

Lab 3: Clock Divider and LED Controller

Submission Due Dates:

Demo: 2023/10/03 17:20
Source Code: 2023/10/03 18:30
Report: 2023/10/08 23:59

Objective

Getting familiar with the clock divider and LED control on the FPGA demo board.

Action Items

1. Module: clock_divider

Write a Verilog module for the clock divider that divides the frequency of the input clock by 2^{25} to get the output clock. Here is the template you should use:

```
module clock_divider
    #(parameter n = 25)
    (input clk,
     output clk_div);

    // add your design here

endmodule
```

We also provide two templates to handle a single module with two clocks. Choose the one you prefer to proceed with the lab assignment.

2. lab3_1.v (30%)

Input Connection:

clk	connected to W5
rst	connected to W16
en	connected to V17
speed	connected to V16

Output Connection:

led	connected to LD15-LD0
-----	-----------------------

Design the LED Controller in Verilog, which is synchronous with the positive clock edges. The clock frequency is obtained by dividing the frequency of Basys3's onboard 100 MHz clock by 2^{25} (i.e., approximately 3 Hz) or 2^{27} (i.e., approximately 0.75 Hz), selected by a switch. Also, download and demonstrate your LED Controller on the FPGA board. Use the clock divider you designed previously.

Here are the input and output constraints:

◆ **rst** signal is an asynchronous positive reset. It will reset the design to its initial condition

(will be shown later).

Notice that rst has the highest priority than any other signals.

◆ **en** signal is a signal that decides whether our module should change its condition or not.

- en == 0: hold the LEDs unchanged.
- en == 1:
 - ✧ If speed == 0: the LEDs will change with the **0.75 Hz** clock.
 - ✧ If speed == 1: the LEDs will change with the **3 Hz** clock.

The LEDs will change in the following order.

(● : led ON , ○ : led OFF)

(LD15) ○○○○○○○○○○○○○○○○○ (LD0) → initial condition

(LD15) ●○○○●○○○●○○○●○○○ (LD0)

(LD15) ●●○○●●○○●●○○●●○○ (LD0)

(LD15) ●●●○●●●○●●●○●●●○ (LD0)

(LD15) ●●●●●●●●●●●●●●●● (LD0)

(LD15) ○○○○○○○○○○○○○○○○○ (LD0)

→ then next round

You should use the following template for your design and name the file as "**lab3_1.v**".

```
module lab3_1(
    input clk,
    input rst,
    input en,
    input speed,
    output [15:0] led
);

    // add your design here

endmodule
```

Demo: <https://youtu.be/dG3Jx8B1-d4>

3. lab3_2.v (30%)

Input Connection:

clk	connected to W5
rst	connected to W16

en	connected to V17
speed	connected to V16
dir	connected to W17

Output Connection:

led	connected to LD15-LD0
-----	-----------------------

Design the LED Controller in Verilog, which is synchronous with the positive clock edges. The clock frequency is obtained by dividing the frequency of Basys3's onboard clock (the onboard clock is 100MHz) by 2^{24} (i.e., approximately 6 Hz) or 2^{26} (i.e., approximately 1.5 Hz), selected by a switch. Also, download and demonstrate your LED Controller on the FPGA board. You should use the clock divider you designed previously.

The LEDs behave as follows:

- ◆ The LEDs will change in three following modes.

1. Regular mode

(LD15) ○○○○○○○○○○○○○○○○○ (LD0) → initial state

(LD15) ●○○○●○○○●○○○●○○○ (LD0)

(LD15) ●●○○●●○○●●○○●●○○ (LD0)

(LD15) ●●●○●●●○●●●○●●●○ (LD0)

(LD15) ●●●●●●●●●●●●●●●● (LD0)

(LD15) ○○○○○○○○○○○○○○○○○ (LD0) → next round

(LD15) ●○○○●○○○●○○○●○○○ (LD0)

.....

After the process repeats three rounds, our machine will go to the **escape mode** when all LEDs are ON.

2. Escape mode

Initially, all LEDs would be ON.

- If dir == 0:

At each clock edge, the leftmost two LEDs which are ON will be turned OFF.

Eg: (LD15) ●●●●●●●●●●●●●●●● (LD0)

→ (LD15) ○○●●●●●●●●●●●●●● (LD0)

→ (LD15) ○○○○●●●●●●●●●●●● (LD0)

→ (LD15) ○○○○○○●●●●●●●●●● (LD0)

→

After all LEDs are OFF, the machine will turn to the **shining mode**.

- If dir == 1: (act as opposed to dir == 0)

At each clock edge, the rightmost two LEDs which are OFF will be turned ON.

Eg: (LD15) ○○○○○○●●●●●●●●●● (LD0)

→ (LD15) ○○○○●●●●●●●●●●●●●● (LD0)

After all LEDs are ON, the machine will go to the **regular mode**.

Notice: If all LEDs are ON when entering this mode, the machine will then go to the regular mode.

When a machine encounters a situation requiring a change in mode, it should wait for one cycle before changing (if the situation does not change in the cycle).

