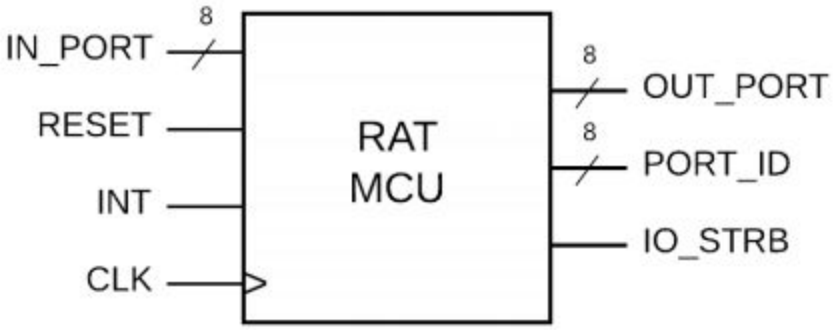
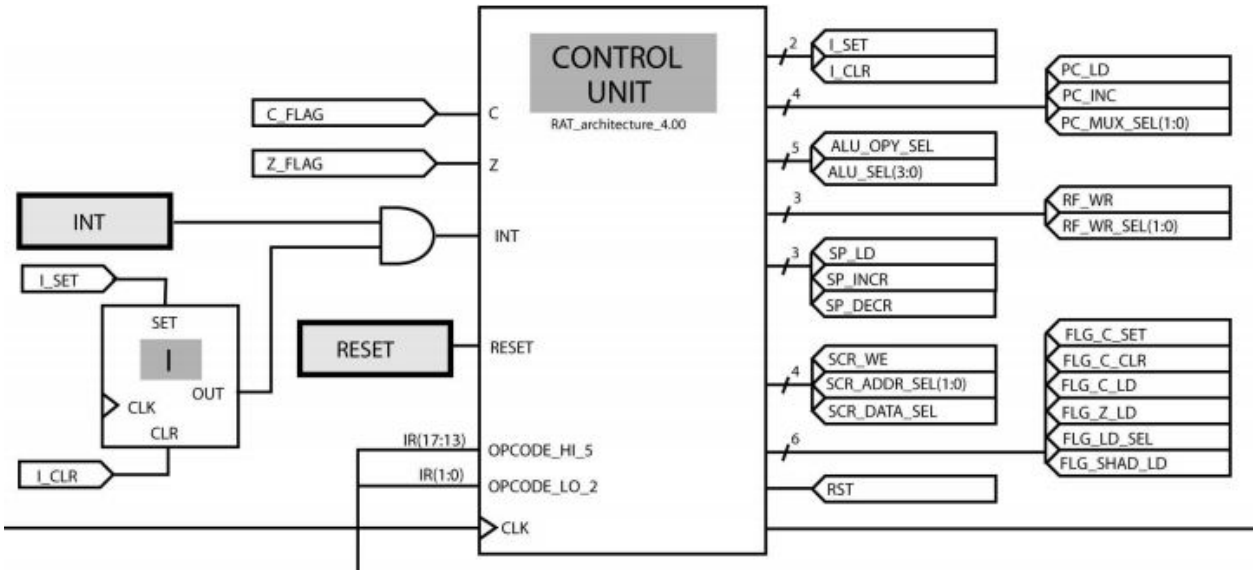
Black Body Diagram:

