Circuit Reliability and Testing

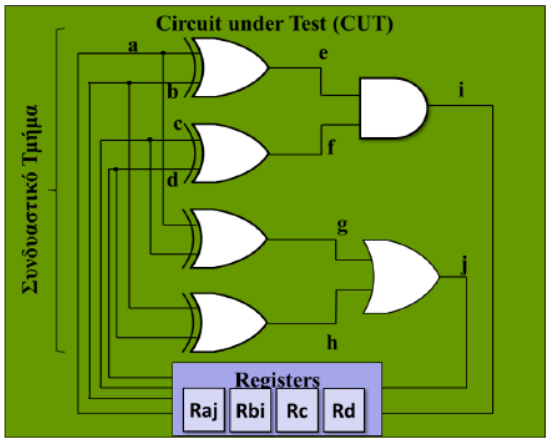
Scan Testing

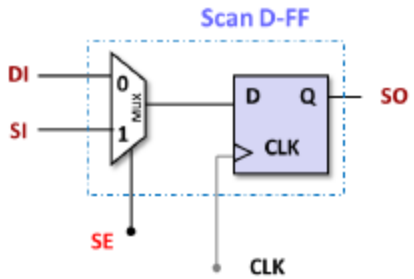
Name: Patsaoglou Panteleimon

AM: 5102

- **Assignment**

The circuit-under-test CUT diagram is given below. The circuit module should be firstly built using RTL Verilog and the functionality should be verified using a Testbench. A scan chain should be implemented for testability using the **Scan D-FF module** which has a typical D-FF and the input is selected via the MUX using **(SE).**





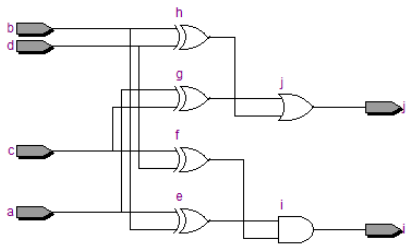
- **Scan Test Flow**

From the diagram we can understand that the scan chain is comprised of 4 **Scan D-FF modules** and the test vectors are inputted via **SI** and **SE** set to logic 1**.** Once a test vector is set after 4 clock cycles and **Control Points** set the inputs of the **CUT, SE is set to logic 0** to perform a capture of the observability points which are going to be stored in the Scan D-FFs in the next clock cycle. SE is set back to logic 1 to shift the next test vector into the scan chain while we simultaneously get the response of the previous capture in the **SO**.

- **CUT RTL in Verilog**

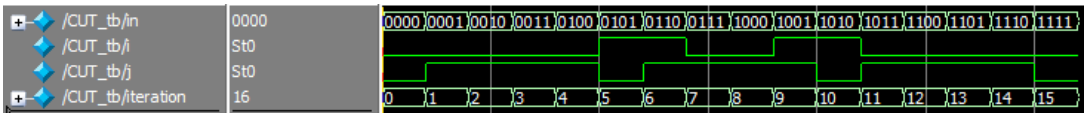
```
module CUT(a,b,c,d,i,j);

input a,b,c,d;
output i,j;
wire e,f,g,h;

assign e = a ^ b;
assign f = c ^ d;
assign g = a ^ c;
assign h = b ^ d;
assign i = e & f;
assign j = g | h;

endmodule
```

- **CUT Synthesized**



- **CUT Testbench**

```
module CUT_tb();

reg[3:0] in;
wire i,j;
integer iteration;

CUT cut (a(in[3]), b(in[2]), c(in[1]), d(in[0]), i(i), j(j));

initial begin
        in = 0;
        for (iteration = 0; iteration<16; iteration = iteration + 1)
        begin
                #2
                in = in + 1;
        end
end
endmodule
```



- **Testable-Ready-CUT**

The testable ready module should have only 3 inputs (SI, SE, CLK) and 1 output (SO). On the 4-bit scan chain Raj has SI the SI of the module and each of the next Scan D-FFs has as SI the SO of the previous Scan D-FF. Also, SO of respective DFFs is setting the inputs of the CUT while SO of the last DFF (Rd) is the SO of the TRCUT module too. More precisely:

