SmartFusion cSoC: Waveform Generation Using ACE DAC

Table of Contents

Introduction

The SmartFusion® customizable system-on-chip (cSoC) device contains a hard embedded microcontroller subsystem (MSS), programmable analog circuitry, and FPGA fabric consisting of logic tiles, static random access memory (SRAM), and phase-locked loops (PLLs). The MSS consists of a 100 MHz ARM® Cortex™-M3 processor, advanced high-performance bus (AHB) matrix, system registers, Ethernet MAC, DMA engine, real-time counter (RTC), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), fabric interface controller (FIC), the Philips Inter-Integrated Circuit (I^2C), serial peripheral interface (SPI), and external memory controller (EMC).

SmartFusion's programmable analog section contains analog computing engine (ACE) and analog front-end (AFE) blocks. The SmartFusion cSoC devices have a versatile AFE that provides a very useful complement to the ARM Cortex-M3 microcontroller and general-purpose FPGA fabric. The SmartFusion cSoC device have a sophisticated controller for the AFE called the ACE, which configures and sequences analog functions and post-processes the results, all independently of the Cortex-M3 processor, allowing it to work on other tasks. The SmartFusion cSoC devices have multiple analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). The A2F200 has two of each, and the A2F500 has three of each. The SmartFusion DACs are of the one-bit sigma-delta type. The ACE has a built-in first-order sigma-delta modulator that feeds each one-bit DAC. The DAC can be operated in either voltage or current output mode. Refer to the *SmartFusion Programmable Analog User's Guide* for more details.

Design Example Overview

This design example demonstrates generating analog waveform (constant signal, positive ramp, negative ramp, sine wave, and square wave) using SmartFusion ACE DAC on the SmartFusion Evaluation Kit Board and the SmartFusion Development Kit Board.

A basic understanding of the SmartFusion cSoC design flow is assumed. Refer to the *Using UART with a SmartFusion cSoC - Libero SoC and SoftConsole Flow Tutorial* to understand the SmartFusion design flow.

This example uses two methods to generate analog waveforms using SmartFusion 1-bit DAC as mentioned below:

1. Using the ARM Cortex-M3 processor as Master
2. Using logic in the fabric as Master

The SmartFusion DACs are 1-bit sigma-delta type. The ACE has a built-in first order sigma-delta modulator that feeds each 1-bit DAC. Refer to the sigma-delta DAC chapter in the *SmartFusion Programmable Analog User's Guide* for a detailed description of the SmartFusion DAC. The ACE block acts as an advanced peripheral bus (APB) slave to the ARM Cortex-M3 processor or APB3 master logic in the FPGA fabric. The ACE interfaces with the ARM Cortex-M3 processor through the AHB matrix without any additional logic or APB3 master in the FPGA fabric via the FIC and AHB to APB3 bridge. Figure 1 shows the functional diagram for waveform generation using the ARM Cortex-M3 processor and logic in the FPGA fabric as masters.

