DIGITAL SYSTEM

FINAL LAB

CODE FOR 16 REGISTER EACH OF 8-BIT:

```
module regfile(clk,rst,in,i,opcode);
parameter n=8;//bit of reg
parameter num=16;//no. of reg
parameter i_bit=4;

input clk;
input rst;
input [n-1:0]in;
input [i_bit-1:0]i;//index of a reg
input [7:0]opcode;

reg[n-1:0]acc;
reg [n-1:0]Reg_arr[num-1:0];

integer k;
always @(posedge clk or negedge rst)
begin
   if (rst==0)
      begin
         for (k=0; k<=num-1; k=k+1)
            begin
            Reg_arr[k] = 0;
            end
      end
end
endmodule
```

CODE FOR PC:

```
module pc(
input wire [3:0]current,
input wire[3:0]branch,
input wire branch_flag,
output reg [3:0]next);

always@(*)
begin
if (branch_flag==1)
   next=branch;
else
   next=current+1;
end
endmodule
```

CODE FOR PROCESSOR:

```
module processor(clk);
parameter n=8;
parameter num=16;
input clk;

reg [7:0]acc;
reg [7:0]opcode;
reg [7:0]data;
reg [3:0]current,branch;
wire[3:0]next;
reg [n-1:0]Reg_arr[num-1:0];
reg [7:0]ext;
reg cb;
```

```
reg flag;
reg [n-1:0]ins_arr[7:0];
reg [3:0]temp;
reg flag2;
reg [7:0] acc_reg;
reg [7:0] data_reg;
reg [7:0] quotient_reg;
reg [7:0] remainder_reg;

pc a1(.current(current),.branch(branch),.branch_flag(flag),.next(next));

initial begin

ins_arr[0]=8'b10010000;ins_arr[1]=8'b00010000;ins_arr[2]=8'b00010011;
ins_arr[3]=8'b00100000;ins_arr[4]=8'b0010011;ins_arr[5]=8'b00110000;
ins_arr[6]=8'b01000000;ins_arr[7]=8'b11111111;
//ins_arr[8]=8'b00000100;
//ins_arr[9]=8'b00000101;ins_arr[10]=8'b01010000;ins_arr[11]=8'b01100000;
//ins_arr[12]=8'b01110000;ins_arr[13]=8'b00000110;ins_arr[14]=8'b00000111;
//ins_arr[15]=8'b10000000;ins_arr[16]=8'b10010000;ins_arr[17]=8'b10100110;
//ins_arr[18]=8'b00000111;

current=4'b0000;
flag=0;
temp=0;
ext=8'b00000000;
opcode=8'b00000000;
acc=8'b00000000;
cb=0;
Reg_arr[0]=8'd10; Reg_arr[1]=8'd10; Reg_arr[2]=8'd10;
Reg_arr[3]=8'd20;Reg_arr[4]=8'd20;Reg_arr[5]=8'd20;
```

```
Reg_arr[6]=8'd30;Reg_arr[7]=8'd30;Reg_arr[8]=8'd30;
Reg_arr[9]=8'd40;Reg_arr[10]=8'd40;Reg_arr[11]=8'd40;
Reg_arr[12]=8'd50;Reg_arr[13]=8'd50;Reg_arr[14]=8'd50;
Reg_arr[15]=8'd60;
end


