# SmartFusion cSoC: Multi-Channel FFT Co-Processor Using FPGA Fabric

## Table of Contents



## Introduction

The SmartFusion® customizable system-on-chip (cSoC) device integrates 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, peripheral direct memory access (PDMA), embedded nonvolatile memory (eNVM), embedded SRAM (eSRAM), embedded FlashROM (eFROM), external memory controller (EMC), Watchdog Timer, the Philips Inter-Integrated Circuit ($I^2C$), serial peripheral interface (SPI), 10/100 Ethernet controller, real-time counter (RTC), GPIO block, fabric interface controller (FIC), in-application programming (IAP), and analog compute engine (ACE).

The SmartFusion cSoC device is a good fit for applications that require interface with many analog sensors and analog channels. SmartFusion cSoC devices have a versatile analog front-end (AFE) that complements the ARM Cortex-M3 processor based MSS and general-purpose FPGA fabric. The SmartFusion AFE includes three 12-bit successive approximation register (SAR) ADCs, one first order sigma-delta DAC (SDD) per ADC, 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) and handles without intervention of Cortex-M3 processor. Refer to the *SmartFusion Programmable Analog User's Guide* for more details.

This application note describes the capability of SmartFusion cSoC devices to compute the Fast Fourier Transform (FFT) in real time. The Multi Channel FFT example design can be used in medical applications, sensor network applications, multi channel audio Spectrum analyzers, Smart Metering, and sensing applications (such as vibration analysis).

This example design uses the Cortex-M3 processor in the SmartFusion MSS as a master and the FFT processor in the FPGA fabric as a slave. All three of the SmartFusion cSoC A2F500's ADCs are used for data acquisition. The example design uses Microsemi's CoreFFT IP and the advanced peripheral bus interface (CoreAPB3). A custom-made APB3 interface has been developed to connect CoreFFT with the MSS via CoreAPB3. The Cortex-M3 processor uses the PDMA controller in the MSS for the data transfer and thus helps to free up the Cortex-M3 processor instruction bandwidth.

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

# Design Overview

This design example demonstrates the capability of the SmartFusion cSoC device to compute the FFT for multiple data channels. The FFT computation is a complex task that utilizes extensive logic resources and computation time. In general, for N number of channels, N number of FFT IP's are needed to be instantiated, which in turn utilize more logic resources on the FPGA. A way to avoid this limitation is to use the same FFT logic for multiple input channels.

This design illustrates the implementation of a Multichannel FFT to process multiple data channels through a single FFT and store FFT points in a buffer. The FFT computes the input data read from each channel and stores the N-point result in the respective channel's allocated buffer. The channel multiplexing is done once each channel buffer has been loaded with the FFT length.

Computing frequency components for a real time data of six channels is described in this application note. For sampling the input signals the AFE is used and the complex FFT computation is implemented in the fabric of the SmartFusion cSoC device. The Cortex-M3 processor in the MSS of the SmartFusion cSoC handles the buffer management and channel muxing.

Figure 1 depicts the block diagram of six channel FFT co-processor in FPGA fabric.