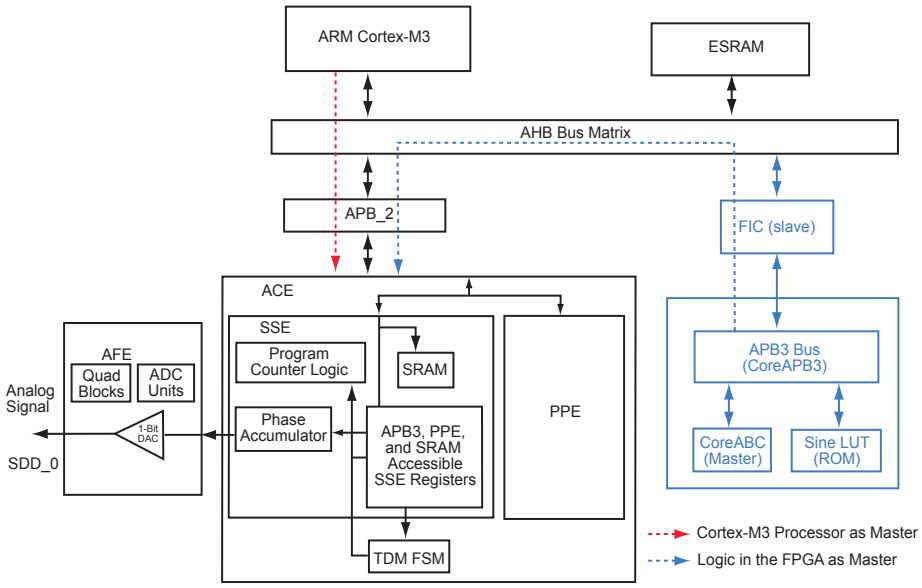

***Figure 1* • Top-Level Functional Diagram for Waveform Generation**

The design steps for waveform generation using the SmartFusion ACE DAC are as follows:

1. Use the ARM Cortex-M3 processor or logic in the FPGA fabric as master to write the computed/stored sample value to the DAC BYTE registers. In the design example, CoreABC IP is used as APB3 master in the fabric.
2. Send the computed/stored sample values to the DAC BYTE register. Writing to the DAC BYTE registers updates all 24 possible bits of the DAC data to the phase accumulator. This updates the 1-bit DAC output accordingly.
3. Probe the SmartFusion DAC output (from the mixed signal header of the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board) and observe the output in an oscilloscope.

Design Description

Digital Signal Pattern Preparation

In order to output any analog signal pattern, you must generate the proper data content in the memory. This stored content is then transferred to the DAC register using the ARM Cortex-M3 processor or logic in the fabric. EQ 1 is used to generate the sine wave samples.

$$Y_{SineSample}(x) = Offset + Amplitude * \sin(2* pi* x/n_s)$$

EQ 1

Here n_s is the number of samples per cycle.

The frequency of the generated analog signal is given in EQ 2:

$$F_{OutputSignal} = Sampling\ rate / n_s$$

EQ 2

Here Sampling rate is the number of samples per second.

For example, to generate a 1 KHz output frequency, the sampling rate and number of samples per cycle could be 100K samples per second and 100 samples per cycle respectively. To produce a high quality sine wave pattern, Microsemi recommends using a high number of samples per cycle (n_s). The following sections explain waveform generation using the ARM Cortex-M3 processor as master and logic in the FPGA fabric as master.

Using the ARM Cortex-M3 Processor as Master

The application program that runs on the ARM Cortex-M3 processor computes the waveform sample values. The ACE SSE registers are addressable and accessible from the ARM Cortex-M3 processor. The ARM Cortex-M3 processor writes the computed sample value to the DAC register and this updates the 1-bit DAC output accordingly. In this design example, 256 sample values are being computed for generating any one of the following five patterns:

1. Constant signal
2. Positive ramp
3. Negative ramp
4. Sinusoidal waveform
5. Square waveform

Using Logic in the Fabric as Master

The ACE SSE registers are addressable and accessible from a master in the fabric. Writing the DAC BYTE register updates the phase accumulator output and generates an equivalent analog voltage. In the design example, current design case DAC0 resolution is set to 8-bit and the DAC0_BYTE0 register is being used as DAC register. Increasing the number of sample values increases the smoothness of the generated analog waveform and reduces the output frequency.

In this design example, CoreABC is used as master in the fabric and a signal LUT/ROM is created with the APB3 interface. CoreABC reads the signal sample words that are stored in the signal LUT/ROM and feeds them to the SSE DAC0_BYTE0 register. The core can be configured using the configuration GUI within SmartDesign. The instructions to the CoreABC are given in the form of a program in the CoreABC GUI configuration window.

The CoreABC program reads a signal word from the signal LUT/ROM over the APB3 interface (CoreAPB3 IP) and writes that word to the ACE DAC register.

