**EENG 2910 Project III – Digital System Design**

Design Project 3

Nathan A Ruprecht

Due 07 October 2016

# Introduction:

For our third project, we were required to design, implement, and test an 8-bit arithmetic logic unit (ALU). An ALU is a combinational circuit that performs arithmetic and bitwise logical operations on integer binary numbers.

# Theory of Operation and Explanation of the Design:

Since an ALU is capable of doing any number of functions, a MUX (case statement) is used to choose which function’s output is sent through. The functions I implemented are an eight bit adder, comparator, XOR, shifter, AND, NAND, 2s complement, and an eight bit subtractor.

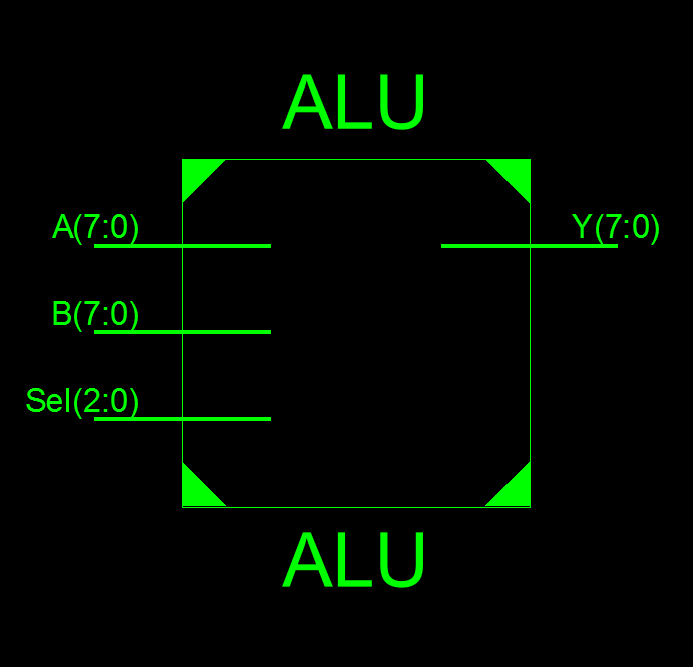


Figure : ALU Schematic

We were given code for a 4 bit adder, comparator, XOR, and shift. I added more full adders to the adder file to make an 8 bit adder, and just changed the log vectors for all the functions to 8 bits.

AND and NAND were simple in that I would AND the 2 input vectors, or AND them then NOT the output. 2s complement just uses 1 input, NOTs it, and uses the 8 bit adder to add 1 to it. 8 bit subtraction adds input A to the 2s complement of B. If 2s complement doesn’t work, neither does the subtractor.

# Experimental Results:

The code itself was not too terrible. Troubleshooting was. I coded all the components within an hour or two, but immediately ran into problems when running the test bench. When I tested it for the first time, only the NAND worked. The first problem was that the output of the adder was showing up as unknown for all 8 bits. It turned out that my adder was fine, and that, no matter which component was tested first, the first output would always show up as unknown.

After I got the adder, I went to troubleshooting the AND since it should be the simplest. The NAND worked, but not the AND which was odd. I heard that there is sometimes a mismatch with signals since there is so much going on, and that it’s sometimes recommended to read the input into an internal signal. When I assigned the select input vector to a signal, and ran my ALU off of that signal instead, the comparator, XOR, shifter, and AND all worked. All that was left was the 2s complement and subtractor.

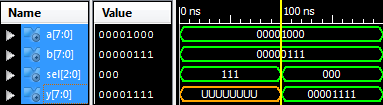
The 8-bit adder uses 8 1-bit adder components to do the math. To the left, I tested 8 plus 7 to show it equals 15.

Figure : 8-bit Adder

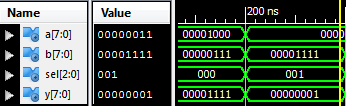
For the comparator, if A is less than B, the output is 0x01. If A is greater than B, the output is 0x80. If they are equal, then Y is 0x10. Any other combination and the output is 0x0. The test shows that A (3) is less than B (15).

