7 Series FPGAs Configurable Logic Block

User Guide

UG474 (v1.7) November 17, 2014

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/01/2011	1.0	Xilinx Initial release.
03/28/2011	1.1	Added devices XC7K355T, XC7K420T, and XC7K480T to Table 1-2. Portions of the text have been revised for clarity.
09/30/2011	1.2	Added last sentence under 7 Series CLB Features. Added Table 1-2 and Table 1-3. Updated CLB features in Table 1-2. Added first sentence under CLB Arrangement, added ASMBL Architecture section, and CLB Slices heading. Added last paragraph under Carry Logic. Added lasts sentence under Using Carry Logic. Added second sentence under Slice Multiplexer Timing Parameters. Modified Table 5-2, Table 5-4, and Table 5-5 for clarity. Added Devices Using Stacked Silicon Interconnect (SSI) Technology section.
01/30/2012	1.3	Revised Table 1-1. Added fifth paragraph under Distributed RAM (Available in SLICEM Only). Clarified last paragraph under Global Controls GSR and GTS.
11/05/2012	1.4	Changed "uniformity" to "optimized" in last bullet under 7 Series CLB Features. Changed "unified" to "scalable" in first sentence under Device Resources. Deleted 7A350T device from Table 1-1. Deleted 7V1500T and 7VH290T devices from Table 1-3. Added reference to 7 Series FPGA Libraries Guide to Distributed RAM (Available in SLICEM Only), Shift Registers (Available in SLICEM Only), and Flip-Flop Primitives. Changed "T_{CEO}" to "T_{CECK}" in Figure 5-2 and first bullet under General Timing Characteristics.

Date	Version	Revision
08/6/2013	1.5	Added Artix®-7 devices. Updated references to implementation tools.
08/11/2014	1.6	Revised footnotes in Table 1-1 through Table 1-3. Revised polarity from *independent* to *programmable* in Control Signals, page 22. Added Primitive column to Table 2-3 and removed footnotes. Renamed or made minor revisions to Figure 2-6 through Figure 2-14. Revised sections Clock – WCLK, page 49, Clock – CLK, page 50, and Clock - C, page 51.
11/17/2014	1.7	Updated Table 1-1 for new Artix 7A15T device.

Table of Contents

Chapter 6: Advanced Topics

About This Guide

Xilinx® 7 series FPGAs include three FPGA families that are all designed for lowest power to enable a common design to scale across families for optimal power, performance, and cost. The Artix®-7 family is optimized for lowest cost and absolute power for the highest volume applications. The Virtex®-7 family is optimized for highest system performance and capacity. The Kintex®-7 family is an innovative class of FPGAs optimized for the best price-performance. This guide serves as a technical reference describing the 7 series FPGAs configurable logic blocks (CLBs).

Usually, logic synthesis assigns the CLB resources without system designer intervention. It can be advantageous for the designer to understand certain CLB details, including the varying capabilities of the look-up tables (LUTs), the physical direction of the carry propagation, the number and distribution of the available flip-flops, and the availability of the very efficient shift registers. This guide describes these and other features of the CLB in detail.

This *7 Series FPGAs Configurable Logic Block User Guide*, part of an overall set of documentation on the 7 series FPGAs, is available on the Xilinx 7 Series documentation website.

Guide Contents

This manual contains these chapters:

- Chapter 1, Overview, provides basic information needed for the majority of users, including:
  - CLB Overview is targeted at the new user.
  - 7 Series CLB Features discusses what is new compared with the Spartan®-6 and Virtex®-6 FPGA families for the experienced user and provides design migration considerations.
  - Device Resources indicates the number of resources per device, and unity between different 7 series families.
  - Recommended Design Flow provides the basics of using CLB resources and lists key aspects to consider.
  - Pinout Planning discusses aspects of CLBs that might affect pin placement for a design.
- Chapter 2, Functional Details, lists architectural specifics for each CLB feature.
- Chapter 3, Design Entry, provides design entry guidelines and primitives for instantiation.
- Chapter 4, Applications, provides examples that use the CLB resources in larger applications.

- Chapter 5, Timing, contains timing models and defines CLB timing specifications from the respective 7 series FPGA data sheet.
- Chapter 6, Advanced Topics, discusses advanced features of the 7 series CLB.

Additional Support Resources

To find additional documentation, see the Xilinx website at:

www.xilinx.com/support/documentation/index

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

www.xilinx.com/support

XILINX

Chapter 1

Overview

CLB Overview

The 7 series configurable logic block (CLB) provides advanced, high-performance FPGA logic:

- Real 6-input look-up table (LUT) technology
- Dual LUT5 (5-input LUT) option
- Distributed Memory and Shift Register Logic capability
- Dedicated high-speed carry logic for arithmetic functions
- Wide multiplexers for efficient utilization

CLBs are the main logic resources for implementing sequential as well as combinatorial circuits. Each CLB element is connected to a switch matrix for access to the general routing matrix (shown in Figure 1-1). A CLB element contains a pair of slices.

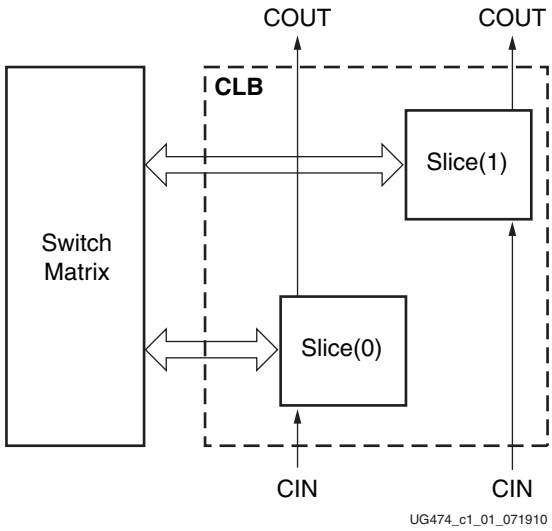

Figure 1-1: **Arrangement of Slices within the CLB**

The LUTs in 7 series FPGAs can be configured as either a 6-input LUT with one output, or as two 5-input LUTs with separate outputs but common addresses or logic inputs. Each 5-input LUT output can optionally be registered in a flip-flop. Four such 6-input LUTs and their eight flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a CLB. Four flip-flops per slice (one per LUT) can optionally be configured as latches. In that case, the remaining four flip-flops in that slice must remain unused.

Approximately two-thirds of the slices are SLICEL logic slices and the rest are SLICEM, which can also use their LUTs as distributed 64-bit RAM or as 32-bit shift registers (SRL32) or as two SRL16s. Modern synthesis tools take advantage of these highly efficient logic, arithmetic, and memory features. Expert designers can also instantiate them.

7 Series CLB Features

The 7 series CLB is identical to that in the Virtex®-6 FPGA family. The CLB is very similar to that of the Spartan®-6 FPGA family with these differences:

- Columnar architecture
  - Scales easily to higher densities
  - More routing between CLBs
- SLICEL and SLICEM only (no Spartan-6 FPGA SLICEX)
  - All slices support carry logic
  - More optimized

The common features in the CLB structure simplify design migration from the Spartan-6 and Virtex-6 families to the 7 series devices. The unique floorplan means that location constraints should be removed before implementing designs originally targeted to earlier FPGAs. The interconnect routing resources are increased in size, quantity, and flexibility relative to the Virtex-6 FPGA family, improving the quality of automatic place and route results.

Device Resources

The CLB resources are scalable across all the 7 series families, providing a common architecture that improves efficiency, IP implementation, and design migration. The number of CLBs and the ratio between CLBs and other device resources differentiates the 7 series families. Migration between the 7 series families does not require any design changes for the CLBs.

Device capacity is often measured in terms of logic cells, which are the logical equivalent of a classic four-input LUT and a flip-flop. The 7 series FPGA CLB six-input LUT, abundant flip-flops and latches, carry logic, and the ability to create distributed RAM or shift registers in the SLICEM, increase the effective capacity. The ratio between the number of logic cells and 6-input LUTs is 1.6:1.

7 Series FPGA CLB Resources

Table 1-1 through Table 1-3 show the available CLB resources for the Artix®-7, Kintex®-7, and Virtex®-7 FPGAs. Refer to DS180, *7 Series FPGAs Overview* for the most up-to-date information.

Table 1-1: **Artix-7 FPGA CLB Resources**

Device	Slices(1)	SLICEL	SLICEM	6-input LUTs	Distributed RAM (Kb)	Shift Register (Kb)	Flip-Flops
7A15T	2,600(2)	1,800	800	10,400	200	100	20,800
7A35T	5,200(2)	3,600	1,600	20,800	400	200	41,600

Table 1-1: **Artix-7 FPGA CLB Resources *(Cont'd)***

Device	Slices[1]	SLICEL	SLICEM	6-input LUTs	Distributed RAM (Kb)	Shift Register (Kb)	Flip-Flops
7A50T	8,150	5,750	2,400	32,600	600	300	65,200
7A75T	11,800[2]	8,232	3,568	47,200	892	446	94,400
7A100T	15,850	11,100	4,750	63,400	1,188	594	126,800
7A200T	33,650	22,100	11,550	134,600	2,888	1,444	269,200

Notes:

1. Each 7 series FPGA slice contains four LUTs and eight flip-flops; only SLICEMs can use their LUTs as distributed RAM or SRLs.
2. Number of slices corresponding to the number of LUTs and flip-flops supported in the device.

Table 1-2: **Kintex-7 FPGA CLB Resources**

Device	Slices[1]	SLICEL	SLICEM	6-input LUTs	Distributed RAM (Kb)	Shift Register (Kb)	Flip-Flops
7K70T	10,250	6,900	3,350	41,000	838	419	82,000
7K160T	25,350	16,600	8,750	101,400	2,188	1,094	202,800
7K325T	50,950	34,950	16,000	203,800	4,000	2,000	407,600
7K355T	55,650	35,300	20,350	222,600	5,088	2,544	445,200
7K410T	63,550	40,900	22,650	254,200	5,663	2,831	508,400
7K420T	65,150[2]	41,400	23,750	260,600	5,938	2,969	521,200
7K480T	74,650	47,500	27,150	298,600	6,788	3,394	597,200

Notes:

1. Each 7 series FPGA slice contains four LUTs and eight flip-flops; only SLICEMs can use their LUTs as distributed RAM or SRLs.
2. Number of slices corresponding to the number of LUTs and flip-flops supported in the device.

Table 1-3: **Virtex-7 FPGA CLB Resources**

Device	Slices[1]	SLICEL	SLICEM	6-input LUTs	Distributed RAM (Kb)	Shift Register (Kb)	Flip-Flops
7V585T	91,050	63,300	27,750	364,200	6,938	3,469	728,400
7V2000T	305,400	219,200	86,200	1,221,600	21,550	10,775	2,443,200
7VX330T	51,000	33,450	17,550	204,000	4,388	2,194	408,000
7VX415T	64,400	38,300	26,100	257,600	6,525	3,263	515,200
7VX485T	75,900	43,200	32,700	303,600	8,175	4,088	607,200
7VX550T	86,600[2]	51,700	34,900	346,400	8,725	4,363	692,800
7VX690T	108,300	64,750	43,550	433,200	10,888	5,444	866,400
7VX980T	153,000	97,650	55,350	612,000	13,838	6,919	1,224,000
7VX1140T	178,000	107,200	70,800	712,000	17,700	8,850	1,424,000

Table 1-3: **Virtex-7 FPGA CLB Resources *(Cont'd)***

Device	Slices[1]	SLICEL	SLICEM	6-input LUTs	Distributed RAM (Kb)	Shift Register (Kb)	Flip-Flops
7VH580T	90,700	55,300	35,400	362,800	8,850	4,425	725,600
7VH870T	136,900	83,750	53,150	547,600	13,275	6,638	1,095,200

Notes:

1. Each 7 series FPGA slice contains four LUTs and eight flip-flops; only SLICEMs can use their LUTs as distributed RAM or SRLs.
2. Number of slices corresponding to the number of LUTs and flip-flops supported in the device.

Recommended Design Flow

CLB resources are inferred for generic design logic and do not require instantiation. Good HDL design is sufficient. A few items to note:

- CLB flip-flops have either a set or a reset. The designer must not use both set and reset.
- Flip-flops are abundant. Pipelining should be considered to improve performance.
- Control inputs are shared across a slice or CLB. The number of unique control inputs required for a design should be minimized. Control inputs include clock, clock enable, set/reset, and write enable.
- A 6-input LUT can be used as a 32-bit shift register for efficient implementation.
- A 6-input LUT can be used as a 64 x 1 memory for small storage requirements.
- Dedicated carry logic implements arithmetic functions effectively.

These steps indicate the recommended design flow:

1. Implement the design using preferred methodologies (HDL, IP, etc.).
2. Evaluate utilization reports to determine resources used.

   Check to make sure arithmetic logic, distributed RAM, and SRL are used, when helpful.
3. Consider flip-flop usage.
   a. Pipeline for performance
   b. Use dedicated flip-flops at the outputs of dedicated resources (block RAM, DSP)
   c. Allow shift registers to use SRL (avoid set/resets)
4. Minimize the use of set/resets.

Pinout Planning

Although the use of most resources affects the resulting device pinout, CLB usage has little effect on pinouts because they are distributed throughout the device. The ASMBL™ architecture provides maximum flexibility with CLBs on both sides of most I/Os.

The best approach is to let the tools choose the I/O locations based on the FPGA requirements. Results can be adjusted if necessary for board layout considerations. The timing constraints should be set so that the tools can choose optimal placement for the design requirements.

Carry logic cascades vertically up a column, so wide arithmetic buses might drive a vertical orientation to other logic, including I/O.

While most 7 series devices are available in flip-chip packages, taking full advantage of the distributed I/O in the ASMBL architecture, the smaller devices are available in wire-bond packages at a lower cost. In these packages, some pins are naturally closer to the I/Os and special resources than others, so pin placement should be done after the internal logic is defined.

XILINX®

Chapter 2

Functional Details

This chapter provides a detailed view of the 7 series FPGAs CLB architecture. These details can be useful for design optimization and verification, but are not necessary for initiating a design. This chapter includes:

- CLB Arrangement
  Overview of slice locations and features within the CLB
- Slice Description
  Complete details of SLICEM and SLICEL
- Look-Up Table (LUT)
  Description of the logical function generators
- Storage Elements
  Description and controls for the latches and flip-flops
- Distributed RAM (Available in SLICEM Only)
  SLICEM ability to use LUTs as writable memory
- Shift Registers (Available in SLICEM Only)
  SLICEM ability to use LUTs as shift registers
- Multiplexers
  Dedicated gates for combining LUTs into wide functions
- Carry Logic
  Dedicated gates and cascading to implement efficient arithmetic functions

CLB Arrangement

The CLBs are arranged in columns in the 7 series FPGAs. The 7 series is the fourth generation to be based on the unique columnar approach provided by the ASMBL™ architecture.

ASMBL Architecture

Xilinx created the Advanced Silicon Modular Block (ASMBL) architecture to enable FPGA platforms with varying feature mixes optimized for different application domains. Through this innovation Xilinx offers a greater selection of devices, enabling customers to select the FPGA with the right mix of features and capabilities for their specific design. Figure 2-1 provides a high-level description of the different types of column-based resources.

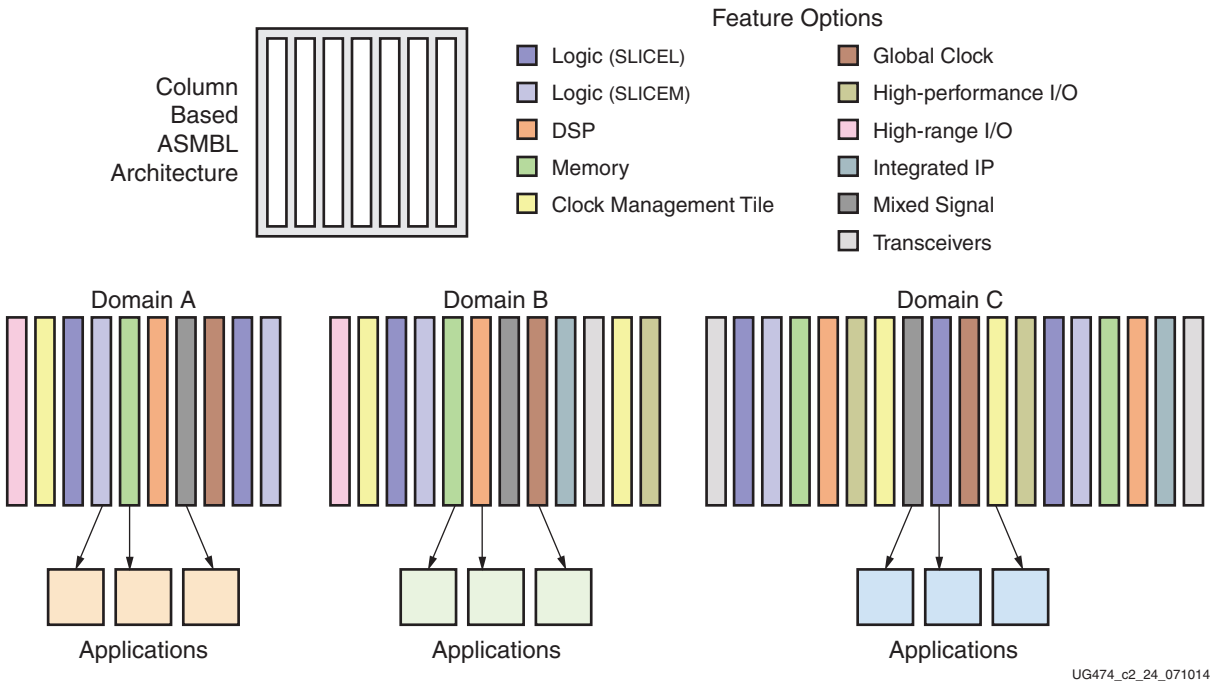

Figure 2-1: **ASMBL Architecture**

The ASMBL architecture breaks through traditional design barriers by:

- Eliminating geometric layout constraints such as dependencies between I/O count and array size.
- Enhancing on-chip power and ground distribution by allowing power and ground to be placed anywhere on the chip.
- Allowing disparate integrated IP blocks to be scaled independent of each other and surrounding resources.

SSI Technology

The 7 series FPGAs extend integration even higher by using the unique stacked silicon interconnect (SSI) technology. SSI technology enables multiple super logic regions (SLRs) to be combined on a passive interposer layer, to create a single FPGA with more than ten thousand inter-SLR connections. See Chapter 6, Advanced Topics for additional information.

CLB Slices

A CLB element contains a pair of slices, and each slice is composed of four 6-input LUTs and eight storage elements.

- SLICE(0) – slice at the bottom of the CLB and in the left column
- SLICE(1) – slice at the top of the CLB and in the right column

These two slices do not have direct connections to each other, and each slice is organized as a column. Each slice in a column has an independent carry chain.

The Xilinx tools designate slices with these definitions:

- An “X” followed by a number identifies the position of each slice in a pair as well as the column position of the slice. The “X” number counts slices starting from the bottom in sequence 0, 1 (the first CLB column); 2, 3 (the second CLB column); etc.
- A “Y” followed by a number identifies a row of slices. The number remains the same within a CLB, but counts up in sequence from one CLB row to the next CLB row, starting from the bottom.

Figure 2-2 shows four CLBs located in the bottom-left corner of the die.

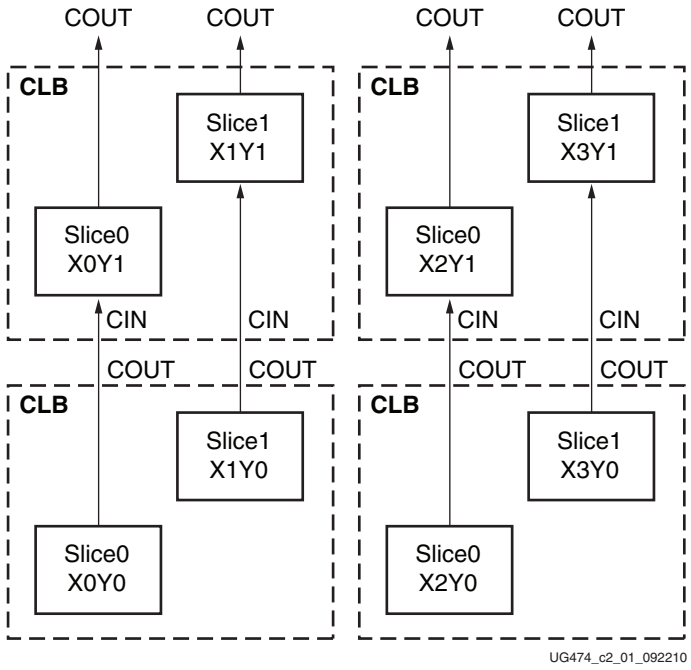

Figure 2-2: **Row and Column Relationship between CLBs and Slices**

CLB/Slice Configurations

Table 2-1 summarizes the logic resources in one CLB. Each SLICEM LUT can be configured as a look-up table, distributed RAM, or a shift register.

