Spartan-6 FPGA Configurable Logic Block

User Guide

UG384 (v1.1) February 23, 2010

$$ XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/24/09	1.0	Initial Xilinx release.
02/23/10	1.1	Clarified the Slice Description section. Added Figure 6. Clarifying edits to Storage Elements, Initialization, Distributed RAM and Memory (SLICEM only), and Fast Lookahead Carry Logic sections. Added Using the Latch Function as Logic and Interconnect Resources. Updated parameter names in Table 9 and Table 10 and Figure 35. Updated description T_{CINA} function and description Table 12.

Table of Contents

About This Guide

This guide serves as a technical reference describing the Spartan®-6 FPGA configurable logic blocks (CLBs). Usually, the logic synthesis software 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.

Additional Documentation

The following documents are also available for download at
http://www.xilinx.com/support/documentation/spartan-6.htm.

- Spartan-6 Family Overview

  This overview outlines the features and product selection of the Spartan-6 family.

- Spartan-6 FPGA Data Sheet: DC and Switching Characteristics

  This data sheet contains the DC and switching characteristic specifications for the Spartan-6 family.

- Spartan-6 FPGA Packaging and Pinout Specifications

  This specification includes the tables for device/package combinations and maximum I/Os, pin definitions, pinout tables, pinout diagrams, mechanical drawings, and thermal specifications.

- Spartan-6 FPGA Configuration User Guide

  This all-encompassing configuration guide includes chapters on configuration interfaces (serial and parallel), multi-bitstream management, bitstream encryption, boundary-scan and JTAG configuration, and reconfiguration techniques.

- Spartan-6 FPGA SelectIO Resources User Guide

  This guide describes the SelectIO™ resources available in all Spartan-6 devices.

- Spartan-6 FPGA Clocking Resources User Guide

  This guide describes the clocking resources available in all Spartan-6 devices, including the DCMs and PLLs.

- Spartan-6 FPGA Block RAM Resources User Guide

  This guide describes the Spartan-6 device block RAM capabilities.

- Spartan-6 FPGA GTP Transceivers User Guide

  This guide describes the GTP transceivers available in the Spartan-6 LXT FPGAs.

- Spartan-6 FPGA DSP48A1 Slice User Guide

  This guide describes the architecture of the DSP48A1 slice in Spartan-6 FPGAs and provides configuration examples.

- Spartan-6 FPGA Memory Controller User Guide

  This guide describes the Spartan-6 FPGA memory controller block, a dedicated embedded multi-port memory controller that greatly simplifies interfacing Spartan-6 FPGAs to the most popular memory standards.

- Spartan-6 FPGA PCB Design Guide

  This guide provides information on PCB design for Spartan-6 devices, with a focus on strategies for making design decisions at the PCB and interface level.

Additional Support Resources

To search the database of silicon and software questions and answers or to create a technical support case in WebCase, see the Xilinx website at:

http://www.xilinx.com/support.

Spartan-6 FPGA CLB

CLB Overview

The Configurable Logic Blocks (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). A CLB element contains a pair of slices. These two slices do not have direct connections to each other, and each slice is organized as a column. For each CLB, the slice in the bottom of the CLB is labeled as SLICE(0), and the slice in the top of the CLB is labeled as SLICE(1).
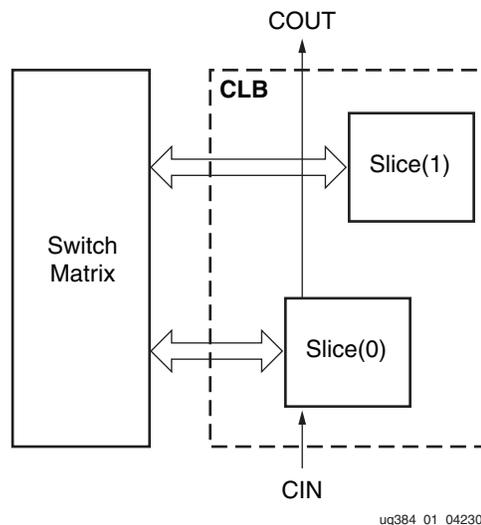
Figure 1: **Arrangement of Slices within the CLB**

The Xilinx tools designate slices with the following 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 shows four CLBs located in the bottom-left corner of the die.
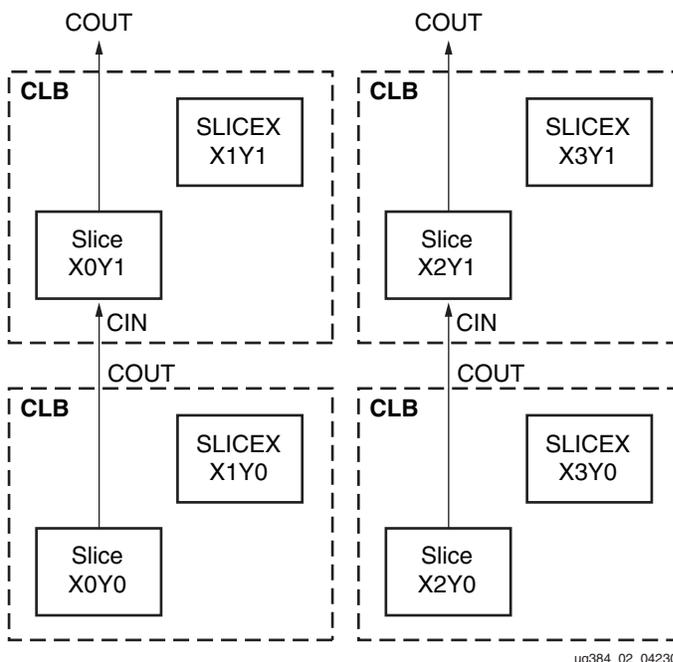
Figure 2: **Row and Column Relationship between CLBs and Slices**

Slice Description

Every slice contains four logic-function generators (or look-up tables, LUTs) and eight storage elements. These elements are used by all slices to provide logic and ROM functions (Table 1). SLICEX is the basic slice. Some slices, called SLICELs, also contain an arithmetic carry structure that can be concatenated vertically up through the slice column, and wide-function multiplexers. The SLICEMs contain the carry structure and multiplexers, and add the ability to use the LUTs as 64-bit distributed RAM and as variable-length shift registers (maximum 32-bit).

Table 1: **Slice Features**

Feature	SLICEX	SLICEL	SLICEM
6-Input LUTs	√	√	√
8 Flip-flops	√	√	√
Wide Multiplexers		√	√
Carry Logic		√	√
Distributed RAM			√
Shift Registers			√

Each column of CLBs contain two slice columns. One column is a SLICEX column, the other column alternates between SLICEL and SLICEMs. Thus, approximately 50% of the available slices are of type SLICEX, while 25% each are SLICEL and SLICEMs. The XC6SLX4 does not have SLICELs (Table 3).

SLICEM (shown in Figure 3) represents a superset of elements and connections found in all slices. SLICEL is shown in Figure 4. SLICEX is shown in Figure 5. All eight SR, CE, and CLK inputs are driven by common control inputs.

Figure 3: **Diagram of SLICEM**
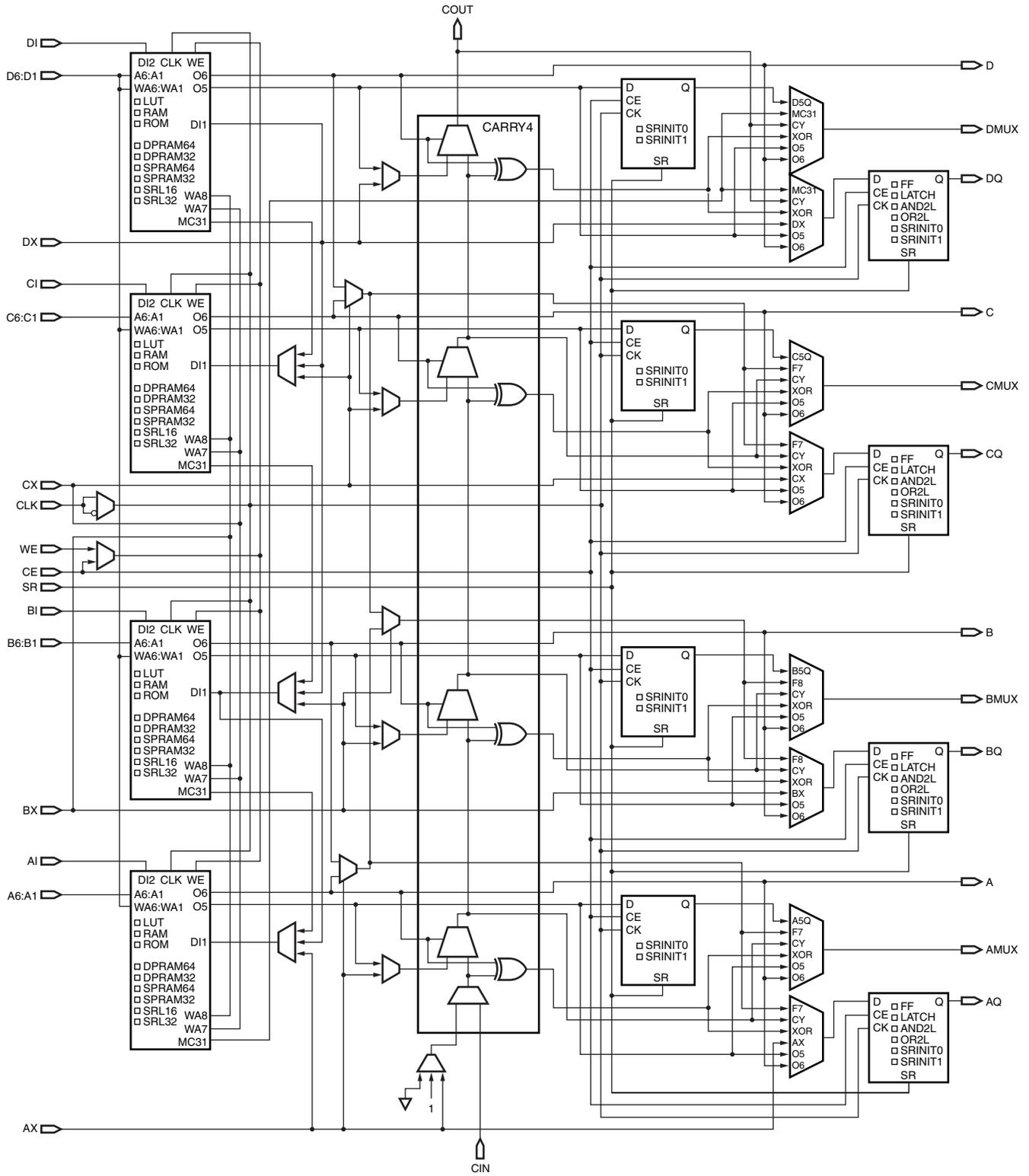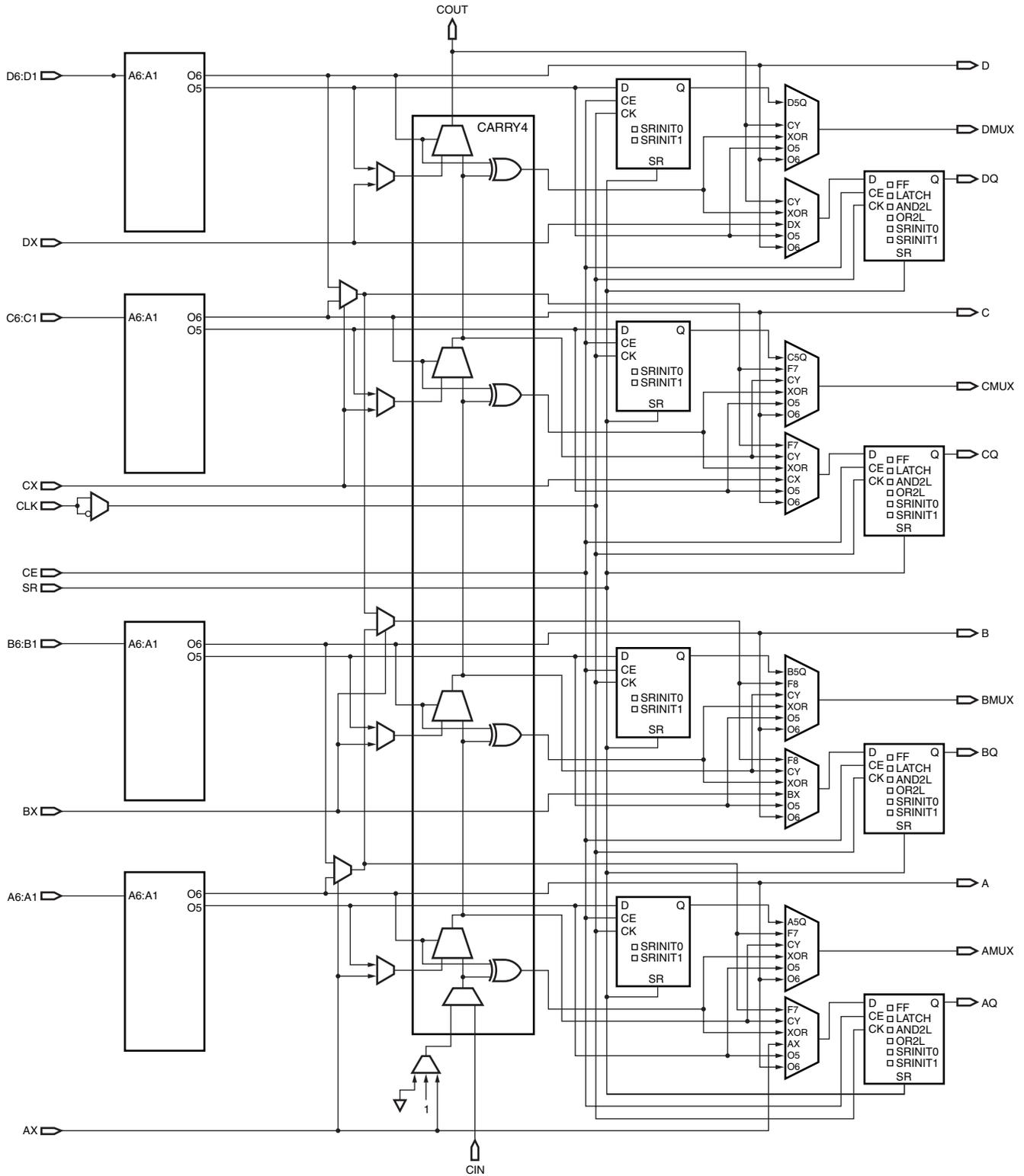
ug384_03_042309

Figure 4: **Diagram of SLICEL**

ug384_04_042309
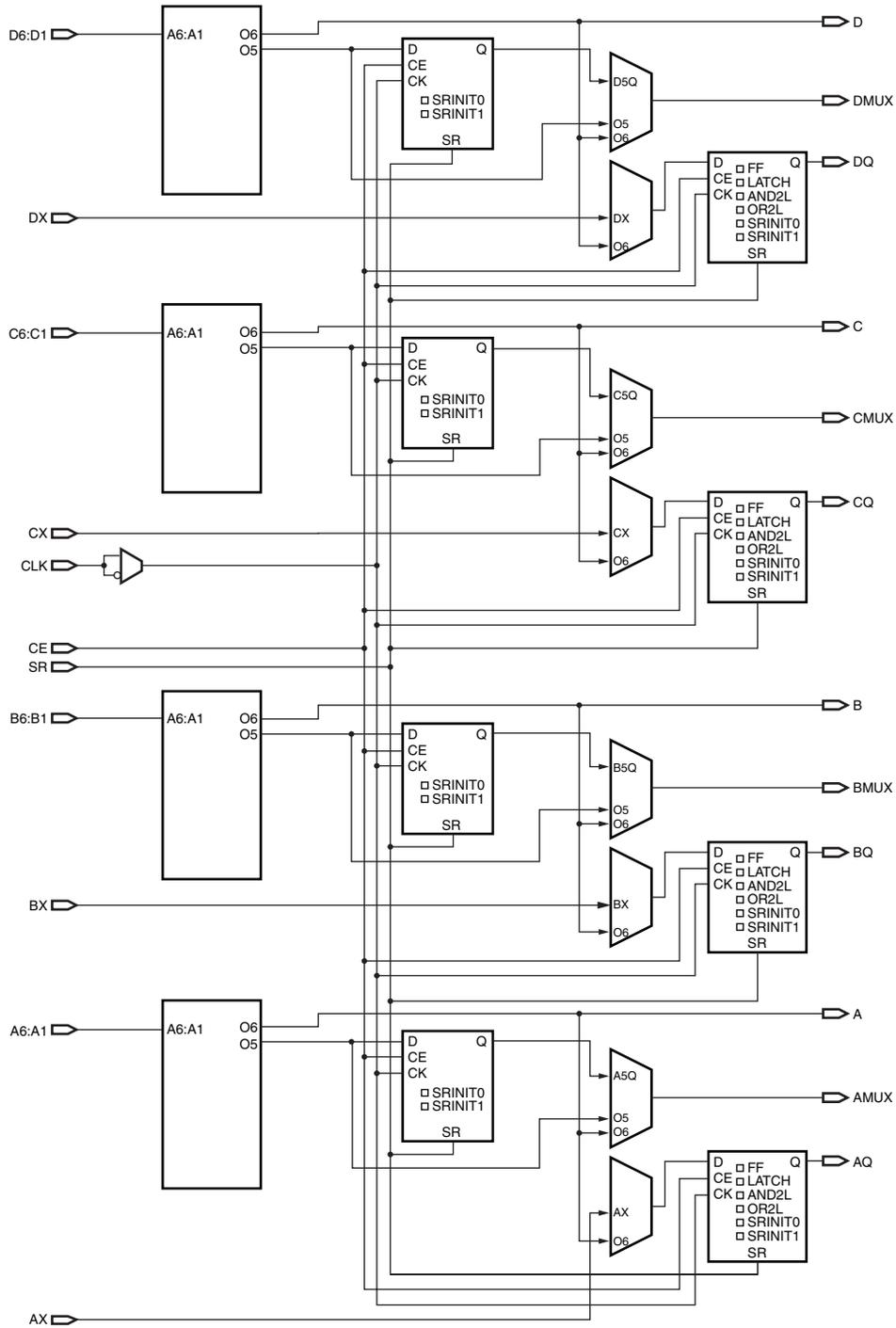
Figure 5: **Diagram of SLICEX**

CLB/Slice Configurations

Table 2 summarizes the logic resources in one CLB. Each CLB or slice can be implemented in one of the configurations listed.

