Rat Assignment 6 – RAT MCU / RAT Wrapper
Stan Carpenco & Luis Gomez

Black Box Diagram

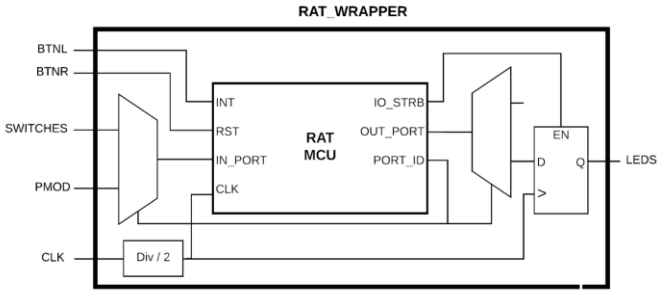

Figure 1: Black Box Diagram of the RAT WRAPPER

Behavior Description

The module RAT_WRAPPER is designed to act as a bridge between the designed RAT micro controller and any attached peripherals. The Wrapper receives the executed software instructions from the hardware design and allows the user to receive feedback from the peripherals. Additionally, the wrapper acts like a temporary storage device for the output signal. The signals that are activated by the wrapper are not limited to the run execution of one cycle of the control unit. For example, if the control unit sets IO_STRB to high, it only stays high only when its activated. On the other hand, an on instruction to the external LEDs will constantly stay on until reset.

Overall, the Rat Wrapper expands the functionality of the RAT MCU by allowing the mjcro controller to communicate to the Basys board and its peripherals. The Wrapper is equipped with two multiplexors that control the input and outputs of the MCU. Additionally, the outputs are equipped with registers that store the data as a constant until it needs to be changed .

Structural Design

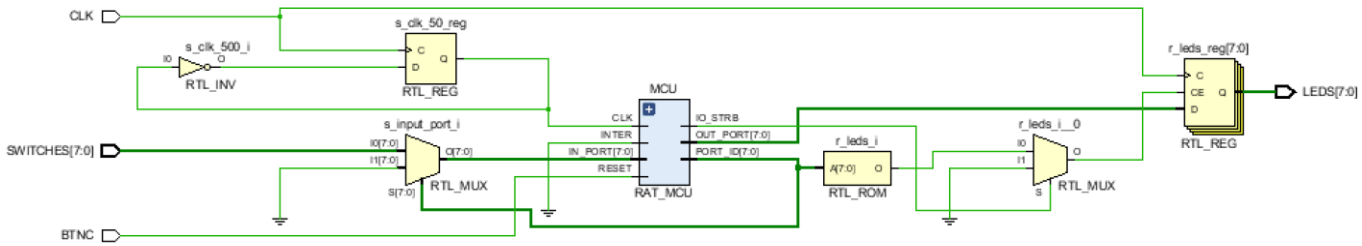

Figure 2: Elaborated Design Schematic of the RAT_WRAPPER Module

Rat Assignment 6 – RAT MCU / RAT Wrapper
Stan Carpenco & Luis Gomez

System Verilog Source Code

Control Unit:

```
//////////////////////////////////////////////////////////////]
// Engineers: Stan Carpenco & Luis Gomez
// Description:
// Finite State Machine that controls all the modules
// of the Rat Micro Controller.
//////////////////////////////////////////////////////////////
module ControlUnit2(
        input C,Z,INTERRUPT, RESET, CLK,
        input [4:0] OPCODE_HI_5,
        input [1:0] OPCODE_LOW_2,

           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 FLG_C_SET, FLG_C_CLR, FLG_C_LD, FLG_Z_LD, RST, IO_STRB
                  );

   typedef enum {ST_INIT, ST_FETCH, ST_EXEC} STATE;
   STATE NS, PS = ST_INIT;
   logic [6:0] opcode;
   assign opcode = {OPCODE_HI_5, OPCODE_LOW_2};

   always_ff @ (posedge CLK)
    begin
          if (RESET == 1) PS <= ST_INIT;
          else            PS <= NS;
    end

   always_comb
    begin
         PC_LD = 0;  PC_INC = 0; PC_MUX_SEL = 0; ALU_OPY_SEL = 0;
         ALU_SEL = 0;  RF_WR = 0; RF_WR_SEL = 0; FLG_C_SET = 0; FLG_C_CLR = 0;
         FLG_C_LD = 0;  FLG_Z_LD = 0;  RST = 0;  IO_STRB = 0;

   begin
     case (PS)
     ST_INIT:
          begin
                RST = 1;  NS = ST_FETCH;
          end
  ST_FETCH:
          begin
                PC_INC = 1;  NS = ST_EXEC;
          end
   ST_EXEC:
           begin
            case(opcode)
 //IN
             7'b1100100, 7'b1100101, 7'b1100110, 7'b1100111:
                begin
                  RF_WR_SEL = 3; RF_WR = 1;
                end
//MOV
       7'b0001001: //reg to reg case
        begin
```

