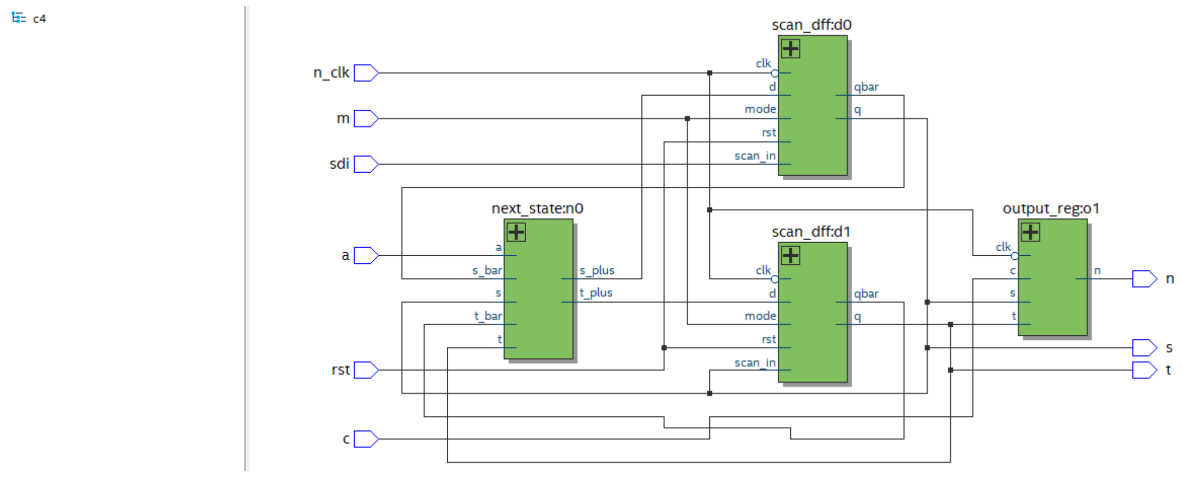
C4 Lab si3g19

Design And Test Of Finite State Machines

3.1 Functional Testing

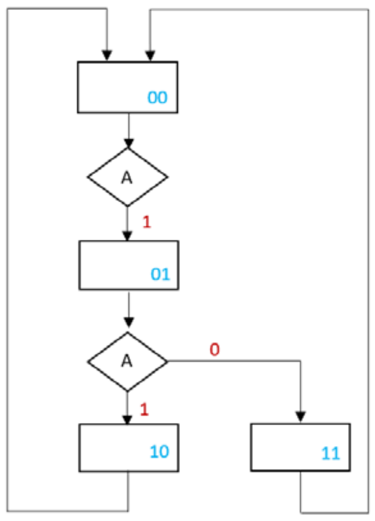


- Do your test sequences verify that the circuit has been implemented correctly?
  FPGA behaviour observed meets the expected
  Behaviour matched modelsim simulation and the circuit was implemented correctly using quartus

3.2 Fault Detection and Diagnosis

(a) Apply a stuck-at-1 fault and step through the test sequence

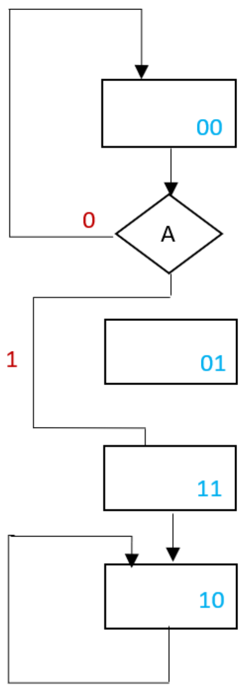
Stuck at 1:



Doesn’t remain in the 10 state when clocked

The fault was detected, can deduce fault location

(b) Apply a stuck-at-0 fault and step through the test sequence.



Gets stuck in 10 state until manually reset

The fault is detected

Can deduce fault point.

3.3 Sequential Output Logic

- With both faults disabled, verify that your previous test sequence finds no fault.

verified

Apply the stuck-at-1 fault

- Is the fault detected? If so, is it possible to diagnose the fault location?

Sequence for stuck at 1:

A=0, rst, a= 1, 1, 1;

Fault is detected by state, as cant go back to S(1) T(0)

Sequence for stuck at 0:

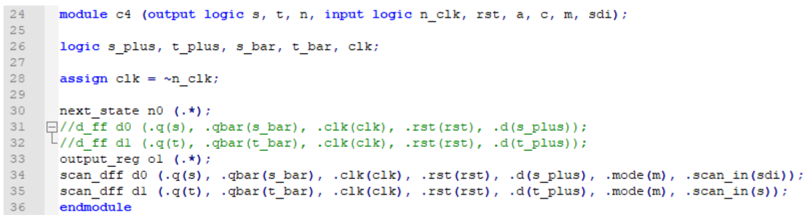
A=0, rst, a= 1, 1;

Fault is detected by state, state skips S(0) T(1)

- How effective is transition verification as a basis for testing an FSM?
  How can you verify transitions from redundant states, and what is the relative cost of doing this?
  Fault will be detected
  Cannot diagnose location of fault
- How easy is it to detect and diagnose faults in non-scan-path FSMs with and without output logic?
  Its difficult to diagnose fault in non-scan path FMSs as we can't set and observe our present state
  Can only go through the state through input.

