EE254L

Divider RTL design in Verilog

Questions for the ee254_divider_simple design:

A. What happens if you divide by zero? Is the behavior of the quotient digit display on SSD1 different if you attempt to divide 3 by 0 vs. if you attempt to divide F by 0. How about 0 divided by 0?

B. If you improve the divider design to move from compute state to done state if X is equal or less than Y (instead of the current X less than Y), will the above behavior change? Does your answer to Q#A above change?

C. Why does the behavior of the next design (**ee254_divider_with_debounce**) appear to be quite different from this design for division by zero? Is it just appearance only or is it really different? Note: Look at the rate at which clk (clock going into the divider) runs in both designs

Questions on the debouncer and the divider with debouncer:

1. Briefly explain why the N_dc parameter was changed to 4 during simulation (from the actual value of 25 for synthesis and implementation). Use words such as “inefficient”, “wasteful”, “readability of waveform”, etc.

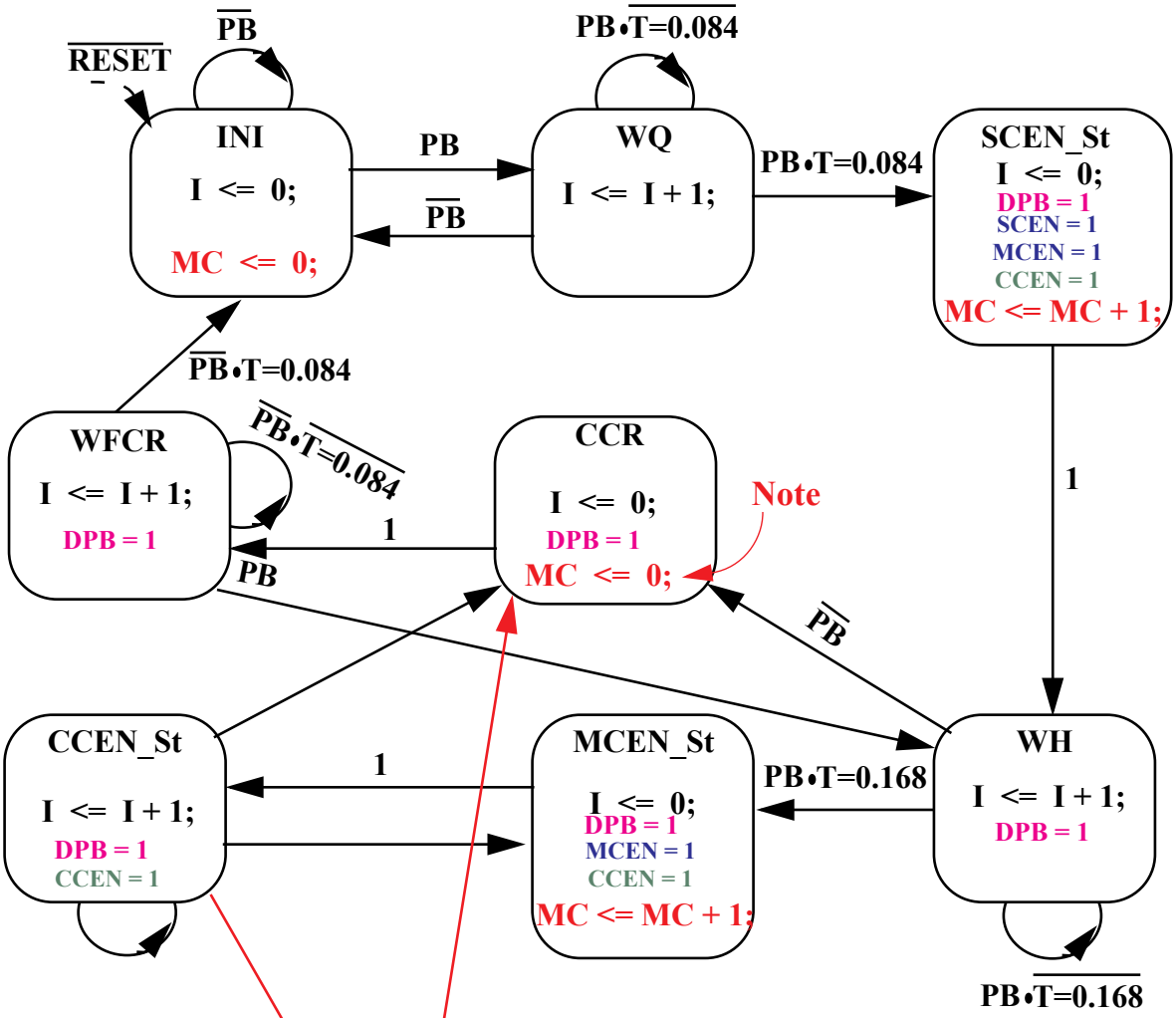
2. When you simulate, zoom into the area of above waveform extract and arrive at your answer for the above question in the waveform extract (why do we see 8 more pulses on MCEN after already seeing two pulses.