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

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

Notes:
1. SLICEM only, SLICEL and SLICEX do not have distributed RAM or shift registers.
2. SLICEM and SLICEL only.

Table 3 shows the available CLB resources for the Spartan-6 FPGAs. The ratio between the number of 6-input LUTs and logic cells is 1.6. This reflects the increased capability of the new 6-input LUT architecture compared to traditional 4-input LUTs.

Table 3: **Spartan-6 FPGA Logic Resources**

| Device	Logic Cells	Total Slices	SLICEMs	SLICELs	SLICEXs	Number of 6-Input LUTs	Maximum Distributed RAM (Kb)	Shift Registers (Kb)	Number of Flip-Flops
XC6SLX4	3,840	600	300	0	300	2,400	75	38	4,800
XC6SLX9	9,152	1,430	360	355	715	5,720	90	45	11,440
XC6SLX16	14,579	2,278	544	595	1,139	9,112	136	68	18,224
XC6SLX25	24,051	3,758	916	963	1,879	15,032	229	115	30,064
XC6SLX45	43,661	6,822	1,602	1,809	3,411	27,288	401	200	54,576
XC6SLX75	74,637	11,662	2,768	3,063	5,831	46,648	692	346	93,296
XC6SLX100	101,261	15,822	3,904	4,007	7,911	63,288	976	488	126,576
XC6SLX150	147,443	23,038	5,420	6,099	11,519	92,152	1,355	678	184,304
XC6SLX25T	24,051	3,758	916	963	1,879	15,032	229	115	30,064
XC6SLX45T	43,661	6,822	1,602	1,809	3,411	27,288	401	200	54,576
XC6SLX75T	74,637	11,662	2,768	3,063	5,831	46,648	692	346	93,296
XC6SLX100T	101,261	15,822	3,904	4,007	7,911	63,288	976	488	126,576
XC6SLX150T	147,443	23,038	5,420	6,099	11,519	92,152	1,355	678	184,304

Look-Up Table (LUT)

The function generators in Spartan-6 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. Each function generator can also implement two arbitrarily defined five-input Boolean functions, as long as these two functions share common inputs. Only the O6 output of the function generator is used when a six-input function is implemented. Both O5 and O6 are used for each of the five-input function generators implemented. In this case, A6 is driven

High by the software. The propagation delay through a LUT is independent of the function implemented, or whether one six-input or two five-input generators are 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 (see Fast Lookahead Carry Logic), enter the carry-logic chain from an O5 output (see Fast Lookahead Carry Logic), enter the select line of the carry-logic multiplexer from an O6 output (see Fast Lookahead Carry Logic), feed the D input of the storage element, or go to F7AMUX/F7BMUX from an O6 output.

Figure 6 shows a simplified view of the connectivity for a single LUT6.

UG384_06new_021210
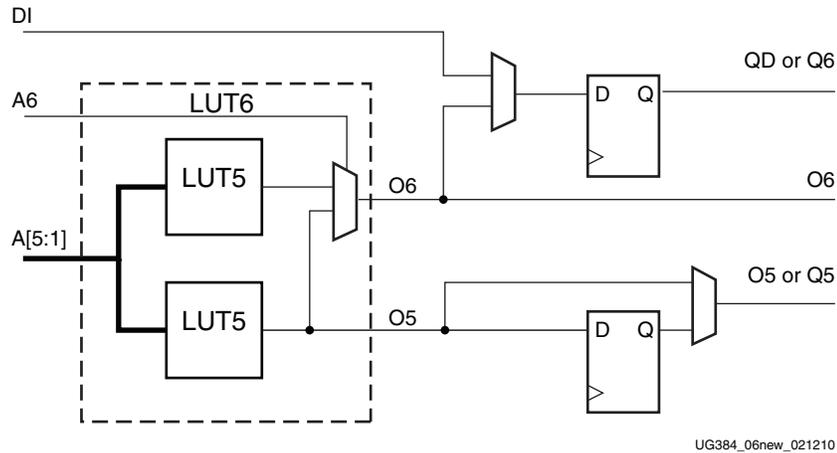
Figure 6: **LUT6**

In addition to the basic LUTs, SLICEL and SLICEM 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 and F7BMUX are used to generate seven input functions from slice A and B, or C and D, while F8MUX is used to combine all slices 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 or between slices, but CLB outputs can be routed through the switch matrix and directly back into the CLB inputs.

Storage Elements

Each slice has eight storage elements. There are four storage elements in a slice that 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.

In Spartan-6 devices, 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. When the original 4 storage elements are configured as latches, these 4 additional storage elements can not be used.

Figure 7 shows both the register only and the register/latch configuration in a slice, both are available.

Registers Only | Registers or Latches

ug384_06_042309
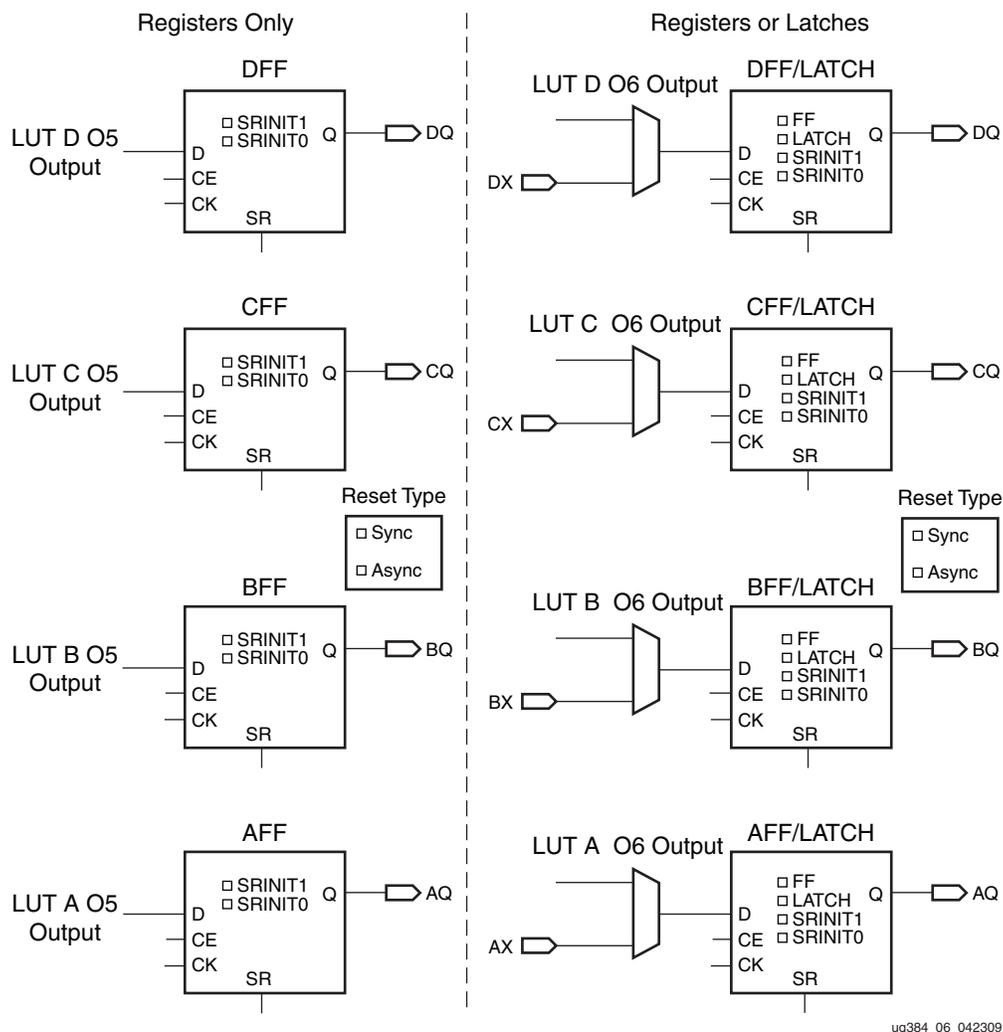
Figure 7: **Configuration in a Slice: 4 Registers Only and 4 Register/Latch**

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 will also have SR or CE enabled by the common signal. Only the CLK signal has independent polarity but applies it to all eight storage elements. Any inverter placed on the clock signal is automatically absorbed. The CE and SR signals are active High. All flip-flop and latch primitives have CE and non-CE versions. The SR signal always has priority over CE.

Initialization

The SR signal forces the storage element into the initial state specified by SRINIT1 or SRINIT0. SRINIT1 forces a logic High at the storage element output when SR is asserted, while SRINIT0 forces a logic Low at the storage element output (see Table 4).

Table 4: **Truth Table When Using SRINIT**

SR	SRINIT	Function
0	SRINIT0 (default)	No Logic Change
1	SRINIT0 (default)	0
0	SRINIT1	No Logic Change
1	SRINIT1	1

SRINIT0 and SRINIT1 can be set individually for each storage element in a slice. The choice of synchronous (SYNC) or asynchronous (ASYNC) set/reset (SRTYPE) is common to all eight storage elements and cannot be set individually for each storage element in a slice.

The initial state after configuration or global initial state is also defined by the same SRINIT option. The initial state is set whenever the Global Set/Reset (GSR) signal is asserted. The GSR signal is always asserted during configuration, and can be controlled after configuration by using the STARTUP_SPARTAN6 primitive. To maximize design flexibility and utilization, use the GSR and avoid local initialization signals.

The initial state of any storage element (SRINIT) is defined in the design either by the INIT attribute or by the use of a set or reset. If both methods are used, they must both be 0 or both be 1. INIT = 0 or a reset selects SRINIT0, and INIT = 1 or a set selects SRINIT1.

The storage element must be initialized to the same value both by the global power-up or GSR signal, and by the local SR input to the slice. A storage element cannot have both set and reset, unless one is defined as a synchronous function so that it can be placed in the LUT. Avoid instantiating primitives with the control input while specifying the INIT attribute in an opposite state, for example, an FDRE with a reset input and the INIT attribute set to 1. Care should be taken when re-targeting designs from another technology to the Spartan-6 architecture. If converting an existing FPGA design, avoid primitives that use both set and reset, such as the FDCPE primitive.

Each of the eight flip-flops in a slice must use the same SR input, although they can be initialized to different values. A second initialization control will require implementation in a separate slice, so minimize the number of initialization signals. The SR could be turned off for all flip-flops in a slice and implemented independently for each flip-flop by implementing it synchronously in the LUT.

The SR signal is available to the flip-flop, independent of whether the LUT is used as a distributed RAM or shift register, which supports a registered read from distributed RAM or an additional pipeline stage in a shift register while still allowing initialization.

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

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

Distributed RAM and Memory (SLICEM only)

The function generators in SLICEMs add a data input and write enable that allows the function generator to be implemented as distributed RAM. RAM resources are

configurable within a SLICEM to implement the distributed RAM shown in Table 5. Multiple LUTs in a SLICEM can be combined in various ways to store more data. Distributed RAM is fast, localized, and ideal for small data buffers, FIFOs, or register files. For larger memory requirements, consider using the 18K block RAM resources.

Distributed RAM are synchronous (write) and asynchronous (read) resources. However, a synchronous read resource can be implemented with a storage element or a flip-flop in the same slice. By placing 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 resources 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 5 shows the number of LUTs (four per slice) occupied by each distributed RAM configuration.

Table 5: **Distributed RAM Configuration**

RAM	Number of LUTs	Description
32 x 2Q[2]	4	Quad-Port 32 x 2-bit RAM
32 x 6SDP[2]	4	Simple Dual-Port 32 x 6-bit RAM
64 x 1S	1	Single-Port 64 x 1-bit RAM
64 x 1D	2	Dual-Port 64 x 1-bit RAM
64 x 1Q[3]	4	Quad-Port 64 x 1-bit RAM
64 x 3SDP[3]	4	Simple Dual-Port 64 x 3-bit RAM
128 x 1S	2	Single-Port 128 x 1-bit RAM
128 x 1D	4	Dual-Port 128 x 1-bit RAM
256 x 1S	4	Single-Port 256 x 1-bit RAM

Notes:
1. S = single-port configuration; D = dual-port configuration; Q = quad-port configuration; SDP = simple dual-port configuration.
2. RAM32M is the associated primitive for this configuration.
3. RAM64M is the associated primitive for this configuration.

For single-port configurations, distributed RAM has a common address port for synchronous writes and asynchronous reads. For dual-port configurations, distributed RAM has one port for synchronous writes and asynchronous reads, and another port for asynchronous reads. In simple dual-port configuration, there is no data out (read port) from the write port. For quad-port configurations, distributed RAM has one port for synchronous writes and asynchronous reads, and three additional ports for asynchronous reads.

In single-port mode, read and write addresses share the same address bus. In dual-port mode, one function generator is connected with the shared read and write port address. The second function generator has the A inputs connected to a second read-only port address and the WA inputs shared with the first read/write port address.

