Cyclone II FFT Co-Processor Reference Design

May 2005 ver. 1.0 Application Note 375

Introduction

The fast Fourier transform (FFT) co-processor reference design demonstrates the use of an Altera® FPGA as a high-performance digital signal processing (DSP) co-processor to the Texas Instruments (TI) TMS320C6000 family of programmable digital signal processors. The reference design uses the *Spectrum Digital DSP Starter Kit (DSK) for the TMS320C6416*, Revision E, which features the TI TMS320C6416 digital signal processor, and the *DSP Development Kit, Cyclone II Edition*, which features the EP2C35 FPGA. The hardware interface is a connection between the TMS320C6416 processor's External Memory Interface (EMIF) and the first-in first-out (FIFO) buffers on the FPGA.

For more information on the Cyclone™ II EP2C35 DSP development board and the EP2C35 FPGA, refer to the *Cyclone II EP2C35 DSP Development Board Reference Manual*.

The reference design includes Verilog hardware description language (HDL) files and C source code for the TMS320C6416 processor.

Background

This section provides background information and describes basic concepts for using an FPGA as a co-processor to a programmable digital signal processor.

Programmable digital signal processors are widely used in a range of signal processing applications. They are designed with optimized instruction sets to execute DSP algorithms such as FFTs and finite impulse response (FIR) filters. Unfortunately, programmable digital signal processor performance has not kept up with the demands of the newest system applications, which often require dramatically higher data rates and increased channel counts. This has forced system designers to implement costly arrays of digital signal processors to satisfy these needs. However, these arrays tend to occupy more board real estate and result in increased power consumption. This affects overall system cost and poses significant implementation challenges, including the arbitration of shared memory between different processors.

The FFT co-processor reference design demonstrates how to implement a design that easily interfaces to a digital signal processor. Designers can adapt the FFT FPGA co-processor reference design to their application because of the programmable nature of the FPGA's fabric. Additionally, designers can customize and construct functions that fully exploit the

parallel nature of the hardware implementation in the FPGA, enabling power-efficient multichannel designs (useful in communication systems) with high data throughputs.

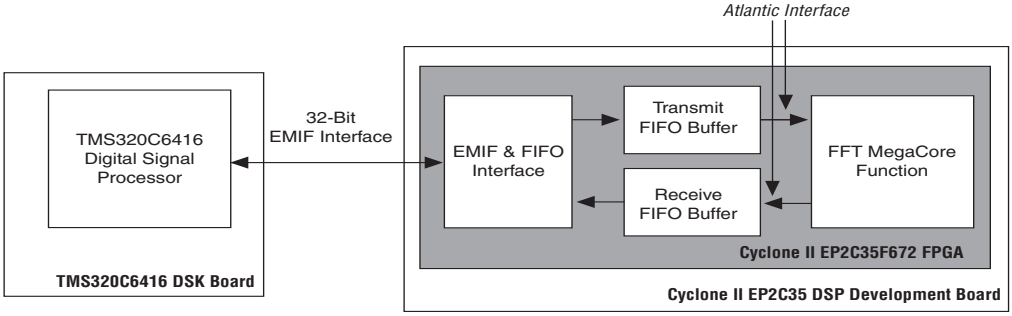
The following steps provide a high-level description of an FPGA co-processor design flow:

1. Profile applications in software to identify computationally intensive algorithms suitable for off-loading to co-processors.
2. Integrate an off-the-shelf co-processor by using an Altera intellectual property (IP) MegaCore® function, or by developing a custom co-processor block using a design tool like DSP Builder, or by using HDL.
3. Evaluate co-processor system architectures and select a suitable processor interface.
4. Integrate the hardware and software design components.
5. Verify the system in hardware.

FFT FPGA Co-Processor Functional Description

Figure 1 shows the FFT FPGA co-processor reference design block diagram.

Figure 1. FFT FPGA Co-Processor Block Diagram



The direct memory access (DMA) controller within the TMS320C6416 processor transmits packets of data to be processed via the TMS320C6416 processor to the EMIF and FIFO interface on the EP2C35 FPGA. The EMIF and FIFO interface sends the data to the transmit FIFO buffer. The reference design monitors the fill level of the transmit FIFO buffer to determine when sufficient data is available for processing by the FFT MegaCore function.

The FFT MegaCore function processes the data in packets the size of the FFT length. The output of the FFT MegaCore function is sent to the receive FIFO buffer. When a whole packet of processed data is available to be read from the receive FIFO buffer, a DMA transfer request is sent to the TMS320C6416 processor. Table 1 lists the scheduling of the data packets through the hardware system blocks to maximize system utilization and data throughput.

Table 1. Data Packet Scheduling Through the FFT Co-Processor Reference Design

Action	Step							
	1	2	3	4	5	6	7	8
EMIF & FIFO Interface	Write 0	-	Write 1	-	Read 0	Write 2	Read 1	Write 3
Transmit FIFO Buffer	-	In 0	-	In 1	-	-	In 2	-
FFT MegaCore Function	-	-	FFT 0	-	FFT 1	-	-	FFT 2
Receive FIFO Buffer	-	-	-	Out 0	-	Out 1	-	-

Table 1 lists the actions that occur in each step. The following list describes each action:

- Write: the TMS320C6416 processor writes to the transmit FIFO buffer
- In: input to transmit FIFO buffer from the EMIF and FIFO interface
- FFT: input to FFT MegaCore function from the transmit FIFO buffer
- Out: output from FFT MegaCore function into the receive FIFO buffer
- Read: the TMS320C6416 processor reads data from the receive FIFO buffer

FFT MegaCore Function

The Altera FFT MegaCore function is high-performance and highly parameterizable. It is optimized for the Stratix® II, Stratix, Stratix GX, Cyclone II, and Cyclone device families. The FFT MegaCore function implements a complex FFT or inverse FFT (IFFT) for high-performance applications.