3. Shining mode

All LEDs will turn ON and OFF repeatedly. After 5 on-off cycles, the controller goes to the **regular mode**.

Here are the input and output constraints:

- ◆ **rst** signal is an asynchronous positive reset. It will reset all the LEDs to OFF and the FSM to the regular mode. All the internal round counters, if any, should be reset as well.
- ◆ **en** signal is to indicate whether our design should change its condition or not.
 - If en == 0: hold the LEDs unchanged.
 - If en == 1:
 - ✧ If speed == 0: the LEDs will change with the **1.5 Hz** clock.
 - ✧ If speed == 1: the LEDs will change with the **6 Hz** clock.

You should use the following template for your design and name the file as "**lab3_2.v**".

```
module lab3_2(
    input clk,
    input rst,
    input en,
    input speed,
    input dir,
    output [15:0] led
);

    // add your design here
endmodule
```

Requirements:

1. You must have at least one finite state machine (FSM) in this design.
2. The FSM should have at least three states. Additional states are acceptable. You may also design additional FSM(s) if necessary. You need to explain your design clearly in the report.

Demo: <https://youtu.be/lwM6iabRjr0>

4. lab3_3.v (40%)**Input Connection:**

clk	connected to W5
rst	connected to W16
en	connected to V17

Output Connection:

led	connected to LD15-LD0
-----	-----------------------

Design the LED Controller in Verilog, which is synchronous with the clock whose frequency is obtained by dividing the frequency of Basys3's onboard 100MHz clock by either 2^{24} (i.e., approximately 6 Hz), 2^{25} (i.e., approximately 3 Hz) or 2^{26} (i.e., approximately 1.5 Hz).

In this design, there are three snakes in a cave, where the cave region equals 16 LEDs on the FPGA board. The first snake is represented as one single LED (i.e., Snake1). The second snake is represented as two consecutive LEDs (i.e., Snake11). The last snake is represented as three consecutive LEDs (i.e., Snake111).

The cave is too narrow for two snakes to be at the same position. Therefore, the snakes change their direction as they meet each other. Additionally, the snakes also change their direction when they are at the edge of the cave. The rules will be discussed in the following.

Here are the input and output constraints:

◆ **rst** signal is an asynchronous positive reset.

It will reset the three snakes at the following position:

Snake1 (LD15) ●○○○○○○○○○○○○○○○○ (LD0)

Snake11 (LD15) ○○○○●●○○○○○○○○○○○○ (LD0)

Snake111 (LD15) ○○○○○○○○○○○○○○○●●● (LD0)

The LEDs will show like:

(LD15) ●○○○●●○○○○○○○○●●● (LD0)

The initial direction of each snake: Snake1 and Snake11 to the right, and Snake111 to the left.

- ◆ **en** signal is a signal to enable the operation:
 - en == 0: Hold the LEDs unchanged.
 - en == 1: The Snakes will move accordingly.
 - ✧ Snake1 moves at 6 Hz.
 - ✧ Snake11 moves at 3 Hz.
 - ✧ Snake111 moves at 1.5 Hz.



- ◆ When moving, each snake moves one step to the right or left.

For Snake1:

Original position:

(LD15) ●○○○○○○○○○○○○○○○○○○ (LD0)

Move right one step:

(LD15) ○●○○○○○○○○○○○○○○○○○○ (LD0)

For Snake11:

Original position:

(LD15) ○○●●○○○○○○○○○○○○○○○○ (LD0)

Move right one step:

(LD15) ○○○●●○○○○○○○○○○○○○○○○ (LD0)

For Snake 111:

Original position:

(LD15) ○○○○○○○○○○○○○○○○○●●● (LD0)

Move left one step:

(LD15) ○○○○○○○○○○○○○○○●●●○ (LD0)

- ◆ The direction policies:

- Snake1 will change to move to the right when it reaches the cave's left edge (LED15).
- Snake111 will change to move to the left when its rightmost body reaches the cave's right edge (LED0).
- As Snake1 and Snake11 meet each other, that is, Snake1 meets the leftmost part of Snake11.

Eg: (LD15) ○○○●●○○○○○○○○○○●●● (LD0)

← →

Assume the dot with the yellow background is Snake1, and the two dots with the blue background denote Snake11.

Snake1 will start to move to the left, and Snake11 will start to move to the right.

- Similarly, as Snake11 and Snake111 meet each other, that is, the rightmost part of Snake11 meets the leftmost part of Snake111.




Eg: (LD15) ○○○●○○○○○   ○○ (LD0)

← →

Assume the dots with the yellow background are Snake11, and the dots with the blue background are Snake111.

The Snake11 will start to move to the left and Snake111 will start to move to the right.

- If Snake1, Snake11 and Snake111 stick together, that is, Snake1 meets the leftmost part of Snake11 and the rightmost of Snake11 meets the leftmost part of Snake111.

Eg: (LD15) ○○○○○○    ○○○○○○ (LD0)

← stay →

Snake1 will start to move to the left. Snake11 will stay still. And Snake111 will start to move to the right.

