Jonathan Gusler 4/23/18 Final Project: 8-bit Processor

Goal: Modify an 8-bit processor from chapter 7 of the text book (Brown and Vranesic) that can only add and subtract numbers so that it can also multiply numbers.

Process:

1) Copy code from textbook to get the basic processor working
2) Verify correct functionality by setting up same simulation as in the textbook and crosschecking results
3) Modify code to incorporate multiplication functionality using the 8-bit by 8-bit multiplier from MP08 (Gusler).
4) Verify correct functionality via waveform that showcases multiplication and addition arithmetic functions. *both simulations use the load and move functions

Definition of signals:

cin is the carry in to the multiplier

reset sets the counter to 0

w tells the processor to start a function

Data is what data is being given to the processor onto the bus from the outside world

F: Function: 000 is load bus into register Rx, 001 is move data from register Ry to Rx, 010 is add the contents of Rx and Ry and store result in Rx, 011 is subtract the data in Ry from Rx and store result in Rx, 100 is multiply value in Rx by Ry and store the bits 15 to 8 of the result in Rx, and bits 7 to 0 in Ry

Rx is Register x, x being a value from 0 to 3 for registers 0 to 3.
Ry is Register y, y being a value from 0 to 3 for registers 0 to 3.

Done goes high when a function has finished

Countread tells what time step for the function the processor is currently on. Move and Load take 1 time step, add and subtract take 3, multiply takes 4.

BusOut tells what data is on the buswires

R1out – R3out show what data are currently on each of the registers.

The remaining "out" signals show what are currently on each of the remaining registers. A goes into the add/sub unit, G takes the result of the add/sub and puts it onto the bus. M goes into the multiplier, P1 takes bits 15 to 8 of the product from the multiplier and puts it on the bus, P2 takes bits 7 to 0.

Simulations: The first simulation, Figure 1, showcases all 4 functions of the processor from the textbook while the second simulation, Figure 2, showcases all the functions that the modified processor is capable of (basic processor and multiplication). As a note, the second simulation is identical to the first except for including the multiplication function at the end. The third simulation, Figure 3, just shows that the invalid function inputs "101", "110", and "111" don't trigger any math operations.

Walkthrough of the simulations: The first things that occurs is the processor is reset to a completely "dead" state. Next, the value 3C is loaded into register R0, B4 into R1, 19 into R2. Then the value in R2 is moved into R3, but not cleared from R2. Then nothing happens for a couple time steps until 1.05us. At that point the value in R2 is subtracted from the value in R1, with the result stored in R1. The next operation once that finishes is adding the values in R2 and R0 together and the result stored in R2. At this point the first simulation is finished. In the second simulation though, the next process is multiplying the value in R3 by the value in R2.

Results: All functions operate correctly with the math functions' results checked against a hexadecimal calculator.