Rat Assignment 6 – RAT MCU / RAT Wrapper
Stan Carpenco & Luis Gomez

```
            ALU_OPY_SEL = 0; ALU_SEL = 4'b1110; RF_WR_SEL = 2'b00;  RF_WR = 1;
        end
     7'b1101100, 7'b1101101, 7'b1101110, 7'b1101111:
               begin
                 ALU_OPY_SEL = 1; ALU_SEL = 4'b1110;  RF_WR_SEL = 0;  RF_WR = 1;
               end
//EXOR
              7'b0000010: //reg to reg case
               begin
                 ALU_OPY_SEL = 0; ALU_SEL = 4'b0111;  FLG_C_CLR = 1; FLG_Z_LD = 1; RF_WR_SEL = 0;
RF_WR = 1;
               end
              7'b1001000, 7'b1001001, 7'b1001010, 7'b1001011:
               begin
                 ALU_OPY_SEL = 1; ALU_SEL = 4'b0111;   FLG_C_CLR = 1; FLG_Z_LD = 1; RF_WR_SEL =
0; RF_WR = 1;
               end
 //OUT
              7'b1101000, 7'b1101001, 7'b1101010, 7'b1101011:
               begin
                 IO_STRB = 1;
               end
//BRN
              7'b0010000:
               begin
                 PC_LD = 1; PC_MUX_SEL = 0;
               end
//AND
              7'b0000000: // reg to reg case
               begin
                 ALU_OPY_SEL = 0;                  ALU_SEL = 4'b0101;  FLG_C_CLR = 1; FLG_Z_LD =
1; RF_WR_SEL = 0; RF_WR = 1;
               end
              7'b1000000, 7'b1000001, 7'b1000010, 7'b1000011:
               begin
                 ALU_OPY_SEL = 1; ALU_SEL = 4'b0101; FLG_C_CLR = 1; FLG_Z_LD = 1;  RF_WR_SEL =
0;  RF_WR = 1;
               end
//OR
              7'b0000001: // reg to reg case
               begin
                 ALU_OPY_SEL = 1'b0; ALU_SEL = 4'b0110;
                 FLG_C_CLR = 1; FLG_Z_LD = 1;  RF_WR_SEL = 0;  RF_WR = 1;
               end
              7'b1000100, 7'b1000101, 7'b1000110, 7'b1000111:
               begin
                 ALU_OPY_SEL = 1; ALU_SEL = 4'b0110;  FLG_C_CLR = 1;  FLG_Z_LD = 1; RF_WR_SEL =
0; RF_WR = 1;
               end
            //TEST
              7'b0000011: // reg to reg case
               begin
                 ALU_OPY_SEL = 1'b0; ALU_SEL = 4'b1000; FLG_C_CLR = 1; FLG_Z_LD = 1; RF_WR_SEL =
2'b00; RF_WR = 0;
               end
              7'b1001100, 7'b1001101, 7'b1001110, 7'b1001111:
               begin
                 ALU_OPY_SEL = 1'b1; ALU_SEL = 4'b1000;  FLG_C_CLR = 1;  FLG_Z_LD = 1;
RF_WR_SEL = 2'b00; RF_WR = 0;
               end
//ADD
              7'b0000100: // reg to reg case
               begin
                 ALU_OPY_SEL = 0; ALU_SEL = 4'b0000; FLG_C_LD = 1;  FLG_Z_LD = 1;  RF_WR_SEL = 0
;RF_WR = 1;
               end
              7'b1010000, 7'b1010001, 7'b1010010, 7'b1010011:
               begin
```

