2. Logic Array Blocks and Adaptive Logic Modules in Stratix III Devices

SIII51002-1.5

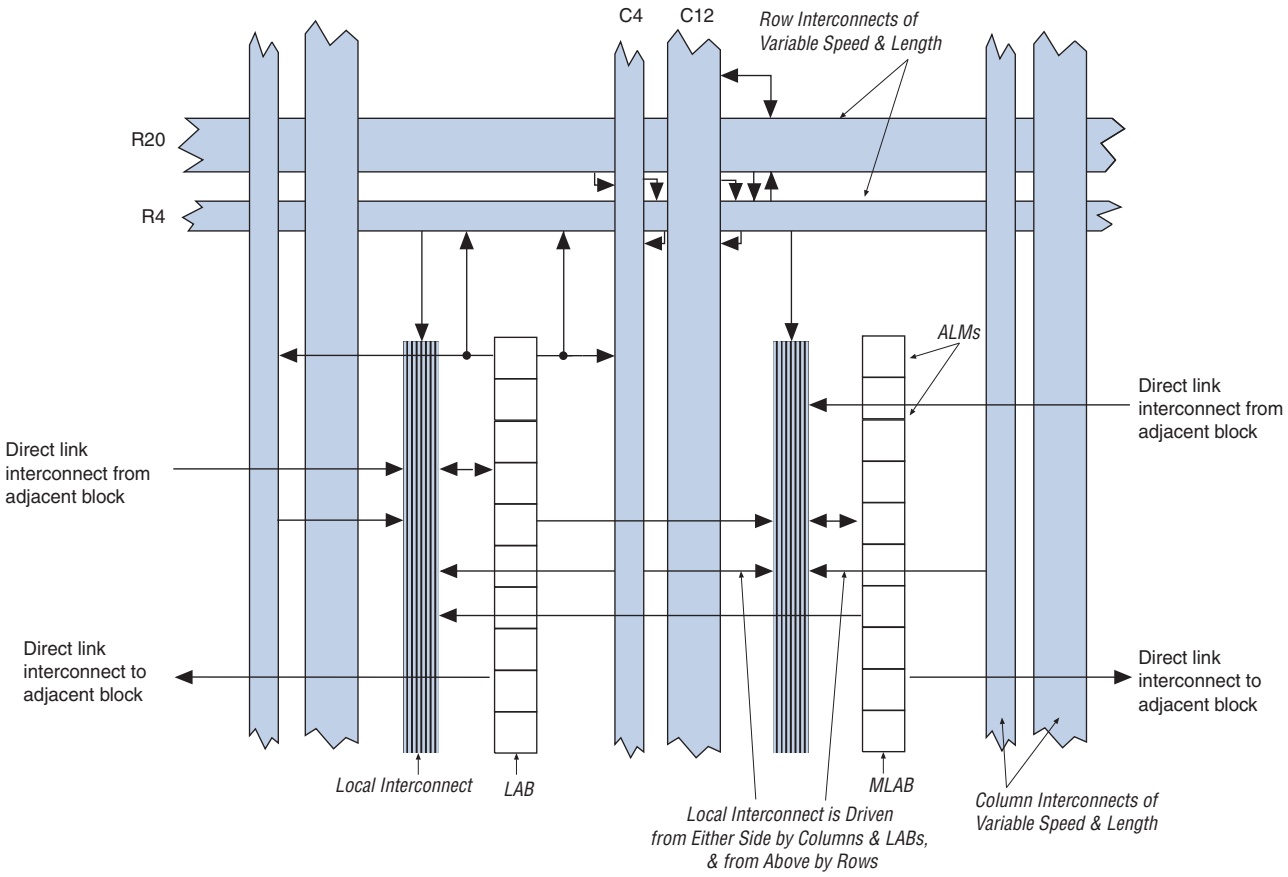
Introduction

This chapter describes the features of the logic array block (LAB) in the Stratix® III core fabric. The logic array block is composed of basic building blocks known as adaptive logic modules (ALMs) that can be configured to implement logic functions, arithmetic functions, and register functions.

Logic Array Blocks

Each LAB consists of ten ALMs, carry chains, shared arithmetic chains, LAB control signals, local interconnect, and register chain connection lines. The local interconnect transfers signals between ALMs in the same LAB. The direct link interconnect allows a LAB to drive into the local interconnect of its left and right neighbors. Register chain connections transfer the output of the ALM register to the adjacent ALM register in an LAB. The Quartus® II Compiler places associated logic in an LAB or adjacent LABs, allowing the use of local, shared arithmetic chain, and register chain connections for performance and area efficiency. Figure 2–1 shows the Stratix III LAB structure and the LAB interconnects.

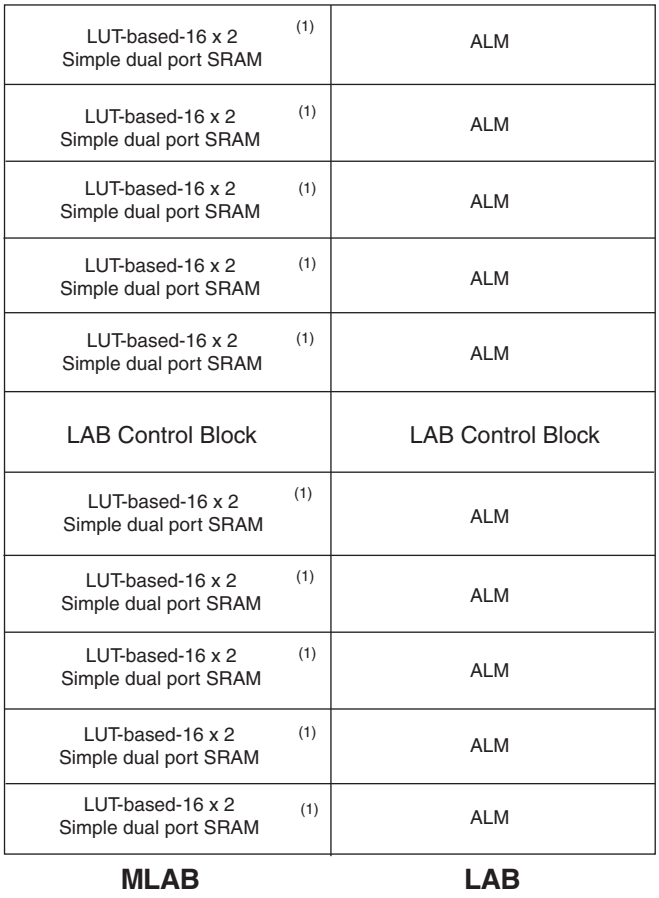
Figure 2–1. Stratix III LAB Structure



The LAB of Stratix III has a new derivative called Memory LAB (MLAB), which adds look-up table (LUT)-based SRAM capability to the LAB as shown in Figure 2–2. The MLAB supports a maximum of 320-bits of simple dual-port static random access memory (SRAM). You can configure each ALM in an MLAB as a 16 × 2 block, resulting in a configuration of 16 × 20 simple dual port SRAM block. MLAB and LAB blocks always co-exist as pairs in all Stratix III families. MLAB is a superset of the LAB and includes all LAB features. Figure 2–2 shows an overview of LAB and MLAB topology.