CoreABC Program

```
// Program to read the sine sample value from sine LUT through slot 2
// and writing to DAC0_BYTE0 register.

DEF DAC0_BYTE0_OFFSET 0x0064

// Delay to change the output frequency
// Vary this delay to change the output frequency

DEF DELAY 150

// Setting the Base Address of SmartFusion ACE for APB3 indirect addressing

APBWRT DAT 1 0x0000 0x40020000

// Initialize the CoreABC Z register with ZERO

LOADZ DAT 0
JUMP $MAIN

// Main loop

$MAIN
$LOOP

    // Read the sample value from sine LUT to CoreABC Accumulator over slot 2

    APBREADZ 2

    // Write the Accumulator data to DAC0_BYTE0 register over slot 0

    APBWRT ACC 0 DAC0_BYTE0_OFFSET

    // Delay Loop

    LOAD DAT DELAY
    $WAIT
    DEC
    JUMP IFNOT ZERO $WAIT

    // Increment the value in the Z register

    INCZ

JUMP $LOOP
// End of CoreABC Program
```

Refer to the *CoreABC Handbook* for more information on the set of instructions supported by CoreABC. The FIC is used to connect the user logic in the fabric to the MSS. In this design example, the FIC is configured in slave mode. The CoreAPB3 bus is used to connect CoreABC, signal LUT/ROM, and the MSS, as shown in Figure 2 on page 5.

The CoreAPB3 bus is configured in Indirect Addressing mode to write to the ACE SSE DAC register. When Indirect Addressing mode is used, slot 0 can access a 32-bit address space. In Indirect Addressing mode, slot 1 of the CoreAPB3 bus is used as a register to store the higher 16 bits of the 32-bit address.

When the peripheral in slot 0 is selected by the master, the lower bits of PADDR (specifically, bits log2(RANGESIZE) — 1 down to 0) are sent to the peripheral on slot 0 along with the upper bits of the Indirect addressing register (specifically, bits 32 down to log2(RANGESIZE)) to create an effective 32-bit address.

In this design example, slot 0 is connected to the MSS and addresses the ACE SSE DAC0_BYTE0 register, which has an address of 0x40020064.

CoreABC reads the sample word from the APB_sine_LUT slave on slot S2 and writes to the DAC0_BYTE0 register on slot S0. Slot S1 is used to store the ACE base address for indirect addressing. APB_sine_LUT is a look-up table containing one full cycle of sine wave sample values. Continuously writing to the DAC0_BYTE0 register with the sine_LUT samples generates a periodic sinusoidal signal. For generation of a different pattern, load the LUT/ROM with appropriate signal sample values. The frequency of the output signal can be modulated by varying the delay between each sample in the CoreABC instructions.

