



**Symbol Interleaver/De-Interleaver
MegaCore Function User Guide
September 1999**

ACCESS, Altera, AMPP, APEX, APEX 20K, Atlas, FLEX, FLEX 10K, FLEX 10KA, FLEX 10KE, FLEX 6000, FLEX 6000A, MAX, MAX+PLUS, MAX+PLUS II, MegaCore, MultiCore, MultiVolt, NativeLink, OpenCore, Quartus, System-on-a-Programmable-Chip, and specific device designations are trademarks and/or service marks of Altera Corporation in the United States and other countries. Product design elements and mnemonics used by Altera Corporation are protected by copyright and/or trademark laws.

Altera Corporation acknowledges the trademarks of other organizations for their respective products or services mentioned in this document, including the following: Verilog is a registered trademark of Cadence Design Systems, Incorporated. Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation.

Altera reserves the right to make changes, without notice, in the devices or the device specifications identified in this document. Altera advises its customers to obtain the latest version of device specifications to verify, before placing orders, that the information being relied upon by the customer is current. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty. Testing and other quality control techniques are used to the extent Altera deems such testing necessary to support this warranty. Unless mandated by government requirements, specific testing of all parameters of each device is not necessarily performed. In the absence of written agreement to the contrary, Altera assumes no liability for Altera applications assistance, customer's product design, or infringement of patents or copyrights of third parties by or arising from use of semiconductor devices described herein. Nor does Altera warrant or represent any patent right, copyright, or other intellectual property right of Altera covering or relating to any combination, machine, or process in which such semiconductor devices might be or are used.

Altera products are not authorized for use as critical components in life support devices or systems without the express written approval of the president of Altera Corporation. As used herein:

1. Life support devices or systems are devices or systems that (a) are intended for surgical implant into the body or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Products mentioned in this document are covered by one or more of the following U.S. patents: 5,873,113; 5,872,463; 5,870,410; 5,861,760; 5,859,544; 5,850,365; 5,850,152; 5,850,151; 5,848,005; 5,847,617; 5,845,385; 5,844,854; RE35,977; 5,838,628; 5,838,584; 5,835,998; 5,834,849; 5,828,229; 5,825,197; 5,821,787; 5,821,773; 5,821,771; 5,815,726; 5,815,024; 5,815,003; 5,812,479; 5,812,450; 5,809,281; 5,809,034; 5,805,516; 5,802,540; 5,801,541; 5,796,267; 5,793,246; 5,790,469; 5,787,009; 5,771,264; 5,768,562; 5,768,372; 5,767,734; 5,764,583; 5,764,569; 5,764,080; 5,764,079; 5,761,099; 5,760,624; 5,757,207; 5,757,070; 5,744,991; 5,744,383; 5,740,110; 5,732,020; 5,729,495; 5,717,901; 5,705,939; 5,699,020; 5,699,312; 5,696,455; 5,693,540; 5,694,058; 5,691,653; 5,689,195; 5,668,771; 5,680,061; 5,672,985; 5,670,895; 5,659,717; 5,650,734; 5,649,163; 5,642,262; 5,642,082; 5,633,830; 5,631,576; 5,621,312; 5,614,840; 5,612,642; 5,608,337; 5,606,276; 5,606,266; 5,604,453; 5,598,109; 5,598,108; 5,592,106; 5,592,102; 5,590,305; 5,583,749; 5,581,501; 5,574,893; 5,572,717; 5,572,148; 5,572,067; 5,570,040; 5,567,177; 5,565,793; 5,563,592; 5,561,757; 5,557,217; 5,555,214; 5,550,842; 5,550,782; 5,548,552; 5,548,228; 5,543,732; 5,543,730; 5,541,530; 5,537,295; 5,537,057; 5,525,917; 5,525,827; 5,523,706; 5,523,247; 5,517,186; 5,498,975; 5,495,182; 5,493,526; 5,493,519; 5,490,266; 5,488,586; 5,487,143; 5,486,775; 5,485,103; 5,485,102; 5,483,178; 5,477,474; 5,473,266; 5,463,328; 5,444,394; 5,438,295; 5,436,575; 5,436,574; 5,434,514; 5,432,467; 5,414,312; 5,399,922; 5,384,499; 5,376,844; 5,375,086; 5,371,422; 5,369,314; 5,359,243; 5,359,242; 5,353,248; 5,352,940; 5,309,046; 5,350,954; 5,349,255; 5,341,308; 5,341,048; 5,341,044; 5,329,487; 5,317,212; 5,317,210; 5,315,172; 5,301,416; 5,294,975; 5,285,153; 5,280,203; 5,274,581; 5,272,368; 5,268,598; 5,266,037; 5,260,611; 5,260,610; 5,258,668; 5,247,478; 5,247,477; 5,243,233; 5,241,224; 5,237,219; 5,220,533; 5,220,214; 5,200,920; 5,187,392; 5,166,604; 5,162,680; 5,144,167; 5,138,576; 5,128,565; 5,121,006; 5,111,423; 5,097,208; 5,091,661; 5,066,873; 5,045,772; 4,969,121; 4,930,107; 4,930,098; 4,930,097; 4,912,342; 4,903,223; 4,899,070; 4,899,067; 4,871,930; 4,864,161; 4,831,573; 4,785,423; 4,774,421; 4,713,792; 4,677,318; 4,617,479; 4,609,986; 4,020,469 and certain foreign patents.

Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights.

Copyright © 1999 Altera Corporation. All rights reserved.



Printed on Recycled Paper.



I.S. EN ISO 9001



About this User Guide

September 1999

This user guide provides comprehensive information about the Altera® symbol interleaver/de-interleaver MegaCore functions.



For the most-up-to-date information about Altera products, go to the Altera world-wide web site at <http://www.altera.com>.

How to Contact Altera

For additional information about Altera products, consult the sources shown in [Table 1](#).

| <i>Table 1. How to Contact Altera</i> | | | |
|---------------------------------------|---|--|--|
| Information Type | Access | USA & Canada | All Other Locations |
| Altera Literature Services | Telephone hotline | (888) 3-ALTERA (1) | (888) 3-ALTERA (1) |
| | Electronic mail | lit_req@altera.com (1) | lit_req@altera.com (1) |
| Non-technical customer service | Telephone hotline | (800) SOS-EPLD | (408) 544-7000 |
| | Fax | (408) 544-7606 | (408) 544-7606 |
| Technical support | Telephone hotline (6:00 a.m. to 6:00 p.m. Pacific Time) | (800) 800-EPLD | (408) 544-7000 (1) |
| | Fax | (408) 544-6401 | (408) 544-6401 (1) |
| | Electronic mail | sos@altera.com | sos@altera.com |
| | FTP site | ftp.altera.com | ftp.altera.com |
| General product information | Telephone | (408) 544-7104 | (408) 544-7104 (1) |
| | World-wide web site | http://www.altera.com | http://www.altera.com |

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The *Symbol Interleaver/De-Interleaver MegaCore Function User Guide* uses the typographic conventions shown in [Table 2](#).

