Digital Design and Computer Organisation Laboratory

UE23CS251A

3rd Semester, Academic Year 2023

Date:22/10/2024

Name: Pranav Hemanth Name: Nishant Holla Name: Aneesh Dutt Name: Nagarjun N H	SRN:PES1UG23CS433 SRN:PES1UG23CS401 SRN:PES1UG23CS371 SRN:PES1UG23CS375	Section G

TITLE: Hackathon

Program Counter:

I. Verilog Code Screenshot

registerfile > lib.v

```
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
```

Hackathon > Final > alu8.v

```
// 8-bit ALU
// operations
// 00 - ADD
// 01 - SUB
// 10 - AND
// 11 - OR

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

module alu (input wire [1:0] op, input wire [7:0] i0, i1,
   output wire [7:0] o, output wire cout);
   wire  [7:0] 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]);
endmodule
```

Hackathon > Final > regfile.v

```
module reg16(input wire clk,reset,load,input wire [7:0]din,output wire [7:0]r);
    dfrl d0(clk,reset,load,din[0],r[0]);
    dfrl d1(clk,reset,load,din[1],r[1]);
    dfrl d2(clk,reset,load,din[2],r[2]);
    dfrl d3(clk,reset,load,din[3],r[3]);
    dfrl d4(clk,reset,load,din[4],r[4]);
    dfrl d5(clk,reset,load,din[5],r[5]);
    dfrl d6(clk,reset,load,din[6],r[6]);
    dfrl d7(clk,reset,load,din[7],r[7]);
endmodule

module reg_file (input wire clk, reset, wr, input wire [2:0] rd_addr_a, rd_addr_b, wr_addr, input wire [7:0] d_in, output wire [7:0] d_out_a, d_out_b);
    wire [0:7]load;
    wire [0:7]r0,r1,r2,r3,r4,r5,r6,r7;
    demux8 dmx(wr,wr_addr[2],wr_addr[1],wr_addr[0],load);
    reg16 reg0(clk,reset,load[0],d_in,r0);
    reg16 reg1(clk,reset,load[1],d_in,r1);
    reg16 reg2(clk,reset,load[2],d_in,r2);
    reg16 reg3(clk,reset,load[3],d_in,r3);
    reg16 reg4(clk,reset,load[4],d_in,r4);
    reg16 reg5(clk,reset,load[5],d_in,r5);
    reg16 reg6(clk,reset,load[6],d_in,r6);
    reg16 reg7(clk,reset,load[7],d_in,r7);
    mux128_16 mm0(r0,r1,r2,r3,r4,r5,r6,r7,rd_addr_a[0],rd_addr_a[1],rd_addr_a[2],d_out_a);
    mux128_16 mm1(r0,r1,r2,r3,r4,r5,r6,r7,rd_addr_b[0],rd_addr_b[1],rd_addr_b[2],d_out_b);
endmodule

module mux128_16(input wire [7:0]i0,i1,i2,i3,i4,i5,i6,i7,input wire s0,s1,s2,output wire [7:0]o);
    mux8 mx0({i0[0],i1[0],i2[0],i3[0],i4[0],i5[0],i6[0],i7[0]},s2,s1,s0,o[0]);
    mux8 mx1({i0[1],i1[1],i2[1],i3[1],i4[1],i5[1],i6[1],i7[1]},s2,s1,s0,o[1]);
    mux8 mx2({i0[2],i1[2],i2[2],i3[2],i4[2],i5[2],i6[2],i7[2]},s2,s1,s0,o[2]);
    mux8 mx3({i0[3],i1[3],i2[3],i3[3],i4[3],i5[3],i6[3],i7[3]},s2,s1,s0,o[3]);
    mux8 mx4({i0[4],i1[4],i2[4],i3[4],i4[4],i5[4],i6[4],i7[4]},s2,s1,s0,o[4]);
    mux8 mx5({i0[5],i1[5],i2[5],i3[5],i4[5],i5[5],i6[5],i7[5]},s2,s1,s0,o[5]);
    mux8 mx6({i0[6],i1[6],i2[6],i3[6],i4[6],i5[6],i6[6],i7[6]},s2,s1,s0,o[6]);
    mux8 mx7({i0[7],i1[7],i2[7],i3[7],i4[7],i5[7],i6[7],i7[7]},s2,s1,s0,o[7]);
endmodule
```

