ECE 3270

LAB 3 REPORT ADDENDUM

June 13, 2022

Tim Driscoll
Clemson University
Department of Electrical and Computer Engineering
`tdrisco@clemson.edu`

1 Addendum

1.1 Adding Instructions

In order to add additional instructions to the simple processor some minor changes and additions would need to be made to the original design. To start the way the instructions are encoded would need to be adjusted, the current encoding only allows for there to be four instructions. The instructions are encoded as eight bits represented by IIXXXYYY where the two I bits represent the desired instruction 0-3. When adding an additional instruction and additional "I" bit would need to be added. This would result in increasing the size of the instruction register and making sure it is assigned to the correct number of bits from DIN. If just a single instruction was added only one "I" bit would be needed and the lower nine bits of DIN could be used. The next potential change to the simple processor design that would need to be made is then addition of any necessary hardware to carry out the new instruction. This would be similar to the addsub unit that was implemented in the original design. This may also result in the addition of more simple registers similar to the A and G registers shown in Figure 2. The largest adjustments that would need to be made to the simple processor to add an extra instruction would occur in the control unit. There would need to be multiple additions to the Mealy state machine implementation, especially if the new instruction took more then three clock cycles (longer then any instruction currently implemented). The Mealy finite state machine would first need to be adjusted by adding in any necessary if statements and adjusting the current if statements to handle what the next state is assigned to based upon the current state and the given instruction. The preexisting if statements would need to be adjusted to handle the new three bit encoding of the different instructions. If there are more then three clock cycles needed to complete the instruction then a state would need to be added for each additional clock cycle, this would be in the form of T4,T5,etc. These states would then be included in the case statement used to update the current state in the same format that is currently implemented. If statements will determine the instruction and dictate how the current state is updated. After the process to determine the current state is updated the process that updates the output would also need to be updated. Within the case statement used to determine the current state the if statements would need to be adjusted to include what actions need to be taken if the new instruction is selected. This would match the preexisting format and would simply depend on what needs to be asserted during each state to carry out the new instruction. Once all these changes are made the new instruction will be implemented and ready to be executed on the simple processor.

1.2 Moore Design vs Mealy Design

The largest difference between a Moore and Mealy finite state machine design is that a Moore machine's output is only dependent on the current state whereas a Mealy's output is dependent on both the current state and the input. Thus the output of a Moore machine is delayed by a clock cycle and requires more states. A Mealy state machine will update on the transition of states as suppose to a Moore machine which updates when the current state changes. In order to switch the design to a Moore machine the number of states necessary would increase and the output process sensitivity list would only be dependent on the current state. A Moore machine would increase the total number of states because there would need to be a state to represent each of the instructions at each of the different clock cycles (T0, T1, T2, T3). This would result in T0 staying the same, T1 would need to be split into three different states, T2 would need to split into two different states and T3 would remain the same. When looking at figure 1 each of the boxes with unique instructions would have to be represented by a different state. This change would model the Moore design because the output would only be dependent on the current state. To summarize, the changes to the current design in order to create a Moore machine would be the inclusion of the extra states and an update to the output process. Extra states would in turn result in the first process in the control unit being more complex due to more states but the implementation would be the same. Then the output process would be updated to include only the current state (remove the instruction register) in its sensitivity list. This would simplify the overall design of this process because all that would be needed is a case statement to determine the current state and then an update of the output based upon each of the different states. The idea of a simple processor better fits a Mealy implementation. As discussed above a Moore machine would complicate the number of different states needed. Whereas in a Mealy implementation each state can represent a different clock cycle and the instruction register (input) can be paired with this to determine the correct output. A Mealy design is also preferred in the implementation of a simple processor because it is faster and is not delayed by a clock cycle like the Moore implementation is. The Mealy design also allows for asynchronous output generation because a change in the input will result in an update of the output. Like stated before a Moore machine is delayed by a clock cycle because the output is only dependent on the current state. Thus in the implementation of a simple processor a Mealy design is preferred.