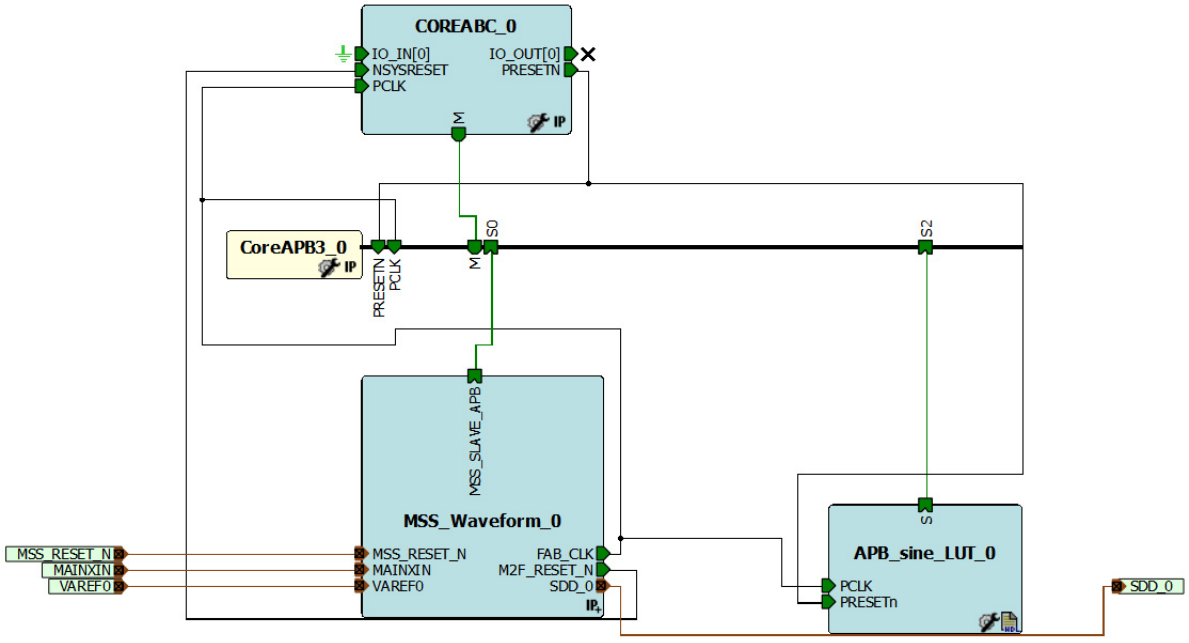

Figure 2 • **Top-Level View of Logic in the Fabric as Master**

Hardware Implementation

The following sections explain the hardware implementation using the Cortex-M3 processor as master and logic in the fabric as master.

Using the ARM Cortex-M3 Processor as Master

1. Choose the main crystal oscillator output as the reference frequency input to the PLL and set the MSS clock frequency to 80 MHz.
2. Enable UART_0 peripheral for serial communication.
3. Enable and configure sigma-delta DAC in the MSS ACE configurator. Figure 3 and Figure 4 on page 6 show the sigma-delta DAC and ACE configuration windows.

Refer to the *MSS ACE Configurator User's Guide* for more information on ACE configuration.

