ECE 385

Fall 2023

Experiment #4

An 8-Bit Multiplier in System Verilog

Yunxuan Yang, Yuqi Wen

2023/9/28

TA: Tianhao Yu

1) Introduction

In this experiment, we design a multiplier in System Verilog for two 8-bit 2's compliment numbers and then run that multiplier on the Urbana board. We implement a add-shift algorithm to achieve the goal.

2) Pre-lab question

a. Rework the multiplication example on page 5.2 of the lab manual, as in compute 11000101 * 00000111 in a table like the example.

Function	X	A	*B*	M	Comments for the next step
Clear A, LoadB, Reset	0	0000 0000	*0000 0111*	1	Since M = 1, multiplicand (available from switches S) will be added to A.
ADD	1	1100 0101	*0000 0111*	1	Shift XAB by one bit after ADD complete
SHIFT	1	1110 0010	1*000 0011*	1	Add S to A since M = 1.
ADD	1	1010 0111	1*000 0011*	1	Shift XAB by one bit after ADD complete
SHIFT	1	1101 0011	11*00 0001*	1	Add S to A since M = 1.
ADD	1	1001 1000	11*00 0001*	1	Shift XAB by one bit after ADD complete
SHIFT	1	1100 1100	011*0 0000*	0	Shift XAB by one bit after ADD complete
SHIFT	1	1110 0110	0011 *0000*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	1	1111 0011	0001 1*000*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	1	1111 1001	1000 11*00*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	1	1111 1100	1100 011*0*	0	Do not add S to A since M = 0. Shift XAB.
SHIFT	1	1111 1110	0110 0011	1	8th shift done. Stop. 16-bit Product in AB.

