Embedded System Design with Programmable Logic

ECE 544 Project Report

Color wheel implementation
And
Pulse width modulation detection
On Nexys4 board

By: Mohak Patel (mohak@pdx.edu)

Guided by: Roy Kravitz (roy.kravitz@pdx.edu)

Sai Bodanki (bodanki@pdx.edu)

Introduction:

Color wheel allows us to choose any color by varying the Hue, Saturation, and Value in the HSV scale and according to the HSV value, it generates the variable duty cycle of red, green and blue color for on board RGB LEDs. We take the output signal of RGB led for the pulse width detection (PWM). Two types of PWM detection methods are used in this project, hardware PWM detection, and software PWM detection.

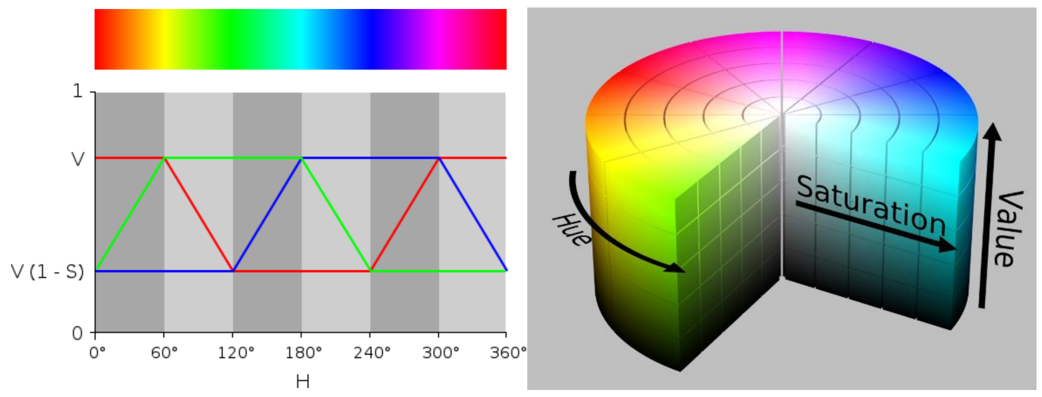

Figure 1: HSV representation and HSV color wheel

(Reference: https://en.wikipedia.org/wiki/HSL_and_HSV#From_HSV)

For software PWM detection, the algorithm is implemented in software. For hardware PWM detection, we required additional hardware module AXI_GPIO in the block design. This module is used for sending the counter value of high/ low pulse time interval. The edge detection algorithm used for hardware detection and software detection.

Functional Specification:

- Slides Switch [0] selects between software (down) and hardware (up) pulse width detection.
- LD0 to indicate software or hardware pulse width detection. If led is on then it indicates the hardware PWM detection and if it is off then it indicates the software detection.
- The system reset is done by pressing the CPU reset button on the Nexys4 DDR.
- The BTNC button is used to terminating the application.
- The BTNR and BTNL buttons are used to vary the Saturation value in the HSV scale. BTNR increment the saturation value and BTNL decrement the saturation value.
- The BTNU and BTND buttons are used to vary the Value in the HSV scale. BTNU for incrementing the Value and BTND for decrementing the Value in HSV scale.
- The rotary encoder knob is used to alter the Hue value. The push button of the rotary encoder is used to terminating the application.
- PmodOLEDrgb display is used for displaying the current Hue, Saturation and Value on the left side of the display and Rectangle showing the color for the corresponding value of the Hue, Saturation, and Value on the right side of the display.

- RGB1 and RGB2 – The tri-color LEDs indicate the color selected from the color wheel. To drive both RGB LEDs, we used HSV to RGB conversation algorithm.
- Seven segment display is used for display the duty cycle of Red signal, Green signal, Blue signal and to display hardware (H)/ software (S) detection.
  1. Digit [0] display the letter "H" for hardware detection and "S" for software detection for few seconds.
  2. Digit [7:6]: display the duty cycle of the RED RGB led.
  3. Digit [4:3]: display the duty cycle of the GREEN RGB.
  4. Digit [1:0]: display the duty cycle of the BLUE RGB led.

Color Wheel implantation and PWM generation:

- The value of hue varies between 0 to 360 degree. The saturation varies between 0 and 1. The Value varies between 0 and 1.
- From figure, we can convert the HSV to RGB using the four formulas [5]:
  level_top= value (1)
  level_bottom = value * (1.0 - saturation); (2)
  slop_down = value * (1.0 - (saturation * delta_hue)); (3)
  slop_up = value * (1.0 - (saturation * (1.0 - delta_hue))); (4)
  where,delta_hue=Hue/60

- The value from rotary encoder and pushbuttons are mapped to the HSV to RGB algorithm to generate the duty cycle for RGB LEDs.
- We use this RGB values to drive the on board RGB leds.
- We convert this RGB values to 16-bit RGB value for displaying the hue, saturation, and value on PmodOLEDrgb.

PWM detection:

Software PWM detection:

- First check for the high pulse
- If the signal is high then we start one counter for counting the time interval for the high pulse
- If the signal is low then we start one counter for counting the time interval for the low pulse
- If we get the low to high transition we send that count for calculate the duty cycle and reset both counters to zero
- PWM_detection() function calculate the duty cycle and Display_duty_cycle() function display the duty cycle on seven segment display.
- FIT(Fix interval Timer) interrupt handler is used for monitoring the GPIO input channel.
- we use XGpio_DiscreteRead( ) function to read that GPIO port.