- Raj: Testability(a), Observability(j)
- Rbi: Testability(a), Observability(i)
- Rc: Testability(c), Observability(c)
- Rd: Testability(d), Observability(d)

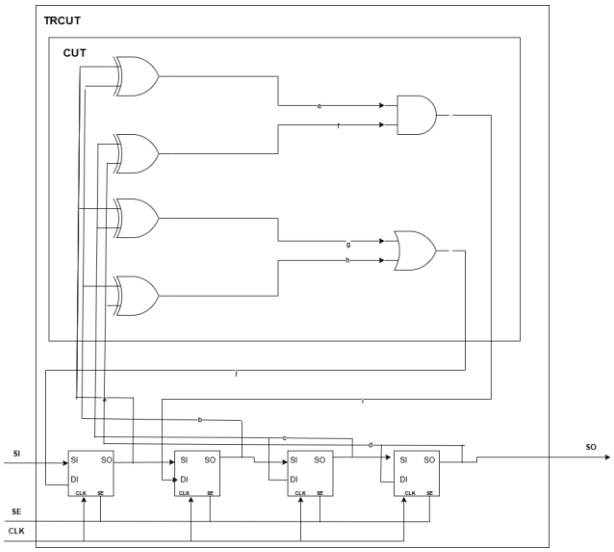
From the structure above, on the testbench after capture of each test vector we expect to see the captured response in this sequence at SO after 4 clock cycles and before the capture of the next test vector:

SO (TRCUT): d->c->i->j (first d last j in SO)

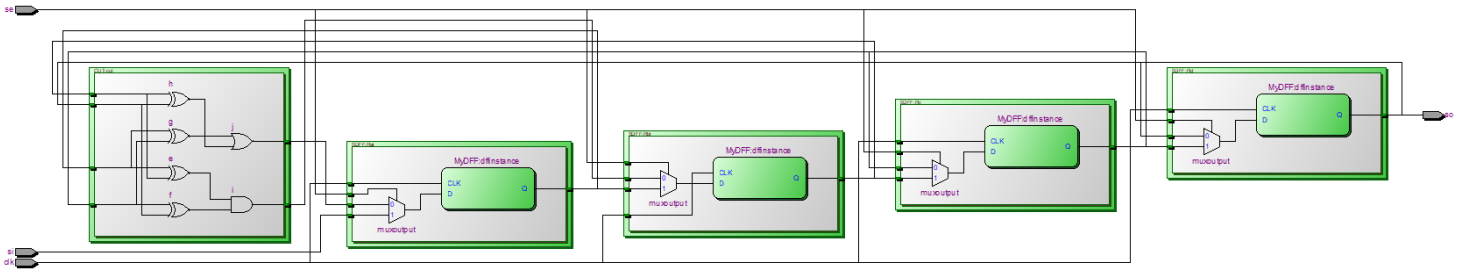
- **Testable-Ready-CUT RTL in Verilog** (1.1)

```
module TRCUT(clk,si,se,so);

input clk,si,se;
output so;

wire a,b,c,d,i,j;

assign so = d;

SDFF Raj(clk, j, si, se, a);
SDFF Rbi(clk, i, a, se, b);
SDFF Rc(clk, c, b, se, c);
SDFF Rd(clk, d, c, se, d);

CUT cut (
        .a(a),
        .b(b),
        .c(c),
        .d(d),
        .i(i),
        .j(j)
);

endmodule
```

- **TRCUT expected**



- **TRCUT Synthesized**



- **TRCUT Testbench**

Reg clk, si, se are declared and used to drive the inputs of the TRCUT. The wire **so** is used to take the output SO of the TRCUT while the reg[3:0] i is used to set the test vectors.

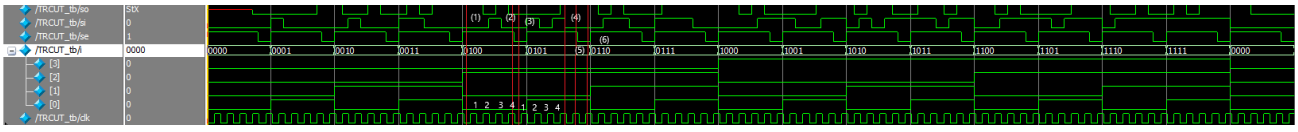
For **clock generation** a forever loop is declared, which is toggling the value of clk each 5 time-units making a 10 time-unit clock period.

