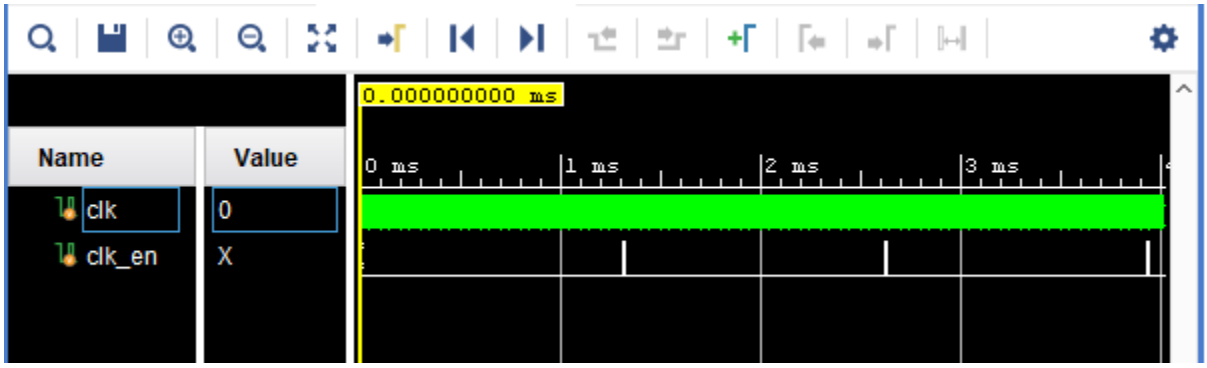
CSM152A
Team 4 (Justin and Krish)

Lab Report 1

Workshop 1:

Clock Dividers

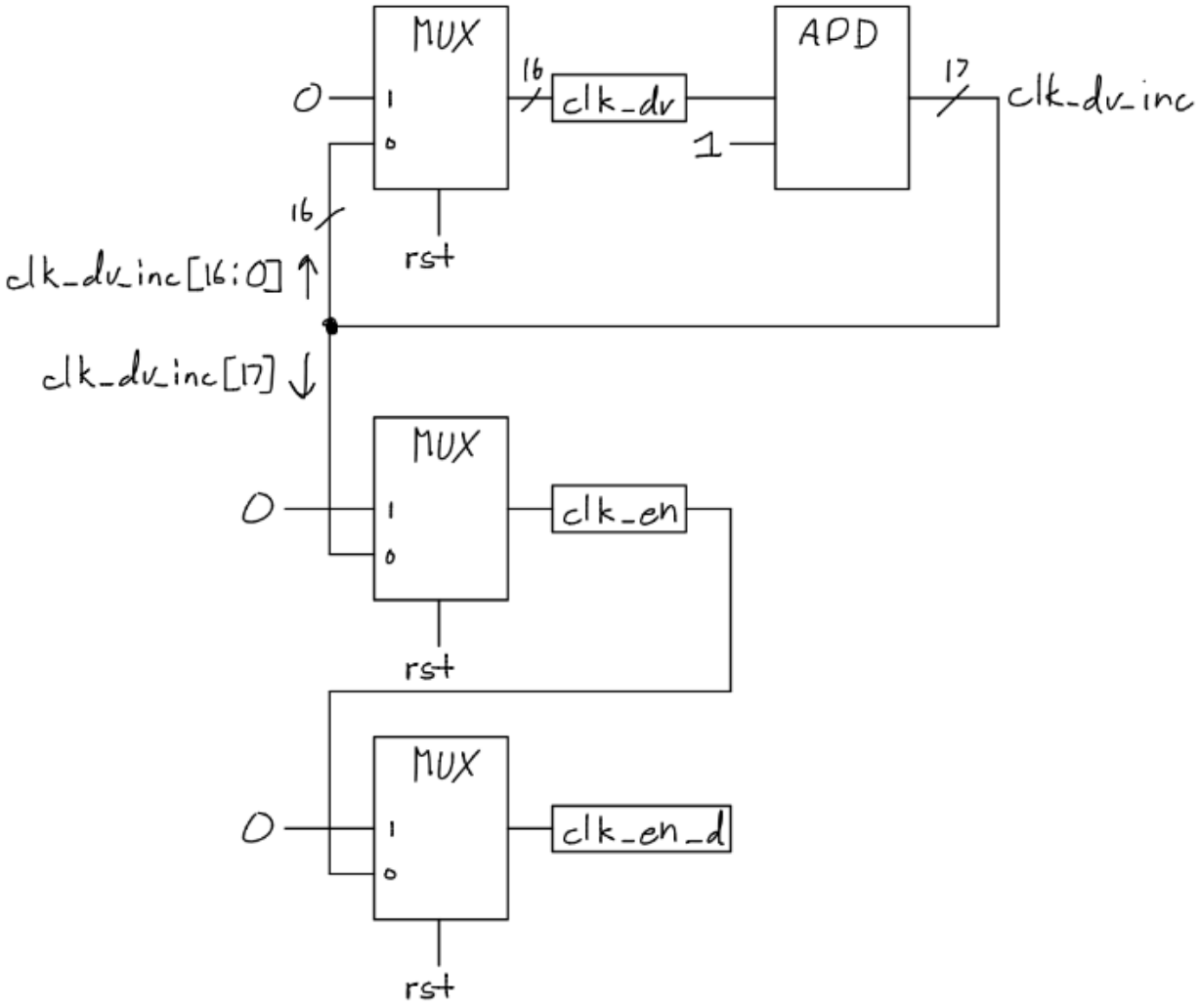
1) 