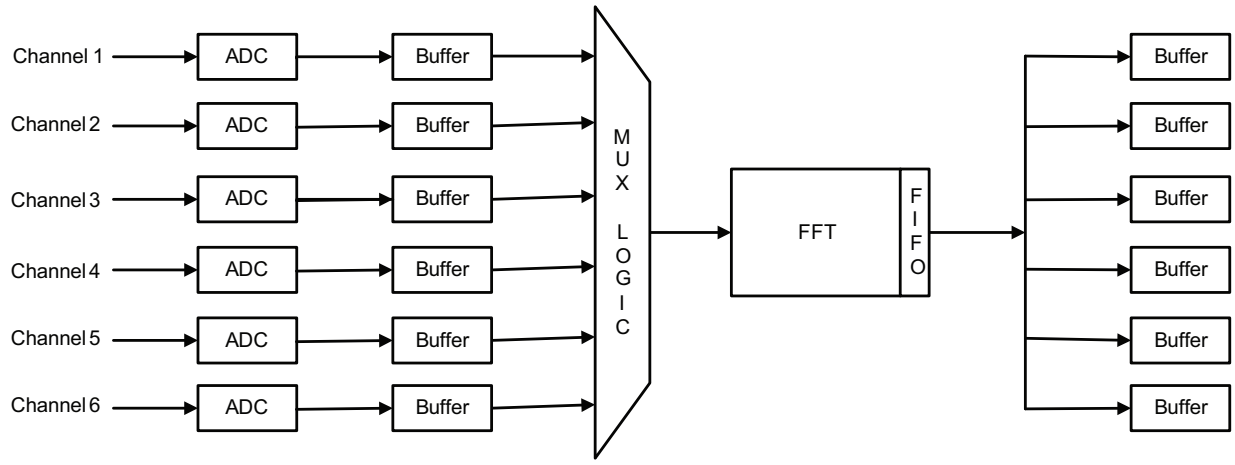


***Figure 1* • Multi Channel FFT Block Diagram**

# Design Description

The design uses CoreFFT for computing the FFT results. You can download the core generator for CoreFFT at www.microsemi.com/soc/portal/default.aspx?r=4&p=m=624,ev=60.

The design example uses a 512-point and 16-bit FFT. A custom-made APB3 interface has been developed to connect CoreFFT IP with the MSS's FIC. The CoreFFT output data is stored in a 512x32 FIFO within the fabric. The FIFO status signals are given in Table 1 on page 3. The status signals indicate that FFT is ready to receive data and data is available in the output of FIFO. These status signals are mapped to the GPIOs in the MSS. The Cortex-M3 processor can read the GPIOs to handle flow control in the data transfer process from the MSS to CoreFFT.

Figure 2 shows the block diagram of logic in the fabric with custom-made APB3 bus.

