XAPP464 (v2.0) March 1, 2005

Using Look-Up Tables as Distributed RAM in Spartan-3 Generation FPGAs

Summary

Each Spartan™-3, Spartan-3L, or Spartan-3E Configurable Logic Block (CLB) contains up to 64 bits of single-port RAM or 32 bits of dual-port RAM. This RAM is distributed throughout the FPGA and is commonly called “distributed RAM” to distinguish it from block RAM. Distributed RAM is fast, localized, and ideal for small data buffers, FIFOs, or register files. This application note describes the features and capabilities of distributed RAM and illustrates how to specify the various options using the Xilinx CORE Generator™ system or via VHDL or Verilog instantiation.

Introduction

In addition to the embedded 18Kbit block RAMs, Spartan-3 FPGAs feature distributed RAM within each Configurable Logic Block (CLB). Each SLICEM function generator or LUT within a CLB resource optionally implements a 16-deep x 1-bit synchronous RAM. The LUTs within a SLICEL slice do not have distributed RAM.

Distributed RAM writes synchronously and reads asynchronously. However, if required by the application, use the register associated with each LUT to implement a synchronous read function. Each 16 x 1-bit RAM is cascadable for deeper and/or wider memory applications, with a minimal timing penalty incurred through specialized logic resources.

Spartan-3 CLBs support various RAM primitives up to 64-deep by 1-bit-wide. Two LUTs within a SLICEM slice combine to create a dual-port 16x1 RAM—one LUT with a read/write port, and a second LUT with a read-only port. One port writes into both 16x1 LUT RAMs simultaneously, but the second port reads independently.

Distributed RAM is crucial to many high-performance applications that require relatively small embedded RAM blocks, such as FIFOs or small register files. The Xilinx CORE Generator™ software automatically generates optimized distributed RAMs for the Spartan-3 architecture. Similarly, CORE Generator creates Asynchronous and Synchronous FIFOs using distributed RAMs.

Single-Port and Dual-Port RAMs

Data Flow

Distributed RAM supports the following memory types:

- Single-port RAM with synchronous write and asynchronous read. Synchronous reads are possible using the flip-flop associated with distributed RAM.
- Dual-port RAM with one synchronous write and two asynchronous read ports. As above, synchronous reads are possible.

As illustrated in Figure 1, dual-port distributed RAM has one read/write port and an independent read port.

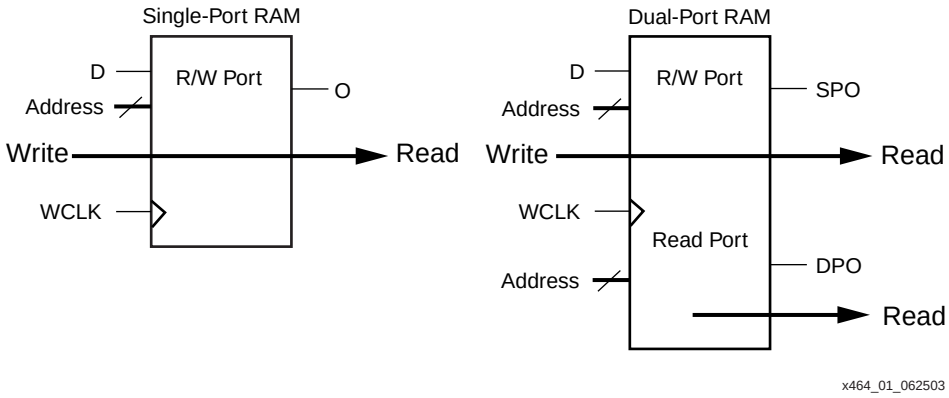

Figure 1: **Single-Port and Dual-Port Distributed RAM**

Any write operation on the D input and any read operation on the SPO output can occur simultaneously with and independently from a read operation on the second read-only port, DPO.

Write Operations

The write operation is a single clock-edge operation, controlled by the write-enable input, WE. By default, WE is active High, although it can be inverted within the distributed RAM. When the write enable is High, the clock edge latches the write address and writes the data on the D input into the selected RAM location.

