Turbo Encoder Co-processor Reference Design

AN-317-1.2 Application Note

Introduction

The turbo encoder co-processor reference design is for implemention in an Stratix™ DSP development board that is connected to a Texas Instruments C6711 DSP Starter Kit (DSK). The DSK has a 32-bit external memory interface (EMIF) and a 16-channel enhanced DMA (EDMA) controller. The design of the DSK limits the use of the EMIF to asynchronous mode only.

The Stratix DSP development board is part of the Altera® DSP Development Kit, Stratix Edition.

The reference design connects the DSK processor's EMIF to Altera's Turbo Encoder MegaCore funtion. The reference design includes an Avalon™ interface (master), which allows you to connect other functions. The Avalon bus is a simple bus architecture designed for connecting processors and peripherals together into a system-on-a-programmable chip (SOPC). The Avalon bus is an interface that specifies the port connections between master and slave components, and specifies the timing by which these components communicate.

For more information on installation and licensing of the Turbo Encoder/Decoder MegaCore function, refer to the *Turbo Encoder/Decoder MegaCore Function User Guide*.

For more information on the Avalon bus, refer to the *Avalon Bus Specification Reference Manual*.

Altera supplies the reference design as Verilog HDL source code. The reference design includes a testbench that allows you to test the Verilog HDL source code.

The purpose of this reference design is to demonstrate that Altera Stratix and Cyclone™ devices are suitable in performance and capacity to implement digital signal processing (DSP) functions as co-processors.

Background

A turbo encoder consists of two recursive convolutional encoders and an internal interleaver. While the convolutional encoders are simple to implement in either hardware or software, the interleaver tends to be complex due to its variability. Any block size from 40 to 5,114 must be supported, and the block size can vary every transmission time interval of 2 ms. This is a significant computational burden for a DSP and adds to the latency, which is a critical parameter in high-speed downlink packet access (HSDPA).

An alternative to using a DSP function to perform this function is to download blocks of data to a turbo encoder accelerator function implemented on a FPGA. This accelerator function removes the need to calculate the lookup table content for the interleaver, and also takes the highly repetitive encoding task off the DSP function, freeing up bandwidth for the other operations the DSP function has to perform.

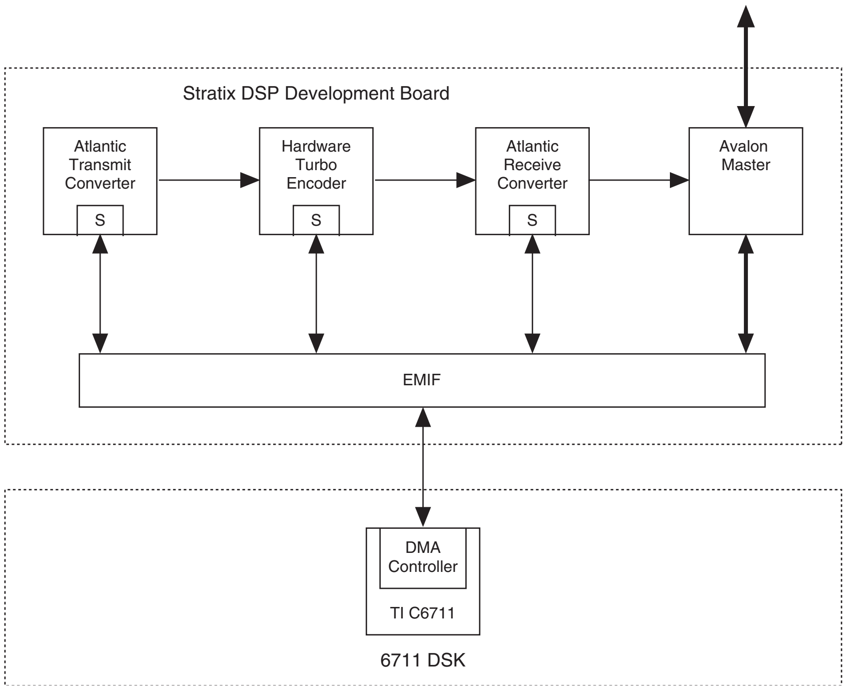
Altera FPGA co-processors interface with a wide range of DSP functions and general purpose processors providing increased system performance and lower system costs.