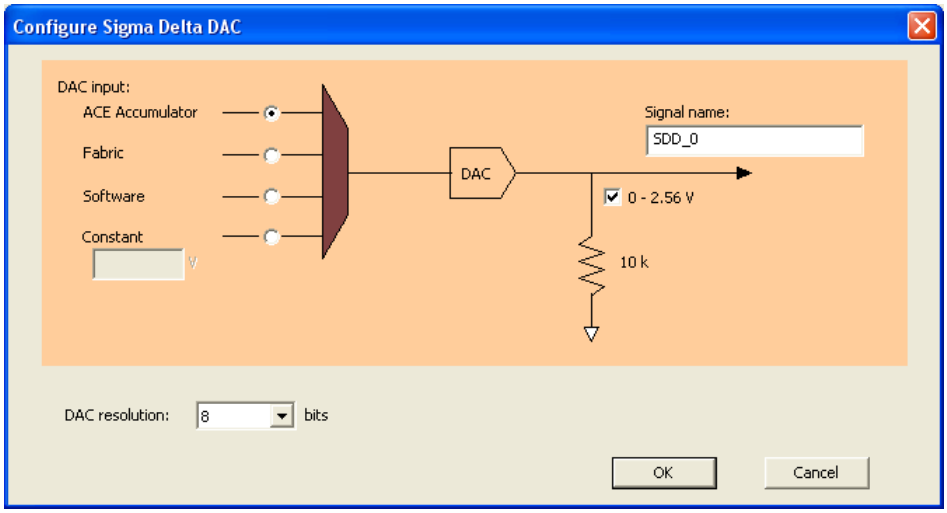

***Figure 3* • Configure Sigma Delta DAC**

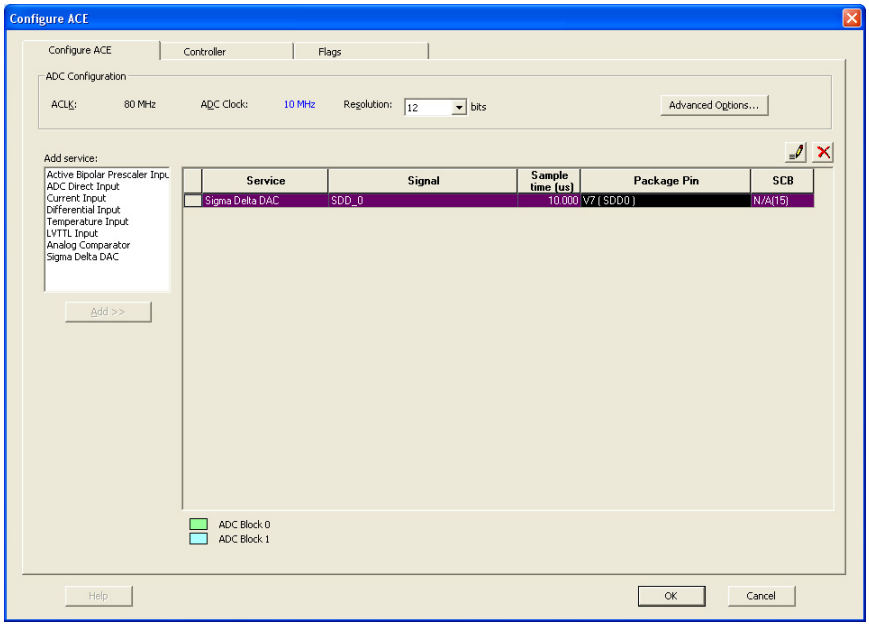

***Figure 4* • Configure ACE Window after DAC Configuration**

4. Generate the MSS component.
5. Create a SmartDesign top-level component and instantiate the generated MSS component in it.
6. Check the design rules for errors if any and generate the design.
7. Complete the synthesis and place-and-route to generate the design programming file.
8. Program the SmartFusion design using FlashPro.

The design files are available in both VHDL and Verilog. Refer to "Appendix A – Design Files" on page 11.

Using Logic in the Fabric as Master

1. Choose the main crystal oscillator output as the reference frequency input to the PLL and set the MSS clock frequency to 80 MHz. Enable FAB_CLK and set the FAB_CLK to 20 MHz.
2. Configure the ACE peripheral as explained in "Using the ARM Cortex-M3 Processor as Master" on page 5.
3. Generate the MSS component.
4. Create a SmartDesign top-level component and instantiate the generated MSS component in it.
5. Add CoreABC and CoreAPB3 and apply the CoreABC program. The CoreABC and CoreAPB3 bus components are instantiated from the **Cores** tab.
6. Create a sine_LUT with APB3 interface and instantiate it in TOP. Refer to *Building an APB3 Core for SmartFusion FPGAs* application note for implementing an APB3 interface in a custom logic block.
7. Interconnect the instantiated components in the TOP, as shown in Figure 2 on page 5. Check the design rules for errors and generate the design.
8. Complete the synthesis and place-and-route to generate the design programming file.
9. Program the SmartFusion design using FlashPro.

The design files are available in both VHDL and Verilog. Refer to "Appendix A – Design Files" on page 11).

Software Implementation

Software implementation is required only when you use the ARM Cortex-M3 processor as master. Software design includes an application program to compute waveform sample values using the Cortex- M3 processor and writing these computed sample words to the DAC register.

Using the Cortex-M3 Processor as Master

Compute the required waveform sample values using the Cortex-M3 processor. Use ACE driver API ACE_set_sdd_value() to write the computed sample value to the selected DAC input register.

The description of the APIs used in the design:

1. ACE_init()

   The ACE_init() function initializes the SmartFusion MSS ACE driver. It initializes the ACE driver's internal data structures. The ACE_init() function must be called before any other MSS ACE driver functions can be called.

2. ACE_set_sdd_value()

   This function is used to set the value of the output of a sigma-delta DAC. It uses the ACE's phase accumulator to generate the 1-bit input bit stream into the SDD, which in turn defines the voltage or current generated at the SDD output.

3. ACE_configure_sdd()

   The ACE_configure_sdd() function is used to configure the operating mode of the sigma delta DAC (SDD) specified as a parameter. It allows selecting whether or not the SDD will output a voltage or a current. A current between 0 and 256 uA is generated in Current mode. A voltage between 0 and 2.56 V is generated in Voltage mode. This function also allows selecting whether or not Return To Zero (RTZ) mode is enabled.

4. ACE_enable_sdd()

   The ACE_enable_sdd() function is used to enable a sigma-delta DAC.

Running the Design

Board Settings

The design example works on the SmartFusion Development Kit Board and the SmartFusion Evaluation Kit Board with default board settings. Refer to the following user's guides for default board settings:

- *SmartFusion Development Kit User's Guide*
- *SmartFusion Evaluation Kit User's Guide*

Program the Design

Program the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board with the provided STAPL file (refer to "Appendix A – Design Files" on page 11) using FlashPro and then power cycle the board.