| Visual Cue | Meaning |
|---|--|
| Bold Type with Initial Capital Letters | Command names and dialog box titles are shown in bold, initial capital letters. Example: Save As dialog box. |
| bold type | External timing parameters, directory names, project names, disk drive names, filenames, filename extensions, and software utility names are shown in bold type. Examples: f_{MAX} , \maxplus2 directory, d: drive, chiptrip.gdf file. |
| <i>Bold italic type</i> | Book titles are shown in bold italic type with initial capital letters. Example: <i>1999 Data Book</i> . |
| <i>Italic Type with Initial Capital Letters</i> | Document titles, checkbox options, and options in dialog boxes are shown in italic type with initial capital letters. Examples: <i>AN 75 (High-Speed Board Design)</i> , the <i>Check Outputs</i> option, the <i>Directories</i> box in the Open dialog box. |
| <i>Italic type</i> | Internal timing parameters and variables are shown in italic type. Examples: <i>t_{PIA}</i> , <i>n + 1</i> . Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <i><file name></i> , <i><project name>.pof</i> file. |
| Initial Capital Letters | Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu. |
| “Subheading Title” | References to sections within a document and titles of MAX+PLUS II Help topics are shown in quotation marks. Example: “Configuring a FLEX 10K or FLEX 8000 Device with the BitBlaster™ Download Cable.” |
| Courier type | Reserved signal and port names are shown in uppercase Courier type. Examples: DATA1, TDI, INPUT. User-defined signal and port names are shown in lowercase Courier type. Examples: my_data, ram_input. Anything that must be typed exactly as it appears is shown in Courier type. For example: c:\max2work\tutorial\chiptrip.gdf. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword SUBDESIGN), as well as logic function names (e.g., TRI) are shown in Courier. |
| 1., 2., 3., and a., b., c.,... | Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure. |
| ■ | Bullets are used in a list of items when the sequence of the items is not important. |
| ✓ | The checkmark indicates a procedure that consists of one step only. |
| ☞ | The hand points to information that requires special attention. |
| ↵ | The angled arrow indicates you should press the Enter key. |
| ☞ | The feet direct you to more information on a particular topic. |

Altera MegaCore Functions

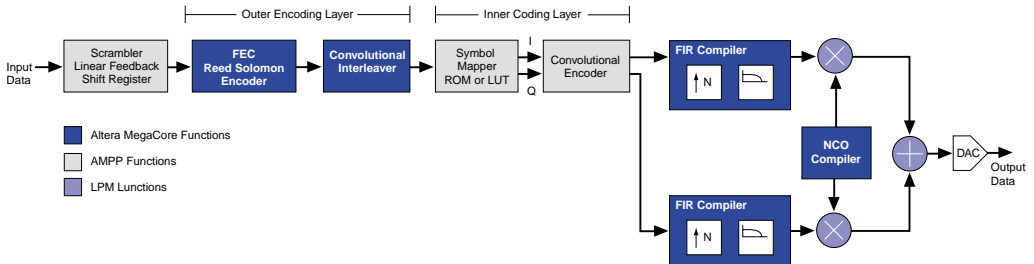
As programmable logic device (PLD) densities grow to over 250,000 gates, design flows must be as efficient and productive as possible. Altera provides ready-made, pre-tested, and optimized megafunctions that let you rapidly implement the functions you need, instead of building them from the ground up. Altera® MegaCore™ functions, which are reusable blocks of pre-designed intellectual property, improve your productivity by allowing you to concentrate on adding proprietary value to your design. When you use MegaCore functions, you can focus on your high-level design and spend more time and energy on improving and differentiating your product.

Traditionally, designers have been forced to make a tradeoff between the flexibility of digital signal processors (DSP) and the performance of DSP application-specific integrated circuit (ASIC) and application-specified standard product (ASSP) solutions. The Altera FLEX® DSP solution eliminates the need for this tradeoff by providing exceptional performance combined with the flexibility of programmable logic devices. The DSP solution ranges from a highly parameterized finite impulse response (FIR) filter compiler to optimized higher-level fast Fourier transform (FFT), adaptive FIR filter, and interleaver/de-interleaver megafunctions.

Altera digital signal processing (DSP) solutions include MegaCore functions developed and supported by Altera. Altera's APEX™ and FLEX devices easily implement DSP applications, while leaving ample room for your custom logic. The devices are supported by Altera's MAX+PLUS® II development system, which allows you to perform a complete design cycle including design entry, synthesis, place-and-route, simulation, timing analysis, and device programming. Altera devices, software, and DSP MegaCore functions provide you with a complete design solution.

Altera DSP solutions include MegaCore functions developed and supported by Altera, and Altera Megafunction Partners Program (AMPP) functions. Additionally, many commonly used functions, such as adders and multipliers, are available from the industry-standard library of parameterized modules (LPM). Altera devices easily implement DSP applications, while leaving ample room for your custom logic. The devices are supported by Altera's MAX+PLUS[®] II and Quartus[™] development systems, which allow you to perform a complete design cycle including design entry, synthesis, place-and-route, simulation, timing analysis, and device programming. Altera devices, software, and DSP MegaCore functions provide you with a complete design solution. Figure 1 shows a complete system and highlights the functions that are available from Altera and other vendors.

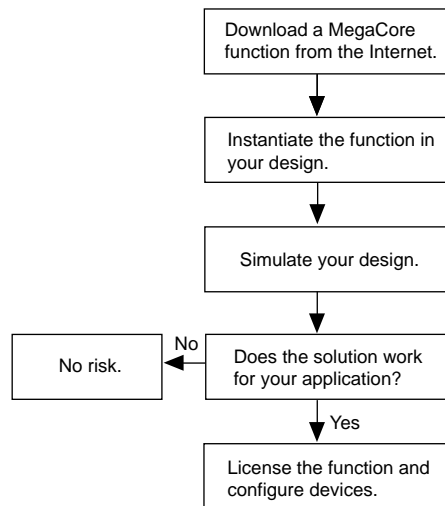
Figure 1. Typical Modulator



This user guide provides detailed information on the symbol interleaver/de-interleaver MegaCore functions, including a technical specification, instructions on how to install and use the functions, and reference designs.

OpenCore Feature

Altera's exclusive OpenCore[™] feature allows you to evaluate MegaCore functions before deciding to license them. You can instantiate a MegaCore function in your design, compile and simulate the design, and then verify the MegaCore function's size and performance. This evaluation provides first-hand functional, timing, and other technical data that allows you to make an informed decision on whether to license the MegaCore function. Once you license a MegaCore function, you can use the MAX+PLUS II or Quartus software to generate programming files as well as EDIF, VHDL, or Verilog HDL output netlist files for simulation in third-party EDA tools. Figure 2 shows a typical design flow using MegaCore functions and the OpenCore feature.

