XAPP465 (v1.0) April 10, 2003

Using Look-Up Tables as Shift Registers (SRL16) in Spartan-3 Devices

Summary

The SRL16 is an alternative mode for the look-up tables where they are used as 16-bit shift registers. Using this Shift Register LUT (SRL) mode can improve performance and rapidly lead to cost savings of an order of magnitude. Although the SRL16 can be automatically inferred by the software tools, considering their effective use can lead to more cost-effective designs.

Introduction

Spartan™-3 FPGAs can configure the Look-Up Table (LUT) in a SLICEM slice as a 16-bit shift register without using the flip-flops available in each slice. Shift-in operations are synchronous with the clock, and output length is dynamically selectable. A separate dedicated output allows the cascading of any number of 16-bit shift registers to create whatever size shift register is needed. Each CLB resource can be configured using four of the eight LUTs as a 64-bit shift register.

This document provides generic VHDL and Verilog submodules and reference code examples for implementing from 16-bit up to 64-bit shift registers. These submodules are built from 16-bit shift-register primitives and from dedicated MUXF5, MUXF6, and MUXF7 multiplexers.

These shift registers enable the development of efficient designs for applications that require delay or latency compensation. Shift registers are also useful in synchronous FIFO and Content-Addressable Memory (CAM) designs. To quickly generate a Spartan-3 shift register without using flip-flops (i.e., using the SRL16 element(s)), use the CORE Generator RAM-based Shift Register module.

Shift Register Architecture

The structure of the SRL16 will be described from the bottom up, starting with the shift register and then building up to the surrounding FPGA structure.

LUT Structure

The Look-Up Table can be described as a 16:1 multiplexer with the four inputs serving as binary select lines, and the values programmed into the Look-Up Table serving as the data being selected (see Figure 1).

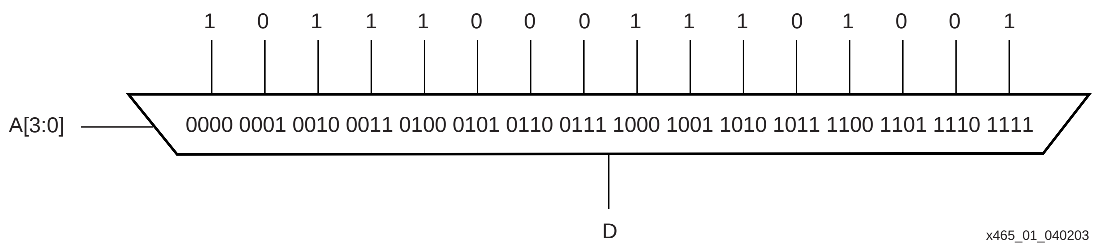

Figure 1: **LUT Modeled as a 16:1 Multiplexer**

With the SRL16 configuration, the fixed LUT values are configured instead as an addressable shift register (see Figure 2). The shift register inputs are the same as those for the synchronous RAM configuration of the LUT: a data input, clock, and clock enable (not shown). A special output for the shift register is provided from the last flip-flop, called Q15 on the library primitives or MC15 in the FPGA Editor. The LUT inputs asynchronously (or dynamically) select one of the 16 storage elements in the shift register.

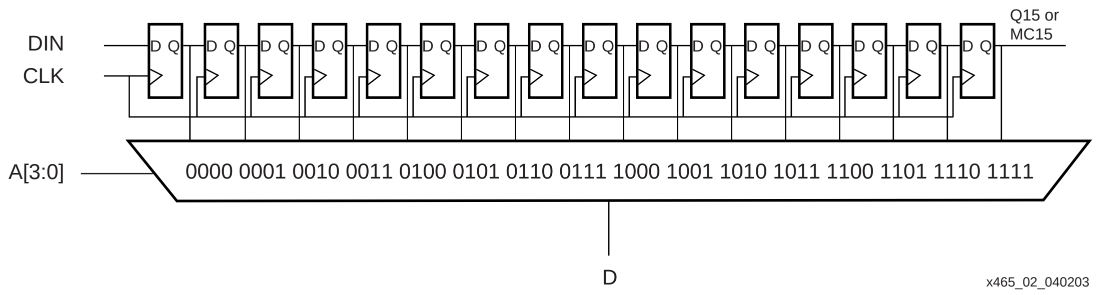

Figure 2: **LUT Configured as an Addressable Shift Register**

Dynamic Length Adjustment

The address can be thought of as dynamically changing the length of the shift register. If D is used as the shift register output instead of Q15, setting the address to 7 (0111) selects Q7 as the output, emulating an 8-bit shift register. Note that since the address lines control the mux, they provide an asynchronous path to the output.

Logic Cell Structure

Each SRL16 LUT has an associated flip-flop that makes up the overall logic cell. The addressable bit of the shift register can be stored in the flip-flop for a synchronous output or can be fed directly to a combinatorial output of the CLB. When using the register, it is best to have fixed address lines selecting a static shift register length. Since the clock-to-output delay of the flip-flop is faster than the shift register, performance can be improved by addressing the second-to-last bit and then using the flip-flop as the last stage of the shift register. Using the flip-flop also allows for asynchronous or synchronous set or reset of the output.

The shift register input can come from a dedicated SHIFTIN signal, and the Q15/MC15 signal from the last stage of the shift register can drive a SHIFTOUT output. The addressable D output is available in all SRL primitives, while the Q15/MC15 signal that can drive SHIFTOUT is only available in the cascadable SRLC16 primitive.

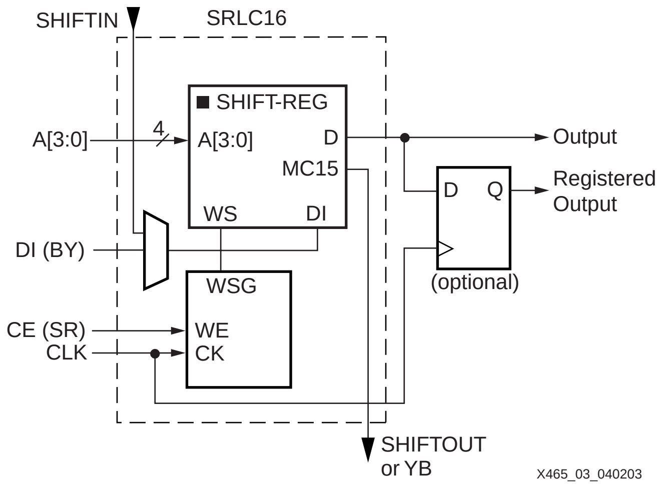

Figure 3: **Logic Cell SRL Structure**

Slice Structure

The two logic cells within a slice are connected via the SHIFTOUT and SHIFTIN signals for cascading a shift register up to 32 bits (see Figure 4). These connect the Q15/MC15 of the first shift register to the DI (or Q0 flip-flop) of the second shift register.

