Mini project 2: Smooth RGB Colorcycle

Ishan Porwal

The goal of this project was to design a digital circuit to drive the RGB LED on the iceBlinkPico board so that it smoothly cycles through the colors on the HSV (360°) color wheel once per second by driving individual LEDs using pulse width modulation according to given waveforms centered around a 12 MHz clock.

The main parameter used in the code was the INTERVAL parameter which is the number of clock steps calculated to complete the time necessary for a 1-degree increment in the HSV color wheel. In sequential logic, every time a logic variable *counter* meets this interval, the counter is reset and a logic variable *hue* is incremented by 1 (reset to 0 at 359 degrees). Also in sequential logic, a logic variable *pwm_counter* increments on every rising edge of the clock signal and is compared against PWM values computed in combinational logic. Since *pwm_counter* is initialized as an 8-bit logic variable, it naturally overflows back to 0 on the clock cycle after it reaches 255. Based on the comparison between *pwm_counter* and the PWM intensity values for each color (RGB), each LED color component is assigned a high (1) or low (0) state, creating a duty cycle proportional to the desired brightness.

The combinational logic driving these PWM intensity values for each color is solely based on the given waveforms. The input is *hue*, which represents the current degree in the HSV color cycle. Based on *hue*, the PWM intensity values for each color are calculated. This logic follows a segmented approach where each segment of the *hue* degree range corresponds to a different combination of RGB intensity values. These segments of increasing, decreasing, and stable intensities ensure a smooth and continuous progression through the HSV color wheel.

For testing operation, a test bench was created to generate a 12 MHz clock signal and visualize the RGB outputs against time. The clock signal toggles every 4.16667 ns which corresponds to a 12 MHz clock frequency. This SystemVerilog test bench is simulated by using Icarus Verilog to compile the test bench into an executable, the Icarus Verilog Runtime Engine to run the simulation, and finally GTKWave to view the waveforms generated to ensure the PWM behavior and RGB outputs were correct.

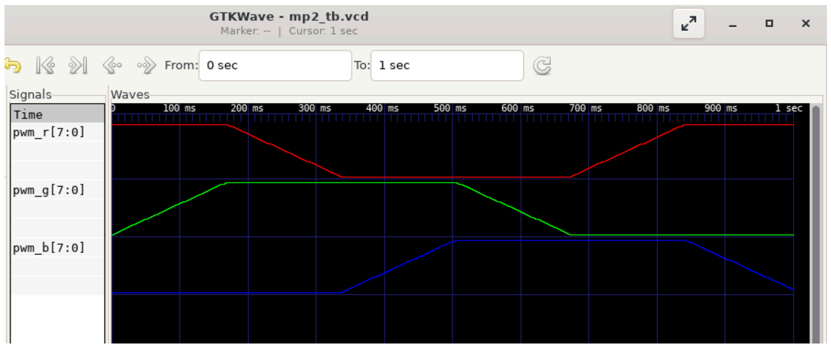

Figure 1. Screengrab of gtkwave plot showing 1 cycle simulation of RGB signal components as a function of time (1 second)