2) Written description of all .SV modules and diagrams of multiplier circuit

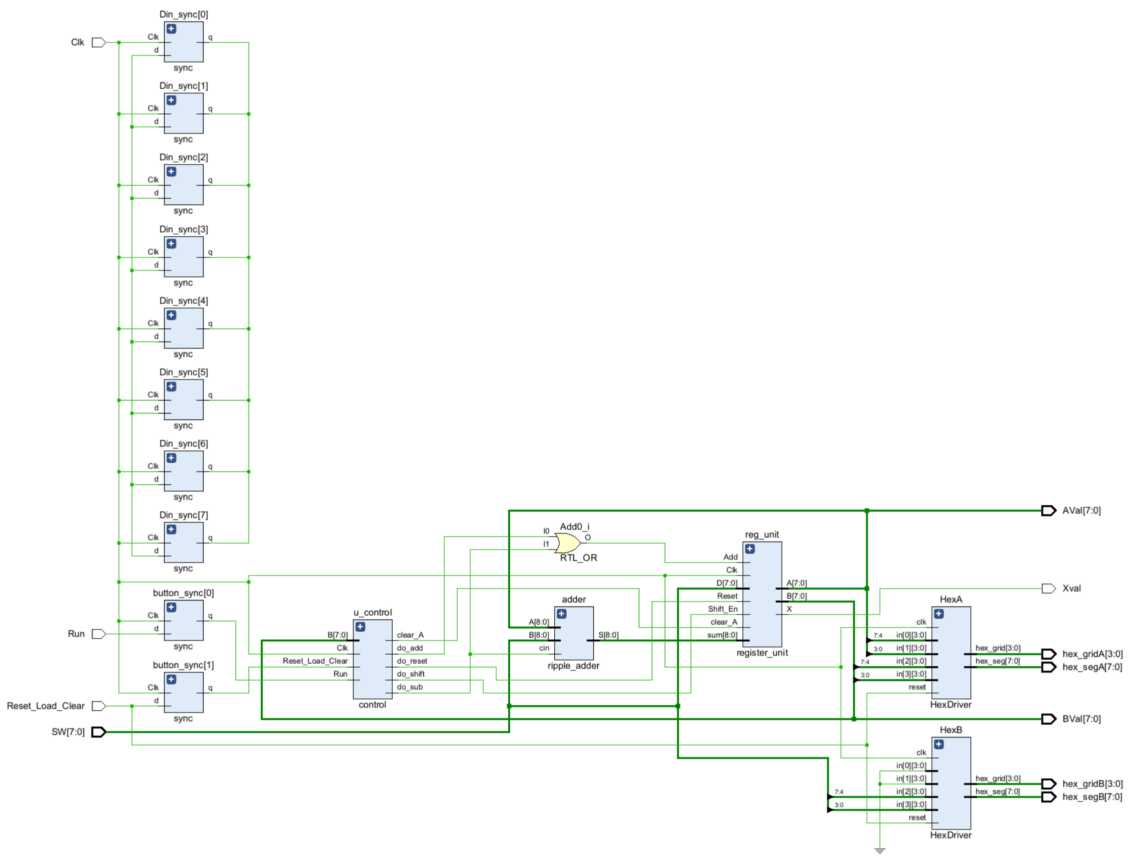
a. Summary of operation

Load: To perform a multiplication, we initially set the switches (S) to represent the multiplier and activate the Reset_Load_Clear button. This action not only inputs the multiplier into Register B but also clears the X and A registers. Next, we adjust the switches (S) for the multiplicand S. Once done, pressing the Run button initiates the calculation.

Compute: The multiplication of B and S utilizes a straightforward add-shift method. Throughout the computation, different states are encountered. In the ADD state, the first step is to inspect the lowest bit of B, referred to as M. If M equals 1, the 9-bit expanded versions of A and S are either added or subtracted, determined by the sign bit, X. The outcome's last 8 bits are then stored in A, with the leading bit going to X. However, if M is 0, no addition is required. Following the ADD state is the SHIFT state, where the combined 17 bits of XAB undergo an arithmetic right-shift by one position. Subsequently, the process reverts to the ADD state. This cycle persists until 16 SHIFT operations have been executed, at which point the result can be found in AB.

Store: After completing the multiplication, the circuit should halt its operations, and the final result, represented by the values in registers A and B, should be displayed on the Hex displays for clarity and confirmation.

