

SmartFusion cSoC: Enhancing Analog Front-End Performance Using Oversampling and Fourth-Order Sigma-Delta Modulator

Table of Contents

Introduction	1
Design Example Overview	2
Design Description	5
Software Implementation	11
Running the Design	12
Conclusion	14
Appendix A – Design Files	15
List of Changes	15

Introduction

The SmartFusion[®] customizable system-on-chip (cSoC) devices integrate FPGA technology with a hardened ARM[®] Cortex[™]-M3 processor based microcontroller subsystem (MSS) and programmable high-performance analog blocks built on a low power flash semiconductor process. The MSS consists of hardened blocks, such as a 100 MHz ARM Cortex-M3 processor, advanced high-performance bus (AHB) matrix, system registers, Ethernet MAC, peripheral DMA (PDMA), real-time counter (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), fabric interface controller (FIC), the Philips Inter-Integrated Circuit (I²C), serial peripheral interface (SPI), external memory controller (EMC), embedded flashROM (eFROM), Watchdog Timer, 10/100 Ethernet Controller, GPIO block, in-application programming (IAP) and system registers. The programmable analog block contains the analog compute engine (ACE) and the analog front-end (AFE) consisting of ADCs, DACs, active bipolar prescalers (ABPS), comparators, current monitors, and temperature monitor circuitry.

The SmartFusion cSoCs have a versatile AFE that complements the ARM Cortex-M3 processor based MSS and general-purpose FPGA fabric. The SmartFusion AFE includes upto three 12-bit successive approximation register (SAR) ADCs, one first order sigma-delta DAC (SDD) per ADC, and high-performance signal conditioning blocks, and comparators. The SmartFusion cSoCs have a sophisticated controller for the AFE called the ACE. The ACE configures and sequences all the analog functions using the sample sequencing engine (SSE) and post-processes the results using the post processing engine (PPE), all independently of the Cortex-M3 processor, thus offloading the Cortex-M3 processor to perform other processing activities. Refer to the [SmartFusion Programmable Analog User's Guide](#) for more details.

The built-in analog capabilities are sufficient for many applications, but for those applications where increased data conversion accuracy is needed, digital signal processing techniques can be implemented in the SmartFusion cSoC FPGA fabric to increase ADC and DAC performance. This application note describes how to enhance the SmartFusion ADC performance using oversampling and FPGA fabric based post processing, and how to enhance the SmartFusion DAC performance by implementing a higher-order sigma-delta modulator in the FPGA fabric. This application note also describes the concept and simulation of a synthetic ADC to achieve high-resolution ADC performance using oversampling and fourth-order sigma-delta modulator for applications such as CD quality audio applications.

The design example associated with this application note includes hardware and software projects. The hardware project demonstrates ADC sampling, APB master implementation for reading ADC output samples into fabric, and passing the samples to the 1-bit DAC (OBD) through a sigma-delta modulator. The software project demonstrates a software implementation that provides a user interface through MSS UART for configuring a higher-order sigma-delta modulator in the FPGA fabric and ACE to control the AFE. A basic understanding of SmartFusion design flow is assumed.

Refer to the [Using UART with SmartFusion cSoC – Libero SoC and SoftConsole Flow Tutorial](#) to understand the SmartFusion design flow.

Design Example Overview

This design example demonstrates the SmartFusion cSoC's built-in ADC and DAC performance by feeding an analog input to the ADC and analyzing the quality of the reconstructed signal at the DAC output with and without signal enhancing techniques. The implementation without oversampling and higher-order sigma-delta modulator in the FPGA fabric is referred to as **simple wire**. The implementation with oversampling and a higher-order sigma-delta modulator in the FPGA fabric is referred to as **improved wire** throughout this document.

Simple Wire

In the simple wire implementation, the ADC samples the analog input signal at the Nyquist sampling rate—44.1 Kilo samples per second (Ksps) for audio signals—and the fabric logic (APB Master) passes these samples to the native first order sigma-delta DAC directly. [Figure 1](#) shows the block diagram of the simple wire implementation.

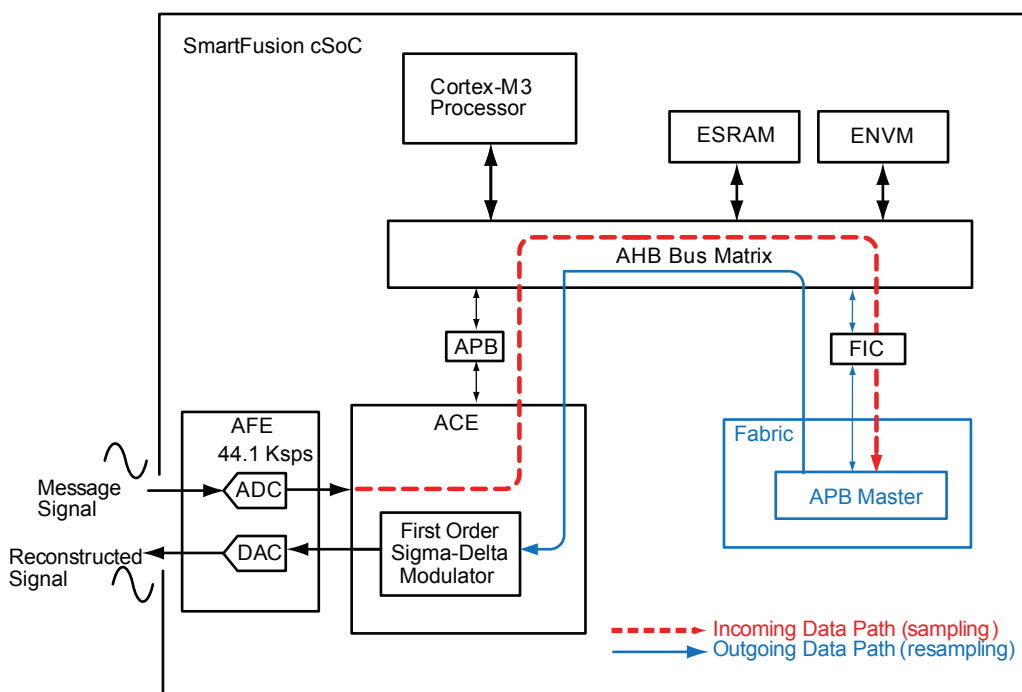


Figure 1 • Simple Wire Block Diagram

Improved Wire

In the improved wire implementation, the ADC oversamples the analog input signal at 400 Ksps and the fabric logic (APB master) passes these samples to the fourth order sigma-delta modulator implemented in the fabric. The fabric sigma-delta modulator converts these 12-bit sample values into a 1-bit pulse-width modulated signal and passes it as input to the 1-bit DAC. Figure 2 shows the block diagram of the improved wire implementation.