When the write enable is Low, no data is written into the RAM.

Read Operation

A read operation is purely combinatorial. The address port—either for single- or dual-port modes—is asynchronous with an access time equivalent to a LUT logic delay.

Read During Write

When synchronously writing new data, the output reflects the data being written to the addressed memory cell, which is similar to the WRITE_MODE=WRITE_FIRST mode on the Spartan-3 block RAMs. The timing diagram in Figure 2 illustrates a write operation with the previous data read on the output port, before the clock edge, followed by the new data.

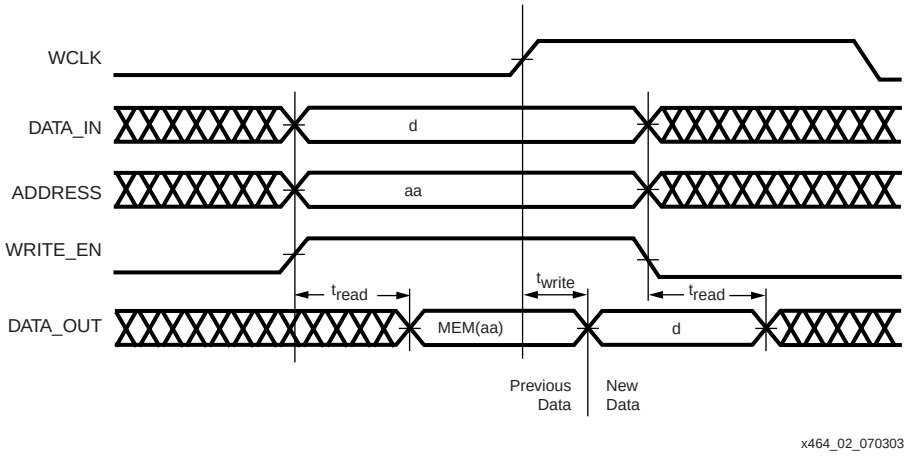

Figure 2: **Write Timing Diagram**

Characteristics

- A write operation requires only one clock edge.
- A read operation requires only the logic access time.
- Outputs are asynchronous and dependent only on the LUT logic delay.
- Data and address inputs are latched with the write clock and have a setup-to-clock timing specification. There is no hold time requirement.
- For dual-port RAM, the A[#:0] port is the write and read address, and the DPRA[#:0] port is an independent read-only address.

Compatibility with Other Xilinx FPGA Families

Each Spartan-3 distributed RAM operates identically to the distributed RAM found in Virtex™, Virtex-E, Spartan-II, Spartan-IIE, Virtex-II, and Virtex-II Pro™ FPGAs.

Table 1 shows the basic memory capabilities embedded within the CLBs on various Xilinx FPGA families. Like Virtex-II/Virtex-II Pro FPGAs, Spartan-3 CLBs have eight LUTs and implement 128 bits of ROM memory. Like the Virtex/Virtex-E and Spartan-II/Spartan-IIE FPGAs, Spartan-3 CLBs have 64 bits of distributed RAM. Although the Spartan-3 and Virtex-II/Virtex-II Pro CLBs are identical for logic functions, the Spartan-3 CLBs have half the amount of distributed RAM within each CLB.

Table 1: **Distributed Memory Features by FPGA Family**

Feature	Spartan-3/ Spartan-3L/ Spartan-3E Families	Virtex/Virtex-E, Spartan-II/Spartan-IIE Families	Virtex-II, Virtex-II Pro Families
LUTs per CLB	8	4	8
ROM bits per CLB	128	64	128
Single-port RAM bits per CLB	64	64	128
Dual-port RAM bits per CLB	32	32	64

Table 2 lists the various single- and dual-port distributed RAM primitives supported by the different Xilinx FPGA families. For each type of RAM, the table indicates how many instances of a particular primitive fit within a single CLB. For example, two 32x1 single-port RAM primitives fit in a single Spartan-3 CLB. Similarly, two 16x1 dual-port RAM primitives fit in a Spartan-3 CLB but a single 32x1 dual-port RAM primitive does not.