Table 2-1: **Logic Resources in One CLB**

Slices	LUTs	Flip-Flops	Arithmetic and Carry Chains	Distributed RAM(1)	Shift Registers(1)
2	8	16	2	256 bits	128 bits

Notes:

1. SLICEM only, SLICEL does not have distributed RAM or shift registers.

Slice Description

Every slice contains:

- Four logic-function generators (or look-up tables)
- Eight storage elements
- Wide-function multiplexers
- Carry logic

These elements are used by all slices to provide logic, arithmetic, and ROM functions. In addition, some slices support two additional functions: storing data using distributed RAM and shifting data with 32-bit registers. Slices that support these additional functions are called SLICEM; others are called SLICEL. SLICEM (shown in Figure 2-3) represents a superset of elements and connections found in all slices. SLICEL is shown in Figure 2-4. Each CLB can contain two SLICEL or a SLICEL and a SLICEM.

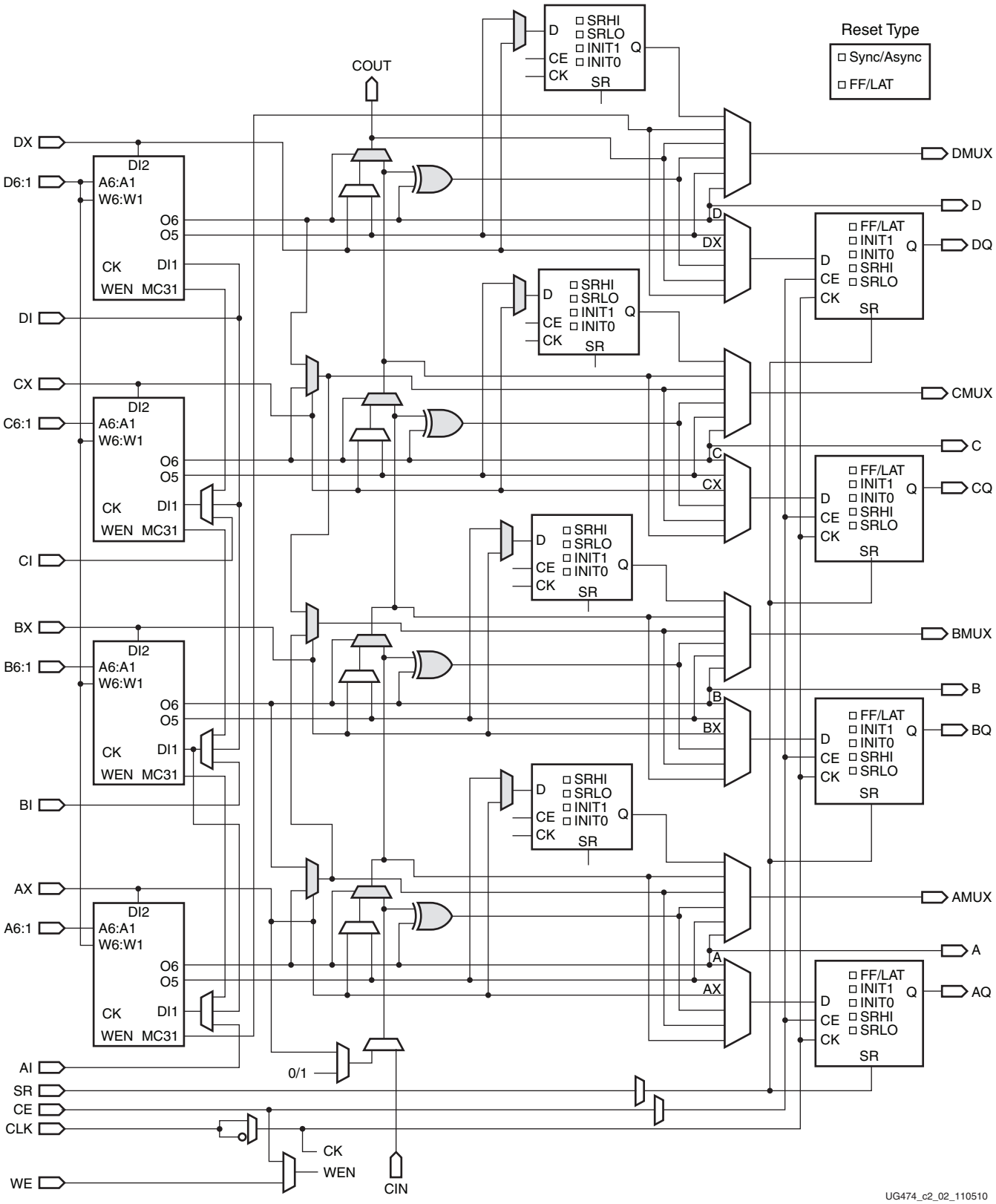

Figure 2-3: **Diagram of SLICEM**

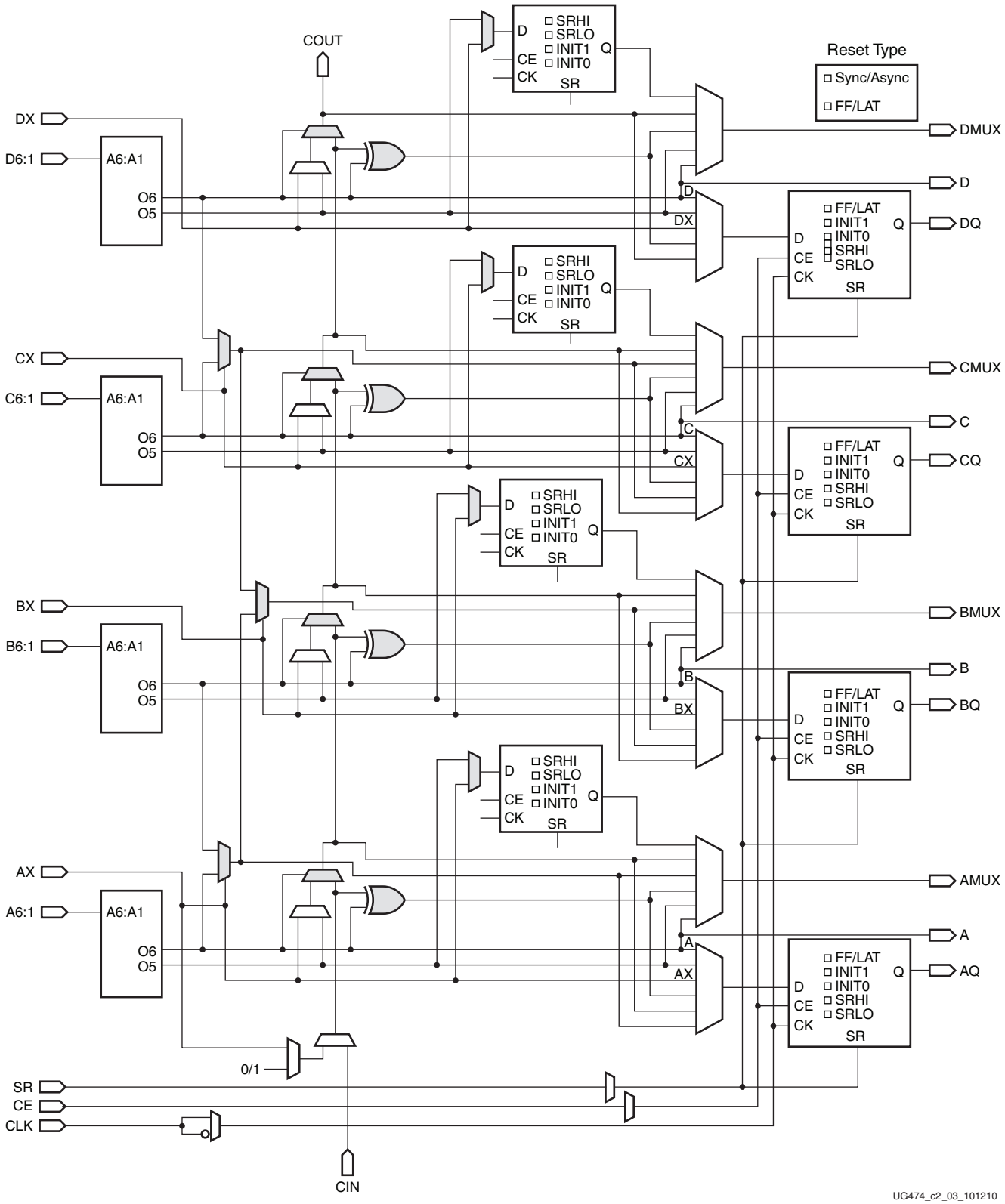

Figure 2-4: **Diagram of SLICEL**

Look-Up Table (LUT)

The function generators in 7 series FPGAs are implemented as six-input look-up tables (LUTs). There are six independent inputs (A inputs - A1 to A6) and two independent outputs (O5 and O6) for each of the four function generators in a slice (A, B, C, and D). The function generators can implement:

- Any arbitrarily defined six-input Boolean function
- Two arbitrarily defined five-input Boolean functions, as long as these two functions share common inputs
- Two arbitrarily defined Boolean functions of 3 and 2 inputs or less

A six-input function uses:

- A1-A6 inputs
- O6 output

Two five-input or less functions use:

- A1–A5 inputs
- A6 driven High
- O5 and O6 outputs

The propagation delay through a LUT is independent of the function implemented. Signals from the function generators can:

- Exit the slice (through A, B, C, D output for O6 or AMUX, BMUX, CMUX, DMUX output for O5)
- Enter the XOR dedicated gate from an O6 output
- Enter the carry-logic chain from an O5 output
- Enter the select line of the carry-logic multiplexer from O6 output
- Feed the D input of the storage element
- Go to F7AMUX/F7BMUX wide multiplexers from O6 output

In addition to the basic LUTs, slices contain three multiplexers (F7AMUX, F7BMUX, and F8MUX). These multiplexers are used to combine up to four function generators to provide any function of seven or eight inputs in a slice.

- F7AMUX: Used to generate seven input functions from LUTs A and B
- F7BMUX: Used to generate seven input functions from LUTs C and D
- F8MUX: Used to combine all LUTs to generate eight input functions.

Functions with more than eight inputs can be implemented using multiple slices. There are no direct connections between slices to form function generators greater than eight inputs within a CLB.

Storage Elements

There are eight storage elements per slice. Four can be configured as either edge-triggered D-type flip-flops or level-sensitive latches. The D input can be driven directly by a LUT output via AFFMUX, BFFMUX, CFFMUX, or DFFMUX, or by the BYPASS slice inputs bypassing the function generators via AX, BX, CX, or DX input. When configured as a latch, the latch is transparent when the CLK is Low.

There are four additional storage elements that can only be configured as edge-triggered D-type flip-flops. The D input can be driven by the O5 output of the LUT or the BYPASS slice inputs via AX, BX, CX, or DX input. When the original four storage elements are configured as latches, these four additional storage elements cannot be used.

Figure 2-5 shows both the register only and the register/latch configuration in a slice.

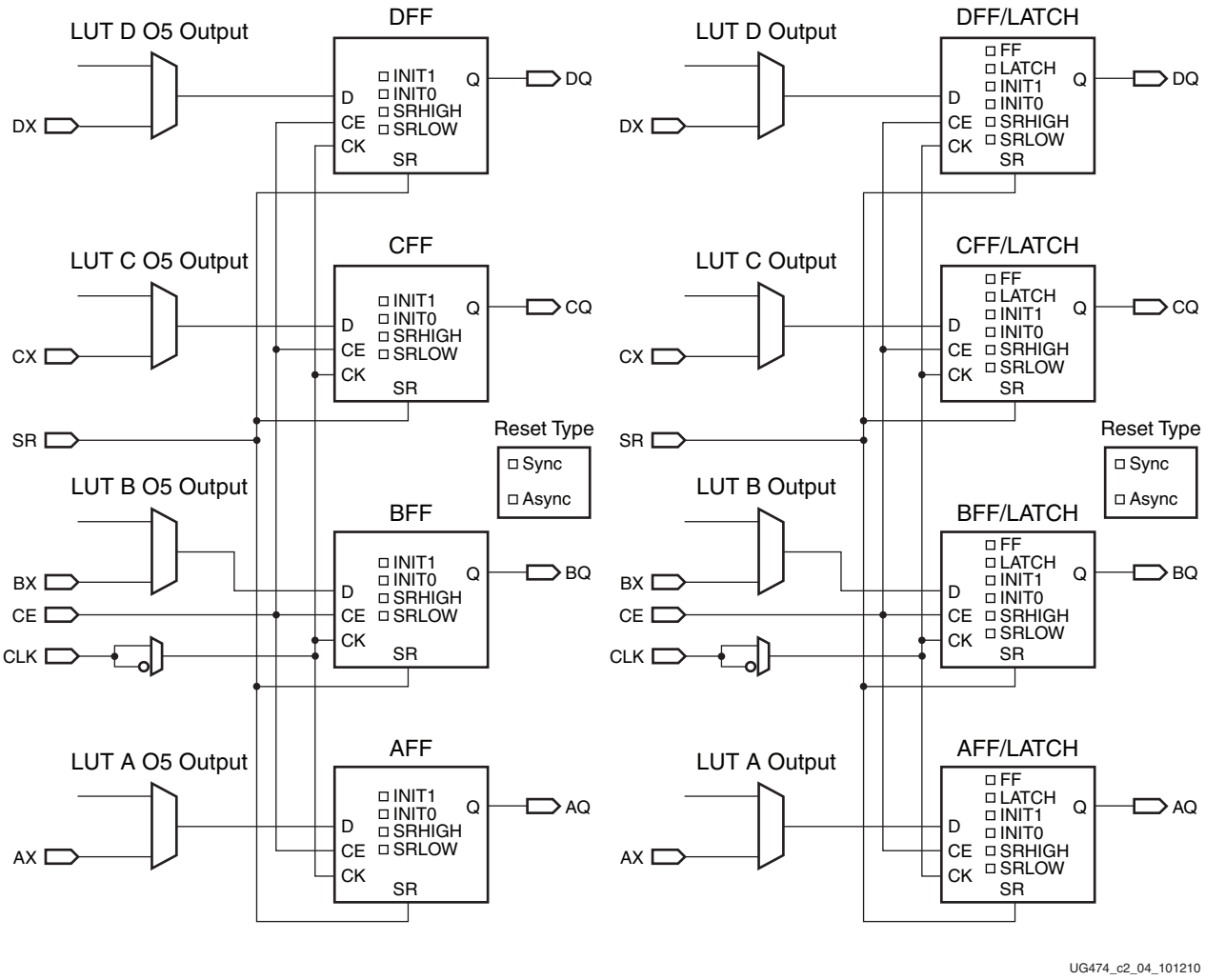

Figure 2-5: **Two Versions of Configuration in a Slice: 4 Registers Only and 4 Register/Latch**

Control Signals

The control signals clock (CLK), clock enable (CE), and set/reset (SR) are common to all storage elements in one slice. When one flip-flop in a slice has SR or CE enabled, the other flip-flops used in the slice also have SR or CE enabled by the common signal. Only the CLK signal has programmable polarity. Any inverter placed on the clock signal is automatically absorbed. The CE and SR signals are active-High.

These initialization options are available for storage elements:

- SRLOW: Synchronous or asynchronous Reset when CLB SR signal is asserted
- SRHIGH: Synchronous or asynchronous Set when CLB SR signal is asserted

- INIT0: Asynchronous Reset on power-up or global Set/Reset (see Global Controls GSR and GTS, page 72)
- INIT1: Asynchronous Set on power-up or global Set/Reset

The SR signal forces the storage element into the state specified by the SRHIGH or SRLOW attribute. SRHIGH forces a logic High at the storage element output when SR is asserted, while SRLOW forces a logic Low at the storage element output (see Table 2-2).

Table 2-2: **Truth Table when using SRLOW and SRHIGH**

SR	SRVAL	Function
0	SRLOW (default)	No Logic Change
1	SRLOW (default)	0
0	SRHIGH	No Logic Change
1	SRHIGH	1

SRHIGH and SRLOW can be set individually for each storage element in a slice. The choice of synchronous (SYNC) or asynchronous (ASYNC) set/reset (SRTYPE) cannot be set individually for each storage element in a slice.

The initial state after configuration or global initial state is defined by separate INIT0 and INIT1 attributes. By default, setting the SRLOW attribute sets INIT0, and setting the SRHIGH attribute sets INIT1. 7 series devices can set INIT0 and INIT1 independent of SRHIGH and SRLOW.

The configuration options for the set and reset functionality of a register or the four storage elements capable of functioning as a latch are:

- No set or reset
- Synchronous set
- Synchronous reset
- Asynchronous set (preset)
- Asynchronous reset (clear)

Distributed RAM (Available in SLICEM Only)

The function generators (LUTs) in SLICEMs can be implemented as a synchronous RAM resource called a distributed RAM element. Multiple LUTs in a SLICEM can be combined in various ways to store larger amount of data. RAM elements are configurable within a SLICEM to implement these configurations:

- Single-Port 32 x 1-bit RAM
- Dual-Port 32 x 1-bit RAM
- Quad-Port 32 x 2-bit RAM
- Simple Dual-Port 32 x 6-bit RAM
- Single-Port 64 x 1-bit RAM
- Dual-Port 64 x 1-bit RAM
- Quad-Port 64 x 1-bit RAM
- Simple Dual-Port 64 x 3-bit RAM
- Single-Port 128 x 1-bit RAM
- Dual-Port 128 x 1-bit RAM

- Single-Port 256 x 1-bit RAM

Distributed RAM modules are synchronous (write) resources. A synchronous read can be implemented with a flip-flop in the same slice. By using this flip-flop, the distributed RAM performance is improved by decreasing the delay into the clock-to-out value of the flip-flop. However, an additional clock latency is added. The distributed elements share the same clock input. For a write operation, the Write Enable (WE) input, driven by either the CE or WE pin of a SLICEM, must be set High.

Table 2-3 shows the number of LUTs (four per slice) occupied by each distributed RAM configuration. See UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide* for details of available distributed RAM primitives.

Table 2-3: **Distributed RAM Configuration**

RAM	Description	Primitive	Number of LUTs
32 x 1S	Single port	RAM32X1S	1
32 x 1D	Dual port	RAM32X1D	2
32 x 2Q	Quad port	RAM32M	4
32 x 6SDP	Simple dual port	RAM32M	4
64 x 1S	Single port	RAM64X1S	1
64 x 1D	Dual port	RAM64X1D	2
64 x 1Q	Quad port	RAM64M	4
64 x 3SDP	Simple dual port	RAM64M	4
128 x 1S	Single port	RAM128X1S	2
128 x 1D	Dual port	RAM128X1D	4
256 x 1S	Single port	RAM256X1S	4

Distributed RAM configurations include:

- Single port
  - Common address port for synchronous writes and asynchronous reads
    - Read and write addresses share the same address bus
- Dual port
  - One port for synchronous writes and asynchronous reads
    - One function generator is connected with the shared read and write port address
  - One port for asynchronous reads
    - Second function generator has the A inputs connected to a second read-only port address, and the WA inputs are shared with the first read/write port address

- Simple dual port
  - One port for synchronous writes (no data out/read port from the write port)
  - One port for asynchronous reads
- Quad port
  - One port for synchronous writes and asynchronous reads
  - Three ports for asynchronous reads

As shown in Figure 2-3, the common write port W6:W1 (WA[6:1] in the following figures) is always physically driven by the inputs to the D LUT using D[6:1]. The read ports are independent for each of the four LUTs. Therefore the D LUT is always effectively single port, even if DPRAM64 is selected as the LUT configuration. The other three LUTs are always effectively dual port, although SPRAM32 can be selected when the read and write addresses are connected together.

Figure 2-6 through Figure 2-14 illustrate various example distributed RAM configurations occupying one SLICEM.

When using x2 configurations (as in 32 X 2 Quad Port in Figure 2-6), A6 and WA6 are driven High by the software to keep O5 and O6 independent.