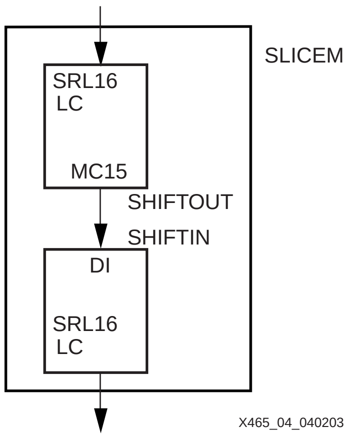

Figure 4: **Shift Register Connections Between Logic Cells in a Slice**

If dynamic addressing (or "dynamic length adjustment") is desired, the two separate data outputs from each SRL16 must be multiplexed together. One of the two SRL16 bits can be selected by using the F5MUX to make the selection (see Figure 5).

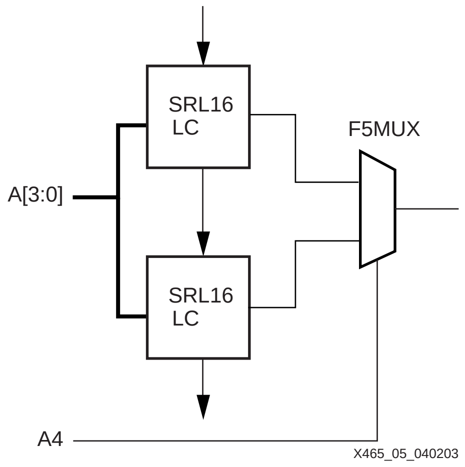

Figure 5: **Using F5MUX for Addressing Multiple SRL16 Components**

CLB Structure

The Spartan-3 CLB contains four slices, each with two Look-Up Tables, but only two allow LUTs to be used as SRL16 components or distributed RAM. The two left-hand SLICEM components allow their two LUTs to be configured as a 16-bit shift register. The same cascading of SHIFTOUT to SHIFTIN available between the LUTs in the SLICEM is also available to connect the two SLICEM components. The four left-hand LUTs of a single CLB can be combined to produce delays up to 64 clock cycles (see Figure 6).

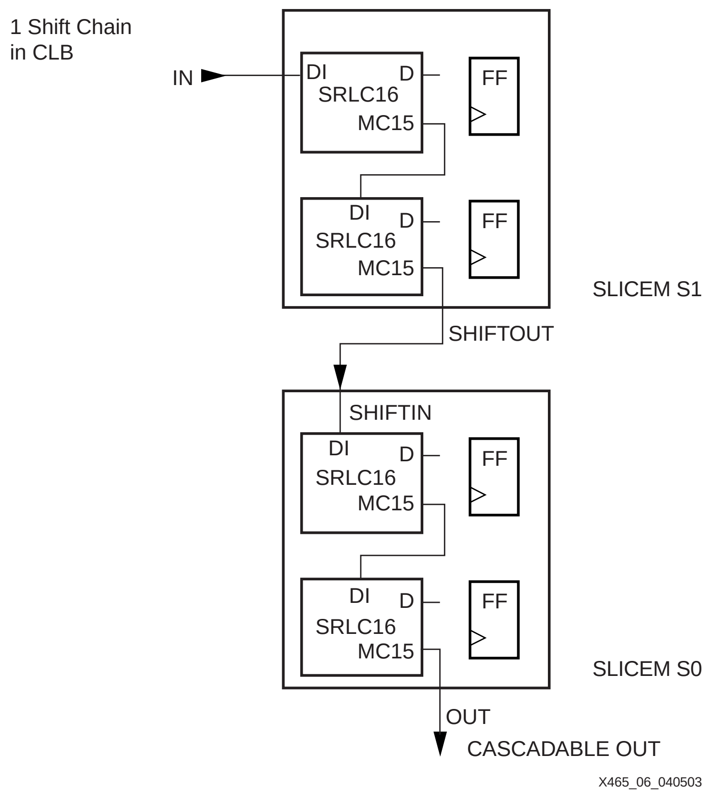

Figure 6: **Cascading Shift Register LUTs in a CLB**

The multiplexers can be used to address multiple SLICEMs similar to the description for combining the two LUTs within a SLICEM. The F6MUX can be used to select from three or four SRL16 components in a CLB, providing up to 64 bits of addressable shift register (see Figure 7).

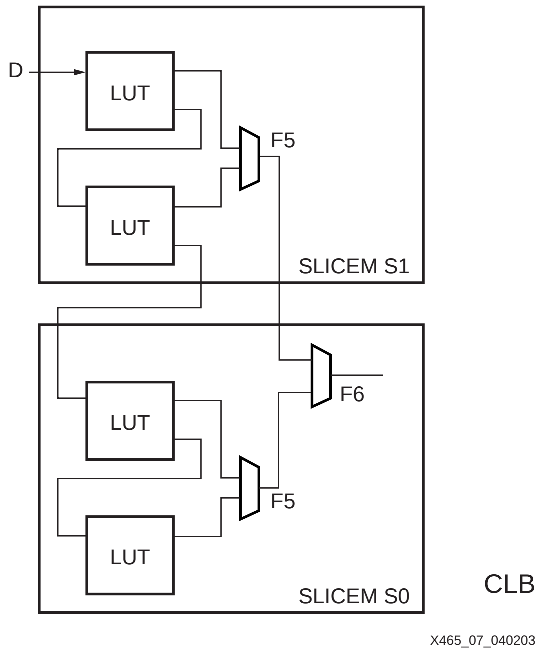

Figure 7: **Using F6MUX to Address a 64-Bit Shift Register**

Library Primitives

Eight library primitives are available that offer optional clock enable (CE), inverted clock ($\overline{\text{CLK}}$) and cascadable output (Q15) combinations.

Table 1 lists all of the available primitives for synthesis and simulation.