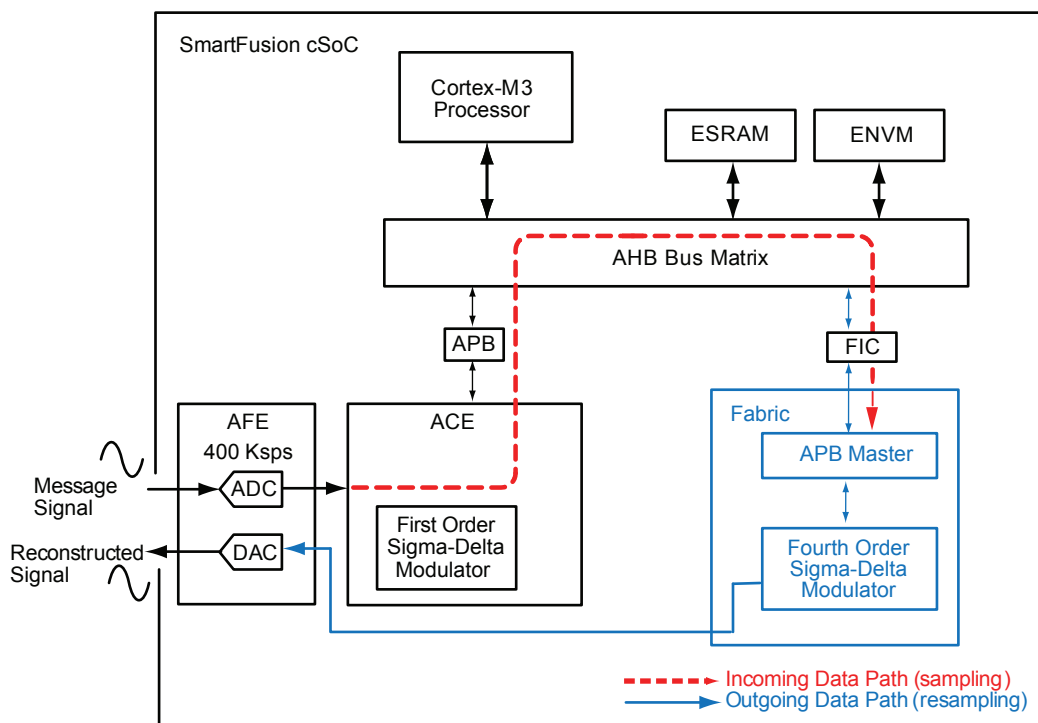


Figure 2 • Improved Wire Block Diagram

Synthetic ADC

The maximum accuracy of built-in ADCs is 12 bits and the effective number of bits is approximately 10 bits. For applications that require higher precision, such as 16-bit accuracy for CD quality audio applications, these native ADCs fall short. The concept of a synthetic ADC is to build a high resolution ADC using the built-in ADC and sigma-delta DAC.

Most of the high resolution ADCs are made with a low resolution ADC (for example, a comparator) in the feed-forward path and a high resolution DAC in the feedback loop. In a closed loop system, when the open-loop gain is large, the transfer function depends primarily upon the feedback components. Similarly, the composite ADC performance depends mostly upon the DAC in the feedback; it does not depend much on the ADC in the feed-forward path, which can be a comparator. The SmartFusion DAC has better performance in terms of resolution and differential non-linearity.

A high performance ADC, referred to as a synthetic ADC, can be constructed using a built-in ADC and DAC by implementing required post-processing logic in the fabric. The overall synthetic ADC architecture can be any of the classic forms: tracking ADC, SAR ADC, or sigma-delta ADC.

Figure 3 shows the block diagram of the synthetic tracking type ADC. The synthetic tracking type ADC uses the SmartFusion cSoCs built-in comparator, up-down counter, higher order sigma-delta modulator implemented in the fabric, and a 1-bit DAC. In this implementation, native ADC is not used. In this architecture, the output of the DAC is fed back to the built-in comparator through the built-in low-pass filter.

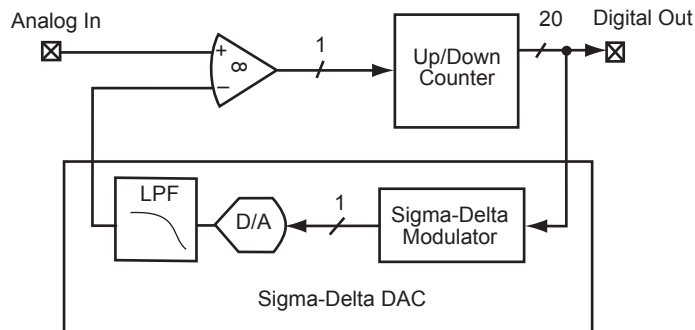


Figure 3 • Synthetic Tracking-Type ADC Block Diagram

Figure 4 shows the synthetic multi-bit sigma-delta type ADC implementation block diagram. A synthetic multi-bit sigma-delta type ADC uses a native ADC, higher-order sigma-delta modulator in the fabric, and a built-in 1-bit DAC. In this architecture, the output of the DAC's low-pass filter is fed back to ADC through an external attenuation circuit and a differential analog integrator.