The SmartFusion DAC output pin (SDD_0) is mapped to the mixed signal header pin 45 on the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board. The generated analog waveform can be observed on an oscilloscope by probing mixed signal header pin 45.

Note that the probe's ground port must be connected to any of the analog ground pins available on the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board—AGND of J5 on the SmartFusion Evaluation Kit Board. Refer to the *SmartFusion Evaluation Kit User's Guide* or the *SmartFusion Development Kit User's Guide* for schematics and pin mapping.

Note: You may observe amplitude attenuation at the output of SDD_0 for higher sampling rates on the SmartFusion Evaluation Kit Board. This is because the SDD_0 output is also looped back to ADC0 through an external low-pass filter for voltage monitoring. Depopulate the resistor R501 to get full scale voltage at SDD_0 or use SDD_1 for generating high frequency waveforms.

Using the ARM Cortex-M3 Processor as Master

Invoke the SoftConsole IDE from the Libero System-on-Chip (SoC) project (refer to "Appendix A – Design Files" on page 11) and launch the debugger. Start a HyperTerminal session with 57600 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 *Configuring Serial Terminal Emulation Programs* tutorial for configuring HyperTerminal, Tera Term, or PuTTY.

When you run the debugger in SoftConsole, the HyperTerminal window provides the user interface to select the pattern to generate. Figure 5 shows the screenshot of the HyperTerminal.

