

Course Project: 8-bit Comparator for signed 2’s complement representation

Course: Advanced Digital Systems Design

Professor: Dr. Abdellatif Abu-Issa

Student: Basil Mari

Student ID: 1191027

Section: 2

Date: 27/12/2021

Table of Contents:

[Table of Contents: 1](#_Toc91705991)

[Introduction and Background: 2](#_Toc91705992)

[Design Philosophy: 2](#_Toc91705993)

[Results: 3](#_Toc91705994)

[Conclusion: 4](#_Toc91705995)

[Appendix: 5](#_Toc91705996)

# Introduction and Background:

The objective of this project is to design and program a signed 8-bit magnitude comparator in VHDL in two stages. In the first stage, the comparator utilizes an 8-bit adder, while in the second stage, it utilizes a 7-bit magnitude comparator plus a sign bit. The circuits are built structurally from the set of basic gates below, with the specified time delays. The combinational circuits are to be converted into synchronous ones by connecting the inputs and outputs to registers and a clock, and the final circuit is to be tested with a test bench. The software used is Aldec Active-HDL Student Edition.

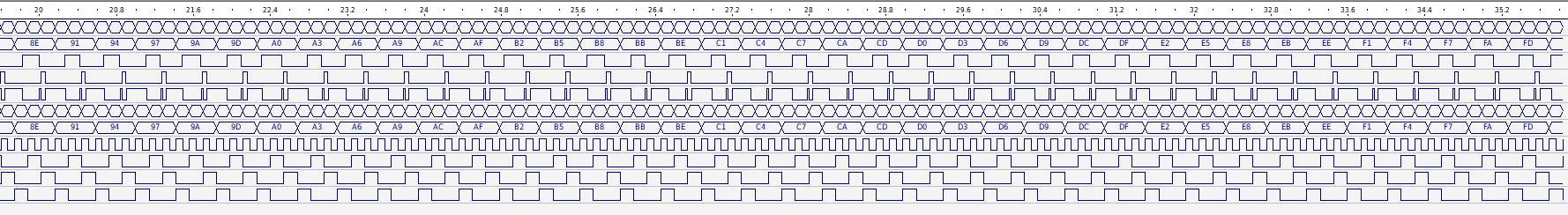
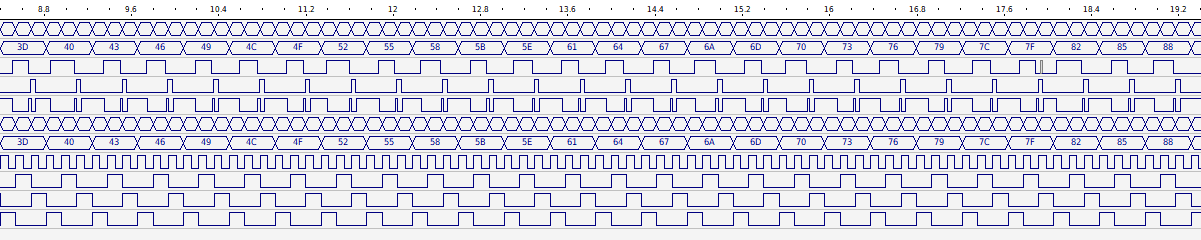
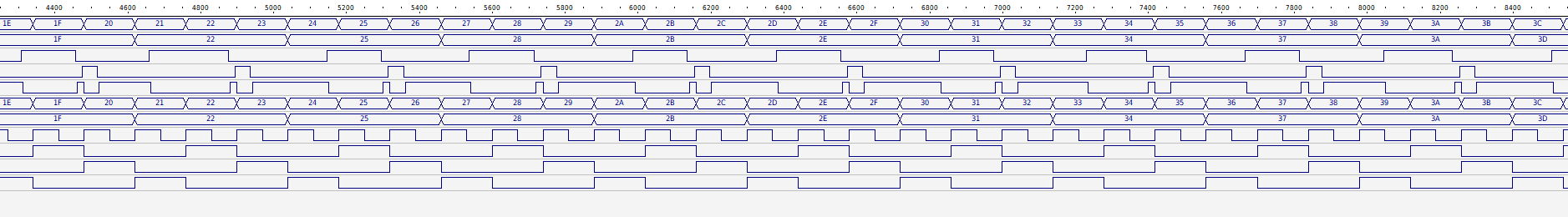
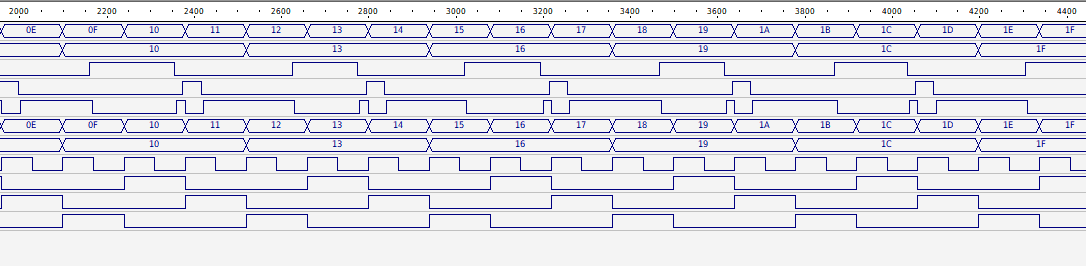
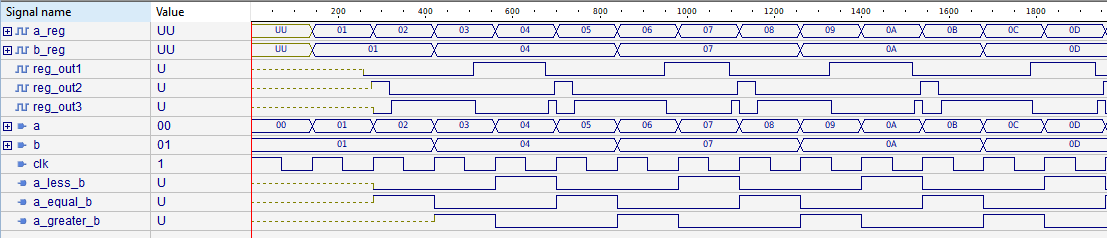
|  |  |
| --- | --- |
| **Gate** | **Delay** |
| Inverter | 2 ns |
| NAND | 5 ns |
| NOR | 5 ns |
| AND | 7 ns |
| OR | 7 ns |
| XNOR | 9 ns |
| XOR | 12 ns |