Figure 8 through Figure 16 illustrate various example distributed RAM configurations occupying one SLICEM. When using x2 configuration (RAM32X2Q), A6 and WA6 are driven High by the software to keep O5 and O6 independent.
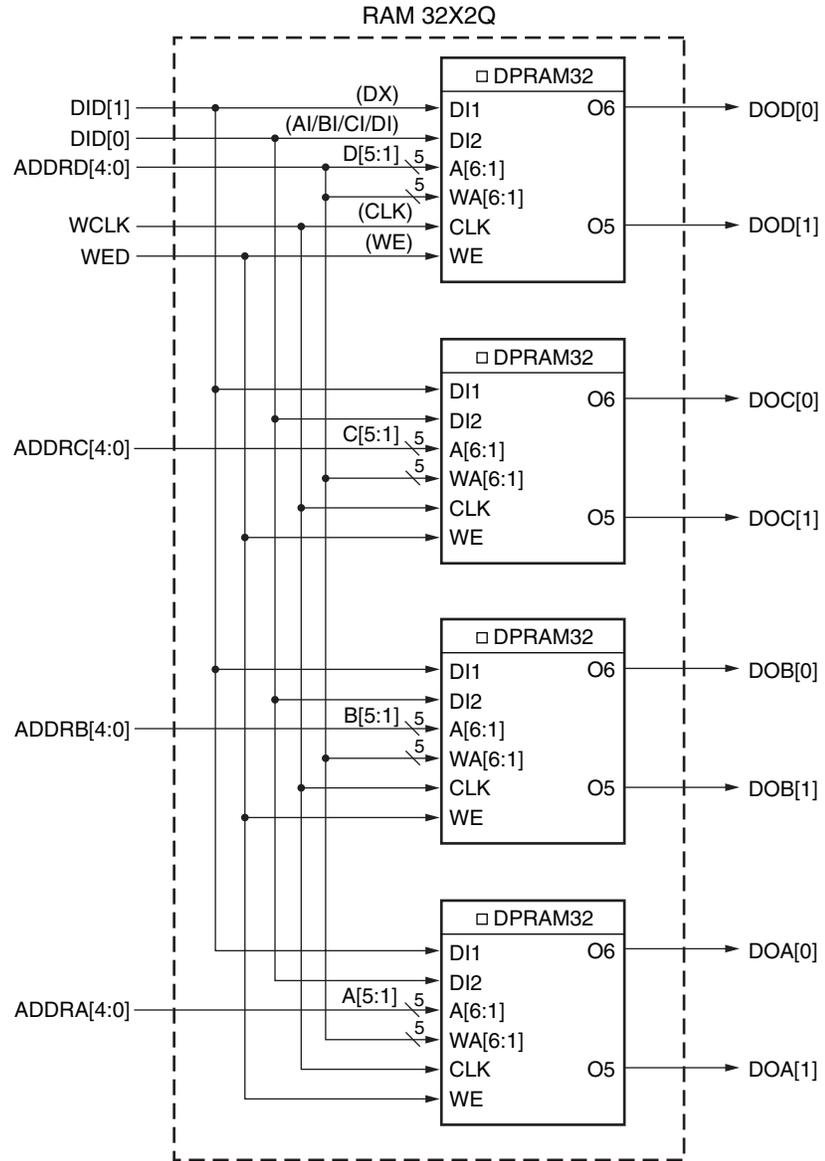
RAM 32X2Q

ug384_07_042309

Figure 8: **Distributed RAM (RAM32X2Q)**
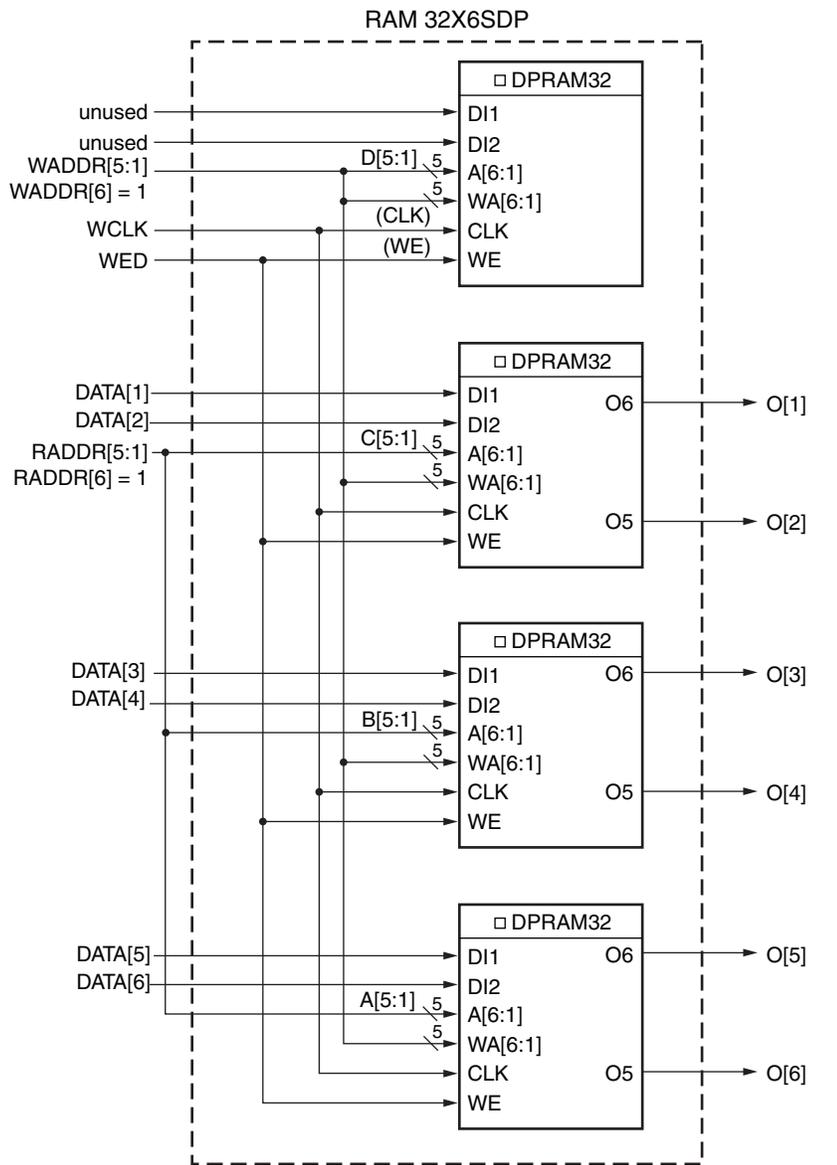
RAM 32X6SDP

Figure 9: **Distributed RAM (RAM32X6SDP)**
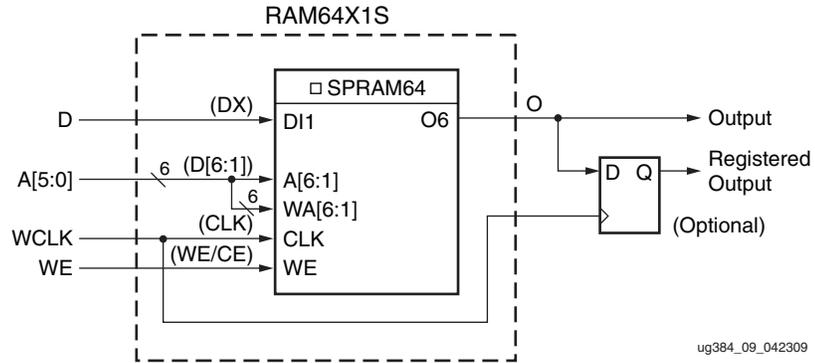
Figure 10: **Distributed RAM (RAM64X1S)**

If four single-port 64 x 1-bit modules are built, 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 64 x 4-bit single-port distributed RAM.
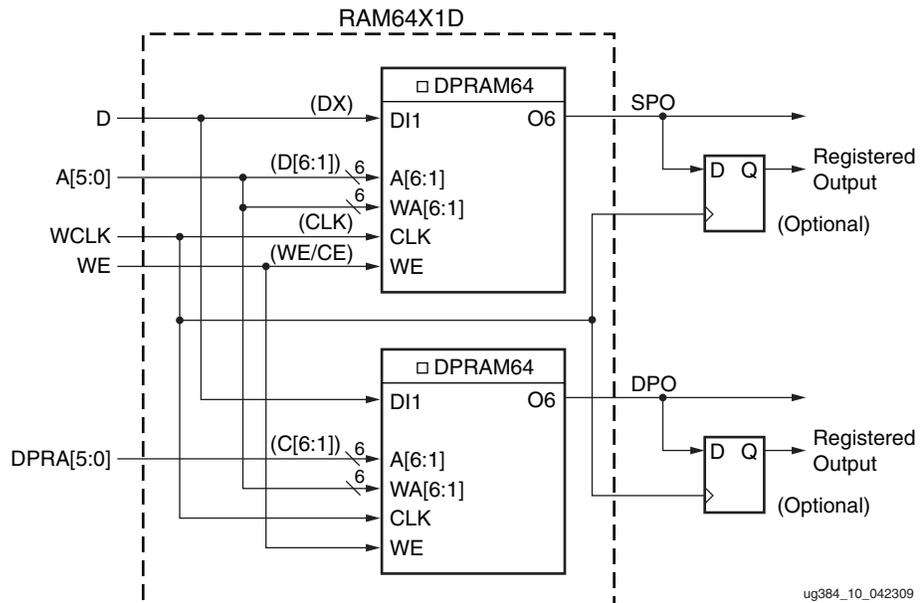
Figure 11: **Distributed RAM (RAM64X1D)**

If two dual-port 64 x 1-bit modules are built, 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 64 x 2-bit dual-port distributed RAM.
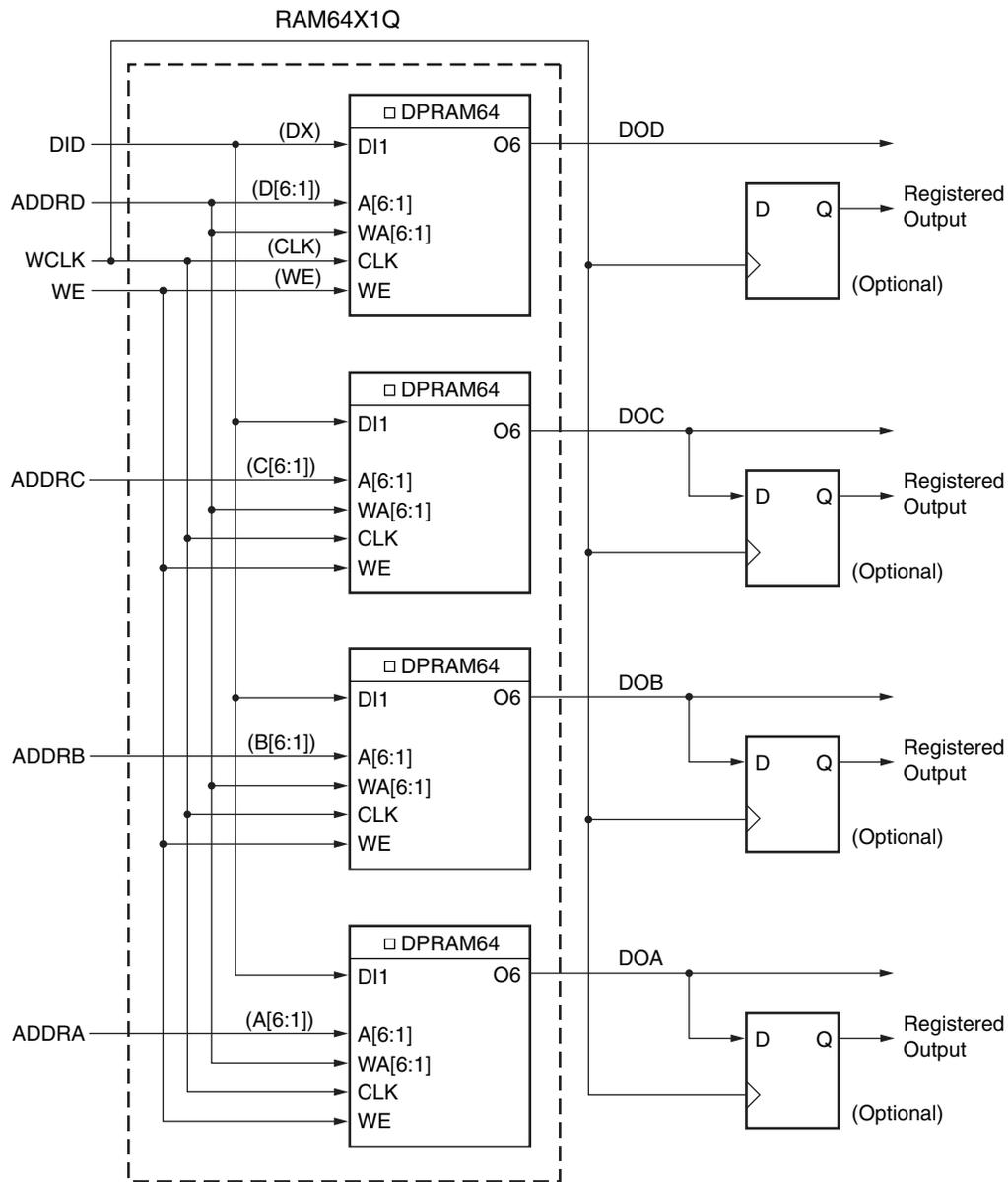
Figure 12: **Distributed RAM (RAM64X1Q)**
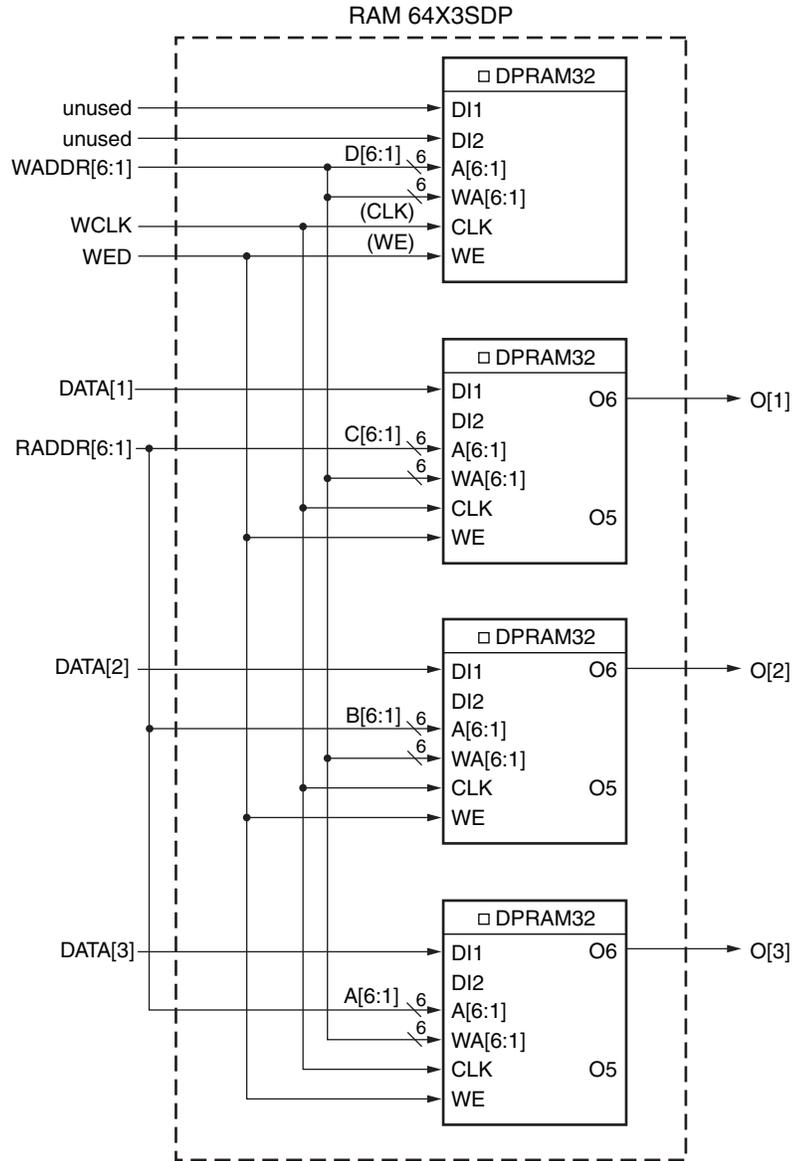
Figure 13: **Distributed RAM (RAM64X3SDP)**

Implementation of distributed RAM configurations with depth greater than 64 requires the usage of wide-function multiplexers (F7AMUX, F7BMUX, and F8MUX).
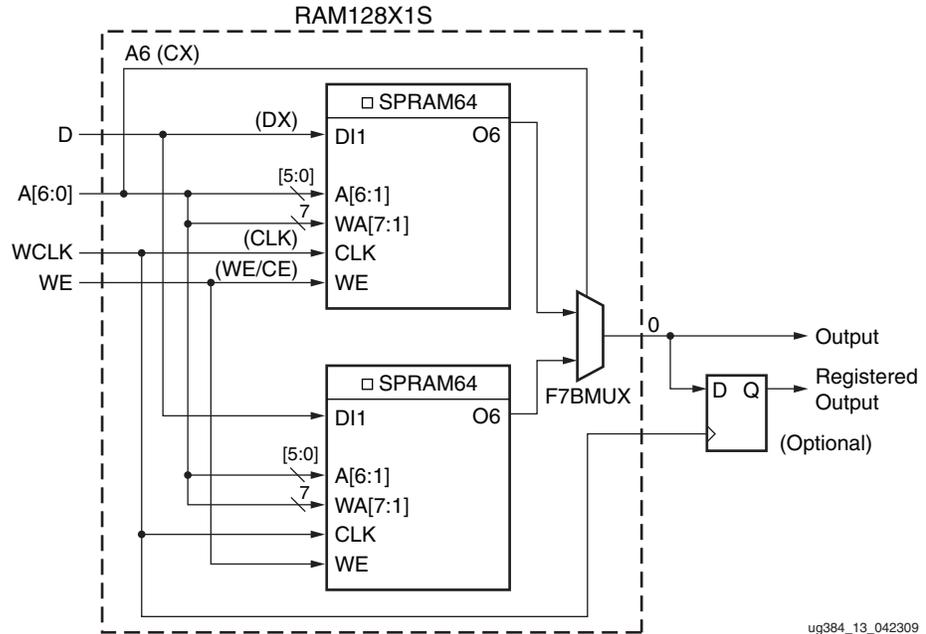
Figure 14: **Distributed RAM (RAM128X1S)**

If two single-port 128 x 1-bit modules are built, 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.