Table 2: **Single- and Dual-port RAM Primitives Supported in a CLB by Family**

Family	Single-Port RAM				Dual-Port RAM		
	16x1	32x1	64x1	128x1	16x1	32x1	64x1
Spartan-3	4	2	1		2		
Spartan-II/Spartan-IIE Virtex/Virtex-E	4	2	1		2		
Virtex-II/Virtex-II Pro	8	4	2	1	4	2	1

Library Primitives

There are four library primitives that support Spartan-3 distributed RAM, ranging from 16 bits deep to 64 bits deep. All the primitives are one bit wide. Three primitives are single-port RAMs and one primitive is dual-port RAM, as shown in Table 3.

Table 3: **Single-Port and Dual-Port Distributed RAMs**

Primitive	RAM Size (Depth x Width)	Type	Address Inputs
RAM16X1S	16 x 1	Single-port	A3, A2, A1, A0
RAM32X1S	32 x 1	Single-port	A4, A3, A2, A1, A0
RAM64X1S	64 x 1	Single-port	A5, A4, A3, A2, A1, A0
RAM16X1D	16 x 1	Dual-port	A3, A2, A1, A0

The input and output data are one bit wide. However, several distributed RAMs, connected in parallel, easily implement wider memory functions.

Figure 3 shows generic single-port and dual-port distributed RAM primitives. The A[#:0] and DPRA[#:0] signals are address buses.

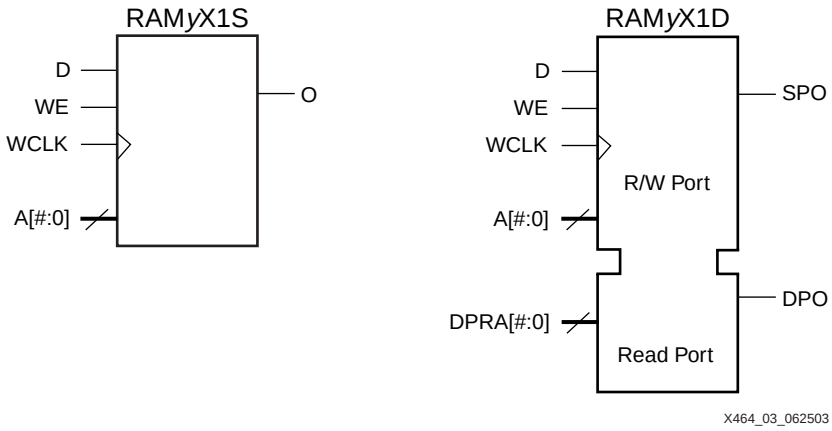

Figure 3: **Single-Port and Dual-Port Distributed RAM Primitives**

As shown in Table 4, wider library primitives are available for 2-bit and 4-bit RAMs.

Table 4: **Wider Library Primitives**

Primitive	RAM Size (Depth x Width)	Data Inputs	Address Inputs	Data Outputs
RAM16x2S	16 x 2	D1, D0	A3, A2, A1, A0	O1, O0
RAM32X2S	32 x 2	D1, D0	A4, A3, A2, A1, A0	O1, O0
RAM16X4S	16 x 4	D3, D2, D1, D0	A3, A2, A1, A0	O3, O2, O1, O0

Signal Ports

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

Clock — WCLK

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

Enable — WE

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

Address — A0, A1, A2, A3 (A4, A5)

The address inputs select the memory cells for read or write. The width of the port determines the required address inputs.

Note: The address inputs are not a bus in VHDL or Verilog instantiations.

Data In — D

The data input provides the new data value to be written into the RAM.

Data Out — O, SPO, and DPO

The data output O on single-port RAM or the SPO and DPO outputs on dual-port RAM reflects the contents of the memory cells referenced by the address inputs. Following an active write clock edge, the data out (O or SPO) reflects the newly written data.

Inverting Control Pins

The two control pins, WCLK and WE, each have an individual inversion option. Any control signal, including the clock, can be active at logic level 0 (negative edge for the clock) or at logic level 1 (positive edge for the clock) without requiring other logic resources.