The MLAB is described in detail in the *TriMatrix Embedded Memory Blocks in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

Figure 2–2. Stratix III LAB and MLAB Structure



Note to Figure 2–2 :

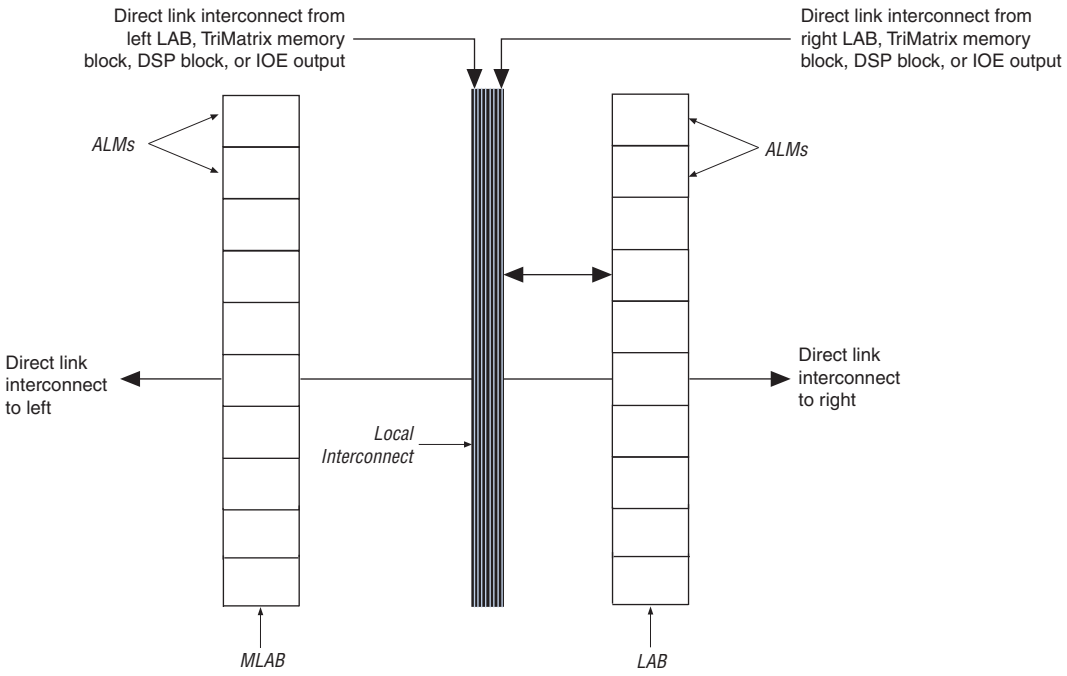
(1) You can use MLAB ALM as a regular LAB ALM or configure it as a dual-port SRAM, as shown.

LAB Interconnects

The LAB local interconnect can drive ALMs in the same LAB. It is driven by column and row interconnects and ALM outputs in the same LAB. Neighboring LABs/MLABs, M9K RAM blocks, M144K blocks, or DSP blocks from the left and right can also drive a LAB's local interconnect through the direct link connection. The direct link connection feature minimizes the use of row and column interconnects, providing higher performance and flexibility. Each ALM can drive 30 ALMs through fast local and direct link interconnects.

Figure 2–3shows the direct link connection.

Figure 2–3. Direct Link Connection



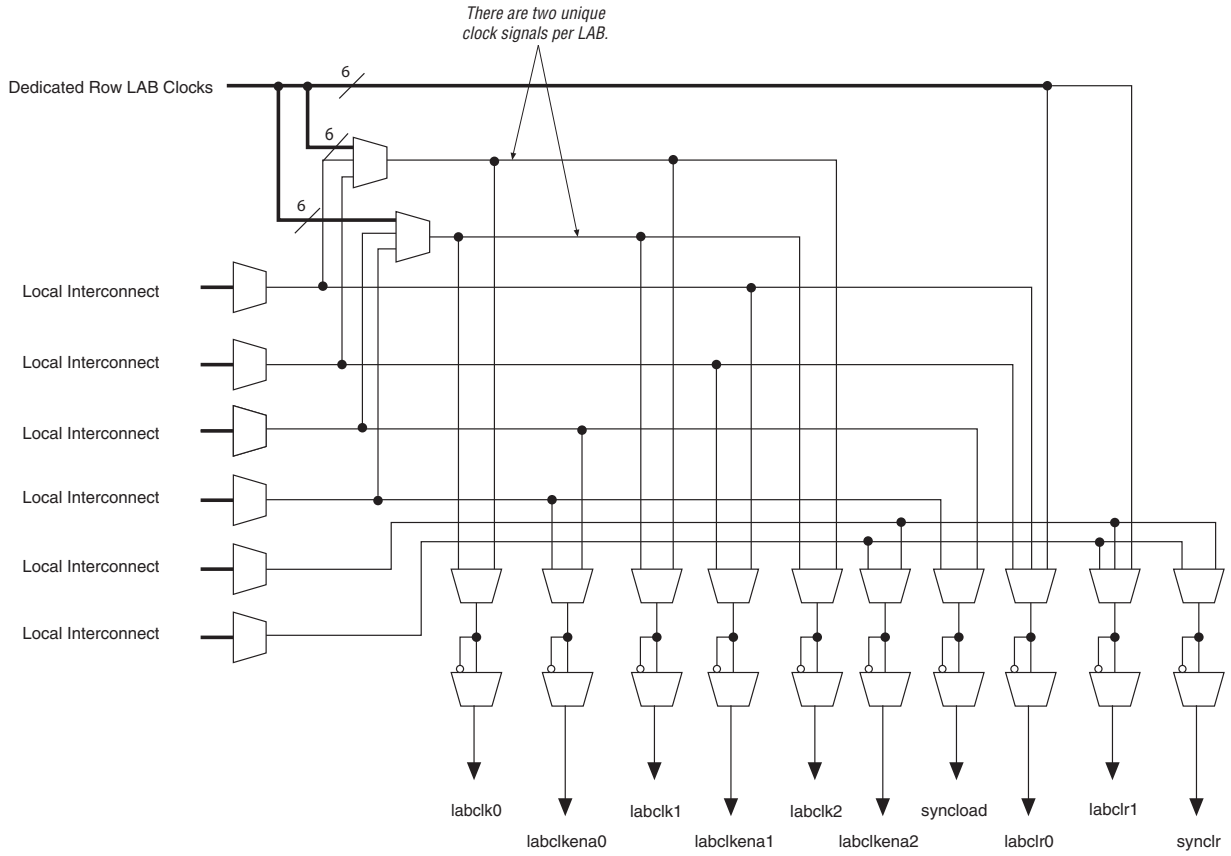
LAB Control Signals

Each LAB contains dedicated logic for driving control signals to its ALMs. The control signals include three clocks, three clock enables, two asynchronous clears, a synchronous clear, and synchronous load control signals. This gives a maximum of 10 control signals at a time. Although you generally use synchronous load and clear signals when implementing counters, you can also use them with other functions.

Each LAB has two unique clock sources and three clock enable signals, as shown in Figure 2–4. The LAB control block can generate up to three clocks using the two clock sources and three clock enable signals. Each LAB's clock and clock enable signals are linked. For example, any ALM in a particular LAB using the `labclk1` signal also uses `labclkena1` signal. If the LAB uses both the rising and falling edges of a clock, it also uses two LAB-wide clock signals. De-asserting the clock enable signal turns off the corresponding LAB-wide clock.

The LAB row clocks [5..0] and LAB local interconnect generate the LAB-wide control signals. The MultiTrack™ interconnect's inherent low skew allows clock and control signal distribution in addition to data. Figure 2–4shows the LAB control signal generation circuit.

Figure 2–4. LAB-Wide Control Signals



Adaptive Logic Modules

The basic building block of logic in the Stratix III architecture, the adaptive logic module (ALM), provides advanced features with efficient logic utilization. Each ALM contains a variety of look-up table (LUT)-based resources that can be divided between two combinational adaptive LUTs (ALUTs) and two registers. With up to eight inputs to the two combinational ALUTs, one ALM can implement various combinations of two functions. This adaptability allows an ALM to be completely backward-compatible with four-input LUT architectures. One ALM can also implement any function of up to six inputs and certain seven-input functions.

In addition to the adaptive LUT-based resources, each ALM contains two programmable registers, two dedicated full adders, a carry chain, a shared arithmetic chain, and a register chain. Through these dedicated resources, an ALM can efficiently implement various arithmetic functions and shift registers. Each ALM drives all types of interconnects: local, row, column, carry chain, shared arithmetic chain, register chain, and direct link interconnects. Figure 2–5 shows a high-level block diagram of the Stratix III ALM while Figure 2–6 shows a detailed view of all the connections in an ALM.

