Laborator S5 AC

5.1 Scrierea fișierelor testbench în Verilog

P.5.1.1 Redactați, utilizând limbajul Verilog, un testbench pentru verificarea exhaustivă a unui multiplexor 4-la-1 numit ***mux_2s_1b***.

Soluție:

```
module mux_2s_1b_tb (
        output reg [3:0] d,
        output reg [3:0] s,
        output o
            );
mux_2s_1b  DUT ( .d(d), .s(s), .o(o) );
integer i;
initial begin
{s,d} = 6'd0;
for ( i=0 ; i < 64 ; i = i+1 )
#50 {s,d} = i[5:0];
end
endmodule
```

P.5.1.2 Construiți, folosind limbajul Verilog, un circuit decodificator 2-la-4 cu o intrare de validare, notată cu ***e***. Dacă ***e*** este inactiv, toate ieșirile vor fi setate pe 0, în caz contrar ieșirea selectată va fi setată pe 1.

a) Desenați și implementați arhitectura detaliată a decodificatorului folosind porți ***AND***.

b) Redactați un testbench pentru verificarea exhaustivă a modulului ***dec_2x4***:

a) ***Soluție:***

```
module dec_2x4  (
        input [1:0]s,
        input en,
        output [3:0] o
                );
assign o[0] = en & (~s[1]) & (~s[0]);
assign o[1] = en & (~s[1]) &   s[0];
assign o[2] = en &   s[1]  & (~s[0]);
assign o[3] = en &   s[1]) &   s[0];
endmodule
```

b) ***Soluție:***

```
module dec_2x4_tb (
        output reg [1:0] s,
        output reg en,
        output [3:0] y
            );
dec_2x4  DUT ( .s(s), .en(en), .y(y) );
initial begin
en=1'd0;
#80 en=1'd1;
end
integer i;
initial begin
s = 2'd0;
for ( i=0 ; i < 8 ; i = i+1 )
#20 s = i[1:0];
end
intial begin
#200 $finish;
endmodule
```

P.5.2.1 Redactați, folosind limbajul Verilog, un modul ***parametrizat*** pentru un registru cu încărcare paralelă pe 8 biți, cu o linie de load (***ld***) și o intrare adițională ***clr*** activă pe 1.

```
module rgst # ( parameter w = 8,
                parameter iv={w{1'b0}} )
    ( input [w-1:0]d,
      input ld, clr, clk, rst_b,
      output reg [w-1:0]q );
always @ ( posedge clk, negedge rst_b) begin
if (!rst_b) q <= iv;
else if (clr) q <=iv;
else if (ld) q <=d;
endmodule
```

P.5.2.2 Construiți, folosind limbajul Verilog, un ***register file*** 4x8 utilizând instanțe ale modulului ***dec_2x4***, modulului parametrizat ***rgst*** și ale modulului ***mux_2s_8b***.

```
module Register_File_4X8  (
input [7:0] wr_data,
input [1:0] wr_addr, rd_addr,
input wr_e, clk, clr, rst_b,
output [7:0] rd_data
);
wire [3:0] w;
wire [7:0] q0, q1, q2, q3;
dec_2x4 decoder ( .s(wr_addr), . e(wr_e), o(w) );


rgst # ( .w(8) ) register1 ( .d(wr_data), .ld(w[0]),
.clk(clk), .rst_b(rst_b), .clr(clr), .q(q0) );
rgst # ( .w(8) ) register2 ( .d(wr_data), .ld(w[1]),
.clk(clk), .rst_b(rst_b), .clr(clr), .q(q1) );
rgst # ( .w(8) ) register3 ( .d(wr_data), .ld(w[2]),
.clk(clk), .rst_b(rst_b), .clr(clr), .q(q2) );
rgst # ( .w(8) ) register4 ( .d(wr_data), .ld(w[3]),
.clk(clk), .rst_b(rst_b), .clr(clr), .q(q3) );


mux #  ( .k(8) ) multiplexer ( .s(rd_addr), .d0(q0),
.d1(q1), .d2(q2), .d3(q3), .o(rd_data) );


endmodule
```