b. Top Level Block Diagram



c. Written Description of .sv Modules

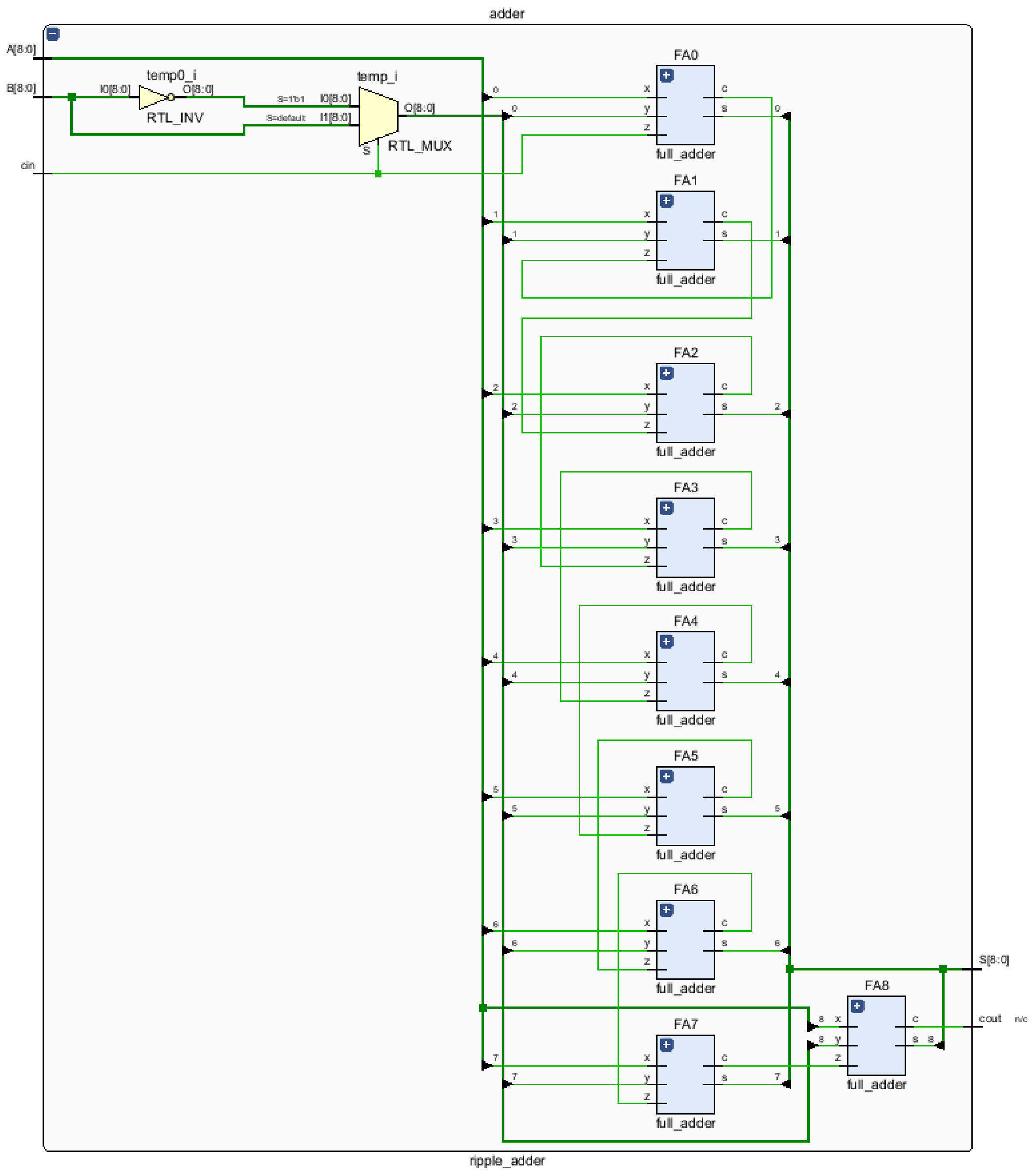
1. Module: ripple_adder.sv

Inputs: [8:0] A, [8;0] B, cin

Outputs: [8:0] S, cout

Description: This module contains two modules: full adder and ripple adder. Declare full adder first and then use it in ripple adder.

Purpose: This module is used to perform 9-bit ripple add calculation.