Hackathon > Final > regfile.v

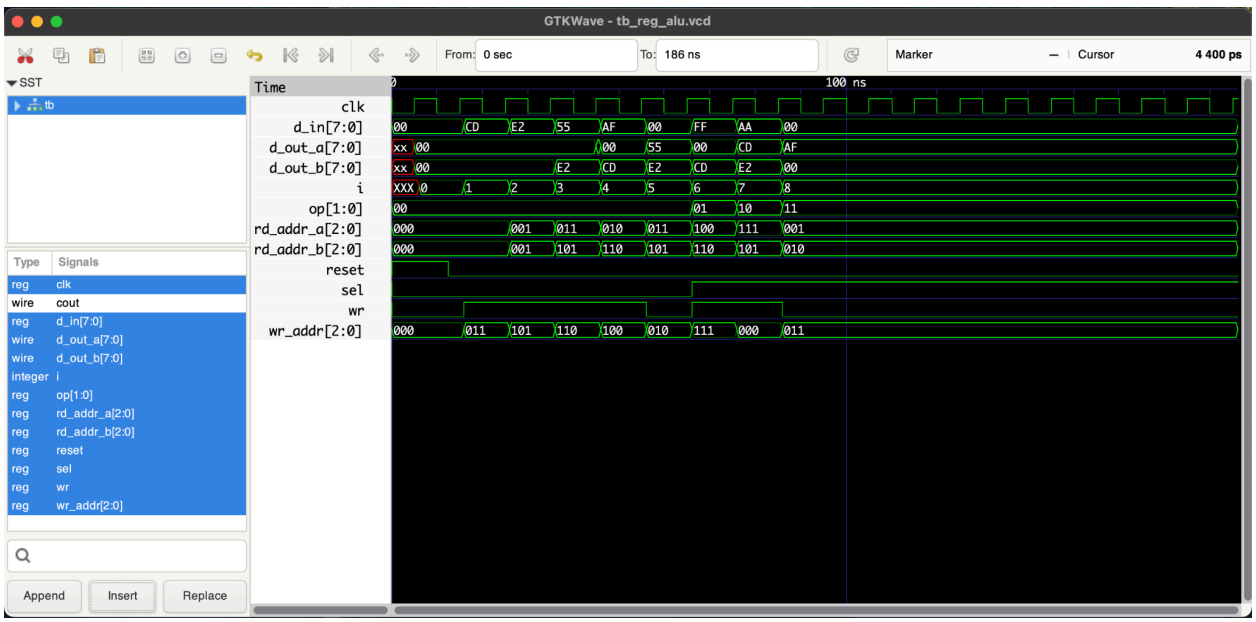
```

module mux128_16(input wire [7:0]i0,i1,i2,i3,i4,i5,i6,i7,input wire s0,s1,s2,output wire [7:0]o);
    mux8 mx0({i0[0],i1[0],i2[0],i3[0],i4[0],i5[0],i6[0],i7[0]},s2,s1,s0,o[0]);
    mux8 mx1({i0[1],i1[1],i2[1],i3[1],i4[1],i5[1],i6[1],i7[1]},s2,s1,s0,o[1]);
    mux8 mx2({i0[2],i1[2],i2[2],i3[2],i4[2],i5[2],i6[2],i7[2]},s2,s1,s0,o[2]);
    mux8 mx3({i0[3],i1[3],i2[3],i3[3],i4[3],i5[3],i6[3],i7[3]},s2,s1,s0,o[3]);
    mux8 mx4({i0[4],i1[4],i2[4],i3[4],i4[4],i5[4],i6[4],i7[4]},s2,s1,s0,o[4]);
    mux8 mx5({i0[5],i1[5],i2[5],i3[5],i4[5],i5[5],i6[5],i7[5]},s2,s1,s0,o[5]);
    mux8 mx6({i0[6],i1[6],i2[6],i3[6],i4[6],i5[6],i6[6],i7[6]},s2,s1,s0,o[6]);
    mux8 mx7({i0[7],i1[7],i2[7],i3[7],i4[7],i5[7],i6[7],i7[7]},s2,s1,s0,o[7]);
endmodule

module mux2for16(input wire [7:0] din_regular, alu_out, input wire selector, output wire [7:0]din_final);
    mux2 m0(din_regular[0], alu_out[0], selector, din_final[0]);
    mux2 m1(din_regular[1], alu_out[1], selector, din_final[1]);
    mux2 m2(din_regular[2], alu_out[2], selector, din_final[2]);
    mux2 m3(din_regular[3], alu_out[3], selector, din_final[3]);
    mux2 m4(din_regular[4], alu_out[4], selector, din_final[4]);
    mux2 m5(din_regular[5], alu_out[5], selector, din_final[5]);
    mux2 m6(din_regular[6], alu_out[6], selector, din_final[6]);
    mux2 m7(din_regular[7], alu_out[7], selector, din_final[7]);
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 [7:0] d_in, output wire [7:0] d_out_a, d_out_b, output wire cout);
    wire [7:0] alu_out;
    wire [7:0] newdin;
    mux2for16 select(d_in, alu_out, sel, newdin);
    reg_file new_reg(clk, reset, wr, rd_addr_a, rd_addr_b, wr_addr, newdin, d_out_a, d_out_b);
    alu calc (op, d_out_a, d_out_b, alu_out, cout);
endmodule

