

ECE 385

Fall 2019

Experiment #5

An 8-Bit Multiplier in SystemVerilog

Eric Dong, Yifu Guo

ericd3, yifuguo3

Section AB3, Thursday 2pm

Gene Shiue

1. Introduction

- a. The multiplier circuit that we designed this week is able to take in two 2's complement binary numbers and compute the multiplication which returns the correct signed answer. The multiplier takes in two 8 bit numbers and returns a sign extended 16 bit answer which is displayed on the led array.

2. Pre-lab question

- a. Since our circuit isn't designed to pick which state to go to, we will write out all the states that the FSM has to go through

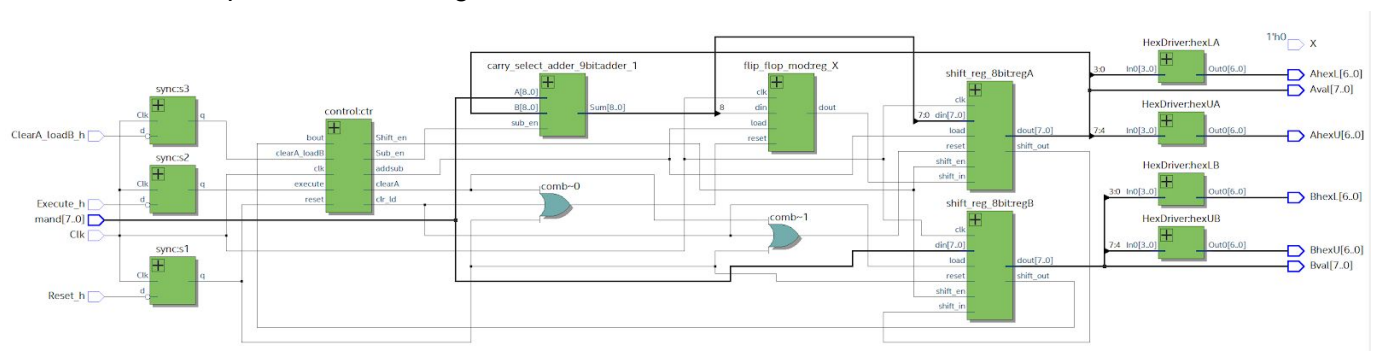
Function	X	A	B	M
Clear A load B	0	00000000	11000101	1
ADD	0	00000111	11000101	1
SHIFT	0	00000011	11100010	0
ADD	0	00000011	11100010	0
SHIFT	0	00000001	11110001	1
ADD	0	00001000	11110001	1
SHIFT	0	00000100	01111000	0
ADD	0	00000100	01111000	0
SHIFT	0	00000010	00111100	0
ADD	0	00000010	00111100	0
SHIFT	0	00000001	00011110	0
ADD	0	00000001	00011110	0
SHIFT	0	00000000	10001111	1
ADD	0	00000111	10001111	1
SHIFT	0	00000011	11000111	1
ADD	1	11111100	11000111	1
SHIFT	1	11111110	01100011	1

3. Written description and diagrams of multiplier circuit

- a. Summary of operation
 - i. Operands are loaded based on the bit that we're working on as well as whether the least significant bit of register B is a 1 or 0. If the least

significant bit of register B is a 0, the multiplier will not allow register A to load the answer, but when the least significant bit of register B is a 1, the control allows register A to be loaded with the result of the addition of register A plus the number being multiplied. When the control is adding the 8th bit of the input number, if that bit is a 1, then the control outputs a subtraction signal that tells the 9 bit adder to do a subtraction instead of addition, then the result of that addition is once again stored into register A. The 9 bit adder is a combination of a 4 bit adder and a 5 bit adder. When a subtraction operation is required, the number being multiplied is XORed against that subtraction enable bit and addition is performed. In order for the subtraction operation to result in the correct 2's complement number, the carry-in bit of the adder is set to the subtraction enable bit so that the sum is right.

b. Top Level Block Diagram



c. Written Description of .sv Modules

i.

Module	Inputs	Outputs
shift_reg_8bit	clk, reset, load, shift_en, shift_in, [7:0] din	shift_out, [7:0] dout
sync	Clk, d	q
carry_select_adder_9bit	sub_en, A, B,	Sum, CO
control	clearA_loadB, reset, execute, clk, bout	Shift_en, Sub_en, clr_ld, addsub, clearA
flip_flop_mod	clk, din, load, reset	dout
HexDriver	In0	Out0
multiplier	[7:0] mand, Clk, Reset_h, Execute_h,	[6:0] AhexU, AhexL, BhexU, BhexL, [7:0]

	ClearA_loadB_h	Aval, Bval, X
--	----------------	---------------

Shift_reg_8bit:

- Description: this is a positive-edge triggered 8bit register with asynchronous reset and synchronous load. When load is high, data is loaded from din into the register on the positive edge of clk.
- Purpose: this module is used to create the registers that store operands A or B in the adder circuit.