Figure 2–5. High-Level Block Diagram of the Stratix III ALM

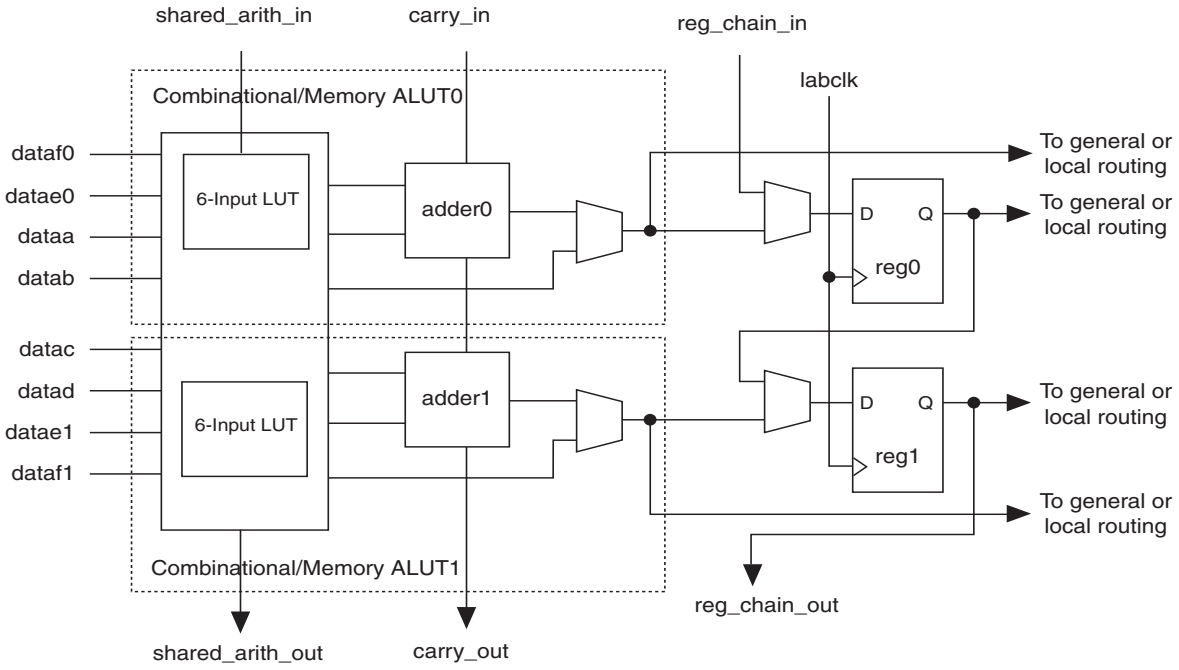
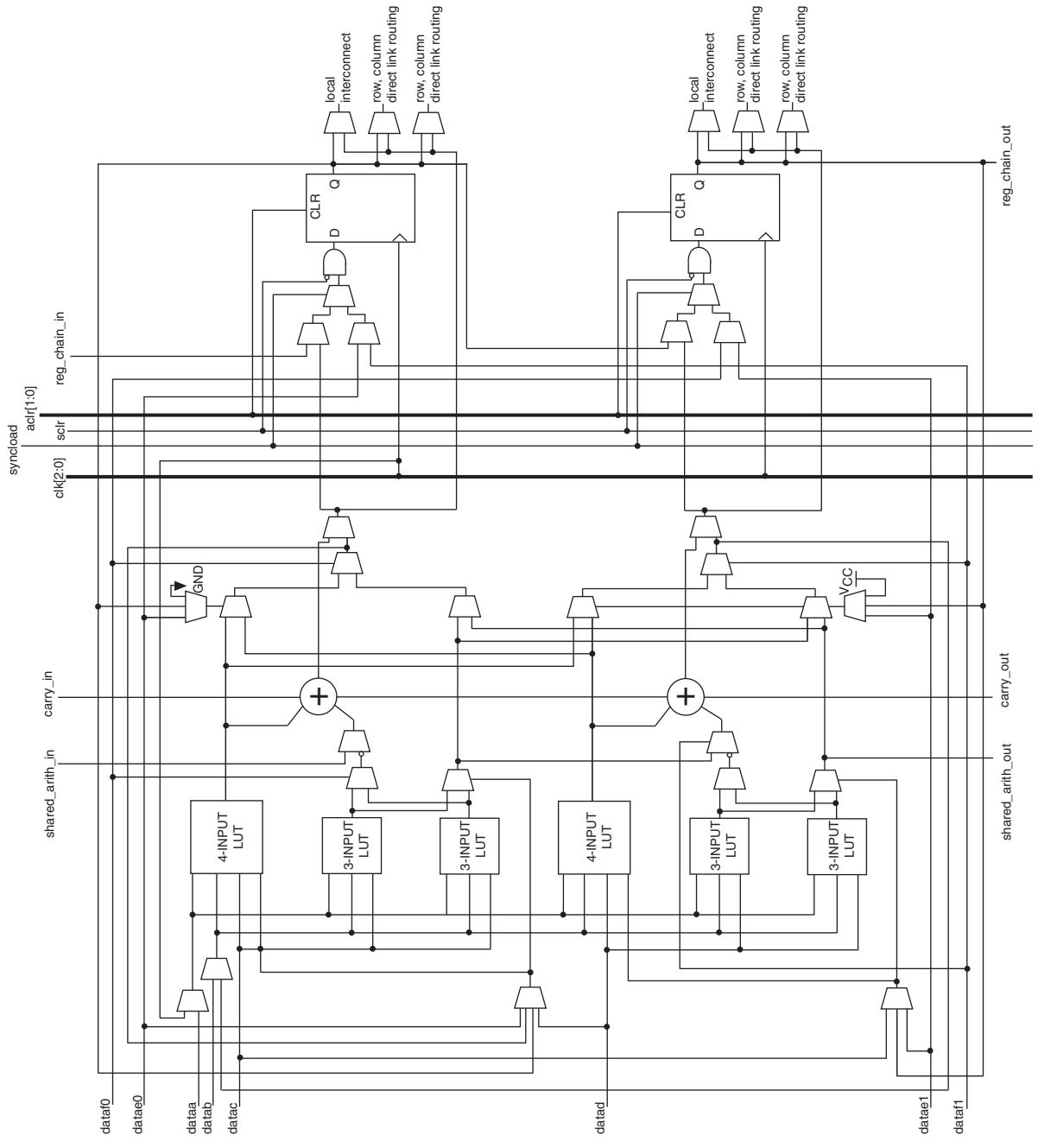


Figure 2–6. Stratix III ALM Details



One ALM contains two programmable registers. Each register has data, clock, clock enable, synchronous and asynchronous clear, and synchronous load/clear inputs. Global signals, general-purpose I/O pins, or any internal logic can drive the register's clock and clear control signals. Either general-purpose I/O pins or internal logic can drive the clock enable. For combinational functions, the register is bypassed and the output of the LUT drives directly to the outputs of an ALM.

Each ALM has two sets of outputs that drive the local, row, and column routing resources. The LUT, adder, or register output can drive these output drivers (refer to Figure 2–6). For each set of output drivers, two ALM outputs can drive column, row, or direct link routing connections, and one of these ALM outputs can also drive local interconnect resources. This allows the LUT or adder to drive one output while the register drives another output.

This feature, called register packing, improves device utilization because the device can use the register and the combinational logic for unrelated functions. Another special packing mode allows the register output to feed back into the LUT of the same ALM so that the register is packed with its own fan-out LUT. This provides another mechanism for improved fitting. The ALM can also drive out registered and unregistered versions of the LUT or adder output.

ALM Operating Modes

The Stratix III ALM can operate in one of the following modes:

- Normal
- Extended LUT Mode
- Arithmetic
- Shared Arithmetic
- LUT-Register

Each mode uses ALM resources differently. In each mode, eleven available inputs to an ALM—the eight data inputs from the LAB local interconnect, carry-in from the previous ALM or LAB, the shared arithmetic chain connection from the previous ALM or LAB, and the register chain connection—are directed to different destinations to implement the desired logic function. LAB-wide signals provide clock, asynchronous clear, synchronous clear, synchronous load, and clock enable control for the register. These LAB-wide signals are available in all ALM modes.

