MAX+PLUS II Advanced Synthesis

User Guide

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
http://www.altera.com

Document Version: 1.0
Document Date: April 2003

About this User Guide

This User Guide provides information about how to install and use the MAX+PLUS® II Advanced Synthesis software. Table 1 shows the user guide revision history.

Table 1. User Guide Revision History

Date	Description
April 2003	First release

How to Find Information

- When using this document in PDF format, the Adobe Acrobat Find feature allows you to search the contents of the document. Click the binoculars toolbar icon in the Adobe Acrobat software to open the Find dialog box.
- Bookmarks serve as an additional table of contents.
- Thumbnail icons, which provide miniature previews of each page, provide a link to the pages.
- Numerous links, shown in green text, allow you to jump to related information.

How to Contact Altera

For the most up-to-date information about Altera® products, go to the Altera web site at **www.altera.com**.

For technical support on this product, go to **www.altera.com/mysupport**. For additional information about Altera products, consult the sources shown in Table 2.

Table 2. How to Contact Altera

Information Type	USA & Canada	All Other Locations
Product literature	**www.altera.com**	**www.altera.com**
Altera literature services	**lit_req@altera.com** *(1)*	**lit_req@altera.com** *(1)*
Non-technical customer service	(800) 767-3753	(408) 544-7000 (7:30 a.m. to 5:30 p.m. Pacific Time)
Technical support	(800) 800-EPLD (3753) (7:30 a.m. to 5:30 p.m. Pacific Time)	(408) 544-7000 *(1)* (7:30 a.m. to 5:30 p.m. Pacific Time)
	www.altera.com/mysupport/	**www.altera.com/mysupport/**
FTP site	**ftp.altera.com**	**ftp.altera.com**

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

Typographic Conventions

The ***MAX+PLUS II Advanced Synthesis User Guide*** uses the typographic conventions shown in Table 3.

Table 3. Conventions

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options 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: $\mathbf{f_{MAX}}$, **\maxplus2** directory, **d:** drive, **chiptrip.gdf** file.
Bold italic type	Book titles are shown in bold italic type with initial capital letters. Example: ***1999 Device Data Book***.
Italic Type with Initial Capital Letters	Document titles are shown in italic type with initial capital letters. Example: *AN 75 (High-Speed Board Design)*.
Italic type	Internal timing parameters and variables are shown in italic type. Examples: t_{PIA}, $n + 1$. Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: *<file name>*, *<project name>***.pof** file.

Table 3. Conventions

Visual Cue	Meaning
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`	Signal and port names are shown in lowercase Courier type. Examples: `data1`, `tdi`, `input.` Active-low signals are denoted by suffix n, e.g., `resetn`. Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\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.

Contents

Introduction

Introduction

The MAX+PLUS II Advanced Synthesis Software synthesizes Verilog HDL or VHDL source files to produce an EDIF Input File (**.edf**) netlist that can be compiled in the MAX+PLUS II software. The MAX+PLUS II Advanced Synthesis software is available free of charge to all Altera customers to augment their existing design flow. Altera recommends that customers use the MAX+PLUS II Advanced Synthesis Software (or a third-party synthesis tool) for Verilog HDL and VHDL synthesis in place of the MAX+PLUS II Compiler's VHDL and Verilog HDL synthesis capabilities.

This synthesis tool is specifically developed to complement the MAX+PLUS II software, and is not tested for compatibility with other software from Altera or other vendors.

Design Flow

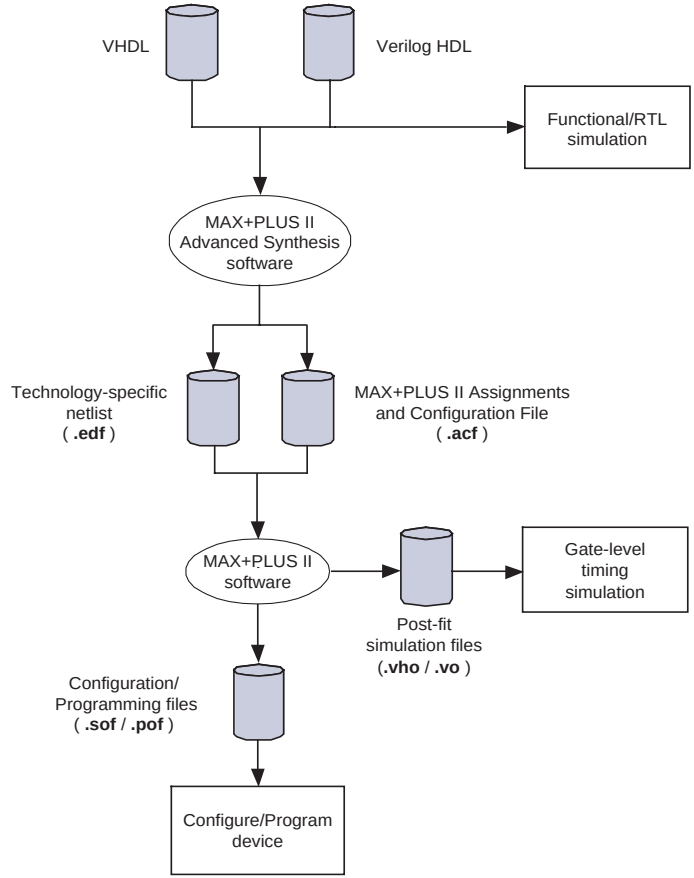
The basic steps in a MAX+PLUS II design flow using the MAX+PLUS II Advanced Synthesis software are as follows:

