

Introduction

MAX 9000 devices extend Altera's third-generation Multiple Array MatriX (MAX) architecture to 12,000 usable gates, and add enhanced features to the MAX device architecture, including in-system programmability (ISP), advanced FastTrack Interconnect, and I/O cells. All MAX 9000 devices are supported by Altera's MAX+PLUS II development system, which uses advanced logic synthesis techniques and a heuristic fitter to map logic designs into MAX 9000 devices. This application note provides tips on how to develop efficient designs that take advantage of the MAX 9000 architecture.

Basic Architectural Features

MAX 9000 devices contain 320 to 560 logic cells, arranged in groups of 16 logic cells called Logic Array Blocks (LABs). Each logic cell has a register and a programmable-AND/fixed-OR array that is used to build combinatorial logic. The following features increase the flexibility and capacity of MAX 9000 devices:

- FastTrack Interconnect—A series of continuous paths that run throughout the length and width of the device, providing signal interconnections between different LABs, and between LABs and pins
- Register control functions—Clock, Clear, and Preset signals that control logic cell flipflops
- Shareable logic expanders—NAND product terms that can be used and shared by logic cells to implement complex logic functions with maximum flexibility
- Parallel expanders—Product terms that can be borrowed by neighboring logic cells to implement complex logic functions with minimal additional timing delays



For more information on the MAX 9000 architecture, refer to the *MAX 9000 Programmable Logic Device Family Data Sheet* in the Altera **1995 Data Book**.

Designing for Speed & Density

Engineers often expect varying levels of performance and density from complex programmable logic devices (CPLDs) and field-programmable gate arrays (FPGAs). In both CPLD and FPGA architectures, tradeoffs are made to optimize designs either for speed or for density. With the MAX 9000 architecture, you can control the speed / density tradeoff to suit the needs of your application with Altera's MAX+PLUS II development system. In addition, you can structure designs to take advantage of the physical characteristics of the MAX 9000 architecture.

You can easily change your design's target device family without manually modifying the design. When you compile a design for MAX 9000 devices, the MAX+PLUS II Compiler automatically synthesizes MAX 9000 designs using macrofunctions optimized for the device family architecture.

Design Entry

MAX+PLUS II provides a wide range of design entry methods. For a fast design cycle, you should describe a design with high-level constructs in a hardware description language (HDL) such as Verilog HDL, VHDL, or the Altera Hardware Description Language (AHDL). Although high-level constructs can simplify the design entry process for complex designs, they can also limit your control over the physical device implementation.

In general, MAX+PLUS II provides excellent area and performance results for MAX 9000 designs that are created with an HDL. However, if a portion of the design must run at a higher performance, you can describe that portion with primitive gates and registers using HDLs or schematics (i.e., create a gate-level design). Although gate-level designs can lengthen the design process, they can also yield higher speeds. MAX+PLUS II supports the library of parameterized modules (LPM), which simplifies gate-level design entry and maintains close control over the target device architecture.

Regardless of your primary design goal, you can select logic options in MAX+PLUS II to guide the logic synthesis for individual logic functions. You can also apply logic synthesis styles, which are combinations of logic option settings saved under a single name. Logic options and logic synthesis styles allow you to optimize a design for any Altera device family.

MAX+PLUS II also offers multi-level synthesis, which provides additional logic options beyond the standard product-term logic synthesis, gives designers control over the fan-in per logic cell, and allows complex projects to be processed without user intervention.



For more information on Altera's LPM support, logic options, and logic synthesis styles, go to MAX+PLUS II Help.

General MAX 9000 Design Guidelines

The following guidelines will help you take advantage of the MAX 9000 architecture to yield maximum speed, reliability, and device resource utilization, while minimizing fitting difficulties:

- Reserve resources in the device for future expansion
- Allow the Compiler to select pin and logic cell assignments
- Use global Clock and Clear signals
- Turn on the XOR Synthesis logic option
- Use parallel expanders only for speed-critical paths
- Reduce logic cell fan-in
- Break down complex logic with LCELL and SOFT buffers
- Evaluate register packing to increase logic cell efficiency
- Use pipelining for complex registered logic

Reserve Resources in the Device for Future Expansion

Altera recommends that you reserve 20% of the logic cells and 20% of the I/O pins in the device to accommodate future design modifications. If the design uses bidirectional pins, you might want to reserve an even larger percentage of the I/O pins, because bidirectional pins require one FastTrack Interconnect routing channel to drive a signal into the device and another FastTrack Interconnect routing channel to receive the signal from the logic core.

