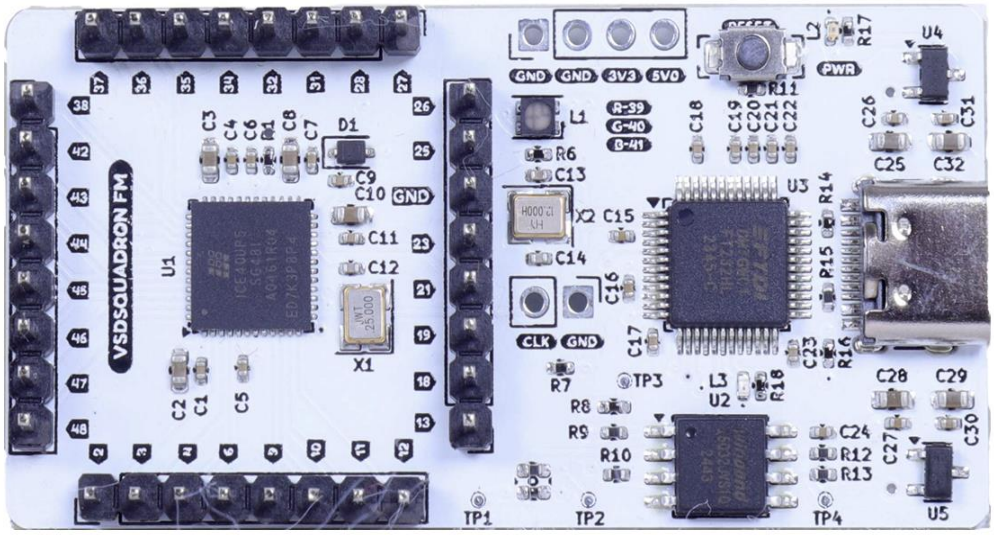

VSDFPGA Task 1

SUBMITTED BY

ISHA SINGH

Introduction

The purpose of this project is to understand, implement, and document the functionality of a Verilog-based LED control system on the VSDSquadron FPGA Mini board. The design utilizes Lattice iCE40 FPGA primitives, specifically the internal high-frequency oscillator (HFOSC) and the RGB LED driver block SB_RGBA_DRV, to control the onboard RGB LED using internal FPGA resources.

This report includes:

- Detailed explanation of the Verilog code (top.v)
- Complete pin mapping using the PCF file
- Integration steps while building and flashing the FPGA design
- Observations and testing results from the FPGA Mini board
- Challenges encountered and their solutions

This report accompanies the GitHub repository containing the working Verilog and PCF files.

SUMMARY OF VERILOG CODE FUNCTIONALITY

1. **Module Declaration**

```
module top (
    output led_red,
    output led_blue,
    output led_green,
    input  hw_clk,
    output testwire
);
```

Explanation: The top module is the top-level entity synthesized onto the VSDSquadron FPGA Mini board.

- **led_red, led_green, led_blue**: These output signals are connected to the physical red, green, and blue channels of the onboard RGB LED through the FPGA pin constraints (PCF file).

- **hw_clk**
  This input represents an external hardware clock source available on the board. In this design, the external clock is not used, as the internal high-frequency oscillator of the iCE40 FPGA is utilized instead. The signal is retained for completeness and future extensibility.

- **testwire**
  This output is a debug signal derived from the internal counter. It is used to verify correct clocking and counter operation by observing a measurable toggle on an external pin or logic analyzer.

2. Internal Signals and Counter Declaration

```
wire  int_osc;
reg  [27:0] frequency_counter_i;
```

- **int_osc**
  This wire carries the clock signal generated by the internal high-frequency oscillator (HFOSC) primitive of the iCE40 FPGA.
- **frequency_counter_i**
  A 28-bit register that functions as a free-running counter. It increments on every rising edge of the internal oscillator clock and is used to derive lower-frequency signals for observation and debugging.

3. Test Signal Assignment

```
assign testwire = frequency_counter_i[5];
```

The testwire output is driven by bit 5 of the internal counter.

- Each bit of a binary counter toggles at half the frequency of the previous bit.
- Bit 5 toggles at a significantly lower frequency than the clock, making it suitable for external observation using an LED, oscilloscope, or logic analyzer.
- This signal confirms that, the internal oscillator is running correctly

4. Counter Logic

```
always @(posedge int_osc) begin
  frequency_counter_i <= frequency_counter_i + 1'b1;
end
```

Explanation: This sequential logic block increments the counter on every rising edge of the internal oscillator.

- The counter operates continuously as long as the FPGA is powered.
- No reset is required for this demonstration, as the counter value is not used for state-critical logic.

- The counter serves as a simple frequency divider and activity indicator within the design.

5. **Oscillator design**

SB_HFOSC #(.CLKHF_DIV ("0b10")) u_SB_HFOSC (.CLKHFPU(1'b1), .CLKHFEN(1'b1), .CLKHF(int_osc));

The SB_HFOSC primitive instantiates the iCE40 FPGA's internal high-frequency oscillator.

- CLKHFPU (Power-Up Enable): Set to logic 1 to power up the oscillator.
- CLKHFEN (Clock Enable): Set to logic 1 to enable clock output.
- CLKHF_DIV = "0b10": Divides the nominal 12 MHz oscillator frequency by 4, resulting in an effective clock frequency of approximately 3 MHz.
- CLKHF output: Provides the internal clock (int_osc) used to drive the counter.

Using the internal oscillator eliminates the need for an external crystal or clock source, simplifying hardware requirements and improving design portability.