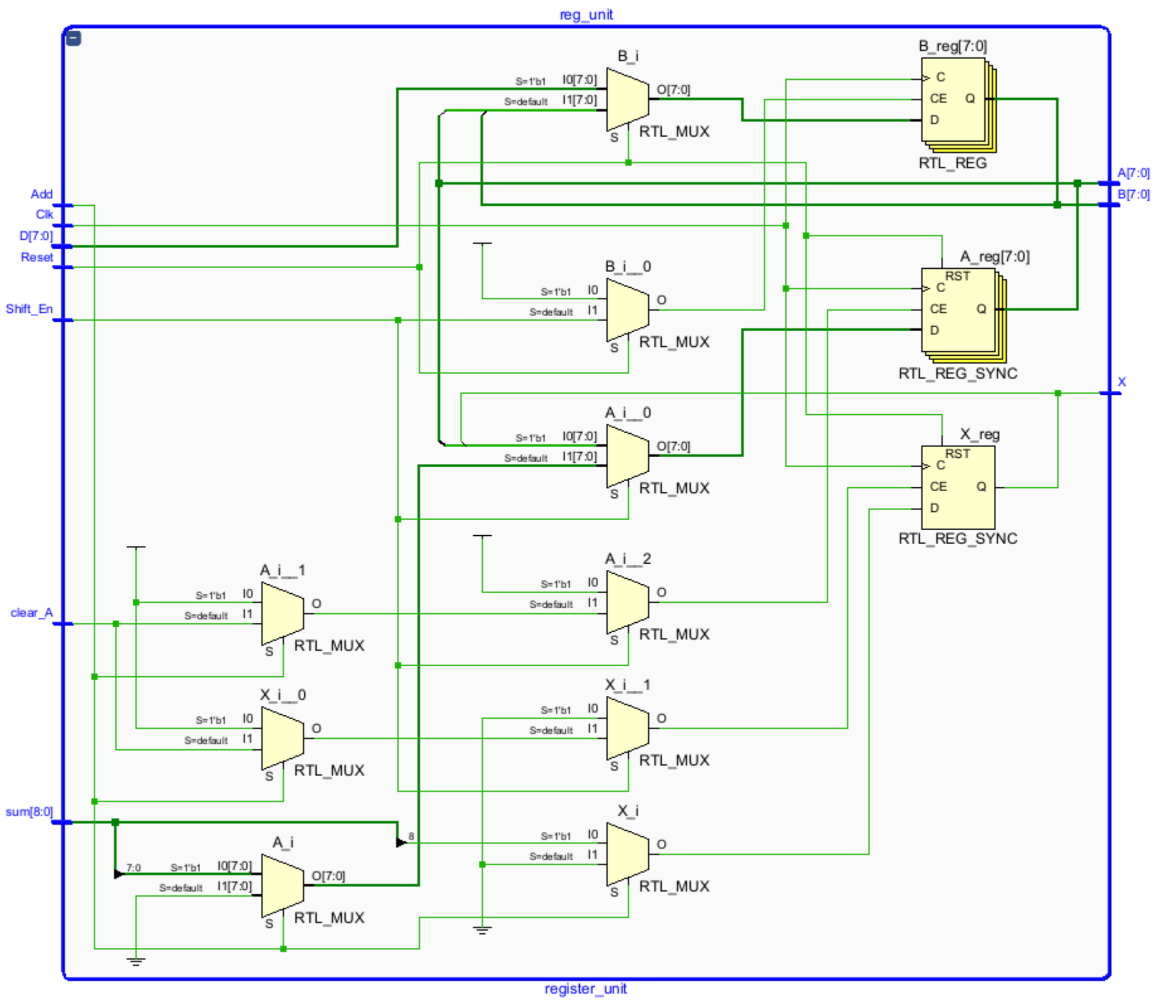
2. Module: Register_unit.sv

Inputs: Clk, Reset, Shift_En, clear_A, Add, [7:0] D, [8:0] sum

Outputs: [7:0] A, [7:0] B, X

Description: This is an 8-bit register with synchronous reset and load capabilities.

Purpose: Store operands A, B, and X in the adder circuit.