# Design Philosophy:

Firstly, the basic gates were implemented with the respective time delays. For the stage one 8-bit comparator, a full adder was constructed structurally from the basic gates. Then, an 8-bit full adder was constructed from the full adder. After that, the 8-bit full adder was converted into a subtractor. An 8-bit comparator was constructed from the subtractor under the following principals: input A is equal to input B if all outputs of the subtractor are 0. A is smaller than B if the most significant of the subtractor is equal to 1. A is larger than B if both the previous statements are false. The 8-bit comparator was connected to registers on its inputs and outputs, which were connected to a clock. Finally, a test bench was created for the synchronous circuit, which tested all three cases for the output, and checked the validity of the results.

As for the stage two 8-bit comparator, firstly, a 1-bit magnitude comparator was constructed. The 1-bit magnitude comparator was used to construct a cascading 1-bit magnitude comparator, which was used to construct a 7-bit magnitude comparator. A 8-bit signed magnitude comparator was created from the 7-bit magnitude comparator, where it was used in conjunction additional logic to account for the sign bits of the two inputs. The logic is as follows: if the sign bits of A and B are “0” and “1”, then A is greater than B. If they are “1” and “0”, then A is smaller than B. If they are “0” and “0”, then the output of the 7-bit magnitude comparator is the final output. If they are “1” and “0”, then the final output is equal to the opposite of the 7-bit magnitude comparator output (e.g. if the output of the 7-bit magnitude comparator is A<B = 1, A=B = 0, and A>B = 0, then the final output will be A<B = 0, A=B = 0, and A>B = 1). The 8-bit signed magnitude comparator was connected to registers through its inputs and outputs, and was tested using the same test bench.

