Digital Design and Computer Organization Laboratory

UE21CS251A

3rd Semester, Academic Year 2022-23

Date: 29/11/2022

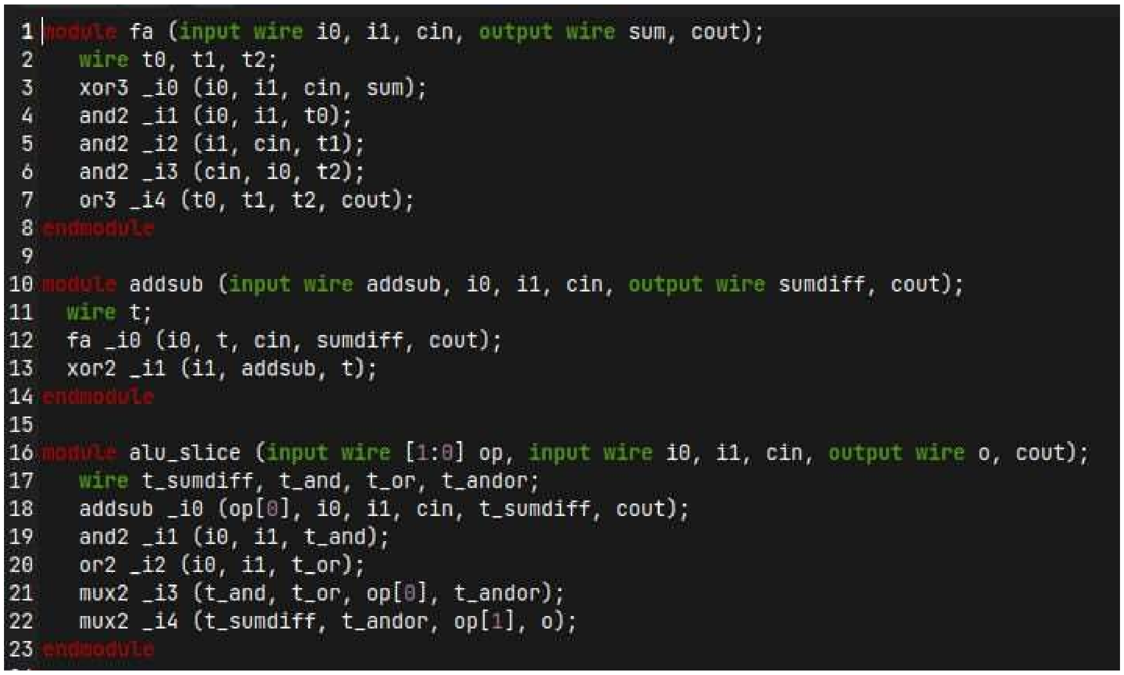
Name : Sujal.S	SRN : PES2UG21CS548	Section : I

Title of the Program:

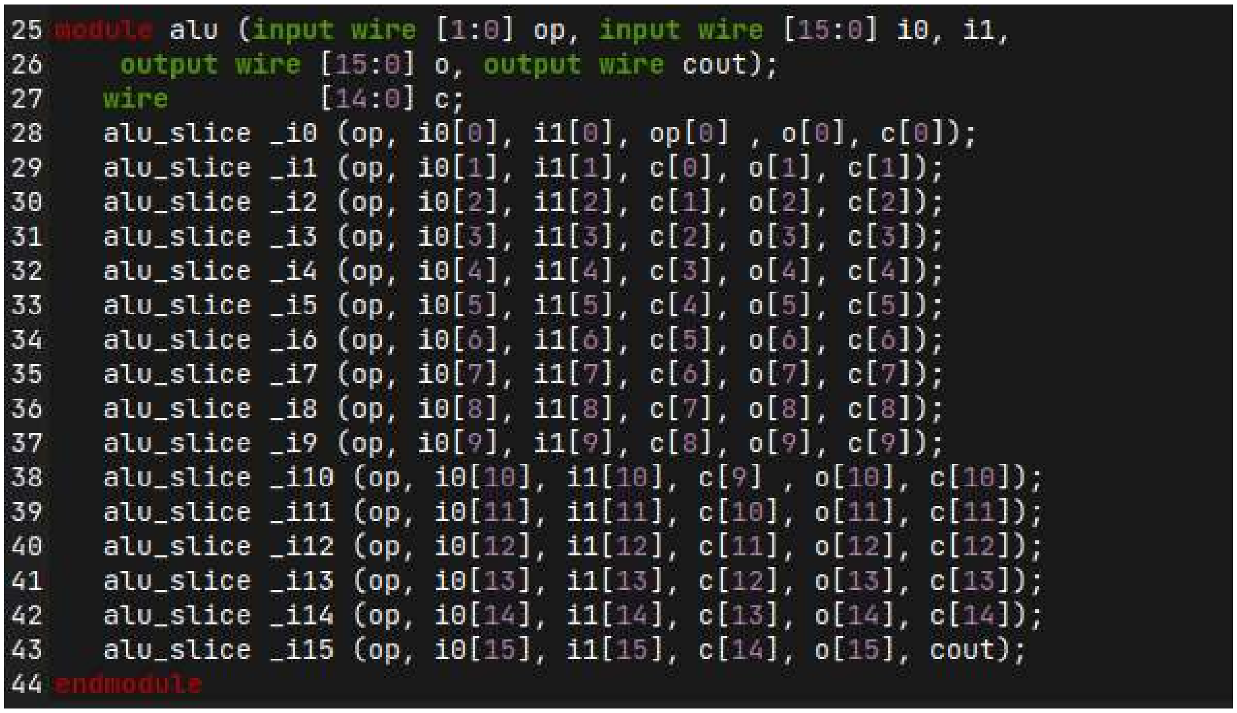
Microprocessor Control Logic-2

Aim of the Program:

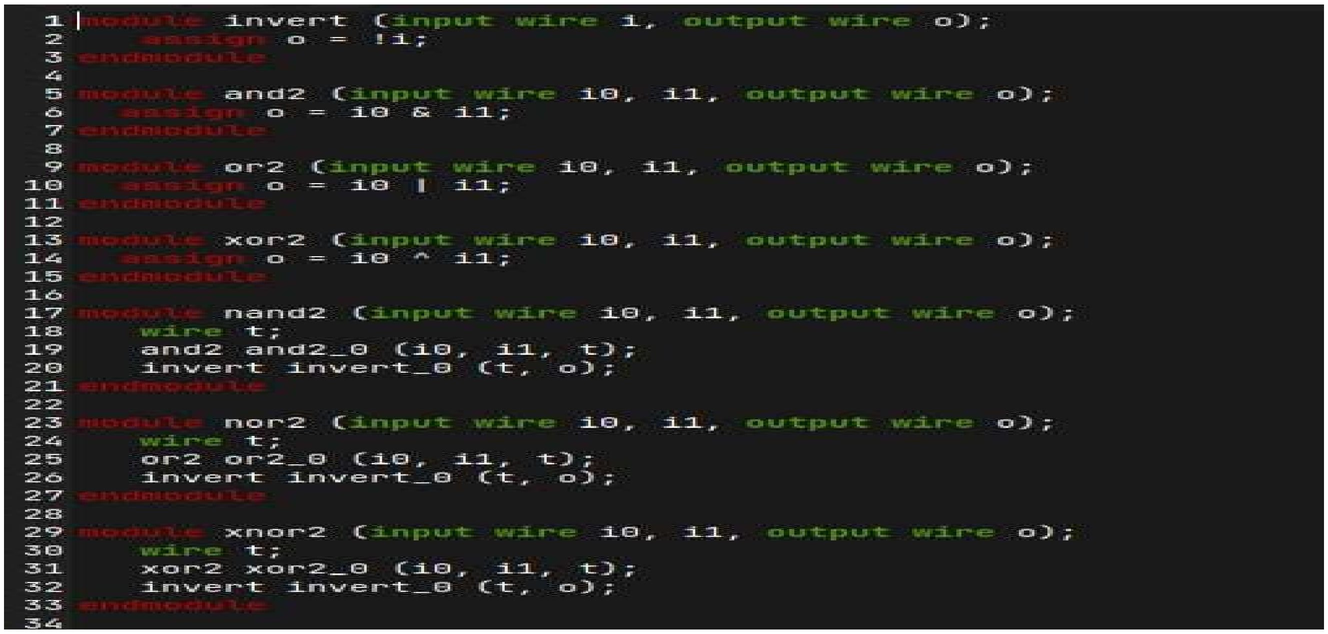
The task in this assignment the intent of this assignment is to enhance the control logic to implement a load and a jump instruction



