

## LAB 2

### ALU and Register File

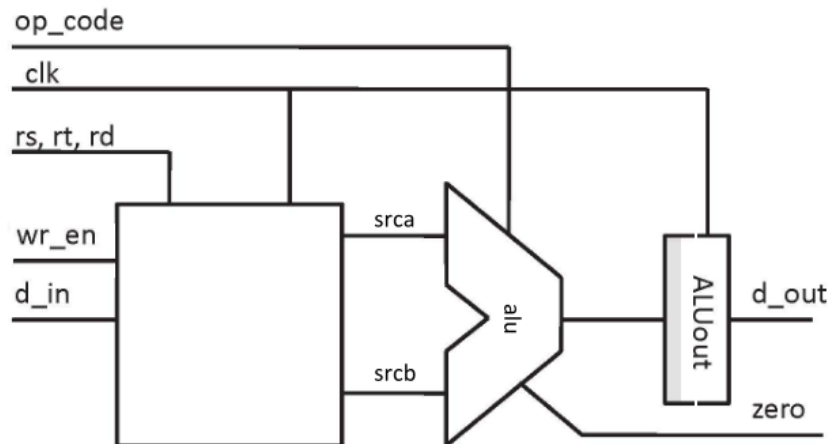


Figure 2.1 Block diagram of ALU and RF

**OBJECTIVE:** Model part of a CPU datapath using structural Verilog. These functional blocks combined will be capable of executing ALU operations on data contained in the register file

Your main task in this assignment is to interconnect the functional blocks to create the system depicted above. The Figure 2.1 shows the top level i.e. system level view so its code must be entered into the **design.sv** window on EDAplayground for it to be the top level module.

The **alu** shown was created in lab1. Copy the **alu** source code from lab 1 into a file named **alu.v**

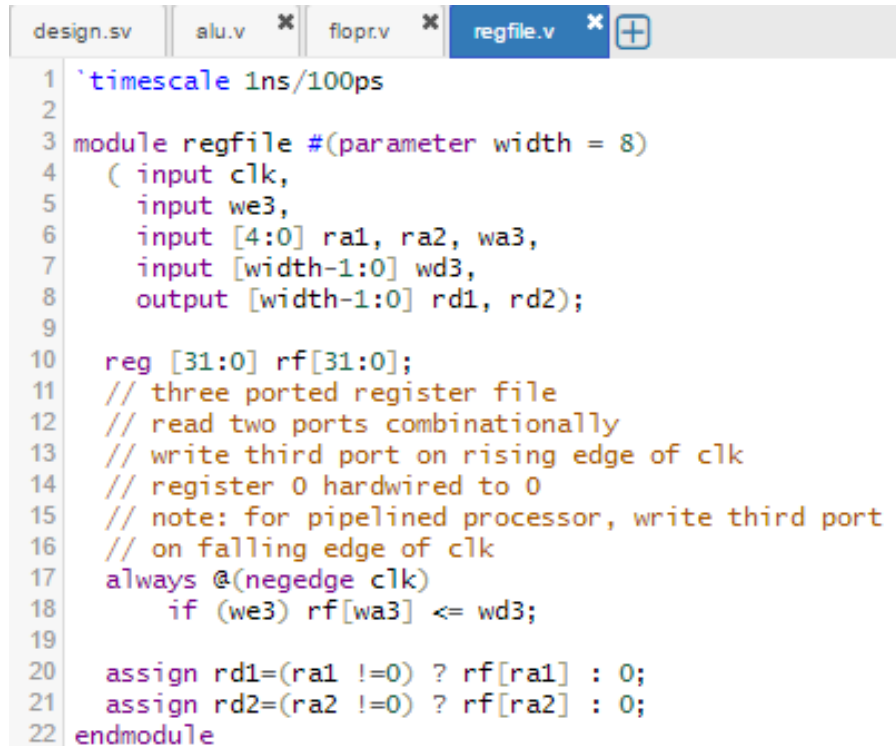
Source codes for the **ALUout** register (flop.v) and **register file** are given in Figures 2.1 and 2.3..

```

design.sv  alu.v  flopr.v  regfile.v  +
1  `timescale 1ns/100ps
2
3  //resettable flip flop with load enable
4  module flopr #(parameter width = 8)
5  ( input clk, reset, load,
6    input [width-1:0] d,
7    output reg [width-1:0] q);
8
9  always @(posedge clk, posedge reset)
10 begin
11     if(reset) q = 0;
12     else if(load) q = d;
13 end
14 endmodule
15

```

Figure 2.2 Verilog code for ALUout register (flop.v)



```

1 `timescale 1ns/100ps
2
3 module regfile #(parameter width = 8)
4   ( input clk,
5     input we3,
6     input [4:0] ra1, ra2, wa3,
7     input [width-1:0] wd3,
8     output [width-1:0] rd1, rd2);
9
10  reg [31:0] rf[31:0];
11  // three ported register file
12  // read two ports combinatorially
13  // write third port on rising edge of clk
14  // register 0 hardwired to 0
15  // note: for pipelined processor, write third port
16  // on falling edge of clk
17  always @(negedge clk)
18    if (we3) rf[wa3] <= wd3;
19
20  assign rd1=(ra1 !=0) ? rf[ra1] : 0;
21  assign rd2=(ra2 !=0) ? rf[ra2] : 0;
22 endmodule

```

Figure 2.3 Verilog code for register file (regfile.v)

Skeleton code for the top level module named **design.sv** is available on *BeachBoard* in the **Content** section, under **Lab Files**, in the **Lab 2** subsection. You will need to instantiate **regfile** and the **alu** according to Figure 2.1 (that is: fill in the port list for the those two modules).

Also the testbench.sv skeleton code is given in the same *BeachBoard* folder as design.sv. In the test bench between lines 42 and 88 there are assignment statements for test cases that must have values filled in on the right side. Use the **Table 1** to help complete the assignment statements:

**Table 1:**

Test Case	Operation	op_code	rs	rt
1	add \$ALUout, \$3, \$4			
2	inc \$ALUout, \$3			
3	and \$ALUout, \$1, \$2			
4	or \$ALUout, \$1, \$2			
5	xor \$ALUout, \$2, \$1			
6	not \$ALUout, \$2			
7	sl \$ALUout, \$4			
8	nop			

Each operation is an instruction where the **destination register** is *always* **\$ALUout** and **does not need to be specified**, **rs** is the number of the **first source register**, **rt** is the number of the *optional second source register* (for operations that take 2 source register operands).

## WHAT TO SUBMIT

Once you have verified proper functionality of your project, copy the contents of the **design.sv** to a text file named **lab2.txt** and upload it to the BeachBoard Dropbox for Lab 2. Additionally, upload your Lab report (see the LabReportTemplate in the Documents folder on BeachBoard), include populated **Table 1** in your lab report.

**NOTE:** keep these lab files as they will be needed for future labs!

Created by Josh Hayter, updated by Jelena Trajkovic