1. Create Verilog HDL and/or VHDL design files within the MAX+PLUS II software or using a text editor.
2. Create a project containing the HDL design files for synthesis in the MAX+PLUS II Advanced Synthesis software. See "Getting Started" on page 9.
3. Select your target device family and make global synthesis settings. See "Change Device and Family Settings" on page 13.
4. Synthesize the project in the MAX+PLUS II Advanced Synthesis software. See "Start MAX+PLUS II Advanced Synthesis" on page 13.
5. Import the technology-specific EDIF netlist and the Assignment and Configuration File (**.acf**) settings file generated by the MAX+PLUS II Advanced Synthesis software into the MAX+PLUS II software for fitting, performance evaluation, and implementation in an Altera device.
6. Configure or program the Altera device.

See "Using the MAX+PLUS II Advanced Synthesis Software" on page 9 for more information.

Figure 1 shows the design flow between the MAX+PLUS II Advanced Synthesis software and the MAX+PLUS II software.

Figure 1. Recommended Design Flow Note (1)



Note:

(1) See "Black-Boxing" on page 20 for information on how certain design elements are black-boxed by the MAX+PLUS II Advanced Synthesis Software.

Note that simulation may be performed at different steps during the flow. In a third-party simulation tool (such as the ModelSim-Altera software) you may perform a functional or RTL simulation using your HDL files. After compiling your design in the MAX+PLUS II you may perform a gate-level timing simulation using the MAX+PLUS II Simulator. To generate a post-fitting Verilog HDL or VHDL netlist during MAX+PLUS II compilation, select **Verilog Netlist Writer** or **VHDL Netlist Writer** (Interfaces menu) when the MAX+PLUS II Compiler window is open.

You may alos perform a functional or timing simulation in the MAX+PLUS II design software by selecting **Functional SNF Extractor** or **Timing SNF Extractor** (Processing menu) before compilation when the MAX+PLUS II Compiler window is open.

Verilog HDL Support

The MAX+PLUS II Advanced Synthesis software supports the Verilog-1995 (IEEE 1364-1995) standard and also partially supports Verilog-2001 (IEEE 1364-2001) standard. Supported Verilog-2001 constructs include:

- Generate statements: `generate` and `genvar`
- `localparam` constants
- Preprocessor statements such as `` `elsif ``, `` `line ``, `` `ifdef ``, and `` `file ``
- Signed declarations for all variables
- Operators such as `**`, `<<<`, and `>>>`
- Attributes using the syntax `(* name = value *)`
- Indexed part selects using `+:` and `-:`
- Combinational logic sensitivity wild card token `@*`

VHDL Support

The MAX+PLUS II Advanced Synthesis software supports the VHDL 1987 (IEEE 1076-1987) and VHDL 1993 (IEEE 1076-1993) standards.

Standard IEEE and vendor VHDL libraries and packages can be called from VHDL code in your project. The **IEEE** library includes the standard VHDL packages **std_logic_1164**, **numeric_std**, and **numeric_bit**. The **STD** library is part of the VHDL language standard and includes packages **standard** (included in every project by default) and **textio**. For compatibility with older designs, the MAX+PLUS II Advanced Synthesis software also supports vendor-specific packages and libraries, including:

- Synopsys packages such as **std_logic_arith** and **std_logic_unsigned** in the **IEEE** library
- Mentor Graphics® packages such as **std_logic_arith** in the **ARITHMETIC** library
- Altera packages such as **maxplus2**, **altera_mf_components**, and **lpm_components** in the **ALTERA** library

To call a user-defined VHDL package in your design, indicate the library and package name using the `LIBRARY` and `USE` commands. You can use any name for your library, including `work`. To include the VHDL package in your MAX+PLUS II Advanced Synthesis project see "Adding HDL Source Files To a Project" on page 10.

Installing the MAX+PLUS II Advanced Synthesis Software

This section describes the requirements and procedures for installing the MAX+PLUS II Advanced Synthesis software on Windows-based PCs and UNIX workstations.

"Setting Environment Variables" on page 6 gives instructions for adding and editing environment variables to customize the MAX+PLUS II Advanced Synthesis software's interaction with other applications installed on the system you are using, such as your preferred text editor.

You must install the MAX+PLUS II Advanced Synthesis software and the MAX+PLUS II design software on the same system. The system hardware and software requirements for both are identical.

Installing on a PC

To install the MAX+PLUS II Advanced Synthesis software on a PC running Microsoft Windows, download the **max2syn_10_pc.exe** installation program. Double-click on the icon for **max2syn_10_pc.exe** and follow the on-screen instructions.

Installing on a UNIX System

To install the MAX+PLUS II Advanced Synthesis software on a UNIX workstation perform the following steps:

1. Make a note of the path to the MAX+PLUS II design software executables as well as the path to the directory where you would like the MAX+PLUS II Advanced Synthesis software to be installed.