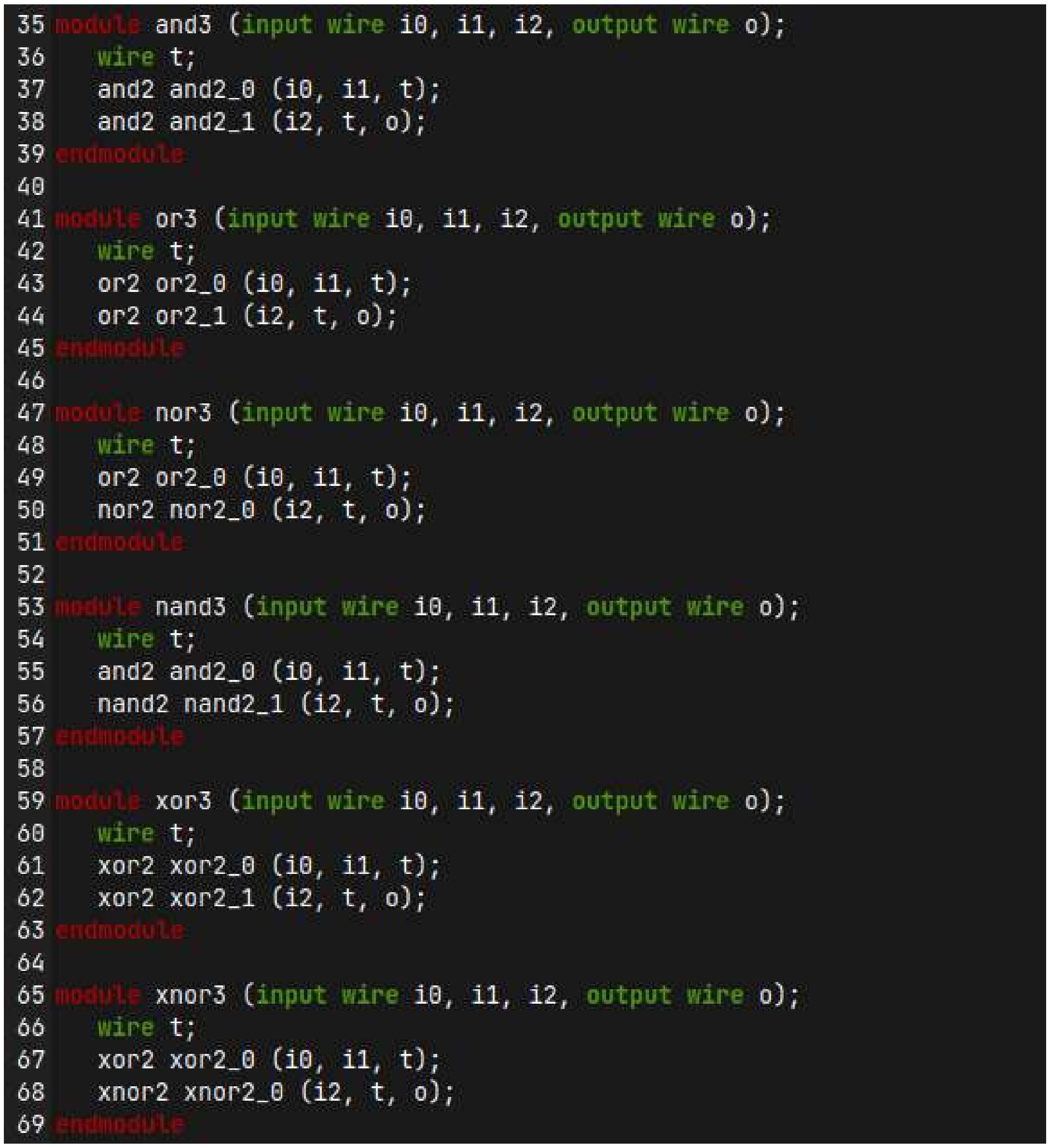
```
module fa (input wire i0, i1, cin, output wire sum, cout);
   wire t0, t1, t2;
   xor3 _i0 (i0, i1, cin, sum);
   and2 _i1 (i0, i1, t0);
   and2 _i2 (i1, cin, t1);
   and2 _i3 (cin, i0, t2);
   or3 _i4 (t0, t1, t2, cout);
endmodule

module addsub (input wire addsub, i0, i1, cin, output wire sumdiff, cout);
  wire t;
  fa _i0 (i0, t, cin, sumdiff, cout);
  xor2 _i1 (i1, addsub, t);
endmodule

module alu_slice (input wire [1:0] op, input wire i0, i1, cin, output wire o, cout);
   wire t_sumdiff, t_and, t_or, t_andor;
   addsub _i0 (op[0], i0, i1, cin, t_sumdiff, cout);
   and2 _i1 (i0, i1, t_and);
   or2 _i2 (i0, i1, t_or);
   mux2 _i3 (t_and, t_or, op[0], t_andor);
   mux2 _i4 (t_sumdiff, t_andor, op[1], o);
endmodule
```



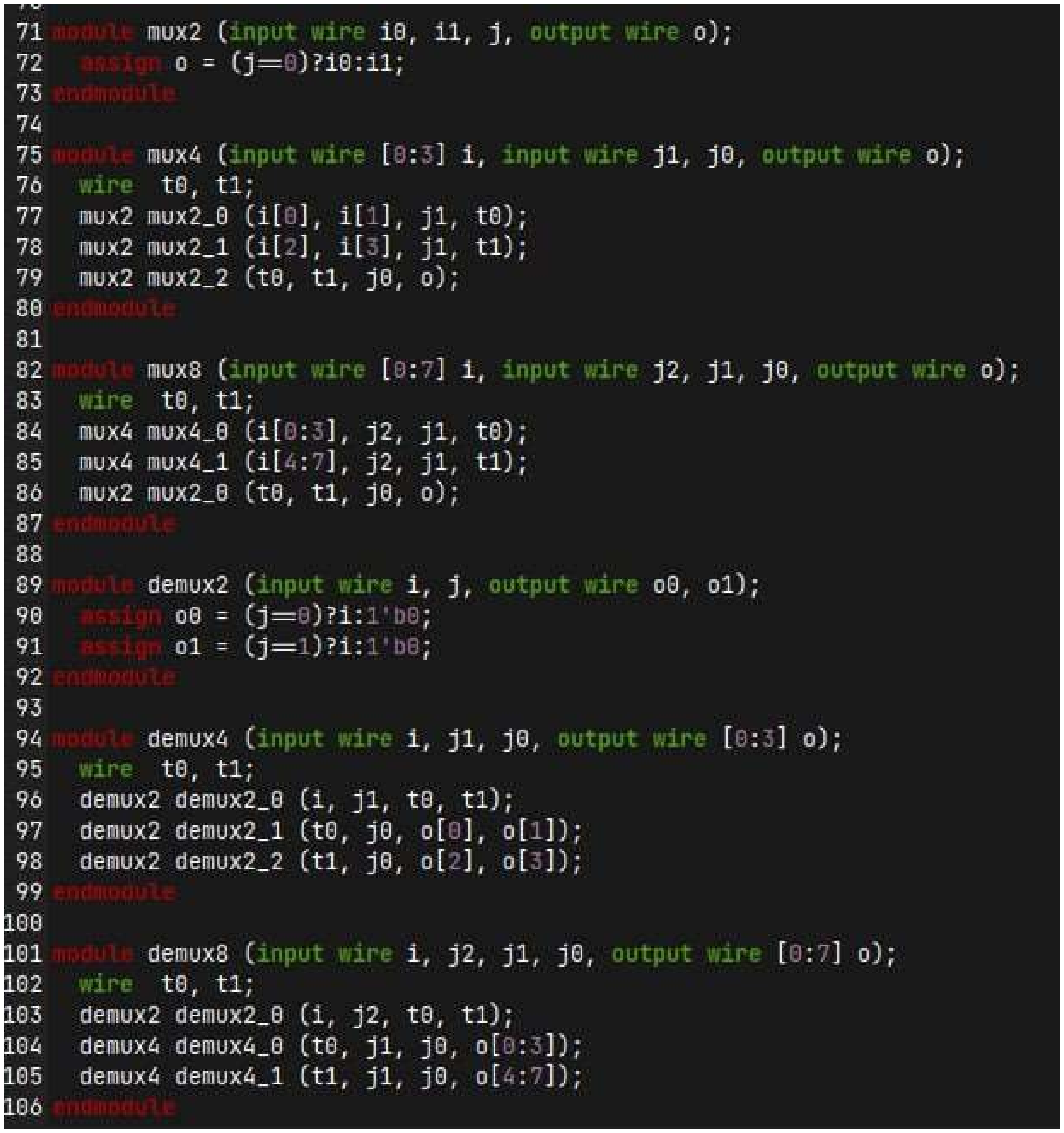
```verilog
module alu (input wire [1:0] op, input wire [15:0] i0, i1,
    output wire [15:0] o, output wire cout);
   wire          [14:0] c;
   alu_slice _i0 (op, i0[0], i1[0], op[0] , o[0], c[0]);
   alu_slice _i1 (op, i0[1], i1[1], c[0], o[1], c[1]);
   alu_slice _i2 (op, i0[2], i1[2], c[1], o[2], c[2]);
   alu_slice _i3 (op, i0[3], i1[3], c[2], o[3], c[3]);
   alu_slice _i4 (op, i0[4], i1[4], c[3], o[4], c[4]);
   alu_slice _i5 (op, i0[5], i1[5], c[4], o[5], c[5]);
   alu_slice _i6 (op, i0[6], i1[6], c[5], o[6], c[6]);
   alu_slice _i7 (op, i0[7], i1[7], c[6], o[7], c[7]);
   alu_slice _i8 (op, i0[8], i1[8], c[7], o[8], c[8]);
   alu_slice _i9 (op, i0[9], i1[9], c[8], o[9], c[9]);
   alu_slice _i10 (op, i0[10], i1[10], c[9] , o[10], c[10]);
   alu_slice _i11 (op, i0[11], i1[11], c[10], o[11], c[11]);
   alu_slice _i12 (op, i0[12], i1[12], c[11], o[12], c[12]);
   alu_slice _i13 (op, i0[13], i1[13], c[12], o[13], c[13]);
   alu_slice _i14 (op, i0[14], i1[14], c[13], o[14], c[14]);
   alu_slice _i15 (op, i0[15], i1[15], c[14], o[15], cout);
endmodule
```