Figure 2. OpenCore Design Flow using the Symbol Interleaver/De-Interleaver

Altera Devices

The symbol interleaver/de-interleaver MegaCore functions have been optimized and targeted for Altera APEX and FLEX devices. You can implement the memory required by the symbol interleaver/de-interleaver function in APEX 20K embedded system blocks (ESBs) or FLEX 10K embedded array blocks (EABs). The MegaWizard Plug-In selects the optimum memory configuration based on the parameters you enter. For block interleaving, the MegaWizard Plug-In uses single-port RAM; for convolutional interleaving the function takes advantage of the dual-port RAM capability of the EABs and ESBs to offer high data rates (e.g., 120 megasamples per second)

APEX 20K Devices

APEX 20K devices offer complete system-level integration on a single device. The APEX MultiCore™ architecture delivers the ultimate in design flexibility and efficiency for high-performance System-on-a-Programmable Chip™ applications. With densities ranging from 100,000 to 1,000,000 gates, the APEX 20K architecture integrates look-up-table (LUT) logic, product-term logic, and memory into a single architecture, eliminating the need for multiple devices, saving board space, and simplifying the implementation of complex designs.

In the APEX MultiCore architecture, embedded system blocks (ESBs) and logic array blocks (LABs) are combined into MegaLAB™ structures. Each APEX 20K ESB can be configured as product-term logic, enabling APEX 20K devices to achieve unmatched integration efficiency, as LUT logic, or as memory. The ESB can be configured as dual-port RAM, with a wide range of RAM widths and depths, or ROM in APEX 20K devices, and as content-addressable memory

(CAM), a memory technology that accelerates applications requiring fast searches, in APEX 20KE devices.

FLEX 10K Devices

The FLEX 10K embedded programmable logic device (PLD) family delivers the flexibility of traditional programmable logic with the efficiency and density of gate arrays with embedded memory. FLEX 10K devices feature EABs, which are 2 Kbits of RAM that can be configured as 256×8 , 512×4 , $1,024 \times 2$, or $2,048 \times 1$ blocks. The new 2.5-V FLEX 10KE devices support efficient implementation of dual-port RAM, and further enhance the performance of the FLEX 10K family.

FLEX 6000 Devices

Altera's 5.0-V and 3.3-V FLEX 6000 devices deliver the flexibility and time-to-market of programmable logic at prices that are competitive with gate arrays. Featuring the OptiFLEX™ architecture, FLEX 6000 devices provide a flexible, high-performance, and cost-effective alternative to ASICs for high-volume production.



For more information on APEX 20K, FLEX 10K, and FLEX 6000 devices, refer to the following documents:

- [*APEX 20K Programmable Logic Device Family Data Sheet*](#)
- [*FLEX 10K Embedded Programmable Logic Family Data Sheet*](#)
- [*FLEX 10KE Embedded Programmable Logic Family Data Sheet*](#)
- [*FLEX 6000 Programmable Logic Device Family Data Sheet*](#)

Software Tools

Altera offers the fastest, most powerful, and most flexible programmable logic development software in the industry. The MAX+PLUS II and Quartus software offer a completely integrated development flow and an intuitive, graphical user interface, making them easy to learn and use.

The MAX+PLUS II software offers a seamless development flow, allowing you to enter, compile, and simulate your design and program devices using a single, integrated tool, regardless of the Altera device you choose. Altera's fourth-generation Quartus software offers designers the ideal platform for processing multi-million gate designs, with state-of-the-art features that shorten design cycles, streamline the development flow, and reduce verification time. The Quartus software shortens design cycles with the revolutionary nSTEP™ Compiler, which permits incremental recompilation, and multiple processor support that can operate locally or across networks and even platforms.

MegaWizard Plug-Ins

MegaWizard Plug-Ins are parameterization tools that help you integrate megafunctions into your designs without requiring the use of third-party tools. You can use this feature in the software (version 8.2 and higher) or as a stand-alone tool with third-party EDA design interfaces. MegaWizard Plug-Ins provide maximum flexibility, allowing you to customize megafunctions without changing your design's source code. You can integrate a parameterized megafunction in any hardware description language (HDL) or netlist file using any EDA tool.

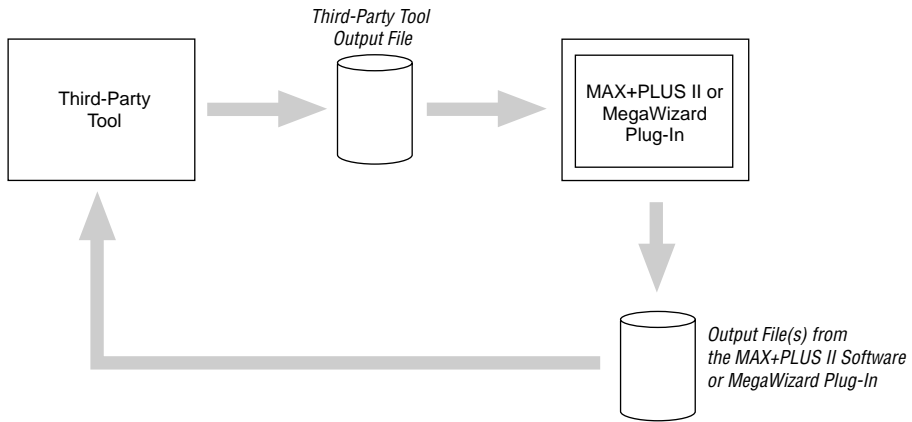
The MAX+PLUS II MegaWizard Plug-In Manager allows you to bring up the megafunction's wizard so that you can set the parameters of the megafunction to fit your design. A custom megafunction variation is then generated that you can instantiate in your design file.

EDA Interfaces

As a standard feature, the MAX+PLUS II software interfaces with all major EDA design tools, including tools for ASIC designers. Once a design is captured and simulated using the tool of your choice, you can transfer your EDIF file directly into the MAX+PLUS II software. After synthesis and fitting, you can transfer your file back into your tool of choice for simulation. The MAX+PLUS II system outputs the full-timing VHDL, Verilog HDL, Standard Delay Format (SDF), and EDIF netlists that can be used for post-route device- and system-level simulation. Altera opened the Quartus interface to various EDA partners to enable them to provide unmatched levels of integration. NativeLink integration provides a truly seamless interface to major EDA software tools to support existing design flows, eliminating the need to learn new design tools. [Figure 3](#) shows a typical design flow.



Some MegaWizard Plug-Ins generate additional output files, such as MATLAB-compatible files, and VHDL and Verilog HDL testbenches.

Figure 3. MAX+PLUS II/EDA Tool Design Flow

To simplify the design flow between Altera software and other EDA tools, Altera has developed the MAX+PLUS II Altera Commitment to Cooperative Engineering Solutions (ACCESSSM) Key Guidelines and the Quartus NativeLink Guidelines. These guidelines provide complete instructions on how to create, compile, and simulate your design with tools from leading EDA vendors. The guidelines are part of Altera's ongoing efforts to give you state-of-the-art tools that fit into your design flow, and to enhance your productivity for even the highest-density devices. The MAX+PLUS II ACCESS Key Guidelines are available on the Altera web site (<http://www.altera.com>) and the MAX+PLUS II CD-ROM. The NativeLink guidelines are integrated into Quartus Help.

Features

- High-speed data rates: 120 megasamples per second (MSPS)
- Supports convolutional interleaving algorithm
- Supports block interleaving algorithm
- Parameterized symbol width and codeword length
- Compatible with discrete and streaming Reed-Solomon encoders/decoders
- Optimized for APEX and FLEX devices
- Internal or external memory architecture
- Test-vector generation
- Contains a burst error distribution analyzer

General Description

Interleaving is a standard DSP function used in many communications systems. Applications that store or transmit digital data require error correction to reduce the effect of spurious noise that can corrupt data. Digital communications systems designers can choose many types of error-correction codes (EECs) to reduce the effect of errors in stored or transmitted data. For example, Reed-Solomon encoders/decoders, which are block-encoding algorithms, are used frequently to perform forward error correction (FEC).