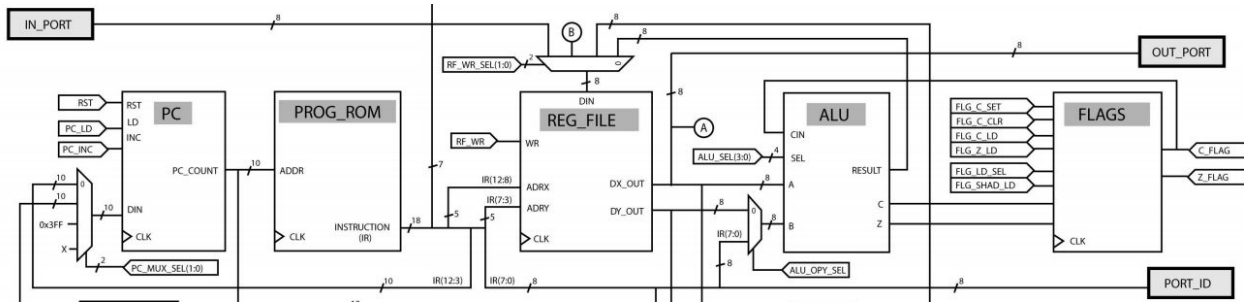


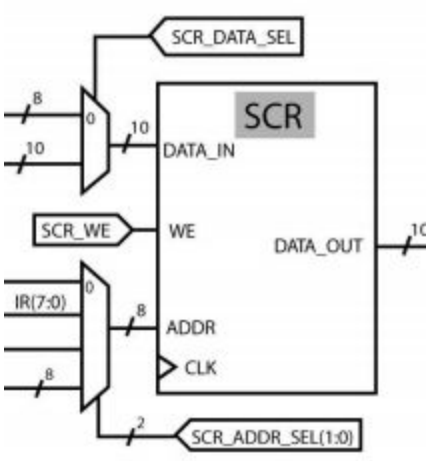
Black Body of Inputs and Outputs

More in depth look at the components in the above

Behavioral Description:





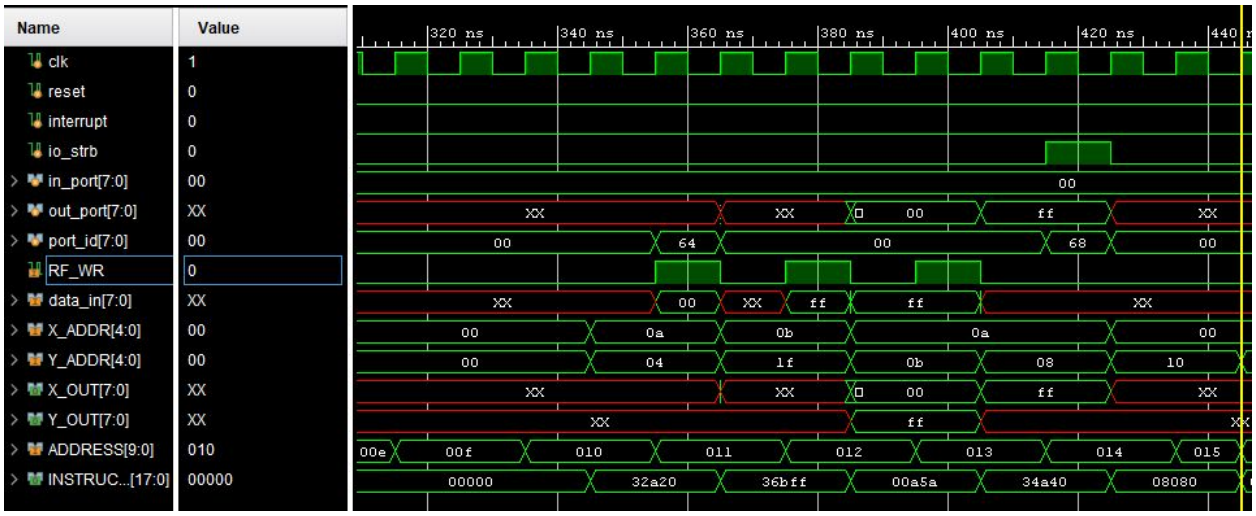


Behavioral Description:

In this assignment, our team created c/z flag register, implemented a control unit, and created the RAT MCU by implementing all past projects with the control unit. Building all of these components together created the RAT MCU. The control unit telling the ALU, scratch, and register file when to operate, the operations set the flags, and the entire unit together as stated before made up the RAT MCU.