# Results:



# Conclusion:

The results of the simulation were successful in achieving the specified design goals. The synchronous comparator correctly outputs values indicating the difference in magnitude between its inputs. The synchronous circuit does not output any errors or glitches when used at or below the maximum frequency, which is equal to 1 over the minimum clock period. This is equal to . This project demonstrated the process of designing, programming, and simulating a synchronous circuit in VHDL with Active-HDL from start to finish. The project showed how one can create a circuit accounting for real-world factors like time delays and glitches. The experience gained from this project is a jumping-off point for the design of even more complex systems with VHDL.

# Appendix:

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity fulladder is

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

sum,cout: out std\_logic);

end fulladder;

architecture struct of fulladder is

signal w1,w2,w3: std\_logic;

begin

g1 : entity work.xorgate(behav) port map (A,B,w1);

g2 : entity work.xorgate(behav) port map (w1,Cin,sum);

g3 : entity work.andgate(behav) port map (A,B,w2);

g4 : entity work.andgate(behav) port map (w1,Cin,w3);

g5 : entity work.orgate(behav) port map (w2,w3,Cout);

end struct;

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity fulladder8bit is

port(a,b: in std\_logic\_vector(7 downto 0);

cin: in std\_logic;

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

cout: out std\_logic);

end fulladder8bit;

architecture struct of fulladder8bit is

signal c1,c2,c3,c4,c5,c6,c7: STD\_LOGIC;

begin

FA1 : entity work.fulladder(struct) port map (A(0),B(0),Cin,s(0),c1);

FA2 : entity work.fulladder(struct) port map (A(1),B(1),C1,s(1),c2);

FA3 : entity work.fulladder(struct) port map (A(2),B(2),C2,s(2),c3);

FA4 : entity work.fulladder(struct) port map (A(3),B(3),C3,s(3),c4);

FA5 : entity work.fulladder(struct) port map (A(4),B(4),C4,s(4),c5);

FA6 : entity work.fulladder(struct) port map (A(5),B(5),C5,s(5),c6);

FA7 : entity work.fulladder(struct) port map (A(6),B(6),C6,s(6),c7);

FA8 : entity work.fulladder(struct) port map (A(7),B(7),C7,s(7),cout);

end struct;

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity subtractor is

port(a,b: in std\_logic\_vector(7 downto 0);

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

bout: out std\_logic);

end subtractor;

architecture struct of subtractor is

signal nb: STD\_LOGIC\_vector(7 downto 0);

begin

N1 : entity work.notgate(behav) port map (B(0),nb(0));

N2 : entity work.notgate(behav) port map (B(1),nb(1));

N3 : entity work.notgate(behav) port map (B(2),nb(2));

N4 : entity work.notgate(behav) port map (B(3),nb(3));

N5 : entity work.notgate(behav) port map (B(4),nb(4));

N6 : entity work.notgate(behav) port map (B(5),nb(5));

N7 : entity work.notgate(behav) port map (B(6),nb(6));

N8 : entity work.notgate(behav) port map (B(7),nb(7));

adder : entity work.fulladder8bit(struct) port map (A,nb,'1',s,bout);

end struct;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity comparator8bit is

port(a,b: in std\_logic\_vector(7 downto 0);

a\_less\_b,a\_equal\_b,a\_greater\_b: out std\_logic);

end comparator8bit;

architecture struct of comparator8bit is

signal s: STD\_LOGIC\_vector(7 downto 0);

signal bout, s0\_nor\_s1, s2\_nor\_s3, s4\_nor\_s5, s6\_nor\_s7, nor\_s, w1,w2: std\_logic;

begin

S1 : entity work.subtractor(struct) port map (a,b,s,bout);

