

## Introduction

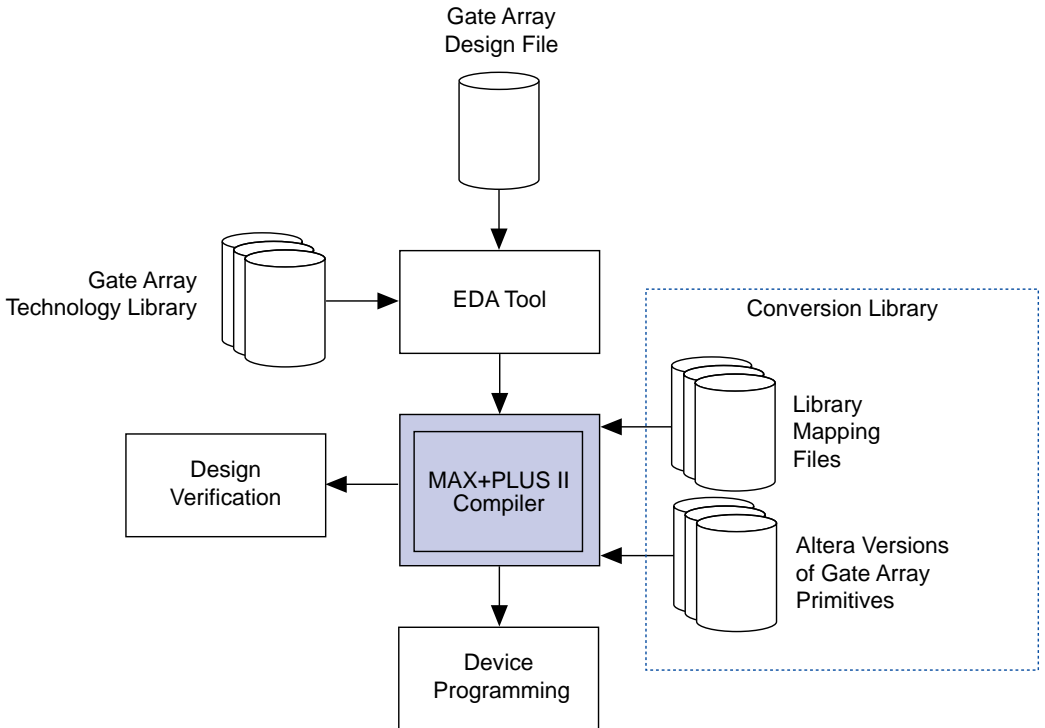
Gate arrays have historically been used for high-volume designs. However, Altera's programmable logic devices (PLDs) are an ideal alternative for prototyping gate array designs and for high-volume production. PLDs offer a high-capacity, high-speed, cost-competitive solution for prototyping and production, and provide designers with the following benefits:

- *Time-to-market advantage*—PLD designs offer quick and easy modification during design debugging. After reprogramming a device, the new design can be checked immediately. Prototyping with PLDs allows several design iterations per day. In contrast, a design iteration for gate arrays can take several weeks or months.
- *Cost*—Altera PLDs are cost-competitive with gate arrays when hidden costs are considered. See the *Gate Array vs. Programmable Logic Cost Analysis White Paper* for a detailed cost comparison.
- *Elimination of dedicated inventory*—Off-the-shelf PLDs eliminate the risk of over- or under-stocking devices because they can be used for any design. In contrast, high inventory risks are inherent in purchasing application-specific devices such as gate arrays.
- *Elimination of test-vector generation*—Altera PLDs are fully tested during manufacturing and provide guaranteed timing and functionality. In contrast, gate arrays require you to create test vectors to detect manufacturing defects. It is difficult to achieve high fault coverage during gate array testing.
- *In-system development*—Hardware engineers can use PLDs to quickly build a prototype system. Software engineers can use the prototype to debug their code rather than waiting for the gate arrays to be fabricated. Prototyping with PLDs allows you to quickly evaluate whether certain features should be implemented in hardware or software. Additionally, in-system programmability (ISP) and in-circuit reconfigurability (ICR) allow you to quickly test different variations of your design.
- *Design entry with gate array primitives*—With the Altera Gate Array-to-Programmable Logic Conversion Kit, engineers can convert architecture-specific designs created with gate array library primitives to designs for Altera PLDs.