Allow the Compiler to Select Pin & Logic Cell Assignments

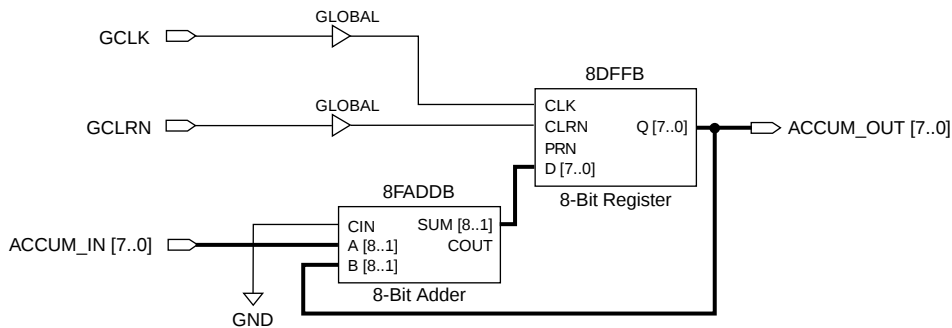
Poorly selected pin and logic cell assignments can limit the MAX+PLUS II Compiler's ability to arrange signals efficiently, which reduces the probability of a successful fit. Therefore, Altera recommends that you permit the MAX+PLUS II Compiler to automatically choose all pin and logic cell assignments. During compilation, MAX+PLUS II efficiently arranges logic cell and I/O pin locations to use the FastTrack Interconnect resources. Back-annotating pins and logic cells lowers the Compiler's flexibility to permute logic cell locations to fit your design. You should also thoroughly simulate designs before back-annotating the MAX+PLUS II Compiler's pin assignments and laying out the printed circuit board.

Use Global Clock & Clear Signals

MAX 9000 devices have two global Clock signals and one global Clear signal that can be used for global register control. These global register control signals can bypass the logic cell array and feed registers directly, preserving product terms for primary logic cell logic. Global signals also bypass logic and data path interconnect resources, since each signal has a dedicated path into the LAB.

Figure 1 shows one method of implementing global control signals by feeding a pin-driven dedicated input signal through a GLOBAL buffer. Without using the GLOBAL signals, the design would require product terms to route the Clock and Clear control signals to the registers, which would further increase logic and expander product term usage.

Figure 1. Saving Resources with Global Controls



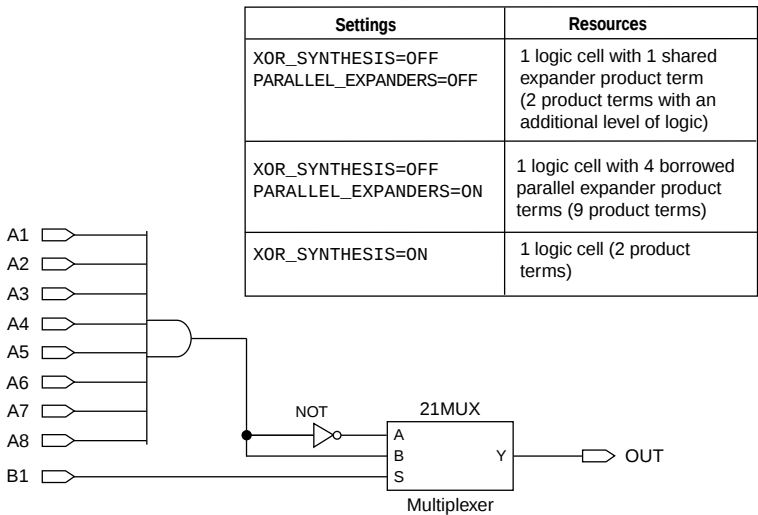
Settings	Resources
With GLOBAL buffers inserted after GCLK and GCLR	Design uses 9 logic cells and 14 shared expanders.
Without GLOBAL buffers	Design uses 13 logic cells and 16 shared expanders.