In the second **begin block**, **SI** is initially set to logic 0 and **SE** to logic 1 so the serial loading of the first test vector can initiate. In the **for loop,** i is incremented on each iteration and on each clock cycle (#10**) SI is set** with **each bit of the test vector** until all **4 bits** are loaded into the chain and **capture is triggered in the next cycle** by setting SE to logic 0. **SE is set back to logic 1** so serial **loading of the next test vector can start** while the response of the observation points are shifted out from SO bus of the TRCUT.

- **TRCUT Testbench in Verilog**

```
module TRCUT_tb();

reg clk, si, se;
wire so;

reg[3:0] i;

TRCUT trcut (
        .clk(clk),
        .si(si),
        .se(se),
        .so(so)
);

initial begin
        clk = 0;
        #1;
        forever begin
                #5 clk =! clk;
        end
end

initial begin
          si = 0;
          se = 1;
          for (i = 0; i<16; i = i + 1) begin
                si = i[0];
                #10 si = i[1];
                #10 si = i[2];
                #10 si = i[3];

                #10 se = 0;
                #10 se = 1;
                end
end

endmodule
```

- **TRCUT Testbench response explanation** (1.2)



(1) Input reg i[3:0] is set to 0100 and SE to logic 1 so we can serially input the test vector.

(2) After 4 clock cycles 0100 vector (a,b,c,d) is serially loaded into the scan chain.

(3) SE is set to logic 0 before the next cycle so capture of the observation points can occur. So, in the next cycle we get d observation point and in the next clock cycles we expect to see the observation points in sequence d->c->i->j.

(4) We get the last bit, value of bit (j) in our case from the previous capture while the last bit of the next test vector is 1 cycle from getting into the chain and the next capture of the observation points can occur. From the test bench we loaded (a,b,c,d) = (0,1,0,0) and we got the expected response of (j,I,c,d) = (1,0,0,0).

(5) The last bit of the next vector is loaded into the chain.

(6) Capture of the observation points of the next test vector occurs.

- **Testing time for Fclk = 10Mhz** (1.3)

For each 4bit test vector we need 4 clock cycles to load it into the scan chain and 1 clock cycle to capture the observation points into the chain. For the complete truth table, we have 16 test vectors in total. Also, we additionally need 4 clock cycles in order to shift out the observation points of the last test vector and get the response in SO of the TRCUT.

So total time:

(1) Tclk = 1/Fclk = 100ns

(2) Ttest = (#of test vectors * (# bits per vector + 1 cycle for capture)* Tclk) + 4 * Tclk= 16*5*100ns + 4*100ns = 8.4us

If for example we had 10bit inputs, we would have 2^10 vectors to test, 10 clock cycles to shift each test vector into the chain plus 1 cycle for capture and additionally 10 clock cycles to get the response of the last capture.

N = 10:

(1) Ttest = (#of test vectors * (# bits per vector + 1 cycle for capture)* Tclk) + 4 * Tclk = 1024*11*100ns + 10*100ns = 1.1274 ms

N = 20:

(1) Ttest = (#of test vectors * (# bits per vector + 1 cycle for capture)* Tclk) + 4 * Tclk = 1,048,576*21*100ns + 20*100ns = 2202.0116 ms

N = 30:

(1) Ttest = (#of test vectors * (# bits per vector + 1 cycle for capture)* Tclk) + 4 * Tclk= 1,073,741,824*30*100ns + 40*100ns = …

N = 40:

(1) Ttest = (#of test vectors * (# bits per vector + 1 cycle for capture)* Tclk) + 4 * Tclk= 1,099,511,627,776*40*100ns + 40*100ns = ….

- **Conclusion**

While basic scan testing for small number of inputs is a feasible procedure, for higher input count circuits it is very Time-to-test Insufficient.