Figure : Comparator

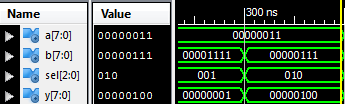
The XOR holds true to show that 0x01 with 0x07 equals 0x03 since the third bit is the only one comparing a 0 and 1.

Figure : XOR

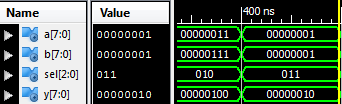
The shifter takes A and shifts it by an index of B. So the larger B is, the more A is shifter to the left. I show a simple example by shifting A just one bit over to the left.

Figure : Shifter

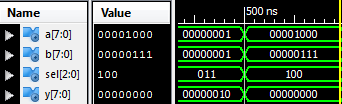
Implement an AND gate is easy to show. With A being 0x08 and B as 0x07, the answer correctly comes out to be 0.

Figure : AND

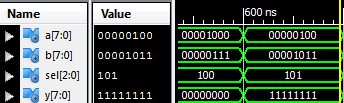
A NAND gate uses the same code as the AND, just inverting the AND. The example ANDs A and B to get 0x0, then inverts it to show 0xFF.

Figure : NAND

Since the subtractor needed the 2s complement, I spent all my time troubleshooting the 2s complement. Since the 2s complement uses the adder component, I wanted to make sure that worked. I assigned an A and B for the component and confirmed the 2s complement was using the adder correctly. Problems came in when I was feeding the logic vector A into the adder. I assigned “not A” to a signal, and sent that signal through the adder, but still nothing. Just to see if A was even going in, I assigned A to a signal, then that signal to the output, and A was not shown as Y.

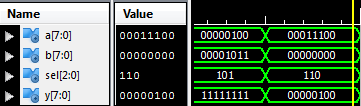
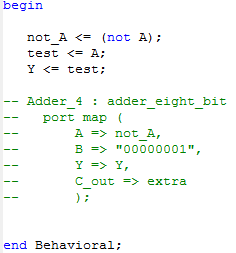


Figure : 2s Complement Test

Since an input vector (A) cannot be assigned directly to an output vector (Y) because of type mismatch, I made a signal vector (test) to act as a median. When A was 0x1C, Y came out to be 0x04. So even if the code is correct, the 2s Complement component won’t work since the input is not getting fed in.

Figure : 2s Complement Test Code

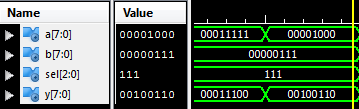


Figure : 8-Bit Subtractor

Since the 2s Complement would not work, that automatically threw off the output for the subtractor since it used the 2s Complement as a component.

# Conclusion:

The goal of the lab was to design, implement, and test an 8-bit arithmetic logic unit (ALU). I successfully took the given code for 4 components (add, compare, XOR, and shifter) and changed them from 4-bit to 8-bit for the ALU. I successfully added in 2 more components (AND and NAND) as 8-bit inputs and output. Although I think the code is correct but cannot find the fault, I failed to implement 2 components (2s Complement and subtractor) after extensive testing and Googling.

# References:

Figures 1-10 are screenshots.

# Source Code:

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 15:00:41 09/28/2016

-- Design Name:

-- Module Name: ALU - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ALU is

Port ( A : in STD\_LOGIC\_VECTOR (7 downto 0);

B : in STD\_LOGIC\_VECTOR (7 downto 0);

Sel : in STD\_LOGIC\_VECTOR (2 downto 0);

Y : out STD\_LOGIC\_VECTOR (7 downto 0));

end ALU;

architecture Behavioral of ALU is

--------------------------------------------------------------------------------

component adder\_eight\_bit

port (

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

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

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

C\_out : out std\_logic

);

end component;

--------------------------------------------------------------------------------

component comparator

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end component;

--------------------------------------------------------------------------------

component ALU\_xor

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end component;