2. Download the **max2syn_10_solaris.tar** file and uncompress it by typing the following:

   ```
   tar -xvf max2syn_10_solaris.tar ↵
   ```

3. Change to the directory created by uncompressing the downloaded archive.

4. To run the install script, type

   ```
   ./install ↵
   ```

   at the UNIX command prompt and follow the on-screen instructions. When prompted, provide the path to the MAX+PLUS II design software executables as well as the path to the directory where you would like the MAX+PLUS II Advanced Synthesis software to be installed.

   ☞ If the specified destination directory for the MAX+PLUS II Synthesis software does not yet exist, the installation script creates it for you.

5. Add the directory containing <*installed directory*>`/bin/max2syn` to your PATH environment variable. (See "Setting Environment Variables" below for instructions.)

Setting Environment Variables

To make full use of the MAX+PLUS II Advanced Synthesis software, certain environment variables should be set so that the MAX+PLUS II Advanced Synthesis software can work smoothly with the MAX+PLUS II design software and your text editor of choice. These variables are set during installation, but if you decide to use a different editor or if other conditions change, use one of the following procedures, depending on your operating system, to change environment variable values.

☞ The procedures below describe setting the `EDITOR` environment variable, but the procedure for setting other environment variables is identical. In every case the path to a certain executable file is the value for a given variable.

Windows NT

Set the `EDITOR` environment variable in Windows NT by performing the following steps:

1. Choose **Settings > Control Panel** (Start menu).
2. Double-click **System**.
3. Click the **Environment** tab in the System Properties dialog box.
4. Click the **System Variable** list to highlight it, and then type EDITOR in the **Variable** box.

5. Type the full path to your preferred text editor (such as `C:\WINNT\NOTEPAD.EXE`) in the **Value** box.

6. Click **Set**, and then click **OK**.

Windows 2000

Set the `EDITOR` environment variable in Windows 2000 by performing the following steps:

1. Choose **Settings > Control Panel** (Start menu).

2. Double-click **System**.

3. Click the **Advanced** tab in the System Properties dialog box.

4. Click **Environment Variables**.

5. Click the System Variable list to highlight it, and then click **New**.

6. Type `EDITOR` in the Variable box.

7. Type the full path to your preferred text editor (such as `C:\WINNT\NOTEPAD.EXE`) in the **Value** box.

Windows XP

Set the `EDITOR` environment variable through the Windows system control panel by performing the following steps:

1. Choose **Control Panel** (Windows Start menu).

2. Click **Switch to Classic View**.

3. Double-click **System** (Control Panel window).

4. Click the **Advanced** tab (System Properties dialog box).

5. Click **Environment Variables**.

6. Click the **System Variable** list to highlight it, and then click **New**.

7. Type EDITOR in the **Variable** box.

8. Type the full path to your preferred text editor (such as `C:\windows\system32\notepad.exe`) in the **Value** box.

Windows 98

Set the EDITOR environment variable in the **autoexec.bat** file by performing the following steps:

1. Open the **autoexec.bat** file with a text editor such as Windows Notepad. This file is found in your **C:** directory. If this file does not exist, create the file and save it in the **C:** directory.

2. Type the environment variable and the value on their own line in the **autoexec.bat** file as follows:

   `set EDITOR=`*<path to editor>* ↵

   where the path to the editor is `C:\windows\notepad.exe` or similar.

3. Save the **autoexec.bat** file. Make sure your text editor does not append **.txt** or other extension to the filename (such as **autoexec.bat.txt**). If in doubt, verify the filename at a command prompt.

4. Restart the PC.

UNIX (Solaris) Workstation

Set the EDITOR environment variable in the shell resource file (such as **.cshrc**) located in the user's home directory by doing the following:

This is an example of the steps required for a C shell dot file (.cshrc). This procedure may be slightly different for other UNIX shells.

1. Enter this environment variable on a separate line in the **.cshrc** file to specify the license location:

   `setenv EDITOR` *<path to editor>* ↵

   Where the path to the editor is `/usr/local` or similar.

2. Save the **.cshrc** file.

3. Type the following commands at the shell (command) prompt:

   `cd` ↵

   `source .cshrc` ↵

Using the MAX+PLUS II Advanced Synthesis Software

Introduction

The MAX+PLUS II Advanced Synthesis software can be controlled using both a graphical user interface (GUI) and a command line, and can interact with the MAX+PLUS II design software in different ways, depending on your preferred workflow. This section will describe the creation and synthesis of a project from VHDL or Verilog HDL source code.

If you haven't already done so, make sure that MAXPLUS2_ROOT environment variable is set to the directory for the MAX+PLUS II executables, and that the directories containing the MAX+PLUS II and MAX+PLUS II Advanced Synthesis executables are specified in the PATH environment variable. See "Setting Environment Variables" on page 6 for more information.

Getting Started

The MAX+PLUS II Advanced Synthesis software organizes and associates source files, report files, and other synthesis output with a project (**.max2syn**) file. This MAX2SYN file represents the project that is referred to in the descriptions below.

