

---

# **Artisan Standard Library 130nm - 250nm SRAM Generator User Manual**

**Confidential**



**June 2007, Revision 2007q2v1**

Copyright © 2002 - 2007 ARM. All rights reserved. Confidential

---

Copyright © 2002 - 2007 ARM. All rights reserved.

Part Number ug-asl-2513-sram-2007q2v1

### **Proprietary Notice**

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Where the term ARM is used it means “ARM or any of its subsidiaries as appropriate”.

### **Confidentiality Status**

This document is Confidential. This document may only be used and distributed in accordance with the terms of the agreement entered into by ARM and the party that ARM delivered this document to.

# Table of Contents

## Preface

Revision History .....	vii
Customer Support .....	viii
Typographic Conventions .....	ix

## Chapter 1

Overview and Installation .....	1-1
1.1 Overview .....	1-3
1.2 ASL SRAM Generator Features .....	1-4
1.3 Generator Installation .....	1-6
1.3.1 System Requirements .....	1-6
1.3.1.1 Operating System Requirements .....	1-6
1.3.1.2 Disk Space Requirements .....	1-6
1.3.2 Installing the Generator .....	1-7
1.3.2.1 Installation Tasks .....	1-7
1.3.2.2 Directory Structure and Executables .....	1-8

## Chapter 2

Using the Generator .....	2-1
2.1 Overview .....	2-3
2.1.1 Running the Generator from the Graphical User Interface (GUI) .....	2-3
2.1.2 Running the Generator from the Command Line .....	2-4
2.2 Views and Output Files .....	2-5
2.3 GUI Components .....	2-7
2.3.1 Generic Parameters Pane .....	2-8
2.3.2 Views Pane .....	2-8
2.3.3 Relative Footprint Pane .....	2-9
2.3.4 ASCII Datatable Pane .....	2-9

2.3.5 Message Pane .....	2-12
2.3.6 File Menu (Exiting the GUI) .....	2-12
2.3.7 Utilities Menu .....	2-13
2.3.8 Help Menu .....	2-13
2.3.9 Balloon Help .....	2-13
2.4 Generating Views from the GUI .....	2-14
2.4.1 Generating Single Views .....	2-14
2.4.2 Generating Multiple Views.....	2-15
2.4.3 Setting View-Specific Parameters .....	2-16
2.5 Generating Views from the Command Line .....	2-17
2.5.1 View Commands.....	2-17
2.5.2 Generating Multiple Views with View-Specific Options.....	2-18
2.6 Generating Specification and Log Files .....	2-20
2.6.1 Using Specification Files .....	2-20
2.6.2 Creating Log Files .....	2-21
2.6.3 Generating Parameter Information .....	2-22
2.7 Generator Options .....	2-23
2.7.1 Command Line Syntax .....	2-23
2.7.2 Basic Options .....	2-24
2.7.3 Setting Advanced Options .....	2-28
2.7.3.1 Setting Advanced Options from the GUI .....	2-28
2.7.3.2 Setting Advanced Options from the Command Line.....	2-30
2.7.4 Advanced Options.....	2-30
2.7.4.1 Advanced View-Specific Options .....	2-34
2.7.5 Selecting Characterization Corners (130nm ra1shd/ra2shd/ra2shd_rd) .....	2-35
2.7.5.1 Selecting Corners from the GUI .....	2-35
2.7.5.2 Selecting Corners from the Command Line .....	2-36

## Chapter 3

Synchronous SRAM Generator Architecture .....	3-1
3.1 Overview .....	3-3
3.2 Synchronous Single-Port SRAM Architecture and Timing Specifications .....	3-4
3.2.1 Single-Port SRAM Description (ra1sh, ra1shd, ra1sd, ra1sl).....	3-4
3.2.2 Single-Port SRAM Pins (ra1sh, ra1shd, ra1sd, ra1sl).....	3-6
3.2.3 Single-Port SRAM Logic Tables (ra1sh, ra1shd, ra1sd, ra1sl) .....	3-7
3.2.4 Single-Port SRAM Parameters (ra1sh, ra1shd, ra1sd, ra1sl).....	3-7
3.2.5 Single-Port High Speed/Density SRAM Block Diagrams (ra1sh, ra1shd, ra1sd).....	3-10
3.2.6 Single-Port Low-Power SRAM Block Diagrams (ra1sl) .....	3-11
3.2.7 Single-Port High-Speed/Density SRAM Core Address Maps (ra1sh, ra1shd) .....	3-12
3.2.8 Single-Port High-Density SRAM Core Address Maps (ra1sd) .....	3-16
3.2.9 Single-Port Low-Power SRAM Core Address Maps (ra1sl).....	3-19
3.2.10 Single-Port SRAM Timing Specifications (ra1sh, ra1shd, ra1sd, ra1sl) .....	3-20
3.2.10.1 Single-Port SRAM Timing Diagrams (ra1sh, ra1shd, ra1sd, ra1sl) .....	3-20
3.2.10.2 Single-Port SRAM Timing Parameters (ra1sh, ra1shd, ra1sd, ra1sl).....	3-23
3.2.10.3 Single-Port SRAM Power Parameters (ra1sh, ra1shd, ra1sd, ra1sl) .....	3-24
3.3 Synchronous Dual-Port SRAM Architecture and Timing Specifications .....	3-25
3.3.1 Dual-Port SRAM Description (ra2sh, ra2shd, ra2shd_rd, ra2sd) .....	3-25
3.3.2 Dual-Port SRAM Pins (ra2sh, ra2shd, ra2shd_rd, ra2sd).....	3-27
3.3.3 Dual-Port SRAM Logic Tables (ra2sh, ra2shd, ra2shd_rd, ra2sd).....	3-28
3.3.4 Dual-Port SRAM Parameters (ra2sh, ra2shd, ra2shd_rd, ra2sd).....	3-28
3.3.5 Dual-Port SRAM Block Diagrams (ra2sh, ra2shd, ra2shd_rd, ra2sd).....	3-31
3.3.6 Dual-Port SRAM Core Address Maps (ra2sh, ra2shd, ra2shd_rd, ra2sd) .....	3-32
3.3.7 Dual-Port SRAM Timing Specifications (ra2sh, ra2shd, ra2shd_rd, ra2sd) .....	3-35
3.3.7.1 Dual-Port SRAM Clock Timing Diagrams (ra2sh, ra2shd, ra2shd_rd, ra2sd).....	3-35
3.3.7.2 Dual-Port SRAM Timing Diagrams (ra2sh, ra2shd, ra2shd_rd, ra2sd) .....	3-37
3.3.7.3 Dual-Port SRAM Timing Parameters (ra2sh, ra2shd, ra2shd_rd, ra2sd) .....	3-39
3.3.7.4 Dual-Port SRAM Power Parameters (ra2sh, ra2shd, ra2shd_rd, ra2sd).....	3-40

3.4 SRAM Power Structure (ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2shd_rd, ra2sd) .....	3-41
3.4.1 Current Calculations .....	3-41
3.4.2 Power Distribution Methodology .....	3-42
3.4.3 Supply Connections to Power Rings .....	3-44
3.4.4 Noise Limits .....	3-45
3.5 SRAM Physical Characteristics (ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2shd_rd, ra2sd) .....	3-46
3.5.1 Top Metal Layer .....	3-46
3.5.2 I/O Connections .....	3-46
3.5.3 Characterization Environments .....	3-47
3.6 SRAM Timing Derating (ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2sd) .....	3-48

## Chapter 4

Generator Views .....	4-1
4.1 Overview .....	4-3
4.2 Tool Verification .....	4-4
4.3 Using the Generator Views .....	4-5
4.3.1 Using the Verilog Model .....	4-5
4.3.2 Using the VHDL Model .....	4-6
4.3.3 Using the Synopsys Model to Generate SDF .....	4-8
4.3.4 Using the Pearl Model to Generate SDF .....	4-9
4.3.5 Using the PrimeTime Model to Generate SDF .....	4-10
4.3.6 Loading the VCLEF Description into Silicon Ensemble .....	4-10
4.3.7 Using Apollo with ARM's Artisan Generators .....	4-11
4.3.7.1 Loading the VCLEF Description into the Milkyway Database .....	4-11
4.3.7.2 Loading the GDSII Layout into the Milkyway Database .....	4-12
4.3.8 Loading the GDSII Layout into a DFII Library .....	4-13
4.3.9 Using the LVS Netlist .....	4-13
4.3.10 Using Hierarchical LVS .....	4-14
4.3.11 Using the Repair Verilog Model .....	4-16
4.3.11.1 Repair Verilog Simulation .....	4-16
4.3.11.2 Repair Verilog Synthesis .....	4-16
4.3.12 Using the Repair VHDL Model .....	4-18
4.3.12.1 Repair VHDL Simulation .....	4-18
4.3.12.2 Repair VHDL Synthesis .....	4-18

---

# Preface

## Revision History

The following table provides the revision history for this manual.

Part Number	Updates (to template)
ug_2002q4v0	<ul style="list-style-type: none"><li>• Initial Release</li></ul>
ug_2002q4v1	<ul style="list-style-type: none"><li>• Drive strength text updated</li></ul>
ug_2003q1v0	<ul style="list-style-type: none"><li>• Table 4-1 updated</li></ul>
ug_2003q2v0	<ul style="list-style-type: none"><li>• EDA package 3.1 added</li></ul>
ug_2003q2v1	<ul style="list-style-type: none"><li>• Text and illustrations for new features added</li></ul>
ug_2003q3v0	<ul style="list-style-type: none"><li>• Format and text updates</li></ul>
ug_2003q3v1	<ul style="list-style-type: none"><li>• Updated wp_size information, for use with redundancy.</li></ul>
ug_2003q3v1.1	<ul style="list-style-type: none"><li>• Equation update in power structure section</li></ul>
ug_2003_q4v2	<ul style="list-style-type: none"><li>• Added low-power and 90nm information where applicable</li></ul>
ug_2003_q4v2.1	<ul style="list-style-type: none"><li>• .hcell file removed from output list</li></ul>
ug_2003_q4v2.2	<ul style="list-style-type: none"><li>• RA2 and RF2 Read/Write Behavior tables updated</li></ul>
ug_2004q1v0	<ul style="list-style-type: none"><li>• Revised RA2 block diagram and RF2 180nm parameters</li></ul>
ug_2004q1v1	<ul style="list-style-type: none"><li>• Revised 90nm SRAM <math>t_{cyc[0-3]_{art}}</math> description</li></ul>
ug_2004q2v0	<ul style="list-style-type: none"><li>• General text revisions</li></ul>
ug_2004q2v1	<ul style="list-style-type: none"><li>• CLI text change for -diodes command</li></ul>
ug_2004q3v1	<ul style="list-style-type: none"><li>• Update Apollo section, Chapter 4</li></ul>
ug_asl_sram_2513_2004q4v1	<ul style="list-style-type: none"><li>• Comprehensive 130nm ra1shd and ra2shd upgrades</li></ul>
ug_asl_sram_2513_2005q1v1	<ul style="list-style-type: none"><li>• Add 130nm DP SRAM redundancy</li></ul>
ug_asl_sram_2513_2005q2v1	<ul style="list-style-type: none"><li>• New Advanced Options GUI features; updated power parameters</li></ul>
ug_asl_sram_2513_2005q2v2	<ul style="list-style-type: none"><li>• Company name/logo changed</li></ul>
ug_asl_sram_2513_2005q2v3	<ul style="list-style-type: none"><li>• Udated Linux support statement, dpccm and asvm text, and behavior table</li></ul>

---

Part Number	Updates (to template)
ug-asl-2513-sram-2005q4v1	• Added Confidentiality statements to footer and cover.
ug-asl-2513-sram-2006q1v1	• Format upgrades
ug-asl-2513-sram-2007q2v1	• Update Copyright and Preface

## Customer Support

For general questions related to ARM Physical IP product availability and their licensing requirements, customers can go to <http://access.arm.com> Log in and click on the “Products and Services > New Business Request” link to enter a request.

Customers with active Support contracts can obtain support for ARM Physical IP products by going to <http://access.arm.com> and clicking on “Products and Services > New Technical Request.” Support contract options are available for review at [www.arm.com/products/physicalip/support.html](http://www.arm.com/products/physicalip/support.html).

You may also contact ARM by telephone or email:

- United States and North America      1-877-ARTILIB (1-877-278-4542)
- International      1-408-548-3298
- email      support-artisan@arm.com



---

# Typographic Conventions

The following typographic conventions are used to assist you in distinguishing special notations, values, and elements described in this manual.

Visual Cue	Meaning
(Bullet) •	Bulleted list of important items.
Courier Type	Commands typed on the keyboard, either examples or instructions.
Dash (- ) Courier Type	Text set in Courier type and preceded by a dash represents a command name (for example, -libname).
<italic type> italic type	Variable names you select, such as file and directory names are enclosed within angle brackets ( < > ). Italic type is used to show variable values, file, and directory names.
(Ellipsis) ...	Indicates commands or options that may be added.
Italic Type with Initial Capital Letters	Document, chapter, section and reference manual names.



---

# 1

## Overview and Installation

This chapter contains the following sections:

- “Overview” on page 1-3
- “ASL SRAM Generator Features” on page 1-4
- “Generator Installation” on page 1-6

## 1.1 Overview

ARM designs the technology that lies at the heart of advanced digital products, from wireless, networking and consumer entertainment solutions to imaging, automotive, security and storage devices. ARM's comprehensive product offering includes 16/32-bit RISC microprocessors, data engines, 3D processors, digital libraries, embedded memories, peripherals, software and development tools, as well as analog functions and high-speed connectivity products. Combined with the company's broad Partner community, they provide a total system solution that offers a fast, reliable path to market for leading electronics companies.

This manual provides information on using ARM's Artisan Standard Library (ASL) 130nm to 250nm SRAM generators to create instances with a variety of parameters and views. This manual provides information about single and dual-port, high speed/density and high density SRAM generators. Where applicable, this manual also provides information for low-power single-port SRAM generators.

——— **Note** ———

Parameters or specifications for your generator may differ from the default ASL generators. Information about deviations to the ASL generators can be found in the README text file that is enclosed with your generator or in an addendum attached to this manual.

See the following sections for more detailed information about this generator.

- This chapter provides basic information about ASL SRAM generators, such as features and views, plus important information about installation requirements and tasks.
- Chapter 2, “Using the Generator,” - Provides details about using the generator GUI or the command line to generate views and instances.
- Chapter 3, “Synchronous SRAM Generator Architecture,” - Lists the architectural details, physical characteristics of memory instances, and characterization/timing information.
- Chapter 4, “Generator Views,” - Lists the tool versions supported by the generator and provides instructions on generating specific views.

## 1.2 ASL SRAM Generator Features

ASL memory generators include the following features:

- Optimized for High Speed/High Density or Low-Power or High Density
- Aspect Ratio Control for Efficient Floor Planning
- Memory Operation and Retention at Low Frequency (Down to 0MHz)
- Low Active Power and Leakage-Only Standby Power
- Timing and Power Models for Industry-Leading Design Tools
- Configurable Word-Write Mask Option
- Flex-Repair™ Redundancy Option (available with a separate Fuse Box generator).  
This redundancy and repair feature is available with most Artisan synchronous single-port SRAM as well as most synchronous 130nm dual-port SRAM generators.

A standard set of views can be generated from ASL SRAM generators. These views are supported by generators verified with the tools defined in the applicable EDA package; EDA tools and version specifics are detailed in the README file. See Chapter 4, “Generator Views,” for details about these tools and using the views. Optional support is available and can be installed in most existing generators without installing a completely new generator. As needed, see the README file in your generator to determine the EDA or tool version(s) for your generator.

Standard and optional tool support is listed below.

**Standard Support**

PostScript Datasheet  
ASCII Datatable  
  
Synopsys Liberty  
Synopsys PrimeTime (STAMP)  
Cadence TLF  
  
Verilog  
VHDL  
  
Repair Verilog  
Repair VHDL  
  
VCLEF Footprint  
GDSII Layout  
LVS Netlist

**Optional Support**

LogicVision IC Memory BIST  
Mentor FastScan  
Mentor MBISTArchitect  
Synopsys TetraMAX

———— **Note** ————

The Repair Verilog and Repair VHDL tools are supported by ARM's Artisan 130nm and 180nm synchronous single-port, and 130nm synchronous dual-port SRAM generators with Flex-Repair.

## 1.3 Generator Installation

This section provides information about system requirements, installation tasks, directory structure, and generator terminology.

### 1.3.1 System Requirements

Make sure that your operating system and disk (CPU) space allocation meet generator requirements to ensure proper functioning of the generator, as described in the following sections.