The FFT MegaCore function has the following features:

- Radix-4 and mixed radix-4/2 implementations
- Block floating-point architecture to maintain the maximum dynamic range of data during processing
- High throughput quad-output radix 4 FFT engine
- Support for multiple single-output and quad-output engines in parallel
- Multiple I/O data flow modes: streaming, buffered burst, and burst
- Parameterization-specific VHDL and Verilog HDL testbench generation
- Transform direction (FFT and IFFT) specifiable on a per-block basis
- Bit-accurate MATLAB models
- Optimized to use Stratix II, Stratix, and Stratix GX DSP blocks and the TriMatrix™ memory architecture
- Altera Atlantic™-compliant input and output interfaces (see "Atlantic Interface")
- Easy-to-use IP Toolbench interface
- IP functional simulation models for use in Altera-supported VHDL and Verilog HDL simulators
- Support for OpenCore® Plus evaluation
- DSP Builder ready

For more information on the FFT MegaCore function, refer to the *FFT MegaCore Function User Guide*.

Table 2 lists the FFT MegaCore function parameters used in the reference design.

Table 2. FFT MegaCore Function Reference Design Parameters

Parameter	Value
Target device family	Cyclone II
Transform length	1,024 points
Data precision	16 bits
Twiddle precision	16 bits
FFT engine architecture	Quad output
Number of parallel FFT engines	1
I/O data flow	Buffered burst
Structure	4 multipliers and 2 adders
Implement multipliers in	Logic cells
Implement appropriate logic functions in RAM	Yes

Atlantic Interface

The FFT MegaCore function uses the Altera Atlantic interface. The Atlantic interface is a flexible interface for high-throughput packet-based data transmission of arbitrary packet length. It provides a synchronous point-to-point connection between two blocks of logic with flexible flow control for master-to-slave and slave-to master directions.

For more information on the Atlantic interface, refer to *FS 13: The Atlantic Interface Functional Specification*.

Transmit & Receive FIFO Buffers

The transmit and receive FIFO buffers handle the flow control of data to and from the FFT MegaCore function. The transmit FIFO buffer receives data from the TMS320C6416 processor across the EMIF interface and buffers an entire packet of data before sending it to the FFT MegaCore function.

As data becomes available on the output of the FFT MegaCore function, it is sent to the receive FIFO buffer. The receive FIFO buffer buffers an entire packet of data before sending it across the EMIF to the TMS320C6416 processor.

The programmable thresholds in the transmit and receive FIFO buffers are set to a little less than the length of one packet. For example, for a 1,024-point 16-bit FFT, the threshold value is set at 1,022 because each packet requires 1,024 32 bit-words. This assumes that the word size of each FIFO is set to 32 bits. The FIFOs are set to a depth of 2,048 to avoid any data overflow that might occur when writing to the FIFO buffers.

EMIF Connection Methods

The EMIF provides an interface to a variety of external memory components including synchronous dynamic random access memory (SDRAM), static random access memory (SRAM), and FIFO buffers. The EMIF is the preferred connection method for the FPGA co-processor because of the data transfer rates available and the possibility of using the enhanced DMA (EDMA) controller integrated within the TMS320C6416 processor. Additionally, the *DSP Development Kit, Cyclone II Edition* includes expansion headers that are compatible with the EMIF expansion headers on the TMS320C6416 board. Table 3 lists the peak data transfer rates achievable by the EMIF for the given clock rates.

In the FFT co-processor reference design, the FPGA co-processor is connected to the TMS320C6416 processor via a 32-bit asynchronous EMIF. The FPGA co-processor appears within the TMS320C6416 processor's memory map in the chip select 3 address space.

Table 3. Peak EMIF Data Rates

EMIF Mode	Peak Transfer Rates (Mbps)		
	@66 MHz	@100 MHz	@133 MHz
32-bit asynchronous *(1)*	53	80	106
32-bit synchronous	264	400	532
64-bit synchronous	528	800	1,064

Note to Table 3:
(1) Each asynchronous access is assumed to be five EMIF clocks.

The FPGA co-processor may be implemented as a memory mapped device using either a synchronous or an asynchronous EMIF connection as determined by the system's performance requirements. FIFO buffers may be used to allow EMIF burst accesses to proceed without wait states, independently of the rate at which the FPGA co-processor consumes or produces data.

Transmit packets (from the processor to the FPGA) are written to the transmit FIFO buffer and receive packets (from the FPGA to the processor) are read from the receive FIFO buffer. FIFO status signals are available to the processor (from the receive FIFO buffer) and to the FFT MegaCore function (from the transmit FIFO buffer).

Table 4 describes the signals in the supplied Verilog source code.

Table 4. Verilog Source Code Signals (Part 1 of 2)

Signal Name	Width (Bits)	Direction	Description
`clk`	1	I	Clock
`rst_n`	1	I	Asynchronous reset, active low
`soft_reset`	1	O	Software controlled reset for FPGA co-processor
`emif_ea [21:0]`	22	I	EMIF address bus
`emif_ce_n`	1	I	EMIF device enable, device selected must be set up as asynchronous
`emif_be_n[3:0]`	4	I	EMIF byte enable

Table 4. Verilog Source Code Signals (Part 2 of 2)

