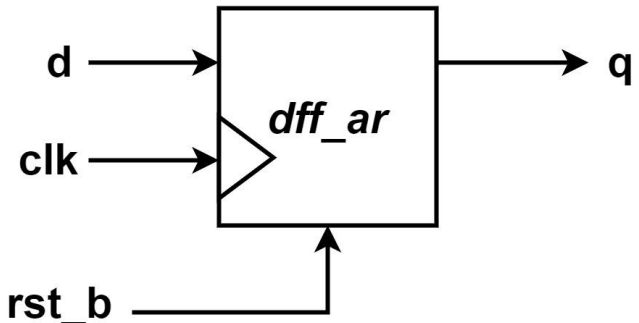
Week 4 – Using always and initial blocks

P.4.1 Implement, using Verilog, a module for a ***D***-type flip-flop (flip-flop) circuit having an asynchronous, active-low ***reset line***. Use a ***sequential always block*** for the behavioral description of the circuit.

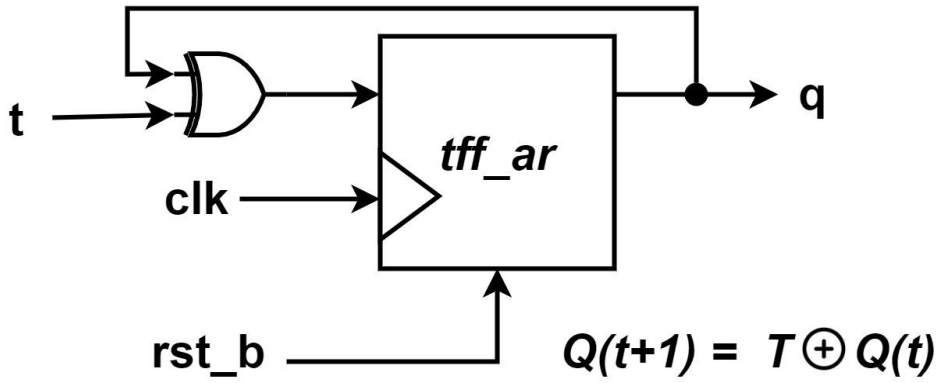
Solution:

```
module dff_ar (
    input d,
    input clk,
    input 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 Implement, using Verilog, a module for a ***T***-type flip-flop circuit, characterized by the relationship: ***Q(t+1) = T ^ Q(t)*** and having the asynchronous ***reset line***, active low. Use a ***sequential always block*** to describe the circuit in the statement.

Solution:

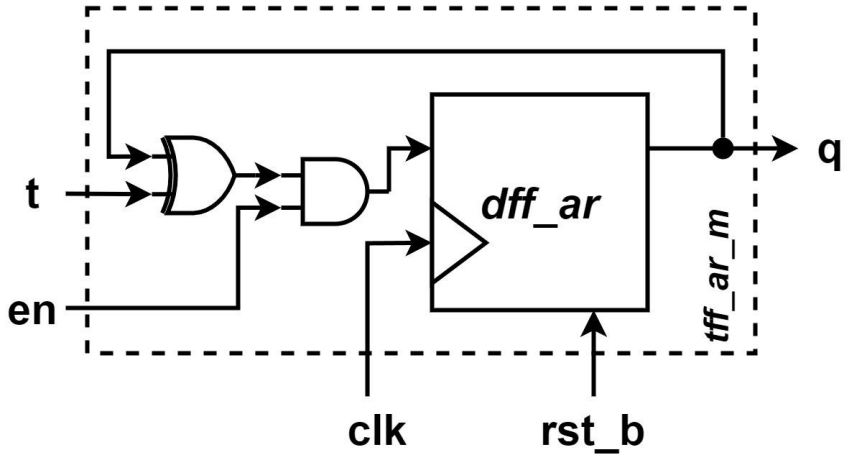



```
module tff_ar (
    input t, clk, rst_b,
    output reg q
);
always @(posedge clk, negedge rst_b) begin
if (!rst_b) q <= 1'b0;
else q <= t ^ q;
end
endmodule
```

P.4.3 Design, using Verilog, the sequential circuit architecture shown below using:

a) a ***dff_ar*** instance for the modified variant in the figure.

b) a ***sequential always block***.



Solution: a)

