Σχεδιασμός Συστημάτων VLSI

Εργαστήριο 2

Λουδάρος Ιωάννης (1067400)

Μπορείτε να δείτε την τελευταία έκδοση του Project εδώ ή σκανάροντας τον κωδικό QR που βρίσκεται στην επικεφαλίδα.

Περιγραφή Αναφοράς

Παρακάτω παραθέτω τις απαντήσεις μας στην "Δεύτερη Εργαστηριακή Άσκηση" του μαθήματος "Σχεδιασμός Συστημάτων VLSI" καθώς και σχόλια τα οποία προέκυψαν κατά την εκπόνηση του.

Περιεχόμενα

Απαντήσεις

1. Ζητούμενο 1.

Ο Decoder

Το κύκλωμα ακολουθεί στην επόμενη σελίδα

```
module decoder_tb #(
    parameter dec_size = 3
);

reg [dec_size-1:0] dec_input;
wire [2**dec_size-1:0] dec_output;
reg enable;

decoder #(.size(dec_size)) test_dec(dec_input, enable, dec_output);

integer i,j;
integer error=0;

initial begin

$display("\n\nThis is a test for: %d inputs! \n ", dec_size);

for (i = 0; i<2**dec_size ;i=i+1) begin
        for (j=0; j<=1; j=j+1) begin

            dec_input = i;
            enable = j;
            #5;
            if (!enable)begin
                if (dec_output!= 8'b1 << i) error = 1;
            end
            else begin
                if (dec_output) error = 1;
            end

            if (error) begin
                $display("❌ Error in simulation when: \n enable:%b,\n dec_input:%b,\n 
ouput:%b ",enable, dec_input, dec_output);
                $finish;
            end
            else $display("✅ enable:%b dec_input:%b ouput:%b ",enable, dec_input, 
dec_output);
        end
    end
    $display("✅ Congrats! It's working.");
end

endmodule
```

```
module decoder #(
    parameter size = 3
)(
    input wire [size-1:0] enc,
    input en,
    output reg [2**size-1:0] dec
);

    always @(*) begin
        if(!en) begin
            dec = 0;
            dec[enc] = 1;
        end
        else begin
            dec = 0;
        end
    end
endmodule
```

