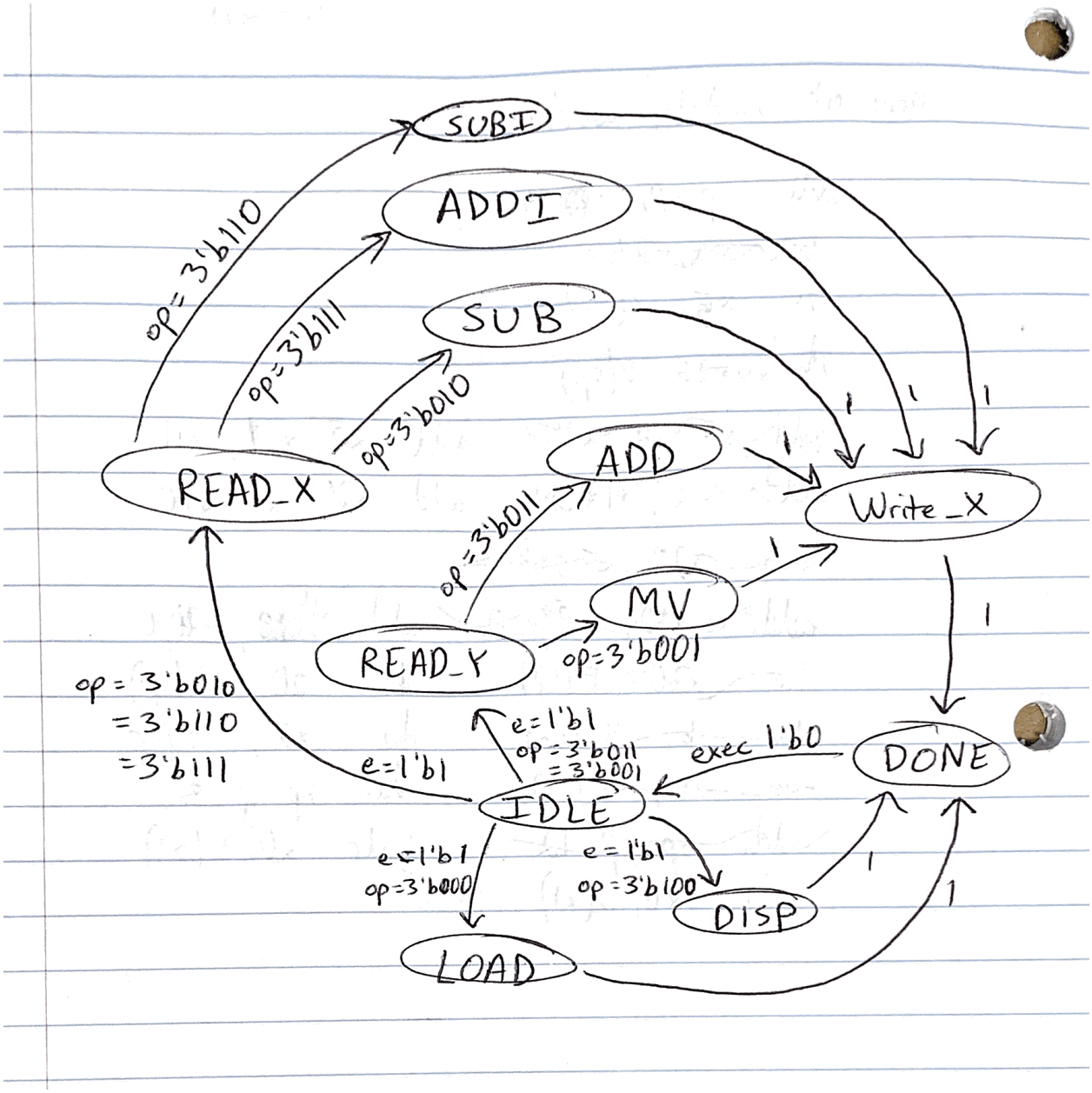