Structural Design:

Verification:



It can be seen from the above that 0xff is being outputted to a specific address on a certain command(the “out” command). All other commands from the sample program can also be seen. As the program runs, it can be seen that the register file is reading and writing, the addresses are being set, the outputs of the register file, the port outputs, and the port id.

Source Code:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/12/2019 01:35:38 PM
// Design Name:
// Module Name: realMain
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module realMain(clk, reset, in_port, interrupt, out_port, port_id, io_strb);

    input interrupt, clk, reset;
    input [7:0] in_port;

    output io_strb;
    output [7:0] port_id, out_port;

    wire [1:0] pc_mux_sel, rf_wr_sel;
    wire pc_reset, sc_reset, pc_inc, alu_opy_sel, flg_c_ld, flg_z_ld, pc_ld;
    wire [3:0] alu_sel;

    wire set_c, set_z, c_out;
    wire [17:0] INSTRUCTION;
    wire [9:0] scr_out, count;
    wire [7:0] regInput, alu_scr_input, scr_addr, alu_result, opy_out;
    wire rf_wr, scr_we;
```

```
assign out_port = alu_scr_input;

realControl mycontrol(
    .clk        (clk),
    .reset      (reset),
    .in_port    (in_port),
    .port_id    (port_id),
    .ophi       (INSTRUCTION[17:13]),
    .oplo       (INSTRUCTION[1:0]),
    .pc_reset   (pc_reset),
    .sc_reset   (sc_reset),
    .pc_inc     (pc_inc),
    .alu_opy_sel(alu_opy_sel),
    .flg_c_ld   (flg_c_ld),
    .flg_z_ld   (flg_z_ld),
    .io_strb    (io_strb),
    .pc_ld      (pc_ld),
    .rf_wr      (rf_wr),
    .scr_we     (scr_we),
    .pc_mux_sel (pc_mux_sel),
    .rf_wr_sel  (rf_wr_sel),
    .alu_sel    (alu_sel)
);

PC myPC(
    .D0     (INSTRUCTION[12:3]),
    .D1     (scr_out),
    .clk    (clk),
    .ld     (pc_ld),
    .up     (pc_inc),
    .SEL    (pc_mux_sel),
    .clr    (pc_reset),
    .rco    (),
    .count  (count)
);

Scratch myScratch(
    .data_in    ({2'b00,alu_scr_input}),
    .SCR_ADDR   (scr_addr),
    .SCR_WE     (scr_we),
    .clk        (clk),
    .data_out   (scr_out)
);
```

```
prog_rom DUT(
        .ADDRESS        (count),
        .INSTRUCTION    (INSTRUCTION),
        .CLK            (clk));

mux_4 regFileMux(
    .SEL    (rf_wr_sel),
    .D0     (alu_result),
    .D1     (scr_out[7:0]),
    .D2     (scr_addr),
    .D3     (in_port),
    .D_OUT  (regInput)
);

regFile myRegFile(
    .data_in    (regInput),
    .X_ADDR     (INSTRUCTION[12:8]),
    .Y_ADDR     (INSTRUCTION[7:3]),
    .RF_WR      (rf_wr),
    .clk        (clk),
    .X_OUT      (alu_scr_input),
    .Y_OUT      (scr_addr)
);

mux_2 alu_sel_MUX(
    .SEL    (alu_opy_sel),
    .D0     (scr_addr),
    .D1     (INSTRUCTION[7:0]),
    .D_OUT  (opy_out));

realALU myalu(
    .A      (alu_scr_input),
    .B      (opy_out),
    .Cin    (c_out),
    .SEL    (alu_sel),
    .result (alu_result),
    .c      (set_c),
    .z      (set_z));

flagReg c_flag(
    .clk    (clk),
    .RF_WR  (rf_wr),
```

```
        .data_in (set_c),
        .flag    (c_out)
    );