Functional Description

Figure 1 shows the reference design block diagram. It shows the connection between the C6711 DSK and the Stratix DSP development board and how the turbo encoder is connected by the Atlantic™ interfaces. The Atlantic interface allows you to insert other hardware modules in place of or in series with the Turbo Encoder. The Avalon master interface allows you to connect other functions through Avalon slaves. The reference design includes a Quartus II project that connects an Avalon parallel I/O (PIO) module driving the 7-segment displays on the Stratix DSP development board.

For more information on the Atlantic interface, refer to the *Atlantic Interface Functional Specification*.

Figure 1. Block Diagram Note (1)


Stratix DSP Development Board
Atlantic Transmit Converter
S
Hardware Turbo Encoder
S
Atlantic Receive Converter
S
Avalon Master
EMIF
DMA Controller
TI C6711
6711 DSK

Note:
(1) The S denotes a slave interface.

The DMA controller transmits packets of data to be encoded to the hardware turbo encoder via the EMIF and an Atlantic transmit converter. The Atlantic transmit converter signals the DMA controller whenever it can accept another packet of data.

The output of the hardware accelerator is connected to the Atlantic receive converter, which is signals to the DMA controller whenever it has a packet of encoded data available to be read via EMIF.

The Atlantic converters and the hardware Turbo Encoder have a slave interface decoded in the EMIF memory space (see the 'S' boxes in Figure 1 on page 3) for communication of control and status.

To maximize the performance of the hardware acceleration it is essential to be able to pipeline data through the system. To allow this we define a number of slots, *n*. The EMIF decodes the EMIF address to allow access to *n* virtual slaves in the Atlantic transmit converter, hardware accelerator, and the Atlantic receive converter. For the turbo encoder co-processor, the number of virtual slaves available, *n*, is 16.

The Atlantic transmit converter has 16 virtual slave interfaces (decoded from the EMIF DMA address) through which data can be written corresponding to each slot. If data is written by the DMA controller to slot *s*, Atlantic address *s* is used when the data is transferred to the hardware turbo encoder. The hardware turbo encoder has 16 sets of control registers within its slave interface. The Atlantic address associated with input data for a packet determines which set of control registers are used to process that packet.

As an example, assume that the encoder is busy processing data for slot 0. The DSP function can write new control information for slot 1 (i.e., the turbo encoder block size) without affecting the current processing. There is no requirement to wait for the hardware turbo encoder to complete the processing of a packet of data before writing new control information, unless the new control information relates to the slot currently being processed.

Figure 2 shows the reference design interface.

Figure 2. Interface

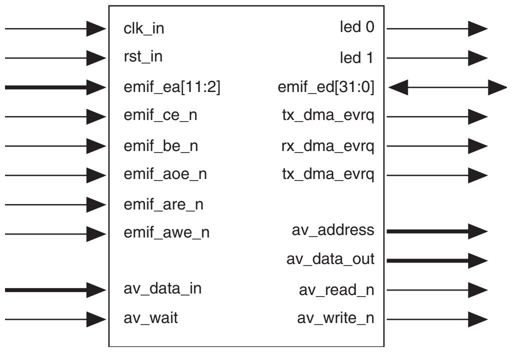


Table 1 shows the the turbo encoder co-processor parameters.

Table 1. Parameters

Parameter	Range	Description
`BANKSWAP`	0 or 1	The value of this parameter is passed to the Turbo Encoder MegaCore® function. 0 = single buffered memory, 1 = double buffered memory in the function.
`DEVICE`	Stratix or Cyclone	The reference design uses this parameter to synthesize memory blocks for the correct device family,

Table 1 shows the turbo encoder co-processor signals.

Table 2. Signals

Name	Width	Direction	Description
`clk_in`	1	I	Clock
`rst_n`	1	I	Asynchronous Reset, active low
`emif_ea[11:2]`	10	I	EMIF Address
`emif_ce_n`	1	I	EMIF Chip Enable. The chip select must be set up as asynchronous
`emif_be_n`	1	I	EMIF Byte Enable
`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_ed[31:0]`	32	I/O	EMIF Data
`tx_dma_evrq`	1	O	Transmit DMA Event Request. Asserted high to request a new block of data to be encoded.
`rx_dma_evrq`	1	O	Receive DMA Event Request. Asserted high to signal that a block of encoded data is available
`av_address`	22	O	Avalon address bus.
`av_data_in`	32	I	Avalon read data input.
`av_data_out`	32	O	Avalon write data output.
`av_read_n`	1	O	Avalon read strobe, active low.
`av_write_n`	1	O	Avalon, write strobe, active low.
`av_wait`	1	I	Avalon wait request.