The MAX+PLUS II Compiler can also automatically choose global signals for the Clock, Clear, and Output Enable controls. You can direct the MAX+PLUS II Compiler to automatically create global signals with the *Automatic Global* options provided in the **Global Project Logic Synthesis** dialog box (Assign menu).

Turn on the XOR Synthesis Logic Option

XOR Synthesis is a MAX+PLUS II logic minimization option that controls the creation of mutually exclusive XOR logic. By default, the MAX+PLUS II Compiler synthesizes mutually exclusive logic with XOR gates. When the XOR Synthesis logic option is turned on, the MAX+PLUS II Compiler minimizes logic by creating new XOR gates that feed combinatorial logic cells, thus decreasing the number of product terms needed by each logic cell. However, XOR gates can create static timing hazards, so you should weigh the tradeoff between logic utilization and XOR gate usage. [Figure 2](#).

Figure 2. XOR Synthesis

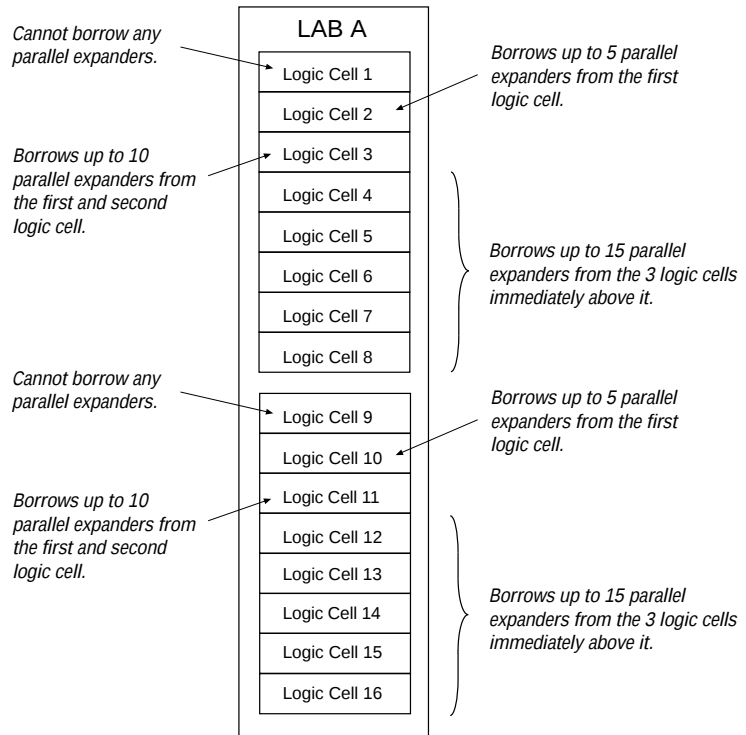


You can incorporate the XOR Synthesis logic option into the default logic synthesis style for an entire project using the **Define Synthesis Style** dialog box that is accessible from the **Global Project Logic Synthesis** dialog box (Assign menu) in any MAX+PLUS II application. You can also apply it on a logic cell-by-logic cell basis with the **Logic Options** command (Assign menu) in any MAX+PLUS II application.

Use Parallel Expanders Only for Speed-Critical Paths

Altera recommends that you use parallel expanders for those parts of your project that require maximum performance. Parallel expanders cause the MAX+PLUS II Compiler to have difficulties in finding and optimizing a fit if pin or logic cell assignments exist for the project. [Figure 3](#) illustrates how parallel expanders are used within an LAB. Where they are not required, the MAX+PLUS II Compiler can implement complex logic by using shared expanders or other logic cells, both of which provide more fitting flexibility than parallel expanders.

The MAX 9000 LABs are composed of two groups of 8 logic cells, forming two chains that can lend or borrow parallel expanders within the same group. The MAX+PLUS II Compiler transparently places logic cells in a location that allow them to lend and borrow parallel expanders appropriately.

Figure 3. Parallel Expanders

You can use the **Define Synthesis Style** dialog box to incorporate the Parallel Expanders logic option into a logic synthesis style. You can then specify this style for the entire project with the **Global Project Logic Synthesis** command, or on a logic cell-by-logic cell basis with the **Logic Options** command. All of these commands are available from the Assign menu in any MAX+PLUS II application.

