ECE 385

EXPERIMENT #5

An 8-Bit Multiplier in SystemVerilog

I. OBJECTIVE

In this experiment, you will design a multiplier in SystemVerilog for two 8-bit 2's compliment numbers and then run that multiplier on the DE2 FPGA board.

II. INTRODUCTION

You will use a simple add-shift algorithm to multiply two numbers. The algorithm is very similar to the pencil-and-paper method of multiplication except the final step for 2's Complement numbers depends on the sign bit. Consider the following example to calculate 8-bit 00000111 (7, Multiplicand) x 11000101 (-59, Multiplier)

```
   00000111                7 (multiplicand)
 x 11000101          x (-)59 (multiplier)
   00000111            (-)413
  +00000000x
 +00000111xx
+00000000xxx
+00000000xxxx
+00000000xxxxx
+00000111xxxxxx
-00000111xxxxxxx   Subtract (or Add 2's comp of 00000111)
1111111001100011 (2's comp of result=0000000110011101=413)
```

Let us see how to perform multiplication using the add-shift method that you will use to multiply the contents of register B and switches S, leaving the result in registers AB:

Initial Values: X = 0, A = 00000000, B = 11000101 (achieved using ClearA_LoadB signal), S = 00000111, M is the least significant bit of the multiplier (Register B).

Function	X	A	*B*	M	Comments for the next step
Clear A, LoadB	0	0000 0000	*11000101*	1	Since M = 1, multiplicand (available from switches S) will be added to A.
ADD	0	0000 0111	*11000101*	1	Shift XAB by one bit after ADD complete
SHIFT	0	0000 0011	1 *1100010*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	0	0000 0001	11 *110001*	1	Add S to A since M = 1.
ADD	0	0000 1000	11 *110001*	1	Shift XAB by one bit after ADD complete
SHIFT	0	0000 0100	011 *11000*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	0	0000 0010	0011 *1100*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	0	0000 0001	00011 *110*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	0	0000 0000	100011 *11*	1	Add S to A since M = 1
ADD	0	0000 0111	100011 *11*	1	Shift XAB by one bit after ADD complete
SHIFT	0	0000 0011	1100011 *1*	1	Subtract S from A since 8^{th} bit M = 1.
SUB	1	1111 1100	1100011 *1*	1	Shift XAB after SUB complete
SHIFT	1	1111 1110	01100011	1	8^{th} shift done. Stop. 16-bit Product in AB.

In the ADD state, the values of A and S are first sign-extended to 9 bits, and then summed together. The 9-bit results (not including the Cout) are then stored into XA. In the SHIFT state, the entire 17 bits of XAB is arithmetically right-shifted by one bit.

When M = 0, an ADD does not need to be performed. In that case, the ADD cycle can be omitted or a zero can be added to A. In addition, since we are using a 2's complement representation, we need to consider negative numbers. If A is negative, then XA will contain the correct partial sum and the sign will be preserved since the shift operation will perform an arithmetic shift on XAB. If B is negative (the most significant bit = 1), then M will be 1 after the seventh shift (see the example above). In that case a subtract operation is performed since the 8^{th} bit of B has negative weight with 2's complement representation.

The 9-bit Adder/Subtractor should be designed using Full Adder primitives that you create. In other words, do not use the available SystemVerilog arithmetic operations "+" (add) and "-" (subtract) for this experiment. In future, you may use these operations in your designs.

You should design your control unit such that it executes one multiply operation when the Run press button is pressed. You can use symbolic states for the state machine in the controller for this experiment. You will need to have a Reset input that will reset the controller in the initial/start state. An **incomplete** block diagram of the circuit is shown in Figure 1:

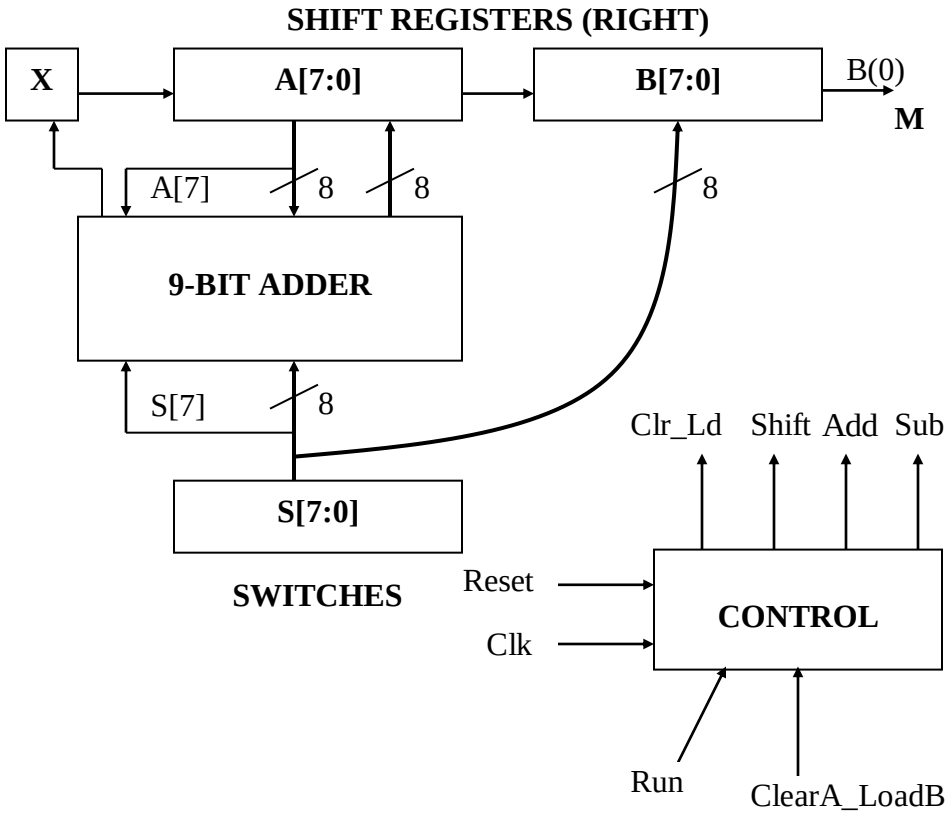

Figure 1: Incomplete Block Diagram

Your circuit should have the following inputs and outputs:

Inputs

S – logic [7:0]

Clk, Reset, Run, ClearA_LoadB – logic

Outputs

AhexU, AhexL, BhexU, BhexL – logic [6:0]

Aval, Bval – logic [7:0]

X –logic

To perform a multiplication, you will first load the multiplier to Register B by setting the switches (S) to represent the multiplier and pressing the ClearA_LoadB button. ClearA_LoadB button should also clears the X and A registers. Then you will set the switches (S) to represent

the multiplicand and press the Run button. ClearA_LoadB should be released before Run button is pushed. Once the Run signal triggers the multiplication, the circuit should complete the multiply operation regardless of the status of Run signal. The circuit should stop once the multiplication is done and the correct result should be displayed by outputting AB on the hex displays. Another multiply operation can be triggered by releasing the Run button and pressing it again.

Your circuit should support consecutive multiplications to receive full demo points. Note that neither ClearA_LoadB nor Reset buttons will be pressed between consecutive presses of the Run button, so your circuit needs to clear X and A before the next multiplication execution starts to get the correct results.

Demo Points Breakdown:

1.0 point: Functional simulation completed successfully

1.0 point: Correct operation of the *Clear_A_Load_B* function on the DE2 board

1.0 point: Correct operation of the *Multiplication* function on the DE2 board. (++, +-, -+, --)

1.0 point: Correct operation of *consecutive* multiplications on the DE2 board. (like -1 × -1 × …)

1.0 point: Execution cycle responds correctly (exactly one execution per press of the "Run" button)

III. PRE-LAB

A. Rework the 8-bit multiplication example presented in the table form at the beginning of this assignment. Use Multiplier B = 7, and Multiplicand S = -59. Note that this is different than the case when B = -59 and S = 7.

B. Design, document, and implement the 8-bit multiplier in SystemVerilog.

You will need to bring the following to the lab:

1. Your code for the 8-bit multiplier. You can bring the code to the lab using a USB storage device, FTP, or any other method.

2. Block diagram of your design, with components, ports, and interconnections labeled.

3. A simulation of your design showing at least one full multiplication. You should set the radix of the Switches, Aval, and Bval signals to signed decimal for readability. (Radix is set by right-clicking on a signal and selecting *Properties*.)

IV. LAB

Follow the Lab 5 demo information on the course website.