Rat Assignment 6 – RAT MCU / RAT Wrapper
Stan Carpenco & Luis Gomez

```
                ALU_OPY_SEL = 1; ALU_SEL = 4'b0000; FLG_C_LD = 1; FLG_Z_LD = 1; RF_WR_SEL = 0;
RF_WR = 1;
              end
//ADDC
            7'b0000101: // reg to reg case
              begin
                ALU_OPY_SEL = 1'b0; ALU_SEL = 4'b0001; FLG_C_LD = 1; FLG_Z_LD = 1; RF_WR_SEL =
0;  RF_WR = 1;
              end
            7'b1010100, 7'b1010101, 7'b1010110, 7'b1010111:
              begin
                ALU_OPY_SEL = 1; ALU_SEL = 4'b0001; FLG_C_LD = 1;  FLG_Z_LD = 1;  RF_WR_SEL =
0; RF_WR = 1;
              end
 //SUB
            7'b0000110: // reg to reg case
              begin
                ALU_OPY_SEL = 0; ALU_SEL = 4'b0010;
                FLG_C_LD = 1; FLG_Z_LD = 1;  RF_WR_SEL = 0;  RF_WR = 1;
              end
            7'b1011000, 7'b1011001, 7'b1011010, 7'b1011011:
              begin
                ALU_OPY_SEL = 1'b1;   ALU_SEL = 4'b0010;  FLG_C_LD = 1;  FLG_Z_LD = 1;
RF_WR_SEL = 0; RF_WR = 1;
              end
//SUBC
            7'b0000111: // reg to reg case
              begin
                ALU_OPY_SEL = 1'b0;  ALU_SEL = 4'b0011;  FLG_C_LD = 1;FLG_Z_LD = 1;  RF_WR_SEL
= 0 ;RF_WR = 1;
              end
            7'b1011100, 7'b1011101, 7'b1011110, 7'b1011111:
              begin
                ALU_OPY_SEL = 1;  ALU_SEL = 4'b0011; FLG_C_LD = 1;  FLG_Z_LD = 1; RF_WR_SEL =
0;   RF_WR = 1;
              end
 //CMP
            7'b0001000: // reg to reg case
              begin
                ALU_OPY_SEL = 1'b0;  ALU_SEL = 4'b0100;   FLG_C_LD = 1; FLG_Z_LD = 1; RF_WR_SEL
= 0; RF_WR = 0;
              end
            7'b1100000, 7'b1100001, 7'b1100010, 7'b1100011:
              begin
                ALU_OPY_SEL = 1'b1; ALU_SEL = 4'b0100;  FLG_C_LD = 1;  FLG_Z_LD = 1;  RF_WR_SEL
= 0; RF_WR = 0;
              end
  //LD
            7'b0001010: // reg to reg case
              begin
              // FILL ME
              end
            7'b1110000, 7'b1110001, 7'b1110010, 7'b1110011:
              begin
              // FILL ME
              end
          //ST
            7'b0001011: // reg to reg case
              begin
              // FILL ME
              end
            7'b1110100, 7'b1110101, 7'b1110110, 7'b1110111:
              begin
              // FILL ME
              end
          //CALL
            7'b0010001:
              begin
              // FILL NE
```

Rat Assignment 6 – RAT MCU / RAT Wrapper
Stan Carpenco & Luis Gomez

```
    end
//BREQ
  7'b0010010:
    begin
      if (Z==1)  PC_LD = 1;
      else       PC_LD = 0;
    end
//BRNE
  7'b0010011:
    begin
      if (Z==0) PC_LD = 1;
      else      PC_LD = 0;
    end
//BRCS
  7'b0010100:
    begin
      if (C==1)  PC_LD = 1;
      else       PC_LD = 0;
    end
//BRCC
  7'b0010101:
    begin
      if (C==0) PC_LD = 1;
      else      PC_LD = 0;
    end
//LSL
  7'b0100000:
    begin
      ALU_SEL = 4'b1001; FLG_Z_LD = 1;  RF_WR_SEL = 0; RF_WR = 1;
    end
//LSR
  7'b0100001:
    begin
      ALU_SEL = 4'b1010;   FLG_Z_LD = 1; RF_WR_SEL = 2'b00; RF_WR = 1;
    end
//ROL
  7'b0100010:
    begin
      ALU_SEL = 4'b1011;  FLG_Z_LD = 1;  RF_WR_SEL = 0; RF_WR = 1;
    end
//ROR
  7'b0100011:
    begin
      ALU_SEL = 4'b1100;  FLG_Z_LD = 1; RF_WR_SEL = 2'b00; RF_WR = 1;
    end
//ASR
  7'b0100100:
    begin
      ALU_SEL = 4'b1101; FLG_Z_LD = 1; RF_WR_SEL = 0; RF_WR = 1;
    end
//PUSH
  7'b0100101:
    begin
    //FILL NE
    end
//POP
  7'b0100110:
    begin
    //FILL ME
    end
//WSP
  7'b0101000:
    begin
    //FILL ME
    end
//RSP
  7'b0101001:
    begin
    //FILL ME
```