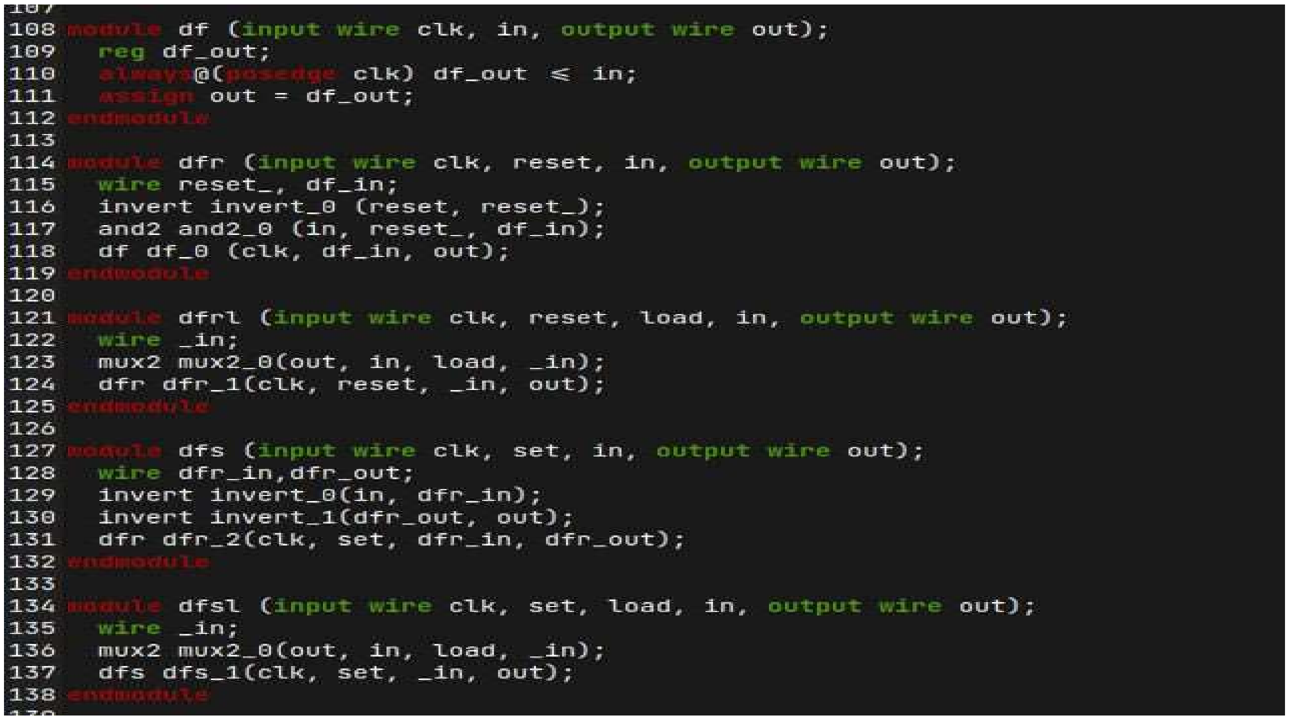
```verilog
module invert (input wire i, output wire o);
   assign o = !i;
endmodule

module and2 (input wire i0, i1, output wire o);
  assign o = i0 & i1;
endmodule

module or2 (input wire i0, i1, output wire o);
  assign o = i0 | i1;
endmodule

module xor2 (input wire i0, i1, output wire o);
  assign o = i0 ^ i1;
endmodule

module nand2 (input wire i0, i1, output wire o);
   wire t;
   and2 and2_0 (i0, i1, t);
   invert invert_0 (t, o);
endmodule

module nor2 (input wire i0, i1, output wire o);
   wire t;
   or2 or2_0 (i0, i1, t);
   invert invert_0 (t, o);
endmodule

module xnor2 (input wire i0, i1, output wire o);
   wire t;
   xor2 xor2_0 (i0, i1, t);
   invert invert_0 (t, o);
endmodule

