CPE 233: RAT assignment 5, MCU

Prof. Bridget Benson

Stan Carpenco, Luis Gomez

BBD

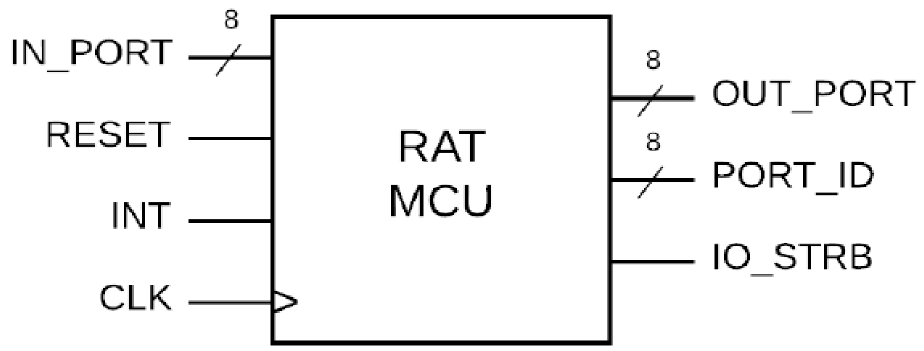
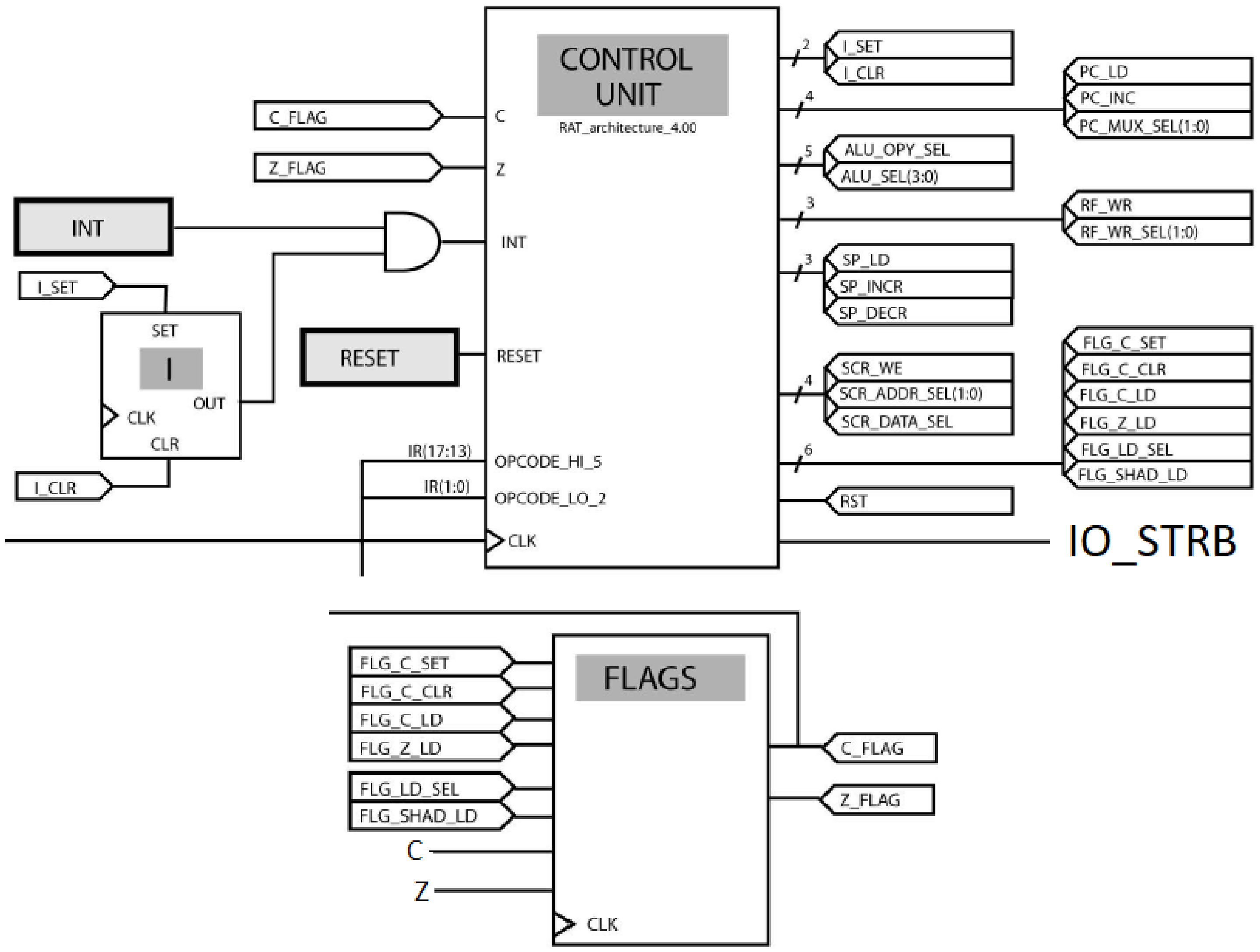