When using the MAX+PLUS II Advanced Synthesis software from the command line, a MAX2SYN file is created by default when source files are first synthesized, named after the top-level entity in the design. See "Using the MAX+PLUS II Advanced Synthesis Software at the Command Line" on page 20 for more information.

Adding HDL Source Files To a Project

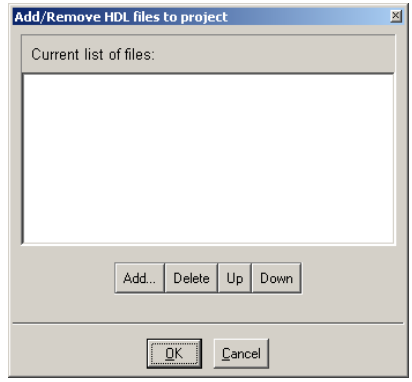
To add HDL source files to a project using the MAX+PLUS II Advanced Synthesis software GUI, perform the following steps:

1. Choose **Programs > Altera > MAX+PLUS II Advanced Synthesis** (Start menu) to start the MAX+PLUS II Advanced Synthesis software GUI.

2. Create a MAX2SYN file using the MAX+PLUS II Advanced Synthesis software GUI by selecting **New Project** (File menu).

3. After creating a project, select **Add/Remove HDL File(s)** (Assign menu). The **Add/Remove HDL Files to Project** dialog box is displayed.

Figure 1. Add/Remove HDL Files to Project Dialog Box



4. In the **Add/Remove HDL Files to Project** dialog box, click **Add** and specify the files you wish to add to the project.

   When adding VHDL files to a project, make sure taht packages are listed first, so that they are synthesized in sequence before the design files that incorporate them.

5. After you have added all the files for your project, click **OK**.

Choosing a Target Device Famiily and Other Options

Before Synthesizing the design select a target device family and make basic global synthesis settings by clicking the **Family and Synthesis Settings** icon or selecting **Family and Synthesis Settings** (Assign menu). See "Changing Settings" on page 14 for more information.

Synthesizing the Project

To synthesize the design, after adding HDL source files to your project, click **Start MAX+PLUS II Advanced Synthesis**. The status of the synthesis process, including any errors, will be reported in the Message Log window below the toolbar.

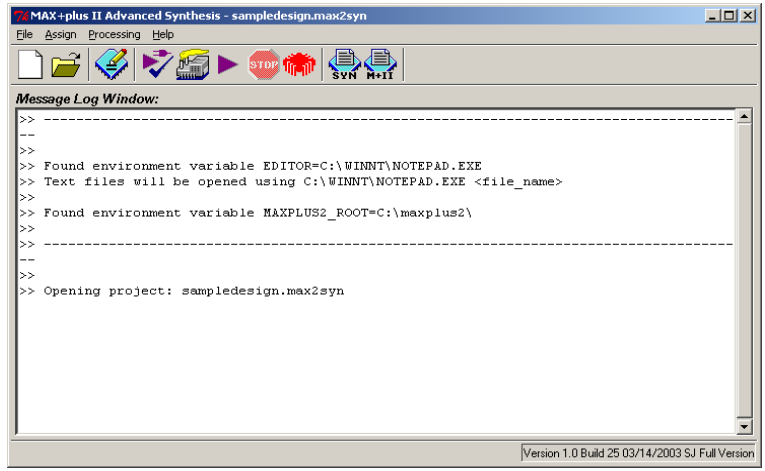
Compiling the Design in the MAX+PLUS II Design Software

After Synthesis, you may compile the project automatically from within the MAX+PLUS II Advanced Synthesis software or you may perform a compilation in the MAX+PLUS II design software manually using the EDIF and ACF files that are the output from synthesis. See "The MAX+PLUS II Advanced Synthesis Window" on page 11 for a description of the different options you can use to compile your project from within the MAX+PLUS II design software.

The MAX+PLUS II Advanced Synthesis Window

When you choose **Programs > Altera > MAX+PLUS II Advanced Synthesis** (Start menu), the MAX+PLUS II Advanced Synthesis window appears (Figure 2).

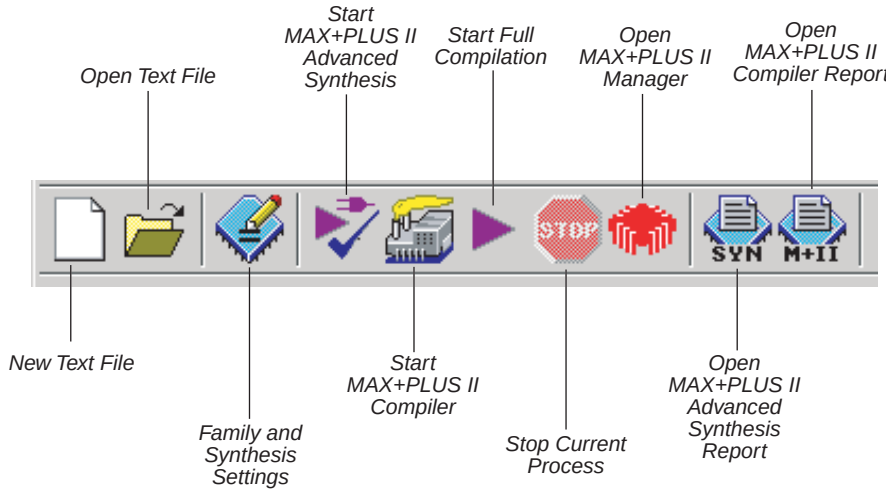
Figure 2. MAX+PLUS II Advanced Synthesis Window



The MAX+PLUS II Advanced Synthesis software GUI can also be started by typing `max2synw -g` ↵ at the Windows command prompt.

