LED Blinking on FPGA – Report

1. Objective

The aim of this project is to implement an RGB LED blinking system on an FPGA using Verilog. The Red, Green, and Blue (RGB) LEDs are controlled individually to create different color patterns. The FPGA will sequentially toggle the three colors at a fixed interval.

---

2. FPGA Board and Connections

This project is implemented on the VSDSquadron Mini FPGA, which is based on the Lattice iCE40 FPGA. The RGB LED consists of three separate LEDs (Red, Green, and Blue) embedded in a single package, with each color controlled independently.

Pin Configuration

Signal	FPGA Pin	Description
clk	35	System clock input (12 MHz)
led_r	21	Red LED output
led_g	22	Green LED output
led_b	23	Blue LED output

- The RGB LED has three control signals for Red, Green, and Blue.
- Each LED is active-low, meaning 0 turns it ON and 1 turns it OFF.

---

3. Verilog Code (rgb_blink_led.v)

The RGB LED is controlled using a counter-based clock divider. Each LED toggles sequentially to create a blinking effect.

```
module rgb_blink_led (
    input clk,        // 12 MHz system clock
    output reg led_r,  // Red LED
    output reg led_g,  // Green LED
    output reg led_b   // Blue LED
);
```

```
reg [23:0] counter;  // 24-bit counter for timing
reg [1:0] state;     // State variable to switch between colors

always @(posedge clk) begin
    counter <= counter + 1;

    if (counter == 24'd6000000) begin  // ~0.5s delay
        state <= state + 1;
        counter <= 0;
    end
end

always @(*) begin
    case (state)
        2'b00: begin
            led_r = 0; led_g = 1; led_b = 1;  // Red ON
        end
        2'b01: begin
            led_r = 1; led_g = 0; led_b = 1;  // Green ON
        end
        2'b10: begin
            led_r = 1; led_g = 1; led_b = 0;  // Blue ON
        end
        default: begin
            led_r = 1; led_g = 1; led_b = 1;  // All OFF
        end
    endcase
end
```

endmodule

Code Explanation:

1. Clock (clk) – Controls the timing of color changes.
2. Counter (counter[23:0]) – Divides the 12 MHz clock to create a 0.5s delay before switching colors.
3. State Variable (state[1:0]) – Determines which LED should be ON.
4. LED Control Logic:
    - State 00 → Red ON, Green OFF, Blue OFF
    - State 01 → Red OFF, Green ON, Blue OFF
    - State 10 → Red OFF, Green OFF, Blue ON
    - Default → All OFF

---

4. FPGA Implementation Steps

1. Synthesis (Yosys)

yosys -p "read_verilog rgb_blink_led.v; synth_ice40 -json rgb_blink_led.json"

- Converts Verilog to FPGA logic gates.

2. Place & Route (NextPNR)

nextpnr-ice40 --json rgb_blink_led.json --pcf pin_constraints.pcf --asc rgb_blink_led.asc --package hx8k

- Maps logic to FPGA resources and assigns pins.

3. Generate Bitstream (IcePack)

icepack rgb_blink_led.asc rgb_blink_led.bin

- Creates the binary file for FPGA programming.

4. Flashing FPGA (IceProg)

iceprog rgb_blink_led.bin

- Uploads the program to the FPGA.
- The RGB LED should start changing colors every 0.5s.

---

5. Simulation & Debugging

Run Simulation

make test

- Simulates the Verilog design using Icarus Verilog.

View Waveform

gtkwave waveform.fst

- Ensures proper color transitions.

Debugging Tips:

- If LED does not change color, check pin assignments (pin_constraints.pcf).
- If LED flickers too fast, increase the counter delay (24'd6000000).

---

6. Conclusion

- Successfully implemented an RGB LED blinking sequence on FPGA.
- Used open-source tools (Yosys, NextPNR, IceProg).
- Verified design through simulation & waveform analysis.

This project demonstrates basic FPGA control of multi-color LEDs, useful for embedded system applications.