#### 1.3.1.1 Operating System Requirements

The following operating system(s) support the applicable EDA package:

- Solaris 8 and the latest SunOS patches

——— **Note** ———

If your operating system does not meet the requirements stated above, you may receive an error message when running the memory generator. In this case, the generator will not function until you upgrade your operating system.

To determine the name and version of your SunOS operating system, enter the following command:

```
uname -a
```

- If LINUX is supported by this generator, LINUX information will be included in the EDA Support section of the generator README.

#### 1.3.1.2 Disk Space Requirements

Make sure you have enough disk space available for your installation. There are different space requirements for the various stages you must complete before you can use the generator.

A generator requires approximately 50 megabytes when you copy it to the installation directory. When you uncompress the generator file approximately 110 megabytes is required. When you extract (tar) the generator file you will have the original file and the uncompressed/extracted version in the installation directory, and will need approximately 160 megabytes of disk space.

## 1.3.2 Installing the Generator

You must determine where you want to install the generator on your system. In this manual, `<install_dir>` refers to the directory you choose for installation. You may also create a working directory, `<working_dir>`, where you actually run the generator to create memory instances.

————— **Note** —————

When copying the installation files, you should create a new directory. Overwriting an existing generator directory may corrupt the generator installation.

### 1.3.2.1 Installation Tasks

Change to the installation directory:

```
cd <install_dir>
```

Uncompress the installation files:

```
gunzip <install_files>.tgz
```

where `<install_files>` represents the generator installation files in your installation directory.

Extract the installation files:

```
tar -xvf <install_files>.tar
```



### 1.3.2.2 Directory Structure and Executables

Installing the generator produces the following directory structure.

aci/<executable>/\*

- bin/    This directory contains the generator executable and platform-specific directories.
- lib/    This directory contains technology files, library files, executables, and subdirectories.
  - vhdl\_lib/    This directory contains the VHDL library file, *artisan\_lib.vhd*.
- doc/    This directory contains generator documentation.

where <executable> refers to the name of the file that is run to generate an instance.

——— **Note** ———

The Artisan Generator GUI is written in Java. For your convenience, the generator source tree includes copies of all required JRE distributions.

The following table provides the general names and executables for available memory generators. Check your generator GUI; the names and executables provided in your generator GUI always supercede those in the table below

Generator	Product Name	Executable
High-Speed/Density Single-Port SRAM	SRAM-SP	ra1sh or ra1shd
High-Speed/Density Dual-Port SRAM	SRAM-DP	ra2sh or ra2shd
High-Speed/Density Dual-Port SRAM with redundancy	SRAM-DP-RD	ra2shd_rd
High-Density Single-Port SRAM	SRAM-SP-HD	ra1sd
High-Density Dual-Port SRAM	SRAM-DP-HD	ra2sd
Low-Power Single-Port SRAM	SRAM-SP-LP	ra1sl



## 2

# Using the Generator

This chapter contains the following sections:

- “Overview” on page 2-3
- “Views and Output Files” on page 2-5
- “GUI Components” on page 2-7
- “Generating Views from the GUI” on page 2-14
- “Generating Views from the Command Line” on page 2-17
- “Generating Specification and Log Files” on page 2-20
- “Generator Options” on page 2-23

## 2.1 Overview

ARM's Artisan memory generators provide integrated circuit designs with the highest levels of density, speed, and power. A wide range of features provides several options, including the ability to increase chip reliability and yield. The generators tailor instances with a large variety of selectable features and create a comprehensive set of views to fit in prevalent EDA tools and flows. You can run the generator by invoking the graphical user interface (GUI) or from the command line. This chapter provides information on using the generator to tailor instances to your design needs.

### 2.1.1 Running the Generator from the Graphical User Interface (GUI)

The generator GUI allows you to configure all generator parameters and generate all views from a single graphical interface. The output views, along with a log file, are placed in the current working directory.

This manual assumes you have added `<install_dir>/aci/<executable>/bin` to your UNIX search path. If you do not wish to do this, preface all generator commands with `<install_dir>/aci/<executable>/bin/`.

To start the GUI from the shell, type:

```
% cd <working_dir>
% <executable>
```

where:

`<working_dir>` refers to the directory where you choose to run the generator. Generator output files are created in this directory; therefore, ARM strongly recommends that you run the generator in a working directory that is different from the source directory `<install_dir>`.

See the table in “Directory Structure and Executables” on page 2-8 for a list of standard generator executables.

## 2.1.2 Running the Generator from the Command Line

You can use command-line options to set parameter values, generate views and use view-specific options.

The syntax described below applies to all options, parameter values and views generated from the command line. All option names and parameter values are case-sensitive.

*<executable> <view\_command> <-option><option\_value> ...*

Commands that generate views do not require a dash (-) in front of the command. Options that set parameters, such as mux or word values, do require a dash.

For example, to obtain an `ra1sh` (or `ra1sl`) instance with Verilog view, mux = 8, words = 256, type:

```
ra1sh verilog -mux 8 -words 256
```

For low-power single-port SRAM memory, replace `ra1sh` with `ra1sl` in the statement above.

The table in “Directory Structure and Executables” on page 2-8 provides a list of standard generator executables. See “Generating Views from the Command Line” on page 2-17 for details about specific commands you can use to generate specific views.

## 2.2 Views and Output Files

You can generate a variety of views from the GUI or the command line. Each view may consist of one, or more, output file. You can apply basic and advanced options or parameters to each view. See "Generating Views from the GUI" on page 2-14, "Generating Views from the Command Line" on page 2-17, and "Generator Options" on page 2-23 for details about adding these parameters to your views.

The following table lists standard and optional views you can generate and output file(s) associated with each view. The instance name is the executable name, in capital letters.

**Table 2-1. Views and Output Files**

<b>Standard Views</b>	
<b>View</b>	<b>Output Files</b>
PostScript datasheet	<instance_name>.ps
ASCII datatable	<instance_name>.dat
GDSII layout file	<instance_name>.gds2
LVS netlist	<instance_name>.cdl
Synopsys model for each corner <sup>1,2,3</sup>	<instance_name>_<corner>_syn.lib
PrimeTime (STAMP) model for each corner <sup>1,2</sup>	<instance_name>.mod <instance_name>_<corner>.data
TLF model for each corner <sup>1,2,3</sup>	<instance_name>_<corner>.tlf
VCLEF footprint	<instance_name>.vclef <instance_name>_ant.lef <instance_name>_ant.clf
Verilog model	<instance_name>.v
VHDL model	<instance_name>.vhd
Repair Verilog <sup>4</sup>	<instance_name>_fr.v
Repair VHDL <sup>4</sup>	<instance_name>_fr.vhd
<b>Optional Views</b>	
<b>View</b>	<b>Output Files</b>
LogicVision IC Memory BIST	<instance_name>.memlib
Mentor MBIST Architect	<instance_name>.mbist
Synopsys TetraMAX <sup>5</sup>	<instance_name>.tv
Mentor FastScan	<instance_name>.fastscan

<sup>1</sup>You can create timing models using any of the Process Voltage Temperature (PVT) corners for which the memory generator was characterized. The generator may support more than four characterization corners; however, you can only create timing models for four corners at a time. The characterization corner name (that is, slow, fast, fast@-40C, typical) is inserted into the output filename (for example, ralsh\_<corner>\_syn.lib).

<sup>2</sup>The typical and slow Synopsys and TLF models are generated with maximum delays, and the fast model is generated with minimum delays. ARM recommends that critical path, setup, and hold analysis be performed for all applicable corners.

<sup>3</sup>Depending on chip design, overall chip level worst case power conditions can occur under the fast corner (PVT conditions) or under the “Maximum Static Power” corner condition. The worst case static power occurs under the maximum temperature, fast process, and maximum VDD. You may need to perform chip level power analysis under both the fast and “static power” corners to determine the maximum overall power dissipation, AC plus static, for your design.

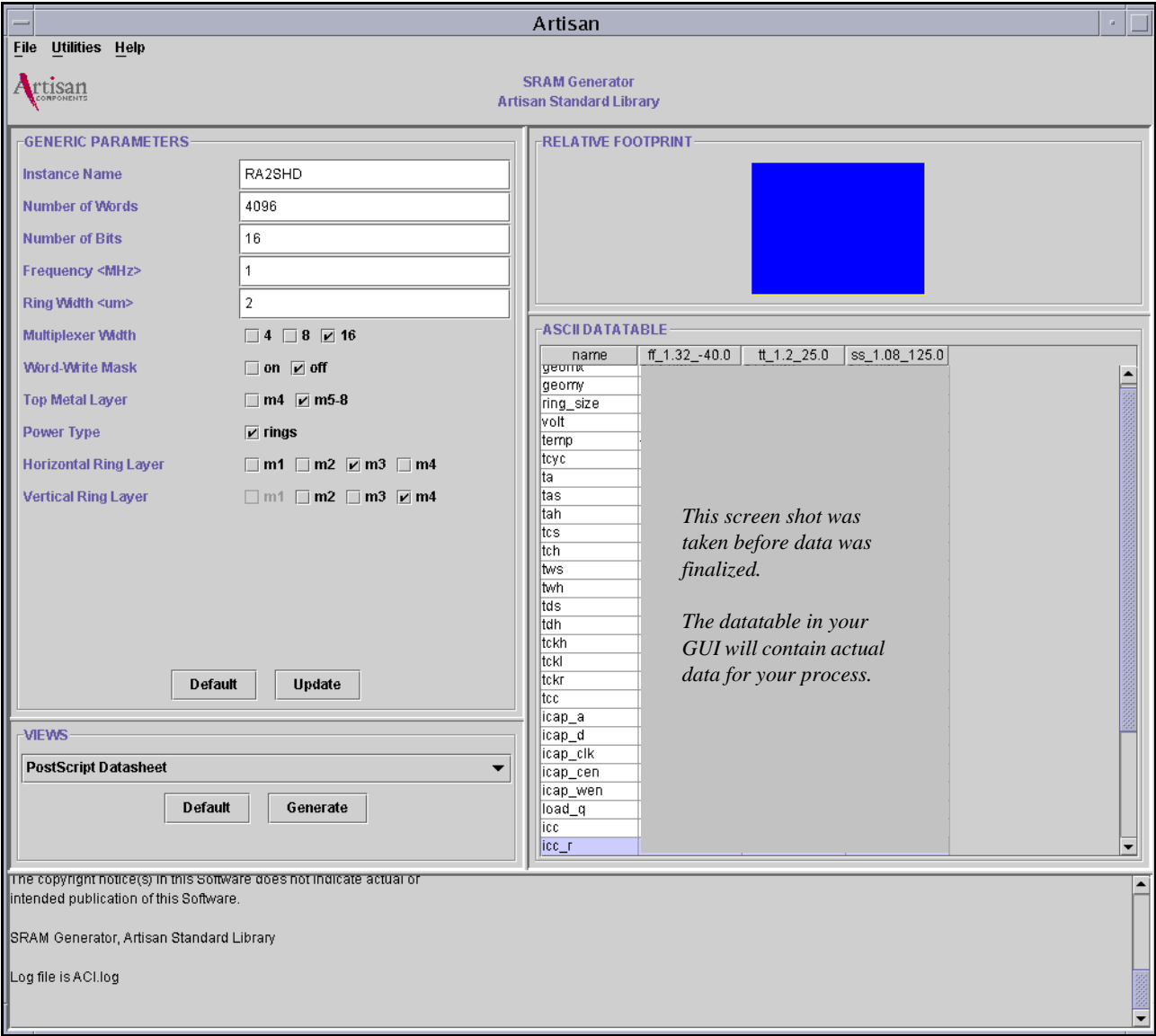
<sup>4</sup>The Repair Verilog and Repair VHDL tools are supported by ARM’s Artisan 130nm and 180nm synchronous single-port and 130nm synchronous dual-port SRAM generators with Flex-Repair.

<sup>5</sup>This optional support is available for free for 180nm, 150nm, and 130nm processes to ARM’s Artisan Access (Free) Library Program licensees under ARM’s Artisan EDAPlus programs based on agreements with our EDA partners.

## 2.3 GUI Components

A sample SRAM generator GUI is shown in Figure 2-1. The GUI for your generator may not look exactly the same as the samples in this chapter. For instance, your generator may have additional characterization corners or features that are not enabled. You can resize the GUI by clicking on its border and dragging it to the desired position.

Figure 2-1. Example: SRAM Generator GUI<sup>1</sup>



<sup>1</sup> Multiplexer Width values may vary.



### 2.3.1 Generic Parameters Pane

The *Generic Parameters* pane of the GUI contains standard input fields and check boxes. The generic parameters are the most commonly used parameters used to configure a generator instance. See Figure 2-1 on page 2-7. You can change the value of a generic parameter by typing the new value in the input field or by selecting the box corresponding to the desired value for each option.

When you want to submit the values of the generic parameters and update the ASCII datatable, click on the *Update* button in the *Generic Parameters* pane.

For example, you can generate views for a specific instance with 256 words, 16 bits, and multiplexer width 8. Enter “256” in the *Number of Words* field, “16” in the *Number of Bits* field, and select the box corresponding to “8” for multiplexer width. Leave all other parameters set to their default values. Click on the *Update* button.

Be sure to enter values that are within the pre-determined ranges for your generator. See Chapter 3, "Synchronous SRAM Generator Architecture," for parameter ranges. You can also use the message pane in the generator GUI to determine parameter ranges. If you attempt to generate views with an out-of-range value, a message identifying the legal (valid) range is displayed in the message pane.

To reset the generic parameters to their default values, click on the *Default* button in the *Generic Parameters* pane.

### 2.3.2 Views Pane

You can generate a single view at a time from the *Views* pane in the generator GUI. To generate a single view, select the view you want from the *Views* pull-down menu and click on the *Generate* button in the *Views* pane. The corresponding view is generated and placed in the current working directory *<working\_dir>*. A list of available views and output files is shown in Table 2-1 on page 2-5. For detailed information about using these views see Chapter 4, "Generator Views,".

You can also generate multiple views at one time. See "Generating Multiple Views" on page 2-15.

You can cancel a view generation from the GUI. When a view is being generated, a window displays a message stating which view is being generated, and a *Cancel* button. Click on the *Cancel* button to cancel generating that view.

### 2.3.3 Relative Footprint Pane

The *Relative Footprint* pane of the GUI shows how the aspect ratio of the SRAM changes as the words, bits, and mux parameters are varied. See Figure 2-1 on page 2-7, which shows the relative footprint in the top right-hand corner of the GUI.

When you change a generic parameter and press the Update button, the relative footprint is automatically updated. The instance without a power ring is shown in the darker color. The power ring is shown in the lighter color.

### 2.3.4 ASCII Datable Pane

When you invoke the GUI, values for the default instance are displayed in the *ASCII Datable* pane. When you change the generic values in the GUI, and click on the *Update* button in the *Generic Parameters* pane, the ASCII datatable automatically updates to reflect the new values.

You can obtain a print-ready copy of these values in two ways. From the Views pane of the GUI, select PostScript datasheet or ASCII datatable. The resulting datasheet (`<instance_name>.ps`) or text file (`<instance_name>.dat`) show the values in the datatable.

Figures 2-2 to 2-4 show sample ASCII datatable panes. The corners and parameters shown may not be the same as those in your ASL generator GUI. Information about minor deviations to the ASL generators can be found in the README text file that is enclosed with your generator or in an addendum attached to this manual.

In the ASCII datable and postscript data sheets, non-power values are displayed in decimal format. Current/power values in datatables and datasheets may be shown in scientific notation or in decimal format.