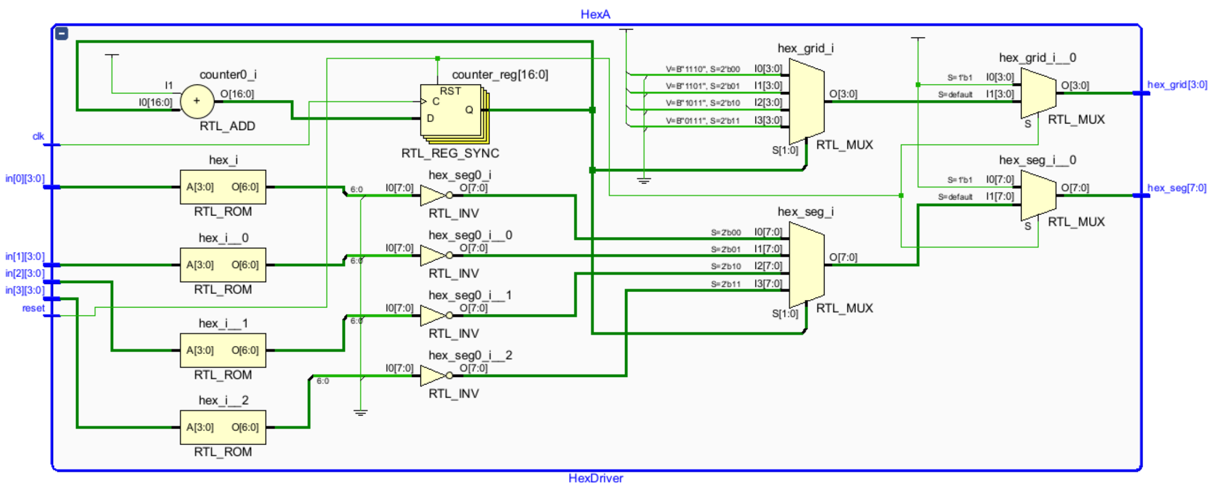
3. Module: HexDriver.sv

Inputs: [3:0] in, clk, reset

Outputs: [7:0] hex_seg, [3:0] hex_grid

Description: This module converts 4bit binary numbers from registers into hexadecimal to display digit on LED.

Purpose: Set the 7-segment LED display on board.



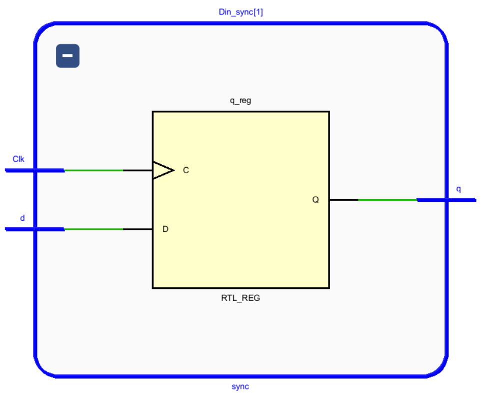
4. Module: Synchronizers.sv

Inputs: Clk, d

Outputs: q

Description: Send signals to FPGA board.

Purpose: These components serve as synchronizers, essential for aligning the input signals received from buttons or switches to ensure proper timing and function within the circuit.



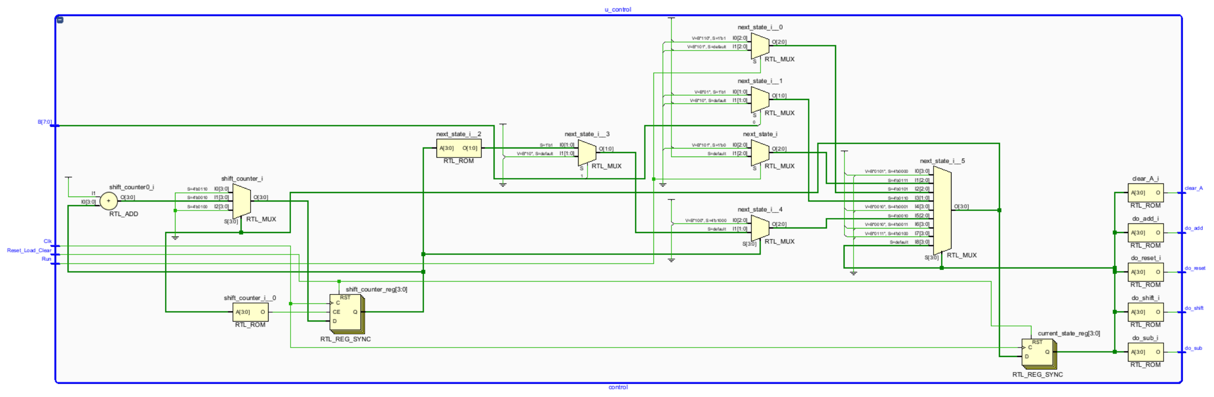
5. Module: Control.sv

Inputs: Clk, Reset_Load_Clear, Run, [7:0] B

Outputs: do_add, do_shift, do_sub, do_reset, clear A

Description: This module uses a Finite State Machine to build a control unit.

Purpose: This module acts as the control unit. It receives input signals from the user and subsequently issues commands to other components within the circuit.



