SmartFusion cSoC: DSP Coprocessor in FPGA Fabric

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 clock (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).

The intent of this application note is to demonstrate the capability of DSP coprocessing engine in the FPGA fabric of SmartFusion cSoC devices. The DSP coprocessor example design can be used where the Cortex-M3 processor in the SmartFusion cSoC device needs to offload the DSP processing (audio/video/image) to the FPGA fabric to boost overall performance.

This example design uses Cortex-M3 processor in the SmartFusion MSS as master and the fast fourier transform processor (CoreFFT IP) in fabric as DSP coprocessor. The Cortex-M3 processor can provide data directly to the coprocessor for processing or it can use the peripheral direct memory access (PDMA) controller in the MSS for the data transfer and thus helps to free up Cortex-M3 processor. This application note demonstrates both methodologies. The example design uses both CoreFFT and advanced peripheral bus interface (CoreAPB3). A custom-made APB3 interface has been developed to connect CoreFFT with the MSS via CoreAPB3.

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

Design Example Overview

This design example demonstrates the capability of the DSP coprocessing engine in the FPGA fabric on the SmartFusion Evaluation Kit Board and the SmartFusion Development Kit Board. The design example consists of a DSP coprocessor block and CoreAPB3 implemented in the FPGA fabric. CoreAPB3 connects the DSP coprocessor with MSS. The DSP coprocessor has CoreFFT IP, 64x16 FIFO, and a custom-made APB3 interface. Figure 1 illustrates how to interface the DSP coprocessor with the MSS. The GPIOs in the MSS are used to handle flow control in data transfer from the MSS to DSP coprocessor. The UART in the MSS is used for printing the FFT values on HyperTerminal.

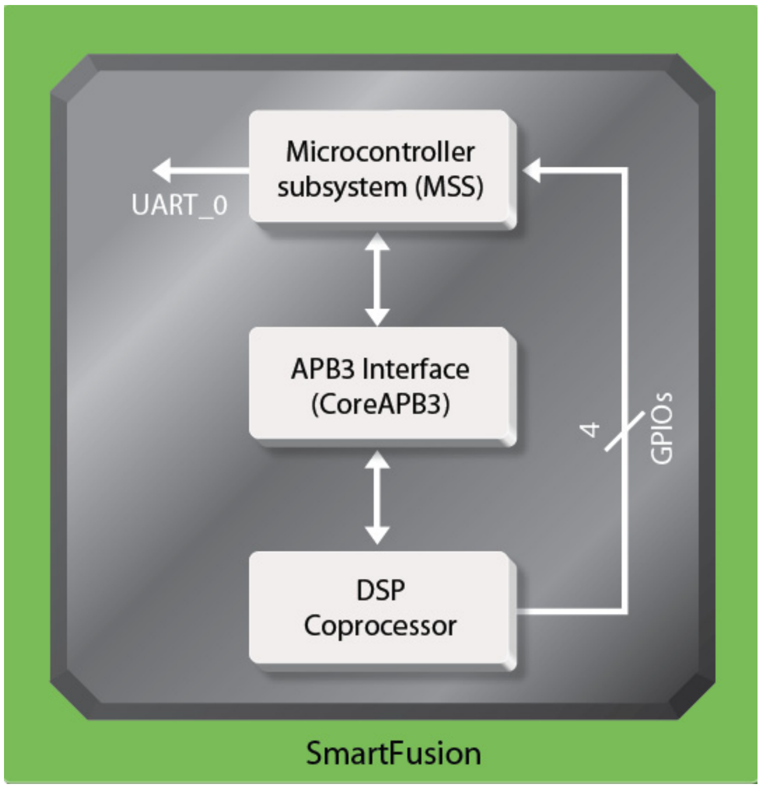

Figure 1 **• System Level Interface Signals**

Description of the Design Example

Description of DSP Coprocessor

The design uses CoreFFT as a DSP coprocessor. You can download the core generator for CoreFFT from www.microsemi.com/soc/products/ip/search/evaluate.aspx?m=624&ev=60. The example design uses a 256-point and 8-bit FFT. A custom-made APB3 interface has been developed to connect CoreFFT with MSS. CoreFFT output data is stored in a 64x16 FIFO. The FIFO status signals are given in Table 1 on page 3. 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 GPIOs to handle flow control in data transfer from the MSS to the DSP coprocessor.

You can implement various methodologies based on the application to handle data transfer, such as polling of the general purpose input/output (GPIOs) for status, using GPIOs as interrupts to the MSS or using PDMA. Figure 2 on page 3 shows the block diagram of DSP coprocessor with custom-made APB3 bus.

