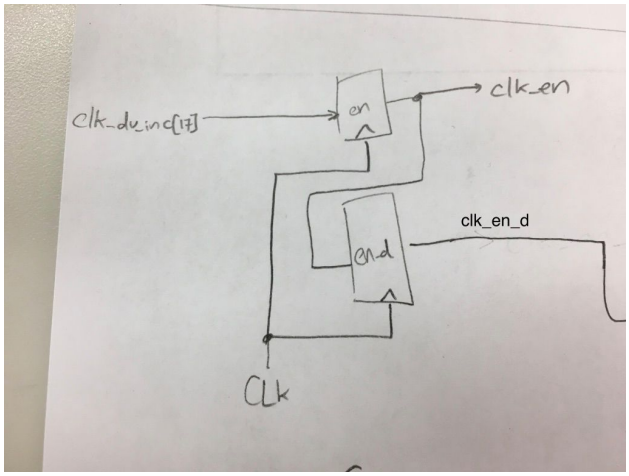
# Lab 1 Report

Robert Zalog, Jeffrey Chan, Sam Ryklansky

## Workshop 1

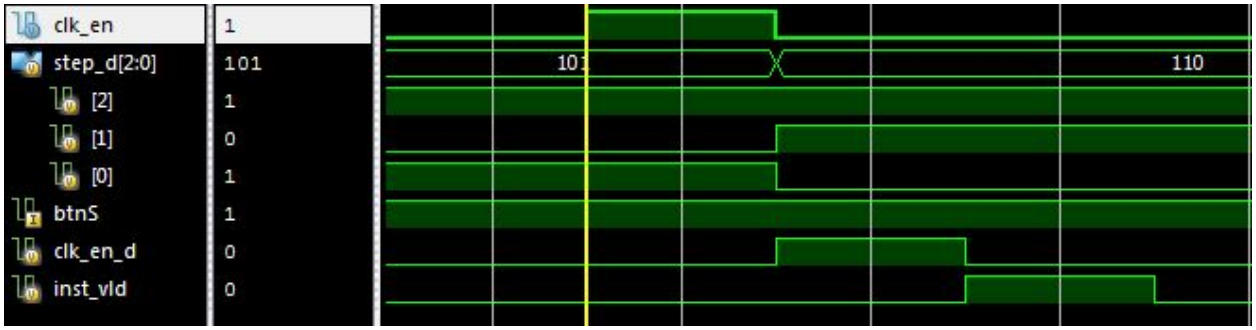
Clock dividers

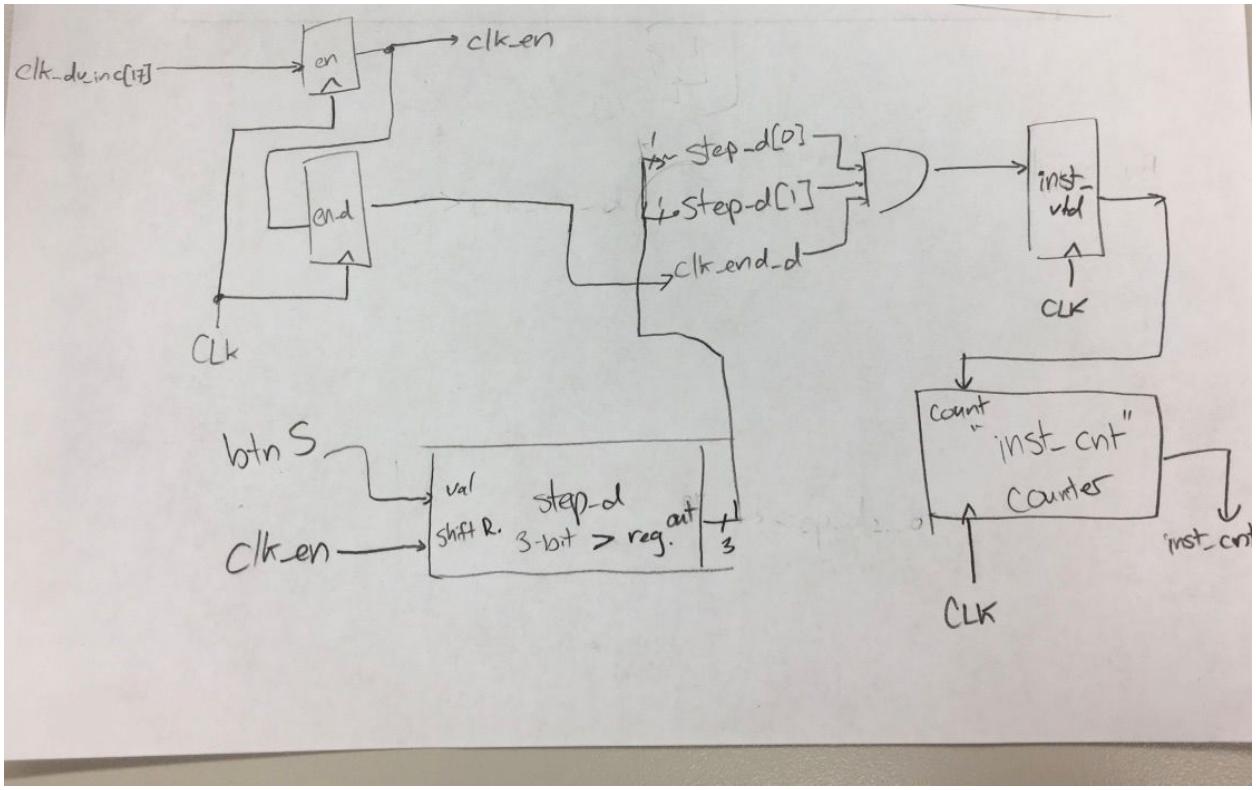
1. 