Global Set/Reset — GSR

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

Global Write Enable — GWE

The global write enable signal, GWE, is asserted automatically at the end of device configuration to enable all writable elements. The GWE signal guarantees that the initialized distributed-RAM contents are not disturbed during the configuration process.

Because GWE is a global signal and automatically connected throughout the device, the distributed RAM primitive does not have a GWE input pin.

Attributes

Content Initialization — INIT

By default, distributed RAM is initialized with all zeros during the device configuration sequence. To specify [non-zero] initial memory contents after configuration, use the INIT attributes. Each INIT is a hexadecimal-encoded bit vector, arranged from most-significant to least-significant bit. In other words, the right-most hexadecimal character represents RAM locations 3, 2, 1, and 0. Table 5 shows the length of the INIT attribute for each primitive.

Table 5: **INIT Attributes Length**

Primitive	Template	INIT Attribute Length
RAM16X1S	RAM_16S	4 digits
RAM32X1S	RAM_32S	8 digits
RAM64X1S	RAM_64S	16 digits
RAM16X1D	RAM_16D	4 digits

Placement Location — LOC

Each Spartan-3 CLB contains four slices, each with its own location coordinate, as shown in Figure 4. Distributed RAM fits only in SLICEMs slices. The ‘M’ in SLICEM indicates that the slice supports memory-related functions and distinguishes SLICEMs from SLICELs. The ‘L’ indicates that the slice supports logic only.

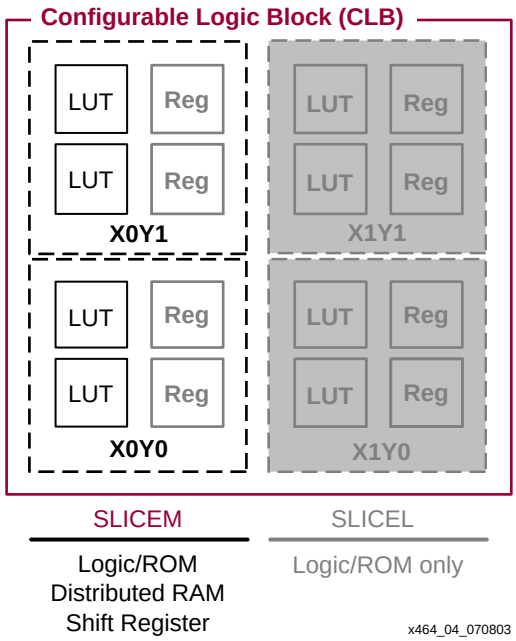

Figure 4: **SLICEM slices within Spartan-3 CLB**

When a LOC property is assigned to a distributed RAM instance, the Xilinx ISE software places the instance in the specified location. Figure 4 shows the X,Y coordinates for the slices in a Spartan-3 CLB. Again, only SLICEM slices support memory.

Distributed RAM placement locations use the slice location naming convention, allowing LOC properties to transfer easily from array to array.

For example, the single-port RAM16X1S primitive fits in any LUT within any SLICEM. To place the instance `U_RAM16` in slice X0Y0, use the following LOC assignment:

```
INST "U_RAM16" LOC = "SLICE_X0Y0";
```

The 16x1 dual-port RAM16X1D primitive requires both 16x1 LUT RAMs within a single SLICEM slice, as shown in Figure 5. The first 16x1 LUT RAM, with output SPO, implements the read/write port controlled by address A[3:0] for read and write. The second LUT RAM implements the independent read-only port controlled by address DPRA[3:0]. Data is presented simultaneously to both LUT RAMs, again controlled by address A[3:0], WE, and WCLK.