Figure 1: RAT MCU Black Box Diagram



Behavior

In this assignment, we built the Micro-Controller (MCU), Control-Unit (CU), and Flags modules in System Verilog. The MCU was assembled using the CU, Flags, PC, Prog Rom, Reg File, SCR, and ALU modules from previous Rat assignments.

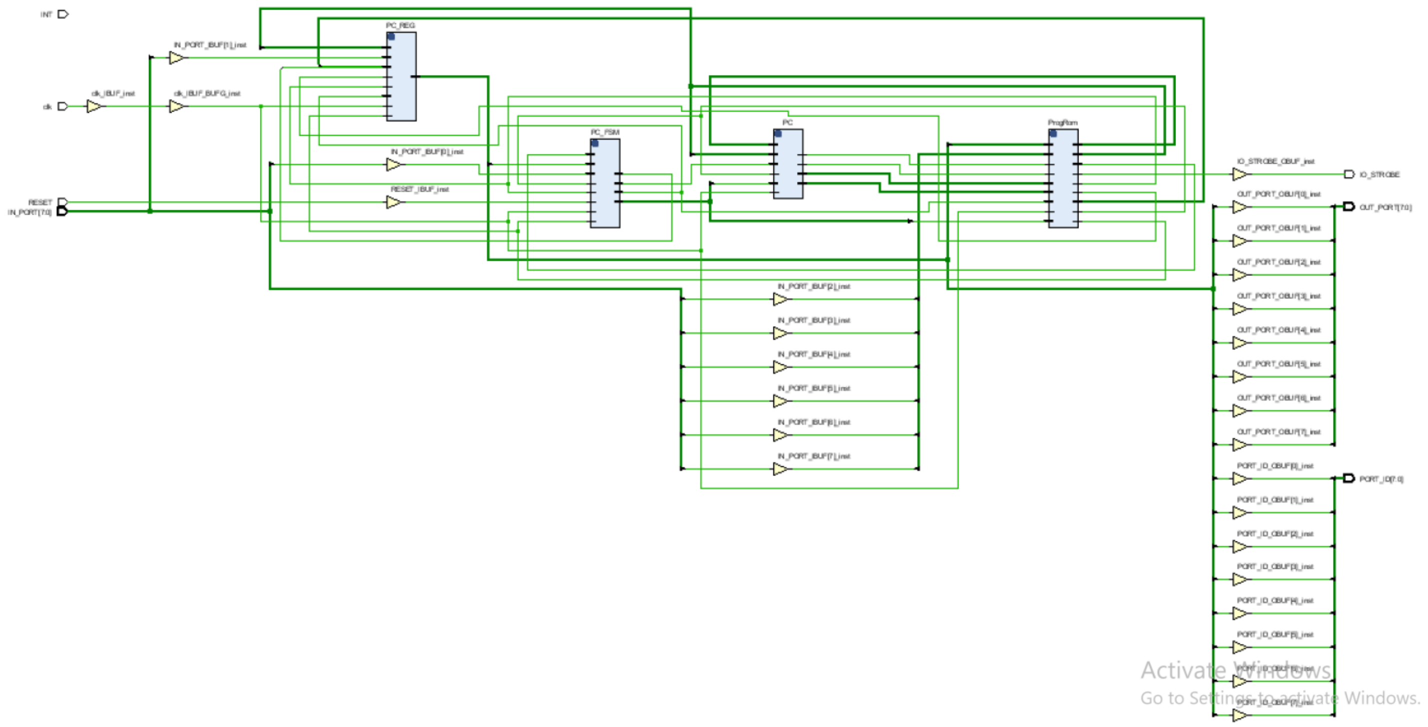
Micro-Controller (MCU): The MCU is a sequential circuit that serves as an interface between the many modules of the RAT computer and the outside world, via the RAT Assembly language. Between the many modules are a variety of combinatorial circuits, namely Muxes, which use incoming flag signals from the CU to control inputs to various modules.

