Chengyu Wang, Simeng Pang and Yaowei Guo

Workshop1

Clock Dividers

1. The period of the clk_en signal is 1.31 ms. The figure is shown below:

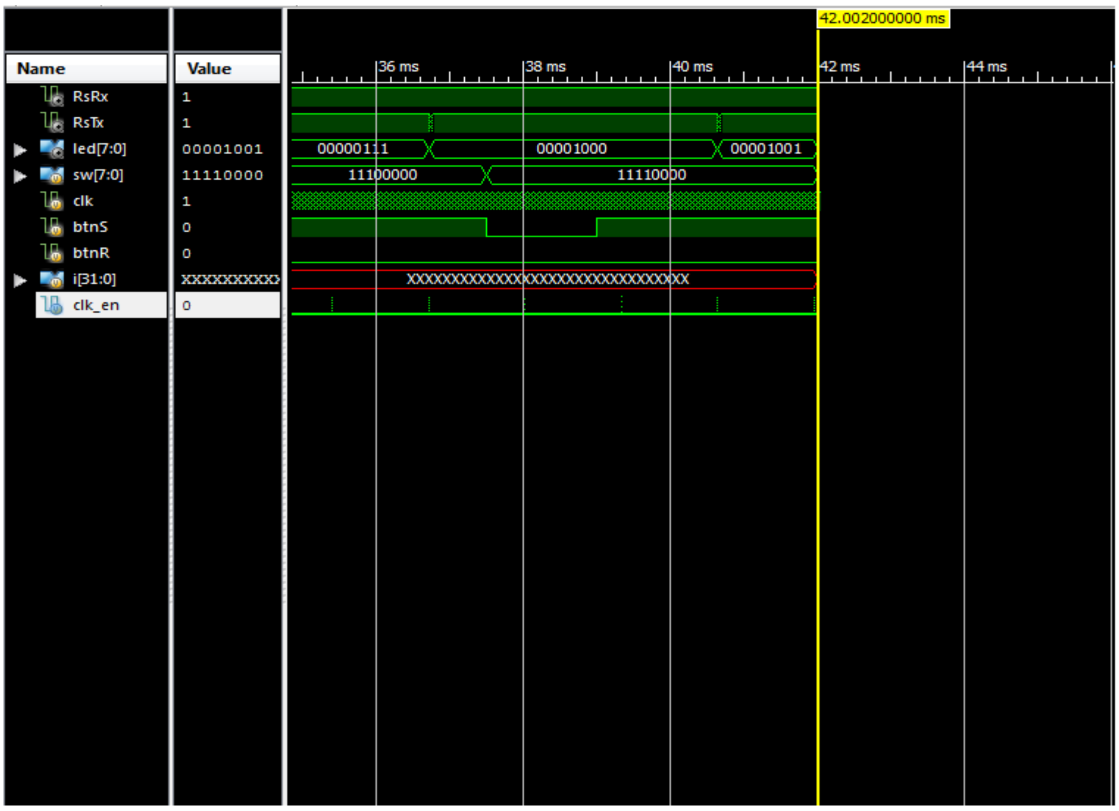


Figure 1: A waveform picture that captures rising edges of clk_en

2. We have P = 1.31ms, and T = 10ns.

$$D = \frac{T}{P} * 100\% = \frac{10ns}{(1.31 * 10^6)ns} * 100\% = 0.00076\%$$

3. The value of clk_dv signal is 0 when clk_en is high.

4.

