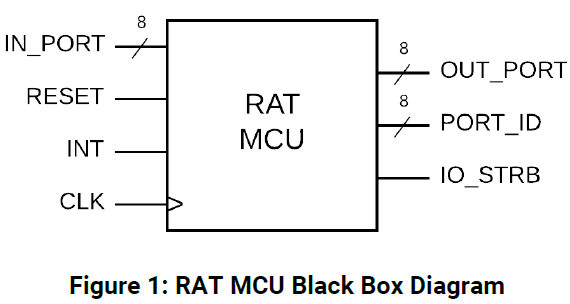
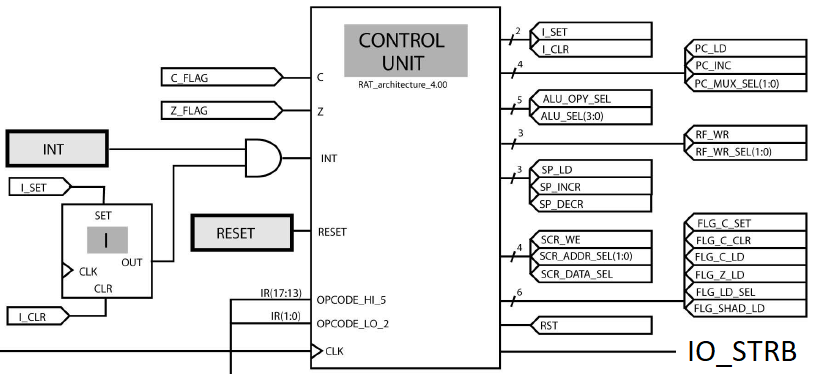
CPE 233: RAT assignment 5, MCU