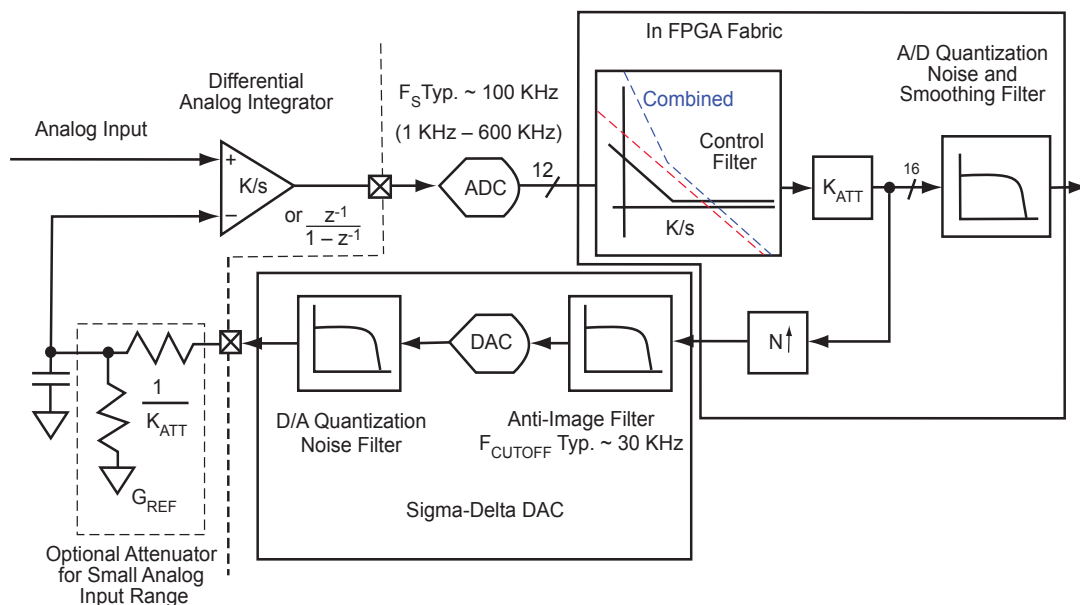


Figure 4 • Synthetic Multi-Bit Sigma-Delta ADC Block Diagram

Design Description

The SmartFusion ADCs are 12-bit SAR ADCs and support a cumulative sampling rate up to 600 Ksps. The SmartFusion cSoCs include an internal 2.56 V reference for ADCs; alternatively you can supply an external reference up to 3.3 V. For this design example, the ADC resolution is set to 12 bits and the internal 2.56 V reference is used, so the full-scale input range of the ADC is 2.56 V and minimum step size (LSB) is 0.625 mV (2.56 V/4096).

Based on the Nyquist-Shannon sampling theorem, the minimum sampling rate must be at least twice the frequency of the highest frequency component in the message signal in order to reconstruct the message signal. However, significant post-processing of the sample data is required to interpolate the values of signal during the time between each sample. Doing oversampling of the input signal can produce more samples to improve the digital representation of the signal. Note that the increase in sampling rate does not directly improve ADC resolution.

The SmartFusion DAC can be used for reconstructing the sampled analog signal. As long as the signal is sampled at or above the Nyquist frequency, post-processing techniques can be used to interpolate the intermediate values and reconstruct the original input signal. Best results can be obtained by implementing a reconstruction filter in the fabric, which is used to interpolate many intermediate values with higher resolution than the original signal. Interpolating many intermediate values increases the effective number of samples, and higher resolution increases the effective number of bits in the sample.

The sigma-delta DAC is comprised of three main blocks: a sigma-delta modulator, a one-bit DAC, and an analog low-pass filter. The SmartFusion cSoC device has built-in first order digital sigma-delta modulators implemented in the ACE and approximates the input binary words with a stream of 1-bit binary symbols using a sigma-delta style algorithm. In the sigma-delta modulator, the goal is to shape the quantization noise frequency spectrum in an ideal way. Due to the highly non-linear nature of the quantizer, first order modulators show some strong frequencies in the power spectrum due to periodic noise or residual tones. One way to reduce this noise is to use a higher order sigma-delta modulator. For higher order sigma-delta modulators, the noise shaping spectrum will be more predictable and pseudo-random in nature with less energy in the signal harmonics.

For applications that do not require extremely fine reproduction of the input signal, alternative methods can enhance digital sampling results with relatively simple post-processing. The details of such techniques are beyond the scope of this application note. Refer to the [Improving ADC Results through Oversampling and Post-Processing of Data](#) white paper for more information.

Simple Wire Implementation

In the simple wire implementation, the SmartFusion ADC0 is configured with 12-bit resolution and the sampling rate is set to approximately 44.1 KHz (Nyquist frequency) to sample signals up to audio frequency range. The SmartFusion MSS is configured as an APB3 slave in the FIC. The APB master in the fabric reads the ADC0 status register content located at 0x40021000. The DATAVALID bit (ADC0_STATUS[12]) of the ADC0 status register indicates the status of the ADC0 conversion. If the DATAVALID bit is set, ADC0 completes the conversion and the converted result is available in the lower 12 bits of the ADC0 status register. The APB master in the fabric continuously polls the status register at a higher frequency than sampling rate. The APB master reads the ADC0 converted result when the ADC0 DATAVALID bit is set and ADC0 is not busy in converting. The APB master forms a simple wire between ADC0 and DAC0 to reconstruct the sampled signal. In this implementation, DAC0 (SDD0) is configured for 16-bit resolution with a built-in first order sigma-delta modulator. The SmartFusion cSoC device's built-in sigma-delta modulator accepts 16-bit sample data and converts to an equivalent pulse-width modulated signal. The APB master reads the 12-bit ADC0 result and makes 16-bit sample data by shifting 4 bits to the left. The resultant 16-bit sample value is written to the SSE_DAC0_BYTE01 register at a fixed sampling rate. The DAC's 16-bit resolution and 600 Ksps sampling rate enables it to play CD quality audio signals.

Figure 5 shows the SmartDesign top-level diagram of a simple wire implementation. Figure 1 on page 2 shows the data flow in the simple wire implementation.