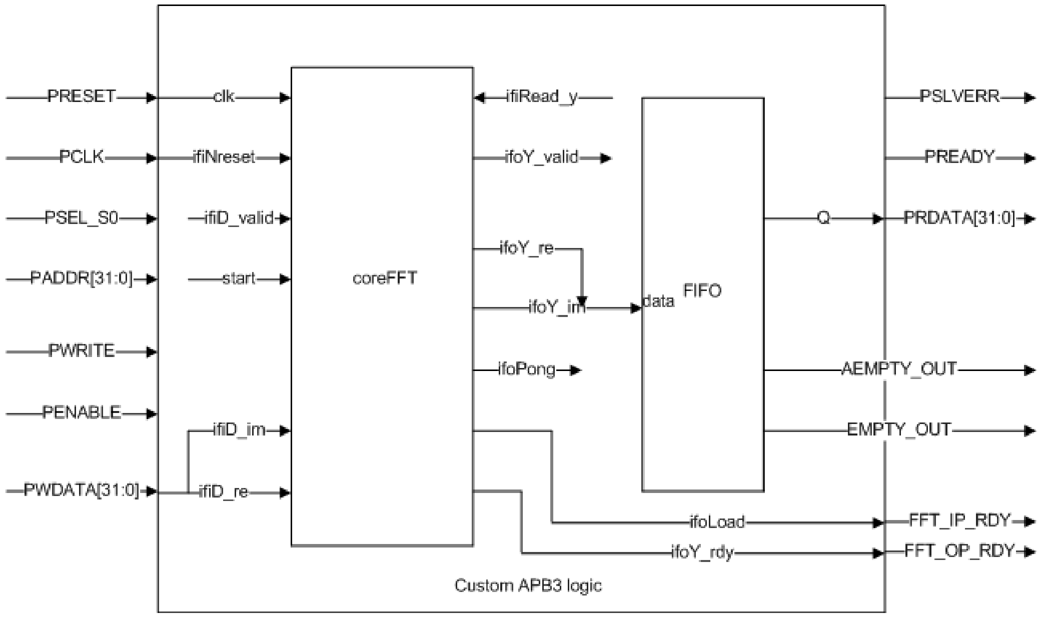

***Figure 2* • DSP Coprocessor with Custom-Made APB3 Bus**

The data valid signal 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. The processed data is stored in FIFO. When FIFO is not ready to receive the 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 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 or not 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

SmartDesign Implementations

The design example consists of the MSS, CoreAPB3, and the DSP coprocessor block. The DSP coprocessor block includes CoreFFT, 64x16 FIFO, and a custom-made APB3 interface. The CoreAPB3 connects the DSP coprocessor with the FIC in the MSS.

The MSS is configured with an FIC, clock conditioning circuit (CCC), GPIOs, and a UART. The CCC generates a 40 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 DSP coprocessor. The UART_0 is configured for printing the FFT values on HyperTerminal.

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

Figure 3 illustrates the DSP Coprocessor Example Design in SmartDesign.

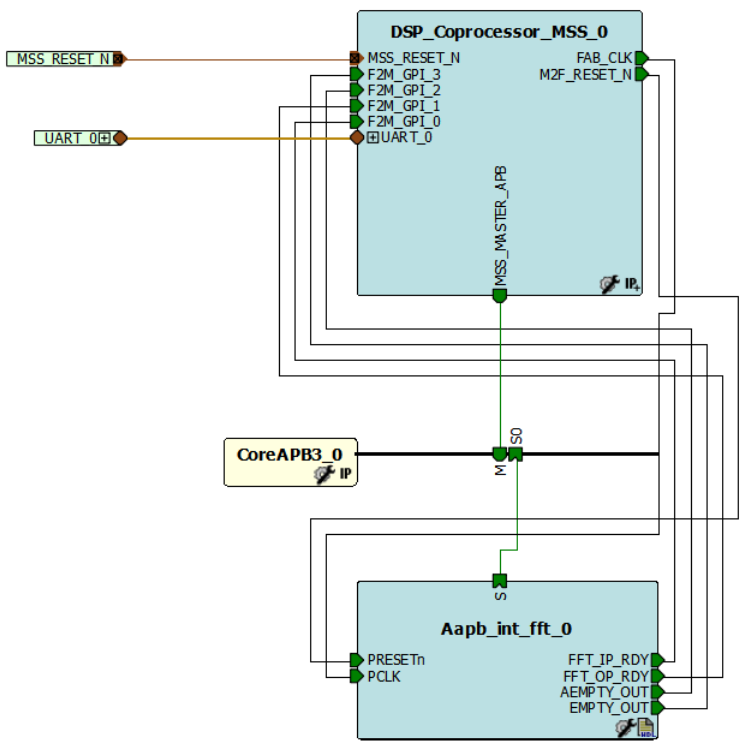

***Figure 3* • DSP Coprocessor Example Design in SmartDesign**

Verilog, VHDL Libero® System-on-Chip (SoC) projects and the SoftConsole project are provided in the design files attached with this design example.