Control-Unit (CU): The CU is an FSM whose behavior is determined by its state (INIT, FETCH, or EXEC) and the input values generated by the binary op-codes of Assembly instructions. Using the 5 MSBs and 2 LSBs of an instruction, the CU toggles a host of output Flag signals. These output flag signals ripple through the RAT computer modules, directing these modules to execute the instructions set out by the op-codes.

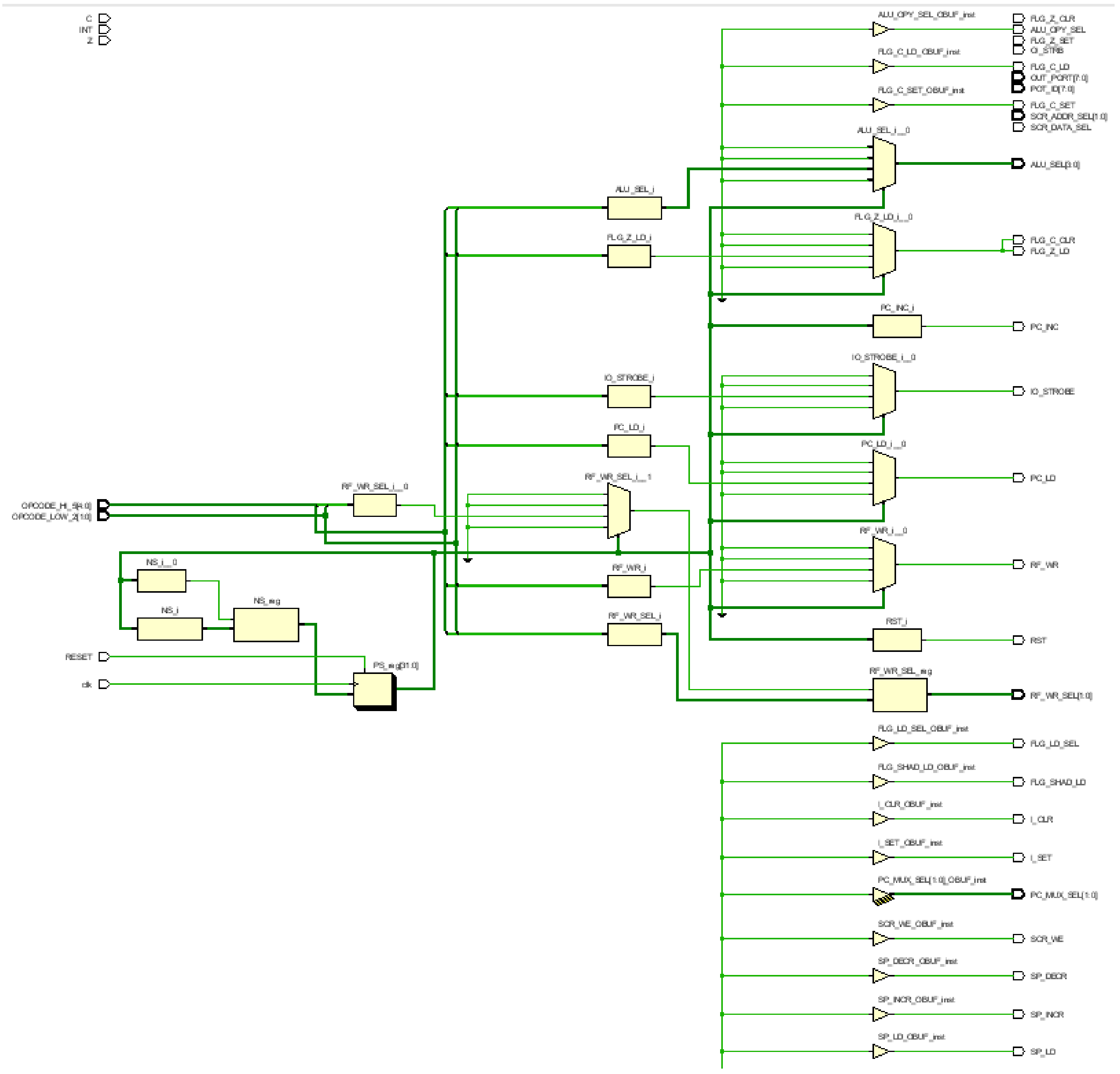
Flags: The Flags module is a sequential circuit whose inputs are a series of flags from the CU and ALU modules. The Flags module is comprised of four registers: C, SHAD C, Z, and SHAD Z. The inputs to the Flags module control the 4 registers and ultimately determine the outputs C_FLAG and Z_FLAG.