Figure 3 shows a close-up view of the toolbar.

Figure 3. MAX+PLUS II Advanced Synthesis Window Toolbar



New Text File

Click this button or choose **New** (File menu) to open a new document in the text editor specified in the EDITOR environment variable. See "Setting Environment Variables" on page 6 for information on how to specify a text editor.

Open Text File

Click this button or choose **Open** (File menu) to specify a text file that will then be opened using the text editor specified in the EDITOR environment variable. See "Setting Environment Variables" on page 6 for information on how to specify a text editor.

Change Device and Family Settings

Click this button or choose **Device and Family Settings** (Assign menu) to display the **Device and Family Settings** dialog box. See "Changing Settings" on page 14 for more information.

Start MAX+PLUS II Advanced Synthesis

Click this button or choose **Start MAX+PLUS II Advanced Synthesis** (Processing menu) to synthesize the source files associated with the currently open synthesis project.

Start MAX+PLUS II Compiler

Click this button or choose **Start MAX+PLUS II Compiler** (Processing menu) to start compilation of a previously synthesized project.

Start Full Compilation

Click this button or choose **Start Full Compilation** (Processing menu) to synthesize a project using the MAX+PLUS II Advanced Synthesis software and automatically compile the design using the MAX+PLUS II Compiler.

Stop Current Process

Click this button or choose **Stop Process** (Processing menu) to stop the current synthesis or compilation process.

Open MAX+PLUS II Manager

Click this button or choose **Open MAX+PLUS II Manager** (Processing menu) to start the MAX+PLUS II software GUI and set the current project to the newly generated EDIF file. No synthesis or compilation takes place.

Open MAX+PLUS II Advanced Synthesis Report

Click this button or choose **Open MAX+PLUS II Advanced Synthesis Report** (Processing menu) to open the synthesis report file (**.syn.rpt**). The SYN RPT file contains settings information as well as any error or warning messages that are reported during synthesis.

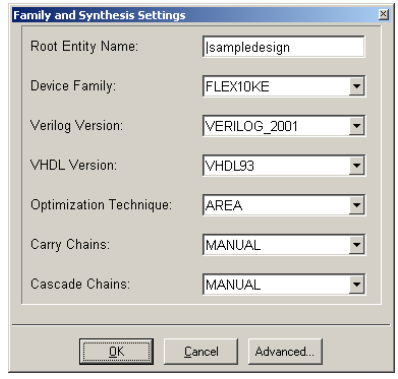
Open MAX+PLUS II Compiler Report

Click this button or choose **Open MAX+PLUS II Compiler Report** (Processing menu) to open the MAX+PLUS II Report File (**.rpt**). The RPT file contains information shows how the project is implemented in one or more devices, including partitioning, resource usage, inputs, outputs, and project timing. The RPT file is generated by the Fitter module of the MAX+PLUS II Compiler.

Changing Settings

Clicking **Change Device and Family Settings** in the toolbar (Figure 3) or selecting **Change Device and Family Settings** (Assign menu) causes the **Family and Synthesis Settings** dialog box to appear (Figure 4).

Figure 4. Device Family and Synthesis Settings Dialog Box



Root Entity Name

Specify the root or top-level entity of your design in the **Root Entity Name** box.

Device Family

In the **Device Family** list, select the device family targeted by your design. The MAX+PLUS II Advanced Synthesis software supports the following Altera device families:

- ACEX® 1K
- FLEX® 6000
- FLEX 8000
- FLEX 10K®
- FLEX 10KA
- FLEX 10KB
- FLEX 10KE
- MAX® 3000A
- MAX 7000
- MAX 7000AE
- MAX 7000B
- MAX 7000S
- MAX 9000

Verilog Version

In the **Verilog Version** box, select the Verilog HDL standard used in your design. The MAX+PLUS II Advanced Synthesis software supports the Verilog-1995 and Verilog-2001 standards. Verilog-2001 is the default setting.

For more information on Verilog HDL support in the MAX+PLUS II Advanced Synthesis software see "Verilog HDL Support" on page 3.

VHDL Version

In the **VHDL Version** box, select the standard of VHDL used in your design. The MAX+PLUS II Advanced Synthesis software supports the VHDL 1987 and VHDL 1993 standards. VHDL 1993 is the default setting.

For more information on Verilog HDL support in the MAX+PLUS II Advanced Synthesis software see "VHDL Support" on page 3.

Optimization Technique

The Optimization Technique logic option specifies the overall goal for logic optimization, i.e., whether to attempt to achieve maximum speed performance or minimum area usage during compilation. Table 1 lists the settings for this option.