Shift Register

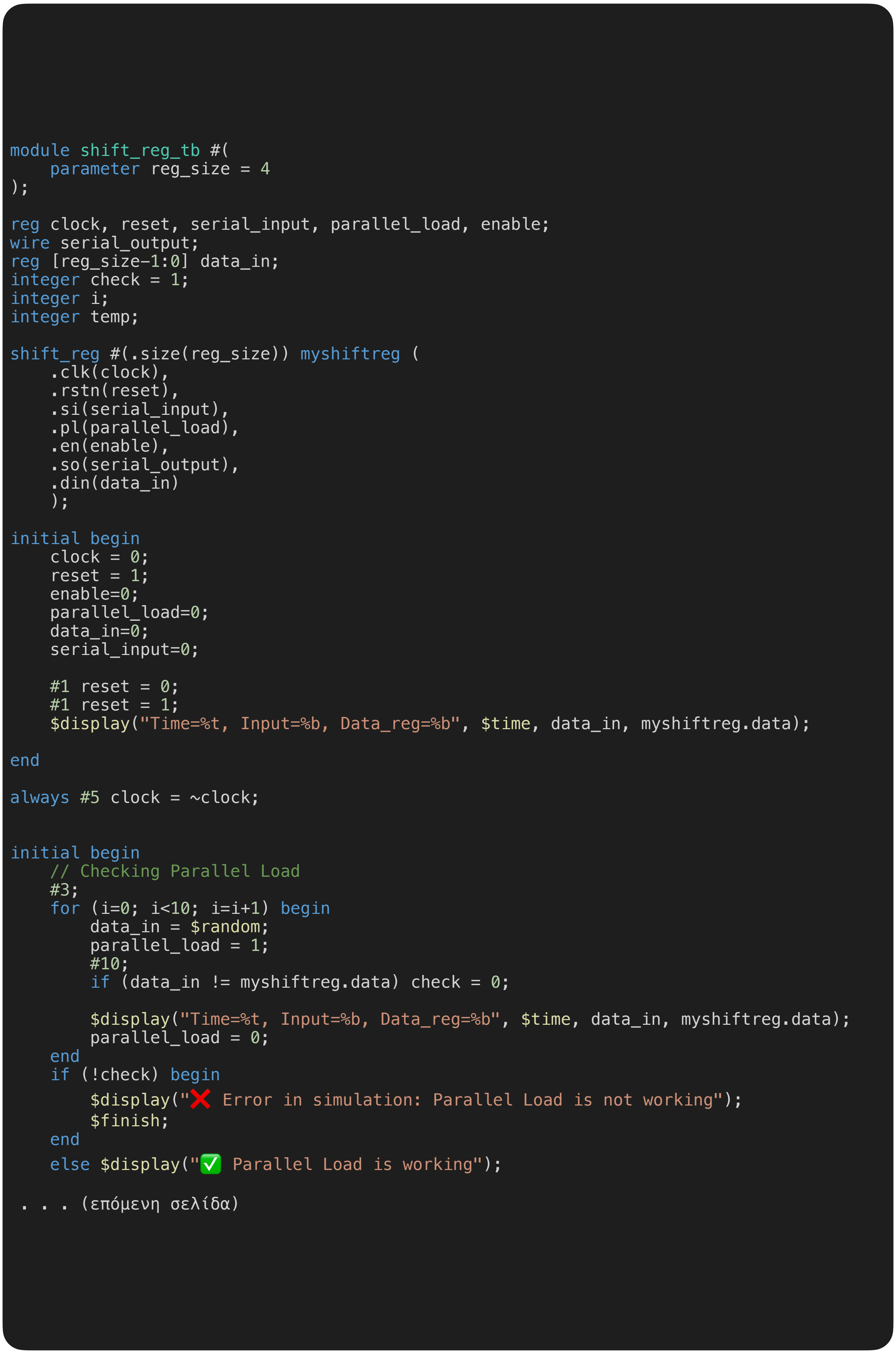
```
module shift_reg #(
    parameter size = 4
) (
    input clk,
    input rstn,
    input [size-1:0] din,
    input pl,
    input en,
    input si,
    output so
);

reg [size-1:0] data;
reg lsb_out;

assign so = lsb_out;


always @(posedge clk, negedge rstn ) begin
    if (pl) data <= din;
    if (en) {data,lsb_out} <= {si,data};
    if (!rstn) begin
        data <=0;
        lsb_out <= 0;
    end
end


endmodule
```



```
module shift_reg_tb #(
    parameter reg_size = 4
);

reg clock, reset, serial_input, parallel_load, enable;
wire serial_output;
reg [reg_size-1:0] data_in;
integer check = 1;
integer i;
integer temp;

shift_reg #(.size(reg_size)) myshiftreg (
    .clk(clock),
    .rstn(reset),
    .si(serial_input),
    .pl(parallel_load),
    .en(enable),
    .so(serial_output),
    .din(data_in)
    );

initial begin
    clock = 0;
    reset = 1;
    enable=0;
    parallel_load=0;
    data_in=0;
    serial_input=0;

    #1 reset = 0;
    #1 reset = 1;
    $display("Time=%t, Input=%b, Data_reg=%b", $time, data_in, myshiftreg.data);

end

always #5 clock = ~clock;


initial begin
    // Checking Parallel Load
    #3;
    for (i=0; i<10; i=i+1) begin
        data_in = $random;
        parallel_load = 1;
        #10;
        if (data_in != myshiftreg.data) check = 0;

        $display("Time=%t, Input=%b, Data_reg=%b", $time, data_in, myshiftreg.data);
        parallel_load = 0;
    end
    if (!check) begin
        $display("❌ Error in simulation: Parallel Load is not working");
        $finish;
    end
    else $display("✅ Parallel Load is working");

 . . . (επόμενη σελίδα)
```

```
// Checking Serial Load
    for (i=0; i<10; i=i+1) begin
        temp = myshiftreg.data;
        serial_input = $random;
        enable=1;
        #10;
        $display("Time=%t, Input=%b, Data_reg=%b, Serial_out=%b", $time, serial_input, 
myshiftreg.data, serial_output);
        enable=0;


        if (!(myshiftreg.data =={serial_input,temp[reg_size-1:1]} && serial_output == 
temp[0])) check=0;
    end
    if (!check) begin
        $display("❌ Error in simulation: Serial Load is not working");
        $finish;
    end
    else $display("✅ Serial Load is working");
    $finish;
end
endmodule
```