always @(posedge clk)
   begin
   opcode=ins_arr[current];

   if (opcode==8'b11111111)
      begin
      end
   else
      begin
      flag=0;
      current=next;
      data=Reg_arr[opcode[3:0]];

      if (opcode[7:4]==4'b0001)
         {cb,acc} = acc+data;
      else if(opcode[7:4]==4'b0010)
         {cb,acc} = data-acc;
      else if(opcode[7:4]==4'b0011)
         {ext,acc}=acc*data;
      else if(opcode[7:4]==4'b0100)
         begin
         if(data!=0)
            begin
            acc_reg = acc;
```

```
        data_reg = data;
        quotient_reg = 0;
        ext = 0;
        while (acc_reg>=data_reg)begin
        acc_reg = acc_reg - data_reg;
        quotient_reg = quotient_reg + 1;
        end
        ext = acc_reg;
        acc=quotient_reg;
        end
    end
else if(opcode[7:4]==4'b0101)
    acc = acc&data;
else if(opcode[7:4]==4'b0110)
    acc = acc^data;
else if(opcode[7:4]==4'b0111)
    begin
    if(acc>=data)
        cb = 0;
    else
        cb = 1;
    end
else if(opcode[7:4]==4'b1001)
    acc=Reg_arr[opcode[3:0]][7:0];
else if(opcode[7:4]==4'b1010)
    Reg_arr[opcode[3:0]][7:0]=acc;

else if(opcode[7:4]==4'b1000)
        begin
        flag=1;
        temp=current;
```

```
        flag2=1;
        branch=opcode[3:0];

        end
else if(opcode[7:4]==4'b1011)
    if(flag2==1)
        begin
        flag=1;
        branch=temp;
        flag2=0;
        end
    else
        begin
        end

else if(opcode[7:4]==4'b1011)
        current=opcode[3:0];
else if(opcode[7:4]==4'b0000)
    begin
    if(opcode[3:0]==4'b0000)
        begin
        end
    else if(opcode[3:0]==4'b0001)
        acc=acc<<1;
    else if(opcode[3:0]==4'b0010)
        acc=acc>>1;
    else if(opcode[3:0]==4'b0011)
        acc = {acc[0], acc[7:1]};
    else if(opcode[3:0]==4'b0100)
        acc = {acc[6:0], acc[7]};
    else if(opcode[3:0]==4'b0101)
```

```
            acc = {acc[7], acc[7:1]};
        else if(opcode[3:0]==4'b0110)
            {cb, acc} = acc + 1;
        else if(opcode[3:0]==4'b0111)
            {cb, acc} = acc -1;

        end
      end
   end
endmodule
```