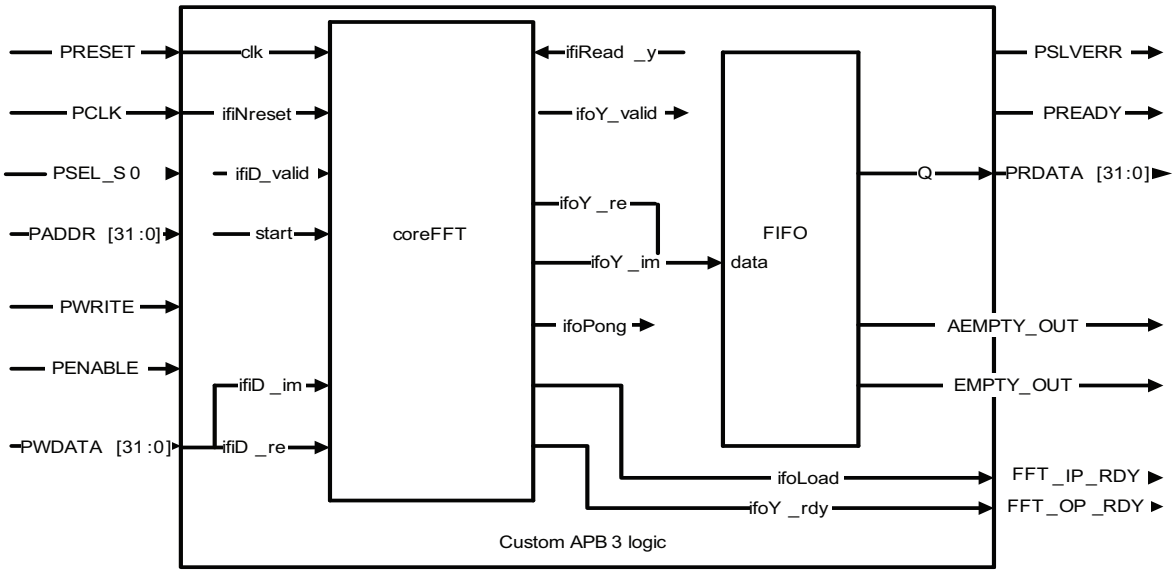


***Figure 2 •*** **CoreFFT with APB Slave Interface**

The data valid signal (ifiD_valid) is generated in custom logic whenever the master needs to write data into the input buffer of the FFT to process through the APB3 interface. The FFT_IP_RDY signal indicates the status of the input buffer of the FFT. If the input buffer is full, the FFT_IP_RDY goes low. The master can read the FFT_IP_RDY signal to get the FFT input buffer status. The FFT generates the processed data with a data valid signal (ifoY_valid). The processed data is stored in the FIFO. When FIFO is not ready to receive output data, it can stop the data fetching from the FFT by pulling down the ifiRead_y signal. The status signal FFT_OP_RDY is used to indicate to the master that processed data is available in the FIFO. FFT_OP_RDY goes High whenever processed data is available in the FFT output buffer. The master can use AEMPTY_OUT or EMPTY_OUT to determine whether the FIFO is empty and all the processed data has been read. Refer to the *CoreFFT Handbook* for more details on architecture and interface signal descriptions.

***Table 1 •*** **FIFO Status Signals with Descriptions**

| Signal | Description |
|---|---|
| FFT_IP_RDY | FFT is ready to receive the Input from the master processor |
| FFT_OP_RDY | Processed data is ready in output buffer of FFT |
| AEMPTY_OUT | Output FIFO is almost empty |
| EMPTY_OUT | Output FIFO is empty |

Three ADCs are configured to have two channels, each channel with 100 ksps sampling rate. The external memory is used for input and output buffers. For each channel, one input buffer having length double to the length of FFT i.e. 1024 words and one output buffer having length equal to the length of FFT i.e. 512 words are used. After each channel's input buffer has 512 points required for the full length of the FFT, each channel, one after the other, streams its points from the FIFO through the FFT. During the FFT computational period, the sampled data values of each channel are stored in the second half of the input buffer. Once the FFT computations for the First half of input buffer completes then the points in the second half of the input buffer will be streamed to FFT. This operation utilizes a ping-pong method.

The Cortex-M3 processor is used for data management, that is, buffering the sampled points and data routing or muxing of these values to the FFT computation block. Sampling of the real time data is done by the ACE. The PDMA handles the data transfer between the external SRAM (eSRAM) buffers and CoreFFT logic in FPGA fabric.

Figure 3 shows the implementation of multi channel FFT on the SmartFusion cSoC device.

## Hardware Implementation

The MSS is configured with an FIC, clock conditioning circuit (CCC), GPIOs, EMC and a UART. The CCC generates 80 MHz clock, which acts as the clock source. The FIC is configured to use a master interface with an AMBA APB3 interface. Four GPIOs in the MSS are configured as inputs that are used to handle flow control in data transfer from MSS to FFT coprocessor. The EMC is configured for Region 0 as Asynchronous RAM and port size as half word. The UART_0 is configured for printing the FFT values to the PC though a serial terminal emulation program.