6. **RGB Driver**

```
SB_RGBA_DRV RGB_DRIVER (
   .RGBLEDEN(1'b1),
   .RGB0PWM (1'b0), // red
   .RGB1PWM (1'b0), // green
   .RGB2PWM (1'b1), // blue
   .CURREN  (1'b1),
   .RGB0    (led_red), //Actual Hardware connection
   .RGB1    (led_green),
   .RGB2    (led_blue)
  );
```

The SB_RGBA_DRV primitive is a dedicated RGB LED driver available in iCE40 devices. It provides regulated current outputs for directly driving an onboard RGB LED without external resistors.

- **RGBLEDEN**
  Enables the RGB LED output drivers.
- **CURREN**
  Enables the internal constant-current source.
- **RGB0PWM, RGB1PWM, RGB2PWM**: These inputs control the red, green, and blue LED channels respectively.
  - Red and green PWM inputs are tied to logic 0, keeping those channels OFF.
  - The blue PWM input is tied to logic 1, enabling the blue LED channel continuously.
- **RGB0, RGB1, RGB2**: Physical output connections from the driver to the FPGA pins wired to the RGB LED on the board.

As a result of this configuration, **only the blue LED remains illuminated**, which is useful for verifying correct pin mapping and RGB driver operation.

7. LED Current Configuration

defparam RGB_DRIVER.RGB0_CURRENT = "0b000001";

defparam RGB_DRIVER.RGB1_CURRENT = "0b000001";

defparam RGB_DRIVER.RGB2_CURRENT = "0b000001";

These parameters configure the drive current for each LED channel.

- The encoded values select a low, safe current level suitable for onboard LEDs.
- Using identical values ensures consistent maximum current capability across all three-color channels.
- Limiting the current prevents LED damage and avoids excessive power consumption.

Integration Steps and Observations While Working with the VSDSquadron FPGA Mini Board

1. The VSDSquadron FPGA Mini board was connected to the host computer using a USB Type-C cable, as specified in the datasheet. Verified that the USB cable supported **data transfer** (not power-only)
2. Ensured the board's power LED was ON
3. Confirmed proper physical seating of the USB-C connector

4. After connecting the board, the system was checked to ensure the FPGA Mini board was detected by the operating system.

5. lsusb command was used

6. The output confirmed the presence of an **FTDI device**, indicating that the onboard programmer was recognized successfully.

7. As specified in the VSDSquadron FPGA Mini datasheet, the open-source iCE40 FPGA toolchain was installed. This included:

- Yosys (for Verilog synthesis)
- nextpnr-ice40 (for place and route)
- icestorm utilities (icepack, iceprog)

8. The provided Makefile was used to automate the build and flash process.

make clean: This command removed previously generated files such as .json, .asc, and .bin, ensuring a clean build environment.

make build: This step performed the Verilog synthesis, place and route, bitstream generation.

9. Successful execution resulted in the generation of the FPGA programming file (.bin).

10. Once the build completed successfully, the bitstream was flashed onto the FPGA Mini board using:

sudo make flash

11. Root privileges were required to access the USB programming interface.

12. The flashing process wrote the bitstream into the SPI flash memory on the board, enabling non-volatile storage of the design.

13. Immediately after flashing:

- Blue channel turned ON

14. This confirmed:

- Correct synthesis and routing
- Proper PCF pin mapping
- Functional internal oscillator (SB_HFOSC)

15. To validate non-volatile programming: the USB cable was disconnected and the board was powered OFF. Then the USB cable was reconnected

16. Upon reconnection, the FPGA automatically reloaded the design from SPI flash and the RGB LED resumed its programmed behavior (steady blue illumination).

17. This verified successful flash memory usage.

18. After flashing the initial Verilog design onto the VSDSquadron FPGA Mini board, it was observed that only the blue channel of the onboard RGB LED was glowing, while the red and green channels remained OFF.

19. This behavior was consistent across multiple power cycles, confirming that the FPGA was correctly programmed and that the design was successfully loaded from the SPI flash memory.

20. The observed output directly corresponded to the PWM input assignments in the Verilog code, where RGB2PWM (blue) was held at logic high (1'b1), while RGB0PWM (red) and RGB1PWM (green) were held at logic low (1'b0).

21. Since the SB_RGBA_DRV primitive relies entirely on external PWM inputs to enable each color channel, only the blue LED was activated, verifying the correct operation of the RGB LED driver.

22. The absence of red and green illumination indicated that the pin mapping in the PCF file was correct and that the issue was related to logic control rather than hardware or connectivity faults.

Observations:

- The signal toggled at a lower frequency corresponding to bit [5] of the internal counter
- This confirmed the internal oscillator and counter logic were functioning correctly.

CHALLENGES FACED

While attempting to flash the code into the FPGA, error was faced due to failure in detecting the board. USB cable was power only and not data cable, board was loosely connected, and the FTDI interface was not recognized by the OS. Solution was to verify

that the USB cable was tightly connected to the board, and the board was checked for /dev/ttyUSB0 / /dev/ttyUSB1.

Another challenge that was faced that even after powering off the virtual machine, the FPGA continued to flash the RGB lights. This led to confusion about how to “stop” the blinking. The FPGA Mini uses an SPI flash chip to store bitstreams. When sudo make flash is used, the bitstream is written to non-volatile memory. This means that the FPGA automatically reloads the design on every power-up and the LED continues to blink until flash is erased or overwritten. To stop blinking, flash memory was erased.

Another thing is to ensure the correct pin mapping from the PCF files, to prevent an invalid signal. The FPGA pins must match the datasheet exactly.

OUTPUT

The following figure shows the VSDSquadron FPGA Mini board after successful programming.
As expected from the Verilog design, only the blue channel of the onboard RGB LED is illuminated, while the red and green channels remain OFF.
This confirms correct pin mapping, RGB driver operation, and successful non-volatile configuration via SPI flash.