Keywords: integral nonlinearity, differential nonlinearity, INL, DNL, analog-to-digital converter, ADC, converter, data record, logic analyzer, data capture, concatenate

APPLICATION NOTE 3495

Combine Multiple Data Files to Optimize INL/DNL Processing

Apr 07, 2005

Abstract: The following application note shows how to utilize some of MATLAB's powerful commands to boost the code density of a memory-limited high-speed data capture system to achieve the high number of code repetitions required for INL and DNL testing of any high-speed analog-to-digital converter (ADC).

This brief application note demonstrates how data processing software such as MATLAB™¹ can be used to increase the number of codes in situations where logic-analyzer memory depths is limited. Any engineer, who has captured ultra-high-speed data with the logic-analyzer (LA) mainframe HP16500C equipped with a HP16517A high-speed state card (for data rates in excess of 135MHz), has probably already experienced these memory limitations.

The standard version of this HP system is usually equipped with a 64k maximum memory depth which allows only 65536 data points to be captured with a single test run. The data points that 64k memory captures are not at all adequate for generating accurate representations of INL/DNL graphs. In fact, for INL and DNL graphs, the logic analyzer should typically capture between 256k and 4M data points which are processed through MATLAB or another signal-processing software.

To use the 64k standard memory depth of the data-capture system described above and achieve precise INL/DNL characterization, multiple data records must be captured and combined into one file. If, for instance, 256k data points are required, then four sets of 64k data points must be captured and combined before processing with the MATLAB program (or similar source code) provided in the application note 2085, "Histogram Testing Determines DNL and INL Errors."

To combine multiple data records in one file large enough for INL/DNL processing, execute the following steps.

1. Capture 64k data points with the high-speed state card (HP16517A).
2. Remove all LA-specific lines within the code file which are not part of the code, and save the file.
3. Assign a new variable to the contents of the file.
   MATLAB command: a=load('inltest1');
4. Capture another 64k data record with the high-speed state card.
5. Again, remove the LA-specific lines within the code file which are not part of the code, and save the file.

6. Assign another new variable to the contents of the new data file.
   MATLAB command: b=load('inltest2');
7. Combine both data records (A and B) and assign them to a new variable.
   MATLAB command: c=vertcat(a,b);
8. If you want to test the size of the combined data record, just type **size(c);** in MATLAB and you should receive a 132k data record.
9. Save this new record as a .TXT or .DAT file, so that MATLAB can process the data for INL/DNL calculation.
   MATLAB command: save inldata.txt c -ascii;
10. The above command saves the file as an ASCII file in the predefined directory and makes it available to the INL/DNL test routine.
11. The above process will work, if all steps were completed in one MATLAB session. This can also be repeated with files that have been previously combined. For that purposeIn that case, just load the file, in which the data is stored, and assign it a new variable.
    MATLAB command: d=load('inldata.txt');
12. Once this task is completed, a new set of data can be taken and added to the existing data. Just capture another 64k data record and assign its contents to yet another new matrix variable.
    MATLAB command: e=load('inltest3');
13. Combine the old text file (now assigned to variable d) with the new data in variable e.
    MATLAB command: f=vertcat(d,e);
14. Follow step 9 to save the new, extended contents of the matrix to an ASCII file.

One can also combine multiple variables at once simply by typing the following command in MATLAB: **y=vertcat(a,b,c,d,e,f,g,h)**. There are many different methods, programs, and setup configurations to accomplish these tests with existing equipment and simple software commands. The process described here is simply one example of using the powerful capabilities of MATLAB.

To close the loop with the INL/DNL test program, simply call up the INL/DNL test routine in MATLAB. Run it, and load the concatenated data set into the program when asked. If you saved the combined data in a .TXT or .DAT file, do not forget to log the data path of the matrix file. When prompted by the program, you will have to provide the exact path and file name to be loaded into test routine. The program also asks for the number of data points (size of your concatenated matrix), the converter resolution, and its mid-code. Make sure that you have this information available before running the INL/DNL test program.

References

1. Application note 283, "INL/DNL Measurements for High-Speed Analog-to-Digital Converters (ADCs)."
2. Application note 2085, "Histogram Testing Determines DNL and INL Errors."

Related Parts		
MAX1121	1.8V, 8-Bit, 250Msps Analog-to-Digital Converter with LVDS Outputs for Wideband Applications	Free Samples
MAX1122	1.8V, 10-Bit, 170Msps Analog-to-Digital Converter with LVDS Outputs for Wideband Applications	Free Samples
MAX1123	1.8V, 10-Bit, 210Msps Analog-to-Digital Converter with LVDS Outputs for Wideband Applications	Free Samples

MAX1124	1.8V, 10-Bit, 250Msps Analog-to-Digital Converter with LVDS Outputs for Wideband Applications	Free Samples
MAX1213	1.8V, 12-Bit, 170Msps ADC for Broadband Applications	Free Samples
MAX19541	12-Bit, 125Msps ADC with CMOS Outputs for Wideband Applications	
MAX19542	12-Bit, 170Msps ADC with CMOS Outputs for Wideband Applications	

More Information
For Technical Support: http://www.maximintegrated.com/support
For Samples: http://www.maximintegrated.com/samples
Other Questions and Comments: http://www.maximintegrated.com/contact

Application Note 3495: http://www.maximintegrated.com/an3495
APPLICATION NOTE 3495, AN3495, AN 3495, APP3495, Appnote3495, Appnote 3495