Table 1. Optimization Technique Settings

Setting	Description
AREA	The MAX+PLUS II Advanced Synthesis software makes the design as small as possible to minimize resource usage
SPEED	The MAX+PLUS II Advanced Synthesis software chooses a design implementation that has the fastest f_{MAX}

The default setting varies by target device family, and is generally optimized to get the best area / speed trade-off. Results are design and device-dependent and can vary depending on which design or device family you use.

Carry Chains

Select an option in the **Carry Chains** box to specify how CARRY primitives will be handled during synthesis. Table 2 lists the settings for this option.

Table 2. Carry Chain Settings

Setting	Description
IGNORE	The MAX+PLUS II Advanced Synthesis software converts all CARRY buffers into wires
AUTO	The MAX+PLUS II Advanced Synthesis software creates carry chains automatically by inserting CARRY buffers into the design
MANUAL	The MAX+PLUS II Advanced Synthesis software uses only the CARRY primitives that have been manually entered in design files

Cascade Chains

Select an option in the **Cascade Chains** box to specify how CASCADE primitives will be handled during synthesis. Table 3 lists the settings for this option.

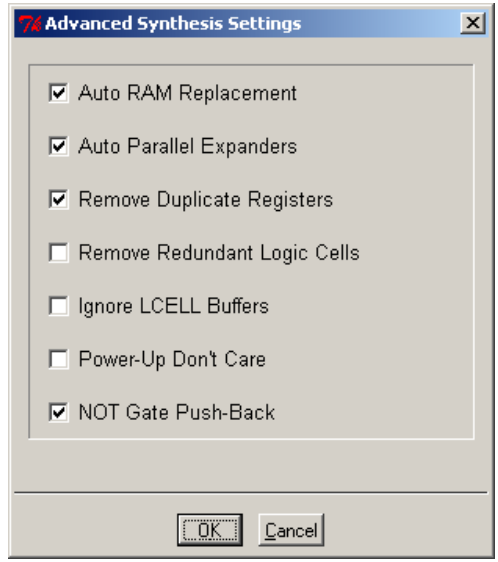
Table 3. Cascade Chain Settings

Setting	Description
IGNORE	The MAX+PLUS II Advanced Synthesis software converts CASCADE buffers into `wires`
AUTO	The MAX+PLUS II Advanced Synthesis software inserts CASCADE buffers automatically
MANUAL	The MAX+PLUS II Advanced Synthesis software uses only the CASCADE primitives that have been manually entered in design files

Advanced Synthesis Settings

Clicking **Advanced** in the **Family and Synthesis Settings** dialog box displays the **Advanced Synthesis Settings** dialog box (Figure 5).

Figure 5. Advanced Synthesis Settings Dialog Box



Auto RAM Replacement

Turning on **Auto RAM Replacement** allows the MAX+PLUS II Advanced Synthesis software to find registers and logic that can be replaced with the `lpm_ram_dp` megafunction. This option is useful for finding areas of the design that can be implemented more efficiently.

☞ This option is available for the ACEX, and FLEX device families only.

☞ Turning on this option may change the functionality of your design in certain situations.

Auto Parallel Expanders

Turning on **Auto Parallel Expanders** allows the MAX+PLUS II Advanced Synthesis software to automatically create chains of parallel expanders in the macrocells of MAX devices.

☞ This option is not supported in ACEX 1K or FLEX devices.

Remove Duplicate Registers

Turning on **Remove Duplicate Registers** causes the MAX+PLUS II Advanced Synthesis software to remove a register if it is identical to another register. If two registers generate the same logic, the second one will be deleted and the first one will be made to fan-out to the second one's destinations. If the deleted register has different logic option assignments, they will be ignored.

Remove Redundant Logic Cells

Turning on **Remove Redundant Logic Cells** causes the MAX+PLUS II Advanced Synthesis software to remove redundant LCELL primitives or WYSIWYG primitives.

Ignore LCELL Buffers

Turning on **Ignore LCELL Buffers** causes the MAX+PLUS II Advanced Synthesis software to ignore LCELL buffers that are instantiated in the design.

Power-Up Don't Care

Turning on **Power-Up Don't Care** causes registers to power up with a "don't care" logic level (X), or the logic level most appropriate for the design. You might use this option to allow the MAX+PLUS II Advanced Synthesis software to change the power-up level of a register to minimize your design's area usage.

For example, a register may have its D input tied to V_{CC}. If you turn this option off, the register powers up low even though it goes high at the first clock signal. If you turn this option on, the MAX+PLUS II Advanced Synthesis software sets the power-up value of the register to high and, therefore, can eliminate the register and connect the output of the register to V_{CC}. If the MAX+PLUS II Advanced Synthesis software performs this type of optimization, you will receive a message indicating that it is doing so.

NOT Gate Push Back

Turning on **NOT Gate Push Back** causes the MAX+PLUS II Advanced Synthesis software to push an inversion (a `NOT` gate) back through a register and implement it on that register's data input if it is necessary to implement the design. If this option is turned on, a register may power-up to an active-high state, needing to be explicitly cleared during initial operation of the device.

