2020-2021 Spring Semester

Lab5 - Preliminary Report

Course: CS223

Name: Alper

Surname: Mumcular

Student ID: 21902740

Lab: 5

Date: 13.04.2021

Trainer Pack: 19

A) Systemverilog code for RegisterFile, ALU and MUX

1) Systemverilog code for RegisterFile

```
module register(
input logic [2:0] rda1,
input logic [2:0] rda2,
input logic [2:0] wra,
input logic [3:0] wrd,
input logic en,
input logic clk,
output logic [3:0] rdd1,
output logic [3:0] rdd2
   );
   reg [3:0] reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7;
   assign rdd1 = rda1 == 0 ? reg0 :
                 rda1 == 1 ? reg1 :
                 rda1 == 2 ? reg2 :
                 rda1 == 3 ? reg3 :
                 rda1 == 4 ? reg4 :
                 rda1 == 5 ? reg5 :
                 rda1 == 6 ? reg6 :
                 rda1 == 7 ? reg7 : 0;
   assign rdd2 = rda2 == 0 ? reg0 :
                 rda2 == 1 ? reg1 :
                 rda2 == 2 ? reg2 :
                 rda2 == 3 ? reg3 :
                 rda2 == 4 ? reg4 :
                 rda2 == 5 ? reg5 :
                 rda2 == 6 ? reg6 :
                 rda2 == 7 ? reg7 : 0;
   always @(posedge clk) begin
      if(en)
         case(wra)
         0: begin
            reg0 <= wrd;
         end
         1: begin
            reg1 <= wrd;
         end
         2: begin
            reg2 <= wrd;
         end
         3: begin
            reg3 <= wrd;
         end
         4: begin
            reg4 <= wrd;
         end
         5: begin
            reg5 <= wrd;
         end
         6: begin
            reg6 <= wrd;
         end
```

```
        7: begin
           reg7 <= wrd;
        end
        endcase
   end
endmodule
```

2) Systemverilog code for ALU

```
module alu(
input logic [3:0] a,
input logic [3:0] b,
input logic [1:0] sel,
output logic [3:0] res
   );
   always @(*)
   begin
      case(sel)
      2'b00 : res = a + 1;
      2'b01 : res = a + b;
      2'b10 : res = a - b;
      2'b11 : res = a | b;
      default : res = a + 1;
      endcase
   end

endmodule
```

3) Systemverilog code for 4-bit 2-to-1 MUX

```
module mux2to1(
input logic [3:0] extdata,
input logic [3:0] res,
input logic isExternal,
output logic [3:0] wrd
   );
   assign wrd = isExternal ? extdata : res;
endmodule
```

B) Testbenches for ALU and RegisterFile modules

1) Testbench for ALU

```
module alutest();
logic [3:0] a;
logic [3:0] b;
logic [1:0] sel;
logic [3:0] res;
alu dut(a,b,sel,res);
initial begin
a <= 4'b0001; b <= 4'b0101; sel <= 2'b00; #100;
a <= 4'b0101; b <= 4'b0010; #100;
sel <= 2'b01; #100;
```

```
a <= 4'b0011; b <= 4'b0101; #100;
a <= 4'b0101; b <= 4'b0011; sel <= 2'b10; #100;
a <= 4'b0011; b <= 4'b0101; sel <= 2'b11; #100;
end
endmodule
```

2) Testbench for RegisterFile

```
module registertest();
logic [2:0] rda1;
logic [2:0] rda2;
logic [2:0] wra;
logic [3:0] wrd;
logic en;
logic clk;
logic [3:0] rdd1;
logic [3:0] rdd2;

register dut(rda1,rda2,wra,wrd,en,clk,rdd1,rdd2);

always
      begin
         clk <= 1; #50;
         clk <= 0; #50;
      end

initial begin
rda1 <= 3'b001; rda2 <= 3'b000; wra <= 3'b001; wrd <= 4'b0101; en <= 1; #50; #50;
wrd <= 4'b0111; #50; #50;
rda1 <= 3'b010; rda2 <= 3'b001; wra <= 3'b010; wrd <= 4'b0110; en <= 1; #50; #50;
end
endmodule
```