Symbol interleaver/de-interleavers can mitigate the effects of burst noise. Typically, these functions are needed for transport channels that require a bit error ratio (BER) on the order of 10^{-6} . Interleaving improves the efficiency of Reed-Solomon encoders/decoders by spreading burst errors across several Reed-Solomon codewords.

The Altera symbol interleaver/de-interleaver function uses internal or external single-port or dual-port RAM. You can implement single-port RAM using FLEX 10K EABs or an external RAM device; you can implement dual-port RAM using the dual-port RAM capability of APEX 20K ESBs or FLEX 10KE embedded array blocks (EABs), or an external RAM device. Dual-port RAM provides a faster and smaller implementation than single-port RAM.

The function uses a 120-MHz frequency range and supports both continuous streaming and discrete mode, making the interleaver/de-interleaver compatible with any type of Reed-Solomon function.

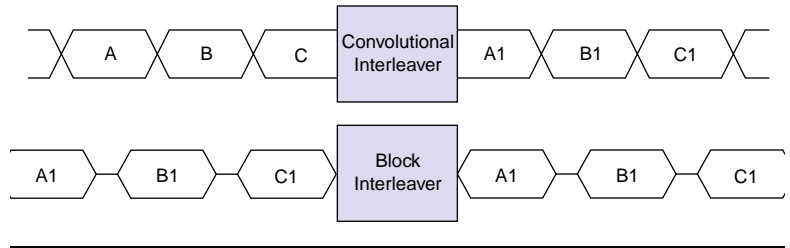
The MegaWizard Plug-In provided with the symbol interleaver/de-interleaver drastically reduces the design creation and simulation cycles from several weeks to several minutes. The wizard generates a highly optimized instance of a custom interleaver/de-interleaver function as well as a MAX+PLUS II Vector File (.vec) that you can use to simulate the function. For example, by choosing a few simple settings you can build an interleaving function as described in a standard, such as DVB 802 or UMTS. Additionally, you can implement a custom interleaving function by specifying the parameter values for your specific transmission channel requirements. The flexibility of programmable logic combined with the symbol interleaver/de-interleaver MegaWizard Plug-In allows you to build real-time systems to analyze and improve parameter values determined by theoretical equations.

Functional Description

Interleaving is the process of reordering the symbols in a group of transmitted codewords such that adjacent symbols in the data stream are not from the same codeword. The receiver reassembles the codewords when it processes the data stream. The interleaving process helps reduce the effects of burst errors (i.e., multiple errors in a row), because rearranging the codeword symbols spreads the errors among multiple codewords.

Depending on your application, you may choose to implement a convolutional or a block interleaver/de-interleaver. Convolutional interleaver/de-interleaver functions process data in a continuous stream, which makes them ideal for high-speed applications that require correction for burst errors (e.g., digital video broadcasting). Typically, these functions are used with Reed-Solomon functions. Block interleaver/de-interleavers process data in a discrete stream and are used in applications such as GSM or UMTS (i.e., mobile phones). These functions are often used with Reed-Solomon functions or Turbo Code encoders/decoders. Compared to block interleavers/de-interleavers, convolutional interleavers/de-interleavers provide reduced delay and lower memory usage for the same distribution of errors. [Figure 1](#) compares the data streaming performed by the functions.

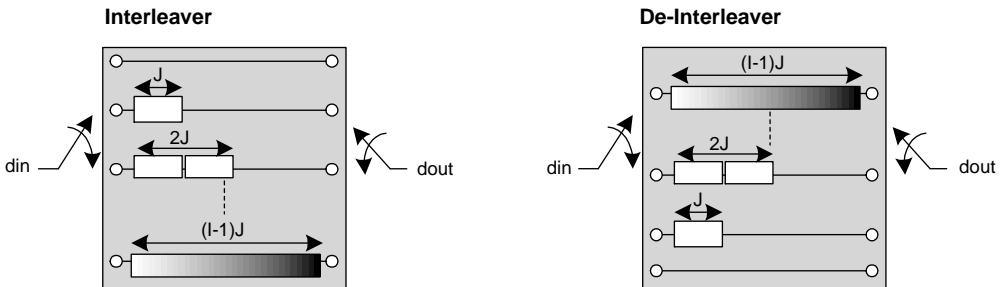
Figure 1. Data Stream Comparison



Convolutional Interleaver/De-Interleaver

Figure 2 illustrates convolutional interleaving and de-interleaving. The number of branches is called the depth of the interleaver. The first branch has no delay. Each consecutive branch introduces an additional symbol delay. Each symbol contains a variable number of bits that the user can modify through the MegaWizard Plug-In. The commutator connects to each branch in order, for every symbol. Therefore, when a symbol enters a branch of the interleaver, it leaves the branch via the output commutator after a variable delay that depends on the branch index.

Figure 2. Convolutional Structure



For the de-interleaver, the delays introduced at each branch are complementary to the interleaver. That is, once a symbol enters the branch of the de-interleaver, it leaves the branch via the output commutator after a variable delay that depends on the branch index. With this interleaving structure, every symbol incurs a fixed delay through the combination of the interleaver and de-interleaver. For synchronization purposes, the sync bytes and inverted sync bytes must be routed to the first branch of the interleaver, which corresponds to a null delay. In principle, the de-interleaver is similar to the interleaver, but the branch indexes are reversed, i.e., branch 1 incurs the largest delay. The de-interleaver synchronization can be performed by routing the first recognized sync byte in branch 1.

The interleaving/de-interleaving process introduces a constant delay between the interleaver input data and the de-interleaver output data. The delay is calculated using the following equation:

$$\text{depth} \times \text{symbol delay} \times (\text{depth} - 1) + 6$$

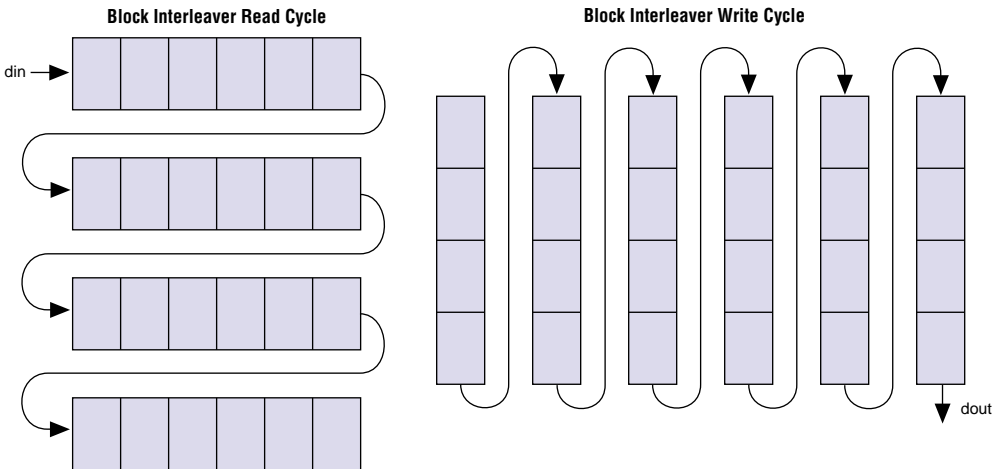
The symbol throughput is equal to the frequency of the function.