Signal Name	Width (Bits)	Direction	Description
emif_aoe_n	1	I	EMIF asynchronous output enable
emif_are_n	1	I	EMIF asynchronous read enable
emif_awe_n	1	I	EMIF asynchronous write enable
emif_ardy	1	O	EMIF asynchronous ready
emif_ed [31:0]	32	I/O	EMIF data
addr	1	O	Control port address
wdata	1	O	Control port write data
hwa_ce	1	O	Control port chip enable
write	1	O	Control port write strobe
hwa_rdata	1	O	Control port read data
tx_dma_evrq	1	O	Transmit DMA event request. Asserted low to request a new block of data to be encoded
rx_dma_evrq	1	O	Receive DMA event request. Asserted low to signal that a block of encoded data is available
tx_full	1	I	Transmit (software to hardware) FIFO buffer status
rx_full	1	I	Receive (hardware to software) FIFO buffer status
tx_dav	1	I	Atlantic master source (transmit) data available
tx_ena	1	O	Atlantic master source (transmit) enable
tx_dat [31:0]	32	O	Atlantic master source (transmit) data
tx_adr [3:0]	4	O	Atlantic master source (transmit) address
tx_eop	1	O	Atlantic master source (transmit) end of packet
tx_sop	1	O	Atlantic master source (transmit) start of packet
rx_ena	1	O	Atlantic master sink (receive) enable
rx_dat [31:0]	32	I	Atlantic master sink (receive) data
rx_eop	1	I	Atlantic master sink (receive) end of packet
rx_sop	1	I	Atlantic master sink (receive) start of packet
rx_err	1	I	Atlantic master sink (receive) error
rx_dav	1	I	Atlantic master sink (receive) data available
rx_val	1	I	Atlantic master sink (receive) data valid

Registers

Table 5 summarizes the registers in the FFT co-processor reference design.

Table 5. FFT Co-Processor Reference Design Registers

Register Mnemonic	Address (Hex) (1)	Access	Description
TX_CREDIT	-	-	Transmit credit register—not directly accessible
RESET_HW	526C	Write	Sends a soft reset to the hardware
TX_CREDIT_INC	523C	Write	Increment TX_CREDIT register

Note to Table 5:
(1) The addresses are offsets from the FPGA base address of 0xB0080000.

Transmit Credit Register (TX_CREDIT)

The FPGA co-processor must signal a DMA event to the EDMA to trigger the writing of packets in the transmit direction. This action is ultimately under control of the TMS320C6416 processor, which must write to the TX_CREDIT register within the FPGA co-processor to give the co-processor permission, or credit, to process a packet, for example, once the associated transmit and receive buffers have been allocated in memory. When the FPGA co-processor has one or more transmit credits, it requests a transmit DMA packet whenever the EMIF is idle. The TX_CREDIT register is decremented when a packet is moved from the transmit FIFO buffer to the FFT MegaCore function. The TX_CREDIT register is not directly accessible by the TMS320C6416 processor.

Soft Hardware Reset Register (RESET_HW)

A soft hardware reset is performed prior to initializing the chip support library, the EDMA, and the DMA receive interrupt service routine. This ensures all registers are at a known reset state prior to initialization. The soft hardware reset is performed once within the entire example C routine included with the reference design.

Transmit Credit Increment Register (TX_CREDIT_INC)

Table 6 lists the transmit credit increment register format.

Table 6. Transmit Credit Increment Register Format

Data Bit	Mnemonic	Description
0	S	A write with the **S** bit asserted causes a soft reset of the FPGA co-processor
1	I	A write with the **I** bit asserted causes the `TX_CREDIT` register to be incremented
31:2	0	Always write `0` for future compatibility

Software Access to the FFT Co-Processor

The reference design contains TMS320C6416 example C code that generates the input data stimuli and demonstrates how blocks of data are streamed through the FFT co-processor. The software project is built using the TMS320C6416 processor and BIOS libraries included with the Texas Instruments (TI) Code Composer Studio software, to configure the EDMA controller and interrupts.

Two EP2C35 general-purpose I/O (GPIO) pins are dedicated for use as event triggers for the EDMA:

- One GPIO pin for transmit data (TMS320C6416 processor to the FPGA co-processor)
- One GPIO pin for receive data (FPGA co-processor to EDMA)

The FPGA co-processor requests a new transmit DMA whenever the `TX_CREDIT` register is non-zero and the FFT MegaCore function is ready for more data. The FPGA co-processor requests a receive DMA whenever a packet of data is available from the FFT MegaCore function in the receive FIFO buffer.

Each time a DMA is completed, the EDMA sends an interrupt request to the TMS320C6416 processor. The software tracks the number of packets transmitted and received. When a predefined number of packets are complete, the software calculates the performance of the FFT co-processor.

Apart from a software reset and incrementing the `TX_CREDIT` register, all accesses to the FPGA co-processor are performed by the EDMA controller. In the reference design the EDMA is set up by calling the `initEdma()` function in the **fft_ping_pong.c** source file.

The example software `main()` routine is in the **fft_ping_pong.c** source file included with the reference design. `main()` performs the following tasks:

- Sets up `timer0` for performance measurement
- Initializes the memory buffers with the sine wave data for the EDMA
- Resets the FPGA co-processor and FIFO buffers
- Initializes the TMS320C6416 processor support library
- Calls `initEdma()` to initialize the EDMA controller
- Starts the timer
- Increments `TX_CREDIT`
- Waits until all blocks are processed
- Calculates average time to process one FFT

Each time a transmit or receive DMA interrupt occurs, `edmaHwi()` is called to handle the interrupt. This function is in **fft_ping_pong.c**. Counters are updated to track the number of transmit and receive interrupts received until a predefined limit is reached, at which point the EDMA and interrupts are disabled. The transmit interrupt handler increments the `TX_CREDIT` register to additional further blocks to be processed.

The example software does not perform any additional data processing.

Figure 2 through Figure 5 show various plots of the FFT input and output data using the TI Code Composer Studio software GUI utility.

The sampled sine wave continues for 1,024 samples. The imaginary input samples are all zero.

Figure 2 shows a plot of the real input data in the TI Code Composer Studio software.