Transmit Converter

One side of the transmit converter presents an EMIF slave interface and a DMA event request signal (which requests more data from the DMA controller). The other side of the converter presents an Atlantic master source for data transfer to the hardware turbo encoder.

The EMIF slave occupies 128 bytes of EMIF address and uses the high order EMIF address lines to provide the Atlantic `ADDR` signals to convey the slot number.

Receive Converter

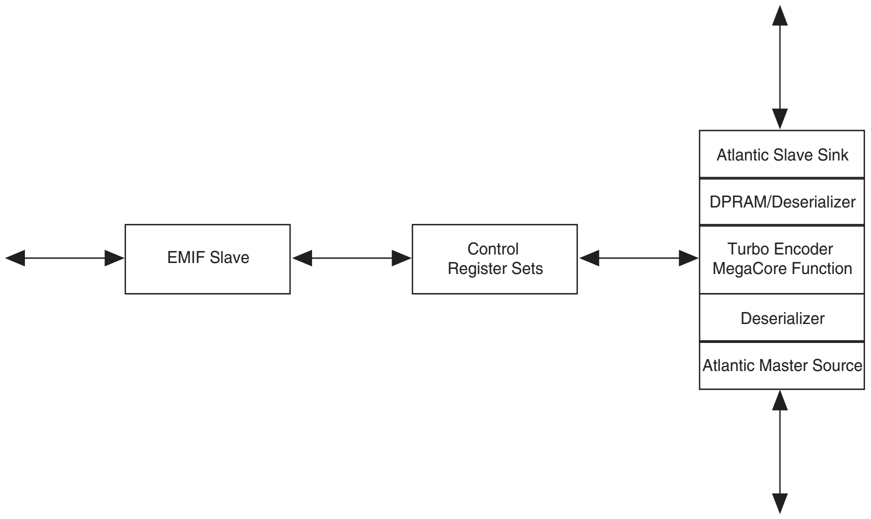
The receive converter accepts data from the Atlantic master in the hardware turbo encoder, through an Atlantic slave sink of the same width as the EMIF slave, and buffers it in a 4 Kb FIFO buffer. The other side of the converter presents an EMIF slave interface and a DMA event request. The DMA event request is asserted when end-of-packet (EOP) is signalled on the Atlantic, indicating the end of the encoded block. The whole encoded block is, thus, available in the FIFO buffer. The system must begin reading the encoded block before the last bit of the next block is available, otherwise a DMA request may be missed. Also the system must read the encoded block fast enough that the FIFO buffer does not overflow as the next encoded block becomes available.

Hardware Turbo Encoder

The hardware turbo encoder comprises the Turbo Encoder MegaCore function with a design file to connect the MegaCore funciton's serial data I/O to the Atlantic interface and its control ports to the control registers in the EMIF slave.

Figure 3 shows the block diagram of the hardware turbo encoder.

Figure 3. Hardware Turbo Encoder



Atlantic data is buffered in a 128 byte dual-port RAM. The read side of the DPRAM is only one-bit wide, so that the data is serialized as it is read out and fed into the serial input of the turbo encoder core.

Control information is stored in a further DPRAM indexed by slot number.

A small FIFO buffer is used to store the Atlantic ADDR (slot no.) at the start of each packet. When the turbo encoder core starts processing that packet the ADDR value from the FIFO buffer is used as the index into the control RAM to get the block size to be used for that packet. If the newly selected block size is different to that previously used then the Turbo coder core is re-initialised. Atlantic data flow is paused during the re-initialisation time.

The serial output of the turbo encoder core is formatted into parallel words for onward transmission through the Atlantic master. The data and two parity bits are packed into nibbles with one unused bit (8 symbols/32 bit word).

There is no FIFO buffer on the output of the hardware accelerator since the Atlantic receive converter has a FIFO buffer.

All control registers are 32 bits wide, and the physical location of parameters that share a single register are fixed, regardless of endian mode. Control registers should always be accessed as 32-bit words.

Avalon Master Interface

You can use the Avalon master interface to connect other user functions to the EMIF via the Avalon slaves. The master port appears as an interface component in SOPC Builder, which you can use to generate a system comprising the turbo encoder co-processor and one or more Avalon slaves. The Avalon master supports peripheral generated wait states.

