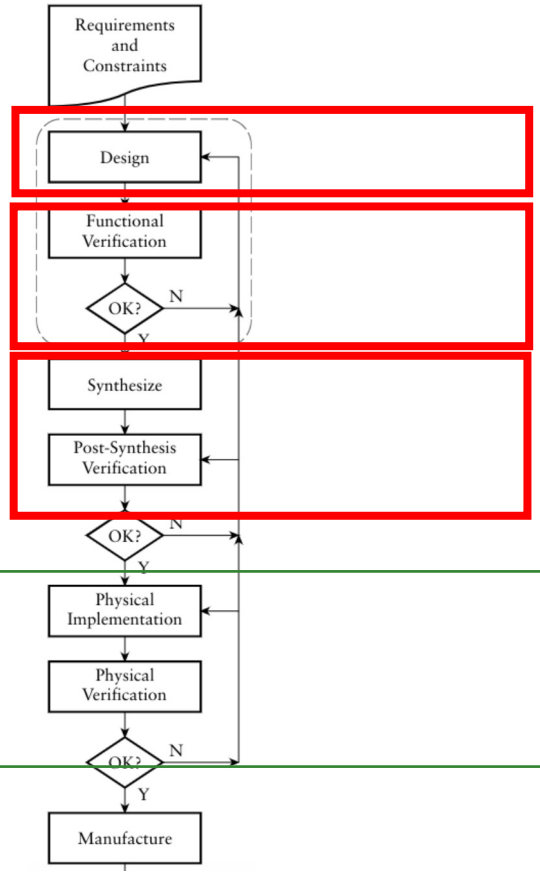
Digital Circuit Design

Li Bai

Overview – Hardware System:

- DE10-Lite Hardware: An alternative FPGA board to run your design
- Basys 3: another FPGA board which can run your design
- Temple ENGR 701 PC's to load sof files to DE10-Lite boards – one to one

Understand Designs (pg. 27)



Design under testing (DUT)

Unit under testing (UUT)

1. Testbench (verification)
2. Timing diagram (verification)

Create an account in EDA Playground

Wrapper implementation

Download sof, bit file on FPGA

Vivado --→ logic sim

Quartus --→ model sim

EDA playground

https://edaplayground.com/

Homework #3 (in group)

Problem 3: Sketch block diagrams and schematics as appropriate for the two SystemVerilog modules below:

```
module one_a (output logic z, input logic a, input logic b, input logic c);
        assign z = a & (b ^ c);
endmodule: one_a


module two_a (output logic m, input logic n, input logic p);
        logic w;
        one_a u1 (.z(m), .a(w), .b(n), .c(p));
        one_a u2 (.z(w), .a(n), .b(p), .c(1'b0));
endmodule: two_a
```

Encoding and decoding

How to represent information into digital data (encoding)

Example

EXAMPLE 2.8 Devise a binary code for the state of a road traffic light. The possible states are red, yellow and green.

SOLUTION Since there are three possible values to represent, we can use a two-bit binary code with one code word unused. One possible code is

red: (0, 0) yellow: (0, 1) green: (1, 0)

In this case, the code word (1, 1) is unused.

$$\lceil \log_2(N) \rceil = \left\lceil \frac{\ln(N)}{\ln(2)} \right\rceil$$

Using Vectors for Binary Code

- wire [4:0] a;
  - a=5'b10110 => a[0]=0 and a[4]=1

https://edaplayground.com/x/ZHiJ

- wire [1:3] m;
  - m =3'b100 => m[1]=1 and m[3]=0

Parity bit

Original Data	Even Parity	Odd Parity
0 0 0 0 0 0 0 0	0	1
0 1 0 1 1 0 1 1	1	0
0 1 0 1 0 1 0 1	0	1
1 1 1 1 1 1 1 1	0	1
1 0 0 0 0 0 0 0	1	0
0 1 0 0 1 0 0 1	1	0

ep_3bit - EDA Playground

Can only detect one bit error, but not knowing which one is wrong

Hamming (7,4) code - Lab 2 next week

Hamming Code (Truth table)

d[4]	d[3]	d[2]	d[1]	p1	p2	p3

Use this example ep_3bit - EDA Playground to create a truth table

```
p1 = d[1] xor d[2] xor d[4]

p2 = d[1] xor d[3] xor d[4]

p3 = d[2] xor d[3] xor d[4]
```

```
e[7:1] = {d[4], d[3], d[2], p3, d[1], p2, p1}
```

```
pc1 = p1 xor d[1] xor d[2] xor d[4] -

pc2 = p2 xor d[1] xor d[3] xor d[4] -

pc3 = p3 xor d[2] xor d[3] xor d[4] -
```

Verification in testbench on edaplayfound

- Inline verification

```
    // this is a command to dump the simulation info
    $dumpfile("dump.vcd");
    $dumpvars(0,uut);   // and which signals to save - all of them
below uut

    d_stim = 3'b000;    // set the input stimulus values to 000
    #10 // wait a little time
    $display("Testing input pattern: %3b", d_stim);
    if (f_sim !== 0)
      $display("Mismatch: input %3b, output expected %b, output
simulated %b",
               d_stim, 1'b0, f_sim);
    d_stim = 3'b001;    // set the input stimulus values to 000
    #10 // wait a little time
    $display("Testing input pattern: %3b", d_stim);
    if (f_sim !== 0)
      $display("Mismatch: input %3b, output expected %b, output
simulated %b",
               d_stim, 1'b1, f_sim);
    d_stim = 3'b010;    // set the input stimulus values to 000
    #10 // wait a little time
    $display("Testing input pattern: %3b", d_stim);
    if (f_sim !== 0)
      $display("Mismatch: input %3b, output expected %b, output
simulated %b",
               d_stim, 1'b1, f_sim);
    d_stim = 3'b011;    // set the input stimulus values to 000
    #10 // wait a little time
    $display("Testing input pattern: %3b", d_stim);
    if (f_sim !== 0)
      $display("Mismatch: input %3b, output expected %b, output
simulated %b",
               d_stim, 1'b0, f_sim);
    d_stim = 3'b100;    // set the input stimulus values to 000
```

- Check design.sv, tb_design.sv, tb_design.txt

```
module and_3_inputs (output logic f, input logic a, b, c);
    // this is the design to test
    assign f = a & b & c;
endmodule
```

and_3_input.sv

```
// instantiate the unit under test
and_3_inputs uut (.f(sim_out), .a(data_in[2]), .b(data_in[1]),
    .c(data_in[0]));

initial begin
    $dumpfile("tb_and_3_inputs.vcd");
    $dumpvars();

    mm_count = 0;   // zero mismatch count

    // read all of the test vectors from a file into
    // array: test_vectors
    $readmemb("tb_and_3_inputs.txt", test_vectors);

    for (i=0; i<ROWS; i=i+1) begin
        // read each vector (row) into the input data and
```

tb_and_3_input.sv

```
000_0
001_0
010_0
011_0
100_0
101_0
110_0
111_1
```

tb_and_3_input.txt

https://edaplayground.com/x/8HLE

VCD wave in EDAplayground

```
and_3_inputs uut (.f(f_sim), .d(d_stim));
```

https://edaplayground.com/x/8HLE

system Verilog with logic

```
$dumpfile("dump.vcd");
$dumpvars(0,uut);  // and which signals to save - all of them below uut
```

Verilog syntax

```
always_comb begin
    f = a^b|c;
end
```

```
assign f=a^b|c;
```

Problem 4

Problem 4:

The module below tests your knowledge of SystemVerilog ***if*** and ***case*** It encodes a three input signals: ***a***, ***b*** and ***c***, into a three output signals: ***x***, ***y*** and ***z***. Complete the truth table.

```
module problem (output logic x, y, z, input logic a, b, c);

        always_comb begin
                {x, y, z} = 3'b101;
                case ({b,c})
                        2'd2: x = 1'b0;
                        2'd3: y = 1'b1;
                        default: z = 1'b0;
                endcase
                if (a == 1'b1) begin
                        y = b ^ c;
                        case ({a,b,c})
                                3'd5: y = 1'b0;
                                3'd3: z = 1'b0;
                                default: ;
                        endcase
                end
        end

endmodule
```

x	y	z	a	b	c

https://www.edaplayground.com/x/ubNS