22 www.xilinx.com Spartan-6 FPGA CLB User Guide
UG384 (v1.1) February 23, 2010
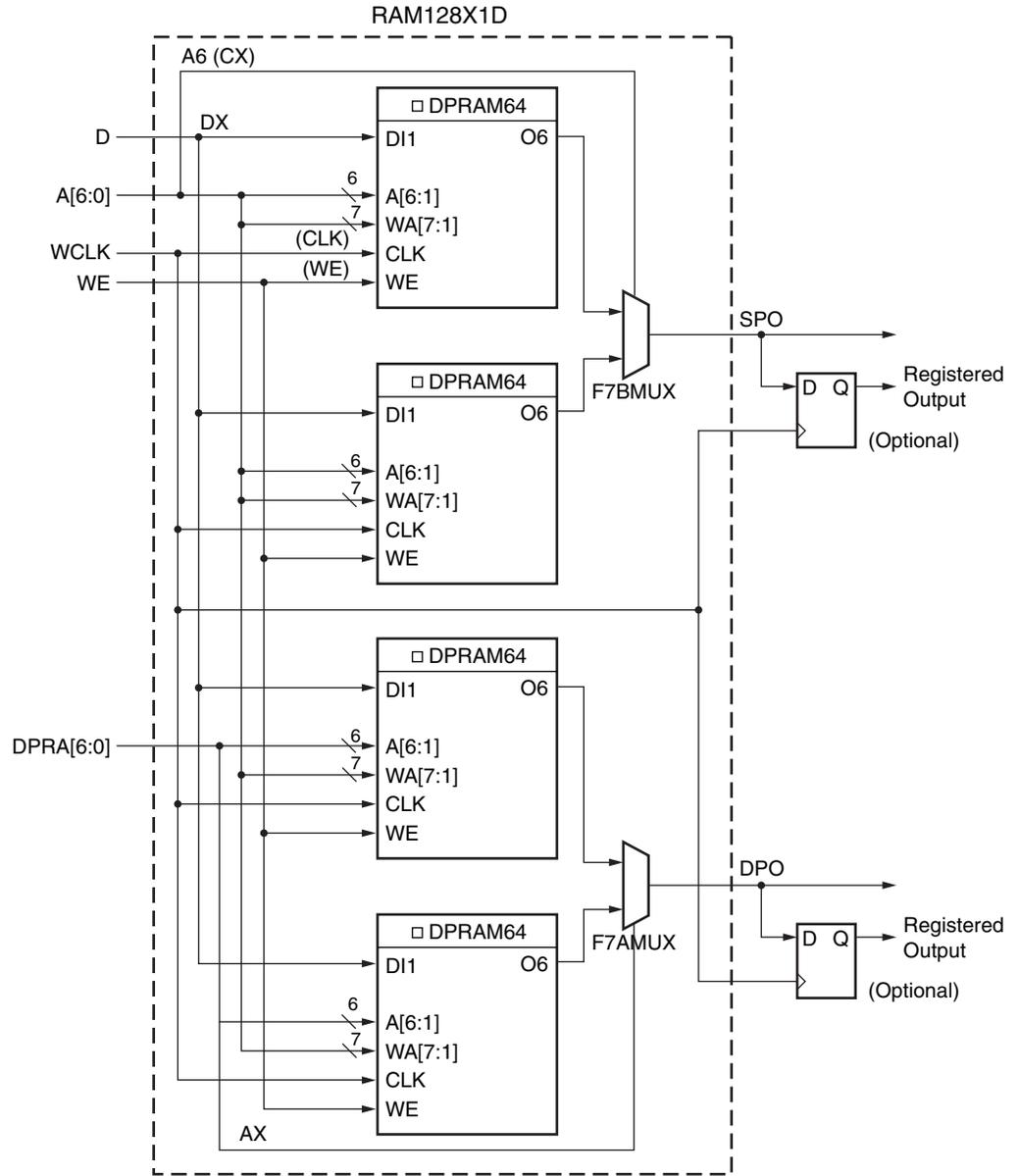
Figure 15: **Distributed RAM (RAM128X1D)**
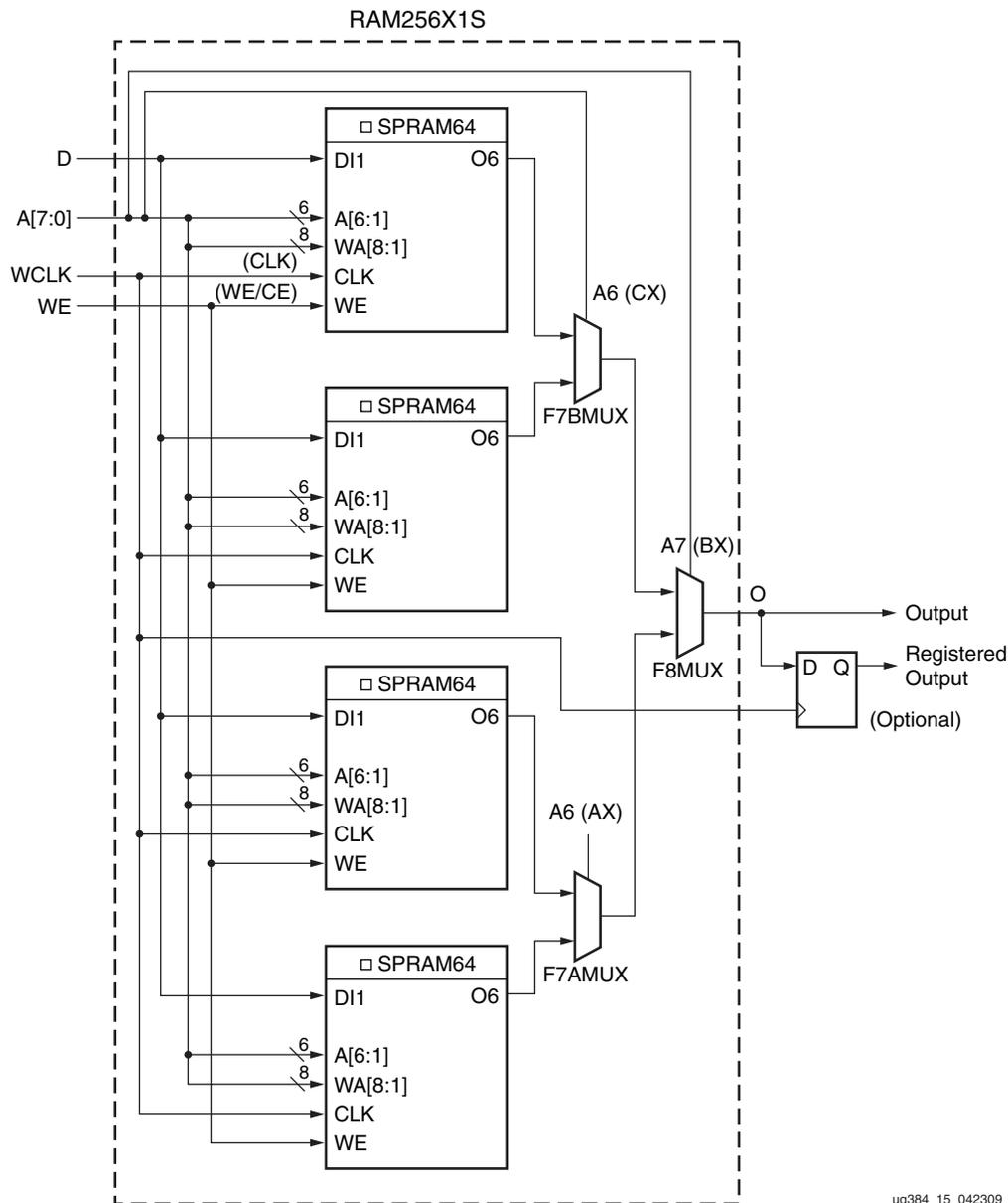
Figure 16: **Distributed RAM (RAM256X1S)**

Distributed RAM configurations larger than the examples provided in Figure 8 through Figure 16 require more than one SLICEM. There are no direct connections to form larger distributed RAM configurations within a CLB or between slices.

Using distributed RAM for memory depths of 64 bits or less is generally more efficient than block RAM in terms of resources, performance, and power. For depths greater than 64 bits but less than or equal to 128 bits, use the following guidelines:

• To conserve LUT resources, use any extra block RAM

• For asynchronous read capability, use distributed RAM

• For widths greater than 16 bits, use block RAM

- For shorter clock-to-out timing and fewer placement restrictions, use registered distributed RAM

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 single-port mode output/SPO output of dual-port mode), or address DPRA (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

- 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 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 6 shows the number of LUTs occupied by each ROM configuration.

Table 6:   **ROM Configuration**

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

Shift Registers (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 anywhere from one 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. Note that 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 requiring delay or latency compensation use these shift registers to develop efficient designs. Shift registers are also useful in synchronous FIFO and content addressable memory (CAM) designs.

The write operation is synchronous with a clock input (CLK) and an optional clock enable (CE). A dynamic read access is performed through the 5-bit address bus, A[4:0]. The LSB of the LUT is unused and the software automatically ties it to a logic High. The configurable shift registers cannot be set or reset. The read is asynchronous; however, a storage element or flip-flop is available to implement a synchronous read. In this case, the clock-to-out of the flip-flop determines the overall delay and improves performance. However, one additional cycle of clock latency is added. Any of the 32 bits can be read out asynchronously (at the O6 LUT outputs) by varying the 5-bit address. This capability is useful in creating smaller shift registers (less than 32 bits). For example, when building a 13-bit shift register, simply set the address to the 13th bit. Figure 17 is a logic block diagram of a 32-bit shift register.

Figure 17: **32-bit Shift Register Configuration**

Figure 18 illustrates an example shift register configuration occupying one function generator.

Figure 18: **Representation of a Shift Register**

Figure 19 shows two 16-bit shift registers. The example shown can be implemented in a single LUT.

Figure 19: **Dual 16-bit Shift Register Configuration**

As mentioned earlier, an additional output (MC31) 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 20 through Figure 22 illustrate various example shift register configurations that can occupy one SLICEM.

Figure 20: **64-bit Shift Register Configuration**

UG384_20_042309

Figure 21:   **96-bit Shift Register Configuration**

Figure 22: **128-bit Shift Register Configuration**

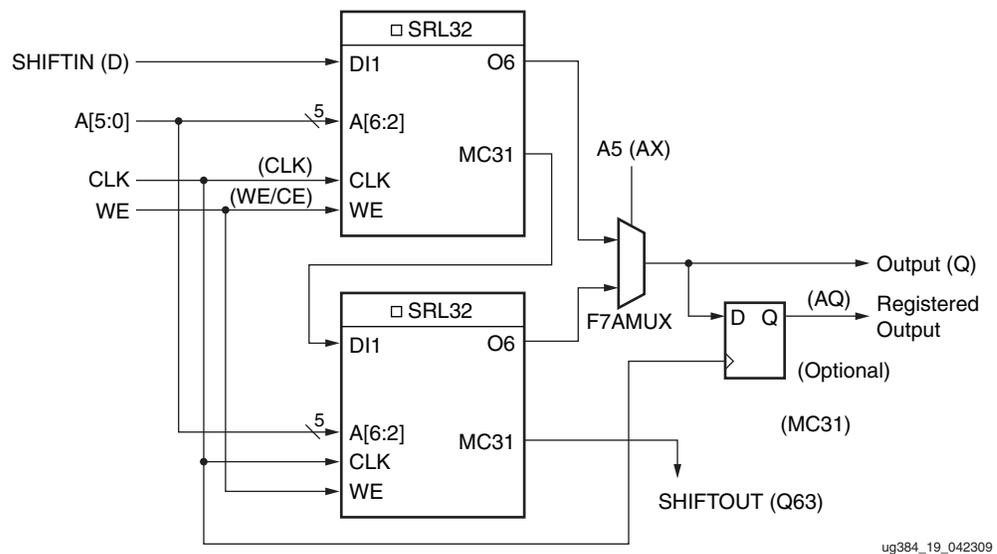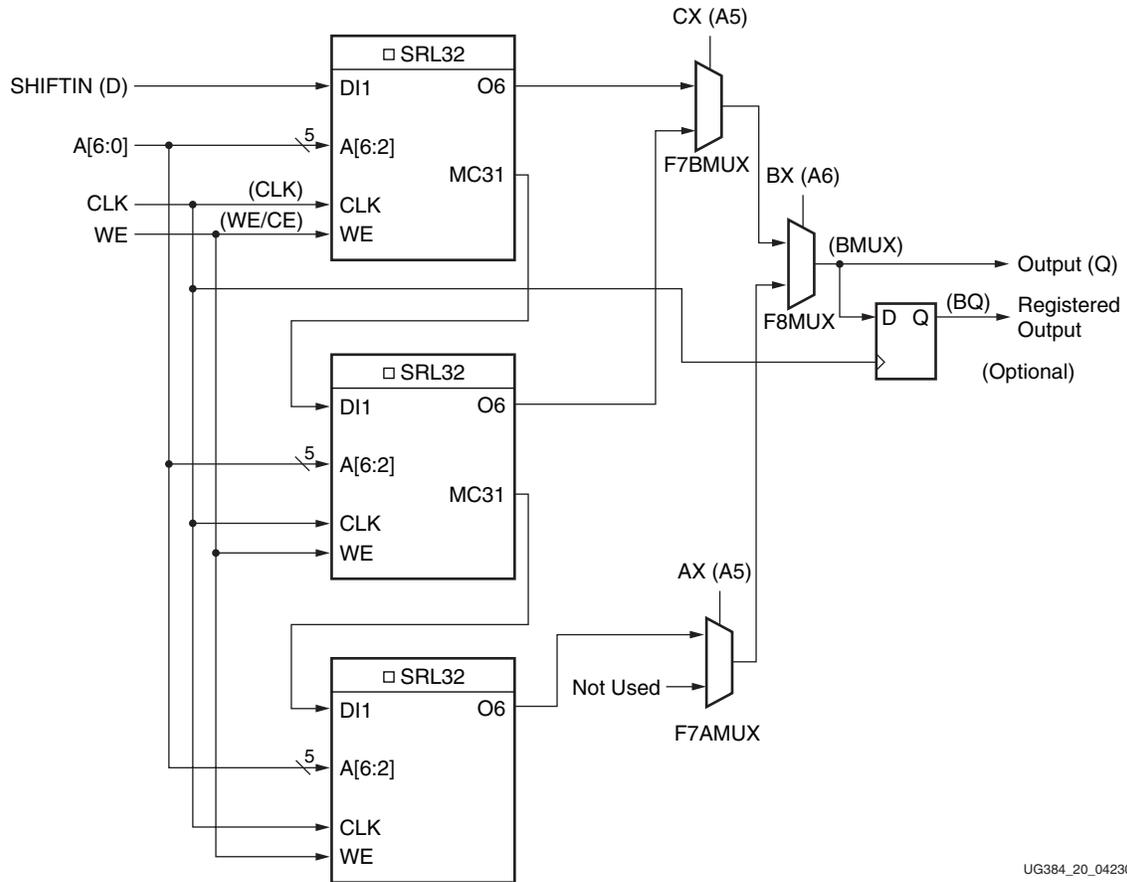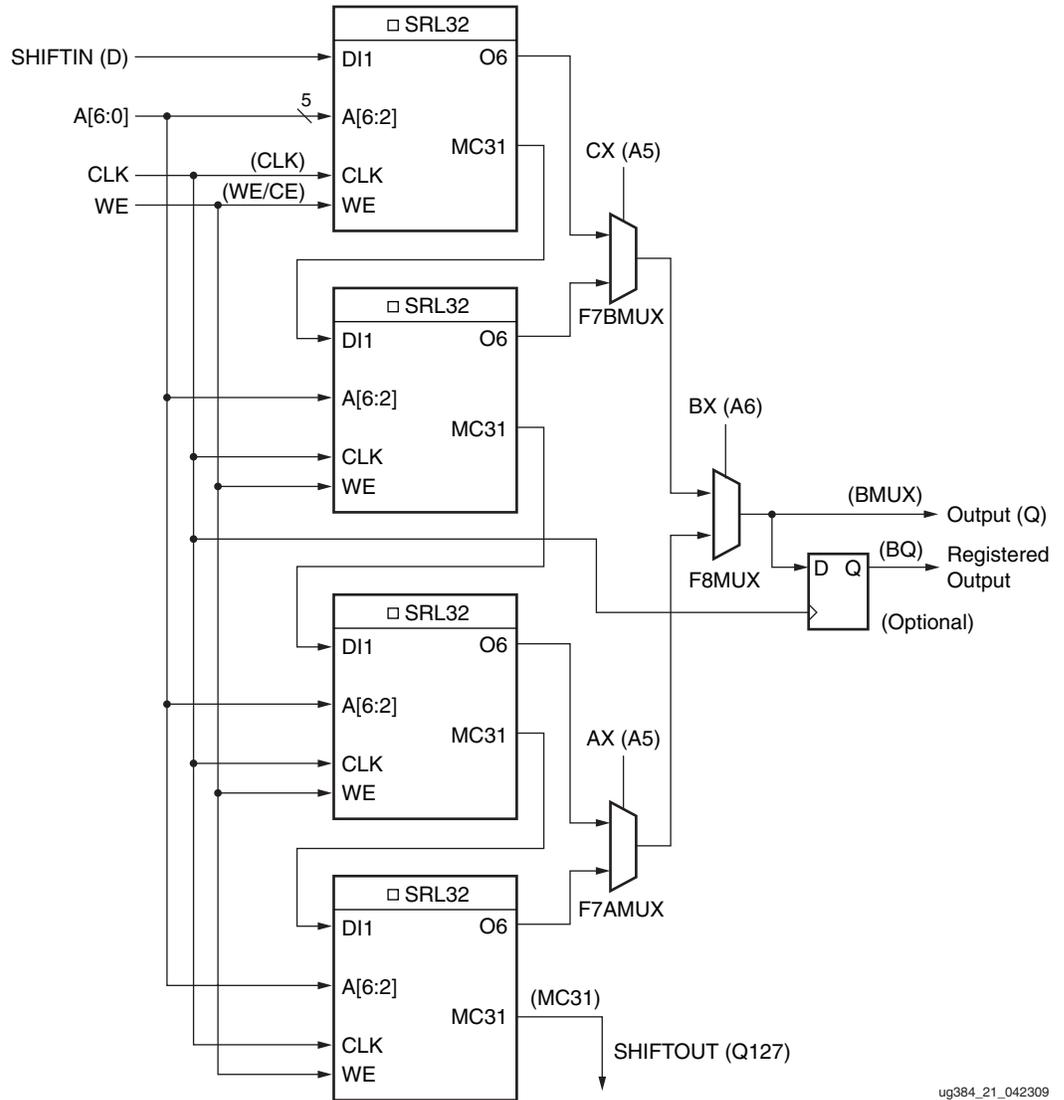It is possible to create shift registers longer than 128 bits across more than one SLICEM. However, there are no direct connections between slices to form these shift registers.