Accumulator

```
module accumulator #(
    parameter m=4,
    parameter n=2,
    parameter k=4
) (
    input [(m+n)*k-1:0] din,
    input pl, clk, rstn,
    output reg ready,
    output reg [m+n+(k-1)-1:0] sum
);

wire shift_enable;
wire load_enable;
reg [$clog2(k):0] counter;
wire [m+n-1:0] sreg_sum_connect;
wire [k-1:0] sreg_din_bus[m+n-1:0];

assign load_enable = pl & ready;
assign shift_enable = ~ready;

integer j;





genvar i,p,q;

generate
    for(p=0; p<m+n ; p = p+1 ) begin : bus_creation_bit_selection
    for (q=0; q<k-1 ; q=q+1) begin : bus_creation_operant_selection

        assign sreg_din_bus[p][q] = din[q*(m+n)+p];
    end
end
endgenerate

generate
    for (i=0; i<m+n; i=i+1) begin : reg_file
        shift_reg #(.size(k-1)) sr(
            .clk(clk),
```

```verilog
generate
    for (i=0; i<m+n; i=i+1) begin : reg_file
        shift_reg #(.size(k-1)) sr(
            .clk(clk),
            .rstn(rstn),
            .din(sreg_din_bus[i]),
            .pl(load_enable),
            .en(shift_enable),
            .si(1'b0),
            .so(sreg_sum_connect[i]));
    end
endgenerate


always @(posedge clk, negedge rstn) begin

    if (!rstn) begin
        ready <= 1;
        counter <= 0;
        sum <= 0 ;
    end
end

always @(posedge clk) begin
if (ready) begin
    counter <= 0;
    if(pl) begin
        ready <= 0;
        sum <= din[(m+n)*k-1:(m+n)*(k-1)];
    end
end
else begin
    counter <= counter+1;
    sum <= sum + sreg_sum_connect;
    if (counter == k-1) ready <=1;
end
end

endmodule
```

ALU

```verilog
module alu #(
    parameter n=4
) (
    input signed [n-1:0] A,B,
    input [3:0] opcode,
    output reg [n-1:0] Y,
    output reg [4:0] status //{Parity, Overflow, Negative, Zero, Carry-out}
);

reg abs_overflow ;
reg [$clog2(n):0] i;



always @(A,B,opcode) begin
  Y = 0;
  abs_overflow = 0;
  status = 0 ;


  case (opcode)
      0: {status[0],Y} = A-B; //A-B
      1: {status[0],Y} = A+B; //A+B
      2: {status[0],Y} = A-1; //A-1
      3: {status[0],Y} = A+1; //A+1
      4: begin    //|A|
          if (A[n-1]) begin
              {abs_overflow,Y} = -A;
              end
              else Y=A;
      end
      5: Y = ~A; //not a
      6: Y = A & B; // A and B
      7: Y = A | B; // A or B
      8: Y = A^B; // A XOR B
      9: Y = A <<< B; // Arithmetic left sift A for B bits
      10: Y = A >>> B; // Arithmetic right sift A for B bits
      11: Y = A >> B;// Right sift A for B bits
      12: for (i=0; i<n; i=i+1) Y[i] = A[(i-B)%n]; // Circular left sift A for B bits

      13: for (i=0; i<n; i=i+1) Y[i] = A[(i+B)%n]; // Circular right sift A for B bits
      14: Y= 0; //0
      15: Y= 1; //1
  endcase



  status[4] = ^Y; //parity
  status[3] = abs_overflow |(~Y[n-1]&A[n-1]&B[n-1]) | (Y[n-1]&(~A[n-1])&(~B[-1]));//
overflow
  status[2] = A>B;//negative
  status[1] = Y==0?1:0; //zero
end

endmodule
```

```
module alu_tb #(
    parameter alu_size = 4,
    parameter fileout = "alu_tb_result.txt"
);

reg [alu_size-1:0] A, B;
reg [4:0] opcode;
wire [alu_size-1:0] Y;
wire [4:0] status;

integer i,f;


alu #(.n(alu_size)) dut(
    .A(A),
    .B(B),
    .Y(Y),
    .opcode(opcode),
    .status(status)
);

initial begin
f = $fopen(fileout,"w");

for (opcode=0;opcode<16;opcode=opcode+1)begin
    for(i=0;i<10;i=i+1)begin
    A=$random;
    B=$random;
    #5;
    $fwrite(f,"%d %d %d %d \n", opcode, A, B, Y );
    #5;
    end
end

end



endmodule
```

