LAB 5
ALU and Register File

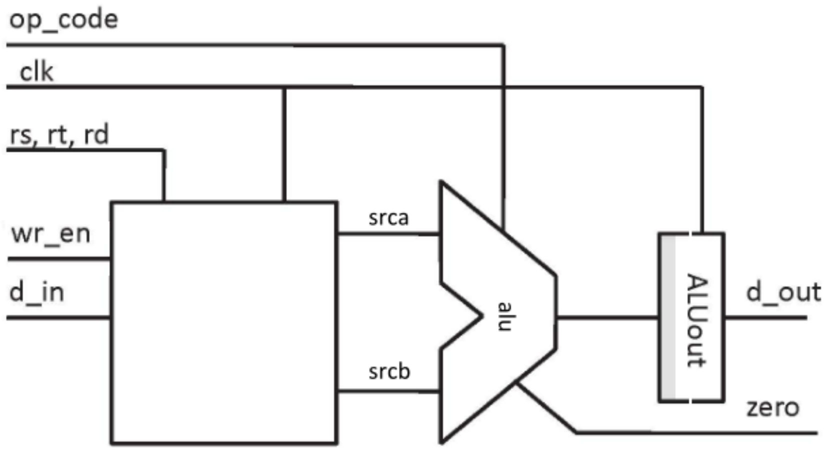


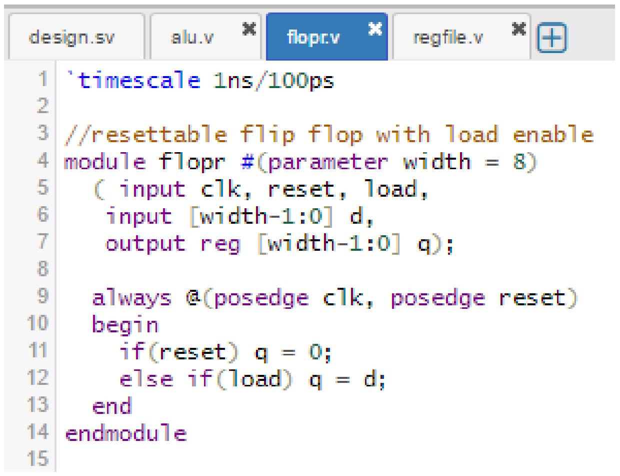
Figure 2.1 Block diagram of ALU and RF

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 lab4. <u>Copy the **alu** source code from lab 4 into a file named **alu.v**</u>