```



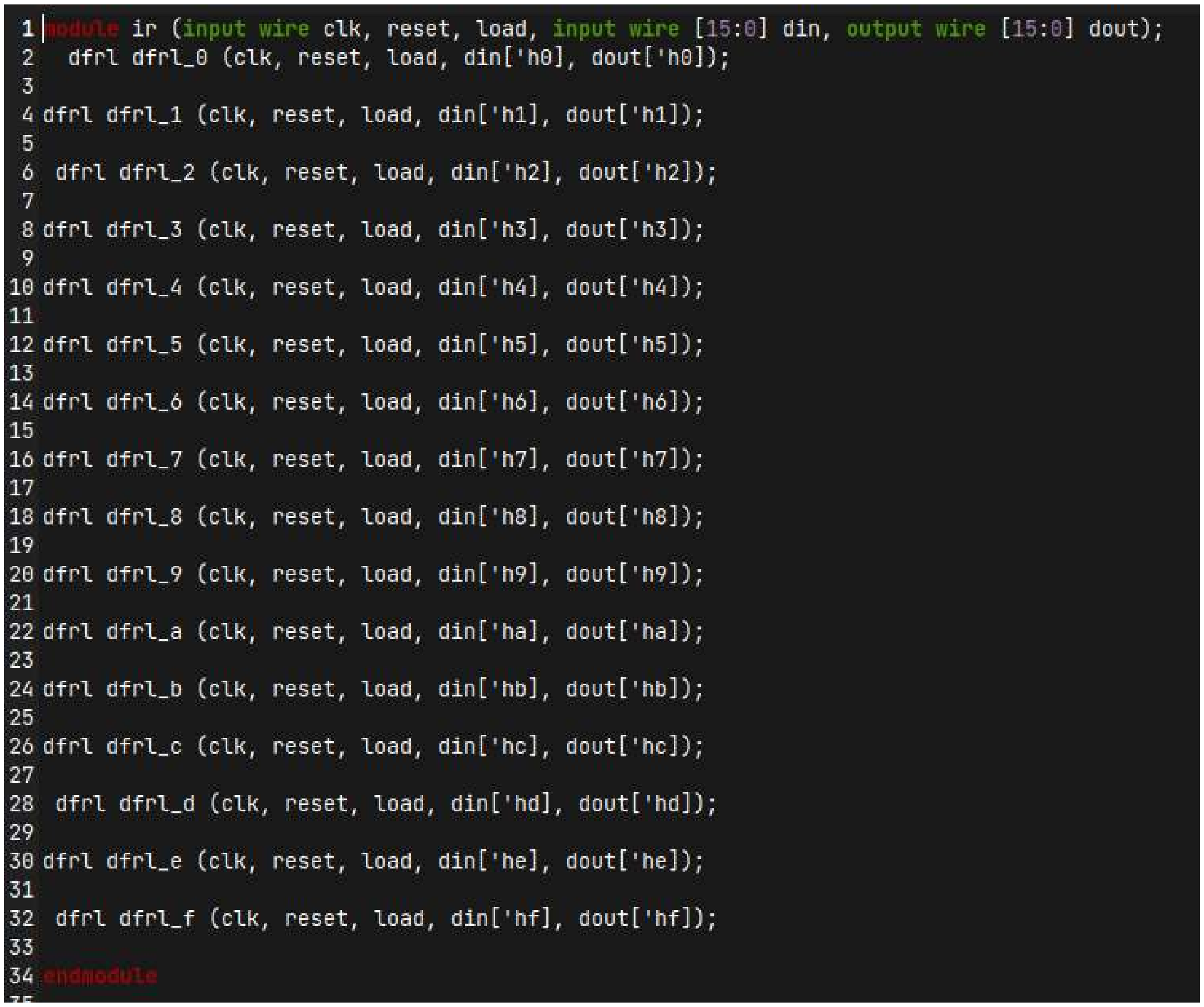
```
module and3 (input wire i0, i1, i2, output wire o);
   wire t;
   and2 and2_0 (i0, i1, t);
   and2 and2_1 (i2, t, o);
endmodule

module or3 (input wire i0, i1, i2, output wire o);
   wire t;
   or2 or2_0 (i0, i1, t);
   or2 or2_1 (i2, t, o);
endmodule

module nor3 (input wire i0, i1, i2, output wire o);
   wire t;
   or2 or2_0 (i0, i1, t);
   nor2 nor2_0 (i2, t, o);
endmodule

module nand3 (input wire i0, i1, i2, output wire o);
   wire t;
   and2 and2_0 (i0, i1, t);
   nand2 nand2_1 (i2, t, o);
endmodule

module xor3 (input wire i0, i1, i2, output wire o);
   wire t;
   xor2 xor2_0 (i0, i1, t);
   xor2 xor2_1 (i2, t, o);
endmodule

module xnor3 (input wire i0, i1, i2, output wire o);
   wire t;
   xor2 xor2_0 (i0, i1, t);
   xnor2 xnor2_0 (i2, t, o);
endmodule
```



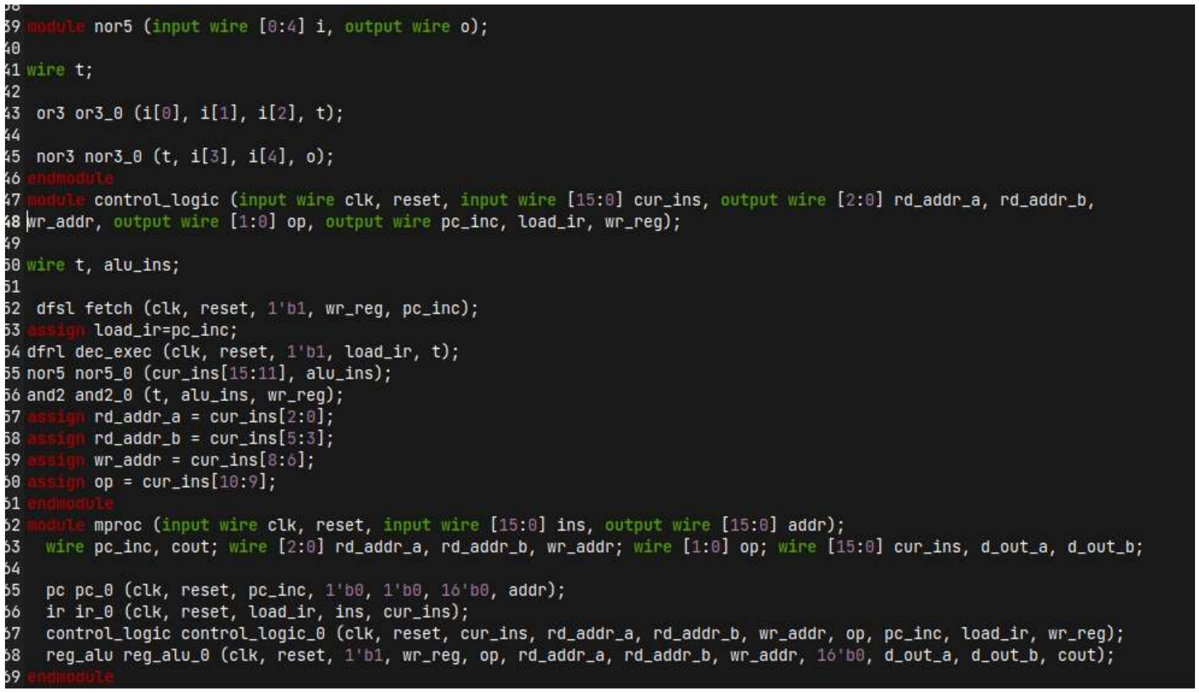
```
module mux2 (input wire i0, i1, j, output wire o);
  assign o = (j==0)?i0:i1;
endmodule

module mux4 (input wire [0:3] i, input wire j1, j0, output wire o);
  wire  t0, t1;
  mux2 mux2_0 (i[0], i[1], j1, t0);
  mux2 mux2_1 (i[2], i[3], j1, t1);
  mux2 mux2_2 (t0, t1, j0, o);
endmodule

module mux8 (input wire [0:7] i, input wire j2, j1, j0, output wire o);
  wire  t0, t1;
  mux4 mux4_0 (i[0:3], j2, j1, t0);
  mux4 mux4_1 (i[4:7], j2, j1, t1);
  mux2 mux2_0 (t0, t1, j0, o);
endmodule

module demux2 (input wire i, j, output wire o0, o1);
  assign o0 = (j==0)?i:1'b0;
  assign o1 = (j==1)?i:1'b0;
endmodule

module demux4 (input wire i, j1, j0, output wire [0:3] o);
  wire  t0, t1;
  demux2 demux2_0 (i, j1, t0, t1);
  demux2 demux2_1 (t0, j0, o[0], o[1]);
  demux2 demux2_2 (t1, j0, o[2], o[3]);
endmodule

module demux8 (input wire i, j2, j1, j0, output wire [0:7] o);
  wire  t0, t1;
  demux2 demux2_0 (i, j2, t0, t1);
  demux4 demux4_0 (t0, j1, j0, o[0:3]);
  demux4 demux4_1 (t1, j1, j0, o[4:7]);
endmodule
```