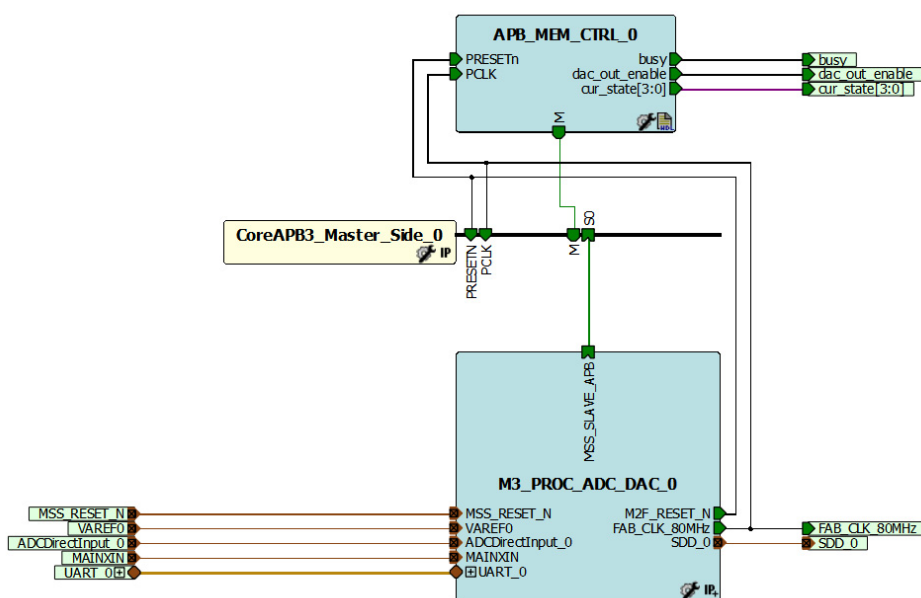


Figure 5 • Simple Wire Implementation

The sigma-delta modulator interpolates the input samples by holding the most recent sample until it is replaced by a new sample. The built-in sigma-delta modulators are clocked by the MSS clock, which is typically in the range of 80 MHz to 100 MHz. Not only is the signal resampled, but the fact that each sample is held (replicated) until the next input sample arrives from the APB master forms a simple reconstruction filter. Holding the sample has the same effect as passing the samples through an N-tap equally-weighted (first order) moving average finite impulse response (FIR) filter, where N is the resampling ratio.

Improved Wire Implementation

For applications that require fine reproduction of the message signal, the samples produced by the ADC with Nyquist sampling frequency are not sufficient. By oversampling the input at the ADC, you can produce more samples to reconstruct the signal more accurately. Since the SmartFusion cSoC device supports higher sampling rates, you can oversample the input signal up to 600 Ksps. As discussed earlier, for a further increase in the signal-to-noise ratio at the output of the 1-bit DAC, you can implement a higher order sigma-delta modulator in the fabric.

In the improved wire implementation, the input signal is oversampled approximately at 400 KHz and a fourth order sigma-delta modulator is implemented in the fabric. The SmartFusion ACE block is configured to bypass the built-in first order sigma-delta modulator. The APB master implemented in the fabric reads the ADC0 status register for a valid ADC converted result and passes this result to sigma-delta modulator implemented in the fabric. The fabric sigma-delta modulator is running at 24 MHz and generates a 1-bit pulse-width modulated output. Input samples are usually provided at a much lower rate; from DC to a few hundred KHz, in most cases. The 1-bit pulse width modulated output is given to the built-in 1-bit DAC to reconstruct the input signal. This design example also provides a software interface to configure the sigma-delta modulator in the fabric.

The ARM Cortex-M3 processor runs the application code to provide a UART interface for the fabric sigma-delta modulator configuration. Using the UART-based communication interface, you can select the order of the sigma-delta modulator—first, second, or fourth order—and a data signaling scheme of NRTZ or RTZ.

Figure 6 shows the SmartDesign top-level diagram for the improved wire implementation. Figure 2 on page 3 shows the data flow in the improved wire implementation.

