Introduction

In this lab, you will design a SystemVerilog based Avalon IP component for the platform you developed as part of the previous lab, named **"led_pwm"**.

The **led_pwm** component will provide an interface between software running on the embedded Nios2 processor and the 26 DE2-115 LEDs. The interface will allow the software, by way of writing a control register, to illuminate any combination of LEDs corresponding to the pattern of 1-bits in the value written.

Immediately after writing a value, **led_pwm** will illumiate the LEDs corresponding to the 1-bits in the written value, and then subsequently dim their brightness to zero after 1.3422 seconds (2^26/50e6).

You will also write software that will illuminate a random LED every 90 ms until one of the buttons are pressed, at which point the system will change the time between the illimination of a random LED to 50 ms, 70 ms, or 90 ms, depending on which key was pressed.

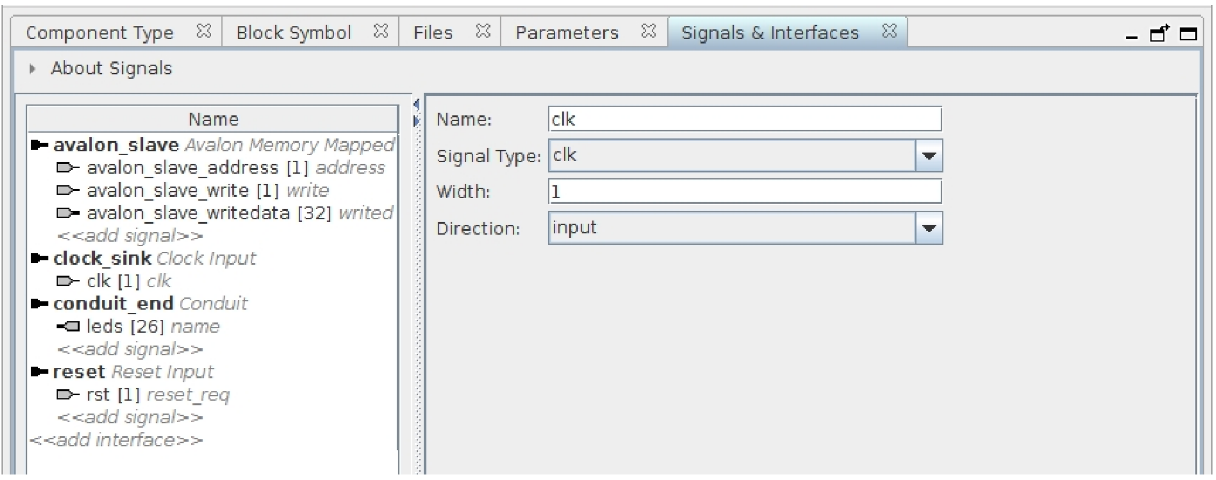
The correct operation of the 90 ms mode is shown at the following link: https://bit.ly/491lab3

Creating an Avalon IP Component

1. In Platform Designer's "IP Catalog" pane, double-click *New Component*.
2. When the *Component Editor* appears, enter "led_pwm" for the *Name* and *Display name*.
3. Under the *Signals & Interfaces* tab in the left panel, click <<*add interface*>> and select *Clock Input*. Once added, you can change the name of the clock from "clock_sink" to something more sensible such as "clk". Then click <<*add signal*>> and select "clk".
4. Under the *Signals & Interfaces* tab in the left panel, click <<*add interface*>> and select *Reset Input*. Once added, you can change the name of the clock from "reset_sink" to something more sensible such as "rst". Then click <<*add signal*>> and select "reset".
5. Under the *Signals & Interfaces* tab in the left panel, click <<*add interface*>> and select *Avalon Memory Mapped Slave*. Once added, change the *Associated Clock* to "clk" and *Associated Reset* to "rst". Then click <<*add signal*>> and select "address", leaving the *Width* field to 1. Click <<*add signal*>> and select "writedata" and set the *Width* field to 32. Click <<*add signal*>> and select "write". At this point you may need to select the avalon_slave interface and re-set the *Associated Reset* to "rst".
6. Under the *Signals & Interfaces* tab in the left panel, click <<*add interface*>> and select *Conduit*. Once added, change its name to "leds". Then click <<*add signal*>> and select "*". Change the name to "leds" and set the *Width* field to 26.
7. Now go to the *Files* tab and click "Create Synthesis File from Signals" and click "Save".