--------------------------------------------------------------------------------

component shifter

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end component;

--------------------------------------------------------------------------------

component ALU\_and

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end component;

--------------------------------------------------------------------------------

component ALU\_nand

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end component;

--------------------------------------------------------------------------------

component twos\_complement

port (

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

Y : out std\_logic\_vector(7 downto 0)

);

end component;

--------------------------------------------------------------------------------

component subtract\_eight\_bit

port (

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

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

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

C\_out : out std\_logic

);

end component;

--------------------------------------------------------------------------------

signal s\_Y0 : std\_logic\_vector(7 downto 0);-- := (others => '0');

signal s\_Y1 : std\_logic\_vector(7 downto 0);

signal s\_Y2 : std\_logic\_vector(7 downto 0);

signal s\_Y3 : std\_logic\_vector(7 downto 0);

signal s\_Y4 : std\_logic\_vector(7 downto 0);

signal s\_Y5 : std\_logic\_vector(7 downto 0);

signal s\_Y6 : std\_logic\_vector(7 downto 0);

signal s\_Y7 : std\_logic\_vector(7 downto 0);

signal s\_sig : std\_logic\_vector(2 downto 0);

signal s\_C\_out : std\_logic\_vector(1 downto 0);

begin

s\_sig<=Sel;

Adder\_1 : adder\_eight\_bit

port map (

A => A,

B => B,

Y => s\_Y0,

C\_out => s\_C\_out(0)

);

Comparator\_1 : comparator

port map(

A => A,

B => B,

Y => s\_Y1

);

XOR\_1 : ALU\_xor

port map(

A => A,

B => B,

Y => s\_Y2

);

Shifter\_1 : shifter

port map(

A => A,

B => B,

Y => s\_Y3

);

AND\_1 : ALU\_and

port map(

A => A,

B => B,

Y => s\_Y4

);

NAND\_1 : ALU\_nand

port map(

A => A,

B => B,

Y => s\_Y5

);

TWOSCOMP\_1 : twos\_complement

port map(

A => A,

Y => s\_Y6

);

Subtract\_1 : subtract\_eight\_bit

port map (

A => A,

B => B,

Y => s\_Y7,

C\_out => s\_C\_out(1)

);

process(s\_sig, A, B)

begin

case s\_sig is

when "000" => Y <= s\_Y0; --add

when "001" => Y <= s\_Y1; --compare

when "010" => Y <= s\_Y2; --XOR

when "011" => Y <= s\_Y3; --shifter

when "100" => Y <= s\_Y4; --AND

when "101" => Y <= s\_Y5; --NAND

when "110" => Y <= s\_Y6; --twos\_complement

when "111" => Y <= s\_Y7; --subtraction

when others => Y <= (others => '0');

end case;

end process;

end Behavioral;

--------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 15:06:46 09/28/2016

-- Design Name:

-- Module Name: E:/2910/DP3/Code/ALU/tb\_ALU.vhd

-- Project Name: ALU

-- Target Device:

-- Tool versions:

-- Description:

--

-- VHDL Test Bench Created by ISE for module: ALU

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

-- Notes:

-- This testbench has been automatically generated using types std\_logic and

-- std\_logic\_vector for the ports of the unit under test. Xilinx recommends

-- that these types always be used for the top-level I/O of a design in order

-- to guarantee that the testbench will bind correctly to the post-implementation

-- simulation model.

--------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity tb\_ALU is

-- Port ( );

end tb\_ALU;

architecture Behavioral of tb\_ALU is

component ALU is

Port ( A : in STD\_LOGIC\_VECTOR (7 downto 0);

B : in STD\_LOGIC\_VECTOR (7 downto 0);

Sel : in STD\_LOGIC\_VECTOR (2 downto 0);

Y : out STD\_LOGIC\_VECTOR (7 downto 0));

end component;

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

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

signal Sel : std\_logic\_vector(2 downto 0);

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

begin

uut : ALU

port map (

A => A,

B => B,

Sel => Sel,

Y => Y

);

sim : process

begin

A <= "00001000";