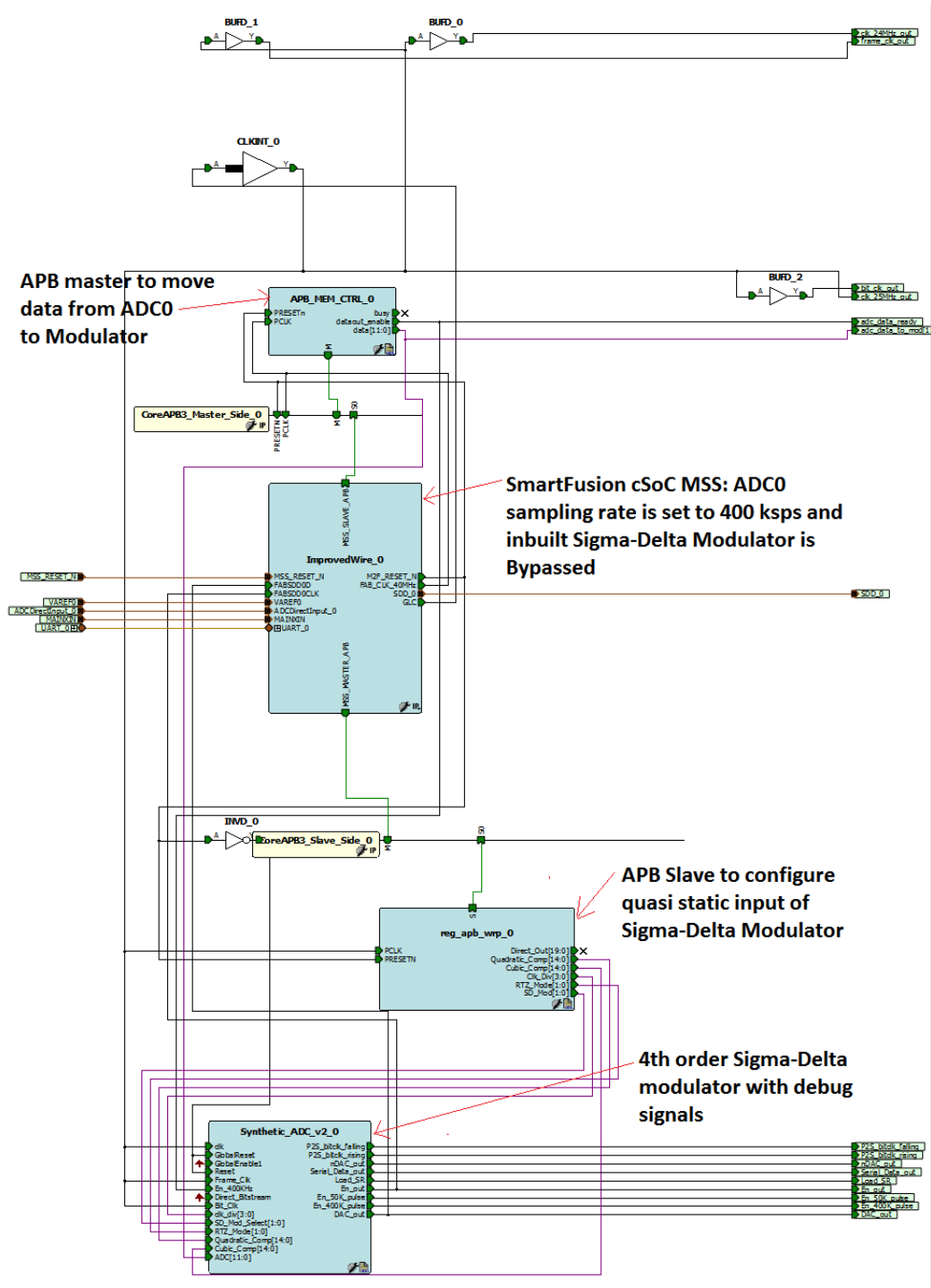


Figure 6 • Improved Wire Implementation

The sigma-delta modulator in the fabric is implemented using the Simulink® tool and converted to HDL using the Symphony HLS RTL generator. Figure 7 shows the Simulink implementation of the fourth order sigma-delta modulator used in this design.

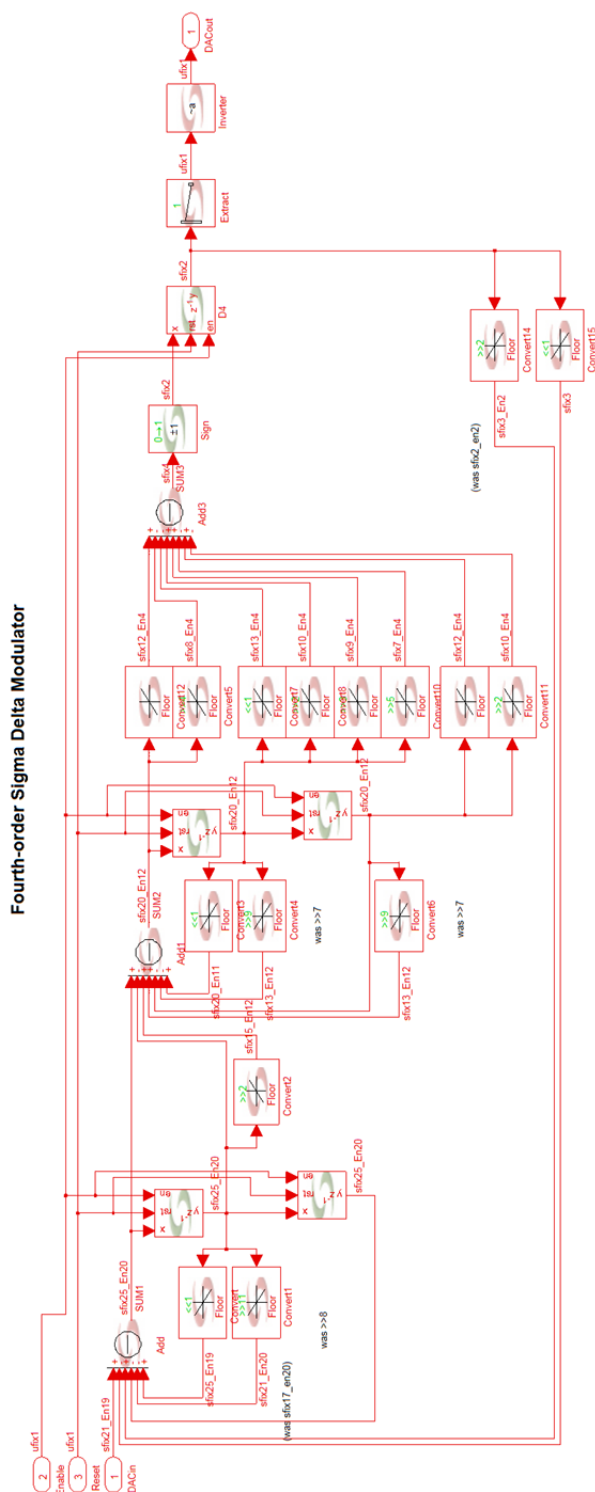


Figure 7 • Fourth-Order Sigma-Delta Modulator Simulink Implementation

For debugging and data analysis purposes, a decimate-by-8 filter and a I2S serializer blocks are implemented in the fabric using Simulink and the Symphony HLS RTL generator.

Synthetic ADC Simulation

This application note shows the simulation of complete synthetic ADC using Simulink. [Figure 8 on page 10](#) shows the synthetic multi-bit sigma-delta type ADC simulation testbench designed in Simulink. As shown in [Figure 8 on page 10](#), the synthetic ADC uses built-in ADC, post-processing blocks implemented in the fabric, high-resolution DAC, and analog feedback circuitry. The output of the sigma-delta DAC model is given as feedback to the ADC model through an analog feedback circuit. The decimate-by-8 filter, I2S serializer and deserializer are implemented to analyze the synthetic ADC output. As shown in [Figure 4 on page 4](#), you can implement the synthetic ADC by adding appropriate external feedback circuitry to the improved wire implementation between the DAC output and ADC input.