The Avalon master interface shares the same EMIF chip select as the EMIF to Atlantic converters. It is selected whenever EMIF address 21 is high during an EMIF read or write.

Registers

Byte and half-word accesses to the Atlantic converter registers have undefined results for some registers (there is no notion of a bus error on EMIF and thus, no way to terminate unexpected accesses).

The Atlantic interface is always big-endian. The symbol size for the Atlantic converters is fixed at 8 bits. Symbol swapping would be required between the EMIF and the Atlantic interface to cope with a DSP function running in little endian mode.

Table 3 shows the registers.

Table 3. Registers

Mnemonic	Address	Access	Description
`TXCONV_DATA`	0h	Write	Transmit converter data register.
`TXCONV_EOP`	40h to 78h	Write	Transmit converter end of packet register.
`RXCONV_DATA`	80h	Read	Receive converter data register.
`RXCONV_DATA`	84h	Read	DMA event status register.
`CONTROL`	C0h, 1C0h, FC0h	Write	Control register.

Transmit Converter Data Register (TXCONV_DATA)

Table 4 shows the transmit control data register.

Table 4. Transmit Converter Data Register (TXCONV_DATA)

Data Bit	Mnemonic	Definition
31:0	`DATA`	32 bits of data to be written to the hardware turbo encoder.

A write to this register strobes the Atlantic ENA signal for one cycle. During this cycle (and subsequent cycles until ENA is 1 again) the Atlantic DATA lines drive the value written to the register. The Atlantic EOP, ERR and MTY (if present) drive 0. The Atlantic ADDR lines drive the value present on A[11:8]

The DMA controller performs non-incrementing writes to this register to write all data values in the packet except the last one.

Transmit Converter End of Packet Register (TXCONV_EOP)

Table 5 shows the transmit converter end of packet register.

Table 5. Transmit Converter End of Packet Register (TXCONV_EOP)

Data Bit	Mnemonic	Definition
31:0	DATA	32 bits of data to be written to the hardware turbo encoder.

A write to this register strobe the Atlantic ENA signal for one cycle. During this cycle (and subsequent cycles until ENA is 1 again) the Atlantic DATA lines drive the value written to the register. The Atlantic EOP drives 1; ERR drives 0; MTY drives a value generated from A[3:2]. The Atlantic ADDR drives the value present on A[11:8].

The DMA controller writes the last word, or partial word, to one of the instances of this register. The address used depends on the number of valid bytes within that word. If MTY = 0, the position of the data transferred (most or least significant bits) depends upon whether the symbol swap was selected.

Receive Converter Data Register (RXCONV_DATA)

Table 6 shows the transmit converter end of packet register.

Table 6. Receive Converter Data Register (RXCONV_DATA|)

Data Bit	Mnemonic	Definition
31:0	DATA	One word of data read from the Atlantic interface .

This register returns a data value read from the Atlantic interface. If this word is the last in the packet, the some of the symbols may be undefined. Whether these are in the most significant or least significant bit positions depends upon whether symbol swapping was selected.

The DMA controller should be programmed to read the correct amount of data for a packet to avoid reading data beyond an end-of-packet on the Atlantic interface.

DMA Event Status Register (RXCONV_DATA)

Table 7 shows the DMA event status register.

Table 7. DMA Event Status Register (RXCONV_DATA)

Data Bit	Mnemonic	Definition
31:2	0	Always read 0.
1	`Rx`	State of receive DMA event request (active high).
0	`Tx`	State of transmit DMA event request (active high).

Control Register (CONTROL)

Table 8 shows the control register.

Table 8. Control Register (CONTROL)

Data Bit	Mnemonic	Definition
31:13	0	Always write 0 for future compatibility.
12:0	`BSIZE`	Block size (40 to 5,114 bits).

The hardware turbo encoder has one control register for each of 16 slots. During operation the channels are selected by the Atlantic `ADDR` as described previously. Bits [11:8] of the register address correspond to the slot number.

Software interface

Software, written in C, provides an API which allows data to be passed to and from the accelerator in an efficient way. The interface is asynchronous—user provided functions are called when data has been processed.