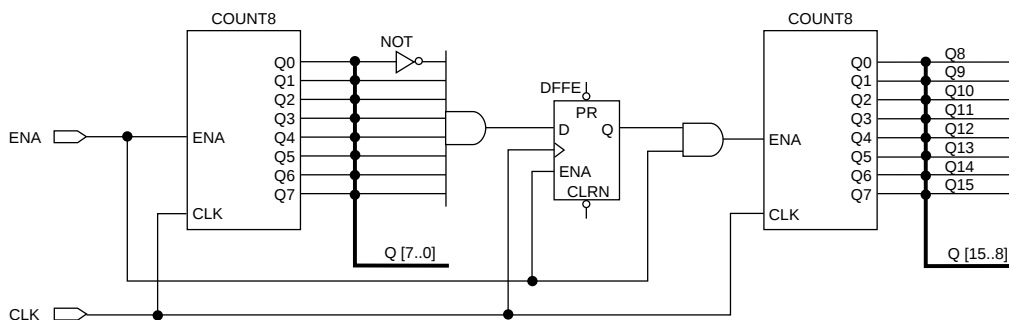
Reduce Logic Cell Fan-In

Logic cells with high fan-in can cause more placement problems for the MAX+PLUS II Compiler than those with low fan-in. When the Compiler places logic cells, it ensures that the fan-in per LAB does not exceed 33. By reducing the fan-in per logic cell, you give the MAX+PLUS II Compiler more flexibility to place logic cells with existing pin assignments. You can control the fan-in for an entire project by choosing the *Multi-Level Synthesis* option in the **Global Project Logic Synthesis** dialog box. The *Multi-Level Synthesis* option allows you to choose the maximum fan-in for all logic cells in a project; by default, the maximum fan-in is 33. Additionally, you can control the fan-in per logic cell with the *Customize* option in the Compiler's **Fitter Settings** dialog box (Processing menu). The *Multi-Level Synthesis* option can help you avoid problems that generate error messages such as:

Logic Array Block<name> requires too many <number>/<number> inputs from row FastTrack Interconnect

Inserting LCELL buffers (or pipelining schemes in sequential designs) can also reduce logic cell fan-in. [Figure 4](#) shows a 16-bit counter design in which a carry look-ahead register reduces the fan-in by seven in the upper eight counter bits. By inserting the look-ahead register before the Enable input to the second counter, you can maintain the performance of your design and make it easier to fit than if the logic were implemented in multiple LABs.

Figure 4. Carry Look-Ahead Counter



Break Down Complex Logic with LCELL & SOFT Buffers

Complex logic, such as multi-level XOR gates, must often be implemented with more than one logic cell. In this case, MAX+PLUS II automatically allocates sharable expanders or additional logic cells (called synthesized logic cells) to supplement the logic resources that are available in a single logic cell. You can also break down complex logic by inserting LCELL and SOFT buffers in the project to reduce the total number of sharable expanders needed. This manual insertion of LCELL or SOFT buffers can provide greater control over speed-critical paths in your design and can also decrease the overall logic cell and sharable expander count.

LCELL buffers provide more control over logic synthesis than SOFT buffers. A good location to insert LCELL buffers is where a single complex logic expression feeds multiple destinations in your design. You can insert an LCELL buffer just after the complex expression; the MAX+PLUS II Compiler extracts this complex expression and places it in a separate logic cell. Rather than duplicating all the logic for each destination, MAX+PLUS II feeds the single output from the logic cell to all destinations.

Alternatively, use SOFT buffers if you anticipate that the logic before or after the buffer will increase in complexity when you make revisions to your design. If the logic that drives the SOFT buffer is complex, then MAX+PLUS II treats the SOFT buffer the same way it would treat an LCELL buffer—it places the logic driving the SOFT buffer into a separate logic cell. However, if the logic that drives the SOFT buffer is simple, MAX+PLUS II ignores the SOFT buffer and combines the logic that drives that SOFT buffer with the logic in the destination logic cells.

