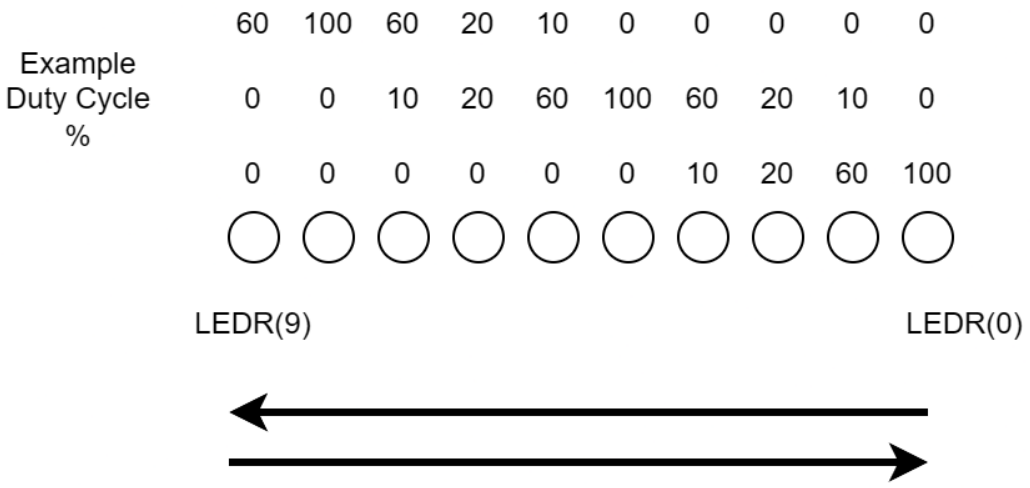
EE457
Lab 8: Pulse Width Modulation – Spring 2025

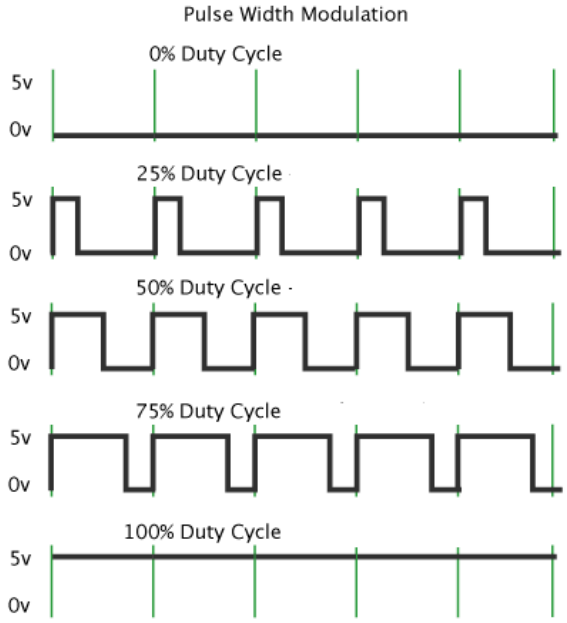
In FPGAs, pulse width modulation is a frequently implemented hardware function. Multiple strings of pulse width modulation can occur much more precisely on an FPGA than a traditional processor.

If you're not old enough to remember a TV show called "Knight Rider" starring David Hasselhoff who drove around in a car that was intelligent enough to converse with it named "Kitt". The car had a string of red LED's which lit up in various duty cycles making it look like a red light was bouncing back and fourth.

By changing the duty cycle of the LEDR lights on the DE10 lite board we can simulate "Kitt's" LED string.

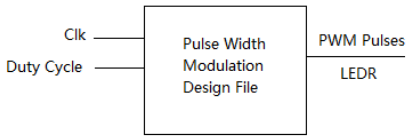


Duty cycle breaks up a specific time-period and acts as a switch turning the light on and off a certain percentage of time. For example, a 0% duty cycle would have the LED off all the time. A 50% duty cycle would have the light on half the time and off half the time. Finally, a 100% duty cycle would have the light on all the time.



Design Requirements:

1) You will simulate "Kitt's" LED strip using the LEDR's on the DE10.
    a. It is suggested that you make the PWM generator its own entity so that you may instantiate it multiple times controlled by the state machine. One possible block diagram for this sub-block looks like this. For those looking for a coding effcient way of dropping in 10 copies, check out the `for generate` VHDL construct. You may also just instantiate 10 copies manually if the for generate doesn't interest you.



    b. Key(0) is the Reset key and when this is pressed the 100% duty cycled LED will return to the right-most LED (LEDR(0)). And sequence starts over. (OK for LEDs to be blank while reset is asserted).

c. SW(0) will change the speed. You will design a slow speed and a fast speed for the LED "bouncing"
d. The Duty cycle will oscillate back and fourth so that LEDR(0) will be at 100% and then LEDR(9) will be at 100% as if to look like the LED's are bouncing back and forth.
    i. See the above image for how the PWM is distributed across the LED array at a few points.
e. Resets and Inputs must be appropriately synchronized
f. ModelSim simulation must run as-submitted
g. Quartus design must compile as-submitted.
h. No inferred latches
i. As in previous labs, your testbench should test the various input combinations to demonstrate your design works correctly.
j. Lab report should discuss how you solved this, including how you determined the PWM period (ie what rationale should be considered when making this choice?).
k. As before, de10_lite_base.vhd is your top, add necessary files, logic and structure below this entity, and implement your testbench in de10_lite_tb.vhd as before.
l. As before, you should add the necessary generic(s) to speed up your simulation (hint, you may need to speed up both the pwm **and** your state machine).

Follow the lab report guidelines in the syllabus/prior labs.

Zip up your design directory and your lab report and submit to Canvas. (as a .zip file).