Section

2

MAX+PLUS II — A Perspective

This section gives an overview of MAX+PLUS II and describes all MAX+PLUS II applications.

Go to MAX+PLUS II Help for complete and up-to-date information on all MAX+PLUS II topics.

MAX+PLUS II Logic Design

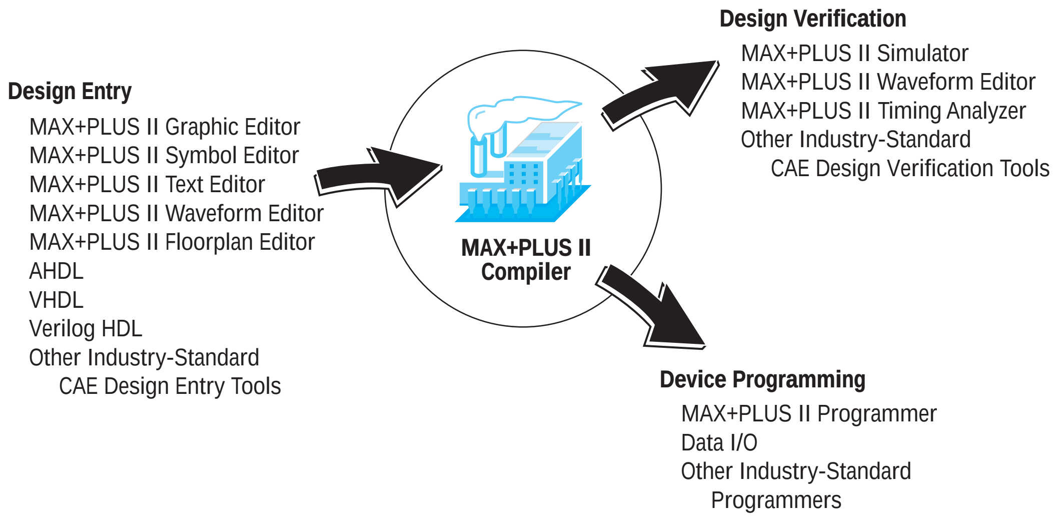
The Altera Multiple Array MatriX Programmable Logic User System (MAX+PLUS II) provides a multi-platform, architecture-independent design environment that easily adapts to your specific design needs. MAX+PLUS II offers easy design entry, quick processing, and straightforward device programming.

MAX+PLUS II development software, shown in Figure 2-1, is a fully integrated package for creating logic designs for Altera programmable logic devices—including the Classic, MAX 5000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000, and FLEX 10K families of devices.

☞ Refer to the MAX+PLUS II **read.me** file for information on other supported Altera device families.

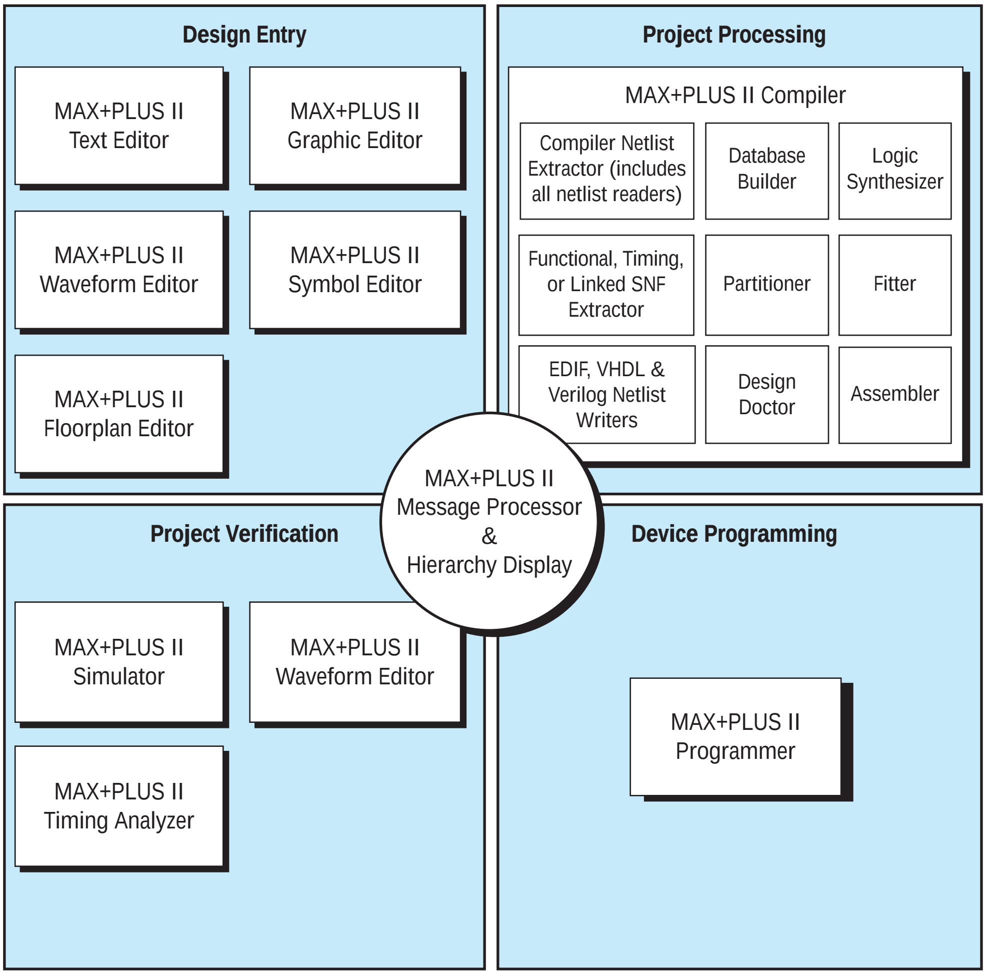
MAX+PLUS II offers a full spectrum of logic design capabilities: a variety of design entry methods for hierarchical designs, powerful logic synthesis, timing-driven compilation, partitioning, functional and timing simulation, linked multi-device simulation, timing analysis, automatic error location, and device programming and verification. MAX+PLUS II both reads *and* writes Altera Hardware Description Language (AHDL) files and standard EDIF netlist files, Verilog HDL files, VHDL files, and OrCAD schematic files. In addition, MAX+PLUS II reads Xilinx netlist files and writes Standard Delay Format (SDF) files for a convenient interface to other industry-standard CAE software.

Figure 2-1. MAX+PLUS II Design Environment


Design Entry
MAX+PLUS II Graphic Editor
MAX+PLUS II Symbol Editor
MAX+PLUS II Text Editor
MAX+PLUS II Waveform Editor
MAX+PLUS II Floorplan Editor
AHDL
VHDL
Verilog HDL
Other Industry-Standard CAE Design Entry Tools
MAX+PLUS II Compiler
Design Verification
MAX+PLUS II Simulator
MAX+PLUS II Waveform Editor
MAX+PLUS II Timing Analyzer
Other Industry-Standard CAE Design Verification Tools
Device Programming
MAX+PLUS II Programmer
Data I/O
Other Industry-Standard Programmers

MAX+PLUS II offers a rich graphical user interface complemented with an illustrated, easy-to-use on-line help system. The complete MAX+PLUS II system includes 11 fully integrated applications that take you through every step of creating a design. (A logic design, including all subdesigns, is called a "project" in MAX+PLUS II.) See Figure 2-2.

Figure 2-2. MAX+PLUS II Applications


Design Entry
MAX+PLUS II Text Editor
MAX+PLUS II Graphic Editor
MAX+PLUS II Waveform Editor
MAX+PLUS II Symbol Editor
MAX+PLUS II Floorplan Editor
Project Processing
MAX+PLUS II Compiler
Compiler Netlist Extractor (includes all netlist readers)
Database Builder
Logic Synthesizer
Functional, Timing, or Linked SNF Extractor
Partitioner
Fitter
EDIF, VHDL & Verilog Netlist Writers
Design Doctor
Assembler
MAX+PLUS II Message Processor & Hierarchy Display
Project Verification
MAX+PLUS II Simulator
MAX+PLUS II Waveform Editor
MAX+PLUS II Timing Analyzer
Device Programming
MAX+PLUS II Programmer

Many features and commands—such as opening files; entering device, pin, and logic cell assignments; and compiling the current project—are shared by many or all MAX+PLUS II applications, so that learning to use one application gives you a head start on learning to use the others. The design editors (the Graphic, Text, and Waveform Editors) and auxiliary editors (the Floorplan and Symbol Editors) also share numerous features. Each design editor allows you to perform similar tasks—such as finding a signal or symbol—in the same way. You can easily combine different types of design files in a hierarchical project, choosing the design entry format that works best for each functional block. A large library of Altera-supplied megafunctions and macrofunctions, including functions from the Library of Parameterized Modules (LPM), provide a wide range of design entry options.

You can work with different MAX+PLUS II applications simultaneously. For example, you can open multiple design files and transfer information between them while compiling or simulating another project; or, you can view an entire project hierarchy and move smoothly from one hierarchical level to another, while MAX+PLUS II automatically starts the appropriate design editor for each file.

The MAX+PLUS II Compiler lies at the heart of the MAX+PLUS II system, providing powerful project processing that you can customize to achieve the best possible silicon implementation of your project. Automatic error location and extensive documentation on error and warning messages make design modifications quick and easy. You can create output files in a variety of formats for functional, timing, and linked multi-device simulation; timing analysis; and device programming. At every step of the design process, MAX+PLUS II makes it easy for you to focus on your project—not on how to use the software.

The superb integration of the MAX+PLUS II software helps you maximize your efficiency and productivity, putting you in control of your logic design environment.

The Design Flow

The process of taking a new project from conception to completion can be simplified as follows:

1. Create a new design file or a hierarchy of multiple design files in any combination of the MAX+PLUS II design editors, i.e., the Graphic, Text, and Waveform Editors.

2. Specify the top-level design file name as the project name.

3. Assign a device family for the project. You can either allow the Compiler to select a device for you or assign a specific device.

4. Open the MAX+PLUS II Compiler window and choose the **Start** button to compile the project. If you wish, you can turn on the Timing SNF Extractor module to create a netlist file for timing simulation and timing analysis.

5. If the project compiles successfully, you can optionally perform a simulation and timing analysis:

   - To run a timing analysis, open the MAX+PLUS II Timing Analyzer window, select an analysis mode, and choose the **Start** button.
   - To run a simulation, you must first create vector inputs in a Simulator Channel File (**.scf**) in the Waveform Editor or in a Vector File (**.vec**) in the Text Editor. Then, open the MAX+PLUS II Simulator window and choose the **Start** button.

6. Open the MAX+PLUS II Programmer window and either insert a device into a programming adapter on the Master Programming Unit (MPU) or connect the BitBlaster, ByteBlaster, or FLEX Download Cable to a device that is mounted in-system.

7. Choose the **Program** button to program an EPROM- or EEPROM-based device, or choose the **Configure** button to configure an SRAM-based device.

Starting MAX+PLUS II

You can start MAX+PLUS II in one of two ways:

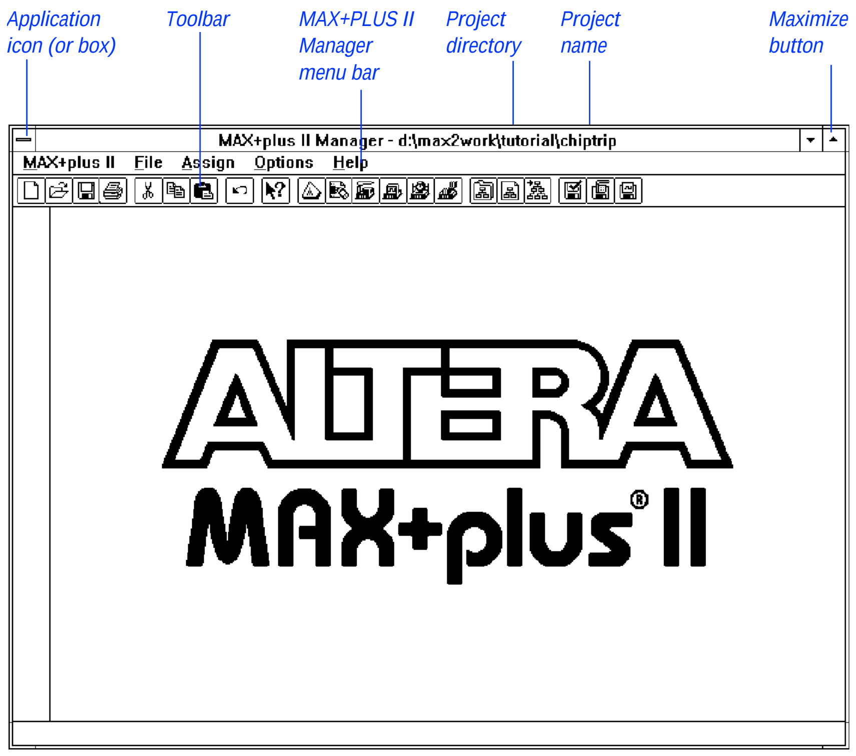
✓ Double-click Button 1 (the left mouse button) on the MAX+PLUS II icon. On a PC running Windows, this icon appears in the MAX+PLUS II program group.

or:

✓ Type `maxplus2` ↵ at the command line.

The MAX+PLUS II Manager window opens. See Figure 2-3.

Figure 2-3. MAX+PLUS II Manager Window

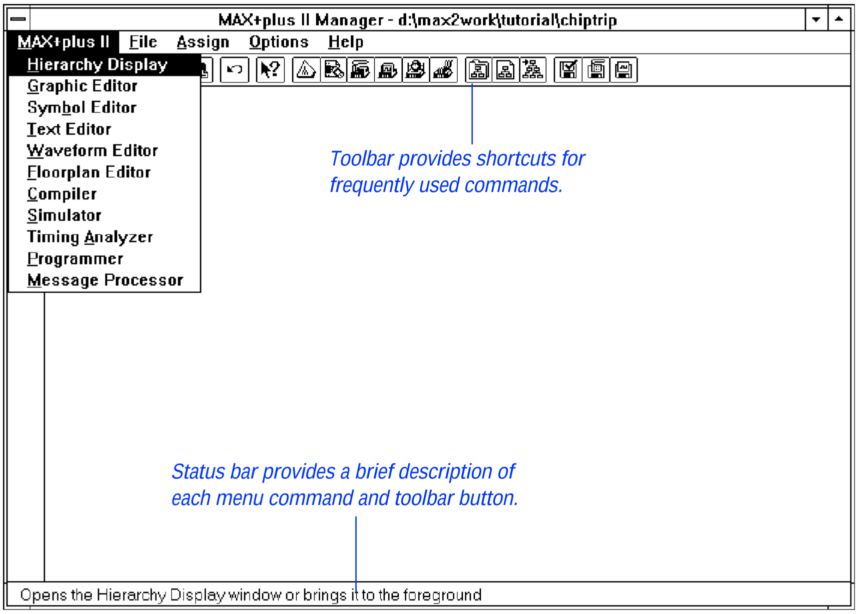

Application icon (or box)
Toolbar
MAX+PLUS II Manager menu bar
Project directory
Project name
Maximize button
MAX+plus II Manager - d:\max2work\tutorial\chiptrip
MAX+plus II
File
Assign
Options
Help
ALTERA
MAX+plus II

☞ If you have not entered an authorization code or specified a network licensing file for MAX+PLUS II, the **Authorization Code** dialog box (Options menu) opens automatically. Go to "Specifying the Authorization Code for a Software Guard Installation" on page 48 or "Specifying the License File for a License File Installation" on page 49 for instructions on how to enter your authorization code or specify your network licensing file.

The MAX+PLUS II Manager

The MAX+PLUS II Manager window opens automatically when you start MAX+PLUS II. From the MAX+PLUS II menu, you can open all other MAX+PLUS II applications. See Figure 2-4.

Figure 2-4. MAX+PLUS II Menu in the MAX+PLUS II Manager Window


MAX+plus II Manager - d:\max2work\tutorial\chiptrip
MAX+plus II
File
Assign
Options
Help
Hierarchy Display
Graphic Editor
Symbol Editor
Text Editor
Waveform Editor
Floorplan Editor
Compiler
Simulator
Timing Analyzer
Programmer
Message Processor
Toolbar provides shortcuts for frequently used commands.
Status bar provides a brief description of each menu command and toolbar button.
Opens the Hierarchy Display window or brings it to the foreground

Commands available from MAX+PLUS II Manager menus are also available in all other MAX+PLUS II applications. For example, these common functions allow you to open a file, compile and simulate the current project, or switch to a different project. You can specify libraries of your custom symbol and design files, archive backup copies of all files in the current project in a separate directory, customize the color scheme, and enter a new authorization code. You can also show or hide the toolbar and status bar, and open MAX+PLUS II Help from the Help menu.

In addition, you can enter, edit, and delete the types of resource, device, and parameter assignments that control project compilation, including logic synthesis, partitioning, and fitting. These functions are available regardless of whether any project design file or application window is open. For information on these functions, go to "Global MAX+PLUS II Design Entry Features" on page 97.

Go to MAX+PLUS II Help for complete and up-to-date information on the MAX+PLUS II Manager.

MAX+PLUS II Applications

MAX+PLUS II software consists of 11 application programs and the MAX+PLUS II Manager. Different design entry applications can be active simultaneously, allowing you to switch between them with a click of the mouse or a menu command. At the same time, you can run one of the background applications—i.e., the Compiler, Simulator, Timing Analyzer, or Programmer. Commands shared by the various applications function in the same way, making your logic design task easier.

You can easily minimize an application window into an icon without closing the application, and restore it later. This feature allows you to keep your screen uncluttered without impairing your efficiency.

Table 2-1 describes the MAX+PLUS II applications and shows their icons.

Table 2-1. MAX+PLUS II Applications (Part 1 of 2)

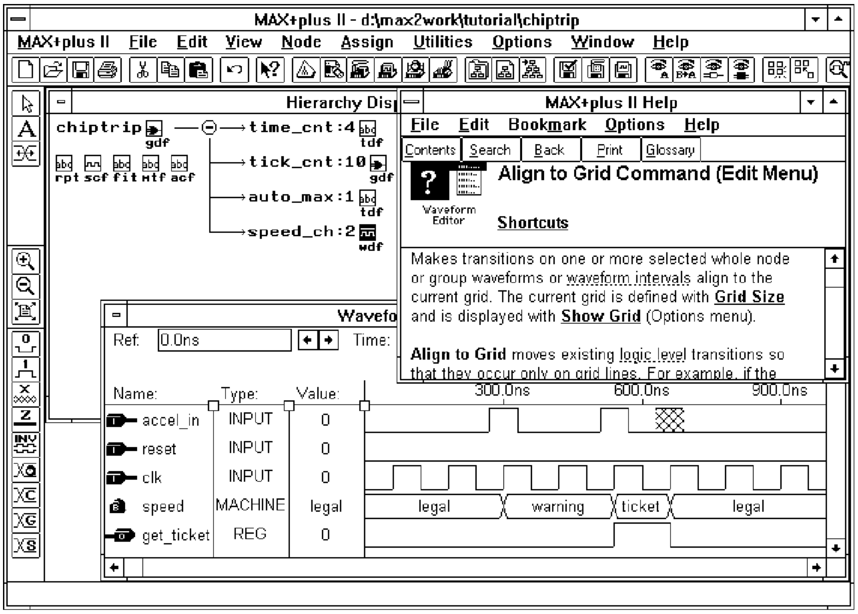
Icon	Application
	Hierarchy Display — Displays the current hierarchy of files as a hierarchy tree with branches that represent subdesigns. You can tell at a glance whether a design file is a schematic, text, or waveform design; which files are currently open; and which user-editable ancillary files are available for the project. You can also directly open or close one or more files in a hierarchy tree and enter resource assignments for them.
	Graphic Editor — Lets you enter a schematic logic design in a true what-you-see-is-what-you-get (WYSIWYG) environment. While the Altera-provided primitives, megafunctions, and macrofunctions serve as your basic building blocks, you can also use custom symbols.
	Symbol Editor — Allows you to edit existing symbols and create new ones.
	Text Editor — The Text Editor lets you create and edit text-based logic design files written in AHDL, VHDL, and Verilog HDL. With the Text Editor, you can also create, view, and edit other ASCII files used with MAX+PLUS II applications. Although you can create HDL files with other text editors, the MAX+PLUS II Text Editor allows you to take advantage of context-sensitive help, syntax coloring, and AHDL, VHDL, and Verilog HDL templates.