Pin Assignment Table

Port Name	Location	Comments
Clk	PIN_Y2	50 MHz Clock from the on-board oscillators
Run	PIN_R24	On-Board Push Button (KEY3)
ClearA_LoadB	PIN_N21	On-Board Push Button (KEY2)
Reset	PIN_M23	On-Board Push Button (KEY0)
S[0]	PIN_AB28	On-board slider switch (SW0)
S[1]	PIN_AC28	On-board slider switch (SW1)
S[2]	PIN_AC27	On-board slider switch (SW2)
S[3]	PIN_AD27	On-board slider switch (SW3)
S[4]	PIN_AB27	On-board slider switch (SW4)
S[5]	PIN_AC26	On-board slider switch (SW5)
S[6]	PIN_AD26	On-board slider switch (SW6)
S[7]	PIN_AB26	On-board slider switch (SW7)
AhexL[0]	PIN_AA25	On-Board seven-segment display segment (HEX2[0])
AhexL[1]	PIN_AA26	On-Board seven-segment display segment (HEX2[1])
AhexL[2]	PIN_Y25	On-Board seven-segment display segment (HEX2[2])
AhexL[3]	PIN_W26	On-Board seven-segment display segment (HEX2[3])
AhexL[4]	PIN_Y26	On-Board seven-segment display segment (HEX2[4])
AhexL[5]	PIN_W27	On-Board seven-segment display segment (HEX2[5])
AhexL[6]	PIN_W28	On-Board seven-segment display segment (HEX2[6])
AhexU[0]	PIN_V21	On-Board seven-segment display segment (HEX3[0])
AhexU[1]	PIN_U21	On-Board seven-segment display segment (HEX3[1])
AhexU[2]	PIN_AB20	On-Board seven-segment display segment (HEX3[2])
AhexU[3]	PIN_AA21	On-Board seven-segment display segment (HEX3[3])
AhexU[4]	PIN_AD24	On-Board seven-segment display segment (HEX3[4])
AhexU[5]	PIN_AF23	On-Board seven-segment display segment (HEX3[5])
AhexU[6]	PIN_Y19	On-Board seven-segment display segment (HEX3[6])
BhexL[0]	PIN_G18	On-Board seven-segment display segment (HEX0[0])
BhexL[1]	PIN_F22	On-Board seven-segment display segment (HEX0[1])
BhexL[2]	PIN_E17	On-Board seven-segment display segment (HEX0[2])
BhexL[3]	PIN_L26	On-Board seven-segment display segment (HEX0[3])
BhexL[4]	PIN_L25	On-Board seven-segment display segment (HEX0[4])
BhexL[5]	PIN_J22	On-Board seven-segment display segment (HEX0[5])
BhexL[6]	PIN_H22	On-Board seven-segment display segment (HEX0[6])
BhexU[0]	PIN_M24	On-Board seven-segment display segment (HEX1[0])
BhexU[1]	PIN_Y22	On-Board seven-segment display segment (HEX1[1])
BhexU[2]	PIN_W21	On-Board seven-segment display segment (HEX1[2])
BhexU[3]	PIN_W22	On-Board seven-segment display segment (HEX1[3])
BhexU[4]	PIN_W25	On-Board seven-segment display segment (HEX1[4])
BhexU[5]	PIN_U23	On-Board seven-segment display segment (HEX1[5])
BhexU[6]	PIN_U24	On-Board seven-segment display segment (HEX1[6])

X	PIN_F17	On-Board LED (LEDG8)

(Assignments for Aval and Bval are intentionally omitted. These outputs are included primarily for simulation, where reading the hex display outputs is not practical. Similarly, reading the outputs of your circuit in binary is not as practical as reading them in hex. You are free to reuse the assignments from Lab 4 for these signals, if you wish.)

V. POST-LAB

1.) Refer to the Design Resources and Statistics in IQT.30-32 and complete the following design statistics table.

LUT	
DSP	
Memory (BRAM)	
Flip-Flop	
Frequency	
Static Power	
Dynamic Power	
Total Power	

Come up with a few ideas on how you might optimize your design to decrease the total gate count and/or to increase maximum frequency by changing your code for the design.

Before the midnight after your lab, create a **zip** file of the code for your multiplier (with comments where appropriate) and email it to your TA with the subject line "ECE 385 – Lab 5 Code". **Include only your source code** (.sv files). The other files in your project directory are generated by Quartus II and will only serve to annoy your TA by artificially filling up his/her inbox. Name your zip file in the format of ***ece385_lab5_netid1_netid2.zip***. Please use the **zip** format only. Other compression formats won't be accepted. (TAs are under no obligation to accept late code, code that hasn't been zipped, or code files that are intermixed with other project files.)

VI. REPORT

In your lab report, should hand in the following:

- An introduction;
- Rework the 8-bit multiplication example;
- Written description of the operation of your circuit;
- Written purpose and operation of each module, including the inputs/outputs of the modules;
- State diagram for your controller;
- Schematic block diagram with components, ports, and interconnections labeled;
- Annotated pre-lab simulation waveforms (4 simulations: +*+, +*-, -*+, -*-);
- Answers to post-lab questions;
- A conclusion regarding what worked and what didn't, with explanations of any possible causes and the potential remedies.