Structural Design

MCU

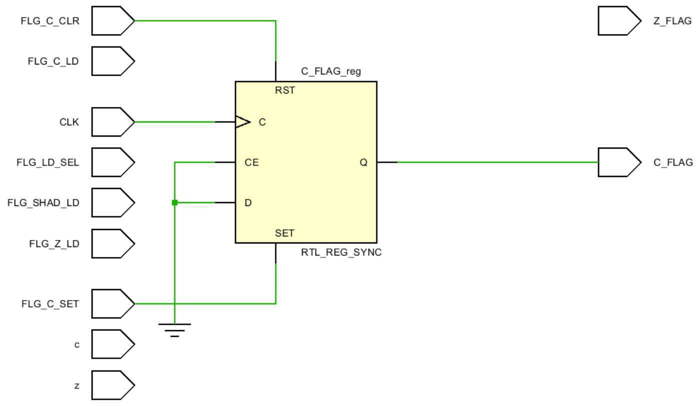


CU


C
INT
Z
OPCODE_HI_5[4:0]
OPCODE_LOW_2[1:0]
RESET
clk
NS_i__0
NS_i
NS_reg
PS_reg[0:0]
RF_WR_SEL_i__0
ALU_SEL_i
FLG_Z_LD_i
IO_STROBE_i
PC_LD_i
RF_WR_SEL_i__1
RF_WR_i
RF_WR_SEL_i
ALU_OPY_SEL_OBUF_inst
FLG_C_LD_OBUF_inst
FLG_C_SET_OBUF_inst
ALU_SEL_i__0
FLG_Z_LD_i__0
PC_INC_i
IO_STROBE_i__0
PC_LD_i__0
RF_WR_i__0
RST_i
RF_WR_SEL_reg
FLG_LD_SEL_OBUF_inst
FLG_SHAD_LD_OBUF_inst
I_CLR_OBUF_inst
I_SET_OBUF_inst
PC_MUX_SEL[1:0]_OBUF_inst
SCR_WE_OBUF_inst
SP_DECR_OBUF_inst
SP_INCR_OBUF_inst
SP_LD_OBUF_inst
FLG_Z_CLR
ALU_OPY_SEL
FLG_Z_SET
OI_STRB
FLG_C_LD
OUT_PORT[7:0]
PORT_ID[7:0]
FLG_C_SET
SCR_ADDR_SEL[1:0]
SCR_DATA_SEL
ALU_SEL[3:0]
FLG_C_CLR
FLG_Z_LD
PC_INC
IO_STROBE
PC_LD
RF_WR
RST
RF_WR_SEL[1:0]
FLG_LD_SEL
FLG_SHAD_LD
I_CLR
I_SET
PC_MUX_SEL[1:0]
SCR_WE
SP_DECR
SP_INCR
SP_LD

Flags

Note, that our flags structural design does not include the four registers. This resulted from the System Verilog code we created for the Registers. Our Register code defines Muxes of variable size such that Vivado cannot generate an elaborated design, presumably because the code describes a module of indefinite size. (**See Source, pg 11)**.



