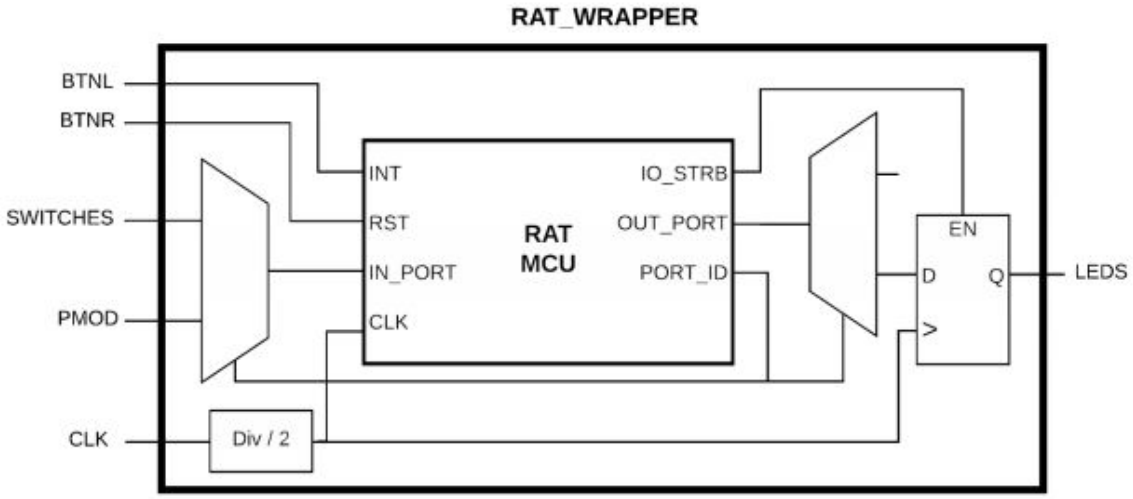
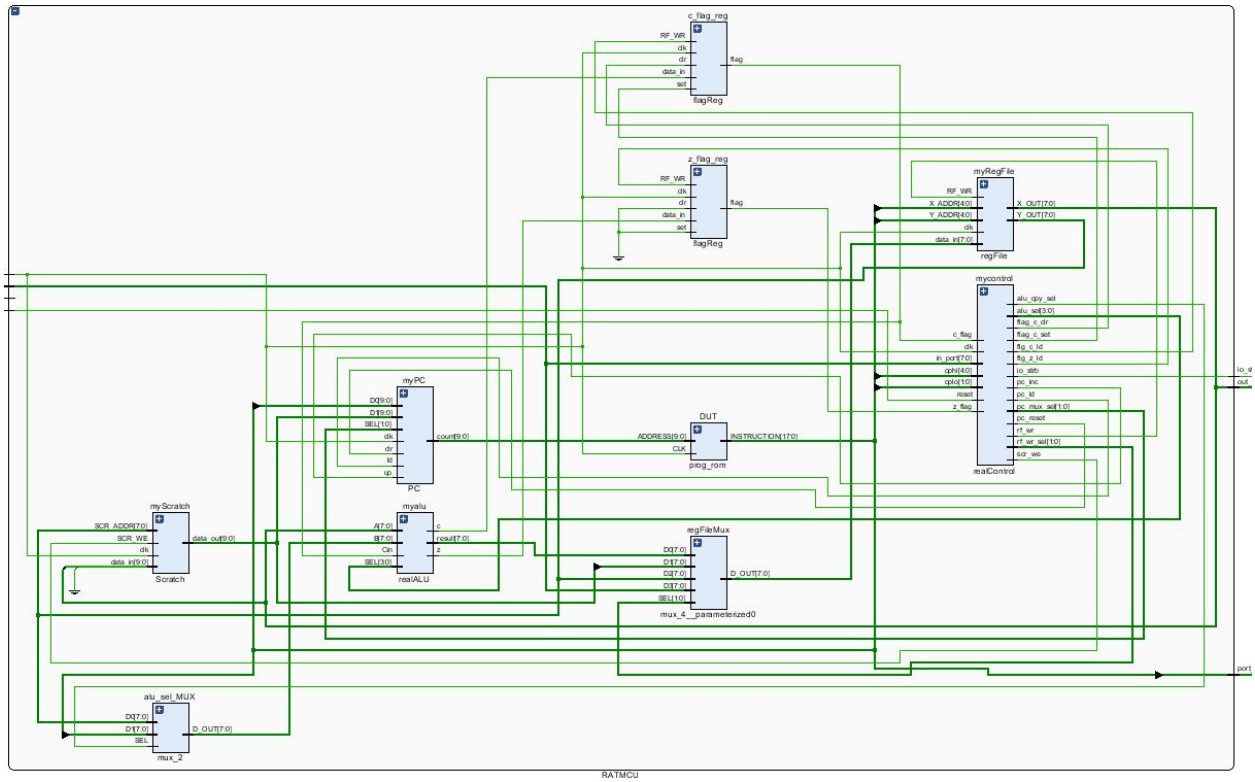
Black Box Diagram:



Behavioral Description:

The RAT_WRAPPER connects the RAT_MCU with the specific modules that interact with the Basys3 development board. Input and output MUXs will connect the IN_PORT and OUT_PORT to a variety of inputs and outputs. Every output will also have a register to keep the output signals constant until specifically changed.

