Designing with FLEX 8000 Devices

May 1994, ver. 2 Application Note 36

Introduction

Historically, programmable logic devices have fallen into two broad categories: Erasable Programmable Logic Devices (EPLDs) and Field-Programmable Gate Arrays (FPGAs). Widespread use of both EPLDs and FPGAs has revealed the strengths of each type of device. Altera's FLEX 8000 architecture combines the strengths of both EPLDs and FPGAs. This application note describes some of the basic characteristics of the FLEX 8000 architecture and offers several design guidelines that can help you use this architecture effectively.

Application Note 40 has been incorporated into the FLEX 8000 Programmable Logic Device Family Data Sheet.

Regardless of your level of familiarity with programmable logic devices, the information in this application note can help you use FLEX 8000 devices to their fullest potential. For detailed descriptions of the FLEX 8000 architecture and device configuration, refer to *Application Note 40* (*FLEX 8000 Architecture*), *Application Note 33 (Configuring FLEX 8000 Devices)*, and *Application Note 38 (Configuring Multiple FLEX 8000 Devices)* in this handbook.

Designing for Speed & Density

Engineers have learned to expect different levels of performance and density from EPLDs and FPGAs. In both EPLD and FPGA architectures, trade-offs are made to optimize designs either for speed or for density. With the FLEX 8000 architecture, you can control speed/density trade-offs to suit the needs of your application. In addition, you can use Altera's MAX+PLUS II software to automatically optimize all or part of a circuit for speed or density. You can also structure designs to take advantage of the physical characteristics of the FLEX 8000 architecture.

Automatic Design Optimization

Altera's MAX+PLUS II software automatically optimizes a design for the architecture of its target device family. Therefore, you can change the target device family for a design without entering any special design modifications. You can also combine one or more existing EPLD designs and re-target them for one or more FLEX 8000 devices.

When you compile a design for the FLEX 8000 device family, the MAX+PLUS II Compiler automatically uses macrofunctions that are optimized for the FLEX 8000 architecture and performs FLEX 8000 family-specific logic synthesis and optimization. You can use logic synthesis styles and logic options that are tailored to the FLEX 8000 architecture to further control design optimization. The automatic design optimization in

MAX+PLUS II allows you to re-target designs to other device families quickly and with minimal effort.

Designing for FLEX 8000 Architecture

In addition to using automatic design optimization, you can take advantage of specific architectural features in FLEX 8000 devices to create circuits that run at higher speeds or use fewer device resources. MAX+PLUS II design entry methods provide the detailed control necessary to achieve the maximum possible speed and density in a FLEX 8000 device.

Basic Architectural Features

The FLEX 8000 architecture is based on logic elements (LEs) containing 4-input look-up tables (LUTs), registers, and several other features that are especially important in creating logic designs:

- ❏ Register control functions—Clock, Clear, and Preset signals that control a programmable flipflop.
- ❏ FastTrack Interconnect—A series of fast, continuous paths that run the entire length and width of the device and provide signal interconnections between different Logic Array Block (LABs) and between LABs and pins.
- ❏ Carry and cascade chains—High-speed data paths that connect adjacent LEs without using other interconnect resources.

The FLEX 8000 LUT can implement any function of four variables. When you compile a FLEX 8000 design, the MAX+PLUS II Compiler automatically divides functions of more than four variables into multiple 4-input functions.

In contrast, Altera's Classic, MAX 5000/EPS464, and MAX 7000 device families use an `AND-OR` array as the fundamental building block for combinatorial logic, with eight, five, and three product terms per macrocell, respectively. When you compile a design for any of these EPLD families, the Compiler divides functions requiring more than the available number of product terms into multiple macrocells.

Design Entry Methods

Altera provides and supports design entry methods that offer a full spectrum of low- to high-level control over actual design implementation. If your primary goal is a fast design cycle, you can 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 help simplify the design entry process, they can limit your control over the physical device implementation because the synthesized logic depends on the synthesis algorithms of the software that processes the HDL.

If you wish to obtain the maximum performance and density in FLEX 8000 devices, you can describe designs with primitive gates and registers (i.e., a "gate-level" design) using HDLs or schematics. MAX+PLUS II also provides family-specific macrofunctions that have been optimized for the FLEX 8000 architecture. Gate-level designs and designs that use FLEX 8000 family-specific macrofunctions provide the greatest control over the physical implementation in a device. Although designing at the gate level may slow the design process, it typically yields the highest speeds and lowest area costs.

Regardless of the design entry method you choose, you can assign logic options in MAX+PLUS II to guide logic synthesis on individual logic functions. You can also apply logic synthesis styles, which are combinations of logic option settings saved under a single name. These logic options and logic synthesis styles can be set to optimize a design for a particular device family. For example, specifying a setting of "Auto" for the Carry Chain and Cascade Chain logic options instructs the Compiler to automatically implement FLEX 8000 carry and cascade chains, which are useful for optimizing designs for high speed or minimum area.