Figure 2. TI Code Composer Studio Real Input Data Plot

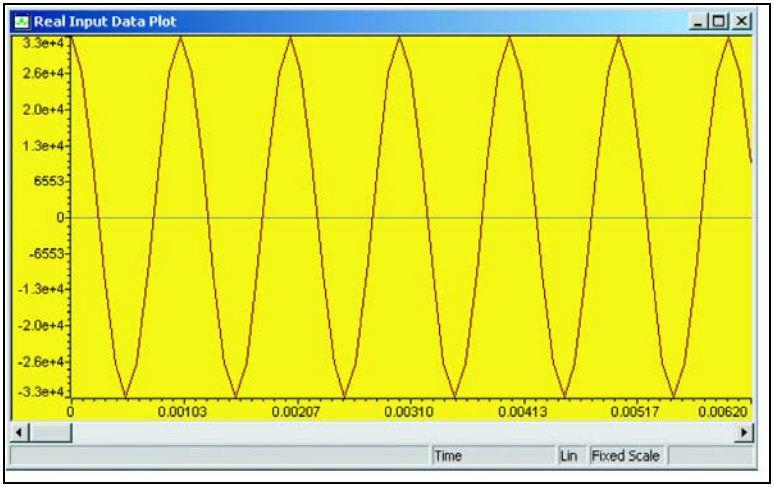


Figure 3 shows a plot of the real output data in the TI Code Composer Studio software.

Figure 3. TI Code Composer Studio Real Output Data Plot

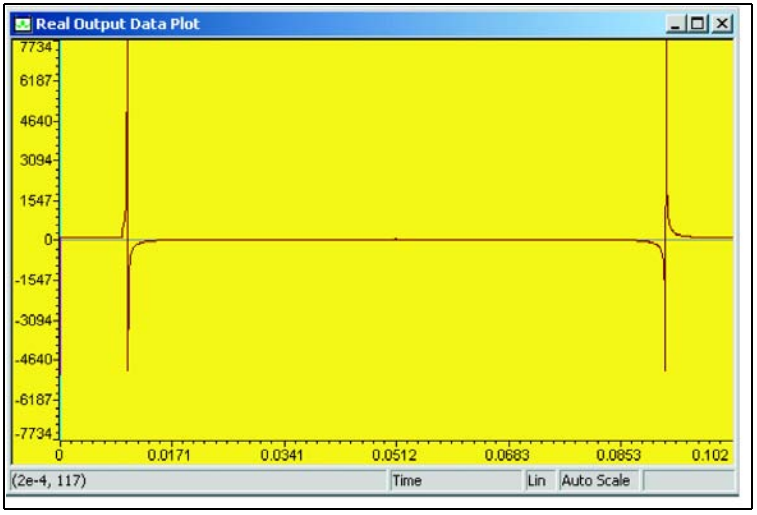


Figure 4 shows a plot of the imaginary output data in the TI Code Composer Studio software.

Figure 4. TI Code Composer Studio Imaginary Output Data Plot

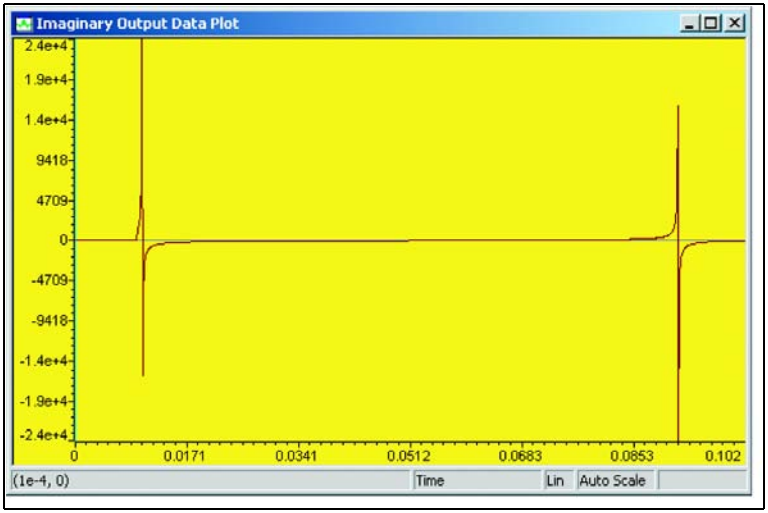
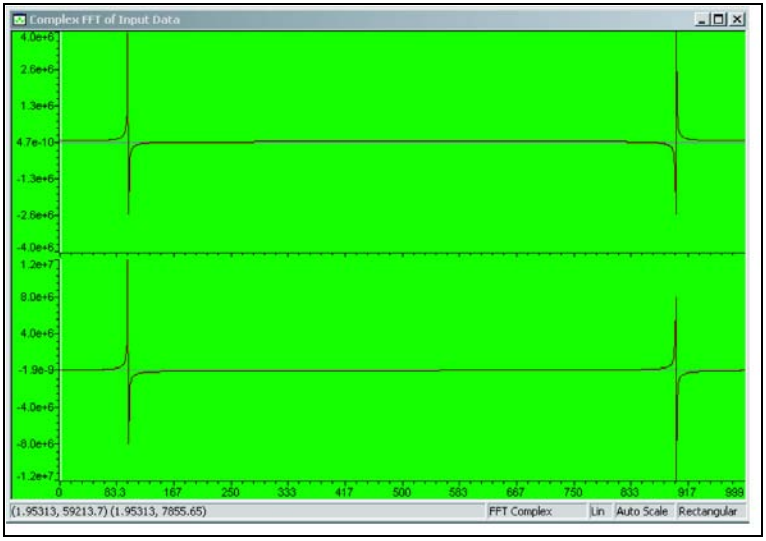


Figure 5 shows the TI Code Composer Studio software plot of a complex FFT applied to the input data implemented entirely in C code.