B <= "00000111";

Sel <= "111";

wait for 100 ns;

A <= "00001000";

B <= "00000111";

Sel <= "000";

wait for 100 ns;

A <= "00000011";

B <= "00001111";

Sel <= "001";

wait for 100 ns;

A <= "00000011";

B <= "00000111";

Sel <= "010";

wait for 100 ns;

A <= "00000001";

B <= "00000001";

Sel <= "011";

wait for 100 ns;

A <= "00001000";

B <= "00000111";

Sel <= "100";

wait for 100 ns;

A <= "00000100";

B <= "00001011";

Sel <= "101";

wait for 100 ns;

A <= "00011100";

B <= "00000000";

Sel <= "110";

wait for 100 ns;

A <= "00011111";

B <= "00000111";

Sel <= "111";

wait for 100 ns;

end process;

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 15:47:08 10/05/2016

-- Design Name:

-- Module Name: adder\_eight\_bit - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity adder\_eight\_bit is

port (

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

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

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

C\_out : out std\_logic

);

end adder\_eight\_bit;

architecture Behavioral of adder\_eight\_bit is

component adder\_one\_bit

Port (

A : in std\_logic;

B : in std\_logic;

C\_in : in std\_logic;

S : out std\_logic;

C\_out : out std\_logic

);

end component;

signal s\_Cin : std\_logic := '0';

signal s\_Cout0 : std\_logic;

signal s\_Cout1 : std\_logic;

signal s\_Cout2 : std\_logic;

signal s\_Cout3 : std\_logic;

signal s\_Cout4 : std\_logic;

signal s\_Cout5 : std\_logic;

signal s\_Cout6 : std\_logic;

signal s\_Cout7 : std\_logic;

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

signal Y\_reg : std\_logic\_vector(7 downto 0);

begin

adder\_0 : adder\_one\_bit

port map (

A => A(0),

B => B(0),

C\_in => s\_Cin,

S => Y\_reg(0),

C\_out => s\_Cout0

);

adder\_1 : adder\_one\_bit

port map (

A => A(1),

B => B(1),

C\_in => s\_Cout0,

S => Y\_reg(1),

C\_out => s\_Cout1

);

adder\_2 : adder\_one\_bit

port map (

A => A(2),

B => B(2),

C\_in => s\_Cout1,

S => Y\_reg(2),

C\_out => s\_Cout2

);

adder\_3 : adder\_one\_bit

port map (

A => A(3),

B => B(3),

C\_in => s\_Cout2,

S => Y\_reg(3),

C\_out => s\_Cout3

);

adder\_4 : adder\_one\_bit

port map (

A => A(4),

B => B(4),

C\_in => s\_Cout3,

S => Y\_reg(4),

C\_out => s\_Cout4

);

adder\_5 : adder\_one\_bit

port map (

A => A(5),

B => B(5),

C\_in => s\_Cout4,

S => Y\_reg(5),

C\_out => s\_Cout5

);

adder\_6 : adder\_one\_bit

port map (

A => A(6),

B => B(6),

C\_in => s\_Cout5,

S => Y\_reg(6),

C\_out => s\_Cout6

);

adder\_7 : adder\_one\_bit

port map (

A => A(7),

B => B(7),

C\_in => s\_Cout6,

S => Y\_reg(7),

C\_out => s\_Cout7

);

C\_out <= s\_Cout7;

Y <= Y\_reg;

result <= s\_Cout7 & Y\_reg;

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 15:02:26 09/28/2016

-- Design Name:

-- Module Name: comparator - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity comparator is

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end comparator;

architecture Behavioral of comparator is

begin

process(A, B)

begin

if (A > B) then

Y <= "10000000"; --if A is bigger than B, output 1000 0000

elsif (A = B) then

Y <= "00010000"; --if A equals B, output 0001 0000

elsif (A < B) then

Y <= "00000001"; --if A is smaller than B, output 0000 0001

else

Y <= "00000000"; --others, output 0000 0000

end if;

end process;