```

II. Verilog VVP Output Screen Shot

```
regfile_tb.v:47: $finish called at 1860 (100ps)
pranavhemanth@Pranavs-MacBook-Pro-M3 Final %iverilog -o output *.v
pranavhemanth@Pranavs-MacBook-Pro-M3 Final %vvp output
VCD info: dumpfile tb_reg_alu.vcd opened for output.
At time                   0: sel = 0, wr = 0, op = 00, rd_addr_a = 0, rd_addr_b = 0, wr_addr = 0, d_in =   0, d_out_a =   x, d_out_b =   x,
At time                  50: sel = 0, wr = 0, op = 00, rd_addr_a = 0, rd_addr_b = 0, wr_addr = 0, d_in =   0, d_out_a =   0, d_out_b =   0,
At time                 160: sel = 0, wr = 1, op = 00, rd_addr_a = 0, rd_addr_b = 0, wr_addr = 3, d_in = 205, d_out_a =   0, d_out_b =   0,
At time                 260: sel = 0, wr = 1, op = 00, rd_addr_a = 1, rd_addr_b = 1, wr_addr = 5, d_in = 226, d_out_a =   0, d_out_b =   0,
At time                 360: sel = 0, wr = 1, op = 00, rd_addr_a = 3, rd_addr_b = 5, wr_addr = 6, d_in =  85, d_out_a =   0, d_out_b = 226,
At time                 450: sel = 0, wr = 1, op = 00, rd_addr_a = 3, rd_addr_b = 5, wr_addr = 6, d_in =  85, d_out_a =  85, d_out_b = 226,
At time                 460: sel = 0, wr = 1, op = 00, rd_addr_a = 2, rd_addr_b = 6, wr_addr = 4, d_in = 175, d_out_a =   0, d_out_b = 205,
At time                 560: sel = 0, wr = 0, op = 00, rd_addr_a = 3, rd_addr_b = 5, wr_addr = 2, d_in =   0, d_out_a =  85, d_out_b = 226,
At time                 660: sel = 1, wr = 1, op = 01, rd_addr_a = 4, rd_addr_b = 6, wr_addr = 7, d_in = 255, d_out_a =   0, d_out_b = 205,
At time                 760: sel = 1, wr = 1, op = 10, rd_addr_a = 7, rd_addr_b = 5, wr_addr = 0, d_in = 170, d_out_a = 205, d_out_b = 226,
At time                 860: sel = 1, wr = 0, op = 11, rd_addr_a = 1, rd_addr_b = 2, wr_addr = 3, d_in =   0, d_out_a = 175, d_
regfile_tb.v:52: $finish called at 1860 (100ps)
pranavhemanth@Pranavs-MacBook-Pro-M3 Final %
```

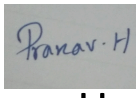
System Se

III. GTKWAVE Screenshot



Disclaimer:

- The programs and output submitted are duly written, verified and executed by me.
- I have not copied from any of my peers nor from external resources such as the internet.
- If found plagiarized, I will abide with the disciplinary action of the University.

Signature: 
Name: Pranav Hemanth
SRN: PES1UG23CS433
Section: G
Date: 22/10/24