Knowing this delay, you can design your application to operate on a continuous stream of codewords. To support continuous streaming, you must use dual-port memory to implement a convolutional interleaver/de-interleaver function. See [“Performance” on page 14](#) for an example of the resource usage and speed of the convolutional function.

Block Interleaver/De-Interleaver

The block interleaver/de-interleaver uses single-port SRAM memory configured as a matrix of n rows by m columns to perform interleaving. During the write cycle, the input symbols are written column by column; during a read cycle, the output symbols are read row by row. The column length is usually equal to the codeword length of the FEC encoder, while the numbers of rows (often called the span) is the interleaver delay. [Figure 3](#) illustrates block function operation using a 6-symbol codeword.

Figure 3. Block Structure for a 6-Symbol Codeword



The block interleaver/de-interleaver operates in discrete mode with a single -port memory used as a buffer. The symbol transmission consists of an alternating sequence of write and read cycles. Each cycle delay is equal to the buffer size, which is the block length multiplied by the span delay.

The total cumulative delay from the transmitter to the receiver can be calculated using the following equation:

$$2 \times \text{Number of Rows} \times (\text{Number of Columns} + 4)$$

MegaWizard Plug-In

You can run the symbol interleaver/de-interleaver wizard using the MAX+PLUS II MegaWizard Plug-In Manager. The wizard generates a custom megafunction variation that you can instantiate in your design file. [Table 1](#) describes the options for the symbol interleaver/de-interleaver wizard.

Table 1. Symbol Interleaver/De-Interleaver Wizard Options

| Option | Function | Description |
|--------------------|------------------------|--|
| Number of columns | Block | Specifies the total length of the codeword (i.e., data symbol + checksum symbol). |
| Number of branches | Convolutional | Specifies the number of branches used by the interleaver. |
| Direction | Block Convolutional | Indicates whether you wish to create an interleaver (transmitter) or a de-interleaver (receiver) |
| Memory Type | Block Convolutional | Indicates whether you wish to use internal or external memory. Convolutional interleaving uses synchronous dual-port RAM. Block interleaving uses synchronous single-port RAM. For internal memory, the MegaWizard Plug-In automatically instantiates the most optimum EAB configuration |
| Number of rows | Block | Specifies the maximum number of codewords in the block interleaver/de-interleaver memory. |
| Unit delay element | Convolutional | Specifies the unit delay for each branch of the interleaver/de-interleaver. |
| Symbol Width | Block Convolutional | Specifies the width of the input symbol. |
| Type | Block Convolutional | Indicates whether you wish to create a block or convolutional interleaver/de-interleaver. |

When you create a new custom interleaver/de-interleaver function, the wizard creates the following files:

- AHDL Text Design File (.tdf), VHDL Design File (.vhd), or Verilog Design File (.v) of the custom function that you can instantiate in your system design
- MAX+PLUS II Vector File (.vec) that you can use for simulation
- Symbol File (.sym) that you can use to include the function in a Graphic Editor design

Signals

The symbol interleaver/de-interleaver function has the signals shown in [Table 2](#). Different signals are required depending on whether you implement a block or convolutional function and whether you use internal or external RAM.

| Signal | Function | Memory Type | Description |
|-------------|------------------------|----------------------|--|
| addr[] | Block | External | Memory address bus. |
| clk | Block Convolutional | Internal External | Input clock signal. |
| clken | Block Convolutional | Internal External | Active-high clock enable. |
| block_full | Block | External | Indicates that a block is full. <code>block_full</code> goes high when the memory block is full. |
| block_empty | Block | External | Indicates that a block is empty. <code>block_empty</code> goes high when the memory block is empty. |
| din[] | Block Convolutional | Internal | Input symbol. |
| next_din | Block | Internal | Input enable. When <code>next_din</code> goes high, the next codeword symbol input is ready to be read. |
| dout[] | Block Convolutional | Internal | Output symbol. |
| dout_valid | Block | Internal | Output enable. When <code>dout_enable</code> goes high, the next codeword symbol output is ready to be read. |
| read_add[] | Block Convolutional | External | Read address bus output. |
| sync_in | Block Convolutional | Internal External | Active-high input resynchronization signal. For convolutional functions, set the branch pointer to branch 0; for block functions, set the branch pointer to column 0, row 0. |

Table 2. Symbol Interleaver/De-Interleaver Signals (Part 2 of 2)

| Signal | Function | Memory Type | Description |
|--------------|------------------------|----------------------|--|
| sync_out | Convolutional | Internal External | Output synchronization signal. The sync_out signal goes high on the first branch of the interleaver. |
| write_add[] | Block Convolutional | External | Write address bus output. |
| write_enable | Block | External | Write enable. The write_enable signal is active high, and should be connected to the write enable of the external RAM. |

Simulation

The wizard generates MAX+PLUS II Vector Files (.vec) that you can use to simulate your custom interleaver/de-interleaver. After compiling your design, you can use the MAX+PLUS II Simulator to view the functionality and timing. The Simulator converts the wizard-generated Vector Files into graphical waveforms. Figures 4, 5, and 6 show sample waveforms for the symbol interleaver/de-interleaver function.

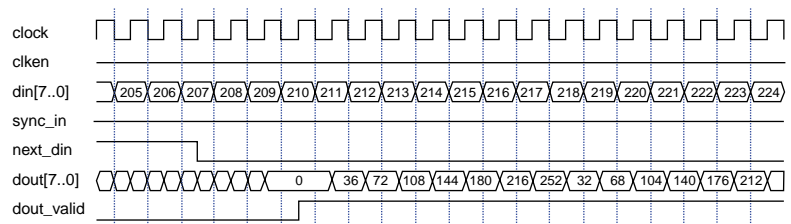
Figure 4. Block Interleaver Read Cycle Waveform

Figure 5. Block Interleaver Write Cycle Waveform

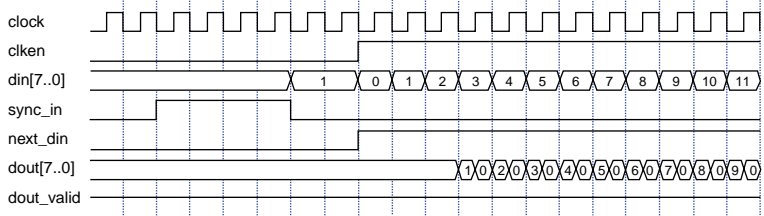
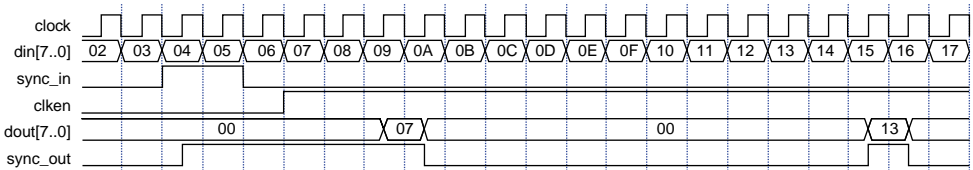


Figure 6. Convolutional Interleaver Waveform



See “MAXPLUS II Simulation” on page 21 and MAX+PLUS II Help for complete instructions on how to perform simulation using these vector files and how to view the resulting waveforms.

