Design Tools for 100,000 Gate Programmable Logic Devices

March 1996, ver. 1 Product Information Bulletin 22

Introduction

The capacity of programmable logic devices (PLDs) has risen dramatically to meet the need for increasing design complexity. Now that PLDs have reached the 100,000-gate density threshold, designers require a powerful set of development tools for design entry and verification. The tools ASIC designers use for 100,000-gate designs provide a good foundation that can be complemented with tools offered by programmable logic vendors. Together, these tools allow PLD designers to create large designs quickly while optimizing the designs for the special silicon features of PLDs.

This product information bulletin discusses the following topics:

- Design entry
- Design synthesis
- Design fitting
- Design verification
- Gate array prototyping strategies

Design Entry

Designing for 100,000-gate devices requires the use of high-level design descriptions developed with hardware description languages (HDLs) such as Verilog HDL and VHDL. To maintain silicon efficiency, ASIC designers use powerful synthesis tools to process their HDL designs. PLD designers can benefit from the same strategy.

Altera's MAX+PLUS II development system supports Verilog HDL and VHDL designs from EDA synthesis tool vendors—including Synopsys, Mentor Graphics, Viewlogic, and Cadence—through an industry-standard EDIF interface. MAX+PLUS II can also directly compile Verilog HDL and VHDL files.

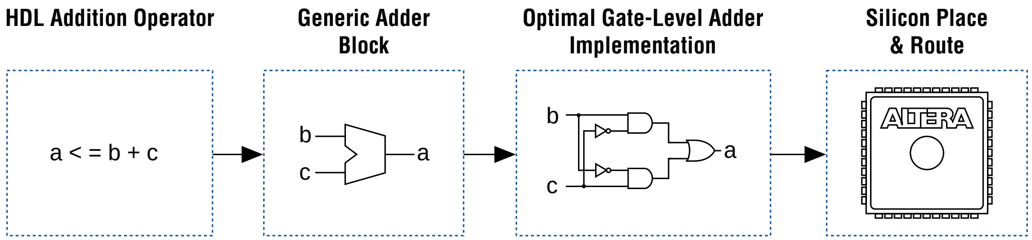
HDLs for Programmable Logic

Verilog HDL- and VHDL-based synthesis tools face the challenge of maintaining the same high efficiency level for PLDs as they have for ASICs. ASICs, like PLDs, have generic building blocks for implementing logic. However, PLDs contain special features (e.g., fast ripple-carry circuitry) for building functions such as adders, comparators, and counters. Synthesis tools support these special features by permitting functions to be implemented through inference and instantiation.

Inference

Functions can be implemented implicitly through inference, i.e., they are automatically recognized from behavioral descriptions. For example, a synthesis tool recognizes a + operator in an HDL design as an adder, and recognizes a Case Statement as a multiplexer. Then, the synthesis tool maps the functions to a gate-level structure that is optimized for the target silicon. Inference allows a designer to use a behavioral design style, which allows the designer to more quickly implement and debug the design. See Figure 1.

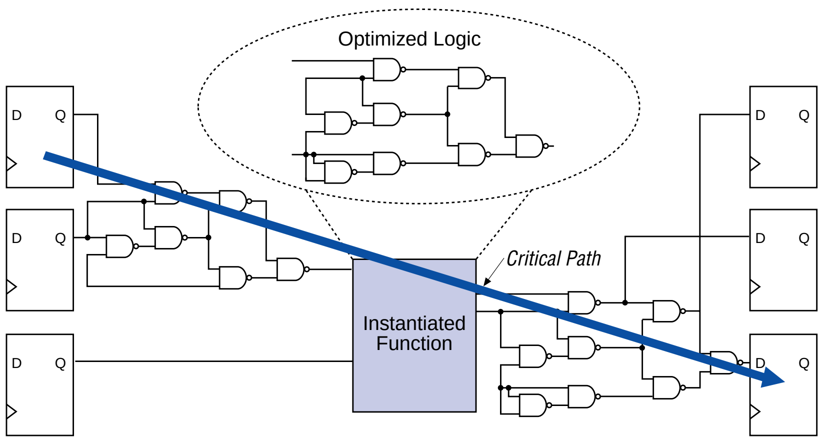
Figure 1. Inference Process



Instantiation

Some functions, such as microcontrollers, microprocessors, and RAM, can be described behaviorally but cannot be automatically inferred and mapped to an optimal gate-level structure. If these functions are area- or speed-critical portions of a design, they should be instantiated for best results (see Figure 2). Many of these functions are available from the library of parameterized modules (LPM) or as megafunctions.

