# Name: Section:

# Homework Assignment: submit via gradescope

1. Complete the development of the example in Lecture 18. Post a screen shot of the terminal window here or in your bitbucket repo, showing the display when you enter “?” with the count greater than zero. Also include a photo of the LEDs on your FPGA board matching the non-zero Count\_Q value.
2. Modify the lec10 component to include a roll signal that is asserted whenever the counter is at the maximum count value. In order to accomplish this with the generic counter width you may need to create a temporary signal that is equal to the highest count value:

 maxCount <= (others => '1');

Add the roll signal as an output to be displayed just like the count value on the help menu (Hint, remember this system tends to count by x26 for each count. If you load a Capital “E” and then count up you can get to max count and see Roll become ‘1’). The demo should show your count incrementing until the Roll signal goes high. Be prepared to demo your circuit at the beginning of Lesson 19 to your instructor or upload a video or upload a terminal screen shot showing the “?” output with the count value just before rollover (while roll is still ‘0’), and then the screen shot showing the “?” with the count as roll = “1”. Also see Lec18\_Install\_short\_version.pdf for more hints.

1. Optional: To get ahead for next class, go ahead and do “step 0” in Lec19\_Install\_short\_version.pdf”
2. **Documentation Statement**: For all assignments in this course, you may work with any faculty members or students **currently** enrolled in ECE383 unless otherwise indicated. We expect all graded work, to include software programs, wired circuits, lab notebooks, and written reports, to be your own work. If they aren't, you've copied and will receive **no academic credit** even if the copying is documented. Further, copying without attribution is dishonorable and will be dealt with as a suspected honor code violation. As in all courses, cadets must document any assistance received in the execution of graded work. If you receive no assistance on an assignment, the use of the **Documentation: None** statement is mandatory. If no documentation statement exists, the assignment will be returned for correction and the work will be considered at least one day late.

## Developed Lecture 18 Block Diagram



## Note

You will most likely get an error that looks like the following:

'Invoking: MicroBlaze Print Size'

mb-size lab3.elf |tee "lab3.elf.size"

 text data bss dec hex filename

 8100 272 8244 16616 40e8 lab3.elf

'Finished building: lab3.elf.size'

Building target: lec18.elf

Invoking: MicroBlaze gcc linker

mb-gcc -Wl,-T -Wl, bla bla bla -Wl,--start-group,-lxil,-lgcc,-lc,--end-group

c:/xilinx/14.7/ise\_ds/edk/gnu/ bla bla /ld.exe: lec18.elf section `.stack' will not fit in region `microblaze\_0\_i\_bram\_ctrl\_microblaze\_0\_d\_bram\_ctrl'

c:/xilinx/14.7/ise\_ds/edk/gnu/ bla bla /ld.exe: region `microblaze\_0\_i\_bram\_ctrl\_microblaze\_0\_d\_bram\_ctrl' overflowed by 80 bytes

collect2: ld returned 1 exit status

make: \*\*\* [lec18.elf] Error 1

Look at the MicroBlaze Print Size address editor and you will find that the value under dec (size of your program (instructions + data) in decimal bytes) is the 32k you allocated in Vivado for the microblaze memory. You should be good but potentially you may have to increase the size of the instruction and data memory in Vivado and recompile (see figure below). As you will see in the Print Size output above, I ran into this problem when my program JUST exceeded the 16K I had allocated for it in Vivado.


After you make this change, you will need to increase the stack and heap space inside SDK. Since you already started with 32K you may only have to increase the stack and heap in SDK. To do this, open the linker script (lscript.ld) and then increase the size of your memory to reflect the size entered in Vivado. Finally you can increase the stack and heap sizes to cover that needed in the error message.