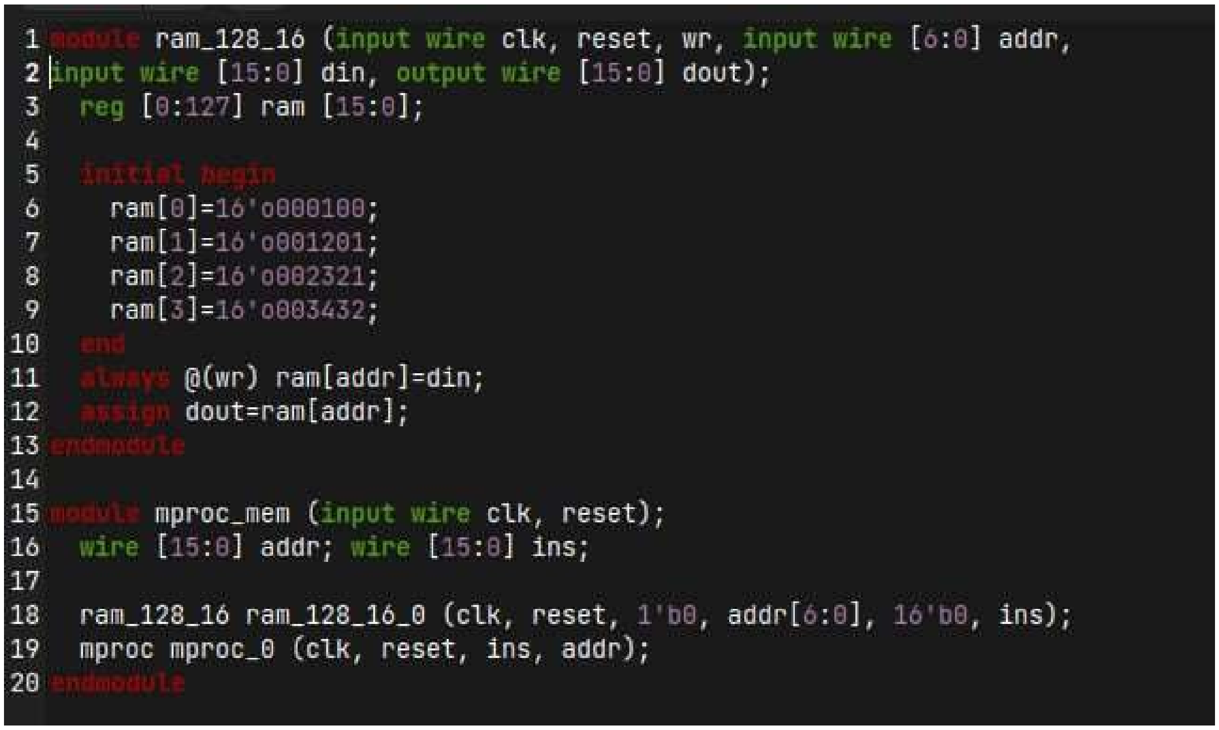
```
module df (input wire clk, in, output wire out);
  reg df_out;
  always@(posedge clk) df_out <= in;
  assign out = df_out;
endmodule

module dfr (input wire clk, reset, in, output wire out);
  wire reset_, df_in;
  invert invert_0 (reset, reset_);
  and2 and2_0 (in, reset_, df_in);
  df df_0 (clk, df_in, out);
endmodule

module dfrl (input wire clk, reset, load, in, output wire out);
  wire _in;
  mux2 mux2_0(out, in, load, _in);
  dfr dfr_1(clk, reset, _in, out);
endmodule

module dfs (input wire clk, set, in, output wire out);
  wire dfr_in,dfr_out;
  invert invert_0(in, dfr_in);
  invert invert_1(dfr_out, out);
  dfr dfr_2(clk, set, dfr_in, dfr_out);
endmodule

module dfsl (input wire clk, set, load, in, output wire out);
  wire _in;
  mux2 mux2_0(out, in, load, _in);
  dfs dfs_1(clk, set, _in, out);
endmodule
```



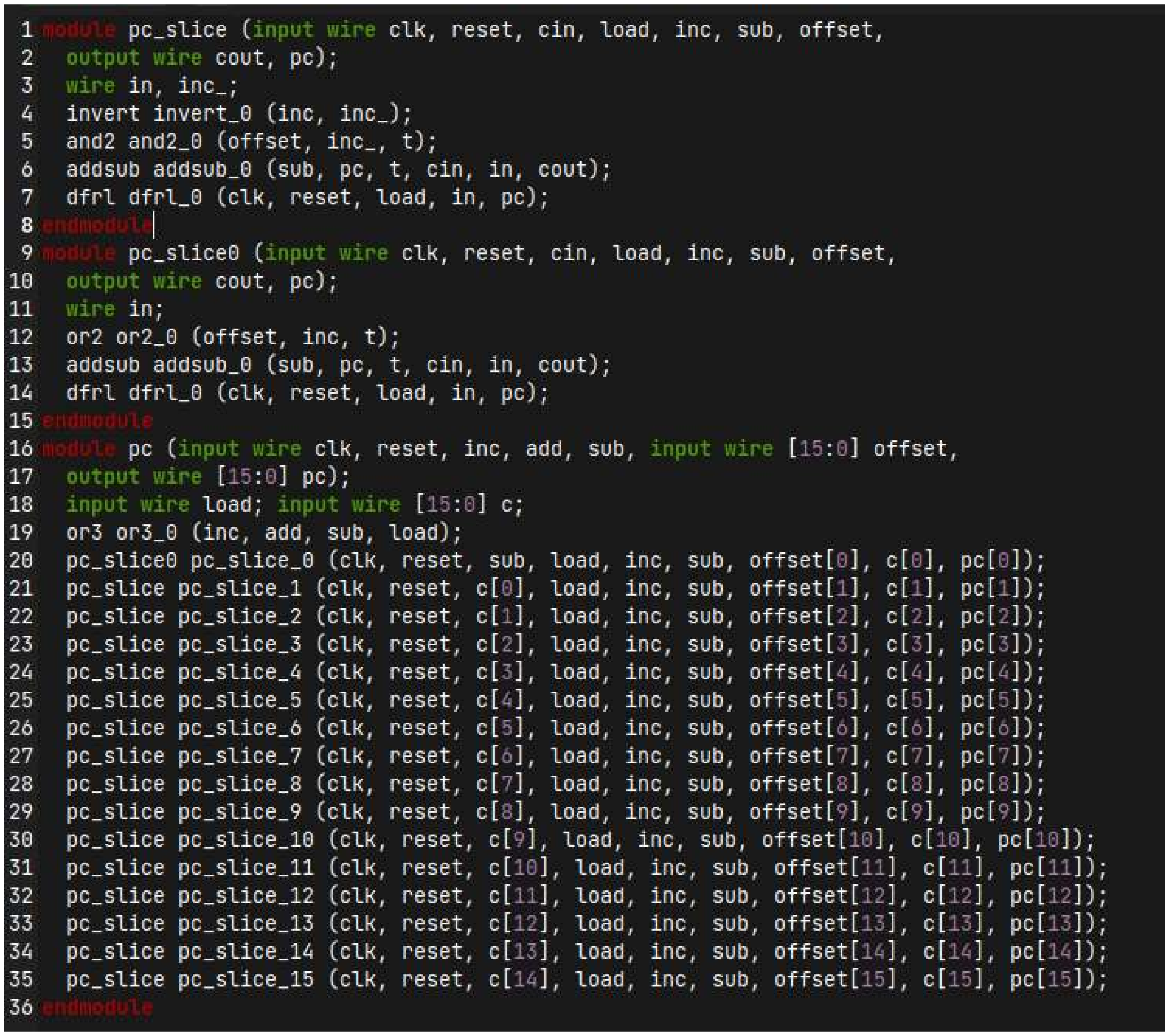
```
module ir (input wire clk, reset, load, input wire [15:0] din, output wire [15:0] dout);
  dfrl dfrl_0 (clk, reset, load, din['h0], dout['h0]);

dfrl dfrl_1 (clk, reset, load, din['h1], dout['h1]);

 dfrl dfrl_2 (clk, reset, load, din['h2], dout['h2]);

dfrl dfrl_3 (clk, reset, load, din['h3], dout['h3]);

dfrl dfrl_4 (clk, reset, load, din['h4], dout['h4]);

dfrl dfrl_5 (clk, reset, load, din['h5], dout['h5]);

dfrl dfrl_6 (clk, reset, load, din['h6], dout['h6]);

dfrl dfrl_7 (clk, reset, load, din['h7], dout['h7]);

dfrl dfrl_8 (clk, reset, load, din['h8], dout['h8]);

dfrl dfrl_9 (clk, reset, load, din['h9], dout['h9]);

dfrl dfrl_a (clk, reset, load, din['ha], dout['ha]);

dfrl dfrl_b (clk, reset, load, din['hb], dout['hb]);

dfrl dfrl_c (clk, reset, load, din['hc], dout['hc]);

 dfrl dfrl_d (clk, reset, load, din['hd], dout['hd]);

dfrl dfrl_e (clk, reset, load, din['he], dout['he]);

 dfrl dfrl_f (clk, reset, load, din['hf], dout['hf]);

endmodule
```



