**Load Counter Hot code**

module parallel\_to\_serial (

input wire clk,

input wire reset,

output wire empty\_o,

input wire[3:0] parallel\_i,

output wire serial\_o,

output wire valid\_o

);

logic [3:0]shift\_ff;

logic [3:0]nxt\_shift;

always\_ff @(posedge clk or posedge reset)

if(reset)

shift\_ff <= 4'h0;

else

shift\_ff <= nxt\_shift;

assign nxt\_shift = empty\_o ? parallel\_i :

{1'b0,shift\_ff[3:1]};

assign serial\_o = shift\_ff[0];

logic [2:0]cnt\_ff;

logic [2:0]nxt\_cnt;

always\_ff @(posedge clk or posedge reset)

if(reset)

cnt\_ff <= 4'h0;

else

cnt\_ff <= nxt\_cnt;

assign nxt\_cnt = (cnt\_ff == 3'h4)?3'h0 :

cnt\_ff + 3'h1;

assign empty\_o = (cnt\_ff == 4'h0);

assign valid\_o = |cnt\_ff;

endmodule

**Test Bench Code**

`timescale 1ns / 1ps

module tb\_parallel\_to\_serial ();

logic clk;

logic reset;

logic empty\_o;

logic[3:0] parallel\_i;

logic serial\_o;

logic valid\_o;

parallel\_to\_serial DAY11 (.\*);

// Clock

always begin

clk = 1'b1;

#5;

clk = 1'b0;

#5;

end

// Stimulus

initial begin

reset <= 1'b1;

parallel\_i <= 4'h0;

@(negedge clk);

reset <= 1'b0;

@(posedge clk);

for (int i=0; i<32; i=i+1) begin

parallel\_i <= $urandom\_range(0, 4'hF);

@(posedge clk);

end

$finish();

end

// VCD

initial begin

$dumpfile("day11.vcd");

$dumpvars(0, tb\_parallel\_to\_serial);

end

endmodule

**Simulation**