Altera's MAX+PLUS II development software greatly simplifies the process of converting and prototyping gate array designs for PLDs. Gate array designs are typically created with design tools supplied by third-party EDA vendors. MAX+PLUS II provides interfaces to popular EDA tools via EDIF 2 0 0 and 3 0 0, and also provides a schematic editor, hardware description language (HDL) synthesis, and timing simulation. MAX+PLUS II supports standard file formats, including EDIF 2 0 0 and 3 0 0, VHDL, VHDL Initiative Toward ASIC Libraries (VITAL), Verilog HDL, and Standard Delay Format (SDF), all of which provide seamless integration with EDA tools. Additionally, MAX+PLUS II directly compiles VHDL and Verilog HDL designs, and creates EDIF, VHDL, and Verilog HDL netlist files for timing simulation. The MAX+PLUS II software is available for 486- and Pentium-based PCs, as well as Sun SPARCstation, HP 9000 Series 700, and IBM RISC System/6000 workstations.

This application note describes how to convert and prototype gate array designs with Altera PLDs. Figure 1 shows the design flow for this conversion process.

Figure 1. Gate Array-to-PLD Design Flow



## Design Flow

The following is a brief description of the gate array to PLD design flow. Each step of the design flow is discussed in greater detail later in this application note.

### Design Entry

You can directly compile and synthesize VHDL and Verilog HDL designs with MAX+PLUS II. You can also convert HDL or schematic designs to EDIF format and import them into MAX+PLUS II.

### Design Processing

You can process gate array designs with EDA tools after design entry. You can also compile designs with the MAX+PLUS II Compiler, which uses architecture-specific logic synthesis, minimization, and fitting algorithms to ensure efficient silicon utilization and performance. Design constraints—such as logic resources and timing—can be passed from the design source files to the MAX+PLUS II Compiler. MAX+PLUS II automatically performs place-and-route, and can partition the design into multiple devices. You can directly control the design implementation with the MAX+PLUS II Floorplan Editor.

The MAX+PLUS II Compiler uses Library Mapping Files (.lbf) to map proprietary symbol and pin names from industry-standard EDA tools to MAX+PLUS II macrofunctions and basic gate library elements. LMFs allow you to convert your logic designs for PLDs without having to restructure architecture-specific design files.

### Design Verification

The MAX+PLUS II Simulator allows you to perform full timing simulation, and the Timing Analyzer allows you to examine registered performance, propagation delay, and setup and hold times for the design. MAX+PLUS II can also create EDIF, Verilog HDL, and VHDL netlist files that contain complete timing information for device- and system-level verification with industry-standard simulation tools. These netlist files enable efficient in-system verification.

## Device Programming

MAX+PLUS II creates files for device programming and configuration. All Altera EPROM-, EEPROM-, and FLASH-based PLDs, as well as Configuration EPROMs for SRAM-based devices can be programmed with standard programming hardware. The EEPROM-based MAX 7000S and MAX 9000 devices also offer ISP, and the SRAM-based FLEX devices offer ICR for on-the-fly design changes. FLASH-based FLASHlogic devices offer both ISP and ICR.

## Design Entry

MAX+PLUS II can support your design flow whether you prefer creating gate array designs with schematic capture or HDL design entry.

- You can use one of the schematic capture tools summarized in Table 1 below to generate an EDIF netlist file. Altera provides design interface kits for major EDA tools so you do not have to recreate your design with an Altera-specific primitive library. You can also prototype a gate array design that has been captured with gate array primitives.
- You have several options if you create a design with an HDL:
  - Create an EDIF netlist file from the synthesized netlist file that targets the gate array technology library.
  - Using Altera synthesis libraries, resynthesize the HDL design with the same tool that you used with the gate array design.
  - Compile a VHDL or Verilog HDL design directly with MAX+PLUS II.
- You can combine schematic and HDL files into a hierarchical design in MAX+PLUS II. For example, a design with multiple state machines can be defined in multiple Verilog HDL design files, while the top-level structure can be defined with a schematic design file.