The header file fragments below show the interface to this code:

```
typedef void (* TXCALLBACK)(
    void * handle);
```

```
typedef void (* RXCALLBACK)(
    uchar * output,
    void * handle);

void turbo_encode(
    const uchar * input,
    uchar * output,
    uint bits,
    TXCALLBACK txcallback,
    RXCALLBACK rxcallback,
    void * handle);

void turbo_register_interrupt(
    uint intnum;

void turbo_poll(void);
```

To encode a block of data user code calls the turbo_encode() function, passing in a block of data to be encoded and a buffer into which the results should be placed. The function queues the block of data and the buffer into which the data should be received and then returns. The user code must not modify the data buffer or the output buffer while they are owned by the accelerator.

When the data has been sent to the accelerator the txcallback function will be called. This informs the user code that it is safe to reuse the input buffer (containing the data passed into the accelerator).

When a result is available, the rxcallback function will be called. This informs the user code that the output buffer now contains valid data which can be processed further.

☞ The `txcallback` and `rxcallback` functions could be called in either order – if both callbacks occur at once then the driver code calls rxcallback first to reduce latency.

The driver can operate in a polled or interrupt driven fashion: for polled operation the turbo_poll function must be called regularly while there are packets pending; for interrupt driven operation the turbo_register_interrupt function must be called before the first packet is submitted.

Testbench

The testbench sends a number of small packets through the co-processor by simulating EMIF read and write cycles. The transmit and receive DMA requests are used to synchronize the packet reads and writes.

Getting Started

This section involves the following steps:

1. "Software Requirements".
2. "Hardware Requirements".
3. "Install the Design".
4. "Generate the System".
5. "Synthesize the Design".
6. "Simulate the Design".
7. "Run the Example Software".

☞ You must synthesize the design to generate the Quartus II Verilog netlist, before you simulate the design.

Software Requirements

The reference design requires the following software:

- Quartus® II software version 3.0, or higher
- Altera Turbo Encoder/Decoder MegaCore function v1.4.0
- ModelSim version 5.6e
- Code Compose Studio version 2.10 or higher (part of the C6711 DSK)

To use the turbo encoder co-processor reference design, you must first install and obtain a license for the Altera Turbo Encoder MegaCore function, which allows the turbo encoder (`aukte_umts_turbo_encoder`) to be instantiated from the turbo encoder co-processor hierarchy (see Figure 5).

For more information on installation and licensing of the Turbo Encoder/Decoder MegaCore function, refer to the *Turbo Encoder/Decoder MegaCore Function User Guide*.

Hardware Requirements

To run the example software, you must have the Stratix DSP development board plugged into the EMIF connector of the C6711 DSK. The C6711 DSK must be connected to your PC according to the C6711 DSK instructions. You must also have a download cable to program the FPGA on the Stratix development board.

The Stratix development board requires the following four wires to be connected between the prototyping area and the EMIF connectors:

- J21/1 to J11/59 (EMIF `RESET`)
- J22/1 to J11/67 (EMIF `EXT_INT6`)
- J23/7 to J11/68 (EMIF `EXT_INT7`)
- J24/1 to J12/77 (EMIF `CE_3`)

☞ Where J*x*/*y* denotes pin *y* of connector J*x* on the Stratix development board. J21 to J24 are in the prototyping area; J11 and J12 are the EMIF connector.

Install the Design

To install the reference design, run the **.exe** and follow the installation instructions. Figure 4 shows the directory structure.