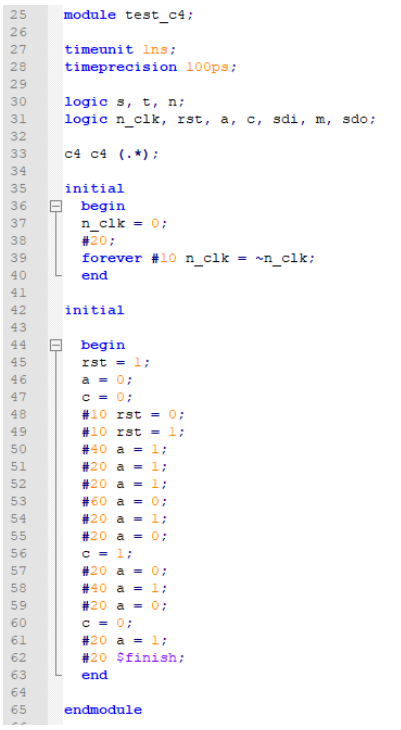
3.4 Scan Design

C4:



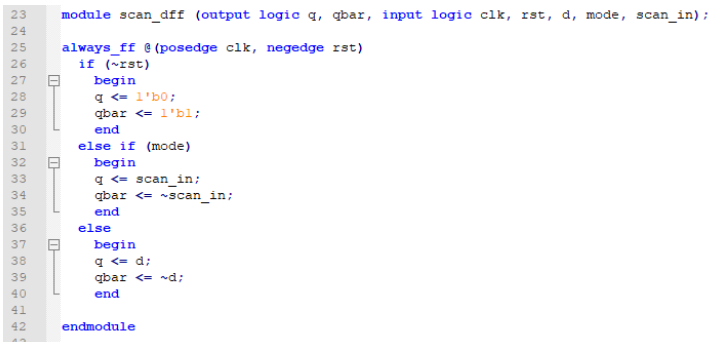
```
module c4 (output logic s, t, n, input logic n_clk, rst, a, c, m, sdi);

logic s_plus, t_plus, s_bar, t_bar, clk;

assign clk = ~n_clk;

next_state n0 (.*);
//d_ff d0 (.q(s), .qbar(s_bar), .clk(clk), .rst(rst), .d(s_plus));
//d_ff d1 (.q(t), .qbar(t_bar), .clk(clk), .rst(rst), .d(t_plus));
output_reg o1 (.*);
scan_dff d0 (.q(s), .qbar(s_bar), .clk(clk), .rst(rst), .d(s_plus), .mode(m), .scan_in(sdi));
scan_dff d1 (.q(t), .qbar(t_bar), .clk(clk), .rst(rst), .d(t_plus), .mode(m), .scan_in(s));
endmodule
```

Test_c4:



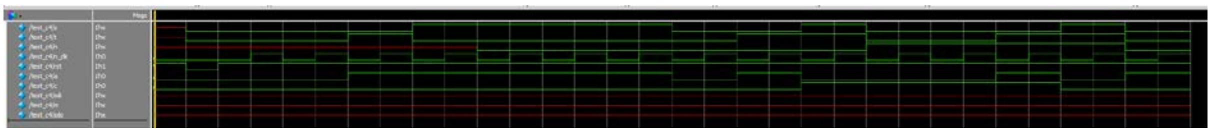
```
module test_c4;

timeunit 1ns;
timeprecision 100ps;

logic s, t, n;
logic n_clk, rst, a, c, sdi, m, sdo;

c4 c4 (.*);

initial
  begin
  n_clk = 0;
  #20;
  forever #10 n_clk = ~n_clk;
  end

initial

  begin
  rst = 1;
  a = 0;
  c = 0;
  #10 rst = 0;
  #10 rst = 1;
  #40 a = 1;
  #20 a = 1;
  #20 a = 1;
  #60 a = 0;
  #20 a = 1;
  #20 a = 0;
  c = 1;
  #20 a = 0;
  #40 a = 1;
  #20 a = 0;
  c = 0;
  #20 a = 1;
  #20 $finish;
  end

endmodule
```

Scan_diff:



```
module scan_dff (output logic q, qbar, input logic clk, rst, d, mode, scan_in);

always_ff @(posedge clk, negedge rst)
  if (~rst)
    begin
    q <= 1'b0;
    qbar <= 1'b1;
    end
  else if (mode)
    begin
    q <= scan_in;
    qbar <= ~scan_in;
    end
  else
    begin
    q <= d;
    qbar <= ~d;
    end

endmodule
```

Waveform:



Fault equation: y= (t*s) + $\bar{A}$

Correct equation: y= (s*t) + $\overline{T}$