```
module tff_ar_mod (
     input t, en, clk, rst_b,
     output q
);
wire w1;
assign w1 = (t^q) & en;


dff_ar t1 (.d(w1), .clk(clk), .rst_b(rst_b), .q(q) );
endmodule
```

Solution: b)

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

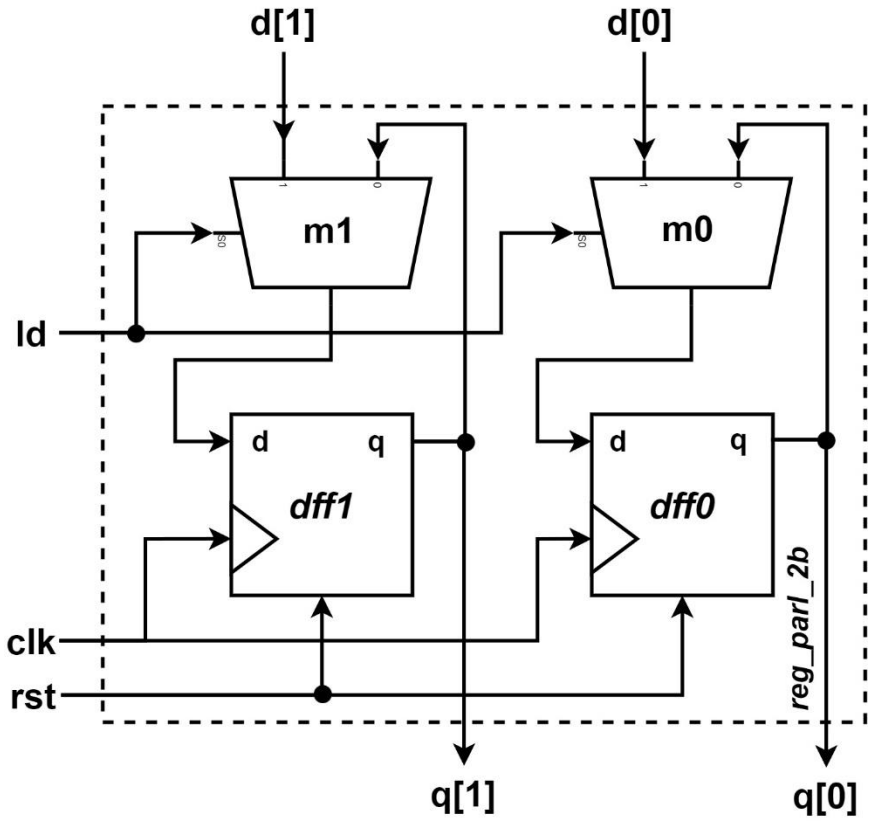
P.4.4 Design, using Verilog, a 2-bit parallel load register with a clock (***clk***), asynchronous reset line (***rst***), a load line (***ld***), a 2-bit ***d*** input, and a 2-bit output denoted by ***q***.

a) Draw the detailed architecture of the 2-bit parallel load register, ***reg_parl_2b***.

b) Construct, by instantiation, the architecture designed in subpoint ***a)***.

```
module reg_parl_2b    (
     input clk, rst,ld,
     output  [3:0] q   );
     wire w1,w2;
//Layer 1
mux_1s_1b m1 (.d1(d[1]), .d0(q[1]), .s(ld), .o(w1);
mux_1s_1b m0 (.d1(d[0]), .d0(q[0]), .s(ld), .o(w2);
// the mux_1s_1b module is reused from previous laboratory
//Layer 2
dff_ar dff1 (.d(w1), .clk(clk), .rst(rst), .q(q[1]));
dff_ar dff0 (.d(w2), .clk(clk), .rst(rst), .q(q[0]));
endmodule
```

P.4.4 a)



P.4.5 Design, using Verilog, a testbench for exhaustive checking of the ***mux_1s_1b*** module. The module's name will be ***mux_1s_1b_tb***.

Solution:

```
module mux_1s_1b_tb (
    output reg s,d1,d0,
    output o
);

// Instantiation
mux_1s_1b DUT (.d0(d0), .d1(d1), .s(s), .o(o));

// Initial begin block
integer i;
initial begin
{s,d1,d0} = 3'b0;
for (i = 0; i < 8; i = i + 1)
# 50 {s,d1,d0} = i[2:0];
end
endmodule
```

P.4.6 Design, using Verilog, a testbench for non-exhaustive checking of the ***dff_ar*** module. The module's name will be ***dff_ar_tb***. The clock signal has ***T*** = 20 ps.

```
module dff_ar_tb (
     output reg d,clk,rst,
     output q       );
// Instantiation
dff_ar DUT (.d(d), .clk(clk), .rst(rst), .q(q));
initial begin // Clock signal generation
clk = 1'b0;
forever #10 clk = ~clk;
end
initial begin // Reset signal generation
rst = 1'b0;
#5 rst = 1'b'1;
#15 rst = 1'b'0;
#10 rst = 1'b1;
initial begin // Data signal generation
d = 1'b0;
#10 d = 1'b'1;
#15 d = 1'b'0;
#10 d = 1'b1;
end
endmodule
```