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 15:02:12 09/28/2016

-- Design Name:

-- Module Name: ALU\_xor - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ALU\_xor is

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end ALU\_xor;

architecture Behavioral of ALU\_xor is

begin

Y(0) <= A(0) xor B(0);

Y(1) <= A(1) xor B(1);

Y(2) <= A(2) xor B(2);

Y(3) <= A(3) xor B(3);

Y(4) <= A(4) xor B(4);

Y(5) <= A(5) xor B(5);

Y(6) <= A(6) xor B(6);

Y(7) <= A(7) xor B(7);

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 15:50:18 09/28/2016

-- Design Name:

-- Module Name: shifter - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity shifter is

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end shifter;

architecture Behavioral of shifter is

begin

process(B)

begin

if ( B(0) = '1' ) then

Y <= A(6 downto 0) & '0';

elsif ( B(1) = '1' ) then

Y <= A(5 downto 0) & "00";

elsif ( B(2) = '1') then

Y <= A(4 downto 0) & "000";

elsif ( B(3) = '1' ) then

Y <= A(3 downto 0) & "0000";

elsif ( B(4) = '1') then

Y <= A(2 downto 0) & "00000";

elsif ( B(5) = '1' ) then

Y <= A(1 downto 0) & "000000";

elsif ( B(6) = '1') then

Y <= A(0) & "0000000";

else

Y <= A;

end if;

end process;

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 10:48:46 10/03/2016

-- Design Name:

-- Module Name: ALU\_and - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ALU\_and is

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end ALU\_and;

architecture Behavioral of ALU\_and is

begin

Y <= (A and B);

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 10:54:13 10/03/2016

-- Design Name:

-- Module Name: ALU\_nand - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ALU\_nand is

port (

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

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

Y : out std\_logic\_vector(7 downto 0)

);

end ALU\_nand;

architecture Behavioral of ALU\_nand is

begin

Y <= not(A and B);

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 10:58:12 10/03/2016

-- Design Name:

-- Module Name: twos\_complement - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity twos\_complement is

port (

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

Y : out std\_logic\_vector(7 downto 0)

);

end twos\_complement;

architecture Behavioral of twos\_complement is

--------------------------------------------------------------------------------

component adder\_eight\_bit

port (

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

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

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

C\_out : out std\_logic

);

end component;

--------------------------------------------------------------------------------

signal not\_A, test : std\_logic\_vector(7 downto 0);

signal extra : std\_logic;

begin

not\_A <= (not A);

Adder\_4 : adder\_eight\_bit

port map (

A => not\_A,

B => "00000001",

Y => Y,

C\_out => extra

);

end Behavioral;

----------------------------------------------------------------------------------

-- Company:

-- Engineer: Nathan A Ruprecht

--

-- Create Date: 11:19:09 10/03/2016

-- Design Name:

-- Module Name: subtract\_eight\_bit - Behavioral

-- Project Name:

-- Target Devices:

-- Tool versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--

----------------------------------------------------------------------------------

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx primitives in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity subtract\_eight\_bit is

port (

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

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

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

C\_out : out std\_logic

);

end subtract\_eight\_bit;

architecture Behavioral of subtract\_eight\_bit is

--------------------------------------------------------------------------------

component adder\_eight\_bit

port (

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

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

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

C\_out : out std\_logic

);

end component;

--------------------------------------------------------------------------------

component twos\_complement

port (

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

Y : out std\_logic\_vector(7 downto 0)

);

end component;

--------------------------------------------------------------------------------

signal s\_Y0 : std\_logic\_vector(7 downto 0);

signal s\_C\_out : std\_logic;

begin

TWOSCOMP\_2 : twos\_complement

port map(

A => B,

Y => s\_Y0

);

Adder\_3 : adder\_eight\_bit

port map (

A => A,

B => s\_Y0,

Y => Y,

C\_out => s\_C\_out

);

end Behavioral;