Table 1: **Shift Register Primitives**

Primitive	Length	Control	Address Inputs	Output
SRL16	16 bits	CLK	A3, A2, A1, A0	Q
SRL16E	16 bits	CLK, CE	A3, A2, A1, A0	Q
SRL16_1	16 bits	CLK	A3, A2, A1, A0	Q
SRL16E_1	16 bits	$\overline{\text{CLK}}$, CE	A3, A2, A1, A0	Q
SRLC16	16 bits	CLK	A3, A2, A1, A0	Q, Q15
SRLC16E	16 bits	CLK, CE	A3, A2, A1, A0	Q, Q15
SRLC16_1	16 bits	CLK	A3, A2, A1, A0	Q, Q15
SRLC16E_1	16 bits	$\overline{\text{CLK}}$, CE	A3, A2, A1, A0	Q, Q15

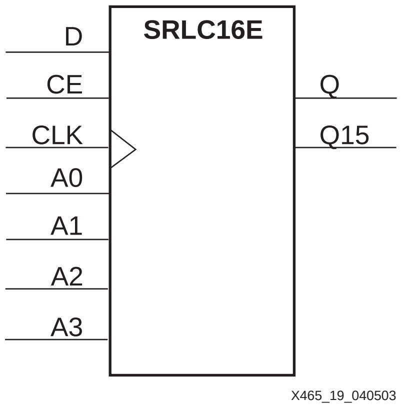

Figure 8: **SRLC16E Primitive**

Initialization in VHDL and Verilog Code

A shift register can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the INIT attribute is attached to the 16-bit shift register instantiation and is copied in the EDIF output file to be compiled by Xilinx Alliance Series tools. The VHDL code simulation uses a `generic` parameter to pass the attributes. The Verilog code simulation uses a `defparam` parameter to pass the attributes.

The S3_SRL16E shift register instantiation code examples (in VHDL and Verilog) illustrate these techniques (see "VHDL and Verilog Templates," page 12). `S3_SRL16E.vhd` and `.v` files are not a part of the documentation.

Port Signals

Clock — CLK

Either the rising edge or the falling edge of the clock is used for the synchronous shift-in. The data and clock enable input pins have set-up 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 (optional)

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 (Q15).

Address — A3, A2, A1, A0

Address inputs select the bit (range 0 to 15) to be read. The nth bit is available on the output pin (Q). Address inputs have no effect on the cascadable output pin (Q15), which is always the last bit of the shift register (bit 15).

Data Out — Q

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

Data Out — Q15 (optional)

The data output Q15 provides the last bit value of the 16-bit shift register. New data becomes available after each shift-in operation.

Inverting Control Pins

The two control pins (CLK, CE) have an individual inversion option. The default is the rising clock edge and active High clock enable.

GSR

The global set/reset (GSR) signal has no impact on shift registers.

Attributes

Content Initialization — INIT

The INIT attribute defines the initial shift register contents. The INIT attribute is a hex-encoded bit vector with four digits (0000).The left-most hexadecimal digit is the most significant bit. By default the shift register is initialized with all zeros during the device configuration sequence, but any other configuration value can be specified.

Location Constraints

Figure 9 shows how the slices are arranged within a CLB. Each CLB has four slices, but only the two at the bottom-left of the CLB can be used as shift registers. These are both designated SLICEM in CLB positions S0 and S1. The relative position coordinates are X0Y0 and X0Y1. To constrain placement, these coordinates can be used in a LOC property attached to the SRL primitive. Note that the dedicated CLB shift chain runs from the top to the bottom, but the start and end of the shift register can be in any of the four SLICEM LUTs.

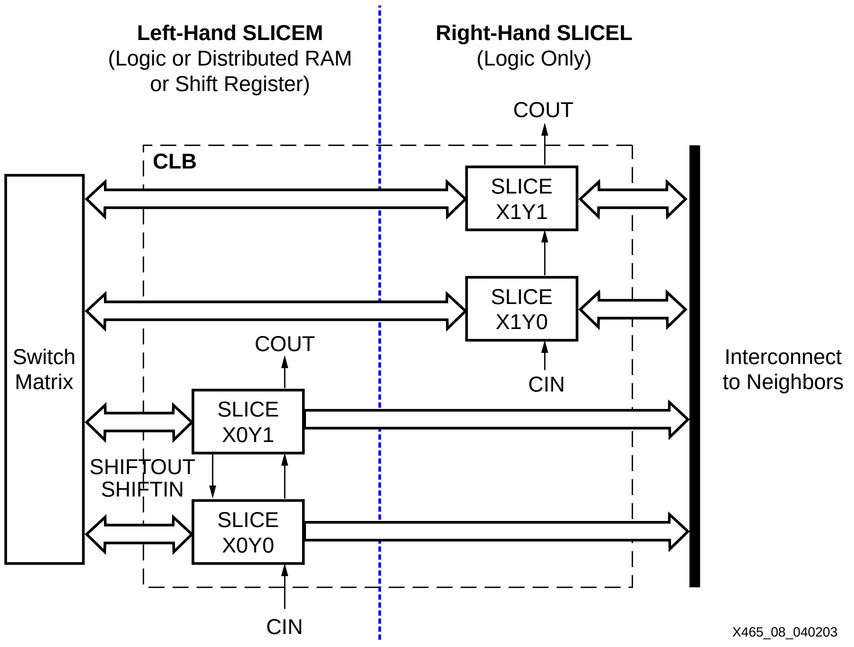

Figure 9: **Arrangement of Slices within the CLB**

Shift Register Operations

Data Flow

Each shift register (SRL16 primitive) supports:

- Synchronous shift-in
- Asynchronous 1-bit output when the address is changed dynamically
- Synchronous shift-out when the address is fixed

In addition, cascadable shift registers (SRLC16) support synchronous shift-out output of the last (16th) bit. This output has a dedicated connection to the input of the next SRLC16 inside the CLB resource. Two primitives are illustrated in Figure 10.