Using the MAX+PLUS II Advanced Synthesis Software at the Command Line

Use the **max2syn** command to invoke the MAX+PLUS II Advanced Synthesis software from a command prompt. In its simple form, this command is entered as shown below:

`max2syn` <*HDL source file*>`.v` ↵

In this case the module name would be the same as the file name (without the **.v** file-type suffix.) To synthesize a design hierarchy contained in several files, or when the module name does not match the file name, type

`max2syn` <*HDL source file 1*>`.v --source=`<*HDL source file 2*>.v
`--source=`<*HDL source file 3*>.v

Type `max2syn --help` ↵ at a command prompt for more information on syntax and usage of the MAX+PLUS II Advanced Synthesis software as a command-line utility.

Black-Boxing

Design files for Altera megafunctions (including MegaWizard-generated files), as well as non-Verilog/VHDL design components, can be read by the MAX+PLUS II Advanced Synthesis software, and the megafunction or design file will be black-boxed automatically by the tool. You will see a message when this occurs. The megafunction will be synthesized during MAX+PLUS II compilation. This is similar to the methodology when using third-party synthesis tools, but no black-boxing attributes or files are required. Be sure not to enclose the code in the `translate_off` and `translate_on` compiler directives. Using these directives will cause the MAX+PLUS II Advanced Synthesis software to ignore the code. See "Translate Off & On" on page 21 for more information.

Altera Hardware Description Language (AHDL) files (**.tdf**) and Graphic Design Files (**.gdf**) are not synthesized by the MAX+PLUS II Advanced Synthesis software. If you have these types of design files instantiated in your design, the MAX+PLUS II Advanced Synthesis software will automatically black-box the files, and a message will be reported. The AHDL or GDF file is synthesized during MAX+PLUS II compilation.

Note that any sub-designs used as part of a GDF or TDF will be black-boxed as a part of the GDF or TDF.

Compiler Directives and Attributes

The MAX+PLUS II Advanced Synthesis software provides a number of compiler directives and attributes to guide the synthesis process. This section defines compiler directives and attributes.

Compiler Directives

The MAX+PLUS II Advanced Synthesis software supports compiler directives, also called pragmas. You include compiler directives in Verilog HDL or VHDL code as comments. These directives are not Verilog HDL or VHDL commands; however, synthesis tools use them to drive the synthesis process in a particular manner. Other tools such as simulators ignore these directives and treat them as comments.

You can enter compiler directives in your code using the following syntax, where *directive* and *value* are variables and the entry within brackets is optional.

Verilog HDL:

```
// synthesis <directive> { <value> }
or
/* synthesis <directive> { <value> } */
```

VHDL:

```
-- synthesis <directive> { <value> }
```

In addition to the `synthesis` keyword shown above, the keywords `pragma`, `synopsys`, and `exemplar` are supported in both Verilog HDL and VHDL for compatibility with other synthesis tools.

Translate Off & On

The `translate_off` and `translate_on` compiler directives indicate whether the MAX+PLUS II Advanced Synthesis software or a third-party synthesis tool should compile a portion of HDL code that is not relevant for synthesis. The `translate_off` directive marks the beginning of code that the synthesis tool should ignore; the `translate_on` directive indicates that synthesis should resume. A common use of these directives is to indicate a portion of code that is intended for simulation only. The synthesis tool reads synthesis-specific directives and processes them during synthesis; however, third-party simulation tools read the directives as comments and ignore them. Figures 6 and 7 show examples of these directives.

Figure 6. Verilog HDL Example of Translate On & Off

```
// synthesis translate_off
parameter       tpd = 2;    // Delay for simulation

#tpd;
// synthesis translate_on
```

Figure 7. VHDL Example of Translate On & Off

```
-- synthesis translate_off
use std.textio.all;
-- synthesis translate_on
```

Read Comments as HDL

The `read_comments_as_HDL` compiler directive indicates that the MAX+PLUS II Advanced Synthesis software should synthesize a portion of HDL code that is commented out. This directive allows you to comment out portions of HDL source code that are not relevant for simulation, while instructing the MAX+PLUS II Advanced Synthesis software to read and synthesize that same source code. Setting the `read_comments_as_HDL` directive to `on` marks the beginning of commented code that the synthesis tool should read; setting the `read_comments_as_HDL` directive to `off` indicates the end of the code.

You can use the directive with `translate_off` and `translate_on` to create one HDL source file that includes both a megafunction instantiation for synthesis and a behavioral description for simulation.

In Figures 8 and 9, the commented code enclosed by `read_comments_as_HDL` is visible to the MAX+PLUS II Advanced Synthesis software and is synthesized.

Compiler directives are case-sensitive in Verilog HDL; you must match the case of the directive as shown in Figure 8

Figure 8. Verilog HDL Example of Read Comments as HDL

```
// synthesis read_comments_as_HDL on
// my_rom lpm_rom (.address (address),
```

```
//  .data      (data));
// synthesis read_comments_as_HDL off
```

Figure 9. VHDL Example of Read Comments as HDL

```
-- synthesis read_comments_as_HDL on
-- my_rom : entity lpm_rom
--   port map (
--             address => address,
--             data    => data,      );
-- synthesis read_comments_as_HDL off
```

Attributes

The MAX+PLUS II Advanced Synthesis software supports attributes, also known as pragmas or directives. Attributes are similar to compiler directives in that they drive the synthesis process; however, attributes always apply to a specific design element.