Rat Assignment 6 – RAT MCU / RAT Wrapper
Stan Carpenco & Luis Gomez

```
                end
              //CLC
                7'b0110000:
                begin
                  FLG_C_CLR = 1;
                end
              //SEC
                7'b0110001:
                begin
                  FLG_C_SET = 1;
                end
              //RET
                7'b0110010:
                begin
                // FILL ME
                end
              //SEI
                7'b0110100:
                begin
                //FILL ME
                end
              //CLI
                7'b0110101:
                begin
                //FILL ME
                end
              //RETID
                7'b0110110:
                begin
                // FILL ME
                end
              //RETIE
                7'b0110111:
                begin
                //FILL ME
                end
             default: RST = 1;
           endcase
          NS = ST_FETCH;
          end
          default : NS = ST_INIT;
          endcase
        end
      end

endmodule
```

Rat Assignment 6 – RAT MCU / RAT Wrapper
Stan Carpenco & Luis Gomez

Rat_Wrapper

```
//////////////////////////////////////////////////////////////////////////////////
// Company: Cal Poly
// Engineer: Paul Hummel
//
// Create Date: 06/28/2018 05:21:01 AM
// Module Name: RAT_WRAPPER
// Target Devices: RAT MCU on Basys3
// Description: Basic RAT_WRAPPER
//
// Revision:
// Revision 0.01 - File Created
//////////////////////////////////////////////////////////////////////////////////

module RAT_WRAPPER(
    input CLK,
    input BTNC,
    input [7:0] SWITCHES,
    output [7:0] LEDS
    );

    // INPUT PORT IDS ////////////////////////////////////////////////////////
    // Right now, the only possible inputs are the switches
    // In future labs you can add more port IDs, and you'll have
    // to add constants here for the mux below
    localparam SWITCHES_ID = 8'h20;

    // OUTPUT PORT IDS ///////////////////////////////////////////////////////
    // In future labs you can add more port IDs
    localparam LEDS_ID      = 8'h40;

    // Signals for connecting RAT_MCU to RAT_wrapper //////////////////////////
    logic [7:0] s_output_port;
    logic [7:0] s_port_id;
    logic s_load;
    logic s_interrupt;
    logic s_reset;
    logic s_clk_50 = 1'b0;     // 50 MHz clock

    // Register definitions for output devices ///////////////////////////////
    logic [7:0]   s_input_port;
    logic [7:0]   r_leds = 8'h00;

    // Declare RAT_CPU ///////////////////////////////////////////////////////
    RAT_MCU MCU (.INTER(s_interrupt), .RESET(s_reset), .CLK(s_clk_50),
     .IN_PORT(s_input_port), .IO_STRB(s_load), .OUT_PORT(s_output_port),
      .PORT_ID(s_port_id)  );

    // Clock Divider to create 50 MHz Clock //////////////////////////////////
    always_ff @(posedge CLK) begin
        s_clk_50 <= ~s_clk_50;
    end


    // MUX for selecting what input to read //////////////////////////////////
    always_comb begin
        if (s_port_id == SWITCHES_ID)
            s_input_port = SWITCHES;
        else
            s_input_port = 8'h00;
    end

    // MUX for updating output registers /////////////////////////////////////
    // Register updates depend on rising clock edge and asserted load signal
    always_ff @ (posedge CLK) begin
        if (s_load == 1'b1) begin
            if (s_port_id == LEDS_ID)
                r_leds <= s_output_port;
            end
```

```
    end


// Connect Signals ////////////////////////////////////////////////////
assign s_reset = BTNC;
assign s_interrupt = 1'b0;  // no interrupt used yet

// Output Assignments /////////////////////////////////////////////////
assign LEDS = r_leds;

endmodule
```