Lab_1: LED Driver

Introduction

In this lab, you will write a complete VHDL description for the *LED_DRIVER* circuit shown below (**Figure 1**). Besides, the circuits ought to be ported to ZedBoard to check the functionality.

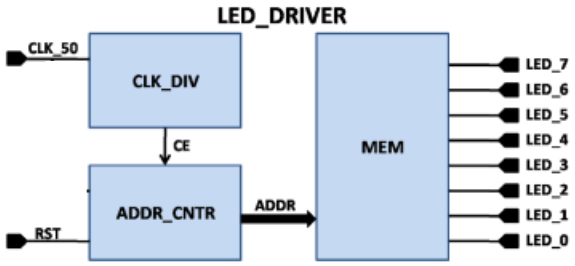

Figure 1 Schematic for the *LED_DRIVER* Circuit

Modules

You should complete the **three components** of this circuit separately and assemble them at the top-level.

1. Clock Divider

You will use a dedicated **100MHz** clock input source on the demo board. This signal needs to be **divided down** to allow the LEDs to change at a rate that you can **visually observe**. As part of this exercise, you will decide exactly which frequency to use for changing the LED pattern. This choice will be reflected in your VHDL source code.

- Input:
  - the clock signal (connect to port in top-level);
- Output:
  - clock signal with lower frequency (connect to port in ADDR_CNTR);

2. n-Bit Binary Counter

You will write a complete VHDL description for the module *ADDR_CNTR* by using a **GENERIC** statement to specify the bit width.

In this experiment, we have determined the number of LED lights, but the counter bit width has not been determined. Perhaps your customers will change their requirements and require you to implement a more complex set of running lights. And we may have to use this module

in other projects, the use of GENERIC can improve the reuse capability of this module.

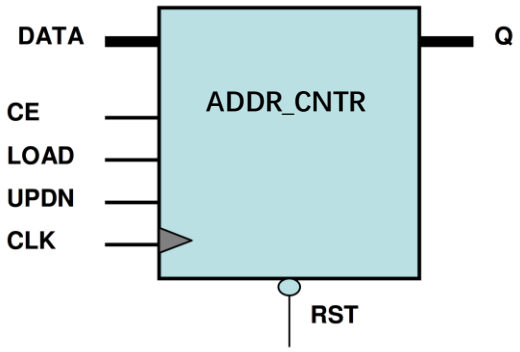

Figure 2 Schematic for a Parameterized (n-bit) Wide Binary Counter

This is an **n-bit binary**, **up/down, loadable** counter, with **active low asynchronous** reset. When and only when the input signal LOAD is set to 1, the data input is read on the rising edge of the clock and stored in the register. When and only when the input signal CE is set to 1 (and the input signal LOAD is set to 0), the number stored in register increase (when the input signal UPDN is 1) or decrease (when the input signal UPDN is 0) on the rising edge of the clock. The output signal Q is always same as the number stored in register.

- Input:
  - DATA (n bits array) (connect to port in top-level);
  - count enable signal (connect to port in top-level);
  - load signal (connect to port in top-level);
  - count forwards or backward signal (connect to port in top-level);
  - the clock signal (connect to port in CLK_DIV);
  - reset signal (connect to port in top-level);
- Output:
  - Q (n bits array) (connect to port in MEM);

3. Read-Only Memory

You will write a complete VHDL description for the MEM submodule of the LED_DRIVER circuit by using a two-dimensional array. The MEM module is an 8-bit wide ROM with 16 words initially. **【The output of the n bit Binary Counter indicates the index of the ROM. And the 8-bits data in ROM corresponds to the 8 LEDs.】** Later (for example, the final project), you may choose to create a deeper structure to hold more content.

We encourage you to use different implementations for this part of the code. Using the "when-case" statement is very inefficient for large amounts of data. Of course, if you insist on doing so, we recommend that you write a python applet to generate the code automatically.

The data in memory is at your discretion, but we recommend that it should be address related so that it can be easily checked by the TA. **It is important to note that the data in memory must be stored exactly, not directly computed by address.** Your module should be able to change the data at any address to any value at the request of the TA.

- Input:
  - Address (4 bits array) (connect to port in ADDR_CNTR);
- Output:
  - LED_# (connect to port in top-level);

Tips

Using

“if rising_edge(clk) then”

instead of

“if clk'event = TRUE and clk = '1' then”

to check the clock rising edge.

Key points

1. **The circuit diagram described in your VHDL code must be exactly the same as Figure 1. Each component should implement and can only implement its own functionality!**
   - The name of the top-level module should be correct (refer to **Figure 1.**).
   - The name of the components should be correct (refer to **Figure 1.**).
   - The name of I/O ports in the top-level should be correct (refer to **Figure 1.**).
   - The name of the signals should be correct (refer to **Figure 1.**).
2. **To ease your load, the output of the MEM can be in the form of an array (STD_LOGIC_VECTOR).**
3. **The result will be checked on Zedboard on Wednesday’s lab course. [Please finish the tutorial 2 and the circuits beforehand.]**

Code format

1. Keywords (except data type) are **all lowercase**.
2. The library names and data types are **all capitalized**.
3. Naming must be **meaningful** (except for the prescribed name). There are no restrictions on naming, and you can reasonably use uppercase letters and underscores to make your

name easier to read and understand.

4. Use **four Spaces** or **a Tab** to indent your code. Reasonable indentation makes your code easier to read and understand.

Submit

1. Compress the HDL code and **name the compressed package as follow the format**:

   Lab1_[YourNameInEnglish]_[YourStudentNumber].zip

2. **File Organization Schema in Package**:

   ```
   Lab1_[YourName]_[YourStudentNumber].zip
    ├──CLK_DIV.vhd
    ├──ADDR_CNTR.vhd
    ├──MEM.vhd
    ├──LED_DRIVER.vhd
    └──testbench.vhd
   ```

   Upload the .zip file to related lab folder on Blackboard.

Deadline: 2020-9-30 17:30

- Submit on time, get all scores.
- Submission time does not exceed 24 hours of the deadline, get half of the score.
- Submitted more than 24 hours from the deadline, get no score.

Any Question?

Any questions on course or labs can be proposed in the Discussing Forum on BB.

We recommend you subscribe the forum to receive the newest topics on time.