Digital Lab 3:

Experiment4:

LED Light Control Circuit

Date: 2023/11/09

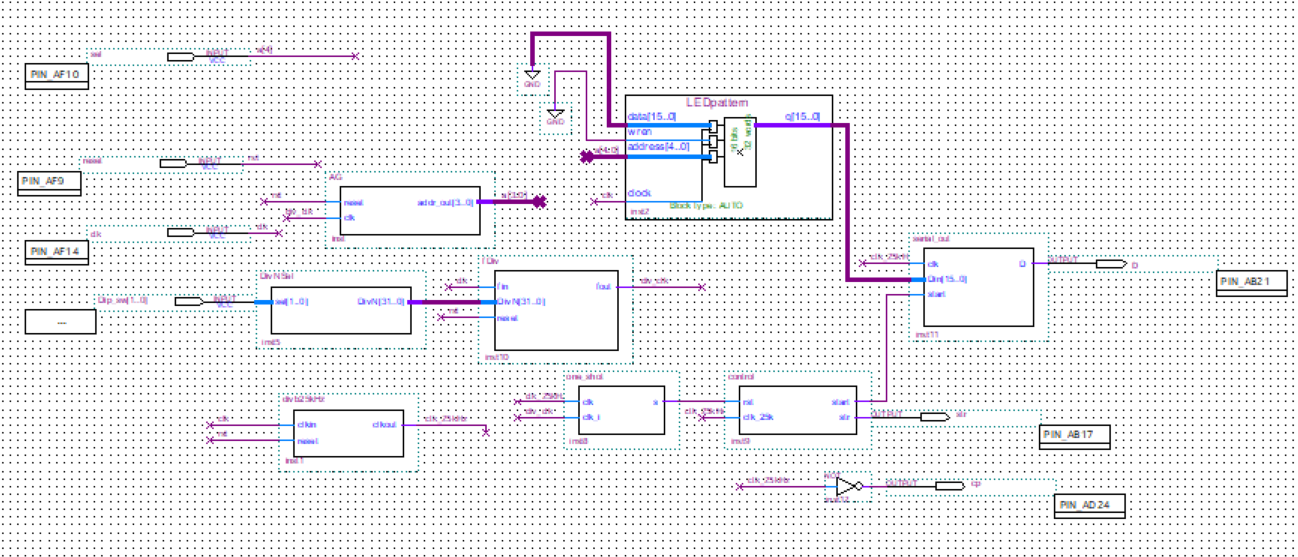
Class: 電機三全英班

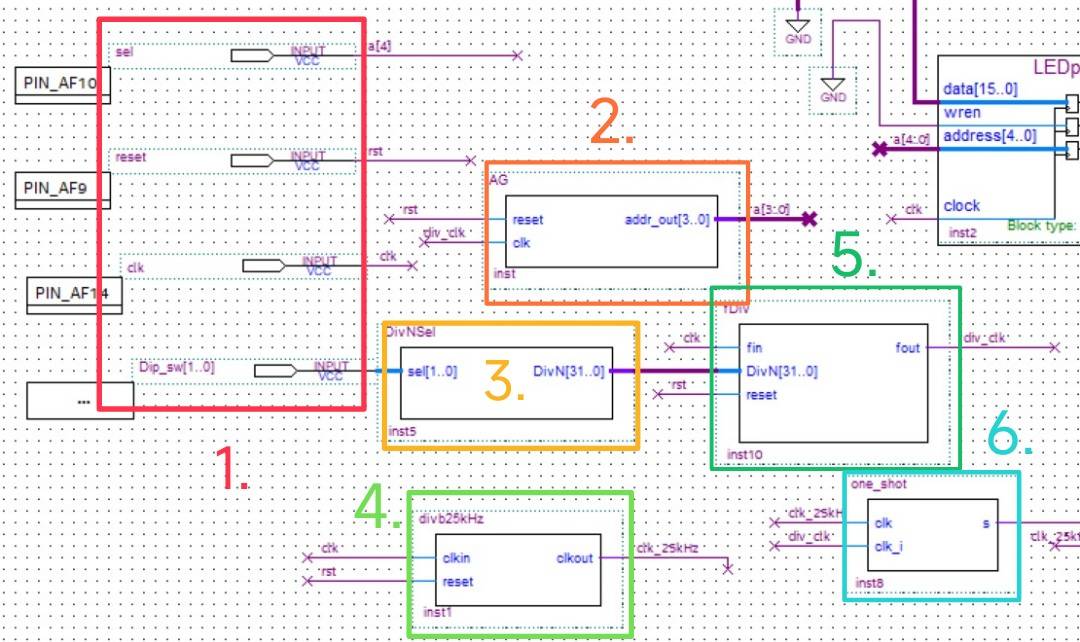
Group: Group 11

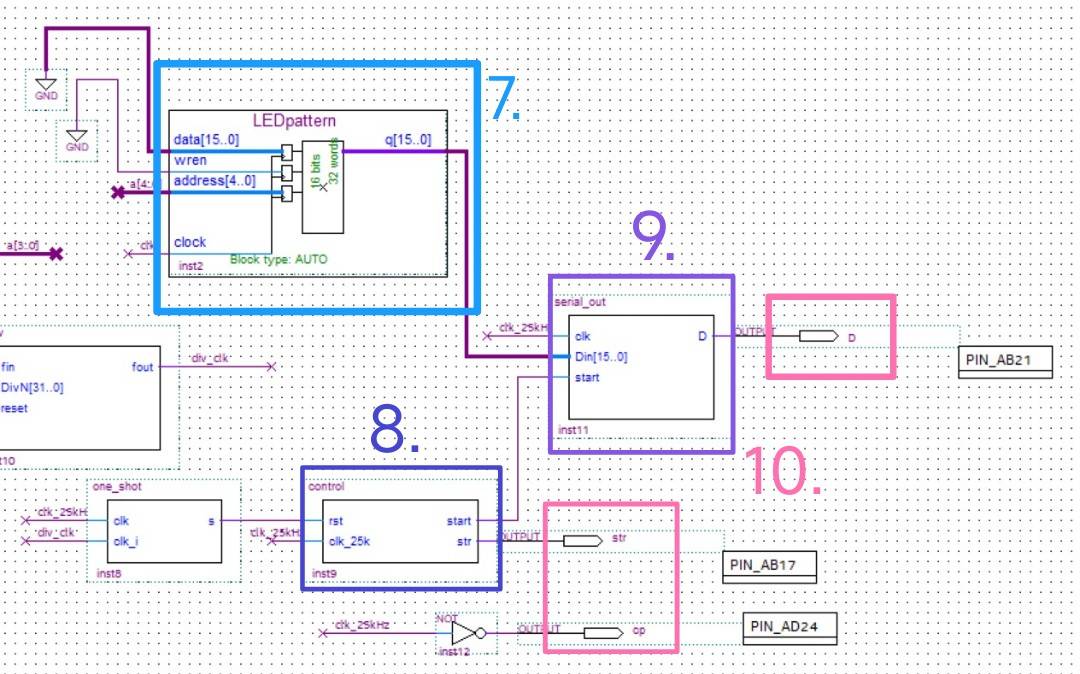
Name: B103105006 胡庭翊

1. **Block Diagram**

Structure:



Function: 



Descriptions:

1. Inputs

Here, to control the circuit, we have one bit of reset, a 50M Hz clock input connected to the FPGA board, a two bit dip switch of speed control for selecting frequency , and a select bit to choose the output pattern mode of the LED which is connected to the memory.

1. Address Generator

Generates the 4-bit address values required for memory. When the "rst" signal is raised, the address is reset to zero, allowing the memory to start outputting data from the beginning.

1. Division Selector

Select the needed frequency corresponding to the 2-bit dip-switch, the possible frequencies that are needed are 2Hz, 4Hz, 6Hz and 8Hz.

1. 25kHz Frequency Divider

Convert 50MHz of the clock on the FPGA board into 25kHz.

1. Frequency Divider 50M/divHz(iHz):

Converts the 50 MHz signal on the board into 2 Hz, 4 Hz, 6 Hz, and 8 Hz signals to control the speed of switching the stored patterns in memory.

1. One Shot

Designed using a state machine, it generates a pulse signal of length 1 clock cycle (25 kHz) after a longer input signal's positive edge trigger.

