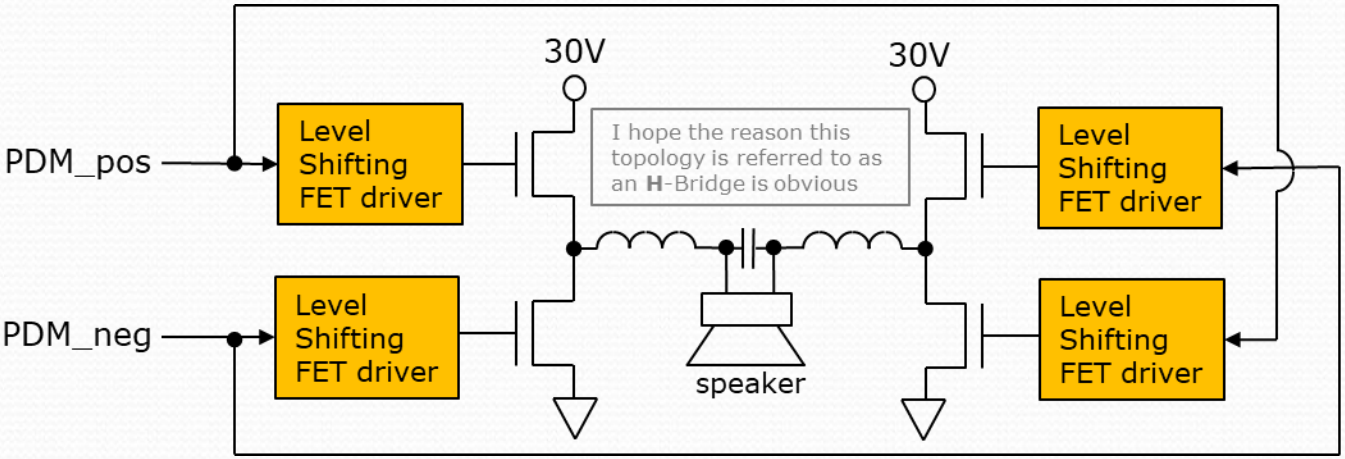
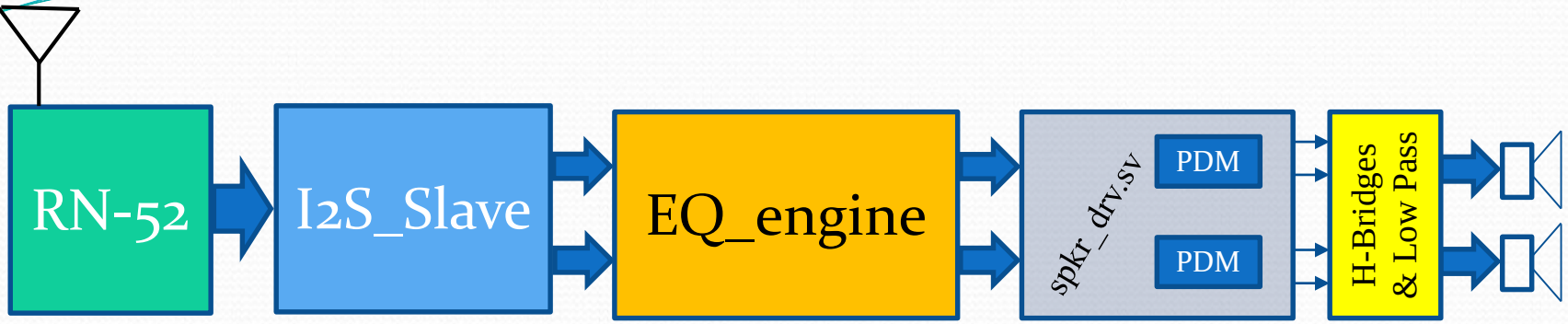
Smoked The Class D Amplifier



- I had my 90W class D amp made from discrete FETs and level shifting drivers working pretty good.
- Then I made some tweaks to my code and it went up in smoke!
- My amp had no real protections built in…had to scrap it.

Exercise 16: PDM revisited & spkr_drv.sv



- A simplified diagram of the flow of audio data is shown above.
- RN-51 provides audio data in I2S format. The I2S_Slave reads that and provides parallel 16-bit left/right audio data to the core equalizer function. The left/right output of the EQ_engine is still represented as two **signed** 16-bit numbers.
- We know we will be using the PDM peripheral you designed in HW3 to convert the music to analog to drive a speaker via H-Bridges.
- The **sprk_drv.sv** block is pretty much a simple registering of the inputs, and two instantiations of the PDM modules.
- However, we now have to revisit your PDM implementation and make some changes.