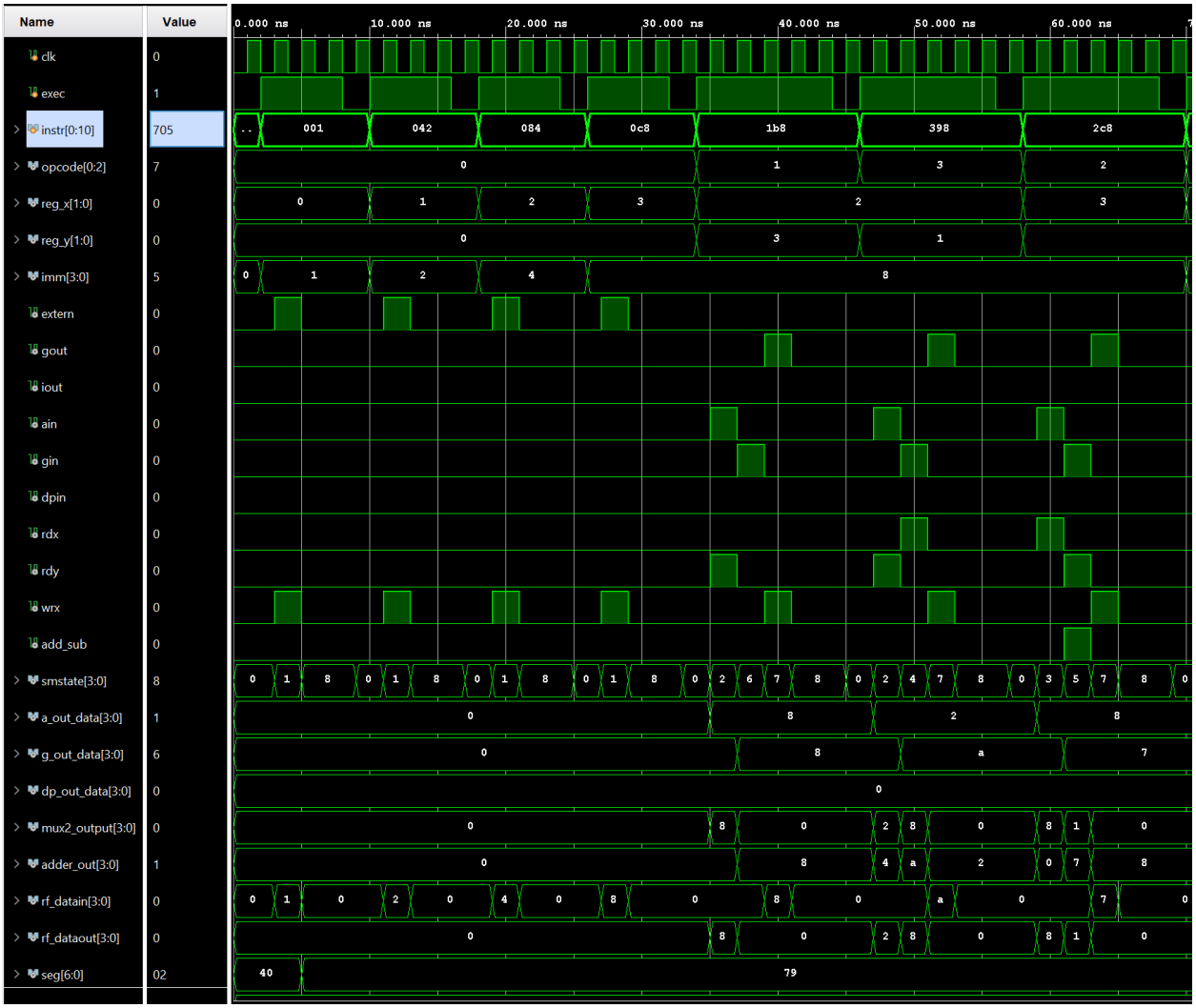
Jack Landers, Michael Kreienkamp