Figure 2. Instance of a Function



Library of Parameterized Modules

The industry-standard LPM contains an efficient set of building blocks for inference and instantiation and is supported by a wide range of silicon vendors. The LPM standard enables efficient mapping of digital designs into divergent architectures, such as PLDs, gate arrays, and standard cells. The LPM standard was accepted as an Electronic Industries Association (EIA) interim standard in April 1993 as an adjunct standard to the Electronic Design Interface Format (EDIF).

Over the past several years, silicon and EDA tool vendors have used the LPM as a standard to provide an architecture-independent design library. The LPM is composed of a compact set of basic building blocks that are parameterized to provide scalability. For example, the parameters of a counter function allow the use of any number of bits, or any combination of loading, direction, and enabling controls. The LPM has several advantages when compared with proprietary libraries supplied by individual vendors. See Table 1.

Table 1. Comparison of Industry-Standard vs. Proprietary Libraries

	High Efficiency	High Level of Abstraction	Architecture Independence	EDA Tool Independence	Compact Set of Functions
Proprietary ASIC vendor library	✓				
7400 TTL library	✓		✓	✓	
Proprietary FPGA vendor library	✓	✓			✓
Industry-standard LPM	✓	✓	✓	✓	✓

Megafunctions

Some designs require large and complex building blocks that go beyond the capability of the LPM. For example, high-density gate arrays are often designed using system-level megafunctions such as RISC or digital signal processing (DSP) cores, offered either by the silicon vendor or by third-party vendors. Cores allow the designer to use complex functions without requiring the time and expertise to build them. High-density PLD designs benefit from the same model.

As part of the Altera Megafunction Partners Program (AMPP), Altera works with third-party, intellectual-property vendors who provide megafunctions. AMPP partners provide a wide variety of functions that are optimized for Altera devices, including microprocessors, ethernet interfaces, and peripheral component interconnect (PCI) interfaces. Altera also provides optimized megafunctions.

Go to Altera's world-wide web site at **http://www.altera.com** for a current listing of AMPP partners and their specialties.

Design Synthesis

In addition to mapping behavioral and structural designs to gates, synthesis tools can map random logic to special architectural features. Synthesis tools can take advantage of special silicon features, for functions such as wide fan-in logic, by mapping the design's logic to these features.

For example, the FLEX 10K architecture contains large blocks of memory that can be configured as ROM, allowing designers to implement many complex combinatorial functions more efficiently than random logic gates. The MAX+PLUS II Compiler can identify functions that are more efficient in ROM and convert them from random logic to ROM. For example, if a user describes a 4×4 multiplier behaviorally, the Compiler can automatically implement it in a 256×8 ROM block.

Design Fitting

Fitting the design into a device is the most computationally intensive step in the development process. Therefore, it is critical that the design fitting tool completes fitting quickly without user intervention. For example, a tool that processes a design in five to eight hours limits design iterations to one per day. In contrast, a tool that processes a design in a fraction of an hour allows many design iterations per day, vastly improving the design's time-to-market. The FLEX 10K architecture is specifically designed for fast-fitting algorithms. A typical 50,000-gate design can be compiled in less than 20 minutes, which enables several design iterations per day.

Floorplanning

Floorplanning tools can augment fitting tools by permitting the designer to view and edit the placement of pins and logic. You can also use floorplanning tools to evaluate critical timing paths, logic equations, and interconnect utilization.

The MAX+PLUS II Floorplan Editor provides a drag-and-drop user interface that facilitates quick pin or logic placement and timing changes. The Floorplan Editor can also trace logic back to the design source files.

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

Design Verification

After design compilation, you can verify the design's functionality and timing with simulation and timing analysis tools.

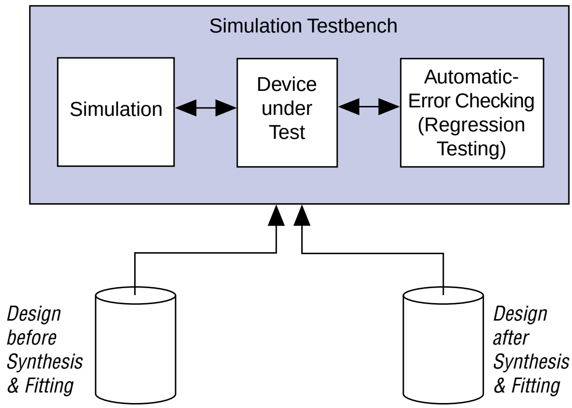
HDL-Based Simulation

The most effective way to simulate 100,000-gate designs is by using Verilog HDL or VHDL. HDL-based simulation offers several advantages:

- Short development cycles
- Reusable simulation routines
- Automatic error checking (regression testing)

After fitting a design, MAX+PLUS II can create Verilog HDL and VHDL simulation netlist files. These netlist files, which include complete timing information, can be used with any industry-standard Verilog HDL or VHDL simulator. Netlist files can be used with the same testbench that verified the design before fitting (see Figure 3), ensuring an exact functional match while also allowing verification of circuit timing.

