EE513 LAB

Experiment 8

Processor Design.

Submitted by,
Patel Ankur
Roll No-224102404
October 16, 2022

Contents

1 AIM

Design an HDL-based 32-bit processor (Instruction decode (Instruction decoder + controller) + Instruction execute (ALU)+register write (Register Bank)) which executes the following 10 instructions:
ADD, SUB, SLL, SLT, SLTU, XOR, SRL, SRA, OR, AND

2 OBJECTIVE

Perform post-synthesis simulations for a basic 32-bit processor which executes the aforementioned instructions. The processor will basically have the following modules:
a. A 32-bit register bank which will have 32 registers and each register can store 32-bit data.
b. An instruction decoder module which will slice the 32-bit instructions into the corresponding fieldsopcode, func and rs1, rs2 and rd values
c. A controller module which will generate a corresponding signal for the respective operations depending func and opcode.
d. 32-bit ALU for processing the data present in rs1 and rs2 registers.
e. Register write to store the result back into rd register in the register bank.

3 Theory

The purpose of theCPUis to processdata. The CPU is where processes such as calculating, sorting and searching take place. Whatever is done on our computers, such as checking emails, playing games and doing homework, the CPU has processed the data we use.The CPU is made up of three main components, the control unit, the immediate access store and the arithmetic and logic unit.
The control unit:
The control unit controls the flow of data within the system.The control unit controls and monitors communications between the hardware attached to the computer. It controls the input and output of data, checks that signals have been delivered successfully, and makes sure that data goes to the correct place at the correct time.
Immediate access store:
The immediate access store is where the CPU holds all the data and programs that it is currently using. You can think of it like the numbers typed into a calculator – they are being stored inside the calculator while it processes the calculations. The immediate access store is often referred to as the registers in the CPU.
Arithmetic and logic unit:
The arithmetic and logic unit (ALU) is where the CPU performs the arithmetic and logic operations. Every task that your computer carries out is completed here. Even typing into a word processor involves adding binary digits to the file, and then calculating which pixels on the screen should change so that you can see the characters.
RegisterBank:
RegisterBank contain alot of General Purpose Register which is going to be use in processing of ALU.

4 Verilog Code

```
‘timescale 1ns / 1ps
module processor(input [31:0] inst, output [31:0] out,input clock);

 wire [4:0] regA_num, regB_num, destreg_num;
 wire [3:0] ctrl;
 wire [31:0] regA, regB,ID_EX_IR,ALUout;

 control CTL (ID_EX_IR,ctrl,inst,regA_num,regB_num,destreg_num,clock);
 RegBank regbank(.rd1(regA),.rd2(regB),.sr1(regA_num),.sr2(regB_num),
 .wr(ALUout),.dr(destreg_num),.clock(clock));
 ALU ALU (regA, regB, ctrl, ALUout);

 assign out = ALUout;
endmodule

module RegBank(rd1,rd2,wr,sr1,sr2,dr,clock);
 input [4:0]sr1,sr2,dr;
 input [31:0] wr;
 input clock;
 output reg [31:0] rd1,rd2;
 reg [31:0] regfile[0:31];

 always @(*)
begin
   rd1 <= regfile[sr1];
   rd2 <= regfile[sr2];
   regfile[dr] <= wr;
 end
endmodule

module control(output reg [31:0] ID_EX_IR, output reg [3:0] ctrl,
input [31:0] inst,output reg [4:0] regA_num,reg [4:0] regB_num,reg [4:0] de-
streg_num,input clock);
```

```
     always@(*)
  begin
   ID_EX_IR <= inst ;
    regA_num <= inst[19:15]; //rs1
    regB_num <= inst[24:20]; //rs2
    destreg_num <= inst[11:7]; //rd
  end

  parameter ADD = 10'b0000001, SUB = 10'b0010001, SLL = 10'b0000011,
  SRL = 10'b0010011, SRA = 10'b0100011, SLT = 10'b0000111,
  SLTU= 10'b0010111, XOR = 10'b0001111, OR = 10'b0011111,
  AND = 10'b0101111; //func_opcode

  always@(posedge clock)
begin
   case(inst[14:12],inst[3:0]) //func_opcode
    ADD : ctrl = 4'h1;
    SUB : ctrl = 4'h2;
    SLL : ctrl = 4'h3;
    SRL : ctrl = 4'h4;
    SRA : ctrl = 4'h5;
    SLT : ctrl = 4'h6;
    SLTU: ctrl = 4'h7;
    XOR : ctrl = 4'h8;
    OR : ctrl = 4'h9;
    AND : ctrl = 4'ha;
    default : ctrl = 4'h0;
   endcase
  end
endmodule

module ALU(input [31:0] regA, input [31:0] regB, input [3:0] ctrl,
output reg [31:0] ALUout);

  always@(*)
begin
  case(ctrl) //func_opcode
   4'h1 : ALUout = regA + regB ;
```

4'h2 : ALUout = regA - regB ;
4'h3 : ALUout = regA << regB[4:0] ;
4'h4 : ALUout = regA >> regB[4:0] ;
$4'h5 : ALUout$ =signed(regA) >>> regB[4:0];
4'h6 : ALUout = ($signed(regA)$ <signed(regB)) ?$32'h00000001 : 32'h00000000$;
4'h7 : ALUout = (regA < regB) ?$32'h00000001 : 32'h00000000$;
$4'h8 : ALUout = regA \wedge regB$;
$4'h9 : ALUout = regA \mid regB$;
$4'ha : ALUout = regA\&regB$;
$default : ALUout = 32'hxxxxxxxx$;
endcase
end
endmodule