Figure 4. Directory Structure

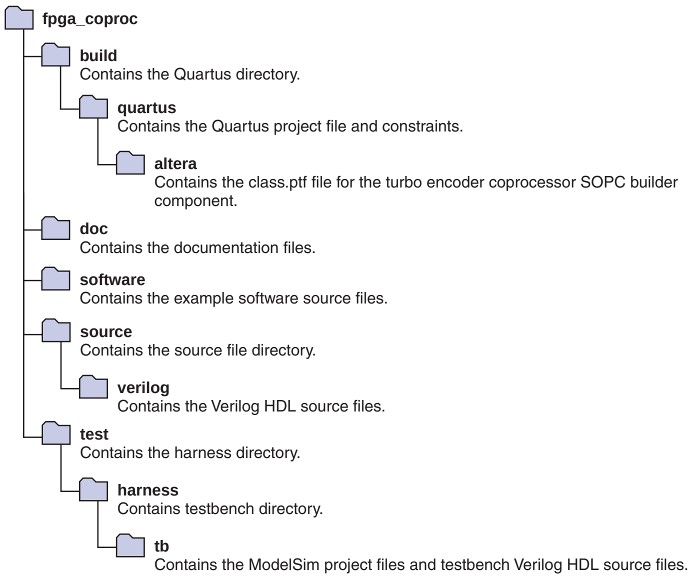

fpga_coproc
build
Contains the Quartus directory.
quartus
Contains the Quartus project file and constraints.
altera
Contains the class.ptf file for the turbo encoder coprocessor SOPC builder component.
doc
Contains the documentation files.
software
Contains the example software source files.
source
Contains the source file directory.
verilog
Contains the Verilog HDL source files.
test
Contains the harness directory.
harness
Contains testbench directory.
tb
Contains the ModelSim project files and testbench Verilog HDL source files.

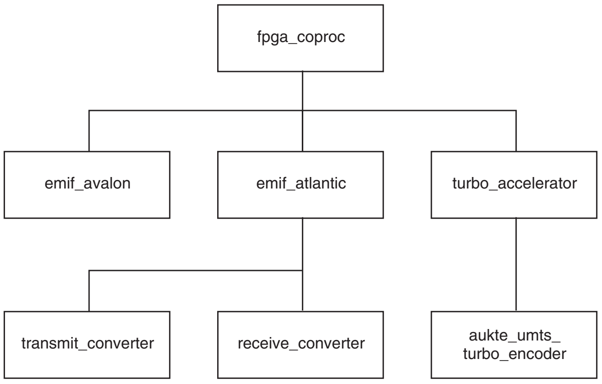
Table 9 shows the Verilog HDL source files. in the **\source\verilog** directory.

Table 9. Verilog HDL Source Files

File name	Description
fpga_coproc.v	The top-level module, fpga_coproc.
emif_atlantic.v	The top-level EMIF to Atlantic interface module, emif_atlantic.
turbo_accelerator.v	The design file for the Turbo Encoder MegaCore function, turbo_accelerator.
transmit_converter.v	The Atlantic transmit converter module, transmit_converter.
receive_converter.v	The Atlantic receive converter module, receive_converter.
emif_avalon.v	EMIF to Avalon master interface.

Figure 5 shows the module heirarchy.

Figure 5. Module Heirarchy



Generate the System

The **build/quartus** directory contains a Quartus II 3.0 project. The constraint file **hwaccel.csf** and **hwaccel.csf** contain the correct pin mapping for the Stratix device on the Stratix DSP board when connected to the TI DSK, and other project settings. The project is configured to generate a Verilog simulation netlist in the **build/quartus/simulation/modelsim** directory.

You can change to VHDL by choosing **EDA Tools Settings** (Assignments menu) in the Quartus II software.

You may skip "Generate the System"and "Synthesize the Design" by using the precompiled **.sof** file (**build/quartus** directory) and the Verilog HDL netlists (**build/quartus/simulation/modelsim** directory). The **.sof** file and the Verilog HDL netlists are overwritten every time you compile the design.

The design passes the `BANKSWAP` parameter to the Turbo Encoder MegaCore function, to select the amount of buffer memory to be used.

For more information on `BANKSWAP`, refer to the *Turbo Encoder/Decoder MegaCore function User Guide*.

To generate the system, perform the following steps:

1. Start the Quartus II software, **Programs** > **Altera** > **Quartus II** (Windows Start menu).

2. Choose **Open Project** (File menu) and browse to the **build/quartus** directory.

3. Choose **fpga_coproc_chip.quartus** and click **Open**.

4. Choose **SOPC Builder** (Tools menu). SOPC Builder shows the turbo encoder co-processor connected to an Avalon PIO module.

5. Click the **System Generation** tab and click **Generate**.

Synthesize the Design

To synthesize the design, perform the following steps:

1. Edit the `BANKSWAP` parameter in **fpga_coproc.v** to your desired value.

2. To target a Cyclone device, you must change the value of the `DEVICE` parameter.

   a. Edit the source file **fpga_coproc.v** to comment out the line `"parameter DEVICE = "Stratix";"`.

   b. Uncomment the line "`parameter DEVICE = "Cyclone";`".

3. Ensure that you specify the MegaCore function’s library folder (**\turbo_codec\lib**) as a user library in the Quartus II software. Search for “User Libraries” in Quartus Help for instructions on how to add a library.