Shift Register Data Flow

Shift Operation

The shift operation is a single clock-edge operation, with an active-High clock enable feature. When enable is High, 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 M31 output.

The bit selected by the 5-bit address port (A[4:0]) appears on the Q output.

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

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 16 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

- A shift operation requires one clock edge.
- Dynamic-length read operations are asynchronous (Q output).
- Static-length read operations are synchronous (Q output).
- 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 SLICEL or SLICEM can implement the following:

- 4:1 multiplexers using one LUT
- 8:1 multiplexers using two LUTs
- 16:1 multiplexers using four LUTs

These wide input multiplexers are implemented in one level or logic (or LUT) using the dedicated F7AMUX, F7BMUX, and F8MUX multiplexers. These multiplexers allow LUT combinations of up to four LUTs in a slice. Dedicated multiplexers can be automatically inferred from the design, or the specific primitives can be instantiated. See WP309: *Targeting and Retargeting Guide for Spartan-6 FPGAs White Paper*.

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 23.
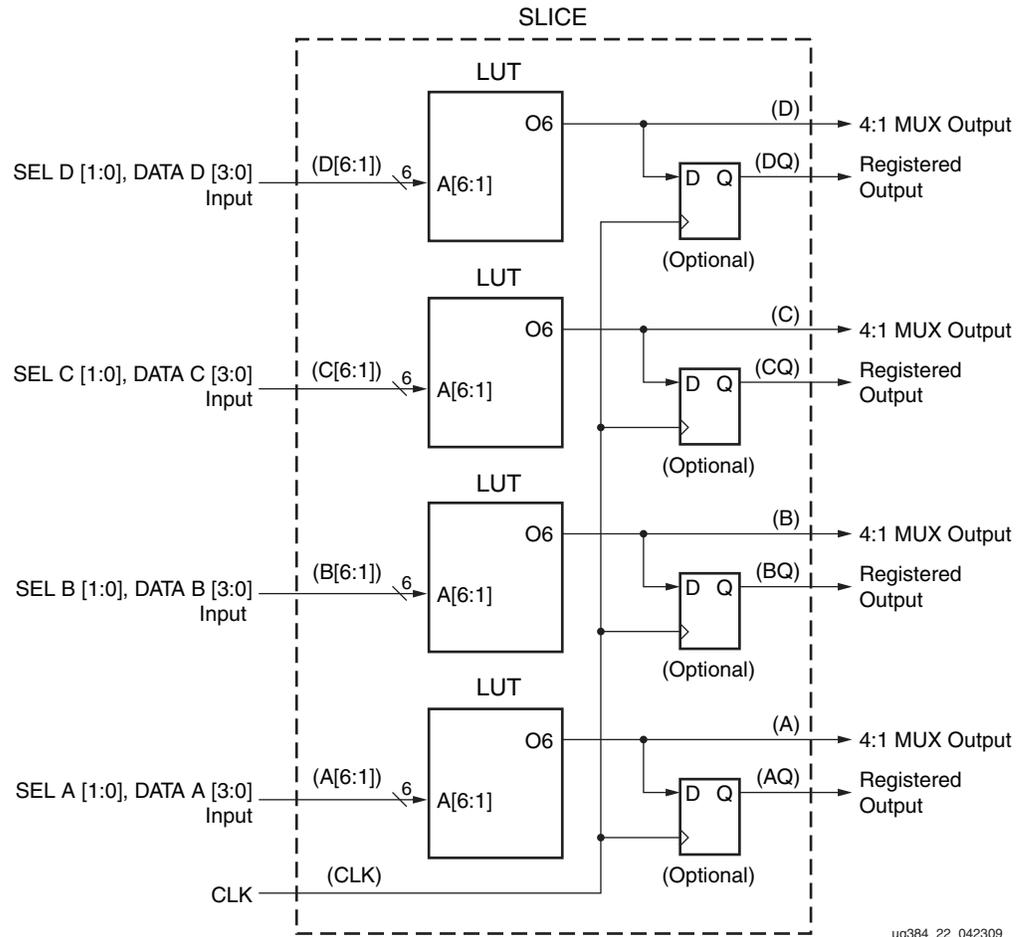
Figure 23: **Four 4:1 Multiplexers in a Slice**

8:1 Multiplexer

Each SLICEL or SLICEM has an F7AMUX and an F7BMUX. These two multiplexers 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 24.
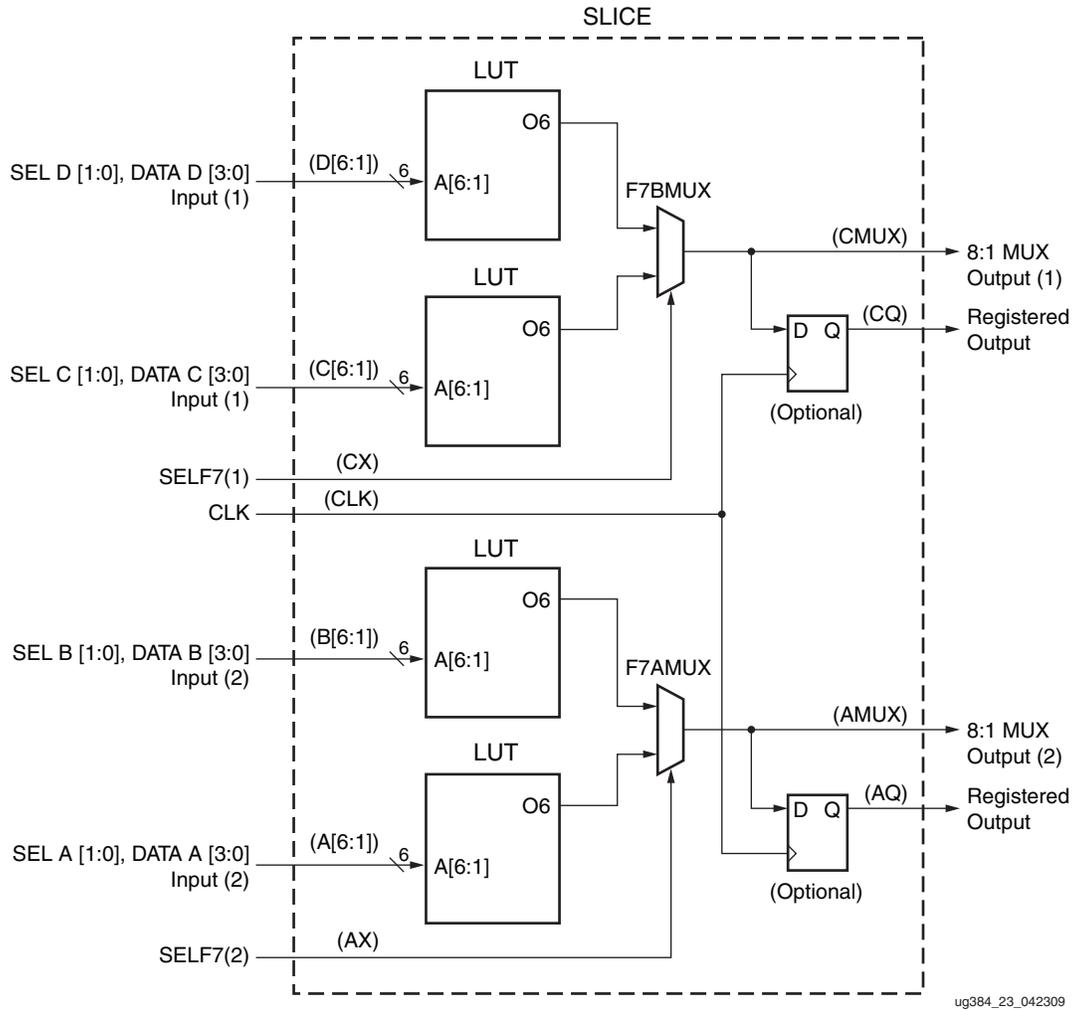
Figure 24: **Two 8:1 Multiplexers in a Slice**

16:1 Multiplexer

Each SLICEL or SLICEM 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 25.
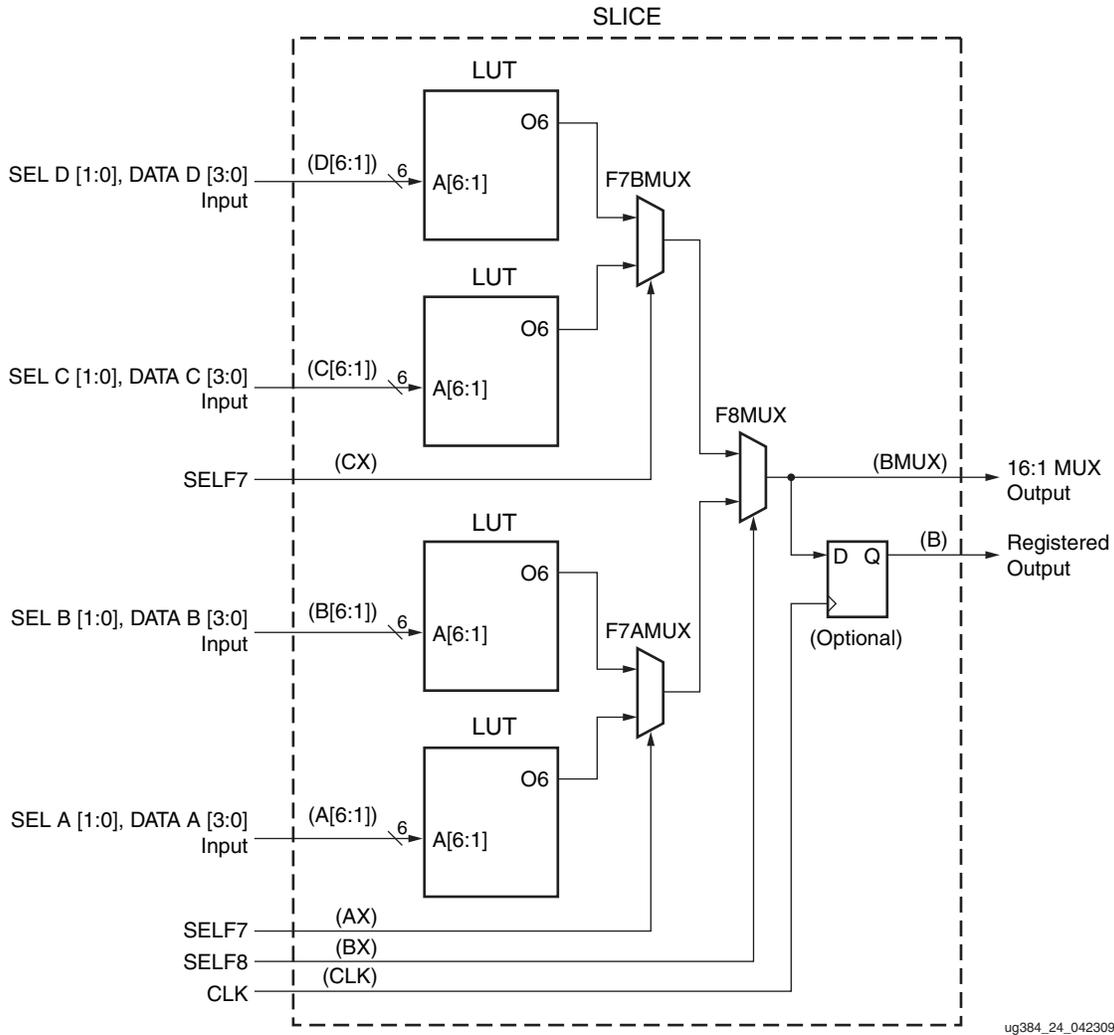
Figure 25: **16:1 Multiplexer in a Slice**

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.

Fast Lookahead Carry Logic

In addition to function generators, SLICEM and SLICEL (but not SLICEX) contain dedicated carry logic to perform fast arithmetic addition and subtraction in a slice. A CLB has one carry chain, as shown in Figure 1. The carry chains are cascadable to form wider add/subtract logic, as shown in Figure 2.

The carry chain in the Spartan-6 device is running 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. Typically, the carry logic

allows four bits of a counter or other arithmetic function to fit in each slice, independent of the function's total size. The dedicated carry path and carry multiplexer (MUXCY) can also be used to cascade function generators for implementing wide logic functions.