   For period, first high = .001311735s, first low = 001311745s, second high = .002622455s. Thus the period is …
2. D = (H1 - L1) / (H2 - H1).
3. When clk_en is high, the value of clk_dv is 0.
4. 


Debouncing

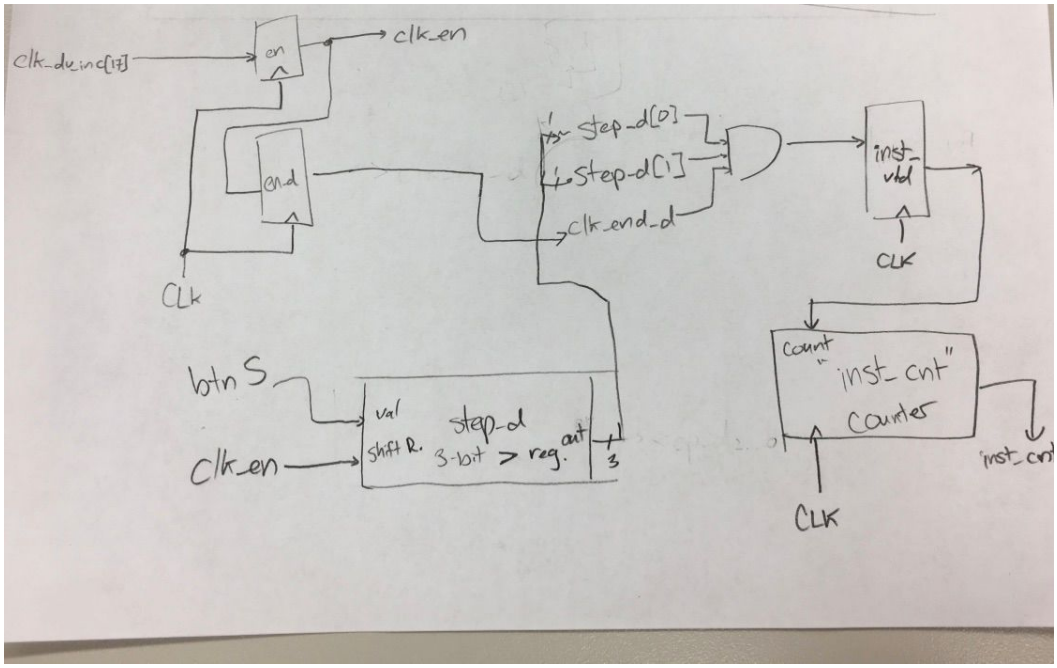
1. Once clk_en is set to 1, the signal gets sent to step_d to shift. Only after this does step_d get set to a value such that “~step_d[0] & step_d[1]” would evaluate to 1. But this value is only actually changed at the end of the clock cycle. But also, at the end of the clock cycle, clk_en goes back down to 0. Thus when inst_vld is updated, although “~step_d[0] & step_d[1]” evaluates to 1, “clk_en” will always be 0, so inst_vld will never be set to 1 and the instructions will never increment. So to fix this we add clk_en_d, which is just a one cycle delayed clk_en, that will be equal to 1 when the step_d register holds the correct value.
2. When we use the clk_en signal, we don’t use it like a normal clock signal where we only say it “goes off” on the positive edge. Instead, we just say that it rises when the value is 1. Thus if we kept it high such that the duty cycle was 50%, it would look as if clk_en fired for multiple times in a row, and all of our registers that update on clk_en would continuously update for every one of our normal clock cycles, which is not what we want. So to fix this, we only set clk_en high for the duration of one internal clock cycle, and then low for the rest of the period.

3. 


4. 


Register File

1. We can see that on line 33, “rf[i_wsel] <= i_wdata”, a register is being written a non-zero value. We can see that this is sequential logic from two things. First is that it is inside an always block that triggers on the positive edge of a clock. Second we are using the asynchronous assign operator, “<=”, rather than just using “assign” and “=” as we would do for combinational logic.
2. These lines of code are lines 35 and 36. This is combinational logic, as we can see from the use of the “assign” statement and the “=” operator. To manually implement this in a circuit you would use a MUX, which takes “i_sel_a” or “i_sel_b” as the select signal, and each value of the output of the four “rf”’s as the input, and assigns the output wires the names “o_data_a” and “o_data”b”.

3. 

4. 


## Workshop 2

1. Identify the part of the tb.v where the instructions are sent to the UUT.
   All the lines that say “tskRun…”, and more specifically in “tskRunInst” task.
2. Which user tasks are called in the process?
   All the “tskRun…” instructions listed, various tskRunPUSH, tskRunADD, tskRunMULT, and tskRunSEND.