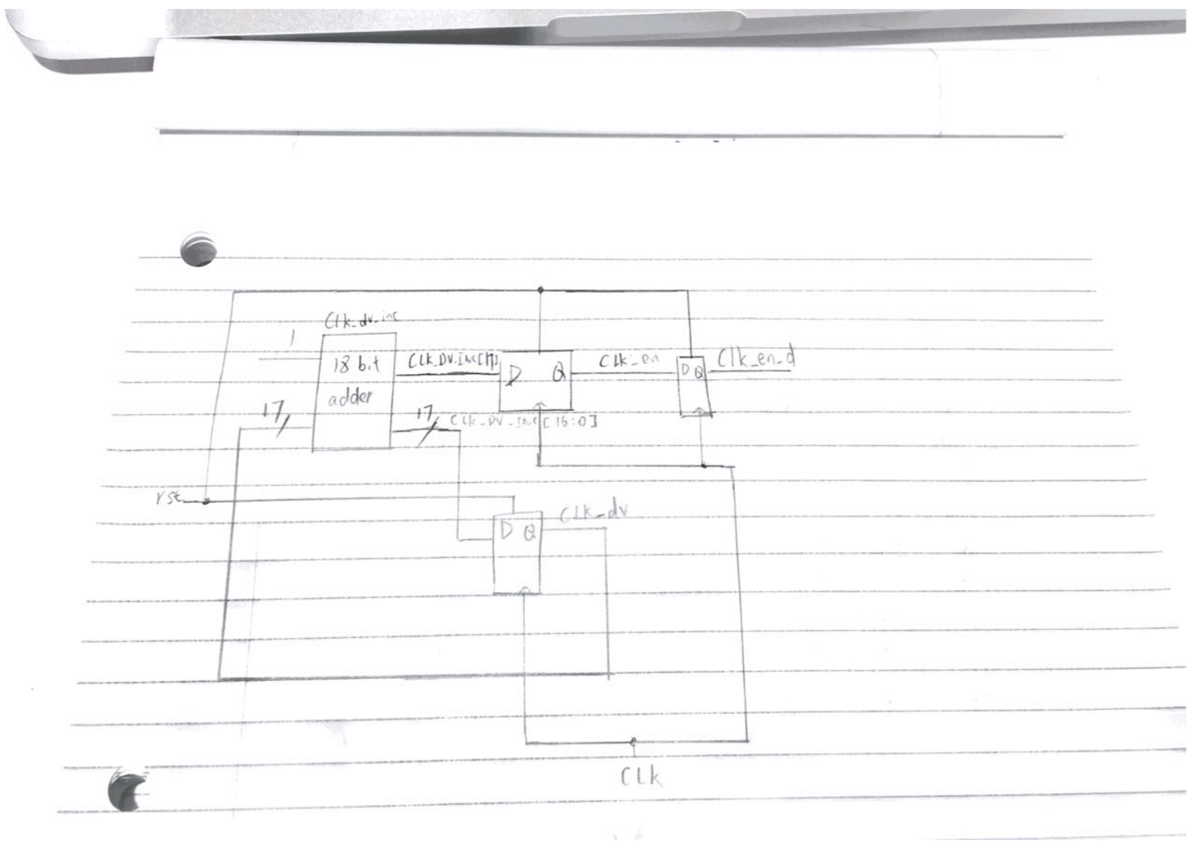


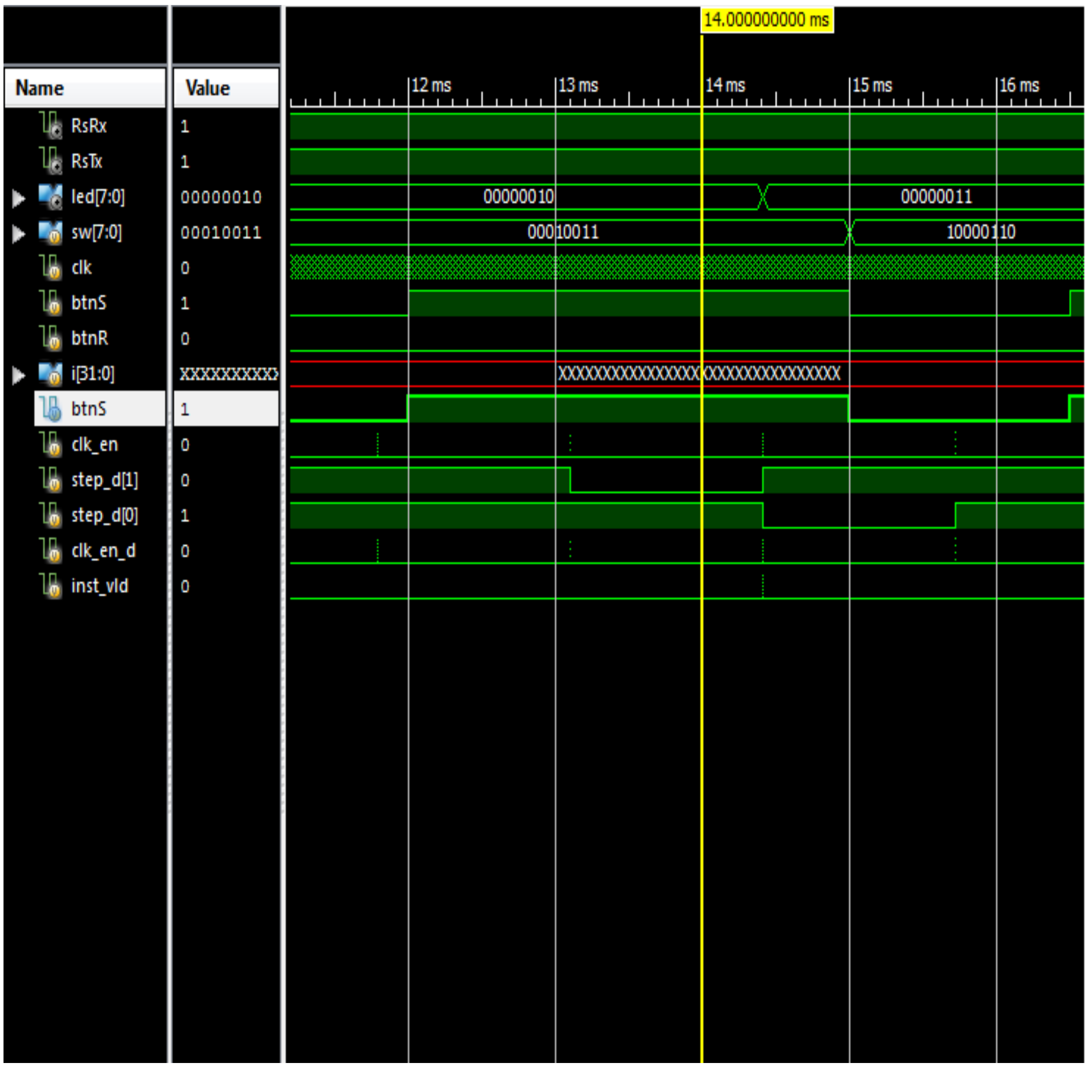
Figure 2: Diagrams of clk_dv, clk_en and clk_en_d

Debouncing

1. This expression is used to record whether the button was pressed one cycle ago. The value of step_d cam be changed only when clk_en is 1, and it's updated in the next cycle since it's sequential logic. However, in the next cycle, clk_en already becomes 0. So, ~step_d[0] & step_d[1] & clk_en will always be evaluated to 0. clk_en_d is the clk_en signal delayed by one cycle. When clk_en_d is 1, step_d has already been updated, so we can determine whether the button was pressed or not one cycle ago.

2. It will not make the duty cycle 50%. In the old case, the clk_en signal is high for very short an interval because once the clk_dv_inc[17] becomes 1, clk_dv becomes will become 0. Then clk_dv_inc will be reset to 1 and clk_dv_inc[17] becomes 0 again. So, clk_en is 1 every 2^{17} cycles. If we use clk_en <= clk_dv[16] instead, clk_en signal will remain 1 for a much longer time because when clk_dv[16] becomes 1, clk_dv_inc won't be reset immediately. So, clk_en will remain 1 until clk_dv_inc[17] becomes 1. It's nearly half of the total cycle time. So, the duty cycle will become much larger instead of 50%.

