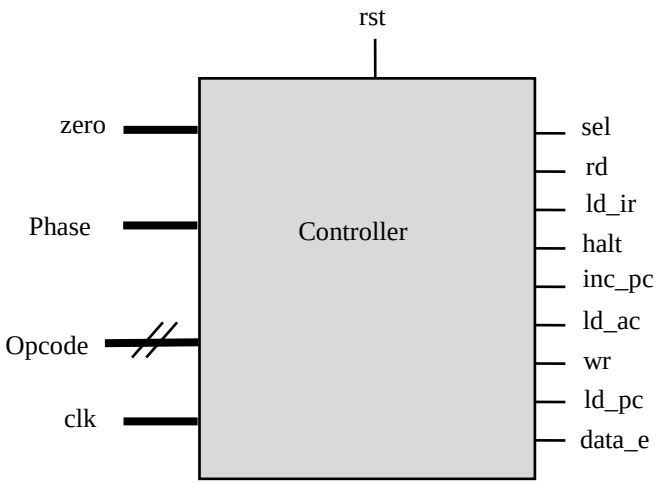
Module 6: Making Procedural Statements

Lab 6-1 Modeling a Controller

Objective: To use the Verilog case statement to describe a controller.

The controller generates all control signals for the VeriRISC CPU. The operation code, fetch-and-execute phase, and whether the accumulator is zero determine the control signal levels.



Specifications

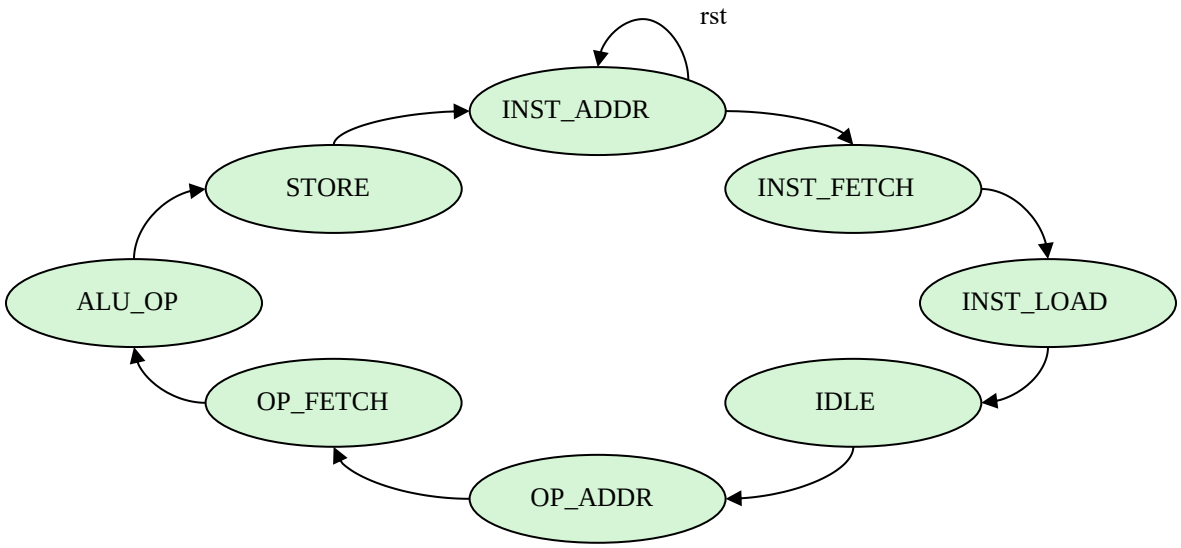
- The controller is clocked on the rising edge of `clk`.
- `rst` is synchronous and active high.
- `zero` is an **input** which is 1 when the CPU accumulator is zero and 0 otherwise.
- `opcode` is a 3-bit **input** for CPU operation, as shown in the following table.

Opcode/ Instruction	Opcode Encoding	Operation	Output
HLT	000	PASS A	in_a => alu_out
SKZ	001	PASS A	in_a => alu_out
ADD	010	ADD	in_a + in_b => alu_out
AND	011	AND	in_a & in_b => alu_out
XOR	100	XOR	in_a ^ in_b => alu_out
LDA	101	PASS B	in_b => alu_out
STO	110	PASS A	in_a => alu_out
JMP	111	PASS A	in_a => alu_out

- There are 7 single-bit **outputs**, as shown in this table.

Output	Function
`sel`	select
`rd`	memory read
`ld_ir`	load instruction register
`halt`	halt
`inc_pc`	increment program counter
`ld_ac`	load accumulator
`ld_pc`	load program counter
`wr`	memory write
`data_e`	data enable

- The controller has a single-bit phase input with a total of 8 phases processed. Phase transitions are unconditional, i.e., the controller passes through the same 8-phase sequence, from `INST_ADDR` to `STORE`, every 8 `clk` cycles. The reset state is `INST_ADDR`.



- The controller outputs will be decoded w.r.t phase and opcode, as shown in this table.

Outputs	**Phase**						**Notes**		
	INST_ ADDR	INST_ FETCH	INST_ LOAD	IDLE	OP_ ADDR	OP_ FETCH	ALU_ OP	STORE	
`sel`	1	1	1	1	0	0	0	0	`ALU_OP = 1` if opcode is `ADD`, `AND`, `XOR` or `LDA`
`rd`	0	1	1	1	0	`ALUOP`	`ALUOP`	`ALUOP`	
`ld_ir`	0	0	1	1	0	0	0	0	
`halt`	0	0	0	0	`HALT`	0	0	0	
`inc_pc`	0	0	0	0	1	0	`SKZ && zero`	0	
`ld_ac`	0	0	0	0	0	0	0	`ALUOP`	
`ld_pc`	0	0	0	0	0	0	`JMP`	`JMP`	
`wr`	0	0	0	0	0	0	0	`STO`	
`data_e`	0	0	0	0	0	0	`STO`	`STO`	

Designing a Controller

1. Change to the *lab7-ctlr* directory and examine the following files.

controller_test.v	Controller test
filelist.txt	File listing all modules to simulate

2. Use your favorite editor to create the *controller.v* file and describe the controller module named "*control* ".

3. Declare a reg for each of these intermediate terms and establish their value immediately before entering the case statement.

Verifying the Controller Design

1. Using the provided test module, check your controller design using the following command with Xcelium™.

```
xrun controller.v controller_test.v (Batch Mode)
```

or

```
xrun controller.v controller_test.v -gui -access +rwc ( GUI Mode)
```

2. You might find it easier to list all the files and simulation options in a text file and pass the file into the simulator using the `-f xrun` option.

```
xrun -f filelist.txt -access rwc
```

You should see the following results.

```
Testing opcode HLT phase 0 1 2 3 4 5 6 7
Testing opcode SKZ phase 0 1 2 3 4 5 6 7
Testing opcode ADD phase 0 1 2 3 4 5 6 7
Testing opcode AND phase 0 1 2 3 4 5 6 7
Testing opcode XOR phase 0 1 2 3 4 5 6 7
Testing opcode LDA phase 0 1 2 3 4 5 6 7
Testing opcode STO phase 0 1 2 3 4 5 6 7
Testing opcode JMP phase 0 1 2 3 4 5 6 7
TEST PASSED
```

3. Correct your controller design as needed.