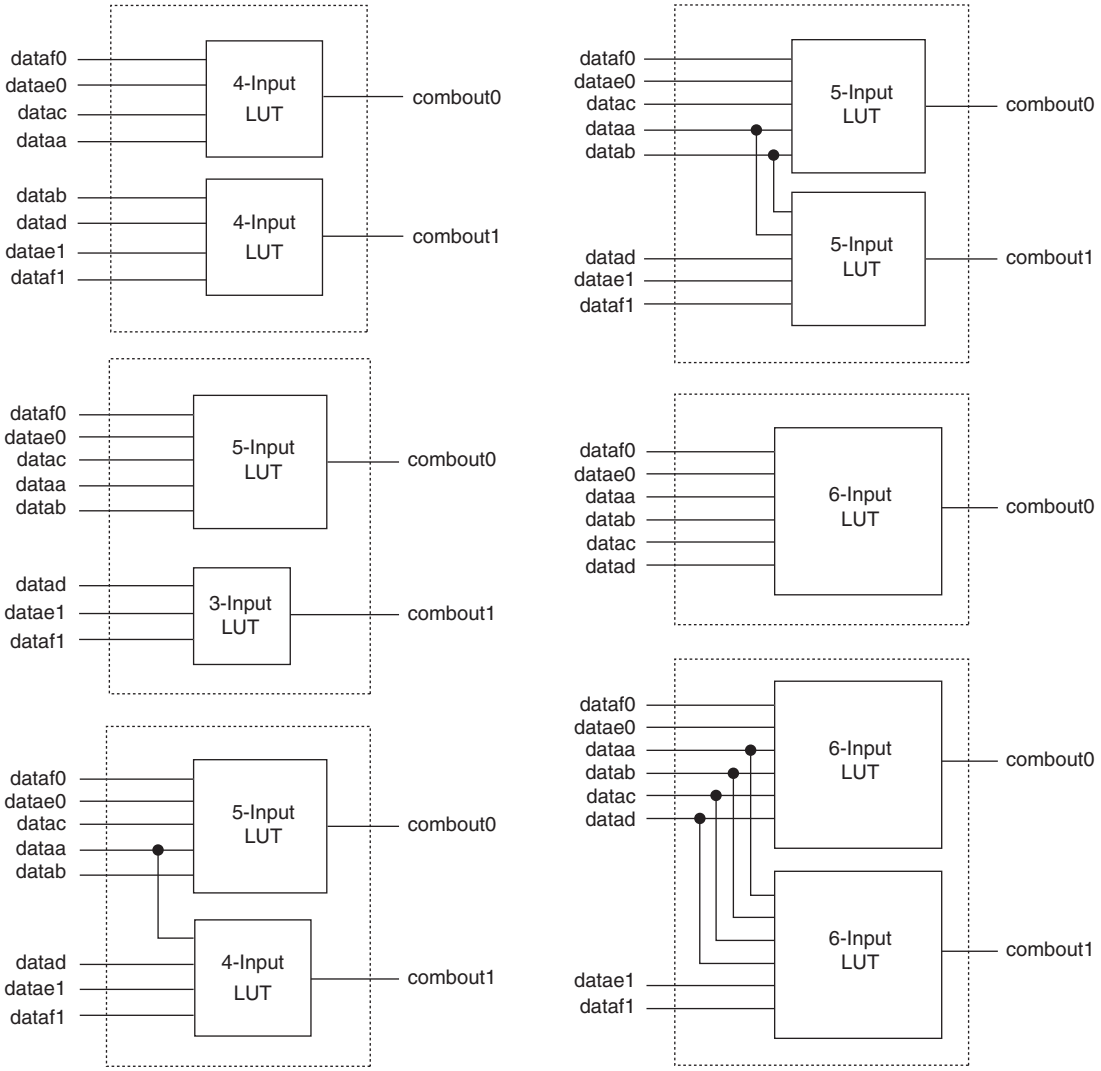
☞ Refer to "LAB Control Signals" on page 2–4 for more information on the LAB-wide control signals.

The Quartus II software and supported third-party synthesis tools, in conjunction with parameterized functions such as the library of parameterized modules (LPM) functions, automatically choose the appropriate mode for common functions such as counters, adders, subtractors, and arithmetic functions.

Normal Mode

The normal mode is suitable for general logic applications and combinational functions. In this mode, up to eight data inputs from the LAB local interconnect are inputs to the combinational logic. The normal mode allows two functions to be implemented in one Stratix III ALM, or an ALM to implement a single function of up to six inputs. The ALM can support certain combinations of completely independent functions and various combinations of functions that have common inputs. Figure 2–7 shows the supported LUT combinations in normal mode.

Figure 2–7. ALM in Normal Mode *(Note 1)*



Note to Figure 2–7:

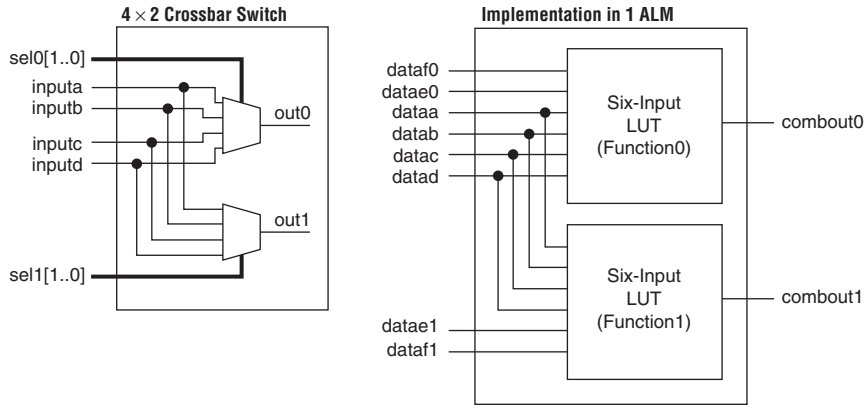
(1) Combinations of functions with fewer inputs than those shown are also supported. For example, combinations of functions with the following number of inputs are supported: 4 and 3, 3 and 3, 3 and 2, 5 and 2.

The normal mode provides complete backward compatibility with four-input LUT architectures.

For the packing of 2 five-input functions into one ALM, the functions must have at least two common inputs. The common inputs are `dataa` and `datab`. The combination of a four-input function with a five-input function requires one common input (either `dataa` or `datab`).

In the case of implementing 2 six-input functions in one ALM, four inputs must be shared and the combinational function must be the same. For example, a 4 × 2 crossbar switch (two 4-to-1 multiplexers with common inputs and unique select lines) can be implemented in one ALM, as shown in Figure 2–8. The shared inputs are `dataa`, `datab`, `datac`, and `datad`, while the unique select lines are `datae0` and `dataf0` for `function0`, and `datae1` and `dataf1` for `function1`. This crossbar switch consumes four LUTs in a four-input LUT-based architecture.

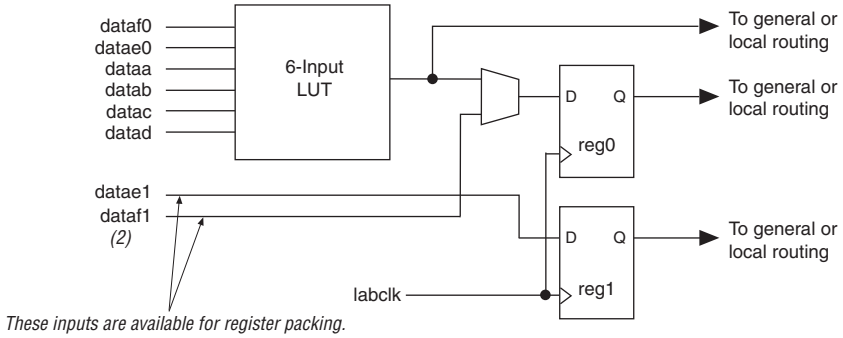
Figure 2–8. 4 × 2 Crossbar Switch Example



In a sparsely used device, functions that could be placed into one ALM may be implemented in separate ALMs by the Quartus II software in order to achieve the best possible performance. As a device begins to fill up, the Quartus II software automatically utilizes the full potential of the Stratix III ALM. The Quartus II Compiler automatically searches for functions of common inputs or completely independent functions to be placed into one ALM and to make efficient use of the device resources. In addition, you can manually control resource usage by setting location assignments.

