ECE333 Computer Architecture Lab, Fall 2011
Assignment 5: Verilog

1. Introduction

Using Verilog, build a 32-bit **single-cycle** processor that implements the following instructions.

Type	Instruction
Arithmetic (unsigned)	add
Logical	and, nand, or, xor, sgt
Shift	sll, srl

Format of the Instruction Set is shown below. Each instruction will be 18 bits.

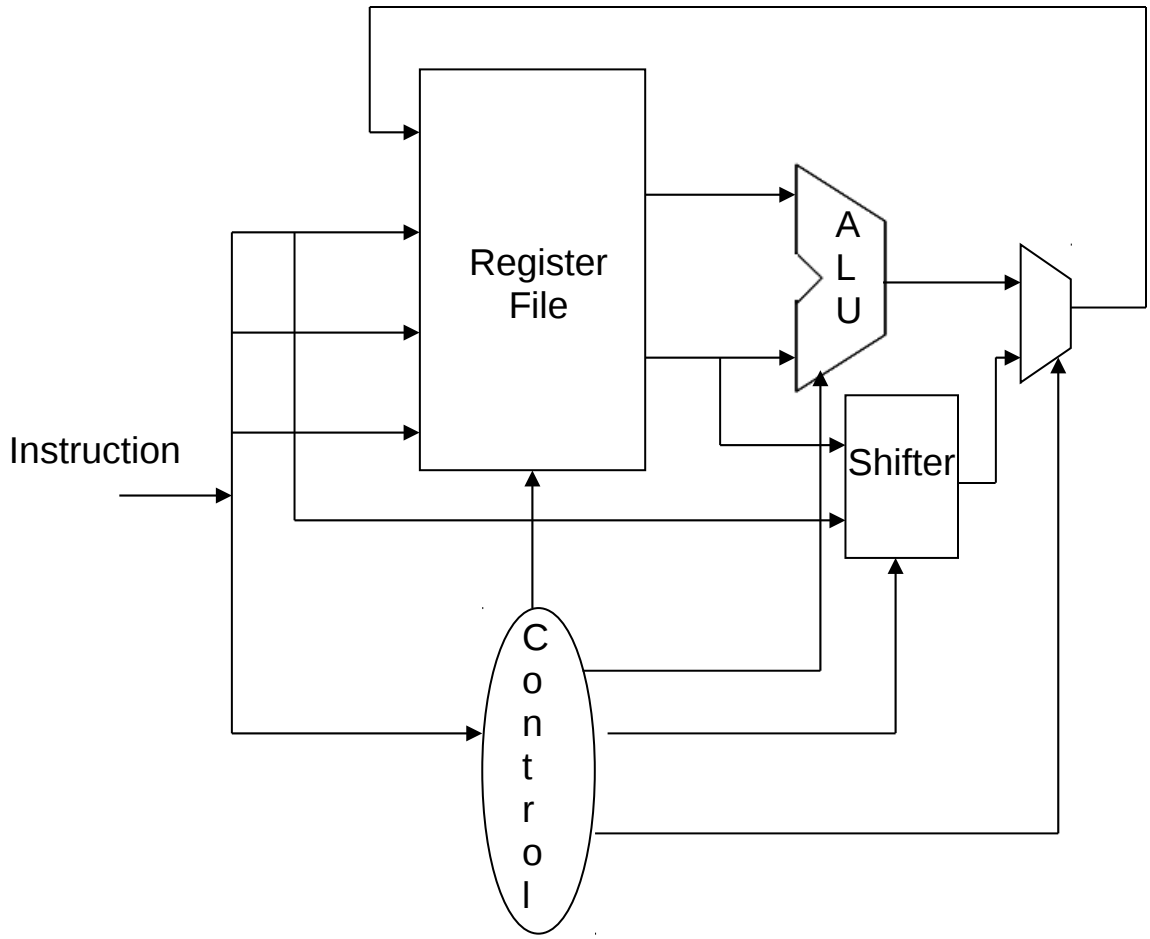
Opcode	Destination Register	Source Register1	Source Register2 / Shift amount
3 bits	**5 bits**	**5 bits**	**5 bits**

Opcode	Instruction Type
000	ADD
001	AND
010	NAND
011	OR
100	XOR
101	SGT (set as 1 if grater then)
110	SLL – Shift Left Logical
111	SRL – Shift Right Logical

The processor is not pipelined. It is single-cycle, which includes instruction decode, execution & writeback. The register file must have 32, 32-bit registers, and the 32 registers would be initialized as 0.

You MUST have the following separate Verilog modules for the processor: **Control Unit, Register File, ALU, Shifter and Multiplexer.** You can describe the modules either behaviorally or structurally. In addition you need to write another separate module **Wiring** to wire the above modules, and connect the processor with the testbench module.

The datapath should be somewhat like the figure below. This is just to give you an idea.



2. Instructions

Step 1. Get Your Assignment

Please visit your Sakai site to download the `verilog-lab-handout.tar` file. Save the .tar file to the directory which you plan to do your work. Then give the command: `tar –xvf verilog-lab-handout.tar`. This will create a directory called `./verilog-lab-handout` with the following files:

- `testbench.v`: Verilog test bench program for testing you processor implementation.
- `wiring.v`: Template for the Wiring module.
- `regfile.v`: Template for the Registers File module.
- `control.v`: Template for the Control Unit module.
- `alu.v`: Template for the ALU module.
- `mux.v`: Template for the Multiplexer module.
- `shifter.v`: Template for the Logical Shifter module.

Step 2. Implementation of the single-cycle processor

Your job is to complete coding the above Verilog module files (except the `testbech.v`, which is already implemented).

Step 3. Debugging and testing

In this lab, we will use Synopsys Verilog Compiler Simulator (VCS) tool to compile and simulate the designs, and use Discovery Visualization Environment (DVE) tool to visualize and view the generated waveforms of the simulation.

There are three main steps in debugging the design:

- Compiling the Verilog source codes using VCS
- Running the simulation
- Viewing and debugging the generated waveforms using DVE

Instructions for performing the above three steps (on ECE lab CentOS) are as below:

First, under the directory where your Verilog .v source codes are located, input the following two commands to set your environment for using VCS and DVE

```
cp   ~vlsiuser/.cshrc .
source  .cshrc
```

Second, input the following command to compile your source code files

```
vcs +v2k -full64 –PP testbench.v wirng.v regfile.v
control.v alu.v mux.v shifter.v
```

Successful execution of the above command would generate the executable simulation program `simv`.

Third, run the simulation with the command below

```
./simv
```

Successful running of the simulation would produce the dump file `cpu.dump`, which could be visualized later using DVE.

Fourth, use DVE to view the generated waveforms

```
dve –full64 –vpd cpu.dump
```

3. Handin

This is an individual project. All handins are electronic.
Specific instructions for submitting your work would be posted on Sakai.