   Period = 1.31072 ms

2) P = 1.31072 ms
   T = 10^-5 ms

$$D = \frac{T}{P} * 100\% = \frac{10^{-5}}{1.31072} * 100\% \approx 7.629 * 10^{-4}\,\%$$

3) clk_dv = 0 when clk_en is high



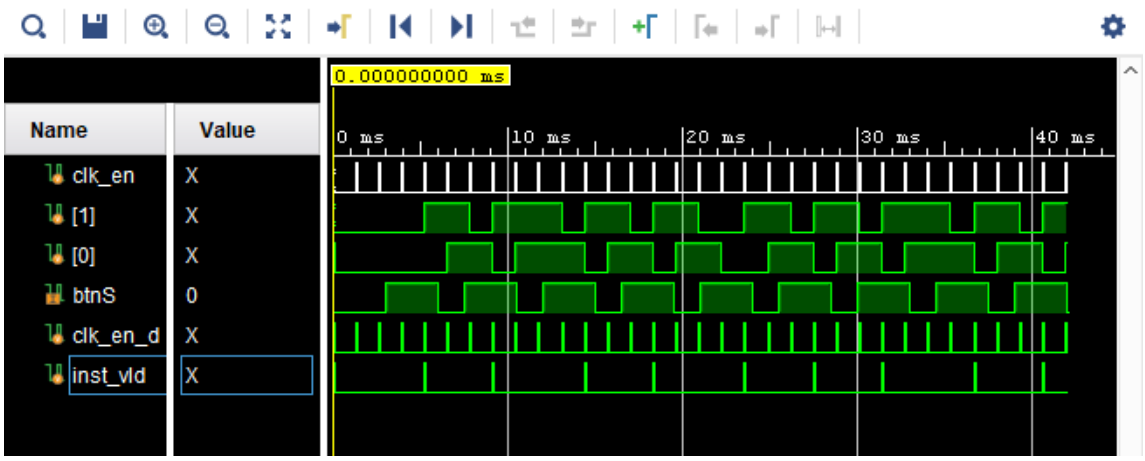
4)