Any six-input function can be implemented utilizing inputs `dataa`, `datab`, `datac`, `datad`, and either `datae0` and `dataf0` or `datae1` and `dataf1`. If `datae0` and `dataf0` are utilized, the output is driven to `register0`, and/or `register0` is bypassed and the data drives out to the interconnect using the top set of output drivers (refer to Figure 2–9). If `datae1` and `dataf1` are utilized, the output drives to `register1` and/or bypasses `register1` and drives to the interconnect using the bottom set of output drivers. The Quartus II Compiler automatically selects the inputs to the LUT. ALMs in normal mode support register packing.

Figure 2–9. Input Function in Normal Mode *(Note 1)*



Notes to Figure 2–9:

(1) If `datae1` and `dataf1` are used as inputs to the six-input function, then `datae0` and `dataf0` are available for register packing.

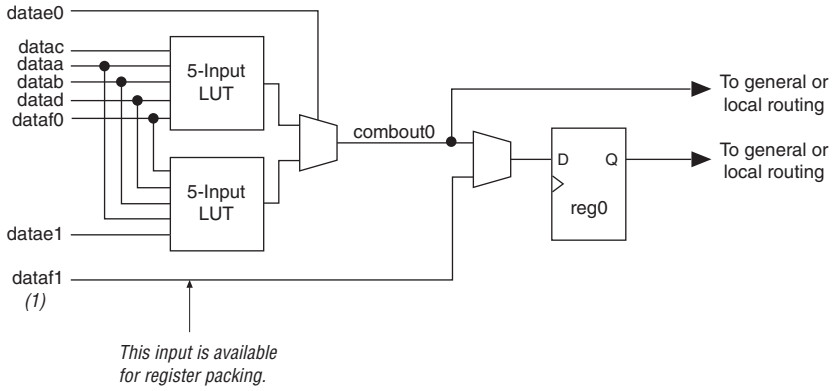
(2) The `dataf1` input is available for register packing only if the six-input function is un-registered.

Extended LUT Mode

Use the extended LUT mode to implement a specific set of seven-input functions. The set must be a 2-to-1 multiplexer fed by two arbitrary five-input functions sharing four inputs. Figure 2–10 shows the template of supported seven-input functions utilizing extended LUT mode. In this mode, if the seven-input function is unregistered, the unused eighth input is available for register packing.

Functions that fit into the template shown in Figure 2–10 occur naturally in designs. These functions often appear in designs as "if-else" statements in Verilog HDL or VHDL code.

Figure 2–10. Template for Supported Seven-Input Functions in Extended LUT Mode



Note to Figure 2–10:

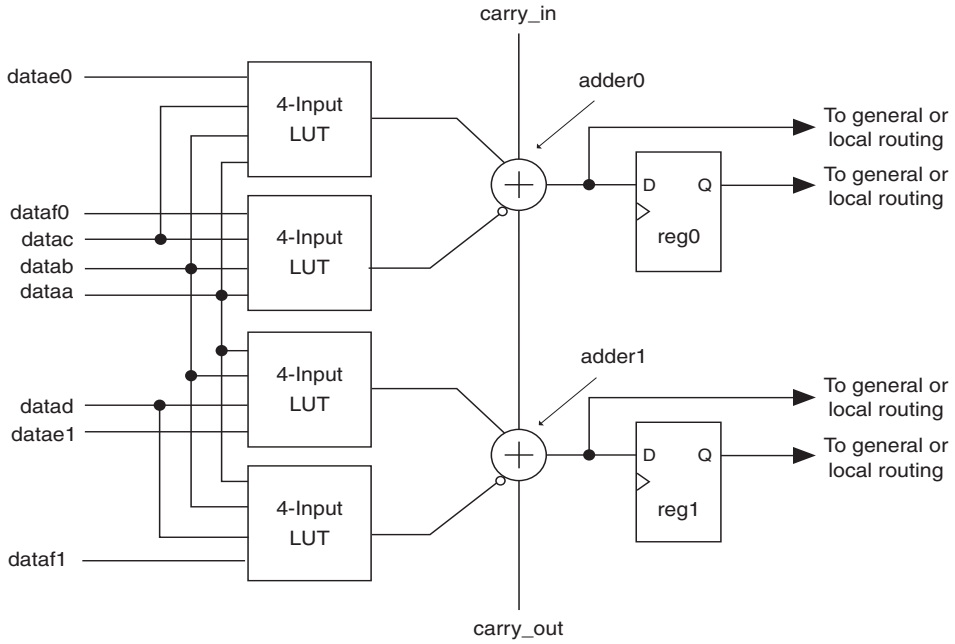
(1) If the seven-input function is unregistered, the unused eighth input is available for register packing. The second register, `reg1`, is not available.

Arithmetic Mode

The arithmetic mode is ideal for implementing adders, counters, accumulators, wide parity functions, and comparators. The ALM in arithmetic mode uses two sets of 2 four-input LUTs along with two dedicated full adders. The dedicated adders allow the LUTs to be available to perform pre-adder logic; therefore, each adder can add the output of 2 four-input functions.

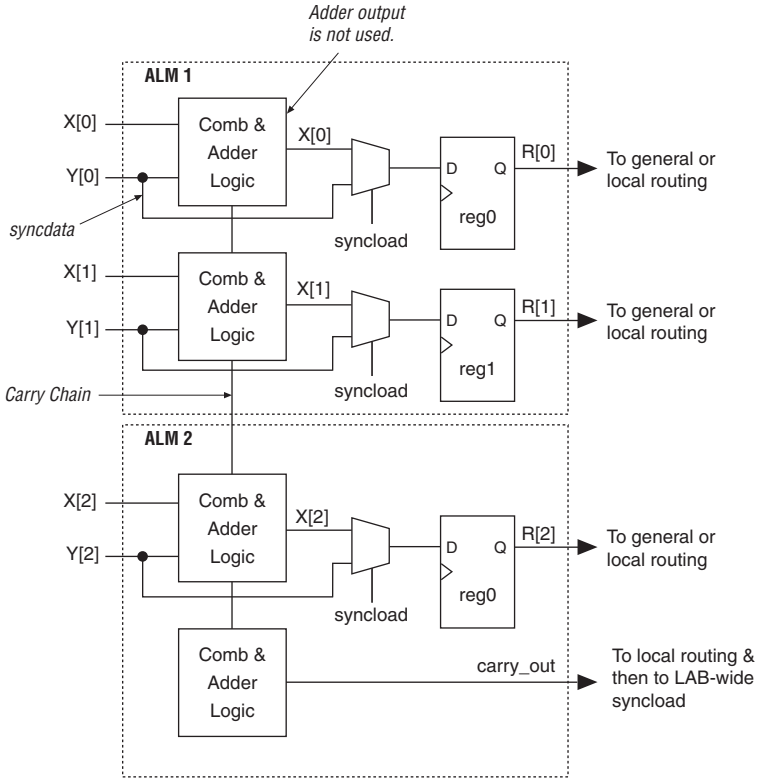
The four LUTs share the `dataa` and `datab` inputs. As shown in Figure 2–11, the carry-in signal feeds to `adder0`, and the carry-out from `adder0` feeds to carry-in of `adder1`. The carry-out from `adder1` drives to `adder0` of the next ALM in the LAB. ALMs in arithmetic mode can drive out registered and/or unregistered versions of the adder outputs.

Figure 2–11. ALM in Arithmetic Mode



While operating in arithmetic mode, the ALM can support simultaneous use of the adder's carry output along with combinational logic outputs. In this operation, the adder output is ignored. This usage of the adder with the combinational logic output provides resource savings of up to 50% for functions that can use this ability. An example of such functionality is a conditional operation, such as the one shown in Figure 2–12.

Figure 2–12. Conditional Operation Example



The equation for this example is:

$R = (X < Y) ? Y : X$

To implement this function, the adder is used to subtract *Y* from *X*. If *X* is less than *Y*, the `carry_out` signal is 1. The `carry_out` signal is fed to an adder where it drives out to the LAB local interconnect. It then feeds to the LAB-wide `syncload` signal. When asserted, `syncload` selects the `syncdata` input. In this case, the data *Y* drives the syncdata inputs to the registers. If *X* is greater than or equal to *Y*, the `syncload` signal is de-asserted and *X* drives the data port of the registers.