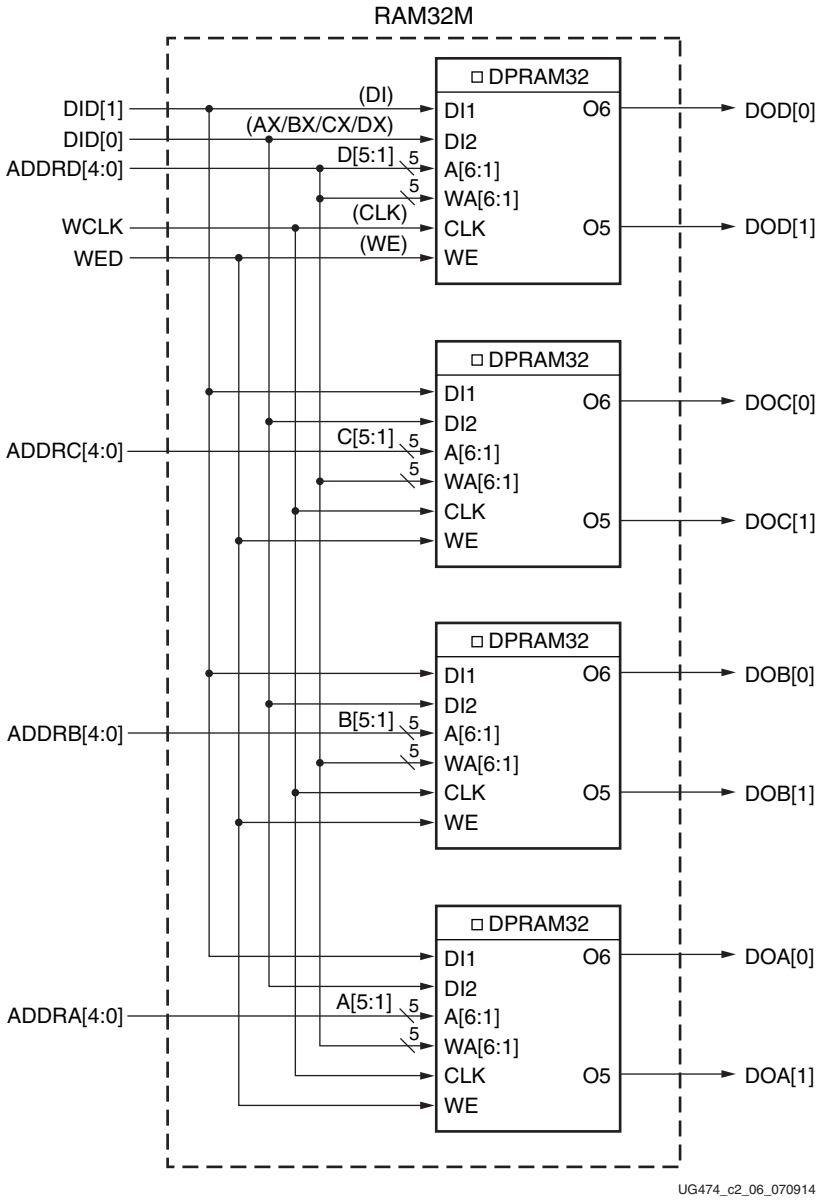

Figure 2-6: **32 X 2 Quad Port Distributed RAM (RAM32M)**

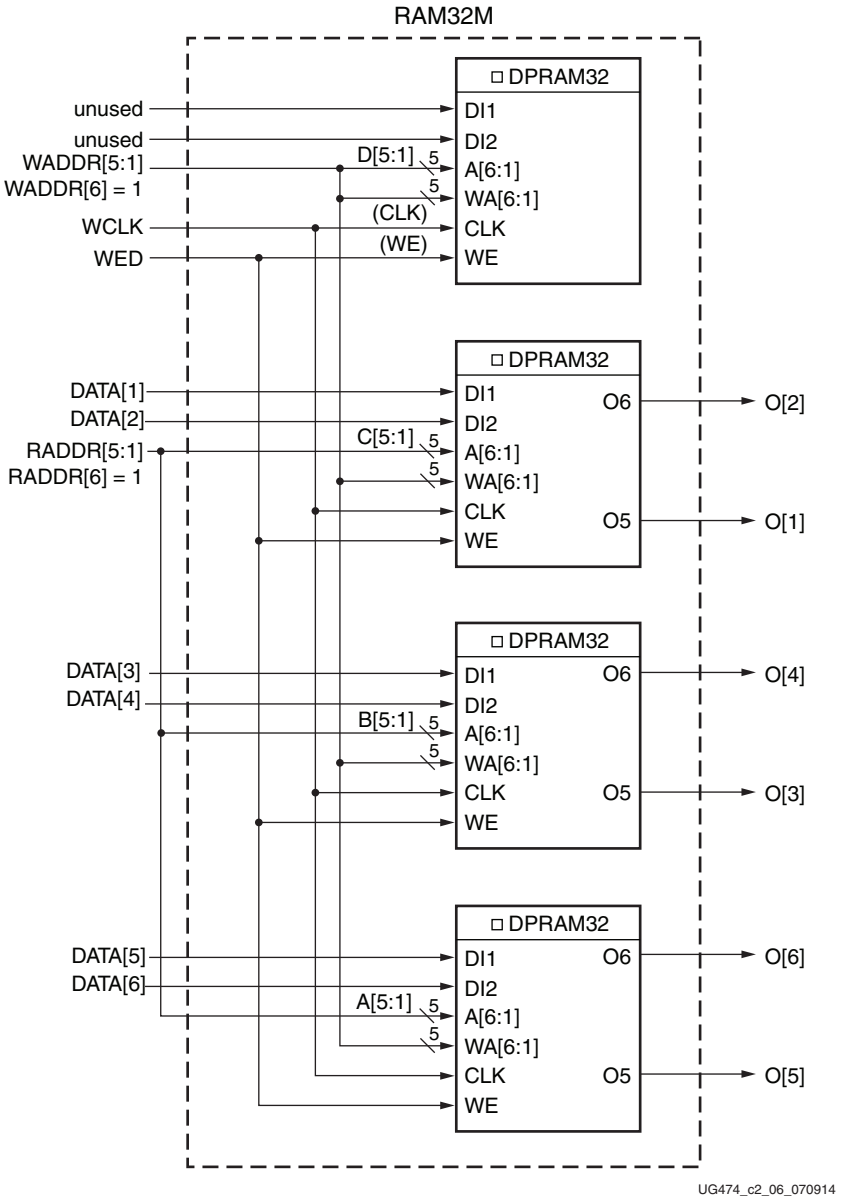

Figure 2-7: **32 X 6 Simple Dual Port Distributed RAM (RAM32M)**

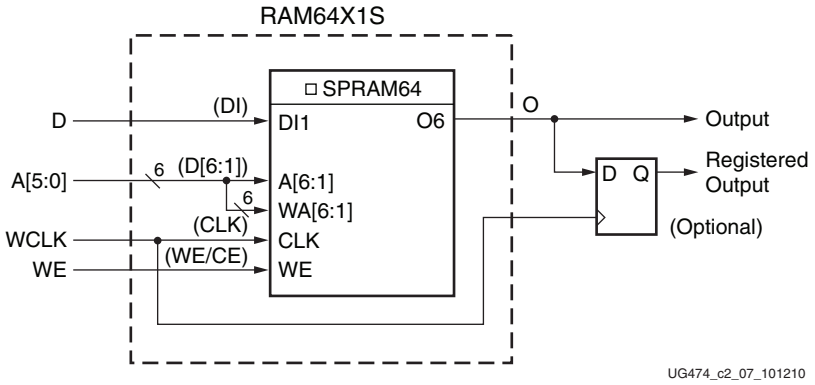

Figure 2-8: **64 X 1 Single Port Distributed RAM (RAM64X1S)**

If four single-port 64 x 1-bit modules are each built as shown in Figure 2-8, the four RAM64X1S primitives can occupy a SLICEM, as long as they share the same clock, write enable, and shared read and write port address inputs. This configuration equates to a 64 x 4-bit single-port distributed RAM.

If two dual-port 64 x 1-bit modules are each built as shown in Figure 2-9, the two RAM64X1D primitives can occupy a SLICEM, as long as they share the same clock, write enable, and shared read and write port address inputs. This configuration equates to a 64 x 2-bit dual-port distributed RAM.

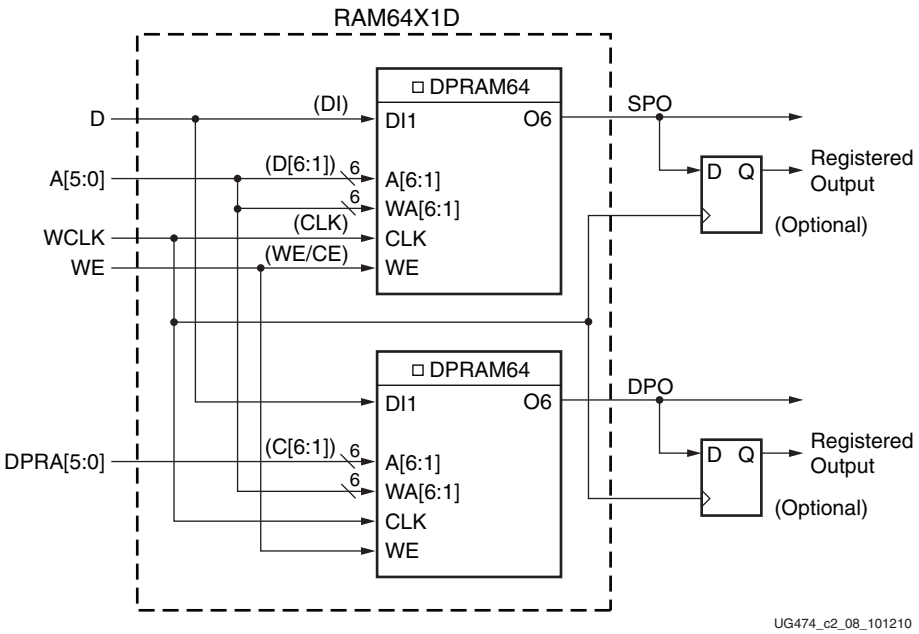

Figure 2-9: **64 X 1 Dual Port Distributed RAM (RAM64X1D)**

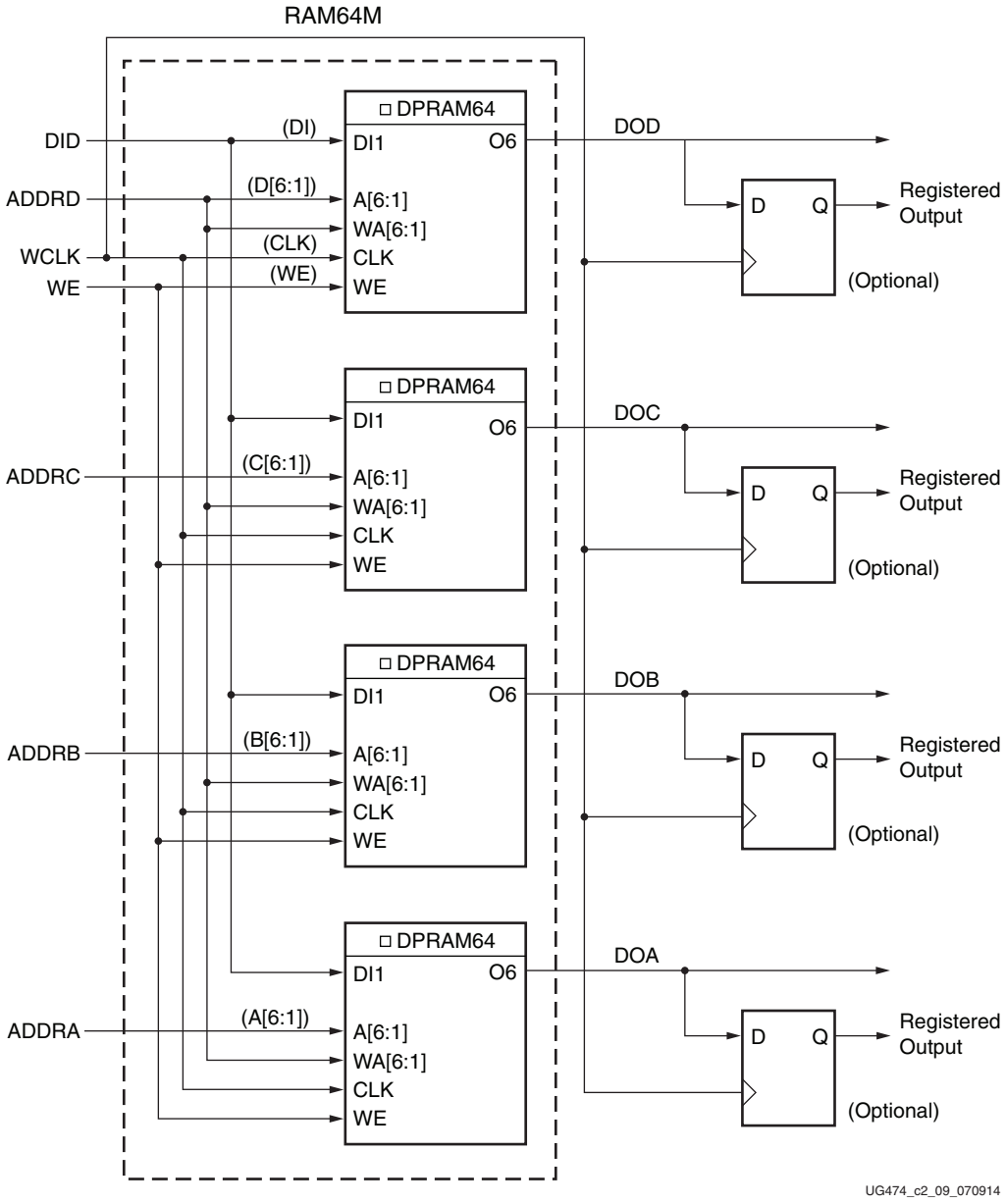

Figure 2-10: **64 X 1 Quad Port Distributed RAM (RAM64M)**

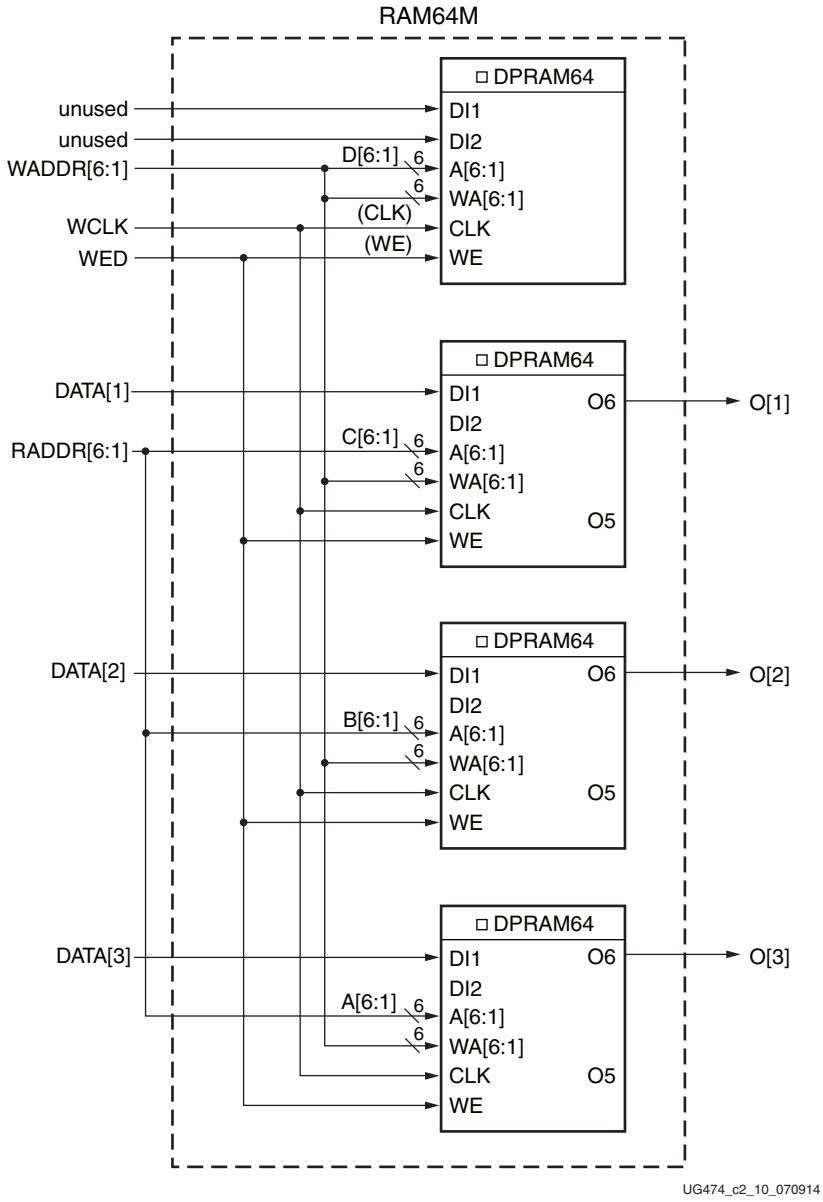

Figure 2-11: **64 X 3 Simple Dual Port Distributed RAM (RAM64M)**

Implementation of distributed RAM configurations with depth greater than 64 requires the usage of wide-function multiplexers (F7AMUX, F7BMUX, and F8MUX), as shown in Figure 2-12 through Figure 2-14.

If two single-port 128 x 1-bit modules are each built as shown in Figure 2-12, the two RAM128X1S primitives can occupy a SLICEM, as long as they share the same clock, write enable, and shared read and write port address inputs. This configuration equates to 128 x 2-bit single-port distributed RAM.