Debouncing

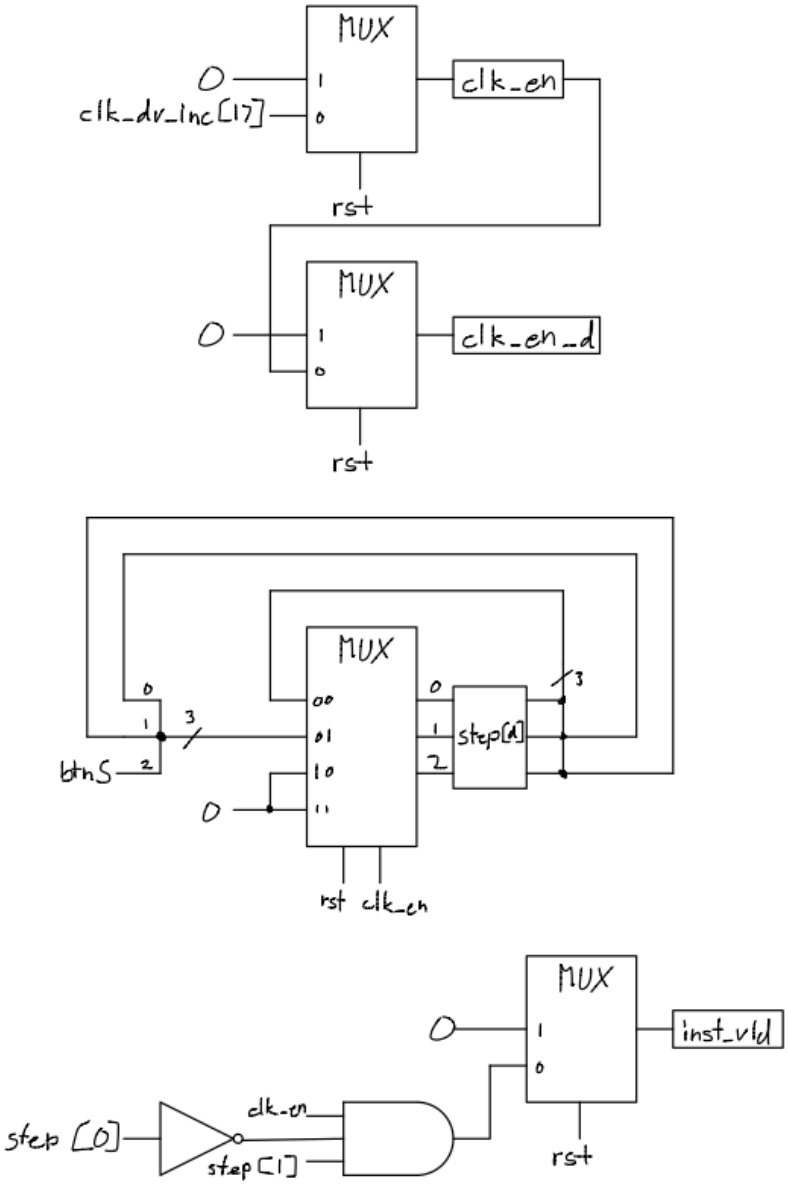
1) Clk_en_d is used for debouncing getting the signal from clk_en, When clk_en switches from a low state to a high state, clk_en_d follows suit, but it does so with a slight delay of one clock cycle. The reason we use this as separate registers is to ensure that the multiple signals aren't sent. Once the clock signal experiences an upward transition (goes from low to high), clk_en_d becomes active and temporarily blocks the next clock pulse from occurring.

2) yes, this works as intended. The value of clk_dv[16] is directly derived from clk_dv_inc[16]. As a result, the most significant bit (bit 16) of clk_dv will switch between 0 and 1 an equal number of times over a sufficient period. It will function correctly because the counts will ultimately balance out.