Table 2-1. MAX+PLUS II Applications (Part 2 of 2)

Icon	Application
	Waveform Editor — Serves a dual role: as a design entry tool and as a tool for entering test vectors and viewing simulation results.
	Floorplan Editor — Lets you assign logic to physical device pin and logic cell resources in a graphical environment. You can edit pin placements in a device package view and assign signals to individual logic cells in a more detailed Logic Array Block (LAB) view. You can also view the results of the last compilation.
	Compiler — Processes logic projects targeted for Altera Classic, MAX 5000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000, and FLEX 10K device families. It performs most tasks automatically. However, you can customize all or part of the compilation process.
	Simulator — Enables you to test the logical operation and internal timing of your logic circuit. Functional simulation, timing simulation, and linked multi-device simulation are available.
	Timing Analyzer — Analyzes the performance of your logic circuit after it has been synthesized and optimized by the Compiler.
	Programmer — Lets you program, configure, verify, examine, and test Altera devices.
	Message Processor — Displays error, warning, and information messages on the status of your project and allows you to locate the source of a message automatically in the original design file(s), ancillary file(s), and assignments floorplan.

Figure 2-5 shows a display of multiple windows: the Hierarchy Display and Waveform Editor windows, and a MAX+PLUS II Help topic.

Figure 2-5. Display of Multiple MAX+PLUS II Applications & Help



Align to Grid Command (Edit Menu)

Makes transitions on one or more selected whole node or group waveforms or waveform intervals align to the current grid. The current grid is defined with **Grid Size** and is displayed with **Show Grid** (Options menu).

Align to Grid moves existing logic level transitions so that they occur only on grid lines. For example, if the

Name:	Type:	Value:
accel_in	INPUT	0
reset	INPUT	0
clk	INPUT	0
speed	MACHINE	legal
get_ticket	REG	0

Design Files, Ancillary Files & Projects

Before you get started with MAX+PLUS II, you should understand the difference between design files, ancillary files, and projects.

Design Files

A *design file* is a graphic, text, or waveform file created with the MAX+PLUS II Graphic, Text, or Waveform Editor, or with another industry-standard schematic or text editor or an EDIF, VHDL, or Verilog HDL netlist writer. It contains logic for a MAX+PLUS II project and is compiled by the Compiler. The Compiler can automatically process the following design files:

- Graphic Design Files (**.gdf**)
- AHDL Text Design Files (**.tdf**)
- Waveform Design Files (**.wdf**)
- VHDL Design Files (**.vhd**)
- Verilog Design Files (**.v**)
- OrCAD Schematic Files (**.sch**)
- EDIF Input Files (**.edf**)
- Xilinx Netlist Format Files (**.xnf**)
- Altera Design Files (**.adf**)
- State Machine Files (**.smf**)

Ancillary Files

Ancillary files are files that are associated with a MAX+PLUS II project but are not part of the project hierarchy tree. Most ancillary files do not contain design logic. Some of these files are generated automatically by a MAX+PLUS II application, others are user-entered. Examples of ancillary files are Assignment & Configuration Files (**.acf**), Symbol Files (**.sym**), Report Files (**.rpt**), and Vector Files (**.vec**).

Projects

A *project* consists of all files in a design hierarchy, including ancillary input and output files. The project name is the name of the top-level design file, without the filename extension. MAX+PLUS II performs compilation, simulation, timing analysis, and programming on one project at a time, although you can always edit files belonging to another project. For example, as you compile **project1**, you may edit a TDF that is part of **project2** and save it; however, if you wish to compile it, you must first specify **project2** as the project name.

You should place each project into a separate subdirectory of the MAX+PLUS II working directory **\max2work**. (On a UNIX workstation, this directory is a subdirectory of the **/usr** directory.)

MAX+PLUS II Help

MAX+PLUS II Help provides *the* complete, up-to-date documentation on MAX+PLUS II software. Help teaches you all you need to know about each MAX+PLUS II application's basic tools, commands, procedures, shortcuts, golden rules, and messages; all primitives, megafunctions, and macrofunctions; and AHDL, VHDL, and Verilog HDL. Help also offers information on all Altera devices and adapters, allowing you to choose the appropriate device before you even begin your logic design. It points you to other Altera technical documents for additional helpful information, and provides tips on how to design most effectively with MAX+PLUS II tools.

Each Help topic contains one or more underlined words, called *jumps*, that provide links to other Help topics or to additional information on the current topic. By default, jumps are shown in green text. To view a topic, point to the jump and click Button 1 (the left mouse button) on it. A jump with a solid underline takes you to a new Help topic. A jump with a dotted underline pops up a glossary entry. A blue jump pops up an example, a list of shortcuts, or an illustration on top of the current Help topic. When you click Button 1 again, the pop-up topic closes. You can also click Button 1 on a *segmented hypergraphic*, which is a picture in a Help topic, such as a picture of a dialog box, that has links to pop-up topics.

Help is only a keystroke or a mouse click away. On-line information is accompanied by a large number of illustrations.

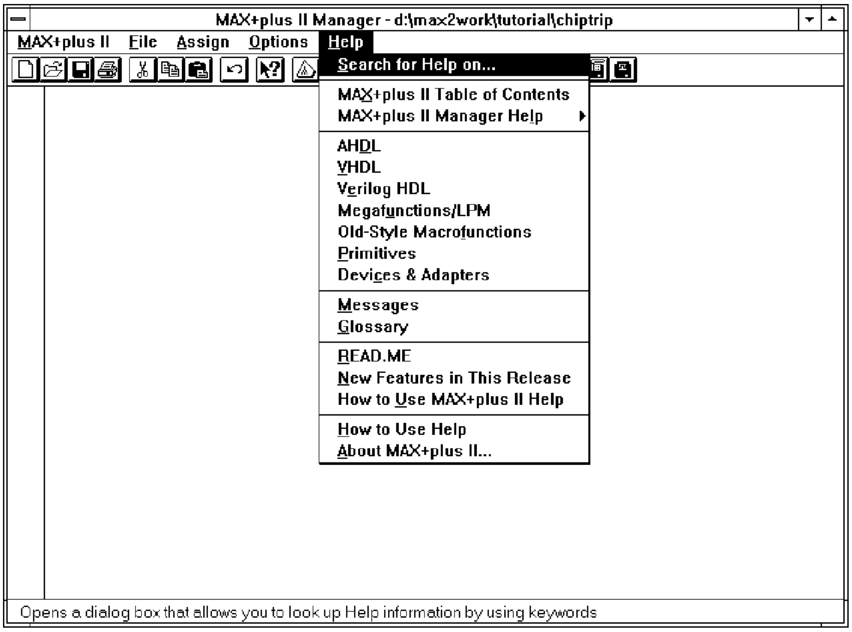
Go to the *MAX+PLUS II Help Poster* provided with your MAX+PLUS II system for colorful and fun explanations of how to use Help.

For information on the mechanics of using Help (e.g., copying or printing a Help topic), choose **How to Use Help** (Help menu).

The Help Menu

The menu bar of each MAX+PLUS II application provides access to the Help menu, shown in Figure 2-6.

Figure 2-6. MAX+PLUS II Help Menu



Opens a dialog box that allows you to look up Help information by using keywords

Table 2-2 describes all Help menu items and, when appropriate, shows the icons that represent them in the Help documentation.

Table 2-2. MAX+PLUS II Help Menu Items (Part 1 of 4)

Help Item	Icon
Search for Help on — Opens the **Help Topics** dialog box (called **Search** in Windows NT 3.51), which allows you to quickly search through Help's extensive index. You can select the word or phrase you want to find. If you start typing, the list box automatically scrolls to the words/ phrases that most closely match what you are typing. You can then list relevant Help topics and go to a topic you wish to view.	—
MAX+PLUS II Table of Contents — A comprehensive table of contents that lists all major topics provided with MAX+PLUS II Help. It is also accessible via the **Contents** button in the button bar at the top of the Help window.	—

Table 2-2. MAX+PLUS II Help Menu Items (Part 2 of 4)

Help Item	Icon
<Application Name> **Help** — Opens a submenu of Help topics for the current MAX+PLUS II application:	*Table 2-1 shows all application icons*
Table of Contents — Table of contents for the current application.	—
Introduction — Overview of the current application, including illustrations and information on how to get started with using the application.	ℹ
Basic Tools — Detailed descriptions of items visible in an application window, as well as input and output files, accompanied by illustrations and examples. The Basic Tools Help category also provides information on primitives and macrofunctions, buttons, fields, icons, etc.	🔧
Commands — Complete details about each command in the current application, accompanied by illustrations and examples. Illustrations of command dialog boxes include pop-up explanations of each option and button in the dialog box.	▤
Procedures — Step-by-step instructions, accompanied by illustrations and examples, on how to perform specific tasks in the current application.	1. 2. 3.
Golden Rules — A summary of essential tips and rules for using the current application.	🧰
Shortcuts — Keyboard, mouse, toolbar, and tool palette shortcuts for the commands and procedures of the current application.	⌨
AHDL — Help on AHDL, including detailed instructions on how to develop a design, and descriptions of basic elements, design structure, Backus-Naur Form (BNF) syntax, and a style guide.	AHDL
VHDL — Help on VHDL, including instructions on how to develop a design with MAX+PLUS II VHDL, and descriptions of supported VHDL constructs, Backus-Naur Form (BNF) syntax, and a style guide.	VHDL

Table 2-2. MAX+PLUS II Help Menu Items (Part 3 of 4)

Help Item	Icon
Verilog HDL — Help on Verilog HDL, including instructions on how to develop a design with MAX+PLUS II Verilog HDL, and descriptions of supported Verilog HDL constructs, Backus-Naur Form (BNF) syntax, and a style guide.	Verilog
Megafunctions/LPM — A list of megafunctions, including Library of Parameterized Modules (LPM) functions and Altera MegaCore/OpenCore functions. If you choose a specific megafunction, its function, usage rules, AHDL Function Prototype, VHDL Component Declaration, and information on device resource usage are displayed.	—
Old-Style Macrofunctions — An alphabetical list of old-style macrofunction categories. You can choose one of the categories listed to display all macrofunction names in that category. If you choose a specific macrofunction, its description, default signal logic levels, AHDL Function Prototype, VHDL Component Declaration, and function table are displayed.	—
Primitives — An alphabetical list of primitives. If you choose a specific primitive, its description, usage rules, AHDL Function Prototype, VHDL Component Declaration, and function table are all displayed.	—
Devices & Adapters — A list of all current Altera devices supported by MAX+PLUS II and their programming adapters. Selection guides for each Altera device family are also included. You can choose one of the devices listed to display a description of the device and the pin and logic cell locations for each supported device package.	?
Messages — An alphabetical list of all MAX+PLUS II information, error, and warning messages. All messages are accompanied by detailed explanations of possible causes and recommended actions. Message explanations also provide jumps to additional helpful information.	
Glossary — A comprehensive list of MAX+PLUS II terms and their definitions.	abc
READ.ME — A copy of the **read.me** file provided with MAX+PLUS II. It gives information on system requirements, known problems, and fixes.	—

Table 2-2. MAX+PLUS II Help Menu Items (Part 4 of 4)

Help Item	Icon
New Features in This Release — A description of all new features in the current release of MAX+PLUS II, including updates to device support and interfaces to third-party EDA tools.	
How to Use Help — Information on the mechanics of using the Windows Help application.	—
How to Use MAX+PLUS II Help — Detailed information on how to use MAX+PLUS II Help, including descriptions of Help categories and documentation conventions.	
About MAX+PLUS II — Displays the MAX+PLUS II version number, current application version number, copyright and patent information, and memory and system resource usage information.	—

The Help Window Button Bar

Table 2-3 describes the buttons in the bar at the top of the Help window, which enable you to move around in Help.

Table 2-3. MAX+PLUS II Help Window Buttons

Name	Function
Contents	Shows the MAX+PLUS II Help table of contents.
Index	Opens the **Help Topics** dialog box (called **Search** in Windows NT 3.51).
Back	Goes back to previously viewed information, i.e., you retrace your path through the topics you have already viewed. The button is dimmed if there is no previous topic.
Print	Prints one or more copies of the current Help topic.
Glossary	Shows the list of MAX+PLUS II terms and their definitions. You can print any glossary entry that you open from this list.
History	Shows the last 40 topics you have viewed. You can read a topic again by double-clicking Button 1 on it. (Available only in Windows NT 3.51.)

Where to Start in Help

Help is versatile. It lets *you* decide how you want to learn about MAX+PLUS II. If you are a first-time user, however, you might find the following approach most efficient:

Step 1: From the Help menu of any application, choose the **MAX+PLUS II Table of Contents** command. This window shows all applications and other topics—including the icons that represent them—on which Help information is available.

Step 2: From the *MAX+PLUS II Table of Contents* topic, choose *Introduction to MAX+PLUS II*. This overview summarizes the features of MAX+PLUS II and suggests starting points ("Where to Start") for becoming acquainted with MAX+PLUS II.

Step 3: From the *Introduction to MAX+PLUS II*, you can choose *How to Use MAX+PLUS II Help* to get basic information about how Help is organized and which format conventions it uses.

Step 4: From the Help menu in any application, choose <*application name*> **Help Procedures** to view a list of all step-by-step procedures that you can use in the current application.

Step 5: From the Help menu in any application, choose <*application name*> **Help Golden Rules** to learn about essential tips and guidelines for the current application. Golden Rules allow you to get a head start on how to design logic circuits with MAX+PLUS II by condensing essential information that is available in other Help topics.

☞ Each MAX+PLUS II application provides its own *Introduction* and *Golden Rules*.

How to Request Help on a Specific Topic

MAX+PLUS II offers both menu-based and context-sensitive help. You can request help with the mouse or with the keyboard in a variety of ways:

- Every MAX+PLUS II application has a Help menu that guides you to application-specific or general MAX+PLUS II information. For example:
  - *To open the MAX+PLUS II index:* Choose **Search for Help on** from the Help menu. In the dialog box that opens, scroll to or type the desired keyword, double-click Button 1 on it to list all related Help topics, then double-click Button 1 on a topic title to open the help topic. You need not type the entire word, since the characters you type are matched with the keywords listed.
  - *To learn the shortcut for a command:* Go to the Help topic describing the command using **Search for Help On** and click Button 1 on the blue "Shortcuts" jump at the top of the topic, or go to the Help menu and choose **Shortcuts** from the <*Application Name*> Help submenu to display a table of all keyboard, mouse, toolbar, and tool palette shortcuts in the current application.
- When you choose the context-sensitive Help button ( ) from the toolbar or press Shift+F1, the pointer turns into a question mark pointer. You can then click Button 1 on any item in the window or any menu command. If context-sensitive help is available for the item, the relevant information is displayed. Otherwise, Help shows a list of all items for which context-sensitive help is available.
- When a menu command is highlighted, or a command dialog box or a pop-up message is displayed, press F1 to get help on that topic. For example:
  - *To get instant information on a command:* Highlight the command on the menu and press F1.
  - *To see a list of all context-sensitive help items in an application:* Press F1. MAX+PLUS II Help shows a list of all items in the current application for which context-sensitive help is available. Click Button 1 on the desired item.

Design Entry

All tools necessary for creating a logic design are readily accessible in MAX+PLUS II. MAX+PLUS II accelerates your design entry with a set of standard logic functions, including primitives, megafunctions, LPM functions, and old-style 74-series-type macrofunctions. It also provides numerous basic and advanced editing features that make logic entry and debugging easier.

MAX+PLUS II provides three design entry editors—the Graphic, Text, and Waveform Editors. It also includes two auxiliary editors—the Floorplan and Symbol Editors—that facilitate design entry.

MAX+PLUS II supports a variety of design entry methods:

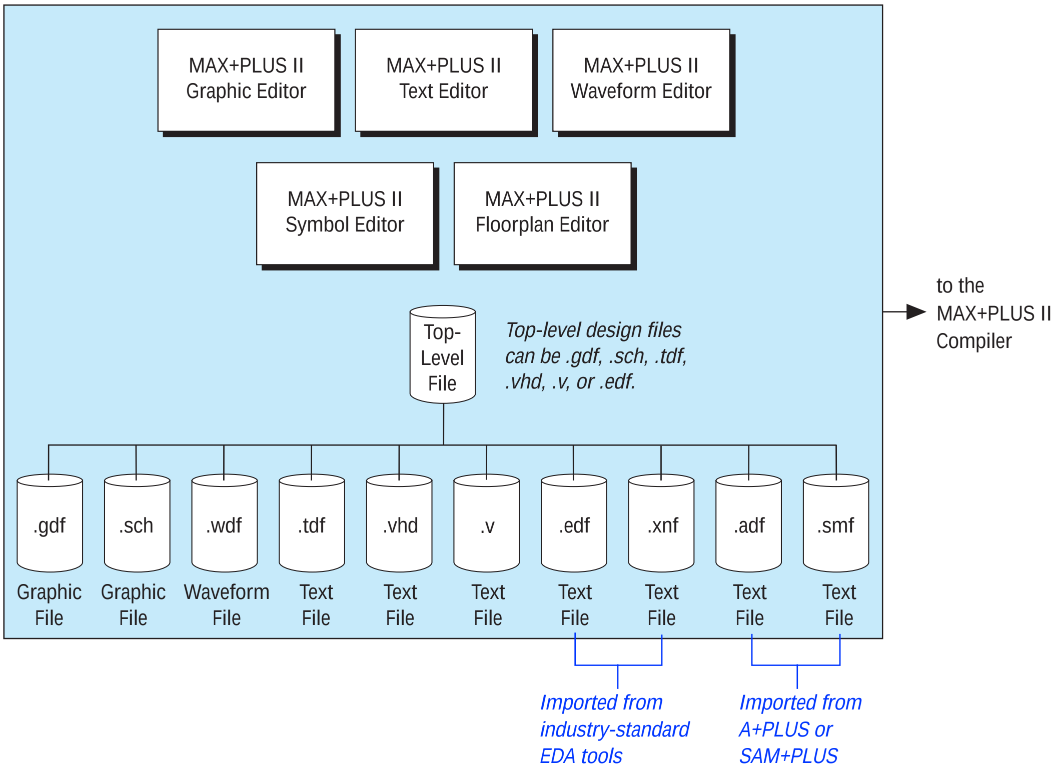
- Schematic designs are entered with the MAX+PLUS II Graphic Editor. You can also open, edit, and save schematics created with the OrCAD Draft schematic editor.
- Altera Hardware Description Language (AHDL), VHDL, and Verilog HDL designs are entered with the MAX+PLUS II Text Editor or another standard text editor.
- Waveform designs are entered with the MAX+PLUS II Waveform Editor.
- EDIF netlist files and Xilinx netlist files generated by other industry-standard EDA tools can be imported into the MAX+PLUS II environment.
- Schematic and text design files created with MAX+PLUS (DOS) and files created with Altera's A+PLUS and SAM+PLUS software packages can be integrated into the MAX+PLUS II environment.
- Physical resource assignments for any node or pin in the current project can be entered in a graphical environment with the Floorplan Editor. The Floorplan Editor saves assignments in the Assignment & Configuration File (**.acf**) for the project, which stores all types of resource, probe, and device assignments, as well as configuration settings for the Compiler, Simulator, and Timing Analyzer.
- Graphic symbols that represent any type of design file can be generated automatically in any MAX+PLUS II design editor. You can

edit the symbols or create your own custom symbols with the Symbol Editor, and use them in any schematic design file.