Transfer the low frequency into an active high frequency circuit module.

1. Memory

Megafunction-generated memory module (16 bits \* 32 words) that stores two different LED patterns, as provided in the attached MIF file.

1. Control

Provides the clock signal to the IC. As Serial Out uses a positive-edge trigger, an inverter is applied to the IC's CLOCK (CP) signal to make it negative-edge triggered.

This ensures that data is written to the IC when it's ready, synchronized with the

negative edge trigger.

1. Serial Output

When "start" is set to 1, the input 16-bit value is stored in the register.

After "start" becomes 0, on each positive edge of the clock, the highest bit of the

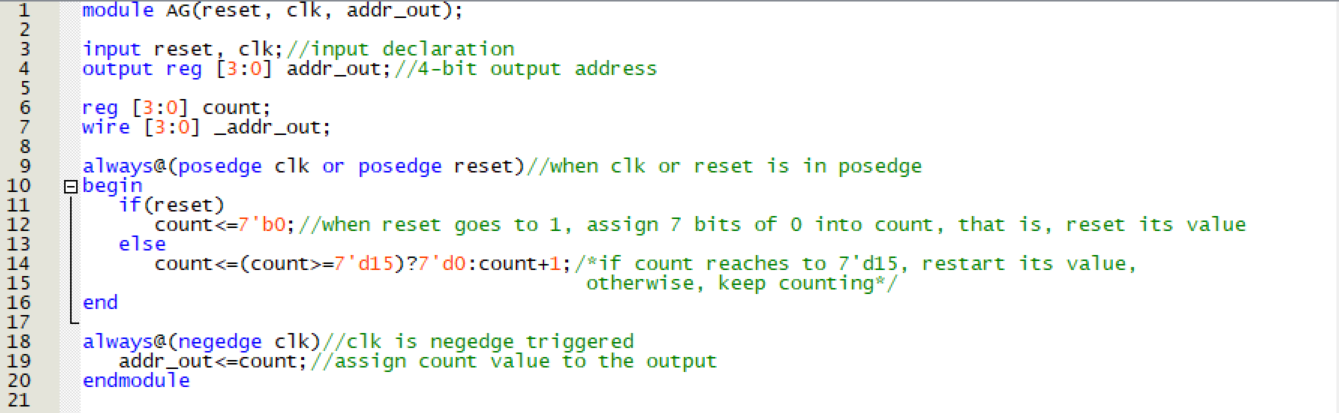
register is sequentially sent out to D

1. Outputs

Pins: str, CP, D

Control the latch enable, input clock, and data for each IC separately.

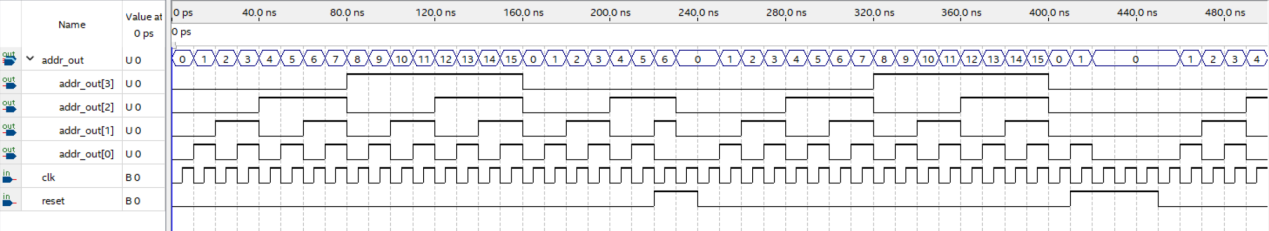
1. **Address Generator**
2. Verilog Code and Comment



