ECE3270 Digital System Design

Lab 3: Simple Processor

Lab Overview: The purpose of this lab is to implement a simple, 4-instruction processor in VHDL. The top level entity should be a structural design, and all lower level components should be kept completely modular and use VHDL Generics. The processor should be tested in ModelSim with appropriate testbench(es) that test for many cases (+/- and varying ranges). You will complete a full report using the LaTeX template and include discussion about the clock cycle latency of the different instructions as well as the state machine diagram generated by quartus. You will submit all edited files to the assign server as Assignment 3.

Part I

Figure 1 shows the layout of the processor you will design. The blue line represents data-paths and is shows as a 16-bit input. You are to make this data-width **GENERIC** so it is easy to change. **DIN**, **Clock**, **Run** and **Resetn** are the only inputs to the system. The **Bus** and **Done** signals are the only outputs from the system.

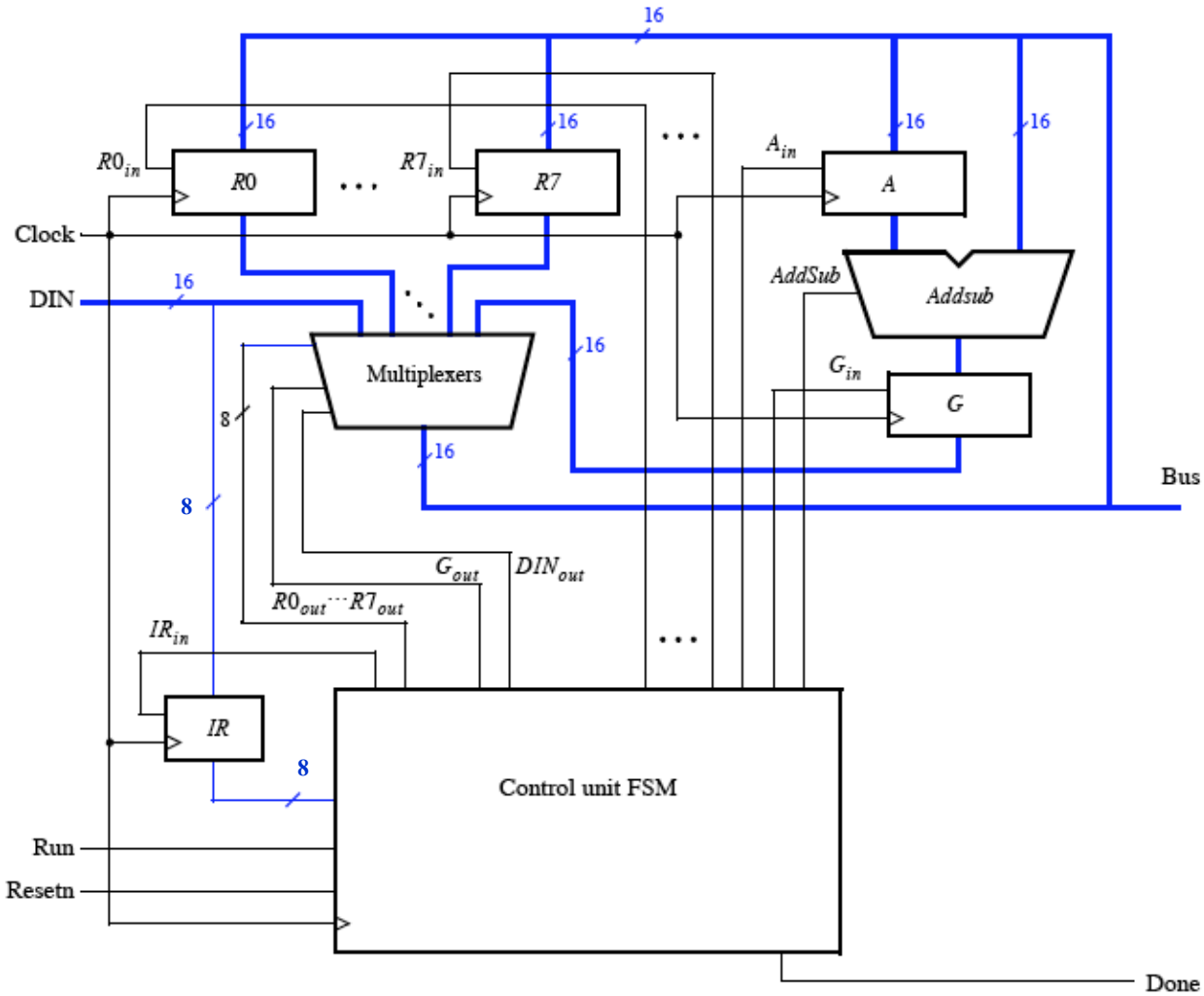

Figure 1: Simple Processor Diagram

This machine operates by reading a value into the instruction register, and on a high "Run" signal it will begin processing. Table 1 shows the 4 instructions you must implement. **mv** will move the data from **RY** into **RX**. **mvi** will move information from **DIN** to **RX**. **add** will add **RX** and **RY** and store the result in **RX**. **sub** will subtract **RY** from **RX** and store the result in **RX**.

Table 1: Required Processor Instructions

Operation	Function performed
mv Rx,Ry	$Rx \leftarrow [Ry]$
mvi Rx,#D	$Rx \leftarrow D$
add Rx, Ry	$Rx \leftarrow [Rx] + [Ry]$
sub Rx, Ry	$Rx \leftarrow [Rx] - [Ry]$

Your instructions will be encoded as **IIXXXYYY**, where **II** represents the instruction, **XXX** represents the X register, and **YYY** represents the Y register. For the **mvi** instruction, **YYY** is still required but has no meaning. When completing **mvi**, the value to be stored in **RX** must be on the **DIN** pins by the next active clock edge after storing the instruction in the **IR** register.