3.



Name	Value
RsRx	1
RsTx	1
led[7:0]	00000010
sw[7:0]	00010011
clk	0
btnS	1
btnR	0
i[31:0]	XXXXXXXXXX
btnS	1
clk_en	0
step_d[1]	0
step_d[0]	1
clk_en_d	0
inst_vld	0

Figure 3: Timing relationship of signals

4.

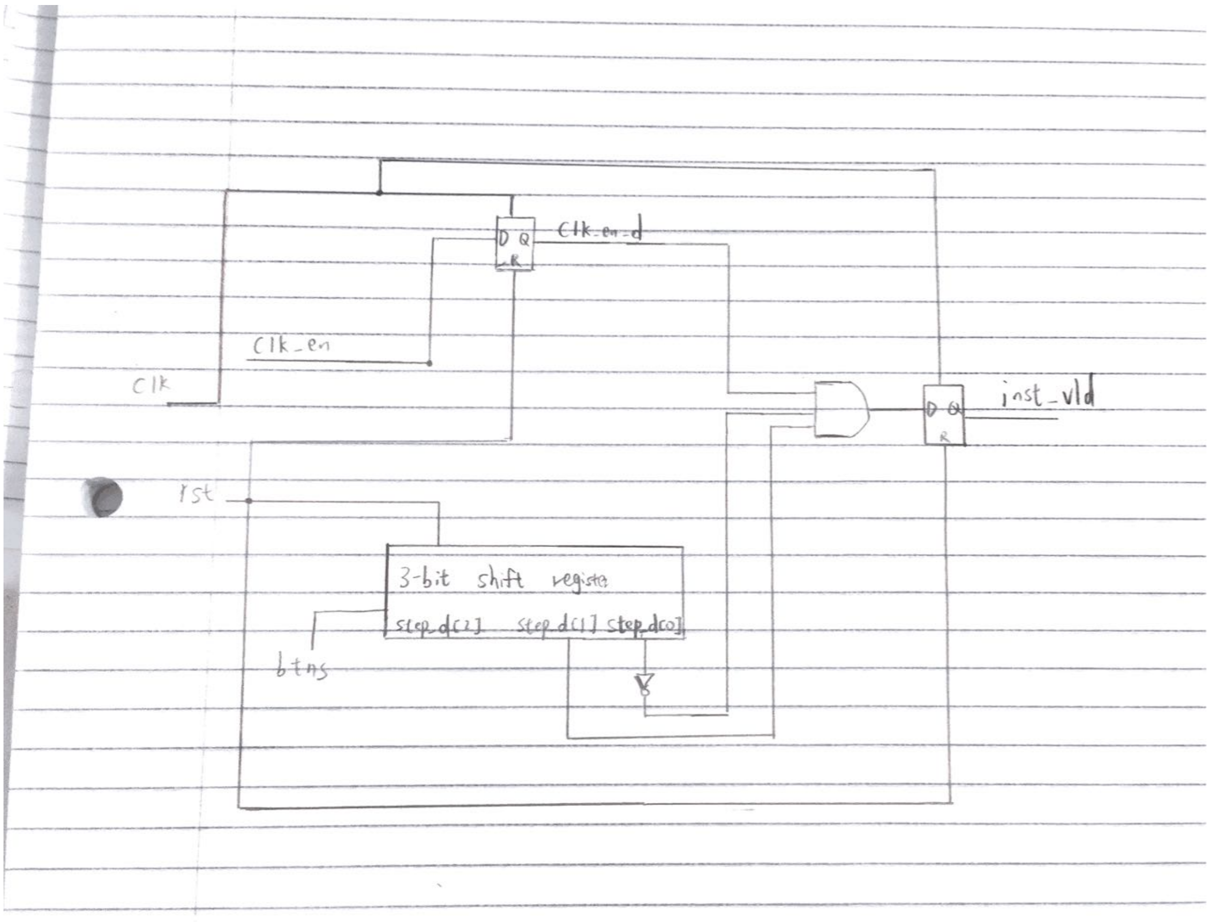


Figure 4:Diagram of signals

Register File:

1. Line 33: rf[i_wsel] <= i_wdata. This is sequential logic. Because this line uses <=, it indicates that this expression uses a non-blocking assignment; the register is written on the positive clock edge.