Figure 2-2. Example: ASCII Datable Panes (except 130nm ra1shd/ra2shd)

**ASCII DATABLE (Left)**

name	fast	typical	slow
geomx	585.820	585.820	585.820
geomy	762.035	762.035	762.035
ring_size	5.200	5.200	5.200
icc	0.075	0.068	0.062
icc_r	0.068	0.062	0.056
icc_w	0.082	0.075	0.069
icc_peak	156.576	99.305	62.562
icc_desel	0.000	0.000	0.000
tcyc	1.026	1.569	2.594
ta	0.994	1.619	2.622
tas	0.292	0.456	0.792
tah	0.054	0.066	0.076
tcs	0.321	0.457	0.719
tch	0.000	0.000	0.000
tws	0.311	0.470	0.752
twh	0.000	0.000	0.000
tds	0.159	0.280	0.514
tdh	0.000	0.000	0.000
thz	0.520	0.730	1.145
tlz	0.471	0.668	1.032
tckh	0.098	0.147	0.254
tckl	0.154	0.234	0.382
tckr	4.000	4.000	4.000
load_q	0.316	0.436	0.641
icap_a	0.053	0.053	0.050

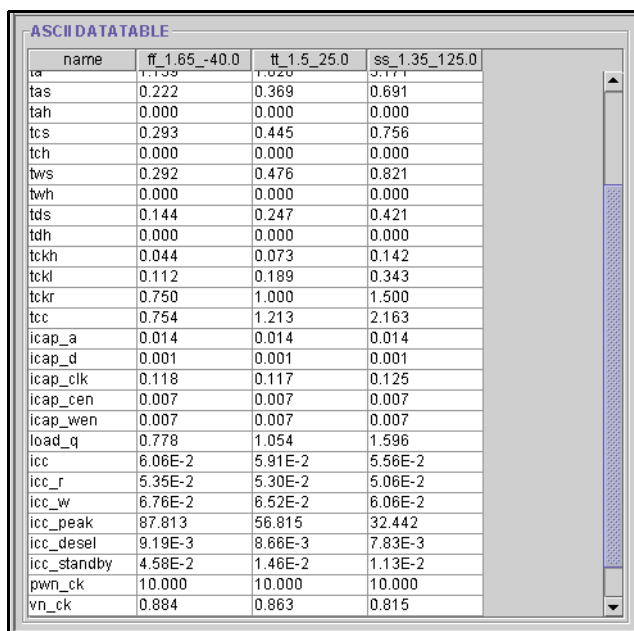
**ASCII DATABLE (Right)**

name	fast@-40C	fast@0C	typical	slow
geomx	416.805	416.805	416.805	416.805
geomy	534.195	534.195	534.195	534.195
ring_size	5.200	5.200	5.200	5.200
icc	0.043	0.044	0.037	0.032
icc_r	0.039	0.040	0.033	0.029
icc_w	0.046	0.048	0.040	0.035
icc_peak	98.536	94.415	66.527	36.463
icc_desel	0.008	0.009	0.007	0.008
icc_stand...	0.000	0.001	0.000	0.002
tcyc	1.094	1.179	1.719	2.932
ta	0.973	1.047	1.694	2.892
tas	0.179	0.190	0.301	0.547
tah	0.009	0.009	0.002	0.000
tcs	0.268	0.286	0.403	0.647
tch	0.000	0.000	0.000	0.000
tws	0.269	0.276	0.392	0.612
twh	0.000	0.000	0.000	0.000
tds	0.124	0.132	0.217	0.364
tdh	0.000	0.000	0.000	0.000
tckh	0.037	0.042	0.058	0.097
tckl	0.106	0.112	0.170	0.289
tckr	4.000	4.000	4.000	4.000
load_q	0.520	0.538	0.713	1.059
icap_a	0.019	0.019	0.018	0.017
icap_d	0.002	0.002	0.002	0.001

Figure 2-3. Example: 130nm (ra1shd/ra2shd) ASCII Datable Pane

**ASCII DATABLE**

name	ff_1.32_-40.0	tt_1.2_25.0	ss_1.08_125.0
geomx	613.390	613.390	613.390
geomy	629.885	629.885	629.885
ring_size	5.200	5.200	5.200
volt	1.320	1.200	1.080
temp	-40.000	25.000	125.000
tcyc	0.985	1.474	2.413
ta	1.082	1.611	2.553
tas	0.200	0.315	0.518
tah	0.008	0.002	0.000
tcs	0.245	0.357	0.533
tch	0.000	0.000	0.000
tws	0.274	0.408	0.646
twh	0.000	0.000	0.000
tds	0.142	0.249	0.441
tdh	0.000	0.000	0.000
tckh	0.044	0.067	0.120
tckl	0.084	0.144	0.247
tckr	0.750	1.000	1.500
tcc	0.736	1.109	1.798
icap_a	0.019	0.018	0.017
icap_d	0.002	0.002	0.002
icap_clk	0.176	0.157	0.138
icap_cen	0.008	0.008	0.008
icap_wen	0.009	0.008	0.008
load_q	0.557	0.782	1.114
icc	0.055	0.075	0.370
icc_r	0.051	0.072	0.367

**Figure 2-4. Example: ASCII Datable Pane with Scientific Notation**


The screenshot shows a window titled "ASCII DATABLE" containing a table with four columns: "name", "ff\_1.65\_-40.0", "tt\_1.5\_25.0", and "ss\_1.35\_125.0". The table lists various parameters such as tas, tah, tcs, tch, tws, twh, tds, tdt, tckh, tckl, tckr, tcc, icap\_a, icap\_d, icap\_clk, icap\_cen, icap\_wen, load\_q, icc, icc\_r, icc\_w, icc\_peak, icc\_desel, icc\_standby, pwn\_ck, and vn\_ck. The values are displayed in scientific notation where applicable.

name	ff_1.65_-40.0	tt_1.5_25.0	ss_1.35_125.0
tas	0.222	0.369	0.691
tah	0.000	0.000	0.000
tcs	0.293	0.445	0.756
tch	0.000	0.000	0.000
tws	0.292	0.476	0.821
twh	0.000	0.000	0.000
tds	0.144	0.247	0.421
tdt	0.000	0.000	0.000
tckh	0.044	0.073	0.142
tckl	0.112	0.189	0.343
tckr	0.750	1.000	1.500
tcc	0.754	1.213	2.163
icap_a	0.014	0.014	0.014
icap_d	0.001	0.001	0.001
icap_clk	0.118	0.117	0.125
icap_cen	0.007	0.007	0.007
icap_wen	0.007	0.007	0.007
load_q	0.778	1.054	1.596
icc	6.06E-2	5.91E-2	5.56E-2
icc_r	5.35E-2	5.30E-2	5.06E-2
icc_w	6.76E-2	6.52E-2	6.06E-2
icc_peak	87.813	56.815	32.442
icc_desel	9.19E-3	8.66E-3	7.83E-3
icc_standby	4.58E-2	1.46E-2	1.13E-2
pwn_ck	10.000	10.000	10.000
vn_ck	0.884	0.863	0.815

You can resize the columns in the ASCII datatable by clicking on the column border and dragging it to the desired width. The columns can be rearranged by clicking on the header cell of a column and dragging it to the desired position.

The “name” column lists the acronym for a characterized parameter. The other columns contain values for these parameters at selectable PVT corners. Characterized parameters are described in the “Timing Parameters” section in Chapter , "Synchronous SRAM Generator Architecture,". Also, by moving your cursor over the parameter name in the GUI, you can get a brief description of that parameter.

The units for parameters in the ASCII datatable are listed in Table 2-2.

**Table 2-2. ASCII Datable Units**

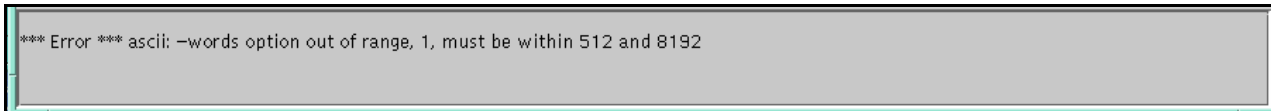
Parameters	Units
Geometry	Microns
Current-consumption	Milliamperes
Timing	Nanoseconds

### 2.3.5 Message Pane

The message pane is located at the bottom of the GUI frame. This pane displays messages when you generate a new instance. Messages include information about successfully generated views and associated output files, an updated ASCII datatable, and when generic parameters are reset to their default values.

The message pane also displays error messages, such as when an invalid value is entered into the GUI or when view generation is not successful. For example, if you enter “1” in the *Number of Words* field, and click the *Update* button, the error message in the message pane indicates that this value is out of range, as shown in Figure 2-5. The valid range, 512 to 8192 in this case, is also provided.

**Figure 2-5. Example: Message Pane**



The log file stores all the messages that appear in the message pane. You can clear the message pane by selecting the *Utilities* Menu, then selecting *Purge Message Area*. The messages are still retained in the log file.

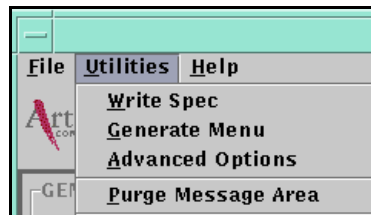
### 2.3.6 File Menu (Exiting the GUI)

To exit the GUI, select the *File* pull-down menu, then select *Exit*.

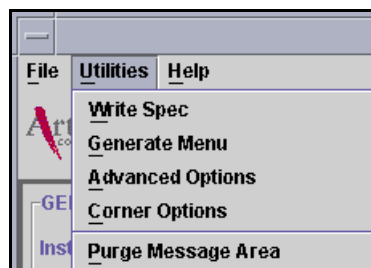
### 2.3.7 Utilities Menu

You can use the Utilities Menu to access other menus and options. You can use it to write specification files, generate multiple views, select corners, and set advanced options. Figures 2-6 and 2-7 show sample Utilities pull-down menus.

**Figure 2-6. Example: Utilities Pull-Down Menu (except 130nm ra1shd/ra2shd)**



**Figure 2-7. Example: 130nm (ra1shd/ra2shd) Utilities Pull-Down Menu**



See "Using Specification Files" on page 2-20, "Generating Multiple Views" on page 2-15, "Setting Advanced Options from the GUI" on page 2-28, for details on how to perform these tasks.

### 2.3.8 Help Menu

The *Help* pull-down menu displays a list of documents that are shipped, in electronic format, with the GUI. When you select a document the Adobe PDF reader, *acroread*, launches to open the document. If the document does not display properly, ask your system administrator to ensure that *acroread* is available to you and is in your path.

### 2.3.9 Balloon Help

The GUI contains balloon help messages that give brief explanations of generator features. The balloon help messages appear as your mouse pauses over an active area such as ASCII datatable parameters. Pausing your mouse over the ASCII datatable allows you to view brief descriptions of the parameters and the process-temperature-voltage (PVT) data for each characterization environment (corner).

## 2.4 Generating Views from the GUI

You can create single or multiple views with specific parameters from the generator GUI.

### 2.4.1 Generating Single Views

You can create a single view for each instance directly from the GUI. For each new instance, update the text fields and check boxes in the Generic Parameters pane and click *Update*. In the Views pane, select a view from the pull-down menu and click *Generate*. When you generate a view, the Message pane at the bottom of the GUI displays a message, showing the success of the generated view and any output file(s) created.

You can also set additional parameters in the Advanced Options dialog box under the Utilities pull-down menu. For additional information, go to "Setting Advanced Options" on page 2-28.

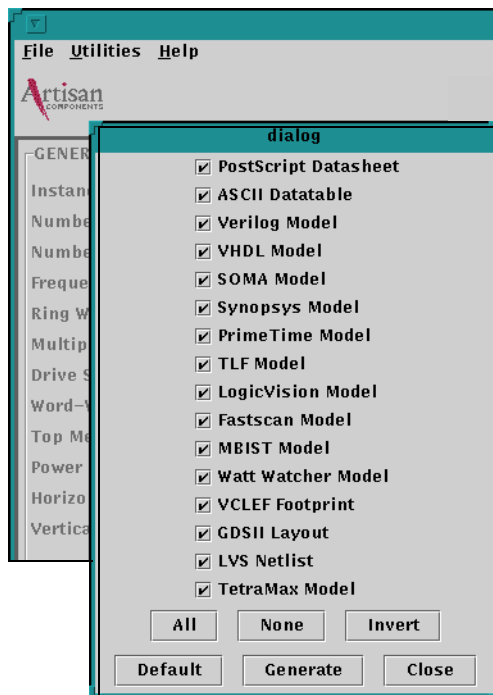
For instance, if you want to create datasheets for multiple instances, you can start by changing the generic parameters shown in the Generic Parameters section or in the Advanced Options pull-down menu of the GUI. Click on Update in the GUI. Select "Postscript Datasheet" from the Views pull-down menu in the Views pane of the GUI. Click on Generate.

An output datasheet called `<instance_name>.ps` is placed in your current `<working_dir>` directory. Now, you can change the parameters and instance name to suit another instance. Click on Generate to create a new datasheet for your new instance.

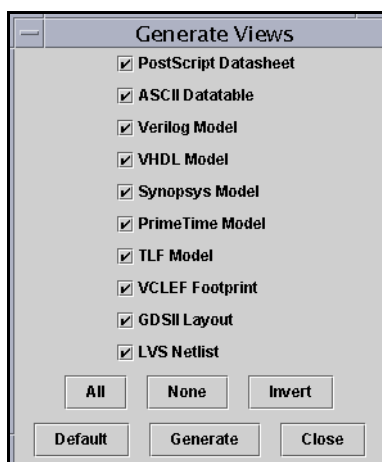
## 2.4.2 Generating Multiple Views

You can also generate all available views, or a selection of views, at one time. Select the *Utilities* pull-down menu from the GUI, as shown in Figures 2-6 and 2-7. Then select *Generate Menu*, to display the Generate View window as shown in Figures 2-8 and 2-9. The *Generate Menu* window shows a list of views that you can generate.

**Figure 2-8. Example: Generate Views Menu (except 130nm ra1shd/ra2shd)**



**Figure 2-9. Example: 130nm (ra1shd/ra2shd) Generate Views Menu**





When this menu is first opened, all views are selected. Click on the box corresponding to a view to toggle between selecting and deselecting the view. When a view is selected, the box contains a check mark. When you click on the *All* button, all views listed are selected. When you click on the *None* button, all views listed are deselected. When you click on the *Invert* button, the selection of views is inverted, or reversed.

When you click on the *Default* button, the parameters for selected views are reset to their default values. When you click on the *Generate* button, all selected views are generated and placed in the current working directory `<working_dir>`. When you generate multiple views, the Message pane at the bottom of the GUI displays a message, that shows the success of generated views and any output files created.

If you close the *Generate-Menu* window and reopen it during the same GUI session, the most recently selected list is recalled.

If you cancel the view generation operation, the current view is cancelled, and the remaining views are not generated.

### 2.4.3 Setting View-Specific Parameters

The *Views* pane of the GUI provides a pull-down menu that displays the views you can generate. When you select certain views, an input field appears. You can enter a view-specific parameter in this field, as described below.

To select a view, click on the corresponding option in the pull-down menu. If the view is not available, a “Not available” message is displayed in the *Views* pane. When you click on a view and click *Generate*, the parameters related to that view appear in the message pane at the bottom of the GUI.

Click *Default* in the *Views* pane to set the view-specific parameters to their default values. As you move from view to view, the parameter values for each view are retained.

---

——— **Note** ———

You can also see "Advanced View-Specific Options" on page 2-34 for more information on options that are specific to particular views.

---

## 2.5 Generating Views from the Command Line

You can generate views directly from the command line. See Chapter 4, "Generator Views," for details about using the views and output files.

### 2.5.1 View Commands

The following table provides the commands you can use to generate standard and optional views from the command line.

**Table 2-3. View Commands**

<i><b>Standard Views</b></i>	
<b>View</b>	<b>View-Command</b>
PostScript datasheet	postscript
ASCII datatable	ascii
GDSII layout file	gds2
LVS netlist	lvs
PrimeTime model	pruntime
Synopsys model	synopsys
TLF model	tlf
VCLEF footprint	vclef-fp
Verilog model	verilog
VHDL model	vhdl
Repair Verilog <sup>1</sup>	frverilog
Repair VHDL <sup>1</sup>	frvhdl
<i><b>Optional Views</b></i>	
<b>View</b>	<b>View-Command</b>
LogicVision IC Memory BIST Model	logicvision
Mentor MBIST Architect model	mbist
Mentor FastScan	fastscan
Synopsys TetraMAX	tmax

<sup>1</sup> The Repair Verilog and Repair VHDL tools are supported by ARM's Artisan 130nm and 180nm synchronous single-port, and 130nm synchronous dual-port SRAM generators with Flex-Repair.

You can run the generator directly from the command line using various commands which instruct the generator to produce the selected view or instance with specific parameters. See "Running the Generator from the Command Line" on page 2-4 for instructions. See "Command Line Syntax" on page 2-23 for details about writing commands to run the generator correctly.

You may use more than one command on your command line.

For example:

```
% ralsh vclef-fp -inst2ring pins -mux 8 ...
```

You may use more than one view command on your command line. The specification file and individual option selections apply to all of the views selected for the run.

For example:

```
% ralsh vclef-fp gds2 synopsys tlf -mux 8 ...
```

## 2.5.2 Generating Multiple Views with View-Specific Options

Certain options are view-specific, they only apply to certain views. For instance, the `inst2ring` and `site_def` options only apply to the VCLEF view, and the `libname` option only applies to the Synopsys and TLF views.

When generating multiple views on your command line, you must specify the view-specific options in detail, as in the following example:

```
% <executable> <view_command_1> <view_command_2>  
  -<view_command_1>.<view-specific_option_1>  
  <view-specific_option_value_1>  
  -<view_command_2>.<view-specific_option_2>  
  <view-specific_option_value_2>
```

For example, to generate both Synopsys and VCLEF-fp views, apply the `-libname` and `-inst2ring` options, and supply a view-specific value for each option, type:

```
% ralsh synopsys vclef-fp -synopsys.libname <syn_userlib> -vclef-fp.inst2ring pins
```

For low-power single-port SRAM memories, replace `ralsh` with `ralsl` in the statement above.

