ADC Final Project – 3 Weeks

Prelab:

The next chip we will test is the ADC. The ADC receives an analog input voltage and produces a digital representation of it. This ADC produces a serial digital output. In this mini-project, you will design the tests for offset error, gain error, INL, and DNL for the 5V rail condition.

You do not need to turn anything in on the prelab, since it is really a project. However, I would make sure this is done before you walk into lab.

Look at the ADC datasheet and the DIB schematic. Be sure to consider the following questions.

a) What is the pinout? What do each of the pins do?
b) How do the pins map to the resources on the DIB? Keep in mind that you will only be testing site 1. What relays should be set?
c) How do the input buffers get their rail supplies? Be sure to look at the voltage regulator page of the DIB schematic.
d) What are the default operating conditions? (e.g. what should the rails be set to? Vref? clock frequency? etc.)
e) Given that this ADC has a serial output, what is the relationship between fclk and fsample?
f) What master clock frequency (between 25Mand 50M) would work to produce both fclk and fsample?
g) What is the VLSB? (in this case, the smallest input voltage that should ideally cause an LSB change in Dout?
h) What is the clocking scheme to get the data off the chip? What should the relative timing be for each of the control bits? For Dout?

ADC Final Project

In the next set of labs, you will be given a new part (and ADC, part number ADS7822), a new datasheet, and a new DIB (although, all the circuits are on the DAC DIB schematic). You are to determine how to test the ADC for Gain Error, Offset Error, INL, and DNL. You will need some additional information about how to capture data; however, all other method have been covered throughout the labs of the quarter. Below, I give you some direction as to how you should proceed. However, many of the details are intended to be determined by you. This is your chance to figure out on your own how to test a new part.

A. Create a New Project

Create a new project and set-up UserInit. This will include adding all analog pins, digital pins in the vector editor, and the clocking scheme. Keep in mind that you will only be driving 1 site.

Keep in mind that all of the timing information may need to be different for this lab since the timing is different for this chip.

B. Create a New Function: GERR

While we should probably start with continuity, none of the pins on this chip have standard ESD protection circuity. Therefore, the first function will test for gain error and offset error. I'm going to recommend several steps to work through this function and minimize debug. However, you don't have to follow these steps as long as you can get it working.

1. Get the chip to produce a digital output

Create the new function and set up the datasheet using the wizards. Set up the relays, analog resources, and the vector editor to control the serial output. On the tester, set an analog voltage on the input resource and verify that the chip is producing a reasonable digital output using the digital channels of the oscope in the tester room. You may need to add some of the filtering circuits on the input and/or the reference voltage to obtain reasonable outputs.

Note: if you're waiting for the tester, move onto step 2. You can test both step 1 and 2 on the tester at the same time. Just use the digital scope probes while testing for step 2.

2. Capture the data

To capture the data, you have to do 2 things:

1) You have to set up the equipment to get the data in the vector editor. When the vector editor is capturing the serial data, you must set a control bit to tell the digital pin unit to save the data in the digital capture memory.
2) Get the data that you captured from the digital capture memory. This should be done after running the clocks, which forces the vector editor to capture the data. Then the data must be retrieved before the next piece of data can be captured.

Test your program to see if this much is working properly. Increment Vin and measure Dout and see if the captured data matches the scope data.

3. Find Gain Error and Offset Error

In order to find the gain error and the offset error, you must find the value of Vin where Dout switches from 0 to 1 (called the Lower Edge) and where Dout switches to full scale (called the Upper Edge). From these edges, you can calculate the offset error and the gain error. However, when you find these, they must be stored in a global double array so that they can also be used for the linearity tests.

Find the Lower Edge (LE) in the same way that you find the offset voltage of a comparator. Sweep Vin until you see Dout change. Use the Vos spec to determine the sweep range. Store the value that forces Dout to change from 0 to 1. (Note: there may be a deadband region due to ADC offset voltage and filter offset voltage.)

Find the Upper Edge (UE) in the same way as LE, except sweeping Vin at higher voltages to force higher Dout values. Use the gain error spec to determine the sweep range. Store the value that forces Dout to change to full scale.

From these edges, calculate and datalog the offset error and gain error.

Test your program on the ETS. Check it with at least 2 different chips to validate your results.

C. Create the Linearity Function

Once you have Gain Error and Vos complete, you have all of the tools you need to create the linearity function. Some guidelines are shown below to perform the linearity function.

1) Recall that both edges have already been found. If they are stored in global arrays, they should be accessible from all functions. If declaring them in the previous function does not make them available, declare them BEFORE UserInit.
2) Create a linear ramp to perform histogram testing for INL and DNL. Determine how long the ramp must be to measure 10 points per code. Run the SPU as fast as it can go to get as many points in the ramp as you can. However, keep in mind that all ADC analog values do not need to be contained in the points since the integrator in the resource DAC will smooth out the digital points providing more resolution.
3) Create a new page in the vector editor and set it up to capture 1 piece of data. Use the programming controls in the vector editor to capture data continuously.
4) Run the analog and digital clock simultaneously.
5) Move the data to the DSP memory.
6) Perform a histogram on the data.
7) Using the histogram and the LE and the UE, find INL and DNL.
8) Run your program and see what happens! Set a break point in the function and check to make sure the following items make sense: (i) number code hits, (ii) Hit Averages, (iii) VLSB's, and (iv) code widths. If you get this working, serious extra credit!!! :)

D. Write up Your Results

Turn in your final code with data, figures captured, etc. and clearly explain how you know that the program is probably correct. Discuss challenges.