Structural Design:



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:
//
//////////////////////////////////////////////////////////////////////////////////
```

```
//RATMCU MCU (.IN_PORT(s_input_port), .OUT_PORT(s_output_port),
//                .PORT_ID(s_port_id), .IO_STRB(s_load),
.RESET(s_reset),
//                .INTR(s_interrupt), .CLK(s_clk_50));


module RATMCU(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, pc_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, z_flag, flag_c_set, flag_c_clr;

    assign out_port = alu_scr_input;
    assign port_id = INSTRUCTION[7:0];

    realControl mycontrol(
        .c_flag     (c_out),
        .z_flag     (z_flag),
        .clk        (clk),
        .reset      (reset),
        .in_port    (in_port),
        .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),
    .flag_c_set (flag_c_set),
    .flag_c_clr (flag_c_clr),
    .pc_sel     (pc_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_reg(
    .clk     (clk),
    .RF_WR   (flg_c_ld),
    .data_in (set_c),
    .flag    (c_out),
    .clr     (flag_c_clr),
    .set     (flag_c_set)
);

flagReg z_flag_reg(
        .clk     (clk),
        .RF_WR   (flg_z_ld),
        .data_in (set_z),
```

```
            .flag    (z_flag),
            .clr     (0),
            .set        (0)
        );



endmodule



`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02/05/2019 01:08:59 PM
// Design Name:
// Module Name: realMain
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module realALU(A, B, Cin, SEL, result, c, z);

    input [7:0] A, B;
    input [3:0] SEL;
    input Cin;

    output logic [7:0] result;
    output logic c, z;
```

```
logic [8:0] temp_result;

always_comb begin

    result = 8'b00000000;
    temp_result = 8'b00000000;
    c = 0;
    z = 0;

    case(SEL)

        'b0: begin //Add
            temp_result = A + B;
            result = temp_result[7:0];
            c = temp_result[8];

            if (result == 'b0)
                z = 'b1;
        end

        'b0001: begin //Addc
            temp_result = A + B + Cin;
            result = temp_result[7:0];
            c = temp_result[8];

            if (result == 'b0)
                z = 'b1;
        end

        'b0010: begin //Sub
            temp_result = {1'b1, A} - B;
            result = temp_result[7:0];
            c = ~temp_result[8];

            if (result == 'b0)
                z = 'b1;
        end

        'b0011: begin //Subc
            temp_result = {1'b1, A} - B - Cin;
            result = temp_result[7:0];
            c = temp_result[8];
```

```
    if (result == 'b0)
        z = 'b1;
end

'b0100: begin //compare
    temp_result = {1'b1, A} - B;
    result = temp_result[7:0];
    c = ~temp_result[8];

    if (result == 'b0)
        z = 'b1;
end

'b0101: begin //And
    result = A & B;

    if (result == 'b0)
        z = 'b1;
end

'b0110: begin //or
    result = A | B;

    if (result == 'b0)
        z = 'b1;
end

'b0111: begin //exor
    result = A ^ B;

    if (result == 'b0)
        z = 'b1;
end

'b1000: begin //test
    temp_result = A & B;

    if (result == 'b0)
        z = 'b1;
end

'b1001: begin //lsl
```

```
        result = {A[6:0], Cin};
        c = A[7];

        if (result == 'b0)
            z = 'b1;
    end

    'b1010: begin //lsr
        result  = {Cin, A[7:1]};
        c = A[0];

        if (result == 'b0)
            z = 'b1;
    end

    'b1011: begin //rol
        c = A[7];
        result = {A[6:0], A[7]};

        if (result == 'b0)
            z = 'b1;
    end

    'b1100: begin //ror
        c = A[0];
        result = {A[0], A[7:1]};

        if (result == 'b0)
            z = 'b1;
    end

    'b1101: begin //asr
        c = A[0];
        result = {A[7], A[7:1]};

        if (result == 'b0)
            z = 'b1;
    end

    'b1110: begin //mov
        result = B;
    end
```

```
            'b1111: begin //unused

            end


        endcase


    end

endmodule
```