The multiplexer decides what data is chosen to be on the bus. These bus lines are then used to transfer data to each of the registers, the add/sub unit, and the output. The control unit sets all signals to latch the registers (including A, G, and IR), select the appropriate output from the multiplexer, and set appropriate done and AddSub signals. For example, if the control unit asserts the signals $\mathbf{R0}_{\mathbf{out}}$ and $\mathbf{A}_{\mathbf{in}}$, the multiplexer will select **R0** as the output to the bus lines and this data will be saved in **A** on the next active clock edge. **DIN** is the *default* output selected by the multiplexer when waiting, but the result of the instruction will be visible on the output when **Done** is asserted.

Not all instructions require the same amount of time to complete: both **add** and **sub** must move data into the **A** register before completing the appropriate operation. The control unit must appropriately step through each instruction, asserting control signals as needed in successive clock cycles until the instruction is completed. The processor begins executing the instruction on DIN when **Run** is asserted and asserts **Done** when the instruction is complete. Table 2 shows the signals asserted in successive time steps for each instruction. Time T_0 is not shown, as *only* signal $\mathbf{IR}_{\mathbf{in}}$ is asserted. You must design a **Mealy** state machine to control these internal signals.

Table 2: Control signals asserted in each instruction/time step

	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*

You will complete this processor and write a testbench to prove operation for 16 bits. You will then map this processor to the board by using a **generic map** to change the processor components' data width to 8 bits. You may select how you choose to map all inputs and outputs and verify functionality on the board.

Rubric	
Report - Proper format - All sections included - Valid images where applicable - Proper grammar, punctuation, and spelling	**50%** - 10% - 30% - 5% - 5%
Demo - Live Demonstration o Includes working code and answering questions from the TA - Proper modular design o Thoughtful I/O and Signal names included o Uses VHDL Generics - Comments o Thoughtful comments, not English translations of code	**40%** - 30% - 5% - 5%
Proper Assign Server Code Submission	**10%**