4.1 Testbench

```
‘timescale 1ns / 1ps
module testbench();
 reg [31:0] inst;
 reg clock;
 wire [31:0] out;
 integer k;

 initial begin
 clock = 0;
 for(k=0;k<32;k=k+1)
  DUT.regbank.regfile[k] = 32’h00000000;
 end
 always 10 clock =  clock;
 processor DUT(inst,out,clock);

 initial begin
  DUT.regbank.regfile[1] = 32’h0000000F;
  DUT.regbank.regfile[2] = 32’h0000000C;
  DUT.regbank.regfile[3] = 32’hFF0000FF;
  DUT.regbank.regfile[4] = 32’h70000000;
  DUT.regbank.regfile[5] = 32’hF0000000;
  DUT.regbank.regfile[6] = 32’h00000004;

  #20 inst = 7’h00,5’h02,5’h01,3’h00,5’h07,7’h01;//ADD
  #20 inst = 7’h00,5’h02,5’h01,3’h01,5’h08,7’h01;//SUB
  #20 inst = 7’h00,5’h06,5’h03,3’h00,5’h09,7’h03;//SLL
  #20 inst = 7’h00,5’h06,5’h03,3’h01,5’h0a,7’h03;//SRL
  #20 inst = 7’h00,5’h06,5’h03,3’h02,5’h0b,7’h03;//SRA
  #20 inst = 7’h00,5’h01,5’h04,3’h00,5’h0c,7’h07;//SLT
  #20 inst = 7’h00,5’h04,5’h01,3’h01,5’h0d,7’h07;//SLTU
  #20 inst = 7’h00,5’h02,5’h01,3’h00,5’h0e,7’h0f;//XOR
  #20 inst = 7’h00,5’h02,5’h01,3’h01,5’h0f,7’h0f;//OR
  #20 inst = 7’h00,5’h02,5’h01,3’h02,5’h10,7’h0f;//AND
  #40 $finish;
 end
endmodule
```

4.2 RTL Schematic and Functionality of Sequence Detector

4.2.1 RTL Schematic

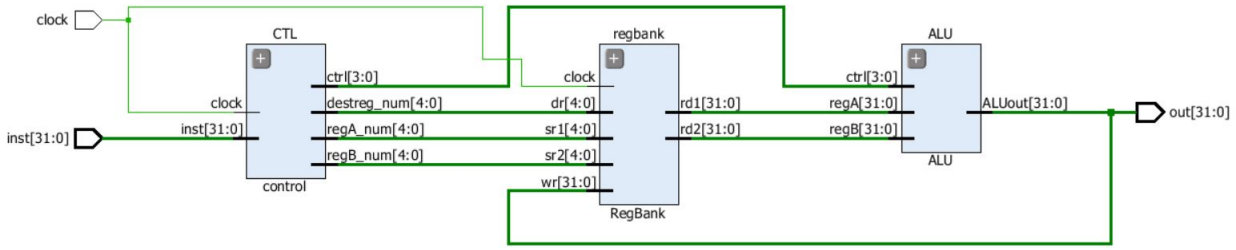

Figure 1

4.2.2 Functionality

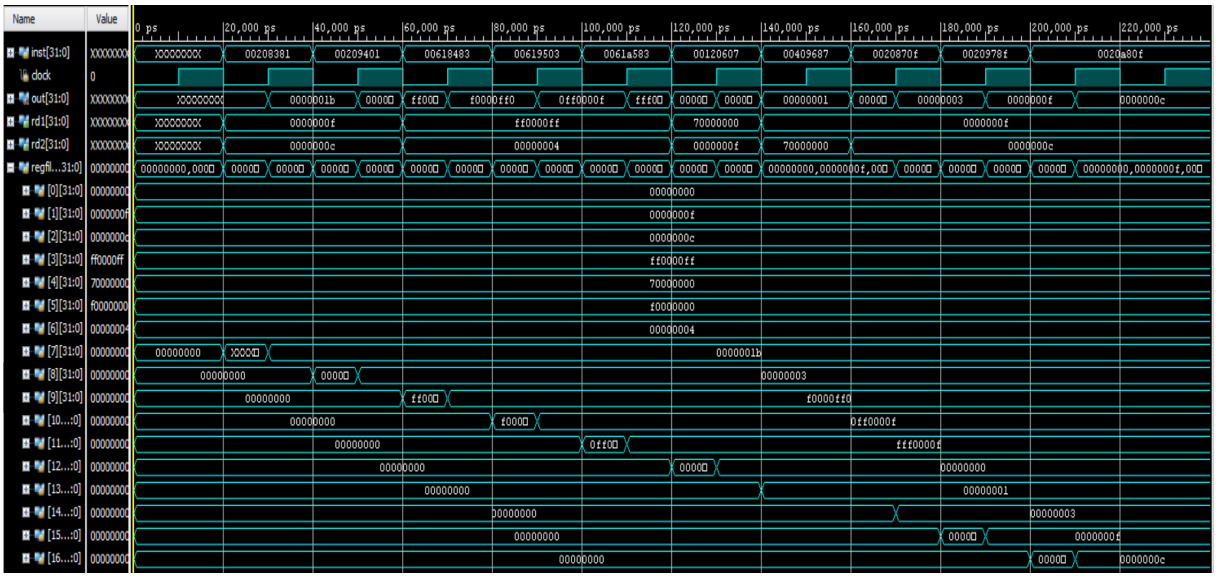

Figure 2

As shown in the waveform for every positive clock cycle output getting store in Registerbank that are getting generated from the Arithmetic Logic Unit of processor.

5 Observation

LUTs	Flip-Flops	Power(W)	Delay(ns)
386	4	0.176	1.503

6 Results

I have verify the functionality of a processor and figure out the number of LUTs, number of flip flops, Power and Delay required in the corresponding designs.