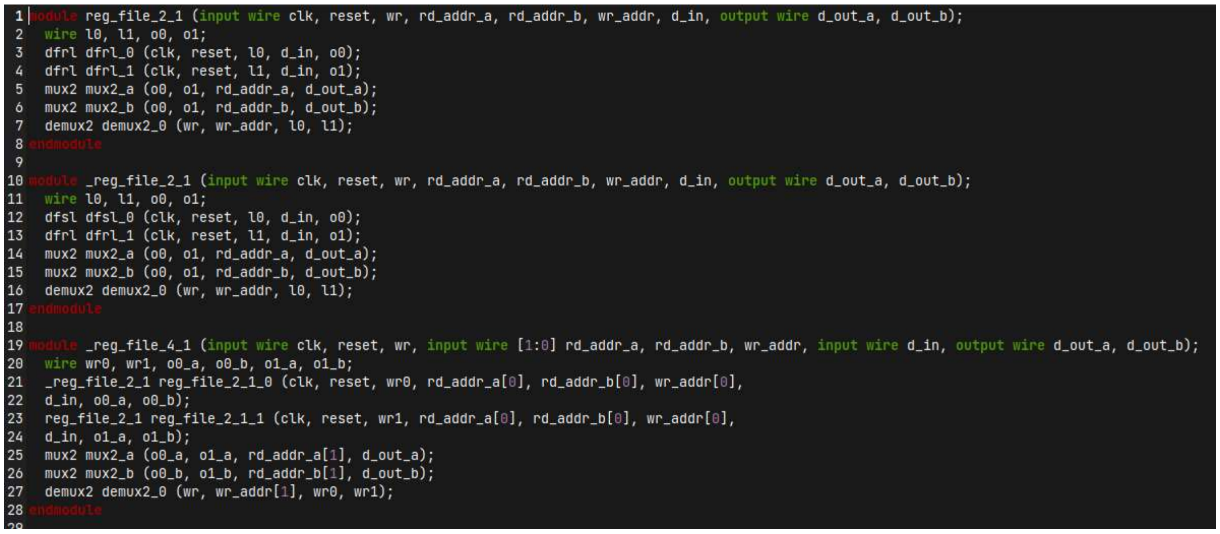
```verilog
module nor5 (input wire [0:4] i, output wire o);

wire t;

 or3 or3_0 (i[0], i[1], i[2], t);

 nor3 nor3_0 (t, i[3], i[4], o);
endmodule
module control_logic (input wire clk, reset, input wire [15:0] cur_ins, output wire [2:0] rd_addr_a, rd_addr_b,
wr_addr, output wire [1:0] op, output wire pc_inc, load_ir, wr_reg);

wire t, alu_ins;

 dfsl fetch (clk, reset, 1'b1, wr_reg, pc_inc);
assign load_ir=pc_inc;
dfrl dec_exec (clk, reset, 1'b1, load_ir, t);
nor5 nor5_0 (cur_ins[15:11], alu_ins);
and2 and2_0 (t, alu_ins, wr_reg);
assign rd_addr_a = cur_ins[2:0];
assign rd_addr_b = cur_ins[5:3];
assign wr_addr = cur_ins[8:6];
assign op = cur_ins[10:9];
endmodule
module mproc (input wire clk, reset, input wire [15:0] ins, output wire [15:0] addr);
  wire pc_inc, cout; wire [2:0] rd_addr_a, rd_addr_b, wr_addr; wire [1:0] op; wire [15:0] cur_ins, d_out_a, d_out_b;

  pc pc_0 (clk, reset, pc_inc, 1'b0, 1'b0, 16'b0, addr);
  ir ir_0 (clk, reset, load_ir, ins, cur_ins);
  control_logic control_logic_0 (clk, reset, cur_ins, rd_addr_a, rd_addr_b, wr_addr, op, pc_inc, load_ir, wr_reg);
  reg_alu reg_alu_0 (clk, reset, 1'b1, wr_reg, op, rd_addr_a, rd_addr_b, wr_addr, 16'b0, d_out_a, d_out_b, cout);
endmodule
```



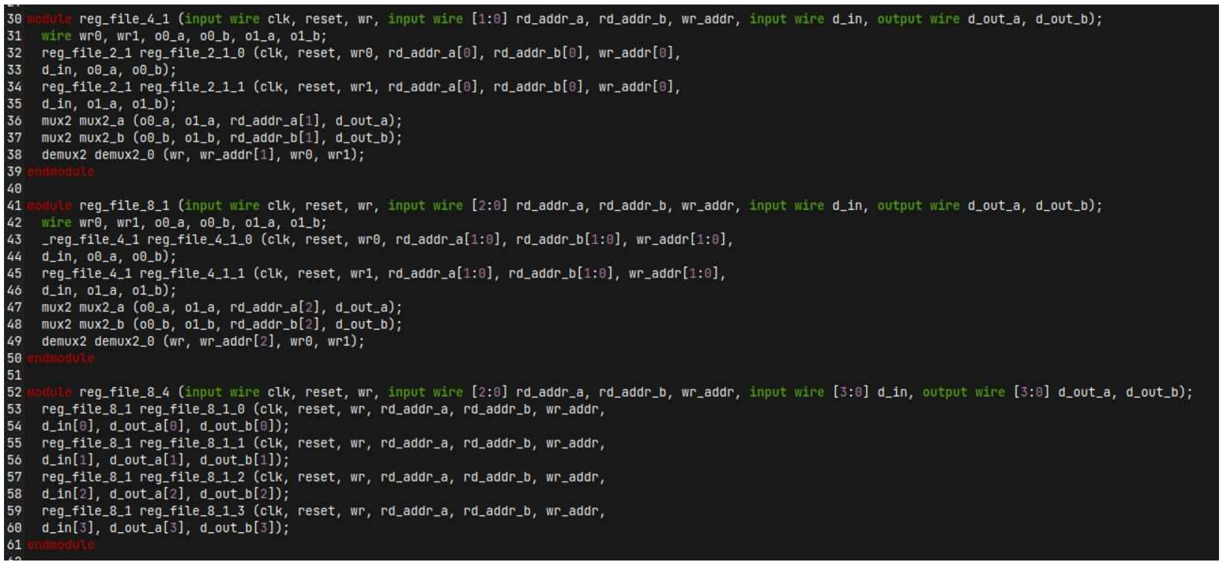
```
module ram_128_16 (input wire clk, reset, wr, input wire [6:0] addr,
input wire [15:0] din, output wire [15:0] dout);
  reg [0:127] ram [15:0];

  initial begin
    ram[0]=16'o000100;
    ram[1]=16'o001201;
    ram[2]=16'o002321;
    ram[3]=16'o003432;
  end
  always @(wr) ram[addr]=din;
  assign dout=ram[addr];
endmodule

module mproc_mem (input wire clk, reset);
  wire [15:0] addr; wire [15:0] ins;

  ram_128_16 ram_128_16_0 (clk, reset, 1'b0, addr[6:0], 16'b0, ins);
  mproc mproc_0 (clk, reset, ins, addr);
endmodule
```