The arithmetic mode also offers clock enable, counter enable, synchronous up/down control, add/subtract control, synchronous clear, and synchronous load. The LAB local interconnect data inputs generate the clock enable, counter enable, synchronous up/down, and add/subtract control signals. These control signals are good candidates for the inputs that are shared between the four LUTs in the ALM. The synchronous clear and synchronous load options are LAB-wide signals that affect all registers in the LAB. These signals can also be individually disabled or enabled per register. The Quartus II software automatically places any registers that are not used by the counter into other LABs.

Carry Chain

The carry chain provides a fast carry function between the dedicated adders in arithmetic or shared arithmetic mode. The two-bit carry select feature in Stratix III devices halves the propagation delay of carry chains within the ALM. Carry chains can begin in either the first ALM or the sixth ALM in an LAB. The final carry-out signal is routed to a ALM, where it is fed to local, row, or column interconnects.

The Quartus II Compiler automatically creates carry chain logic during design processing, or you can create it manually during design entry. Parameterized functions such as LPM functions automatically take advantage of carry chains for the appropriate functions.

The Quartus II Compiler creates carry chains longer than 20 (10 ALMs in arithmetic or shared arithmetic mode) by linking LABs together automatically. For enhanced fitting, a long carry chain runs vertically allowing fast horizontal connections to TriMatrix™ memory and DSP blocks. A carry chain can continue as far as a full column.

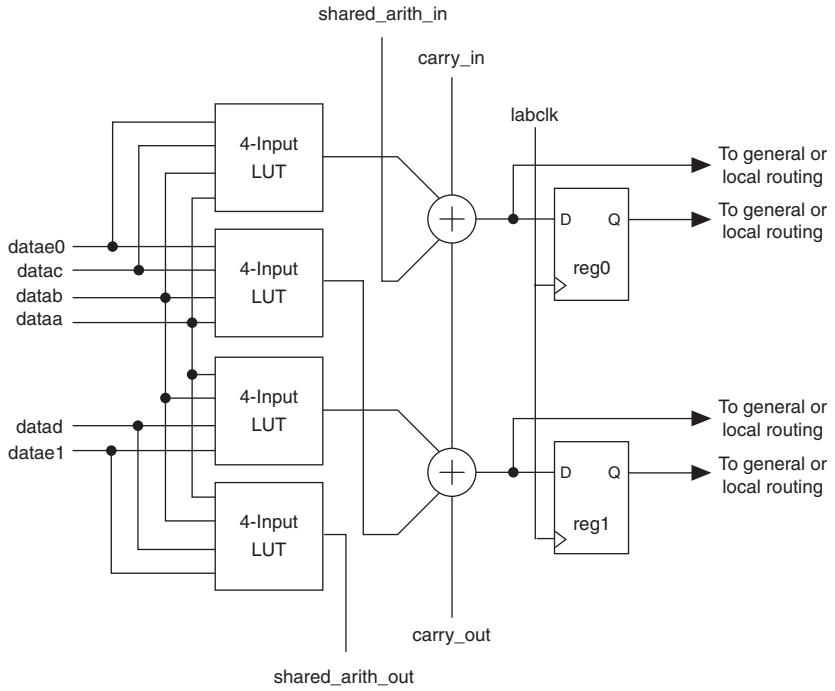
To avoid routing congestion in one small area of the device when a high fan-in arithmetic function is implemented, the LAB can support carry chains that only utilize either the top half or the bottom half of the LAB before connecting to the next LAB. This leaves the other half of the ALMs in the LAB available for implementing narrower fan-in functions in normal mode. Carry chains that use the top five ALMs in the first LAB carry into the top half of the ALMs in the next LAB within the column. Carry chains that use the bottom five ALMs in the first LAB carry into the bottom half of the ALMs in the next LAB within the column. In every alternate LAB column, the top half can be bypassed; in the other MLAB columns, the bottom half can be bypassed.

For more information on carry chain interconnect, refer to "ALM Interconnects" on page 2–20.

Shared Arithmetic Mode

In shared arithmetic mode, the ALM can implement a three-input add within an ALM. In this mode, the ALM is configured with 4 four-input LUTs. Each LUT either computes the sum of three inputs or the carry of three inputs. The output of the carry computation is fed to the next adder (either to `adder1` in the same ALM or to `adder0` of the next ALM in the LAB) via a dedicated connection called the shared arithmetic chain. This shared arithmetic chain can significantly improve the performance of an adder tree by reducing the number of summation stages required to implement an adder tree. Figure 2–13 shows the ALM using this feature.

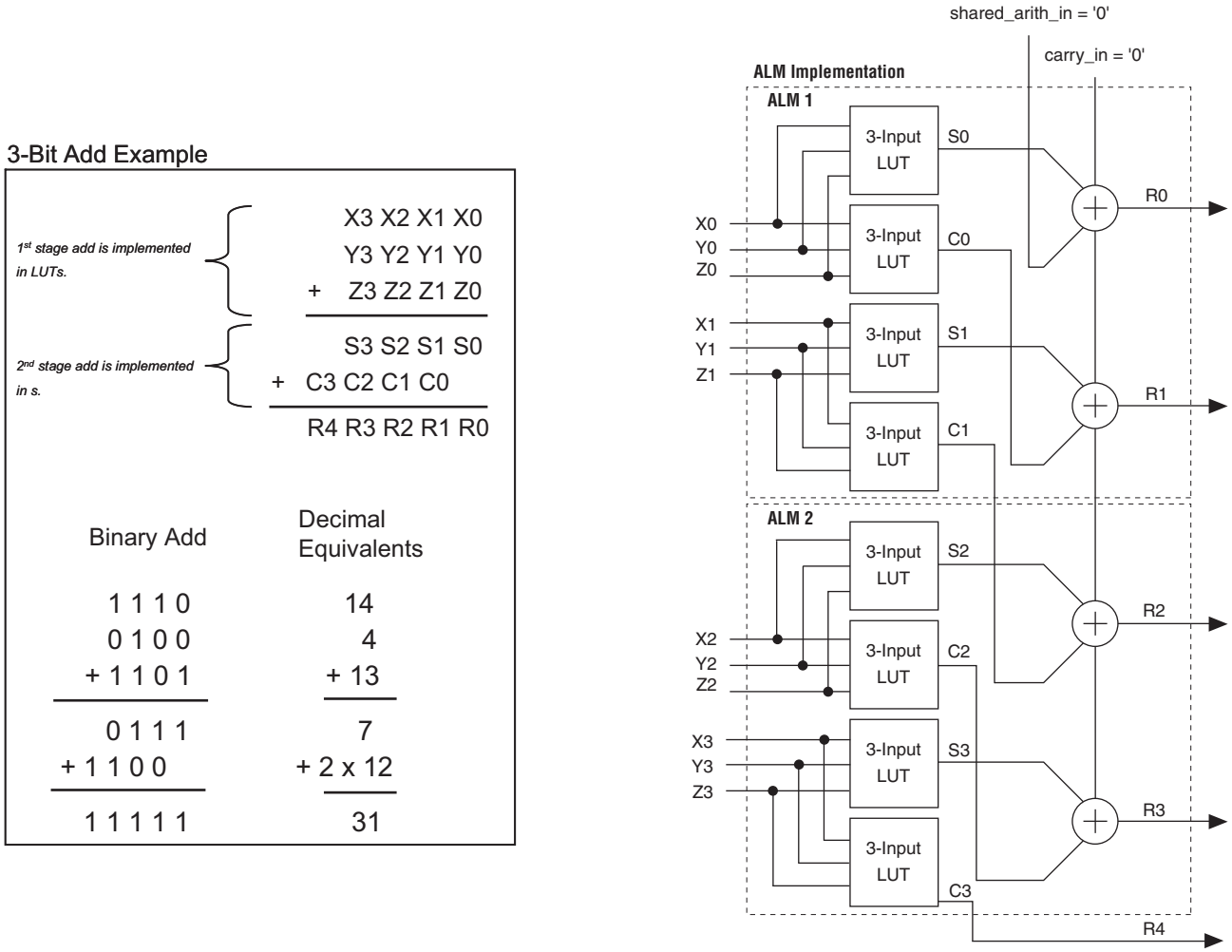
Figure 2–13. ALM in Shared Arithmetic Mode



You can find adder trees in many different applications. For example, the summation of the partial products in a logic-based multiplier can be implemented in a tree structure. Another example is a correlator function that can use a large adder tree to sum filtered data samples in a given time frame to recover or to de-spread data that was transmitted utilizing spread spectrum technology.

