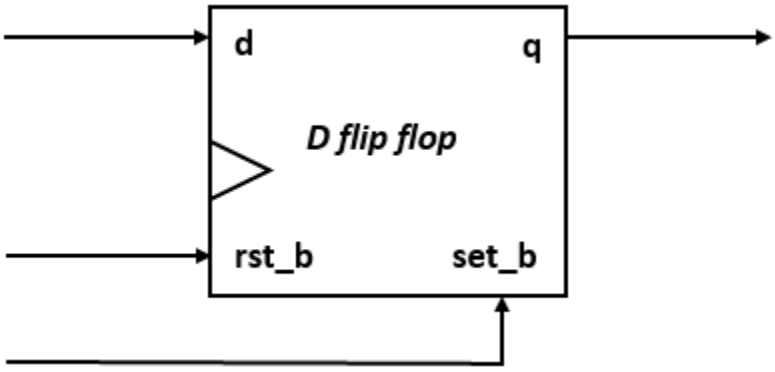
Week 8 CA – Error Detection Architectures

Built-In Self-Test Architectures

P.8.1 Design a type ***D flip-flop*** with an additional asynchronous set line, as described in the given architecture:



a) Implement, using Verilog language, the ***dff_ar*** module attached to the given design.
b) Write a testbench that verifies the functionality of the ***dff_ar*** module by using the non-exhaustive checking procedure.

```
// Solution a):
module dff_ar (
input d, rst_b, set_b, clk,
output reg q
);
always @(posedge clk, negedge rst_b) begin
if (set_b==0)
q <= 1'b1;
else
if (rst_b==0)
q <= 1'b0;
else q <= d;
end
endmodule
// Solution b):
module dff_ar_tb (
output reg d, rst_b, set_b, clk,
output q
);
initial begin
clk=1'b0;
repeat (4) #20 clk = ~clk;
end
initial begin
rst_b=1'b0;
#5 rst_b = 1'b1;
end
```

```
initial begin
set_b=1'b1;
#60 set_b = 1'b0;
#10 set_b=1'b1;
end
initial begin
d=1'b0;
#40 d = 1'b1;
#20 d = 1'b0;
end
endmodule
```

P.8.2. Design a 4-bit serial right shift register with feedback, the outputs of which are generated in the given truth table as support. This requires the following aspects:

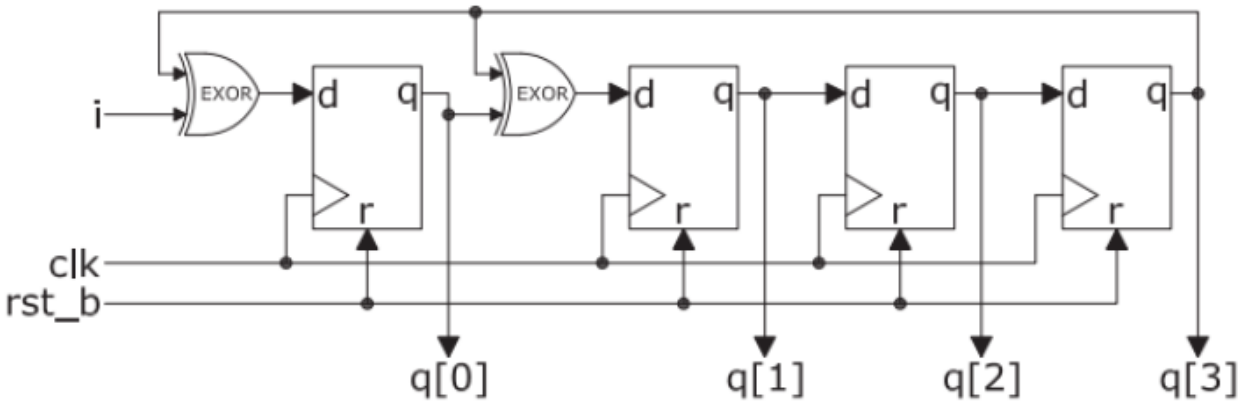
a) Fill in the truth table with the missing 4-bit vectors and determine the periodicity of the output sequence.

b) Implement, using Verilog language, the 4-rank Linear Feedback Shift Register (LFSR) unit, using instances of the dff_ar module designed in ***P.9.1***.