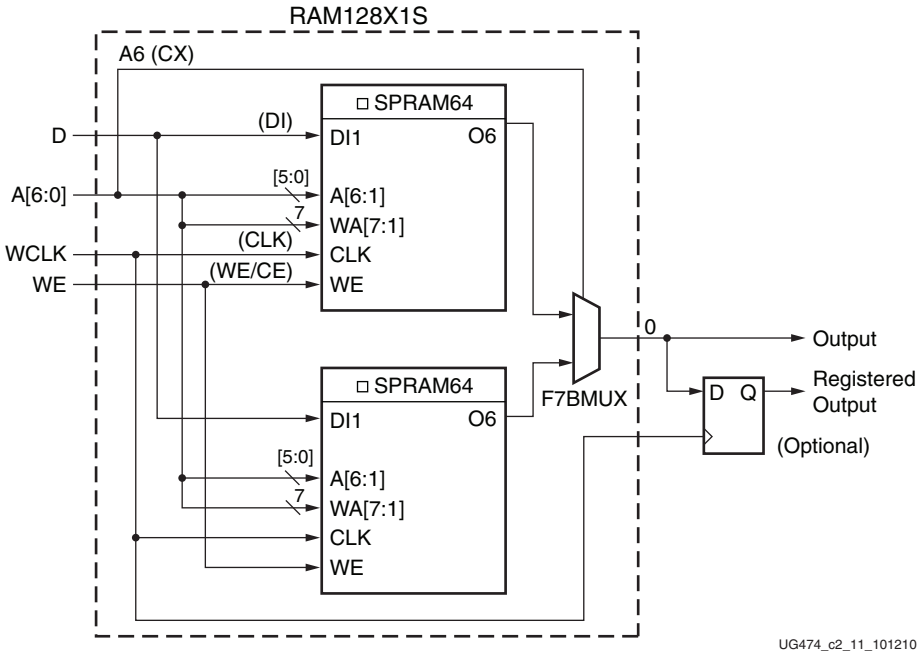

Figure 2-12: **128 X 1 Single Port Distributed RAM (RAM128X1S)**

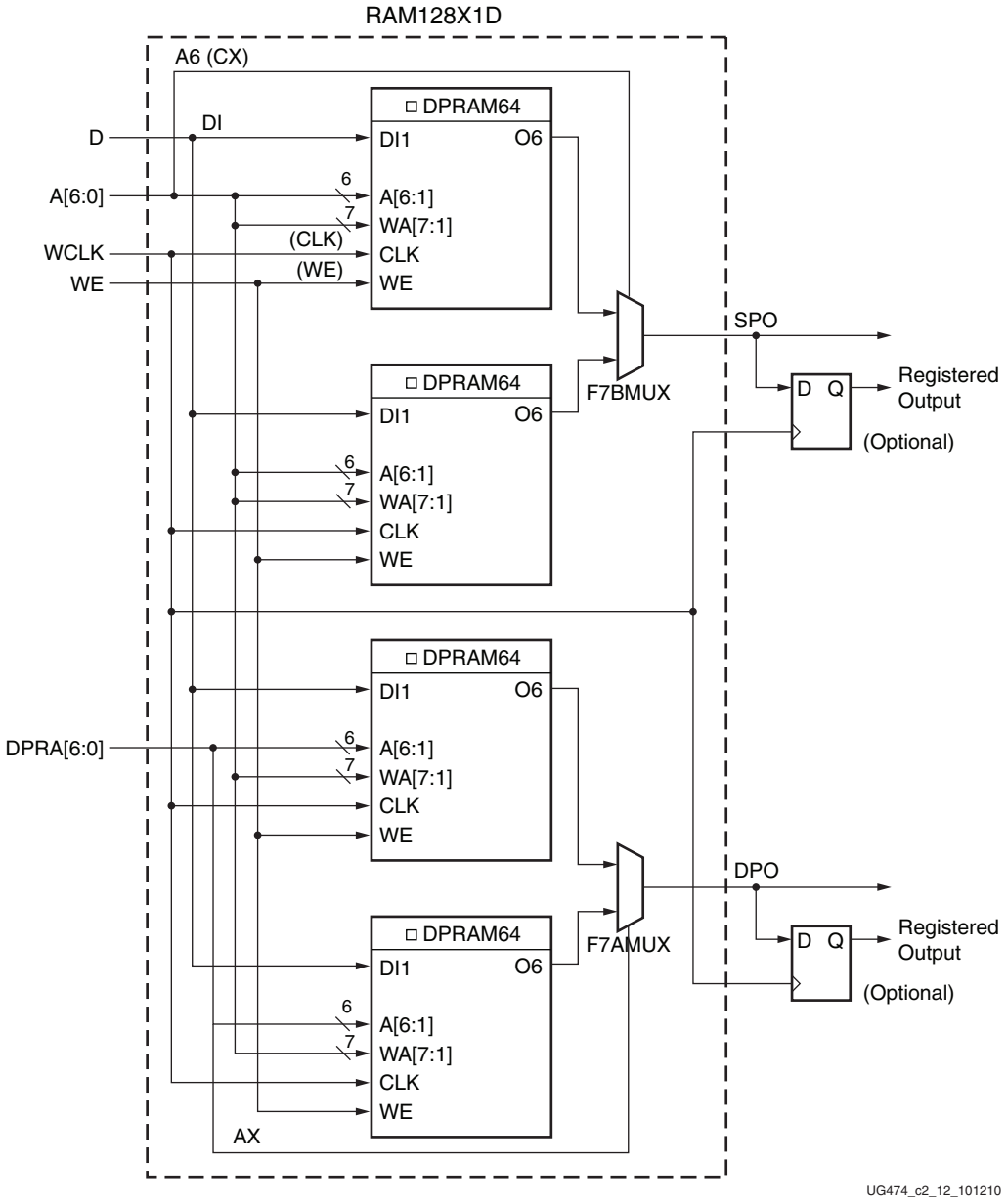

Figure 2-13: **128 X 1 Dual Port Distributed RAM (RAM128X1D)**

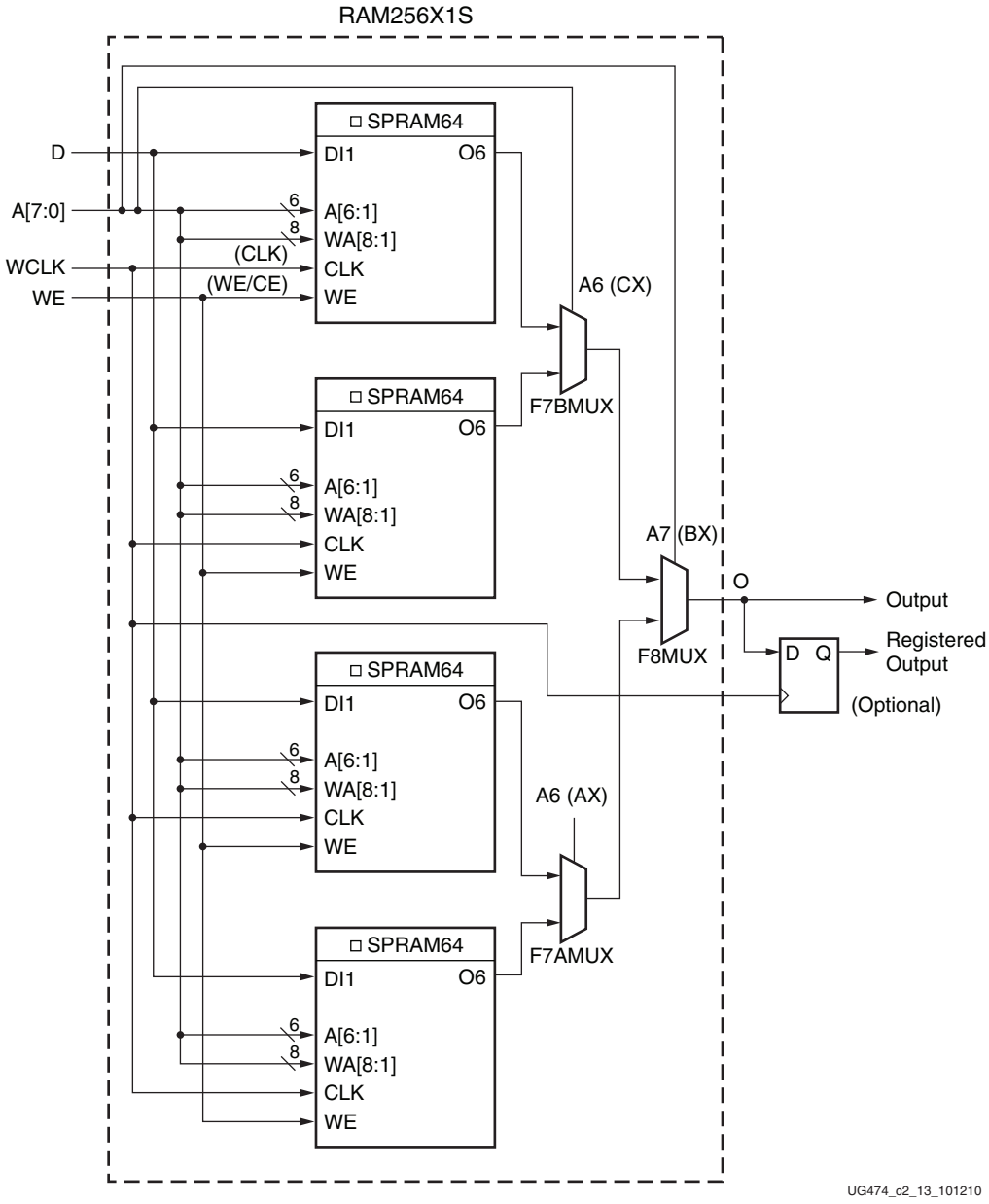

Figure 2-14: **256 X 1 Single Port Distributed RAM (RAM256X1S)**

Distributed RAM configurations greater than the provided examples require more than one SLICEM. There are no direct connections between slices to form larger distributed RAM configurations within a CLB or between slices.

Distributed RAM Data Flow

Synchronous Write Operation

The synchronous write operation is a single clock-edge operation with an active-High write-enable (WE) feature. When WE is High, the input (D) is loaded into the memory location at address A.

Asynchronous Read Operation

The output is determined by the address A for the single-port mode output SPO of dual-port mode, or address DPRA for the DPO output of dual-port mode. Each time a new address is applied to the address pins, the data value in the memory location of that address is available on the output after the time delay to access the LUT. This operation is asynchronous and independent of the clock signal.

Distributed RAM Summary

Here is a summary of distributed RAM features:

- Single-port and dual-port modes are available in SLICEMs
- A write operation requires one clock edge
- Read operations are asynchronous (Q output)
- The data input has a setup-to-clock timing specification

Read Only Memory (ROM)

Each function generator in both SLICEMs and SLICELs can implement a 64 x 1-bit ROM. Three configurations are available: ROM64X1, ROM128X1, and ROM256X1. ROM contents are loaded at each device configuration. Table 2-4 shows the number of LUTs occupied by each ROM configuration size.

Table 2-4: **ROM Configuration**

ROM	Number of LUTs
64 x 1	1
128 x 1	2
256 x 1	4

Shift Registers (Available in SLICEM Only)

A SLICEM function generator can also be configured as a 32-bit shift register without using the flip-flops available in a slice. Used in this way, each LUT can delay serial data from 1 to 32 clock cycles. The shiftin D (DI1 LUT pin) and shiftout Q31 (MC31 LUT pin) lines cascade LUTs to form larger shift registers. The four LUTs in a SLICEM are thus cascaded to produce delays up to 128 clock cycles. It is also possible to combine shift registers across more than one SLICEM. There are no direct connections between slices to form longer shift registers, nor is the MC31 output at LUT B/C/D available. The resulting programmable delays can be used to balance the timing of data pipelines.

Applications for shift registers include:

- Delay or latency compensation
- Synchronous FIFO and content addressable memory (CAM)

Shift register functions include:

- Write operation
  - Synchronous with a clock input (CLK) and an optional clock enable (CE)
- Fixed read access to Q31
- Dynamic read access
  - Performed through the 5-bit address bus, A[4:0]
    - The LSB of the LUT address is unused and the software automatically ties it to a logic High.
  - Any of the 32 bits can be read out asynchronously (at the O6 LUT outputs, referred to as Q on the primitive) by varying the address
  - This capability is useful in creating smaller shift registers (less than 32 bits).
    - For example, when building a 13-bit shift register, set the address to the 13th bit.
  - A storage element or flip-flop is available to implement a synchronous read.
    - The clock-to-out of the flip-flop determines the overall delay and improves performance.
    - However, one additional cycle of clock latency is added.
- Set or reset of the shift register is not supported.

Figure 2-15 is a logic block diagram of a 32-bit shift register.

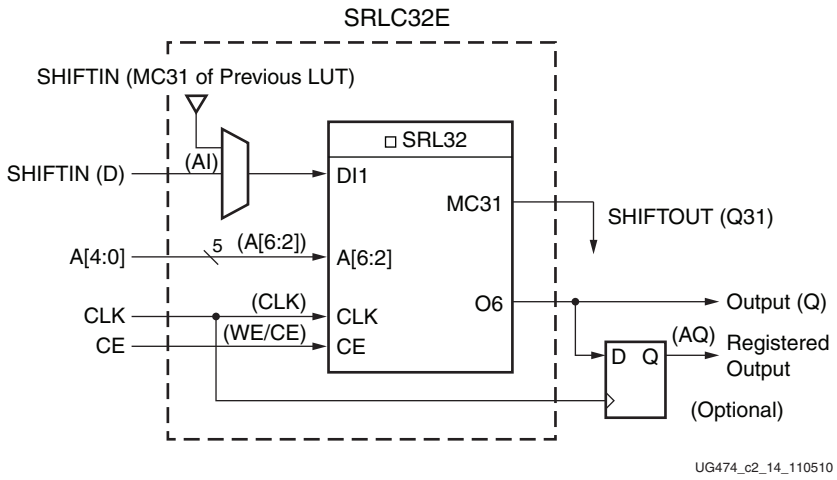

Figure 2-15: **32-Bit Shift Register Configuration**

Figure 2-16 illustrates an example shift register configuration occupying one function generator.

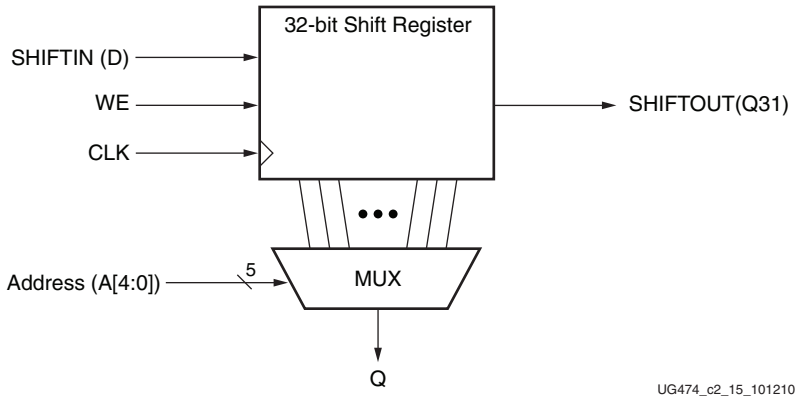

Figure 2-16: **Representation of a Shift Register**

Figure 2-17 shows two 16-bit shift registers. The example shown can be implemented in a single LUT. For more information on the SRLC32E and SRL16E primitives, see UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*.

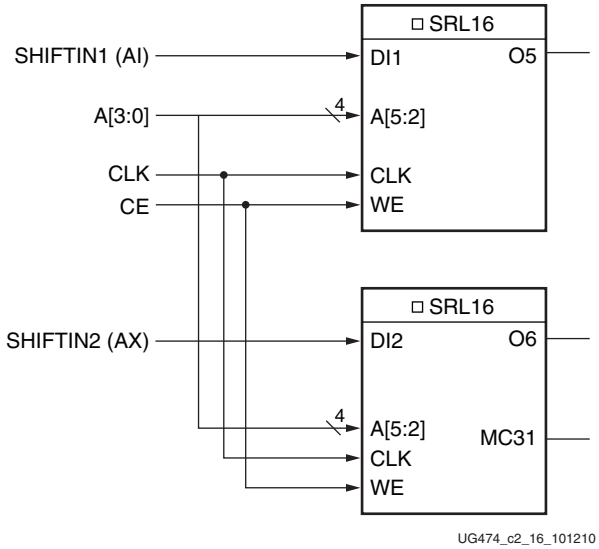

Figure 2-17: **Dual 16-Bit Shift Register Configuration**

As mentioned earlier, the MC31 output and a dedicated connection between shift registers allows connecting the last bit of one shift register to the first bit of the next, without using the LUT O6 output. Longer shift registers can be built with dynamic access to any bit in the chain. The shift register chaining and the F7AMUX, F7BMUX, and F8MUX multiplexers allow up to a 128-bit shift register with addressable access to be implemented in one SLICEM. Figure 2-18 through Figure 2-20 illustrate various example shift register configurations that can occupy one SLICEM.

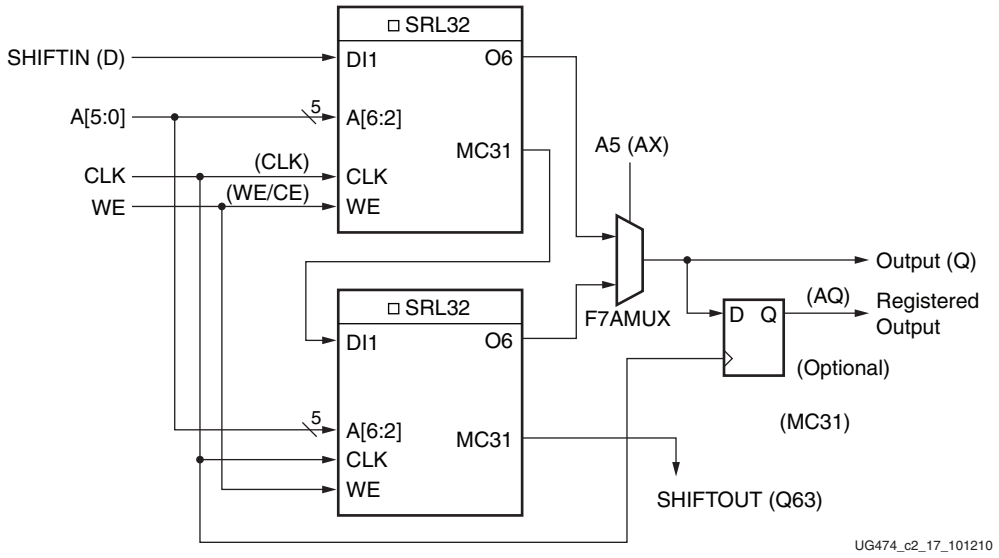

Figure 2-18: **64-Bit Shift Register Configuration**

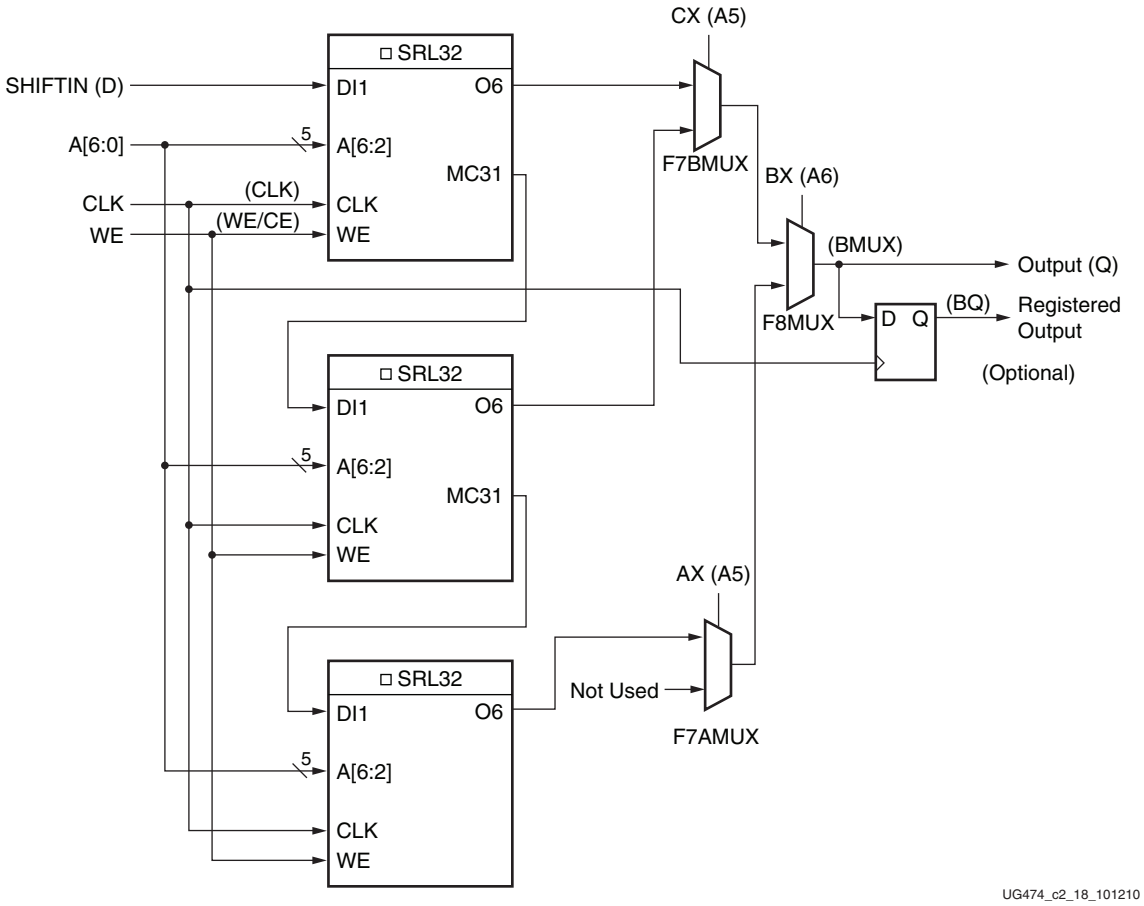

Figure 2-19: **96-Bit Shift Register Configuration**

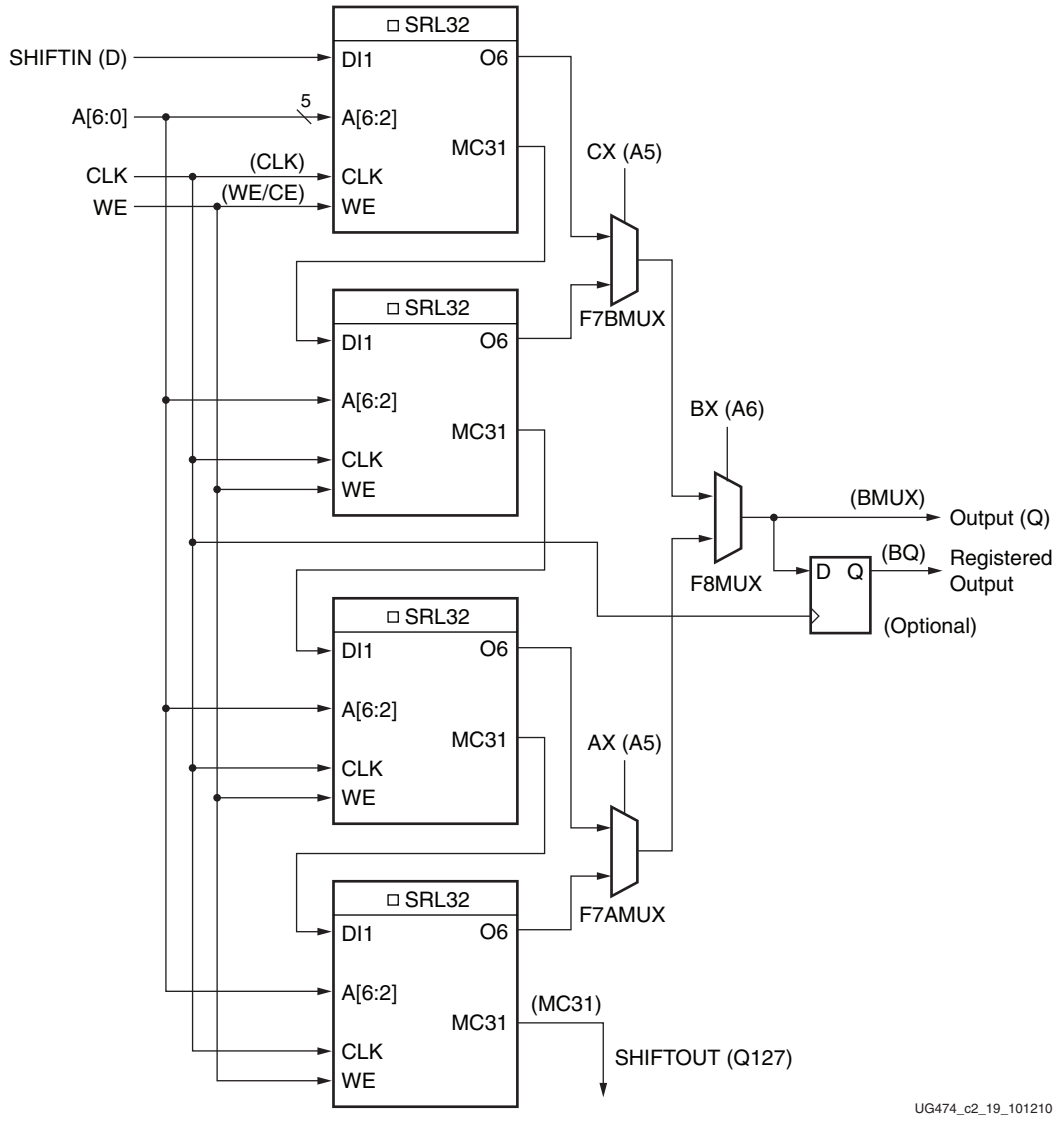

Figure 2-20: **128-Bit Shift Register Configuration**

It is possible to create shift registers longer than 128 bits across more than one SLICEM. However, there are no direct connections between slices to form these shift registers.