The following table shows these view-specific commands, in sequential order, compared to the entries in the previous examples.

Commands (in sequence)	Example
%<executable>	%ralsh <sup>1</sup>
<view_command_1> <view_command_2>	synopsys vclef-fp
-<view_command_1>.<view-specific_option_1>	-synopsys.libname
<view-specific_option_value_1>	syn_userlib
-<view_command_2>.<view-specific_option_2>	-vclef-fp.inst2ring
<view-specific_option_value_2>	pins

<sup>1</sup> For low-power single-port SRAM memories, replace `ralsh` with `ralsl`.

## 2.6 Generating Specification and Log Files

You can generate files that save the parameters and specifications of each instance. This section tells you how to write a specification file for each instance and create a log file to record commands and output files. In addition, this section describes instance parameter information that displays in the top of most views. This feature allows you to easily identify the parameters generated with each view.

### 2.6.1 Using Specification Files

You can create an ASCII text file for each instance you generate, with all the specific parameters and options you selected for that instance.

When you select the Utilities pull-down menu, then select *Write Spec*, a specification file is generated and placed in the current working directory *<working\_dir>*. The name of the generated specification file is *<instance\_name>.spec*, where *instance\_name* is the name shown in the *Generic Parameters* pane of the GUI at the time the specification file is generated. This file may be edited and used for subsequent runs.

You can create or modify your own specification file using any ASCII text editor. The format for this file is a list of the options you want to apply to your model. You may use either a space or an equal sign “=” between the option name and the value. When using options in a specification file, do not place a dash “-” in front of the option name. Figure 2-10 is an example of a simple specification file that includes a few options.

**Figure 2-10. Example: Specification File**

```
prefix MY_  
instname <instname>  
mux 8  
words 256  
vclef-fp.site_def=off  
vclef-fp.inst2ring=blockages  
top_layer=m8  
write_mask=off
```

Name your specification file, and save it. Your specification file can now be used to configure the generated instance the next time generator is invoked. Launch the GUI with the parameter values from your specification file as the defaults:

```
% <executable> -spec <spec_file>
```

where *<spec\_file>* is the name of your specification file.

The specification file may also be used with the generator commands. See the "Generator Options" on page 2-23.

## 2.6.2 Creating Log Files

A log file containing a record of GUI-generated instances and output messages is placed in the same working directory. A log file is not created when you generate instances from the command line.

When a view is generated or parameters are initialized to default values, a message is recorded in the log file and shown in the message pane of the GUI. The usual name for this file is ACI.log. Although the message pane clears when you select the *Utilities* Menu and then select *Purge Message Area*, the log file retains a full record of messages.

By default, each time the GUI is invoked, the log file created for that run overwrites the existing log file. You may choose to keep the existing log file(s) and create a new log file for the current session. To launch the GUI and keep existing logs, creating a new log file:

```
% <executable> -keeplogs
```

The next log file is an incremental name generated by the system, for example, ACI.log.1.

### 2.6.3 Generating Parameter Information

Parameter information identifies the strings and selections in the generator's fields and options. When you generate an instance from the GUI, a message containing parameter information appears in the message pane at the bottom of the GUI. These values are shown as you would enter them on the command line, as a set of parameters/options and their values.

Parameter information for an instance also outputs at the beginning of all generated views, except the postscript datasheet and ASCII datatable views. An example is shown in Figure 2-11. You must change the instance name in the GUI to retain information unique to each instance. If the instance name does not change, the previous information will be overwritten when you regenerate.

**Figure 2-11. Example: Parameter Information**

```
* name:          xxx Generator
*                Artisan x.xxum Process
* version:       2003Q2V0
* comment:       030522_6:38_03
* configuration: -instname INSTANCExxx -words 4096 -bits 16
                -frequency 1 -ring_width 2 -mux 16 -drive 6 -write_mask off
                -wp_size 8 -top_layer met8 -power_type rings -horiz met3 -
                vert met3 -cust_comment "030522_6:38_03" -left_bus_delim
                "[" -right_bus_delim "]" -pwr_gnd_rename "VDD:VDD,GND:VSS"
                -prefix "" -pin_space 0.0 -name_case upper -check_instname
                on -diodes on -inside_ring_type VDD
```

Included in this parameter information is the customer comment option, which allows you to add a unique identifier such as the date and time you generate a particular instance. You can use this option to add more detail than in the instance name. The example above shows a parameter information string, with “030522\_6:38\_03” as the customer comment. For more information on the customer comment string, see “Setting Advanced Options” on page 2-28.

## 2.7 Generator Options

The standard options described in this section allow you to customize your generator to create specific memory instances. These options can be accessed from the command line and from the generator GUI.

The option is listed first, followed by the type of parameter, and then a description of the option. For instance the parameter for the "mux" option is a number. You can see the parameters tables in Chapter 3, "Synchronous SRAM Generator Architecture," for standard parameter ranges.

In some generators, some options or parameters may not be available when other options are selected. The options will still be visible, but are shown in grey when disabled.

### 2.7.1 Command Line Syntax

Use the following syntax for all options, including as many options as you want to set. See "Directory Structure and Executables" on page 2-8, for information about the executable for your generator.

```
<executable> [<view_command>] [-spec <filename>] [-mux <number>]  
[-write_mask on|off]...
```

You may supply any combination of options and specification files on the command line. On the command line, use the dash '-' in front of the option. In the case of duplicate options or parameters, the last option set takes precedence.



## 2.7.2 Basic Options

### `-spec filename`

The specification file contains a list of basic, advanced, and view-specific options and values for the generator. You may create a new specification file, or files, to customize the options that you want to use repeatedly. If you are creating new specification files, be sure to read the "Using Specification Files" on page 2-20.

### `-help`

You can access the help information for a generator or generator view. Information such as available views and options is displayed as a text message on the command line. You can access help information that applies to specific views by including the view command for that view. See "View Commands" on page 2-17 for a list of view commands.

To access the help for a generator, be sure you have added `<install_dir>/aci/<executable>/bin` to your UNIX search path. If you do not wish to do this, preface all generator commands with `<install_dir>/aci/<executable>/bin/` and type:

```
<executable> -help
```

To access the help for a generator view, type:

```
<executable> <view_command> -help
```

For example, type `"ra1sh ascii -help"` to view the help information related to the ASCII datatable, such as generator parameters. For low-power single-port SRAM memories, replace `ra1sh` with `ra1sl` in the statement.

**-instname** *name*

The default instance name is the same as the executable name, in capital letters. For instance, if the executable is `ra1sh`, the default instance name is `RA1SH`. See “Directory Structure and Executables” on page 2-8, regarding the executable name for your generator. For low-power single-port SRAM memories, replace `ra1sh` with `ra1sl` in the statement.

You can set the instance name to any alphanumeric value. To avoid name conflicts for instances within the same library, you must enter a unique instance name. To avoid tool compatibility issues, an instance name of 16 characters or fewer is recommended, and it should begin with a letter. See the additional information in the `-check_instname` and `-prefix` option descriptions.

**-words** *number*

The generator GUI shows the default words value. The range for words depends on the multiplexer width, limited by the physical array of memory cells. See Chapter 3, "Synchronous SRAM Generator Architecture," for the word ranges for standard generators.

**-bits** *number*

The generator GUI shows the default bits value. The range for bits depends on the multiplexer width, limited by the physical array of memory cells. See Chapter 3, "Synchronous SRAM Generator Architecture," for the bit ranges for standard generators.

**-frequency** *number*

The generator GUI shows the default frequency value, in megahertz (MHz). The frequency can be set to any positive integer value, up to the inverse of the cycle time in nanoseconds (ns) multiplied by 1000. The frequency parameter is used to scale the AC current-consumption datatable values. If it is left as the default value of 1.0 megahertz, the units on the AC current-consumption datasheet values will be milliamperes per megahertz.

`-ring_width number`

The generator GUI shows the default ring width value, in  $\mu\text{m}$ , and is intended as a place holder only. In order for the generated memory to function correctly, you must analyze the ring width requirements based on the design methodology and supply an appropriate ring width value. For details, see "Supply Connections to Power Rings" on page 3-44.

`-mux number`

The generator GUI shows the default mux value and any choices for this option. See Chapter 3, "Synchronous SRAM Generator Architecture," for the mux values for specific generators.

`-drive number`

The generator GUI shows the output drive strength.

`-write_mask on|off`

The generator GUI shows the default value for the Word-Write Mask option and any choices for this option. When it is selected, the associated option, Word Partition Size, is displayed in the GUI.

`-wp_size number`

The default value is 8. The range for word partition size is 1 to min (36, bits-1) increment = 1. When you enable the redundancy option, the only legal value of word partition size is 1.

`-redundancy on/off`

The default value is off. The choice for redundancy is on or off. When the value on is selected, the companion options are displayed in the GUI:



---

**Note**

The redundancy feature (Flex-Repair tool) is offered with most Artisan synchronous 130nm and 180nm single-port SRAM as well as most synchronous 130nm dual-port SRAM generators. For detailed information about the redundancy option, see the “Synchronous SRAM Generator with 130nm/180nm Flex-Repair Solution Application Note.”

---

**-redundancy\_bits** *1*

The default/only value is 1. The choice for redundant bits is 1. Redundant bits are added to the core memory bits "-bits <number>" so that the memory is generated with the total number (redundant bits + memory bits) of bits.

———— **Note** ————

The redundancy feature (Flex-Repair tool) is offered with most Artisan synchronous 130nm and 180nm single-port SRAM as well as most synchronous 130nm dual-port SRAM generators.

**-top\_layer** *number*

The generator GUI shows the default value for the top metal layer and any choices for this option. For more details, see "Top Metal Layer" on page 3-46.

**-power\_type** *rings*

Power is supplied through a ring structure in standard generators.

**-horiz** *string*

The generator GUI shows the default value of the Horizontal Ring Layer option and any choices for this option.

The horizontal and vertical ring layers may be set to any two sequential metal layers or the same layer (for example, m1 and m2, or m1 and m1). When a horizontal value is selected, the vertical ring layer is automatically set to a valid value.

**-vert** *string*

The generator GUI shows the default value of the Vertical Ring Layer option and any choices for this option.

The horizontal and vertical ring layers may be set to any two sequential metal layers or the same layer (for example, m1 and m2, or m1 and m1). When a vertical value is selected, the horizontal ring layer is automatically set to a valid value.

## 2.7.3 Setting Advanced Options

You can set advanced options from the generator GUI or by entering the options on the command line. This section demonstrates the methods for setting these options, including some options that apply only to specific views. Some of these options depend on the setting of the generic parameters in the main GUI.

### 2.7.3.1 Setting Advanced Options from the GUI

From the GUI, select the *Utilities* pull-down menu, then select the *Advanced Options* menu as shown in Figures 2-12 and 2-13. Enter a string or number in the fields, or select the check boxes as needed.

———— **Note** ————

The number of advanced options may vary and depends on your specific generator. The redundancy feature (Flex-Repair tool) is offered with some Artisan synchronous single-port and synchronous dual-port SRAM generators. For some generators, you may see Dual Port Clock Collision Modeling (DPCCM) and Read Address Setup Violation Modeling (ASVM) options or just a single option for ASVM.

**Figure 2-12. Example: Advanced Options Menu-Redundancy On**

Advanced Options	
Customer Comment	
Left Bus Delimeter	[
Right Bus Delimeter	]
Power Ground Rename	VDD:VDD,GND:VSS
Instname Prefix	
Pin Space (um)	0.0
Name Case	<input checked="" type="checkbox"/> upper <input type="checkbox"/> lower
Check Instance Name	<input checked="" type="checkbox"/> on <input type="checkbox"/> off
Diodes	<input checked="" type="checkbox"/> on <input type="checkbox"/> off
Inside Ring Type	<input type="checkbox"/> VDD <input checked="" type="checkbox"/> GND
Memory to Ring Wires	<input type="checkbox"/> pins <input checked="" type="checkbox"/> blockages
Site Definitions	<input type="checkbox"/> on <input checked="" type="checkbox"/> off
— Redundancy Options —	
Fuse-encoding	<input type="checkbox"/> 1 hot <input checked="" type="checkbox"/> encoded
Include FuseBox	<input checked="" type="checkbox"/> yes <input type="checkbox"/> no
FuseBox Instance Name	FUSE
Control Block Style	<input checked="" type="checkbox"/> mux <input type="checkbox"/> 3state
<input type="button" value="Default"/> <input type="button" value="Close"/>	

Figure 2-13. Example: Advanced Options Menu - Redundancy Off; with DPCCM and ASVM<sup>1</sup>

Advanced Options

Customer Comment

Bus-notation ☒ on ☐ off

Left Bus Delimeter [

Right Bus Delimeter ]

Power Ground Rename VDD:VDD,GND:VSS

Instname Prefix

Pin Space (um) 0.0

Name Case ☒ upper ☐ lower

Check Instance Name ☒ on ☐ off

Diodes ☒ on ☐ off

Inside Ring Type ☐ VDD ☒ GND

Drive Strength ☒ 6

DP Clock Collision Modeling ☒ on ☐ off

Read Address Setup Violation Modeling ☐ on ☒ off

Memory to Ring Wires ☐ pins ☒ blockages

Site Definitions ☐ on ☒ off

Default Close

<sup>1</sup> DPCCM does not apply to single-port SRAM generators.

### 2.7.3.2 Setting Advanced Options from the Command Line

From the command line, type the executable name, then the option preceded by a dash "-" and then the parameter you want to specify. You can type as many options and parameters as you need. See the "Command Line Syntax" on page 2-23 to be sure you are entering commands correctly.

### 2.7.4 Advanced Options

`-customer_comment string`

A customer-specified text field of up to 64 characters. The allowed character set is alphanumeric, plus the following characters: '\_', '-', '.', ':', '=', and '+.

The customer comment string is included in the parameter information that appears at the top of all views, except the postscript datasheet and ASCII datatable. You can use this string to differentiate between different instances with more characters than the instance name allows. For more information about the customer comment string, see "Generating Parameter Information" on page 2-22.

`-bus_notation on` (130nm ralshd/ra2shd only)

By default, all bus pins are represented by bus notation in the interface of models (for example, A[3:0]).

`-left_bus_delim /|<|/`

The Advanced Options menu shows the default value for the Left Bus Delimiter option. This option specifies the left bus delimiter for physical views, including VCLEF, GDSII, and LVS. Delimiter choices are "[," "<," or "{." Bus delimiters for front-end views are tool specific, and are not affected by using this option.

`-right_bus_delim /|>|/`

The Advanced Options menu shows the default value for the Right Bus Delimiter option. This option specifies the right bus delimiter for physical views, including VCLEF, GDSII, and LVS. Delimiter choices are "],," ">," or }." Bus delimiters for front-end views are tool specific, and are not affected by using this option.

`-pwr_gnd_rename` *string*

This option allows you to name the power and ground net names for backend views. An example string, VDD:VCC,VSS:GND, will rename power "VCC" and ground "GND."

`-name_case` *upper|lower*

The default is upper. This option specifies the case for subcircuit and net names, excluding the top-level instance name and power/ground names.

`-prefix` *name*

This option allows you to assign a prefix for the instance name. If this option is left blank, no prefix is applied to the instance name. The prefix is counted when using `-check_instname`.

`-inside_ring_type` *VDD|GND*

The Advanced Options dialog box shows the default value for the Inside Ring Type option and any choices for this option. The inside ring power type can be either VDD or GND (VSS). The outside ring power type will be of the opposite polarity. See the "Supply Connections to Power Rings" section on page 3-44 for more information about selecting the ring type.

`-pin_space` *number*

The Advanced Options menu shows the default value for the Pin Space option. This option specifies the space, in microns, between the pins of the memory core and the inner power ring segment.

`-check_instname` *on|off*

The Advanced Options menu shows the default value for the Check Instance Name option and any choices for this option. An instance name should begin with a letter, and should not exceed 16 characters. If this option is turned on and the name does not meet these requirements, the GUI issues error messages, and does not generate views. If the option is turned off, the GUI issues warning messages, but it will generate views. Both errors and warnings display in the message pane, and are recorded in the log file.

If the generator was launched from the command line and the instance name is greater than 16 characters, the error and warning messages appear on the terminal.



The following advanced options are implemented by the memory generator RTL. Use these options to customize the design for your single-port 130nm or 180nm SRAM and dual-port 130nm SRAM, with Flex-Repair, instances.

