CA Lab: Lab 12

student: Dimitri Tabatadze

June 18, 2023

Task Description

We have already written all ofthe components of MIPS Processor:Branch control evaluation unit, general purpose registers, instruction decoder, ALU MEMORY, shifter, signal extension. Now it is time to combine all of our modules and obtain MIPS Processor.

Solution

The code for the MIPS Processor:

```
module Processor (
    input Clock,
    input Reset,
    output reg [31:0] aluresout, shift_resultout, GP_DATA_INout
);

    wire [31:0] PC, I, data_in, data_out;
    wire [31:0] Instruction, SrcA, SrcB;
    wire [31:0] Mout;
    wire [31:0] alu_result, shift_result, immediate_out;
    wire [31:0] data_out_A, data_out_B;
    wire [3:0] af, bf;
    wire i, ALU_MUX_SEL, GP_WE_Org, DM_WE, bcres;
    wire [2:0] shift_type;
    wire [1:0] GP_MUX_SEL, PC_MUX_Select;
    wire [4:0] Cad;
  wire E;
    wire [31:0] next_SrcA;
    reg [31:0] next_PC;
  reg [31:0] next_SrcB;
    reg GP_WE;

    ID ID (
        .Instruction(I),
        .I(i),
        .ALU_MUX_SEL(ALU_MUX_SEL),
        .GP_WE(GP_WE_Org),
        .DM_WE(DM_WE),
        .Af(af),
        .Bf(bf),
```

```
        .Shift_type(shift_type),
        .GP_MUX_SEL(GP_MUX_SEL),
        .PC_MUX_Select(PC_MUX_Select),
        .Cad(Cad)
    );

    ALU ALU (
        .SrcA(next_SrcA),
        .SrcB(next_SrcB),
        .af(af),
        .i(i),
        .Alures(alu_result)
    );

    IEU IEU (
        .U(i),
        .imm(data_in[15:0]),
        .res(immediate_out)
    );

    BCE BCE (
        .SrcA(SrcA),
        .SrcB(SrcB),
        .BF(bf),
        .bcres(bcres)
    );

    Shifter S (
        .Funct(shift_type[1:0]),
        .Number(SrcA),
        .Shift_Amount(SrcB[4:0]),
        .Result(shift_result)
    );

    GPR GP (
        .clk(Clock),
        .WE(GP_WE),
        .addrA(Instruction[25:21]),
        .addrB(Instruction[20:16]),
        .addrC(Cad),
        .data_in_C(GP_DATA_INout),
        .data_out_A(data_out_A),
        .data_out_B(data_out_B)
    );

    Memory M (
        .Clock(Clock),
        .Reset(Reset),
        .S(DM_WE),
        .Next_PC(next_PC),
        .data_addr_in(GP_DATA_INout),
        .data_in(data_out_A),
        .PC_out(PC),
        .Iout(I),
        .Mout(Mout),
    .E(E)
    );
```

```
88
89      always @* begin
90          case(ALU_MUX_SEL)
91              1'b0: next_SrcB = data_out_B;
92              1'b1: next_SrcB = immediate_out;
93          endcase
94      end
95
96      always @* begin
97          case(GP_MUX_SEL)
98              2'b00: GP_DATA_INout = alu_result;
99              2'b01: GP_DATA_INout = shift_result;
100             2'b10: GP_DATA_INout = Mout;
101             default: GP_DATA_INout = next_PC;
102         endcase
103     end
104
105     assign next_SrcA = data_out_A;
106
107     always @(posedge Clock) begin
108         if (Reset) begin
109             next_PC <= 32'b0;
110             GP_WE <= 1'b0;
111         end else begin
112             GP_WE <= GP_WE_Org & E;
113             case(PC_MUX_Select)
114                 2'b00: next_PC <= PC + 4;
115                 2'b01: next_PC <= PC + (immediate_out << 2);
116                 2'b10: next_PC <= data_out_A;
117                 2'b11: next_PC <= (bcres) ? PC + (immediate_out << 2) :
     PC + 4;
118             endcase
119         end
120     end
121 endmodule
```

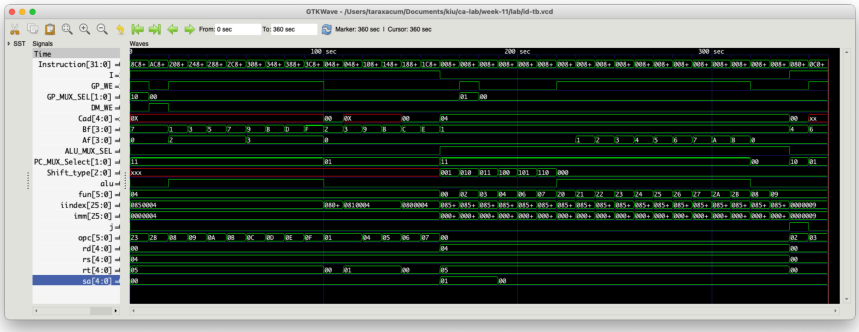
Here, I used all the modules necessary and cobined them into one Processor (Top Level) module.

Simulation and Testing

This is the testbench:

```
1  module testbench();
2
3    reg clock = 0;
4    reg reset = 0;
5    wire [31:0] aluresout, shift_resultout, GP_DATA_INout;
6
7    Processor UUT (
8      .Clock(clock),
9      .Reset(reset),
10     .aluresout(aluresout),
11     .shift_resultout(shift_resultout),
12     .GP_DATA_INout(GP_DATA_INout)
13   );
```

```
14
15    always #10 clock = ~clock;
16
17    initial begin
18      $dumpfile("testbench.vcd");
19      $dumpvars(0, testbench);
20      #10;
21      reset = 1;
22      #10;
23      reset = 0;
24      #100;
25      $finish;
26    end
27
28  endmodule
```



Reference

- me