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

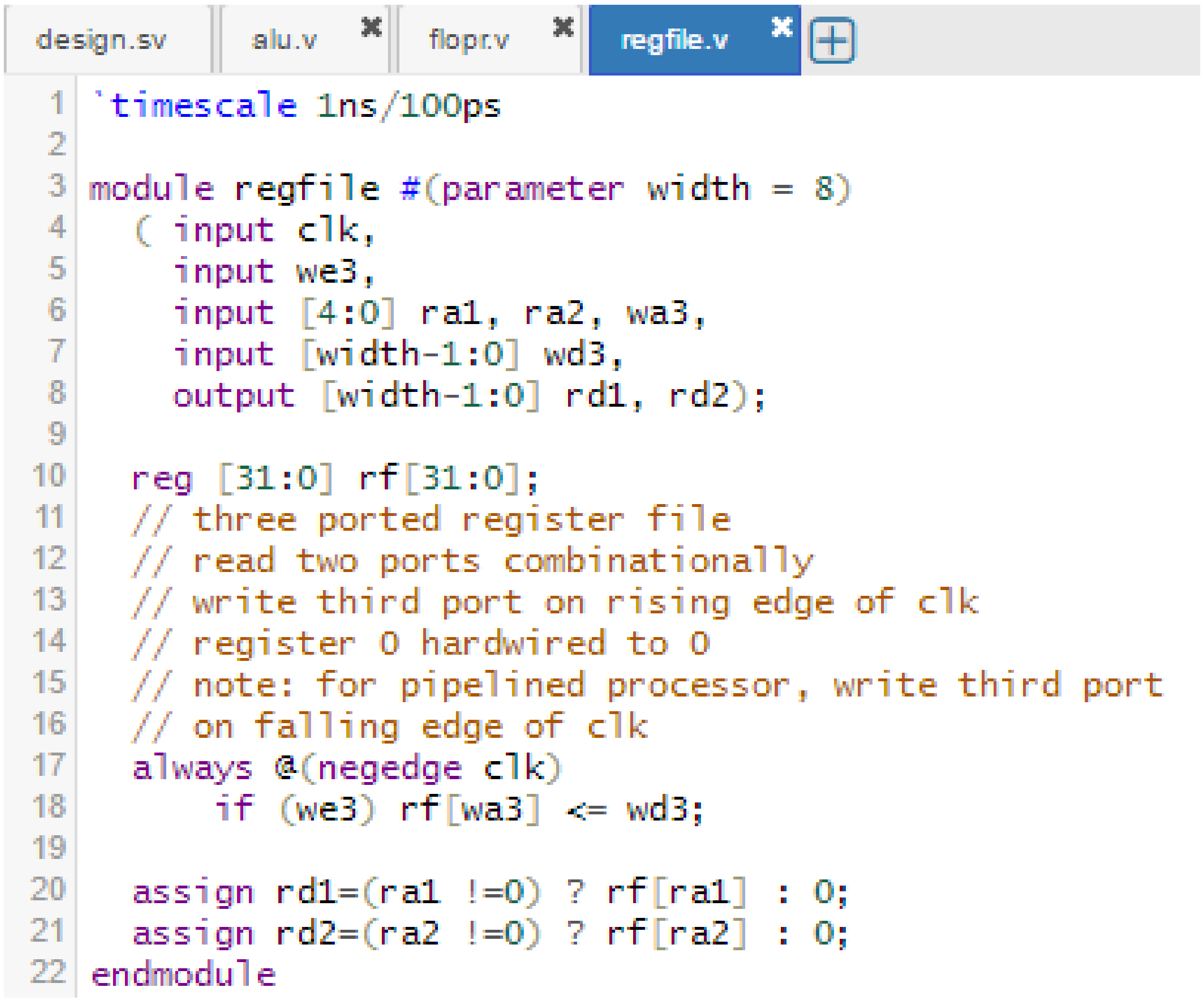


```
`timescale 1ns/100ps

//resettable flip flop with load enable
module flopr #(parameter width = 8)
  ( input clk, reset, load,
   input [width-1:0] d,
   output reg [width-1:0] q);

  always @(posedge clk, posedge reset)
  begin
    if(reset) q = 0;
    else if(load) q = d;
  end
endmodule
```

Figure 2.2 Verilog code foe ALUout refister (flopr.v)



```
`timescale 1ns/100ps

module regfile #(parameter width = 8)
  ( input clk,
    input we3,
    input [4:0] ra1, ra2, wa3,
    input [width-1:0] wd3,
    output [width-1:0] rd1, rd2);

  reg [31:0] rf[31:0];
  // three ported register file
  // read two ports combinationally
  // write third port on rising edge of clk
  // register 0 hardwired to 0
  // note: for pipelined processor, write third port
  // on falling edge of clk
  always @(negedge clk)
      if (we3) rf[wa3] <= wd3;

  assign rd1=(ra1 !=0) ? rf[ra1] : 0;
  assign rd2=(ra2 !=0) ? rf[ra2] : 0;
endmodule
```

Figure 2.3 Verilog code foe 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 5** subsection. You will need to instantiate **regfile** and the **alu** according to Figure 2.1 (that is: fill in the port list for 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	0	3	4
2	inc $ALUout, $3	1	3	1
3	and $ALUout, $1, $2	2	1	2
4	or $ALUout, $1, $2	3	1	2
5	xor $ALUout, $2, $1	4	2	1
6	not $ALUout, $2	5	2	null
7	sl $ALUout, $4	6	[illegible]4 << 1	null
8	nop	7	null	null

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 **lab5.txt** and upload it to the BeachBoard Dropbox for Lab 5. Additionally, upload your Lab report (see the LabReportTemplete in the Documents folder on BeachBoard), include populated **Table 1** in your lab repot.

NOTE:

- Comment your code
- Keep these lab files as they will be needed for future labs!

Created by Josh Hayter, updated by Gayathri Venna and Jelena Trajkovic