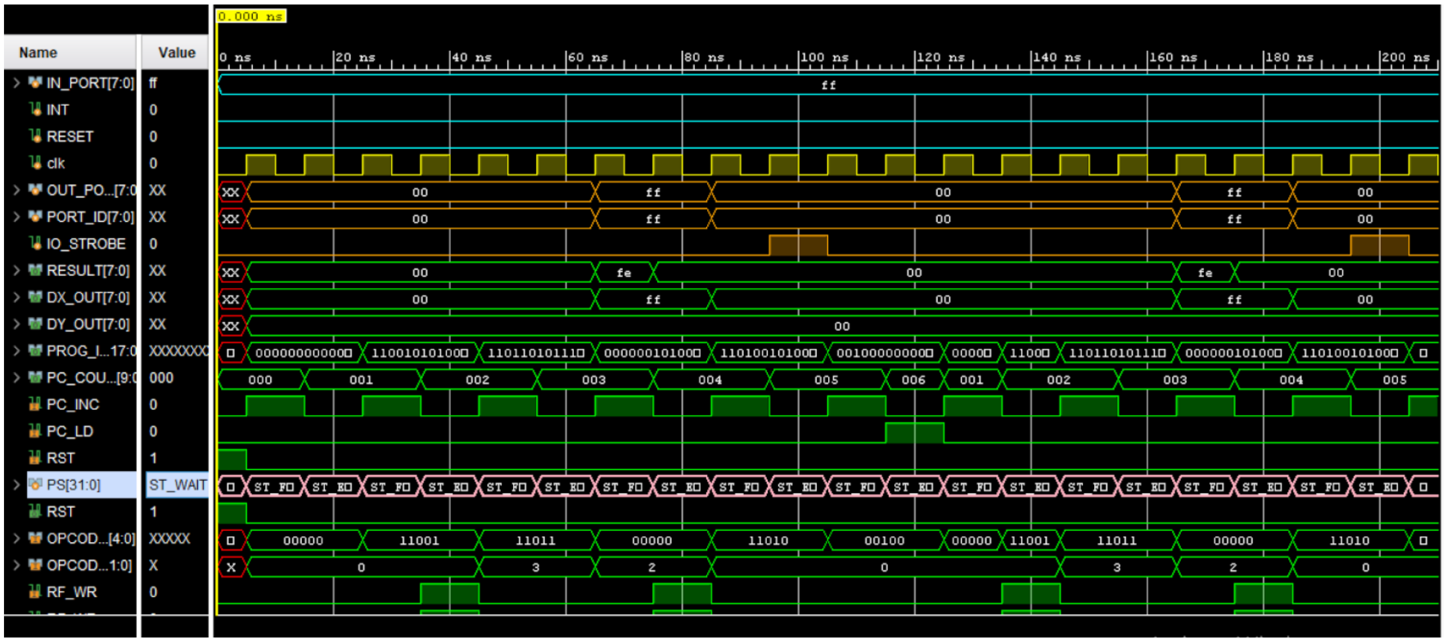
Verification

To verify the behavior of the MCU, we simulated the following assembly code via a System Verilog test bench. The test bench System Verilog code can be found on the last few pages of the report, under Source (**pg 12**).