Το test bench αυτό εξάγει έντελα και αντίστοιχα αποτελέσματα στο αρχείο “alu_tb_result.txt”. Τα περιεχόμενα αυτού του αρχείου ύστερα ελέγχονται από το python script “test_alu_results.py” που ακολουθεί

```
import sys

A=[]
B=[]
opcode=[]
Y=[]

operation = {
    0: "A-B                                 :-:",
    1: "A+B                                 :-:",
    2: "A-1                                 :-:",
    3: "A+1                                 :-:",
    4: "|A|                                 :-:",
    5: "not A                               :-:",
    6: "A and B                             :-:",
    7: "A or B                              :-:",
    8: "A ^ B                               :-:",
    9: "A <<< B                             :-:",
    10:"A >>> B                             :-:",
    11:"A >> B                              :-:",
    12:"A left Circular shift by B Bits     :-:",
    13:"A right Circular shift by B Bits    :-:",
    14:"0                                   :-:",
    15:"1                                   :-:"
}

with open(sys.argv[1]) as f:
    for line in f:
        nums = [int(n) for n in line.split()]
        opcode.append(nums[0])
        A.append(nums[1])
        B.append(nums[2])
        Y.append(nums[3])

    for i in range(len(opcode)):
        if opcode[i]==0:
            expected = A[i]-B[i]


        elif opcode[i]==1:
            expected = A[i]+B[i]

        elif opcode[i]==2:
            expected = A[i]-1

        elif opcode[i]==3:
            expected = A[i]+1

        elif opcode[i]==4:
            if (A[i]<0) : expected = -A[i]
            else : expected = A[i]

        elif opcode[i]==5:
            expected = ~A[i]

        elif opcode[i]==6:
            expected = A[i] & B[i]

        elif opcode[i]==7:
            expected = A[i] | B[i]

        elif opcode[i]==8:
            expected = A[i] ^ B[i]


        elif opcode[i]==9:
            expected = A[i] << B[i]
```

. . . συνέχεια στην επόμενη σελίδα

```
( . . .)

        elif opcode[i]==9:
            expected = A[i] << B[i]

        elif opcode[i]==10:
            expected = A[i] >> B[i]

        elif opcode[i]==11:
            expected = A[i] >> B[i]


        elif opcode[i]==12:
            lala=1

        elif opcode[i]==13:
            lala=1

        elif opcode[i]==14:
            expected = 0

        elif opcode[i]==15:
            expected = 1




        if expected == Y[i]:
            print("✅ Line ",i+1," OK    :", "Operation", operation[opcode[i]]," A:
",A[i]," B: ",B[i], " Y: ", Y[i])
        else:
            print("❌ Line ",i+1," ERROR :", "Operation", operation[opcode[i]]," A:
",A[i]," B: ",B[i], " Y: ", Y[i])
            error=1

if error==1 :
    print("The test failed!")
else:
    print("The test was succesful!. Μπράβο μωρή Φιλενάδα <3")
```

[Προσοχή, το test bench περιέχει λάθη, με το test bench που δίνεται, η alu αποδεικνύεται ότι είναι πλήρως λειτουργική.]