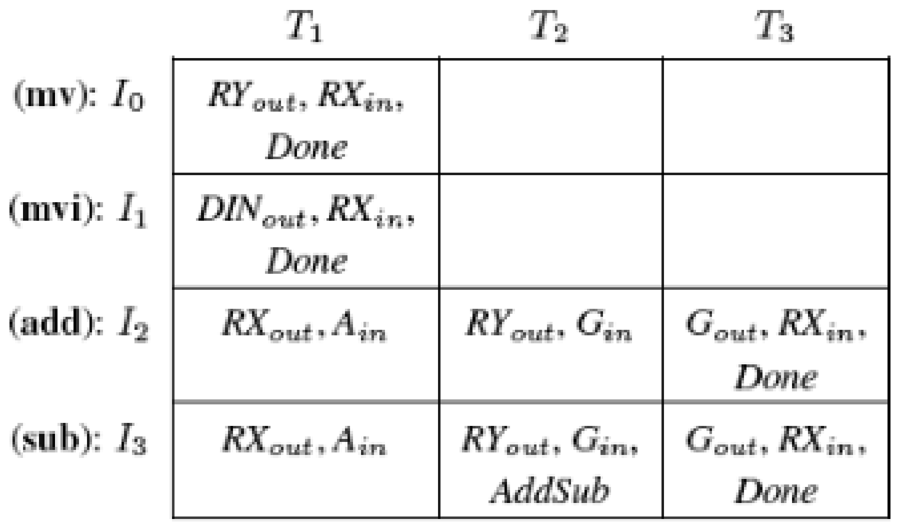
2 References

M.Smith ”Digital Computer Design” Clemson University Holcombe Department of Electrical and Computer Engineering, January 2020

M.Smith ”ECE3270 Digital System Design Lab 3: Simple Processor” Clemson University Holcombe Department of Electrical and Computer Engineering, January 2020

W. Daily, R. Curtis Harting and T. Aamodt, Digital Design Using VHDL a systems approach. Cambridge University Press

3 Appendix



	T_1	T_2	T_3
(mv): I_0	RY_{out}, RX_{in}, *Done*		
(mvi): I_1	DIN_{out}, RX_{in}, *Done*		
(add): I_2	RX_{out}, A_{in}	RY_{out}, G_{in}	G_{out}, RX_{in}, *Done*
(sub): I_3	RX_{out}, A_{in}	RY_{out}, G_{in}, *AddSub*	G_{out}, RX_{in}, *Done*

Figure 1: State Assertions Table

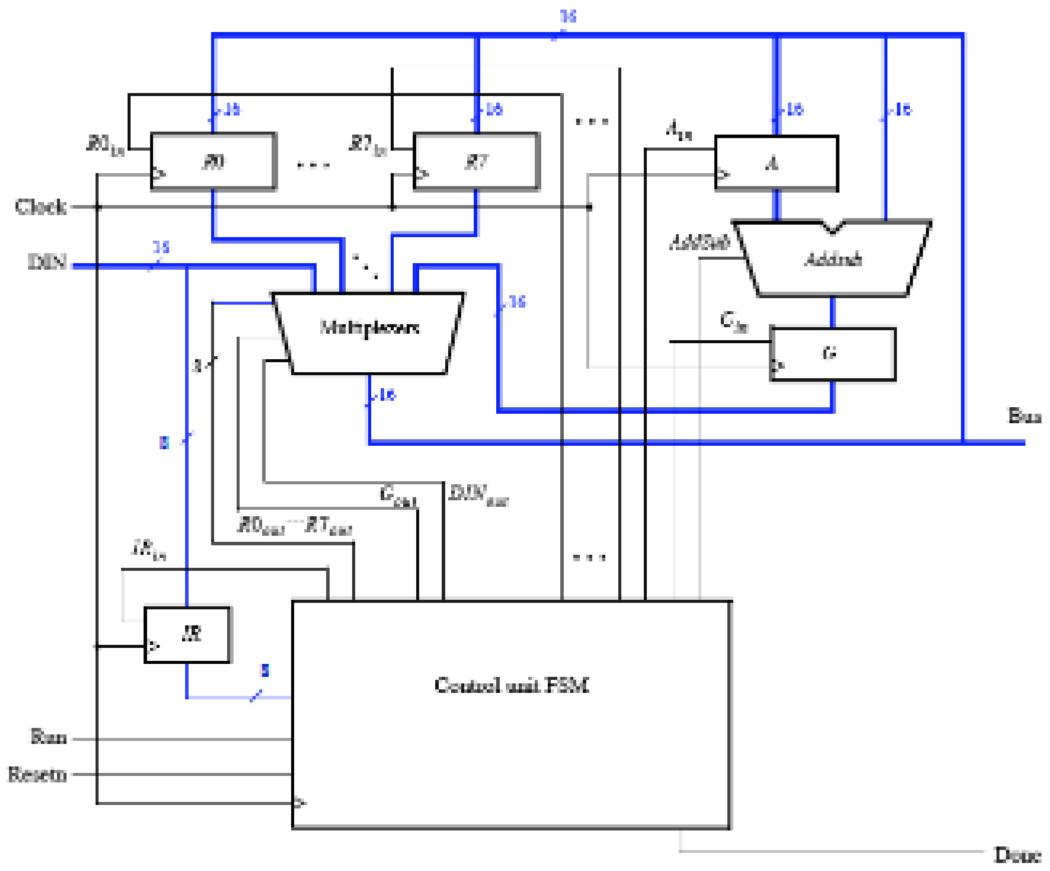


Figure 2: Final Processor Circuit