- When two or more snakes meet at the edge and results in a traffic jam (i.e., some snake cannot follow the direction policies, which also called that the specific snake is locked). Simply let the locked snake stay in the same position until other snakes move away. The locked snake will resume moving if there is available space.



Eg: (LD15)   ○○○○○○○○○○●●● (LD0)

In this example, Snake1 is locked. It must stay in the same position until Snake11 moves. In this case, Snake1 must wait for one clock cycle of 6 Hz.

NOTICE:

For simplicity, the direction policies only apply when the snake needs to move. For example:

At t cycle of 6Hz clock: when both Snake1 and Snake11 are triggered to move:

(LD15) ○○  ○○  ○○○○○○○○●●● (LD0)

→ ←

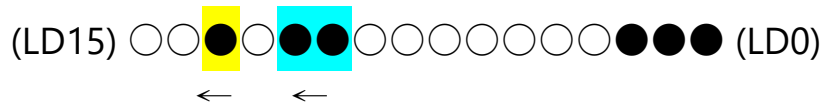
At t+1 cycle of 6Hz clock: now they meet each other. At this point, Snake1 is triggered to move again, while Snake11 is not yet triggered. Therefore, Snake1 needs to follow the direction policy for its next move. But Snake11 doesn't need to move. So Snake11 simply keeps its direction.

(LD15) ○○○   ○○○○○○○○●●● (LD0)

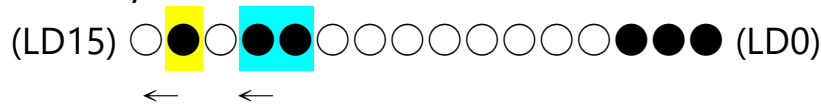
← ←

At t+2 cycle of 6Hz clock: Both Snake1 and Snake11 are triggered to move. They move

to the left.



At t+3 cycle of 6Hz clock: Both Snake1 and Snake11 move to the left again.



You should use the following template for your design and name the file as “lab3_3.v”.

```
module lab3_3 (
    input clk,
    input rst,
    input en,
    output [15:0] led
);

    // add your design here

endmodule
```

Demo: <https://youtu.be/ZFykig5vi18>

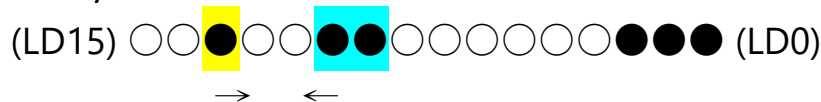
Questions and Discussion

Please answer the following questions in your report.

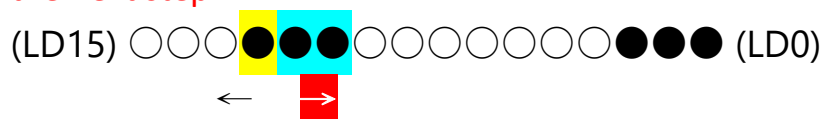
- In lab3_1, rst has the highest priority than any other signal (in most hardware designs, it's true as well). How do you implement that?
- In lab3_3, we simplify the direction policies of snakes to only apply at the moment the snake is going to move. What if the policies affect every time snakes meet each other? (Simply explain what you are going to change in your code).

Ex:

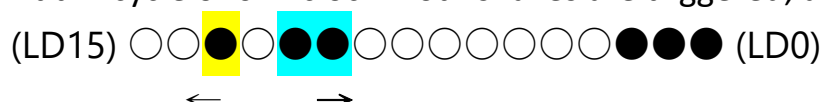
At t cycle of 6Hz clock: when both Snake1 and Snake11 are triggered, they move.



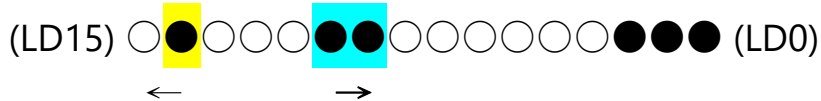
At t+1 cycle of 6Hz clock: now they meet each other. At this point, Snake1 is triggered but Snake11 isn't yet. This time, **we want both snakes to change their direction for the next step.**



At t+2 cycle of 6Hz clock: Both snakes are triggered, and they will both move.



At t+3 cycle of 6Hz clock:



Report Guidelines

Your report should include but not limit to the items (discussed in Labs 1 and 2):

- A. Lab Implementation (35%)
- B. Questions and Discussions (50%)
- C. Problem Encountered (10%)
- D. Suggestions (5%)

Attention

1. You should hand in three Verilog files, **lab3_1.v**, **lab3_2.v**, and **lab3_3.v**.
2. Finish the modules from the template, and integrate them in **lab3_1.v**, **lab3_2.v**, and **lab3_3.v**. **Put your clock divider inside lab3_1.v, lab3_2.v, and lab3_3.v as well.**
3. You should also hand in your report as **lab3_report_[StudentID].pdf** (e.g. **lab3_report_111456789.pdf**).
4. Please do not hand in any compressed files, which will be considered as an incorrect format.
5. You should be able to answer the questions from TA during the demo.
6. You need to generate the bitstream before the demo.
7. If you have any questions about the spec, feel free to ask on the EECLASS forum.