2. Line 35 and 36:

```
assign o_data_a = rf[i_sel_a];
assign o_data_b = rf[i_sel_b];
```
This is combination logic. We can use multiplexers to manually implement.

3.

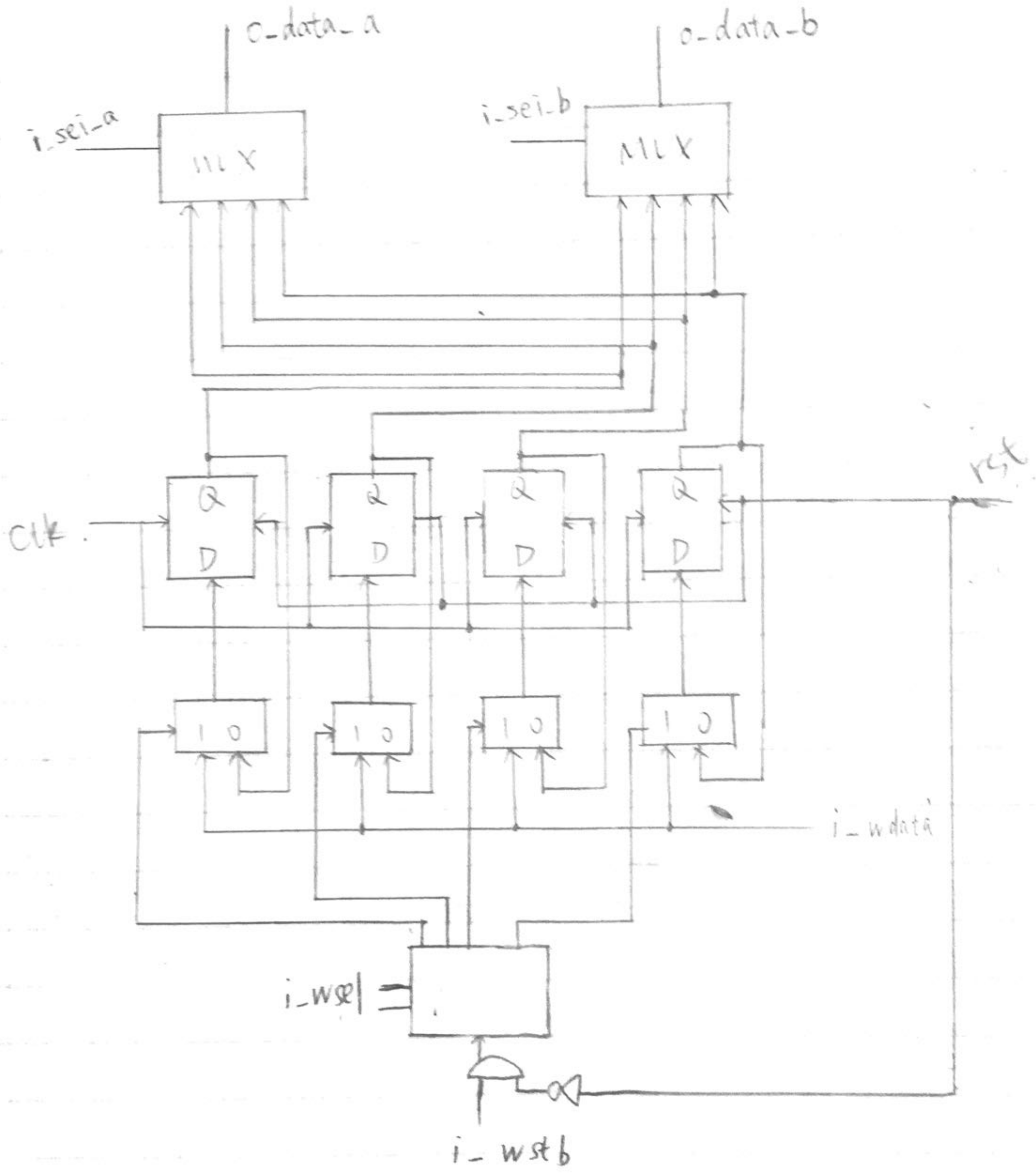


Figure 5: Circuit diagram of register files

4.