PROCESSOR CODE USED IN FPGA:

```
`timescale 1ns / 1ps

module clock_divider(
 input main_clk,
 output slow_clk
);
 reg [31:0] counter;
 always@ (posedge main_clk)
 begin
 counter <=counter+1;
 end
 assign slow_clk=counter[28];
endmodule

module pc(
input wire [3:0]current,
input wire[3:0]branch,
input wire mode,
```

```
output reg [3:0]next);

always@(*)
begin
if (mode==1)
   next=branch;
else
   next=current+1;
end
endmodule

module miniprocessor(clk,switch,switch2,out,out2);

parameter n=8;
parameter num=16;
input clk;
input [7:0]switch;
input switch2;
output reg [7:0]out,out2;

wire slow_clk;
clock_divider clk_divider_inst (.main_clk(clk),.slow_clk(slow_clk));

reg [n-1:0]acc;
reg [n-1:0]opcode;
reg [n-1:0]data;
reg [3:0]current,branch;
wire[3:0]next;
reg [n-1:0]Reg_arr[num-1:0];
reg [n-1:0]ext;
reg cb;
```

```
reg mode;
reg [n-1:0]ins_arr[8:0];
reg [3:0]temp;
reg mode2;
reg [n-1:0] acc_reg;
reg [n-1:0] data_reg;
reg [n-1:0] quotient_reg;
reg [n-1:0] remainder_reg;

pc a1(.current(current),.branch(branch),.mode(mode),.next(next));

initial begin
ins_arr[0]=8'b10010000;ins_arr[1]=8'b01100000;ins_arr[2]=8'b00010000;
ins_arr[3]=8'b00010011;ins_arr[4]=8'b00110000;ins_arr[5]=8'b00000110;
ins_arr[6]=8'b00000111;ins_arr[7]=8'b10010001;ins_arr[8]=8'b11111111;

current=4'b0000;
mode=0;
temp=0;
ext=8'b00000000;
opcode=8'b00000000;
acc=8'b00000000;
cb=0;
Reg_arr[0]=8'd10; Reg_arr[1]=8'd10; Reg_arr[2]=8'd10;
Reg_arr[3]=8'd10;Reg_arr[4]=8'd20;Reg_arr[5]=8'd20;
Reg_arr[6]=8'd30;Reg_arr[7]=8'd30;Reg_arr[8]=8'd30;
Reg_arr[9]=8'd40;Reg_arr[10]=8'd40;Reg_arr[11]=8'd40;
Reg_arr[12]=8'd50;Reg_arr[13]=8'd50;Reg_arr[14]=8'd50;
Reg_arr[15]=8'd60;
end
```

```
always @(posedge slow_clk)
   begin
   opcode=ins_arr[current];

   if (opcode==8'b11111111)
      begin
      end
   else
      begin
      mode=0;
      current=next;
      data=Reg_arr[opcode[3:0]];
      if (opcode[7:4]==4'b0001)
         {cb,acc} = acc+data;
      else if(opcode[7:4]==4'b0010)
         {cb,acc} = data-acc;
      else if(opcode[7:4]==4'b0011)
         {ext,acc}=acc*data;
      else if(opcode[7:4]==4'b0101)
         acc = acc&data;
      else if(opcode[7:4]==4'b0110)
         acc = acc^data;
      else if(opcode[7:4]==4'b0111)
         begin
         if(acc>=data)
            cb = 0;
         else
            cb = 1;
         end
      else if(opcode[7:4]==4'b1001)
         acc=Reg_arr[opcode[3:0]][7:0];
```

```
else if(opcode[7:4]==4'b1010)
   Reg_arr[opcode[3:0]][7:0]=acc;

else if(opcode[7:4]==4'b1000)
       begin
       mode=1;
       temp=current;
       mode2=1;
       branch=opcode[3:0];

       end
else if(opcode[7:4]==4'b1011)
   if(mode2==1)
       begin
       mode=1;
       branch=temp;
       mode2=0;
       end
   else
       begin
       end

else if(opcode[7:4]==4'b1011)
       current=opcode[3:0];
else if(opcode[7:4]==4'b0000)
   begin
   if(opcode[3:0]==4'b0000)
       begin
       end
   else if(opcode[3:0]==4'b0001)
       acc=acc<<1;
```

```
        else if(opcode[3:0]==4'b0010)
            acc=acc>>1;
        else if(opcode[3:0]==4'b0011)
            acc = {acc[0], acc[7:1]};
        else if(opcode[3:0]==4'b0100)
            acc = {acc[6:0], acc[7]};
        else if(opcode[3:0]==4'b0101)
            acc = {acc[7], acc[7:1]};
        else if(opcode[3:0]==4'b0110)
            {cb, acc} = acc + 1;
        else if(opcode[3:0]==4'b0111)
            {cb, acc} = acc -1;
        end
    end
if(switch[0]==1)
    out=Reg_arr[0];
else if(switch[1]==1)
    out=Reg_arr[1];
else if(switch[2]==1)
    out=Reg_arr[2];
else if(switch[3]==1)
    out=Reg_arr[3];
else if(switch[4]==1)
    out=Reg_arr[4];
else if(switch[5]==1)
    out=Reg_arr[5];
else if(switch[6]==1)
    out=Reg_arr[6];
else if(switch[7]==1)
    out=ext;
else
```

```
        out=ext;
    if (switch2==1)
        out2=acc;
    else
        out2=0;
    end
endmodule
```