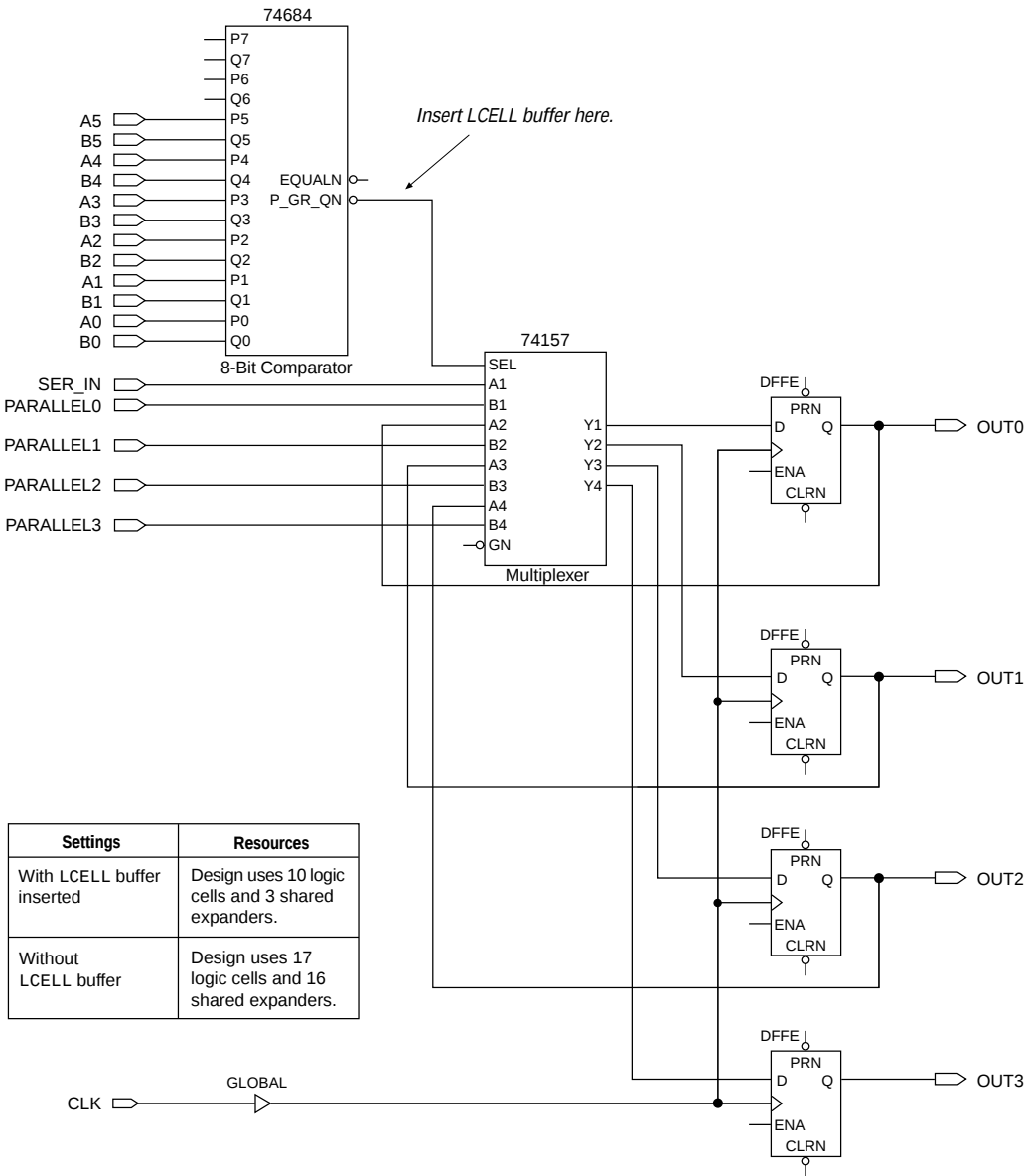
Figure 5 illustrates the insertion of an LCELL buffer after a complex combinatorial expression that feeds multiple destinations (commonly referred to as duplicate logic extraction by MAX+PLUS II). Without the buffer, the design requires 17 logic cells and 16 shared expanders; with the buffer, the design requires only 10 logic cells and 3 shared expanders.

When the *Multi-Level Synthesis* option (available from the **Global Project Logic Synthesis** dialog box) is turned on, the Compiler can identify duplicate logic and can automatically insert LCELL buffers in advantageous locations. The Compiler can also provide advanced synthesis features—such as subfactor extraction and register optimization—to improve the logic cell and shareable expander utilization of your design. When compiling the design in Figure 5 with the *Multi-Level Synthesis* option turned on, MAX+PLUS II automatically inserts an LCELL buffer after the 74684 equality comparator output, which gives the same results as manually inserting an LCELL buffer.