In a hierarchical project, you can freely mix Graphic Design Files (**.gdf**), Text Design Files (**.tdf**), VHDL Design Files (**.vhd**), Verilog Design Files (**.v**), EDIF Input Files (**.edf**), and OrCAD Schematic Files (**.sch**) at any level of the hierarchy. However, Waveform Design Files (**.wdf**), Xilinx Netlist Format Files (**.xnf**), Altera Design Files (**.adf**), and State Machine Files (**.smf**) must be either at the lowest level of a project hierarchy or be the only design file in a project. See "Project Hierarchy" on page 125.

See Figure 2-7.

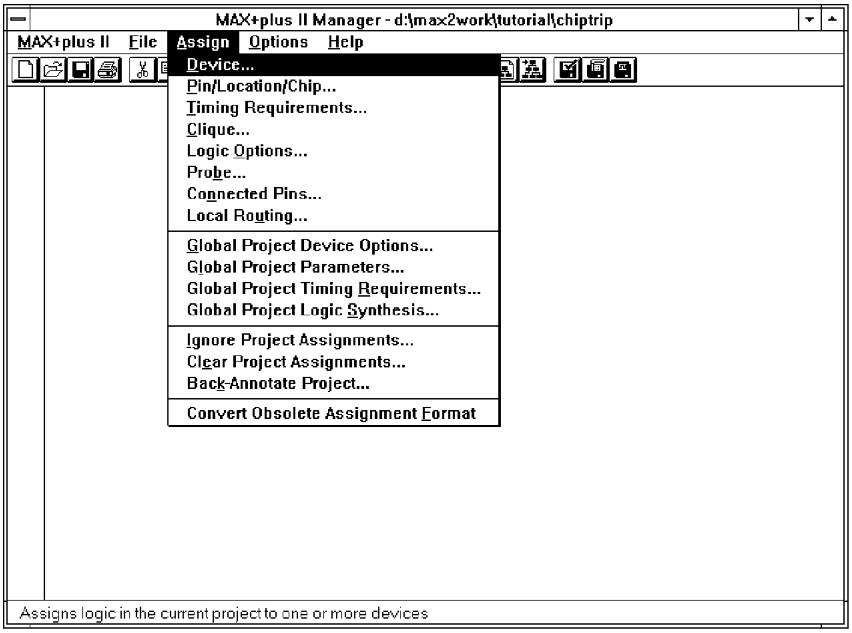
Figure 2-7. MAX+PLUS II Design Entry Methods


MAX+PLUS II Graphic Editor
MAX+PLUS II Text Editor
MAX+PLUS II Waveform Editor
MAX+PLUS II Symbol Editor
MAX+PLUS II Floorplan Editor
Top-Level File
Top-level design files can be .gdf, .sch, .tdf, .vhd, .v, or .edf.
to the MAX+PLUS II Compiler
.gdf Graphic File
.sch Graphic File
.wdf Waveform File
.tdf Text File
.vhd Text File
.v Text File
.edf Text File
.xnf Text File
.adf Text File
.smf Text File
Imported from industry-standard EDA tools
Imported from A+PLUS or SAM+PLUS

Global MAX+PLUS II Design Entry Features

All MAX+PLUS II applications allow you to enter, edit, and delete the types of resource, device, and parameter assignments that control project compilation—including logic synthesis, partitioning, and fitting—with commands on the Assign menu. Figure 2-8 shows the MAX+PLUS II Assign menu. You can enter assignments for the current project regardless of whether any project design file or application window is open. MAX+PLUS II saves the information in an Assignment & Configuration File (**.acf**) for the project. Assignment edits made in the Floorplan Editor window are also saved in the ACF. In addition, you can edit a project's ACF manually with the Text Editor.

Figure 2-8. MAX+PLUS II Assign Menu


MAX+plus II Manager - d:\max2work\tutorial\chiptrip
MAX+plus II
File
Assign
Options
Help
Device...
Pin/Location/Chip...
Timing Requirements...
Clique...
Logic Options...
Probe...
Connected Pins...
Local Routing...
Global Project Device Options...
Global Project Parameters...
Global Project Timing Requirements...
Global Project Logic Synthesis...
Ignore Project Assignments...
Clear Project Assignments...
Back-Annotate Project...
Convert Obsolete Assignment Format

Assigns logic in the current project to one or more devices

The following functions are common to all MAX+PLUS II applications:

- Device, resource, and probe assignments
- Back-annotation
- Global project device options
- Global project parameters

- Global project timing requirements
- Global project logic synthesis

Device, Resource & Probe Assignments

A resource is a portion of an Altera device, such as a pin or logic cell, that performs a specific user-defined task. You can assign logic to device resources to ensure that the MAX+PLUS II Compiler fits a project exactly as you wish. The following types of assignments are available:

- *Clique assignment* Specifies which logic functions must remain together. Grouping logic functions into a clique helps ensure that they are implemented in the same Logic Array Block (LAB), Embedded Array Block (EAB), row, or device.

- *Chip assignment* Specifies which logic functions must be implemented in the same device when a project is partitioned into multiple devices.

- *Pin assignment* Assigns the input or output of a single logic function, such as a primitive or megafunction, to a specific pin, row, or column within a chip.

- *Location assignment* Assigns a single logic function, such as the output of a primitive or megafunction, to a specific location within a chip, such as a logic cell, I/O cell, embedded cell, LAB, EAB, row, or column.

- *Probe assignment* Assigns an easy-to-remember, unique name to an input or output of a logic function.

- *Connected pin assignment* Specifies how two or more pins are connected externally on your circuit board. This information is also useful for timing simulation and linked multi-project simulation.

- *Local routing assignment* Assigns a fan-out of a node to a logic cell in the same LAB as the node or an adjacent LAB to the node, using shared local interconnect. Local routing is also available between a node that is placed in an LAB on the periphery of a device and the output pin that it feeds.

- *Device assignment* Assigns project logic to a device. In a multi-device project, device assignments map chip assignments to specific devices.

You can assign a specific device and its package type, speed grade, and operating temperature. You can also assign an "AUTO" device and allow the Compiler to select a device from a target device family. This automatic device selection process can be controlled by specifying both the range and number of devices to use from the target device family. If a project is too large to fit into a specified device, you can also specify the type and number of additional devices.

- *Logic option assignment* Guides logic synthesis on individual logic functions during compilation with logic synthesis style and/or individual logic synthesis options.

  Altera provides numerous logic options, as well as three "ready-made" synthesis styles, each of which represents a collection of logic option settings combined under a single style name. You can use these styles or create your own custom styles. Synthesis styles let you tailor your synthesis options for a specific device family to take advantage of that family's architecture.

- *Timing assignment* Guides logic synthesis and fitting on individual logic functions to achieve the desired performance for input to non-registered output delays ($\mathbf{t_{PD}}$), Clock to output delays ($\mathbf{t_{CO}}$), Clock setup time ($\mathbf{t_{SU}}$), and Clock frequency ($\mathbf{f_{MAX}}$). You can also cut the connections between the timing path for a particular signal (called a "node" in MAX+PLUS II) and other nodes in the project.

Go to the current Altera ***Data Book*** and individual device data sheets for complete information on all devices.

Go to *Devices & Adapters* in MAX+PLUS II Help for pin locations for all currently available Altera device packages.

Back-Annotation

After you have entered your entire project, you can compile your project with MAX+PLUS II, then preserve, i.e., back-annotate, the resource assignments that the Compiler made during the most recent compilation so that you can produce the same fit with subsequent compilations.

Global Project Device Options

You can specify the global device options for the Compiler to use for all devices when it processes a project. To reserve additional logic capacity for future use, you can specify the percentage of pins and logic cells that must remain unused during the current compilation. You can also specify the settings for device option bits and dual-purpose device configuration pins. For example, you can specify the global default setting for the Security Bit, which prevents an EPROM- or EEPROM-based device from being interrogated.

Global Project Parameters

You can specify the names and global settings for the Compiler to use for parameters in all parameterized functions in your project.

Global Project Timing Requirements

You can enter global timing requirements for a project to specify the overall requirements for input to non-registered output delays ($\mathbf{t_{PD}}$), Clock to output delays ($\mathbf{t_{CO}}$), Clock setup time ($\mathbf{t_{SU}}$), and Clock frequency ($\mathbf{f_{MAX}}$). You can also cut the connections between all bidirectional feedback, Preset signal, and Clear signal timing paths and other timing paths in the project.

Global Project Logic Synthesis

You can select global synthesis settings for the Compiler to use when it synthesizes a project. You can specify a default logic synthesis style, specify a speed/area optimization preference, and instruct the Compiler to select automatic global control signals, such as the Clock, Clear, Preset, and Output Enable signals. You can also direct the Compiler to use standard or multi-level synthesis, one-hot state machine encoding, and automatic register packing. In addition, you can specify preferences to automatically implement logic in fast input or output logic cells and I/O cells, open-drain pins, and embedded array blocks (EABs).

Common Editor Functions

Many functions are shared by all five MAX+PLUS II editors or by the three design editors (the Graphic, Text, and Waveform Editors), making it easier for you to design logic. For example, standard functions such as saving and retrieving a file are available in all design editors.

In addition, the following functions are common to MAX+PLUS II editor applications:

Symbol & Include File Generation

You can automatically generate and update a symbol for any type of MAX+PLUS II–supported design file with the **Create Default Symbol** command (File menu). The Symbol File has the same name as the design file, with the extension **.sym**. It can be incorporated into any GDF or OrCAD Schematic File (**.sch**) that is higher up in the project hierarchy. You can also create a custom symbol for a design file.

In a process analogous to creating a default symbol, you can automatically generate and update an Include File containing an AHDL Function Prototype for any MAX+PLUS II design file. The **Create Default Include File** command (File menu) creates an Include File with the same name as the design file, with the extension **.inc**. You can use an AHDL Include Statement to incorporate the Include File and its Function Prototype into a TDF that is higher up in the project hierarchy. These Function Prototypes are required to implement instances of mega- and macrofunctions in a TDF. The MAX+PLUS II Compiler also uses the information in Include Files containing AHDL Function Prototypes to process instances of logic functions in Verilog Design Files (**.v**).

Node Location

You can select a node in an ancillary file or the current floorplan, and locate it instantly in the original design file with the **Find Node in Design File** command (Utilities menu). Similarly, you can select a node or clique in a design or ancillary file and locate it instantly in the floorplan for a project with **Find Node in Floorplan** or **Find Clique in Floorplan** (Utilities menu). The **Find Node in Floorplan** and **Find Clique in Floorplan** commands are also available in the Hierarchy Display.

Hierarchy Traversal

You can move up, down, or to the top of the current hierarchy. MAX+PLUS II opens the selected file or brings it to the front, and automatically starts the appropriate editor.

Context-Sensitive Menu Commands

Context-sensitive menu commands are available on pop-up menus in all editors and in the Hierarchy Display. These menus, which display commands that are appropriate for the selected object(s) at the current mouse pointer location, pop up automatically when you click Button 2 (the right mouse button).

Timing Analysis

You can tag nodes as sources and destinations for timing analysis, and calculate point-to-point propagation delays, setup and hold time requirements, and the maximum Clock frequency for each Clock signal in a project. You can also completely cut off a node so that only the signal path that leads to the node is included in the analysis.

Find & Replace Text

You can find and replace text—including the names of nodes and probes—in any design file in the current hierarchy.

Undo, Cut, Copy, Paste & Delete

You can undo the most recent editing step—and undo the undo. You can also copy, cut, paste, and delete one or more selected items, and paste to other MAX+PLUS II applications or Windows applications outside MAX+PLUS II.

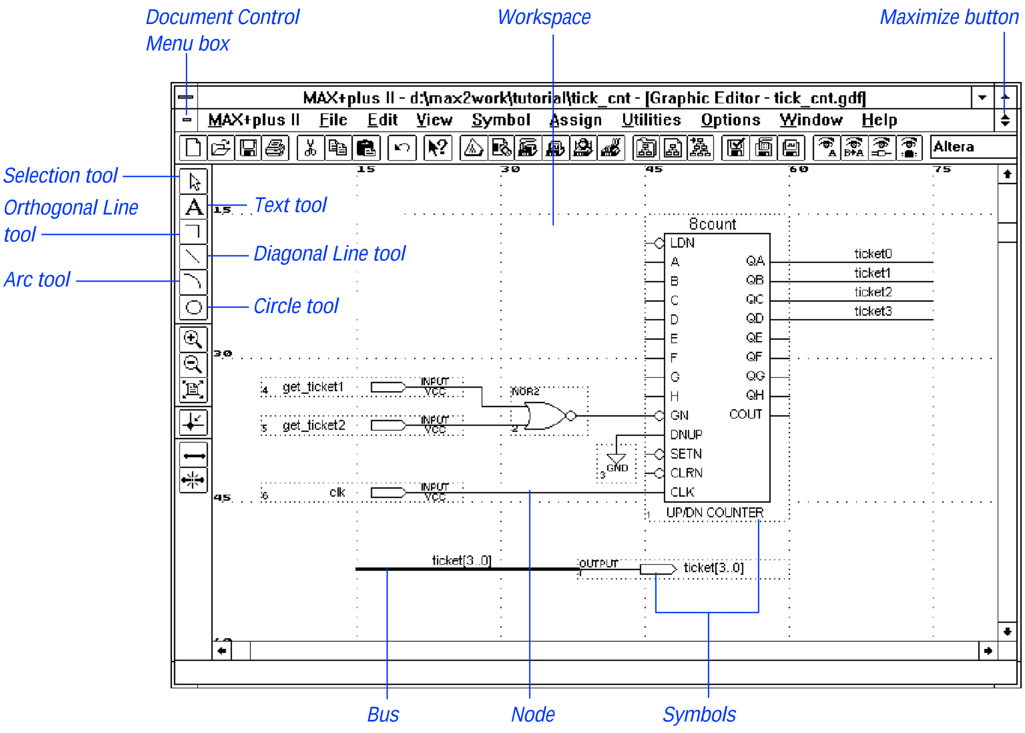
Print

You can print all or part of the current file, specify the printer or plotter, and determine the printer configuration.

MAX+PLUS II Graphic Editor

The MAX+PLUS II Graphic Editor, shown in Figure 2-9, offers a what-you-see-is-what-you-get design environment. You open a new, untitled Graphic Editor window with the **New** command (File menu), or, if no Graphic Editor window is open, by choosing **Graphic Editor** from the MAX+PLUS II menu.

Figure 2-9. MAX+PLUS II Graphic Editor


Document Control Menu box
Workspace
Maximize button
Selection tool
Orthogonal Line tool
Arc tool
Text tool
Diagonal Line tool
Circle tool
MAX+plus II - d:\max2work\tutorial\tick_cnt - [Graphic Editor - tick_cnt.gdf]
MAX+plus II File Edit View Symbol Assign Utilities Options Window Help
Altera
8count
LDN A B C D E F G H GN DNUP SETN CLRN CLK
QA QB QC QD QE QF QG QH COUT
UP/DN COUNTER
ticket0
ticket1
ticket2
ticket3
get_ticket1
get_ticket2
clk
INPUT VCC
NOR2
GND
ticket[3..0]
OUTPUT
Bus
Node
Symbols

The Graphic Editor is a sophisticated schematic capture program that allows you to enter even complex designs quickly and easily. The extensive primitive, megafunction, and macrofunction libraries—including the Library of Parameterized Modules (LPM)—provide basic building blocks for constructing a design, while the symbol-generation capability lets you build your own libraries of custom functions.

A Graphic Design File (**.gdf**) or OrCAD Schematic File (**.sch**) created with the Graphic Editor can include any combination of primitive, megafunction, and macrofunction symbols. Symbols may represent any type of design file, including other GDFs and OrCAD Schematic Files, AHDL Text Design Files (**.tdf**), VHDL Design Files (**.vhd**), Verilog Design Files (**.v**), Waveform Design Files (**.wdf**), EDIF Input Files (**.edf**), Xilinx Netlist Format Files (**.xnf**), Altera Design Files (**.adf**), and State Machine Files (**.smf**).

The following features highlight the Graphic Editor's versatility:

- The "smart" Selection tool shown in Figure 2-9 makes design entry easy. This tool allows you to move and copy items and enter new symbols. When you move the Selection tool over a pinstub or the end of a line, it changes automatically to an orthogonal line-drawing tool; when you click on text, such as a pin or node name, it changes automatically into a text editing tool.

- Symbols are connected with signal lines, called nodes, or with bus lines that represent multiple logically grouped nodes. When you assign a name to a node, you can connect it to other nodes or symbols by name only. Buses are connected by name: a graphical connection is optional.

- You can customize the ports used in each separate instance of a mega- or macrofunction symbol, and optionally invert them. Any bit of a bus port can be inverted. A `NOT` "bubble" appears automatically to indicate an inverted port.

- You can select and edit multiple objects in a rectangular area or as discontiguous items. When you move a selection, the rubberbanding feature preserves signal connectivity.

- You can view probe, pin, location, chip, clique, timing, local routing, logic option, and parameter assignments on each symbol. To facilitate simulation, you can also create connected pin group assignments that specify the external device connections between pins.

- Altera-provided primitives, megafunctions, and macrofunctions reduce design entry time. You can also create your own libraries of custom functions. When you edit a symbol or regenerate a default symbol, you can automatically update selected instances or all instances of that symbol in a Graphic Editor file.

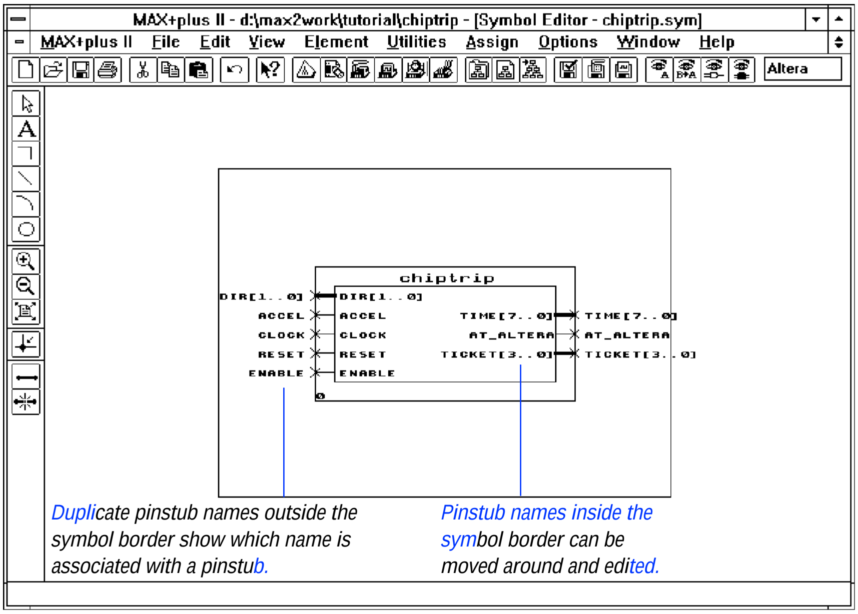
The MAX+PLUS II Graphic Editor provides many other features. For example, you can zoom in and out to various display scales so that you can see an entire design file at a glance or a detailed portion of it. You can also select various text fonts, text sizes, and line styles, and display and set the spacing of guidelines. You can copy, cut, paste, and delete one or more selected items; flip them horizontally or vertically; rotate them by 90, 180, or 270 degrees; and specify the size and horizontal or vertical orientation of the current drawing sheet.

Go to MAX+PLUS II Help for complete information on all MAX+PLUS II Graphic Editor functions and features.