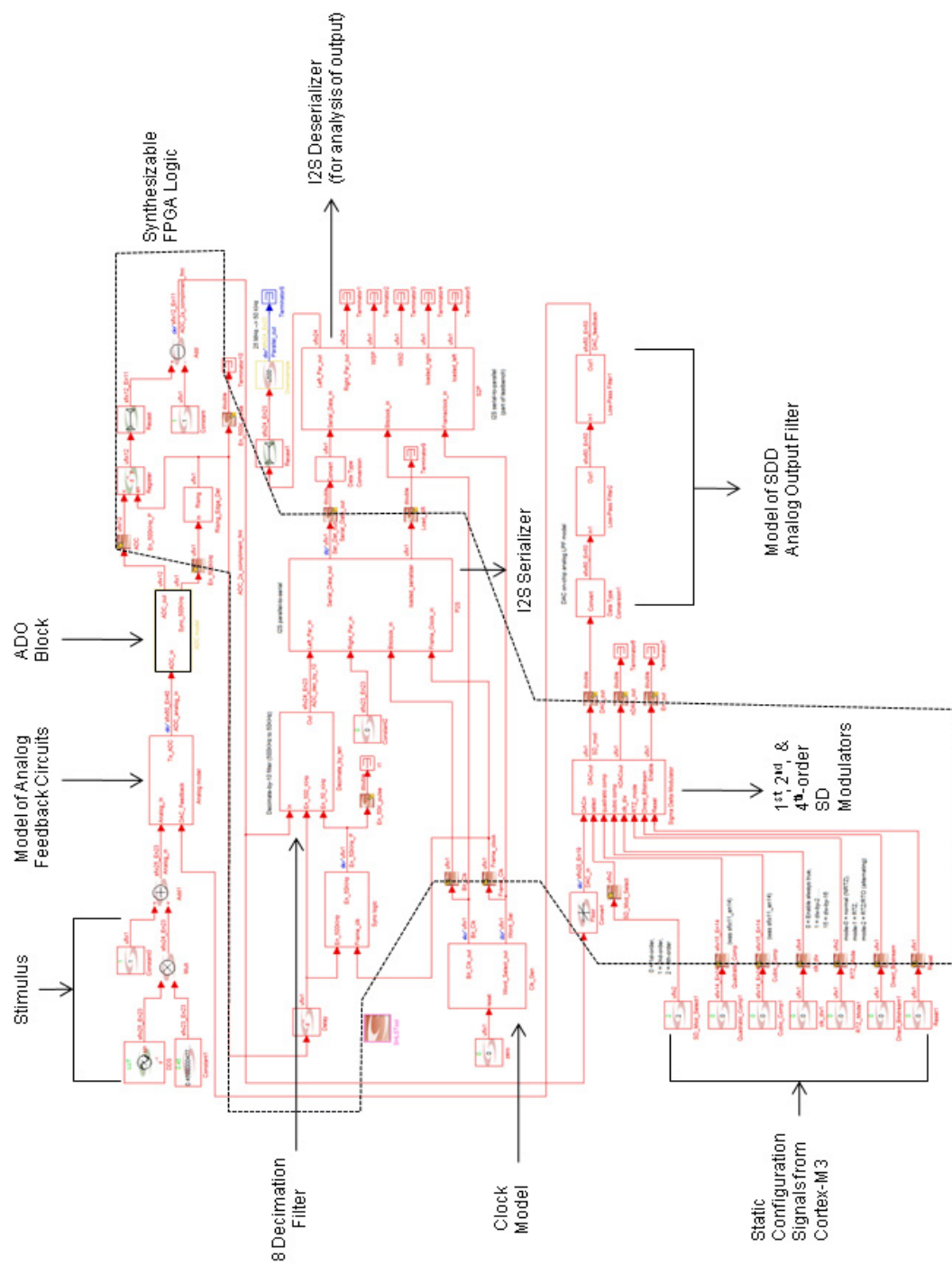


Figure 8 • Synthetic ADC Simulation Testbench

Figure 9 gives the simulated power spectrum of the synthetic ADC, showing the spectral purity.

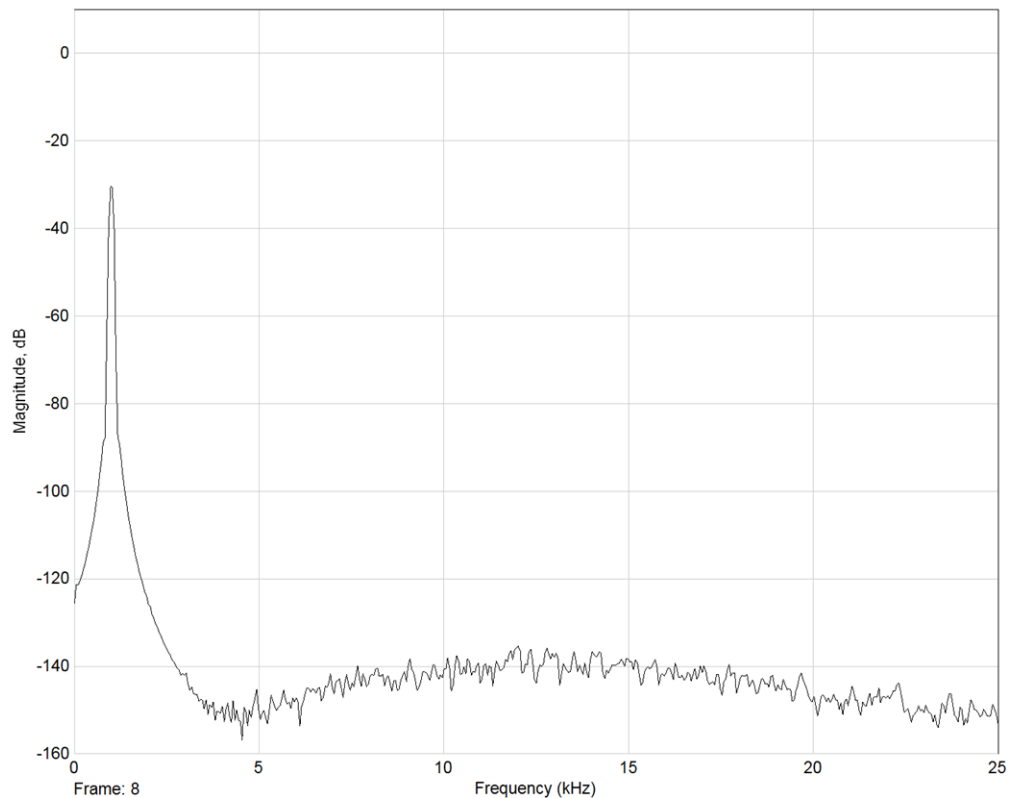


Figure 9 • Simulated Power Spectrum of Synthetic ADC

Software Implementation

The example software design configures the SmartFusion DAC0 with the required resolution and operating mode. In the improved wire implementation, software also provides a UART-based user interface for configuring fabric sigma-delta modulator static parameters such as order, quadratic and cubic compensation values, signaling mode (RTZ/NRTZ), and clock divider values.

The example software project provided in the design files is in the debug mode. Refer to the [SmartFusion cSoC: Building Executable Image in Release Mode and Loading into eNVM Tutorial](#) for generating the application in release mode.

The descriptions of the APIs used in the design are as follows:

1. ACE_configure_sdd() – Function to configure the SmartFusion cSoC sigma-delta DACs.
2. Modify_Q_C_Comp() – Function to modify the quadratic and cubic compensation values of the fabric sigma-delta modulator.
3. Modify_Quasi_Static_Inputs() – Function to set the fabric sigma-delta modulator quasi static parameters such as order, signaling mode (RTZ/NRTZ), and clock divider values.