Figure 5. FFT Plot of the Input Data in the TI Code Composer Studio



Performance Analysis and Design Enhancement

This section discusses performance analysis and suggests improvements that can enhance the performance of an FFT co-processor. Designers can consider several enhancements to the solution presented in this reference design, if the designer is starting a new board-level FPGA co-processor design.

- The designer can integrate the TMS320C6416 processor and the EP2C35 FPGA on the same board. The designer must take care when routing board-level interconnections between the TMS320C6416 processor and the FPGA co-processor to determine if it is possible to leverage the synchronous EMIF maximum data throughput and clock rate to reduce the round-trip data delay time. The current TI C64x family of digital signal processors supports a maximum of 64-bits of EMIF data at clock rates up to 133 MHz.
- The designer can use FIFO buffers in the transmit and receive paths that allow the FFT co-processor function (or any other co-processor function) to run at a different clock rate than the EMIF. If the FFT co-processor function is a performance bottleneck, it can be run at a higher clock rate than the EMIF, decreasing the overall processing time. Alternatively, the co-processor may be reconfigured to exploit further parallelism of the FPGA. If system issues, other than the FFT co-processor function, cause performance bottlenecks, the co-processor clock rate can be reduced, lowering the dynamic power consumption in the FPGA.
- The FFT co-processor function in this reference design is a relatively simple example. For larger co-processing systems that might consist of several co-processor functions, design considerations need to be made to maximize the data processing within the FPGA. The larger co-processing reduces the data-transfer overhead between the digital signal processor and the FPGA relative to the data processing time, maximizing overall system throughput.
- Alternative high-speed interfaces—for example, Serial Rapid I/O, or SerialLite can be considered to improve the high-speed data transfers between the digital signal processor and the FPGA co-processor, increasing overall system performance.

Getting Started

This section describes the software and hardware requirements, as well as the required steps to install the reference design files, program the Cyclone II EP2C35 DSP development board, and load the TMS320C6416 DSK binary executable program.

Software Requirements

The reference design requires the following software:

- Quartus® II software version 5.0
- Altera FFT MegaCore function version 2.1.3
- Texas Instruments Code Composer Studio IDE, version 2.21, which is included in the TMS320C6416 DSK

Hardware Requirements

You must have the following development kits to run the FFT FPGA co-processor reference design:

- *DSP Development Kit, Cyclone II Edition*, version 1.0.0
- Spectrum Digital *DSP Starter Kit (DSK) for the TMS320C6416*, Revision E

Connect each board to their power supplies and programming cables according to the instructions listed in their respective data sheets and reference manuals.

For more information on the *DSP Development Kit, Cyclone II Edition*, refer to the *Cyclone II EP2C35 DSP Development Board Reference Manual*.

For more information on the TMS320C6416 DSK, refer to the *TMS320C6416 DSK Technical Reference* or the *DSP Starter Kit (DSK) for the TMS320C6416 (720 MHz) Quick Start Installation Guide*. Both documents are included in the TMS320C6416 DSK.

Connecting the Cyclone II EP2C35 DSP Development Board to the TMS320C6416 Board

To connect the boards, follow these steps:

1. The Cyclone II EP2C35 DSP development board should be on top of the TMS320C6416 board.

For the Cyclone II EP2C35 DSP development board and the TMS320C6416 board, turn off the power to the board by disconnecting the power cable.

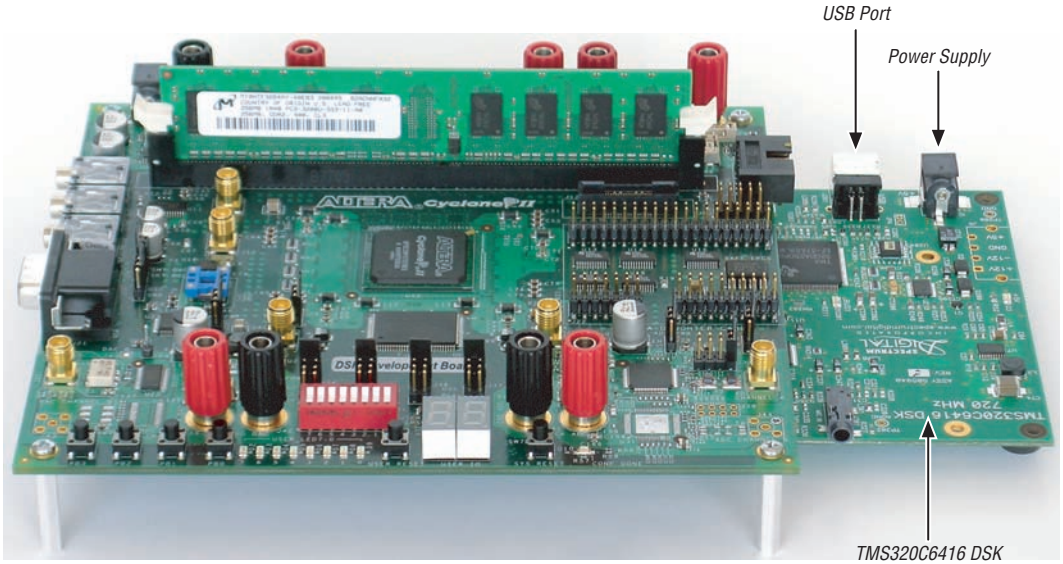
2. Connect U34 (Peripheral Expansion Interface), on the bottom of the Cyclone II EP2C35 DSP development board, to J3 (Peripheral Expansion Connector) on the TMS320C6416 board.