Shift Register Data Flow

Shift Operation

Figure 5-7 shows a timing diagram of the shift operation. The shift operation has these characteristics:

- Operates on a single clock edge
- Enabled by an active-High clock enable
- The input (D) is loaded into the first bit of the shift register
- Each bit is also shifted to the next highest bit position

- In a cascadable shift register configuration, the last bit is shifted out on the MC31 output
- The bit selected by the 5-bit address port (A[4:0]) appears on the Q output

Dynamic Read Operation

In a dynamic read operation:

- The Q output is determined by the 5-bit address
- Each time a new address is applied to the 5-input address pins, the new bit position value is available on the Q output after the time delay to access the LUT
- This operation is asynchronous and independent of the clock and clock-enable signals

Static Read Operation

In a static read operation:

- If the 5-bit address is fixed, the Q output always uses the same bit position
- This mode implements any shift-register length from 1 to 32 bits in one LUT
- The shift register length is (N + 1), where N is the input address (0–31)
- The Q output changes synchronously with each shift operation
- The previous bit is shifted to the next position and appears on the Q output

Shift Register Summary

Here is a summary of the shift registers:

- A shift operation requires one clock edge
- Dynamic-length read operations to the Q output of the LUT are asynchronous
- Static-length read operations to the Q output of the LUT are synchronous
- The data input has a setup-to-clock timing specification
- In a cascadable configuration, the Q31 output always contains the last bit value
- The Q31 output changes synchronously after each shift operation

Multiplexers

Function generators and associated multiplexers in 7 series FPGAs can implement these functions:

- 4:1 multiplexers using one LUT
  - Four 4:1 MUXes per slice
- 8:1 multiplexers using two LUTs
  - Two 8:1 MUXes per slice
- 16:1 multiplexers using four LUTs
  - One 16:1 MUX per slice

These wide input multiplexers are implemented in one level of logic (or LUT) using the dedicated F7AMUX, F7BMUX, and F8MUX multiplexers. These multiplexers allow LUT combinations of up to four LUTs in a slice. The wide multiplexers can also be used to create general-purpose functions of up to 13 inputs in two LUTs or 27 inputs in four LUTs (one slice). Although the multiplexer outputs are combinatorial, they can be registered in the CLB storage elements.

Designing Large Multiplexers

4:1 Multiplexer

Each LUT can be configured into a 4:1 MUX. The 4:1 MUX can be implemented with a flip-flop in the same slice. Up to four 4:1 MUXes can be implemented in a slice, as shown in Figure 2-21.

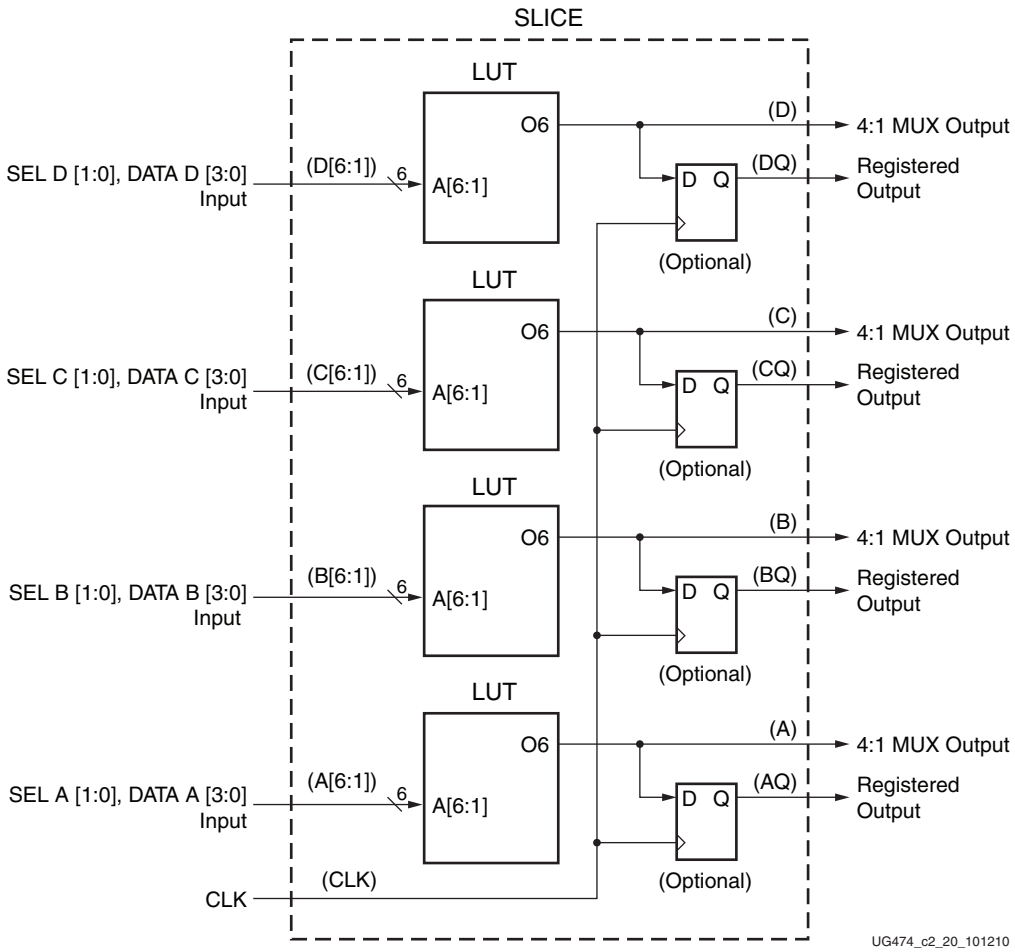

Figure 2-21: **Four 4:1 Multiplexers in a Slice**

8:1 Multiplexer

Each slice has an F7AMUX and an F7BMUX. These two MUXes combine the output of two LUTs to form a combinatorial function up to 13 inputs (or an 8:1 MUX). Up to two 8:1 MUXes can be implemented in a slice, as shown in Figure 2-22.

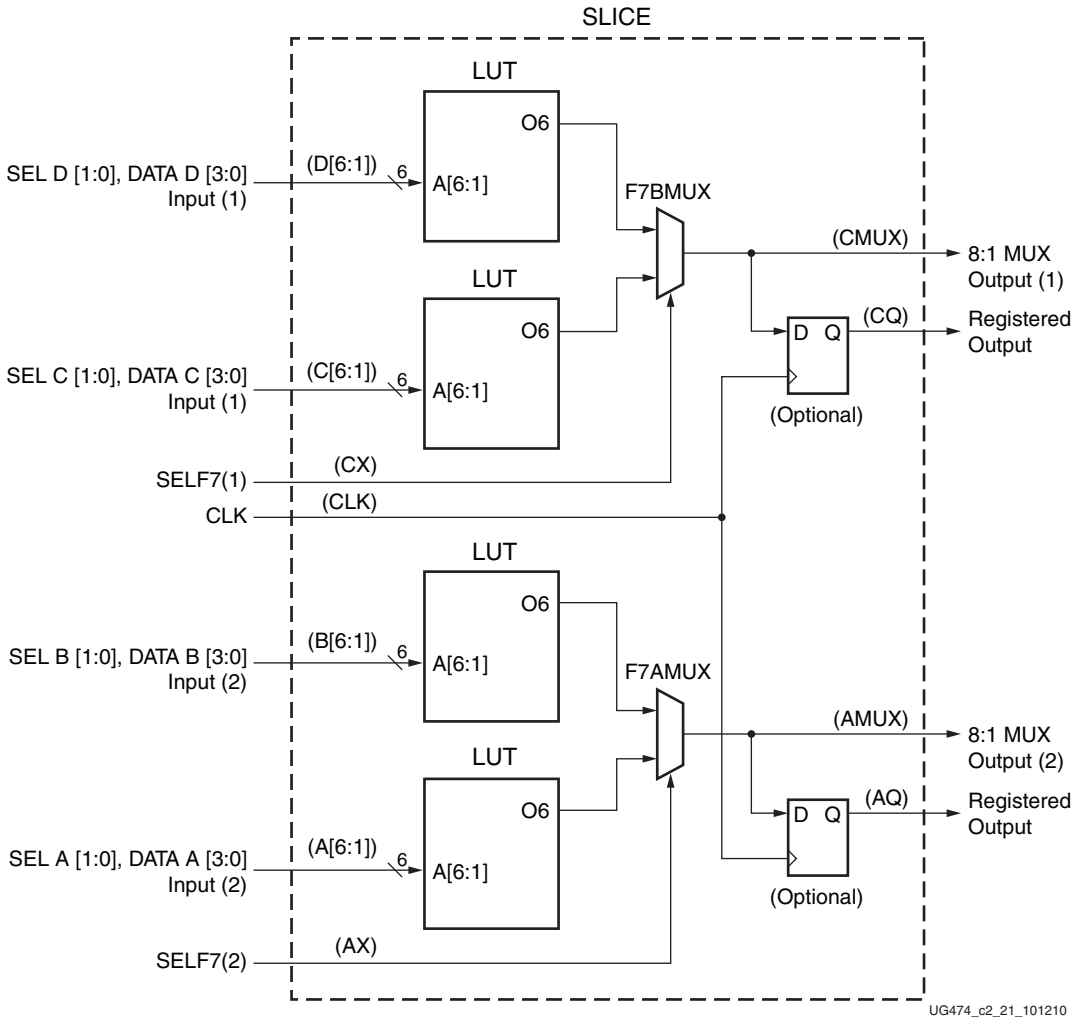

Figure 2-22: **Two 8:1 Multiplexers in a Slice**

16:1 Multiplexer

Each slice has an F8MUX. F8MUX combines the outputs of F7AMUX and F7BMUX to form a combinatorial function up to 27 inputs (or a 16:1 MUX). Only one 16:1 MUX can be implemented in a slice, as shown in Figure 2-23.

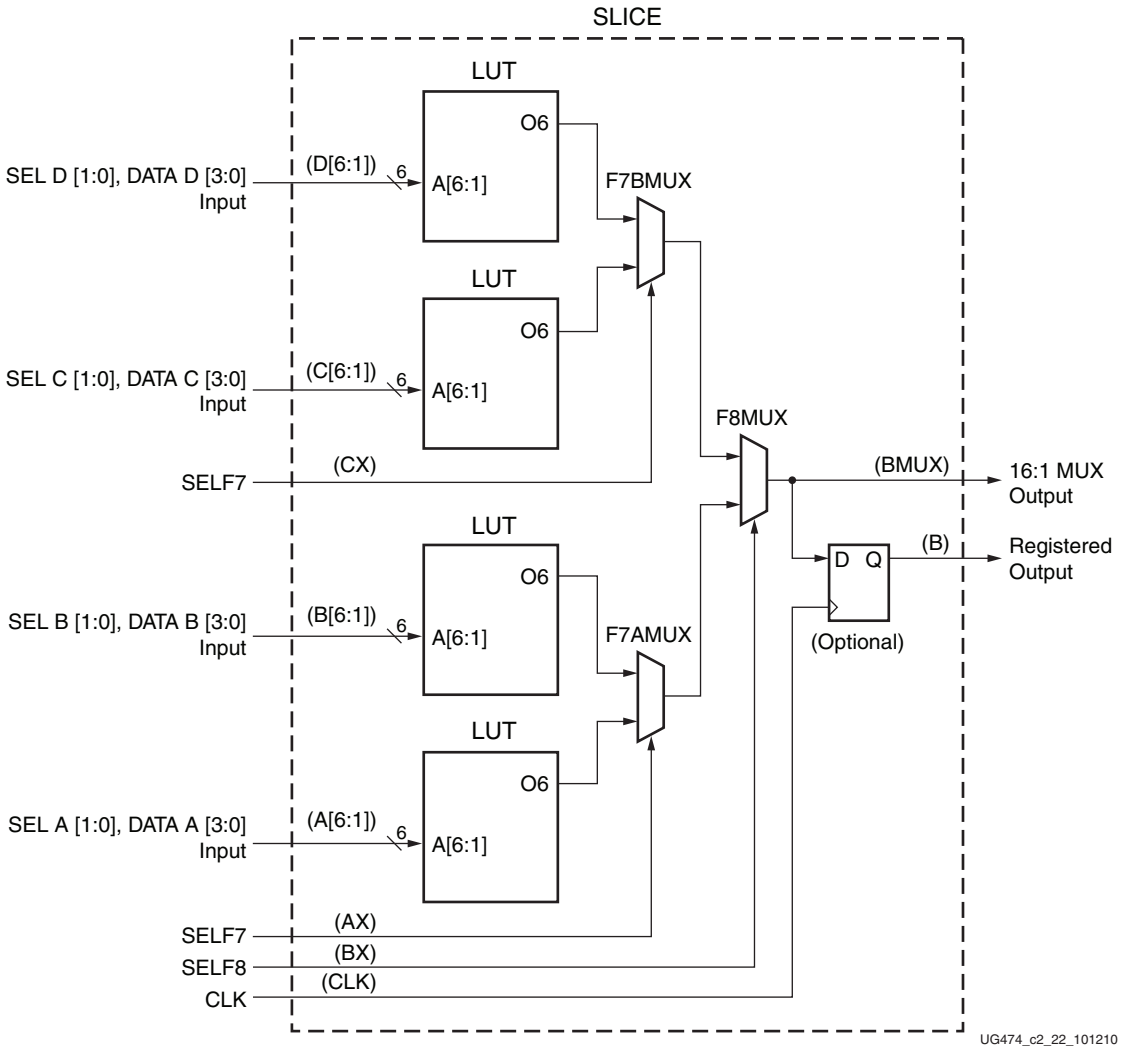

Figure 2-23: **16:1 Multiplexer in a Slice**

It is possible to create multiplexers wider than 16:1 across more than one SLICEM. However, there are no direct connections between slices to form these wide multiplexers.

Carry Logic

In addition to function generators, dedicated fast lookahead carry logic is provided to perform fast arithmetic addition and subtraction in a slice. A 7 series FPGA CLB has two separate carry chains, as shown in Figure 1-1. The carry chains are cascadable to form wider add/subtract logic, as shown in Figure 2-2.

The carry chain runs upward and has a height of four bits per slice. For each bit, there is a carry multiplexer (MUXCY) and a dedicated XOR gate for adding/subtracting the operands with a selected carry bits. The dedicated carry path and carry multiplexer (MUXCY) can also be used to cascade function generators for implementing wide logic functions.

Figure 2-24 illustrates the carry chain with associated logic in a slice.

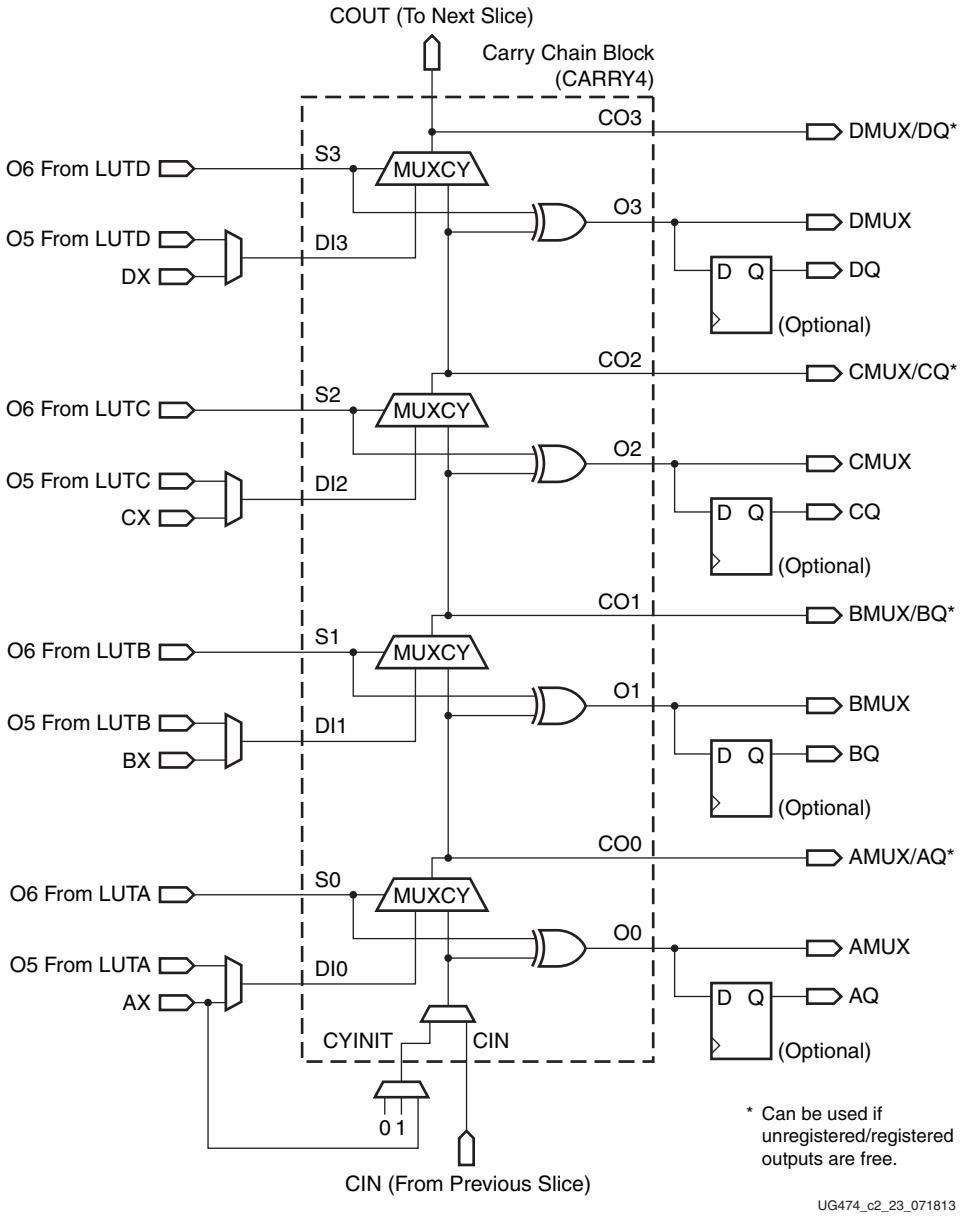

Figure 2-24: **Fast Carry Logic Path and Associated Elements**

The carry chains use carry lookahead logic along with the function generators. There are 10 independent inputs and 8 independent outputs:

- Inputs
  - S inputs - S0 to S3
    - The “propagate” signals of the carry lookahead logic
    - Sourced from the O6 output of a function generator
  - DI inputs - DI1 to DI4
    - The “generate” signals of the carry lookahead logic
    - Sourced from either the O5 output of a function generator
      - To create a multiplier
    - Or the BYPASS input (AX, BX, CX, or DX) of a slice
      - To create an adder/accumulator
  - CYINIT
    - CIN of the first bit in a carry chain
      - 0 for add
      - 1 for subtract
      - AX input for the dynamic first carry bit
  - CIN
    - Cascade slices to form a longer carry chain
- Outputs
  - O outputs - O0 to O3
    - Contain the sum of the addition/subtraction
  - CO outputs - CO0 to CO3
    - Compute the carry out for each bit
    - CO3 is connected to COUT output of a slice to form a longer carry chain by cascading multiple slices

The propagation delay for an adder increases linearly with the number of bits in the operand, as more carry chains are cascaded. The carry chain can be implemented with a storage element or a flip-flop in the same slice.

Carry logic cascading is limited only by the height of the column of slices. Carry logic cannot be cascaded across super logic regions (SLRs) in devices using stacked silicon interconnect (SSI) technology. See Devices Using Stacked Silicon Interconnect (SSI) Technology in Chapter 6.