An example of a three-bit add operation utilizing the shared arithmetic mode is shown in Figure 2–14. The partial sum (S[3..0]) and the partial carry (C[3..0]) is obtained using the LUTs, while the result (R[3..0]) is computed using the dedicated adders.

Figure 2–14. Example of a 3-Bit Add Utilizing Shared Arithmetic Mode



Shared Arithmetic Chain

The shared arithmetic chain available in enhanced arithmetic mode allows the ALM to implement a three-input add. This significantly reduces the resources necessary to implement large adder trees or correlator functions.

The shared arithmetic chains can begin in either the first or sixth ALM in an LAB. The Quartus II Compiler creates shared arithmetic chains longer than 20 (10 ALMs in arithmetic or shared arithmetic mode) by linking LABs together automatically. For enhanced fitting, a long shared arithmetic chain runs vertically allowing fast horizontal connections to TriMatrix memory and DSP blocks. A shared arithmetic chain can continue as far as a full column.

Similar to the carry chains, the top and bottom half of shared arithmetic chains in alternate LAB columns can be bypassed. This capability allows the shared arithmetic chain to cascade through half of the ALMs in a LAB while leaving the other half available for narrower fan-in functionality. Every other LAB column is top-half bypassable, while the other LAB columns are bottom-half bypassable.

Refer to "ALM Interconnects" on page 2–20 for more information on shared arithmetic chain interconnect.

LUT-Register Mode

LUT-Register mode allows third register capability within an ALM. Two internal feedback loops allow combinational `ALUT1` to implement the master latch and combinational `ALUT0` to implement the slave latch needed for the third register. The LUT register shares its clock, clock enable, and asynchronous clear sources with the top dedicated register. Figure 2–15 shows the register constructed using two combinational blocks within the ALM. Figure 2–16 shows the ALM in LUT-Register mode.

Figure 2–15. LUT Register from Two Combinational Blocks

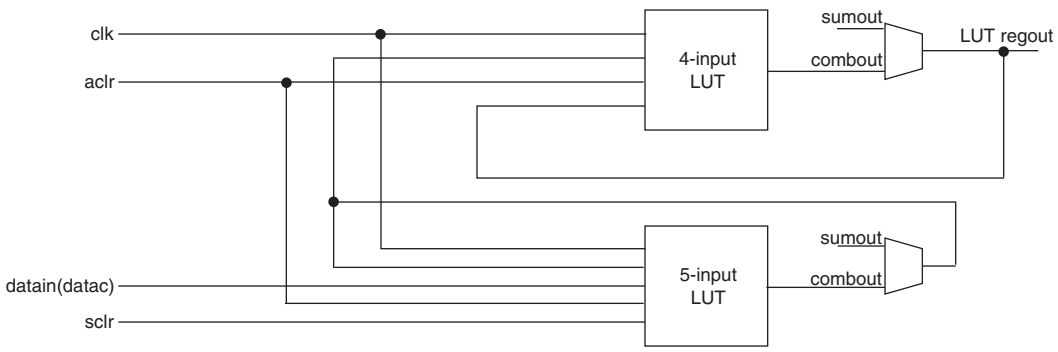
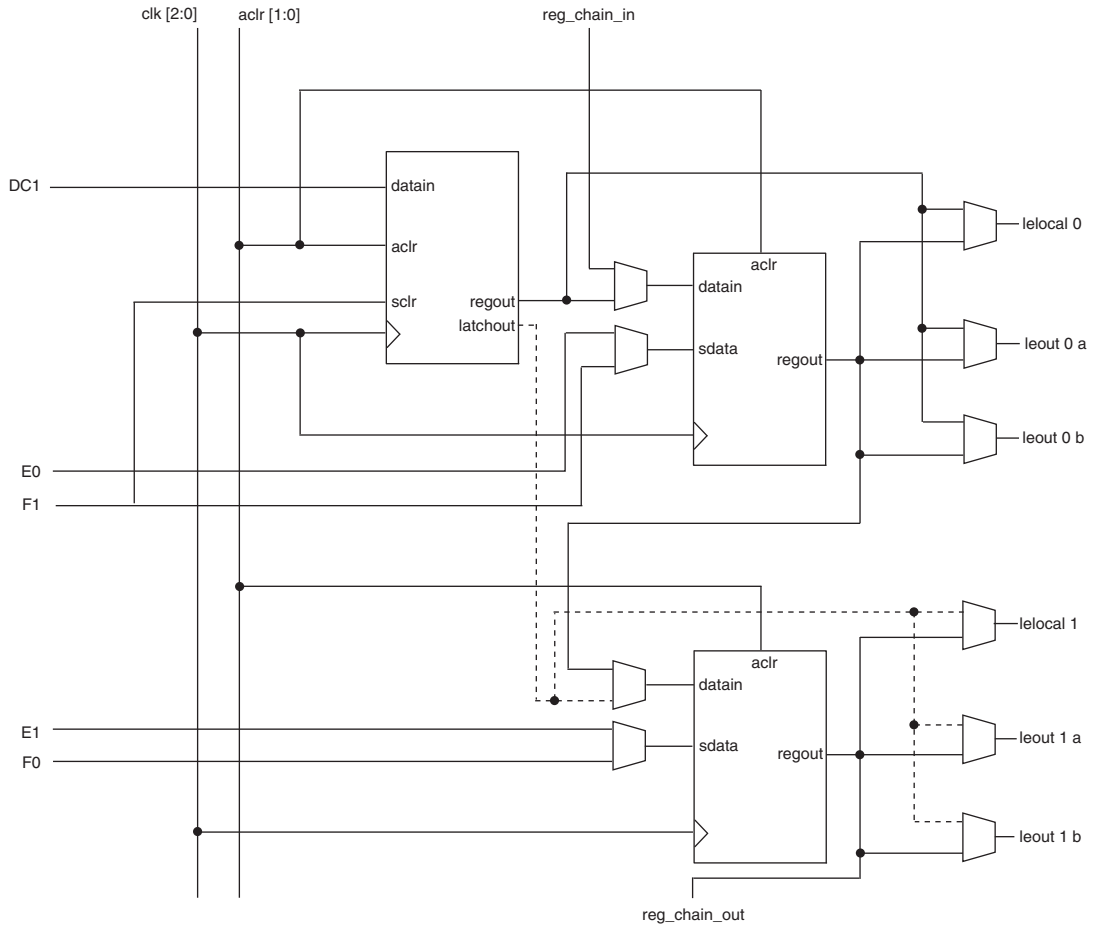