N2 : entity work.norgate(behav) port map (s(0),S(1),s0\_nor\_s1);

N3 : entity work.norgate(behav) port map (s(2),s(3),s2\_nor\_s3);

N4 : entity work.norgate(behav) port map (s(4),s(5),s4\_nor\_s5);

N5 : entity work.norgate(behav) port map (s(6),s(7),s6\_nor\_s7);

A6 : entity work.andgate(behav) port map (s0\_nor\_s1,s2\_nor\_s3,w1);

A7 : entity work.andgate(behav) port map (s4\_nor\_s5,s6\_nor\_s7,w2);

A8 : entity work.andgate(behav) port map (w1,w2,nor\_s);

O10 : entity work.norgate(behav) port map (nor\_s,s(7),a\_greater\_b);

a\_equal\_b <= nor\_s;

a\_less\_b <= s(7);

end struct;

architecture behav of comparator8bit is

signal Answer: STD\_Logic\_Vector(2 downto 0);

begin

a\_less\_b<=Answer(0);

a\_greater\_b<=Answer(1);

a\_equal\_b<=Answer(2);

Answer<="001" WHEN A<B

Else "010" when A>B

Else "100" when A=B;

end behav;

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity sync\_comparator is

port(a,b: in std\_logic\_vector(7 downto 0);

clk: in std\_logic;

a\_less\_b,a\_equal\_b,a\_greater\_b: out std\_logic);

end sync\_comparator;

architecture behav of sync\_comparator is

signal a\_reg,b\_reg: STD\_LOGIC\_vector(7 downto 0);

signal reg\_out1,reg\_out2,reg\_out3: std\_logic;

begin

C1 : entity work.comparator8bit(struct) port map (a\_reg,b\_reg,reg\_out1,reg\_out2,reg\_out3);

process(CLK)

begin

if(rising\_edge(clk)) then

a\_reg <= a;

b\_reg <= b;

a\_less\_b <= reg\_out1;

a\_equal\_b <= reg\_out2;

a\_greater\_b <= reg\_out3;

end if;

end process;

end architecture behav;

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

use ieee.numeric\_std.all;

library work;

ENTITY testbench IS

END testbench;

ARCHITECTURE behavior OF testbench IS

signal A, B : std\_logic\_vector(7 downto 0) := (others => '0');

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

signal A\_less\_B, A\_equal\_B, A\_greater\_B, less\_correct, equal\_correct, greater\_correct : std\_logic;

constant clock\_period : time := 140 ns;

BEGIN

uut: entity work.sync\_comparator(behav) PORT MAP (a,b,clk,a\_less\_b,a\_equal\_b,a\_greater\_b);

correct: entity work.comparator8bit(behav) port map (a,b,less\_correct,equal\_correct,greater\_correct);

clk <= not clk after clock\_period / 2;

stim\_proc: process

begin

for i in 0 to 85 loop

B <= std\_logic\_vector(to\_signed(1+3\*i,B'length));

for o in 0 to 2 loop

A <= std\_logic\_vector(to\_signed(3\*i+o,A'length));

wait for clock\_period;

assert ((a\_equal\_b = equal\_correct) and (a\_less\_b = less\_correct) and (a\_greater\_b = greater\_correct)) report "<error message>" severity error;

end loop;

end loop;

wait;

end process;

END;

ARCHITECTURE behavior2 OF testbench IS

signal A, B : std\_logic\_vector(7 downto 0) := (others => '0');

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

signal A\_less\_B, A\_equal\_B, A\_greater\_B, less\_correct, equal\_correct, greater\_correct : std\_logic;

constant clock\_period : time := 140 ns;