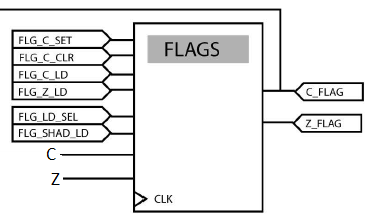
Prof. Bridget Benson

Stan Carpenco, Luis Gomez

# BBD







# 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***

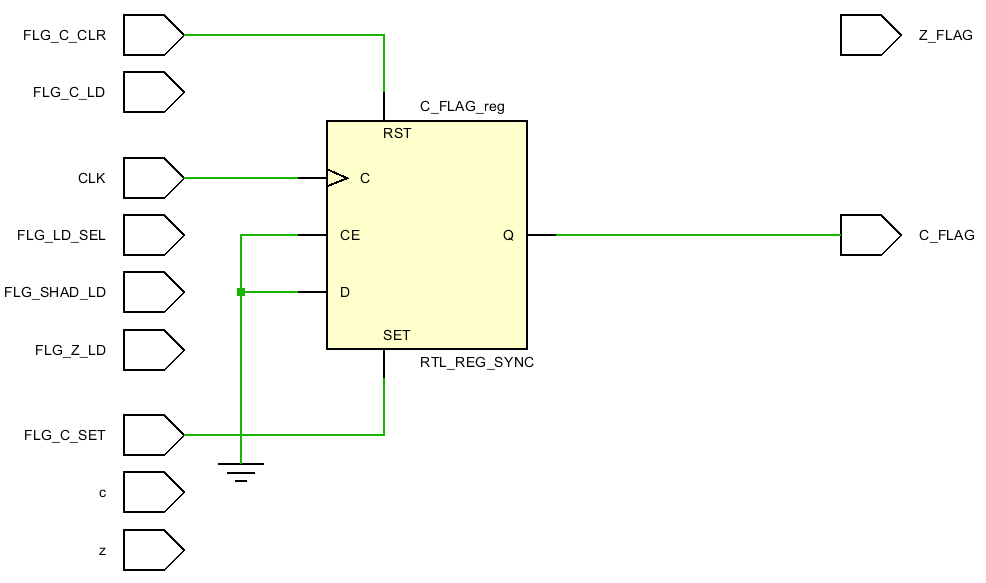
# 

***CU***

# 

***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

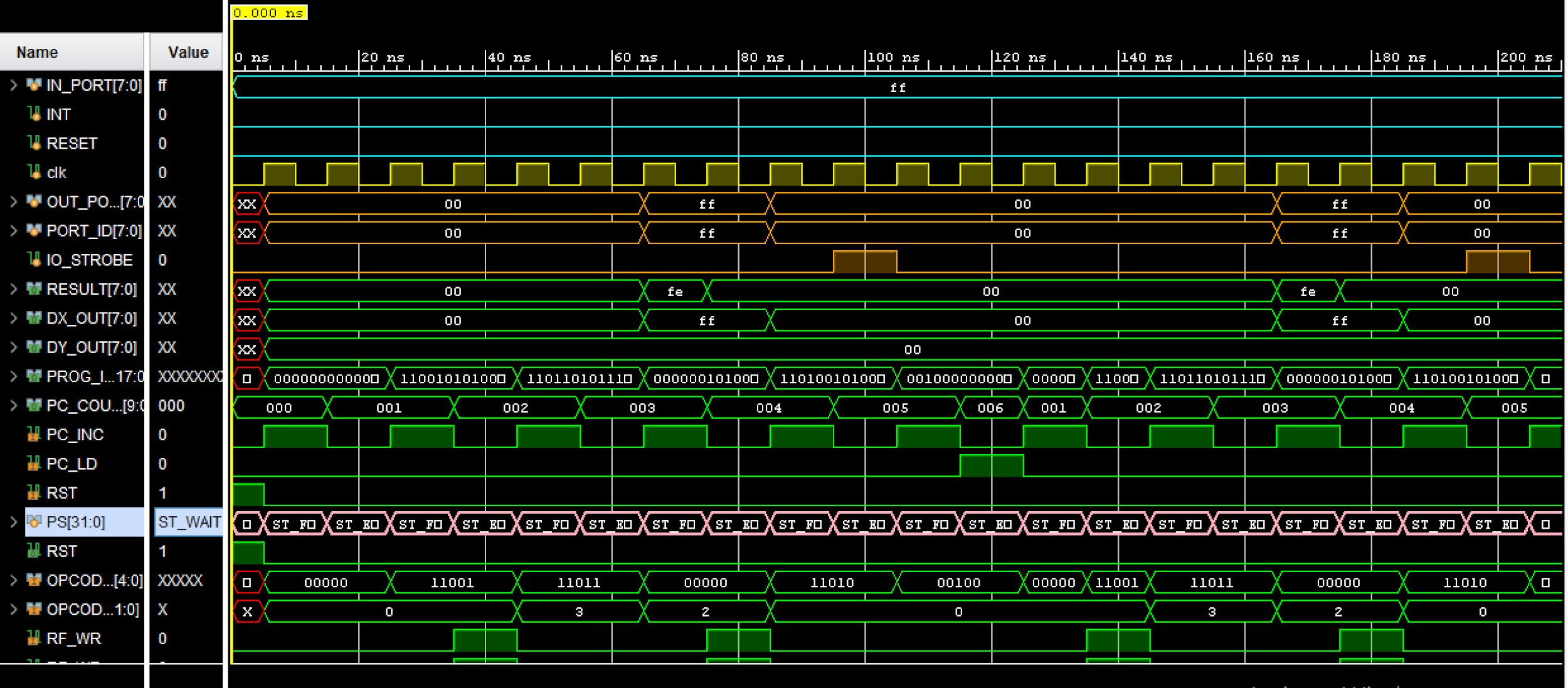
EXOR R10, R11

OUT R10, LED\_PORT

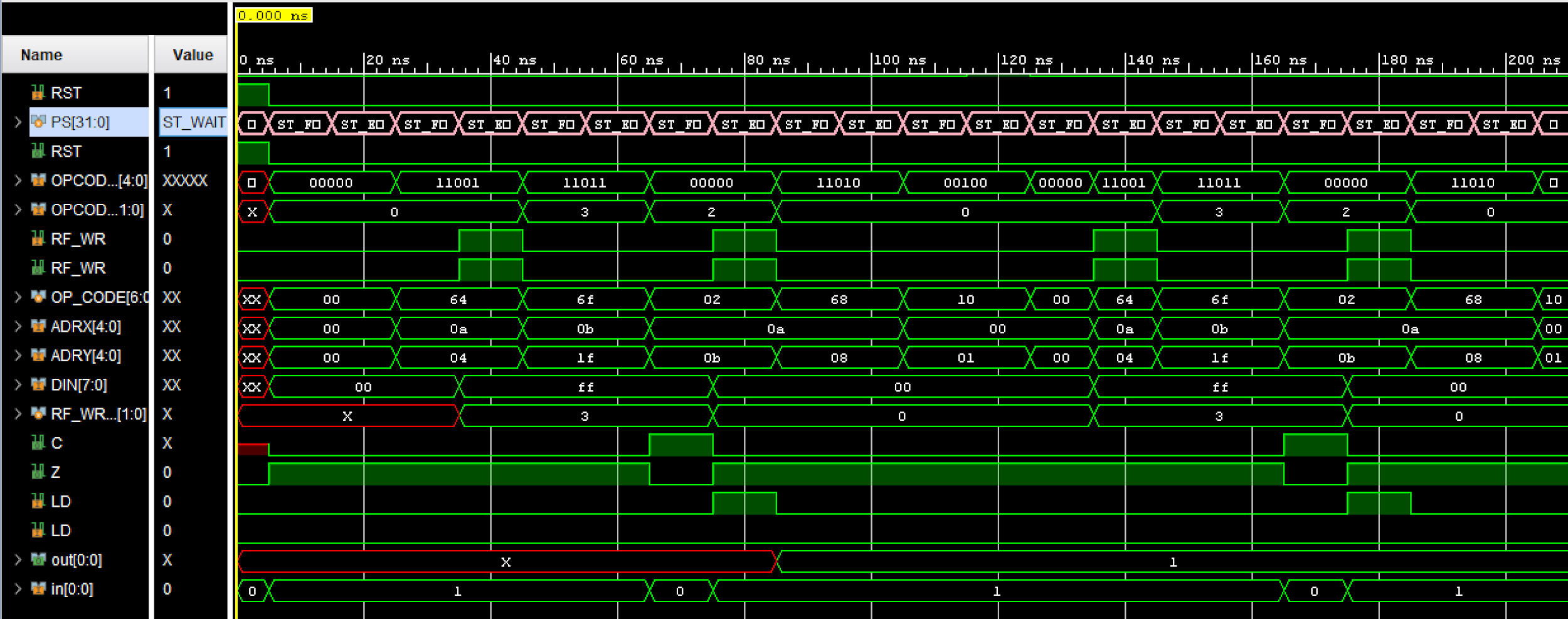
BRN main

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