<b>Table 1. EDA Support for Gate Array Designs</b>			
<b>Vendor</b>	<b>Schematic Capture</b>	<b>Synthesis</b>	<b>Simulation</b>
Cadence	Composer Concept	Synergy	Verilog-XL Leapfrog RapidSIM
Mentor Graphics	Design Architect	AutoLogic AutoLogic II	QuickSim II QuickVHDL
Synopsys		Design Analyzer Design Compiler DesignWare	VSS

## Design Processing

### Schematic Design Guidelines

To compile a schematic design in MAX+PLUS II, you must first create an EDIF netlist file with the schematic capture tool. When you enter the design, use primitives from the gate array technology library of your choice. Choose the appropriate MAX+PLUS II library for the technology library that the design uses. The Gate Array-to-Programmable Logic Conversion Kit contains these MAX+PLUS II libraries. Configure MAX+PLUS II, compile the design, and then generate EDIF, VHDL, or Verilog HDL netlist files for simulation.



See the *Gate Array-to-Programmable Logic Conversion Kit User Guide* for specific information on technology libraries.

The method of creating files from a schematic varies depending on the tool you use. See “[References](#)” on page 12 of this application note for a list of Altera-supplied documents on configuring gate array designs to Altera devices.

### HDL Design Guidelines

You can easily retarget a behavioral HDL design to Altera devices and instantiate primitives specific to a particular gate array vendor (e.g., output drivers). Although MAX+PLUS II cannot directly interpret these primitives, they can be mapped to MAX+PLUS II macrofunctions. The mapping method varies depending on how the design is processed.

#### *EDIF Netlist Files*

You can create an EDIF netlist file from an HDL design file that uses a gate array technology library; this design process does not require you to resynthesize the design. The Altera Gate Array-to-Programmable Logic Conversion Kit allows you to map all gate array primitives to the equivalent Altera macrofunctions. You can also create conversion libraries for unsupported gate array technologies.

#### *HDL Design Resynthesis*

You can resynthesize the HDL design with the same EDA tool using Altera synthesis libraries instead of gate array synthesis libraries. When using this method, the Altera Gate Array-to-Programmable Logic Conversion Kit allows you to map instantiations of gate array primitives. Standard EDA synthesis tools—such as AutoLogic, Exemplar Logic, Synergy, Design Compiler, and FPGA Compiler—support this method, which provides better speed and area results than using the gate array technology library. For example, the Synopsys synthesis libraries include libraries for DesignWare, giving optimal results for adders and counters.

If you resynthesize a design with Altera synthesis libraries, MAX+PLUS II can use both the LMF that maps the gate array primitives and the LMF that is designed for use with the Altera synthesis libraries. Using LMFs together with the libraries in the Altera Gate Array-to-Programmable Logic Conversion Kit is an efficient method of mapping gate array primitives that are instantiated in the HDL design.

### *VHDL Design*

You can open a VHDL design directly in MAX+PLUS II. This method gives better speed and area results than using an EDIF netlist file. This approach works best with purely behavioral VHDL because it does not require you to map technology-specific cells. Search for “MAX+PLUS II VHDL Support” in Help for information on which VHDL constructs are supported by MAX+PLUS II.

If gate array primitives are referenced in the VHDL design, you will need to use Altera macrofunctions that represent the gate array primitives. LMFs do not work with VHDL files.

Without an LMF, a distinct macrofunction is required for every gate array primitive. It is impossible to share macrofunctions between primitives because each macrofunction must have the same name and port names as the gate array primitive.

### *Verilog HDL*

You can use Altera’s Verilog HDL synthesis tool to read a Verilog HDL design in MAX+PLUS II. This method gives better speed and area results than using the output from gate array synthesis. However, minor syntax changes can be required. This approach works best with purely behavioral Verilog HDL because it does not require you to map technology-specific cells.