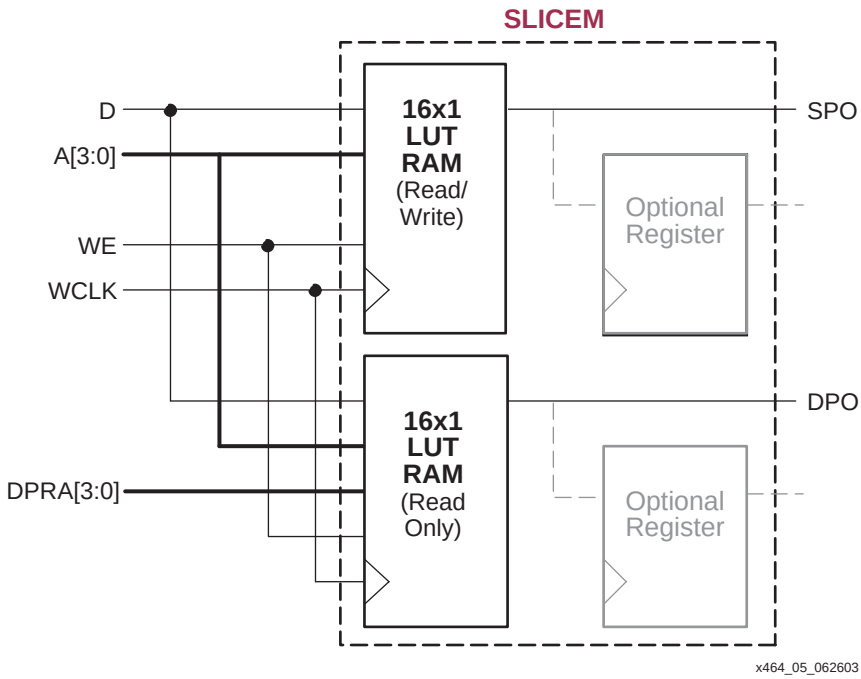

Figure 5: **RAM16X1D Placement**

A 32x1 single-port RAM32X1S primitive fits in one slice, as shown in Figure 6. The 32 bits of RAM are split between two 16x1 LUT RAMs within the SLICEM slice. The A4 address line selects the active LUT RAM via the F5MUX multiplexer within the slice.

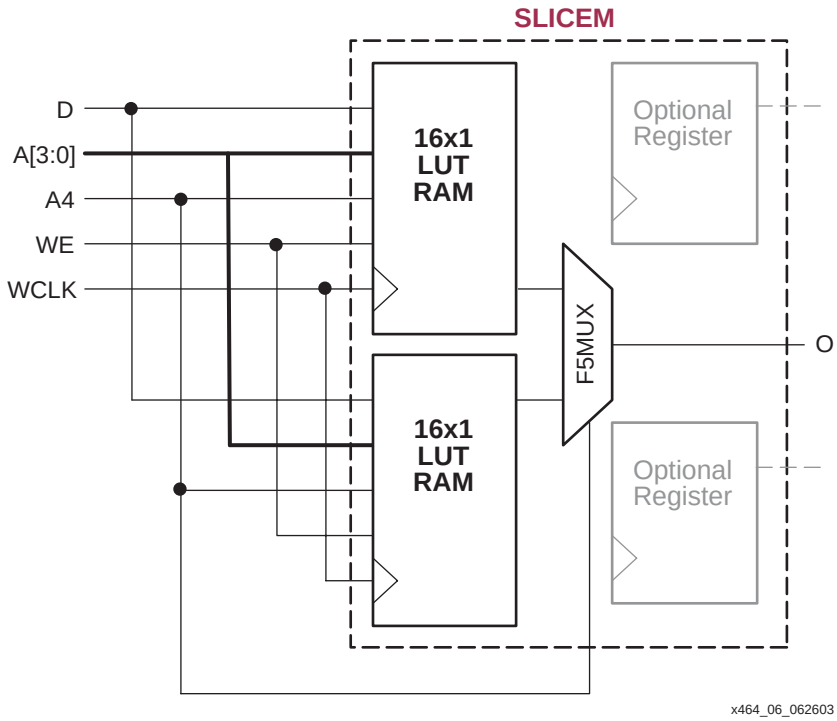

Figure 6: **RAM32X1S Placement**

The 64x1 single-port RAM64X1S primitive occupies both SLICEM slices in the CLB. The read path uses both F5MUX and F6MUX multiplexers within the CLB.

Distributed RAM Design Entry

To specify distributed RAM in an application, use one of the various design entry tools, including the Xilinx CORE Generator software or VHDL or Verilog.

Xilinx CORE Generator System

The Xilinx CORE Generator system creates distributed memory designs for both single-port and dual-port RAMs, ROMs, and even SRL16 shift-register functions.

The Distributed Memory module is parameterizable. To create a module, specify the component name and choose to include or exclude control inputs, then choose the active polarity for the control inputs.

Optionally, specify the initial memory contents. Unless otherwise specified, each memory location initializes to zero. Enter user-specified initial values via a Memory Initialization File, consisting of one line of binary data for every memory location. A default file is generated by the CORE Generator system. Alternatively, create a coefficients file (`.coe`) as shown in Figure 7, which not only defines the initial contents in a radix of 2, 10, or 16, but also defines all the other control parameters for the CORE Generator system.

```
memory_initialization_radix=16;
memory_initialization_vector= 80, 0F, 00, 0B, 00, 0C, …, 81;
```

Figure 7: **A Simple Coefficients File (`.coe`) Example for a Byte-Wide Memory**

The output from the CORE Generator system includes a report on the options selected and the device resources required. If a very deep memory is generated, then some external multiplexing may be required; these resources are reported as the number of logic slices required. For simulation purposes, the CORE Generator system creates VHDL or Verilog behavioral models.

The CORE Generator synchronous and asynchronous FIFO modules support both distributed and block RAMs.

- **CORE Generator**: Distributed Memory module
  http://www.xilinx.com/ipcenter/catalog/logicore/docs/dist_mem.pdf
- **CORE Generator**: Synchronous FIFO module
  http://www.xilinx.com/ipcenter/catalog/logicore/docs/sync_fifo.pdf
- **CORE Generator**: Asynchronous FIFO module
  http://www.xilinx.com/ipcenter/catalog/logicore/docs/async_fifo.pdf

VHDL and Verilog

VHDL and Verilog synthesis-based designs can either infer or directly instantiate block RAM, depending on the specific logic synthesis tool used to create the design.

Inferring Block RAM

Some VHDL and Verilog logic synthesis tools, such as the Xilinx Synthesis Tool (XST) and Synplicity Synplify, infer block RAM based on the hardware described. The Xilinx ISE Project Navigator includes templates for inferring block RAM in your design. To use the templates within Project Navigator, select **Edit → Language Templates** from the menu, and then select **VHDL** or **Verilog**, followed by **Synthesis Templates → RAM** from the selection tree. Finally, select the preferred distributed RAM template. Cut and paste the template into the source code for the application and modify it as appropriate.

It is still possible to directly instantiate distributed RAM, even if portions of the design infer distributed RAM.

Instantiation Templates

For VHDL- and Verilog-based designs, various instantiation templates are available to speed development. Within the Xilinx ISE Project Navigator, select **Edit → Language Templates** from the menu, and then select **VHDL** or **Verilog**, followed by **Component Instantiation → Distributed RAM** from the selection tree. Cut and paste the template into the source code for the application and modify it as appropriate.

There are also downloadable VHDL and Verilog templates available for all single-port and dual-port primitives. The RAM_*x*S templates (where *x* = 16, 32, or 64) are single-port modules and instantiate the corresponding RAM*x*X1S primitive. The ‘S’ indicates single-port RAM. The RAM_16D template is a dual-port module and instantiates the corresponding RAM16X1D primitive. The ‘D’ indicates dual-port RAM.

- VHDL Distributed RAM Templates
  ftp://ftp.xilinx.com/pub/applications/xapp/xapp464_vhdl.zip
- Verilog Distributed RAM Templates
  ftp://ftp.xilinx.com/pub/applications/xapp/xapp464_verilog.zip

The following are single-port templates:

- RAM_16S
- RAM_32S
- RAM_64S

The following is a dual-port template:

- RAM_16D

In VHDL, each template has a component declaration section and an architecture section. Insert both sections of the template within the VHDL design file. The port map of the architecture section must include the design signal names.

Templates for the RAM_16S module are provided below as examples in both VHDL and Verilog code.

VHDL Template Example

```
--
-- Module: RAM_16S
--
-- Description: VHDL instantiation template
-- Distributed RAM
-- Single Port 16 x 1
-- Can also be used for RAM16X1S_1
--
-- Device: Spartan-3 Family
--
-----------------------------------------------------------------------
--
-- Components Declarations:
--
component RAM16X1S
-- pragma translate_off
generic (
-- RAM initialization ("0" by default) for functional simulation:
INIT : bit_vector := X"0000"
);
-- pragma translate_on
port (
    D    : in std_logic;
    WE   : in std_logic;
    WCLK : in std_logic;
    A0   : in std_logic;
    A1   : in std_logic;
    A2   : in std_logic;
    A3   : in std_logic;
    O    : out std_logic
);
end component;
--
-----------------------------------------------------------------------
--
-- Architecture section:
--
-- Attributes for RAM initialization ("0" by default):
attribute INIT: string;
--
attribute INIT of U_RAM16X1S: label is "0000";
--
-- Distributed RAM Instantiation
U_RAM16X1S: RAM16X1S
port map (
    D    => , -- insert Data input signal
    WE   => , -- insert Write Enable signal
    WCLK => , -- insert Write Clock signal
    A0   => , -- insert Address 0 signal
    A1   => , -- insert Address 1 signal
    A2   => , -- insert Address 2 signal
    A3   => , -- insert Address 3 signal
    O    =>   -- insert Data output signal
);
--
-----------------------------------------------------------------------
```

Verilog Template Example

```
//
// Module: RAM_16S
//
// Description: Verilog instantiation template
// Distributed RAM
// Single Port 16 x 1
// Can also be used for RAM16X1S_1
//
// Device: Spartan-3 Family
//
//-----------------------------------------------------------------------
//
// Syntax for Synopsys FPGA Express
// synopsys translate_off
defparam
//RAM initialization ("0" by default) for functional simulation:
U_RAM16X1S.INIT = 16'h0000;
// synopsys translate_on
//Distributed RAM Instantiation
RAM16X1S U_RAM16X1S (
    .D(),     // insert input signal
    .WE(),    // insert Write Enable signal
    .WCLK(),  // insert Write Clock signal
    .A0(),    // insert Address 0 signal
    .A1(),    // insert Address 1 signal
    .A2(),    // insert Address 2 signal
    .A3(),    // insert Address 3 signal
    .O()      // insert output signal
);
// synthesis attribute declarations
/* synopsys attribute
INIT "0000"
*/
```

Wider Distributed RAM Modules

Table 6 shows the VHDL and Verilog distributed RAM examples that implement *n*-bit-wide memories.

Table 6: **VHDL and Verilog Submodules**

Submodules	Primitive	Size	Type
XC3S_RAM16XN_S_SUBM	RAM16X1S	16 words x n-bit	Single-port
XC3S_RAM32XN_S_SUBM	RAM32X1S	32 words x n-bit	Single-port
XC3S_RAM64XN_S_SUBM	RAM64X1S	64 words x n-bit	Single-port
XC3S_RAM16XN_D_SUBM	RAM16X1D	16 words x n-bit	Dual-port

Initialization in VHDL or Verilog Codes

Distributed RAM structures can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attributes are attached to the distributed RAM instantiation and are copied in the EDIF output file to be compiled by Xilinx ISE 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.

Related Materials and References

Refer to the following documents for additional information:

- “Elements within a Slice” and “Function Generator” sections, *Spartan-3 Data Sheet (Module 2)*. Describes the CLB slice structure and distributed RAM function. http://www.xilinx.com/bvdocs/publications/ds099-2.pdf
- *Libraries Guide*, for ISE 6.3i by Xilinx, Inc. Distributed RAM primitives. http://toolbox.xilinx.com/docsan/xilinx6/books/docs/lib/lib.pdf

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/08/03	1.0	Initial Xilinx release.
03/01/05	2.0	Added references to Spartan-3L and Spartan-3E FPGAs.