MAX+PLUS II Symbol Editor

The MAX+PLUS II Symbol Editor, shown in Figure 2-10, enables you to view, create, and edit a symbol that represents a logic circuit. You open a new, untitled Symbol Editor window with the **New** command (File menu), or, if no Symbol Editor window is open, by choosing **Symbol Editor** from the MAX+PLUS II menu.

Figure 2-10. MAX+PLUS II Symbol Editor



Duplicate pinstub names outside the symbol border show which name is associated with a pinstub.

Pinstub names inside the symbol border can be moved around and edited.

A Symbol File has the same name as the design file it represents, with the extension **.sym**. The **Create Default Symbol** command, available from the File menu of the Graphic, Text, and Waveform Editors, creates the symbol for any design file.

The following list highlights MAX+PLUS II Symbol Editor features:

- You can customize the symbol that represents a design file.
- You can enter and edit pinstubs and pinstub names for input, output, and bidirectional pins, and specify whether pinstub names should be displayed when the symbol is entered in a Graphic Editor file.

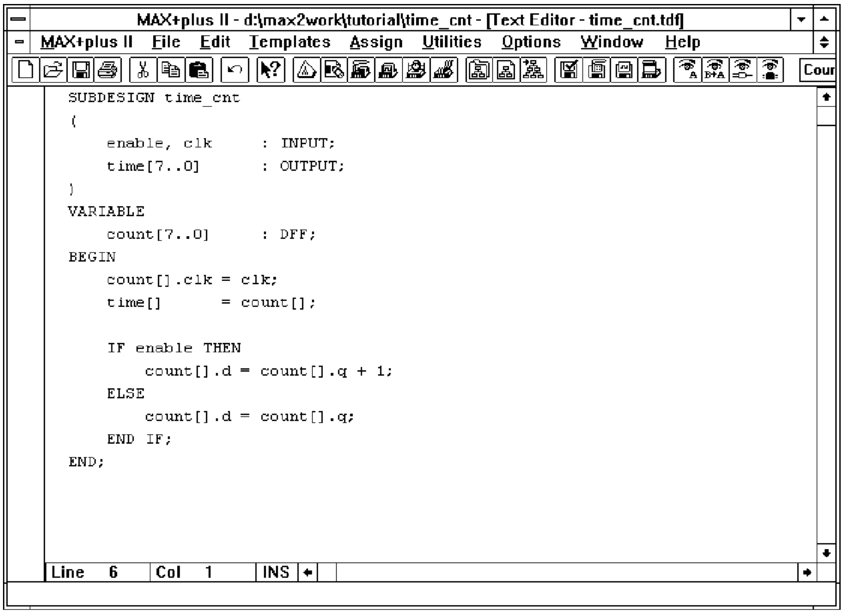
  You can choose to display the full pinstub name in a symbol or change the visible pinstub name, for example, to make it more compact or informative. Therefore, the full port name and the name displayed in a Graphic Editor file can be different.
- Pinstub names are automatically duplicated outside a symbol boundary to give you a visual reference of which name and pinstub go together. When you move a pinstub name inside the symbol boundary, the identical name outside the boundary, which cannot be moved, helps you keep track of its pinstub connection.
- You can specify parameters and their optional default values.
- The grid and guidelines help you to align objects precisely.
- You can insert comments or helpful notes in a symbol. They will also appear when the symbol is entered in a Graphic Editor file.

Go to MAX+PLUS II Help for complete information on all MAX+PLUS II Symbol Editor functions and features.

MAX+PLUS II Text Editor

The MAX+PLUS II Text Editor, shown in Figure 2-11, is a flexible tool for entering Text Design Files (.**tdf**) in the Altera Hardware Description Language (AHDL), VHDL Design Files (**.vhd**) in the Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL), and Verilog Design Files (**.v**) in the Verilog HDL. You can also view, enter, and edit any other ASCII file with the MAX+PLUS II Text Editor. You open a new, untitled Text Editor window with the **New** command (File menu), or, if no Text Editor window is open, by choosing **Text Editor** from the MAX+PLUS II menu.

Figure 2-11. MAX+PLUS II Text Editor



```
SUBDESIGN time_cnt
(
    enable, clk      : INPUT;
    time[7..0]       : OUTPUT;
)
VARIABLE
    count[7..0]      : DFF;
BEGIN
    count[].clk = clk;
    time[]      = count[];

    IF enable THEN
        count[].d = count[].q + 1;
    ELSE
        count[].d = count[].q;
    END IF;
END;
```

Although you can use any ASCII text editor to create AHDL, VHDL, and Verilog HDL design files, only the MAX+PLUS II Text Editor gives you the advantage of the unique design entry, compilation, and debugging features available in MAX+PLUS II.

The following list highlights MAX+PLUS II Text Editor features:

- Because AHDL, VHDL, Verilog HDL, and the Text Editor are completely integrated into the MAX+PLUS II system, you can process an AHDL, VHDL, or Verilog HDL file with the Compiler, and the Message Processor automatically locates any syntax errors in the Text Editor. The Text Editor also provides templates for AHDL, VHDL, and Verilog HDL language constructs. (See "Altera Hardware Description Language" on page 117, "VHDL" on page 119, and "Verilog HDL" on page 121.)

- You can turn on the syntax coloring feature to allow you to clearly view language syntax in AHDL, VHDL, Verilog HDL, and a variety of text-based ancillary files.

- You can automatically find the matching section or comment delimiter for a selected delimiter, allowing you to easily move around your design file.

- You can use the drag-and-drop editing feature to move selected text to a new location within the file.

- You can manually edit Assignment & Configuration Files (**.acf**) that specify probe, resource, and device assignments, as well as project configuration settings for the Compiler, Simulator, and Timing Analyzer.

- You can create Vector Files (**.vec**) that are used as the input for simulation, functional testing, or waveform design entry. You can also create Command Files (**.cmd**) for use with the MAX+PLUS II Simulator, as well as EDIF Command Files (**.edc**) and Library Mapping Files (**.lmf**) for use with the MAX+PLUS II Compiler. You can also edit any other ASCII file.

- When you run a compilation or simulation, the MAX+PLUS II Message Processor automatically locates any syntax errors in text-based ancillary files in the Text Editor.

- With context-sensitive help, you can get immediate help on AHDL syntax elements, keywords, and statements. You can also get help on all Altera-provided primitives, megafunctions, and macrofunctions in AHDL, VHDL, and Verilog HDL design files. In addition, you can get context-sensitive help on keywords and syntax elements in other text files, such as Assignment & Configuration Files, Vector Files, Command Files, and EDIF Command Files.

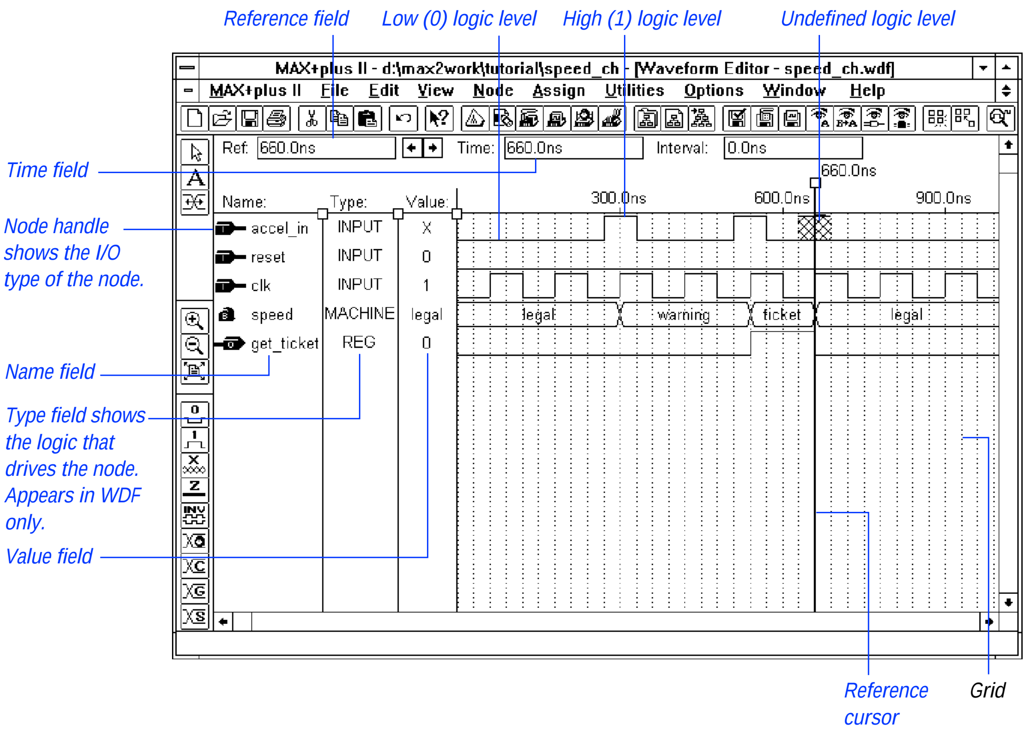
The MAX+PLUS II Text Editor provides many other features. For example, you can find, cut, copy, paste, insert, and delete text; select different text fonts and sizes; set tab stops; and use automatic indentation.

Go to MAX+PLUS II Help for complete information on all MAX+PLUS II Text Editor functions and features.

MAX+PLUS II Waveform Editor

The MAX+PLUS II Waveform Editor, shown in Figure 2-12, serves two roles: as a design entry tool and as a tool for entering test vectors and viewing simulation results. You can create Waveform Design Files (**.wdf**) that contain design logic for a project and Simulator Channel Files (**.scf**) that contain input vectors for simulation and functional testing. You open a new, untitled Waveform Editor window with the **New** command (File menu), or, if no Waveform Editor window is open, by choosing **Waveform Editor** from the MAX+PLUS II menu.

Figure 2-12. MAX+PLUS II Waveform Editor



Waveform design entry offers an alternative to graphic and text design entry. You create a waveform design by specifying combinations of input logic levels and desired outputs as graphical waveforms. A WDF can contain both logical and state machine inputs, as well as combinatorial, registered, and state machine outputs. Buried nodes can also be used to help define desired outputs.

Waveform design entry is best suited for circuits with well-defined sequential inputs and outputs, such as state machines, counters, and registers. They are ideal for combinatorial functions decoded from counters and for other repeating functions.

The Waveform Editor offers a variety of features. You can easily transform whole or partial waveforms and create and edit nodes and groups. With a few simple commands, you can create an ASCII Table File (**.tbl**) or import an ASCII Vector File (**.vec**) to create an SCF or WDF. You can also save a WDF as an SCF for simulation, or convert an SCF to a WDF for use as a design file.

The following list highlights MAX+PLUS II Waveform Editor features:

- You can create or edit a node to have an I/O type that represents an input pin, output pin, or buried logic.
- When you create a WDF, you can specify the type of logic that drives each node as pin input, registered, combinatorial, or state machine.
- You can specify a default high (1), low (0), undefined (X), or high-impedance (Z) logic level on a logic node, or any default state name on a state machine-type node.
- You can easily add any or all nodes from the Simulator Netlist File (**.snf**) for a fully optimized, compiled project to an SCF to simplify test vector creation.
- You can combine from 2 to 256 nodes to create a new group (bus), or undo grouping to expand a group into its original members. Groups can also be combined into other groups. The group value can be displayed in binary, decimal, hexadecimal, or octal radix, with or without conversion to Gray code.
- You can copy, paste, move, or delete a selected portion ("interval") of a waveform, a whole waveform, or an entire node or group (i.e., the node or group name plus waveform). With a single operation, you can edit multiple intervals, whole waveforms, and entire nodes and groups. Copies of entire nodes and groups are linked, so that waveform edits in one copy are reflected in all copies. You can also invert, insert, overwrite, repeat, expand, or compress a waveform interval of any length with any logic level, clock signal, count sequence, or state name.
- You can define and optionally display a drawing grid for aligning logic level transitions either before or after they are created.

- You can insert comments between waveforms at any point within a file.
- You can zoom in and out to any scale.
- To show the differences between simulation outputs and actual device outputs, you can superimpose any outputs on the current file, or superimpose a second Waveform Editor file to compare its node and group waveforms with those in the current file.

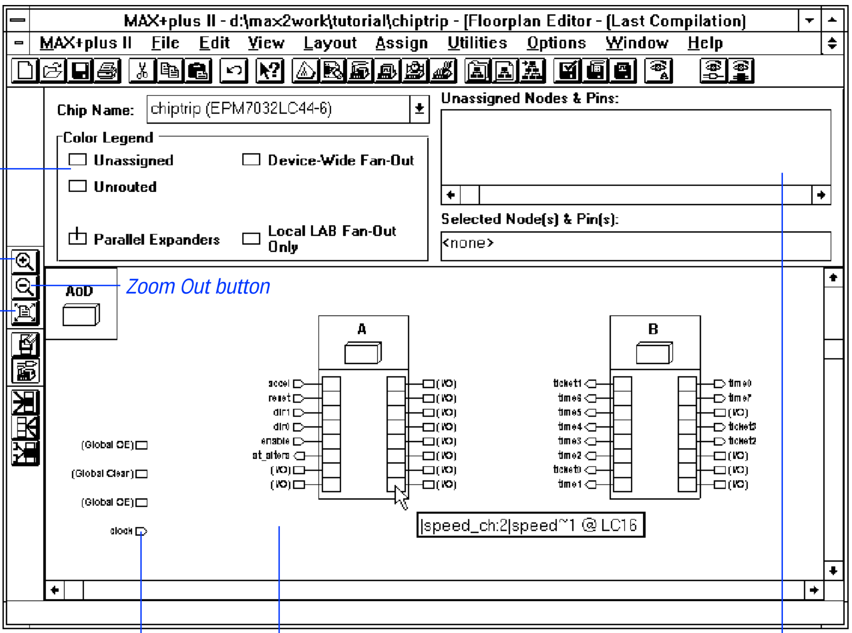
For additional information on waveform editing, go to "MAX+PLUS II Waveform Editor" under "Project Verification" on page 141.

Go to MAX+PLUS II Help for complete information on all MAX+PLUS II Waveform Editor functions and features.

MAX+PLUS II Floorplan Editor

You can use the MAX+PLUS II Floorplan Editor, shown in Figure 2-13, to assign physical device resources and to view Compiler partitioning and fitting results. You open the Floorplan Editor window by choosing **Floorplan Editor** from the MAX+PLUS II menu.

Figure 2-13. MAX+PLUS II Floorplan Editor


MAX+plus II - d:\max2work\tutorial\chiptrip - [Floorplan Editor - (Last Compilation)]
MAX+plus II File Edit View Layout Assign Utilities Options Window Help
Chip Name: chiptrip (EPM7032LC44-6)
Unassigned Nodes & Pins:
Color Legend
Unassigned
Device-Wide Fan-Out
Unrouted
Parallel Expanders
Local LAB Fan-Out Only
Selected Node(s) & Pin(s):
<none>
AoD
A
B
(Global OE)
(Global Clear)
(Global OE)
clock
|speed_ch:2|speed~1 @ LC16

The color legend shows the colors used to identify unassigned pins, logic cells, and I/O cells.

Zoom In button

Zoom Out button

Fit In Window button

Dedicated Global pins are shown separately from LABs for some devices.

The LAB View is currently displayed.

You can drag a node or pin name to the device to assign it to a pin, logic cell, LAB, I/O cell, embedded cell, row, column, or chip, depending on the selected view (all are currently assigned).

The Floorplan Editor provides a convenient method to enter and edit physical device resource assignments for your project. Two displays are available:

- The Device View shows all pins on a device package and their functions.
- The LAB View shows the interior of the device, including all Logic Array Blocks (LABs) and the individual logic cells within each LAB. In

devices that include Embedded Array Blocks (EABs), you can view individual embedded cells within each EAB. In MAX 9000, FLEX 6000, FLEX 8000, and FLEX 10K devices, I/O cell locations are also displayed. In addition, pins are displayed around the edges of the device packages.

The Floorplan Editor provides a list of unassigned node and pin names in your project. Each name has a handle that you can drag to an individual pin, logic cell, I/O cell, or embedded cell in the Device View or LAB View display. You can also drag a node or pin with an existing assignment back to the list of unassigned nodes or to a different location on the device.

You can also make a more general assignment to the assignment "bin" for an entire LAB, EAB, or a device, and then allow the Compiler to select the most appropriate location within the LAB or device. In MAX 9000, FLEX 6000, FLEX 8000, and FLEX 10K devices, you can also assign nodes and pins to row and column bins. Each generic assignment bin displays a number showing the number of nodes or pins assigned to it.

The following list highlights MAX+PLUS II Floorplan Editor features:

- You enter physical resource assignments in a graphical drag-and-drop environment. The Compiler's assignments can also be back-annotated and edited.
- A color legend clearly indicates unassigned and assigned pins, logic cells, and I/O cells; the type of fan-out from each item; and `VCC`, `GND`, and reserved pins.
- You can view and edit your current assignments, which are stored in the project's Assignment & Configuration File (**.acf**). You can also display a non-editable (read-only) view of the results of the last compilation, which are stored in the Fit File (**.fit**), regardless of whether fitting was successful. Any items with illegal assignments are highlighted in the list of unassigned node and pin names. Nodes that have been placed but not routed are indicated in red.
- You can automatically display the fan-in and fan-out of any selected item(s), or the paths between multiple selected items. You can also view detailed routing statistics for selected item(s) and for the most congested area of a chip.
- A Report File (**.rpt**) equation viewer allows you to select one or more items in the window and view their equations and the names of all nodes and pins that feed or are fed by any of the selected item(s). You

can select a fan-in or fan-out node and view its equation, tracing signal paths throughout the floorplan.

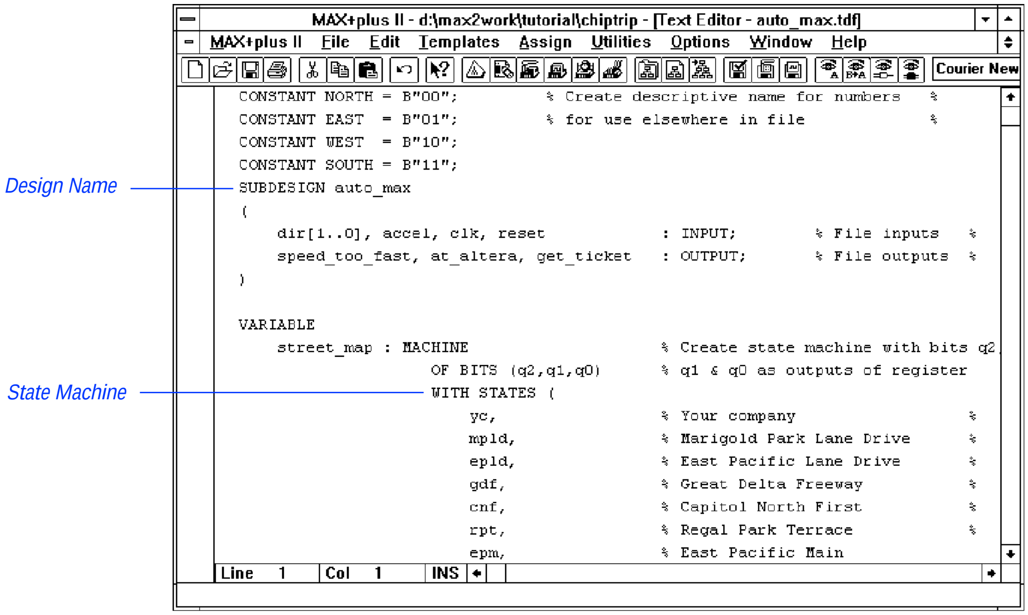
- The name of a logic function assigned to a pin or logic cell is displayed automatically in “balloon text” as the mouse pointer passes over it.
- If multiple items are assigned to a single location, you can view a list of all items and select a single item to edit.
- You can assign the same pin name to the output of one device and the input of another to control partitioning in a multi-device project.
- With the Compiler’s “smart recompile” feature, you can fine-tune assignments in the Floorplan Editor and quickly recompile.