8. Click Finish and Save.
9. Now you should be able to wire your new IP component into your system. Be sure to connect the clock input to "sys_clk", the reset input to the global reset, and the avalon slave to the Nios2's data master.
10. The Verilog file for your module can be found in your project directory.

The screenshow below shows the final configuration of the component. Your signal names do not need to match these exactly, as long as your have one clock signal, one reset signal, one output conduit signal, and one Avalon slave with corresponding write, address, and data signals.



```
Name
avalon_slave Avalon Memory Mapped
    avalon_slave_address [1] address
    avalon_slave_write [1] write
    avalon_slave_writedata [32] writed
    <<add signal>>
clock_sink Clock Input
    clk [1] clk
conduit_end Conduit
    leds [26] name
    <<add signal>>
reset Reset Input
    rst [1] reset_req
    <<add signal>>
<<add interface>>
```

Name: clk
Signal Type: clk
Width: 1
Direction: input

Figure 1: Screenshot

Your Code

Your hardware component will perform the following functions:

1. Maintain a 26-bit duty cycle value for each of the 26 individual LEDs, for a total of 676 bits.
2. Decrement each duty cycle value containing a value greater than zero by 1 every clock cycle.
3. Reset any duty cycle value to `0x3FFFFFF` if its corresponding bit is set to 1 after an Avalon write.
4. Drive each LED value with a PWM signal having a period of 67108864 cycles (2^26) and its corresponding duty cycle, treated as a percentage of its value relative to 67108864.

Your software will perform the following:

1. Write a value to the led_pwm every 90 ms containing a 26-bit value where a random bit is 1 and the others are 0.

2. Continuously monitor the KEY input. If the user presses KEY[3] (the leftmost button), the code will change the write interval to 50 ms. If the user presses KEY[2], the code will change the interval to 70 ms. If the user presses KEY[1], the code will change the interval to 90 ms.
3. Repeat infinitely.

Parallel I/O Modules

- Be sure to remove the Parallel IO (PIO) component that you previously used for the LEDs in both Platform Designer and in the top-level System Verilog module (lights.sv).
- The input value associated with the KEY[3:0] input can be accessed in the software by adding a new Parallel I/O (PIO) component in Platform Designer set up as a 4-bit Input. Be sure to change its name "key", wire it up to your system, and add its external connection as "key". In the top-level SystemVerilog file, connect this PIO externally within your nios_system port map as:

```
.key_export (KEY)
```

After this, the key value can be read using in the software using `IORD(KEY_BASE,0)`.

Requirements

During your group demo, we will ask you to change the speed setting of the system by pressing the KEYs. We will verify the correctness of your code by visual inspection of the DE2-115 board while in operation. We will also ask you questions about how your code works.

Submitting Your Code

When you are ready to turn in your code for grading, each group should submit an archive of their project directory to Dropbox. To do this, enter the following commands:

```
rm -Rf <project_dir>/db <project_dir>/incremental_db <project_dir>/
    output_files
tar cvf lab3.tar <project_dir>
gzip lab3.tar
```

Avalon Slave Interface

The Avalon Slave Interface is shown in the waveform below.

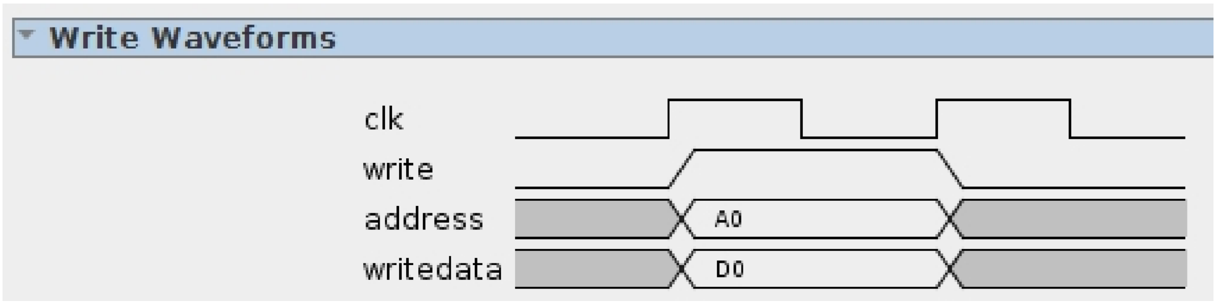


Figure 2: Screenshot

Rubric

- Correct operation of the led_pwm component - **60 points**
- Correct operation of the software in 90 ms mode - **20 points**
- Correct operation of the software in switching to 50 ms, 70 ms, and 90 ms modes - **20 points**

Total points possible: 100.