The solution for a) is provided in the laboratory material.

```
module LFSR (
   input clk, rst_b, set_b,
   output [3:0] q
);
// Instantiation
dff_ar dff0 ( .d(q[3]),
              .clk(clk),
              .rst_b (1'b1),
              .set_b (set_b),
               .q (q[0]) );
dff_ar dff1 ( .d(q[3] ^ q[0]),
              .clk(clk),
              .rst_b (1'b1),
              .set_b (set_b),
               .q (q[1]) );
dff_ar dff2 ( .d(q[1]),
              .clk(clk),
              .rst_b (1'b1),
              .set_b (set_b),
               .q (q[2]) );
dff_ar dff3 ( .d(q[2]),
              .clk(clk),
              .rst_b (1'b1),
              .set_b (set_b),
              .q (q[3]) );
endmodule
```

P.8.3. Implement a 4-bit Single Input Signature Register (***SISR***) using Verilog language, as shown below.

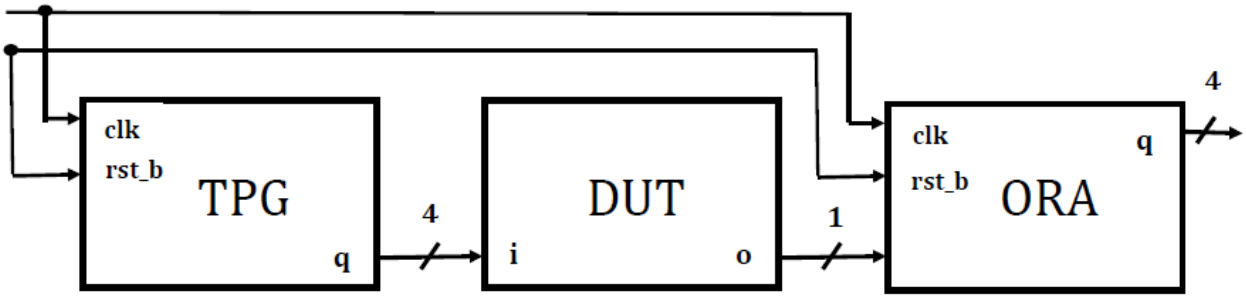



```
module SISR (
    input i, clk, rst_b, set_b,
    output [3:0] q
);
// Instantiation
dff_ar dff0 ( .d(i ^q[3]),
                .clk(clk),
                .rst_b (rst_b),
                .set_b (1'b1),
                 .q (q[0]) );
dff_ar dff1 ( .d(q[3] ^ q[0]),
                .clk(clk),
                .rst_b (rst_b),
                .set_b (1'b1),
                 .q (q[1]) );
```

```
dff_ar dff2 ( .d(q[1]),
              .clk(clk),
              .rst_b (rst_b),
              .set_b (1'b1),
               .q (q[2]) );
dff_ar dff3 ( .d(q[2]),
              .clk(clk),
              .rst_b (rst_b),
              .set_b (1'b1),
              .q (q[3]) );
endmodule
```

P.8.4. Design, using Verilog language, a Built-In Self-Test (***BIST***) architecture that uses the following components in its structure:

- The 4-bit ***LFSR*** implemented in ***E10.2*** will replace the TPG unit.
- The ***DUT*** will be replaced by a combinational circuit that provides a bit of ***1*** if the input ***i*** of the block is a multiple of 3, and ***0*** otherwise.
- The 4-bit ***SISR*** will replace the ***ORA*** unit in the given diagram.




```
module BIST (
   input clk, rst_b, set_b,
   output [3:0] q
);
wire [3:0] w1;
wire w2;
// Instantiation
LFSR TPG (   .clk(clk),
             .rst_b (1'b1),
             .set_b (set_b),
              .q (w1)
);
MUL_3 DUT ( .i (w1),
              .o (w2)
);
SISR ORA  (  .clk(clk),
             .rst_b (rst_b),
             .set_b (1'b1),
             .i(w2),
             .q (q)
);
endmodule
```