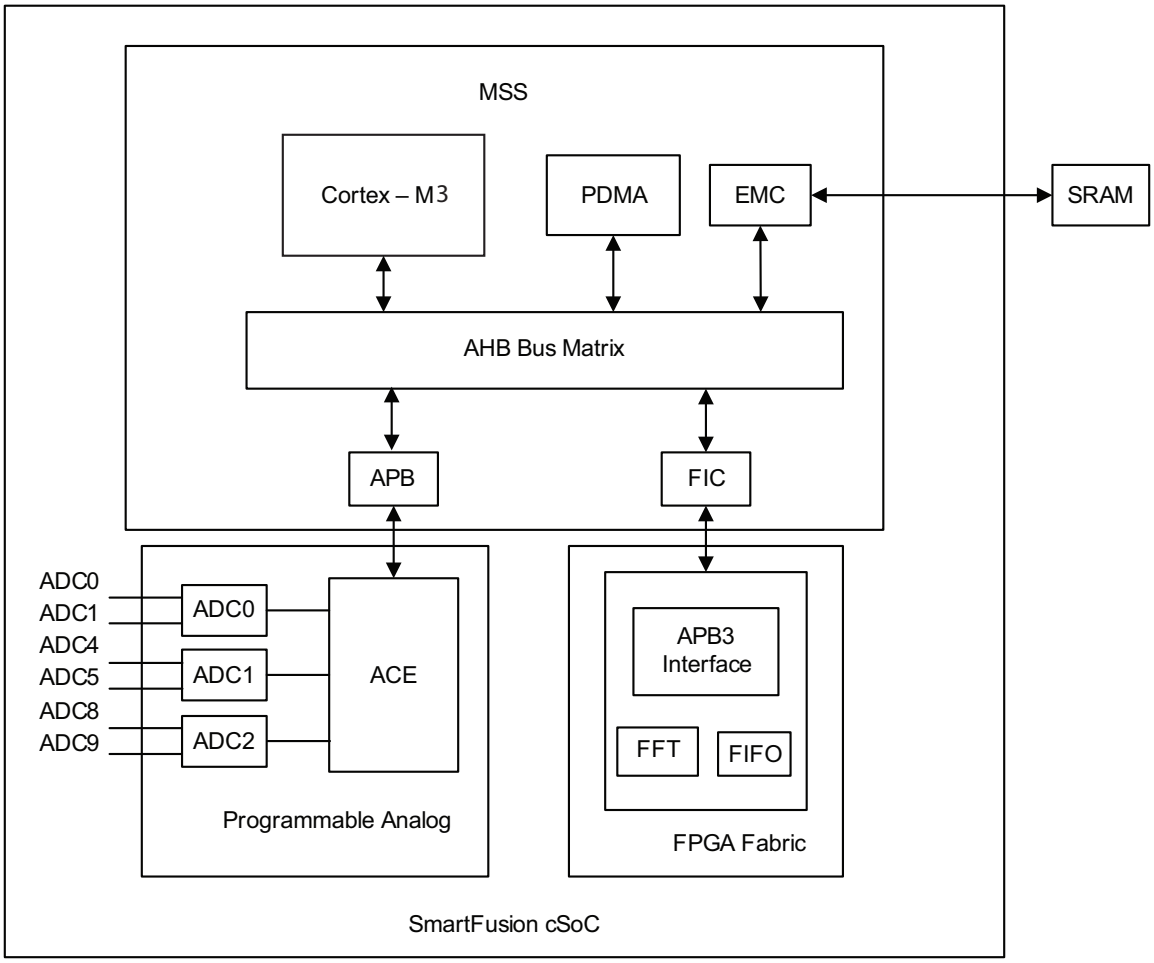


***Figure 3* • Implementation of Multi Channel FFT on the SmartFusion cSoC**

ADC0, ADC1, and ADC2 are configured with 12-bit resolution, two channels and the sampling rate is set to approximately 100 KHz. Figure 4 on page 5 shows the ACE configuration window.