Sync:

- Description: this is a D flip-flop. The data will go from D to Q when it is on the positive edge of clock.
- Purpose: we use this module as a synchronizer to negate the reset, execute and clearA_loadB signal synchronously.

Carry_select_adder_9bit:

- Description: this is a 9-bit carry_select_adder.
- Purpose: this module is used to perform the operation of add or subtract.

Control:

- Description: this module is a state machine.
- Purpose: we use this module to control the operation of adders and registers.

Flip_flop_mod:

- Description: this is a positive-edge triggered flip-flop.
- Purpose: we use this flip-flop to store the extension bit X.

HexDriver:

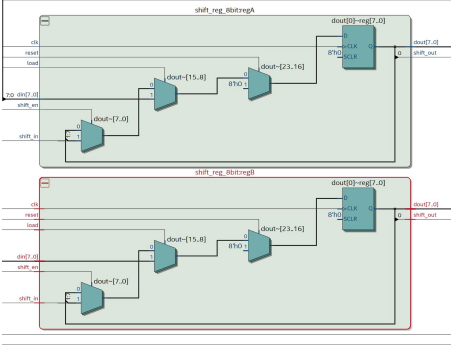
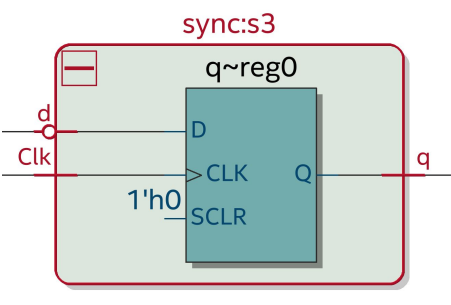
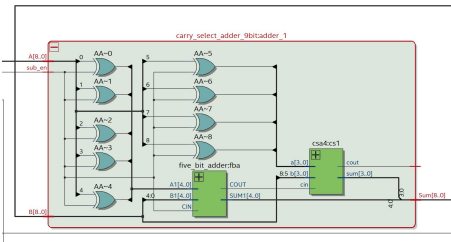
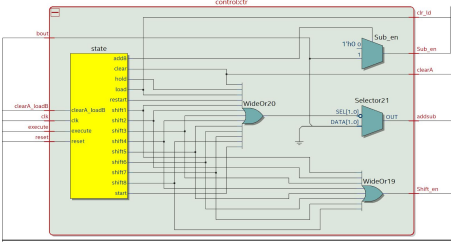
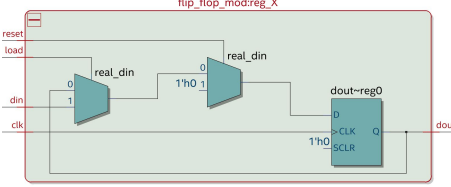
- Description: this is a hex driver.
- Purpose: it helps to present the output hex-value on the board.

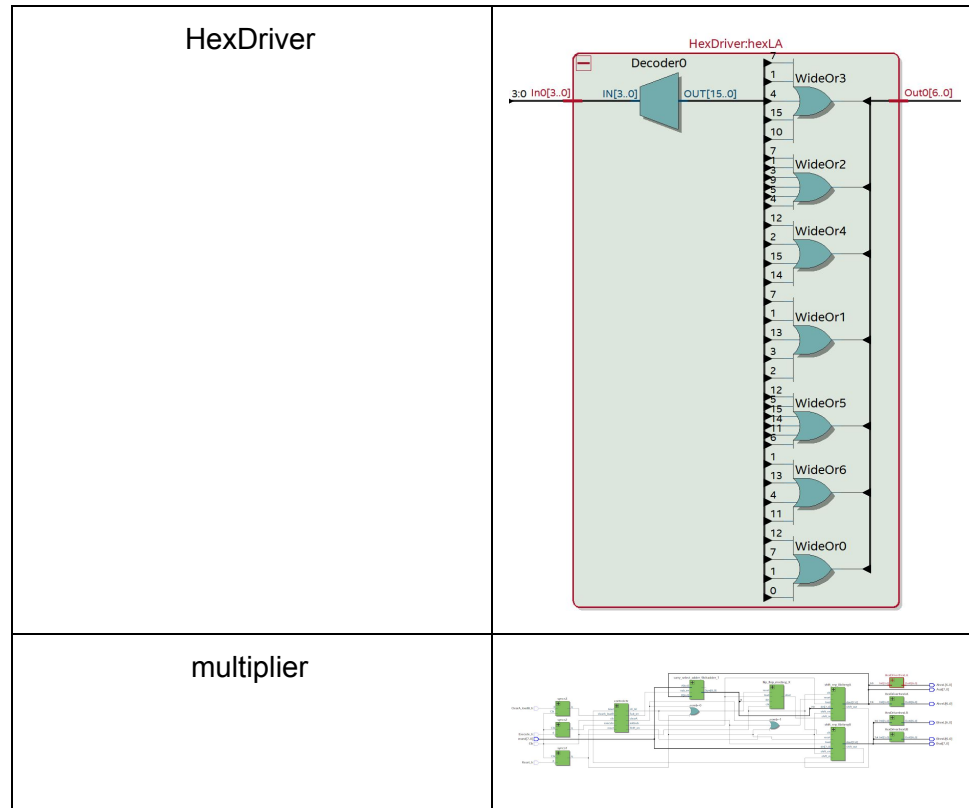
Multiplier:

- Description: it is a top-level module.
- Purpose: this module organizes and works with other modules to perform the intended operation.

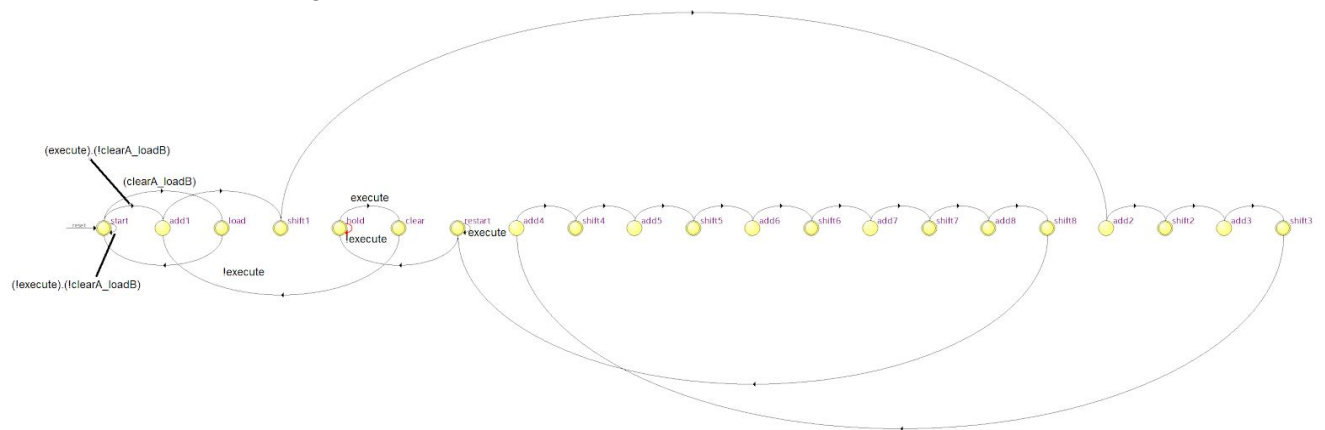
ii.

Module	Picture
--------	---------

<p>shift_reg_8bit</p>	 <p>The diagram shows two identical 8-bit shift registers, labeled 'shift_reg_8bitregA' and 'shift_reg_8bitregB'. Each register has inputs for 'clk', 'reset', 'load', 'din', and 'en'. The output is 'dout'. The internal structure shows a chain of 8 D flip-flops. The first flip-flop's D input is 'din'. The output of each flip-flop is connected to the D input of the next flip-flop. The output of the last flip-flop is 'dout'. The 'en' input is connected to the 'en' input of each flip-flop.</p>
<p>sync</p>	 <p>The diagram shows a D flip-flop labeled 'q~reg0'. The D input is 'd'. The CLK input is 'Clk'. The SCLR input is '1'h0'. The Q output is 'q'.</p>
<p>carry_select_adder_9bit</p>	 <p>The diagram shows a 9-bit carry-select adder, labeled 'carry_select_adder_9bitadder_1'. It has inputs 'A[8:0]', 'B[8:0]', and 'cin'. The output is 'sum[8:0]'. The circuit uses a 5-bit adder to calculate the sum of the lower 5 bits of A and B. The result is then used to select between two 4-bit adders for the upper 4 bits of A and B. The final output is the sum of the two 4-bit adders.</p>
<p>control</p>	 <p>The diagram shows a control unit, labeled 'controlctr'. It has inputs 'load', 'clk', 'en', 'exec', 'reset', and 'state'. The output is 'dout'. The circuit uses a state machine to control the execution of the adder. The state machine has four states: 'idle', 'load', 'exec', and 'reset'. The 'load' state is the initial state. The 'exec' state is reached when the 'exec' input is asserted. The 'reset' state is reached when the 'reset' input is asserted. The 'idle' state is reached when the 'load' input is deasserted.</p>
<p>flip_flop_mod</p>	 <p>The diagram shows a modified D flip-flop, labeled 'flip_flop_mod:reg_X'. It has inputs 'reset', 'load', 'din', and 'clk'. The output is 'dout'. The circuit uses a D flip-flop with a 'real_din' input. The 'real_din' input is connected to 'din' when 'load' is asserted. The 'real_din' input is connected to 'dout' when 'load' is deasserted. The 'clk' input is connected to the 'CLK' input of the flip-flop. The 'reset' input is connected to the 'SCLR' input of the flip-flop.</p>



