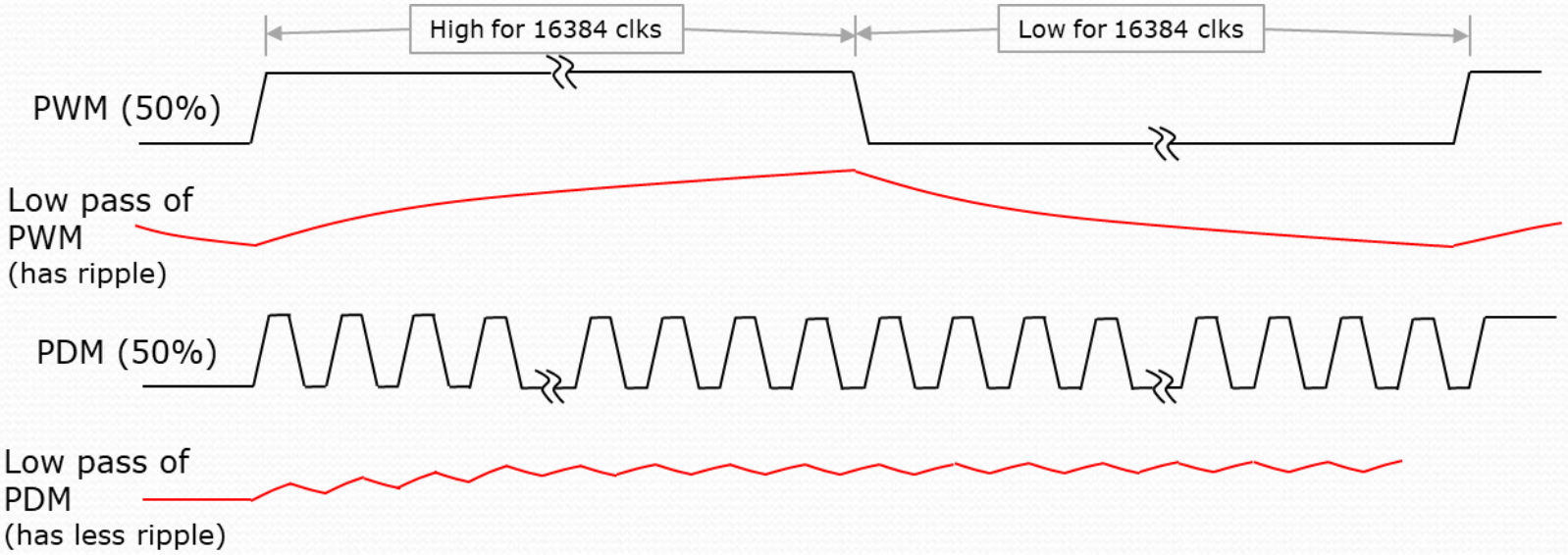
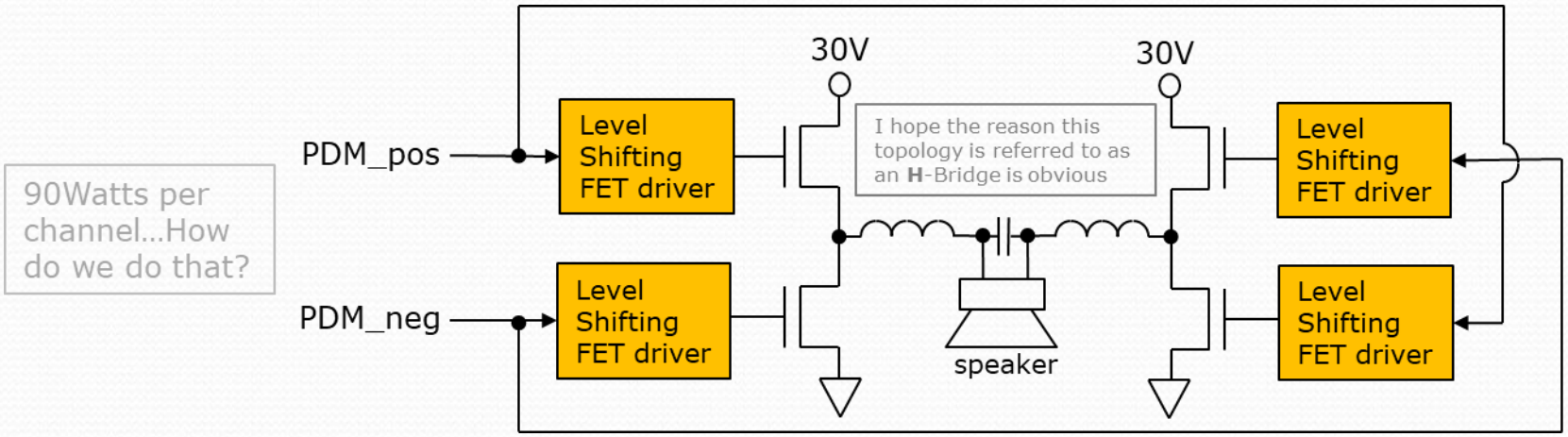
Exercise 7 (HW3 Problem2) (PDM):

- To drive the speakers we eventually need an analog signal. So we need a form of **D**igital to **A**nalog **C**onversion (**DAC**).
- There are many forms of DAC. One of the simplest is **P**ulse **W**idth **M**odulation (**PWM**) followed by low pass filtering.
- A variation of PWM is PDM. PDM and PWM are conceptually similar, but PDM is easier to low pass filter than PWM.
- Consider a 15-bit PWM signal vs a 15-bit PDM signal. Both driving 50% duty cycle.