Software API

The CALL_FFT () API can be used whenever DSP coprocessor is required to transform the input data. This API configures the PDMA for data transfer from the Cortex-M3 processor to DSP coprocessor and from DSP coprocessor to the Cortex-M3 processor.

Without PDMA, the Cortex-M3 processor transfers data by polling the status signals.

Data Transfer with PDMA

The PDMA in the MSS consists of eight instances of a single DMA channel design. Each channel can be configured to perform 8-, 16-, and 32-bit transfers from the peripheral to memory, memory to peripheral, or between memory and memory. Channels can be assigned to peripherals or memory arbitrarily.

The Cortex-M3 processor can use PDMA to transfer data to the DSP coprocessor and thus free itself for other tasks. Custom APB3 logic causes the PREADY signal to be low whenever FFT is not ready for the data transfer.

Data Transfer Without PDMA

The Cortex-M3 processor can transfer data from memory for processing or processed data to the memory. The Cortex-M3 processor polls the status signals for the flow control with coprocessor. These status signals are mapped to the GPIOs in the MSS.

Simulation

System level simulation is done by modifying the user.bfm script available in the simulation folder of the MSS to verify the custom interface. Refer to the *DirectCore AMBA BFM User's Guide* for further details on system level simulation with the SmartFusion cSoC device. Figure 4 illustrates the FFT write operation. When data is written to FFT for processing, the data valid signal, if oY_valid, goes High.

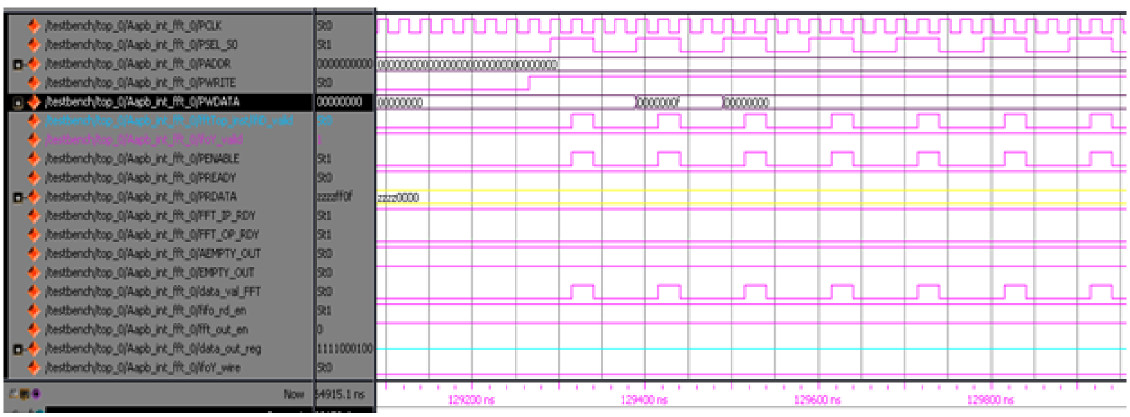

Figure 4 **• CoreFFT Write Operation**

Figure 5 illustrates the FFT read operation. If a read request is received from the Cortex-M3 processor, the fifo_rd_en signal is asserted and data is transferred. Figure 5 shows fft_out_en signal requesting the processed data from the FFT.