Performance

Table 3 shows the interleaver/de-interleaver function’s performance as calculated using the MAX+PLUS II version 9.2 software.

| Table 3. Symbol Interleaver/De-Interleaver Performance | | | | | | |
|---|--|---------------|--------------------|-----------------|------------------|------------------------------|
| Function | Parameters | Device | Speed Grade | LEs Used | EABs Used | f_{MAX} (MHz) |
| Convolutional interleaver using FLEX 10KE EABs | Depth = 12, Unit Delay = 17, Symbol Width = 8 bits (digital video broadcast settings) | FLEX 10KE | 1 | 392 | 8 | 110 |
| Block interleaver using single-port RAM | Block length = 36, Span delay = 20, Data width = 8, (UTRA) ITU-R RTT | FLEX 10KE | -1 | 40 | 4 | 120 |

Altera® DSP MegaCore™ functions provide solutions for integrating symbol interleavers and de-interleavers into your digital communications system. The functions are optimized for Altera APEX® and FLEX® devices, greatly enhancing your productivity by allowing you to focus efforts on the custom logic in the system. This section describes how to obtain the interleaver/de-interleaver MegaCore functions, explains how to install them on your PC, and walks you through the process of implementing the function in a design. You can test-drive MegaCore functions using Altera's OpenCore™ feature to simulate the functions within your custom logic. When you are ready to license a function, contact your local Altera sales representative.

Before You Begin

Before you can start using the symbol interleaver/de-interleaver MegaCore functions, you must obtain the MegaCore files and install them on your PC. The following instructions describe this process.

Obtaining the Functions

If you have Internet access, you can download MegaCore functions from Altera's web site at <http://www.altera.com>. Follow the instructions below to obtain the MegaCore functions via the Internet. If you do not have Internet access, you can obtain the MegaCore functions from your local Altera representative.

1. Run your web browser (e.g., Netscape Navigator or Microsoft Internet Explorer).
2. Open the URL <http://www.altera.com>.
3. Click the Tools icon on the home page toolbar.
4. Click the MegaCore Functions link.
5. Click the link for the symbol interleaver/de-interleaver function.
6. Follow the on-line instructions to download the function and save it to your hard disk.

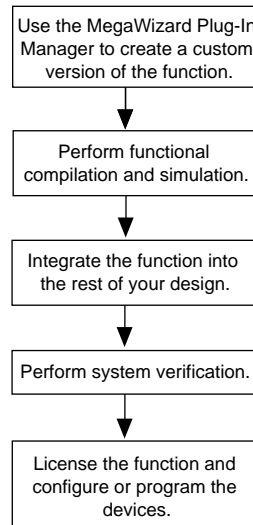
Installing the MegaCore Files

To install the files on Windows 95/98 or Windows NT 4.0, follow the instructions below:

1. Click **Run** (Start menu).
2. Type `<path name>\<filename>.exe`, where `<path name>` is the location of the downloaded MegaCore function and `<filename>` is the filename of the function.
3. Click **OK**. The **MegaCore Installer** dialog box appears. Follow the on-line instructions to finish installation.
4. After you have finished installing the MegaCore files, you must specify the directory in which you installed them as a user library in the MAX+PLUS II software. Search for “User Libraries” in MAX+PLUS II Help for instructions on how to add these libraries.

Walk-Through Overview

This section describes an entire design flow using Altera interleaver/de-interleaver MegaCore function and the MAX+PLUS II development system. The MegaWizard Plug-In Manager, which you can use within the Quartus™ or MAX+PLUS II software or as a stand-alone application, lets you create or modify design files that contain custom megafunction variations. You can then instantiate the custom megafunction in your design file. The symbol interleaver/de-interleaver MegaCore is parameterized, and you can customize the function to meet the needs of your application. Altera provides a MegaWizard Plug-In with the MegaCore function, which you can use to specify custom parameters quickly. See [Figure 1](#).

Figure 1. Example Design Flow

The instructions assume that:

- You are using a PC.
- You are familiar with the MAX+PLUS II software.
- MAX+PLUS II version 9.1 or higher is installed in the default location (c:\maxplus2).
- You are using the OpenCore feature to test-drive the function or you have licensed the function.

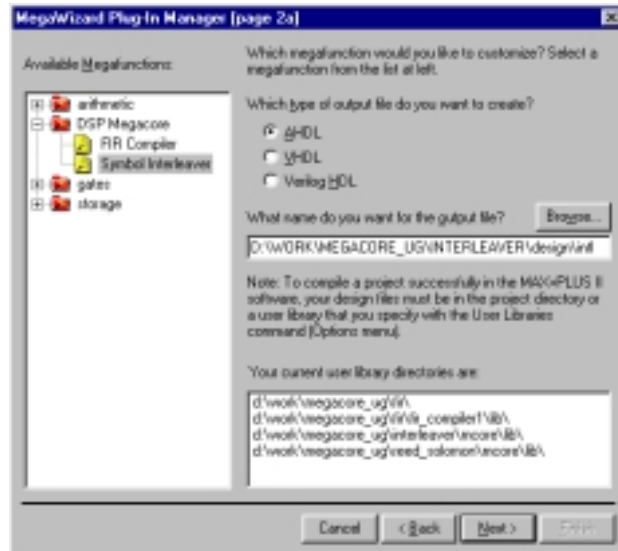
You can use Altera's OpenCore feature to compile and simulate the MegaCore functions, allowing you to evaluate the functions before deciding to license them. However, you must obtain a license from Altera before you can generate programming files or EDIF, VHDL, or Verilog HDL netlist files for simulation in third-party EDA tools.

Design Entry

To create custom version of the interleaver/de-interleaver function, follow these steps:

1. Start the MegaWizard Plug-in Manager by choosing the **MegaWizard Plug-In Manager** command (File menu) in any MAX+PLUS II application, or by start the stand-alone version of the MegaWizard Plug-In Manager by typing the command `megawiz 9` at a command or UNIX prompt. The **MegaWizard Plug-In Manager** dialog box is displayed. Refer to MAX+PLUS II Help for detailed instructions on how to use the MegaWizard Plug-In Manager.
2. Specify that you want to create a new custom megafunction and click **Next**.
3. On the second page of the wizard, select the interleaver function from the *DSP MegaCore* drop-down list in the *Available Megafunctions* box, choose whether you wish to create a TDF, VHDL Design File, or Verilog Design File, specify the directory and name for the files the wizard creates, and click **Next**. See [Figure 2](#).