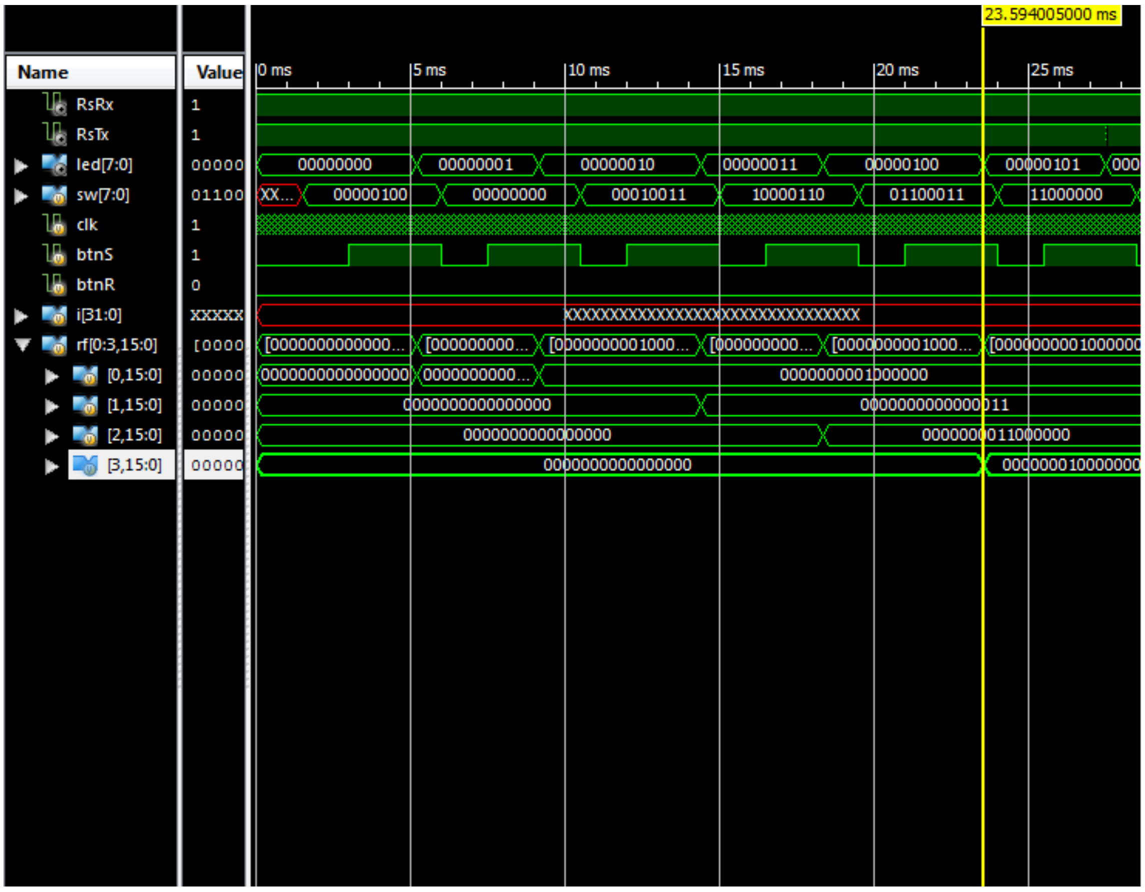


Figure 6: Waveform of the first time register 3 is written with a non-zero value