These logic options are also set to Auto in the Altera-provided Fast logic synthesis style in MAX+PLUS II. In contrast, specifying a setting of "Ignore" for the Carry Chain and Cascade Chain logic options directs the Compiler to ignore any user-specified carry and cascade logic. These logic options are set to Ignore in the Altera-provided Normal logic synthesis style. Since the LEs in a carry or cascade chain must be adjacent to each other, long carry or cascade chains can limit fitting flexibility and may reduce the routability of a design. Therefore, you may wish to use different logic option settings in different portions of your design. For more information on logic options and logic synthesis styles, refer to MAX+PLUS II Help; for more information on carry and cascade chains, refer to *Application Note 40* (*FLEX 8000 Architecture*) in this handbook.

Application Note 40 has been incorporated into the FLEX 8000 Programmable Logic Device Family Data Sheet.

General FLEX 8000 Design Guidelines

The following design guidelines will help you use the FLEX 8000 architecture as efficiently as possible. Following these guidelines will yield maximum speed, reliability, and device resource utilization, and minimize fitting problems.

Reserve Resources in the Device for Future Expansion

The design process generally includes many modification cycles for logic changes or additional logic. Altera recommends that you leave 20% of the device's logic cells and I/O pins unused to accommodate future design modifications.

Allow the Compiler to Select Pin & Logic Cell Assignments

Although you can use FLEX 8000 device resources extremely efficiently, poorly or arbitrarily selected resource assignments can prevent a design from fitting. During compilation, MAX+PLUS II arranges and permutes logic cell and I/O pin locations to use the partially populated multiplexers in the FastTrack Interconnect as efficiently as possible. Pin and/or logic cell assignments, however, can limit the MAX+PLUS II Compiler's ability to arrange signals efficiently, thus reducing the probability of a successful fit. Therefore, Altera recommends that you allow the Compiler to choose all pin and logic cell locations automatically. You should also simulate a design as thoroughly as possible before you lay out your printed circuit board or back-annotate the Compiler's pin assignments.

Balance Ripple-Carry & Carry Look-Ahead Usage

Each FLEX 8000 LE contains high-speed carry and cascade generation logic. The dedicated carry chain in the FLEX 8000 architecture can propagate a ripple-carry for short- and medium-length counters and adders with minimum delay and maximum efficiency. Long carry chains, however, restrict the Compiler's ability to fit a design because the LEs in the chain must be contiguous.

You can design counters using either a ripple-carry or a carry look-ahead. In contrast to ripple-carry counters, logic cells used in carry look-ahead counters can be non-adjacent. When the Compiler processes a carry look-ahead counter, it can arrange and permute the LEs to map the design into the device more efficiently.

Altera does not recommend using ripple-clocked counters, i.e., counters in which the output of one flipflop clocks another flipflop.

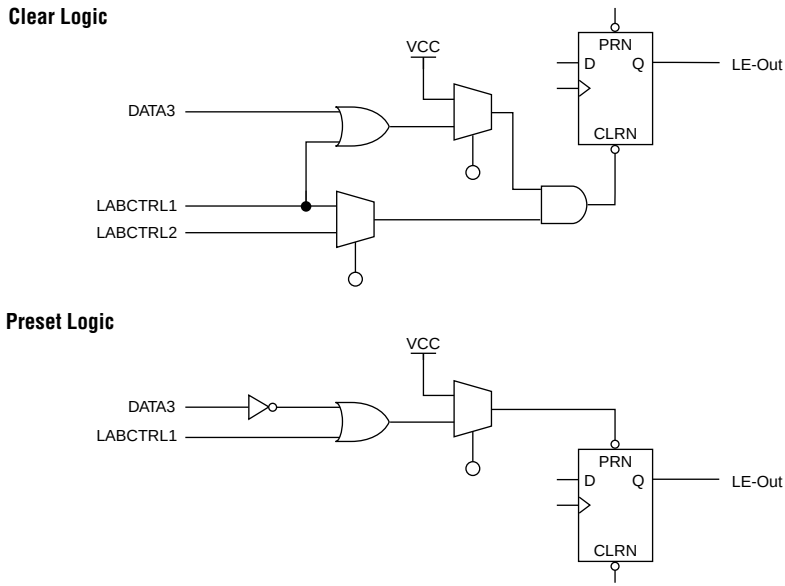
Altera recommends that you use carry chains only in the portions of a design that require maximum performance. You can choose between using ripple-carry and carry look-ahead counters on a case-by-case basis. In some cases, you may wish to trade the speed and silicon efficiency of a ripple-carry implementation for the increased routability and logic cell usage of a carry look-ahead implementation. For more information on counters, refer to *Application Brief 121* (*Designing Counters in FLEX 8000 Devices*) in this handbook.