PDM Changes (verbal description):

- Input was originally intended to be a 15-bit unsigned number that represented and amplitude of speaker drive. The sign of the orginal 16-bit source would determine if the H-bridge of class D was driven positive or negative.

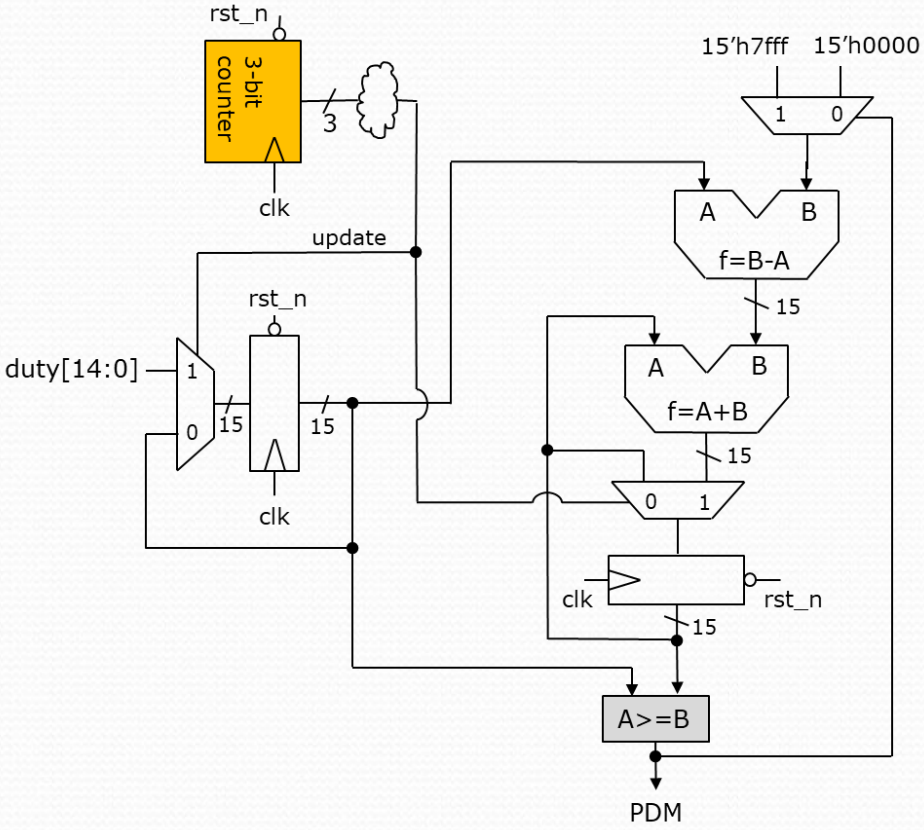
- Now input is a 16-bit signed number, however, PDM really operates in the unsigned domain so we have to convert this 16-bit signed number to a 16-bit unsigned number by adding 0x8000 to it.

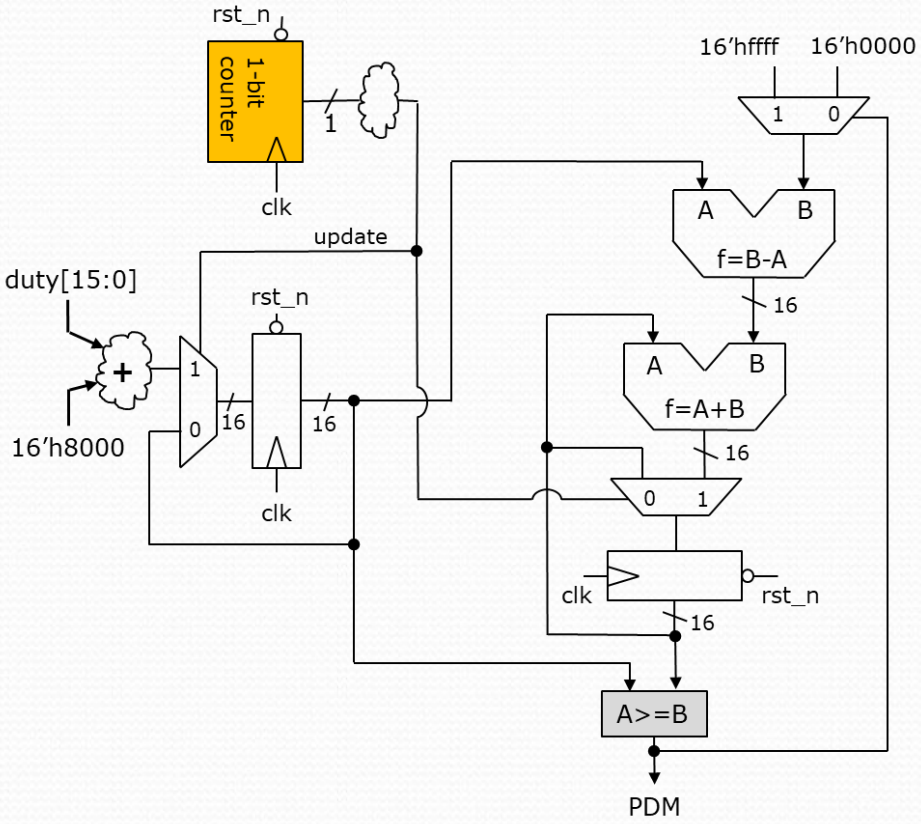
- The PDM is now 16-bits wide (all registers and arithmetic functions 16-bits) (they had been 15-bits).