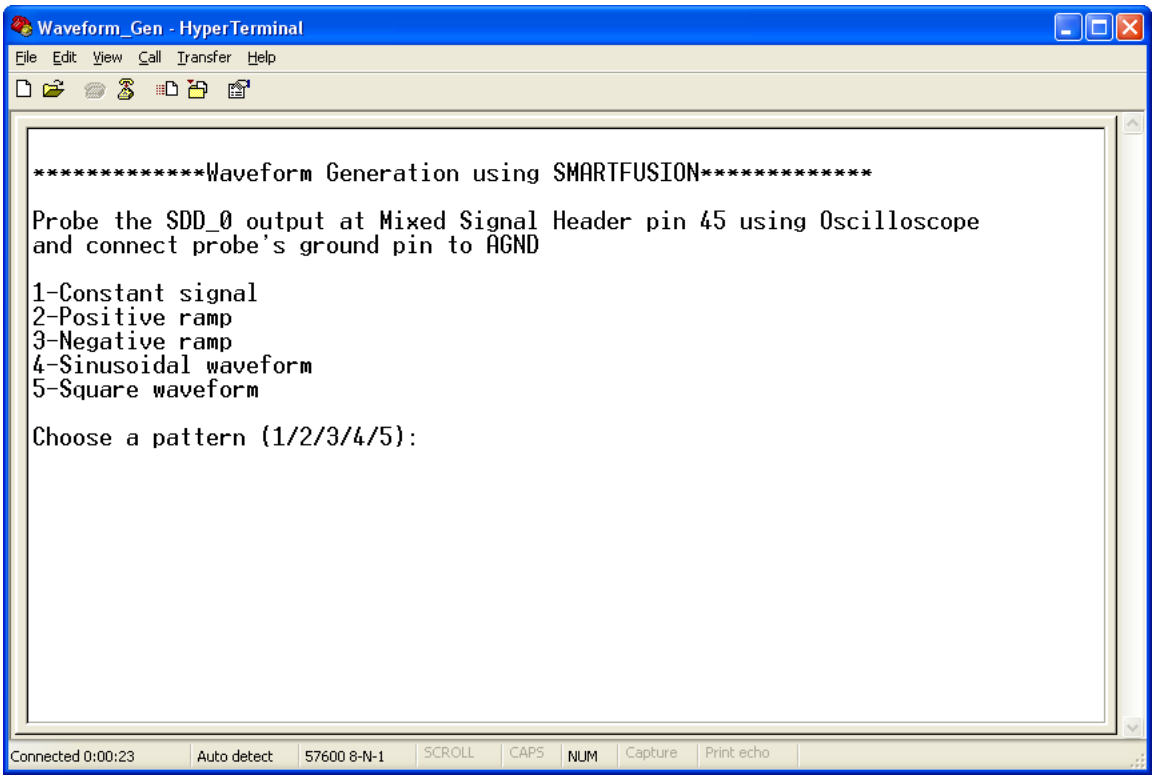

***Figure 5* • Screenshot of HyperTerminal**

Press a number from 1 to 5 to generate the corresponding waveform pattern. Observe the output on oscilloscope. Figure 6 shows the generated output waveform with 256 samples. Increasing the number of samples increases smoothness of the waveform and reduces the frequency of the waveform.