4. Choose **Start Compilation** (Processing menu).

The Quartus II software generates a Verilog HDL netlist in the **build/quartus/simulation/modelsim** directory that you can use to simulate the design.

Simulate the Design

The **test/harness/tb** directory contains a ModelSim project. A ModelSim **.do** file, **fpga_coproc_stratix.do**, contains all the commands necessary to compile and run the turbo encoder co-processor testbench.

☞ The **.do** file may need editing to specify the correct installation directories and versions for the Turbo Encoder MegaCore function and the Quartus II simulation libraries.

The **.do** file performs the following steps:

1. Refreshes the Turbo Encoder MegaCore simulation libraries (may not be required depending upon the version of the ModelSim simulator).

2. Compiles the testbench and source files.

3. Compiles the Verilog HDL simulation netlist output from Quartus.

3. Compiles the Quartus II simulation library.

4. Opens the waveform window.

5. Runs the simulation.

☞ If your ModelSim license supports mixed language simualtion, you can use **fpga_coproc_real.do** to simulate the source files instead of the Quartus II output netlist.

The testbench passes a number of small blocks of data through the turbo encoder co-processor. You may change the number and sizes of these blocks to investigate the behaviour of the turbo encoder co-processor.

To run the **.do** file, perform the following steps:

1. Start the ModelSim simulator.

2. Choose **Open** > **Project** (File menu) and browse to the **fpga_coproc\test\harness\tb** directory.

3. Choose **fpga_coproc.mpf** and click **Open**.

4. Choose **Execute Macro** (Tools menu).

5. Choose **fpga_coproc_stratix.do**, or **fpga_coproc_cyclone.do** and click **Open**.

Run the Example Software

To run the example software, perform the following steps:

☞ You can skip steps 3 to 5, by using the precompiled object file **accel.out** in the **software\debug** directory.

1. Ensure the C6711 DSK is connected and powered up.

2. Start Code Composer Studio.

3. Choose **Workspace** > **Load Workspace** (File menu).

4. Browse to the **\software** directory and open **workspace.wks**.

5. Choose **Build** (Project menu), to compile the project.

6. Choose Load Program (File menu).

7. Browse to the **\software\debug** directory and open **accel.out**, to load the software into the C6711 DSK.

8. Choose **Run** (Debug menu), to run the software on the C6711 DSK.

The software runs the turbo encoder co-processor with the following block sizes:

- 40 bits
- 80 bits
- 160 bits
- 320 bits
- 640 bits
- 1280 bits
- 2560 bits
- 5114 bits

The results are printed on `stdout` in the following form:

```
    N bits: Tµs/iteration - 3630 ns/bit (A B iterations)
```

where:

N is the number of bits
T is the time per iteration, which includes the overhead of setting up the DMA transfers—a fixed overhead (more or less). As the block size increases, its effect decreases and the time per bit improves.
A and *B* show the number of iterations.
A is the number of blocks transmitted by the DMA controller.
B is the number of encoded blocks received by the DMA controller.

The test runs for a fixed time, so *A* and *B* are not equal if one or more blocks are being encoded when the test finishes.

Performance

Table 10 shows the performance for the default configuration of the reference design with various devices, using Quartus II version 3.0 with push-button synthesis. The performance results include the Turbo MegaCore function with `BANKSWAP` = 1, and also include the Avalon PIO.

Table 10. Performance

Device	Logic Elements (LEs)	Memory(Bits)	Frequency (MHz)
Stratix EP1S25F780C5	2,735	34,896	106
Cyclone EP1C3T144C8	2,544	34,896	106

☞ The Turbo encoder co-processor reference design has not been hardware tested in Cyclone devices.

Summary

Congratulations! You have successfully used the turbo encoder co-processor reference design, to demonstrate that Altera Stratix and Cyclone devices are suitable in performance and capacity to implement digital signal processing (DSP) functions as co-processors.

The design is typically used to transfer large blocks of data at high speed and works particularly well with high-speed downlink packet access (HSDPA) applications.

The following strategies are two of many that are available to further develop the software for a particular application:

- Assembler code the DMA routines on the DSP to improve performance
- Implement programed transfers of data for smaller block sizes to avoid DMA setup overhead

Also, the use of a synchronous EMIF interface (not possible with the C6711 DSK) may improve the performance of data transfers between the DSP and the co-processor.