Figure 26 illustrates the carry chain with associated logic elements in a slice.
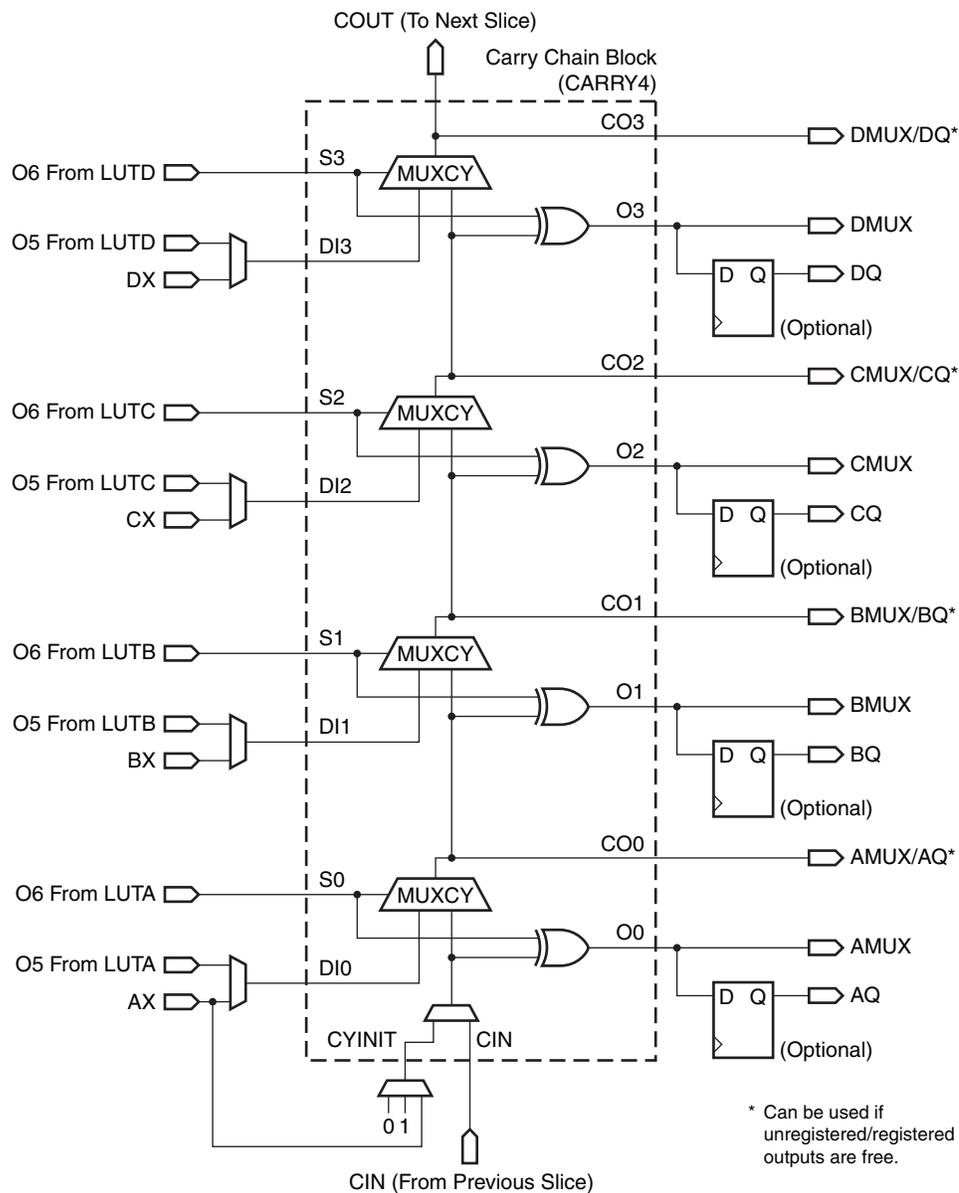
Figure 26: **Fast Carry Logic Path and Associated Elements**

The carry chains carry lookahead logic along with the function generators. There are ten independent inputs (S inputs – S0 to S3, DI inputs – DI1 to DI4, CYINIT and CIN) and eight independent outputs (O outputs – O0 to O3, and CO outputs – CO0 to CO3).

The S inputs are used for the "propagate" signals of the carry lookahead logic. The "propagate" signals are sourced from the O6 output of a function generator. The DI inputs are used for the "generate" signals of the carry lookahead logic. The "generate" signals are sourced from either the O5 output of a function generator or the BYPASS input (AX, BX,

CX, or DX) of a slice. The former input is used to create a multiplier, while the latter is used to create an adder/accumulator. CYINIT is the CIN of the first bit in a carry chain. The CYINIT value can be 0 (for add), 1 (for subtract), or AX input (for the dynamic first carry bit). The CIN input is used to cascade slices to form a longer carry chain. The O outputs contain the sum of the addition/subtraction. The CO outputs 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.

Consider using the DSP48A1 slice adders (see the *Spartan-6 FPGA DSP48A1 Slice User Guide*) for designs consuming too many carry logic resources.

To conserve carry logic resources when designing with adder trees, the 6-input LUT architecture can efficiently create ternary addition (A + B + C = D) using the same amount of resources as simple 2-input addition.

Using the Latch Function as Logic

Since 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 27.

AND2B1L   OR2L

ug384_27_012710
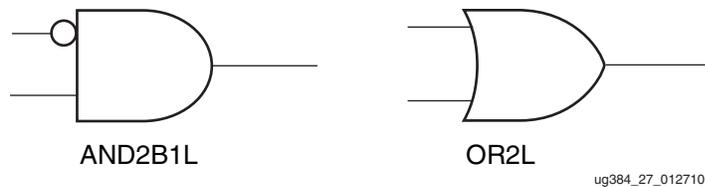
Figure 27:   **AND2B1L and OR2L Components**

As shown in Figure 28, 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. Since there is only one SR input per slice, using more than one AND2B1L or OR2L per slice requires a shared common external input.

ug384_28_021610
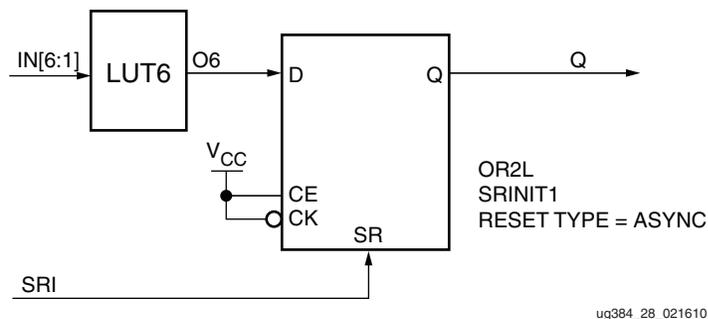
Figure 28:   **Implementation of OR2L (Q = D or SRI)**

The device model shows these functions as AND2L and OR2L configurations of the storage element (Figure 3 through Figure 5). The ISE™ software reports these as **AND/OR Logics** within the slice utilization report. As shown in Table 7, the two inputs of the OR2L gate are not architecturally equivalent; DI is the D input to the latch, and SRI is the SR input.

Table 7:   **OR2L Logic Table**

Inputs		Outputs
DI	**SRI**	**O**
0	0	0
0	1	1
1	0	1
1	1	1

The AND2B1L and OR2L two-input gates save LUT resources and are initialized to a known state on power-up and on GSR assertion. 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 Place and Route (PAR) tool within the ISE Design Suite software exploits the rich interconnect array to deliver optimal system performance and the fastest compile times.

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 the *Spartan-6 FPGA Clocking Resources User Guide*. Two global control signals, GTS and GSR, are selected by using the STARTUP_SPARTAN6 primitive, which is described in Global Controls. Knowledge of the general-purpose routing resources is helpful when considering floorplanning the layout of a design.

Spartan-6 FPGA Interconnect Types

The Spartan-6 FPGA 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 (Figure 29). A similar switch matrix connects other resources, such as the DSP slices and block RAM resources.
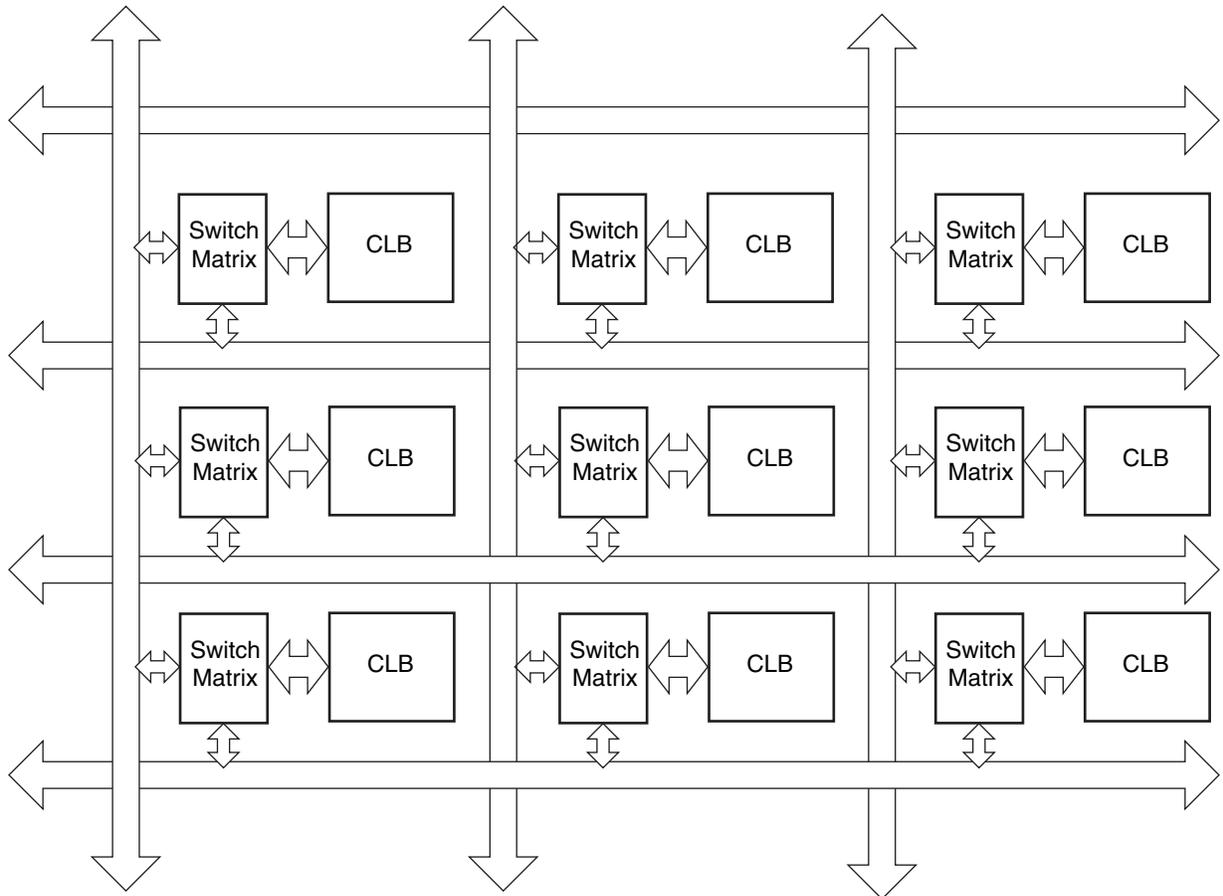
ug384_29_012710

Figure 29: **CLB Array and Interconnect Channels**

The various types of routing in the Spartan-6 architecture are primarily defined by their length (Figure 30). Longer routing elements are faster for longer distances.

Fast Interconnects

Fast connects route block outputs back to block inputs. Along with the larger size of the CLB, fast connects provide higher performance for simpler functions.

Single Interconnects

Singles route signals to neighboring tiles, both vertically and horizontally.

Double Interconnects

Doubles connect to every other tile, both horizontally and vertically, in all four directions, and to the diagonally adjacent tiles.

Quad Interconnects

Quads connect to one out of every four tiles, horizontally and vertically, and diagonally to tiles two rows and two columns distant. Quad lines provide more flexibility than the single-channel long lines of earlier generations.
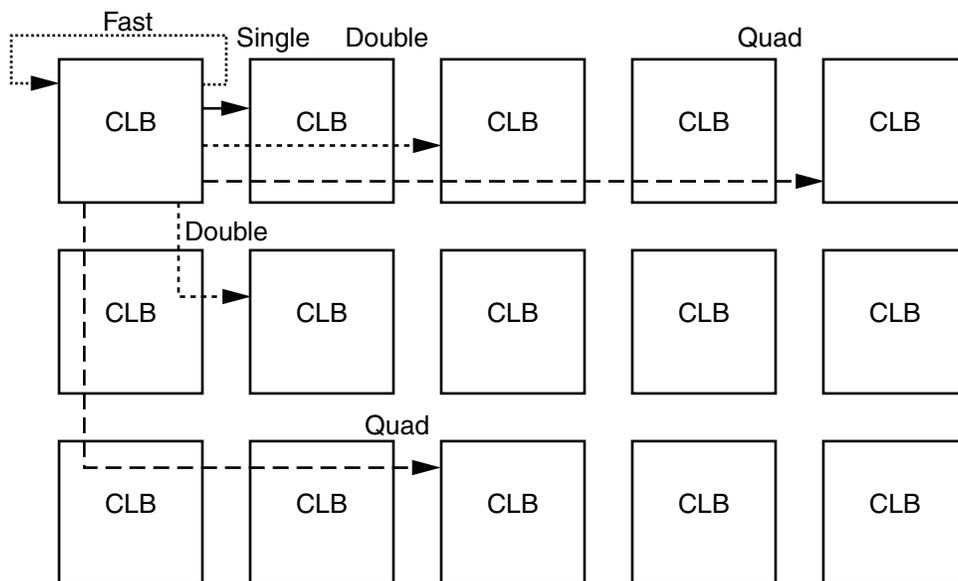
Figure 30: **Examples of Interconnect Types**

Interconnect Delay and 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. A good estimate of interconnect delay is to use the same value as the block delays in a path.

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 PAR 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.

Interconnect delays are typically improved not by changing the interconnect but by changing the placement. This is the Floorplanning process.

Viewing Interconnect Details with FPGA Editor

The FPGA Editor is used to view the interconnect of a blank device or to view the interconnect used in an implemented design. FPGA Editor is a graphical application for displaying and configuring FPGAs.

To access the FPGA Editor, first run place-and-route on your design. Then double-click on the **View/Edit Routed Design** (FPGA Editor) process to open FPGA Editor.

For details on using FPGA Editor, see the on-line help within the FPGA Editor application.

Global Controls

In addition to the general-purpose interconnect, Spartan-6 FPGAs have two global logic control signals, as described in Table 8.

Table 8: **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 (see Storage Elements). Asserted automatically during the FPGA configuration process (see the Configuration Sequence section in the *Spartan-6 FPGA Configuration User Guide*.
GTS	Global 3-State: When High, asynchronously forces all I/O pins to a high-impedance state (High-Z, 3-state).

Use the GSR control in a design instead of a separate 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. The GSR signal is asserted automatically during the FPGA configuration process, guaranteeing that the FPGA starts up in a known state. Using GSR and GTS does not use any general-purpose routing resources.

STARTUP_SPARTAN6 Primitive

The GSR and GTS signal sources are defined and connected using the STARTUP_SPARTAN6 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 STARTUP 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 STARTUP primitive also includes other signals used specifically during configuration. For more information, read the *Spartan-6 FPGA Configuration User Guide*.

Interconnect Summary

The flexible interconnect resources of the Spartan-6 family allows efficient implementation of almost any configuration of logic and I/O resources. The ISE software automatically places and routes designs to take best advantage of these general-purpose resources. Dedicated resources for clocks are used when clock buffers are used in a design. Dedicated resources for global set/reset and global 3-state are controlled by using the STARTUP_SPARTAN6 primitive.

Floorplanning

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 PlanAhead Design Analysis tool provides a graphical view of placement, and helps the designer make choices between RTL coding and synthesis and implementation, with extensive design exploration and analysis features. More information on the PlanAhead tool is available at: http://www.xilinx.com/tools/planahead.htm.

For floorplanning and design analysis it is important to understand the general layout and naming designations for the CLB resources. As shown in Figure 2, CLBs each contain two

Spartan-6 FPGA CLB User Guide
UG384 (v1.1) February 23, 2010

www.xilinx.com

39

slices, with columns of SLICEM/SLICEX CLBs alternating with columns of SLICEL/SLICEX CLBs. The numbering begins from the bottom left corner. Although all the I/O blocks are outside the CLB array, the IOI cells sometimes take the place of CLBs on the perimeter. In addition, columns of block RAM, DSP slices, and clock management tiles are embedded in the columns of CLBs. Lastly, the GTP transceivers and integrated blocks for PCI Express are embedded in the top of the array, as well as in the bottom of the array in the larger devices.

An example floorplan for the XC6SLX45T is shown in Figure 31.