3. Connect U40 (Memory Expansion Interface), on the bottom of the Cyclone II EP2C35 DSP development board, to J4 (Memory Expansion Connector) on the TMS320C6416 DSK board.

CAUTION

Be sure to connect U34 to J3 and U40 to J4. Figure 6 shows the correct positioning of the TMS320C6416 DSK board underneath the Cyclone II EP2C35 DSP development board. If the TMS320C6416 DSK board is connected in the reverse position, it will be severely damaged.

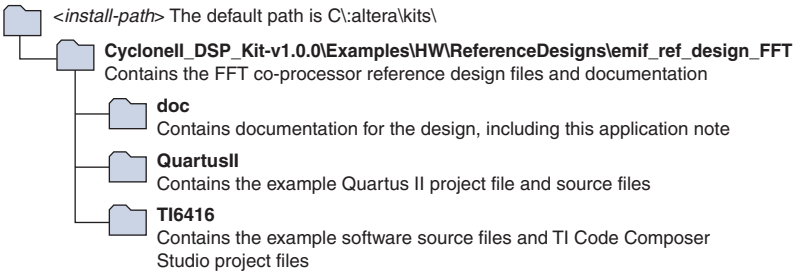
Figure 6. Correct Positioning of the TMS320C6416 DSK Board Underneath the Cyclone II EP2C35 DSP Development Board



Installing the FFT Co-Processor Reference Design Files

When you install the software from the *DSP Development Kit, Cyclone II Edition* CD-ROM, the design files for the FFT co-processor reference design are installed in the directory structure shown in Figure 7.

Figure 7. Cyclone II FFT Co-Processor Reference Design Directory Structure



Configuring the Cyclone II DSP Development Board

The compilation of the FFT co-processor design produces a Sram Object File (**.sof)** that is used to configure the EP2C35 FPGA. The SOF file is included with the reference design in the *DSP Development Kit, Cyclone II Edition*.

To configure the EP2C35 FPGA, follow these steps:

1. Start the Quartus II software.
2. Choose **Open Project** (File menu) and browse to the directory:

   <*install-path*>**\CycloneII_DSP_Kit-v1.0.0\Examples\HW \ReferenceDesigns\emif_ref_design_FFT\QuartusII**
3. Select **emif_ref_ex.qpf** and click **Open**.
4. If a message appears asking whether you want to overwrite the database written with a prior version of the Quartus II software, click **OK**.
5. In the Quartus II software, choose **Programmer** (Tools menu).

   The **emif_ref_ex.sof** is automatically detected by the **emif_ref_ex.cdf** (Chain Description File) and loaded into the Quartus II software.

If the file does not load, click **Add File**, and browse to the directory:

*<install-path>***CycloneII_DSP_Kit-v1.0.0\Examples\HW \ReferenceDesigns\emif_ref_design_FFT\QuartusII**

a. Select **emif_ref_ex.sof** and click **Open** (see Figure 8).

b. Ensure that **USB-Blaster** is selected in the **Quartus II Programmer** window. If it is not, click **Hardware Setup**. Select **USB-Blaster** and click **Select Hardware**. Click **Close** (see Figure 8).

For details on installing the USB-Blaster™ driver on your PC, refer to the *USB-Blaster Download Cable User Guide*. The driver files are installed at *<quartus-install-dir>***drivers\usb-blaster**.

6. Turn on **Program/Configure** (see Figure 8).

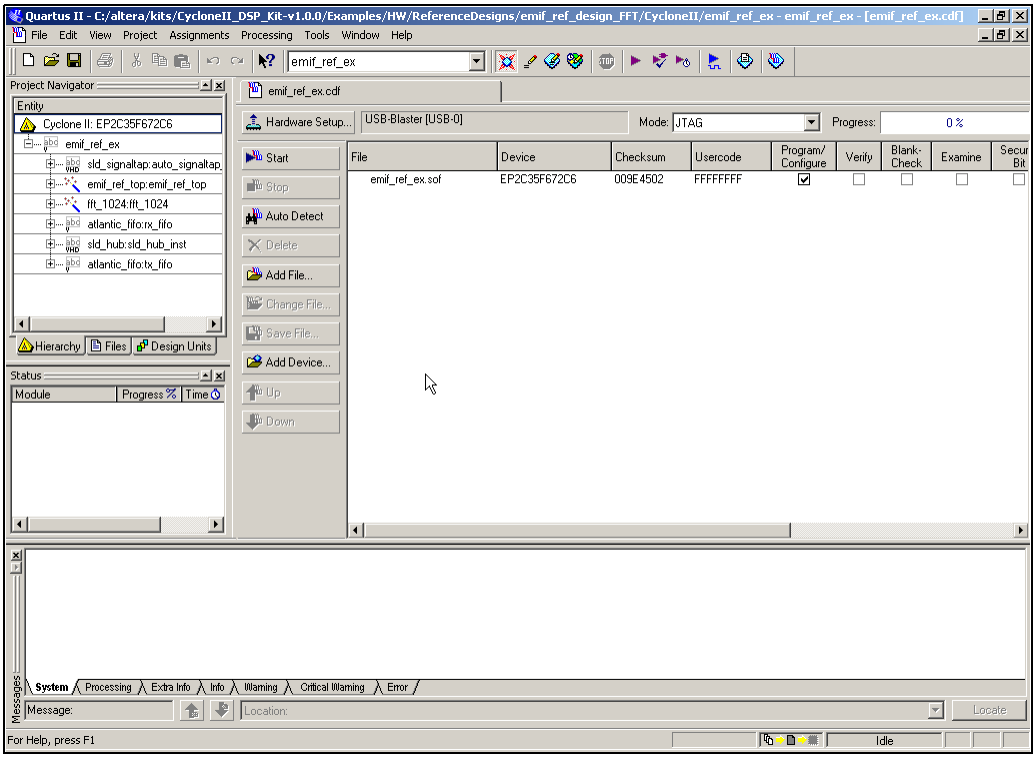
7. Click **Start** to begin programming and configuring the Cyclone II EP2C35 DSP development board (see Figure 8).

