NDSU ECE 444/644 : HW #2
Due: Wed., 9/23/2010

1. Using your HW #1 "digital wire" as a starting point, hardware-implement a spectral inverter system, which is (ideally) mathematically described as

$$y[n] = (-1)^n x[n].$$

Mathematically explain (prove) why this system is a spectral inverter. Using your hardware implementation, show that an input sinusoid of frequency f (where $0 < f < \frac{F_s}{2}$) produces an output sinusoid of frequency $\frac{F_s}{2} - f$. What kind of output is produced if the input is a square wave? What kind of output is produced if the input is a triangle wave?

CAUTION: You may not be able to implement an ideal spectral inverter given hardware limitations of our DSK. That is, you need to modify the spectral inverter idea to accommodate the dc offset/unipolar nature of the DAC. Carefully explain your modification when writing up your solution to this HW.

2. Consider a twist on the spectral inverter designed in Prob. 1: let's "kill" every other sample by making it zero, and then, much like the spectral inverter, alternate the sign of every other remaining sample. Taken together, this is equivalent to multiplying samples by the 4-periodic signal that sequences as $\ldots, 0, 1, 0, -1, \ldots$. Using your HW #1 "digital wire" as a starting point, hardware-implement this system. Explain the effect of this system on the input signal's spectrum. See how the system responds to the types of input signals (sinusoids, square waves, triangle waves) used in Prob. 1. See the same caution as given in the first problem.

3. Using your HW #1 "digital wire" as a starting point, hardware-implement an "adjustable-quality digital wire". Each ADC sample of your input is represented using a finite number of bits N ($N = 12$ for the K22F). Mask this result (logical AND with ones in the upper bits, zeros in the lower bits) to produce an N'-bit output, where $0 \leq N' \leq N$. You can adjust the bit resolution in code (recompile and download for each change) if you want, but I'd be more impressed if you found a method for run-time adjustment (e.g., select bit resolution using a variable that can be run-time adjusted using a variable watch window or – more impressively – change bit resolution by ± 1 using input buttons on the DSP board).

4. Book problem 2.1-5(c).

5. Book problem 2.2-1.

6. Book problem 2.3-3.

7. Book problem 2.4-3.

8. Book problem 2.5-1.

9. Book problem 2.6-2(b,c).

10. Book problem 2.6-4(a,c,e).

NOTE: Students need to demonstrate hardware-based problems (1, 2, and 3) to the instructor. While relevant design details and code should be turned in with the other problems, expect oral questioning during hardware check-off.