The MAX+PLUS II Floorplan Editor provides many other features. For example, you can zoom in and out to various display scales, so that you can see an entire device at a glance or a detailed portion of it. You can also copy, cut, paste, and delete one or more selected assignments, and search for an assignment by its clique name, node name, or pin, logic cell, I/O cell, or embedded cell number.

Altera Hardware Description Language

The Altera Hardware Description Language (AHDL) is a high-level, modular language that is completely integrated into the MAX+PLUS II system. You can use the MAX+PLUS II Text Editor or another text editor to create AHDL Text Design Files (**.tdf**), which you compile and simulate in MAX+PLUS II. See Figure 2-14.

Figure 2-14. AHDL Text Design File



```
CONSTANT NORTH = B"00";         % Create descriptive name for numbers   %
CONSTANT EAST  = B"01";         % for use elsewhere in file             %
CONSTANT WEST  = B"10";
CONSTANT SOUTH = B"11";
SUBDESIGN auto_max
(
    dir[1..0], accel, clk, reset            : INPUT;        % File inputs   %
    speed_too_fast, at_altera, get_ticket   : OUTPUT;       % File outputs  %
)

VARIABLE
    street_map : MACHINE                    % Create state machine with bits q2
                OF BITS (q2,q1,q0)          % q1 & q0 as outputs of register
                WITH STATES (
                    yc,                     % Your company                  %
                    mpld,                   % Marigold Park Lane Drive      %
                    epld,                   % East Pacific Lane Drive       %
                    gdf,                    % Great Delta Freeway           %
                    cnf,                    % Capitol North First           %
                    rpt,                    % Regal Park Terrace            %
                    epm,                    % East Pacific Main
```

You can use any ASCII text editor to create TDFs. However, when you enter AHDL files with the MAX+PLUS II Text Editor, you can take advantage of the unique design entry, compilation, and debugging features available only in MAX+PLUS II editors. For example, you can take advantage of AHDL templates; use syntax coloring to easily view different sections of the file; get context-sensitive help about AHDL syntax elements, keywords, and statements, as well as Altera-provided primitives, megafunctions, and macrofunctions; make resource and device assignments; and use the MAX+PLUS II automatic error location feature during and after compilation.

AHDL consists of a variety of elements and behavioral statements that describe logic. The following list highlights the features that make AHDL an ideal tool for describing functions such as state machines, truth tables, Boolean equations, conditional logic, and group operations:

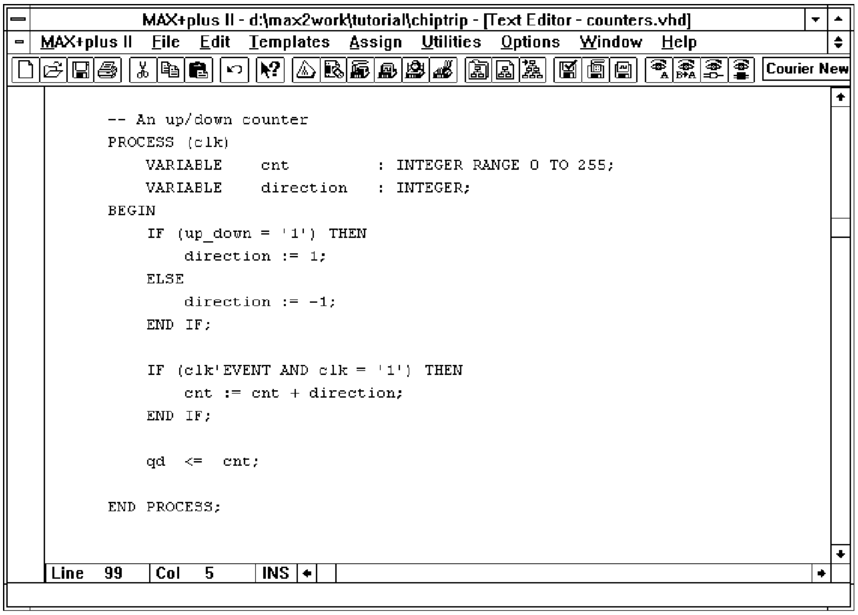
- You can use a variety of logic functions from the Library of Parameterized Modules (LPM) to implement logic. The LPM provides gate, arithmetic, and storage components that implement combinatorial logic such as decoders, multiplexers, and adders, and sequential logic such as registers and counters.
- As an alternative to using LPM functions, you can implement combinatorial and sequential logic with Boolean expressions and equations, macrofunctions, and truth tables.
- You can create your own parameterized designs in AHDL using iterative and conditional logic generation.
- You can store frequently used constants, evaluated functions, parameters, and function prototypes in Include Files (**.inc**) and incorporate them into any TDF.
- AHDL is ideal for state machine designs. The language is structured so that you can either assign state bits and state values yourself, or let the Compiler do the work for you. You can also import and export AHDL state machines between TDFs and other design files in a design hierarchy.
- The MAX+PLUS II Compiler can generate AHDL Text Design Export Files (**.tdx**) and Text Design Output Files (**.tdo**) when you compile a project. Regardless of your original design entry method, you can then rename the file as a TDF and use it to replace the original design file(s).

Go to MAX+PLUS II Help or to the ***MAX+PLUS II AHDL*** manual for complete information on AHDL.

VHDL

The Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) is a high-level, modular language that is completely integrated into the MAX+PLUS II system. VHDL is an industry-standard hardware description language that describes the inputs and outputs, behavior, and function of circuits. This language is defined by the IEEE 1076-1987 and 1076-1993 Standards. You can use the MAX+PLUS II Text Editor or another text editor to create VHDL Design Files (**.vhd**) in VHDL 1987 or 1993 syntax, which you compile and simulate in MAX+PLUS II. See Figure 2-15.

Figure 2-15. VHDL Design File


MAX+plus II - d:\max2work\tutorial\chiptrip - [Text Editor - counters.vhd]
MAX+plus II File Edit Templates Assign Utilities Options Window Help
Courier New
Line 99 Col 5 INS

```
-- An up/down counter
PROCESS (clk)
    VARIABLE    cnt          : INTEGER RANGE 0 TO 255;
    VARIABLE    direction    : INTEGER;
BEGIN
    IF (up_down = '1') THEN
        direction := 1;
    ELSE
        direction := -1;
    END IF;

    IF (clk'EVENT AND clk = '1') THEN
        cnt := cnt + direction;
    END IF;

    qd  <=  cnt;

END PROCESS;
```

You can use any ASCII text editor to create VHDL Design Files (**.vhd**). However, when you enter VHDL Design Files with the MAX+PLUS II Text Editor, you can take advantage of the unique design entry, compilation, and debugging features available only in MAX+PLUS II editors. For example, you can take advantage of VHDL templates; use MAX+PLUS II context-sensitive help to learn about Altera-provided primitives, megafunctions, and macrofunctions; use syntax coloring to easily view different sections of the file; make resource and device assignments; and use the MAX+PLUS II automatic error location feature during and after compilation.

VHDL Design Files can contain any combination of MAX+PLUS II-supported constructs. They can also contain Altera-provided primitives, megafunctions, and macrofunctions, i.e., lower-level design files, as well as user-defined mega- and macrofunctions.

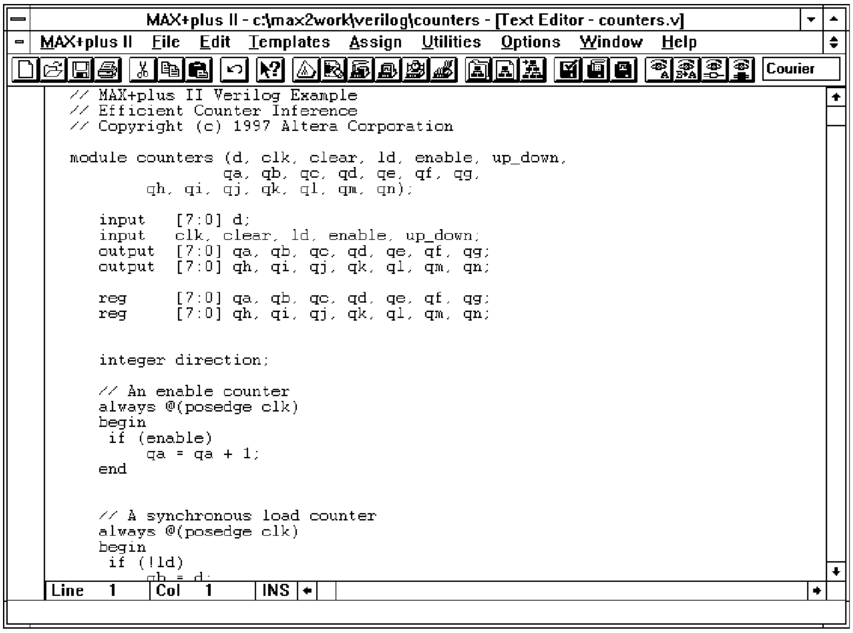
The MAX+PLUS II Compiler can generate VHDL Output Files (**.vho**) containing a project's post-synthesis functional and timing information. These files can be exported to an industry-standard simulator for simulation. Timing information can also be written to Standard Delay Format (SDF) Output Files (**.sdo**).

Go to MAX+PLUS II Help or to the ***MAX+PLUS II VHDL*** manual for information on MAX+PLUS II VHDL support.

Verilog HDL

The Verilog Hardware Description Language (HDL) is a high-level, modular language that is completely integrated into the MAX+PLUS II system. Verilog HDL is an industry-standard hardware description language that describes the inputs and outputs, behavior, and function of circuits. This language is defined by the IEEE Std 1364. You can use the MAX+PLUS II Text Editor or another text editor to create Verilog Design Files (**.v**), which you compile and simulate in MAX+PLUS II.
See Figure 2-16.

Figure 2-16. Verilog Design File



```
// MAX+plus II Verilog Example
// Efficient Counter Inference
// Copyright (c) 1997 Altera Corporation

module counters (d, clk, clear, ld, enable, up_down,
                qa, qb, qc, qd, qe, qf, qg,
        qh, qi, qj, qk, ql, qm, qn);

   input   [7:0] d;
   input   clk, clear, ld, enable, up_down;
   output  [7:0] qa, qb, qc, qd, qe, qf, qg;
   output  [7:0] qh, qi, qj, qk, ql, qm, qn;

   reg     [7:0] qa, qb, qc, qd, qe, qf, qg;
   reg     [7:0] qh, qi, qj, qk, ql, qm, qn;


   integer direction;

   // An enable counter
   always @(posedge clk)
   begin
    if (enable)
        qa = qa + 1;
   end


   // A synchronous load counter
   always @(posedge clk)
   begin
    if (!ld)
```

You can use any ASCII text editor to create Verilog Design Files (**.v**). However, when you enter Verilog Design Files with the MAX+PLUS II Text Editor, you can take advantage of the unique design entry, compilation, and debugging features available only in MAX+PLUS II editors. For example, you can take advantage of Verilog HDL templates; use MAX+PLUS II context-sensitive help to learn about Altera-provided primitives, megafunctions, and macrofunctions; use syntax coloring to easily view different sections of the file; make resource and device assignments; and use the MAX+PLUS II automatic error location feature during and after compilation.

Verilog Design Files can contain any combination of MAX+PLUS II-supported constructs. They can also contain Altera-provided logic functions—including primitives, megafunctions, and macrofunctions—as well as user-defined logic functions.

The MAX+PLUS II Compiler can generate Verilog Output Files (**.vo**) containing a project's post-synthesis functional and timing information. These files can be exported to an industry-standard simulator for simulation. Timing information can also be written to Standard Delay Format (SDF) Output Files (**.sdo**).

Go to MAX+PLUS II Help or to the ***MAX+PLUS II Verilog HDL*** manual for information on MAX+PLUS II Verilog HDL support.

Primitives, Megafunctions, & Macrofunctions

Altera provides libraries of logic functions—primitives, megafunctions, and old-style (e.g., 74-series) macrofunctions—including functions that are optimized for the architecture of a particular device family. During installation, all logic functions are copied to subdirectories of the **\maxplus2\max2lib** and **\maxplus2\vhdl*nn*** directories, where ***nn*** is "87" or "93." (On a UNIX workstation, the **maxplus2** directory is a subdirectory of the **/usr** directory.)

MAX+PLUS II Help provides extensive information on all Altera-provided logic functions, including the default input logic levels, an AHDL Function Prototype, VHDL Component Declaration, and a function table. You simply choose the context-sensitive Help button on the toolbar or press Shift+F1 in the Graphic or Text Editor, then click Button 1 on a logic function symbol or logic function name to open the appropriate Help topic.

Primitives

Primitives—buffer, flipflop, latch, input/output, and logic primitives—are basic functional blocks used to design circuits with MAX+PLUS II. They can be used in GDFs, and in AHDL, VHDL, and Verilog HDL design files.

Primitives in HDL design files are a subset of the primitive symbols used in GDFs. Other primitive functions can be represented by logical operators, ports, and various statements. AHDL Function Prototypes for primitives are built into the MAX+PLUS II software. VHDL Component Declarations for primitives are provided in the `maxplus2` package in the **altera** library.

In a Graphic Editor schematic, you can create a primitive array, in which a single primitive connected to one or more named bus lines represents a series of identical primitives. During project processing, the Compiler automatically translates a primitive array into the correct number of individual primitives. Primitive arrays provide an alternative to using parameterized functions.

Megafunctions

Megafunctions are complex or high-level building blocks that can be used together with primitives and other mega- and macrofunctions to create a logic design.

Many megafunctions, including functions from the Library of Parameterized Modules (LPM), are inherently parameterized for size, behavior, and silicon implementation. The scalability of LPM and other parameterized functions can greatly simplify design entry. Megafunctions can be used freely in GDFs and in all HDL design files. When the Compiler analyzes the complete logic circuit, it automatically uses any available device-family-specific megafunction logic, and removes all unused gates and flipflops to ensure optimum design efficiency.

Old-Style Macrofunctions

Old-style macrofunctions are high-level building blocks that can be used together with primitives and mega- and macrofunctions to create a logic design. They can be used freely in GDFs and in all HDL design files. When the Compiler analyzes the complete logic circuit, it automatically uses any available device-family-specific macrofunction logic, and removes all unused gates and flipflops to ensure optimum design efficiency. All macrofunction inputs also have default input signal levels so that unused pins can be left unconnected.

Many macrofunctions have bus equivalents, which are functionally identical to the macrofunction, but have input and output pins that are grouped into buses.

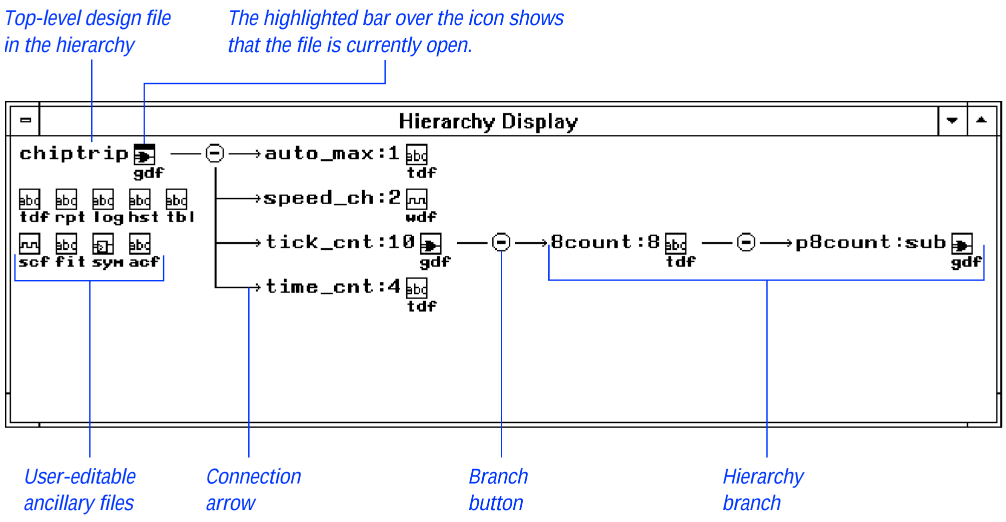
Old-style macrofunctions are not inherently parameterized. However, some Altera-specific parameters can be applied to macrofunctions to determine their style of implementation.

☞ Altera recommends using LPM megafunctions rather than equivalent old-style macrofunctions. LPM and other parameterized functions are easier to use, scalable, and are implemented more efficiently in silicon.

Project Hierarchy

The MAX+PLUS II Hierarchy Display shows a hierarchical logic design as a hierarchy tree where lower-level design files are represented as branches. Different design entry methods can be mixed in a single project. See Figure 2-17.

Figure 2-17. MAX+PLUS II Hierarchy Display



When you open the Hierarchy Display, it shows the full hierarchy of design files—called a "hierarchy tree"—for the current project or another hierarchy of design files. If one or more files in the hierarchy are open, the top of its file icon displays a highlighted bar. The Hierarchy Display shows the entire hierarchy of design files, as well as all user-editable ancillary files for the top-level design file, if the project has been compiled with the Compiler Netlist Extractor module.

The Hierarchy Display features make it easy for you to move between the different types of files for a project. For example, you can open and close one or more files in the Hierarchy Display window; the appropriate editors are then automatically opened or closed. You can also zoom in and out to various display scales to see all or part of the hierarchy, or choose a compact display to view as many branches as possible of a large hierarchy tree.

In addition, the Hierarchy Display offers the following features:

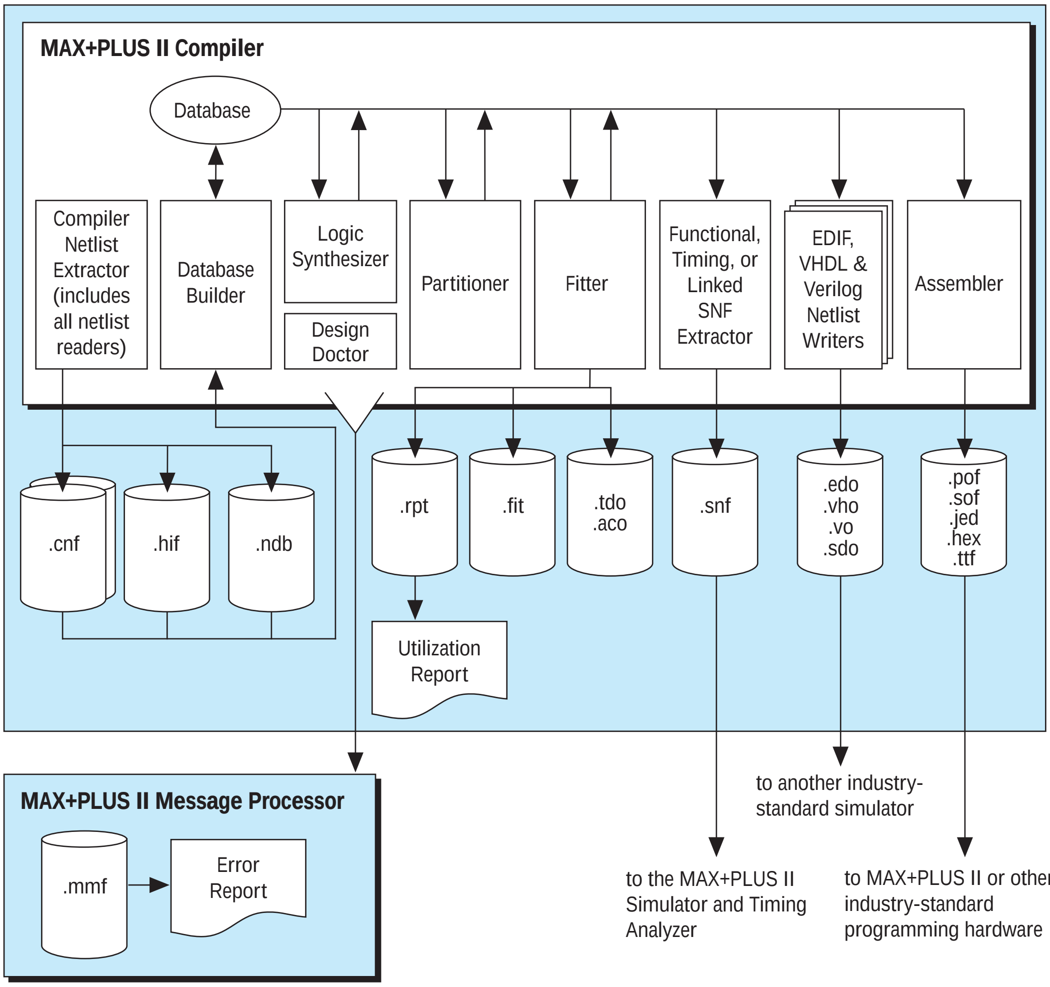
- You can easily open an editor window onto any design or ancillary file in the current hierarchy.
- Branch buttons at the intersections between hierarchy tree branches allow you to hide or display the lower-level branches.
- All filenames in the hierarchy tree are accompanied by the appropriate MAX+PLUS II editor icon and filename extension. The top-level file also shows icons and filename extensions for one or more ancillary files.
- When a file is open, a highlighted bar is displayed over the file icon. The highlighting disappears when you close the file.
- You can select a design file and view its physical implementation in the LAB View of the Floorplan Editor.
- You can select a design file and enter resource assignments for the entire file. This behavior is analogous to entering assignments on a symbol in a Graphic Editor file.
- You can display any one of multiple open hierarchies.
- You can display the hierarchy in horizontal or vertical orientation.
- You can print the current hierarchy tree or any combination of project design files and ancillary files.