8. Programming and configuration is complete when the **Progress** bar reaches 100%. The Quartus II software reports:

```
Info: Ended Programmer operation at <date, time>
```

9. The CONF DONE LED (D10) turns on, indicating successful completion of the configuration.

Figure 8. Quartus II Programmer Dialog Box



Running the TI Code Composer Studio Example Software

To run the example software in the TI Code Composer Studio software, follow these steps:

1. Ensure that the USB cable is connected between the TMS320C6416 DSK board and your PC. The USB port on the TMS320C6416 DSK board is shown in Figure 6 on page 16.

2. Start **Code Composer Studio**. This opens the The TI Code Composer Studio software window. (see Figure 9 on page 22).

3. Ensure that the correct GEL file is loaded for the TMS320C6416 DSK under GEL files in the TI Code Composer Studio software Project window. To load the appropriate GEL file, right click on **GEL files** under **Files** in the left pane, and select **Load GEL**. Browse to the directory:

   <*CCS-install-path*>**\cc\gel**

4. Select **DSK6416.gel** and click **Open**.

5. Choose **Project > Open** (Project menu) and browse to the directory:

   <*install-path*>**\CycloneII_DSP_Kit-v1.0.0\Examples\HW \ReferenceDesigns\emif_ref_design_FFT\TI6416**

6. Select **fft_ping_pong.pjt** and click **Open**.

7. If a message appears saying the **csl6416.lib** file cannot be found, click **Browse** and browse to the directory:

   <*CCS-install-path*>**\c6000\bios\lib**

8. Select **csl6416.lib** and click **Open**.

9. Choose **Workspace** > **Load Workspace** (File menu) and open **fft_ping_pong.wks**. This opens a saved TI Code Composer Studio software workspace that allows you to view the FFT input and output data plots in the TI Code Composer Studio software user interface.

   Ignore any TI Code Composer Studio software messages that refer to insufficient resources or GEL file location.

10. Choose **Resets > ClearBreakPts_Reset_EMIFset** (GEL menu).

    Ignore any TI Code Composer Studio software messages that refer to start address identifiers.

11. Choose **Memory Map > SetMemoryMap** (GEL menu).

12. To load the design onto the TMS320C6416 DSK board, choose **Load Program** (File menu) and browse to the **debug** directory.

13. Select **fft_ping_pong.out** and click **Open**.

14. Choose **Enable Clock** (Profiler menu) so that the profiler can keep count of the number of clock cycles it took to run the FFT function.

You may want to choose **Enable Clock** again in the profiler menu to ensure that **Enable Clock** is turned on.

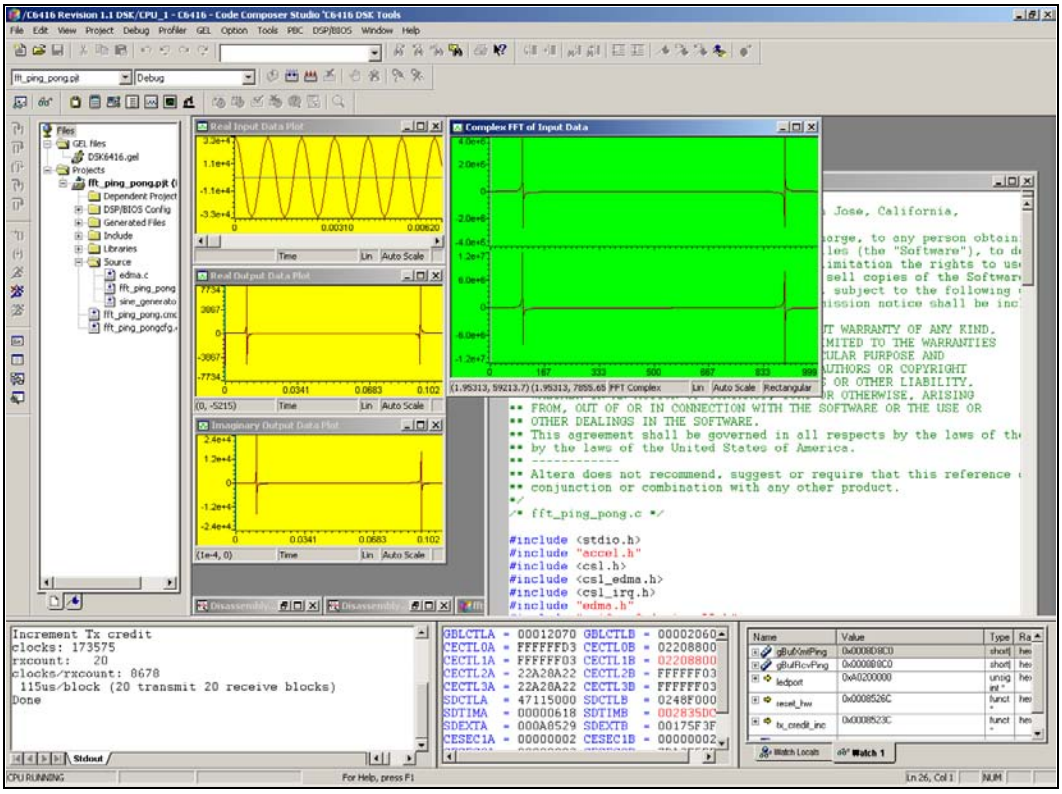
15. Choose **Run** (Debug menu) to run the software on the TMS320C6416 DSK board.

