LED Color Controller

The aim of this exercise is to design a circuit that adjusts/controls the color of an RGB LED. To this end, a pulse width modulation (PWM) circuit is used where the duration of the pulses controls the three color components (RGB) of the LED. In the following, you will design a circuit and draw its block diagram for different components of the color controller.

Hand-in instructions: Prepare a small report with your solutions (detailed). Submit your report as a PDF through the lecture moodle per the moodle submission deadline.

Designing the Individual Components

For the following components, the registers use a clock `CLKxC` and an active-high asynchronous reset `RSTxR` which is only asserted once during power-up (HIGH) and inactive (LOW) afterward.

Task 1: Rising Edge-Detector

Draw a block diagram for a circuit that can detect a transition from '0' to '1' (rising edge-detector) of an input signal `PUSHxSI` and generate an output pulse `EdgexSO` that is active (HIGH) for a duration of one clock cycle. If you completed Task 3 of the previous exercise (Exercise 1b), then you can use your solution directly here.

Task 2a: 20-Bit Counter with Enable

Draw a block diagram for a 20-bit counter (0 to $2^{20}-1$) that increments only when an input enable signal is '1'. The counter should overflow once it reaches the maximum value. That is, when the counter is equal to $2^{20}-1$, it should go to 0 in the next clock cycle if you add a 1 to $2^{20}-1$. You can use your block diagram from Task 2 of the previous exercise (Exercise 1b) as a starting point.

Circuit designer's toolbox

To better understand how an overflow occurs, try to perform binary addition on paper as shown below assuming a 2-bit counter which currently has value $3_{10} = 11_2$ (subscript indicates number system) and what happens when you add a 1 to this. We have sign-extended with 0's as the result of adding two 2-bit numbers requires 3 bits.

Carry		1	1	
A		0	1	1
B	$+$	0	0	1
C		1	0	0

The result is $100_2 = 4_{10}$ as expected, but if your counter register can only save a 2-bit result, you will have to choose which bits to save! If you only save the lower bits, you see that the actual result into your register is 0. In this exercise, we use the overflow to reset our counter register, but an overflow is in many cases undesirable.

It is critical that you understand how numbers grow under addition, subtraction, and multiplication as the result may otherwise not be what you expect.

Task 2b: 20-Bit Counter with Enable and 2^{17} Increment

Draw a block diagram for a 20-bit counter (0 to $2^{20}-1$) that **increments by 2^{17}** only when an input enable signal is '1'. The counter should overflow and go to 0 in the next cycle once the result of the addition is larger than the maximum result for the counter register, $2^{20}-1$, same as in Task 2a. You can use/modify your block diagram from Task 2a.

Think about how the addition can be optimized by considering the binary representation of 2^{17}, i.e., try to perform binary addition as shown in the **Circuit designer's toolbox** on the previous page.

Hints and common errors

The reason for choosing a large counter width (20-bit) lies in the slow reaction time of the LEDs. The LED would simply not be able to follow a short period for the PWM. Therefore, we, choose a rather long period (almost $10\,\text{ms}$). However, with this, the threshold counter needs to increment in larger steps to avoid the need to press the button a million times before reaching full illumination.

Task 3: PWM Pulse Generator

Design a circuit with the following functionality. The circuit has an output signal `PWMxS` which is '1' when a counter value is smaller than a threshold signal and is otherwise '0'. The output `PWMxS` is called a PWM pulse, and a higher threshold will make the width of this pulse larger. This circuit is similar to the pulse generator from Task 1 of the previous exercise (Exercise 1b in week 1), but here the pulse width can be controlled using the threshold.

You can use the following components:

- A free-running 20-bit counter with an increment of 1.
- A 20-bit input threshold signal to which the counter value is compared.
- A comparator with a 1-bit output signal indicating whether the counter value is less than the input threshold.

Draw the block diagram of your circuit with all the above components.

Circuit designer's toolbox

In this exercise, we show you a way of simplifying the design process by:

- **Listing the components (counter, comparator, etc.) you need to solve a task.**

This is similar to what we did in Exercise 1a! Designing digital circuits is often a task of connecting standard components in different ways. A counter can be used for a wide range of applications but it is still only a counter and breaking your design into many smaller pieces that you are familiar with can significantly speed up the design process.

Previously in Exercise 1b we proposed that you (when creating a better problem statement):

- Write down a list of ports with their directions and purpose.
- Draw out a timing diagram showing the behavior of the signals in your circuit.

Designing the Color Controller

We now consider designing the color controller using the blocks implemented in the previous tasks. **This design will be used as a starting point for the lab session next week.**

Task 4: LED Color Controller

Connect the circuits that you designed in the previous tasks so that the output signal of the edge-detector in Task 1 provides the enable for the 20-bit counter in Task 2b, and its 20-bit counter value provides the input threshold of the PWM pulse generator in Task 3.

The RBG LED is controlled by turning each of the color components (RGB) on and off. Copying this circuit three times will therefore allow individual control of each color, with the pulse width controlling the brightness of each color component.