COUNTER

```
module counter8bit(input clk,input reset,output reg [7:0]count);
always@(posedge clk or negedge reset)
begin
if(~reset)
count<=0;
else
count<=count+1;
end
endmodule
```

TESTBENCH

```
module counter_tb;
reg clk;
reg reset;
wire [7:0] count;
counter8bit tb(.clk(clk),
.reset(reset),
.count(count));
initial begin
clk=0;
reset=0;
#10 reset=1;
end
always
#5 clk=~clk;
endmodule
```

Constraintfile

```
create_clock -name "clk" -add -period 16.0 -waveform {0.0 8.0} [get_ports clk]
set_input_delay -clock [get_clocks clk] -add_delay 0.3 [get_ports rst]
set_output_delay -clock [get_clocks clk] -add_delay 0.3 [get_ports count]


create_clock -name clk -period 10 -waveform {0 5} [get_ports "clk"]
set_clock_transition -rise 0.1 [get_ports "clk"]
set_clock_transition -fall 0.1 [get_ports "clk"]
set_clock_uncertainty 0.01 [get_ports "clk"]
set_input_delay -max 1.0 [get_ports "rst"] -clock [get_ports "clk"]
set_output_delay -max 1.0 [get_ports "count"] -clock [get_ports "clk"]
```

SERIAL ADDER

```
module serial_adder(A, B, reset, clock, sum);
   input [7:0] A, B;
   input reset, clock;
   output [7:0] sum;
   reg [3:0] count;
   reg s, y, Y;
   wire [7:0] qa, qb, sum;
   wire run;
   parameter G = 0 , H = 1;

   shiftreg A1(A, reset, 1'b1, 1'b0, clock, qa);
   shiftreg B1(B, reset, 1'b1, 1'b0, clock, qb);
   shiftreg C1(8'b0, reset, run, s, clock, sum);
```

```
    always @(qa, qb, y) // Full Adder FSM
    begin
        case (y)
            G: begin
                s = qa[0] ^ qb[0];
                if (qa[0] & qb[0])
                    Y = H;
                else
                    Y = G;
            end
            H: begin
                s = qa[0] ~^ qb[0];
                if (~qa[0] & ~qb[0])
                    Y = H;
                else
                    Y = G;
            end
            default: Y = G;
        endcase
    end

    always @(posedge clock)
        if (reset)
            y = G;
        else
            y = Y;

    always @(posedge clock)
        if (reset)
            count = 8;
        else if (run)
            count = count - 1;

    assign run = |count;
endmodule

module shiftreg(R, L, E, W, clock, q);
    parameter n = 8;
    input [n - 1:0] R;
    input L, E, W, clock;
    output [n - 1:0] q;
    reg [n - 1:0] q;
    integer k;

    always @(posedge clock)
        if (L)
            q <= R;
        else if (E)
        begin
            for (k = n - 1; k > 0; k = k - 1)
                q[k - 1] <= q[k];
            q[n - 1] <= W;
        end
endmodule
```

TESTBENCH

```
module serial_adder_tb;
reg [7:0] A,B;
reg reset,clock;
wire [7:0] sum;
serial_adder s1(A,B,reset,clock,sum);
initial
begin
A=8'hA0;
B=8'h1F;
clock=0;
reset=1;
#10 reset =0;
end
always
#5 clock=~clock;
endmodule
```

Constraint file

```
current_design serial_adder
create_clock -name "clock" -add -period 16.0 -waveform {0.0 8.0} [get_ports clock]
set_input_delay -clock [get_clocks clock] -add_delay 0.3 [get_ports reset]
set_output_delay -clock [get_clocks clock] -add_delay 0.3 [get_ports sum]
```

```
current_design serial_adder
create_clock -name "clock" -add -period 16.0 -waveform {0.0 0.8} [get_ports clock]
set_input_delay -clock [get_clocks clock] -add_delay 0.3 [get_ports reset]
set_input_delay -clock [get_ports clock] -add_delay 0.3 [get_ports A]
set_input_delay -clock [get_ports clock] -add_delay 0.3 [get_ports B]
set_output_delay -clock [get_clocks clock] -add_delay 0.3 [get_ports sum]
```

COMMANDS

```
set_db lib_search_path {./lib}
set_db hdl_search_path {./Digital}
set_db library {./lib/slow.lib}
read_hdl counter.v
elaborate
syn_gen
syn_map
syn_opt
report_timing
report_area
report_power
```