Altera’s Verilog HDL package generates an Altera Hardware Description Language (AHDL) Text Design File (.tdf) file containing references to any gate array primitives that are in the Verilog HDL file. To compile the design, you need to use MAX+PLUS II macrofunctions that represent the gate array primitives; these macrofunctions are supplied by Altera or created by the designer. MAX+PLUS II does not require special settings to use Verilog HDL. Because LMFs will not work with AHDL TDFs, a distinct macrofunction is required for every gate array primitive. It is not possible for primitives to share macrofunctions because each macrofunction must have the same name and port names as the gate array primitive.

For VHDL and Verilog HDL designs, you can enter timing constraints in the original design if the design is synthesized with Synopsys design tools.



Go to the *Synopsys & MAX+PLUS II Software Interface Guide* for more information on timing analysis and timing constraints with Synopsys tools. For more information on how to create a custom library, search for “Library Mapping File Format” in MAX+PLUS II Help.

## Design Compilation

After you set up the EDIF Netlist Reader and user libraries, MAX+PLUS II can compile the design. To compile the current design, choose the **Start** button in the MAX+PLUS II Compiler window.

MAX+PLUS II offers many features that support gate array conversion and prototyping, including the following:

- *Logic synthesis*—MAX+PLUS II performs architecture-specific logic synthesis on all designs.
- *Design-rule checking*—MAX+PLUS II provides a design-rule checker called the Design Doctor. Once a design is implemented in a device, the Design Doctor looks for potential problems—such as static hazards, race conditions, and complex logic feeding control signals on flipflops—and suggests possible solutions. Search for “Design Doctor utility” in MAX+PLUS II Help for more information.
- *Timing-driven compilation*—With timing-driven compilation, you can optimize individual speed paths by specifying timing constraints (e.g., required Clock frequency). These constraints can be specified with the following methods.
  - MAX+PLUS II can read timing assignments that have been entered with Synopsys tools, allowing for a complete architecture-independent design.
  - You can use the **Global Project Timing Requirements** or **Timing Requirements** commands (Assign menu) to enter a variety of timing assignments directly in MAX+PLUS II.
- *Multi-device partitioning*—If a design is too large to fit into one device, MAX+PLUS II can partition the design into multiple devices automatically, or according to your specifications. Search for “Partitioning a Project” in MAX+PLUS II Help for more information.

## Design Verification

After design compilation, you can verify the design’s functionality and timing with the MAX+PLUS II Simulator and Timing Analyzer.

## Timing Analyzer

The MAX+PLUS II Timing Analyzer shows the worst-case delays in the design. It operates in three modes:

- *Registered performance*—Shows the maximum speed of a design. If the design has multiple or internally generated Clocks, the Registered Performance analysis shows the maximum speed for each individual Clock.
- *Setup/hold matrix*—Shows the setup and hold times for each pin.
- *Delay matrix*—Shows delays from point to point. This mode can be used to look at pin-to-pin delays, such as combinatorial delays or clock-to-output delays. It can also look at node-to-node delays internal to the design.



For more information on how to use the Timing Analyzer, go to MAX+PLUS II Help.

## MAX+PLUS II Simulator

The MAX+PLUS II Simulator verifies the function of the design and supports full timing simulation. MAX+PLUS II also exports EDIF, Verilog HDL, VHDL, VITAL, and SDF files for simulation in standard EDA simulators. See [Table 2](#).

<b>Table 2. MAX+PLUS II Support for Simulation Tools</b>	
<b>Vendor</b>	<b>Simulation Tool</b>
Cadence	RapidSIM Leapfrog Verilog-XL
Mentor Graphics	QuickSim II QuickVHDL
Synopsys	VSS
Viewlogic	ViewSim PROsim PROvhdl Vantage-VHDL VCS

For more information on MAX+PLUS II support for simulation tools, see [“References” on page 12](#) of this application note.

## Device Programming

Traditionally, designers have used programming hardware to program EPROM- and EEPROM-based PLDs before they are mounted on printed circuit boards (PCBs). MAX+PLUS II automatically creates programming files that support industry-standard programming hardware. For more information on device programming, search for “Programmer” in MAX+PLUS II Help.