Design Entry

CLB resources are used automatically and efficiently by FPGA synthesis tools without any special FPGA-specific coding required. Some HDL coding suggestions and techniques can help optimize designs for maximum efficiency.

Design Checklist

These guidelines are provided as a quick checklist of design suggestions for effective use of the 7 series CLBs:

- Resource Utilization
  - Use generic HDL code and allow the synthesis and mapping tools to choose the specific FPGA CLB resources.
  - Consider instantiation of specific resources only when necessary to meet density or performance requirements.
  - Compare the results to an estimated slice count to verify design efficiency.
  - If a design is running out of resources in the target device, examine which resource is the limiting factor and consider using alternative resources, such as moving registers to SRLs or distributed RAM, or distributed RAM to block RAM, or carry logic to DSP slices.
  - When migrating a design from another architecture, remove resource instantiation and any mapping and floorplanning constraints (refer to UG429, *7 Series FPGAs Migration Methodology Guide*).

Refer to www.xilinx.com for good HDL coding techniques for FPGAs, such as in UG687, *XST User Guide for Virtex-6, Spartan-6, and 7 Series Devices*.

- Pipelining
  - The designer should use sequential design techniques and pipelining to take advantage of abundant flip-flops for performance.
- Control Signals
  - Use control signals only as necessary.
  - Avoid using a routed global reset signal and minimize use of local resets to maximize opportunity to use FPGA resources.
  - Use active-High control signals.
  - Avoid having both set and reset on the same flip-flop.
  - Avoid control signals on small shift registers and storage arrays in order to use LUTs instead of flip-flops, to maximize utilization and minimize power.

- Software Options
  - To improve performance automatically, use timing constraints and trade off longer implementation run times through software options.

Using the CLB Resources

Xilinx recommends using generic HDL code and allowing the tools to infer the usage of CLB resources. Using IP solutions designed for 7 series FPGAs can help make full use of the CLB resources. Although any feature in the CLB can be instantiated directly, including the LUTs, carry logic, and sequential elements, instantiation should be reserved primarily for specifying when resources outside the CLB should be used, such as the DSP slice. Instantiation might also be needed if the synthesis tool does not infer a desired special CLB resource, such as the wide multiplexers, distributed RAM, or SRL feature.

Primitives

This section provides an overview of the most commonly used CLB primitives. For instantiation examples and for information on other CLB primitives, see UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*.

Multiplexer Primitives

The multiplexer primitives provide direct instantiation of the dedicated multiplexers in each slice, allowing wider multiplexers to be built. Table 3-1 describes the two primitives.

Table 3-1: **Multiplexer Primitives**

Primitive	Inputs	Resource	Output Function
MUXF7	LUT outputs (4:1 multiplexer)	F7AMUX or F7BMUX	8:1 multiplexer
MUXF8	F7AMUX and F7BMUX outputs (8:1 multiplexer)	F8MUX	16:1 multiplexer

The port signals are the same for both multiplexer primitives. Figure 3-1 shows MUXF7.

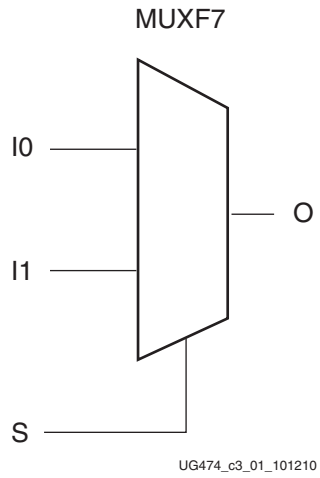

Figure 3-1: **MUXF7 Primitive**

Port Signals

Data In - I0, I1

The data input provides the data to be selected by the select signal (S).

Control In - S

The select input signal determines the data input signal to be connected to the output O. Logic 0 selects the I0 input, while logic 1 selects the I1 input.

Data Out - O

The data output O provides the data value (one bit) selected by the control inputs.

Carry Chain Primitive

The CARRY4 primitive instantiates the fast carry logic available in each slice. This primitive works in conjunction with LUTs to build adders and multipliers. Figure 3-2 shows the CARRY4 primitive. Synthesis tools generally infer this logic from arithmetic HDL code, automatically connecting this function properly.

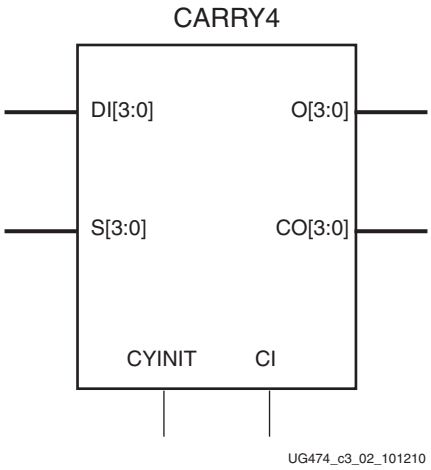

Figure 3-2: **CARRY4 Primitive**

Port Signals

Sum Outputs - O[3:0]

The sum outputs provide the final result of the addition/subtraction. They connect to the slice AMUX/BMUX/CMUX/DMUX outputs.

Carry Outputs - CO[3:0]

The carry outputs provide the carry out for each bit. CO[3] is equivalent to COUT. A longer carry chain can be created if CO[3] is connected through COUT to the CI input of another CARRY4 primitive, and dedicated routing connects the carry chain up a column of slices. The carry outputs also optionally connect to the slice AMUX/BMUX/CMUX/DMUX outputs.

Carry In - CI

The carry in input, also called CIN, is used to cascade slices to form longer carry chains.

Data Inputs - DI[3:0]

The data inputs are used as "generate" signals to the carry lookahead logic. The "generate" signals are sourced from LUT outputs.

Select Inputs - S[3:0]

The select inputs are used as "propagate" signals to the carry lookahead logic. The "propagate" signals are sourced from LUT outputs.

Carry Initialize - CYINIT

The carry initialize input is used to select the first bit in a carry chain. The value for this pin is either 0 (for add), 1 (for subtract), or AX input (for the dynamic first carry bit).

SLICEM Distributed RAM Primitives

Eight primitives are shown in Table 3-2. Three primitives are single-port RAM, three primitives are dual-port RAM, and two primitives are quad-port RAM.

Table 3-2: **Single-Port, Dual-Port, and Quad-Port Distributed RAM**

Primitive	RAM Size	Type	Address Inputs
RAM32X1S	32-bit	Single-port	A[4:0] (read/write)
RAM32M	32-bit	Quad-port	ADDRA[4:0] (read) ADDRB[4:0] (read) ADDRC[4:0] (read) ADDRD[4:0] (read/write)
RAM64X1S	64-bit	Single-port	A[5:0] (read/write)
RAM64X1D	64-bit	Dual-port	A[5:0] (read/write) DPRA[5:0] (read)
RAM64M	64-bit	Quad-port	ADDRA[5:0] (read) ADDRB[5:0] (read) ADDRC[5:0] (read) ADDRD[5:0] (read/write)
RAM128X1S	128-bit	Single-port	A[6:0] (read/write)
RAM128X1D	128-bit	Dual-port	A[6:0], (read/write) DPRA[6:0] (read)
RAM256X1S	256-bit	Single-port	A[7:0] (read/write)

The input and output data are one bit wide (with the exception of the quad-port RAM).

Figure 3-3 shows generic single-port, dual-port, and quad-port distributed RAM primitives. The A, ADDR, and DPRA signals are address buses.

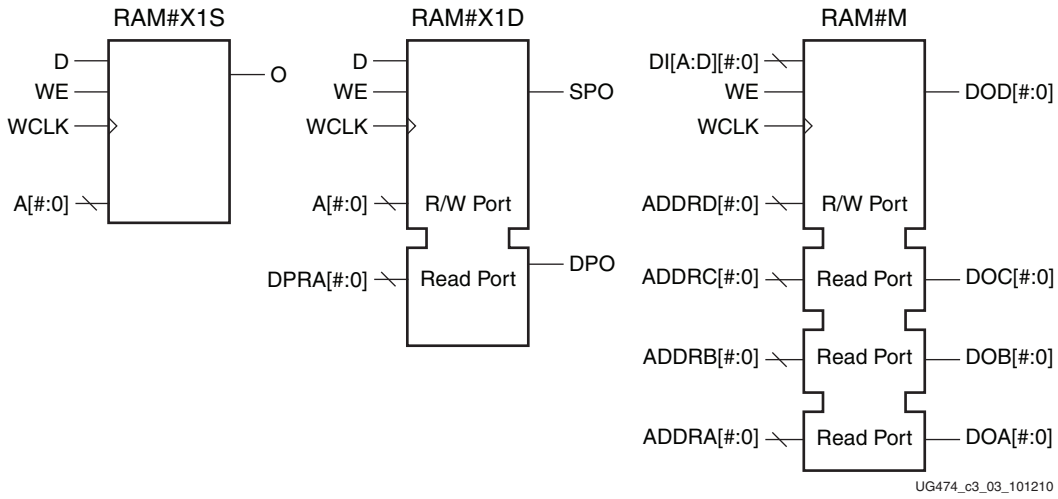

Figure 3-3: **Single-Port, Dual-Port, and Quad-Port Distributed RAM Primitives**

Instantiating several distributed RAM primitives can be used to implement wide memory blocks.

Port Signals

Each distributed RAM port operates independently of the other while reading the same set of memory cells.

Clock – WCLK

The clock is used for the synchronous write. The data and the address input pins have setup times referenced to the WCLK pin.

The clock pin (WCLK) has an inversion option at the slice level. The clock signal can be active at the negative edge of the clock or the positive edge of the clock without requiring other logic resources. The default is at the positive clock edge.

Enable – WE/WED

The enable pin affects the write functionality of the port. An inactive write enable prevents any writing to memory cells. An active write enable causes the clock edge to write the data input signal to the memory location pointed to by the address inputs.

Address – A[#:0], DPRA[#:0], and ADDRA[#:0] – ADDRD[#:0]

The address inputs A[#:0] (for single-port and dual-port), DPRA[#:0] (for dual-port), and ADDRA[#:0] – ADDRD[#:0] (for quad-port) select the memory cells for read or write. The width of the port determines the required address inputs. Some of the address inputs are not buses in VHDL or Verilog instantiations. Table 3-2 summarizes the function of each address pin.

Data In – D, DID[#:0]

The data input D (for single-port and dual-port) and DID[#:0] (for quad-port) provide the new data value to be written into the RAM.

Data Out – O, SPO, DPO and DOA[#:0] – DOD[#:0]

The data out O (single-port or SPO), DPO (dual-port), and DOA[#:0] – DOD[#:0] (quad-port) reflects the contents of the memory cells referenced by the address inputs. Following an active write clock edge, the data out (O, SPO, or DOD[#:0]) reflects the newly written data.

SLICEM SRL Shift Register Primitive

One primitive is available for the 32-bit shift register (SRLC32E), which uses one LUT in a SLICEM. Figure 3-4 shows the 32-bit shift register primitive.

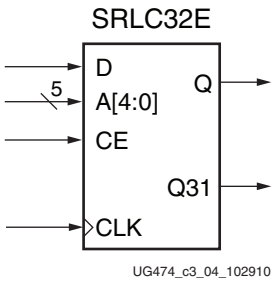

Figure 3-4: **32-Bit Shift Register**

Port Signals

Clock – CLK

Either the rising edge or the falling edge of the clock is used for the synchronous shift operation. The data and clock enable input pins have setup times referenced to the chosen edge of CLK.

The clock pin (CLK) has an inversion option at the slice level. The clock signal can be active at the negative or positive edge without requiring other logic resources. The default is positive clock edge.

Data In – D

The data input provides new data (one bit) to be shifted into the shift register.

Clock Enable – CE

The clock enable pin affects shift functionality. An inactive clock enable pin does not shift data into the shift register and does not write new data. Activating the clock enable allows the data in (D) to be written to the first location and all data to be shifted by one location. When available, new data appears on output pins (Q) and the cascadable output pin (Q31).

Address – A[4:0]

The address input selects the bit (range 0 to 31) to be read. The nth bit is available on the output pin (Q). Address inputs have no effect on the cascadable output pin (Q31). Q31 is always the last bit of the shift register (bit 31).

Data Out – Q

The data output Q provides the data value (1 bit) selected by the address inputs.

Data Out – Q31

The data output Q31 provides the last bit value of the 32-bit shift register. New data becomes available after each shift-in operation.

Flip-Flop Primitives

There are several primitives for the CLB storage elements, including both flip-flops and latches, with different combinations of control signals available. The FDRE primitive is shown in Figure 3-5 for an example. For more information on the flip-flop and latch primitives, see UG953, *Vivado Design Suite 7 Series FPGA and Zynq-7000 All Programmable SoC Libraries Guide*.

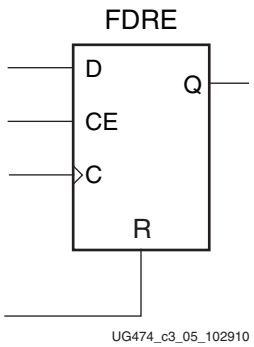

Figure 3-5: **FDRE Primitive**

Port Signals

Data In - D

The data input provides new data (one bit) to be clocked into the flip-flop.

Data Out - Q

Q is the one-bit registered data output from the flip-flop.

Clock - C

Either the rising edge or the falling edge of the clock is used to capture data and toggle the output. The data and clock enable input pins have setup times referenced to the chosen edge of the clock. The clock pin (C) has an inversion option at the slice level. The clock signal can be active at the negative or positive edge of the clock without requiring other logic resources. The default is the positive clock edge. All flip-flops in a slice must use the same clock and same clock polarity.

Clock Enable - CE

When CE is Low, clock transitions are ignored with respect to the D input. The set and reset inputs have priority over the clock enable.

Synchronous Reset - R

When R is High, all other inputs are overridden and the data output (Q) is driven Low on the active clock transition. This signal is available in the FDRE component. The FDRE flip-flop is also cleared by default on power-up.

Synchronous Set - S

When S is High, all other inputs are overridden and the data output (Q) is driven High on the active clock transition. This signal is available in the FDSE component. The FDSE flip-flop is also preset by default on power-up.

Asynchronous Clear - CLR

When CLR is High, all other inputs are overridden and the data output (Q) is driven Low. This signal is available in the FDCE component. The FDCE flip-flop is also cleared by default on power-up.

Asynchronous Preset - PRE

When PRE is High, all other inputs are overridden and the data output (Q) is driven High. This signal is available in the FDPE component. The FDPE flip-flop is also preset by default on power-up.

Note: Using both asynchronous clear and preset on the same flip-flop requires additional resources and timing paths.

XILINX®

Chapter 4

Applications

This chapter provides guidance regarding using the CLB resources as parts of larger functions. Trade-offs are described between alternative methods of implementing these functions.

Distributed RAM Applications

Distributed RAM provides a trade-off between using storage elements for very small arrays and block RAM for larger arrays. It is recommended to infer memory where possible to provide the greatest flexibility. Distributed RAM can also be targeted by instantiation or through the use of Xilinx LogiCORE™ IP.

In general, distributed RAM should be used for all memories that consist of 64 bits or less, unless there is a shortage of SLICEM or logic resources for the target device. Distributed RAM is more efficient in terms of resources, performance, and power.

For depths greater than 64 bits but less than or equal to 128 bits, the decision on the best resource to use depends on these factors:

1. The availability of extra block RAMs. If not available, distributed RAM should be used.
2. The latency requirements. If asynchronous read capability is needed, distributed RAMs must be used.
3. The data width. Widths greater than 16 bits should use block RAM, if available.
4. The necessary performance requirements. Registered distributed RAMs generally have shorter clock-to-out timing and fewer placement restrictions than block RAMs.

Shift Register Applications

Synchronous Shift Registers

The shift-register primitive does not use the register available in the same slice. To implement a fully synchronous read and write shift register, output pin Q must be connected to a flip-flop. Both the shift register and the flip-flop share the same clock, as shown in Figure 4-1.

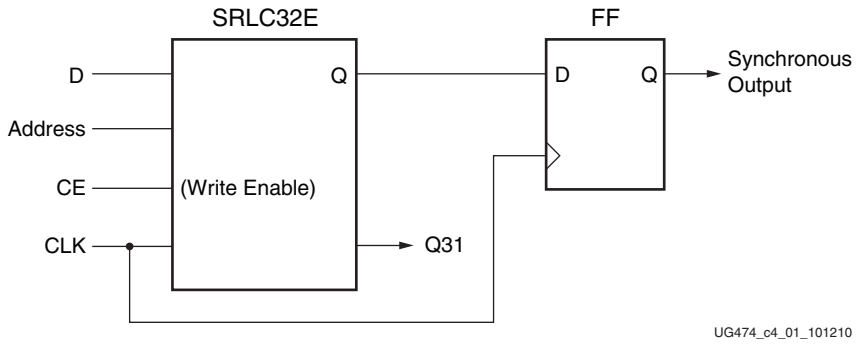

Figure 4-1: **Synchronous Shift Register**

This configuration provides a better timing solution and simplifies the design. Because the flip-flop must be considered to be the last register in the shift-register chain, the static or dynamic address should point to the desired length minus one. If needed, the cascadable output can also be registered in a flip-flop.

Static-Length Shift Registers

The cascadable 32-bit shift register implements any static length mode shift register without the dedicated multiplexers (F7AMUX, F7BMUX, and F8MUX). Figure 4-2 illustrates a 72-bit shift register. Only the last SRLC32E primitive needs to have its address inputs tied to `0b00111`. Alternatively, shift register length can be limited to 71 bits (address tied to `0b00110`) and a flip-flop can be used as the last register. (In an SRLC32E primitive, the shift register length is the address input + 1).

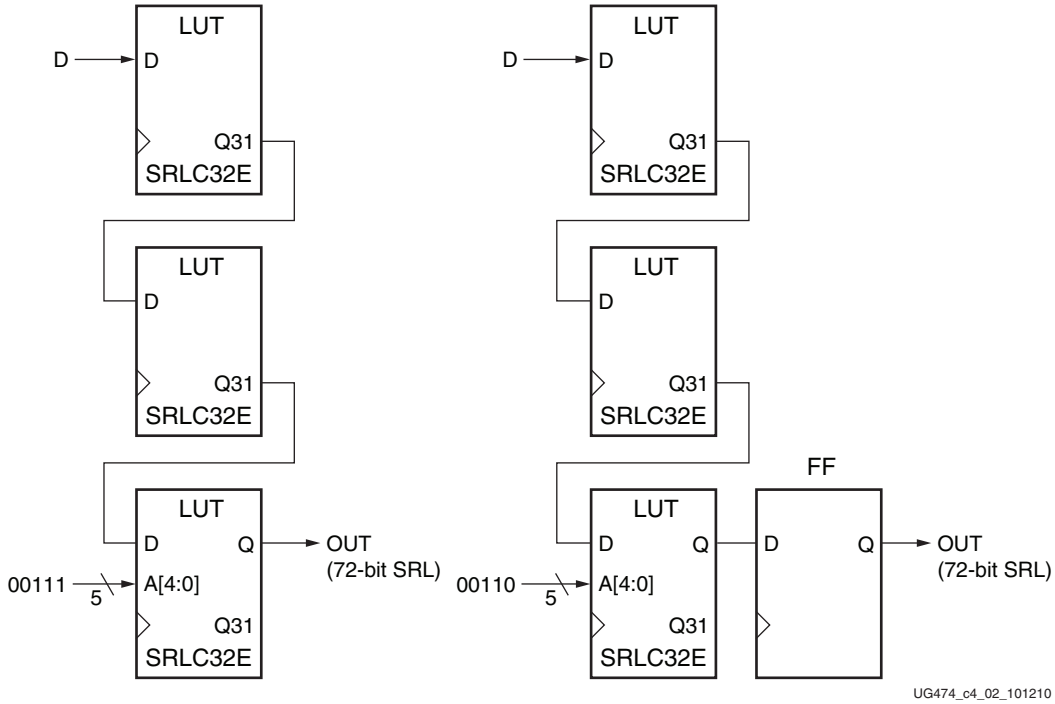

Figure 4-2: **Example Static-Length Shift Register**

Carry Logic Applications

Dedicated carry logic in the CLB improves the performance of arithmetic functions such as adders, counters, and comparators. Designs that include simple counters or adder/subtractors automatically infer the carry logic. More complex functions including multipliers can be implemented using the separate DSP48E1 slice.

The DSP48E1 slices in every device support many independent functions including multiply, multiply accumulate, multiply add, three-input add, barrel shift, wide-bus multiplexing, magnitude comparator, bitwise logic functions, pattern detection, and wide counter. The architecture also supports cascading multiple DSP48E1 slices to form wide math functions, DSP filters, and complex arithmetic. For more details on the DSP48E1 slice, see UG479, *7 Series DSP48E1 Slice User Guide*.

The choice between using CLB carry logic and the DSP48E1 slice depends on the application. A small arithmetic function can be faster and lower power using the CLB carry logic rather than using an entire DSP48E1 slice. When the DSP48E1 slices are fully utilized, the CLB slices and carry logic provide an efficient alternative.