`-fuse_encoding` *1hot|encoded*

The default value is encoded. The choices for fuse encoding are 1hot or encoded. The encoded option provides binary encoding at the faulty bit location. The 1hot option provides one fuse per bit at the faulty location. See the “Synchronous SRAM Generator with 130nm/180nm Flex-Repair Solution Application Note” for further information about this option.

`-insert_fuse` *yes|no*

The default value is yes. The choices for insert fuse are yes and no.

If yes is selected the wrapper module in the Register Transfer Logic (RTL) includes the fuse box inside. This option should be used if you are creating a fuse box instance dedicated to this memory instance. A direct mapping of fuse box instance to memory instance results.

If no is selected you must instantiate the fuse box outside and connect the outputs to the wrapper. The no option is useful if you want to create one fuse box to be used for multiple memories.

`-inst_name` *name*

The fuse box instance name can be set to any alphanumeric value. To avoid tool compatibility issues an instance name of 16 characters or fewer is recommended, and should begin with a letter.

The instance name of your fuse box must be created with the same name by running the fuse box generator. See the `check_instname` and `prefix` options described in this section for further information.

`-rtl_style` *mux|3state*

The default value is 3-state. The choices for rtl\_style are mux (multiplexer) and 3-state. This option allows you to select the logic architecture in the repair RTL.

`-clustsize` *number*

This option is not active.

*-dpccm on/off*

This option enables you to select front-end (FE) model behavior, for dual-port generators, when clock collisions occur during Read and Write operations. See your generator GUI for the default setting for your generator.

During clock collision, if `dpccm` is `off`, Read operation fails and the output lines show X. Also, Write operation fails and the memory location and write-through (if applicable) show X.

If `dpccm` is `on` during clock collision, Read operation fails and the output lines show X, but in this case Write operation to the memory location succeeds.

*-asvm on/off*

This option enables you to select front-end (FE) model behavior when address setup violations occur during a Read operation. This option allows you to choose memory FE model behavior when address setup violations occur. See your generator GUI for the default setting for your generator.

During an address setup violation, if `asvm` is `off` during Read operation, Read fails, and the output lines show X and are invalidated (X-ed-out) at all memory locations.

If `asvm` is `on` during Read operation, Read fails and the output lines show X, but in this case all memory locations preserve their state.

### 2.7.4.1 Advanced View-Specific Options

This section lists advanced options that apply only to certain views.

`-libname userlib`

If you are using Synopsys or TLF models, the default library name is *userlib*. Use this option to specify your choice for `-libname`. This option applies only to Synopsys and TLF models.

`-diodes on|off`

The Advanced Options menu shows the default value for the Diodes option and any choices for this option. Because the default value indicates how the generator was validated, the LVS rule set may not support the non-default value.

If the Diodes option is off, antenna diodes present in the GDSII view are omitted in the LVS Netlist view.

`-inst2ring pins|blockages`

The Memory to Ring Wires option shows the default value, blockages. Choose whether the power connections from the memory to the ring wires are modeled as pins or blockages in VCLEF. This option applies only to the VCLEF model.

`-site_def on|off`

The default value is off. This option specifies whether VCLEF contains a site definition. This option applies to only the VCLEF model.

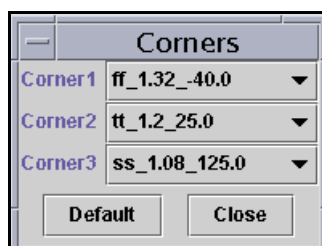
## 2.7.5 Selecting Characterization Corners (130nm ra1shd/ra2shd/ra2shd\_rd)

You can set characterization corners from the generator GUI or by entering the option on the command line. ARM recommends that critical path, setup and hold analyses be performed for all applicable corners.

### 2.7.5.1 Selecting Corners from the GUI

From the Utilities pull-down menu in the generator GUI, select the “Corners Option.” The Corners dialog box contains several corner options; Corner1, Corner2, and Corner3. The number of corner options may vary per generator. Figure 2-14 shows a Corners dialog box with three fields.

**Figure 2-14. Example: Corners Dialog Box**



Each field contains predetermined PVT selections. The first item in each selection is the process indicator, such as fast (ff), typical (tt), and slow (ss). There may be more than one fast corner, each with differing combinations of voltages and temperatures. After the process indicator, the first value is the voltage for that corner. The second value is the temperature for that corner. In the example above, for the first selection in Corner 1, the process is fast (ff), the first value is 1.32V, and the second value is -40°C.

The Corners menu and the ASCII datatable in the GUI show the default corners.

Select the PVT information that you would like to generate for each corner option. All timing models, PostScript datasheet, and ASCII datatable will have delays set at the chosen PVT corners. For more information about PVT corners, see "Characterization Environments" on page 3-47.

### **2.7.5.2 Selecting Corners from the Command Line**

`-corners "string, string, string, {string}"`

You can specify up to four PVT corners for characterization at one time. For more information about PVT corners, see "Characterization Environments" on page 3-47. All timing models, PostScript datasheet, and ASCII datatable have delays set at the chosen corners. Timing model filenames have a corner name embedded in them.



# 3

## Synchronous SRAM Generator Architecture

This chapter contains the following sections:

- “Overview” on page 3-3
- “Synchronous Single-Port SRAM Architecture and Timing Specifications” on page 3-4
- “Synchronous Dual-Port SRAM Architecture and Timing Specifications” on page 3-25
- “SRAM Power Structure (ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2shd\_rd, ra2sd)” on page 3-41
- “SRAM Physical Characteristics (ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2shd\_rd, ra2sd)” on page 3-46
- “SRAM Timing Derating (ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2sd)” on page 3-48

## 3.1 Overview

This chapter describes the architecture, features, timing characterization, and physical characteristics for synchronous single- and dual-port, high speed/density and high density SRAM generators. This chapter also applies to synchronous single-port, low-power SRAM generators.

————— **Note** —————

Information about deviations to the ASL generators can be found in the README text file enclosed with your generator or in an addendum attached to this manual.

Where applicable, separate sections are provided for single- and dual-port SRAM generators. The following table lists the generator names, product names, and executable names for the generators described in this section. Check your generator GUI; the names provided in your generator GUI always supercede those in the table below.

**Table 3-1. SRAM Generator Naming Conventions**

<b>Generator</b>	<b>Product Name</b>	<b>Executable</b>
High-Speed/Density Single-Port SRAM	SRAM-SP	ra1sh or ra1shd
High-Speed/Density Dual-Port SRAM	SRAM-DP	ra2sh or ra2shd
High-Speed/Density Dual-Port SRAM with redundancy	SRAM-DP-RD	ra2shd_rd
High-Density Single-Port SRAM	SRAM-SP-HD	ra1sd
High-Density Dual-Port SRAM	SRAM-DP-HD	ra2sd
Low-Power Single-Port SRAM	SRAM-SP-LP	ra1sl



## 3.2 Synchronous Single-Port SRAM Architecture and Timing Specifications

This section describes the synchronous single-port SRAM generator architecture, which includes pin descriptions, logic tables, block diagrams, core address maps, timing diagrams, and timing and power parameters. Executable names for applicable generators are provided for your reference.

### 3.2.1 Single-Port SRAM Description (ra1sh, ra1shd, ra1sd, ra1sl)

The synchronous single-port SRAM is produced by a parameterized block generator which allows great flexibility in the SRAM organization.

SRAM access is synchronous and is triggered by the rising edge of the clock, CLK. Address, input data, write enable, and chip enable are latched by the rising edge of the clock, subject to individual setup and hold times.

The value of chip enable must be low (CEN=0) for a read or write operation to occur. The SRAM enters read mode when the value of chip enable is low and the value of write enable is high. During read mode, data is read from the memory location specified on the address bus A[j:0] and appears on the data output bus Q[i:0].

If the word-write mask feature is *not* implemented (word-write mask = off), the SRAM enters write mode when the values of chip enable and write enable are low (CEN=0, WEN=0). During write mode, data on the data input bus D[i:0] is written into the memory location specified on the address bus A[j:0].

If the word-write mask feature is implemented (word-write mask = on), data on the data input bus D[i:0] is partitioned to the write enable bus WEN[k:0]. Each WEN[k] pin has a distinct latched value, making each partition individually selectable for reading or writing. When the value of a write enable pin WEN[k] is low, the corresponding data partition is selected, and its data is written to the memory location specified on the address bus A[j:0], and driven through to the data output bus Q[i:0].

For example, an SRAM with 32 bits and a word partition size of 8 (wp\_size = 8) will have four write enable pins: WEN[0], WEN[1], WEN[2], and WEN[3]. The write enable pin WEN[0] corresponds to the data partition D[7:0]; the write enable pin WEN[1] corresponds to the data partition D[15:8]; the write enable pin WEN[2] corresponds to the data partition D[23:16]; the write enable pin WEN[3] corresponds to the data partition D[31:24]. If the values of WEN[0] and

WEN[3] are low, and the values of WEN[1] and WEN[2] are high, the data bits D[7:0] and D[31:24] will be written to the memory and driven through to the data output bus Q[7:0], Q[31:24]. Even if data is presented on D[15:8] and D[23:16], this data is *not* written to the memory and is *not* driven through to the data output bus. Instead, Q[15:8] and Q[23:16] will be the result of a read operation.

When an address is accessed beyond the address space for reading or writing, operations are not known.

Some 180nm, and older, generators have an output enable (OEN) pin. If the value of the output enable signal is high (OEN=1), data on the output bus Q[i:0] is placed in a high impedance state. However, while it is in the high impedance state, every read or write operation continues to update the internal output data latch. When the value of the output enable signal is low (OEN=0), the data appears on the output bus Q[i:0].

When the memory redundancy feature is selected, in single-port SRAM generators with Flex-Repair capability, the generator adds redundant bits to the user-defined bits before creating the desired views. For example, if you define a 32 bit/word memory with one bit of redundancy, the generator creates a  $32 + 1 = 33$  bit/word memory.

Word write-mask is supported with memory redundancy, however the only legal value for word partition size is 1.

Power dissipation is minimized using static circuit implementations. A standby mode is provided to further reduce power dissipation during periods of non-operation (CEN=1). While in standby mode, address and data inputs are disabled; data stored in the memory is retained, but the memory cannot be accessed for reads or writes. You can eliminate switching current in the input stages and reduce deselected current to near leakage by holding all input pins steady during standby mode.

Static power consumption is limited to leakage provided that all input signals except CLK are held steady.

### 3.2.2 Single-Port SRAM Pins (ra1sh, ra1shd, ra1sd, ra1sl)

Figure 3-1 shows basic pins for the single-port SRAM generator.

**Figure 3-1. Single-Port SRAM Basic Pins**

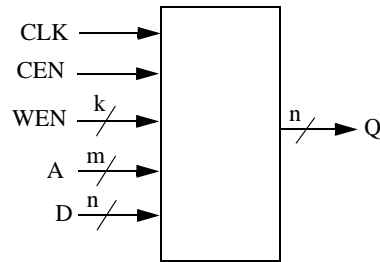


Table 3-2 provides the single-port SRAM generator pin descriptions.

**Table 3-2. Pin Descriptions for Single-Port SRAM Generators**

Name	Type	Description
<b>Basic Pins</b>		
A[m-1:0]	Input	Addresses (A[0] = LSB)
D[n-1:0]	Input	Data inputs (D[0] = LSB)
CEN	Input	Chip Enable, active low
WEN [*]	Input	Write Enable, active low. *If word-write mask is enabled, this becomes a bus.
CLK	Input	Clock
Q[n-1:0]	Output	Data outputs (Q[0] = LSB)
OEN	Output	Tristate Output Enable <sup>1</sup>

<sup>1</sup> The OEN pin applies to some 180nm generators.

### 3.2.3 Single-Port SRAM Logic Tables (ra1sh, ra1shd, ra1sd, ra1sl)

This section provides logic tables for basic single-port SRAM functions and for the individual test and repair functions.

In following table, wen could be WEN when word-mask is off and WEN[ ] when it is on.

Logic functions for the basic single-port SRAM generator features are shown in Table 3-3.

**Table 3-3. Single-Port SRAM Basic Functions**

CEN	wen	Data Output	Mode	Function
H	X	Last Data	Standby	Address inputs are disabled; data stored in the memory is retained, but memory cannot be accessed for new reads or writes. Data outputs remain stable.
L	L	Data In	Write	<b>Word-write:</b> Data on the data input bus, D[n-1:0], is written to the memory location specified by the address bus, A[m-1:0]; and is driven through to the data output bus Q[n-1:0] <b>Bit-write:</b> The corresponding data partition is selected by the write enables, WEN[k-1:0]; and that data is written to the memory location specified by the address bus, A[m-1:0], and is driven through to the data output bus Q[n-1:0]. Data on the data input bus, D[n-1:0], is written to the memory location specified by the address bus, A[m-1:0]; and is driven through to the data output bus Q[n-1:0]
L	H	SRAM data	Read	Data on the data output bus Q[n-1:0] is read from the memory location specified on the address bus A[m-1:0].

### 3.2.4 Single-Port SRAM Parameters (ra1sh, ra1shd, ra1sd, ra1sl)

The standard input and block parameters of a synchronous single-port SRAM are listed in Tables 3-4 and 3-5. The values differ between 150nm/130nm and 250nm/180nm generators. See your generator GUI for specific input ranges. If you enter an invalid value and update the GUI, the message pane in the GUI displays an error message and the specific range for your generator.

**Table 3-4. Single-Port 150nm/130nm SRAM Parameters**

Input Parameters		
Parameter	Ranges	
number of words	mux = 8	256 to 4096, increment = $\text{mux} \bullet 2$
	mux = 16	512 to 8192, increment = $\text{mux} \bullet 2$
	mux = 32	1024 to 16384, increment = $\text{mux} \bullet 2$
number of bits	mux = 8	2 to 128, increment = 1
	mux = 16	2 to 64, increment = 1
	mux = 32	2 to 32, increment = 1
frequency (MHz)	1 to $1/t_{\text{cyc}} \bullet 1000$ , increment = 1	
word partition size <sup>1,2</sup>	1 to min (36, bits-1) increment = 1	
top chip metal layer support	m4 to top metal layer supported by design process	
top generator metal layer	m4	
horizontal ring layer	m1, m2, m3, or m4	
vertical ring layer	m1, m2, m3, or m4	
ring width (μm)	2 to 10,000	
redundancy (130nm only)	on/off, default is off; when ‘on’ is selected, the default/only redundancy bit value is 1	
Block Parameters		
Parameter	Ranges	
total memory bits	512 to 524,288, total bits = words • bits	
rows in memory matrix	32 to 512, increment = 2, rows = words / mux	
columns in memory matrix	16 to 1024, increment = mux, columns = bits • mux	
address lines	mux = 8	8 to 12
	mux = 16	9 to 13
	mux = 32	10 to 14
output drive strength	See“Running the Generator from the Graphical User Interface (GUI)” on page 2-3.	

<sup>1</sup> The input pin capacitance for each pin of the write enable bus is proportional to the size of the word partition. For example, an instance with bits=32 and wp\_size=24 will have two partitions, one with 24 bits and one with 8 bits. The write enable pin for the 24 bit partition will have a significantly larger input pin capacitance than the write enable pin for the 8 bit partition. When modeling write enable timing, the write enable pin with the largest capacitance is used in the typical and slow corner timing models. The write enable pin with the smallest capacitance is used in the fast corner timing models. ARM recommends that the critical path, setup and hold analysis be performed for all corners.

<sup>2</sup> When you enable the redundancy option, the only legal value of word partition size is 1.