6. Module: multiplier

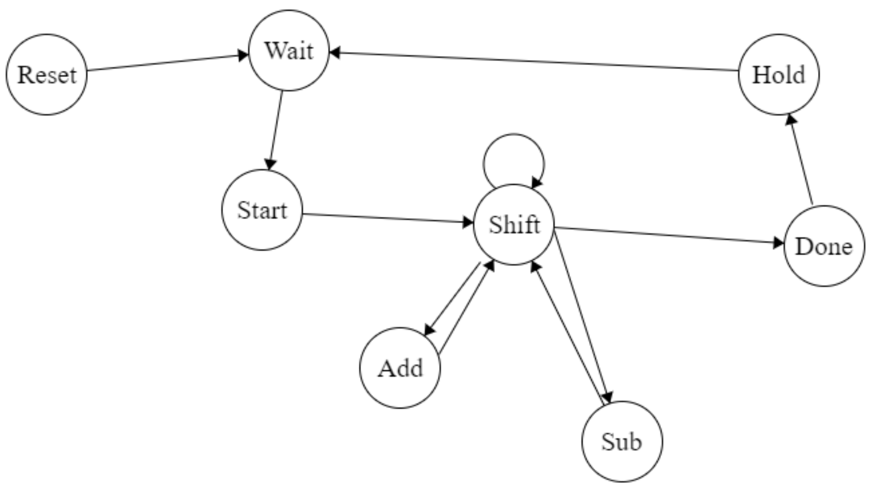
Inputs: Clk, Reset_Clear, Run_Accumulate, [7:0] SW

Outputs: [7:0] hex_segA, [3:0] hex_gridA, [7:0] hex_segB, [3:0] hex_gridB, [7:0] AVal, [7:0] BVal, XVal

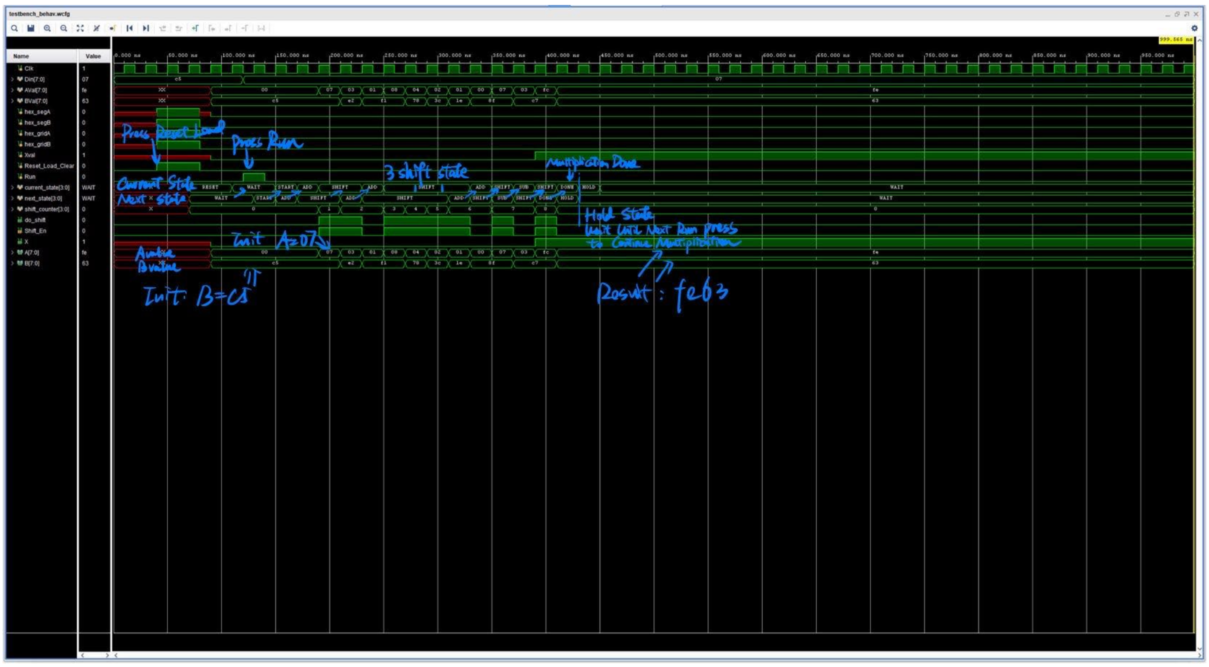
Description: This is a top-level module of the 8-bit multiplier.