Figure 3. Simulation Testbench



Timing Analysis

HDL-based simulation is ideal for verifying the logical functionality of a design. For timing verification, however, static timing analyzers offer overwhelming advantages over simulation. Static timing analysis uses exhaustive measurement of all paths in the circuit, determines the maximum design performance with very little processing time, and does not require the user to specify even a single vector.

In contrast, design simulation does not provide comprehensive timing analysis. As designs become more complex, the number of simulation vectors required to measure a similar percentage of timing paths becomes prohibitive. Timing simulation can only determine success or failure at a specific speed; if the design fails, simulation cannot determine the number of failed paths or the margin by which the paths failed.

MAX+PLUS II contains a full-featured static Timing Analyzer that can determine Clock frequency, pin-to-pin propagation delays, setup and hold times, and Clock-to-output times. MAX+PLUS II can also generate Verilog HDL and VHDL netlist files for use with industry-standard static analysis tools.

In-System Verification

With PLDs, changes to a design can be made and quickly verified by reloading the modified design into the device. Devices that support in-system programmability (ISP) and in-circuit reconfigurability (ICR) can be reprogrammed or reconfigured without being removed from the circuit board, which saves time during design changes and prevents device or lead damage.

MAX+PLUS II can download a compiled design through an RS-232 serial port into devices soldered on the circuit board and connected in a Joint Test Action Group (JTAG) chain. Multiple devices can be programmed at once, even if non-Altera devices are in the JTAG chain.

Gate Array Prototyping Strategies

Designers can reduce the risks associated with gate arrays by first prototyping the design with PLDs, using one of the following prototyping design strategies.

- *Behavioral/register transfer level (RTL) design*—A design described with high-level HDL constructs can be easily retargeted using standard synthesis tools. This strategy is easy and safe to use because a single design is used for both prototyping and production. However, the results will not be optimal for either gate arrays or PLDs.

- *PLD-optimized design*—A behavioral design can be optimized for a specific device architecture by structuring the critical paths in the design. Structuring a design for PLDs typically achieves significant gains in speed and efficiency, permitting true in-system verification at the required system speed. The LPM makes structuring simple and effective.

- *Gate array-optimized design*—Hand-optimizing a behavioral design for a gate array returns relatively small improvements in speed and area for the gate array. The hand-optimization usually requires locking the design into a specific silicon vendor library, which makes prototyping difficult or impossible.

- *Dual-optimized design*—A design can be independently optimized for both PLDs and gate arrays. However, the prototype and production designs may not be functionally equivalent.

Table 2 summarizes the prototyping design strategies.

Table 2. Summary of Prototyping Design Strategies

Criteria	Behavioral/ RTL	PLD-Optimized	Gate Array-Optimized	Dual-Optimized
Improved efficiency in gate array over behavioral code	None	None	Minimal	Minimal
Improved efficiency in PLD over behavioral code	None	High	None	High
Risk	Low	Low	Low	High
Ease of design entry	High	High	Medium	Low
Retargetability	High	High	Low	Low
Advantages	Short, reliable design cycle.	Increased ability to prototype a large design at the required system speed.	Slightly improved efficiency in the gate array.	Optimal results can be achieved for both architectures.
Disadvantages	Not optimized for either silicon architecture.	No increase in efficiency for the gate array.	Reduced ability to prototype the design at the required system speed in a single device.	Increased engineering time. Risk of mismatch between prototype and production.

Go to *Application Note 51 (Using Programmable Logic for Gate Array Designs)* for more information on prototyping gate arrays with PLDs.

Conclusion

As PLDs approach 100,000 gates, designers require sophisticated software tools. The tools ASIC designers use for 100,000-gate designs provide a good foundation that can be complemented with powerful tools offered by the PLD silicon vendors. Used together, these tools offer the capabilities required to take advantage of the special silicon features offered by PLDs—even at 100,000-gate densities.

Altera's powerful fitting and floorplanning tools, combined with tight integration to traditional ASIC design entry and verification tools, and support of the industry-standard LPM, provide an ideal platform for developing 100,000-gate PLD designs.

Go to one of the following documents for information on designing for high-density Altera devices using industry-standard EDA tools.

- *Cadence & MAX+PLUS II Software Interface Guide*
- *Mentor Graphics & MAX+PLUS II Software Interface Guide*
- *Synopsys & MAX+PLUS II Software Interface Guide*
- *Viewlogic & MAX+PLUS II Software Interface Guide*

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