Go to MAX+PLUS II Help for complete information on all MAX+PLUS II Hierarchy Display functions and features.

Project Processing

MAX+PLUS II processes projects for Altera Classic, MAX 5000, MAX 7000, MAX 9000, FLEX 6000, FLEX 8000, and FLEX 10K devices. MAX+PLUS II compiles projects automatically, but you can also make detailed processing specifications. Figure 2-18 shows how MAX+PLUS II compiles projects.

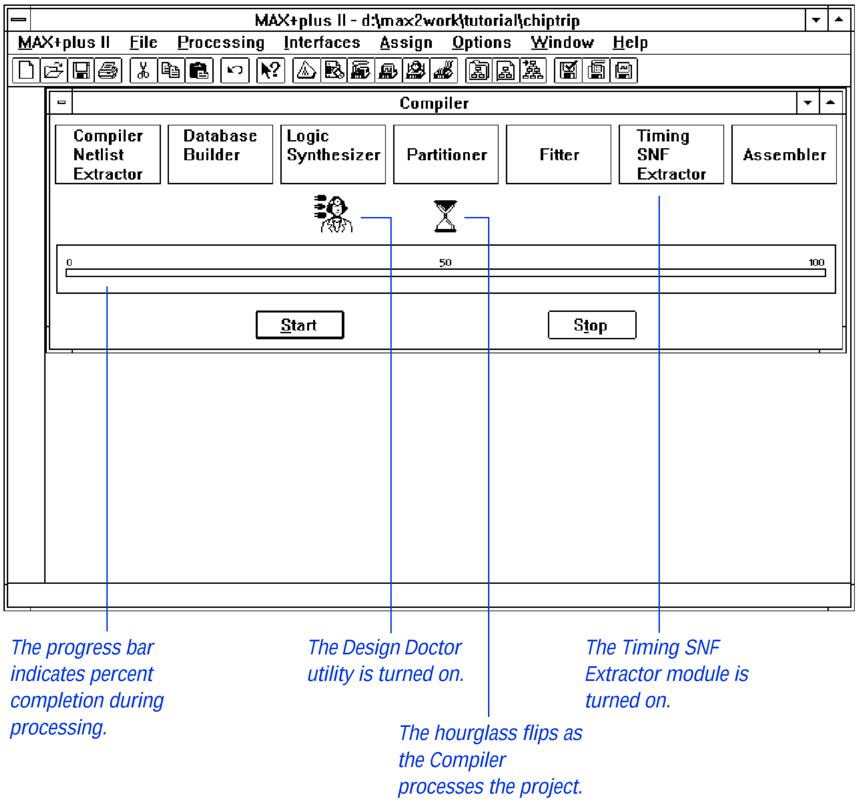
Figure 2-18. Project Processing


MAX+PLUS II Compiler
Database
Compiler Netlist Extractor (includes all netlist readers)
Database Builder
Logic Synthesizer
Design Doctor
Partitioner
Fitter
Functional, Timing, or Linked SNF Extractor
EDIF, VHDL & Verilog Netlist Writers
Assembler
.cnf
.hif
.ndb
.rpt
.fit
.tdo
.aco
.snf
.edo
.vho
.vo
.sdo
.pof
.sof
.jed
.hex
.ttf
Utilization Report
to another industry-standard simulator
to the MAX+PLUS II Simulator and Timing Analyzer
to MAX+PLUS II or other industry-standard programming hardware
MAX+PLUS II Message Processor
.mmf
Error Report

MAX+PLUS II Compiler

The MAX+PLUS II Compiler consists of a series of modules and a utility that check a project for errors, synthesize the logic, fit the project into one or more Altera devices, and generate output files for simulation, timing analysis, and device programming. The Compiler links the MAX+PLUS II design entry applications—the Graphic, Text, Waveform, Symbol, and Floorplan Editors—with the post-processing Timing Analyzer, Simulator, and Programmer applications. Figure 2-19 shows the MAX+PLUS II Compiler window.

Figure 2-19. MAX+PLUS II Compiler



MAX+plus II - d:\max2work\tutorial\chiptrip
MAX+plus II
File
Processing
Interfaces
Assign
Options
Window
Help
Compiler
Compiler Netlist Extractor
Database Builder
Logic Synthesizer
Partitioner
Fitter
Timing SNF Extractor
Assembler
0
50
100
Start
Stop
The progress bar indicates percent completion during processing.
The Design Doctor utility is turned on.
The Timing SNF Extractor module is turned on.
The hourglass flips as the Compiler processes the project.

Compiler Input Files

The MAX+PLUS II Compiler uses the following input files:

- Graphic Design Files (**.gdf**) created with the MAX+PLUS II Graphic Editor.
- Text Design Files (**.tdf**) created in the Altera Hardware Description Language (AHDL).
- VHDL Design Files (**.vhd**) created in VHDL 1987 or 1993 syntax.
- Verilog Design Files (**.v**) created in the Verilog HDL.
- EDIF version 2 0 0 or 3 0 0 Input Files (**.edf**) generated with any standard EDIF netlist writer.
- Waveform Design Files (**.wdf**) created with the MAX+PLUS II Waveform Editor.
- OrCAD Schematic Files (**.sch**) created with the OrCAD Draft schematic editor or with the MAX+PLUS II Graphic Editor.
- Xilinx Netlist Format Files (**.xnf**) created with Xilinx software.
- Altera Design Files (**.adf**) created with Altera's A+PLUS software. ADFs use a netlist format and Boolean equations to describe a design. The MAX+PLUS II Compiler automatically translates an ADF into a Compiler Netlist File (**.cnf**) during project compilation.
- State Machine Files (**.smf**) that contain a state machine design created for use with Altera's A+PLUS or SAM+PLUS software. The MAX+PLUS II Compiler automatically translates an SMF into an Altera Design File (**.adf**) and a Compiler Netlist File (**.cnf**) during compilation.
- Hexadecimal (Intel-format) Files (**.hex**) and/or Memory Initialization Files (**.mif**) containing the initial values for a memory block.
- An EDIF Command File (**.edc**) used to customize the format of EDIF Output Files (**.edo**) created by the MAX+PLUS II Compiler.

- An Assignment & Configuration File (**.acf**) that stores the project's probe, resource, and device assignments, as well as configuration settings for the Compiler, Simulator, and Timing Analyzer.

  ☞ Assignment and configuration information from pre-version 5.0 releases of MAX+PLUS II—which was stored in TDFs, Probe & Resource Assignment Files (**.prb**), and *<project name>***.ini** files—can be converted automatically to the ACF format.

- Symbol Files (**.sym**) created with the MAX+PLUS II Symbol Editor.

- Include Files (**.inc**) that are imported into an AHDL Text Design File (**.tdf**) by an AHDL Include Statement. The Include File replaces the Include Statement that calls it. Include Files can contain Function Prototype, Define, Parameters, or Constant Statements. The Compiler also uses AHDL Function Prototypes in Include Files to process logic functions in Verilog Design Files (**.v**).

- Library Mapping Files (**.lmf**) used to map cells in EDIF Input Files and OrCAD Schematic Files to corresponding MAX+PLUS II logic functions.

Compilation Process

The Compiler first extracts information that defines the hierarchical connections between a project's design files and checks the project for basic design entry errors. It creates an organizational map of the project and then combines all design files into a fully flattened database that can be processed efficiently.

The Compiler applies a variety of techniques to increase the efficiency of your project and minimize device resource usage. If your project is too large to fit into a single device, the Compiler can automatically partition it into multiple devices from the same device family, while minimizing the number of connections between devices. A Report File (**.rpt**) then shows how a project will be implemented in one or more devices.

The Compiler also creates programming files that the MAX+PLUS II Programmer or another industry-standard programmer uses to program one or more Altera devices.

While the Compiler can compile a project with minimal assistance, it also allows you to customize design processing to your exact specifications. For example, you can specify a default project logic synthesis style and other project-wide logic synthesis settings that tailor logic synthesis to your needs, enter project-wide timing requirements, specify precisely how to divide a large project into multiple devices, and set various device options on a project-wide basis. You can also choose how many pins and logic cells must remain unused during the current compilation to reserve additional logic capacity for future use.

Running the Compilation

You can start project compilation from any MAX+PLUS II application or from the Compiler. The Compiler automatically processes all input files for the current project, and you can monitor the compilation process in the Compiler window:

- The hourglass empties and flips, indicating that the Compiler is active.
- The module boxes are highlighted in turn as the Compiler completes each stage of processing.
- Icons representing output files appear below the boxes representing the Compiler modules that generate them. You can double-click Button 1 on an icon to open the corresponding file.
- The percent completion shown in the progress bar moves toward 100%.
- During partitioning and fitting, the Compiler's **Stop** button turns into a **Stop/Show Status** button, which you can choose to open a dialog box that shows the project's current partitioning and fitting status.
- If any errors or potential problems are detected during compilation, the Message Processor window opens automatically; lists information, error, and warning messages; and provides immediate help on how to correct an error; and allows you to locate message sources in the project's design files or its assignments floorplan.

The Compiler can run in the background. You can minimize it while it is processing a project, and continue working on other files. A progress bar under the minimized Compiler icon allows you to keep an eye on the compilation progress while you focus your attention on a different task.

Compiler Modules & Output Files

The MAX+PLUS II Compiler processes a project with the following modules and utilities:

- Compiler Netlist Extractor (including built-in EDIF, VHDL, Verilog, and XNF Netlist Readers)
- Database Builder
- Logic Synthesizer
- Partitioner
- Fitter
- Functional SNF Extractor
- Timing SNF Extractor
- Linked SNF Extractor
- EDIF Netlist Writer
- Verilog Netlist Writer
- VHDL Netlist Writer
- Assembler
- Design Doctor Utility

Compiler Netlist Extractor (Including Built-In EDIF Netlist Reader, VHDL Netlist Reader, Verilog Netlist Reader & XNF Netlist Reader)

The Compiler Netlist Extractor converts each design file in the project into one or more binary Compiler Netlist Files (**.cnf**). Because the Compiler Netlist Extractor resolves the values of any parameters used in parameterized functions, the contents of a CNF can change in a subsequent compilation if the parameter values change. The Compiler Netlist Extractor also creates a Hierarchy Interconnect File (**.hif**) that documents the hierarchical connections between the project files, and provides the information necessary to show the project's hierarchy tree in the Hierarchy Display. In addition, the Compiler Netlist Extractor generates the Node Database File (**.ndb**) that contains project node names for the resource assignment database.

The built-in EDIF, VHDL, Verilog HDL, and XNF netlist readers automatically translate the design information in EDIF Input Files (**.edf**), VHDL Design Files (**.vhd**), Verilog Design Files (**.v**), and Xilinx Netlist Format Files (**.xnf**), respectively, into a MAX+PLUS II-compatible format. The EDIF Netlist Reader processes EDIF Input Files with the help of Library Mapping Files (**.lmf**) that map logic functions provided with other industry-standard EDA tools to MAX+PLUS II functions. The XNF Netlist Reader optionally generates a Text Design Export File (**.tdx**) that contains the AHDL equivalent of a Xilinx Netlist Format File (**.xnf**) so that you can easily edit your project in AHDL.

Database Builder

The Database Builder uses the HIF to link the CNFs that describe the project. Based on HIF data, the Database Builder copies each CNF into a single, fully flattened project database. The database thus preserves the electrical connectivity of the project.

As it creates the database, the Database Builder examines the logical completeness and consistency of the project, and checks for boundary connectivity and syntactical errors (e.g., a node without a source or destination). Most errors are detected and can be easily corrected at this stage of processing. Each Compiler module subsequently processes and updates this database.

The first time the Compiler processes a project, all design files of that project are compiled. You can use the Compiler's "smart recompile" feature to create an expanded project database that helps to accelerate subsequent compilations. This database allows you to change physical device resource assignments, such as pin and logic cell assignments, and recompile the project without rebuilding the database and resynthesizing the project logic. With the "total recompile" feature, you can choose between recompiling only those files that have been edited since the last compile or fully recompiling the project.

Logic Synthesizer

The Logic Synthesizer module applies a number of algorithms that reduce resource usage and remove redundant logic to ensure that the logic cell structure is used as efficiently as possible for the architecture of the target device family. This Compiler module also applies logic synthesis techniques to help implement user-specified timing and other implementation requirements. In addition, the Logic Synthesizer searches the logic for unconnected nodes. If it finds an unconnected node, it removes the primitives associated with that node.

A large number of logic options, as well as three "ready-made" synthesis styles, are available to help you guide the outcome of logic synthesis.

You can enter timing assignments and logic option assignments, and define logic synthesis styles, in any MAX+PLUS II application. You can specify global default logic synthesis and timing options for an entire project, and entire additional logic option and timing assignments on individual logic functions.

Go to "Global Project Timing Requirements" and "Global Project Logic Synthesis" on page 100 for more information.

Partitioner

If a project does not fit into a single device, the Partitioner divides the database updated by the Logic Synthesizer into multiple devices from the same device family, attempting to split the project into the smallest possible number of devices. A project is partitioned along logic cell boundaries, and the number of pins used for inter-device communication is minimized.

Partitioning can be totally automatic, partially user-controlled, or fully user-controlled. Device assignments and automatic device selection settings allow you to exercise the level of control that is appropriate for your project.

While the Partitioner and Fitter modules are running, you can pause the compilation. The Compiler then displays information on the current status of the partitioning and fitting process, including a comparison of required and available resources, so that you can decide whether to continue the compilation.

Fitter

Using the database updated by the Partitioner, the Fitter matches the requirements of the project with the known resources of one or more devices. It assigns each logic function to the best logic cell location and selects appropriate interconnection paths and pin assignments. The Fitter attempts to match your resource assignments—i.e., the pin, logic cell, I/O cell, embedded cell, chip, clique, device, local routing, timing, and connected pin assignments in the project's Assignment & Configuration File (**.acf**)—with the available resources. The Fitter provides options to allow you to customize its fitting techniques to help achieve a fit, for example, by automatically inserting logic cells or limiting fan-in. If it cannot find a fit, the Fitter issues a message and gives you the option of ignoring some or all of your assignments or terminating compilation.

Regardless of whether a fit is achieved, the Fitter generates a Report File (**.rpt**) that documents fitting information on project partitioning, input and output pin names, project timing, and unused resources for each device in the project. You can optionally include Report File sections that show user assignments, file hierarchy, logic cell interconnections, and equations.

The Compiler also automatically generates a Fit File (**.fit**) that documents resource and device assignments for the entire project, as well as routing information. Regardless of whether a successful fit was achieved, you can view the fitting, partitioning, and routing information from the Fit File with the Floorplan Editor. You can also back-annotate Fit File assignments to the project's ACF for editing.

You can optionally direct the Fitter to generate AHDL Text Design Output Files (**.tdo**) for the fully optimized, fitted project. Since one file is generated for each device in a multi-device project, you can split your project into multiple single-device projects if you wish to "lock down" the logic in some of the devices. You can then edit the logic for a single device, save the TDO File for that device as a Text Design File (**.tdf**), and recompile the logic for that device while maintaining the logic synthesis results of a previous compilation.

Functional SNF Extractor

The optional Functional SNF Extractor creates the functional Simulator Netlist File (**.snf**) required for functional simulation. The Compiler generates this file before it synthesizes the project; therefore, it contains all nodes present in the original design files. The functional SNF does not contain timing information, but is generated quickly. The file is created only if a project compiles without errors.

Timing SNF Extractor

The optional Timing SNF Extractor creates the timing Simulator Netlist File (**.snf**), which contains the timing data for the fully optimized project. This file is used for timing simulation and timing analysis. The Compiler's EDIF Netlist Writer, Verilog Netlist Writer, and VHDL Netlist Writer modules also use timing SNFs to generate EDIF, Verilog HDL, and VHDL output files, as well as optional Standard Delay Format (SDF) Output Files (**.sdo**). The timing SNF is created only if a project compiles without errors.

You can optionally instruct the Compiler to generate an optimized SNF containing dynamic models that represent types of combinatorial logic. Optimizing the SNF increases the compilation time, but can save you time during simulation and timing analysis.

Linked SNF Extractor

The optional Linked SNF Extractor creates a linked Simulator Netlist File (**.snf**), which contains the functional and/or timing data for multi-project, board-level-type simulation. The linked SNF combines information from the timing SNFs and/or functional SNFs for multiple separate projects. Projects that are linked can use devices from different device families. If a linked SNF contains timing information only, you can also use it to run a timing analysis. The file is created only if a project compiles without errors.

Go to "Project Verification" on page 141 and to MAX+PLUS II Help for more information on the MAX+PLUS II Simulator.

EDIF Netlist Writer

The MAX+PLUS II Compiler can interface with most industry-standard CAE tools that can read a netlist file in the EDIF 2 0 0 or 3 0 0 standard format. The optional EDIF Netlist Writer produces one or more EDIF Output Files (**.edo**) containing post-synthesis functional and optional timing information. Timing information can also be written to separate Standard Delay Format (SDF) Output Files (**.sdo**). These files can be used with an industry-standard simulator. EDIF and SDF output files are created only if a project compiles without errors.

Verilog Netlist Writer

The optional Verilog Netlist Writer produces one or more Verilog Output Files (**.vo**) that contain the project's post-synthesis functional and optional timing information. Timing information can also be written to separate SDF Output Files. These files can be used with an industry-standard Verilog HDL simulator. Verilog HDL and SDF output files are created only if a project compiles without errors.

VHDL Netlist Writer

The optional VHDL Netlist Writer produces one or more VHDL Output Files (**.vho**), in VHDL 1987 or 1993 syntax, which contain the project's post-synthesis functional and optional timing information. Timing information can also be written to separate SDF Output Files. These files can be used with an industry-standard VHDL simulator. VHDL and SDF output files are created only if a project compiles without errors.

