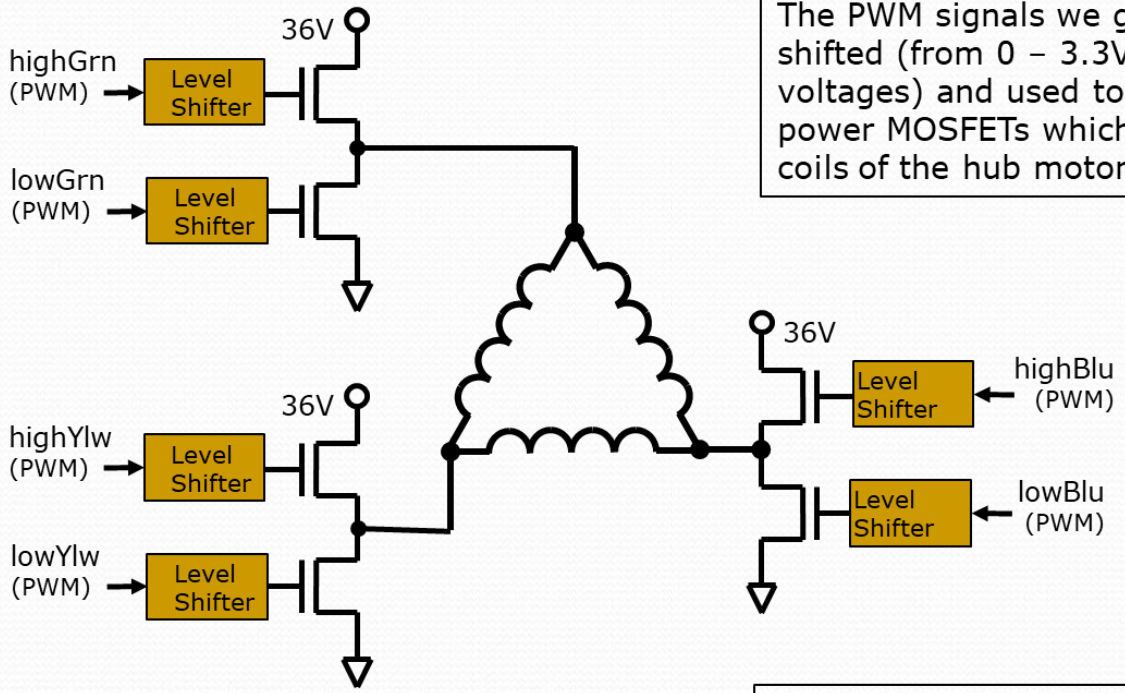
Exercise 8 (HW3 Problem3) (non-overlap):

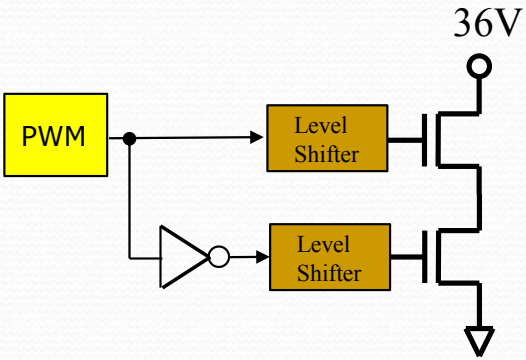


The PWM signals we generate are level shifted (from 0 – 3.3V signals to higher voltages) and used to drive the gates of power MOSFETs which in turn drive the coils of the hub motor.

The level shifters have some delay in their rise/fall times (in the 1 to 2usec vicinity). There is also some variation in the delay of the high driver from the low driver.

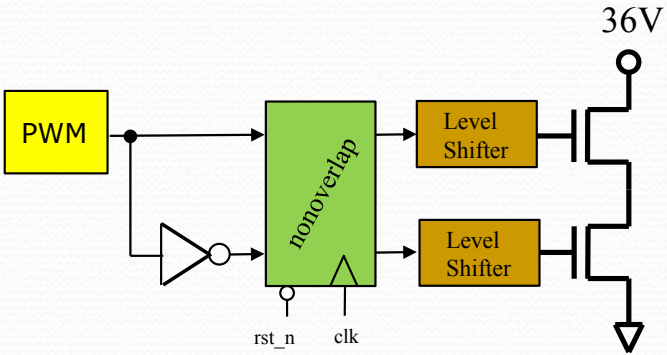
The power MOSFETs are very powerful (very low ohmic and capable of passing a lot of current). The low gate drive is logically just the inverse of the high gate drive.

Exercise 8 (HW3 Problem 3):



Given the power of the MOSFETs and the slow slope and variation in the level shifting gate drivers…do you see a problem with this configuration?

If both the high and low FETs are on at the same time (even for a fraction of a usec) then hundreds of amps could flow from 36V to GND.



Signal:	Dir:	Description:
clk, rst_n	In	50MHz clock, and reset
highIn	In	Control for high side FET
lowIn	In	Control for low side FET
highOut	Out	Control for high side FET with ensured non-overlap
lowOut	Out	Control for low side FET with ensured non-overlap

We need a non-overlap block that ensures the high gate drive and low gate drive (after level shifting) will never overlap. This non-overlap block will create a dead time (32 clocks) where both output signals are low for a while whenever an input changes.