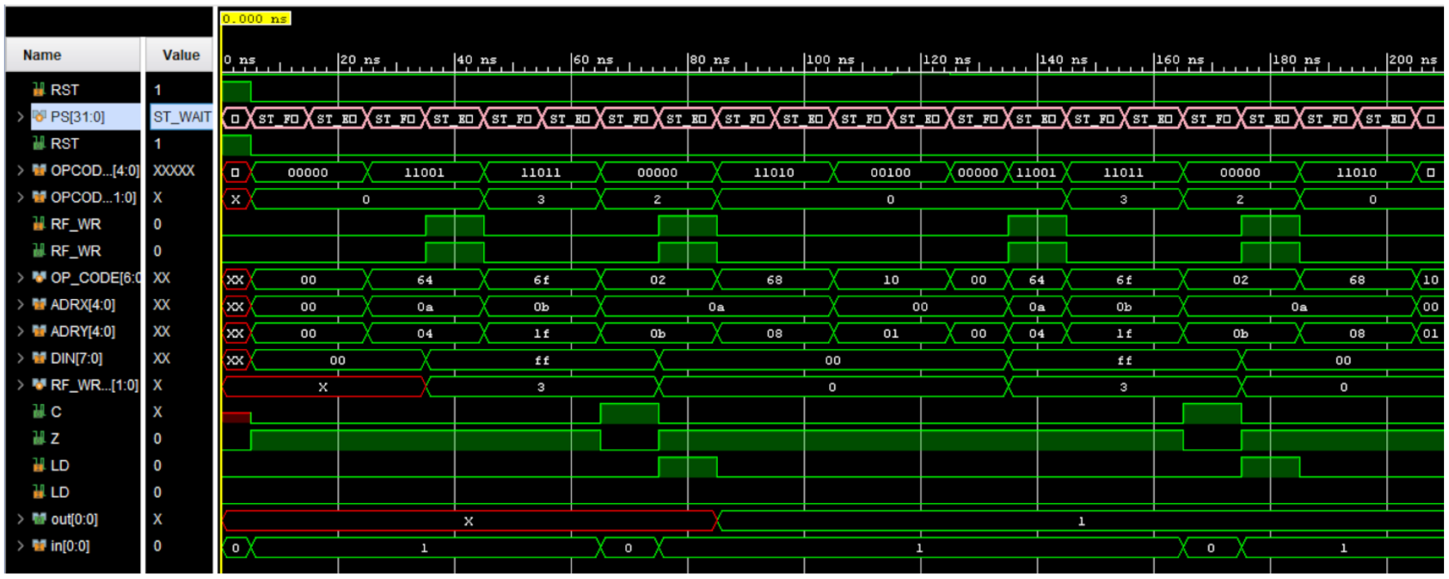
```
.EQU SWITCH_PORT = 0x20   ; port for switch input
.EQU LED_PORT = 0x40      ; port for LED output

.CSEG
.ORG 0x10

main:    IN   R10, SWITCH_PORT
         MOV R11, 0xFF
         EXOR R10, R11
         OUT R10, LED_PORT
             BRN main
```

The entire waveform can be found on the following page. For clarity, we colored input signals a light blue and output signals, orange. The internal clock signal is yellow and the present state of the MCU is labeled in pink.

