Mini Project #6 & 7: Mini-Multiplier

TWO Due dates

Part I: beginning of class Friday 3/24/17

Final Report: beginning of class Friday 3/31/17

Notes:

1. This should be written up like a lab assignment. Write your document in Word (or similar) and document your code and input/output waveforms in figures.
2. There should be a separate figure in your report for each VHDL file, each VHDL package file, and each timing diagram as well as any additional requested figures.
3. Use the **Cyclone II** target architecture in Quartus II.
4. Timing diagrams should be plotted on timescales less than 400 ns.
5. Small letter variables in this assignment represent bits, capital letter variables are vectors. They should be plotted as such in timing diagrams (since this is a multiplier with 8 input bits and 10 output bits, it makes the most sense to display the vectors in HEX)
6. You must use the provided structures to do the required logic work. You cannot use process statements in the main VHDL file to simply perform the work for you. This means structural, not behavioral code in the top-level VHDL file.

VHDL Assignment:

Design an 8-bit multiplier circuit using structural code in VHDL (not behavioral!) that can multiply an input value $W(w_7 - w_0)$ by 1, 2, 3, or 4 depending on the input signals s_1 and s_1 as shown below:

Input $s_1 s_0$	Result P
0 0	$W \times 1$
0 1	$W \times 2$
1 0	$W \times 3$
1 1	$W \times 4$

Restrictions

You only have the following to make your multiplier work!

- Four (4) 4-bit left-right shifters (from project #5)
- One (1) 8-bit ripple carry adder (from project #4)
- One (1) (1-bit) full adder (from project #4)
- Four (4) 8-bit vectorized 4-to-1 multiplexers (see below)
- Four (4) 4-to-1 multiplexers
- Twenty (20) (*total*) AND, OR, NOT, NAND, NOR, XOR gates

You do not have to use *all* of the provided structures.

SUMMARY OF SIGNALS

$W = w_7 w_6 w_5 w_4 w_3 w_2 w_1 w_0$	input vector to be multiplied
$P = p_9 p_8 p_7 p_6 p_5 p_4 p_3 p_2 p_1 p_0$	output vector result (product)
s_1, s_0	input bits to control multiplication

Show:

Part I: beginning of class Friday 3/13/15

a. (**40 points**) A schematic for your multiplier design with all input/output signals (PowerPoint or draw.io are actually fairly handy for this![1]) **DO NOT DO THIS BY HAND AND SCAN IT**. If it is not neat and clear, **YOU WILL LOSE POINTS**. Include a brief description (a convincing one) describing how the circuit works.

b. (**15 point BONUS**) Do the job with only two (2) of your 4-bit left-right shifters and NO added logic gates. You can use whatever other hardware is left.

Part II: beginning of class Friday 3/27/15

c. (**20 points**) Include VHDL code (possibly multiple files) that performs the functionality of the circuit schematic in (a). Instantiate your sub-structures by using package files as needed. Include all code for the main circuit and any sub-circuits you created. Include all package files. If you use code from the book for a sub-circuit, reference the appropriate figure.

d. (**40 points**) Run a functional simulation with input/output vectors and control bit waveforms proving functionality of the multiplier. Think about how you would have a set of test cases that would ensure everything is working correctly. What bits need to be checked for functionality under what conditions assuming that your sub-structures work properly?

[1] http://www.eehomepage.com/symbols.ppt may be helpful for this.

e. Include everything from part (a) as part of your write-up.

Appendix:

The following VHDL structure is provided for use in this assignment.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

-- Structure works the same as a normal 4to1 MUX
-- except it controls 8 bits in a vector on all four inputs
-- and the output.

ENTITY vector8_mux4to1 IS
     PORT(w0,w1,w2,w3    :IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
               s1,s0     :IN  STD_LOGIC;
               f         :OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END vector8_mux4to1;

ARCHITECTURE Behavior OF vector8_mux4to1 IS
BEGIN
     PROCESS(w0,w1,w2,w3,s1,s0)
     -- Pick the input to present at the output based on
     -- choice of s1 and s0.
     BEGIN
          IF (s1 = '0') AND (s0 = '0') THEN
               f <= w0;
          ELSIF (s1 = '0') AND (s0 = '1') THEN
               f <= w1;
          ELSIF (s1 = '1') AND (s0 = '0') THEN
               f <= w2;
          ELSE
               f <= w3;
          END IF;
     END PROCESS;
END Behavior;
```

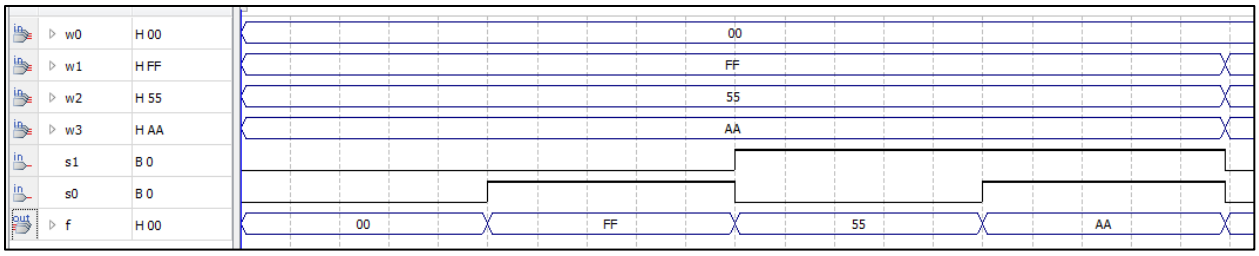

Figure 1 Output waveform showing functionality of the 8-bit vectorized 4-to-1 multiplexer