**Table 3-5. Single-Port 250nm/180nm SRAM Parameters**

Input Parameters		
Parameter	Ranges	
number of words	mux = 4	16 to 2048, increment = mux • 2
	mux = 8	32 to 4096, increment = mux • 2
	mux = 16	64 to 8192, increment = mux • 2
number of bits	mux = 4	2 to 128, increment = 1
	mux = 8	2 to 128, increment = 1
	mux = 16	2 to 64, increment = 1
frequency (MHz)	1 to $1/t_{cyc} \bullet 1000$ , increment = 1	
word partition size <sup>1, 2</sup>	1 to min (36, bits-1) increment = 1	
top chip metal layer support	m4 to top metal layer supported by design process	
top generator metal layer	m4	
horizontal ring layer	m1, m2, m3, or m4	
vertical ring layer	m1, m2, m3, or m4	
ring width (µm)	2 to 10,000	
redundancy (some 180nm only)	on/off, default is off; when ‘on’ is selected, the default/only redundancy bit value is 1	
Block Parameters		
Parameter	Ranges	
total memory bits	32 to 524,288, total bits = words • bits	
rows in memory matrix	4 to 512, increment = 2, rows = words / mux	
columns in memory matrix	8 to 1024, increment = mux, columns = bits • mux	
address lines	mux = 4	4 to 11
	mux = 8	5 to 12
	mux = 16	6 to 13
output drive strength	See“Running the Generator from the Graphical User Interface (GUI)” on page 2-3	

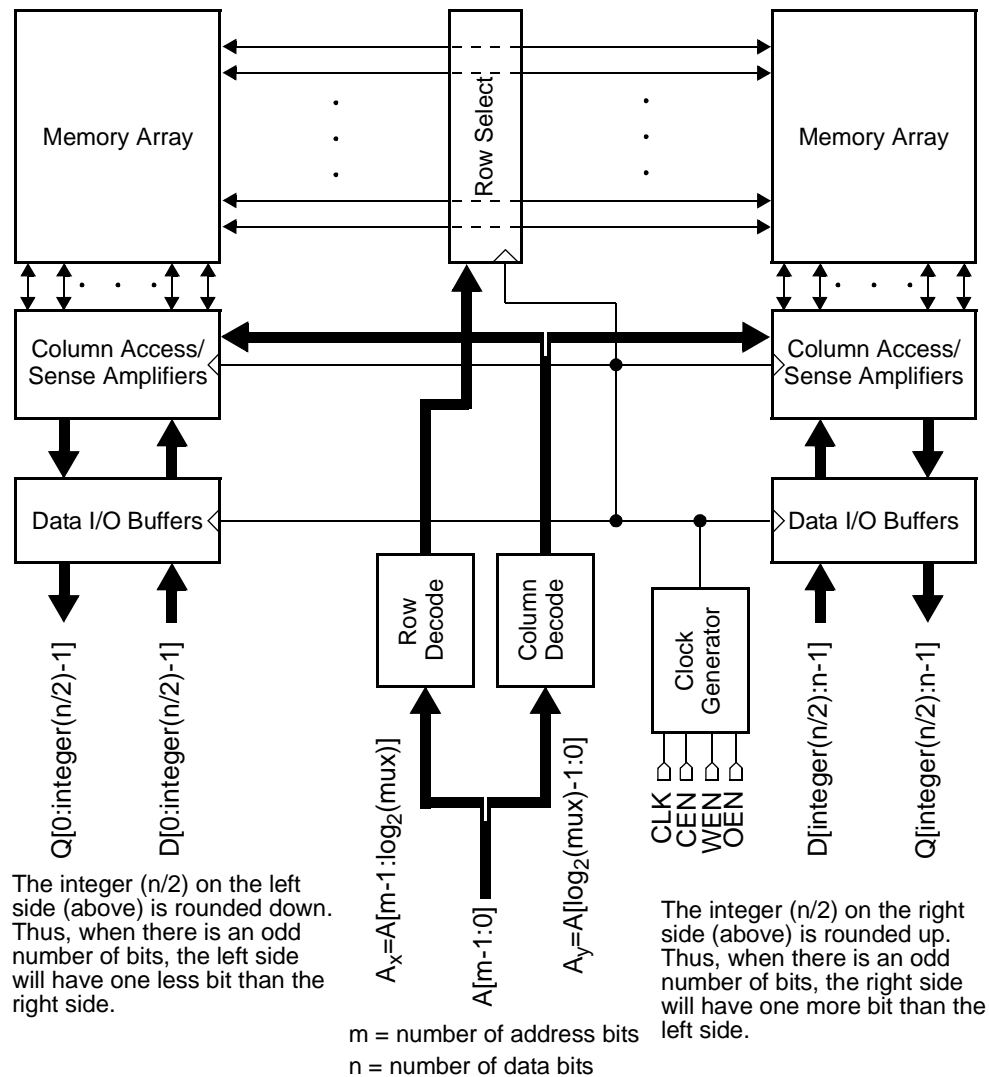
<sup>1</sup> The input pin capacitance for each pin of the write enable bus is proportional to the size of the word partition. For example, an instance with bits=32 and wp\_size=24 will have two partitions, one with 24 bits and one with 8 bits. The write enable pin for the 24 bit partition will have a significantly larger input pin capacitance than the write enable pin for the 8 bit partition. When modeling write enable timing, the write enable pin with the largest capacitance is used in the typical and slow corner timing models. The write enable pin with the smallest capacitance is used in the fast corner timing models. ARM recommends that the critical path, setup, and hold analysis be performed for all corners.

<sup>2</sup> When you enable the redundancy option, the only legal value of word partition size is 1.

### 3.2.5 Single-Port High Speed/Density SRAM Block Diagrams (ra1sh, ra1shd, ra1sd)

The synchronous high-speed/density single-port SRAM instance block diagram is shown in Figure 3-2.

**Figure 3-2. Single-Port High-Speed/Density and High-Density SRAM Block Diagram**



#### Notes:

When the word-write mask (wwm) option is turned on, WEN is a bus. When wwm is turned off, WEN is a signal pin. Output enable (OEN) pins apply to some 180nm and older generators.





### 3.2.7 Single-Port High-Speed/Density SRAM Core Address Maps (ra1sh, ra1shd)

An example of the standard physical core mapping for all possible high-speed/density mux values is shown in Figures 3-4 to 3-7. Your generator may not include each mux.

**Figure 3-4. Single-Port High-Speed/Density SRAM Mux 4: Core Address Mapping**

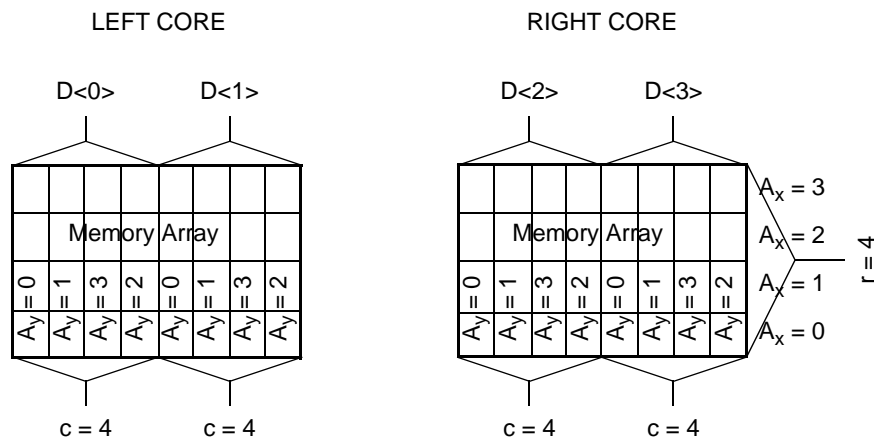


Figure 3-5. Single-Port High-Speed/Density SRAM Mux 8: Core Address Mapping

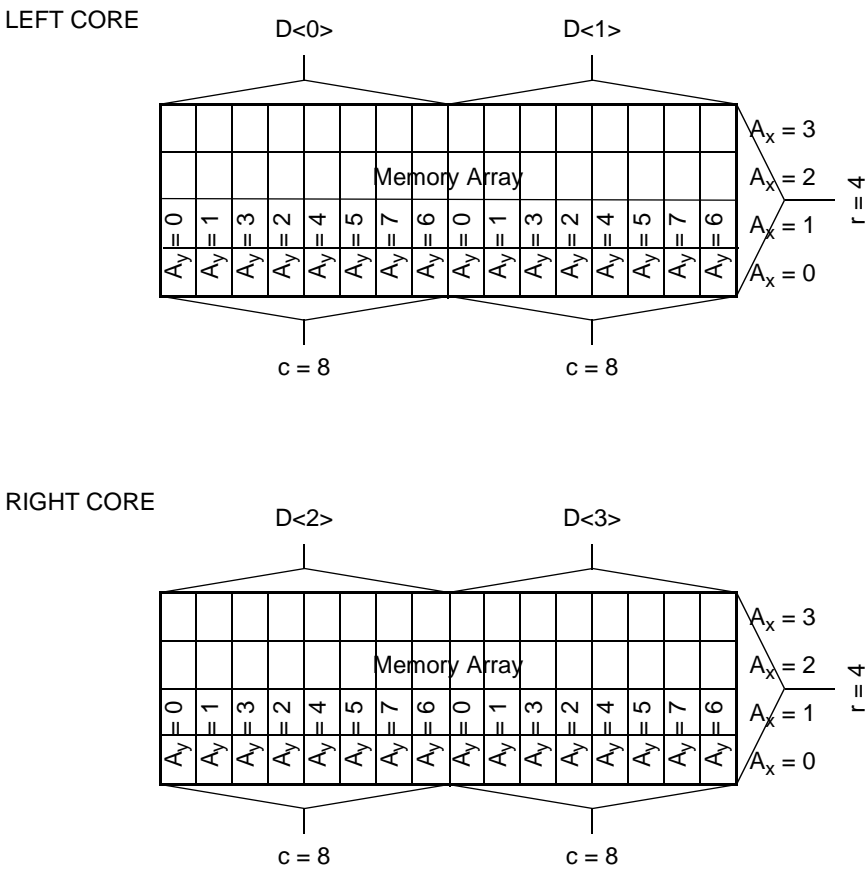


Figure 3-6. Single-Port High-Speed/Density SRAM Mux 16: Core Address Mapping

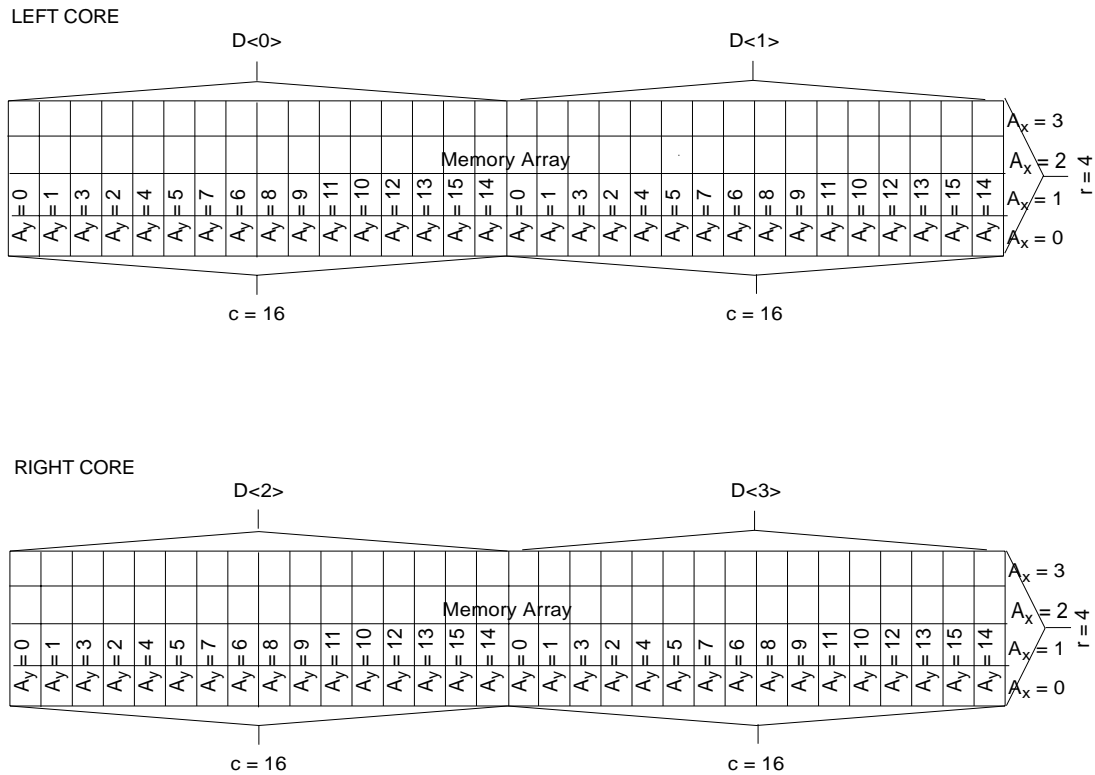
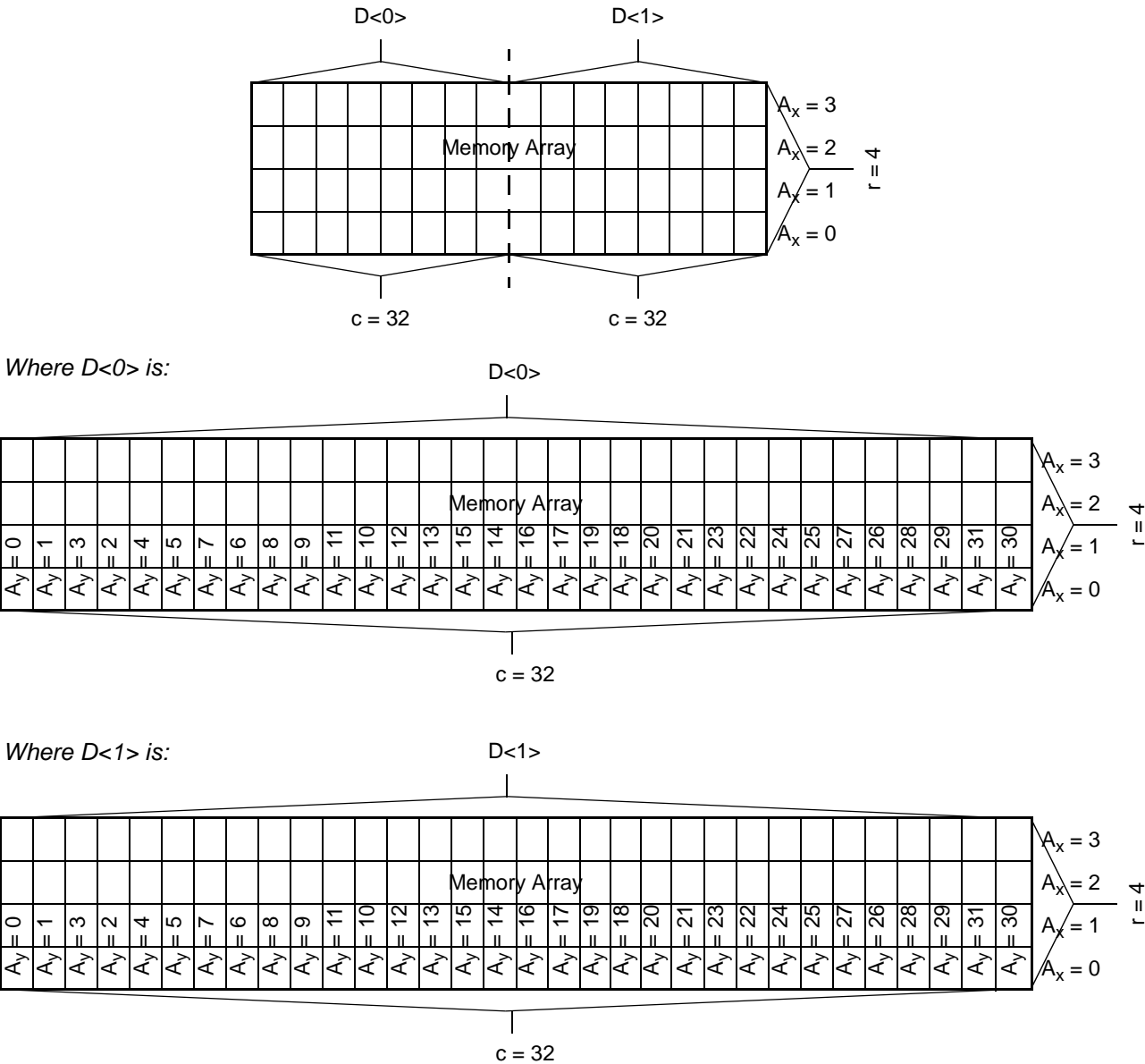


Figure 3-7. Single-Port High-Speed/Density SRAM Mux 32: Core Address Mapping



### 3.2.8 Single-Port High-Density SRAM Core Address Maps (ra1sd)

An example of the physical core mapping for all possible high-density mux values is shown in Figures 3-8 to 3-10.

