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

Daniel Ackuaku

April 4, 2019.

ENGR 304 Lab 7

Hardware Lab 1

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

I used VHDL code to receive input of two numbers from the switches on the DE2 board, display the ASCII equivalent HEX value of the respective inputs, sum the two inputs and display the result of the two numbers.

I learned the difference between VHDL and other programming languages we have encountered in previous labs. VHDL unlike C or Assembly is encoding actual hardware components with instructions.

Testing strategy.

I tested the addition of the following ASCII values:

30 + 31

30 + 32

30 + 3x (….9)

These values are considered normal values. This ensured that the basic logic was working as I expected.

Next, I tested the upper bounds of the addition to make sure the carry over bit displays appropriately:

46 + 46

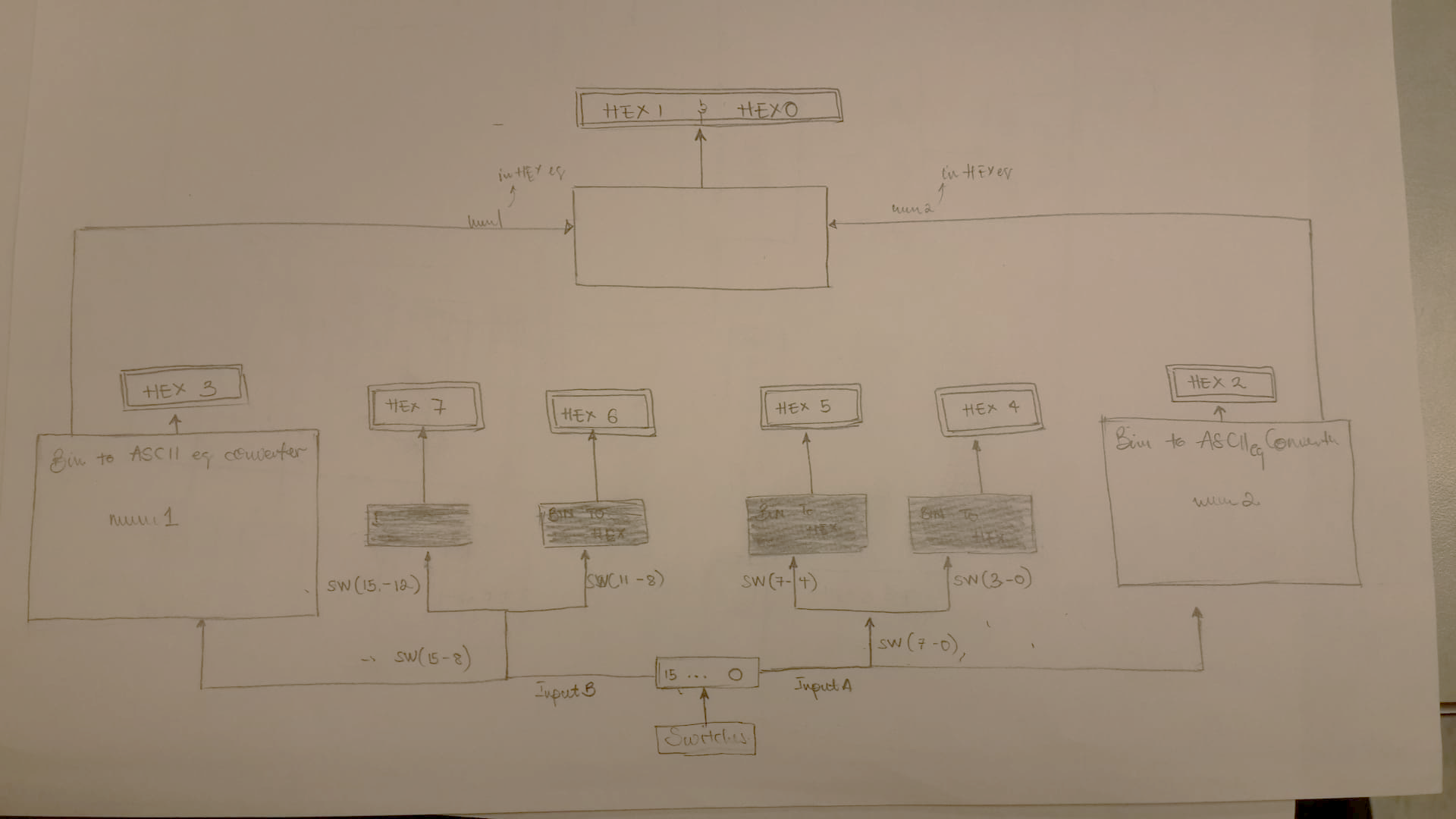


Figure 1 System block diagram

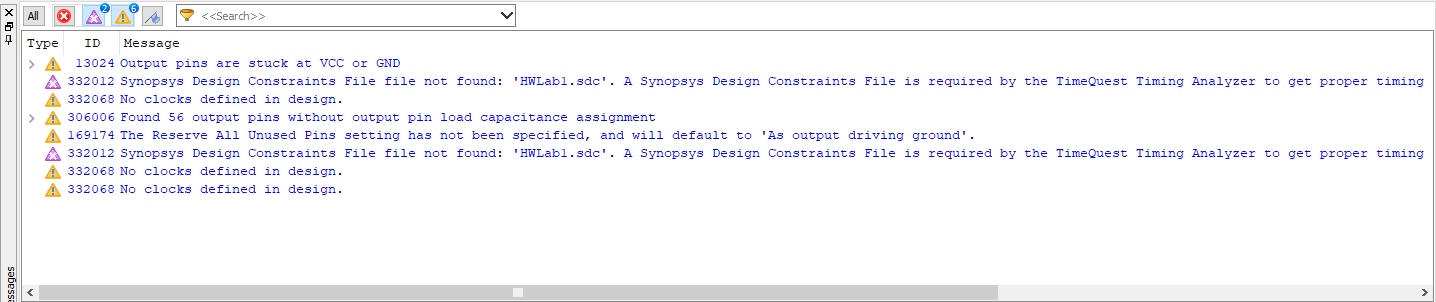


Figure 2 screenshot of the Quartus compiler warnings

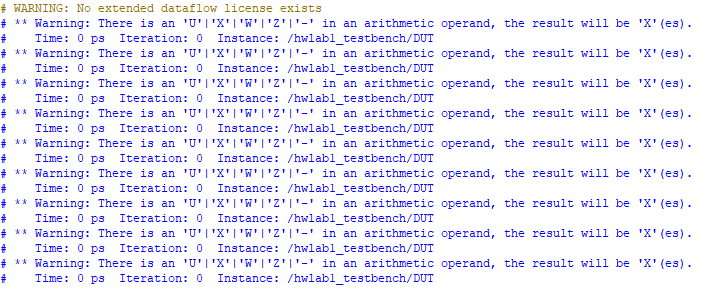