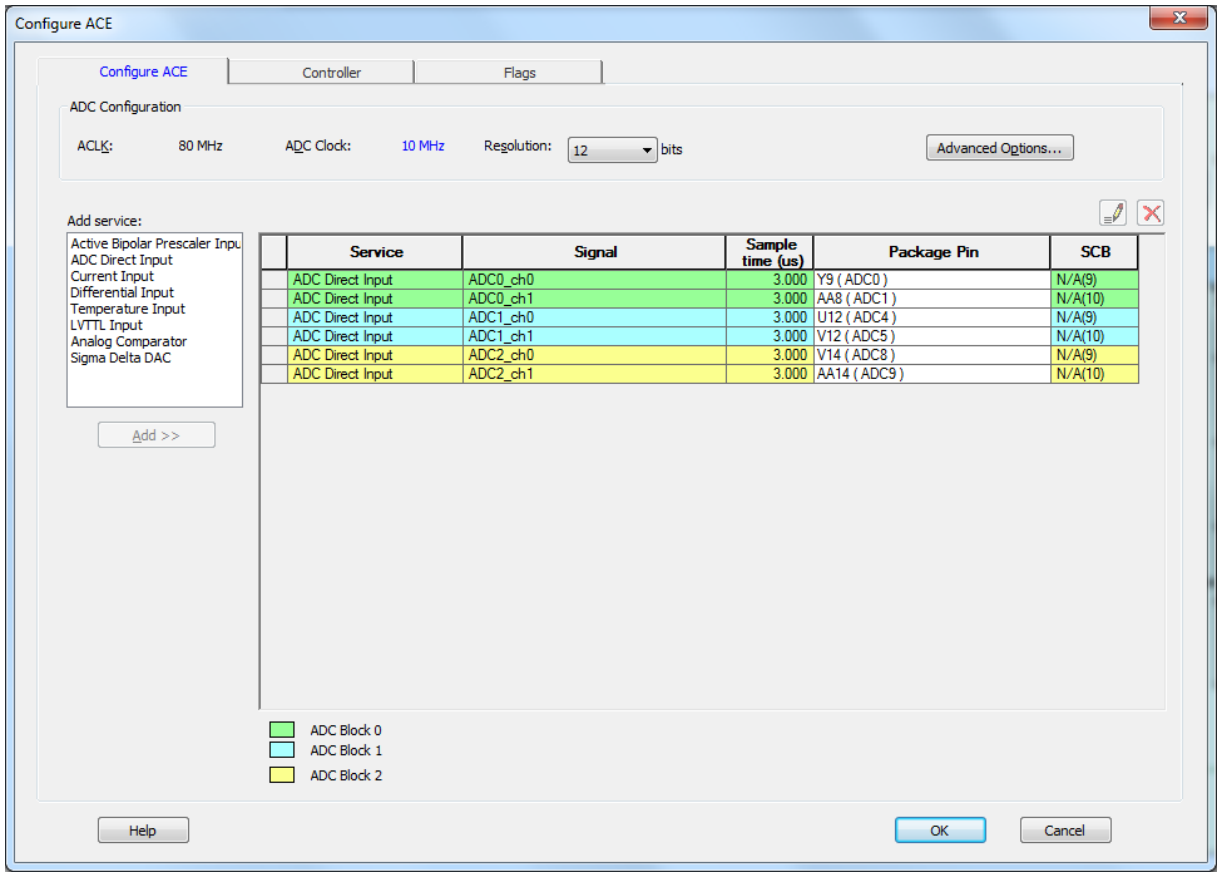


***Figure 4 •*** **Configure ACE**

The APB wrapper logic is implemented on the top of CoreFFT and connected to CoreAPB3. A FIFO of size 512*32 is used to connect to CoreFFT output.

CoreAPB3 acts as a bridge between the MSS and the FFT coprocessor block. It provides an advanced microcontroller bus architecture (AMBA3) advanced peripheral bus (APB3) fabric supporting up to 16 APB slaves. This design example uses one slave slot (Slot 0) to interface with the FFT coprocessor block and is configured with direct addressing mode. Refer to the *CoreAPB3 Handbook* for more details on CoreAPB3 IP.

For more details on how to connect FPGA logic MSS, refer to the *Connecting User Logic to the SmartFusion Microcontroller Subsystem* application note.

The logic in the FPGA fabric consumes 18 RAM blocks out of 24. We cannot use eSRAM blocks for implementing CoreFFT as the transactions between these SRAM blocks and FFT logic are very high and are time critical.

Figure 5 on page 6 illustrates the multi channel FFT example design in the SmartDesign.