Figure 2. Specify the Function to Customize

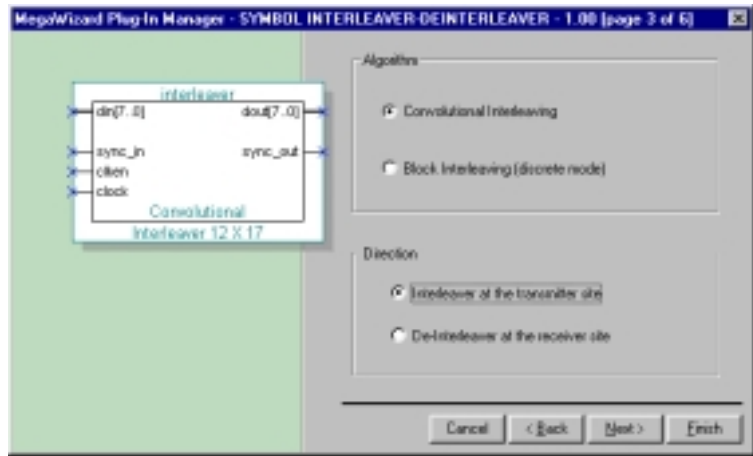




If you do not specify the directory in which you installed the interleaver/de-interleaver files as a user library in the MAX+PLUS II software, the function will not appear in the *Available Megafunctions* box. Search for “User Libraries” in MAX+PLUS II Help for information on how to specify these libraries.

- Specify the the type of algorithm (convolutional or block) and the direction (interleaver or de-interleaver) and click **Next**. See [Figure 3](#).

Figure 3. Choose the Algorithm & Direction



- Specify the characteristics of the function. As you adjust the settings, the wizard graphically displays a block diagram of the function. [Table 1](#) describes the options for the characteristics.

| Function | Variable | Description |
|---------------|----------|---------------------|
| Block (1) | <i>I</i> | Number of columns. |
| | <i>J</i> | Number of rows. |
| Convolutional | <i>I</i> | Unit delay value. |
| | <i>J</i> | Number of branches. |

Note:

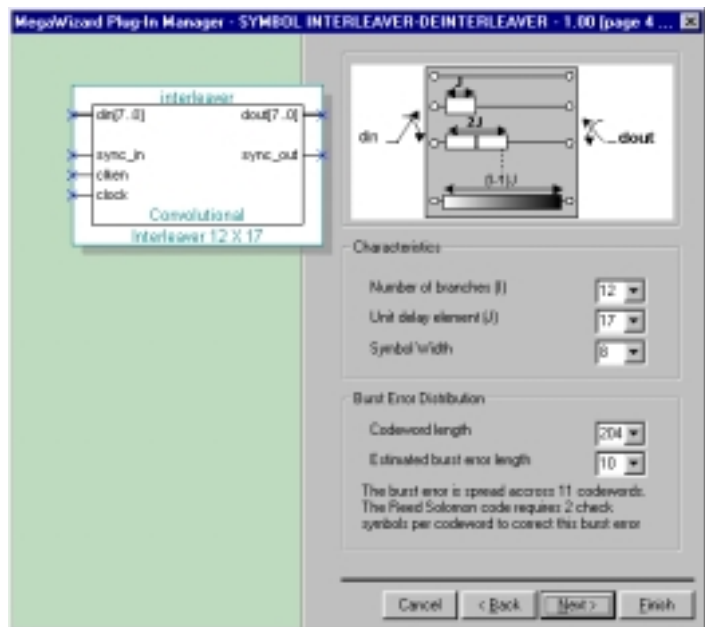
- Typically, the Reed-Solomon codeword length is the number of columns and the number of rows is the number of codewords to interleave.



In a given system, you should use the same interleaving algorithm for both the transmitter and receiver, as well as using the same characteristics.

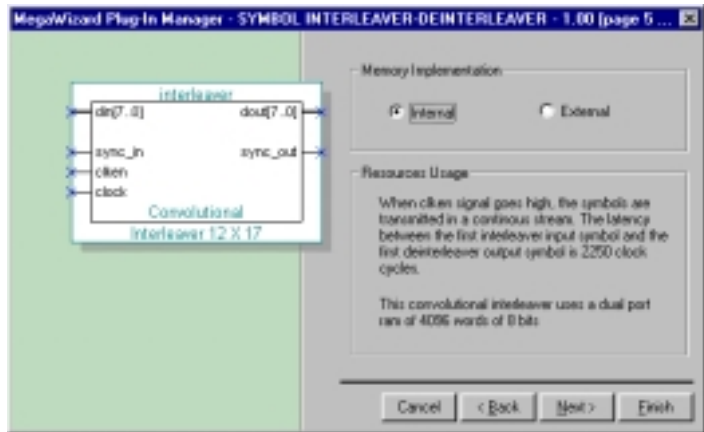
- You can use the drop-down list boxes in the *Burst Error Distribution* box to perform burst error analysis (see [Figure 4](#)). Indicate the codeword length and the estimated burst error length. The wizard displays the number of codewords the burst error is spread across before interleaving and indicates the number of Reed-Solomon check symbols required to correct the error. The calculation assumes that two Reed-Solomon check symbols are required to fix one symbol error per codeword. Click **Next** when you are finished.

Figure 4. Specify the Characteristics



- Specify whether you wish to use internal or external memory. The wizard estimates the resources required by the function (see [Figure 5](#)). Click **Next**.

Figure 5. Memory & Resource Usage



8. Click **Finish** to accept your selections and generate the megafunction variation.

Once you have created a megafunction variation, you can instantiate it into your design. After you have finished your design, you are ready to perform functional simulation to verify that your circuit is working correctly.

MAXPLUS II Simulation

The following steps explain how to simulate your design after you have compiled it in the MAX+PLUS II software.

1. Run the MAX+PLUS II Simulator. The vector file created by the MegaWizard Plug-In Manager for your custom interleaver/de-interleaver function is loaded automatically. Click Start to begin simulation.
2. Once simulation has completed, click the **Open SCF** button to view the waveform for the design.

After you have verified that the custom MegaWizard-generated function is compatible with your design requirements, you can insert the function into your system-level design. The wizard generates the following files, which can be referenced in your top-level file:

- MAX+PLUS II Symbol File (**.sym**)
- Altera Hardware Description Language (AHDL) TDF (**.tdf**)
- Verilog Design File (**.v**)
- VHDL Design File (**.vhd**)

Using Third-Party EDA Tools

As a standard feature, Altera's MAX+PLUS II software works seamlessly with tools from all EDA vendors, including Cadence, Exemplar Logic, Mentor Graphics, Synopsys, Synplicity, and Viewlogic. After you have licensed the MegaCore function, you can generate EDIF, VHDL, Verilog HDL, and Standard Delay output files from the MAX+PLUS II software and use them with your existing EDA tools to perform functional modeling and post-route simulation of your design.

To simplify the design flow between the MAX+PLUS II software and other EDA tools, Altera has developed the MAX+PLUS II Altera Commitment to Cooperative Engineering Solutions (ACCESSSM) Key Guidelines. These guidelines provide complete instructions on how to create, compile, and simulate your design with tools from leading EDA vendors. The MAX+PLUS II ACCESS Key Guidelines are part of Altera's ongoing efforts to give you state-of-the-art tools that fit into your design flow, and to enhance your productivity for even the highest-density devices. The MAX+PLUS II ACCESS Key Guidelines are available on the Altera web site (<http://www.altera.com>) and the MAX+PLUS II CD-ROM.