3)

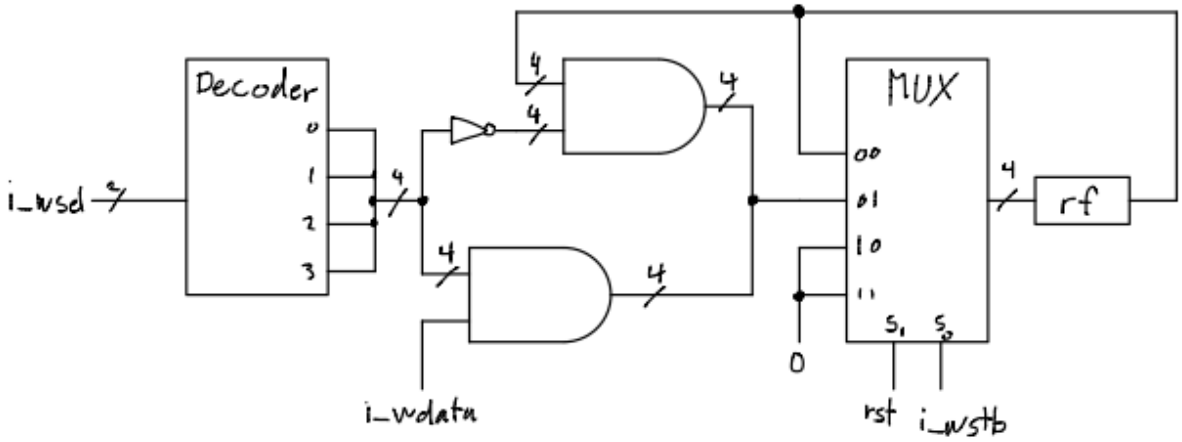

0.000000000 ms
Name
Value
clk_en
X
[1]
X
[0]
X
btnS
0
clk_en_d
X
inst_vld
X
0 ms
10 ms
20 ms
30 ms
40 ms

4)

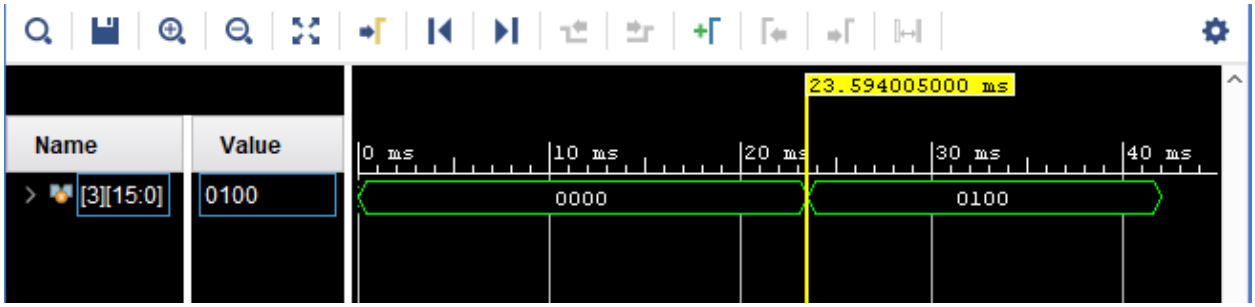

MUX
0
clk_dv_inc[17]
clk_en
rst
MUX
0
clk_en_d
rst
MUX
btnS
0
step[1]
rst
clk_en
MUX
0
inst_vld
clk_en
step[0]
step[1]
rst

Register File

1) Line 33, sequential as it uses non-blocking assignments (<=) -> rf[i_wsel] <= i_data
2) Lines 35 and 36, which is combinatorial as it uses blocking assignments (=) and thus the values would only be assigned once the statements above were executed.. If we were to manually implement this, we would use two multiplexers(4*1) to implement these two lines
3)



4)



Workshop 2

1) Instructions are sent to UUT at lines 71-73 in the task tskRunInst.
2) The tasks which are run are tskRunPUSH, tskRunADD, tskRunMULT, and tskRunSEND which then call tskRunInst.