Waveform Upper Half



Waveform Lower Half



Source Code

MCU (2 pages)

```
`timescale 1ns / 1ps

module RAT_MCU(
    input clk,
    input INT,
    input RESET,
    input [7:0] IN_PORT,
    output logic [7:0] OUT_PORT,
    output logic [7:0] PORT_ID,
    output logic IO_STROBE
    );
    logic [9:0] tFromPc;
    logic [17:0] tRom2Reg;
    logic [7:0] tIN, t2ALU, t2ALUMUX, tRESULT, t8,
        tFROMMUX, t11;
    logic [9:0] t6, t10;
    logic c, z;
    logic I_SEL, I_CLR, PC_LD, PC_INC, ALU_OPY_SEL,
        tRF_WR, SP_LD, SP_INCR, SP_DECR;
    logic SCR_WE, SCR_DATA_SEL, FLG_C_SET, FLG_C_CLR,
        FLG_C_LD, FLG_Z_LD, FLG_LD_SEL;
    logic FLG_SHAD_LD, RST;
    logic [1:0] PC_MUX_SEL, RF_WR_SEL, SCR_ADDR_SEL;
    logic [3:0] ALU_SEL; logic C_FLAG, Z_FLAG;

    PC PC(  .clk(clk), .FROM_IMMED(tRom2Reg[12:3]),
            .FROM_STACK(t6), .PC_MUX_SEL(PC_MUX_SEL),
            .PC_LD(PC_LD), .PC_INC(PC_INC), .RST(RST),
            .PC_COUNT(tFromPc));

    ProgRom ProgRom (   .PROG_CLK(clk), .PROG_ADDR(tFromPc),
                        .PROG_IR(tRom2Reg));

    PC_REGISTER PC_REG (.clk(clk), .DIN(tIN), .RF_WR(tRF_WR),
            .ADRX(  tRom2Reg[12:8]), .ADRY(tRom2Reg[7:3]),
                    .DX_OUT(t2ALU), .DY_OUT(t2ALUMUX));

    ALU ALU(.CIN(C_FLAG), .SEL(ALU_SEL),
            .A(t2ALU), .B(tFROMMUX),
            .RESULT(tRESULT), .C(c), .Z(z));

    ControlUnit PC_FSM (.clk(clk), .C(C_FLAG), .Z(Z_FLAG),
                        .RESET(RESET), .OPCODE_HI_5(tRom2Reg[17:13]),
                        .OPCODE_LOW_2(tRom2Reg[1:0]), .I_SET(I_SEL),
                        .I_CLR(I_CLR), .PC_LD(PC_LD), .PC_INC(PC_INC),
                        .ALU_OPY_SEL(ALU_OPY_SEL), .RF_WR(tRF_WR),
                        .SP_LD(SP_LD), .SP_INCR(SP_INCR), .SP_DECR(SP_DECR),
                        .SCR_WE(SCR_WE), .SCR_DATA_SEL(SCR_DATA_SEL),
                        .FLG_C_SET(FLG_C_SET), .FLG_C_CLR(FLG_C_CLR),
                        .FLG_C_LD(FLG_C_LD), .FLG_Z_LD(FLG_Z_LD),
                        .FLG_LD_SEL(FLG_LD_SEL), .FLG_SHAD_LD(FLG_SHAD_LD),
                        .RST(RST), .IO_STROBE(IO_STROBE),
                        .PC_MUX_SEL(PC_MUX_SEL), .RF_WR_SEL(RF_WR_SEL),
                        .SCR_ADDR_SEL(SCR_ADDR_SEL),.ALU_SEL(ALU_SEL) );
                        //Lacking INT assign PORT_ID = t2[7:0];
```

```
Reg4Flags  zflag (  .clk(clk), .in(z), .SET(0),
                    .LD(FLG_Z_LD), .CLR(0), .out(Z_FLAG));

Reg4Flags  cflag (  .clk(clk), .in(c), .SET(FLG_C_SET),
                    .LD(FLG_C_LD), .CLR(FLG_C_CLR), .out(C_FLAG));

//Muxes
mux2bits #8 REG_FILE_MUX (.zero(tRESULT), .one(t6[7:0]), .two(t8),
                          .three(IN_PORT), .sel(RF_WR_SEL), .muxout(tIN));

mux2bits #8 ALU_MUX (   .zero(t2ALU), .one(t2ALUMUX[7:0]),
                        .sel({1'b0, ALU_OPY_SEL}), .muxout(tFROMMUX));

mux2bits #10 SCR_DATA_IN_MUX (  .zero(t2ALU), .one(tFromPc),
                                .sel(SCR_DATA_SEL), .muxout(t10));

mux2bits #8 SCR_ADDR_MUX (  .zero(t2ALUMUX), .one(tRom2Reg[7:0]),
                            .two(t8), .three(-t8), .sel(SCR_ADDR_SEL),
                            .muxout(t11));

assign PORT_ID = t2ALU[7:0];
assign OUT_PORT = t2ALU;

endmodule
```

