_logic;

AGTB : out std\_logic;

AEQB : out std\_logic;

ALTB : out std\_logic);

end comparator\_stage2;

architecture behavior of comparator\_stage2 is

component D\_FF

port (d : in std\_logic;

clk : in std\_logic;

reset : in std\_logic;

q : out std\_logic);

end component;

component FF\_register

port (d : in std\_logic\_vector (7 downto 0);

clk : in std\_logic;

reset : in std\_logic;

**Page21**

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

end component;

component magnitude\_comparator

port (A : in std\_logic\_vector (7 downto 0);

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

AGTB : out std\_logic;

AEQB : out std\_logic;

ALTB : out std\_logic);

end component;

signal A\_GT\_B,A\_EQ\_B,A\_LT\_B,GT,EQ,LT,An,Bn,GT0,GT1,GT2,LT0,LT1,LT2 : std\_logic;

signal A,B : std\_logic\_vector (7 downto 0);

begin

A\_register : FF\_register port map(d => Ain, clk => clk, reset => reset, q => A);

B\_register : FF\_register port map(d => Bin, clk => clk, reset => reset, q => B);

magnitudeComparator : magnitude\_comparator port map(A => A, B => B, AGTB => A\_GT\_B, AEQB =>

A\_EQ\_B, ALTB => A\_LT\_B);

AGTB\_register : D\_FF port map(d => GT, clk => clk, reset => reset, q => AGTB);

AEQB\_register : D\_FF port map(d => EQ, clk => clk, reset => reset, q => AEQB);

ALTB\_register : D\_FF port map(d => LT, clk => clk, reset => reset, q => ALTB);

An <= not A(7) after 2 ns;

Bn <= not B(7) after 2 ns;

GT0 <= An and A\_GT\_B after 7 ns;

**Page22**

GT1 <= B(7) and A\_GT\_B after 7 ns;

GT2 <= An and B(7) after 7 ns;

GT <= GT0 or GT1 or GT2 after 7 ns;

EQ <= A\_EQ\_B ;

LT0 <= Bn and A\_LT\_B after 7 ns;

LT1 <= A(7) and A\_LT\_B after 7 ns;

LT2 <= A(7) and Bn after 7 ns;

LT <= LT0 or LT1 or LT2 after 7 ns;

end behavior;

**5.8 Test-bench Stage 2**

library ieee;

use ieee.std\_logic\_1164.all;

entity testbench\_stage2 is

end testbench\_stage2;

architecture behavior of testbench\_stage2 is

component comparator\_stage2

port (Ain : in std\_logic\_vector (7 downto 0);

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

clk : in std\_logic;

reset : in std\_logic;

AGTB : out std\_logic;

AEQB : out std\_logic;

ALTB : out std\_logic);

**Page23**

end component;

signal A,B : std\_logic\_vector (7 downto 0):="00000000";

signal clk,reset : std\_logic :='0';

signal AGTB,AEQB,ALTB : std\_logic;

begin

uut: comparator\_stage2 port map(A,B,clk,reset,AGTB,AEQB,ALTB);

process

begin

wait for 25 ns;

clk <= not clk;

end process;

process

begin

reset<='1';

wait for 10 ns;

reset<='0';

wait for 400 ns;

A <="00000111"; B<="00000101";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000111, B=00000101" severity error;

A <="00000101"; B<="00000111";

wait for 400 ns;

**Page24**

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=00000101, B=00000111" severity error;

A <="00000111"; B<="00000111";

wait for 400 ns;

assert (AGTB='0' and AEQB='1' and ALTB='0') report "Test Failed A=00000111, B=00000111" severity error;

A <="00000101"; B<="11111000";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000101, B=11111000" severity error;

A <="00000101"; B<="11111110";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000101, B=11111110" severity error;

A <="00000111"; B<="11111000";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000111, B=11111000" severity error;

A <="00000111"; B<="11111110";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=00000111, B=11111110" severity error;

A <="01111111"; B<="11111110";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=01111111, B=11111110" severity error;

**Page25**

A <="11111110"; B<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='1' and ALTB='0') report "Test Failed A=11111110, B=11111110" severity error;

B <="00000101"; A<="11111000";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111000, B=00000101" severity error;

B <="00000101"; A<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111110, B=00000101" severity error;

B <="00000111"; A<="11111000";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111000, B=00000111" severity error;

B <="00000111"; A<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111110, B=00000111" severity error;

B <="01111111"; A<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111110, B=01111111" severity error;

B <="01111111"; A<="01111111";

**Page26**

wait for 400 ns;

assert (AGTB='0' and AEQB='1' and ALTB='0') report "Test Failed A=01111111, B=01111111" severity error;

A <="11111111"; B<="11111110";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=11111111, B=11111110" severity error;

A <="11111000"; B<="11111110";

wait for 400 ns;

assert (AGTB='0' and AEQB='0' and ALTB='1') report "Test Failed A=11111000, B=11111110" severity error;

A <="11111110"; B<="11111000";

wait for 400 ns;

assert (AGTB='1' and AEQB='0' and ALTB='0') report "Test Failed A=11111110, B=11111000" severity error;

A <="11111000"; B<="11111000";

wait for 400 ns;

assert (AGTB='0' and AEQB='1' and ALTB='0') report "Test Failed A=11111000, B=11111000" severity error;

wait;

end process;

end behavior;

**Page27**