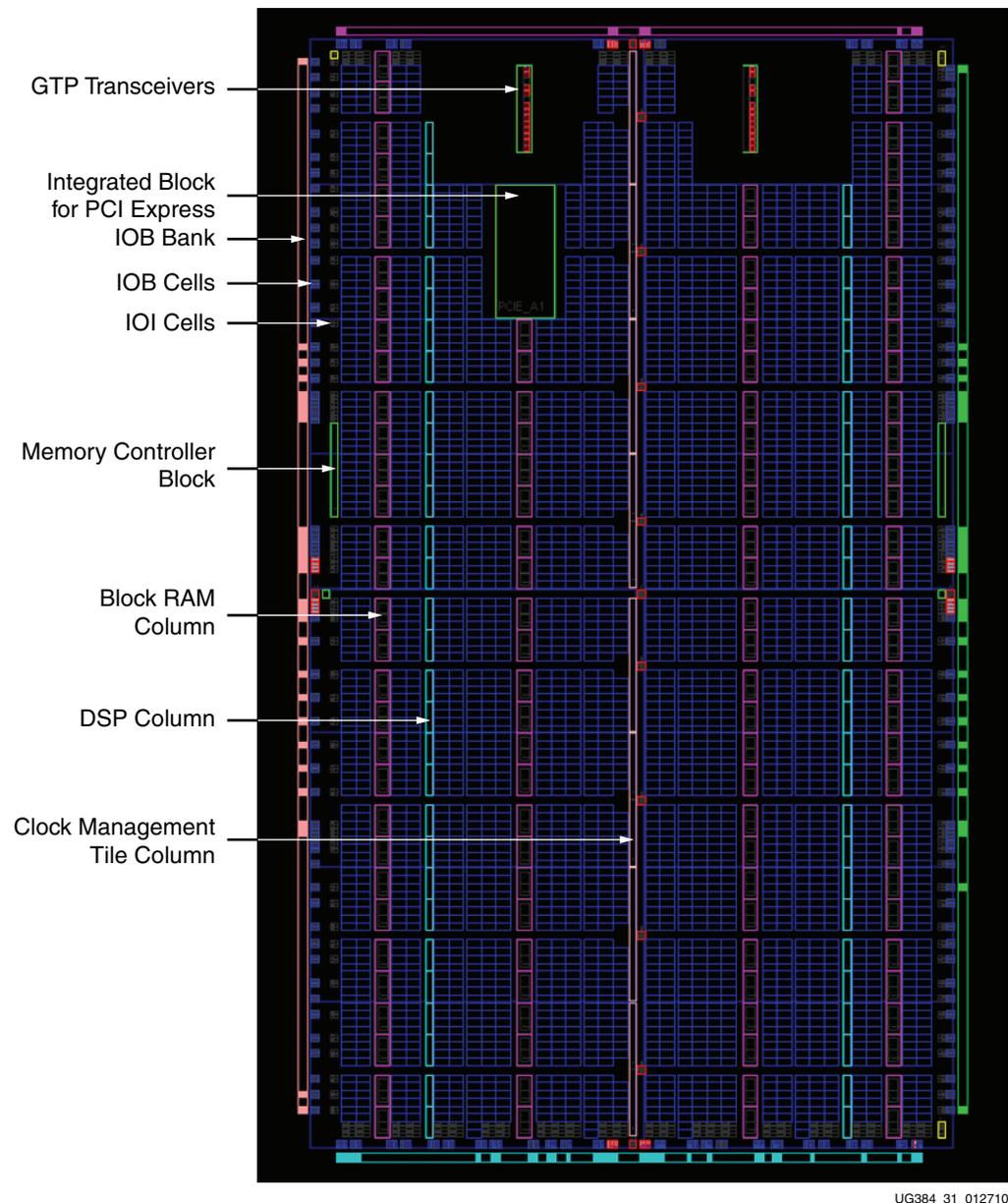UG384_31_012710

Figure 31: **XC6SLX45T Floorplan View in PlanAhead**

CLB / Slice Timing Models

Due to the large size and complexity of Spartan-6 FPGAs, understanding the timing associated with the various paths and functional elements is a difficult and important task. 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.

Three timing model sections are described:

- Functional element diagram – basic architectural schematic illustrating pins and connections
- Timing parameters – definitions of *Spartan-6 FPGA Data Sheet* timing parameters
- Timing Diagram - illustrates functional element timing parameters relative to each other

Use the models in this chapter in conjunction with both the Xilinx Timing Analyzer software (TRCE) and the section on switching characteristics in the *Spartan-6 FPGA Data Sheet*. All pin names, parameter names, and paths are consistent with the post-route timing and pre-route static timing reports. Most of the timing parameters found in the section on switching characteristics are described in this chapter.

All timing parameters reported in the *Spartan-6 FPGA Data Sheet* are associated with slices and CLBs. The following sections correspond to specific switching characteristics sections in the *Spartan-6 FPGA Data Sheet*:

- Slice (LUT and Storage Element) Timing Models
- Slice Distributed RAM Timing Model (SLICEM only)
- Slice SRL Timing Model (SLICEM only)
- Slice Carry-Chain Timing Model (SLICEM and SLICEL only)

Slice (LUT and Storage Element) Timing Models

A simplified Spartan-6 FPGA slice is shown in Figure 32. Some elements of the slice are omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.
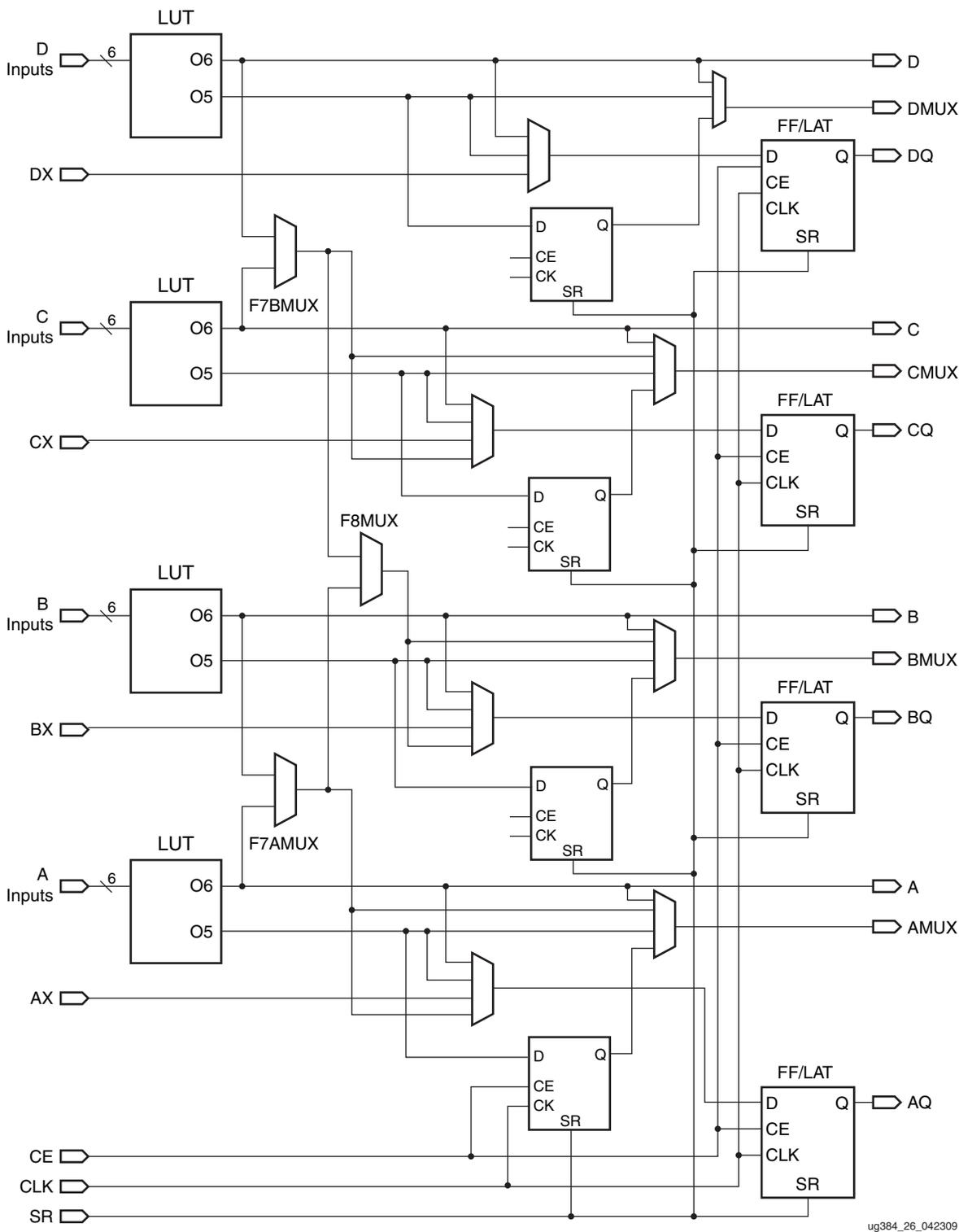
Figure 32: **Simplified Spartan-6 FPGA Slice**

Timing Parameters

Table 9 shows the general slice timing parameters for a majority of the paths in Figure 32.

Table 9: **Slice (LUT and Storage Element) 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 inputs of the slice, through the look-up tables (LUTs), to the A/B/C/D outputs of the slice (six-input function).
T_{ILOF}	A/B/C/D inputs to AMUX/CMUX outputs	Propagation delay from the A/B/C/D inputs of the slice, through the LUTs and F7AMUX/F7BMUX to the AMUX/CMUX outputs (seven-input function).
T_{OPAB}	A/B/C/D inputs to BMUX output	Propagation delay from the A/B/C/D inputs of the slice, through the LUTs, F7AMUX/F7BMUX, and F8MUX to the BMUX output (eight-input function).
Sequential Delays		
T_{CKO} Flip-Flop/ Latch element	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_{CKO} Flip-Flop only element	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.
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).
Setup and Hold Times for Slice Sequential Elements[2]		
T_{DICK}/T_{CKDI} Flip-Flop/ Latch element	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 at the D input of the slice sequential elements (configured as a flip-flop).
T_{DICK}/T_{CKDI} Flip-Flop only element	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 at the D input of the slice sequential elements.
T_{CECK}/T_{CKCE} Flip-Flop/ Latch element	CE input	Time before/after the CLK that the CE input of the slice must be stable at the CE input of the slice sequential elements (configured as a flip-flop).
T_{CECK}/T_{CKCE} Flip-Flop only element	CE input	Time before/after the CLK that the CE input of the slice must be stable at the CE input of the slice sequential elements.
T_{SRCK}/T_{CKSR} Flip-Flop/ Latch element	SR input	Time before/after the CLK that the SR inputs of the slice must be stable at the SR inputs of the slice sequential elements (configured as a flip-flop).

*Table 9:* **Slice (LUT and Storage Element) Timing Parameters** *(Cont'd)*

Parameter	Function	Description
T_{SRCK}/T_{CKSR} Flip-Flop only element	SR input	Time before/after the CLK that the SR inputs of the slice must be stable at the SR inputs of the slice sequential elements
Set/Reset		
T_{SRMIN}		Minimum Pulse Width for the SR.
T_{RQ}		Propagation delay for an asynchronous set/reset of the slice sequential elements from the SR inputs to the AQ/BQ/CQ/DQ outputs.
F_{TOG}		Toggle Frequency – Maximum frequency that a CLB flip-flop can be clocked: $1 / (T_{CH} + T_{CL})$.

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).

Timing Characteristics

Figure 33 illustrates the general timing characteristics of a Spartan-6 FPGA slice.
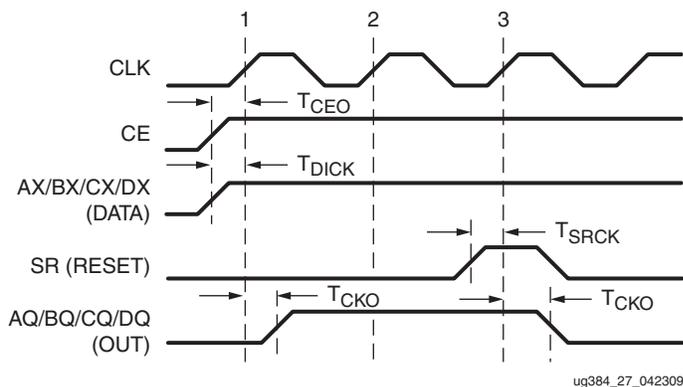
Figure 33: **General Slice Timing Characteristics**

- At time T_{CEO} 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).

Slice Distributed RAM Timing Model (SLICEM only)

Figure 34 illustrates the details of distributed RAM implemented in a Spartan-6 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.
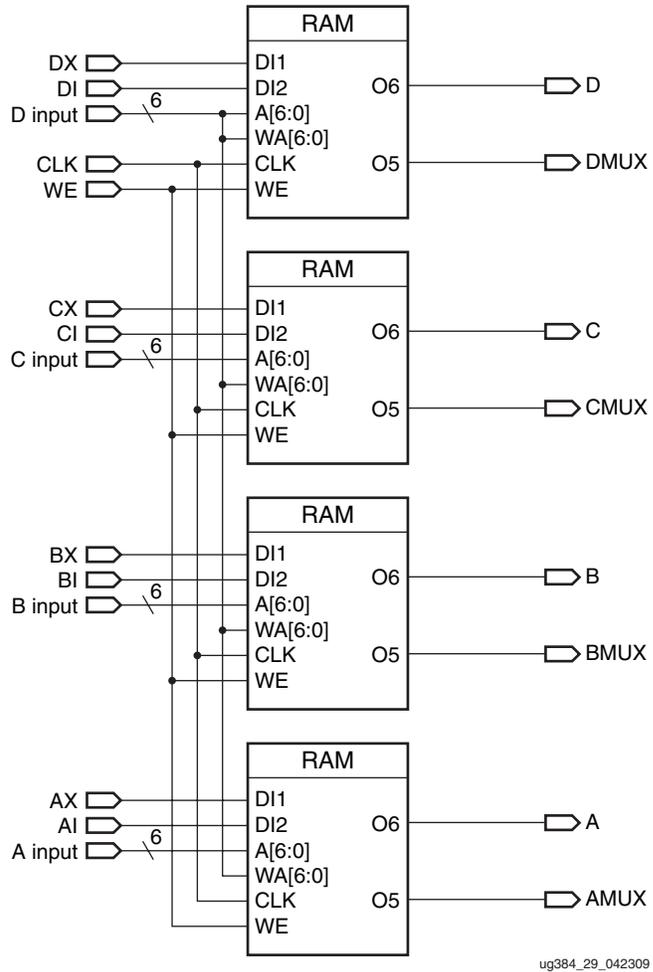

Figure 34: **Simplified Spartan-6 FPGA SLICEM Distributed RAM**

Distributed RAM Timing Parameters

Table 10 shows the timing parameters for the distributed RAM in SLICEM for a majority of the paths in Figure 34.

Table 10: **Distributed RAM Timing Parameters**

Parameter	Function	Description
Sequential Delays for a Slice LUT Configured as RAM (Distributed RAM)		
T_{SHCKO}[1]	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 output of the slice.
Setup and Hold Times for a Slice LUT Configured as RAM (Distributed RAM)[2]		
T_{DS}/T_{DH}[3]	AX/BX/CX/DX configured as data input (DI1)	Time before/after the clock that data must be stable at the AX/BX/CX/DX input of the slice.
T_{AS}/T_{AH}	A/B/C/D address inputs	Time before/after the clock that address signals must be stable at the A/B/C/D inputs of the slice LUT (configured as RAM).
T_{WS}/T_{WH}	WE input	Time before/after the clock that the write enable signal must be stable at the WE input of the slice LUT (configured as RAM).
Clock CLK		
T_{MPW}		Minimum pulse width.
T_{MCP}		Minimum clock period to meet address write cycle time.

Notes:
1. This parameters includes a LUT configured as a two-bit distributed RAM.
2. T_{XXCK} = Setup Time (before clock edge), and T_{CKXX} = Hold Time (after clock edge).
3. Parameter includes AI/BI/CI/DI configured as a data input (DI2).

Distributed RAM Timing Characteristics

The timing characteristics of a 16-bit distributed RAM implemented in a Spartan-6 FPGA slice (LUT configured as RAM) are shown in Figure 35.
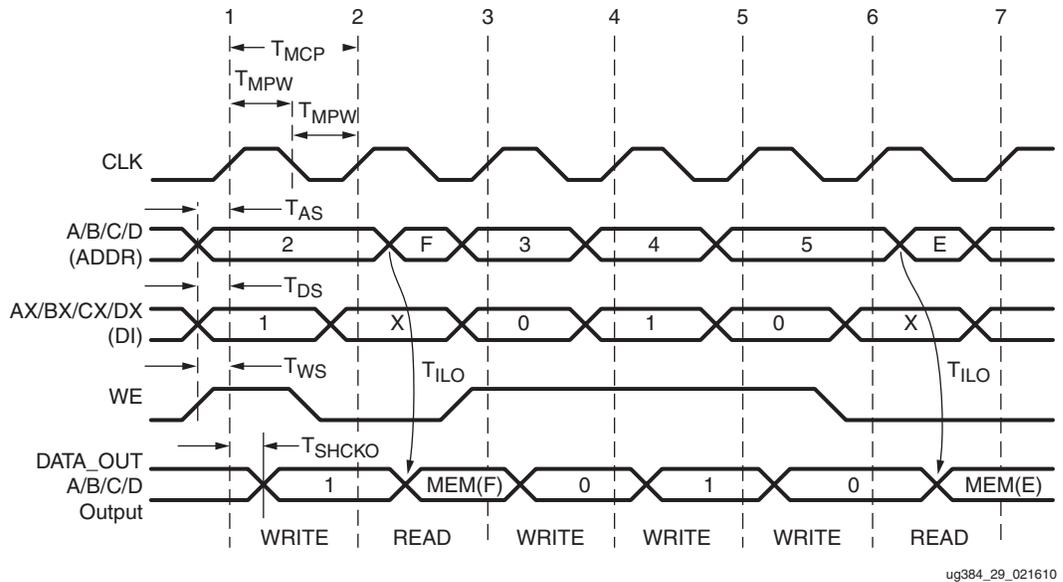

Figure 35: **Slice Distributed RAM Timing Characteristics**

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 output 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} and T_{WOSCO} 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}.