Assembler

The Assembler converts the Fitter's logic cell, pin, and device assignments into a programming image for the device(s) in the form of one or more binary Programmer Object Files (**.pof**) or SRAM Object Files (**.sof**); for some devices, the Compiler also generates JEDEC Files (**.jed**), Tabular Text Files (**.ttf**), and Hexadecimal (Intel-format) Files (**.hex**). The POFs, SOFs, or JEDEC Files are then processed by the MAX+PLUS II Programmer and Altera programming hardware, or another industry-standard programmer, to produce working devices. The Hex Files and TTFs can be used to configure FLEX 6000, FLEX 8000, and FLEX 10K devices by other means. The Assembler creates programming files only if the project compiles without errors.

After compilation has been completed, the MAX+PLUS II Compiler and Programmer allow you to generate additional device programming files for use in other programming environments. For example, you can create Serial Bitstream Files (**.sbf**) and Raw Binary Files (**.rbf**) for configuring FLEX 6000, FLEX 8000, and FLEX 10K devices. You can also create Serial Vector Format Files (**.svf**) and Jam Files (**.jam**) for programming devices in automated test equipment (ATE)-type and embedded processor-type programming environments, respectively.

Design Doctor Utility

The optional Design Doctor utility checks each design file for logic that may cause system-level reliability problems that are usually discovered only after a design has entered production. You can choose one of three predefined sets of design rules with different levels of design-rule checking, or create a custom set of design rules.

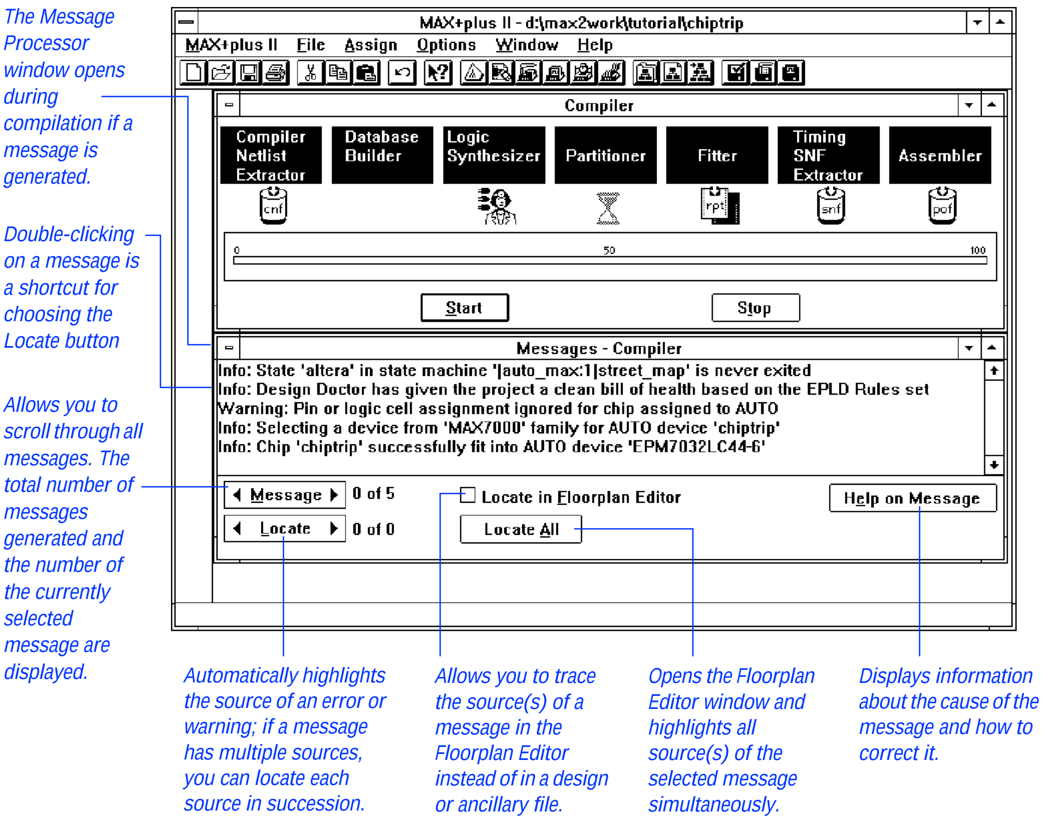
Design rules are based on reliability guidelines that cover logic containing features such as asynchronous inputs, ripple Clocks, multi-level logic on Clocks, Preset and Clear configurations, and race conditions. MAX+PLUS II Help explains rule violations to help you determine which edits are needed in the design files.

Go to MAX+PLUS II Help for complete information on all Compiler functions and features.

Error Detection & Location

The MAX+PLUS II Message Processor communicates with all MAX+PLUS II applications, recording error, information, and warning messages as you process, verify, or program a project. If you double-click Button 1 on a message, the Message Processor automatically opens the design file, ancillary file, or the project floorplan that contains the source of the message, and highlights the location of the error. See Figure 2-20.

Figure 2-20. MAX+PLUS II Message Processor


The Message Processor window opens during compilation if a message is generated.
Double-clicking on a message is a shortcut for choosing the Locate button
Allows you to scroll through all messages. The total number of messages generated and the number of the currently selected message are displayed.
Automatically highlights the source of an error or warning; if a message has multiple sources, you can locate each source in succession.
Allows you to trace the source(s) of a message in the Floorplan Editor instead of in a design or ancillary file.
Opens the Floorplan Editor window and highlights all source(s) of the selected message simultaneously.
Displays information about the cause of the message and how to correct it.
MAX+plus II - d:\max2work\tutorial\chiptrip
MAX+plus II
File
Assign
Options
Window
Help
Compiler
Compiler Netlist Extractor
Database Builder
Logic Synthesizer
Partitioner
Fitter
Timing SNF Extractor
Assembler
cnf
rpt
snf
pof
0
50
100
Start
Stop
Messages - Compiler
Message
0 of 5
Locate
0 of 0
Locate in Floorplan Editor
Locate All
Help on Message

```
Info: State 'altera' in state machine '|auto_max:1|street_map' is never exited
Info: Design Doctor has given the project a clean bill of health based on the EPLD Rules set
Warning: Pin or logic cell assignment ignored for chip assigned to AUTO
Info: Selecting a device from 'MAX7000' family for AUTO device 'chiptrip'
Info: Chip 'chiptrip' successfully fit into AUTO device 'EPM7032LC44-6'
```

The Message Processor offers the following features:

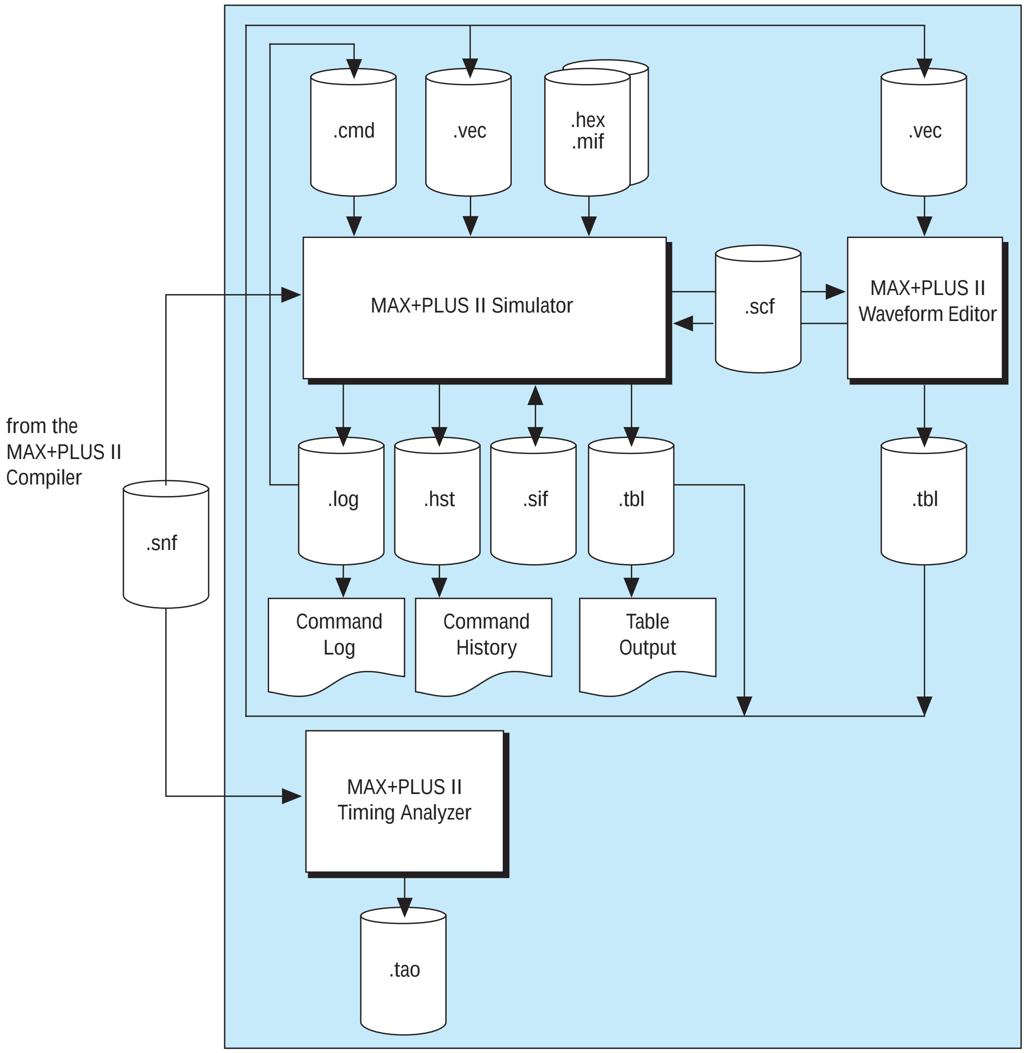
- If a message is caused by a problem in multiple locations, the Message Processor allows you to find each location.
- You can locate the source of a message in the project design files and ancillary files (e.g., in a Simulator Channel File), or locate to the floorplan for the project.
- When you list signal paths with the **List Paths** button in the MAX+PLUS II Timing Analyzer, the Message Processor displays messages that show all propagation delays between a pair of nodes.
- If you choose to locate messages in the project floorplan, you can locate each source of a message in succession, or locate all sources simultaneously. Locating all sources simultaneously provides a convenient method for viewing critical timing paths.
- You can print the current messages or save the messages in an ASCII Message Text File (**.mtf**).
- When a message is highlighted in the Message Processor window and you choose the **Help on Message** button, you get detailed information on the likely cause of the message and the suggested corrective action.

Go to MAX+PLUS II Help for complete information on all Message Processor functions and features.

Project Verification

MAX+PLUS II provides three applications—the MAX+PLUS II Simulator, Timing Analyzer, and Waveform Editor—to help you test the logic of a compiled project. See Figure 2-21.

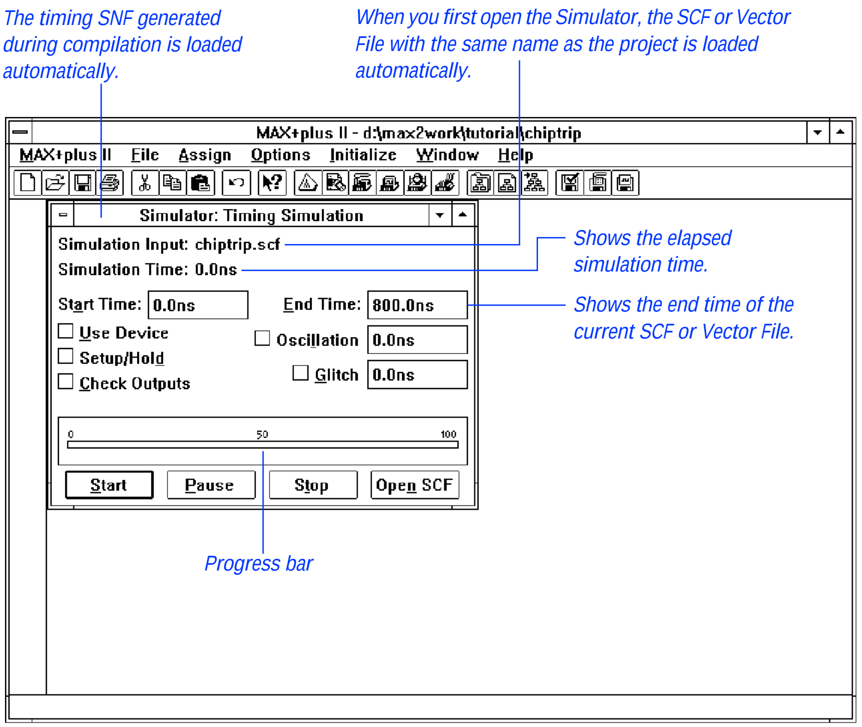
Figure 2-21. MAX+PLUS II Project Verification



MAX+PLUS II Simulator

The MAX+PLUS II Simulator tests the logical operation and internal timing of a project, allowing you to model a circuit design before it is programmed into a device. You can run the Simulator either in interactive mode or in batch mode. Figure 2-22 shows the Simulator window.

Figure 2-22. MAX+PLUS II Simulator


The timing SNF generated during compilation is loaded automatically.
When you first open the Simulator, the SCF or Vector File with the same name as the project is loaded automatically.
MAX+plus II - d:\max2work\tutorial\chiptrip
MAX+plus II
File
Assign
Options
Initialize
Window
Help
Simulator: Timing Simulation
Shows the elapsed simulation time.
Shows the end time of the current SCF or Vector File.
Use Device
Setup/Hold
Check Outputs
Oscillation
Glitch
0
50
100
Start
Pause
Stop
Open SCF
Progress bar

Simulation Input: chiptrip.scf
Simulation Time: 0.0ns
Start Time: 0.0ns
End Time: 800.0ns
Oscillation: 0.0ns
Glitch: 0.0ns

To simulate a project, you must first compile it and instruct the Compiler to generate a Simulator Netlist File (**.snf**) for functional, timing, or linked multi-project simulation. The functional, timing, or linked SNF for the current project is then loaded automatically when you open the Simulator.

The Simulator uses a graphical waveform Simulator Channel File (**.scf**) or an ASCII Vector File (**.vec**) as the source of input vectors. For projects that contain memory, you can specify the initial memory contents with Hexadecimal (Intel-format) Files (**.hex**) or with Memory Initialization Files **(.mif**). The Waveform Editor can automatically create a default SCF, which you can edit to provide the desired input vectors; if you create a Vector File instead, the Simulator automatically generates an SCF from it.

Go to *Vector File Format* in MAX+PLUS II Help for a description of the Vector File format. You should also try out "Session 10: Simulate the Project" on page 255 in the *MAX+PLUS II Tutorial*.

The Simulator allows you to check the outputs of simulation against any outputs in the SCF, such as user-defined expected outputs or outputs from a previous simulation. With appropriate programming hardware, you can also perform functional testing to test the actual outputs of a programmed device against the simulation outputs.

Using options in the Simulator, you can monitor your project for glitches, oscillations, and setup and hold time violations. Once simulation is completed, you can open the Waveform Editor to view the updated SCF or save the outputs to a Table File (**.tbl**) and view the results in the Text Editor.

Functional Simulation

When the MAX+PLUS II Compiler creates a functional SNF, it generates the SNF before it synthesizes the project. Consequently, in a functional simulation, all nodes in the project can be simulated.

During functional simulation, the Simulator ignores all propagation delays. Because there are no delays in the functional SNF, output logic levels change at the same time as the input vectors.

Timing Simulation

When the MAX+PLUS II Compiler creates a timing SNF, it generates the SNF after the project has been fully synthesized and optimized. Therefore, a timing SNF contains only those nodes that have not been eliminated during logic synthesis.

The Simulator uses the information from the timing SNF, which contains hardware information from Device Model Files (**.dmf**) provided with MAX+PLUS II, to simulate the project.

If a project has been partitioned into two or more devices, the Compiler creates an SNF for the project as a whole and for each device. However, timing simulation is performed for the entire project only.

You can accelerate timing simulation by instructing the Compiler to generate an optimized SNF containing dynamic models that represent various types of combinatorial logic. The Compiler's processing time increases when it generates the optimized SNF; however, the resulting SNF can reduce simulation time, because the Simulator can refer to a representative dynamic model instead of interpreting all logic in a combinatorial network.

Linked Multi-Project Simulation

When the MAX+PLUS II Compiler creates a linked SNF, it combines the functional and/or timing SNFs for multiple individual projects. The separate "sub-projects" in the linked SNF can be targeted for different Altera device families. In addition, because functional SNFs are not fully compiled, you can incorporate sub-projects that represent logic that is not implemented in an Altera device.

You can use the linked SNF to perform a board-level-type simulation. In addition, if the linked SNF contains timing information only, you can use it to run a timing analysis in the MAX+PLUS II Timing Analyzer.

Simulator Highlights

Together with other MAX+PLUS II applications, the Simulator lets you perform the following tasks:

- Specify expected output logic levels that can be compared to simulation outputs.
- Simulate individual or grouped nodes. You can combine bits of a state machine in the project, simulate them as a group, and refer to them with a state name.

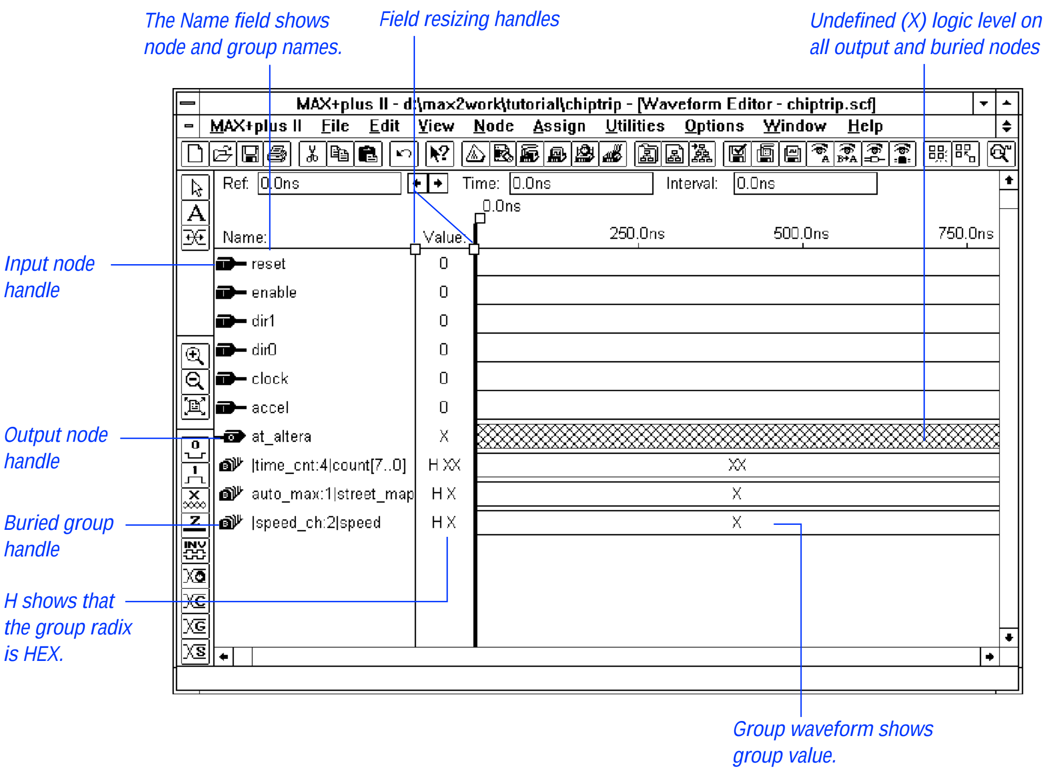
- Define a time interval that constitutes an oscillation or glitch, and analyze the project for one or both conditions.
- Monitor the project for register setup and hold time violations.
- Record actual device outputs instead of simulation outputs.
- Perform functional testing. You can check whether simulation outputs are functionally equivalent to actual device outputs.
- Create breakpoint conditions that cause the Simulator to pause when these conditions are met during simulation.
- List the name and logic level of any combination of nodes and groups and initialize node and group logic levels before simulation.
- Initialize the contents of memory blocks (RAM or ROM) before simulation.
- Save initialized node and group values, including initialized memory values, in a Simulator Initialization File (**.sif**), or reload the initialized values stored in the file.
- Log Simulator commands to an ASCII Log File (**.log**) with the same format as the Command File (**.cmd**) used for batch-mode simulation, and use the Log File to repeat an earlier simulation. You can also record Simulator commands and their output in an ASCII History File (**.hst**).