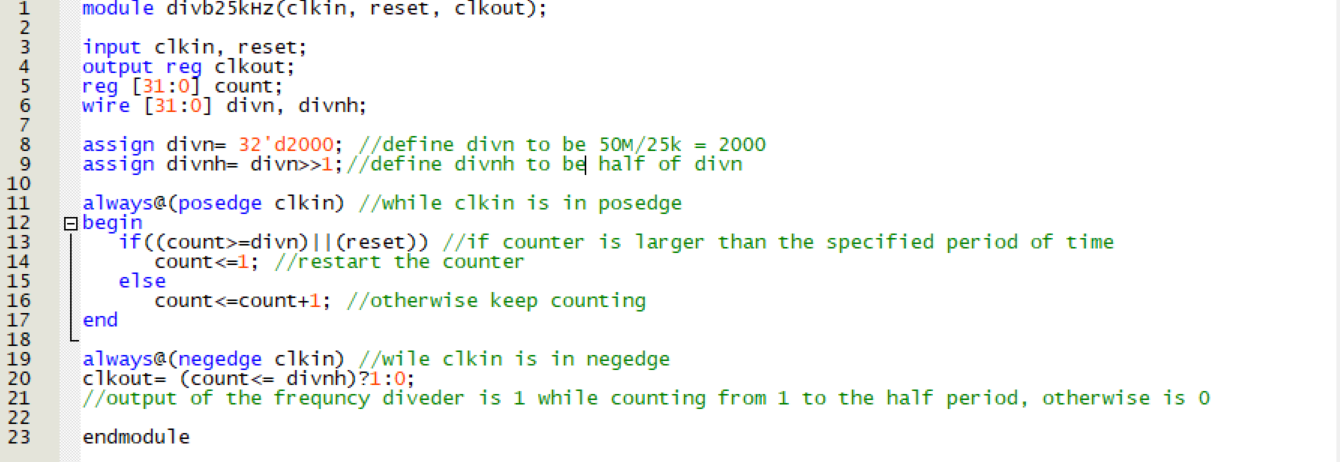
1. Simulation

The address would be generated when either the clock or reset is in posedge.

The value of count will be reset to 0 if reset is 1, otherwise, it will keep increasing until it reaches to 15, and will restart counting from 0.



1. **25k frequency divider(divb25kHz)**
2. Verilog Code and Comment

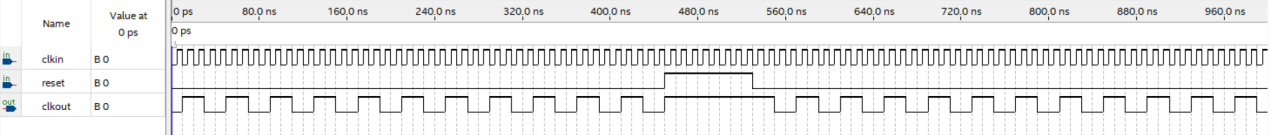


1. Simulation

We replace the frequency division ratio 32’d2000 into 32’d4 in order to make the simulation better be seen.



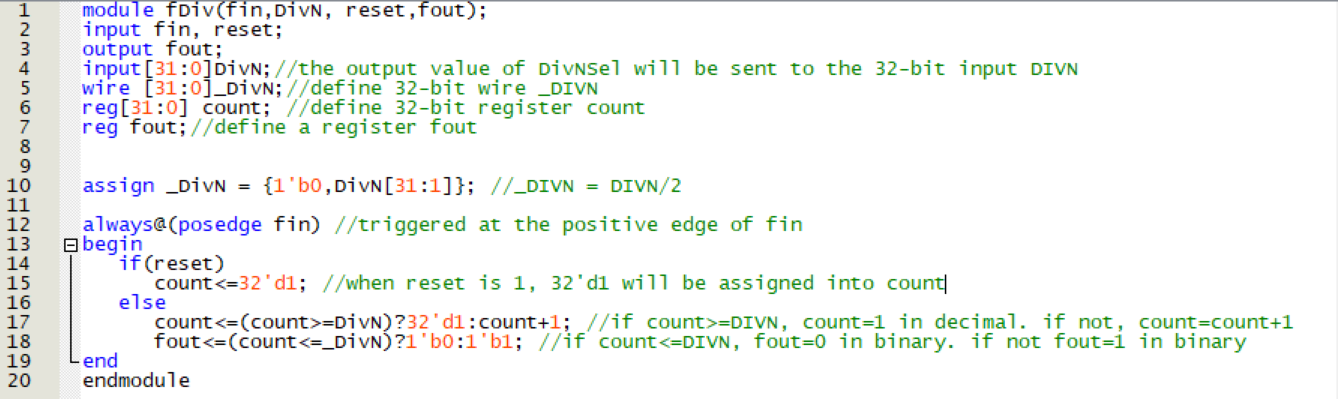
After modifying, we generate a frequency divider that can convert the input frequency into 1/4 times of the frequency.