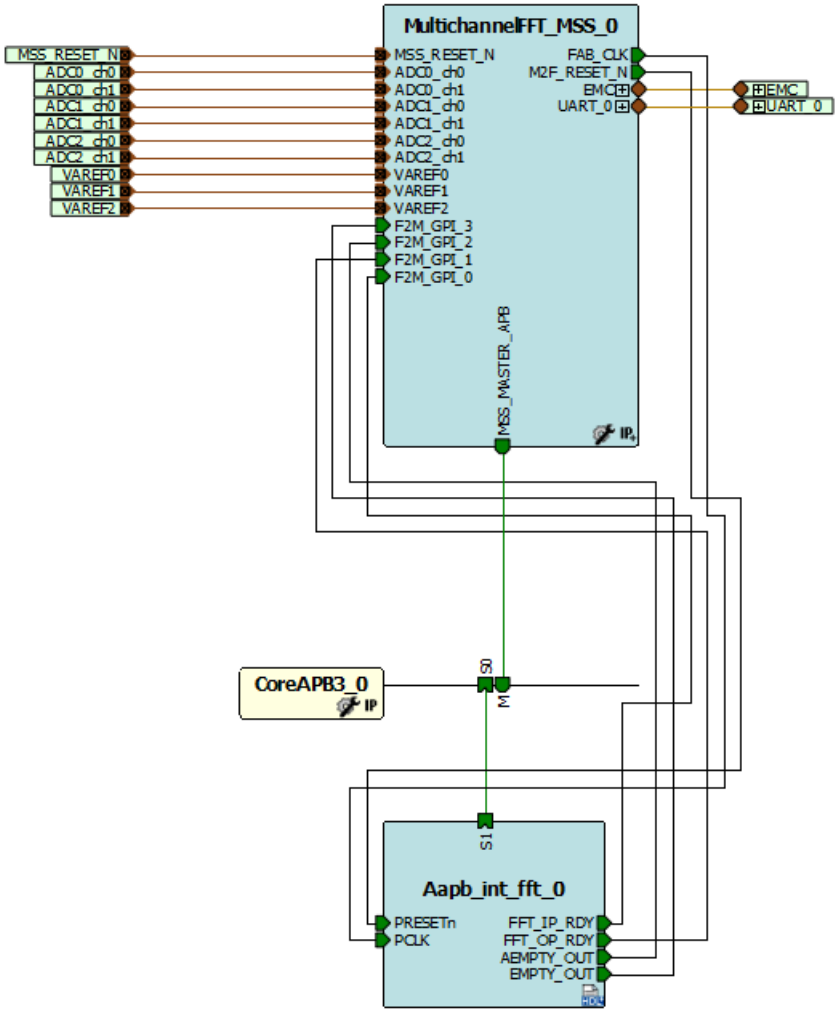


***Figure 5* • SmartDesign Implementation of Multi Channel FFT**

Table 2 summarizes the logic resource utilization of the design on the A2F500M3F device.

***Table 2* • Logic Utilization of the Design on A2F500M3F**

| CoreFFT | | Other Logic in Fabric | Total |
|---|---|---|---|
| Ram Blocks | 14 | 4 | 18 (75%) |
| Tiles | 7842 | 471 | 8313 (72.1%) |

## Software Implementation

The Cortex-M3 processor continuously reads the values from ACE and stores the values into the input buffers. If the first 512 points are filled then the processor initiates the FFT process. In the FFT process, the input buffers are streamed one after other to the CoreFFT with the help of PDMA. Using another channel of PDMA the output of FFT is moved to the corresponding channel output buffers.

During the FFT process the Cortex-M3 processor stores the sampled values into the second half of the input buffers. Once the FFT process completes the first half of input buffer, then the second half of the input buffer are streamed to CoreFFT.

The CALL_FFT(int *) application programmable interface (API) initiates the PDMA to transfer input buffer data to the FFT in the fabric. Before initiating PDMA it checks for FFT whether or not it is ready to read the data. The CALL_FFT(int *) API also checks if the output FIFO is empty so that all the FFT out values have been already read. When the input buffer has points equal to the full length of FFT, then it will be called.

The Read_FFT() API initiates the PDMA for reading the FFT output values from FIFO in fabric to the corresponding output buffer. After reading all the values it calls the CALL_FFT() API with the next channel buffer to compute the FFT for next channel. This is done for all channels. After completion of FFT computation for all channels, if the continuous variable is not defined, it will print the FFT output values on the serial terminal. When FFT_OP_READY interrupt occurs then this API will be called.

The GPIO1_IRQHandler() interrupt service routine occurs on the positive edge of FFT_OP_READY signal. It calls Read_FFT() API. This interrupt mechanism is used to read the sample values continuously while computing the FFT.

If **continuous** variable is defined, then the FFT is computed without any loss of data samples. If **#define continuous** line is commented then after every completion of FFT computation of all channels the FFT output is printed on serial terminal. The printed values are in the form of complex numbers.

The ping-pong mechanism is used for input data buffer to store the samples continuously. For each channel the input buffer length is double of the full FFT length. While computing the FFT for the first half of the buffer, the new sample values are stored in the second half of the input buffer and while computing the FFT for second half of buffer, the new sample values are stored in first half of the input buffer.

## Customizing the Number of Channels

You can change the design depending on your requirement. Configure the ADC (Figure 4 on page 5) with the required number of channels and required sampling rate. In SoftConsole project change the parameter value NUM_CHANNELS according to the ADC configuration. Edit the main code for reading ADCs data into buffers according to ACE configuration.

## Throughput Calculations

The actual time to get 512 samples with 100 ksps is 5.12 ms. Each channel is configured to 100 ksps, so for every 5.12 ms we will have 512 samples in the input buffers.