Running the Design

Start a HyperTerminal session with a 57,600 baud rate, 8 data bits, 1 stop bit, no parity, and no flow control. If your computer does not have the HyperTerminal program, use any free serial terminal emulation program such as PuTTY or Tera Term. Refer to the [Configuring Serial Terminal Emulation Programs Tutorial](#) for configuring HyperTerminal, Tera Term, or PuTTY.

Program the SmartFusion A2F500 Development Kit Board (A2F500-DEV-KIT) with the provided simple wire implementation programming file (refer to "[Appendix A – Design Files](#)" on page 15) using FlashPro software.

Generate a positive sine wave with a maximum amplitude of 2.56 V and in audio frequency range using a function generator. Supply the generated sine wave as direct input to ADC0 at pin-4 of JP4 on the SmartFusion Development Kit Board. To observe the reconstructed signal, connect an oscilloscope or spectrum analyzer to the output of SDD0 at pin 3 of JP4 or pin 45 of the mixed signal header. Power cycle the board and observe the frequency spectrum of both input signal and reconstructed signal on the spectrum analyzer. [Figure 10](#) shows the simple wire test setup.

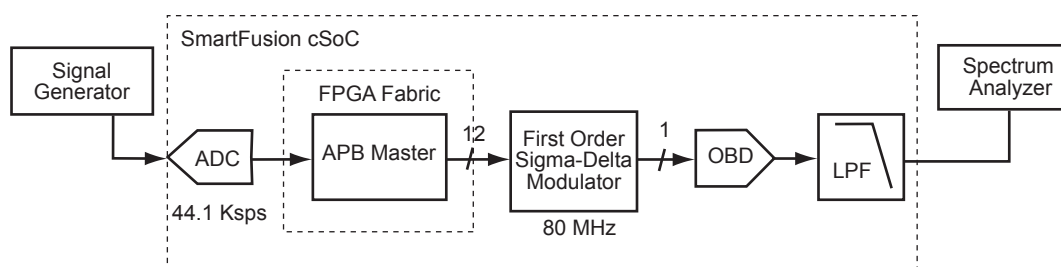


Figure 10 • Simple Wire Test Setup

Now program the SmartFusion Development Kit Board with the provided improved wire implementation programming file using FlashPro (refer to "[Appendix A – Design Files](#)" on page 15). Power cycle the board and observe the frequency spectrum of both the input signal and reconstructed signal on the spectrum analyzer. [Figure 11](#) shows the improved wire test setup.

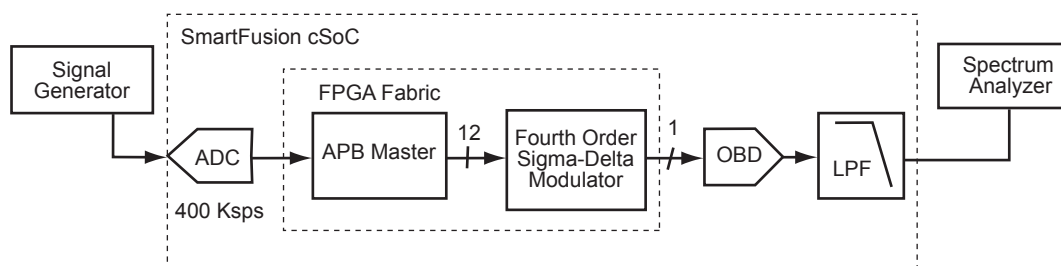
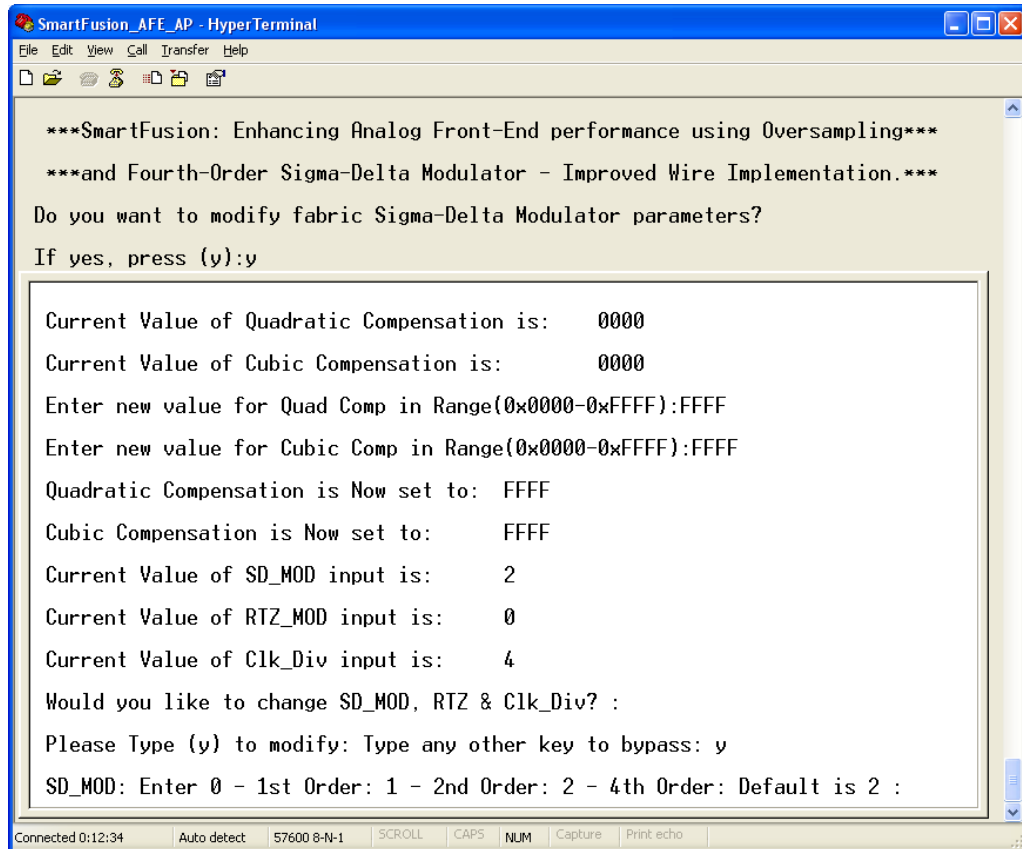