When clkin is in posedge, the value of count will be varied, while if reset is 1 at this moment, then 1 will be assigned into count.

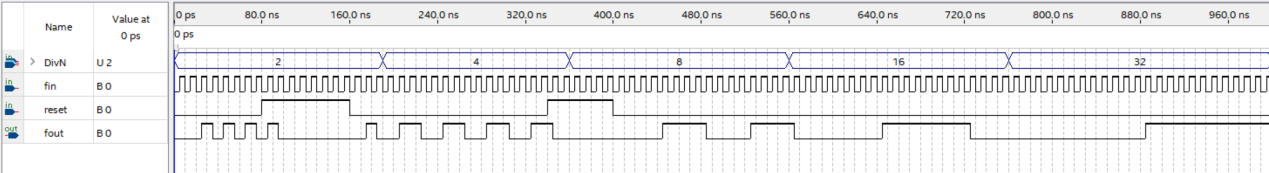
When clkin is in negedge, clkout will compare the value of count and the set value divnh, and output the correspond signal.

1. **fDiv (i frequency divider)**
2. Verilog Code and Comment



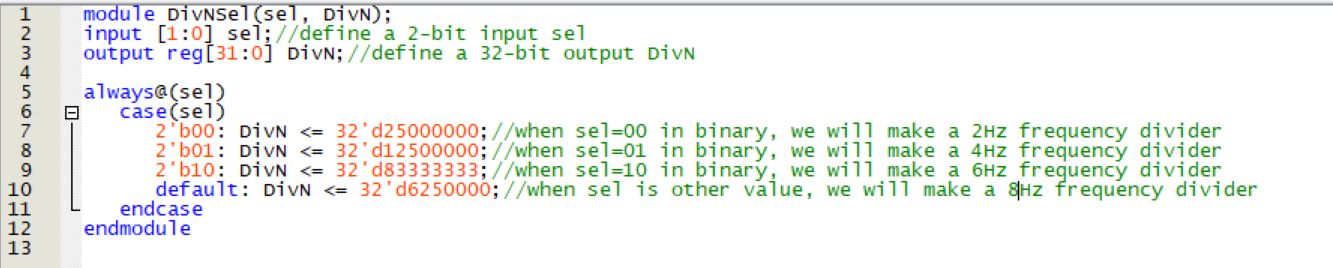
1. Simulation

The value of DivN is the output of the DivNSel, we can modified our frequency by simply changing the input frequency.



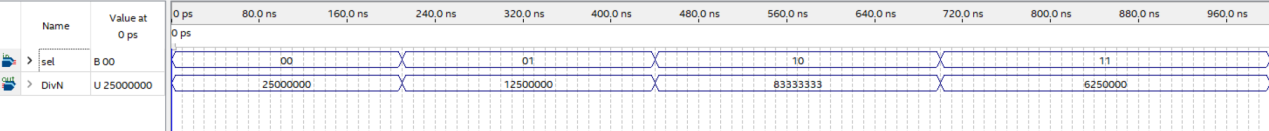
The operation of output signal and reset both happens when fin is in posedge, this is the reson why when the input signals change, there is a slice of delay in output.