3. Did we use the DPB (Debounced Push-Button) pulse or SCEN (Single-Clock enable) pulse to act as the Start signal and the Acknowledge signal? Could we have used anyone of them?

Read the code (`ee254_debounce_DPB_SCEN_CCEN_MCEN.v`) and complete the state diagram below.

Complete the Debouncing State Machine with the added state MCEN_cont

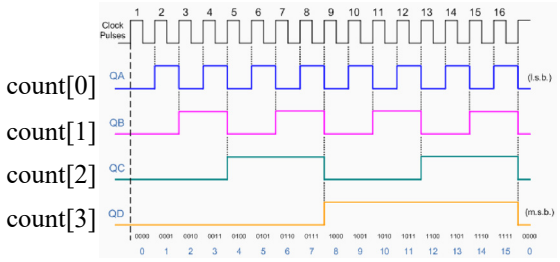
Complete the missing state transition conditions and also any RTL in the state MCEN_Cont



MC stands for MCEN count.
After certain count of MCN, control is transferred to MCEN_Cont.

MCEN_Cont

MCEN_Continuous state
Here MCEN behaves like CCEN.
See the output coding table given before.



count[3] becomes 1 after 8 (=2^3) clocks of the CLOCK.
count[23] becomes 1 after 2^{23} clocks of the CLOCK.
2^{23} clocks each of 10 ns make 0.084 sec. Hence T1 = 0.084 sec
2^{24} clocks each of 10 ns make 0.168 sec. Hence T2 = 0.168 sec

Nexys 3 board clock is at 100MHz.
100MHz frequency corresponds to 10ns clock period.

Names of the students submitting:

1.

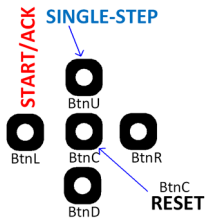
2.

3.

Questions on ee254_divider_with_single_step:

A. Is it possible to use SCEN to control one state (or a few states), MCEN to control another state, and further CCEN to control yet another state? When we say "control a state" here, we mean control the RTL operations in the state and also the state-transitions going away from the state. How about looping-around state transitions?
If we are not going away from the state (because of absence of the SCEN pulse) then we will remain in the state, whether originally there is a loop-around state-transition or not.

B. Can we choose to place **all three states** of the divider design under single-stepping control and *simultaneously* combine `Start` and `Ack` under one button (say BtnL)?
Is this just not possible or it works if we produce a BtnL_ SCEN and use it as START as well as ACK, or ...?
Can you press two buttons exactly at the same time to 10ns or 5ns accuracy? Even if you press at the same time to that accuracy, can you guarantee that they bounce for the same length of time and the two instances of the debouncing state machine would produce their respective SCEN pulses at the same time?



C. We took time to design output-coded state machine with no OFL at all, there by avoiding any glitches in the SCEN, MCEN, etc. Are glitches really harmful in our design or we have just shown a way to produce glitch-free outputs?