Figure 3 Screenshot that shows the Modelsim command window after the running of the testbench.

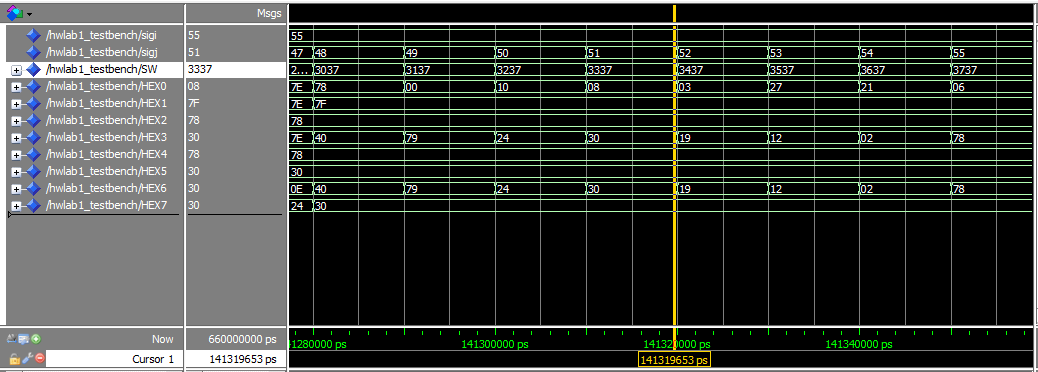


Figure 4 Modelsim simulation waveforms

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

-- DANIEL ACKUAKU

-- FILE: HWLab1

-- This file contains VHDL that implements Lab 1 for the Altera DE2 Board.

-- DESCRIPTION:

-- In this lab, the switches are used to input digits. HEX displays are used

-- to show the hex value of the operation result and the operands.

-- COURSE: Engineering 304 - Spring 2018

-- DESIGN TOOL: HEX\_TO\_ASCII\_EQUI

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

-- MODIFICATION HISTORY:

-- Revision 1.1 Fill\_In

-- Fill\_In

-- Revision 1.0 3/27/18 2:00 PM Prof. Brouwer

-- File template as supplied by the professor.

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

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

-- Add a USE statement for the package containing function calls that support

-- the arithmetic operations on unsigned data. Similar libraries are available

-- for signed (2's complement) arithmetic operations.

