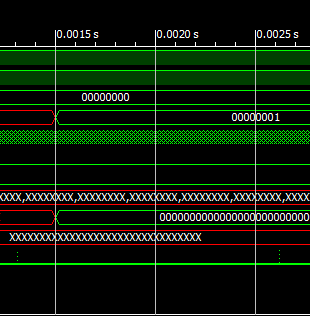
Workshop 2:

1. The part of tb.v that is responsible for sending the instructions to the UUT is the combination of the lines between 30-38 that call the tasks defined on lines 77-116. The actual instructions are sent in the task defined on lines 67-75 called tskRunInst.
2. The user tasks that are called are: tskRunInst, tskRunPUSH, tskRunSEND, tskRunADD and tskRunMULT.

Workshop 1:

Clock Dividers:

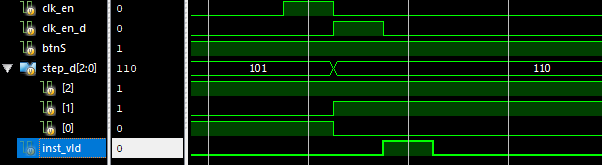
1. periodicity of 0.00131072



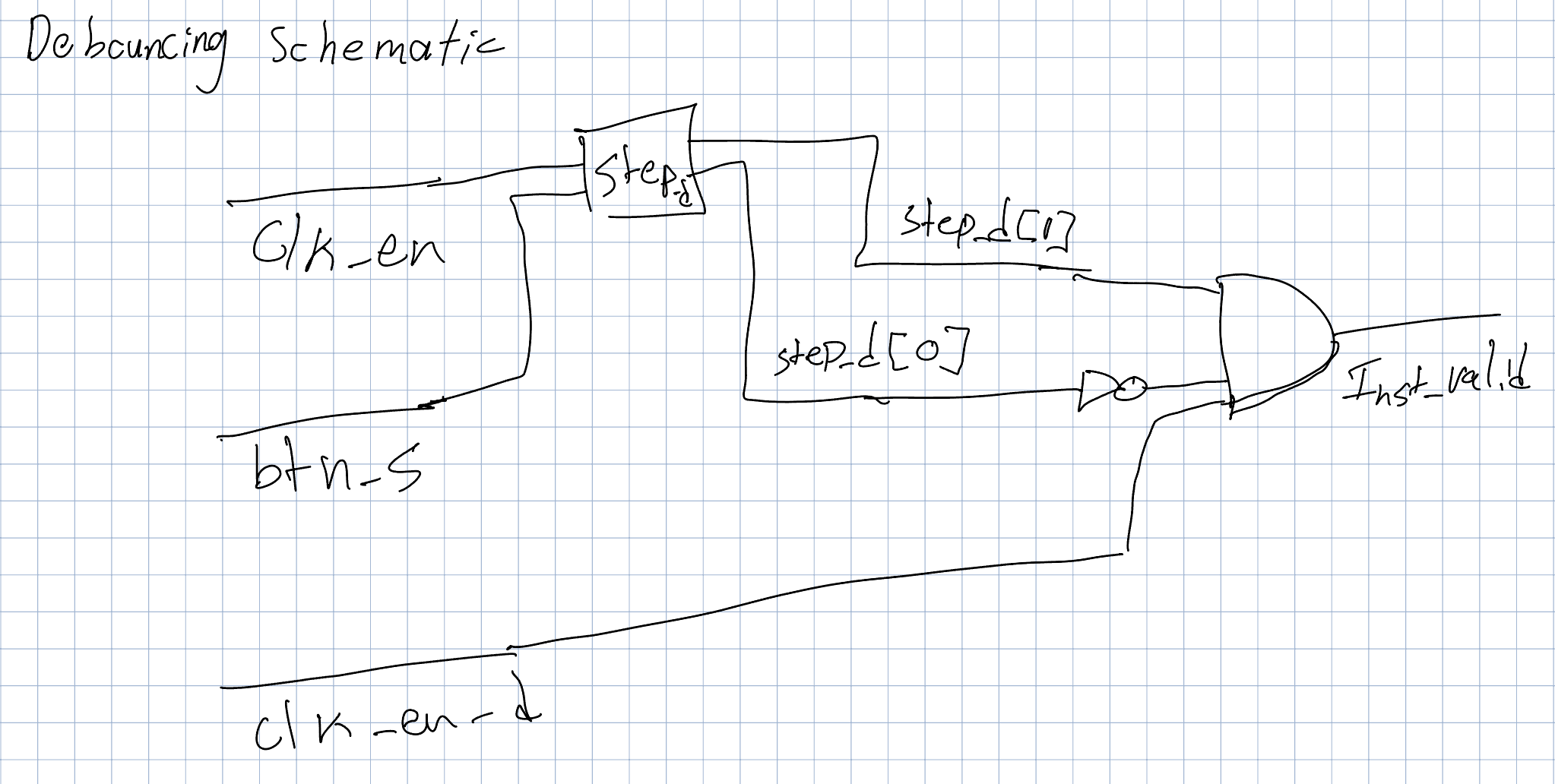
1. 0.000762939453125%
2. 00000000000000000

Debouncing:

1. Clk\_en\_d is used in this expression to make sure that instructions are only set to valid on the timestep of the divided clk (clk\_dv). We use clk\_en\_d instead of clk\_en because step\_d is set using clk\_en, and clk\_en\_d is delayed by one clock cycle from clk\_en, so if we were to use clk\_en to set inst\_vld then it would look at the old value of step\_d.
2. It will work because none of the functionality requires for clk\_en to only be high for one clock cycle. Inst\_vld still is only high for one clock cycle so instructions are only sent once and timing is set based on the base clock.







Register File:

1. Non-zero data is written on line 33: “rf[i\_wsel] <= i\_wdata;” which Is part of the control structure on lines 26-33. This is combinatorial logic because it only relies on input data.
2. Lines 35 and 36. This is sequential logic because it relies on previous data. I would use a mux to select from the register file data using sel\_a and sel\_b.