BEGIN

uut: entity work.sync\_comparator(struc2) PORT MAP (a,b,clk,a\_less\_b,a\_equal\_b,a\_greater\_b);

correct: entity work.comparator8bit(behav) port map (a,b,less\_correct,equal\_correct,greater\_correct);

clk <= not clk after clock\_period / 2;

stim\_proc: process

begin

for i in 0 to 85 loop

B <= std\_logic\_vector(to\_signed(1+3\*i,B'length));

for o in 0 to 2 loop

A <= std\_logic\_vector(to\_signed(3\*i+o,A'length));

wait for clock\_period;

assert ((a\_equal\_b = equal\_correct) and (a\_less\_b = less\_correct) and (a\_greater\_b = greater\_correct)) report "<error message>" severity error;

end loop;

end loop;

wait;

end process;

END;

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity mag\_comp is

port(a,b: in std\_logic;

a\_less\_b, a\_equal\_b, a\_greater\_b: out std\_logic);

end mag\_comp;

architecture struct of mag\_comp is

signal w1,w2: std\_logic;

begin

g1 : entity work.notgate(behav) port map (B,w1);

g2 : entity work.notgate(behav) port map (A,w2);

g3 : entity work.xnorgate(behav) port map (A,B,a\_equal\_b);

g4 : entity work.andgate(behav) port map (A,w1,a\_greater\_b);

g5 : entity work.andgate(behav) port map (w2,B,a\_less\_b);

end struct;

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity cascading\_comp is

port(a,b,less\_in,equal\_in,greater\_in: in std\_logic;

a\_less\_b, a\_equal\_b, a\_greater\_b: out std\_logic);

end cascading\_comp;

architecture struct of cascading\_comp is

signal w1,w2,w3,w4,w5,w6: std\_logic;

begin

g1 : entity work.mag\_comp(struct) port map (A,B,w1,w2,w3);

g2 : entity work.andgate(behav) port map (w2,less\_in,w4);

g3 : entity work.andgate(behav) port map (w2,equal\_in,w5);

g4 : entity work.andgate(behav) port map (w2,greater\_in,w6);

g5 : entity work.orgate(behav) port map (w1,w4,a\_less\_b);

g6 : entity work.orgate(behav) port map (w2,w5,a\_equal\_b);

g7 : entity work.orgate(behav) port map (w3,w6,a\_equal\_b);

end struct;

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity mag\_comp7bit is

port(a,b: in std\_logic\_vector(6 downto 0);

a\_less\_b, a\_equal\_b, a\_greater\_b: out std\_logic);

end mag\_comp7bit;

architecture struct of mag\_comp7bit is

signal w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16,w17,w18: std\_logic;

begin

g1 : entity work.cascading\_comp(struct) port map (A(0),B(0),'0','0','0',w1,w2,w3);

g2 : entity work.cascading\_comp(struct) port map (A(1),B(1),w1,w2,w3,w4,w5,w6);

g3 : entity work.cascading\_comp(struct) port map (A(2),B(2),w4,w5,w6,w7,w8,w9);

g4 : entity work.cascading\_comp(struct) port map (A(3),B(3),w7,w8,w9,w10,w11,w12);

g5 : entity work.cascading\_comp(struct) port map (A(4),B(4),w10,w11,w12,w13,w14,w15);

g6 : entity work.cascading\_comp(struct) port map (A(5),B(5),w13,w14,w15,w16,w17,w18);

g7 : entity work.cascading\_comp(struct) port map (A(6),B(6),w16,w17,w18,a\_less\_b,a\_equal\_b,a\_greater\_b);

end struct;