endmodule



`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/12/2019 01:38:58 PM
// Design Name:
// Module Name: realControl
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module realControl(clk, reset, in_port, port_id, ophi, oplo,
pc_reset, sc_reset, pc_inc, alu_opy_sel, flg_c_ld, flg_z_ld, io_strb, pc_ld, rf_wr,
 pc_mux_sel, rf_wr_sel, alu_sel, scr_we);

    input clk, reset;
    input [7:0] in_port;
    input [4:0] ophi;
    input [1:0] oplo;

    output logic [7:0] port_id;

    parameter RESET = 2'b00;
    parameter FETCH = 2'b01;
```

```
    parameter EXECUTE = 2'b10;

    logic [1:0] ps, ns;
    output logic [1:0] pc_mux_sel, rf_wr_sel;
    output logic pc_reset, sc_reset, pc_inc, alu_opy_sel, flg_c_ld, flg_z_ld, io_strb, pc_ld, rf_wr,
scr_we;
    output logic [3:0] alu_sel;

    always_ff @(posedge clk)
        if(reset)
            ps <= RESET;
        else
            ps <= ns;

    always_comb begin
        pc_reset <= 1'b0;
        sc_reset <= 'b0;
        pc_inc <= 'b0;
        alu_opy_sel <= 'b0;
        flg_c_ld <= 'b0;
        flg_z_ld <= 'b0;
        io_strb <= 'b0;
        pc_ld <= 'b0;
        rf_wr <= 'b0;
        scr_we <= 'b0;
        alu_sel <= 'b0;
        rf_wr_sel <= 2'b00;
        alu_sel <= 4'b0000;
        pc_mux_sel <= 'b0;
        port_id <= 'b0;

        case(ps)
            RESET: begin
                pc_reset <= 1'b1;
                sc_reset <= 1'b1;
                ns <= FETCH;
            end

            FETCH: begin
                pc_inc <= 1'b1;
                ns <= EXECUTE;
            end
```

```
EXECUTE: begin
   ns <= FETCH;
   pc_inc <= 1'b0;

   case(ophi[4])
      1'b0: begin
         case({ophi, oplo})
            7'b0010000: begin //BRN
               pc_ld <= 1'b1;
            end

            7'b0000010: begin //exor 2 registers
               rf_wr <= 1'b1;
               rf_wr_sel <= 2'b00;
               alu_sel <= 4'b0111;
               alu_opy_sel <= 1'b0;
               flg_c_ld <= 1'b1;
               flg_z_ld <= 1'b1;
            end

            7'b0001001: begin //mov with regs
               rf_wr <= 1'b1;
               rf_wr_sel <= 2'b00;
               alu_sel <= 4'b1110;
               alu_opy_sel <= 1'b0;
               flg_c_ld <= 1'b0;
               flg_z_ld <= 1'b0;
            end

            default: ns <= FETCH;
         endcase
      end

      1'b1:
         case(ophi)
            5'b10010: begin //exor with an immediate
               rf_wr <= 1'b1;
               rf_wr_sel <= 2'b00;
               alu_sel <= 4'b0111;
               alu_opy_sel <= 1'b1;
               flg_c_ld <= 1'b1;
               flg_z_ld <= 1'b1;
            end
```

```
                5'b11001: begin //in
                    rf_wr <= 1'b1;
                    rf_wr_sel <= 2'b11;
                    alu_sel <= 4'b0000;
                    alu_opy_sel <= 1'b0;
                    flg_c_ld <= 1'b0;
                    flg_z_ld <= 1'b0;
                    port_id <= {ophi, oplo};
                end

                5'b11011: begin //mov with immediate
                    rf_wr <= 1'b1;
                    rf_wr_sel <= 2'b00;
                    alu_sel <= 4'b1110;
                    alu_opy_sel <= 1'b1;
                    flg_c_ld <= 1'b0;
                    flg_z_ld <= 1'b0;
                end

                5'b11010: begin //out
                    io_strb <= 1'b1;
                    port_id <= {ophi, oplo};
                end

                default: ns <= FETCH;
            endcase

        default: ns <= FETCH;
    endcase
  end
endcase
end

endmodule
```

-