Using Carry Logic

Carry logic can be inferred or instantiated. Using macros designed for the 7 series FPGA can provide the most flexibility and efficiency, especially for more complex functions.

Unimacros include adders, counters, comparators, multipliers, and multiplier/accumulators. The unimacros use the DSP48E1 slice.

The Xilinx IP includes a similar set of functions. When defining these functions, the user can specify whether the implementation should be in the CLB carry logic or in the DSP48E1 slice.

The carry logic runs vertically up every other column of slices (SLICEL and SLICEM). The Xilinx tools automatically place logic in a column when the carry logic is used. When floorplanning, carry logic based functions should always be specified as a group to avoid unnecessary breaks in the carry chain. Carry logic cannot be cascaded across super logic regions (SLRs) in devices using stacked silicon interconnect (SSI) technology.

XILINX®

Chapter 5

Timing

Although it is not necessary to understand the various timing parameters to implement most designs using Xilinx software, a thorough timing model can assist advanced users in analyzing critical paths or planning speed-sensitive designs, especially in the larger and more complex 7 series FPGAs.

The user is recommended to use the models in this chapter in conjunction with both the Xilinx® Timing Analyzer and the section on switching characteristics in the respective 7 series FPGAs data sheet. All pin names, parameter names, and paths are consistent with the Timing Analyzer reports. Most of the CLB timing parameters found in the data sheet section on switching characteristics are described in this chapter.

Five categories of functions are described in this chapter:

- CLB General Slice Timing Model and Parameters, page 58
- CLB Slice Multiplexer Timing Model and Parameters, page 60
- CLB Slice Carry-Chain Timing Model and Parameters, page 61
- CLB Slice Distributed RAM Timing Model and Parameters (Available in SLICEM Only), page 63
- CLB Slice SRL Shift Register Timing Model and Parameters (Available in SLICEM Only), page 66

For each category, three timing model sections are described:

- Functional element diagram - basic architectural schematic illustrating pins and connections
- Timing parameters - definitions of the respective 7 series FPGAs data sheet timing parameters
- Timing diagram - illustrates functional element timing parameters relative to each other

The general form of a timing parameter name is T subscripted with a combination of the starting pin and the ending pin. For example, T_{CECK} is the setup time from CE to the clock, and T_{CKCE} is the hold time from the clock to CE being removed. In some cases, a single parameter name can apply to multiple paths through the slice, and can have different values for each path.

CLB General Slice Timing Model and Parameters

A simplified 7 series FPGA slice is shown in Figure 5-1. Only the elements relevant to the timing paths described in this section are shown.

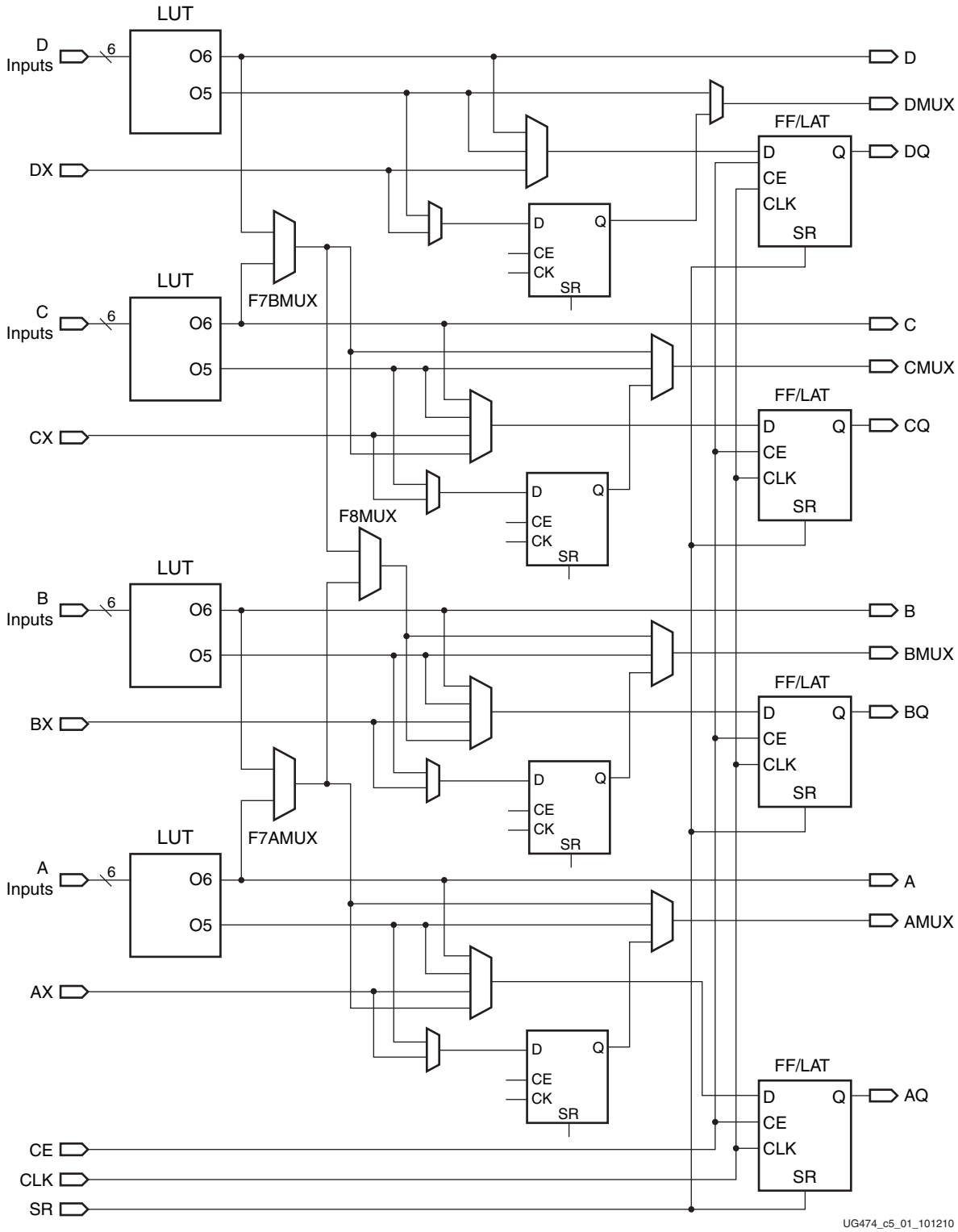

Figure 5-1: **Simplified 7 Series FPGA Slice**

General Timing Parameters

Table 5-1 shows the general slice timing parameters for a majority of the paths in Figure 5-1.

Table 5-1: **General Slice Timing Parameters**

Parameter	Function	Description
Combinatorial Delays		
T_{ILO}[1]	A/B/C/D inputs to A/B/C/D outputs	Propagation delay from the A/B/C/D LUT inputs of the slice, through the look-up tables (LUTs), to the A/B/C/D outputs of the slice.
T_{ITO}	A/B/C/D inputs through transparent latch to AQ/BQ/CQ/DQ outputs	Propagation delay from the A/B/C/D LUT inputs of the slice, through the LUTs to the AQ/BQ/CQ/DQ outputs of the slice sequential elements (configured as a latch).
Sequential Delays		
T_{CKO}	FF Clock (CLK) to AQ/BQ/CQ/DQ outputs	Time after the clock that data is stable at the AQ/BQ/CQ/DQ outputs of the slice sequential elements (configured as a flip-flop).
T_{CKLO}	Latch Clock (CLK) to AQ/BQ/CQ/DQ outputs	Time after the clock that data is stable at the AQ/BQ/CQ/DQ outputs of the slice sequential elements (configured as a latch).
T_{SHCKO}	FF Clock (CLK) to AMUX/BMUX/CMUX/DMUX outputs	Time after the clock that data is stable at the AMUX/BMUX/CMUX/DMUX outputs (through the slice flip-flops).
Setup and Hold Times for Slice Sequential Elements[2]		
T_{AS}/T_{AH}	A/B/C/D inputs	Time before/after the CLK that data from the A/B/C/D LUT inputs of the slice must be stable.
T_{DICK}/T_{CKDI}	AX/BX/CX/DX inputs	Time before/after the CLK that data from the AX/BX/CX/DX inputs of the slice must be stable.
T_{CECK}/T_{CKCE}	CE input	Time before/after the CLK that the CE input of the slice must be stable.
T_{SRCK}/T_{CKSR}	SR input	Time before/after the CLK that the SR (Set/Reset) input of the slice must be stable.
Set/Reset		
T_{RPW} (T_{SRMIN})	SR input	Minimum pulse width for the SR (Set/Reset).
T_{RQ}	SR input	Propagation delay for an asynchronous Set/Reset of the slice sequential elements.
Miscellaneous		
T_{CEO}	CE input	Propagation delay from CE enable to latch output at AQ/BQ/CQ/DQ outputs.
F_{TOG}	CLK input	Toggle Frequency – Maximum frequency that a slice flip-flop can be clocked. For export control.

Notes:

1. This parameter includes a LUT configured as two five-input functions.
2. T_{XXCK} = Setup Time (before clock edge), and T_{CKXX} = Hold Time (after clock edge).

General Timing Characteristics

Figure 5-2 illustrates the general timing characteristics of a 7 series FPGA slice.

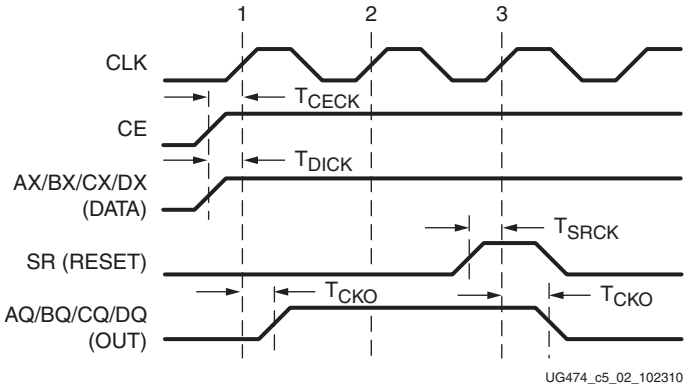

Figure 5-2: **General Slice Timing Characteristics**

- At time T_{CECK} before clock event (1), the clock-enable signal becomes valid-High at the CE input of the slice register.
- At time T_{DICK} before clock event (1), data from either AX, BX, CX, or DX inputs become valid-High at the D input of the slice register and is reflected on either the AQ, BQ, CQ, or DQ pin at time T_{CKO} after clock event (1).
- At time T_{SRCK} before clock event (3), the SR signal (configured as synchronous reset) becomes valid-High, resetting the slice register. This is reflected on the AQ, BQ, CQ, or DQ pin at time T_{CKO} after clock event (3).

CLB Slice Multiplexer Timing Model and Parameters

Figure 2-22, page 41 and Figure 2-23, page 42 illustrate the wide multiplexers in a 7 series FPGA slice.

Slice Multiplexer Timing Parameters

Table 5-2 shows the slice multiplexer timing parameters for a majority of the paths in Figure 2-22 and Figure 2-23. Note that many of these parameter names are also used for paths that do not include the wide multiplexers.

Table 5-2: **Slice Multiplexer Timing Parameters**

Parameter	Function	Description
Propagation Delays for Slice Using the Wide Multiplexers		
$T_{AXA}/T_{AXB}/T_{BXB}/T_{CXB}/T_{CXC}$	AX/BX/CX inputs to AMUX/BMUX/CMUX outputs	Propagation delay from the AX/BX/CX inputs of the slice through the select inputs of the multiplexers to the AMUX/BMUX/CMUX outputs of the slice.

Table 5-2: **Slice Multiplexer Timing Parameters *(Cont'd)***

Parameter	Function	Description
T_{OPAB}/ T_{OPBA}/T_{OPBB}/ T_{OPCB}/ T_{OPDB}/ T_{OPDC}	A/B/C/D inputs to AMUX/ BMUX/CMUX outputs	Propagation delay from the A/B/C/D LUT inputs of the slice through the multiplexers to the AMUX/ BMUX/CMUX outputs of the slice.
T_{ILO_2}	A/C inputs to AMUX/CMUX outputs	Propagation delay from the A LUT inputs of the slice through the F7AMUX multiplexer to the AMUX output of the slice, or propagation delay from the C LUT inputs of the slice through the F7BMUX multiplexer to the CMUX output of the slice.

CLB Slice Carry-Chain Timing Model and Parameters

Figure 2-24, page 43 illustrates a carry chain in a 7 series FPGA slice.

Slice Carry-Chain Timing Parameters

Table 5-3 shows the slice carry-chain timing parameters for a majority of the paths in Figure 2-24.

Table 5-3: **Slice Carry-Chain Timing Parameters**

Parameter	Function	Description
Propagation Delays for Slice Configured as a Carry Chain		
$T_{AXA}/T_{AXB}/T_{AXC}/T_{AXD}$/ $T_{BXB}/T_{BXC}/T_{BXD}$/ T_{CXC}/T_{CXD}/ T_{DXD}	AX/BX/CX/DX inputs to AMUX/BMUX/ CMUX/ DMUX outputs	Propagation delay from the AX/BX/CX/DX inputs of the slice through the carry logic to the AMUX/BMUX/ CMUX/DMUX outputs of the slice.
$T_{AXCY}/T_{BXCY}/T_{CXCY}/T_{DXCY}$	AX/BX/CX/DX inputs to COUT output	Propagation delay from the AX/BX/CX/DX inputs of the slice to the COUT output of the slice.
T_{BYP}	CIN input to COUT output	Propagation delay from the CIN input of the slice to the COUT output of the slice.
$T_{OPCYA}/T_{OPCYB}/T_{OPCYC}$/ T_{OPCYD}	A/B/C/D inputs to COUT output	Propagation delay from the A/B/C/D LUT inputs of the slice to the COUT output of the slice.

Table 5-3: **Slice Carry-Chain Timing Parameters *(Cont'd)***

Parameter	Function	Description
$T_{CINA}/T_{CINB}/T_{CINC}/T_{CIND}$	CIN input to AMUX/ BMUX/ CMUX/DMUX outputs	Propagation delay from the CIN input of the slice to AMUX/BMUX/CMUX/ DMUX outputs of the slice using XOR (sum).
Setup and Hold Times for a Slice Configured as a Carry Chain(1)		
T_{CINCK}/T_{CKCIN}	CIN input	Time before/after the CLK that the CIN input of the slice must be stable.

Notes:

1. T_{XXCK} = Setup Time (before clock edge), and T_{CKXX} = Hold Time (after clock edge).

Slice Carry-Chain Timing Characteristics

Figure 5-3 illustrates the timing characteristics of a slice carry chain implemented in a 7 series FPGA slice.

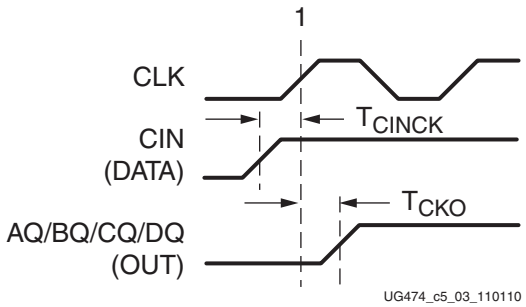

Figure 5-3: **Slice Carry-Chain Timing Characteristics**

Note relevant to Figure 5-3:

- At time T_{CINCK} before clock event 1, data from the CIN input becomes valid-High at the Data input of the slice register. This is reflected on any of the AQ/BQ/CQ/DQ pins at time T_{CKO} after clock event 1.

CLB Slice Distributed RAM Timing Model and Parameters (Available in SLICEM Only)

Figure 5-4 illustrates the details of distributed RAM implemented in a 7 series FPGA slice. Some elements of the slice are omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

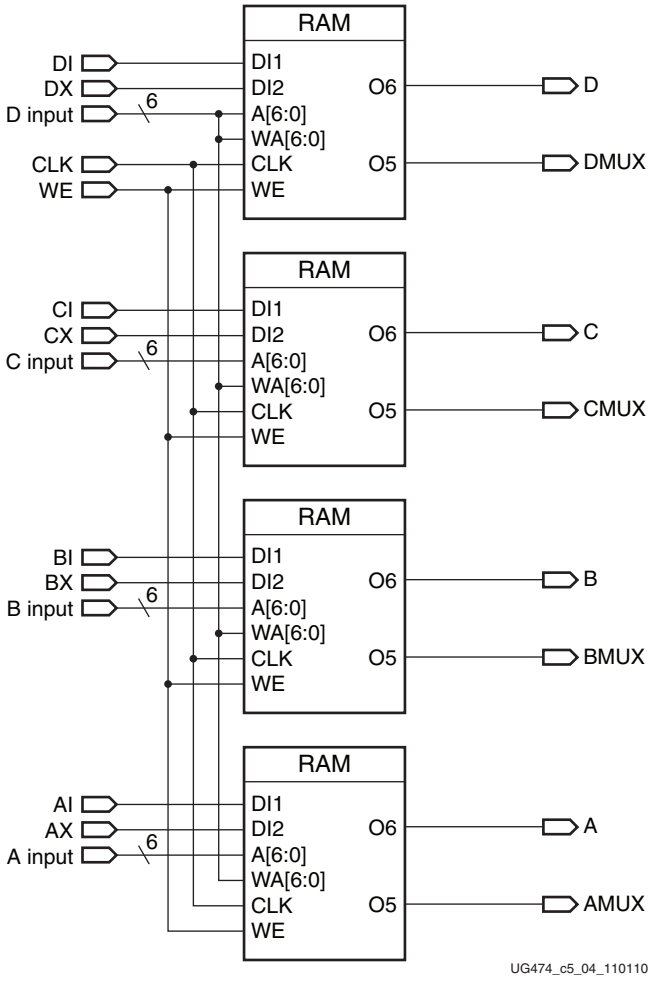

Figure 5-4: **Simplified 7 Series FPGA SLICEM Distributed RAM**

Distributed RAM Timing Parameters

Table 5-4 shows the timing parameters for the distributed RAM in SLICEM for a majority of the paths in Figure 5-4.

Table 5-4: **Distributed RAM Timing Parameters**

Parameter	Function	Description
Sequential Delays for a Slice LUT Configured as RAM (Distributed RAM)		
T_{SHCKO}	CLK to A/B/C/D outputs	Time after the CLK of a write operation that the data written to the distributed RAM is stable on the A/B/C/D outputs of the slice.
T_{SHCKO} (T_{SHCKO_1})	CLK to AMUX/BMUX/CMUX/DMUX outputs	Time after the CLK of a write operation that the data written to the distributed RAM is stable on the AMUX/BMUX/CMUX/DMUX outputs of the slice.
Setup and Hold Times for a Slice LUT Configured as RAM (Distributed RAM)(1)		
T_{DS}/T_{DH} (T_{DS_LRAM}/T_{DH_LRAM})	AI/BI/CI/DI configured as data inputs (DI1)(2)	Time before/after the clock that data must be stable at the AI/BI/CI/DI inputs of the slice (configured as RAM).
T_{AS}/T_{AH} (T_{AS_LRAM}/T_{AH_LRAM})	A/B/C/D address inputs	Time before/after the clock that address signals must be stable at the A/B/C/D LUT inputs of the slice (configured as RAM).
T_{WS}/T_{WH} (T_{WS_LRAM}/T_{WH_LRAM})	WE input	Time before/after the clock that the write enable signal must be stable at the WE input of the slice (configured as RAM).
Clock CLK		
T_{MPW} (T_{MPW_LRAM})		Minimum clock pulse width for distributed RAM.
T_{MCP}		Minimum clock period to meet address write cycle time.

Notes:

1. T_{XS} = Setup Time (before clock edge), and T_{XH} = Hold Time (after clock edge).
2. Parameter includes AX/BX/CX/DX configured as a data input (DI2).

Distributed RAM Timing Characteristics

The timing characteristics of a distributed RAM implemented in a 7 series FPGA slice (LUT configured as RAM) are shown in Figure 5-5.