Figure 11 • Improved Wire Test Setup

Using a UART-based serial terminal interface, you can configure the fabric sigma-delta modulator quasi static parameters and observe the output signal quality. Figure 12 shows the screenshot of the HyperTerminal window with instructions to the user. Enabling return to zero (RTZ) mode improves the linearity of the SDD output to the detriment of accuracy. This mode can be used if linearity is more important than accuracy.



```

SmartFusion_AFE_AP - HyperTerminal
File Edit View Call Transfer Help

***SmartFusion: Enhancing Analog Front-End performance using Oversampling***
***and Fourth-Order Sigma-Delta Modulator - Improved Wire Implementation.***
Do you want to modify fabric Sigma-Delta Modulator parameters?
If yes, press (y):y

Current Value of Quadratic Compensation is: 0000
Current Value of Cubic Compensation is: 0000
Enter new value for Quad Comp in Range(0x0000-0xFFFF):FFFF
Enter new value for Cubic Comp in Range(0x0000-0xFFFF):FFFF
Quadratic Compensation is Now set to: FFFF
Cubic Compensation is Now set to: FFFF
Current Value of SD_MOD input is: 2
Current Value of RTZ_MOD input is: 0
Current Value of Clk_Div input is: 4
Would you like to change SD_MOD, RTZ & Clk_Div? :
Please Type (y) to modify: Type any other key to bypass: y
SD_MOD: Enter 0 - 1st Order: 1 - 2nd Order: 2 - 4th Order: Default is 2 :

```

Figure 12 • Screenshot of HyperTerminal Window for Improved Wire Implementation

In a comparison of the output spectrum of the simple wire and improved wire implementations, the improved wire implementation shows the high signal-to-noise ratio with proper noise shaping and weak harmonic frequency components. Figure 13 shows the power spectrums of DAC0 outputs for a 1 KHz analog signal with the simple wire and improved wire implementations.

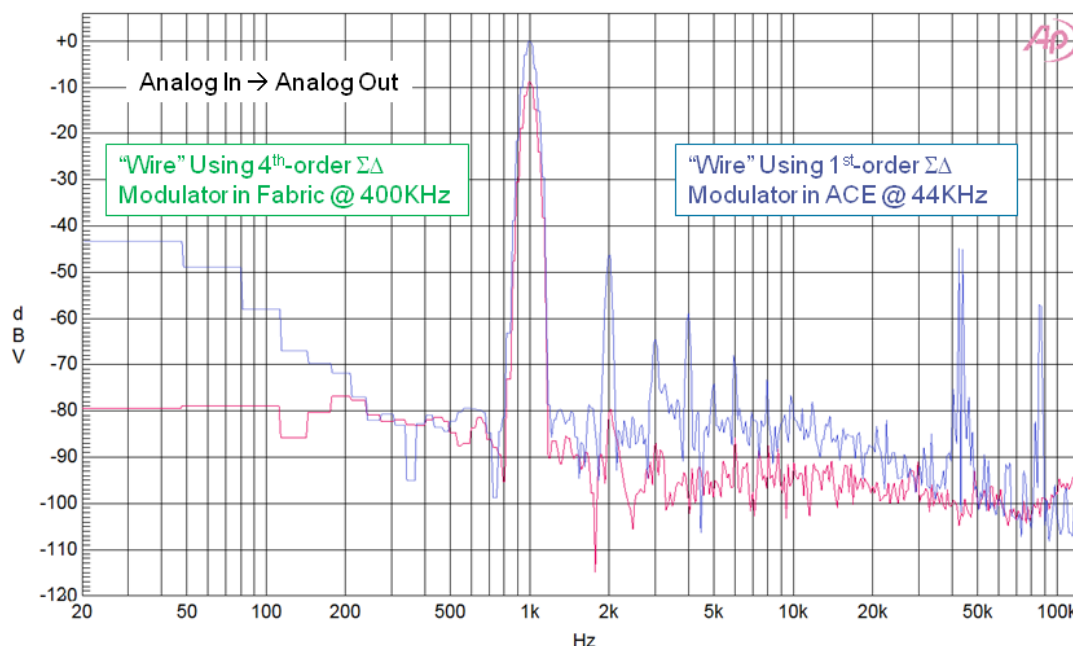


Figure 13 • Power Spectrums of Simple Wire and Improved Wire Implementations

Instead of a sine wave, you can use any audio signal (mono-type) at the input of the ADC0 with proper impedance matching circuitry and listen to the reconstructed CD quality audio signal at the DAC0 output with proper termination and a speaker with built-in amplifier. Observe the quality improvement in the improved wire implementation compared to the simple wire implementation.

Release Mode

The release mode programming file (STAPL) is also provided. Refer to the Readme.txt file included in the programming zip file for more information.

Building [Refer to the SmartFusion cSoC: Building Executable Image in Release Mode and Loading into eNVM Tutorial](#) or more information on building an application in release mode.

Conclusion

This application note shows the capabilities of the SmartFusion cSoC built-in ADCs and DACs and presents techniques such as oversampling and use of a higher order sigma-delta modulator to improve the performance of the built-in ADCs and DACs. It also presents the concept and simulation of synthetic ADC design using built-in resources, including the ADC, DAC, and fabric, for applications that require higher ADC resolution.

Appendix A – Design Files

You can download the design files from the Microsemi SoC Products Group website:
www.microsemi.com/soc/download/rsc/?f=A2F_AC375_DF.

The design files consists of a Libero® System-on-Chip (SoC) Verilog project, and a programming file (*.STP) for both simple wire and improved implementations. Refer to the Readme.txt file included in the design files for directory structure and description.

You can download the programming files (*.stp) in release mode from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC375_PF.

The programming file consists of STAPL programming files (*.stp) for simple wire and improved wire implementations and a Readme.txt file.

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 2 (February 2012)	Removed the '.zip' from URL in the application note (SAR 36763).	15
Revision 1 (January 2012)	Incorporated new Figure 5 and Figure 6 (SAR 35797).	6,7
	Added a new section called " Release Mode " (SAR 35797).	14
	Modified the section paragraph under " Appendix A – Design Files " section (SAR 35797).	15

Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.



Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at www.microsemi.com.

© 2012 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.