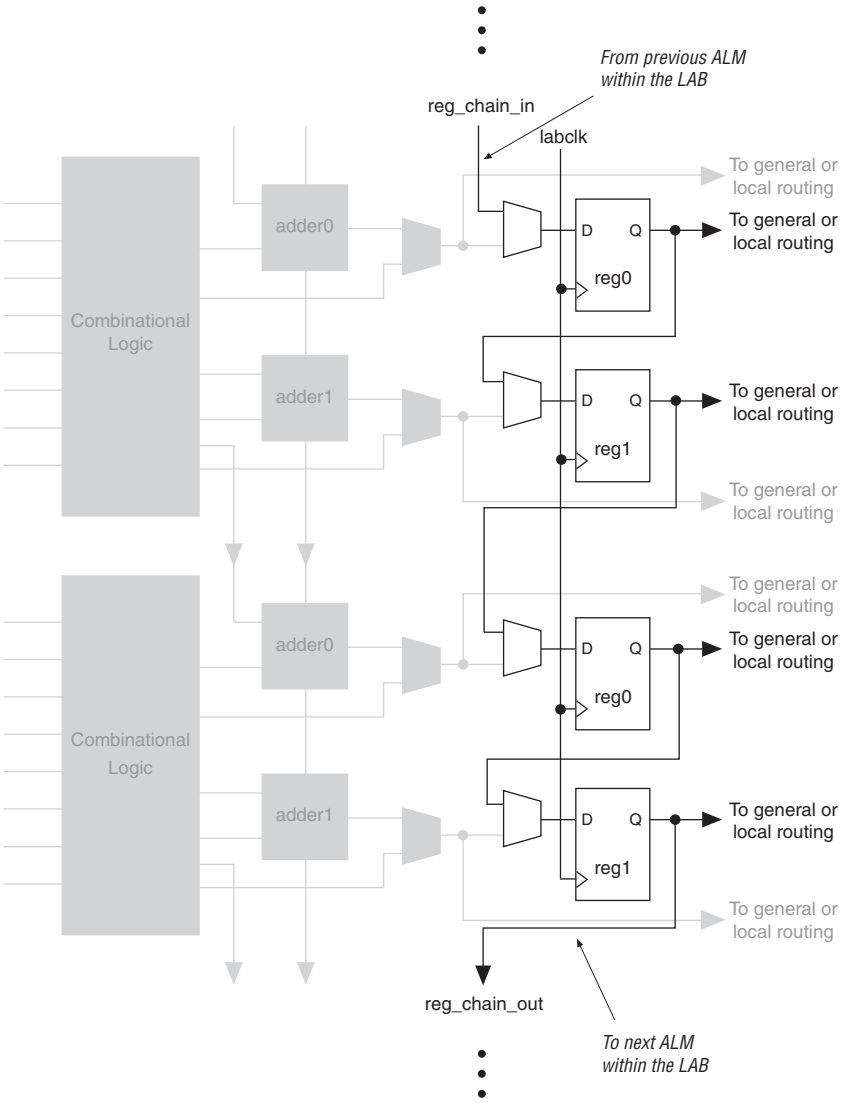
Figure 2–16. ALM in LUT-Register Mode with 3-Register Capability



Register Chain

In addition to the general routing outputs, the ALMs in an LAB have register chain outputs. The register chain routing allows registers in the same LAB to be cascaded together. The register chain interconnect allows a LAB to use LUTs for a single combinational function and the registers to be used for an unrelated shift register implementation. These resources speed up connections between ALMs while saving local interconnect resources (refer to Figure 2–17). The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance.

Figure 2–17. Register Chain within an LAB *(Note 1)*



Note to Figure 2–17:

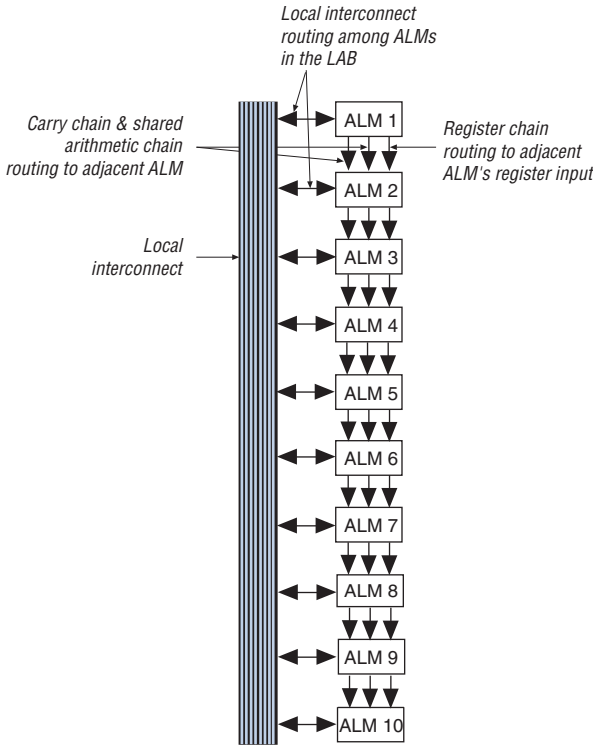
(1) You can use the combinational or adder logic to implement an unrelated, un-registered function.

For more information on register chain interconnect, refer to "ALM Interconnects" on page 2–20.

ALM Interconnects

There are three dedicated paths between ALMs: Register Cascade, Carry-chain, and Shared Arithmetic chain. Stratix III devices include an enhanced interconnect structure in LABs for routing shared arithmetic chains and carry chains for efficient arithmetic functions. The register chain connection allows the register output of one ALM to connect directly to the register input of the next ALM in the LAB for fast shift registers. These ALM-to-ALM connections bypass the local interconnect. The Quartus II Compiler automatically takes advantage of these resources to improve utilization and performance.Figure 2–18 shows the shared arithmetic chain, carry chain, and register chain interconnects.

Figure 2–18. Shared Arithmetic Chain, Carry Chain, and Register Chain Interconnects



For information about routing between LABs, refer to the *MultiTrack Interconnect in Stratix III Devices* chapter in volume 1 of the *Stratix III Device Handbook*.

Clear and Preset Logic Control

LAB-wide signals control the logic for the register's clear signal. The ALM directly supports an asynchronous clear function. You can achieve the register preset through the Quartus II software's **NOT-gate push-back logic** option. Each LAB supports up to two clears.

Stratix III devices provide a device-wide reset pin (`DEV_CLRn`) that resets all registers in the device. An option set before compilation in the Quartus II software controls this pin. This device-wide reset overrides all other control signals.

LAB Power Management Techniques

The following techniques are used to manage static and dynamic power consumption within the LAB:

- Stratix III low-voltage devices (L ordering code suffix) offer selectable core voltage to reduce both DC and AC power.
- To save AC power, Quartus II forces all adder inputs low when ALM adders are not in use.
- Stratix III LABs operate in high-performance mode or low-power mode. The Quartus II software automatically chooses the appropriate mode for an LAB based on the design to optimize speed vs. leakage trade-offs.
- Clocks represent a significant portion of dynamic power consumption due to their high switching activity and long paths. The LAB clock that distributes a clock signal to registers within a LAB is a significant contributor to overall clock power consumption. Each LAB's clock and clock enable signal are linked. For example, a combinational ALUT or register in a particular LAB using the `labclk1` signal also uses the `labclkena1` signal. To disable LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock. The Quartus II software automatically promotes register-level clock enable signals to the LAB-level. All registers within an LAB that share a common clock and clock enable are controlled by a shared gated clock. To take advantage of these clock enables, use a clock enable construct in your HDL code for the registered logic.

Refer to the *Power Optimization* chapter in section 3 of the *Quartus II Handbook* for details on implementation.

For detailed information about Stratix III programmable power capabilities, refer to the *Programmable Power and Temperature Sensing Diode in Stratix III Devices chapter in volume 1* of the *Stratix III Device Handbook*.

Conclusion

Logic array block and adaptive logic modules are the basic building blocks of the Stratix III device. You can use these to configure logic functions, arithmetic functions, and register functions. The ALM provides advanced features with efficient logic utilization and is completely backward-compatible.

Chapter Revision History

Table 2–1shows the revision history for this document.

Table 2–1. Chapter Revision History(Sheet 1 of 2)

Date and Revision	Changes Made	Summary of Changes
February 2009, version 1.5	Removed “Referenced Documents” section.	—
October 2008, version 1.4	■ Updated “LAB Control Signals”, and “Carry Chain” Sections. ■ Updated New Document Format.	—

Table 2–1. Chapter Revision History(Sheet 2 of 2)

Date and Revision	Changes Made	Summary of Changes
May 2008, version 1.3	Updated Figure 2–2 and Figure 2–6.	—
October 2007, version 1.2	■ Added section “Referenced Documents”. ■ Added live links for references.	Minor changes.
May 2007, version 1.1	■ Minor formatting changes. ■ Updated Figure 2–6 to include a missing connection.	Minor changes.
November 2006, version 1.0	Initial Release.	—