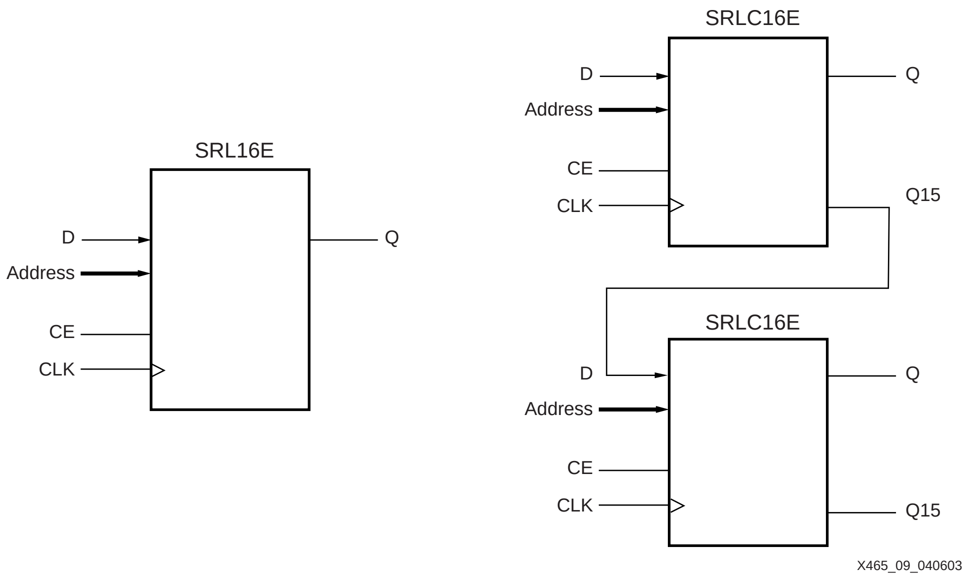

Figure 10: **Shift Register and Cascadable Shift Register**

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, and each bit is shifted to the next highest bit position. In a cascadable shift register configuration (such as SRLC16), the last bit is shifted out on the Q15 output.

The bit selected by the 4-bit address appears on the Q output.

Dynamic Read Operation

The Q output is determined by the 4-bit address. Each time a new address is applied to the 4-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.

Figure 11 illustrates the shift and dynamic read operations.

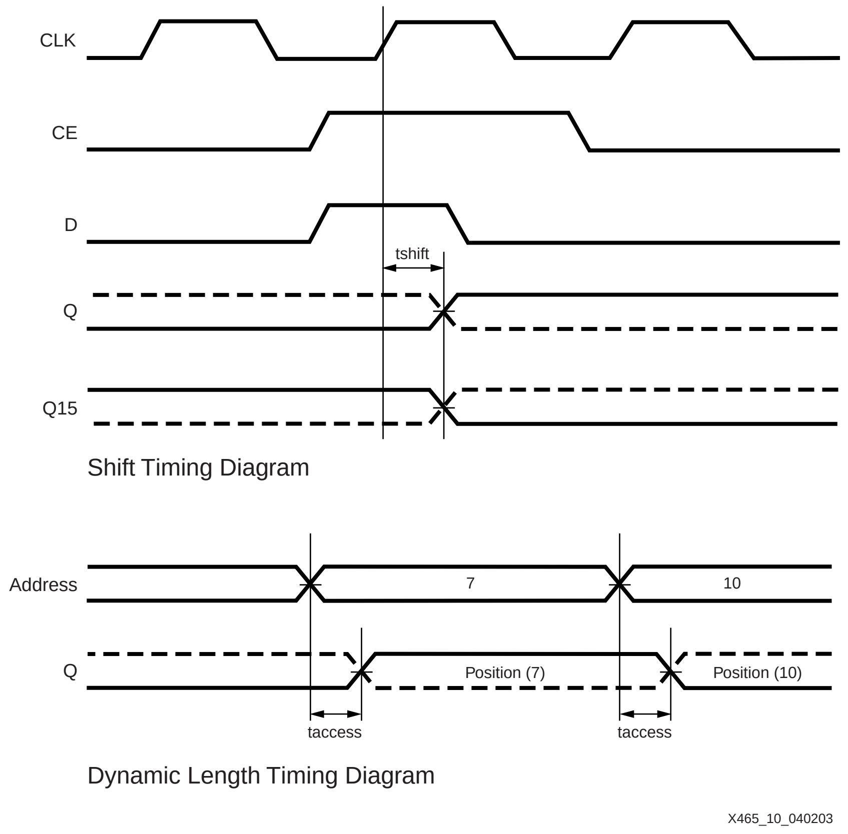

Figure 11: **Shift- and Dynamic-Length Timing Diagrams**

Static Read Operation

If the 4-bit address is fixed, the Q output always uses the same bit position. This mode implements any shift register length up 1 to 16 bits in one LUT. Shift register length is (N+1) where N is the input address.

The Q output changes synchronously with each shift operation. The previous bit is shifted to the next position and appears on the Q output.

Characteristics

- 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 Q15 output always contains the last bit value.
- The Q15 output changes synchronously after each shift operation.

Shift Register Inference

When a shift register is described in generic HDL code, synthesis tools will infer the use of the SRL16 component. Note that since the SRL16 does not have either synchronous or asynchronous set or reset inputs, and does not have access to all bits at the same time, using such capabilities will preclude the use of the SRL16, and the function will be implemented in flip-flops. The cascadable shift register (SRLC16) may be inferred if the shift register is larger than 16 bits or if only the Q15 is used.

Although the SRL16 shift register does not have a parallel load capability, an equivalent function can be implemented simply by anticipating the load requirement and shifting in the proper data. This requires predictable timing for the load command.

VHDL Inference Code

The following code infers an SRL16 in VHDL.

```
architecture Behavioral of srl16 is

signal Q_INT: std_logic_vector(15 downto 0);

begin

process(C)
begin
  if (C’event and C=’1’) then
    Q_INT <= Q_INT(14 downto 0) & D;
  end if;
end process;

Q <= Q_INT(15);

end Behavioral;
```

An inverted clock (SRL16_1) is inferred by replacing `C='1'` with `C='0'`. A clock enable (SRL16E) is inferred by inserting `if (CE='1') then` after the first if-then statement.

Verilog Inference Code

The following code infers an SRL16 in Verilog.

```
always @ (posedge C)
begin
  Q_INT <= {Q_INT[14:0],D};
end

always @(Q_INT)
begin
  Q <= Q_INT[15];
end
```

An inverted clock (SRL16_1) is inferred by replacing (`posedge C`) with (`negedge C`). A clock enable (SRL16E) is inferred by inserting `if(CE)` after the begin statement.

Shift Register Submodules

In addition to the 16-bit primitives, two submodules that implement 32-bit and 64-bit cascadable shift registers are provided in VHDL and Verilog code. Table 2 lists available submodules.