16. As the design runs, messages appear in the Stdout window. The messages indicate the status of the program and the number of clock cycles used to run the FFT co-processor function.

17. The program prints **Done** in the Stdout window as the last message, indicating that the program successfully executed.

18. The graphs in the TI Code Composer Studio software user interface show the various input and output FFT plots (see Figure 9). If the signals do not look correct, right-click in the respective graph windows and click **Refresh** to update the signals.

Figure 9. TI Code Composer Studio Workspace Graphs & User Interface



Compiling the FFT Co-Processor Reference Design (Optional)

To compile the FFT co-processor reference design, you must first install the FFT MegaCore function. The function can be run with a full license or using free OpenCore Plus evaluation.

For more information on OpenCore Plus, refer to *AN 320: OpenCore Plus Evaluation of Megafunctions*.

The following directory contains the Quartus II project file, **emif_ref_ex.qpf**, which contains the project files and the pin mappings for the *DSP Development Kit, Cyclone II Edition*:

*<install-path>***\CycloneII_DSP_Kit-v1.0.0\Examples\HW\ReferenceDesigns\emif_ref_design_FFT\QuartusII**

To compile the FFT co-processor reference design, follow these steps:

1. Start the Quartus II software.

2. Choose **Open Project** (File menu) and browse to the directory:

   <*install-path*>**\CycloneII_DSP_Kit-v1.0.0\Examples\HW \ReferenceDesigns\emif_ref_design_FFT\QuartusII**

3. Select **emif_ref_ex.qpf** and click **Open**.

4. If a message appears asking whether you want to overwrite the database written with a prior version of the Quartus II software, click **OK**.

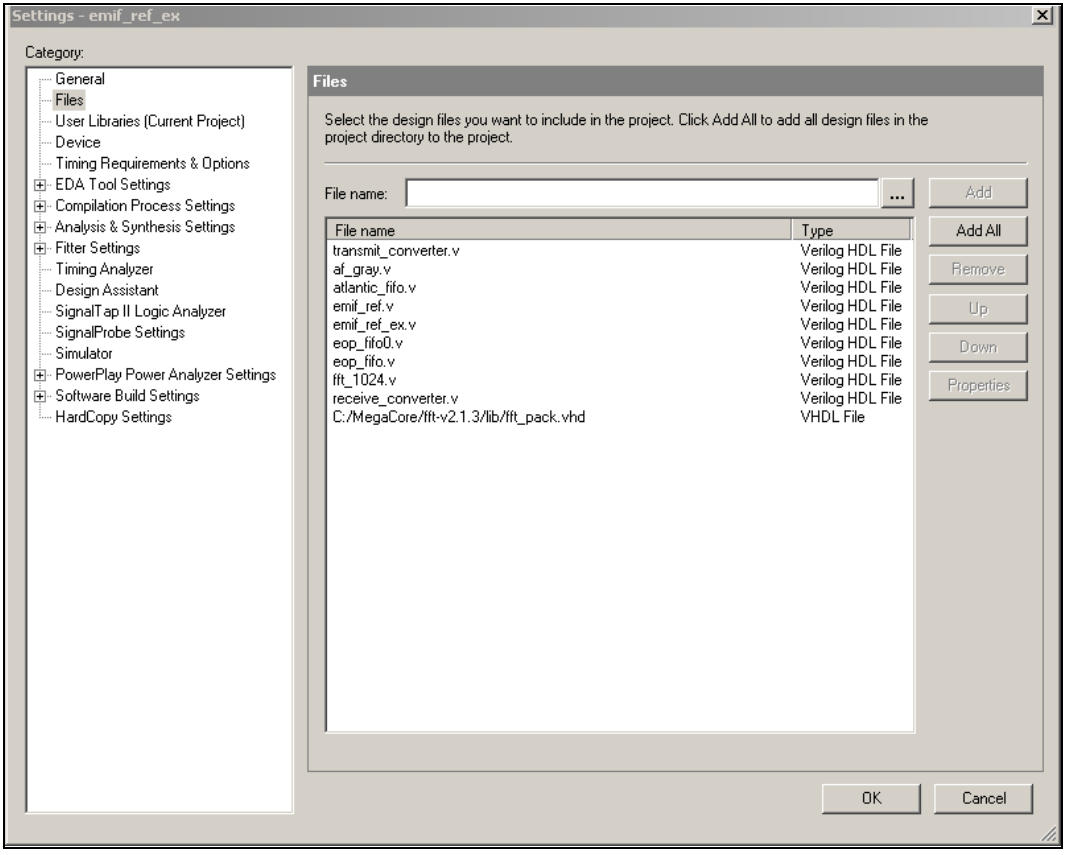
5. Choose **Add/Remove Files in Project** (Project menu) and verify that the **fft_pack.vhd** file (for the FFT MegaCore function) is located in the path listed in the display (see Figure 10).

6. If the path for the **fft_pack.vhd** file is incorrect, remove the **fft_pack.vhd** file by selecting it and click **Remove**. To add the **fft_pack.vhd** file from the directory where it is installed, in the **File name** list, browse to the directory:

   <*install_path*>**\MegaCore\fft-v2.1.3\lib**

7. Select **fft_pack.vhd** and click **Add** (see Figure 10).

Figure 10. Verifying the fft_pack.vhd File in the Settings Dialog Box



8. Choose **Start Compilation** (Processing menu) to begin compiling the design.

9. Click **OK** in the message window that appears.

Conclusion

The Cyclone II FFT co-processor reference design provides designers the flexibility to implement a design that easily interfaces the Cyclone II FPGA to a wide range of digital signal processors. Designers can adapt the FFT co-processor model to fit virtually any target application because of the programmable nature of the FPGA's fabric.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com
Applications Hotline:
(800) 800-EPLD
Literature Services:
lit_req@altera.com