PDM Changes (verbal description):

- Originally we had to seriously restrict the PDM update rate because we were driving high voltage level shifters with the PDM signal that could not operate too fast.
- Now we will be low pass filtering a PDM signal in the 0 to 3.3V domain, so we can crank up the frequency.
- The update rate had been 1:8 (3-bit counter). Now we will make it 1:2 (1-bit counter).
- A pictorial view is given on the next slide.

PDM Changes (pictorial):

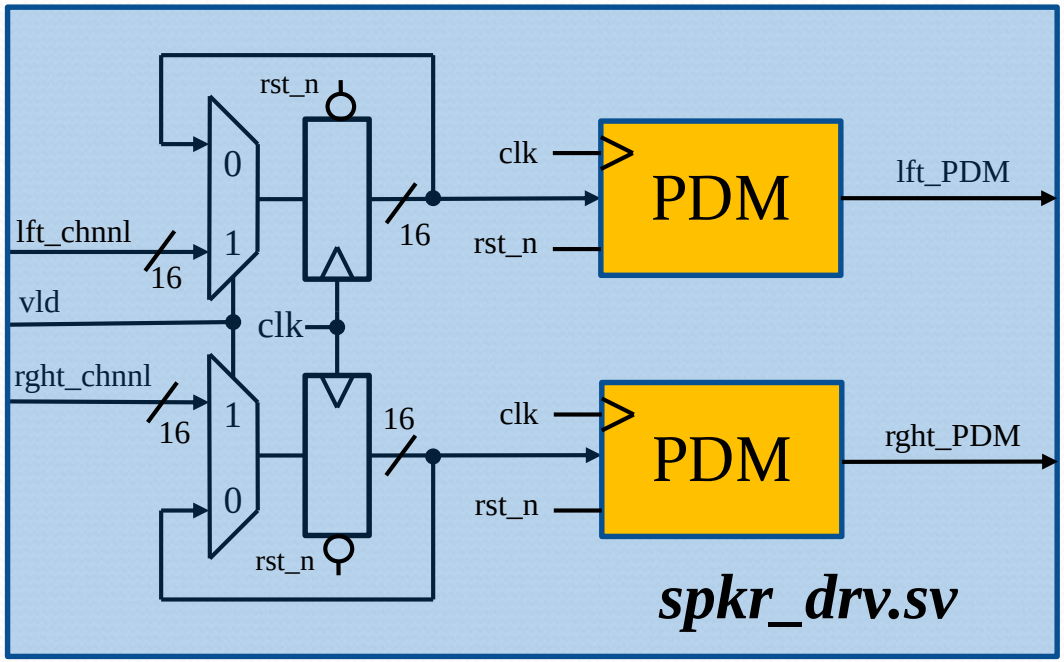




This → becomes This

Exercise 16: spkr_drv (verbal description of functionality)

- The left/right inputs (**lft_chnnl**[15:0] & **rght_chnnl**[15:0]) from the EQ_engine are not static vectors. They may only be valid for one clock cycle. They are accompanied by a valid signal (**vld**) that indicates when the values are valid.
- The first thing your **sprk_drv** block needs to do is register (buffer) these inputs.
- Then simply instantiate two (modified) PDM units and route the registered flops to them.



- Create a test bench (**spkr_drv_tb.sv**) to test it.

Turn In:

Modified **PDM.sv**
spkr_drv.sv
spkr_drv_tb.sv