Purpose: The module is used as a top-level module to connect each module in this project, receive input signal, compute it with adder, change it to hex representation, and display it on the board.

d. State Diagram for Control Unit



4) Annotated pre-lab simulation waveforms



5) Answers to the post-lab questions

1. Fill in the table

LUT	92
DSP	0
Memory	0
Flip-Flop	43
Latches	0
Frequency	30.3 MHz
Static Power	72 mW
Dynamic Power	40 mW
Total	112 mW

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.
Instead of having distinct ADD and SHIFT states, we could combine them. If M=1, do the addition and then always do the shift. This reduces the FSM complexity.

2. What is the purpose of the X register? When does the X register get set/cleared?
The primary role of the X register is to store the highest bit of the 9-bit sign-extended value from register A (the multiplicand). This specific bit indicates whether the number in register A is positive or negative, given that in a 2's complement representation, the topmost bit represents the sign of the number. The bit is stored in the X register. During the multiplication process, this register is referenced to ascertain whether the subsequent action should be addition, subtraction, or merely a bit shift. The register is updated with each addition or subtraction operation and is initialized to a clear state at the beginning of the computation.

3. What would happen if you used the carry out of an 8-bit adder instead of output of a 9-bit adder for X?
When the X register is set as the carry-out of an 8-bit adder, and the two numbers being added have their most significant bit set to 1, the adder results in a 0 for its most significant bit. According to our procedure, the X bit captures the sign of register A. In the described scenario, X would wrongly indicate 1, even though 0 implies that the number is positive. Situations like this can lead to a loss of information about the sign of register A. As a result, to avoid such inaccuracies, we opt to use the 9th bit from the sign-extended value of register A for the X bit.

4. What are the limitations of continuous multiplications? Under what circumstances will the implemented algorithm fail?
The limit to continuous multiplications in our system arises when the outcome of any operation surpasses 16 bits. The range of number representable using 16 bits with 1bit sign extension is 65535 to -65,536. If the product of any preceding multiplications goes beyond this limit, our method would be ineffective for subsequent multiplication tasks.

5. What are the advantages (and disadvantages?) of the implemented multiplication algorithm over the pencil-and-paper method discussed in the introduction?
The primary benefit of employing the multiplication algorithm we've used, relative to the traditional pencil-and-paper technique, is its increased speed. This is mainly due to the algorithm's ability to bypass the ADD state when it's unnecessary, opting to just shift the bits. For instance, when adding bits 0 and 0, the algorithm avoids the ADD state and directly transitions to the SHIFT state. However, a drawback of this approach is its intricate state machine, making it less straightforward to implement.

6) Conclusion

a. Discuss functionality of your design. If parts of your design did not work, discuss what could be done to fix it.

For the control unit part, initially we didn’t realized that the run button is active low, instead we set the FSM of transition into wait state as receive high signal of run which causing the state enter directly after the done state. To fix it, we determine the state transition by receiving low on the run button.

b. ambiguous, incorrect, or unnecessarily

I think the manual could explain more about the add-shift part, why we have to do 9-bit additions instead of just 8 bits.

c. summary

This lab served as an excellent primer on this particular multiplier design, which was not covered in our earlier courses. We gained an understanding of how a combination of additions, bit shifts, and subtractions can result in a multiplication function. We also delved into the importance of the X and M bits in our circuit, which are pivotal in deciding whether the subsequent operation should be an addition, subtraction, or a shift. In conclusion, we successfully achieved our objective of constructing an 8-bit 2's complement multiplier.