Both ICR and ISP can shorten development time by allowing the board layout to be completed earlier in the design cycle; thus, accelerating the time-to-market. ICR and ISP allow you to perform more extensive board-level testing, and allow you to perform remote field upgrades to installed systems. See [Table 3](#). In addition, Altera supports the following methods that allow devices to be programmed or configured after they are mounted on a PCB:

- ICR allows SRAM-based devices to be configured during normal system operation. ICR capability enables quick and easy on-board design iterations during prototyping, which provides a mechanism for exhaustive testing that would otherwise be impossible in software simulation. ICR capability has stimulated the growth of reconfigurable hardware applications because it allow you to implement software algorithms in hardware, and change them on-the-fly. ICR makes reconfigurable hardware products a viable solution for reconfigurable computing and algorithm acceleration.
- ISP allows EEPROM- and FLASH-based devices to be programmed during the system manufacturing flow. ISP capability reduces the cost and complexity of manufacturing, prevents lead damage, and enables in-system debugging.

<b>Feature</b>	<b>ICR</b>	<b>ISP</b>
Reconfigurable element	SRAM	EEPROM, FLASH
On-board device programming	Yes	Yes
Programming speed	Fastest (measured in ms)	Fast (measured in s)
Serial programming	Yes	Yes
Parallel programming	Yes	No
Memory volatility	Volatile	Non-volatile
Number of reconfigurations	Unlimited	< 100
On-the-fly reconfiguration during normal system operation	Yes	No

Table 4 provides information on ICR and ISP availability for Altera devices.

Family	Usable Gates	ICR	ISP
FLEX 10K	10,000 to 100,000	✓	
FLEX 8000	2,500 to 16,000	✓	
MAX 9000	6,000 to 12,000		✓
MAX 7000S	600 to 5,000		✓
FLASHlogic	800 to 3,200	✓	✓

For high-volume designs, Altera's Mask-Programmed Logic Device (MPLD) program offers a low-cost migration path. This fully-automated path guarantees function- and pin-compatibility with the original programmable logic design. Refer to the *MPLD Conversion Information & Order Forms*, available from your local Altera representative.

## MAX+PLUS II Conversion Libraries

MAX+PLUS II uses conversion libraries to interpret the gate array primitives in an EDIF netlist file. Conversion libraries consist of an LMF and a library that includes the Altera versions of gate array primitives. Altera provides LSI Logic, Fujitsu, NEC, and Toshiba conversion libraries. Consult Altera for availability.

If your design contains primitives that are not included in an Altera conversion library, you can create these primitives and add them to the library. Creating or modifying a conversion library involves creating an LMF and Altera versions of the gate array primitives. MAX+PLUS II uses LMFs and Altera functions when compiling a design.

To create a conversion library:

1. Examine each gate array primitive to determine which Altera primitive or macrofunction corresponds to the gate array primitive. If no exact match is available, create a MAX+PLUS II design file to match the functionality of the gate array primitive.
2. Create an LMF that maps the gate array primitives to Altera equivalents or to user-created functions. You can also edit a copy of an Altera-supplied LMF to incorporate additional functions.

## LMF Information

An LMF maps gate array primitives in an EDIF netlist file to Altera macrofunctions and primitives. It lists the port names for each EDIF cell that represents a gate array primitive and relates them to the port names of the appropriate Altera macrofunction or primitive.



Search for “Library Mapping File Format” in MAX+PLUS II Help for more information.

## Creating Altera Equivalents of Gate Array Primitives

MAX+PLUS II includes over 300 macrofunctions and primitives. Most gate array primitives can be directly mapped to one of these built-in macrofunctions or primitives by using an LMF. However, some gate array primitives (e.g., gate array AND\_OR cell) might not have an exact match in MAX+PLUS II. Therefore, you must create a design file for this primitive.

