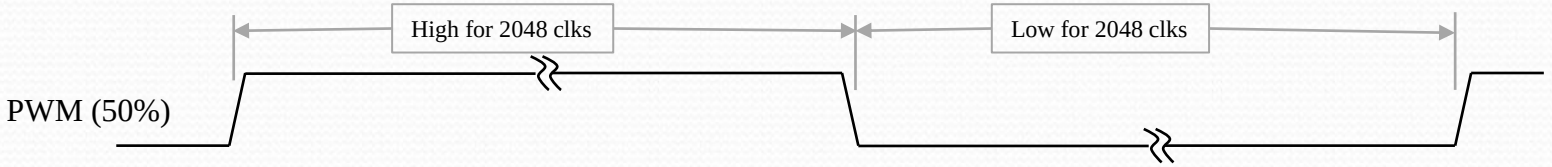
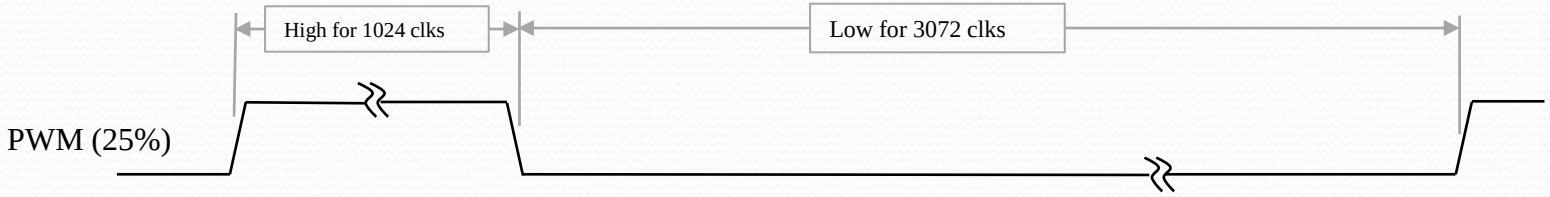
Exercise 10: PWM12

- We have to have a way of varying the strength/direction of the drive to the MazeRunners motors. This will be done through **P**ulse **W**idth **M**odulation (PWM).
- PWM is commonly used as a simple way of varying intensity. It can be used on an LED. Turn the LED on at full brightness for 100usec then off for 100usec. The human eye will average the light intensity (your retina integrates), so the light will look like the LED is driven at ½ intensity.
- The same works with motor control. Drive the motor coil at full voltage for 50usec and off for 150usec. The inductance in the coil will "average" the current and it will look like the motor is driven at 25%.
- Consider an 12-bit PWM signal being driven at 50% duty cycle. The period of the PWM waveform is 4096 clocks (2^{12}). Since our system clock is 50MHz this is 80usec.

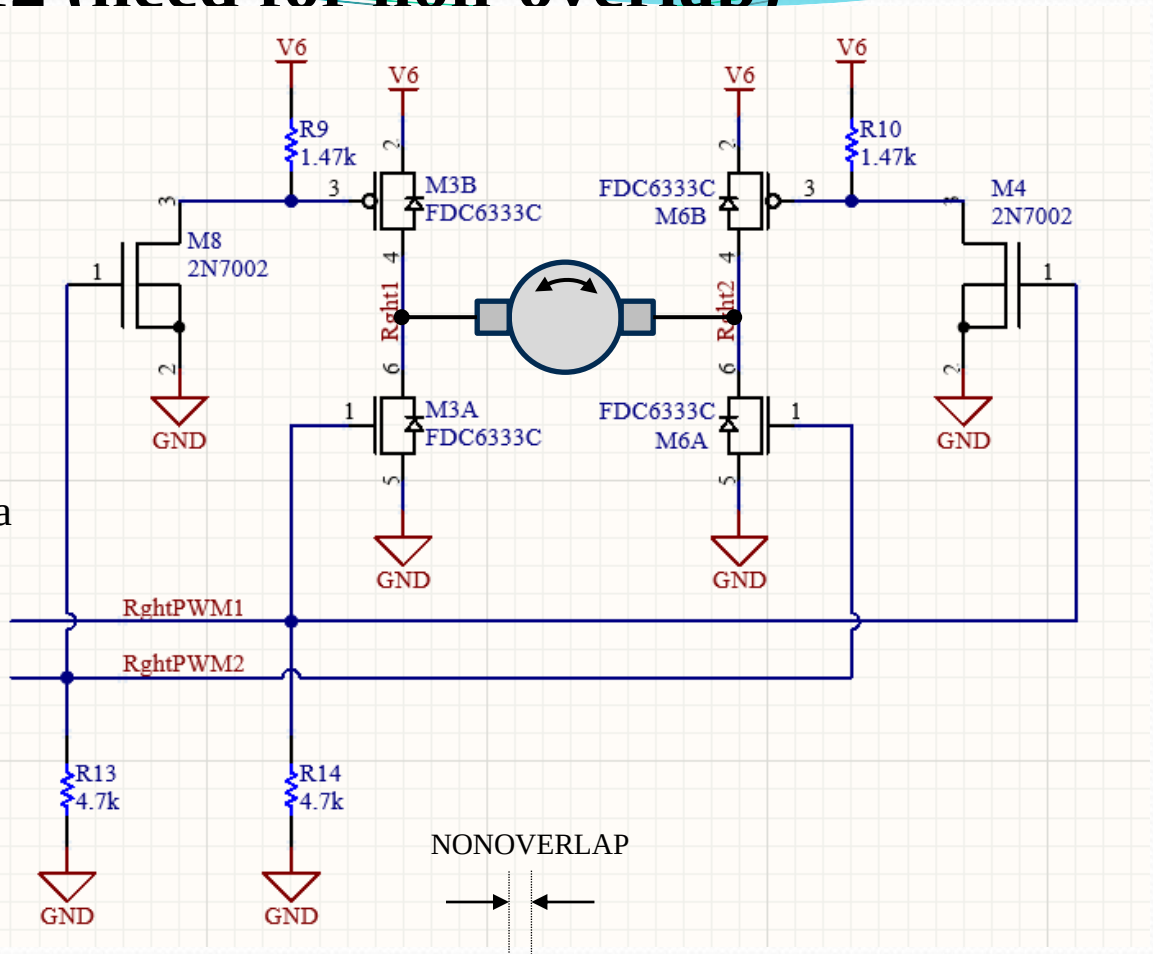


- Second example is 25% duty cycle



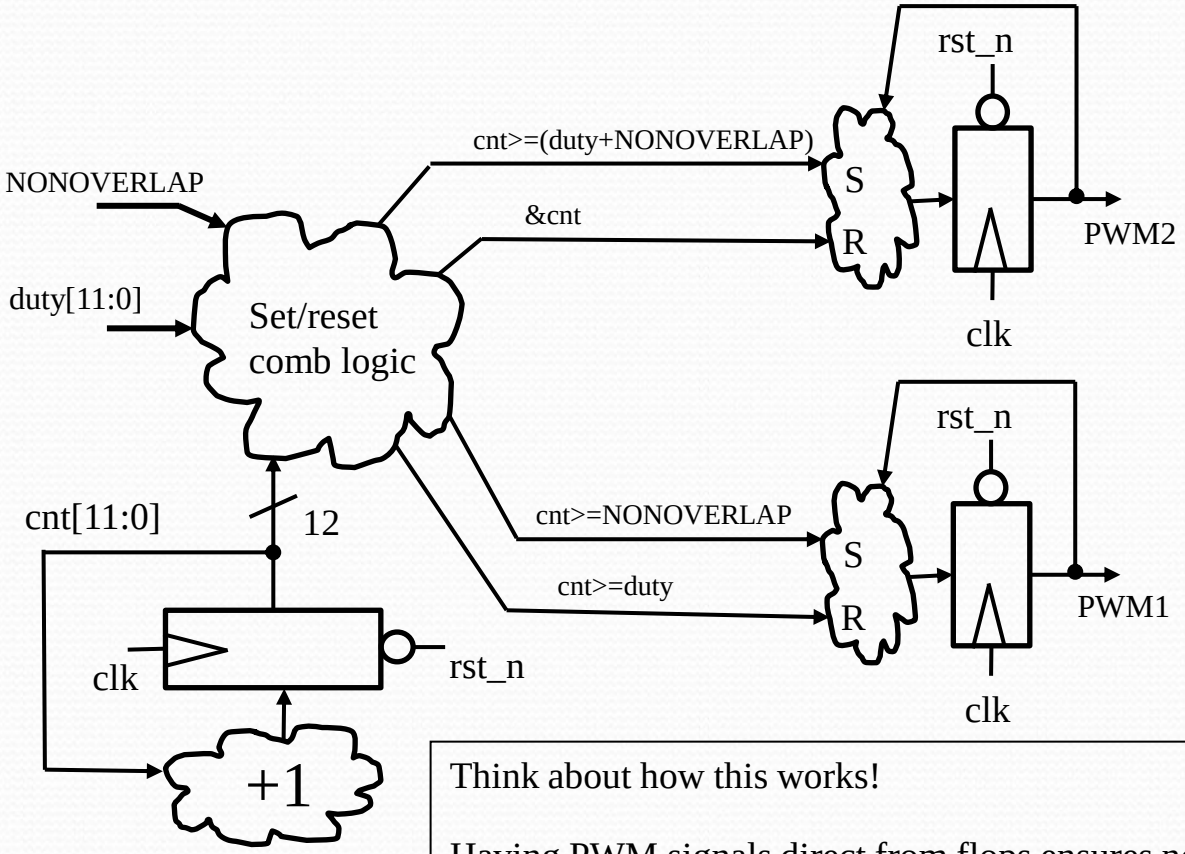
Exercise 10: PWM12 (need for non-overlap)

- The motors can be driven forward or reverse from the 6V battery through the circuit shown.
- PWM1 & PWM2 are complementary signals *(opposite of each other)* but they also have a non-overlap time *(time in which they are both low)*
- Why is the non-overlap necessary? Think about it… If both the NMOS and PMOS power transistors were on, then current would be shooting through the transistors from 6V supply to ground. We need to ensure both NMOS and PMOS in a stack are never on at same time.



Exercise 10 (PWM12 & PWM12_tb):

- We will have a localparam called NONOVERLAP and set it to 12'h02C. i.e. **localparam** NONOVERLAP = 12'h02C



Signal:	Dir:	Description:
clk	in	50MHz system clk
rst_n	in	Asynch active low
duty[11:0]	in	Specifies duty cycle (unsigned 12-bit)
PWM1 PWM2	out	Complementary glitch free PWM signals with non-overlap

Submit: **PWM12.sv** & **waveforms** showing various duty cycles

Duty cycles less than NONOVERLAP will result in zero duty cycle. *(this is OK for our application)*

Think about how this works!

Having PWM signals direct from flops ensures no glitching. You already made an S/R flop of this style in HW2.

Test at a low duty cycle, a mid duty cycle, and a high duty cycle. Self-checking not necessary.