The actual time taken to compute the FFT for each channel is the sum of time taken to transfer 512 points to CoreFFT, FFT computation time, and time to read FFT output to the output buffer.

- Total time for computing FFT = (time taken to receive 512 data + computational latency for 512 points + time taken to store 512 data) = 512*5 + 23292 + 512*5 =28412 clks
- Time to compute FFT for 6 channels = 28412*6 = 170472 clks

Time to compute FFT for six channels is 2.1309 ms (If CLK is 80 MHz). It is less than half the sample rate of 5.12 ms.

If only one channel is configured with maximum sampling rate (600 ksps) then time to get 512 samples with 600 ksps is 0.853 ms. Time to compute FFT for these 512 samples is 0.355 ms. If you configure three ADCs with maximum sampling rate (1800 ksps) then time to compute the FFT for these three channels will be 1.065 ms which is higher than the sampling time. In this there is a loss of some samples. The design works fine up to 1440 ksps.

# Implementing Multi Channel FFT on EVAL KIT BOARD

To implement the design on the SmartFusion Evaluation Kit Board the FFT must be 256 point and 8 bit because the A2F200 device has less RAM blocks and logic cells. The ADC channels must be selected for only ADC0 and ADC1. Figure 6 on page 8 shows the implementation of multi channel FFT on the SmartFusion cSoC (A2F200M3F) device.

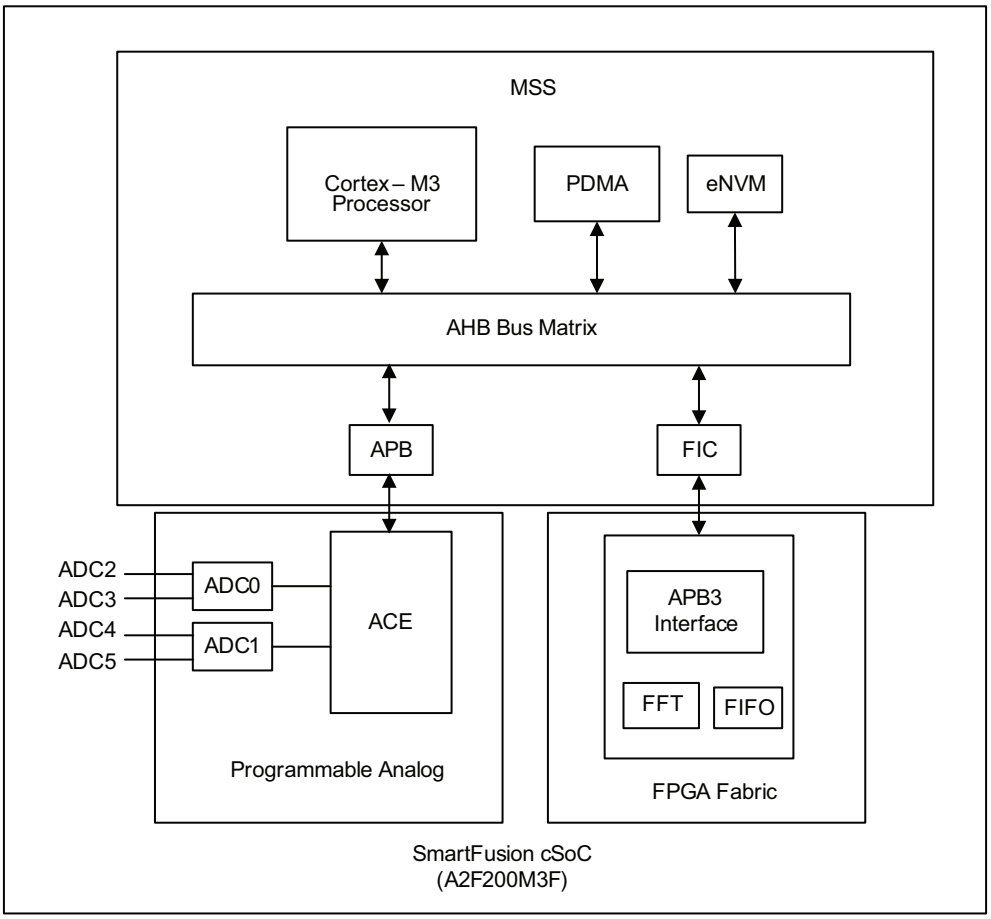


***Figure 6 •*** **Implementation of Multi Channel FFT on the SmartFusion Evaluation Kit Board**

Table 3 summarizes the logic resource utilization of the design with 256 points 8-bit FFT on A2F200M3F device.