3) Top module for the datapath and an overall testbench

1) Systemverilog code for datapath

```
module datapath(

input logic [3:0] extdata,

input logic isExternal,

input logic [2:0] AddrSrc1,

input logic [2:0] AddrSrc2,

input logic [2:0] AddrDest,

input logic pushButn,

input logic clk,

input logic [1:0] ALUSel,

output logic [6:0] seg, //
```

```
output logic dp, //
output logic [3:0] an  // In testbench = output logic [3:0] res
   );
   logic [3:0] wrd;
   logic en;
   logic [3:0] rdd1;
   logic [3:0] rdd2;
   logic [3:0] res;

   debouncer deb(clk,pushButn,en);
   mux2to1 mux(extdata,res,isExternal,wrd);
   register rg(AddrSrc1,AddrSrc2,AddrDest,wrd,en,clk,rdd1,rdd2);
   alu al(rdd1,rdd2,ALUSel,res);
   SevenSegmentDisplay svn(clk,res[3],res[2],res[1],res[0],seg,dp,an); // not included in testbench

endmodule
```

2) Testbench for the datapath

```
module datapathtest();
logic [3:0] extdata;
logic isExternal;
logic [2:0] AddrSrc1;
logic [2:0] AddrSrc2;
logic [2:0] AddrDest;
logic pushButn;
logic clk;
logic [1:0] ALUSel;
logic [3:0] res;
datapath dut(extdata,isExternal,AddrSrc1,AddrSrc2,AddrDest,pushButn,clk,ALUSel,res);

initial begin
pushButn <= 1; #0;
end
always
      begin
         clk <= 1;#5;
         clk <= 0;#5;
      end

initial begin
extdata <= 4'b0100; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0000;
ALUSel <= 2'b00; #30;
extdata <= 4'b0010; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0011;
ALUSel <= 2'b01; #30;
isExternal <= 0; AddrDest <= 4'b0011; ALUSel <= 2'b10; #30;
```

```
extdata <= 4'b0100; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0000;
ALUSel <= 2'b00; #30;
extdata <= 4'b0010; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0011;
ALUSel <= 2'b01; #30;
isExternal <= 0; AddrDest <= 4'b0011; ALUSel <= 2'b10; #30;
extdata <= 4'b0100; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0000;
ALUSel <= 2'b00; #30;
extdata <= 4'b0010; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0011;
ALUSel <= 2'b01; #30;
isExternal <= 0; AddrDest <= 4'b0011; ALUSel <= 2'b10; #30;
extdata <= 4'b0100; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0000;
ALUSel <= 2'b00; #30;
extdata <= 4'b0010; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0011;
ALUSel <= 2'b01; #30;
isExternal <= 0; AddrDest <= 4'b0011; ALUSel <= 2'b10; #30;
AddrSrc1 <= 4'b0011; AddrSrc2 <= 4'b0000; AddrDest <= 4'b0000; ALUSel <= 2'b00; #30;
extdata <= 4'b0100; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0000;
ALUSel <= 2'b00; #30;
extdata <= 4'b0010; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0011;
ALUSel <= 2'b01; #30;
isExternal <= 0; AddrDest <= 4'b0011; ALUSel <= 2'b10; #30;
AddrSrc1 <= 4'b0011; AddrSrc2 <= 4'b0000; AddrDest <= 4'b0000; ALUSel <= 2'b00; #30;
extdata <= 4'b0100; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0000;
ALUSel <= 2'b00; #30;
extdata <= 4'b0010; isExternal <= 1; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0011;
ALUSel <= 2'b01; #30;
isExternal <= 0; AddrDest <= 4'b0011; ALUSel <= 2'b10; #30;
AddrSrc1 <= 4'b0011; AddrSrc2 <= 4'b0000; AddrDest <= 4'b0000; ALUSel <= 2'b00; #30;
extdata <= 4'b0100; isExternal <= 0; AddrSrc1 <= 4'b0000; AddrSrc2 <= 4'b0011; AddrDest <= 4'b0011;
ALUSel <= 2'b11; #30;
end

endmodule
```