USE ieee.std\_logic\_unsigned.ALL;

-- Add a USE statement for the package containing function calls that support

-- the 7-segment displays on the DE2 board.

USE work.SevenSeg\_pkg.ALL;

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

-- ENTITY: HWLab1

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

ENTITY HWLab1 IS

PORT (

-- The entity port names are int ed to match the names found in the pin

-- assignment file. Matching the names makes pin assignment easier, but

-- it is not required.

SW : IN std\_logic\_vector(15 downto 0);

HEX0 : OUT std\_logic\_vector(6 downto 0);

HEX1 : OUT std\_logic\_vector(6 downto 0);

HEX2 : OUT std\_logic\_vector(6 downto 0);

HEX3 : OUT std\_logic\_vector(6 downto 0);

HEX4 : OUT std\_logic\_vector(6 downto 0);

HEX5 : OUT std\_logic\_vector(6 downto 0);

HEX6 : OUT std\_logic\_vector(6 downto 0);

HEX7 : OUT std\_logic\_vector(6 downto 0)

);

END HWLab1;

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

-- ARCHITECTURE: behav

-- This architecture is implemented with behavioural VHDL

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

ARCHITECTURE behav OF HWLab1 IS

-- This area is used to define types and any internal signals

-- Fill\_In as necessary to define signals like "Result", etc

signal BIN\_TO\_HEX\_ASCII\_EQU1 : std\_logic\_vector(7 downto 0);

signal BIN\_TO\_HEX\_ASCII\_EQU2 : std\_logic\_vector(7 downto 0);

signal num1 : std\_logic\_vector(4 downto 0);

signal num2 : std\_logic\_vector(4 downto 0);

signal Result : std\_logic\_vector(4 downto 0);

BEGIN

-- concurrent signal assignments placed here

-- Assign internal signals to external port names (e.g. SW15..8 = InputB)

-- use the conversion function to display the least signif 4-bits of Result

HEX7 <= convert\_to\_7seg(SW(15 downto 12));

HEX6 <= convert\_to\_7seg(SW(11 downto 8));

HEX5 <= convert\_to\_7seg(SW(7 downto 4));

HEX4 <= convert\_to\_7seg(SW(3 downto 0));

-- Fill\_In as necessary to drive the remaining HEX displays

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

-- process

-- Based on the binary value of the fisrt 8 SWs. A 4bit ASCII equivalent HEX

-- is generated.

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

process (SW) is

BEGIN

BIN\_TO\_HEX\_ASCII\_EQU1 <= SW(7 downto 0);

end process ;

process (BIN\_TO\_HEX\_ASCII\_EQU1) is

BEGIN

CASE BIN\_TO\_HEX\_ASCII\_EQU1 IS

WHEN "00110000" => num1 <= "00000" ; -- 30 in HEX to EQU ASCII of 0

WHEN "00110001" => num1 <= "00001" ; -- 31 in HEX to EQU ASCII of 1

WHEN "00110010" => num1 <= "00010" ; -- 32 in HEX to EQU ASCII of 2

WHEN "00110011" => num1 <= "00011" ; -- 33 in HEX to EQU ASCII of 3

WHEN "00110100" => num1 <= "00100" ; -- 34 in HEX to EQU ASCII of 4

WHEN "00110101" => num1 <= "00101" ; -- 35 in HEX to EQU ASCII of 5

WHEN "00110110" => num1 <= "00110" ; -- 36 in HEX to EQU ASCII of 6

WHEN "00110111" => num1 <= "00111" ; -- 37 in HEX to EQU ASCII of 7

WHEN "00111000" => num1 <= "01000" ; -- 38 in HEX to EQU ASCII of 8

WHEN "00111001" => num1 <= "01001" ; -- 39 in HEX to EQU ASCII of 9

WHEN "01100001" => num1 <= "01010" ; -- 61 in HEX to EQU ASCII of a

WHEN "01100010" => num1 <= "01011" ; -- 62 in HEX to EQU ASCII of b

WHEN "01100011" => num1 <= "01100" ; -- 63 in HEX to EQU ASCII of c

WHEN "01100100" => num1 <= "01101" ; -- 64 in HEX to EQU ASCII of d

WHEN "01100101" => num1 <= "01110" ; -- 65 in HEX to EQU ASCII of e

WHEN "01100110" => num1 <= "01111" ; -- 66 in HEX to EQU ASCII of f

WHEN "01000001" => num1 <= "01010" ; -- 41 in HEX to EQU ASCII of A

WHEN "01000010" => num1 <= "01011" ; -- 42 in HEX to EQU ASCII of B

WHEN "01000011" => num1 <= "01100" ; -- 43 in HEX to EQU ASCII of C

WHEN "01000100" => num1 <= "01101" ; -- 44 in HEX to EQU ASCII of D

WHEN "01000101" => num1 <= "01110" ; -- 45 in HEX to EQU ASCII of E

WHEN "01000110" => num1 <= "01111" ; -- 46 in HEX to EQU ASCII of F

WHEN OTHERS => num1 <= "11111"; -- error handling

END CASE;

end process;

process (num1) is

BEGIN

if (num1 = "11111") THEN

HEX2 <= "1111110"; -- display err mark (-)