***Table 3 •*** **Logic Utilization of the Design on A2F200M3F Device**

| CoreFFT | | Other Logic in Fabric | Total |
|---|---|---|---|
| Ram Blocks | 7 | 1 | 8 (100%) |
| Tiles | 3201 | 85 | 3286 (66%) |

# Running the Design

Program the SmartFusion Evaluation Kit Board or the SmartFusion Development Kit Board with the generated or provided *.stp file (refer to "Appendix A – Design Files" on page 10) using FlashPro and then power cycle the board.

For computing continuous FFT values for the all six signals sampled through the ADCs, uncomment the line **#define continuous** in the main program. The FFT output values are stored in the rdata buffer. This buffer is updated for every computation of FFT.

For printing the FFT values on serial terminal (HyperTerminal or PuTTy), comment the line **#define continuous** in the main program.

Connect the analog inputs to the SmartFusion Kit Board with the information provided in Table 4.

***Table 4 •*** **Settings**

| Channel | Evaluation Kit | Development Kit |
|---|---|---|
| Channel 1 | 73 of J21 (signal header) | ADC0 of JP4 |
| Channel 2 | 74 of J21 (signal header) | ADC1 of JP4 |
| Channel 3 | 77 of J21 (signal header) | 77 of J21 (signal header) |
| Channel 4 | 78 of J21 (signal header) | 78 of J21 (signal header) |
| Channel 5 | | 85 of J21 (signal header) |
| Channel 6 | | 86 of J21 (signal header) |

Invoke the SoftConsole IDE, by clicking on **Write Application code** under Develop Firmware in Libero® System-on-Chip (SoC) project (refer to "Appendix A – Design Files") and launch the debugger. Start HyperTerminal or PuTTY with a baud rate of 57600, 8 data bits, 1 stop bit, no parity, and no flow control. If your PC 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 the HyperTerminal, Tera Term, or PuTTY.

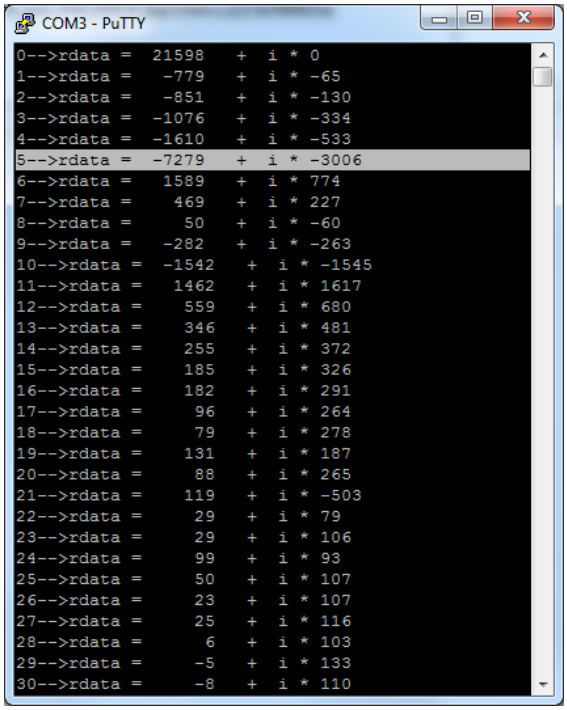


***Figure 7 •*** **FFT Output Data for 1 kHz Sinusoidal Signal on PUTTY**

# Conclusion

This application note describes the capability of the SmartFusion cSoC devices to compute the multi channel FFT. The Cortex-M3 processor, AFE, and FPGA fabric together gives a single chip solution for real time multi channel FFT system. This design example also shows the 6-channel data acquisition system.

# Appendix A – Design Files

The Design files are available for download on the Microsemi SoC Product Groups website: www.microsemi.com/soc/download/rsc/?f=A2F_AC381_DF.

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

**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**.



51900249-0/02.12