**Figure 3-8. Single-Port High-Density SRAM Mux 4: Core Address Mapping**

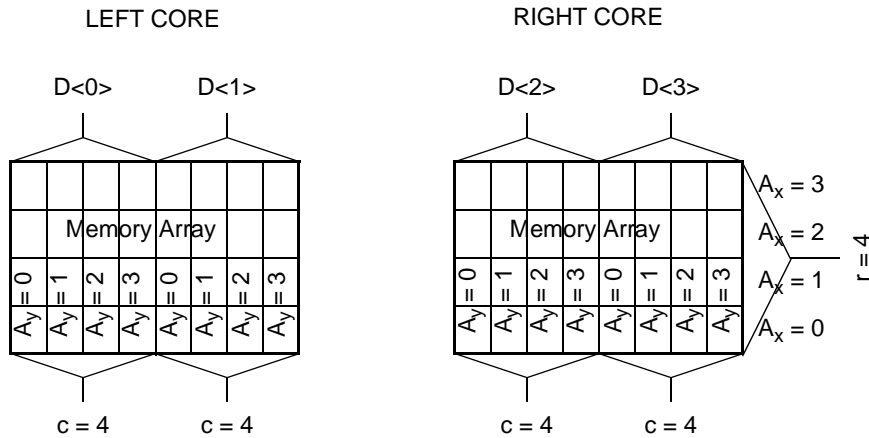


Figure 3-9. Single-Port High-Density SRAM Mux 8: Core Address Mapping

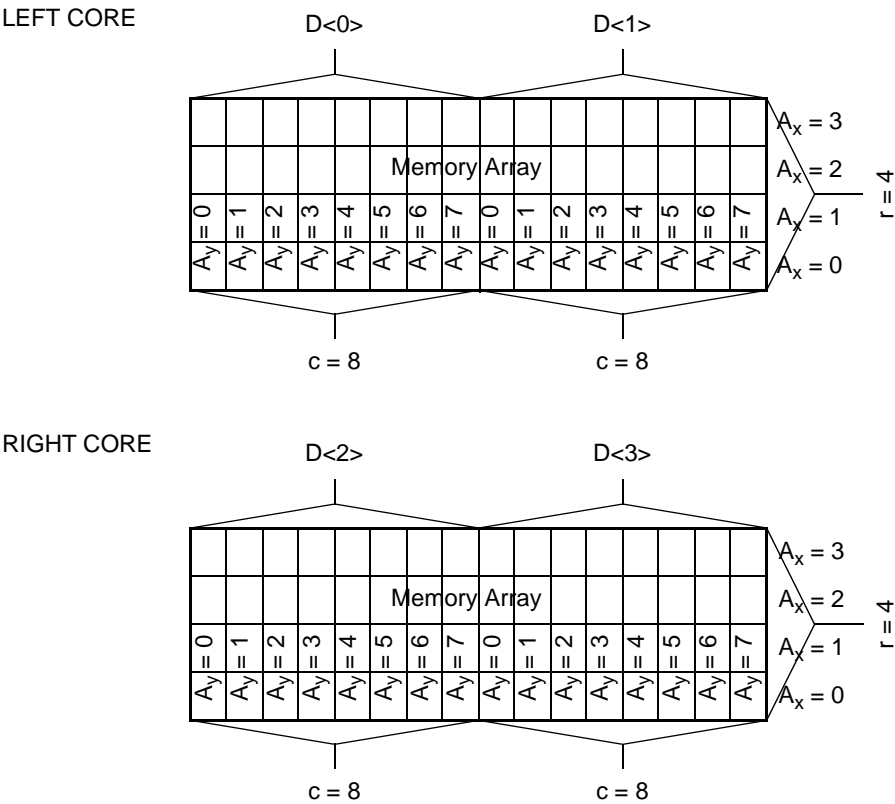
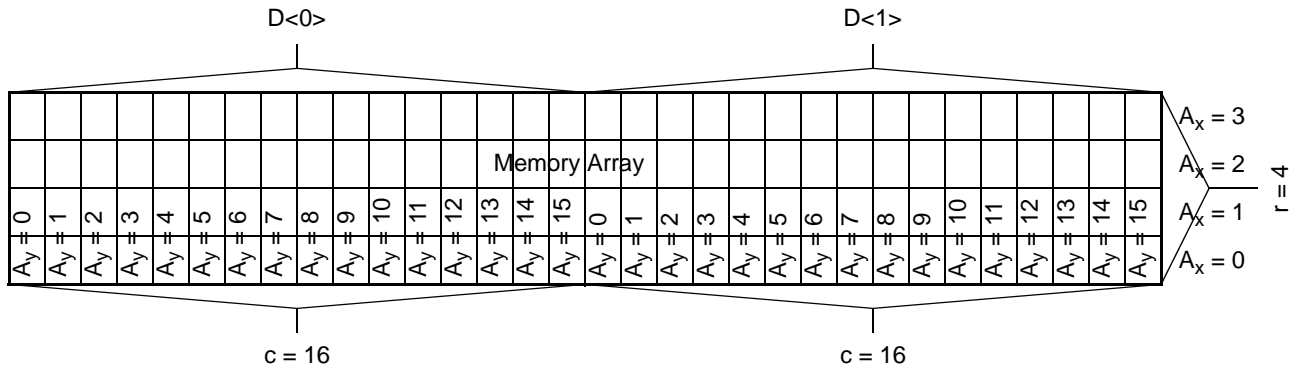
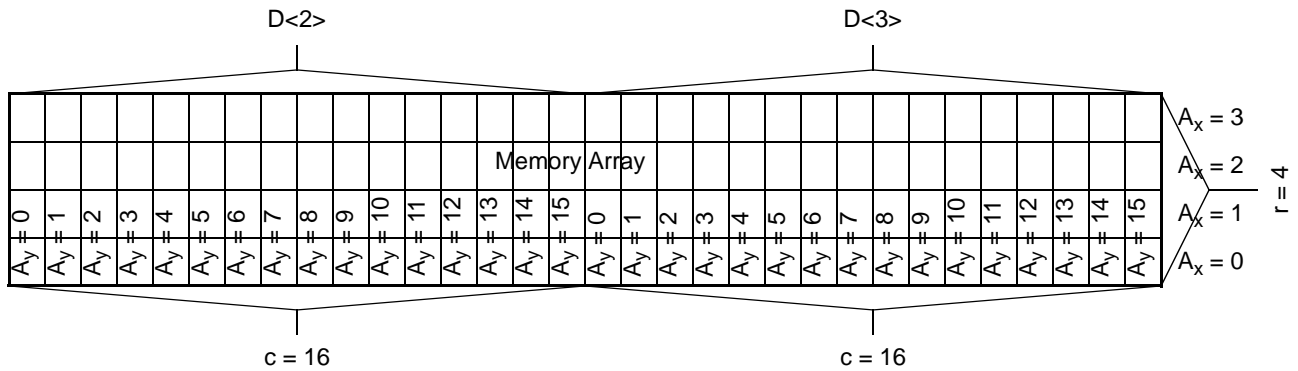


Figure 3-10. Single-Port High-Density SRAM Mux 16: Core Address Mapping

LEFT CORE



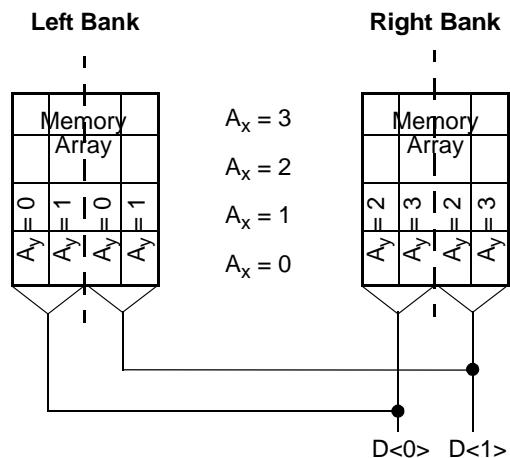
RIGHT CORE



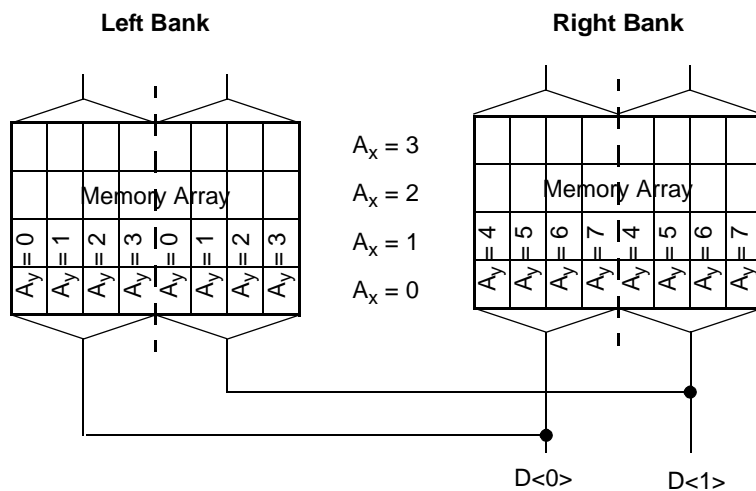
### 3.2.9 Single-Port Low-Power SRAM Core Address Maps (ra1sl)

An example of the standard physical core mapping for all possible low-power mux values is shown in Figures 3-11 to 3-13.

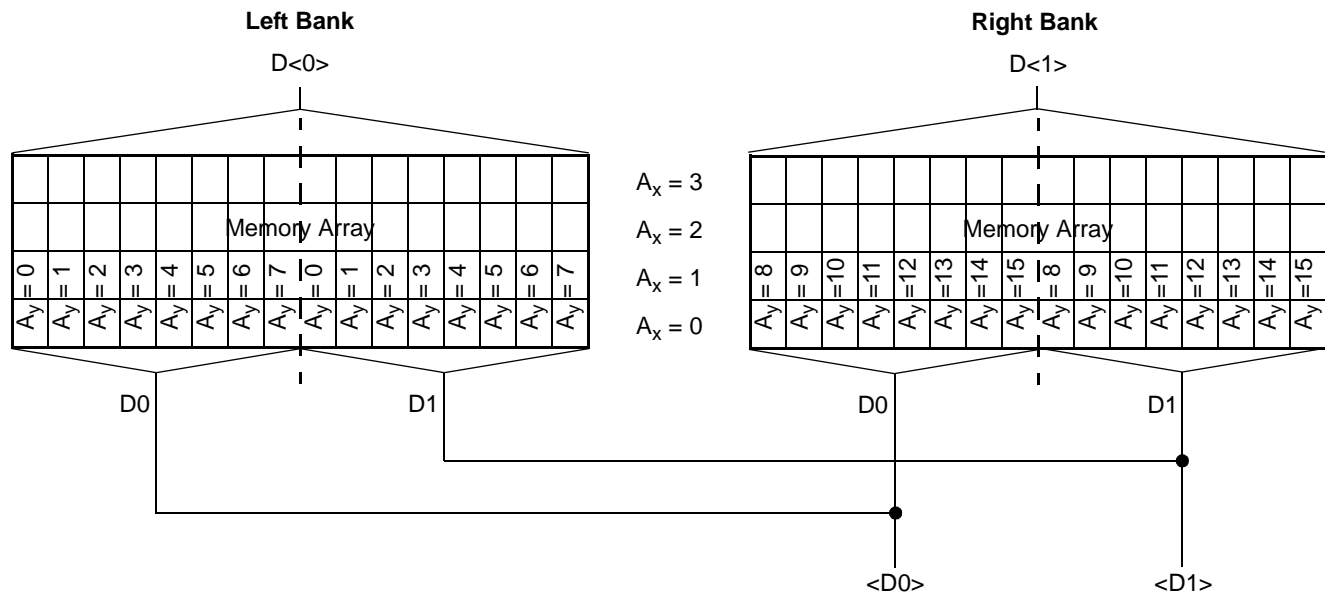
**Figure 3-11. Mux 4: Core Address Mapping**



**Figure 3-12. Mux 8: Core Address Mapping**





**Figure 3-13. Mux 16: Core Address Mapping**

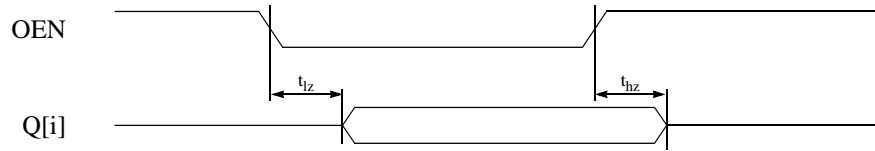
### 3.2.10 Single-Port SRAM Timing Specifications (ra1sh, ra1shd, ra1sd, ra1sl)

This section contains the timing diagrams, timing parameters, and power parameters for the synchronous single-port SRAMs. For detailed pin descriptions, see Table 3-2 on page 3-6.

#### 3.2.10.1 Single-Port SRAM Timing Diagrams (ra1sh, ra1shd, ra1sd, ra1sl)

Figures 3-14 to 3-16 show timing diagrams for high-speed/density, low-power, or high-density synchronous single-port SRAMs. Standard rising/falling delays and slews percentages are shown in these diagrams. Some generators may be designed with different percentages. Check a GUI generated postscript datasheet to verify the delay and slew values for a particular instance.

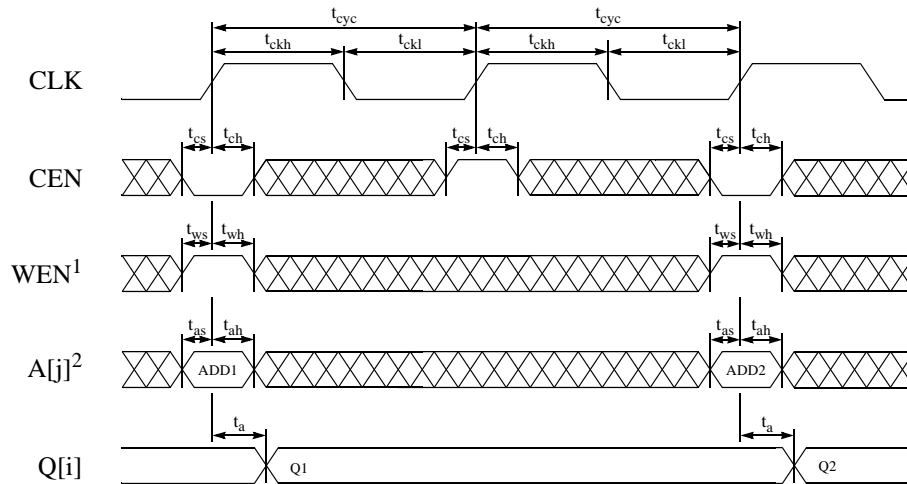
**Figure 3-14. Single-Port SRAM Output-Enable Timing<sup>1</sup>**



Rising delays are measured at 50% VDD and falling delays are measured at 50% VDD.  
Rising and falling slews are measured from 10% VDD to 90% VDD.

<sup>1</sup> Output enable capability applies to some 180nm and older, generators.

**Figure 3-15. Single-Port SRAM Read-Cycle Timing**

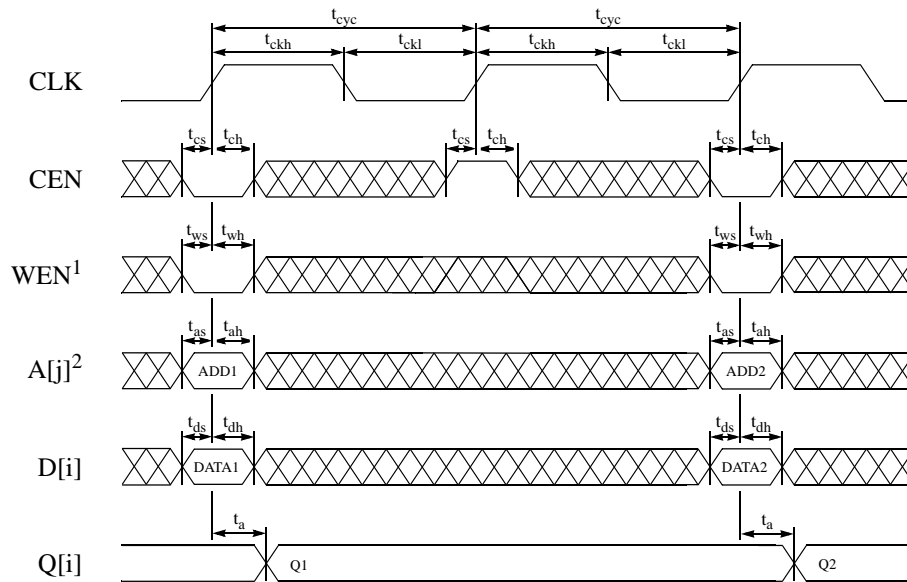


Rising delays are measured at 50% VDD and falling delays are measured at 50% VDD.  
Rising and falling slews are measured from 10% VDD to 90% VDD.

<sup>1</sup> When word-write mask is turned off, WEN is a signal pin as shown in this diagram.  
When word-write mask is turned on, WEN is a bus.

<sup>2</sup> For 130nm ra1shd/ra2shd processes, ADD1 and ADD2 are changed to ADDR1 and ADDR2, respectively.

Figure 3-16. Single-Port SRAM Write-Cycle Timing



Rising delays are measured at 50% VDD and falling delays are measured at 50% VDD. Rising and falling slews are measured from 10% VDD to 90% VDD.

<sup>1</sup> When word-write mask is turned off, WEN is a signal pin as shown in this diagram. When word-write mask is turned on, WEN is a bus.