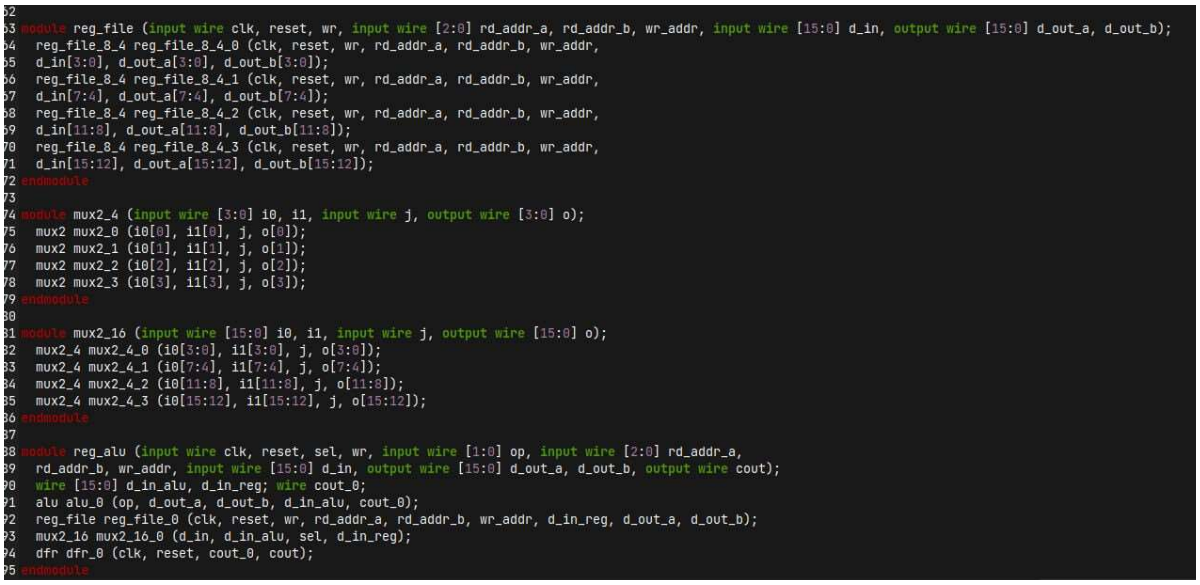
```
module pc_slice (input wire clk, reset, cin, load, inc, sub, offset,
  output wire cout, pc);
  wire in, inc_;
  invert invert_0 (inc, inc_);
  and2 and2_0 (offset, inc_, t);
  addsub addsub_0 (sub, pc, t, cin, in, cout);
  dfrl dfrl_0 (clk, reset, load, in, pc);
endmodule
module pc_slice0 (input wire clk, reset, cin, load, inc, sub, offset,
  output wire cout, pc);
  wire in;
  or2 or2_0 (offset, inc, t);
  addsub addsub_0 (sub, pc, t, cin, in, cout);
  dfrl dfrl_0 (clk, reset, load, in, pc);
endmodule
module pc (input wire clk, reset, inc, add, sub, input wire [15:0] offset,
  output wire [15:0] pc);
  input wire load; input wire [15:0] c;
  or3 or3_0 (inc, add, sub, load);
  pc_slice0 pc_slice_0 (clk, reset, sub, load, inc, sub, offset[0], c[0], pc[0]);
  pc_slice pc_slice_1 (clk, reset, c[0], load, inc, sub, offset[1], c[1], pc[1]);
  pc_slice pc_slice_2 (clk, reset, c[1], load, inc, sub, offset[2], c[2], pc[2]);
  pc_slice pc_slice_3 (clk, reset, c[2], load, inc, sub, offset[3], c[3], pc[3]);
  pc_slice pc_slice_4 (clk, reset, c[3], load, inc, sub, offset[4], c[4], pc[4]);
  pc_slice pc_slice_5 (clk, reset, c[4], load, inc, sub, offset[5], c[5], pc[5]);
  pc_slice pc_slice_6 (clk, reset, c[5], load, inc, sub, offset[6], c[6], pc[6]);
  pc_slice pc_slice_7 (clk, reset, c[6], load, inc, sub, offset[7], c[7], pc[7]);
  pc_slice pc_slice_8 (clk, reset, c[7], load, inc, sub, offset[8], c[8], pc[8]);
  pc_slice pc_slice_9 (clk, reset, c[8], load, inc, sub, offset[9], c[9], pc[9]);
  pc_slice pc_slice_10 (clk, reset, c[9], load, inc, sub, offset[10], c[10], pc[10]);
  pc_slice pc_slice_11 (clk, reset, c[10], load, inc, sub, offset[11], c[11], pc[11]);
  pc_slice pc_slice_12 (clk, reset, c[11], load, inc, sub, offset[12], c[12], pc[12]);
  pc_slice pc_slice_13 (clk, reset, c[12], load, inc, sub, offset[13], c[13], pc[13]);
  pc_slice pc_slice_14 (clk, reset, c[13], load, inc, sub, offset[14], c[14], pc[14]);
  pc_slice pc_slice_15 (clk, reset, c[14], load, inc, sub, offset[15], c[15], pc[15]);
endmodule
```



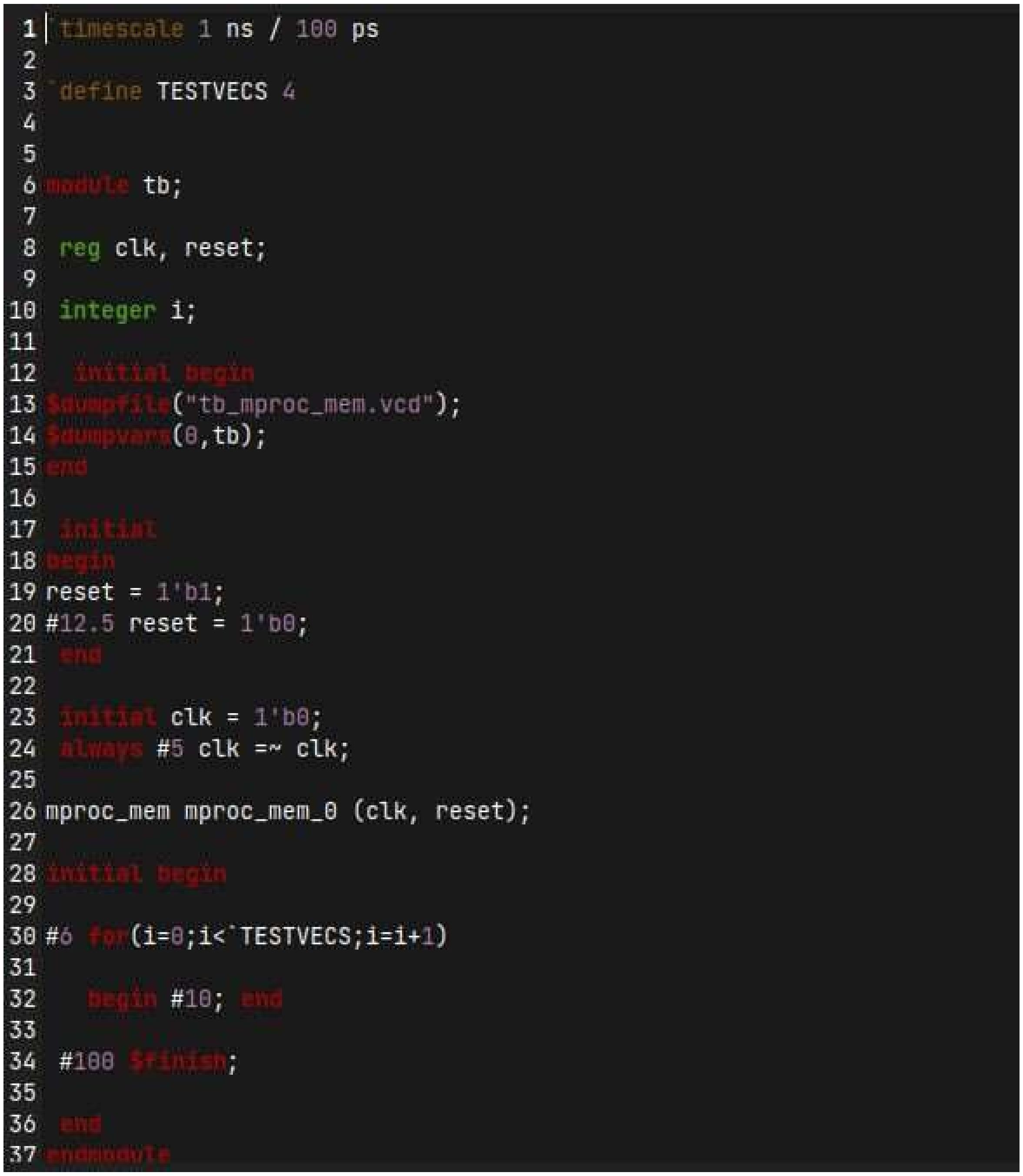
```verilog
module reg_file_2_1 (input wire clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, d_in, output wire d_out_a, d_out_b);
  wire l0, l1, o0, o1;
  dfrl dfrl_0 (clk, reset, l0, d_in, o0);
  dfrl dfrl_1 (clk, reset, l1, d_in, o1);
  mux2 mux2_a (o0, o1, rd_addr_a, d_out_a);
  mux2 mux2_b (o0, o1, rd_addr_b, d_out_b);
  demux2 demux2_0 (wr, wr_addr, l0, l1);
endmodule

module _reg_file_2_1 (input wire clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, d_in, output wire d_out_a, d_out_b);
  wire l0, l1, o0, o1;
  dfsl dfsl_0 (clk, reset, l0, d_in, o0);
  dfrl dfrl_1 (clk, reset, l1, d_in, o1);
  mux2 mux2_a (o0, o1, rd_addr_a, d_out_a);
  mux2 mux2_b (o0, o1, rd_addr_b, d_out_b);
  demux2 demux2_0 (wr, wr_addr, l0, l1);
endmodule

module _reg_file_4_1 (input wire clk, reset, wr, input wire [1:0] rd_addr_a, rd_addr_b, wr_addr, input wire d_in, output wire d_out_a, d_out_b);
  wire wr0, wr1, o0_a, o0_b, o1_a, o1_b;
  _reg_file_2_1 reg_file_2_1_0 (clk, reset, wr0, rd_addr_a[0], rd_addr_b[0], wr_addr[0],
  d_in, o0_a, o0_b);
  reg_file_2_1 reg_file_2_1_1 (clk, reset, wr1, rd_addr_a[0], rd_addr_b[0], wr_addr[0],
  d_in, o1_a, o1_b);
  mux2 mux2_a (o0_a, o1_a, rd_addr_a[1], d_out_a);
  mux2 mux2_b (o0_b, o1_b, rd_addr_b[1], d_out_b);
  demux2 demux2_0 (wr, wr_addr[1], wr0, wr1);
endmodule
```



```verilog
module reg_file_4_1 (input wire clk, reset, wr, input wire [1:0] rd_addr_a, rd_addr_b, wr_addr, input wire d_in, output wire d_out_a, d_out_b);
  wire wr0, wr1, o0_a, o0_b, o1_a, o1_b;
  reg_file_2_1 reg_file_2_1_0 (clk, reset, wr0, rd_addr_a[0], rd_addr_b[0], wr_addr[0],
  d_in, o0_a, o0_b);
  reg_file_2_1 reg_file_2_1_1 (clk, reset, wr1, rd_addr_a[0], rd_addr_b[0], wr_addr[0],
  d_in, o1_a, o1_b);
  mux2 mux2_a (o0_a, o1_a, rd_addr_a[1], d_out_a);
  mux2 mux2_b (o0_b, o1_b, rd_addr_b[1], d_out_b);
  demux2 demux2_0 (wr, wr_addr[1], wr0, wr1);
endmodule

module reg_file_8_1 (input wire clk, reset, wr, input wire [2:0] rd_addr_a, rd_addr_b, wr_addr, input wire d_in, output wire d_out_a, d_out_b);
  wire wr0, wr1, o0_a, o0_b, o1_a, o1_b;
  _reg_file_4_1 reg_file_4_1_0 (clk, reset, wr0, rd_addr_a[1:0], rd_addr_b[1:0], wr_addr[1:0],
  d_in, o0_a, o0_b);
  reg_file_4_1 reg_file_4_1_1 (clk, reset, wr1, rd_addr_a[1:0], rd_addr_b[1:0], wr_addr[1:0],
  d_in, o1_a, o1_b);
  mux2 mux2_a (o0_a, o1_a, rd_addr_a[2], d_out_a);
  mux2 mux2_b (o0_b, o1_b, rd_addr_b[2], d_out_b);
  demux2 demux2_0 (wr, wr_addr[2], wr0, wr1);
endmodule

module reg_file_8_4 (input wire clk, reset, wr, input wire [2:0] rd_addr_a, rd_addr_b, wr_addr, input wire [3:0] d_in, output wire [3:0] d_out_a, d_out_b);
  reg_file_8_1 reg_file_8_1_0 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
  d_in[0], d_out_a[0], d_out_b[0]);
  reg_file_8_1 reg_file_8_1_1 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
  d_in[1], d_out_a[1], d_out_b[1]);
  reg_file_8_1 reg_file_8_1_2 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
  d_in[2], d_out_a[2], d_out_b[2]);
  reg_file_8_1 reg_file_8_1_3 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
  d_in[3], d_out_a[3], d_out_b[3]);
endmodule
```

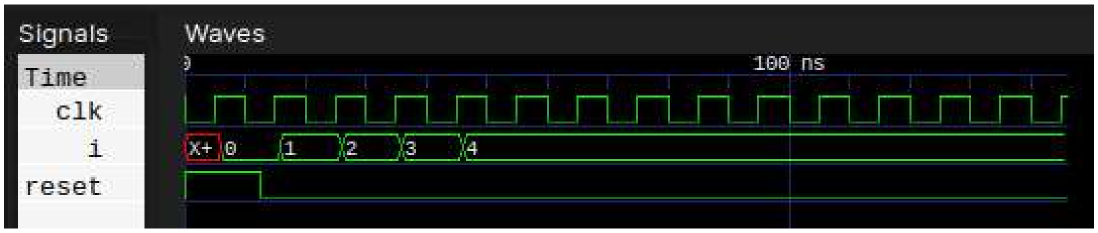


```verilog
module reg_file (input wire clk, reset, wr, input wire [2:0] rd_addr_a, rd_addr_b, wr_addr, input wire [15:0] d_in, output wire [15:0] d_out_a, d_out_b);
  reg_file_8_4 reg_file_8_4_0 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
  d_in[3:0], d_out_a[3:0], d_out_b[3:0]);
  reg_file_8_4 reg_file_8_4_1 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
  d_in[7:4], d_out_a[7:4], d_out_b[7:4]);
  reg_file_8_4 reg_file_8_4_2 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
  d_in[11:8], d_out_a[11:8], d_out_b[11:8]);
  reg_file_8_4 reg_file_8_4_3 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr,
  d_in[15:12], d_out_a[15:12], d_out_b[15:12]);
endmodule

module mux2_4 (input wire [3:0] i0, i1, input wire j, output wire [3:0] o);
  mux2 mux2_0 (i0[0], i1[0], j, o[0]);
  mux2 mux2_1 (i0[1], i1[1], j, o[1]);
  mux2 mux2_2 (i0[2], i1[2], j, o[2]);
  mux2 mux2_3 (i0[3], i1[3], j, o[3]);
endmodule

module mux2_16 (input wire [15:0] i0, i1, input wire j, output wire [15:0] o);
  mux2_4 mux2_4_0 (i0[3:0], i1[3:0], j, o[3:0]);
  mux2_4 mux2_4_1 (i0[7:4], i1[7:4], j, o[7:4]);
  mux2_4 mux2_4_2 (i0[11:8], i1[11:8], j, o[11:8]);
  mux2_4 mux2_4_3 (i0[15:12], i1[15:12], j, o[15:12]);
endmodule

module reg_alu (input wire clk, reset, sel, wr, input wire [1:0] op, input wire [2:0] rd_addr_a,
  rd_addr_b, wr_addr, input wire [15:0] d_in, output wire [15:0] d_out_a, d_out_b, output wire cout);
  wire [15:0] d_in_alu, d_in_reg; wire cout_0;
  alu alu_0 (op, d_out_a, d_out_b, d_in_alu, cout_0);
  reg_file reg_file_0 (clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, d_in_reg, d_out_a, d_out_b);
  mux2_16 mux2_16_0 (d_in, d_in_alu, sel, d_in_reg);
  dfr dfr_0 (clk, reset, cout_0, cout);
endmodule
```



```verilog
`timescale 1 ns / 100 ps

`define TESTVECS 4


module tb;

 reg clk, reset;

 integer i;

  initial begin
$dumpfile("tb_mproc_mem.vcd");
$dumpvars(0,tb);
end

 initial
begin
reset = 1'b1;
#12.5 reset = 1'b0;
 end

 initial clk = 1'b0;
 always #5 clk =~ clk;

mproc_mem mproc_mem_0 (clk, reset);

initial begin

#6  for(i=0;i<`TESTVECS;i=i+1)

   begin #10; end

 #100 $finish;

 end
endmodule
```

Output waveform


Signals
Waves
Time
clk
i
reset
100 ns
X+