For more information on multi-level synthesis, see MAX+PLUS II Help.

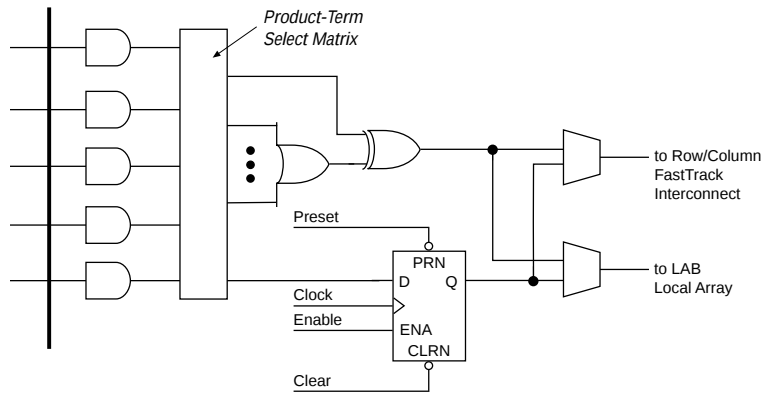
Figure 5. Breaking Down Complex Logic with LCELL Insertion



Evaluate Register Packing to Increase Logic Cell Efficiency

Register packing allows two logic functions—a combinatorial logic function and a register with a single data input—to be implemented in the same logic cell. Combinatorial logic signals passing through the XOR gate can bypass the logic cell's register. The signals are routed either to the rest of the LAB via the local array feedback, or to the rest of the device via the FastTrack Interconnect. A single product term drives the D input of the logic cell's register, and then drives out to the LAB or the FastTrack Interconnect. See [Figure 6](#).

Figure 6. MAX 9000 Register Packing



Register packing can result in improved logic cell utilization, although it consumes more FastTrack Interconnect resources because both outputs of the logic cell are used. Packed logic cells always use the LAB local array feedback and the FastTrack routing resources. However, logic cells that are not packed can use either the local array feedback to reach logic cells within the same LAB, or FastTrack routing to reach logic cells in other LABs and I/O pins. FastTrack routing resources are conserved when a logic cell uses only the LAB local array feedback, making it easier for MAX+PLUS II to fit the design. You should explore the tradeoffs for your design between improved logic cell utilization and tighter fitting constraints for the MAX+PLUS II Compiler.

To select register packing for the entire design, turn on the *Automatic Register Packing* option in the **Global Project Logic Synthesis** dialog box (Assign menu). Once register packing is turned on, MAX+PLUS II packs registers that can use a single product term for their D input.

Use Pipelining for Complex Registered Logic

Maintaining the system Clock speed at or above a certain frequency is often a major goal in a circuit design. For example, if you have a fully synchronous system that must run at 25 MHz, the longest delay path from the output of any register to the input(s) of the register(s) it feeds must be less than 40 ns. Maintaining system Clock speed can be difficult if there are long delay paths through the more complex logic. To reduce delays through complex logic, pipeline these blocks of combinatorial logic by inserting flipflops between them. Although pipelining can consume device resources, it also lowers the propagation delay between registers, allowing you to maintain high system Clock speeds.

The benefits of pipelining can be demonstrated with a 4-bit pipelined adder that adds two 4-bit numbers. The adder is based on two 2-bit adders with outputs that are registered using D flipflops. Figure 7 shows one of the 2-bit pipelined adders. The function 2ADD is the 2-bit adder that feeds both sum bits (SUM1 and SUM2) and a carry bit (COUT) to the D flipflops in 4REG.