Use Global Clock & Clear Signals

In FLEX 8000 devices, a programmable flipflop is used to support sequential functions. Sequential logic circuits are most reliable if they are fully synchronous, i.e., if every register in the design is clocked by the same global Clock signal and reset by the same global Clear signal. The FLEX 8000

architecture is optimized for this type of highly reliable, fully synchronous design. Four dedicated high-speed, low-skew global signals are available throughout each device, independent of the FastTrack Interconnect resources. Using these global signals for Clock and Clear functions will ensure a more reliable design and a much more efficient fit. Figure 1 shows the register control signals in FLEX 8000 devices.

Figure 1. Clear & Preset Logic



The Preset and Clear functions of the register can be functions of LABCTRL 1, LABCTRL 2 and DATA 3 This structure is especially useful for sequential functions that require an asynchronous Clear with loading capability.

The asynchronous load (with or without a Clear input signal) and asynchronous Preset modes can be implemented within a single FLEX 8000 LE. Figure 2 shows an asynchronous load with a Clear input signal. Since the Clear signal has priority over the load signal, it does not need to feed the Preset circuitry.

Figure 2. Asynchronous Load with a Clear Input Signal

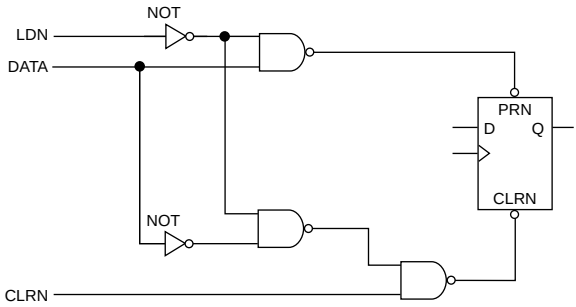


Figure 3 shows an asynchronous load without a Clear input signal.

Figure 3. Asynchronous Load without a Clear Input Signal

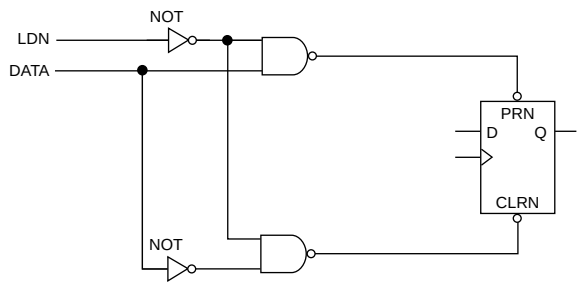
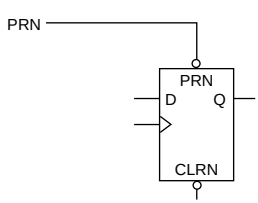


Figure 4 shows an asynchronous Preset signal. Asynchronous Preset signals are actually implemented as asynchronous loads in FLEX 8000 devices. The FLEX 8000 device loads a "1" into the register to implement a Preset. MAX+PLUS II uses the Clear input to the register for simple Preset signals, thus preserving the data input for use in the LUT while providing correct Preset functionality.

Figure 4. Asynchronous Preset



Use One-Hot State Bit Encoding

One-hot state bit encoding increases both the system speed and routability of a design. This type of encoding uses one register per state and allows only one state bit to be active at any time. Although one-hot encoding increases the number of registers, it also reduces the average fan-in to the state bits. This reduced fan-in minimizes the number of LEs required to implement the state decoding logic and yields a design that runs faster and uses less interconnect.

MAX+PLUS II automatically uses one-hot encoding when compiling state machines written in AHDL, VHDL, or Verilog HDL and targeted for FLEX 8000 devices. Altera also recommends using the options provided by other industry-standard CAE tools, such as the Mentor Graphics Autologic and Synopsys Design Compiler tools, to synthesize state machines described in VHDL or Verilog HDL with one-hot state bit encoding. For more information, refer to *Application Brief 131 (State Machine Encoding)* in this handbook.

Use Pipelining for Complex Combinatorial 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 some of the delay paths through the more complex logic are long. In these cases, Altera recommends pipelining complex blocks of combinatorial logic by inserting flipflops between combinatorial logic. Although pipelining may increase device resource usage, it lowers the propagation delay between registers and allows 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. This adder is based on two 2-bit adders that have outputs that are registered using D flipflops. Figure 5 shows one of the 2-bit pipelined adders. The function `2ADD` is the 2-bit adder that feeds both sum bits (`SUM 1`and `SUM 2`) and a carry bit (`COUT`) to the D flipflops in `4REG`.