Table 2: **Shift Register Submodules**

Submodule	Length	Control	Address Inputs	Output
SRLC32E_SUBM	32 bits	CLK, CE	A4, A3, A2, A1, A0	Q, Q31
SRLC64E_SUBM	64 bits	CLK, CE	A5, A4, A3, A2, A1, A0	Q, Q63

The submodules are based on SRLC16E primitives, which are associated with dedicated multiplexers (MUXF5, MUXF6, and so forth). This implementation allows a fast static- and dynamic-length mode, even for very large shift registers.

Figure 12 represents the cascadable shift registers (32-bit and 64-bit) implemented by the submodules in Table 2.

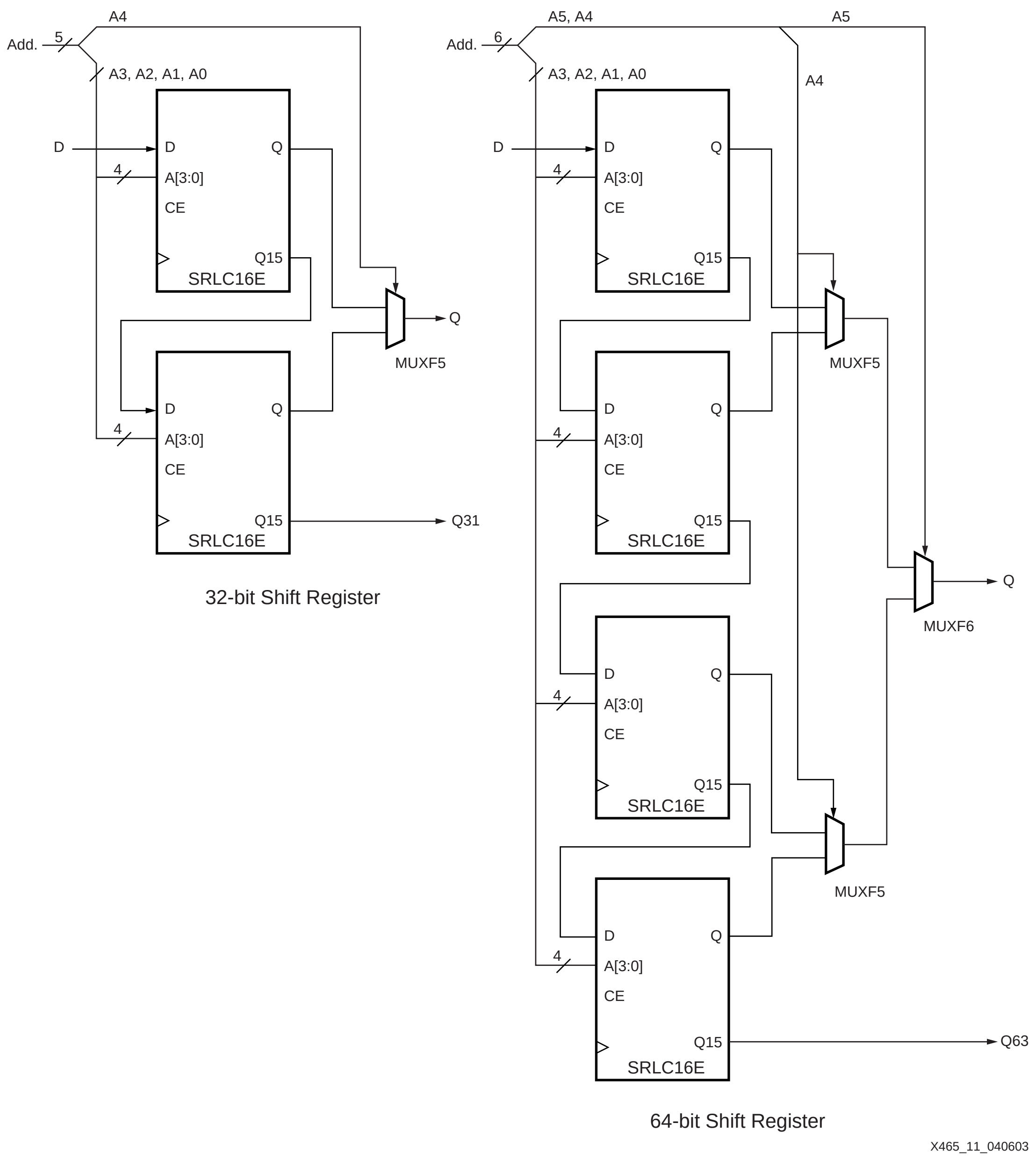

Figure 12: **Shift-Register Submodules (32-bit, 64-bit)**

All clock enable (CE) and clock (CLK) inputs are connected to one global clock enable and one clock signal per submodule. If a global static- or dynamic-length mode is not required, the SRLC16E primitive can be cascaded without multiplexers.

Fully Synchronous Shift Registers

All shift-register primitives and submodules do not use the register(s) available in the same slice(s). 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 13.

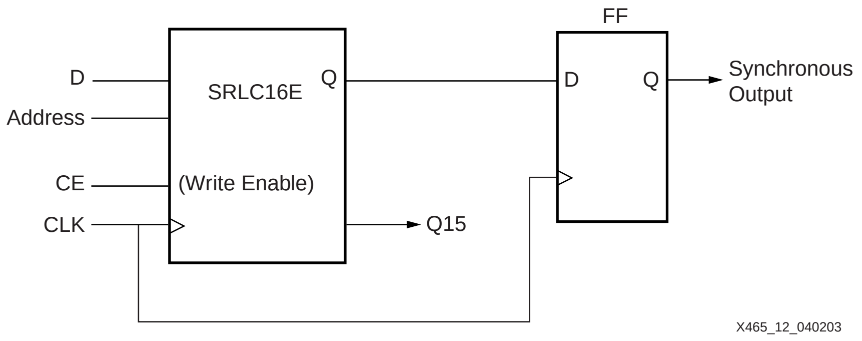

Figure 13: **Fully Synchronous Shift Register**

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 16-bit shift register implements any static length mode shift register without the dedicated multiplexers (MUXF5, MUXF6, and so on). Figure 14 illustrates a 40-bit shift register. Only the last SRLC16E primitive needs to have its address inputs tied to “0111”. Alternatively, shift register length can be limited to 39 bits (address tied to “0110”) and a flip-flop can be used as the last register. (In an SRLC16E primitive, the shift register length is the address input + 1.)