Figure 7. 2-Bit Pipelined Adder (2REGADD)

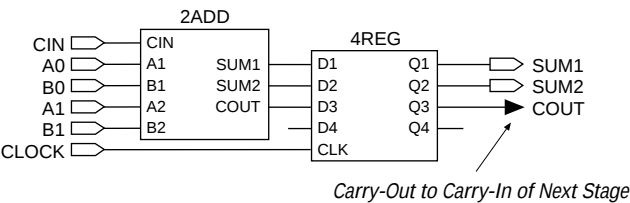
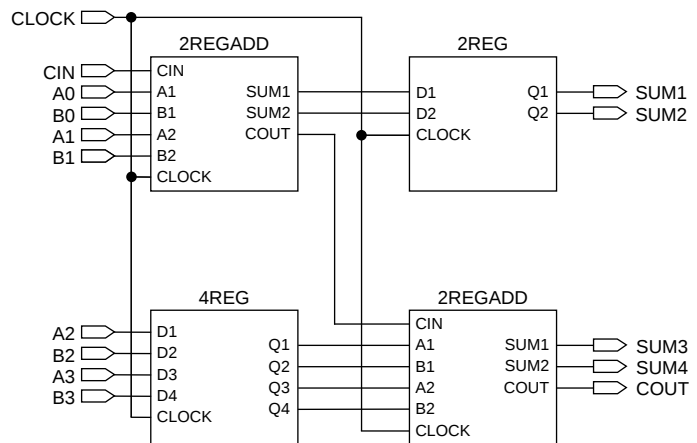


Figure 8 shows two 2-bit adders (2REGADD) that are combined to form a 4-bit pipelined adder. The most significant bits (MSBs) of the 4-bit adder (A3, B3, A2, and B2) require the carry from the least significant bits (LSBs) to complete the sum. Because the time it takes to generate the carry varies, the carry-in and the MSB data inputs arrive at different times. Pipelining this design ensures that the MSBs are presented to the adder's inputs along with the carry-out signal from the previous stage. In Figure 8, both sets of LSBs (A0, B0, A1, and B1) are added in the first Clock cycle, while the other bits (A2, B2, A3, and B3) are added in the following Clock cycle.

Figure 8. Pipelined Adder



Fitting Techniques

Occasionally, a design requires more interconnect resources than the device has available. When a design does not fit, the MAX+PLUS II Compiler issues one or more error messages. In many cases, the MAX+PLUS II Compiler lets you insert LCELL buffers, or change the compilation settings, pin assignments, and logic cell assignments to adjust the fit during compilation.



Go to MAX+PLUS II Help for more information on error messages.

If you have followed the design guidelines in this application note and your project does not fit, here are several techniques to help the MAX+PLUS II Compiler fit your project:

If you are willing to discard your pin assignments, you can let the MAX+PLUS II Compiler automatically ignore all the assignments, the minimum number of assignments, or specific assignments.

If you need to maintain your pin assignments, Altera recommends the following techniques, in the order listed below:

1. Delete logic cell assignments or allow the Compiler to ignore them.
2. Turn on the *Multi-Level Synthesis* option in the **Global Project Logic Synthesis** dialog box. Do not change the default fan-in per logic cell. If the project still does not fit, reduce the maximum fan-in per logic cell in the **Fitter Settings** dialog box. Make sure the *Multi-Level Synthesis* option is turned on.
3. If a row input pin has a high degree of fan-out, reduce the fan-out by inserting LCELL buffers between the pin and some of its destinations.
4. Turn off the *Multi-Level Synthesis* option in the **Global Project Logic Synthesis** dialog box. Select the *Advanced Try Harder/Longer Compilation* fitting option in the **Fitter Settings** dialog box. If your project still does not fit, select the *Advanced Try Harder/Longer Compilation* fitting option with the *Multi-Level Synthesis* option turned on.



Refer to MAX+PLUS II Help for information on entering pin and logic cell assignments, adjusting the fit during compilation, and improving MAX 9000 fitting.

Conclusion

Altera's MAX+PLUS II software allows you to quickly enter new designs or re-target existing designs for MAX 9000 devices. The MAX+PLUS II Compiler automatically optimizes your design for the MAX 9000 architecture. In addition, MAX+PLUS II design entry methods provide control over physical device implementation so you can use your knowledge of the MAX 9000 architecture to achieve the performance and density goals for your design.



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, and FLEX10K. Altera acknowledges the trademarks of other organizations for their respective products or services mentioned in this document, specifically: Verilog and Verilog-XL are registered trademarks of Cadence Design Systems, Inc. Mentor Graphics is a registered trademark of Mentor Graphics Corporation. Synopsys is a registered trademark of Synopsys, Inc. Viewlogic is a registered trademark 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.

Copyright © 1996 Altera Corporation. All rights reserved.



I.S. EN ISO 9001