Figure 5. 2-Bit Pipelined Adder (2REGADD)

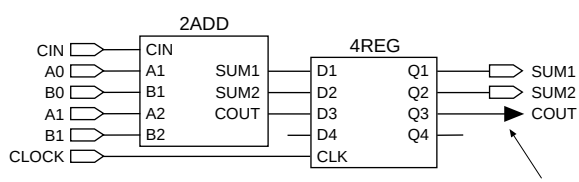
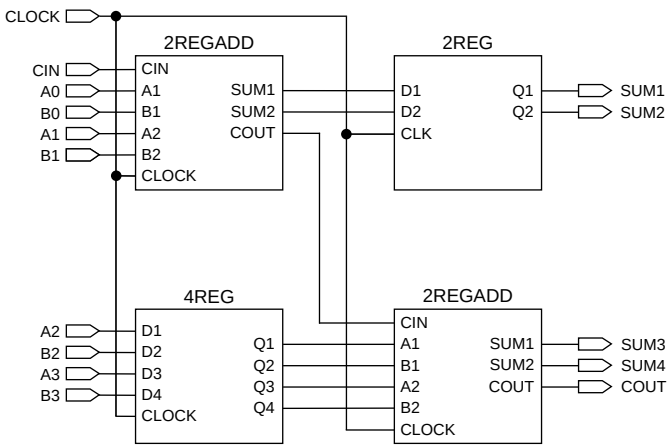


Figure 6 shows two 2-bit adders that are combined to form a 4-bit pipelined adder. The most significant bits (MSBs) of the 4-bit adder (A3, B3, A2, and B2) require the carry from the least significant bits (LSBs) for their sum. However, the MSB data inputs to the adder and the carry-in arrive at different times, due to the time it takes to generate the carry. Pipelining this design ensures that the MSBs are presented to the inputs of the adder at the same time as the carry-out signal from the previous stage. In Figure 6, the two sets of LSBs (A0, B0, A1, and B1) are added on the first Clock cycle, while bits A2, B2, A3, and B3 are added on the next Clock cycle. The outputs of 2REGADD are registered.

Figure 6. 4-Bit Pipelined Adder



Pipelining is most effective with register-intensive devices such as FLEX 8000 devices. While it can be used in product-term-based architectures such as those of the MAX 5000 and MAX 7000 devices, it may be less effective than in the FLEX 8000 architecture. Since each MAX 5000 and MAX 7000 logic cell has higher fan-in than a 4-input LUT, complex functions that require several FLEX 8000 LUTs may need only a single MAX 5000 or MAX 7000 logic cell.

Fitting Techniques

Occasionally, a design may require more interconnect resources than are available in the device. When a design does not fit, the MAX+PLUS II Compiler issues one or more error messages; up-to-date information on these error messages is available from on-line help. In many cases, it allows you to change compilation settings and pin and logic cell assignments or insert logic cells to adjust the fit during the compilation.

If the project does not fit after you have followed all of the design guidelines provided in this application note, you can use several techniques to help the Compiler fit the design:

- If you are willing to discard your pin assignments, you can allow the Compiler to automatically ignore all assignments, the minimum number of assignments possible, or specific individual assignments.

- If you wish to maintain your pin assignments, Altera recommends trying each of the following techniques, in order:

    1. Direct the Compiler to automatically insert logic cells between the design logic and the pins. Inserting logic cells gives the Compiler more fitting flexibility by separating device inputs and outputs from the design logic.

    2. Delete any logic cell assignments or allow the Compiler to ignore them.

    3. Allow the Compiler to ignore explicitly entered carry and cascade chain logic on a case-by-case basis or throughout the design.

    4. Break long carry chains by inserting logic cells into the chain.

    5. Redesign functions with long carry chains (e.g., adders and counters) with techniques such as carry look-ahead.

    6. Place input and bidirectional pins on column interconnects when possible.

    7. If an input pin has a high degree of fan-out, break the fan-out down by inserting `LCELL` primitives between the pin and some of its destinations.

Refer to MAX+PLUS II Help for additional information on entering pin, logic cell, and clique assignments; implementing carry and cascade logic; and adjusting the fit during compilation.

Conclusion

Altera has combined the strengths of both EPLDs and FPGAs into the FLEX 8000 architecture. Altera's MAX+PLUS II software allows you to quickly enter new designs or re-target existing designs for FLEX 8000 devices with design compilation that is automatically optimized for the FLEX 8000 architecture. In addition, MAX+PLUS II design entry methods offer detailed control over physical device implementation so that you can use your knowledge of the FLEX 8000 architecture to achieve the maximum speed and density for your designs.