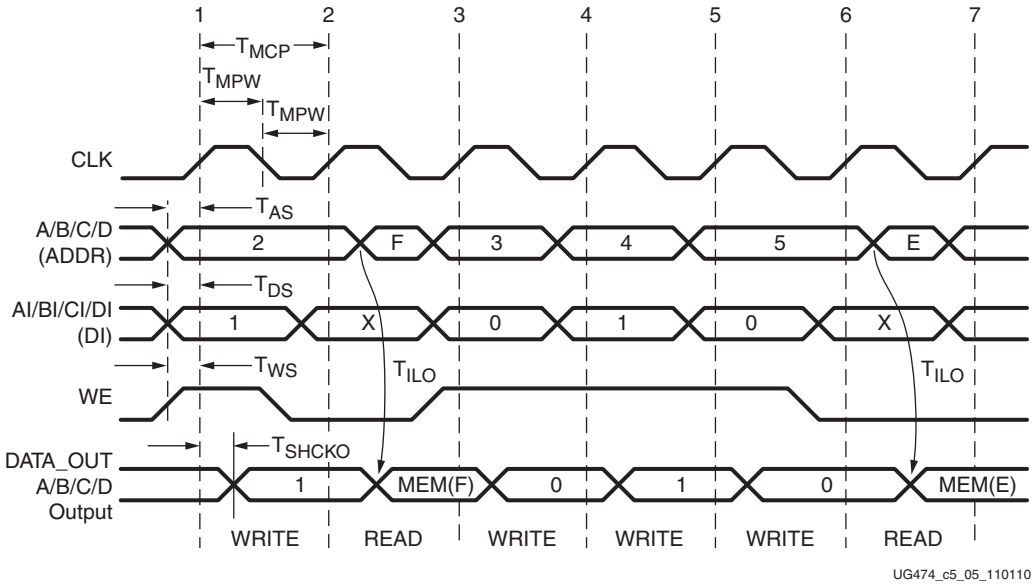

Figure 5-5: **Slice Distributed RAM Timing Characteristics**

Clock Event 1: Write Operation

During a Write operation, the contents of the memory at the address on the ADDR inputs are changed. The data written to this memory location is reflected on the A/B/C/D outputs synchronously.

- At time T_{WS} before clock event 1, the write-enable signal (WE) becomes valid-High, enabling the RAM for a Write operation.
- At time T_{AS} before clock event 1, the address (2) becomes valid at the A/B/C/D inputs of the RAM.
- At time T_{DS} before clock event 1, the DATA becomes valid (1) at the DI input of the RAM and is reflected on the A/B/C/D outputs at time T_{SHCKO} after clock event 1. This is also applicable to the AMUX, BMUX, CMUX, DMUX, and COUT outputs at time T_{SHCKO} after clock event 1.

Clock Event 2: Read Operation

All Read operations are asynchronous in distributed RAM. As long as WE is Low, the address bus can be asserted at any time. The contents of the RAM on the address bus are reflected on the A/B/C/D outputs after a delay of length T_{ILO} (propagation delay through a LUT). The address (F) is asserted after clock event 2, and the contents of the RAM at address (F) are reflected at the output after a delay of length T_{ILO}.

CLB Slice SRL Shift Register Timing Model and Parameters (Available in SLICEM Only)

Figure 5-6 illustrates shift register implementation in a 7 series FPGA slice. Some elements of the slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

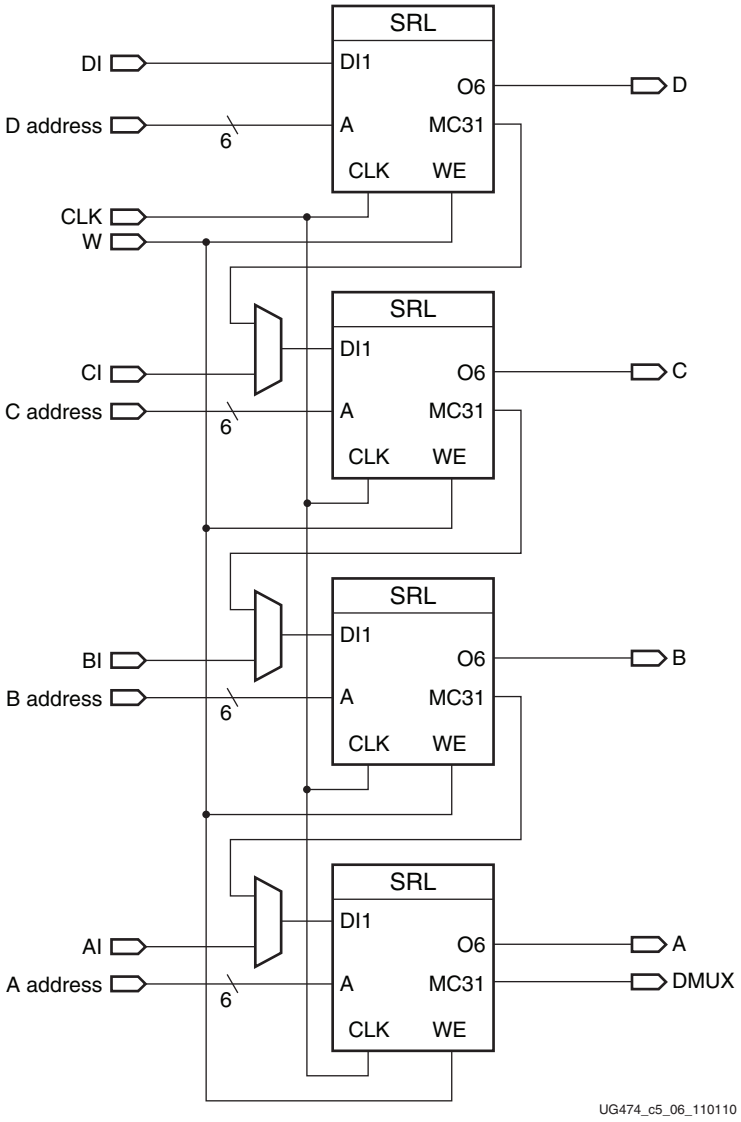

Figure 5-6: **Simplified 7 Series FPGA Slice SRL**

Slice SRL Timing Parameters

Table 5-5 shows the SLICEM SRL timing parameters for a majority of the paths in Figure 5-6.

Table 5-5: **Slice SRL Timing Parameters**

Parameter	Function	Description
Sequential Delays for a Slice LUT Configured as an SRL		
T_{REG}(1)	CLK to A/B/C/D outputs	Time after the CLK of a write operation that the data written to the SRL is stable on the A/B/C/D outputs of the slice.
T_{REG} (T_{REG_MUX})(1)	CLK to AMUX - DMUX outputs	Time after the CLK of a write operation that the data written to the SRL is stable on the AMUX/BMUX/CMUX/DMUX outputs of the slice.
T_{REG} (T_{REG_M31})	CLK to DMUX output via MC31 output	Time after the CLK of a write operation that the data written to the SRL is stable on the DMUX output via MC31 output.
Setup and Hold Times for a Slice LUT Configured as an SRL(2)		
T_{CECK}/T_{CKCE} ($T_{CECK\_SHFREG}/T_{CKCESHFREG}$)	CE input (CE)	Time before/after the clock that the clock-enable signal must be stable at the CE input of the slice LUT (configured as an SRL).
T_{WS}/T_{WH} ($T_{WS\_SHFREG}/T_{WH\_SHFREG}$)	CE input (WE)	Time before/after the clock that the write-enable signal must be stable at the WE input of the slice LUT (configured as an SRL).
T_{DS}/T_{DH}(3) ($T_{DS\_SHFREG}/T_{DH\_SHFREG}$)	AI/BI/CI/DI configured as data inputs (DI)	Time before/after the clock that the data must be stable at the AI/BI/CI/DI inputs of the slice (configured as an SRL).
Clock CLK		
T_{MPW}		Minimum clock pulse width for distributed RAM.

Notes:

1. This parameter includes a LUT configured as a two-bit shift register.
2. T_{XS} = Setup Time (before clock edge), and T_{XH} = Hold Time (after clock edge).
3. Parameter includes AX/BX/CX/DX configured as a data input (DI2) or two bits with a common shift.

Slice SRL Timing Characteristics

Figure 5-7 illustrates the timing characteristics of a shift register implemented in a 7 series FPGA slice (a LUT configured as an SRL).

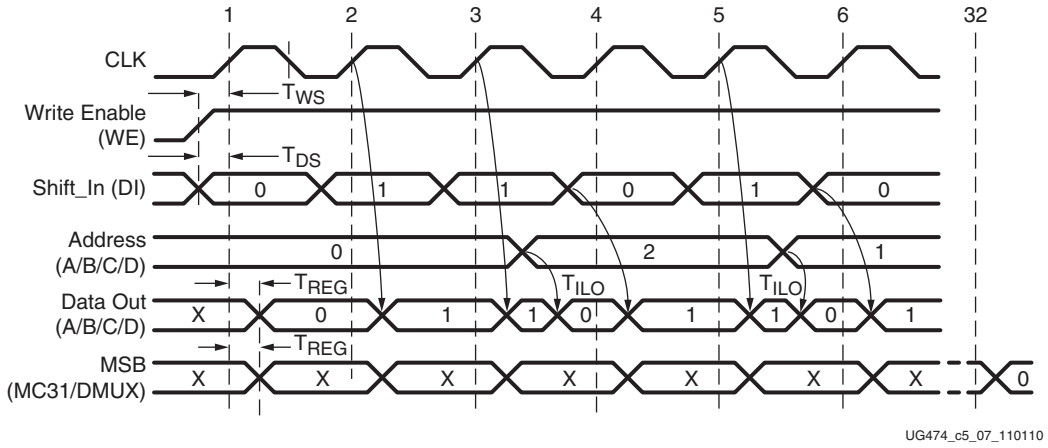

Figure 5-7: **Slice SRL Timing Characteristics**

Clock Event 1: Shift In

During a write (Shift In) operation, the single-bit content of the register at the address on the A/B/C/D inputs is changed, as data is shifted through the SRL. The data written to this register is reflected on the A/B/C/D outputs synchronously, if the address is unchanged during the clock event. If the A/B/C/D inputs are changed during a clock event, the value of the data at the addressable output (A/B/C/D outputs) is invalid.

- At time T_{WS} before clock event 1, the write-enable signal (WE) becomes valid-High, enabling the SRL for the Write operation that follows.
- At time T_{DS} before clock event 1, the data becomes valid (0) at the DI input of the SRL and is reflected on the A/B/C/D output after a delay of length T_{REG} after clock event 1. Because address 0 is specified at clock event 1, the data on the DI input is reflected at A/B/C/D output, because it is written to register 0.

Clock Event 2: Shift In

- At time T_{DS} before clock event 2, the data becomes valid (1) at the DI input of the SRL and is reflected on the A/B/C/D output after a delay of length T_{REG} after clock event 2. Because address 0 is still specified at clock event 2, the data on the DI input is reflected at the D output, because it is written to register 0.

Clock Event 3: Shift In/Addressable (Asynchronous) Read

All Read operations are asynchronous to the CLK signal. If the address is changed (between clock events), the contents of the register at that address are reflected at the addressable output (A/B/C/D outputs) after a delay of length T_{ILO} (propagation delay through a LUT).

- At time T_{DS} before clock event 3, the data becomes valid (1) at the DI input of the SRL and is reflected on the A/B/C/D output T_{REG} time after clock event 3.

- The address is changed (from 0 to 2). The value stored in register 2 at this time is a 0 (in this example, this was the first data shifted in), and it is reflected on the A/B/C/D output after a delay of length T_{ILO}.

Clock Event 32: MSB (Most Significant Bit) Changes

- At time T_{REG} after clock event 32, the first bit shifted into the SRL becomes valid (logical 0 in this case) on the DMUX output of the slice via the MC31 output of LUT A (SRL). This is also applicable to the AMUX, BMUX, CMUX, DMUX, and COUT outputs at time T_{REG} after clock event 1.

Advanced Topics

Using the Latch Function as Logic

Because the latch function is level-sensitive, it can be used as the equivalent of a logic gate. The primitives to specify this function are AND2B1L (a 2-input AND gate with one input inverted) and OR2L (a 2-input OR gate), as shown in Figure 6-1.

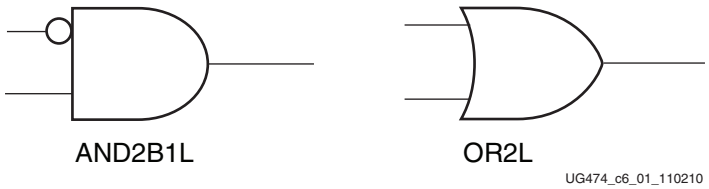

Figure 6-1: **AND2B1L and OR2L Components**

As shown in Figure 6-2, the data and SR inputs and Q output of the latch are used when the AND2B1L and OR2L primitives are instantiated, and the CK gate and CE gate enables are held active-High. The AND2B1L combines the latch data input (the inverted input on the gate, DI) with the asynchronous clear input (SRI). The OR2L combines the latch data input with an asynchronous preset. Generally, the latch data input comes from the output of a LUT within the same slice, extending the logic capability to another external input. Because there is only one SR input per slice, using more than one AND2B1L or OR2L per slice requires a shared common external input.

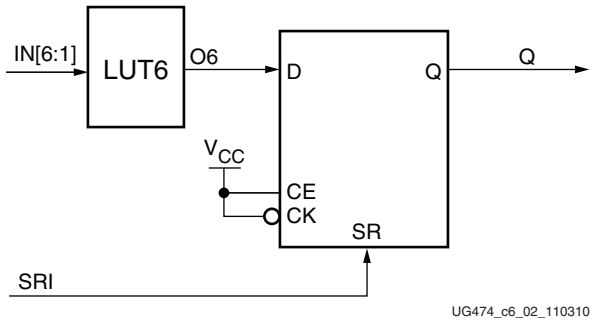

Figure 6-2: **Implementation of OR2L (Q = D or SRI)**

The device model shows these functions as AND2L and OR2L configurations of the storage element.

The AND2B1L and OR2L two-input gates save LUT resources and are initialized to a known state on power-up and on GSR assertion (Global Set/Reset). Using these primitives can reduce logic levels and increase logic density of the device by trading register/latch resources for logic. However, due to the static inputs required on the clock and clock enable inputs, specifying one or more AND2B1L or OR2L primitives can cause register packing and density issues in a slice disallowing the use of the remaining registers and latches.

Interconnect Resources

Interconnect is the programmable network of signal pathways between the inputs and outputs of functional elements within the FPGA, such as IOBs, CLBs, DSP slices, and block RAM. Interconnect, also called routing, is segmented for optimal connectivity. The Xilinx placement and routing tools exploit the rich interconnect array to deliver optimal system performance and the fastest compile times.

The 7 series CLBs are arranged in a regular array inside the FPGA. Each connects to a switch matrix for access to the general-routing resources, which run vertically and horizontally between the CLB rows and columns. A similar switch matrix connects other resources, such as the DSP slices and block RAM resources.

Most of the interconnect features are transparent to FPGA designers. Knowledge of the interconnect details can be used to guide design techniques but is not necessary for efficient FPGA design. Only selected types of interconnect are under user control. These include the clock routing resources, which are selected by using clock buffers and discussed in more detail in UG472, *7 Series FPGAs Clocking Resources User Guide*. Two global control signals, GSR and GTS, are selected by using the STARTUPE2 primitive.

Global Controls GSR and GTS

In addition to the general-purpose interconnect, 7 series FPGAs have two global logic control signals, as described in Table 6-1.

Table 6-1: **Global Logic Control Signals**

Global Control Input	Description
GSR	Global Set/Reset: When High, asynchronously places all registers and flip-flops in their initial state.
GTS	Global 3-State: When High, asynchronously forces all I/O pins to a high-impedance state (High-Z, 3-state).

If a common initialization signal is needed for every flip-flop in the design, use the GSR control in a design instead of a separate routed global reset signal to make CLB inputs available, which results in a smaller more efficient design. The GSR signal must always re-initialize every flip-flop. Using GSR and GTS does not use any general-purpose routing resources. The GSR signal is asserted automatically during the FPGA configuration process, guaranteeing that the FPGA starts up in a known state. At configuration, because flip-flops are enabled by the configuration clock and then clocked by a user clock, it is recommended to enable key flip-flops or key clock signals on the user clock after configuration.

STARTUPE2 Primitive

The GSR and GTS signal sources are defined and connected using the STARTUPE2 primitive. This primitive allows the user to define the source of these dedicated nets. GSR and GTS are always active during configuration, and connecting signals to them on the STARTUPE2 primitive defines how they are controlled after configuration. By default, they are disabled after configuration on a selected clock cycle of the start-up phase, enabling the flip-flops and I/Os in the device. The STARTUPE2 primitive also includes other signals used specifically during configuration. For more information, refer to the UG470, *7 Series FPGAs Configuration User Guide*.

Interconnect Optimization

Interconnect delays vary according to the specific implementation and loading in a design. The type of interconnect, distance required to travel in the device, and number of switch matrices to traverse factor into the total delay. Most timing issues are addressed by examining the block delays and determining the impact of using fewer levels or faster paths. If interconnect delays seem too long, increase implementation effort levels or iterations to improve performance along with making sure that the required timing is in the constraints file.

Nets with critical timing or that are heavily loaded can often be improved by replicating the source of the net. The dual 5-input LUT configuration of the slice simplifies the replication of logic in the same slice, which minimizes any additional loads on the inputs to the source function. Replicating logic in multiple slices gives the software more flexibility to place the sources independently.

Floorplanning is the process of specifying user-placement constraints. Floorplanning can be done either before or after automatic place and route, but automatic place and route is always recommended first before specifying user floorplanning. The Xilinx implementation tool provides a graphical view of placement. It helps the designer choose between RTL coding and synthesis and implementation, with extensive design exploration and analysis features.

Devices Using Stacked Silicon Interconnect (SSI) Technology

As noted in the DS180, *7 Series FPGAs Overview*, some of the Virtex-7 devices use stacked silicon interconnect (SSI) technology. These devices provide a unique additional type of interconnect resource called a super long line or SLL. These special routing resources can be treated like other interconnect resources as they are abundant (more than 10,000) and fast (about 1 ns). The SLL provides a connection between super logic regions (SLRs), as shown in Figure 6-3. Combining multiple SLRs effectively increases the height of the ASMBL™ architecture columns and increases the overall capacity of the device. The software can be allowed to take best advantage of this configuration, or users can apply floorplanning to control placement within and between SLRs. Note that carry logic cascading is limited to within an SLR and does not continue through the SLL connections. The same is true of other types of cascading including block RAM, DSP, and DCI. Boundaries are visible in floorplanning tools and reported in timing reports. The number of SLRs per Virtex-7 device is listed in the *7 Series FPGAs Overview*.

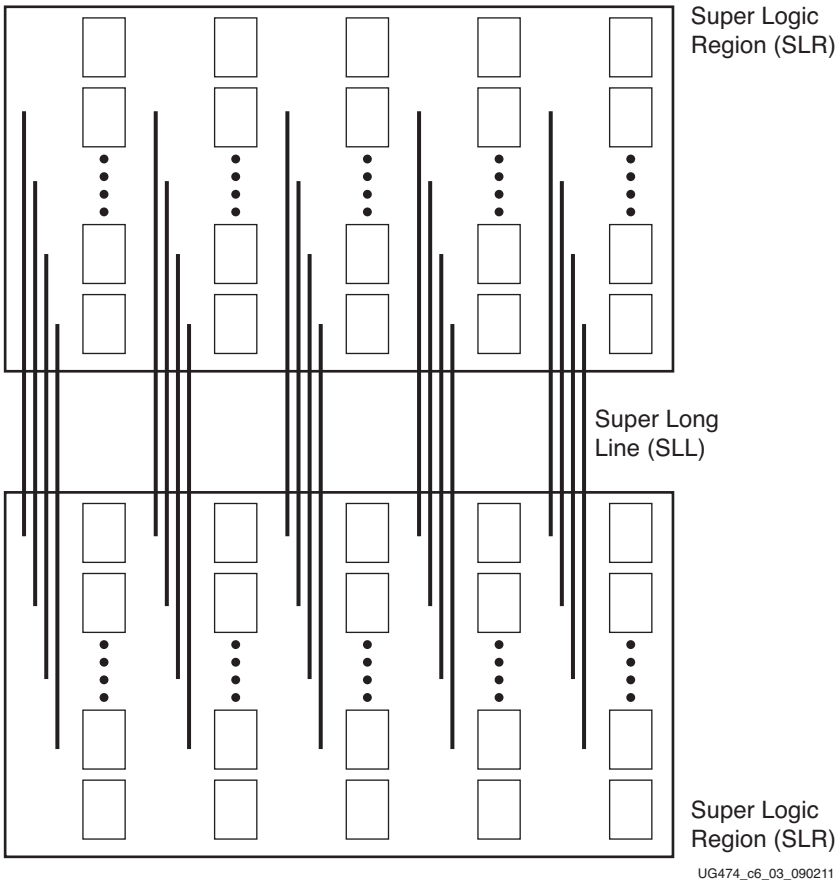

Figure 6-3: **Stacked Silicon Interconnect Technology**

Designs targeting devices using SSI technology only require one STARTUPE2 instantiation. One source for GSR and GTS signals propagates across the SLR boundaries to all elements of the device.

Optimization for devices using SSI technology can start with the same techniques as for any standard device, including using proper design techniques, timing constraints or options to automatically find the best implementation. Floorplanning for these devices can use the same graphical tools.

For more information on devices using SSI technology, see the following:

- DS180, *7 Series FPGAs Overview*
- Stacked Silicon Interconnect Technology page
  www.xilinx.com/technology/roadmap/stacked-silicon-interconnect
- WP380, *Xilinx Stacked Silicon Interconnect Technology Delivers Breakthrough FPGA Capacity, Bandwidth, and Power Efficiency*