The following sections describe how to generate a VHDL or Verilog HDL model, and describe the design flow to compile and simulate your custom MegaCore design with a third-party EDA tool. Refer to Figure 3 on page 6, which shows the design flow for interfacing your third-party EDA tool with the MAX+PLUS II software.

Synthesis Compilation & Post-Routing Simulation

To synthesize your design in a third-party EDA tool and perform post-route simulation, perform the following steps:

1. Create your custom design instantiating a symbol interleaver/de-interleaver MegaCore function.
2. Synthesize the design using your EDA tool. Your EDA tool should treat the MegaCore instantiation as a black box by either setting attributes or ignoring the instantiation.



For more information on setting compiler options in your EDA tool, refer to the MAX+PLUS II ACCESS Key Guidelines.

3. After compilation, generate a hierarchical EDIF netlist file in your EDA tool.
4. Open your EDIF file in the MAX+PLUS II software.
5. Set your EDIF file as the current project in the MAX+PLUS II software.

6. Choose **EDIF Netlist Reader Settings** (Interfaces menu).
7. In the **EDIF Netlist Reader Settings** dialog box, select the vendor for your EDIF netlist file in the *Vendor* drop-down list box and click **OK**.
8. Make logic option and/or place-and-route assignments for your custom logic using the commands in the Assign menu.
9. In the MAX+PLUS II Compiler, make sure **Functional SNF Extractor** (Processing menu) is turned off.
10. Turn on the **Verilog Netlist Writer** or **VHDL Netlist Writer** command (Interfaces menu), depending on the type of output file you want to use in your third-party simulator. Set the netlist writer settings.
11. Compile your design. The MAX+PLUS II Compiler synthesizes and performs place-and-route on your design, and generates output and programming files.
12. Import your MAX+PLUS II-generated output files (**.edo**, **.vho**, **.vo**, or **.sdo**) into your EDA tool for post-route, device-level, and system-level simulation.

When you install the symbol interleaver/de-interleaver functions, the installation program creates the directory **symbol_interleaver/reference_design/dvb**. This directory contains the reference design files for a digital video broadcast application that conforms to IEEE Std. 802.14. This example illustrates how to combine and synchronize the interleaver and de-interleaver.

The system parameters, which meet IEEE Std. 802.14, are as follows:

- Depth = 12
- Unit delay = 17
- Symbol width = 8

To use the reference design, perform the following steps:

1. Open the file **top_dvb.gdf** in the **symbol_interleaver/reference_design/dvb** directory.
2. Choose **Project Set Project to Current File** (File menu).
3. Open the MAX+PLUS II Compiler and click the **Start** button to begin compilation.

Convolutional Interleaver

Example: DVB IEEE Std. 802.14 Transmitter & Receiver

4. Open the MAX+PLUS II Simulator; the wizard-generated vector file is loaded automatically.
5. Click the **Start** button to begin simulation.
6. When simulation is complete, click **Open SCF** to view the waveform for the design.

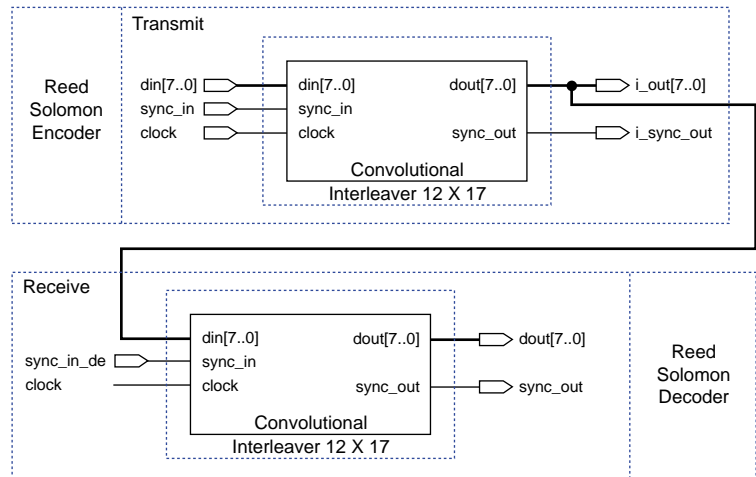
You can change the parameters of the reference design by editing the interleaver and de-interleaver variations. For example, in the Graphic Editor, you can double-click on the symbols in the **top_dvb.gdf** file to open the variation for editing.



A system implementing both an interleaver and de-interleaver only operates correctly if the parameter values of the interleaver and de-interleaver are identical.

Figure 6 shows a block diagram of a system using the convolutional interleaver/de-interleaver with a Reed Solomon encoder/decoder.

Figure 6. Interleaver/De-Interleaver & Reed Solomon Functions



Block Interleaver Example: UMTS Transmitter & Receiver

When you install the symbol interleaver/de-interleaver functions, the installation program creates the directory **symbol_interleaver/reference_design/umts**, which contains the reference design files for a UMTS application. The system parameters for this application are as follows:

- Number of columns = 36
- Number of rows = 20

To use the reference design, perform the following steps:

1. Open the file **top_umts.gdf** in the **symbol_interleaver/reference_design/umts** directory.
2. Choose **Project Set Project to Current File** (File menu).
3. Open the MAX+PLUS II Compiler and click the **Start** button to begin compilation.
4. Open the MAX+PLUS II Simulator; the wizard-generated vector file is loaded automatically.
5. Click the **Start** button to begin simulation.
6. When simulation is complete, click **Open SCF** to view the waveform for the design.

You can change the parameters of the reference design by editing the interleaver and de-interleaver variations. For example, in the Graphic Editor, you can double-click on the symbols in the **top_umts.gdf** file to open the variation for editing.

Glossary

bit error ratio (BER) The BER is the ratio of error bits to transmitted bits.

codeword A block of data to be interleaved and transmitted.

embedded array block (EAB) The building block of embedded arrays in FLEX 10K devices. Each EAB provides 2,048 or 4,096 bits of configurable RAM, ROM, FIFO, or dual-port RAM.

embedded system block (ESB) The embedded system block resides in the MultiCore architecture of the APEX 20K. Each ESB contains 2,048 programmable bits that can be configured as product-term logic, look-up table-based logic, or dual-port RAM, ROM, or content addressable memory (CAM). Each ESB can be configured with up to 16 macrocells, and can contain up to 32 product terms, XOR logic, 16 D-flipflops, and parallel expanders.

FEC Forward Error Correction

logic element (LE) The basic building block of a FLEX or APEX device. A logic element consists of a look-up table (LUT)—i.e., a function generator that quickly computes any function of four variables—and a programmable register to support sequential functions.

MSPS Megasample per second

MBPS Megabits per second

Reed Solomon functions Reed Solomon encoders/decoders convert a data stream into a number of information symbols, followed by several check, or parity, symbols.

span The numbers of rows used by an interleaver.

symbol An individual data bit; a codeword is composed of one or more symbols.

References

Andrews, Kenneth, Chris Heegard, and Dexter Kozen. *A Theory of Interleavers*. Ithaca, New York. N.p., n.d.

European Telecommunications Standards Institute. *DE/JTC-DVB-6, Digital broadcasting systems for television, sound and data services*.