Slice SRL Timing Model (SLICEM only)

Figure 36 illustrates shift register implementation in a Spartan-6 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.
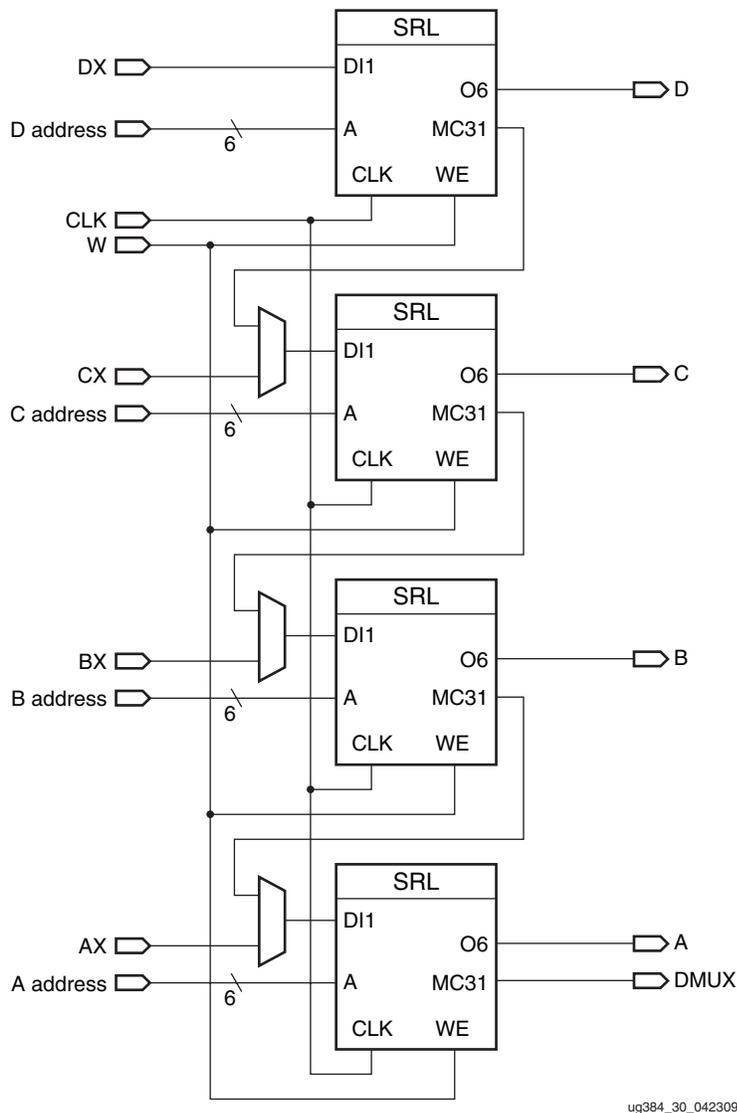

Figure 36:   **Simplified Spartan-6 FPGA Slice SRL**

Slice SRL Timing Parameters

Table 11 shows the SLICEM SRL timing parameters for a majority of the paths in Figure 36.

Table 11: **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_MUX}[1]	CLK to AMUX - DMUX output	Time after the CLK of a write operation that the data written to the SRL is stable on the DMUX output of the slice.
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 SRL[2]		
T_{WS}/T_{WH}	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]	AX/BX/CX/DX configured as data input (DI)	Time before the clock that the data must be stable at the AX/BX/CX/DX input of the slice (configured as an SRL).

Notes:
1. This parameter includes a LUT configured as a two-bit shift register.
2. T_{XXCK} = Setup Time (before clock edge), and T_{CKXX} = Hold Time (after clock edge).
3. Parameter includes AI/BI/CI/DI configured as a data input (DI2) or two bits with a common shift.

Slice SRL Timing Characteristics

Figure 37 illustrates the timing characteristics of a 16-bit shift register implemented in a Spartan-6 FPGA slice (a LUT configured as an SRL).
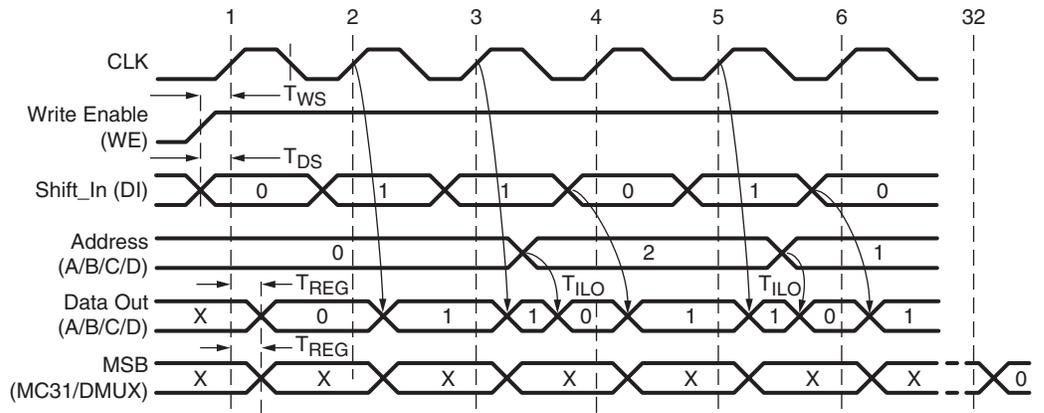
ug384_31_042309

Figure 37: **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. Since the 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. Since the 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} and T_{WOSCO} after clock event 1.

Slice Carry-Chain Timing Model (SLICEM and SLICEL only)

Figure 26, page 34 illustrates a carry chain in a Spartan-6 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.

Slice Carry-Chain Timing Parameters

Table 12 shows the slice carry-chain timing parameters for a majority of the paths in Figure 26, page 34.

Table 12: **Slice Carry-Chain Timing Parameters**

Parameter	Function	Description
Sequential Delays for Slice LUT Configured as Carry Chain		
$T_{AXCY}/T_{BXCY}/T_{CXCY}/T_{DXCY}$	AX/BX/CX/DX input 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 input to COUT output	Propagation delay from the A/B/C/D inputs of the slice to the COUT output of the slice.
$T_{CINA}/T_{CINB}/T_{CINC}/T_{CIND}$	CIN input to AMUX/BMUX/CMUX/DMUX output	Propagation delay from the CIN input of the slice to AMUX/BMUX/CMUX/DMUX output of the slice using XOR (sum).
Setup and Hold Times for a Slice LUT Configured as a Carry Chain[1]		
T_{CINCK}/T_{CKCIN}	CIN Data inputs	Time before the CLK that data from the CIN input of the slice must be stable at the D input of the slice sequential elements (configured as a flip-flop).

Notes:

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

Slice Carry-Chain Timing Characteristics

Figure 38 illustrates the timing characteristics of a slice carry chain implemented in a Spartan-6 FPGA slice.
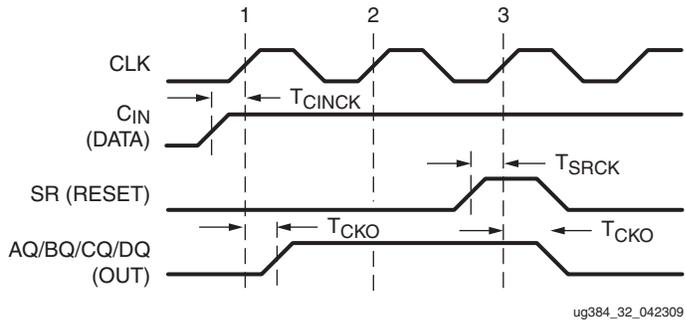

ug384_32_042309

Figure 38: **Slice Carry-Chain Timing Characteristics**

• At time T_{CINCK} before clock event 1, data from CIN input becomes valid-high at the D 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.

• 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 any of the AQ/BQ/CQ/DQ pins at time T_{CKO} after clock event 3.

CLB Primitives

More information on the CLB primitives are available in the software libraries guide.

Distributed RAM Primitives

Seven primitives are available; from 32 x 2 bits to 256 x 1 bit. Three primitives are single-port RAM, two primitives are dual-port RAM, and two primitives are quad-port RAM, as shown in Table 13.

Table 13: **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)
RAM32X1D	32-bit	Dual-port	A[4:0] (read/write) DPRA[4:0] (read)
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 1-bit wide (with the exception of the 32-bit RAM).

Figure 39 shows generic single-port, dual-port, and quad-port distributed RAM primitives. The A, ADDR, and DPRA signals are address buses.
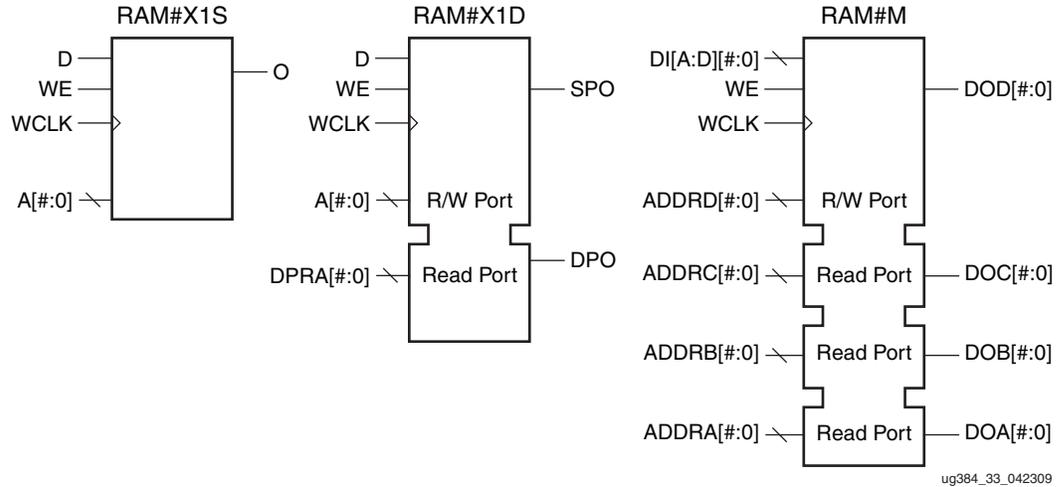
Figure 39: **Single-Port, Dual-Port, and Quad-Port Distributed RAM Primitives**

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.

Enable – WE/WED

The enable pin affects the write functionality of the port. An active 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 13 summarizes the function of each address pins.

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.

Inverting Clock Pins

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

Global Set/Reset – GSR

The global set/reset (GSR) signal does not affect distributed RAM modules.

Shift Registers (SRLs) Primitive

One primitive is available for the 32-bit shift register (SRLC32E). Figure 40 shows the 32-bit shift register primitive.
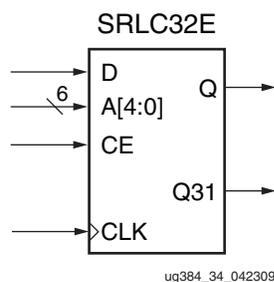

Figure 40: **32-bit Shift Register**

Instantiating several 32-bit shift register with dedicated multiplexers (F7AMUX, F7BMUX, and F8MUX) allows a cascadable shift register chain of up to 128-bit in a slice. Figure 20 through Figure 22 in the *Shift Registers (SLICEM only)* section of this document illustrate the various implementation of cascadable shift registers greater than 32 bits.

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.

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). It 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 (optional)

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

Inverting Clock Pins

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

Global Set/Reset – GSR

The global set/reset (GSR) signal does not affect the shift registers.

Other 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 41.
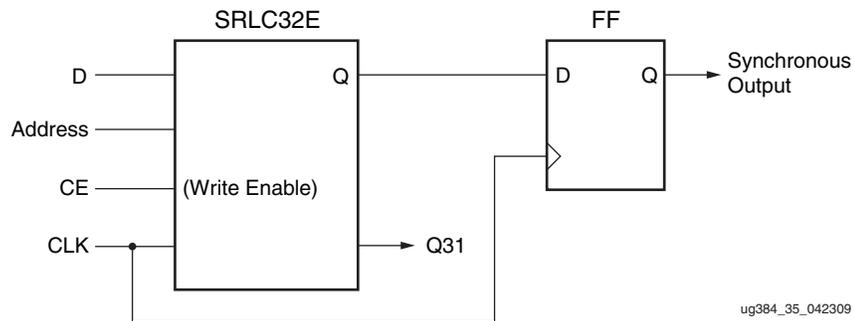

Figure 41: **Synchronous Shift Register**

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 42 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).
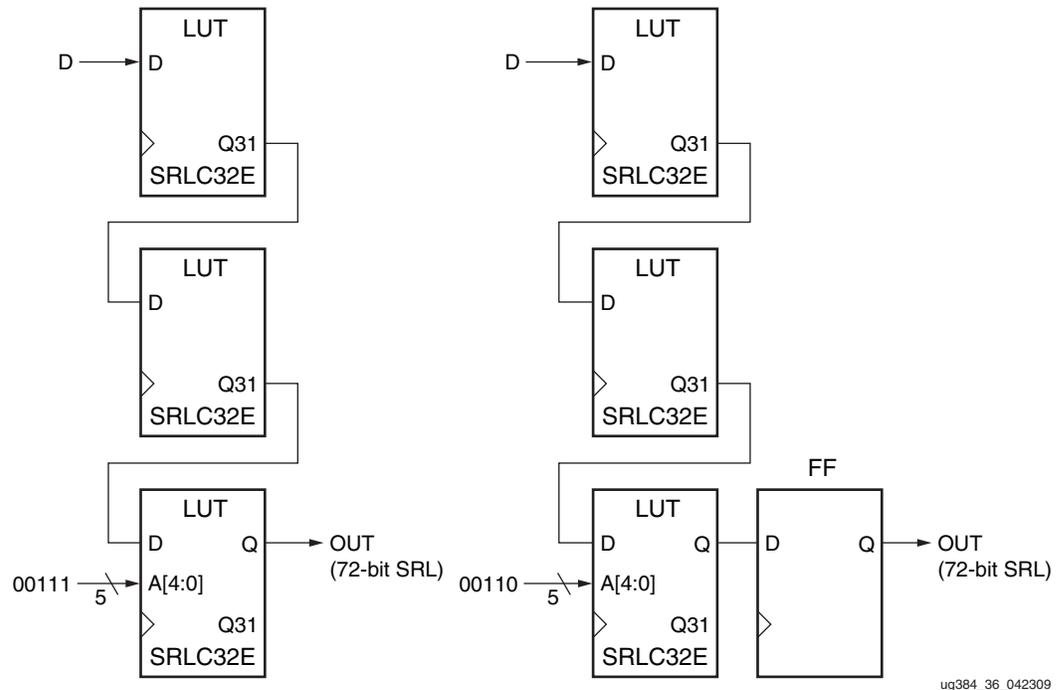
Figure 42: **Example Static-Length Shift Register**

Multiplexer Primitives

Two primitives (MUXF7 and MUXF8) are available for access to the dedicated F7AMUX, F7BMUX and F8MUX in each SLICEM or SLICEL. Combined with LUTs, these multiplexer primitives are also used to build larger width multiplexers (from 8:1 to 16:1). The Designing Large Multiplexers section provides more information on building larger multiplexers.

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 represents the fast carry logic for a SLICEM or SLICEL in the Spartan-6 architecture. This primitive works in conjunction with LUTs in order to build adders and multipliers. This primitive is generally inferred by synthesis tools from standard RTL code. The synthesis tool can identify the arithmetic and/or logic functionality that best maps to this logic in terms of performance and area. It also automatically uses and connects this function properly. Figure 26, page 34 illustrates the CARRY4 block diagram.

Port Signals

Sum Outputs – O[3:0]

The sum outputs provide the final result of the addition/subtraction.

Carry Outputs – CO[3:0]

The carry outputs provide the carry out for each bit. A longer carry chain can be created if CO[3] is connected to CI input of another CARRY4 primitive.

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).

Carry In – CI

The carry in input is used to cascade slices to form longer carry chain. To create a longer carry chain, the CO[3] output of another CARRY4 is simply connected to this pin.