

ECEN 4243:Computer Architecture

Instructor: Rose Thompson

Lab 0: Revisiting Digital Logic and SystemVerilog Simulation

Group:

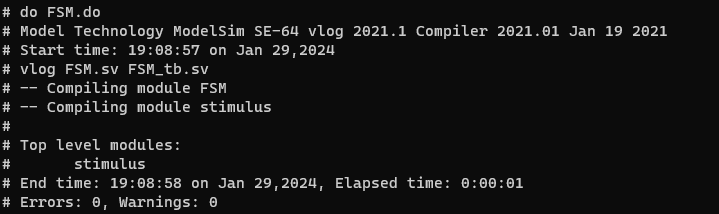
Garrett Page

Hridi Prova Debnath

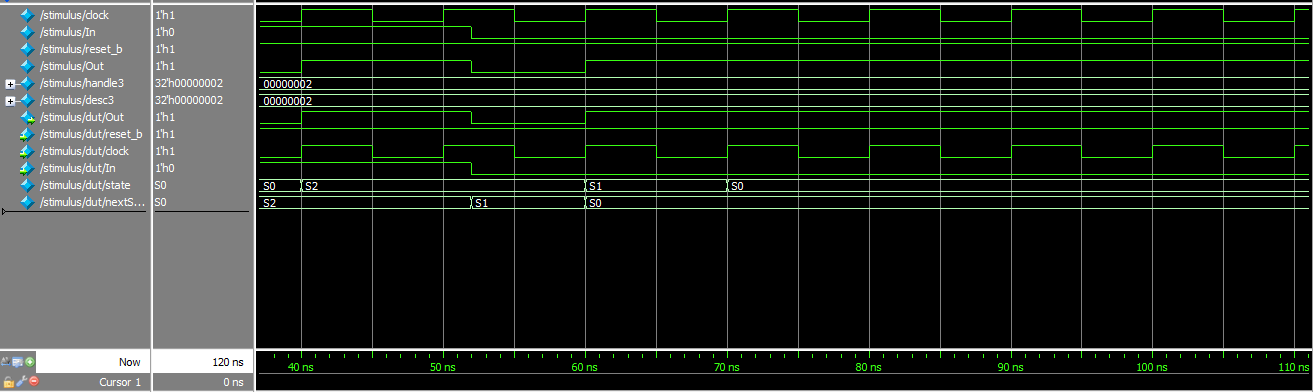
This lab0 aims to introduce the use of HDL (Hardware Description Language) through Siemens ModelSim/Questasim. The focus will be on creating a Finite State Machine (FSM) and a register file(RF).

Finite State Machine (FSM):

Here, we use a Mealy Finite State Machine (FSM). This type of FSM is characterized by its output values being determined by both its current state and the current input signals. For this task, we will retrieve three files from the repository: FSM.sv (the FSM module), FSM\_tb.sv (the testbench for the FSM), and FSM.do (a script for simulation). These files are compiled in the terminal using the command *“****vsim -do FSM.do -c”.***



For getting waveform, we use ***“vsim -do FSM.do***” command.

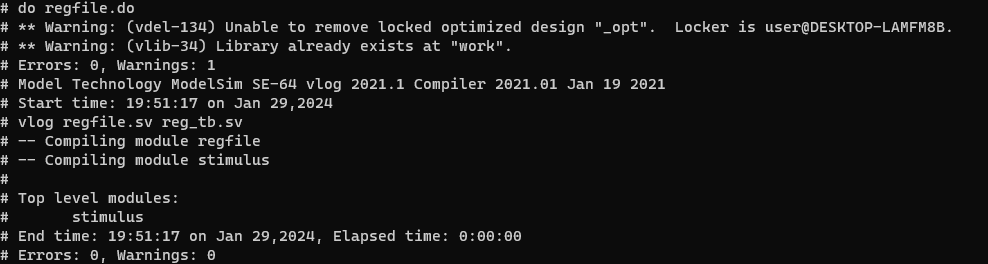


The FSM has three states (S0, S1, S2) and transitions between them based on the input 'In'. The output 'Out' depends on both the current state and the input:

1. In state S0, the FSM goes to S2 if 'In' is 1 (output 'Out' is 0), and stays in S0 if 'In' is 0 (output 'Out' is 1).
2. In state S1, it always transitions to S0, with the output 'Out' set to 1.
3. In state S2, it remains there if 'In' is 1 (output 'Out' is 1), and goes to S1 if 'In' is 0 (output 'Out' is 0).
4. If in an undefined state, it defaults to S0 with an undefined output ('Out' is '1'bx).

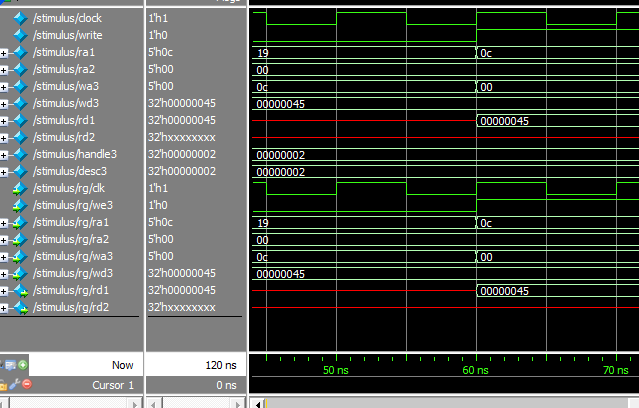
Register file:

A register file is a collection of registers that can be read and written by a digital circuit. Registers are used to store operands for ALU operations.For this task, we will retrieve three files from the repository: FSM.sv (the FSM module), FSM\_tb.sv (the testbench for the FSM), and FSM.do (a script for simulation). These files are compiled in the terminal using the command “***vsim -do regfile.do -c”***.



This register file has two read ports and one write port for a processor. It features inputs for clock, write enable, read and write addresses, and write data, and outputs for read data. The module includes a 32x32-bit register array, supports clock-triggered write operations (excluding register 0), and provides immediate output updates for read operations.

For getting waveform, we use ***‘vsim -do regfile.do’*** command.



Initial setup involves setting read and write addresses without initiating a write. The write operation begins at 20 time units and then stops. A read from a written register occurs at 40 time units. An attempt to write to the protected register 0 is made and ceases after 20 time units.

**Self-checking testbench**

A self-checking testbench was added to the regfile to ensure operability.

The test vector file used is **stimulus.tv** which includes comments explaining what each auto-test does.

The test vector file includes annotations that must be removed by running **compile\_stimulus.py**, which creates a valid tv file called *reg.tv* that is actually used by the testbench.

Run **vsim -do reg.do -c** to use the self-checking testbench. (uses *reg\_tb\_auto.sv*)

Run **vsim -do regfile.do** to use the manual waveform version. (uses *reg\_tb.sv*)

Reference:

1] S. Harris and D. Harris, Digital Design and Computer Architecture, RISC-V Edition. Elsevier Science,