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity signed\_comp8bit is

port(a,b: in std\_logic\_vector(7 downto 0);

a\_less\_b, a\_equal\_b, a\_greater\_b: out std\_logic);

end signed\_comp8bit;

architecture struct of signed\_comp8bit is

signal w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16,w17: std\_logic;

begin

g1 : entity work.mag\_comp7bit(struct) port map (A(6 downto 0),B(6 downto 0),w1,w2,w3);

g2 : entity work.notgate(behav) port map (A(7),w4);

g3 : entity work.notgate(behav) port map (B(7),w5);

g4 : entity work.andgate(behav) port map (w4,B(7),w6); --01

g5 : entity work.andgate(behav) port map (w5,A(7),w7); --10

g6 : entity work.andgate(behav) port map (w4,w5,w8); --00

g7 : entity work.andgate(behav) port map (w8,w1,w9); --a<b when 00

g8 : entity work.andgate(behav) port map (w8,w2,w10); --a=b when 00

g9 : entity work.andgate(behav) port map (w8,w3,w11); --a>b when 00

g10 : entity work.andgate(behav) port map (A(7),B(7),w12); --11

g11 : entity work.andgate(behav) port map (w12,w3,w13); --a<b when 11

g12 : entity work.andgate(behav) port map (w12,w2,w14); --a=b when 11

g13 : entity work.andgate(behav) port map (w12,w1,w15); --a>b when 11

g14 : entity work.orgate(behav) port map (w9,w13,w16);

g15 : entity work.orgate(behav) port map (w16,w7,a\_less\_b);

g16 : entity work.orgate(behav) port map (w10,w14,a\_less\_b);

g17 : entity work.orgate(behav) port map (w11,w15,w17);

g18 : entity work.orgate(behav) port map (w17,w6,a\_greater\_b);

end struct;

library ieee;

use ieee.std\_logic\_1164.all;

library work;

entity sync\_comparator is

port(a,b: in std\_logic\_vector(7 downto 0);

clk: in std\_logic;

a\_less\_b,a\_equal\_b,a\_greater\_b: out std\_logic);

end sync\_comparator;

architecture struct2 of sync\_comparator is

signal a\_reg,b\_reg: STD\_LOGIC\_vector(7 downto 0);

signal reg\_out1,reg\_out2,reg\_out3: std\_logic;

begin

C1 : entity work.signed\_comp8bit port map (a\_reg,b\_reg,reg\_out1,reg\_out2,reg\_out3);

process(CLK)

begin

if(rising\_edge(clk)) then

a\_reg <= a;

b\_reg <= b;

a\_less\_b <= reg\_out1;

a\_equal\_b <= reg\_out2;

a\_greater\_b <= reg\_out3;

end if;

end process;

end architecture struct2;

library ieee;

use ieee.std\_logic\_1164.all;

entity notgate is

port(a: in std\_logic; o: out std\_logic);

end notgate;

architecture behav of notgate is

begin

o <= not a after 2 ns;

end behav;

library ieee;

use ieee.std\_logic\_1164.all;

entity nandgate is

port(a,b: in std\_logic; o: out std\_logic);

end nandgate;

architecture behav of nandgate is

begin

o <= a nand b after 5 ns;

end behav;

library ieee;

use ieee.std\_logic\_1164.all;

entity norgate is

port(a,b: in std\_logic; o: out std\_logic);

end norgate;

architecture behav of norgate is

begin

o <= a nor b after 5 ns;

end behav;

library ieee;

use ieee.std\_logic\_1164.all;

entity andgate is

port(a,b: in std\_logic; o: out std\_logic);

end andgate;

architecture behav of andgate is

begin

o <= a and b after 7 ns;

end behav;

library ieee;

use ieee.std\_logic\_1164.all;

entity orgate is

port(a,b: in std\_logic; o: out std\_logic);

end orgate;

architecture behav of orgate is

begin

o <= a or b after 7 ns;

end behav;

library ieee;

use ieee.std\_logic\_1164.all;

entity xorgate is

port(a,b: in std\_logic; o: out std\_logic);

end xorgate;

architecture behav of xorgate is

begin

o <= a xor b after 9 ns;

end behav;

library ieee;

use ieee.std\_logic\_1164.all;

entity xnorgate is

port(a,b: in std\_logic; o: out std\_logic);

end xnorgate;

architecture behav of xnorgate is

begin

o <= a xnor b after 12 ns;

end behav;