ELSE

HEX2 <= convert\_to\_7seg(num1(3 downto 0));

end if;

end process;

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

-- process

-- Based on the binary value of the last 8 SWs. A 4bit ASCII equivalent HEX

-- is generated.

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

process (SW) is

BEGIN

BIN\_TO\_HEX\_ASCII\_EQU2 <= SW(15 downto 8);

end process ;

process (BIN\_TO\_HEX\_ASCII\_EQU2) is

BEGIN

CASE BIN\_TO\_HEX\_ASCII\_EQU2 IS

WHEN "00110000" => num2 <= "00000" ; -- 30 in HEX to EQU ASCII of 0

WHEN "00110001" => num2 <= "00001" ; -- 31 in HEX to EQU ASCII of 1

WHEN "00110010" => num2 <= "00010" ; -- 32 in HEX to EQU ASCII of 2

WHEN "00110011" => num2 <= "00011" ; -- 33 in HEX to EQU ASCII of 3

WHEN "00110100" => num2 <= "00100" ; -- 34 in HEX to EQU ASCII of 4

WHEN "00110101" => num2 <= "00101" ; -- 35 in HEX to EQU ASCII of 5

WHEN "00110110" => num2 <= "00110" ; -- 36 in HEX to EQU ASCII of 6

WHEN "00110111" => num2 <= "00111" ; -- 37 in HEX to EQU ASCII of 7

WHEN "00111000" => num2 <= "01000" ; -- 38 in HEX to EQU ASCII of 8

WHEN "00111001" => num2 <= "01001" ; -- 39 in HEX to EQU ASCII of 9

WHEN "01100001" => num2 <= "01010" ; -- 61 in HEX to EQU ASCII of a

WHEN "01100010" => num2 <= "01011" ; -- 62 in HEX to EQU ASCII of b

WHEN "01100011" => num2 <= "01100" ; -- 63 in HEX to EQU ASCII of c

WHEN "01100100" => num2 <= "01101" ; -- 64 in HEX to EQU ASCII of d

WHEN "01100101" => num2 <= "01110" ; -- 65 in HEX to EQU ASCII of e

WHEN "01100110" => num2 <= "01111" ; -- 66 in HEX to EQU ASCII of f

WHEN "01000001" => num2 <= "01010" ; -- 41 in HEX to EQU ASCII of A

WHEN "01000010" => num2 <= "01011" ; -- 42 in HEX to EQU ASCII of B

WHEN "01000011" => num2 <= "01100" ; -- 43 in HEX to EQU ASCII of C

WHEN "01000100" => num2 <= "01101" ; -- 44 in HEX to EQU ASCII of D

WHEN "01000101" => num2 <= "01110" ; -- 45 in HEX to EQU ASCII of E

WHEN "01000110" => num2 <= "01111" ; -- 46 in HEX to EQU ASCII of F

WHEN OTHERS => num2 <= "11111"; -- error handling

END CASE;

END process;

process (num2) is

BEGIN

IF (num2 = "11111") THEN

HEX3 <= "1111110"; -- display err mark (-)

ELSE

HEX3 <= convert\_to\_7seg(num2(3 downto 0));

END IF;

END process;

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

-- process

-- Sum

-- this block calculates the Sum of the hex digits num1 and num2

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

process (num1, num2) is

BEGIN

IF ( num1 = "11111" or num2 = "11111") THEN

Result <= "00000";

ELSE

Result <= num1 + num2;

END if;

end process;

process (Result, num1, num2) is

BEGIN

IF ( num1 = "11111" or num2 = "11111") THEN

HEX1 <= "1111110";

HEX0 <= "1111110";

-- If else pair that takes care of the carry bit.

ELSIF (Result < "10000") THEN -- If result is less than 16.

HEX0 <= convert\_to\_7seg(Result(3 downto 0));

HEX1 <= (others => '1'); -- turn off HEX1.

ELSE

HEX0 <= convert\_to\_7seg(Result(3 downto 0));

HEX1 <= "1111001"; -- set HEX1 to 1.

END IF;

end process;

END behav;