**Verilog code for top module Soc**

module SoC (

input clk, // System clock

input reset, // System reset

output [7:0] led\_out // LED output (8-bit)

);

// Wires to connect Timer peripheral

wire [7:0] timer\_count;

wire timer\_interrupt;

// Internal memory (Single-Port RAM: 255 x 8-bit)

reg [7:0] memory [0:254];

// LED output register

reg [7:0] led\_reg;

assign led\_out = led\_reg;

// Timer Instance

Timer timer\_inst (

.clk(clk),

.reset(reset),

.count(timer\_count),

.interrupt(timer\_interrupt)

);

// State machine for interrupt response

reg [1:0] state;

reg [15:0] delay\_counter; // simple delay counter

localparam IDLE = 2'b00,

ON\_STATE = 2'b01,

DELAY = 2'b10,

OFF\_STATE= 2'b11;

always @(posedge clk or posedge reset) begin

if (reset) begin

led\_reg <= 8'h00;

memory[0] <= 8'h00;

state <= IDLE;

delay\_counter <= 16'd0;

end else begin

case (state)

IDLE: begin

if (timer\_interrupt) begin

led\_reg <= 8'hFF; // Turn on LEDs

memory[0] <= 8'h55; // Write 0x55 to memory[0]

delay\_counter <= 16'd50000; // Some delay cycles

state <= DELAY;

end

end

DELAY: begin

if (delay\_counter > 0)

delay\_counter <= delay\_counter - 1;

else

state <= OFF\_STATE;

end

OFF\_STATE: begin

led\_reg <= 8'h00; // Turn off LEDs

memory[0] <= 8'h00; // Clear memory[0]

state <= IDLE;

end

endcase

end

end

endmodule

module Timer (

input clk, // Clock input

input reset, // Reset input

output reg [7:0] count, // 8-bit count value (from 0F to 00)

output reg interrupt // Interrupt signal when count hits 00

);

always @(posedge clk or posedge reset) begin

if (reset) begin

count <= 8'h0F; // Initialize counter to 0F on reset

interrupt <= 1'b0;

end else begin

if (count == 8'h00) begin

interrupt <= 1'b1; // Raise interrupt when count reaches 00

end else begin

count <= count - 1'b1; // Decrement count

interrupt <= 1'b0; // Clear interrupt if not at 00

end

end

end

endmodule

**Verilog code for testbench**

`timescale 1ns / 1ps

module SoC\_tb;

// Testbench signals

reg clk;

reg reset;

wire [7:0] led\_out;

// Instantiate the SoC

SoC uut (

.clk(clk),

.reset(reset),

.led\_out(led\_out)

);

// Clock generation: 10ns period (100 MHz)

initial begin

clk = 0;

forever #5 clk = ~clk;

end

// Stimulus block

initial begin

// Monitor output

$monitor("Time: %0t | Reset: %b | LED: %h", $time, reset, led\_out);

// Initialize

reset = 1;

#20;

// Deassert reset

reset = 0;

// Run simulation for some time to observe behavior

#500000;

// Finish simulation

$finish;

end

endmodule

**Assembly code**

AREA RESET, CODE, READONLY

ENTRY

LDR R0, =0x40000000 ; LED GPIO

LDR R1, =0x20000000 ; RAM base

LDR R2, =0x00000010 ; Timer ISR vector

; Infinite loop

main\_loop

B main\_loop

AREA ISR, CODE, READONLY

EXPORT Timer\_Handler

Timer\_Handler

; Write FF to LED

LDR R3, =0xFF

STRB R3, [R0]

; Write 55 to RAM[0]

LDR R3, =0x55

STRB R3, [R1]

; Simple delay

MOV R4, #0xFF

delay\_loop

SUBS R4, R4, #1

BNE delay\_loop

; Clear LED

MOV R3, #0x00

STRB R3, [R0]

; Clear RAM[0]

STRB R3, [R1]

; Clear timer interrupt flag if needed

BX LR

**Results**