You can create a design file for a gate array primitive in a variety of formats, including Verilog HDL, VHDL, EDIF, MAX+PLUS II Graphic Design File (.gdf), or Text Design File (.tdf) format. For example, the following AHDL file creates an AND\_OR primitive:

```
SUBDESIGN and_or
    (in0, in1, in2, in3    : INPUT;
     out                  : OUTPUT; )
BEGIN
    out = (in0 & in1) # (in2 & in3);
END;
```

The Subdesign Section of the TDF defines the inputs and outputs of the design file. The Logic Section defines the Boolean equations that describe the design. In the equation shown above, the ampersand (&) represents a logical AND, and the pound symbol (#) represents a logical OR. See MAX+PLUS II AHDL Help for more information.

## Conclusion

Altera PLDs are an efficient and cost-effective solution for converting gate array designs. Altera provides many tools for prototyping gate array designs and for high-volume production. You can use MAX+PLUS II together with EDA development tools to create your design. In addition, Altera LMFs provide easy mapping of gate array primitives to Altera MAX+PLUS II functions. The Altera Gate Array-to-Programmable Logic Conversion Kit provides design libraries for quick and easy conversion of gate array designs to programmable logic. Moreover, Altera’s application engineers can answer any questions about PLDs, MAX+PLUS II, or the conversion process.

## References

The list below summarizes Altera-supplied literature that provides information related to converting gate array designs.

- *FLEX 10K Embedded Programmable Logic Family Data Sheet*
- *FLEX 8000 Programmable Logic Device Family Data Sheet*
- *MAX 9000 Programmable Logic Device Family Data Sheet*
- *MAX 7000S Programmable Logic Device Family Data Sheet*
- *MAX 7000 Programmable Logic Device Family Data Sheet*
- *Cadence & MAX+PLUS II Software Interface Guide*
- *Mentor Graphics & MAX+PLUS II Software Interface Guide*
- *Synopsys & MAX+PLUS II Software Interface Guide*
- *Powerview & MAX+PLUS II Software Interface Guide*
- *Application Brief 96 (Generating Library Mapping Files)*
- *Application Brief 106 (Simulating MAX+PLUS II Verilog Output)*
- *MAX+PLUS II AHDL Help*

For more information about ICR and ISP, see the following documents:

- *Product Information Bulletin 19 (ICR & ISP)*
- *Application Brief 141 (In-System Programmability in MAX 9000 Devices)*
- *Application Brief 145 (Designing for In-System Programmability in MAX 7000S Devices)*
- *Application Note 33 (Configuring FLEX 8000 Devices)*
- *Application Note 38 (Configuring Multiple FLEX 8000 Devices)*
- *Application Note 59 (Configuring FLEX 10K Devices)*



2610 Orchard Parkway  
San Jose, CA 95134-2020  
(408) 894-7000  
Applications Hotline:  
(800) 800-EPLD  
Customer Marketing:  
(408) 894-7104  
Literature Services:  
(408) 894-7144

Altera, MAX, MAX+PLUS, and FLEX are registered trademarks of Altera Corporation. The following are trademarks of Altera Corporation: MAX+PLUS II, AHDL, FLEX 10K, FLEX 8000, MAX 9000, MAX 7000S, MAX 7000, FLASHlogic, MAX 5000, and Classic. Altera acknowledges the trademarks of other organizations for their respective products or services mentioned in this document, specifically: Fujitsu is a trademark of Fujitsu. LSI Logic is a trademark of LSI Logic, Inc. NEC is a trademark of NEC Electronics Inc. Mentor Graphics is a registered trademark of Mentor Graphics Corporation. Pentium is a trademark of International Business Machines Corporation. SPARCstation is a trademark of SPARC International, Inc. and is licensed exclusively to Syn Microsystems, Inc. Sun is a trademark of Sun Microsystems, Inc. Synopsys is a registered trademark of Synopsys, Inc. Toshiba is a trademark of Toshiba Corporation. Verilog and Verilog-XL are registered trademarks of Cadence Design Systems, Inc. Powerview and Viewlogic are registered trademarks of Viewlogic Systems, Inc. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

U.S. and foreign patents pending.

Copyright © 1996 Altera Corporation. All rights reserved.



I.S. EN ISO 9001