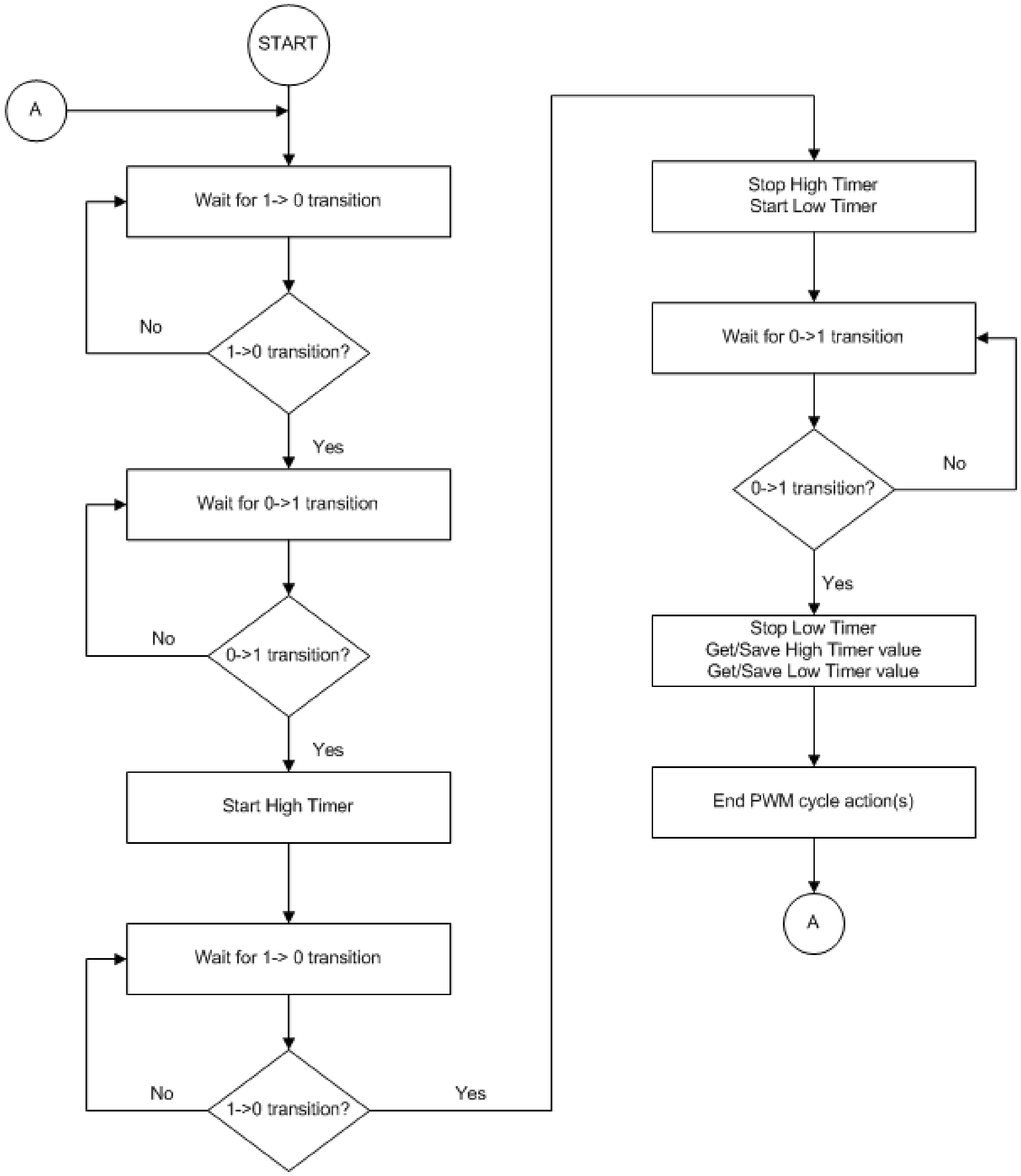

Figure: State diagram of edge detection algorithm using counter [1]

<u>Hardware PWM detection:</u>

We require the additional hardware module AXI_GPIO for the hardware detection algorithm. We add three AXI_GPIO module for the red, green and blue signal. Each module has two 32 bit input channels, one for the high counter value and second for low counter value. We also require additional 100 MHz clock. The module is written in Verilog to calculate the high and low interval time of the signal.

- First check for the high pulse

    - If the signal is high then we start one counter for counting the time interval for a high pulse.
    - If the signal is low then we start one counter for counting the time interval for the low pulse
    - If we get the low to high transition we send that low count to calculate the duty cycle and reset the low counters to zero.
    - If we get the high to low transition we send that high count to calculate the duty cycle and reset the high counter to zero.
    - PWM_detection() function calculate the duty cycle and Display_duty_cycle() function display the duty cycle on seven segment display.
    - FIT(Fix interval Timer) interrupt handler is used for monitoring the GPIO input channels.
    - For hardware detection, first initialize the GPIO instance using XGpio_Initialize() function, second we set the direction of GPIO port using XGpio_SetDirection() function.
    - Last we use XGpio_DiscreteRead( ) function to read that GPIO port.
    - Added n4fpga_PWM_detection module for hardware PWM detection in n4fpga.v file. And create three instances of that module in n4fpga.v file.

- To calculate the duty cycle:
  Duty Cycle = (High Time * 100) / (High Time + Low Time)

Challenges:

- Sometimes forget to change the High and Low base address of Nexys4IO when creating new board support package, and I took time to figure it out.
- Took time to understand the functionality of FIT interrupt handler. It should be as light as possible.
- It took time to familiarize with Vivado tool.

Extension:

- Added indication of software or hardware PWM detection by displaying "H" for hardware detection and "S" for software detection on seven segment display. It displays H or S on first seven segments whenever we switch from software to hardware, vice versa. It displays H or S for few seconds and after that, it will display the duty cycle in that mode.

References:

[1] Lecture notes by Roy Kravitz
[2] Getting Started in ECE 544 manual
[3] Xilinx forum
[4] Nexys4 manual
[5] http://www.vagrearg.org/content/hsvrg