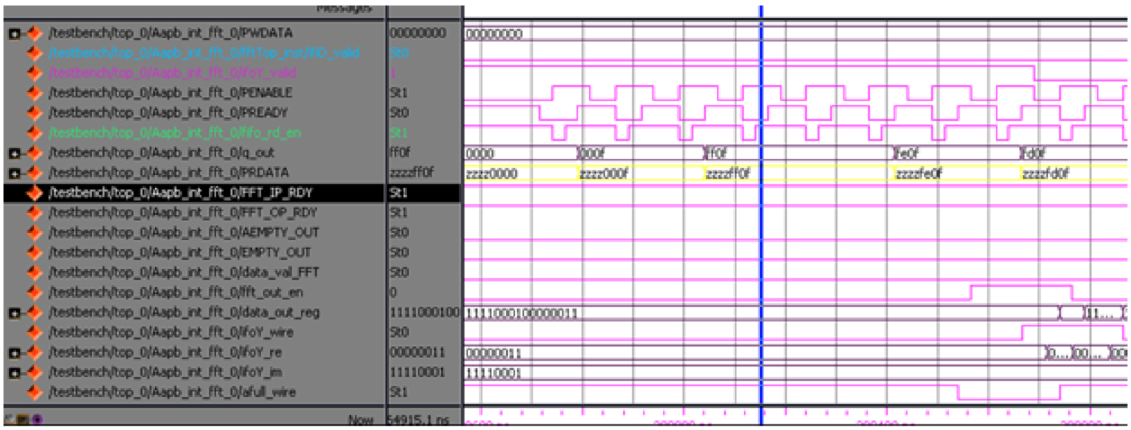

***Figure 5* • CoreFFT Read Operation**

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 and Programming Files" on page 7) using FlashPro and then power cycle the board.

Running the Application

Invoke the SoftConsole IDE from the Libero SoC Project (refer to "Appendix A - Design and Programming Files" on page 7) and launch the debugger. Start a HyperTerminal with a baud rate of 57600, 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 the HyperTerminal, Tera Term, and PuTTY.

When you run the debugger in SoftConsole, the application starts printing the FFT values of a sine wave on HyperTerminal. Figure 6 shows an example of HyperTerminal with FFT values.

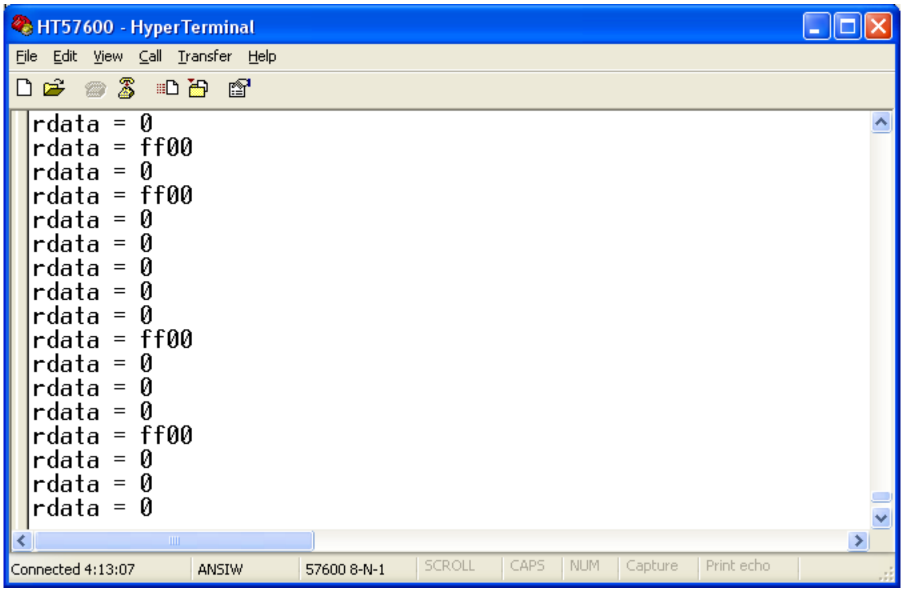

Figure 6 **• Screenshot of HyperTerminal with FFT Values**

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.

Refer to *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 described the capability of DSP coprocessing engine in FPGA fabric of SmartFusion cSoC devices. The DSP coprocessor design example demonstrates the usage where the ARM Cortex-M3 processor in the SmartFusion cSoC devices need to offload the DSP processing (audio/video/image) to FPGA fabric to boost the overall performance. This application note also describes the use of PDMA for further improvements in overall performance.

Appendix A - Design and Programming Files

You can download the design files from the Microsemi SoC Products Group website: www.microsemi.com/soc/download/rsc/?f=A2F_AC355_DF.

The design file consists of Libero Verilog, VHDL projects, and programming files (*.stp). 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_AC355_PF.

The programming zip 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 "Running the Application" section (SAR 43469).	6
Revision 2 (February 2012)	Removed ".zip" extension in the Design and Programming files link (SAR 36763).	7
Revision 1 (January 2012)	Incorporated new Figure 3 (SAR 35791).	4
	Modified the second and third lines in "Running the Design" section (SAR 35791).	6
	Added new section "Release Mode" (SAR 35791).	7
	Modified the "Appendix A - Design and Programming Files" section (SAR 35791).	7

*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

51900223-3/01.13