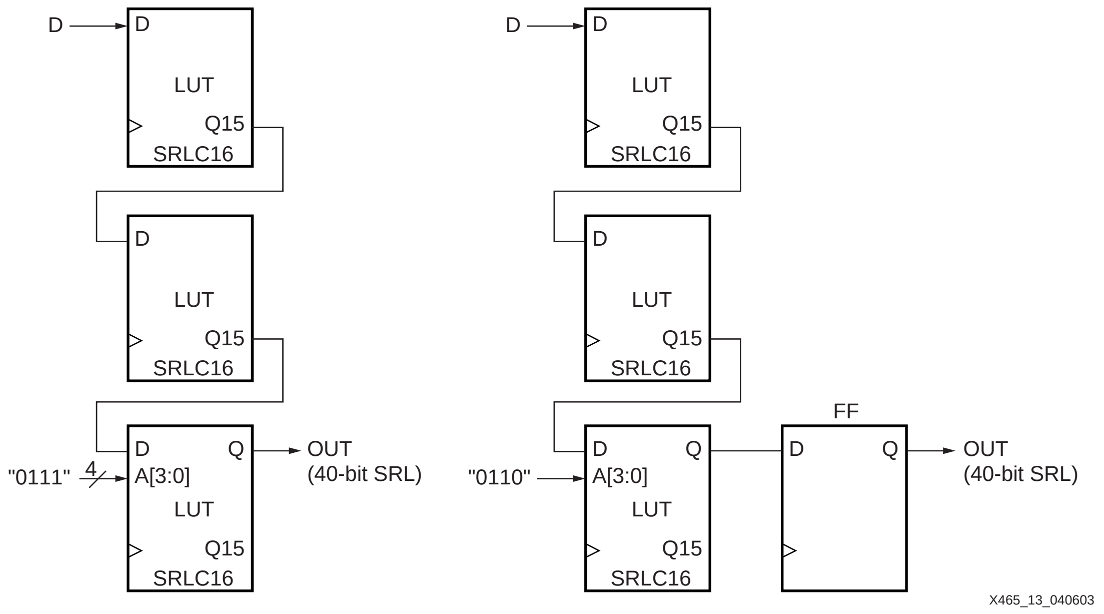

Figure 14: **40-bit Static-Length Shift Register**

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available for all primitives and submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The ShiftRegister_C_x (with x = 16, 32, or 64) templates are cascadable modules and instantiate the corresponding SRLCxE primitive (16) or submodule (32 or 64).

The ShiftRegister_16 template can be used to instantiate an SRL16 primitive.

VHDL and Verilog Templates

In template names, the number indicates the number of bits (for example, SHIFT_SELECT_16 is the template for the 16-bit shift register) and the "C" extension means the template is cascadable.

The following are templates for primitives:

- SHIFT_REGISTER_16
- SHIFT_REGISTER_16_C

The following are templates for submodules:

- SHIFT_REGISTER_32_C (submodule: SRLC32E_SUBM)
- SHIFT_REGISTER_64_C (submodule: SRLC64E_SUBM)

The corresponding submodules have to be synthesized with the design.

Templates for the SHIFT_REGISTER_16_C module are provided in VHDL and Verilog code as an example.

VHDL Template:

```
-- Module: SHIFT_REGISTER_C_16
-- Description: VHDL instantiation template
--  CASCADABLE 16-bit shift register with enable (SRLC16E)
-- Device: Spartan-3 Family
---------------------------------------------------------------------
-- Components Declarations:
--
component SRLC16E
-- pragma translate_off
  generic (
-- Shift Register initialization ("0" by default) for functional
simulation:
        INIT : bit_vector := X"0000"
  );
-- pragma translate_on
  port (
        D : in std_logic;
        CE   : in std_logic;
        CLK  : in std_logic;
        A0   : in std_logic;
        A1   : in std_logic;
        A2   : in std_logic;
        A3   : in std_logic;
        Q    : out std_logic;
        Q15  : out std_logic
  );
end component;
-- Architecture  Section:
--
```

```
-- Attributes for Shift Register initialization (“0” by default):
attribute INIT: string;
--
attribute INIT of U_SRLC16E: label is “0000”;
--
-- ShiftRegister Instantiation
U_SRLC16E: SRLC16E
  port map (
  D       => , -- insert input signal
  CE      => , -- insert Clock Enable signal (optional)
  CLK     => , -- insert Clock signal
  A0      => , -- insert Address 0 signal
  A1      => , -- insert Address 1 signal
  A2      => , -- insert Address 2 signal
  A3      => , -- insert Address 3 signal
  Q       => , -- insert output signal
  Q15     =>   -- insert cascadable output signal
  );
```

Verilog Template:

```
// Module: SHIFT_REGISTER_16
// Description: Verilog instantiation template
// Cascadable 16-bit Shift Register with Clock Enable (SRLC16E)
// Device: Spartan-3 Family
//---------------------------------------------------------------------
// Syntax for Synopsys FPGA Express
// synopsys translate_off

  defparam

//Shift Register initialization ("0" by default) for functional simulation:
  U_SRLC16E.INIT = 16'h0000;
// synopsys translate_on

//SelectShiftRegister-II Instantiation
   SRLC16E U_SRLC16E   ( .D(),
                         .A0(),
                         .A1(),
                         .A2(),
                         .A3(),
                         .CLK(),
                         .CE(),
                         .Q(),
                         .Q15()
          );

// synthesis attribute declarations
  /* synopsys attribute
  INIT "0000"
  */
```

CORE Generator System

The Xilinx CORE Generator™ system generates fast, compact, FIFO-style shift registers, delay lines, or time-skew buffers using the SRL16. The RAM-based Shift Register module shown in Figure 15 provides a very efficient multibit wide shift for widths up to 256 and depths to 1024. Fixed-length shift registers and variable-length shift registers can be created. An option is also provided to register the outputs of the module. If output registering is selected, there are additional options for Clock Enable, Asynchronous Set, Clear, and Init, and Synchronous Set, Clear and Init of the output register. The module can optionally be generated as a relationally placed macro (RPM) or as unplaced logic.