TESTBENCH:

```
module tb();

    reg clk;
    processor uut(clk);

    initial begin
    clk = 0;
        forever #5 clk = ~clk;
    end

    initial begin
    #100 ;
    $finish;
    end

endmodule
```

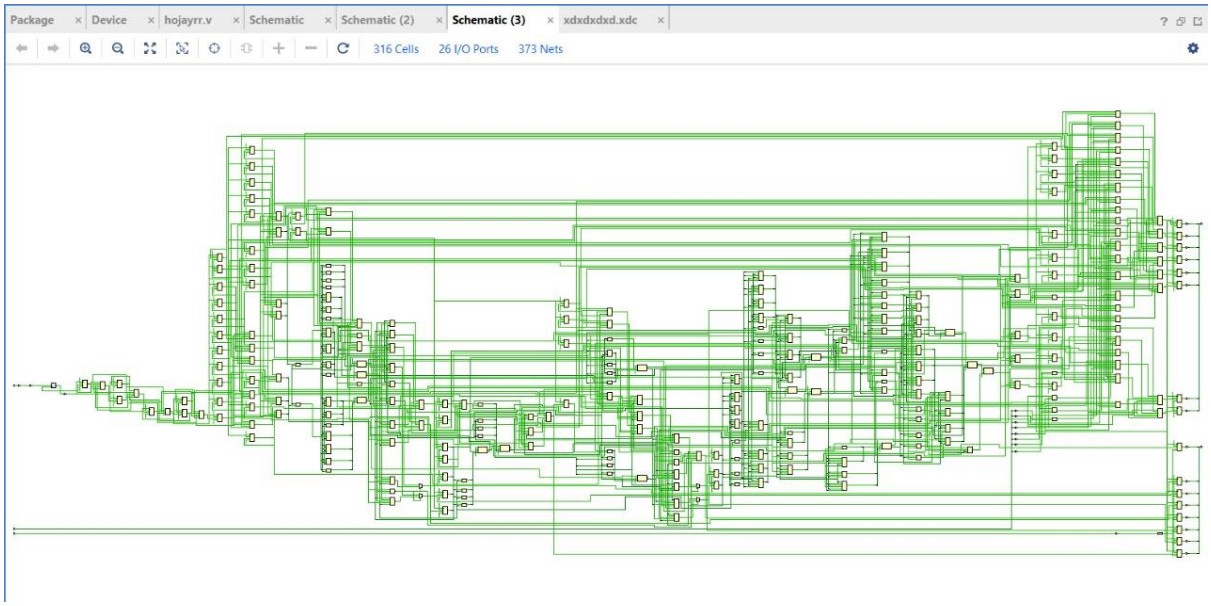
XDX FILE:

```
set_property PACKAGE_PIN U16 [get_ports {out[0]}]
set_property PACKAGE_PIN E19 [get_ports {out[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[7]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {out[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[0]}]
set_property PACKAGE_PIN U19 [get_ports {out[2]}]
set_property PACKAGE_PIN V19 [get_ports {out[3]}]
set_property PACKAGE_PIN W18 [get_ports {out[4]}]
set_property PACKAGE_PIN U15 [get_ports {out[5]}]
set_property PACKAGE_PIN U14 [get_ports {out[6]}]
set_property PACKAGE_PIN V14 [get_ports {out[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out2[0]}]
set_property PACKAGE_PIN V13 [get_ports {out2[0]}]
set_property PACKAGE_PIN V3 [get_ports {out2[1]}]
set_property PACKAGE_PIN W3 [get_ports {out2[2]}]
set_property PACKAGE_PIN U3 [get_ports {out2[3]}]
set_property PACKAGE_PIN P3 [get_ports {out2[4]}]
set_property PACKAGE_PIN N3 [get_ports {out2[5]}]
set_property PACKAGE_PIN P1 [get_ports {out2[6]}]
set_property PACKAGE_PIN L1 [get_ports {out2[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[6]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {switch[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {switch[0]}]
set_property PACKAGE_PIN V17 [get_ports {switch[0]}]
set_property PACKAGE_PIN V16 [get_ports {switch[1]}]
set_property PACKAGE_PIN W16 [get_ports {switch[2]}]
set_property PACKAGE_PIN W17 [get_ports {switch[3]}]
set_property PACKAGE_PIN W15 [get_ports {switch[4]}]
set_property PACKAGE_PIN V15 [get_ports {switch[5]}]
set_property PACKAGE_PIN W14 [get_ports {switch[6]}]
set_property PACKAGE_PIN W13 [get_ports {switch[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports switch2]
set_property IOSTANDARD LVCMOS33 [get_ports switch3]
set_property PACKAGE_PIN W5 [get_ports clk]
set_property PACKAGE_PIN R2 [get_ports switch2]
set_property PACKAGE_PIN T1 [get_ports switch3]
```

BLOCK DIAGRAM:



SIMULATION RESULT: