

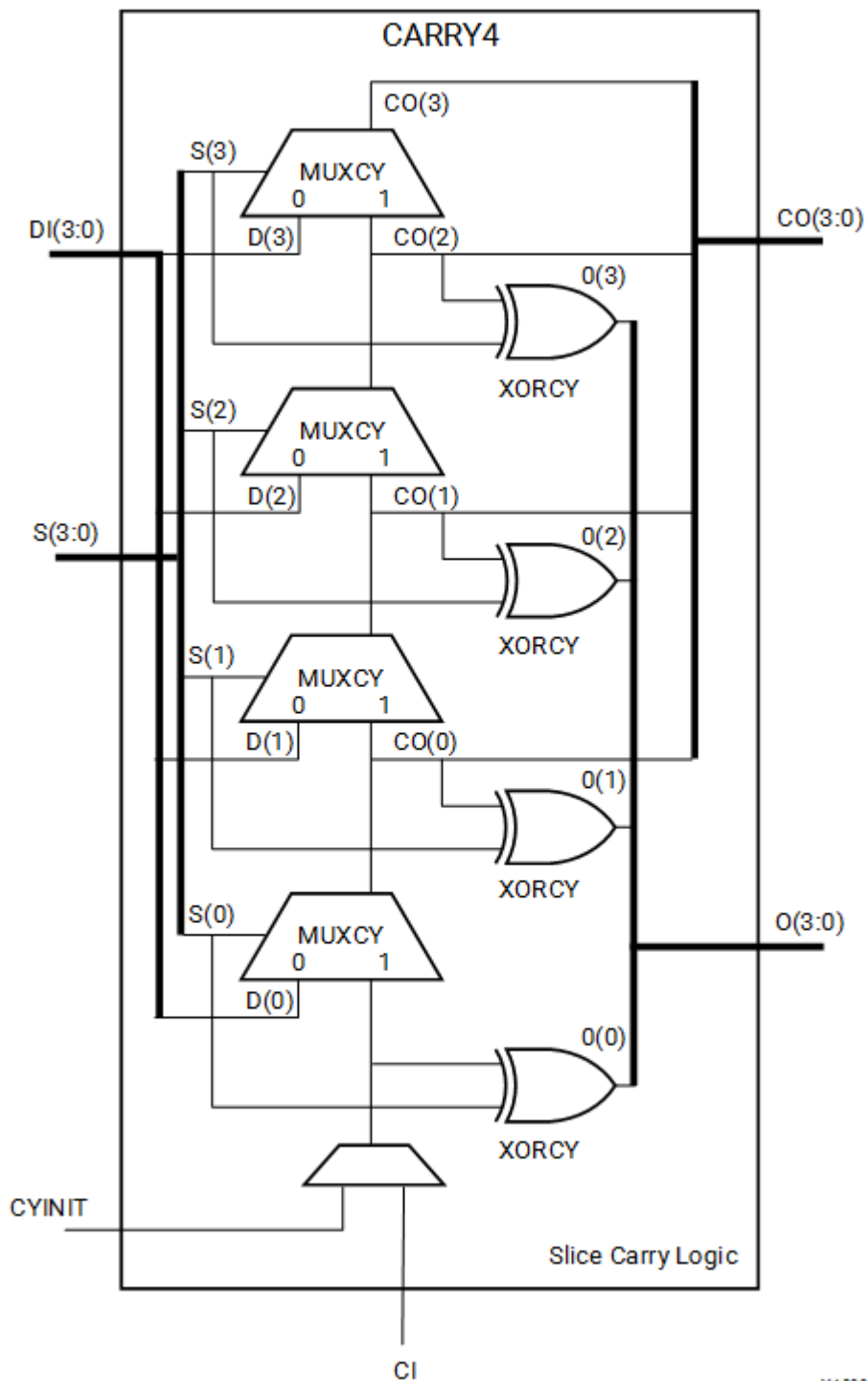


Vivado Design Suite 7 Series FPGA and Zynq 7000 SoC Libraries Guide (UG953)

CARRY4
CFGLUT5
LUT1
LUT2
LUT3
LUT4
LUT5
LUT6
LUT6_2
MUXF7
MUXF8
SRL16E
SRLC32E

CARRY4

Primitive: Fast Carry Logic with Look Ahead



X10937

Introduction

This circuit design represents the fast carry logic for a slice. The carry chain consists of a series of four MUXes and four XORs that connect to the other logic

(LUTs) in the slice via dedicated routes to form more complex functions. The fast carry logic is useful for building arithmetic functions like adders, counters, subtractors and add/subs, as well as such other logic functions as wide comparators, address decoders, and some logic gates (specifically, AND and OR).

Port Descriptions

Port	Direction	Width	Function
O	Output	4	Carry chain XOR general data out.
CO	Output	4	Carry-out of each stage of the carry chain.
DI	Input	4	Carry-MUX data input.
S	Input	4	Carry-MUX select line.
CYINIT	Input	1	Carry-in initialization input.
CI	Input	1	Carry cascade input.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;  
use UNISIM.vcomponents.all;
```

```

-- CARRY4: Fast Carry Logic Component
--           7 Series
-- Xilinx HDL Language Template, version 2025.1

CARRY4_inst : CARRY4
port map (
    CO => CO,          -- 4-bit carry out
    O => O,            -- 4-bit carry chain XOR data out
    CI => CI,          -- 1-bit carry cascade input
    CYINIT => CYINIT,  -- 1-bit carry initialization
    DI => DI,          -- 4-bit carry-MUX data in
    S => S             -- 4-bit carry-MUX select input
);

-- End of CARRY4_inst instantiation

```

Verilog Instantiation Template

```

// CARRY4: Fast Carry Logic Component
//           7 Series
// Xilinx HDL Language Template, version 2025.1

CARRY4 CARRY4_inst (
    .CO(CO),          // 4-bit carry out
    .O(O),            // 4-bit carry chain XOR data out
    .CI(CI),          // 1-bit carry cascade input
    .CYINIT(CYINIT), // 1-bit carry initialization
    .DI(DI),          // 4-bit carry-MUX data in
    .S(S)             // 4-bit carry-MUX select input
);

// End of CARRY4_inst instantiation

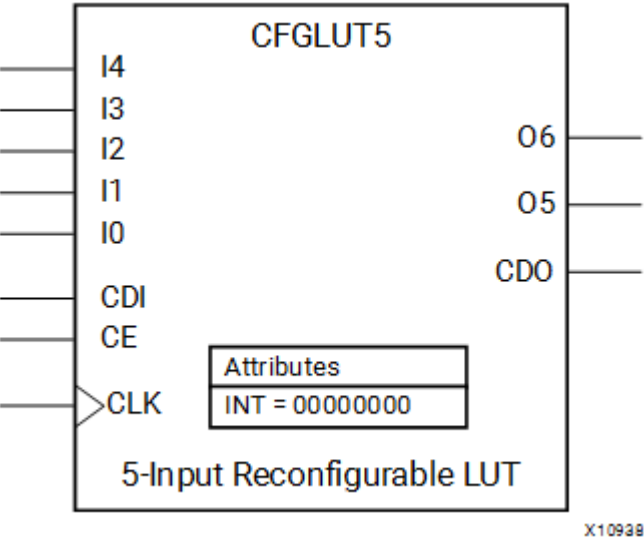
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

CFGLUT5

Primitive: 5-input Dynamically Reconfigurable Look-Up Table (LUT)



Introduction

This element is a runtime, dynamically reconfigurable, 5-input look-up table (LUT) that enables the changing of the logical function of the LUT during circuit operation. Using the CDI pin, a new INIT value can be synchronously shifted in serially to change the logical function. The O6 output pin produces the logical output function, based on the current INIT value loaded into the LUT and the currently selected I0-I4 input pins. Optionally, you can use the O5 output in combination with the O6 output to create two individual 4-input functions sharing the same inputs or a 5-input function and a 4-input function that uses a subset of the 5-input logic (see the following tables). This component occupies one of the four LUT6 components within a Slice-M.

To cascade this element, connect the CDO pin from each element to the CDI input of the next element. This will allow a single serial chain of data (32-bits per LUT) to reconfigure multiple LUTs.

Port Descriptions

Port	Direction	Width	Function
O6	Output	1	5-LUT output.

Port	Direction	Width	Function
O5	Output	1	4-LUT output.
I0, I1, I2, I3, I4	Input	1	LUT inputs.
CDO	Output	1	Reconfiguration data cascaded output (optionally connect to the CDI input of a subsequent LUT).
CDI	Input	1	Reconfiguration data serial input.
CLK	Input	1	Reconfiguration clock.
CE	Input	1	Active-High reconfiguration clock enable.

Design Entry Method

Instantiation	Recommended
Inference	No
IP Catalog	No
Macro support	No

- Connect the CLK input to the clock source used to supply the reconfiguration data.
- Connect the CDI input to the source of the reconfiguration data.
- Connect the CE pin to the active-High logic if you need to enable/disable LUT reconfiguration.
- Connect the I4-I0 pins to the source inputs to the logic equation. The logic function is output on O6 and O5.
- To cascade this element, connect the CDO pin from each element to the CDI input of the next element to allow a single serial chain of data to reconfigure multiple LUTs.

The INIT attribute should be placed on this design element to specify the initial logical function of the LUT. A new INIT can be loaded into the LUT any time during circuit operation by shifting in 32-bits per LUT in the chain, representing the new

INIT value. Disregard the O6 and O5 output data until all 32-bits of new INIT data has been clocked into the LUT. The logical function of the LUT changes as new INIT data is shifted into it. Data should be shifted in MSB (INIT[31]) first and LSB (INIT[0]) last.

In order to understand the O6 and O5 logical value based on the current INIT, see the following table.

Table: Logic Table

I4 I3 I2 I1 I0	O6 Value	O5 Value
1 1 1 1 1	INIT[31]	INIT[15]
1 1 1 1 0	INIT[30]	INIT[14]
...
1 0 0 0 1	INIT[17]	INIT[1]
1 0 0 0 0	INIT[16]	INIT[0]
0 1 1 1 1	INIT[15]	INIT[15]
0 1 1 1 0	INIT[14]	INIT[14]
...
0 0 0 0 1	INIT[1]	INIT[1]
0 0 0 0 0	INIT[0]	INIT[0]

For instance, the INIT value of FFFF8000 would represent the following logical equations:

- $O6 = I4 \text{ or } (I3 \text{ and } I2 \text{ and } I1 \text{ and } I0)$
- $O5 = I3 \text{ and } I2 \text{ and } I1 \text{ and } I0$

To use these elements as two, 4-input LUTs with the same inputs but different functions, tie the I4 signal to a logical one. The INIT[31:16] values apply to the logical values of the O6 output and INIT [15:0] apply to the logical values of the O5 output.

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-bit value	All zeros	Specifies the initial logical expression of this element.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```

Library UNISIM;
use UNISIM.vcomponents.all;

-- CFGLUT5: Reconfigurable 5-input LUT (Mapped to SliceM
LUT6)
--          7 Series
-- Xilinx HDL Language Template, version 2025.1

CFGLUT5_inst : CFGLUT5
generic map (
    INT => X"00000000")
port map (
    CD0 => CD0, -- Reconfiguration cascade output
    O5 => O5,   -- 4-LUT output
    O6 => O6,   -- 5-LUT output
    CDI => CDI, -- Reconfiguration data input
    CE  => CE,  -- Reconfiguration enable input
    CLK => CLK, -- Clock input
    I0  => I0,  -- Logic data input
    I1  => I1,  -- Logic data input
    I2  => I2,  -- Logic data input
    I3  => I3,  -- Logic data input
    I4  => I4   -- Logic data input
);

-- End of CFGLUT5_inst instantiation

```

Verilog Instantiation Template

```
// CFGLUT5: Reconfigurable 5-input LUT (Mapped to a SliceM
LUT6)
//          7 Series
// Xilinx HDL Language Template, version 2025.1

CFGLUT5 #(
    .INIT(32'h00000000) // Specify initial LUT contents
) CFGLUT5_inst (
    .CD0(CD0), // Reconfiguration cascade output
    .O5(O5),   // 4-LUT output
    .O6(O6),   // 5-LUT output
    .CDI(CDI), // Reconfiguration data input
    .CE(CE),   // Reconfiguration enable input
    .CLK(CLK), // Clock input
    .I0(I0),   // Logic data input
    .I1(I1),   // Logic data input
    .I2(I2),   // Logic data input
    .I3(I3),   // Logic data input
    .I4(I4)    // Logic data input
);

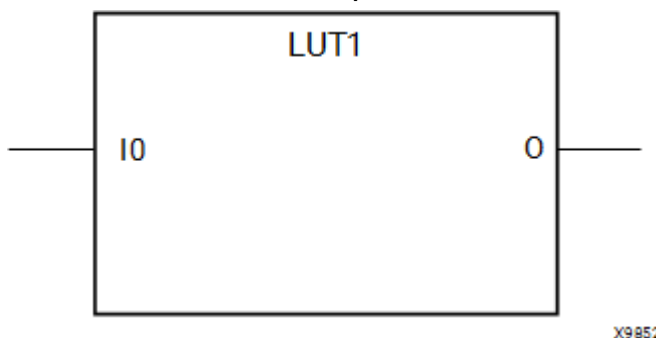
// End of CFGLUT5_inst instantiation
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

LUT1

Primitive: 1-Bit Look-Up Table with General Output



Introduction

This design element is a 1-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation. The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- *The Logic Table Method* Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- *The Equation Method* Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs	Outputs
I0	O
0	INIT[0]
1	INIT[1]
INIT = Binary number assigned to the INIT attribute	

Design Entry Method

Instantiation	Yes
---------------	-----

Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 2-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```

Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT1: 1-input Look-Up Table with general output
--       7 Series
-- Xilinx HDL Language Template, version 2025.1

LUT1_inst : LUT1
generic map (
    INIT => "00")
port map (
    0 => 0,    -- LUT general output
    I0 => I0   -- LUT input
);

-- End of LUT1_inst instantiation

```

Verilog Instantiation Template

```
// LUT1: 1-input Look-Up Table with general output (Mapped to
a LUT6)
//      7 Series
// Xilinx HDL Language Template, version 2025.1

LUT1 #(
    .INIT(2'b00) // Specify LUT Contents
) LUT1_inst (
    .O(0), // LUT general output
    .I0(I0) // LUT input
);

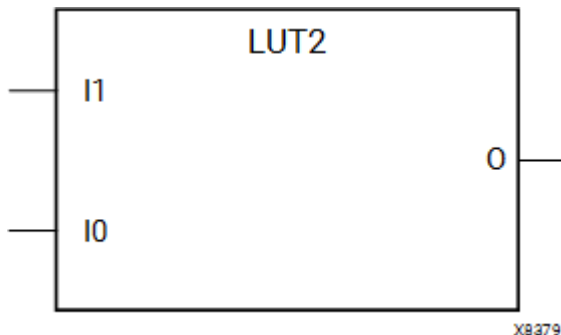
// End of LUT1_inst instantiation
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

LUT2

Primitive: 2-Bit Look-Up Table with General Output



Introduction

This design element is a 2-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Multiple variants of LUTs accommodate additional types of outputs that can

be used by different timing models for more accurate pre-layout timing estimation. The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- *The Logic Table Method* Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- *The Equation Method* Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs		Outputs
I1	I0	O
0	0	INIT[0]
0	1	INIT[1]
1	0	INIT[2]
1	1	INIT[3]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute		

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No

Macro support	No
---------------	----

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 4-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```

Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT2: 2-input Look-Up Table with general output
--       7 Series
-- Xilinx HDL Language Template, version 2025.1

LUT2_inst : LUT2
generic map (
    INIT => X"0")
port map (
    O => O,    -- LUT general output
    I0 => I0,  -- LUT input
    I1 => I1   -- LUT input
);

-- End of LUT2_inst instantiation

```

Verilog Instantiation Template

```

// LUT2: 2-input Look-Up Table with general output (Mapped to

```

```

a LUT6)
//      7 Series
// Xilinx HDL Language Template, version 2025.1

LUT2 #(
    .INIT(4'h0) // Specify LUT Contents
) LUT2_inst (
    .O(0), // LUT general output
    .I0(I0), // LUT input
    .I1(I1) // LUT input
);

// End of LUT2_inst instantiation

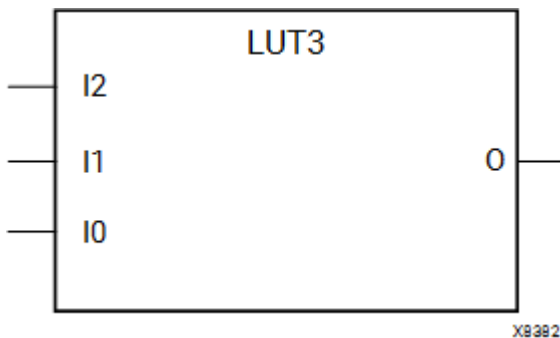
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

LUT3

Primitive: 3-Bit Look-Up Table with General Output



Introduction

This design element is a 3-bit look-up table (LUT) with general output (O). A mandatory INIT attribute, with an appropriate number of hexadecimal digits for the number of inputs, must be attached to the LUT to specify its function.

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building

blocks. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation. The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- *The Logic Table Method* Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- *The Equation Method* Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs			Outputs
I2	I1	I0	O
0	0	0	INIT[0]
0	0	1	INIT[1]
0	1	0	INIT[2]
0	1	1	INIT[3]
1	0	0	INIT[4]
1	0	1	INIT[5]
1	1	0	INIT[6]
1	1	1	INIT[7]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute			

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 8-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;  
use UNISIM.vcomponents.all;
```

```
-- LUT3: 3-input Look-Up Table with general output (Mapped to  
a LUT6)  
--      7 Series  
-- Xilinx HDL Language Template, version 2025.1
```

```
LUT3_inst : LUT3  
generic map (  
    INIT => X"00")  
port map (  
    0 => 0,    -- LUT general output  
    I0 => I0,  -- LUT input  
    I1 => I1,  -- LUT input  
    I2 => I2   -- LUT input
```

```
);  
  
-- End of LUT3_inst instantiation
```

Verilog Instantiation Template

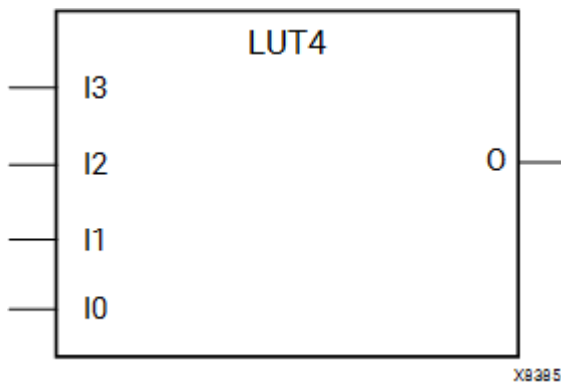
```
// LUT3: 3-input Look-Up Table with general output (Mapped to  
a LUT6)  
//      7 Series  
// Xilinx HDL Language Template, version 2025.1  
  
LUT3 #(  
    .INIT(8'h00) // Specify LUT Contents  
) LUT3_inst (  
    .O(0), // LUT general output  
    .I0(I0), // LUT input  
    .I1(I1), // LUT input  
    .I2(I2) // LUT input  
) ;  
  
// End of LUT3_inst instantiation
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

LUT4

Primitive: 4-Bit Look-Up-Table with General Output



Introduction

This design element is a 4-bit look-up table (LUT) with general output (O).

An INIT attribute with an appropriate number of hexadecimal digits for the number of inputs must be attached to the LUT to specify its function. This element provides a look-up table version of a buffer or inverter. These elements are the basic building blocks. Multiple variants of LUTs accommodate additional types of outputs that can be used by different timing models for more accurate pre-layout timing estimation. The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- *The Logic Table Method* Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- *The Equation Method* Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs				Outputs
I3	I2	I1	I0	O

Inputs				Outputs
I3	I2	I1	I0	O
0	0	0	0	INIT[0]
0	0	0	1	INIT[1]
0	0	1	0	INIT[2]
0	0	1	1	INIT[3]
0	1	0	0	INIT[4]
0	1	0	1	INIT[5]
0	1	1	0	INIT[6]
0	1	1	1	INIT[7]
1	0	0	0	INIT[8]
1	0	0	1	INIT[9]
1	0	1	0	INIT[10]
1	0	1	1	INIT[11]
1	1	0	0	INIT[12]
1	1	0	1	INIT[13]
1	1	1	0	INIT[14]
1	1	1	1	INIT[15]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute				

Design Entry Method

Instantiation	Yes
Inference	Recommended

IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 16-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```

Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT4: 4-input Look-Up Table with general output
--       7 Series
-- Xilinx HDL Language Template, version 2025.1

LUT4_inst : LUT4
generic map (
    INIT => X"0000")
port map (
    0 => 0,    -- LUT general output
    I0 => I0,  -- LUT input
    I1 => I1,  -- LUT input
    I2 => I2,  -- LUT input
    I3 => I3   -- LUT input
);

-- End of LUT4_inst instantiation

```

Verilog Instantiation Template

```
// LUT4: 4-input Look-Up Table with general output (Mapped to
a LUT6)
//      7 Series
// Xilinx HDL Language Template, version 2025.1

LUT4 #(
    .INIT(16'h0000) // Specify LUT Contents
) LUT4_inst (
    .O(0), // LUT general output
    .I0(I0), // LUT input
    .I1(I1), // LUT input
    .I2(I2), // LUT input
    .I3(I3) // LUT input
);

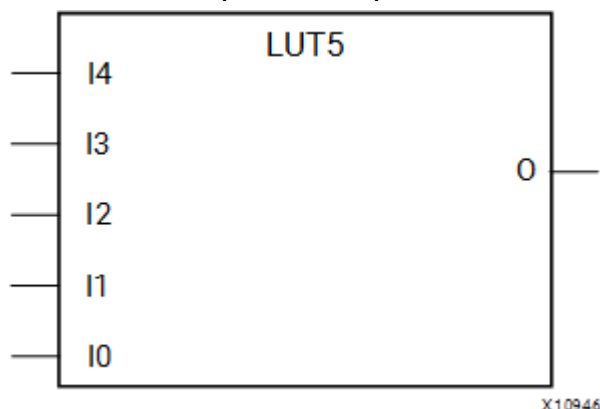
// End of LUT4_inst instantiation
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

LUT5

Primitive: 5-Input look-up Table with General Output



Introduction

This design element is a 5-input, 1-output look-up table (LUT) that can either act as

an asynchronous 32-bit ROM (with 5-bit addressing) or implement any 5-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. One LUT5 is packed into a LUT6 within a slice, or two LUT5s can be packed into a single LUT6 with some restrictions. The functionality of the LUT5, LUT5_L and LUT5_D is the same. However, the LUT5_L and LUT5_D allow the additional specification to connect the LUT5 output signal to an internal slice or CLB connection using the LO output. The LUT5_L specifies that the only connections from the LUT5 will be within a slice or CLB, while the LUT5_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT5 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 32-bit hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to the corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 32'h80000000 (X"80000000" for VHDL) makes the output zero unless all of the inputs are one (a 5-input AND gate). A Verilog INIT value of 32'hfffffffe (X"FFFFFFFE" for VHDL) makes the output one unless all zeros are on the inputs (a 5-input OR gate).

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined in order to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- *The Logic Table Method* Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- *The Equation Method* Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs	Outputs
--------	---------

I4	I3	I2	I1	I0	LO
0	0	0	0	0	INIT[0]
0	0	0	0	1	INIT[1]
0	0	0	1	0	INIT[2]
0	0	0	1	1	INIT[3]
0	0	1	0	0	INIT[4]
0	0	1	0	1	INIT[5]
0	0	1	1	0	INIT[6]
0	0	1	1	1	INIT[7]
0	1	0	0	0	INIT[8]
0	1	0	0	1	INIT[9]
0	1	0	1	0	INIT[10]
0	1	0	1	1	INIT[11]
0	1	1	0	0	INIT[12]
0	1	1	0	1	INIT[13]
0	1	1	1	0	INIT[14]
0	1	1	1	1	INIT[15]
1	0	0	0	0	INIT[16]
1	0	0	0	1	INIT[17]
1	0	0	1	0	INIT[18]
1	0	0	1	1	INIT[19]
1	0	1	0	0	INIT[20]
1	0	1	0	1	INIT[21]
1	0	1	1	0	INIT[22]

1	0	1	1	1	INIT[23]
1	1	0	0	0	INIT[24]
1	1	0	0	1	INIT[25]
1	1	0	1	0	INIT[26]
1	1	0	1	1	INIT[27]
1	1	1	0	0	INIT[28]
1	1	1	0	1	INIT[29]
1	1	1	1	0	INIT[30]
1	1	1	1	1	INIT[31]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute					

Port Descriptions

Port	Direction	Width	Function
O	Output	1	5-LUT output
I0, I1, I2, I3, I4	Input	1	LUT inputs

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 32-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```

Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT5: 5-input Look-Up Table with general output (Mapped to
-- SliceM LUT6)
--      7 Series
-- Xilinx HDL Language Template, version 2025.1

LUT5_inst : LUT5
generic map (
    INIT => X"00000000") -- Specify LUT Contents
port map (
    0 => 0,  -- LUT general output
    I0 => I0,  -- LUT input
    I1 => I1,  -- LUT input
    I2 => I2,  -- LUT input
    I3 => I3,  -- LUT input
    I4 => I4   -- LUT input
);

-- End of LUT5_inst instantiation

```

Verilog Instantiation Template

```

// LUT5: 5-input Look-Up Table with general output (Mapped to
// a LUT6)
//      7 Series

```

```
// Xilinx HDL Language Template, version 2025.1
```

```
LUT5 #(
    .INIT(32'h00000000) // Specify LUT Contents
) LUT5_inst (
    .O(0), // LUT general output
    .I0(I0), // LUT input
    .I1(I1), // LUT input
    .I2(I2), // LUT input
    .I3(I3), // LUT input
    .I4(I4) // LUT input
);

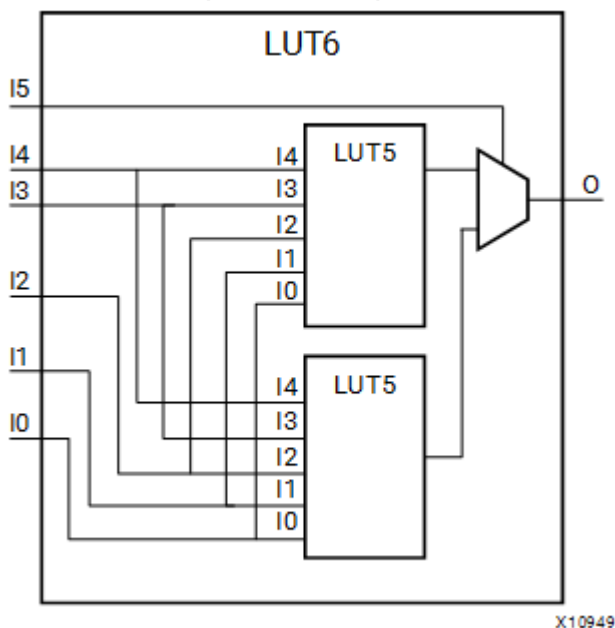
// End of LUT5_inst instantiation
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

LUT6

Primitive: 6-Input Look-Up Table with General Output



Introduction

This design element is a 6-input, 1-output look-up table (LUT) that can either act as an asynchronous 64-bit ROM (with 6-bit addressing) or implement any 6-input logic function. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. A LUT6 is mapped to one of the four look-up tables in the slice. The functionality of the LUT6, LUT6_L and LUT6_D is the same.

However, the LUT6_L and LUT6_D allow the additional specification to connect the LUT6 output signal to an internal slice, or CLB connection, using the LO output. The LUT6_L specifies that the only connections from the LUT6 will be within a slice, or CLB, while the LUT6_D allows the specification to connect the output of the LUT to both inter-slice/CLB logic and external logic as well. The LUT6 does not state any specific output connections and should be used in all cases except where internal slice or CLB signal connections must be implicitly specified.

An INIT attribute consisting of a 64-bit Hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 64'h8000000000000000 (X"8000000000000000" for VHDL) makes the output zero unless all of the inputs are one (a 6-input AND gate). A Verilog INIT value of 64'hffffffffffffe (X"FFFFFFFFFFFFFFFE" for VHDL) makes the output one unless all zeros are on the inputs (a 6-input OR gate).

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- *The Logic Table Method* Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- *The Equation Method* Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs	Outputs
--------	---------

I5	I4	I3	I2	I1	I0	O
0	0	0	0	0	0	INIT[0]
0	0	0	0	0	1	INIT[1]
0	0	0	0	1	0	INIT[2]
0	0	0	0	1	1	INIT[3]
0	0	0	1	0	0	INIT[4]
0	0	0	1	0	1	INIT[5]
0	0	0	1	1	0	INIT[6]
0	0	0	1	1	1	INIT[7]
0	0	1	0	0	0	INIT[8]
0	0	1	0	0	1	INIT[9]
0	0	1	0	1	0	INIT[10]
0	0	1	0	1	1	INIT[11]
0	0	1	1	0	0	INIT[12]
0	0	1	1	0	1	INIT[13]
0	0	1	1	1	0	INIT[14]
0	0	1	1	1	1	INIT[15]
0	1	0	0	0	0	INIT[16]
0	1	0	0	0	1	INIT[17]
0	1	0	0	1	0	INIT[18]
0	1	0	0	1	1	INIT[19]
0	1	0	1	0	0	INIT[20]
0	1	0	1	0	1	INIT[21]
0	1	0	1	1	0	INIT[22]

0	1	0	1	1	1	INIT[23]
0	1	1	0	0	0	INIT[24]
0	1	1	0	0	1	INIT[25]
0	1	1	0	1	0	INIT[26]
0	1	1	0	1	1	INIT[27]
0	1	1	1	0	0	INIT[28]
0	1	1	1	0	1	INIT[29]
0	1	1	1	1	0	INIT[30]
0	1	1	1	1	1	INIT[31]
1	0	0	0	0	0	INIT[32]
1	0	0	0	0	1	INIT[33]
1	0	0	0	1	0	INIT[34]
1	0	0	0	1	1	INIT[35]
1	0	0	1	0	0	INIT[36]
1	0	0	1	0	1	INIT[37]
1	0	0	1	1	0	INIT[38]
1	0	0	1	1	1	INIT[39]
1	0	1	0	0	0	INIT[40]
1	0	1	0	0	1	INIT[41]
1	0	1	0	1	0	INIT[42]
1	0	1	0	1	1	INIT[43]
1	0	1	1	0	0	INIT[44]
1	0	1	1	0	1	INIT[45]
1	0	1	1	1	0	INIT[46]

1	0	1	1	1	1	INIT[47]
1	1	0	0	0	0	INIT[48]
1	1	0	0	0	1	INIT[49]
1	1	0	0	1	0	INIT[50]
1	1	0	0	1	1	INIT[51]
1	1	0	1	0	0	INIT[52]
1	1	0	1	0	1	INIT[53]
1	1	0	1	1	0	INIT[54]
1	1	0	1	1	1	INIT[55]
1	1	1	0	0	0	INIT[56]
1	1	1	0	0	1	INIT[57]
1	1	1	0	1	0	INIT[58]
1	1	1	0	1	1	INIT[59]
1	1	1	1	0	0	INIT[60]
1	1	1	1	0	1	INIT[61]
1	1	1	1	1	0	INIT[62]
1	1	1	1	1	1	INIT[63]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute						

Port Descriptions

Port	Direction	Width	Function
O	Output	1	6/5-LUT output
I0, I1, I2, I3, I4, I5	Input	1	LUT inputs

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 64-Bit Value	All zeros	Initializes look-up tables.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;  
use UNISIM.vcomponents.all;
```

```
-- LUT6: 6-input Look-Up Table with general output  
--       7 Series  
-- Xilinx HDL Language Template, version 2025.1
```

```
LUT6_inst : LUT6  
generic map (  
    INIT => X"0000000000000000") -- Specify LUT Contents  
port map (  
    0 => 0,  -- LUT general output  
    I0 => I0,  -- LUT input  
    I1 => I1,  -- LUT input  
    I2 => I2,  -- LUT input  
    I3 => I3,  -- LUT input
```

```

        I4 => I4,    -- LUT input
        I5 => I5     -- LUT input
    );

    -- End of LUT6_inst instantiation

```

Verilog Instantiation Template

```

// LUT6: 6-input Look-Up Table with general output
//      7 Series
// Xilinx HDL Language Template, version 2025.1

LUT6 #(
    .INIT(64'h0000000000000000) // Specify LUT Contents
) LUT6_inst (
    .O(0),    // LUT general output
    .I0(I0), // LUT input
    .I1(I1), // LUT input
    .I2(I2), // LUT input
    .I3(I3), // LUT input
    .I4(I4), // LUT input
    .I5(I5)  // LUT input
);

// End of LUT6_inst instantiation

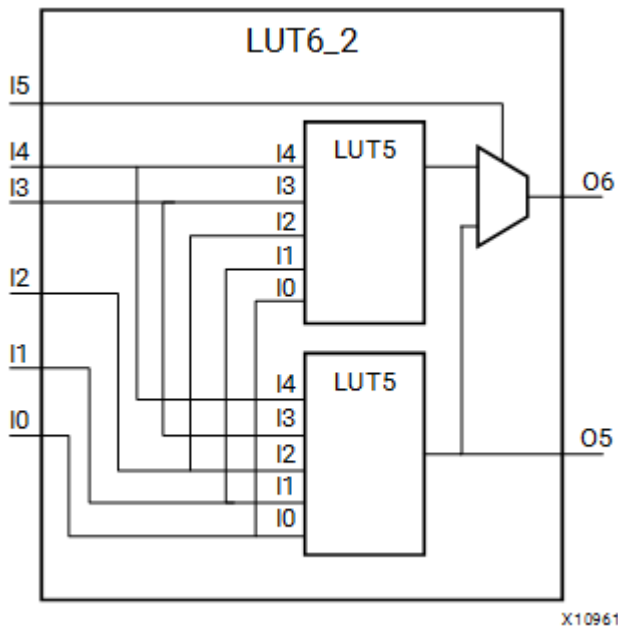
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

LUT6_2

Primitive: Six-input, 2-output, Look-Up Table



Introduction

This design element is a 6-input, 2-output look-up table (LUT) that can either act as a dual asynchronous 32-bit ROM (with 5-bit addressing), implement any two 5-input logic functions with shared inputs, or implement a 6-input logic function and a 5-input logic function with shared inputs and shared logic values. LUTs are the basic logic building blocks and are used to implement most logic functions of the design. A LUT6_2 will be mapped to one of the four look-up tables in the slice.

An INIT attribute consisting of a 64-bit hexadecimal value must be specified to indicate the LUTs logical function. The INIT value is calculated by assigning a 1 to corresponding INIT bit value when the associated inputs are applied. For instance, a Verilog INIT value of 64'hffffffffffffe (X"FFFFFFFFFFFFFFFFFE" for VHDL) makes the O6 output 1 unless all zeros are on the inputs and the O5 output a 1, or unless I[4:0] are all zeros (a 5-input and 6-input OR gate). The lower half (bits 31:0) of the INIT values apply to the logic function of the O5 output.

The INIT parameter for the FPGA LUT primitive gives the LUT its logical value. By default, this value is zero, thus driving the output to a zero regardless of the input values (acting as a ground). However, in most cases a new INIT value must be determined to specify the logic function for the LUT primitive. There are at least two methods by which the LUT value can be determined.

- *The Logic Table Method:* Create a binary logic table of all possible inputs, specify the desired logic value of the output and then create the INIT string from those output values.
- *The Equation Method:* Define parameters for each input to the LUT that correspond to their listed truth value and use those to build the logic equation you are after. This method is easier to understand once you have grasped the concept and is more self-documenting than the above method. However, this method does require the code to first specify the appropriate parameters.

Logic Table

Inputs						Outputs	
I5	I4	I3	I2	I1	I0	O5	O6
0	0	0	0	0	0	INIT[0]	INIT[0]
0	0	0	0	0	1	INIT[1]	INIT[1]
0	0	0	0	1	0	INIT[2]	INIT[2]
0	0	0	0	1	1	INIT[3]	INIT[3]
0	0	0	1	0	0	INIT[4]	INIT[4]
0	0	0	1	0	1	INIT[5]	INIT[5]
0	0	0	1	1	0	INIT[6]	INIT[6]
0	0	0	1	1	1	INIT[7]	INIT[7]
0	0	1	0	0	0	INIT[8]	INIT[8]
0	0	1	0	0	1	INIT[9]	INIT[9]
0	0	1	0	1	0	INIT[10]	INIT[10]
0	0	1	0	1	1	INIT[11]	INIT[11]
0	0	1	1	0	0	INIT[12]	INIT[12]
0	0	1	1	0	1	INIT[13]	INIT[13]
0	0	1	1	1	0	INIT[14]	INIT[14]

Inputs						Outputs	
I5	I4	I3	I2	I1	I0	O5	O6
0	0	1	1	1	1	INIT[15]	INIT[15]
0	1	0	0	0	0	INIT[16]	INIT[16]
0	1	0	0	0	1	INIT[17]	INIT[17]
0	1	0	0	1	0	INIT[18]	INIT[18]
0	1	0	0	1	1	INIT[19]	INIT[19]
0	1	0	1	0	0	INIT[20]	INIT[20]
0	1	0	1	0	1	INIT[21]	INIT[21]
0	1	0	1	1	0	INIT[22]	INIT[22]
0	1	0	1	1	1	INIT[23]	INIT[23]
0	1	1	0	0	0	INIT[24]	INIT[24]
0	1	1	0	0	1	INIT[25]	INIT[25]
0	1	1	0	1	0	INIT[26]	INIT[26]
0	1	1	0	1	1	INIT[27]	INIT[27]
0	1	1	1	0	0	INIT[28]	INIT[28]
0	1	1	1	0	1	INIT[29]	INIT[29]
0	1	1	1	1	0	INIT[30]	INIT[30]
0	1	1	1	1	1	INIT[31]	INIT[31]
1	0	0	0	0	0	INIT[0]	INIT[32]
1	0	0	0	0	1	INIT[1]	INIT[33]
1	0	0	0	1	0	INIT[2]	INIT[34]
1	0	0	0	1	1	INIT[3]	INIT[35]
1	0	0	1	0	0	INIT[4]	INIT[36]

Inputs						Outputs	
I5	I4	I3	I2	I1	I0	O5	O6
1	0	0	1	0	1	INIT[5]	INIT[37]
1	0	0	1	1	0	INIT[6]	INIT[38]
1	0	0	1	1	1	INIT[7]	INIT[39]
1	0	1	0	0	0	INIT[8]	INIT[40]
1	0	1	0	0	1	INIT[9]	INIT[41]
1	0	1	0	1	0	INIT[10]	INIT[42]
1	0	1	0	1	1	INIT[11]	INIT[43]
1	0	1	1	0	0	INIT[12]	INIT[44]
1	0	1	1	0	1	INIT[13]	INIT[45]
1	0	1	1	1	0	INIT[14]	INIT[46]
1	0	1	1	1	1	INIT[15]	INIT[47]
1	1	0	0	0	0	INIT[16]	INIT[48]
1	1	0	0	0	1	INIT[17]	INIT[49]
1	1	0	0	1	0	INIT[18]	INIT[50]
1	1	0	0	1	1	INIT[19]	INIT[51]
1	1	0	1	0	0	INIT[20]	INIT[52]
1	1	0	1	0	1	INIT[21]	INIT[53]
1	1	0	1	1	0	INIT[22]	INIT[54]
1	1	0	1	1	1	INIT[23]	INIT[55]
1	1	1	0	0	0	INIT[24]	INIT[56]
1	1	1	0	0	1	INIT[25]	INIT[57]
1	1	1	0	1	0	INIT[26]	INIT[58]

Inputs						Outputs	
I5	I4	I3	I2	I1	I0	O5	O6
1	1	1	0	1	1	INIT[27]	INIT[59]
1	1	1	1	0	0	INIT[28]	INIT[60]
1	1	1	1	0	1	INIT[29]	INIT[61]
1	1	1	1	1	0	INIT[30]	INIT[62]
1	1	1	1	1	1	INIT[31]	INIT[63]
INIT = Binary equivalent of the hexadecimal number assigned to the INIT attribute.							

Port Descriptions

Port	Direction	Width	Function
O6	Output	1	6/5-LUT output.
O5	Output	1	5-LUT output.
I0, I1, I2, I3, I4, I5	Input	1	LUT inputs.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
-----------	------	----------------	---------	-------------

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 64-Bit Value	All zeros	Specifies the LUT5/6 output function.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```

Library UNISIM;
use UNISIM.vcomponents.all;

-- LUT6_2: 6-input 2 output Look-Up Table
--          7 Series
-- Xilinx HDL Language Template, version 2025.1

LUT6_2_inst : LUT6_2
generic map (
    INIT => X"0000000000000000") -- Specify LUT Contents
port map (
    O6 => O6,  -- 6/5-LUT output (1-bit)
    O5 => O5,  -- 5-LUT output (1-bit)
    I0 => I0,   -- LUT input (1-bit)
    I1 => I1,   -- LUT input (1-bit)
    I2 => I2,   -- LUT input (1-bit)
    I3 => I3,   -- LUT input (1-bit)
    I4 => I4,   -- LUT input (1-bit)
    I5 => I5    -- LUT input (1-bit)
);

-- End of LUT6_2_inst instantiation

```

Verilog Instantiation Template

```

// LUT6_2: 6-input, 2 output Look-Up Table
//          7 Series

```



```
// Xilinx HDL Language Template, version 2025.1

LUT6_2 #(
    .INIT(64'h0000000000000000) // Specify LUT Contents
) LUT6_2_inst (
    .O6(O6), // 1-bit LUT6 output
    .O5(O5), // 1-bit lower LUT5 output
    .I0(I0), // 1-bit LUT input
    .I1(I1), // 1-bit LUT input
    .I2(I2), // 1-bit LUT input
    .I3(I3), // 1-bit LUT input
    .I4(I4), // 1-bit LUT input
    .I5(I5) // 1-bit LUT input (fast MUX select only
available to O6 output)
);

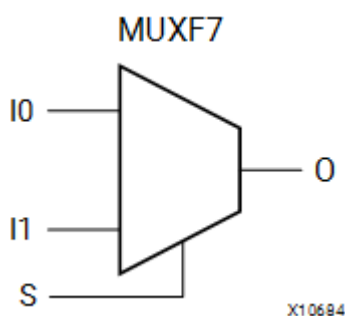
// End of LUT6_2_inst instantiation
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

MUXF7

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output



Introduction

This design element is a two input multiplexer which, in combination with two LUT6 elements will let you create any 7-input function, an 8-to-1 multiplexer, or other logic functions up to 12-bits wide. Local outputs of the LUT6 element are connected to

the I0 and I1 inputs of the MUXF7. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1. The O output is a general interconnect.

Logic Table

Inputs			Outputs
S	I0	I1	O
0	I0	X	I0
1	X	I1	I1
X	0	0	0
X	1	1	1

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Output of MUX to general routing.
I0	Input	1	Input (tie to LUT6 LO out).
I1	Input	1	Input (tie to LUT6 LO out).
S	Input	1	Input select to MUX.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- MUXF7: CLB MUX to tie two LUT6's together with general
output
--          7 Series
-- Xilinx HDL Language Template, version 2025.1

MUXF7_inst : MUXF7
port map (
    0 => 0,    -- Output of MUX to general routing
    I0 => I0,  -- Input (tie to LUT6 06 pin)
    I1 => I1,  -- Input (tie to LUT6 06 pin)
    S => S     -- Input select to MUX
);

-- End of MUXF7_inst instantiation
```

Verilog Instantiation Template

```
// MUXF7: CLB MUX to tie two LUT6's together with general
output
//          7 Series
// Xilinx HDL Language Template, version 2025.1

MUXF7 MUXF7_inst (
    .0(0),    // Output of MUX to general routing
    .I0(I0),  // Input (tie to LUT6 06 pin)
    .I1(I1),  // Input (tie to LUT6 06 pin)
    .S(S)     // Input select to MUX
);
```

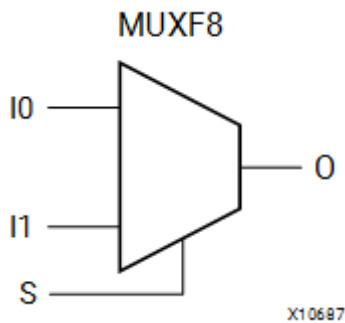
```
// End of MUXF7_inst instantiation
```

Related Information

- 7 Series FPGAs Configurable Logic Block User Guide ([UG474](#))

MUXF8

Primitive: 2-to-1 Look-Up Table Multiplexer with General Output



Introduction

This design element is a two input multiplexer which, in combination with two MUXF7 multiplexers and their four associated LUT6 elements, will let you create any 8-input function, a 16-to-1 multiplexer, or other logic functions up to 24-bits wide. Local outputs of MUXF7 are connected to the I0 and I1 inputs of the MUXF8. The S input is driven from any internal net. When Low, S selects I0. When High, S selects I1. The O output is a general interconnect.

Logic Table

Inputs			Outputs
S	I0	I1	O
0	I0	X	I0
1	X	I1	I1

Inputs			Outputs
S	I0	I1	O
X	0	0	0
X	1	1	1

Port Descriptions

Port	Direction	Width	Function
O	Output	1	Output of MUX to general routing.
I0	Input	1	Input (tie to MUXF7 LO out).
I1	Input	1	Input (tie to MUXF7 LO out).
S	Input	1	Input select to MUX.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- MUXF8: CLB MUX to tie two MUXF7's together with general
```

```

output
--          7 Series
-- Xilinx HDL Language Template, version 2025.1

MUXF8_inst : MUXF8
port map (
    0 => 0,      -- Output of MUX to general routing
    I0 => I0,    -- Input (tie to MUXF7 L/L0 out)
    I1 => I1,    -- Input (tie to MUXF7 L/L0 out)
    S => S       -- Input select to MUX
);

-- End of MUXF8_inst instantiation

```

Verilog Instantiation Template

```

// MUXF8: CLB MUX to tie two MUXF7's together with general
output
//          7 Series
// Xilinx HDL Language Template, version 2025.1

MUXF8 MUXF8_inst (
    .0(0),      // Output of MUX to general routing
    .I0(I0),    // Input (tie to MUXF7 L/L0 out)
    .I1(I1),    // Input (tie to MUXF7 L/L0 out)
    .S(S)       // Input select to MUX
);

// End of MUXF8_inst instantiation

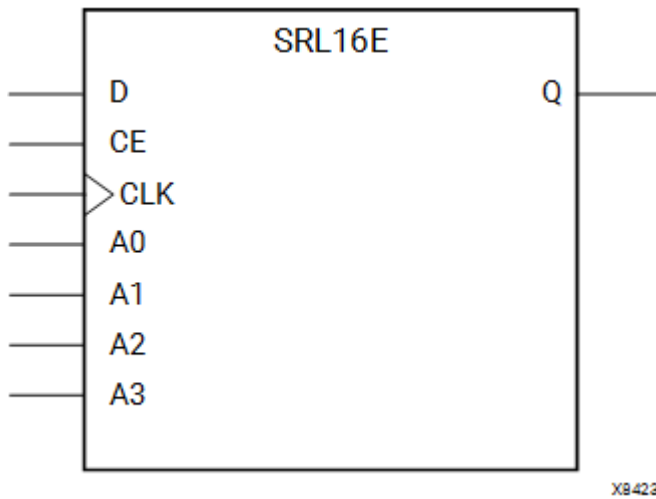
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

SRL16E

Primitive: 16-Bit Shift Register Look-Up Table (LUT) with Clock Enable



Introduction

This design element is a shift register look-up table (LUT). The inputs A3, A2, A1, and A0 select the depth of the shift register. The shift register can be of a fixed, static depth or it can be dynamically adjusted.

To create a fixed-depth shift register: Drive the A3 through A0 inputs with static values. The depth of the shift register can vary from 1 bit to 16 bits, as determined by the following formula:

$$\text{Depth} = (8 \times A3) + (4 \times A2) + (2 \times A1) + A0 + 1$$


If A3, A2, A1, and A0 are all zeros (0000), the shift register is one bit deep. If they are all ones (1111), it is 16 bits deep.

To change the depth of the shift register dynamically: Change the values driving the A3 through A0 inputs. For example, if A2, A1, and A0 are all ones (111) and A3 toggles between a one (1) and a zero (0), the depth of the shift register changes from 16 bits to 8 bits. Internally, the depth of the shift register is always 16 bits and the input lines A3 through A0 select which of the 16 bits reach the output. The shift register LUT contents are initialized by assigning a four-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of four zeros (0000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the clock (CLK) transition. During subsequent clock transitions, when CE is High, data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached. When CE is Low, the register ignores clock transitions and retains current data within the shift register.

Two SLR16E components may be placed within the same LUT within a CLBM as

long as they have the same clock, clock enable and depth selection address signals as well as the same IS_CLK_INVERTED attribute value. This allows up to 16 SRL16E components to be placed into a single CLB. Optionally, LUTNM or HLUTNMs may be placed on two SRL16E components to specify specific grouping within a LUT.

 **Note:** When using SRLs with initialized values, you should use safe clock start-up techniques to ensure the initialized data is not corrupted upon completion of configuration. Refer to UG949: UltraFast Design Methodology Guide for details on controlling and synchronizing clock startup.

Logic Table

Inputs				Output
Am	CE	CLK	D	Q
Am	0	X	X	Q(Am)
Am	1	↑	D	Q(Am - 1)
m= 0, 1, 2, 3				

Port Descriptions

Port	Direction	Width	Function
CE	Input	1	Active-High clock enable
CLK	Input	1	Shift register clock. Polarity is determined by the IS_CLK_INVERTED attribute.
D	Input	1	SRL data input.
Q	Output	1	SRL data output.
A0	Input	1	The value placed on the A0 - A3 inputs specifies the shift register depth. <i>Depth = (8 x A3) + (4 x A2) + (2 x</i>

Port	Direction	Width	Function
			$A1) + A0 + 1$
A1	Input	1	The value placed on the A0 - A3 inputs specifies the shift register depth. $Depth = (8 \times A3) + (4 \times A2) + (2 \times A1) + A0 + 1$
A2	Input	1	The value placed on the A0 - A3 inputs specifies the shift register depth. $Depth = (8 \times A3) + (4 \times A2) + (2 \times A1) + A0 + 1$
A3	Input	1	The value placed on the A0 - A3 inputs specifies the shift register depth. $Depth = (8 \times A3) + (4 \times A2) + (2 \times A1) + A0 + 1$

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Values	Default	Description
INIT	HEX	Any 16-Bit Value	All zeros	Specifies the initial contents in the shift register upon completion of configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;

-- SRL16E: 16-bit shift register LUT with clock enable
-- operating on posedge of clock (Mapped to SliceM LUT6)
--          7 Series
-- Xilinx HDL Language Template, version 2025.1

SRL16E_inst : SRL16E
generic map (
    INIT => X"0000")
port map (
    Q => Q,          -- SRL data output
    A0 => A0,        -- Select[0] input
    A1 => A1,        -- Select[1] input
    A2 => A2,        -- Select[2] input
    A3 => A3,        -- Select[3] input
    CE => CE,        -- Clock enable input
    CLK => CLK,      -- Clock input
    D => D           -- SRL data input
);

-- End of SRL16E_inst instantiation
```

Verilog Instantiation Template

```
// SRL16E: 16-bit shift register LUT with clock enable
// operating
//          on posedge of clock (Mapped to a SliceM LUT6)
//          7 Series
// Xilinx HDL Language Template, version 2025.1
```

```

SRL16E #(
    .INIT(16'h0000) // Initial Value of Shift Register
) SRL16E_inst (
    .Q(Q),           // SRL data output
    .A0(A0),         // Select[0] input
    .A1(A1),         // Select[1] input
    .A2(A2),         // Select[2] input
    .A3(A3),         // Select[3] input
    .CE(CE),         // Clock enable input
    .CLK(CLK),       // Clock input
    .D(D)            // SRL data input
);

// End of SRL16E_inst instantiation

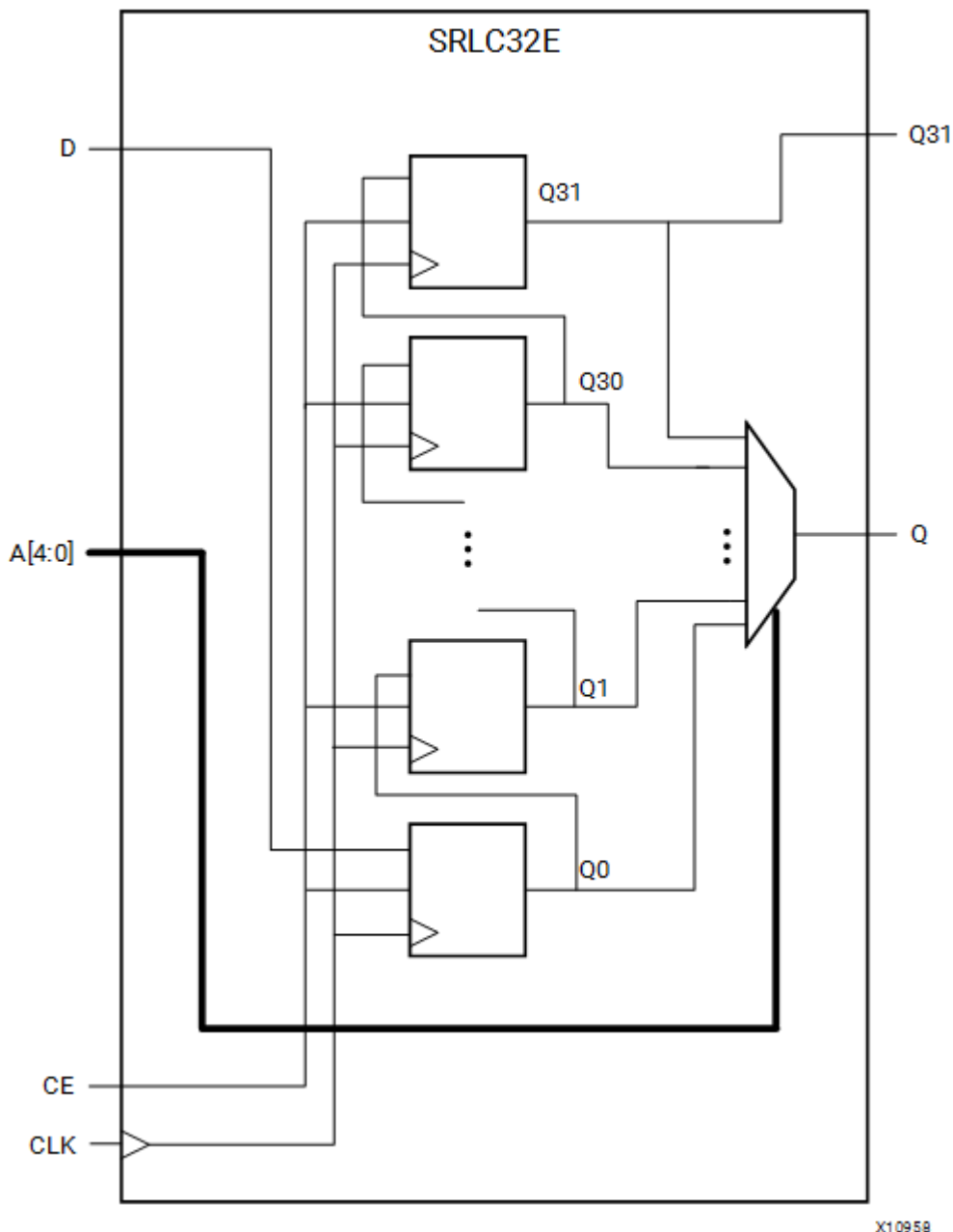
```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))

SRLC32E

Primitive: 32 Clock Cycle, Variable Length Shift Register Look-Up Table (LUT) with Clock Enable



Introduction

This design element is a shift register look-up table (LUT). The inputs A4, A3, A2, A1, and A0 select the depth of the shift register.

The shift register can be of a fixed, static depth or it can be dynamically adjusted.

To create a fixed-depth shift register: Drive the A4 through A0 inputs with static values. The depth of the shift register can vary from 1 bit to 32 bits, as determined by the following formula:

$$\text{Depth} = (16 \times A4) + (8 \times A3) + (4 \times A2) + (2 \times A1) + A0 + 1$$


If A4, A3, A2, A1, and A0 are all zeros (00000), the shift register is one bit deep. If they are all ones (11111), it is 32 bits deep.

To change the depth of the shift register dynamically: Change the values driving the

A4 through A0 inputs. For example, if A3, A2, A1, and A0 are all ones (1111) and A4 toggles between a one (1) and a zero (0), the depth of the shift register changes from 32 bits to 16 bits. Internally, the depth of the shift register is always 32 bits and the input lines A4 through A0 select which of the 32 bits reach the output. The shift register LUT contents are initialized by assigning a eight-digit hexadecimal number to an INIT attribute. The first, or the left-most, hexadecimal digit is the most significant bit. If an INIT value is not specified, it defaults to a value of eight zeros (00000000) so that the shift register LUT is cleared during configuration.

When CE is High, the data (D) is loaded into the first bit of the shift register during the clock (CLK) transition. During subsequent clock transitions, when CE is High, data shifts to the next highest bit position as new data is loaded. The data appears on the Q output when the shift register length determined by the address inputs is reached. When CE is Low, the register ignores clock transitions and retains current data within the shift register.

Two or more SLRC32E components may be cascaded to create deeper than 32-bit shift registers. To do so, connect the Q31 output of one SRLC32E component to the D input of another.

 **Note:** When using SRLs with initialized values, you should use safe clock start-up techniques to ensure the initialized data is not corrupted upon completion of configuration. Refer to UG949: UltraFast Design Methodology Guide for details on controlling and synchronizing clock startup.

Port Descriptions

Port	Direction	Width	Function
A<4:0>	Input	5	The value placed on the A0 - A3 inputs specifies the shift register depth. $Depth = (16 \times A4) + (8 \times A3) + (4 \times A2) + (2 \times A1) + A0 + 1$
CE	Input	1	Active-High clock enable.
CLK	Input	1	Shift register clock. Polarity is determined by the IS_CLK_INVERTED attribute.
D	Input	1	SRL data input.

Port	Direction	Width	Function
Q	Output	1	SRL data output.
Q31	Output	1	SRL data output used to connect more than one SRLC32E component to form deeper than 32-bit shift registers.

Design Entry Method

Instantiation	Yes
Inference	Recommended
IP Catalog	No
Macro support	No

Available Attributes

Attribute	Type	Allowed Value	Default	Description
INIT	HEX	Any 32-Bit Value	All zeros	Specifies the initial contents in the shift register upon completion of configuration.

VHDL Instantiation Template

Unless they already exist, copy the following two statements and paste them before the entity declaration.

```
Library UNISIM;
use UNISIM.vcomponents.all;
```

```
-- SRLC32E: 32-bit variable length shift register LUT
--           with clock enable (Mapped to a SliceM LUT6)
```

```

--          7 Series
-- Xilinx HDL Language Template, version 2025.1

SRLC32E_inst : SRLC32E
generic map (
    INIT => X"00000000")
port map (
    Q => Q,          -- SRL data output
    Q31 => Q31,      -- SRL cascade output pin
    A => A,          -- 5-bit shift depth select input
    CE => CE,        -- Clock enable input
    CLK => CLK,      -- Clock input
    D => D           -- SRL data input
);

-- End of SRLC32E_inst instantiation

```

Verilog Instantiation Template

```

// SRLC32E: 32-bit variable length cascadable shift register
// LUT (Mapped to a SliceM LUT6)
//          with clock enable
//          7 Series
// Xilinx HDL Language Template, version 2025.1

SRLC32E #(
    .INIT(32'h00000000) // Initial Value of Shift Register
) SRLC32E_inst (
    .Q(Q),          // SRL data output
    .Q31(Q31),     // SRL cascade output pin
    .A(A),          // 5-bit shift depth select input
    .CE(CE),        // Clock enable input
    .CLK(CLK),      // Clock input
    .D(D)           // SRL data input
);

// End of SRLC32E_inst instantiation

```

Related Information

- *7 Series FPGAs Configurable Logic Block User Guide* ([UG474](#))