ELEN 122 Lab 3

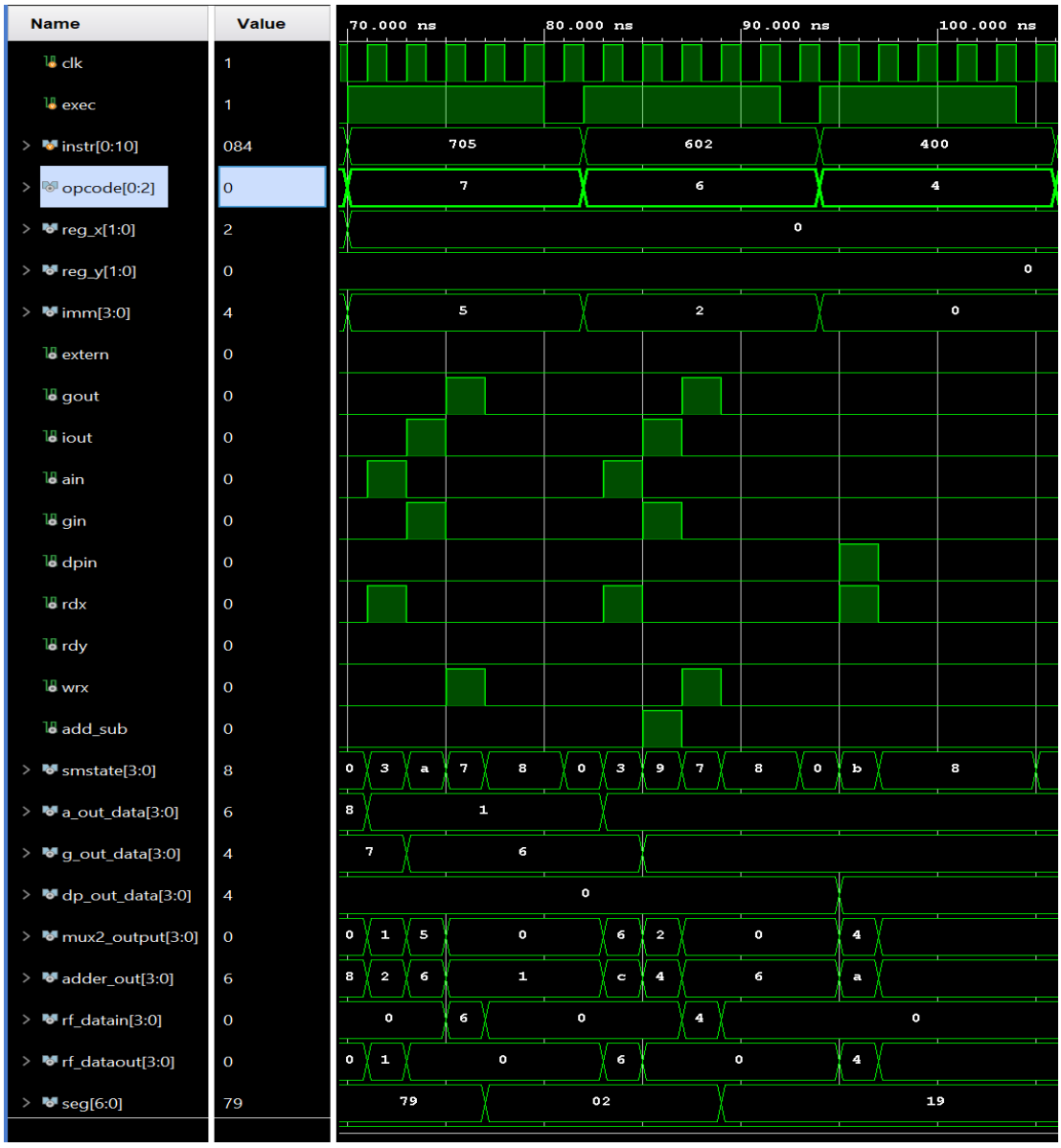
February 9, 2023

Updated State Diagram:





Simulation Picture 1: This simulation data above shows the operations from last week which remained the same for this week's lab. Opcode 0 was used to LOAD, opcode 1 was used for MOVE, opcode 2 was used for SUB, and opcode 3 was used for ADD. All of these operations remained the same with the same things being asserted to reach them.



Simulation Picture 2: This simulation picture shows the new operations which were added for this week's lab. Opcode 7 was ADDI which took the path of states Read_X, ADDI, Write_X, then DONE. You can see that Iout is asserted which means the immediate value=mux2_output and that equals 5. rf_dataout is 1 so 1+5=6 which is shown in adder_out. Opcode 6 was SUBI which took the path of states Read_X, SUBI, Write_X, then DONE. Iout is asserted again and imm=2. rf_dataout is 6 so 6-2=4 which can be seen in adder_out. Opcode 4 is DISP. This is just a single state after IDLE and before DONE which uses RdX and DPin which are clearly both on.

What problems did you encounter while implementing and testing your system?

We had a lot of issues during the actual lab period with debugging because we believed we had all the code, but we could not find why the simulation did not want to run even though the synthesis was working. We probably just had some random typos or punctuation errors. We decided to completely restart, and we worked carefully to use the code which we already had, while making sure not to cut anything we needed or put any wrong punctuation anywhere. Everything ended up working the second time, but we did have to change DISP a little because we had DISP coming after state Read_X and before state Write_X when in reality, DISP only needed RdX and DPin asserted and could come after IDLE then go immediately to DONE.

Did any problems arise when demonstrating for the TA? In other words, did the TA ask you to demonstrate something that you did not think of yourself? What was the scenario that you were asked to demonstrate? Provide some thoughts about why you didn't think of this yourself.

Yes, we had some initial problems explaining what was going on, but after looking at the actual diagram with the datapath, we realized how to explain our simulation. We had some problems with DISP as described above because we thought we had to go through state Read_X when in reality, we should have just asserted RdX in the DISP state. We also did not need to go through state Write_X. We thought DISP would go through other states like ADDI and SUBI did so that is why we didn't immediately think of having DISP go right after IDLE and then go to DONE without any other states being used for the operation. While it can be advantageous to use existing states (like for SUBI's and ADDI's implementation), DISP is so simple that it should not have to reuse any other states. It should be its own state, not needing Read_X or Write_X.

In this design, we moved to a 3-bit operation encoding yet only implemented 7 operations in total. That leaves one encoding unused. Think of another operation you could envision implementing with this unused operation encoding, and describe/explain the datapath changes that would be necessary to support execution of this new operation.

For this operation code, 101, we could make a greater than operation, this operation would also perform a read x and sub execution, but after the fact there would be another datapath where the greater number is output. This operation would work by using the first bit of the output as a selection for a multiplexer that would take in both the same inputs as for the subtraction operation. This works because if x - y and the output is negative, then the first bit would be 1 and select y as the output. If the result was positive instead, the first bit would be 0 and select x as the output. This would work because the outputs would be signed numbers, so a negative number would always start with a 1 and a positive number would always start with a 0. After this, like the other operations, the next state would be done.