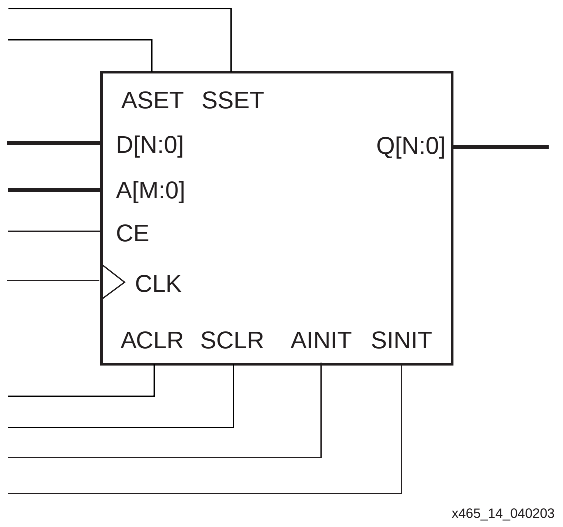

Figure 15: **CORE Generator RAM-Based Shift Register Module**

Applications

Delay Lines

The register-rich nature of the Xilinx FPGA architecture allows for the addition of pipeline stages to increase throughput. Data paths must be balanced to keep the desired functionality. The SRL16 can be used when additional clock cycles of delay are needed anywhere in the design (see Figure 16).

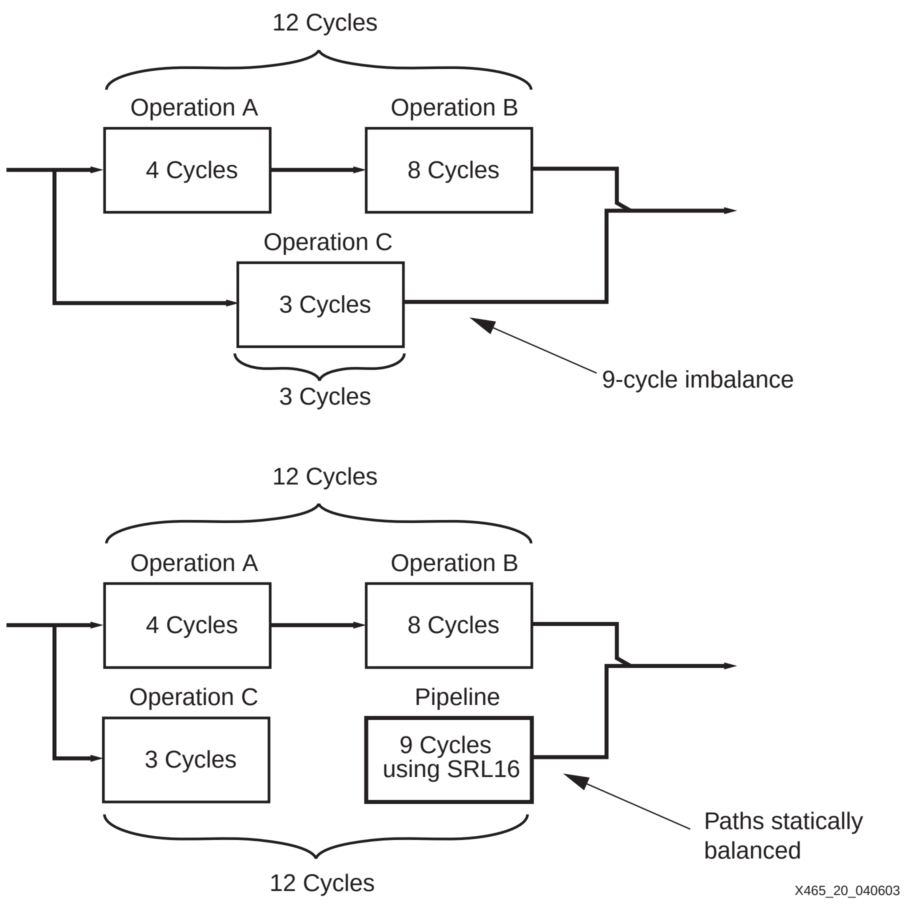

Figure 16: **Using SRL16 as a Delay Line**

Linear Feedback Shift Registers

Linear Feedback Shift Registers (LFSRs) sequence through 2^n-1 states, where n is the number of flip-flops. The sequence is created by feeding specific bits back through an XOR or XNOR gate. LFSRs can replace conventional binary counters in performance critical applications where the count sequence is not important (e.g., FIFOs). LFSRs are also used as pseudo-random number generators. They are important building blocks in encryption and decryption algorithms.

Maximal-length LFSRs need taps taken from specific positions within the shift register. There are multiple ways these taps can be made available in the SRL16 configuration. One is by addressing the necessary bit in a given SRL16 while allowing the Q15 to cascade to the next SRL16. Another is to use flip-flops to "extend" the SRL16 where necessary to access the tap points. For example, Figure 17 shows how a 52-bit LFSR can be implemented in one CLB with the feedback coming from bits 49 and 52. A third method is to duplicate the LFSR in multiple SRLs and address different bits from each one. Yet another is to generate multiple addresses in one SRL clock cycle to capture multiple bit positions. The XNOR gate required for any LFSR can be conveniently located in the SLICEL part of the CLB. More detail is available in XAPP210.

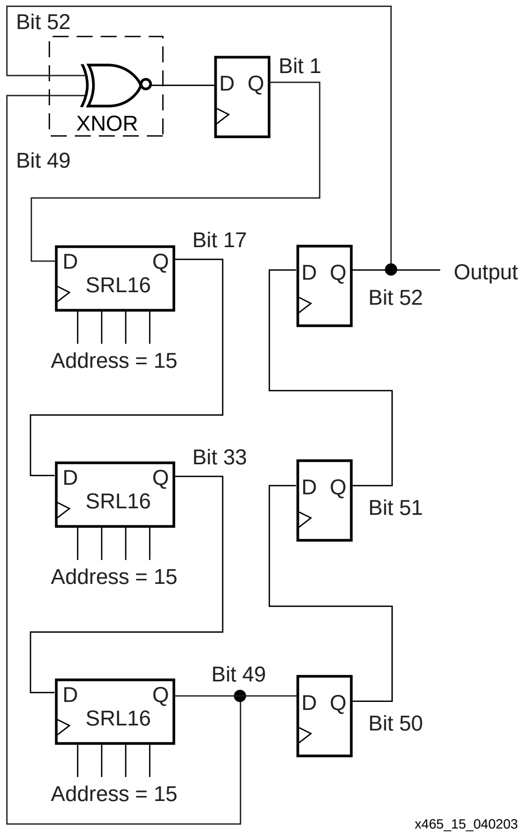

Figure 17: **52-bit LFSR in One CLB**

Gold Code Generator

Gold code generators are used in CDMA systems to generate code sequences with good correlation properties (see Figure 18). R. Gold suggested that sets of small correlation codes could be generated by modulo 2 addition of the results of two LFSRs, primed with factor codes. The result is a set of codes ideally suited to distinguish one code from another in a spectrum full of coded signals. Figure 18 shows an implementation of a Gold code generator. The logic required to initially fill the LFSR and provide the feedback can be located in the SLICEL parts of the CLB. See XAPP217 for more details.