Go to MAX+PLUS II Help for complete information on all MAX+PLUS II Simulator functions and features.

MAX+PLUS II Waveform Editor

The MAX+PLUS II Waveform Editor is used as a design entry tool and as a tool for entering input vectors and viewing simulation results. You can create Waveform Design Files (**.wdf**) that contain design logic for the project and Simulator Channel Files (**.scf**) that contain input vectors for simulation. See Figure 2-23.

Figure 2-23. MAX+PLUS II Waveform Editor


The Name field shows node and group names.
Field resizing handles
Undefined (X) logic level on all output and buried nodes
MAX+plus II - d:\max2work\tutorial\chiptrip - [Waveform Editor - chiptrip.scf]
MAX+plus II File Edit View Node Assign Utilities Options Window Help
Ref: 0.0ns
Time: 0.0ns
Interval: 0.0ns
0.0ns
Name:
Value:
250.0ns
500.0ns
750.0ns
Input node handle
reset 0
enable 0
dir1 0
dir0 0
clock 0
accel 0
Output node handle
at_altera X
|time_cnt:4|count[7..0] H XX
XX
auto_max:1|street_map H X
X
Buried group handle
|speed_ch:2|speed H X
X
H shows that the group radix is HEX.
Group waveform shows group value.

To simulate a project, you must provide input vectors. With an SCF, you can describe simulation input vectors as waveforms, which are a graphical alternative to the ASCII Vector File (**.vec**) input for the Simulator.

You create an SCF containing the input vector waveforms that will drive simulation, and the buried node and output node names to be simulated. Buried and output nodes have undefined logic levels, or can be edited to include expected logic values. The Waveform Editor can use the Simulator Netlist File (**.snf**) to create a default SCF that contains some or all nodes and groups in the compiled project. You can edit this SCF to meet your specifications or you can create an SCF "from scratch." In addition, you can import a Vector File to automatically create its graphical waveform equivalent.

With the Waveform Editor, you can view and interpret the outputs of simulation in an SCF. The Simulator generates an SCF automatically if a Vector File is the source of input vectors; otherwise, the SCF that was the source of vectors is simply updated during simulation. Buried logic and output node waveforms are overwritten with logic levels based on simulation inputs.

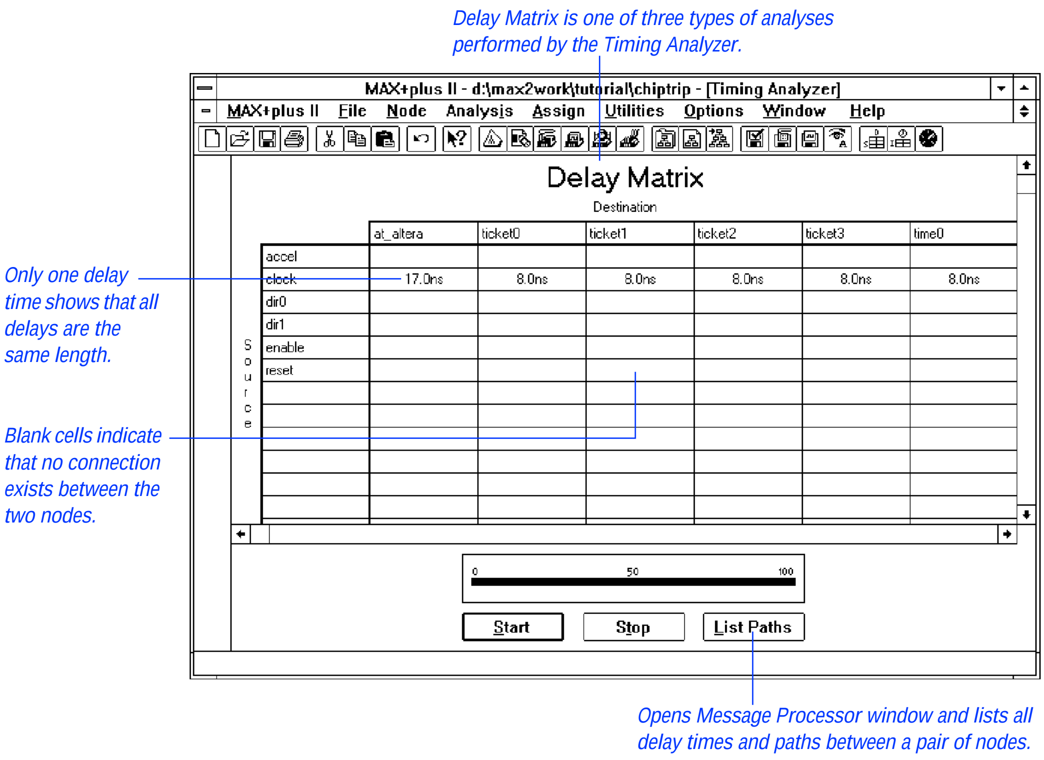
For more details, go to "MAX+PLUS II Waveform Editor" on page 146.

Go to MAX+PLUS II Help for complete information on the MAX+PLUS II Waveform Editor.

MAX+PLUS II Timing Analyzer

With the MAX+PLUS II Timing Analyzer, you can analyze the timing performance of a project after it has been optimized by the Compiler. You can trace all signal paths in the project, determining critical speed paths and paths that limit the project's performance. See Figure 2-24.

Figure 2-24. MAX+PLUS II Timing Analyzer


MAX+plus II - d:\max2work\tutorial\chiptrip - [Timing Analyzer]
MAX+plus II File Node Analysis Assign Utilities Options Window Help
Delay Matrix
Destination
Source
0 50 100
Start
Stop
List Paths

Delay Matrix is one of three types of analyses performed by the Timing Analyzer.

Only one delay time shows that all delays are the same length.

Blank cells indicate that no connection exists between the two nodes.

Opens Message Processor window and lists all delay times and paths between a pair of nodes.

	at_altera	ticket0	ticket1	ticket2	ticket3	time0
accel						
clock	17.0ns	8.0ns	8.0ns	8.0ns	8.0ns	8.0ns
dir0						
dir1						
enable						
reset						

The Timing Analyzer uses the network and timing information from a timing Simulator Netlist File (**.snf**) generated by the Compiler. The Timing Analyzer can also use a linked SNF that links the timing SNFs of other projects.

The Timing Analyzer generates three types of analyses:

- The *Delay Matrix* shows the shortest and longest propagation delay paths between multiple source and destination nodes in a project.
- The *Setup/Hold Matrix* shows the minimum required setup and hold times from input pins to the data, Clock, Clock Enable, Latch Enable, address, and Write Enable inputs to flipflops, latches, and asynchronous RAM.
- The *Registered Performance Display* shows the results of a registered performance analysis, including a user-defined number performance-limiting delays, minimum Clock period, and maximum circuit frequency.

With the Timing Analyzer, you can tag multiple source and destination nodes so that they are included in an analysis. You can tag these nodes directly in design files in the Graphic, Text, and Waveform Editors, and in the project floorplan with the Floorplan Editor. You can also use the default timing tagging available for each type of analysis. Moreover, you can specify maximum and minimum delays, and completely cut off a signal path from the timing analysis so that only the signal that leads to the node is included in the analysis. You can also exclude I/O pin feedback, Clear, and Preset signal delay paths from the analysis, and restrict the number of paths to list per Clock in a Registered Performance analysis.

After the Timing Analyzer completes an analysis, you can select a source or destination node and list all delay paths associated with it. The Message Processor automatically opens and lists the paths for the selected node, so that you can locate a specific path in the original design file or in the project floorplan.

You can save the results of a timing analysis to a Timing Analyzer Output File (**.tao**).

Go to MAX+PLUS II Help for complete information on all MAX+PLUS II Timing Analyzer functions and features.

Device Programming

Altera provides all hardware and software necessary for programming, configuring, and verifying Altera devices. The hardware includes an add-on Logic Programmer card (for 486- or Pentium-based PCs) that drives the Altera Master Programming Unit (MPU). The MPU performs continuity checking to ensure adequate electrical contact between the programming adapter and the device. With the appropriate programming adapter, the MPU also supports functional testing, so that you can apply vectors created for simulation to a programmed device to verify its functionality.

Altera also supports in-system programmability (ISP) and in-circuit reconfiguration (ICR) with the FLEX Download Cable (for PCs), the ByteBlaster parallel download cable (for PCs), and the BitBlaster serial download cable (for PCs and UNIX workstations). The FLEX Download Cable can connect any Configuration EPROM programming adapter, which is installed on the MPU, to target FLEX devices in a prototype system. The ByteBlaster connects a parallel port (i.e., printer port), and the BitBlaster serial download cable connects a standard RS-232 port (called a "COM port" on a PC), to devices mounted on system boards. These cables allow you to program or configure one or more ICR- or ISP-compatible devices in a FLEX chain or a JTAG chain. See Table 2-4.

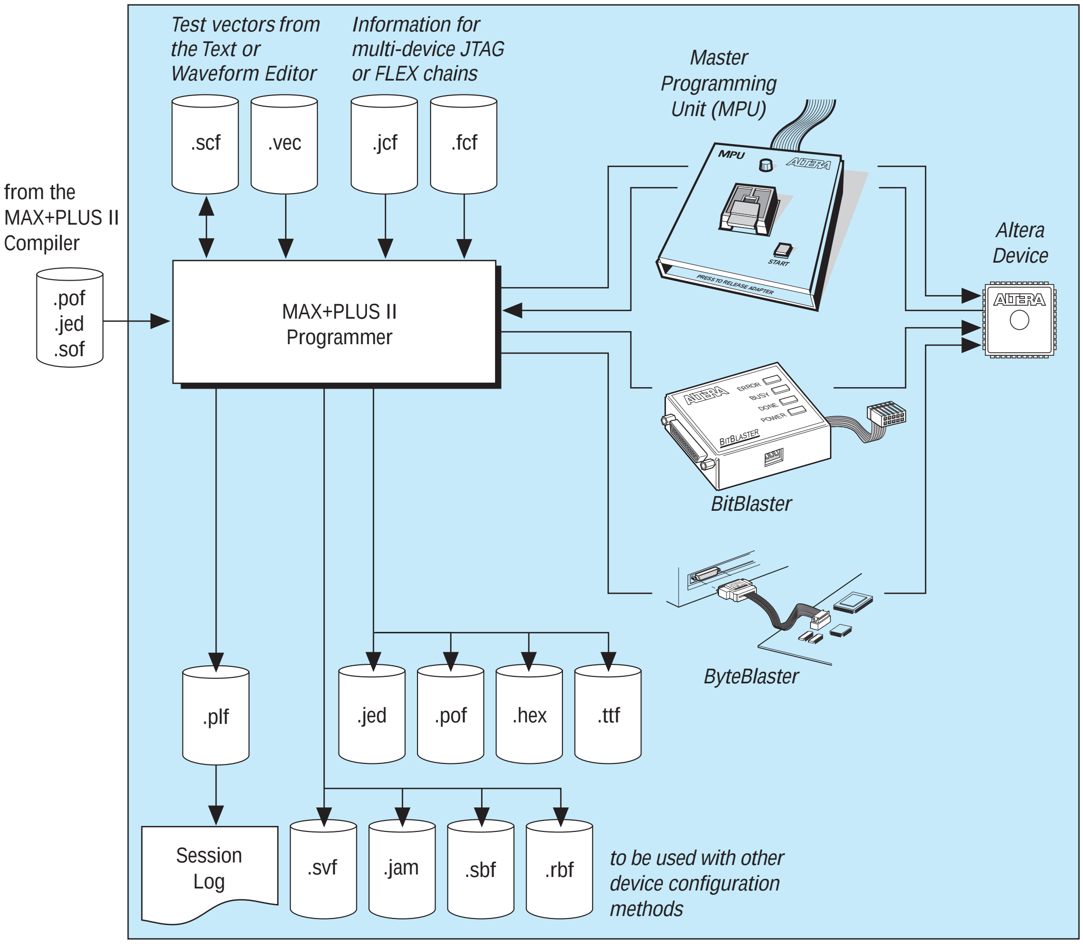
Table 2-4. Altera Programming Hardware

Programming Hardware Option	**PCs**	**UNIX Workstations**	**Classic & MAX 5000 Devices**	**MAX 7000 Devices (excluding MAX 7000S)**	**MAX 7000S & MAX 9000 Devices**	**FLEX 6000, FLEX 8000 & FLEX 10K Devices**	**In-System Programming/ Configuration**
Logic Programmer card, PL-MPU Master Programming Unit, and device-specific adapters	✓		✓	✓	✓		
FLEX Download Cable	✓					✓	✓
BitBlaster Download Cable	✓	✓			✓	✓	✓
ByteBlaster Download Cable	✓				✓	✓	✓

Go to "Installing the Programming Hardware" on page 53 for more information about Altera programming hardware.

Go to *Application Note 87 (Configuring FLEX 6000 Devices)*, *Application Note 33 (Configuring FLEX 8000 Devices)*, *Application Note 38 (Configuring Multiple FLEX 8000 Devices)*, and *Application Note 59 (Configuring FLEX 10K Devices)*, for instructions on how to configure SRAM-based FLEX 6000, FLEX 8000, and FLEX 10K devices.

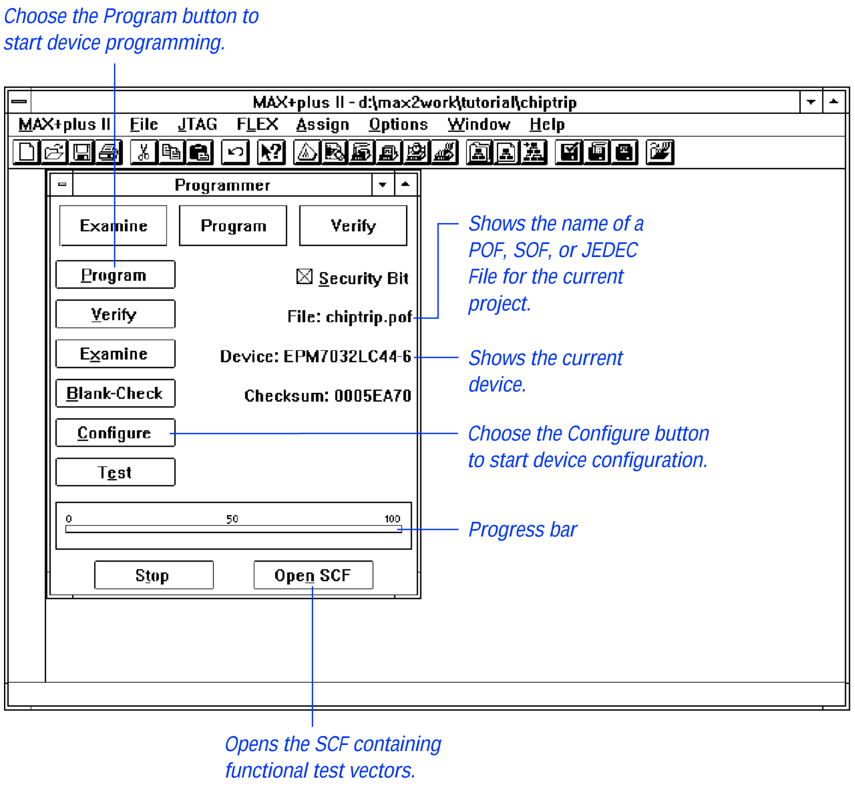
Figure 2-25. MAX+PLUS II Device Programming



MAX+PLUS II Programmer

The MAX+PLUS II Programmer, shown in Figure 2-26, uses programming files generated by the Compiler to program Altera devices. It allows you to program, configure, verify, examine, blank-check, and functionally test devices.

Figure 2-26. MAX+PLUS II Programmer


Choose the Program button to start device programming.
MAX+plus II - d:\max2work\tutorial\chiptrip
MAX+plus II
File
JTAG
FLEX
Assign
Options
Window
Help
Programmer
Examine
Program
Verify
Program
Verify
Examine
Blank-Check
Configure
Test
0
50
100
Stop
Open SCF
Shows the name of a POF, SOF, or JEDEC File for the current project.
Shows the current device.
Choose the Configure button to start device configuration.
Progress bar
Opens the SCF containing functional test vectors.

☒ Security Bit

File: chiptrip.pof

Device: EPM7032LC44-6

Checksum: 0005EA70

Altera programming hardware enables you to program Altera devices with the programming file(s) generated by the Compiler. When you open the Programmer window, the programming file for the current project is automatically loaded. The MAX+PLUS II Programmer accepts the following file formats:

- A Programmer Object File (**.pof**) to program Altera Classic, MAX 5000, MAX 7000, and MAX 9000 devices, as well as the Configuration EPROMs used to configure FLEX 6000, FLEX 8000, and FLEX 10K devices.

- An SRAM Object File (**.sof**) to configure Altera FLEX 6000, FLEX 8000, and FLEX 10K devices.

- A JEDEC File (**.jed**) to program Altera Classic devices and the EPM5016 and EPM5032 devices from the MAX 5000 family.

- A JTAG Chain File (**.jcf**) that describes the order in which POFs, SOFs, and JEDEC Files for multiple devices on a circuit board are to be programmed or configured in a chain of multiple devices that are connected by JTAG circuitry.

- A FLEX Chain File (**.fcf**) that describes the order in which SOFs for multiple FLEX devices on a circuit board are to be configured in a FLEX chain.

The Programmer and Compiler can also generate Hexadecimal (Intel-Format) Files (**.hex**), Tabular Text Files (**.ttf**), Serial Bitstream Files (**.sbf**), Raw Binary Files (**.rbf**), Serial Vector Format Files (**.svf**), and Jam Files (**.jam**) for configuring and programming ISP- and ICR-compatible devices in other programming environments.

You start device programming or configuration by choosing the **Program** or **Configure** button. If any errors or problems are detected during programming or configuration, the Message Processor window lists messages and provides immediate help on how to correct an error.

The MAX+PLUS II Programmer can perform the following tasks:

- Programs POF or JEDEC File data into a blank Classic, MAX 5000, MAX 7000, or MAX 9000 device to produce a working device. In multi-device JTAG chain mode, multiple devices are programmed with additional information from a JCF.

- Downloads configuration data from an SRAM Object File (**.sof**) or a JEDEC File (**.jed**) to configure FLEX 6000, FLEX 8000, or FLEX 10K devices. In multi-device JTAG chains, multiple devices are configured with additional information from a JCF. In multi-device FLEX chains, devices are configured with additional information from an FCF.

- Creates and reads JCFs and FCFs that detail the order of devices that will be programmed or configured in a multi-device JTAG or FLEX chain, respectively.

- Converts a POF into JEDEC File format or vice versa, and optionally saves functional testing vectors in the file, so that you can program and test a device with other industry-standard programming hardware and software.

- Verifies the programming file contents against the contents of a programmed device.

- Examines a programmed device and saves the programming data and test vector data in POF or JEDEC File format.

- Checks to ensure that an erasable device is blank or completely erased.

- Turns the Security Bit on or off before project data is programmed into a device. When the Security Bit is on, the device cannot be interrogated. EPROM-based devices also cannot be reprogrammed.

- Tests a programmed device with the input vectors in the current programming file, SCF, or Vector File, and reports whether the output logic levels in the file are functionally equivalent to actual device outputs.

- Optionally creates an output Programmer Log File (**.plf**) that records programming session commands and messages for future reference.

Go to MAX+PLUS II Help for complete information on all Programmer functions and features.