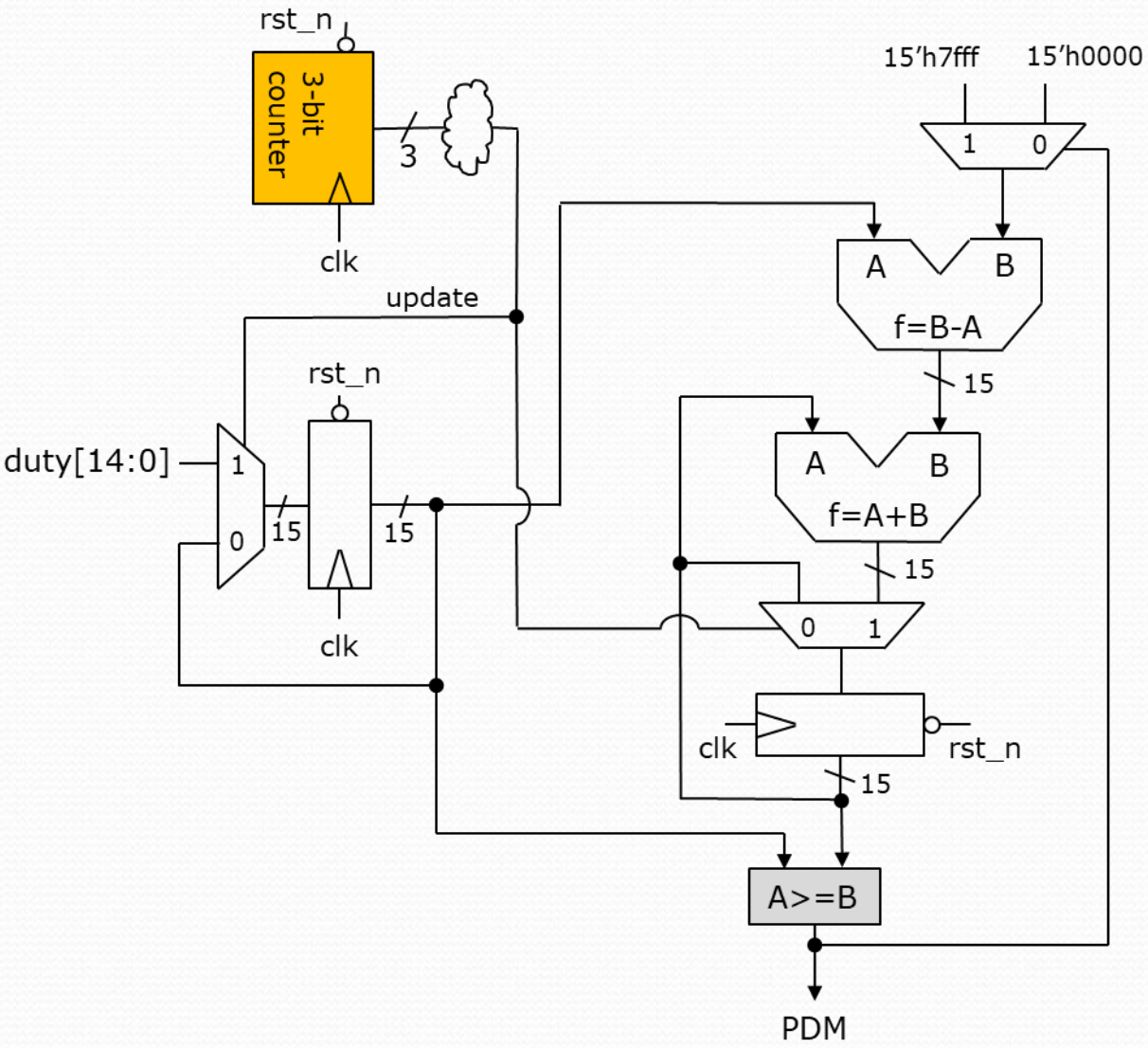
- As is shown above for a low pass filter with a given time constant, the PDM signal filters better and gives a lower ripple. This is an exaggerated view. The time constant of the LPF on the board will result in tiny ripple, and thus very little distortion in music quality.

Exercise 7 (HW3 Problem2):



- Being a child of the 80's I don't believe in headphones. If your neighbors aren't complaining you are not doing it right.
- The puny 0 to 3.3V PDM signals will be stepped up to 30V signals with 4A drive capability through the use of power MOSFETs in an H-Bridge topology. An LC circuit built across the H-Bridges forms the lowpass filter we need.
- The level shifting FET drivers are pretty neat chips, but they have limits. They cannot operate at the 50MHz of our main system clock. So our PDM signal cannot be operating at 50MHz. We will need to slow it down by a factor of 8 (6.25MHz).
- We will create a signal called **update** in our PDM block. This signal will be true when a 3-bit counter is full. Most all functions in our PDM block will only update when **update** is true. Thus we slow down the PDM to 6.25MHz without doing something stupid like running off a divided clock (see next page).

Exercise 7 (HW3 Problem2):



Signal:	Dir:	Description:
clk	in	50MHz system clk
rst_n	in	Asynch active low
duty[14:0]	in	Specifies duty cycle (unsigned 15-bit)
PDM	out	PDM signal out (glitch free)

Pretty much code what you see using mixture of dataflow and ***always*** blocks. Code it flat. No need to introduce hierarchy here.

Create a testbench. Test it at multiple levels of **duty[14;0]**. Theoretically you need to run for a minimum of 32768*8 clocks at each setting. **Turn in: PDM.sv, PDM_tb.sv** and proof you ran and it worked.