# VSDSquadron FPGA Project: Technical Documentation

This document outlines the functionality of the provided Verilog code, the corresponding hardware pin configuration, and the process for integrating the design with the VSDSquadron FPGA Mini board.

### 1. Verilog Code Functional Overview

The top-level Verilog module is designed to drive the VSDSquadron board's onboard RGB LED. It utilizes two primary Lattice primitives: an internal high-frequency oscillator (SB\_HFOSC) and an RGB LED driver (SB\_RGBA\_DRV).

#### **Internal Oscillator and Counter**

* **SB\_HFOSC Primitive:** This block generates a stable internal clock without requiring external components. With the CLKHF\_DIV parameter set to 2, it produces a **12 MHz** clock (int\_osc) from the base 48 MHz oscillator.
* **Frequency Counter:** A 28-bit counter (frequency\_counter\_i) increments on every rising edge of the 12 MHz clock. The higher-order bits of this counter toggle at a much slower, human-visible rate, making them ideal for creating LED blink patterns.

#### **RGB LED Driver (SB\_RGBA\_DRV)**

The module generates Pulse Width Modulation (PWM) signals for the RGB LED by logically combining the slow-toggling counter bits. The SB\_RGBA\_DRV primitive is instantiated to control the LED channels based on these signals.

* **Green Channel (led\_green):** Activated when frequency\_counter\_i[24] AND frequency\_counter\_i[23] are high.
* **Blue Channel (led\_blue):** Activated when frequency\_counter\_i[24] is high AND frequency\_counter\_i[23] is low.
* **Red Channel (led\_red):** Activated when frequency\_counter\_i[24] is low AND frequency\_counter\_i[23] is high.

This logic causes the red, green, and blue LEDs to illuminate in a distinct sequence as the counter cycles, creating a simple, repeating blink pattern.

### 2. PCF Pin Mapping and Hardware Correlation

The Physical Constraints File (PCF) maps the logical signals from the Verilog design to the physical pins of the FPGA, as specified in the VSDSquadron board datasheet.

* led\_red -> **Pin 41** (Corresponds to RGB2 on the board)
* led\_blue -> **Pin 40** (Corresponds to RGB1 on the board)
* led\_green -> **Pin 39** (Corresponds to RGB0 on the board)
* hw\_clk -> **Pin 20** (Input for the external hardware oscillator)
* testwire -> **Pin 17** (General-purpose test output)

Each mapping was verified against the board's official schematic to ensure the Verilog outputs correctly drive the intended physical components.

### 3. Integration and Deployment Workflow

The design is built and flashed onto the FPGA using a toolchain within the provided VirtualBox VM.

#### **Software and Board Setup**

1. **VM Preparation:** The VSDSquadron FM VirtualBox image is configured and launched.
2. **Board Connection:** The FPGA board is connected to the host PC via USB-C, and the FTDI interface is passed through to the VM. The lsusb command is used within the VM to verify a successful connection.

#### **Build and Flash Process**

The provided Makefile automates the entire synthesis and programming flow. The process is executed from the project directory in the VM's terminal:

1. **make clean**: Clears any artifacts from previous builds.
2. **make build**: Synthesizes the Verilog code into a bitstream file using Yosys and NextPNR.
3. **sudo make flash**: Programs the FPGA's SRAM with the generated bitstream.

Upon successful flashing, the RGB LED on the board immediately began blinking in the red, green, and blue sequence, confirming that the hardware, Verilog code, and pin constraints were all functioning correctly together.

### 4. Challenges and Solutions

The integration process was largely smooth, but required careful attention to the following areas:

* **Pin Assignment Verification:** The primary challenge was ensuring the PCF pin assignments for led\_red, led\_green, and led\_blue perfectly matched the board's datasheet. Relying solely on the official datasheet for the RGB0/1/2 to pin mapping was crucial for success.
* **VM Device Connection:** The FTDI USB device can sometimes disconnect from the VM. It was important to ensure the device was correctly captured by VirtualBox before running the make flash command to avoid programming errors.
* **Oscillator Configuration:** The design relies on the CLKHF\_DIV parameter to slow the internal oscillator to 12 MHz. Verifying this parameter was set correctly was essential for achieving the intended blink rate.

By systematically verifying these key points, the design was implemented successfully without modifications to the source code.