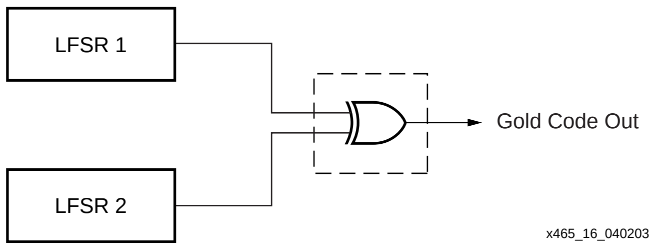

Figure 18: **Gold Code Generator**

FIFOs

Synchronous FIFOs can be built out of the SRL16 components. These are useful when other resources become scarce, providing up to 64 bits per CLB. For larger FIFOs, the block RAM is the most efficient resource to use. See XAPP256 for more detail.

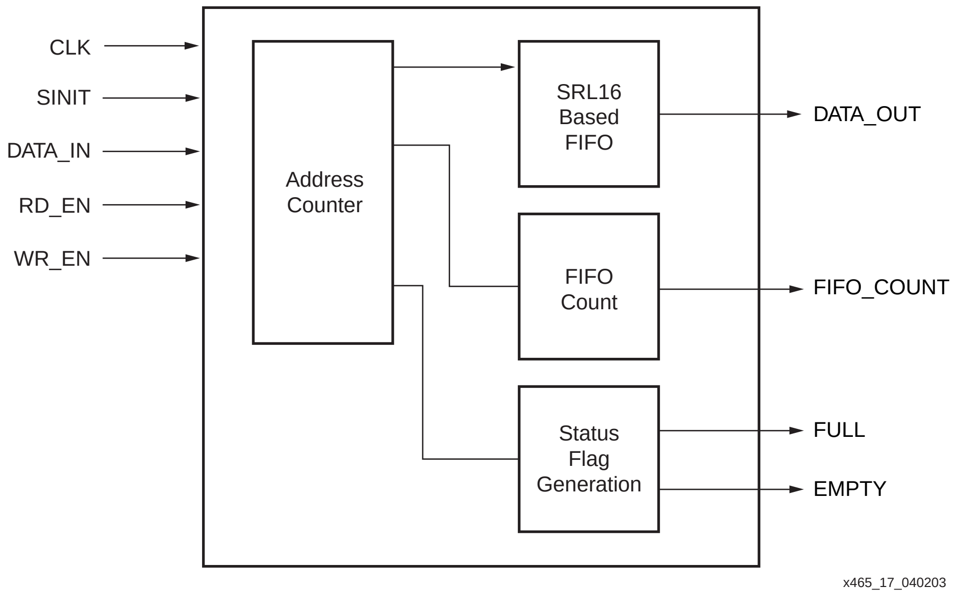

Figure 19: **Synchronous FIFO Using SRLC16 Shift Registers**

Counters

Any desired repeated sequence of 16 states can be achieved by feeding each output with an SRL16. Cascading the SRL16 allows even longer arbitrary count sequences. A terminal count can be generated by using the standard carry chain (see Figure 20).

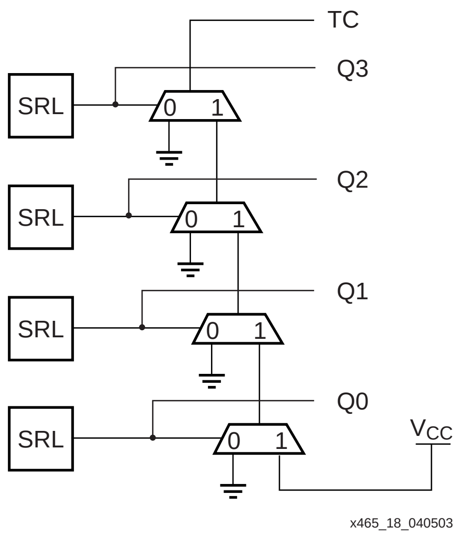

Figure 20: **SRL-Based Counter with Terminal Count**

Related Materials and References

- XAPP210: Linear Feedback Shift Registers in Virtex Devices
  Linear Feedback Shift Registers are very efficient counters in the FPGA architecture. Using the SRL16 as the basis of the shift register, a 15-bit counter can fit in one slice and a 52-bit counter in two slices.
- XAPP211: PN Generators Using the SRL Macro
  Pseudo-random Noise sequences are used to code and spread signals across a wide band of transmission frequencies for spread spectrum modulation. PN generators are based upon LFSRs, which can be effectively built from the SRL16 components.
- XAPP217: Gold Code Generators in Virtex Devices
  A special type of PN sequence is a Gold code generator, which can be created from SRL16-based LFSRs.
- XAPP220: LFSRs as Functional Blocks in Wireless Applications
  Further discussion of the usage of LFSRs such as Gold Code Generators in applications such as CDMA.
- XAPP256: FIFOs Using Virtex-II Shift Registers
  The SRL16 is ideal for building smaller synchronous FIFOs. FIFOs can be built in any width while producing a 1-bit resolution. With cascaded SRL16 shift registers, a flexible depth in multiples of 16 is available. These techniques are useful for even larger FIFOs when block RAM resources are not available.
- TechXclusive: The SRL16E: How Using this Exciting Mode Can Lead to Cost Saving of an Order of Magnitude
  Describes the SRL16 function and its application in pipeline compensation, pseudo random noise generators, serial frame synchronizers, running averages, pulse generation and clock division, pattern generation, state machines, dynamically addressable shift registers, FIFOs, and an RS232 receiver.
- DS228: RAM-Based Shift Register LogiCORE Module
  Generates fast, compact, FIFO-style shift registers, delay lines or time-skew buffers using the SRL16.
- SRL16 Primitives in Libraries Guide
  Describes the usage and functionality of the SRL16 primitive and its variations.

Conclusion

The SRL16 configuration of the Spartan-3 look-up table provides a space-efficient shift register that would otherwise require 16 flip-flops. This feature will be automatically used when a small shift register is described in HDL code. However, creative consideration of the uses of the SRL16 as described here can provide even more significant advantages in many applications.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
04/10/03	1.0	Initial Xilinx release.