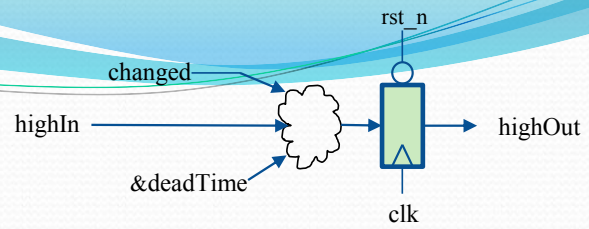
Exercise 8 (HW3 Problem 3):

nonoverlap.sv specifications:

- Whenever ***highIn*** or ***lowIn*** change both ***highOut*** and ***lowOut*** should go low on the next clock cycle (Next rising clk edge).
- Once ***highOut*** and ***lowOut*** are forced low (from a change in either) they should remain forced low for 32 system clocks.
- Both ***highOut*** and ***lowOut*** should come directly from flops so they cannot glitch (it is always possible for the output of combination logic to glitch).
- If ***highOut*** and ***lowOut*** are not being forced low (from a change) they should simply take their value from ***highIn*** and ***lowIn*** respectively.

See next slide for implementation hints

Exercise 8 (HW3 Problem 3):



nonoverlap.sv implementation hints:

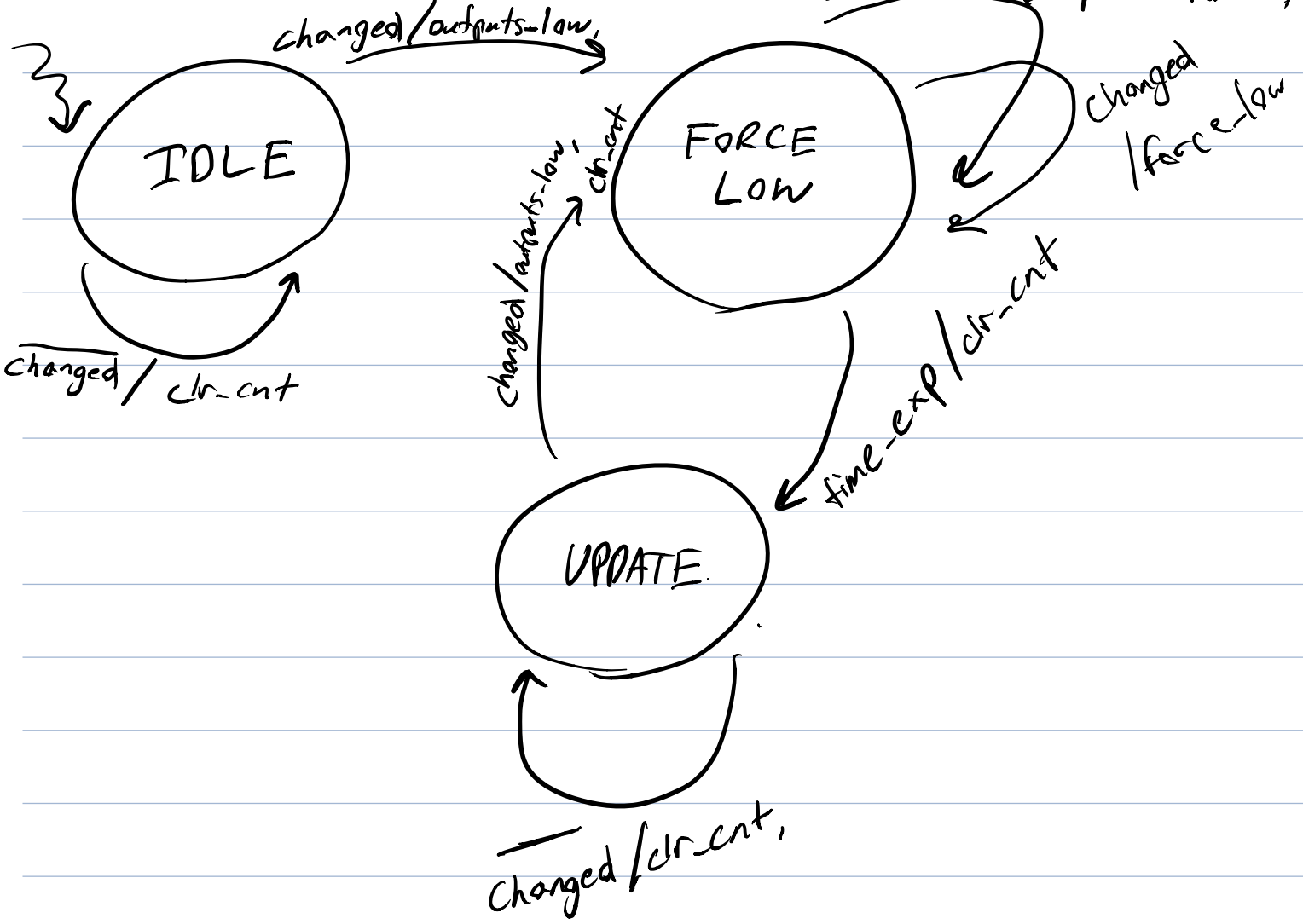
- You will need a 5-bit counter (dead time counter) that can be cleared via a signal (like when there is a change). It could possibly be free running, but you might want to have an enable signal.
- ***highOut*** and ***lowOut*** should come from flops that are asynch reset to zero, synchronously set to zero (under the condition of changed inputs) or a copy of ***highIn*** & ***lowIn*** if dead time has expired.
- You may want a simple state machine to control it, although it can be implemented without.

nonoverlap.sv testing:

- The testbench can have a single stimulus signal and its inverse feeding highIn/lowIn. Upon a change in this signal both outputs should go low for 32-clocks, then one of them should go high. (Testbench should self check)

Turn in **nonoverlap.sv** along with a testbench (**nonoverlap_tb.sv**) and proof the test bench was run/debugged.

time_exp / outputs_low


changed/outputs_low,
IDLE
FORCE LOW
changed /force_low
changed/outputs_low, clr_cnt
time_exp/clr_cnt
changed/clr_cnt
UPDATE
changed/clr_cnt,