Generated Waveforms Using Cortex-M3 Processor as Master

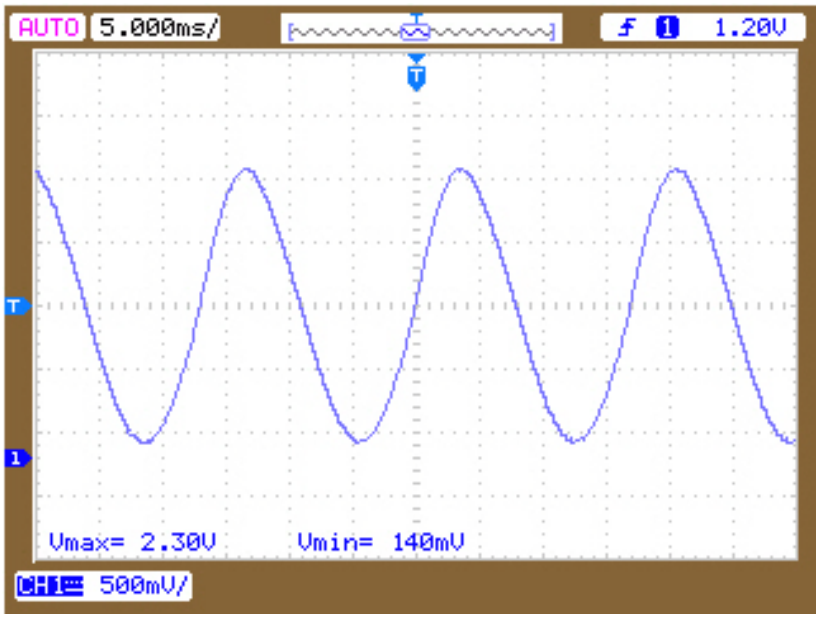

***Figure 6* • Generated Sinusoidal Waveform**

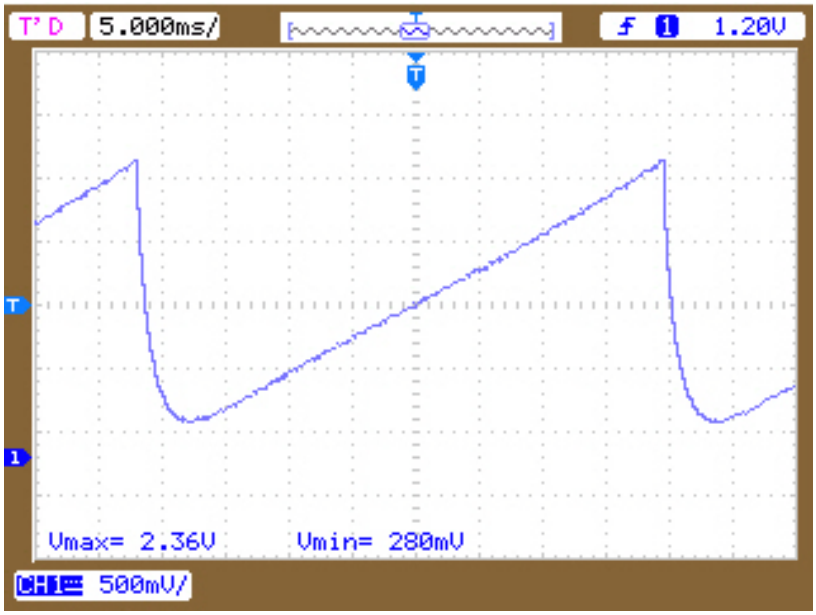

***Figure 7* • Generated Positive Ramp**

Using Logic in the Fabric as Master

After programming the SmartFusion cSoC device with the corresponding STAPL file that is provided, press the RESET switch on the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board. Pressing the RESET switch begins execution of the design. The waveform generated can be observed on the oscilloscope. Increasing the number of sample values increases the output signal smoothness and decreases the harmonics. Figure 8 shows the output waveform with 256 sine samples in the ROM.

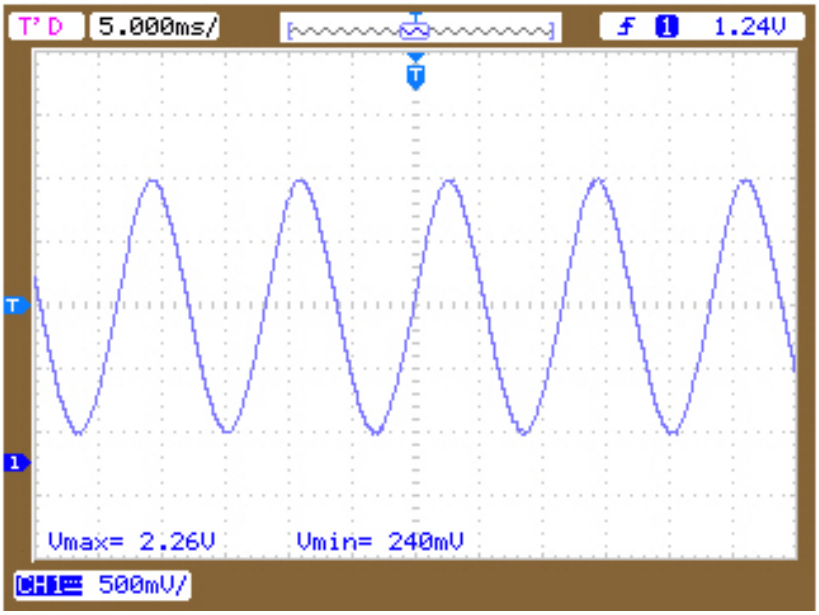

***Figure 8* • Generated Sinusoidal Waveform with Logic in the Fabric as Master**

Release Mode

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

Refer to the *Building Executable Image in Release Mode and Loading into eNVM tutorial* for more information on building an application in Release mode.

Conclusion

This application note introduced the features of SmartFusion cSoC FPGAs and highlighted the features of the ACE DAC. This application note also described how to use the 1-bit DAC on the SmartFusion cSoC device to generate an analog waveform using the Cortex-M3 processor as master or logic in the FPGA fabric as masters.

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_AC350_DF.

The design zip file consists of Libero SoC projects and programming file (*.stp) for A2F500 and A2F200. Refer to the Readme.txt file included in the design file 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_AC350_PF.

The programming file consists of STAPL programming files (*.stp) for A2F500 and A2F200, 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 3 (January 2013)	Added "Board Settings" section and "Program the Design" section (SAR 43469).	8
Revision 2 (February 2012)	Removed ".zip" extension in the Design files link (SAR 36763).	11
Revision 1 (January 2012)	Added Figure 2 (SAR 35789).	5
	Modified "Using the ARM Cortex-M3 Processor as Master" section (SAR 35789).	8
	Modified "Running the Design" section (SAR 35789).	8
	Added section called "Release Mode" (SAR 35789).	11
	Modified the "Appendix A – Design Files" section (SAR 35789).	11

*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 the industry's most comprehensive portfolio of semiconductor technology. Committed to solving the most critical system challenges, Microsemi's products include high-performance, high-reliability analog and RF devices, mixed signal integrated circuits, FPGAs and customizable SoCs, and complete subsystems. Microsemi serves leading system manufacturers around the world in the defense, security, aerospace, enterprise, commercial, and industrial markets. Learn more at **www.microsemi.com**.

51900218-3/01.13