1. **Division Selector (DivNSel)**
2. Verilog Code and Comment

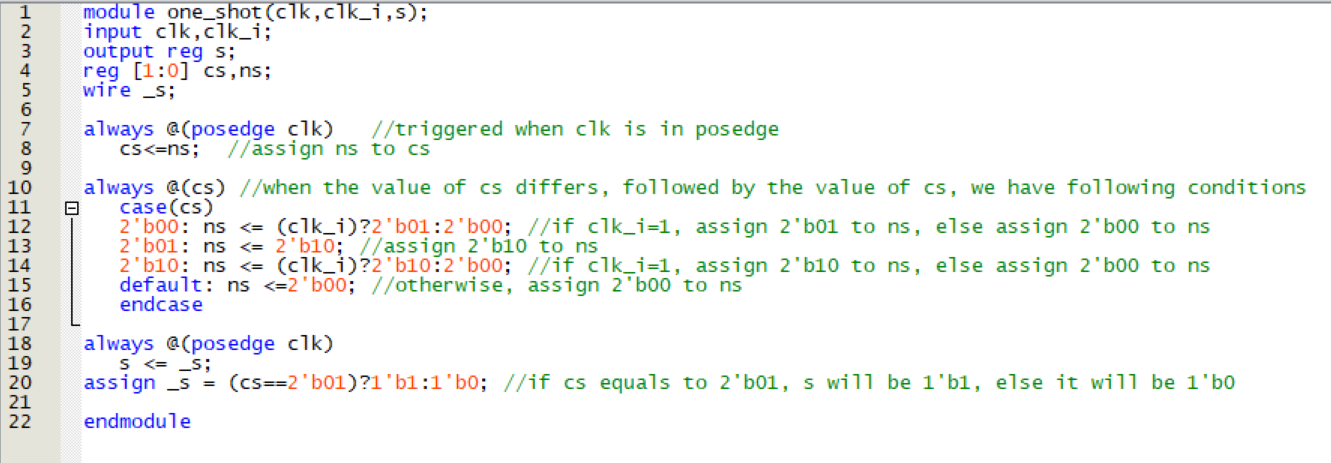


1. Simulation

The output value of DivN is depended on the input value of sel.

****

1. **One Shot**
2. Verilog Code and Comment

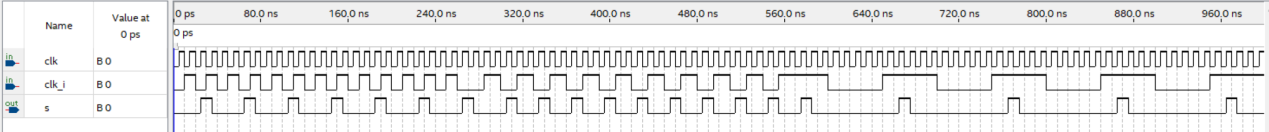


1. Simulation

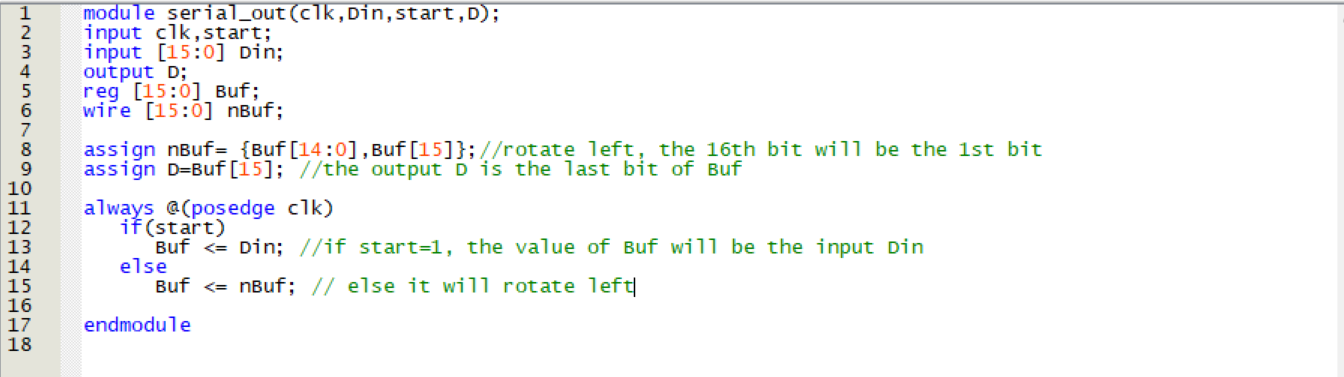
The input of clk is the output of 25kHz frequency divider, while the input of clk\_i is the output of changeable frequency divider.

When clk is in posedge, one\_shot will judge the value of clk\_i, and assign the value of cs corresponding to the states.

After that, the value of s would be assigned corresponding to the status of cs.