The Verilog-2001 HDL and VHDL language definitions provide specific syntax for specifying attributes. However in Verilog-1995 HDL you must use comments similar to compiler directives. You can enter attributes in your code using the following syntax, where *attribute*, *attribute type*, *value*, *object*, and *object type* are variables and the entry within brackets is optional.

Verilog-1995 HDL:

```
// synthesis <attribute> { <value> }
```
or
```
/* synthesis <attribute> { <value> }  */
```

☞ You cannot use the open one-line comment in Verilog HDL when a semicolon is required at the end of the line because it is not clear to which HDL element the attribute applies.

☞ In addition to the `synthesis` keyword shown above, the keywords `pragma`, `synopsys`, and `exemplar` are supported for compatibility with other synthesis tools.

Verilog-2001 HDL:

```
(* <attribute> { = <value> }  *)
```

VHDL:

```
attribute <attribute> : <attribute type> ;
attribute <attribute> of <object> : <object type> is <value> ;
```

The examples in the following sections demonstrate each syntax form.

Full Case

The `full_case` attribute indicates that a Verilog HDL case statement should be considered full, that is, either all possible binary combinations of cases are specified or a default case is present. A full case attribute attached to a `case` statement header that is not full forces the unspecified states to be treated as logic "don't care" values. Using this attribute on a `case` statement that is not full avoids latch inference. VHDL case statements must be full, so the attribute does not apply.

The case statement in Figure 10 is not full because not all binary values for `sel` are specified. Because the `full_case` attribute is used, synthesis treats the output as "don't care" when the `sel` input is `2'b11`.

Figure 10. Sample Verilog HDL Code with a full_case Attribute

```
module full_case (a, sel, y);
    input [3:0] a;
    input [1:0] sel;
    output y;
    reg y;

    always @(a or sel)
    case (sel) // synthesis full_case
       2'b00: y=a[0];
       2'b01: y=a[1];
       2'b10: y=a[2];
    endcase
endmodule
```

Verilog-2001 syntax also accepts the following statements in the `case` header instead of the comment form as shown in Figure 10:

```
(* full_case *) case (sel)
```

When using the `full_case` attribute, there is a potential cause for simulation-mismatch between third-party Verilog HDL functional and post-MAX+PLUS II Fitter simulation because unknown `case` statement cases may still function like latches during functional simulation. For example, this mismatch occurs if a `full_case` attribute is specified, affecting synthesized output. In this example, when `sel` is `2'b11,` a functional HDL simulation output behaves like a latch while the post MAX+PLUS II simulation output behaves like "don't care."

Parallel Case

The `parallel_case` attribute indicates that a Verilog HDL case statement should be considered parallel, that is, only one case item can be matched at a time. A parallel case attribute attached to a `case` statement header forces the synthesis tool to generate multiplexer logic instead of a priority encoder. VHDL `case` statements have no overlap in any of the case items, so they are always parallel and this attribute does not apply.

Use this attribute only when the `case` statement is truly parallel and you want to use one-hot style encoding. If you use the attribute in any other situation, the generated logic will not match the functional HDL simulation.

The example in Figure 11 shows a `casez` statement with non-parallel cases. Functional HDL simulation behaves as a priority encoder with bits of `sel` having high to low priority order, `sel[2]`, `sel[1]`, and `sel[0]`. However, the synthesized version of this design may simulate differently because it will be implemented with multiplexer logic.

Figure 11. Sample Verilog HDL Code with a parallel_case Attribute

```
module parallel_case (sel, a, b, c);
    input [2:0] sel;
    output a, b, c;
    reg a, b, c;

    always @(sel)
    begin
       {a, b, c} = 3'b0;
       casez (sel) // synthesis parallel_case
          3'b1??: a = 1'b1;
          3'b?1?: b = 1'b1;
          3'b??1: c = 1'b1;
       endcase
    end
endmodule
```

Verilog-2001 HDL syntax also accepts the following statements in the `case` header instead of the comment form as shown in Figure 11:

```
(* parallel_case *) casez (sel)
```

Preserve Registers

The `preserve` attribute directs the MAX+PLUS II Advanced Synthesis software not to minimize or remove a specified register during synthesis optimization. Optimizations can eliminate redundant registers and registers with constant drivers. This option can preserve a register so you can observe it during simulation. Additionally, it can preserve registers if you are creating a preliminary version of the design in which secondary signals are not specified. The option cannot preserve registers that have no fan-out.

You can set the `preserve` attribute in your HDL code as shown below. In this example, the `my_reg` register is preserved.

In addition to `preserve`, the MAX+PLUS II Advanced Synthesis software supports the `syn_preserve` attribute name for compatibility with other synthesis tools.

Verilog HDL:

```
reg my_reg /* synthesis preserve */;
```

Verilog-2001:

```
(* preserve *) reg my_reg;
```

VHDL:

```
signal my_reg : stdlogic;

attribute preserve : boolean;
attribute preserve of my_reg : signal is true;
```

Notes: