Laborator AC – Săptămâna 4

Utilizarea blocurilor always secvențiale

P.4.1 Implementați, utilizând Verilog, un modul pentru un circuit basculant de tip ***D*** (flip-flop), având o linie de reset asincronă, activă pe low. Folosiți un bloc always secvențial pentru descrierea comportamentală a circuitului.

Soluție:

```
module dff_ar  (
        input d, clk, rst_b,
        output reg q
            );
always @ (posedge clk, negedge rst_b) begin
        if (!rst_b) q <= 1'b0;
          else q <= d;
    end
endmodule
```

P.4.2 Implementați, utilizând Verilog, un modul pentru un circuit basculant de tip ***T*** (flip-flop), caracterizat prin relația: ***Q(t+1) = T ^ Q(t)*** și având linia de reset asincronă, activă pe low. Folosiți un bloc always secvențial pentru descrierea circuitului din enunț.

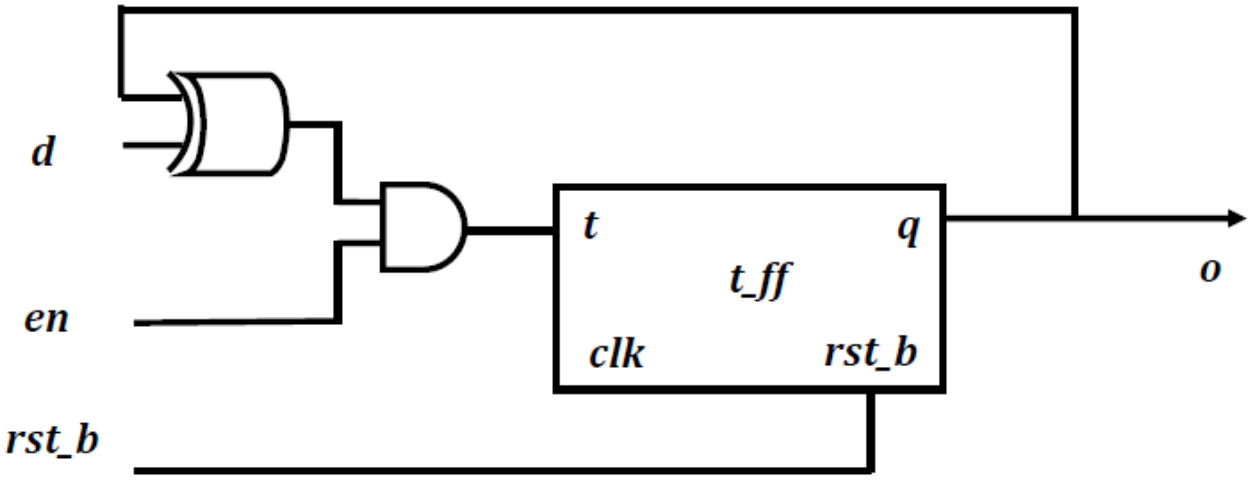
Soluție:

```
module tff_ar  (
        input t, clk, rst_b,
        output reg q
            );
always @ (posedge clk, negedge rst_b) begin
        if (!rst_b) q <= 1'b0;
          else q <= t ^ q;
    end
endmodule
```

P.4.3 Implementați, folosind limbajul Verilog, arhitectura circuitului secvențial prezentată mai jos uzitând :

a) o instanță pentru varianta modificată de ***tff_ar***.

b) un bloc always secvențial.



a) ***Soluție:***

```
module tff_ar_modified  (
        input d, en, clk, rst_b,
        output o
            );
tff_ar t1 ( .t((d^o) & en),
.clk(clk),
.rst_b(rst_b),
.q(o));
endmodule
```

b) *Soluție:*

```
module tff_ar_modified  (
        input d, en, clk, rst_b,
        output o
            );
always @ (posedge clk, negedge rst_b) begin
        if (!rst_b) o <= 1'b0;
          else if (en) q <= d ^ o;
    end
endmodule
```

P.4.4 Proiectați, utilizând Verilog, un registru serial de deplasare la dreapta pe 4 biți, folosind:

a) instanțe ale modulului ***dff_ar*** implementat la P.4.1.

b) un bloc always secvențial.

a) ***Soluție:***

```
module Right_Shift_Register_4b  (
    input shin,
    input clk, rst_b,
    output [3:0] q
);

dff_ar dff3 ( .d(shin), .clk(clk), .rst_b(rst_b), .q(q[3]) );
dff_ar dff2 ( .d(q[3]), .clk(clk), .rst_b(rst_b), .q(q[2]) );
dff_ar dff1 ( .d(q[2]), .clk(clk), .rst_b(rst_b), .q(q[1]) );
dff_ar dff0 ( .d(q[1]), .clk(clk), .rst_b(rst_b), .q(q[0]) );

endmodule
```

b) *Soluţie:*

```
module Right_Shift_Register_4b (
        input shin,
        input clk, rst_b,
        output reg [3:0] q
            );
always @ (posedge clk, negedge rst_b) begin
        if (!rst_b) q <= 4'b0000;
          else q <= {shin, q};
    end
endmodule
```

P.4.5 Proiectați, utilizând Verilog, un registru paralel cu linie de ***load*** (ld), pe 2 biți, folosind unitatea ***dff_ar*** implementată la ***P.4.1***.

Soluție:

```
module Parallel_Load_Register_2b  (
input [1:0]d,
input ld, clk, rst_b,
output [1:0] q
);

wire f, g ;

mux_1s_1b m1 ( .d0(q[1]), .d1(d[1]), .s(ld), .o(f) );
mux_1s_1b m0 ( .d0(q[0]), .d1(d[0]), .s(ld), .o(g) );
dff_ar dff1 ( .d(f), .clk(clk), .rst_b(rst_b), .q(q[1]) );
dff_ar dff0 ( .d(g), .clk(clk), .rst_b(rst_b), .q(q[0]) );

endmodule
```