CU (3 pages)

```
`timescale 1ns / 1ps

module ControlUnit(
input  C, Z , INT, RESET, clk,
input [4:0] OPCODE_HI_5,
input [1:0] OPCODE_LOW_2,
output logic OI_STRB,  RST,

output logic I_SET, I_CLR,

output  logic PC_LD, PC_INC,
output logic [1:0] PC_MUX_SEL,

output logic ALU_OPY_SEL,
output logic [3:0] ALU_SEL,

output logic RF_WR,
output logic [1:0] RF_WR_SEL,

output logic SP_LD, SP_INCR, SP_DECR,
output logic SCR_WE, SCR_DATA_SEL,
output logic [1:0] SCR_ADDR_SEL,
output logic FLG_C_SET, FLG_C_CLR, FLG_C_LD, FLG_Z_SET, FLG_Z_CLR,
FLG_Z_LD, FLG_LD_SEL, FLG_SHAD_LD,

output logic IO_STROBE,
output logic [7:0] OUT_PORT, POT_ID

 );

logic [6:0] OP_CODE;
assign OP_CODE = {OPCODE_HI_5 , OPCODE_LOW_2};

typedef enum  { ST_WAIT, ST_FETCH, ST_EXEC} STATE;
STATE NS; STATE PS= ST_WAIT;

 always_ff @ ( posedge clk)
 begin
    if(RESET==1)
    PS <= ST_WAIT;
    else
    PS <= NS;
end
```

```
always_comb
begin
  I_SET = 0;
    I_CLR = 0;
    PC_LD = 0;
    PC_INC = 0;
    PC_MUX_SEL = 0;
    ALU_OPY_SEL = 0;
    RF_WR = 0;
    SP_LD = 0;
    SP_INCR = 0;
    SP_DECR = 0;
    SCR_WE = 0;
    ALU_SEL=0;
    FLG_C_SET = 0;
    FLG_C_CLR = 0;
    FLG_C_LD = 0;
    FLG_Z_LD = 0;
    FLG_LD_SEL = 0;
    FLG_SHAD_LD = 0;
    IO_STROBE=0;
    RST = 0;

case (PS)
    ST_WAIT:
            begin
            RST = 1;
            NS= ST_FETCH;
            end
   ST_FETCH:
            begin
            PC_INC=1;
            NS= ST_EXEC;
                end
```

```
ST_EXEC:
            begin
            case (OP_CODE)
           //In

                 7'b1100100, 7'b1100101, 7'b1100110, 7'b1100111:
                 begin
                    RF_WR =1; RF_WR_SEL = 3 ;
                 end
          // Mov Reg-Imed
                 7'b0001001:
                 // 7'b1101100, 7'b1101101, 7'b1101110, 7'b1101111:
                 begin
                    RF_WR=1; ALU_SEL= 4'b1110; RF_WR_SEL=0;
//ALU_OPY_SEL=1;
                 end
            //Exor
                 7'b0000010:
                 begin
                    ALU_SEL= 4'B0111; RF_WR=1; FLG_Z_LD = 1;
FLG_C_CLR=1; RF_WR_SEL=0;
                 end
            // OUT
                 7'b1101000, 7'b1101001, 7'b1101010, 7'b1101011:
                 begin
                      IO_STROBE=1; //RF_WR=1;
                 end
           //BRN
                7'B0010000:
                begin
                    PC_LD=1; PC_MUX_SEL=0;
                end
          //default: RST = 1; //never gets here

          endcase

       NS = ST_FETCH;
       end
       //default: NS = ST_WAIT;
 endcase
 end
endmodule
```

Flags & Registers

```
module FLAGS(
    input CLK,
    input FLG_C_SET,c,z,
    input FLG_C_CLR,
    input FLG_C_LD,
    input FLG_Z_LD,
    input FLG_LD_SEL,
    input FLG_SHAD_LD,

    output logic C_FLAG,
    output logic Z_FLAG
    );

    always_ff@ (posedge CLK)
    begin
    if (FLG_C_CLR == 1)
    C_FLAG <= 0 ;
    else if (FLG_C_SET == 1)
    C_FLAG <= 1;
    else if (FLG_C_LD == 1)
    C_FLAG  = C_FLAG;
    end
endmodule



module Reg4Flags #(parameter WIDTH = 1)(
input clk,
input [WIDTH - 1 : 0] in,
input SET, LD, CLR,
output logic [WIDTH - 1 : 0] out );

 always_ff @ (posedge clk)
 begin
   if (CLR) out = 0;
   else if (LD) out = in;
   else if (SET) out = 1;
    end
endmodule
```

Testbench

```
`timescale 1ns / 1ps
module RAT_MCU_tb(

    );
            logic [7:0] IN_PORT;
            logic INT;
            logic RESET;
            logic clk;
            logic [7:0] OUT_PORT;
            logic [7:0] PORT_ID;
            logic IO_STROBE;

           RAT_MCU RAT_MCU_INST (.*);

              always
              begin
               clk = 0; #5;
               clk = 1; #5;
               end


                 initial
                 begin

                     INT= 0;
                     RESET = 0; IN_PORT = 8'hFF;
                     #120;
                 end
          endmodule
```