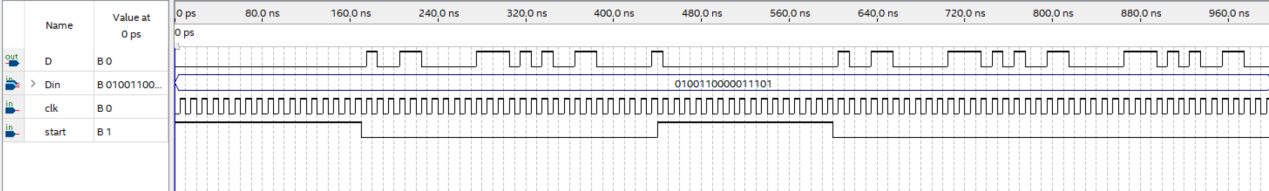


1. **Serial Out**
2. Verilog Code and Comment

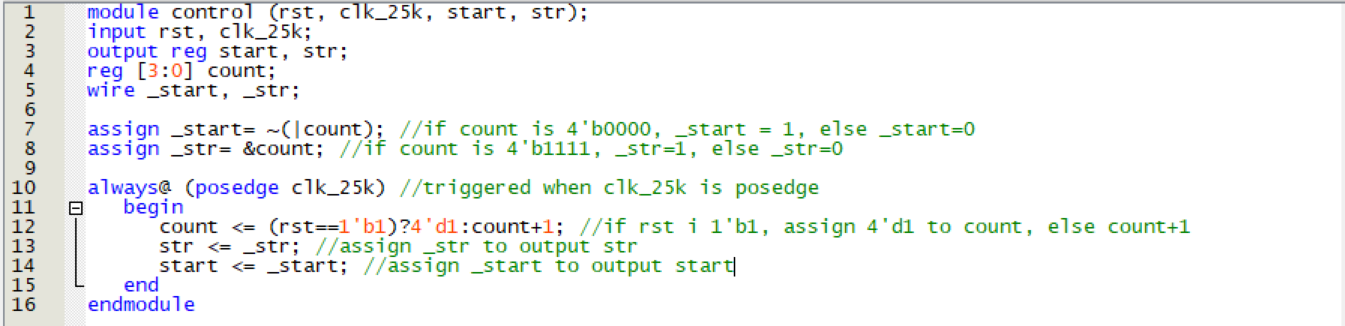


1. Simulation

When clk is in posedge with start =1, output the largest bit of current registor Buf, else when clk is in posedge but start=0, rotate left the current registor and output its largest bit.

****

1. **Control**
2. Verilog Code and Comment

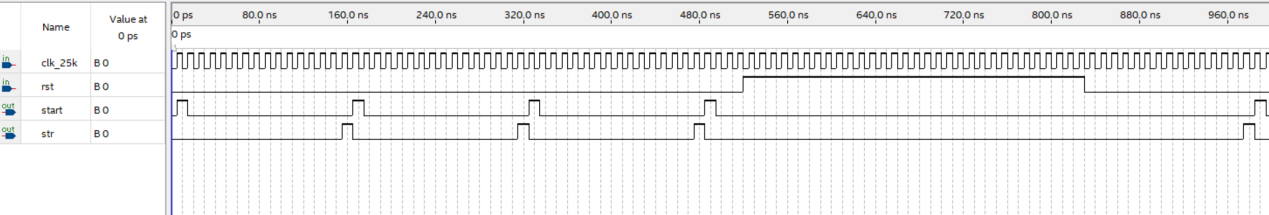


1. Simulation

In every posedge clk\_25k, count will keep counting until rst is 1’b1, and the value of str and start will be assign at this moment too.

When count is 4’b0000, the value of start will be 1, and when count is 4’b1111, the value of str will be 1, in other case, they are 0.

We know that after 1111, count will be 0000 again, that is the reason why after str=1, start will be 1 too.



1. **Reflection**

The practical part of the experiment required us to solder circuits, a task I found a bit tricky and felt like it took up time and resources without a clear need. Even though the first time soldering circuit boards was kind of fun, thinking about future experiments involving soldering made me a bit tired, especially remembering the five hours we spent on it last time.

Moreover, the experiment guide didn't come with any ready-to-use code. This meant we had to tweak our existing modules, which was a bit tough. However, it turned out to be a good chance to improve our skills.