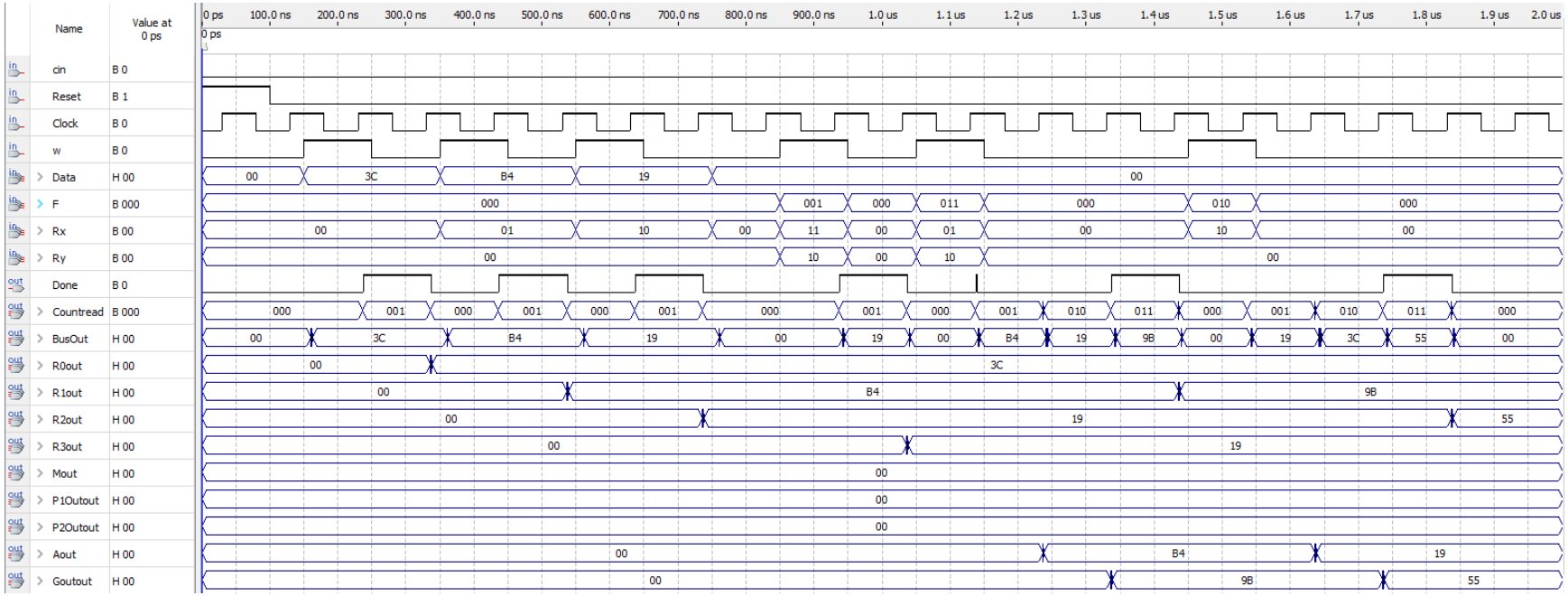

Figure 1: Shows processor simulation featuring the add and subtract functions

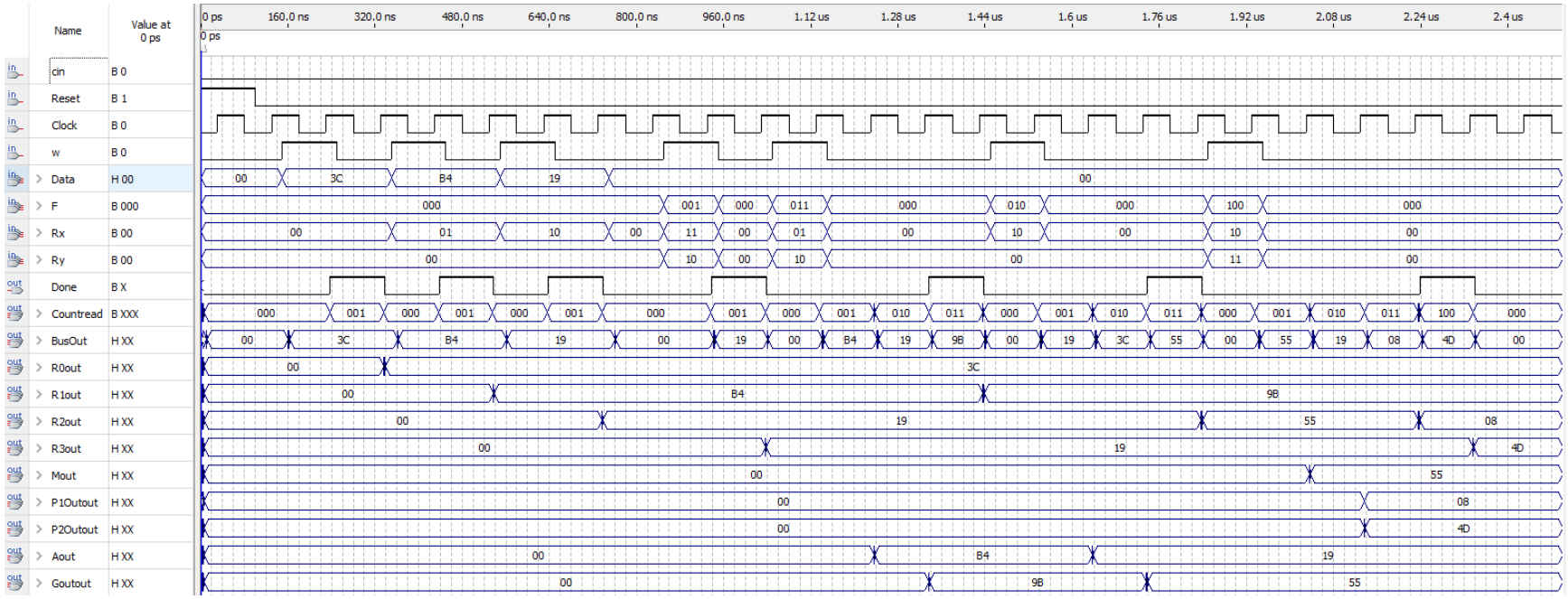

Figure 2: Shows processor simulation featuring the multiplication function in addition to the addition and subtraction functions

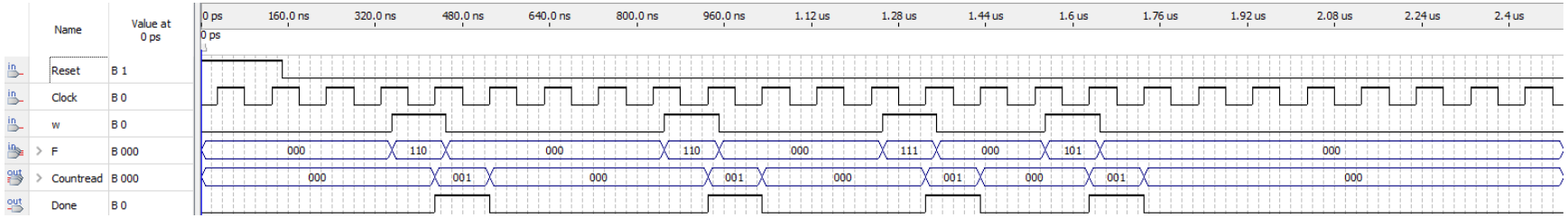

Figure 3: Simulation to show invalid function inputs don't perform operations

References

Brown, Stephen and Zvonko Vranesic. *Fundamentals of Digital Logic with VHDL Design*. McGraw Hill, 2009.

Gusler, Jonathan. "MP08 8-bit Multiplier." VHDL project code. 2018.