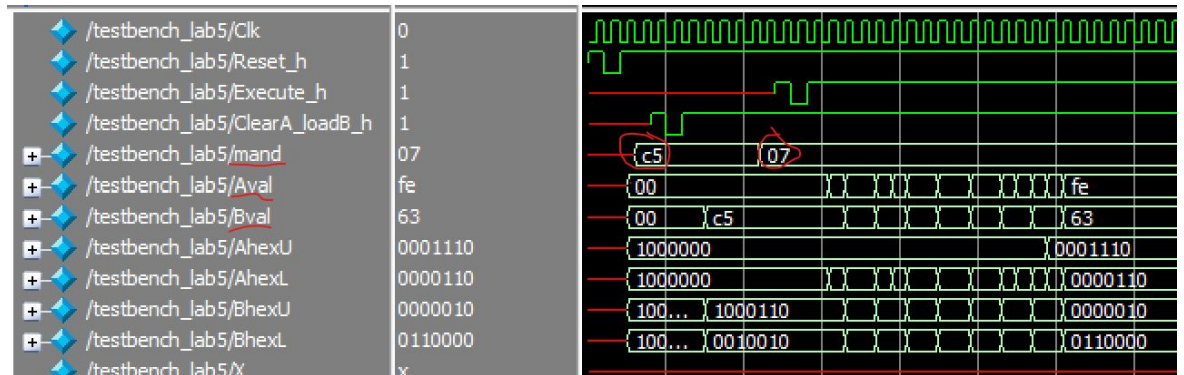
d. State Diagram for Control Unit



4. Annotated pre-lab simulation waveforms

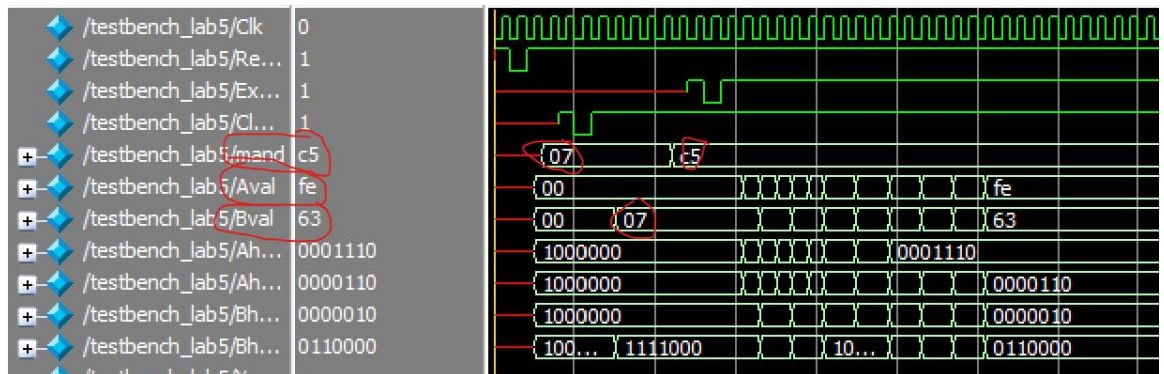
a. $+^*$

The operands are xC5 and x07, the result is xFE63: look at the mand values



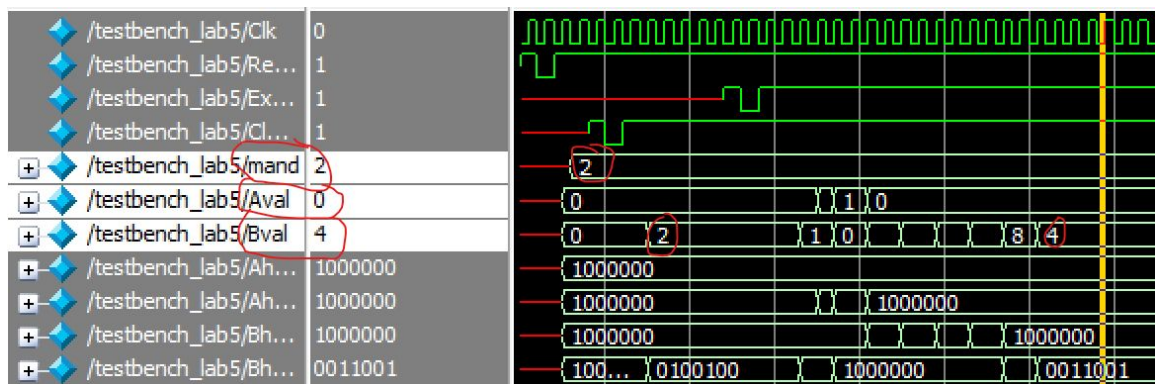
b. $-^*+$

The operands are x07 and xC5, the result is xFE63: look at the mand values



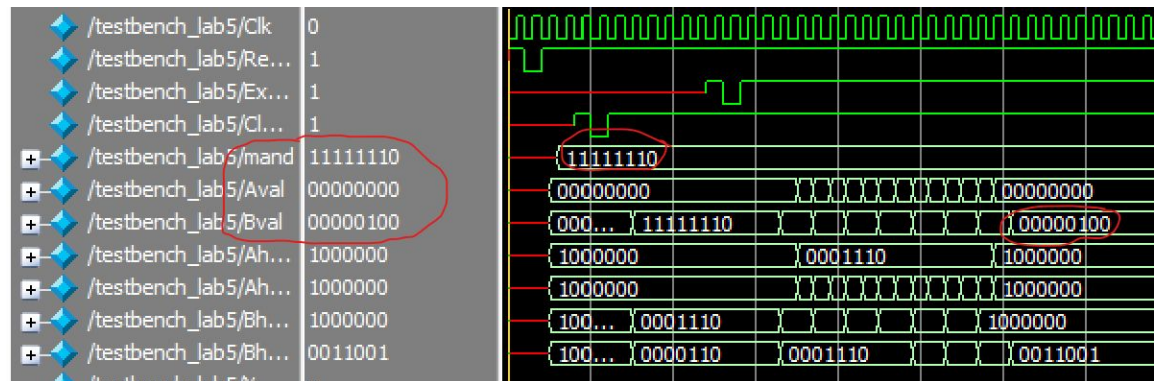
c. $+^{*+}$

The operands are x02 and x02, the result is x0004: look at the mand values



d. -*-

The operands are xFE and xFE, the result is x0004: look at the mand values



5. Answers to post-lab questions

- a. I think in order to optimize the gate usage and other resources, we should change the state diagram so that the next state is dependent on the least significant bit of register B instead of just doing add and shift over and over again. This way there are less states that the machine has to go through.

LUT	90
DSP	N/A
Memory	0
Flip-Flop	9
Frequency	74.14 MHz
Static Power	98.51 mW
Dynamic Power	2.92 mW
Total Power	144.95 mW

- b. The purpose of the register X was to hold the sign extension bit. The register only gets set by the addition of the 9 bit adder. If the addition results in a positive number then the register holds a 0 otherwise a 1.
- c. One of the limitations of continuous multiplication is the overflow problem. When the result of multiplication is more than 16 bits, the overflow will happen and the algorithm will fail.
- d. Compared to the pencil-and-paper method, the implemented algorithm will have the overflow problem but we wouldn't have to keep track of the subtraction problem since it automatically determines whether it should subtract or not, the adder is "smart" and automatically does the subtraction.

6. Conclusion

- a. The function of our design was to build a multiplier to do 8 bit 2's complement multiplication and display that result into 16 bit LEDs. The design was to do sequential additions for each digit of the multiplicand and if the most significant digit of one of the multiplicand were a 1, then the last operation would be changed into a subtraction instead. All the parts of our design worked we just had to perfect our finite state machine and make sure that all the states made sense.
- b. I think that the manual could have done a better job explaining why we had to do 9 bit additions instead of just 8 bits since the sign extension part was not obvious to everyone.