<sup>2</sup> For 130nm ra1shd/ra2shd processes, ADD1 and ADD2 are changed to ADDR1 and ADDR2, respectively.

### 3.2.10.2 Single-Port SRAM Timing Parameters (ra1sh, ra1shd, ra1sd, ra1sl)

The GUI generated postscript datasheets and the ASCII datatable contain timing parameters listed in Table 3-6.

**Table 3-6. Single-Port SRAM Timing Parameters**

Parameter	Symbol
Cycle time	$t_{cyc}$
Access time <sup>1,2</sup>	$t_a$
Address setup	$t_{as}$
Address hold	$t_{ah}$
Chip enable setup	$t_{cs}$
Chip enable hold	$t_{ch}$
Write enable setup	$t_{ws}$
Write enable hold	$t_{wh}$
Data setup	$t_{ds}$
Data hold	$t_{dh}$
Output enable to hi-Z <sup>3</sup>	$t_{hz}$
Output enable active <sup>1, 3</sup>	$t_{lz}$
Clock high (minimum pulse width)	$t_{ckh}$
Clock low (minimum pulse width)	$t_{ckl}$
Clock rise slew (maximum transition time)	$t_{ckr}$

<sup>1</sup> The ASCII datatable and postscript datasheet show fixed delay values. These parameters have a load dependence ( $K_{load}$ ), which is used to calculate:

TotalDelay = FixedDelay + ( $K_{load} \times C_{load}$ ), for timing views.

<sup>2</sup> Access time is defined as the slowest possible output transition for the typical and slow corners, and the fastest possible output transition for the fast corner.

<sup>3</sup> The OEN pin applies to some 180nm, and older, generators.

Typical and slow timing models are generated with maximum delays, and the fast model is generated with minimum delays. ARM recommends that critical path, setup and hold analysis be performed for all corners.

### 3.2.10.3 Single-Port SRAM Power Parameters (ra1sh, ra1shd, ra1sd, ra1sl)

The GUI contains an ASCII datatable that provides characterization values for each corner, per instance. These values are also available in a generated postscript datasheet. Table 3-7 shows the single-port SRAM power parameters.

**Table 3-7. Single-Port SRAM Power Parameters**

Parameter	Symbol
AC Current <sup>1, 4</sup>	$i_{cc}$
Read AC Current <sup>4</sup>	$i_{cc\_r}$
Write AC Current <sup>4</sup>	$i_{cc\_w}$
Peak Current <sup>4</sup>	$i_{cc\_peak}$
Deselected Current <sup>2, 4</sup>	$i_{cc\_desel}$
Standby Current <sup>3</sup>	$i_{cc\_standby}$

<sup>1</sup> Value assumes 50% read and write operations, where all addresses and 50% of input and output pins switch. This value is an average of the read and write current ( $i_{cc\_r}$ ,  $i_{cc\_w}$ ) values.

<sup>2</sup> Value assumes the memory is deselected, all addresses switch, and 50% of data input pins switch. The logic-switching component of deselected power becomes negligibly small if the input pins are held stable by externally controlling these signals with chip select.

<sup>3</sup> Value is independent of frequency and assumes all inputs and outputs are stable.

<sup>4</sup> For most generators, value shows dynamic current without leakage (standby) component. See the Power table in your generator's postscript datasheet to determine if your generator includes a DC leakage component.

The current values shown in datasheets and datatables are based on certain assumptions. See "Current Calculations" on page 3-41 for instructions on recalculating the current for a specific design. See "Noise Limits" on page 3-45 for more information related to power considerations.

## 3.3 Synchronous Dual-Port SRAM Architecture and Timing Specifications

This section describes the synchronous dual-port SRAM generator architecture, which includes pin descriptions, logic tables, block diagrams, core address maps, timing diagrams, and timing and power parameters. Executable names for applicable generators are provided for your reference.

### 3.3.1 Dual-Port SRAM Description (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

Details about the architecture of port A in the RAM is provided below. This explanation is also applicable to port B.

The synchronous dual-port SRAM has two ports for the same memory location. Both ports can be independently accessed for read or write operations.

SRAM access is synchronous and is triggered by the rising edge of the clock, CLKA. Input address, input data, write enable, and chip enable are latched by the rising edge of the clock, subject to individual setup and hold times.

The value of chip enable must be low ( $CENA=0$ ) for a read or write operation to occur. The SRAM enters read mode when the value of chip enable is low and the value of write enable is high. During read mode, data is read from the memory location specified on the address bus  $AA[j:0]$  and appears on the data output bus  $QA[i:0]$ .

If the word-write mask feature is *not* implemented (word-write mask = off), the SRAM enters write mode when the value of chip enable is low ( $CENA=0$ ) and the value of write enable is low ( $WENA=0$ ). During write mode, data on the data input bus  $DA[i:0]$  is written into the memory location specified on the address bus  $AA[j:0]$ .

If the word-write mask feature is implemented (word-write mask = on), data on the data input bus  $DA[i:0]$  is partitioned to the write enable bus  $WENA[k:0]$ . Each  $WENA[k]$  pin has a distinct latched value, making each partition individually selectable. When the value of a write enable pin  $WENA[k]$  is low, the corresponding data partition is selected, and its data is written to the memory location specified on the address bus  $A[j:0]$ , and driven through to the data output bus  $QA[i:0]$ .

For example, a SRAM with 32 bits and a word partition size of 8 ( $wp\_size = 8$ ) will have four write enable pins: WENA[0], WENA[1], WENA[2], and WENA[3]. The write enable pin WENA[0] corresponds to the data partition DA[7:0]; the write enable pin WENA[1] corresponds to the data partition DA[15:8]; the write enable pin WENA[2] corresponds to the data partition DA[23:16]; the write enable pin WENA[3] corresponds to the data partition DA[31:24]. If the values of WENA[0] and WENA[3] are low, and the values of WENA[1] and WENA[2] are high, the data bits DA[7:0] and DA[31:24] will be written to the memory and driven through to the data output bus QA[7:0], QA[31:24]. Even if data is presented on DA[15:8] and DA[23:16], this data is *not* written to the memory and is *not* driven through to the data output bus. Instead, QA[15:8] and QA[23:16] will be the result of a read operation.

Some 180nm or older generators have an output enable (OEN) pin. If the value of the output enable signal is high (OENA=1), data on the output bus QA[i:0] is placed in a high impedance state. However, while it is in the high impedance state, every read or write operation continues to update the internal output data latch. When the value of the output enable signal is low (OENA=0), the data appears on the output bus QA[i:0]. The SRAM is allowed to access non-existing physical addresses, but the outputs will be unknown.

For 130nm dual-port SRAM generators only, when the memory redundancy feature is selected, the generator adds redundant bits to the user-defined bits before creating the desired views. For example, if you define a 32 bit/word memory with one bit of redundancy, the generator creates a  $32 + 1 = 33$  bit/word memory.

Word write-mask is supported with memory redundancy, however the only legal value for word partition size is 1.

Power dissipation is minimized using static circuit implementations. A standby mode is provided to further reduce power dissipation during periods of non-operation (CENA=1). While in standby mode, address and data inputs are disabled; data stored in the memory is retained, but the memory cannot be accessed for reads or writes. You can eliminate switching current in the input stages and reduce deselected current to near leakage by holding all input pins steady during standby mode.

Static power consumption of the SRAM is limited to leakage provided all of the input signals except CLK are held steady.

Address contention occurs when both ports simultaneously access the same address. In this case, both ports will read the same data.

3.3.2 Dual-Port SRAM Pins (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

Figure 3-17 shows basic pins for the dual-port SRAM generator.

Figure 3-17. Dual-Port SRAM Basic Pins

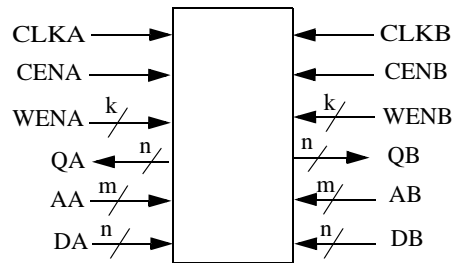


Table 3-8 provides the dual-port (ports A and B) SRAM generator pin descriptions.

Table 3-8. Pin Descriptions for Dual-Port SRAM Generators

Name	Type	Description
<b>Basic Pins</b>		
AA[m-1:0], AB[m-1:0]	Input	Addresses (AA[0] = LSB), (AB[0] = LSB)
DA[n-1:0], DB[n-1:0]	Input	Data inputs (DA[0] = LSB), (DB[0] = LSB)
CENA, CENB	Input	Chip Enables, active low
WENA[*], WENB[*]	Input	Write Enables, active low. *If word-write mask is enabled, this becomes a bus.
CLKA, CLKB	Input	Clocks
QA[n-1:0], QB[n-1:0]	Output	Data Outputs (QA[0] = LSB), (QB[0] = LSB)
OENA, OENB	Output	Tristate Output Enables <sup>1</sup>

<sup>1</sup> The OEN pin applies to some 180nm generators.



### 3.3.3 Dual-Port SRAM Logic Tables (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

This section provides logic tables for basic dual-port SRAM functions and for the individual test and repair functions. Information applies to both port A and port B. The wen pin is WEN when word-mask is on and WEN[ ] when it is off.

Table 3-9 shows the logic functions for basic dual-port SRAM generator functions.

**Table 3-9. Dual-Port SRAM Basic Functions**

CEN	wen	Data Output	Mode	Function
H	X	Last Data	Standby	Address inputs are disabled; data stored in the memory is retained, but memory cannot be accessed for new reads or writes. Data outputs remain stable.
L	L	Data In	Write	<p><b>Word-write:</b> Data on the data input bus, D[n-1:0], is written to the memory location specified by the address bus, A[m-1:0]; and is driven through to the data output bus Q[n-1:0]</p> <p><b>Bit-write:</b> The corresponding data partition is selected by the write enables. WEN[k-1:0]; and that data is written to the memory location specified by the address bus, A[m-1:0], and is driven through to the data output bus Q[n-1:0].</p> <p>Data on the data input bus, D[n-1:0], is written to the memory location specified by the address bus, A[m-1:0]; and is driven through to the data output bus Q[n-1:0]</p>
L	H	SRAM data	Read	Data on the data output bus Q[n-1:0] is read from the memory location specified on the address bus A[m-1:0].

### 3.3.4 Dual-Port SRAM Parameters (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

The standard input and block parameters of a synchronous dual-port SRAM are described in Tables 3-10 and 3-11. Note that the ranges differ between 150nm/130nm and 250nm/180nm generators. You can also see your generator GUI for the specific ranges for your generator. If you enter an invalid value and update the GUI, the message pane at the bottom of the GUI displays an error message and the specific range for your generator.

**Table 3-10. Dual-Port 150nm/130nm SRAM Parameters**

Input Parameters		
Parameter	Ranges	
number of words	mux = 4	128 to 2048, increment = mux • 2
	mux = 8	256 to 4096, increment = mux • 2
	mux = 16	512 to 8192, increment = mux • 2
number of bits	mux = 4	2 to 128, increment = 1
	mux = 8	2 to 64, increment = 1
	mux = 16	2 to 32, increment = 1
frequency (MHz)	1 to 1/t <sub>cyc</sub> • 1000 , increment = 1	
word partition size <sup>1</sup>	1 to min (36, bits-1) increment = 1	
top chip metal layer support	m4 to top metal layer supported by design process	
top generator metal layer	m4	
horizontal ring layer	m1, m2, m3, or m4	
vertical ring layer	m1, m2, m3, or m4	
ring width (μm)	2 to 10,000	
redundancy (130nm only)	on/off, default is off; when 'on' is selected, the default/only redundancy bit value is 1	
Block Parameters		
Parameter	Ranges	
total memory bits	256 to 262,144, total bits = words • bits	
rows in memory matrix	32 to 512, increment = 2, rows = words / mux	
columns in memory matrix	8 to 512, increment = mux, columns = bits • mux	
address lines	mux = 4	7 to 11
	mux = 8	8 to 12
	mux = 16	9 to 13
output drive strength	See“Running the Generator from the Graphical User Interface (GUI)” on page 2-3	

<sup>1</sup> The input pin capacitance for each pin of the write enable bus is proportional to the size of the word partition. For example, an instance with bits=32 and wp\_size=24 will have two partitions, one with 24 bits and one with 8 bits. The write enable pin for the 24 bit partition will have a significantly larger input pin capacitance than the write enable pin for the 8 bit partition. When modeling write enable timing, the write enable pin with the largest capacitance is used in the typical and slow corner timing models. The write enable pin with the smallest capacitance is used in the fast corner timing models. ARM recommends that the critical path, setup and hold analysis be performed for all corners.

**Table 3-11. Dual-Port 250nm/180nm SRAM Parameters**

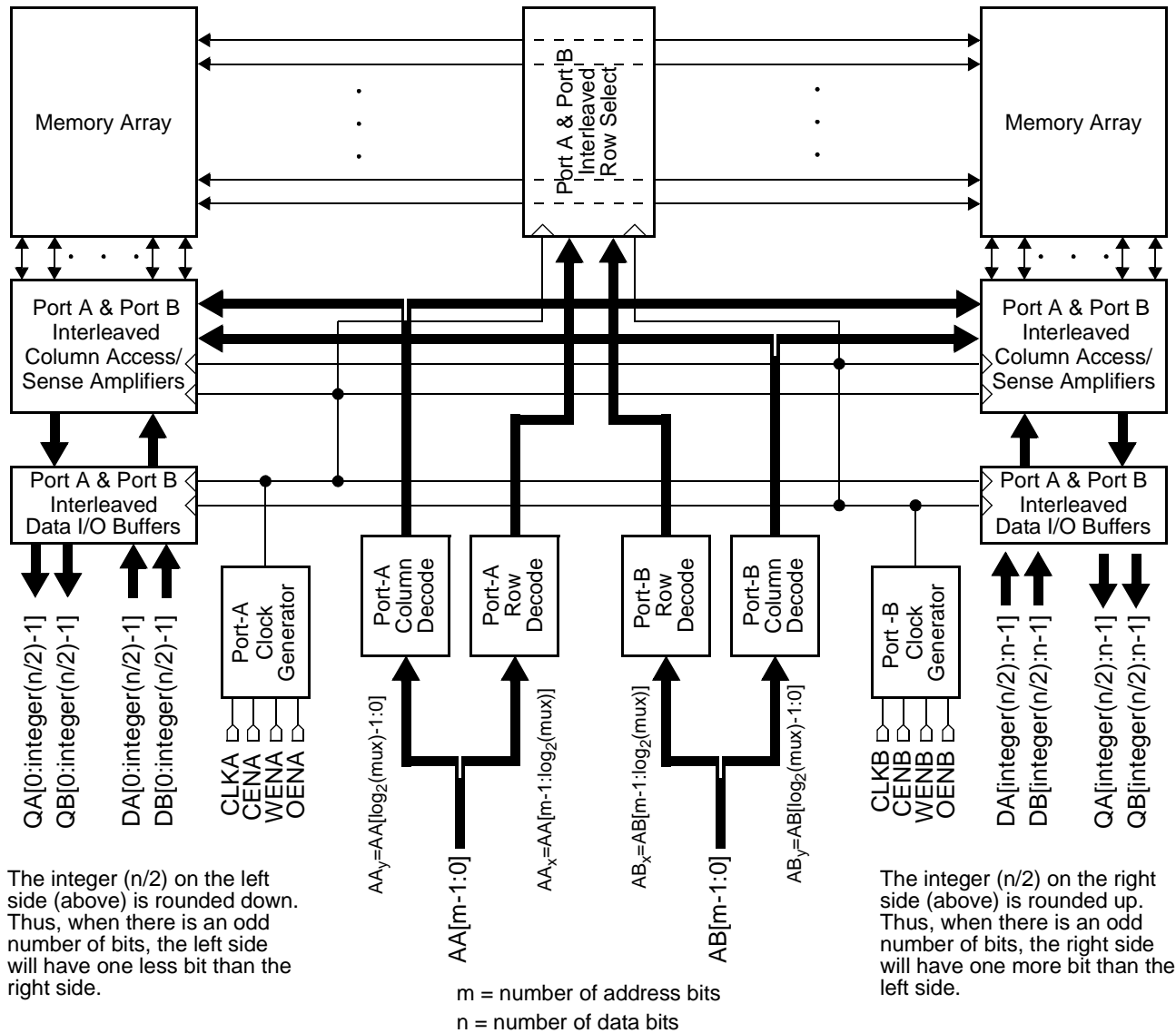
Input Parameters		
Parameter	Ranges	
number of words	mux = 4	16 to 2048, increment = mux • 2
	mux = 8	32 to 4096, increment = mux • 2
	mux = 16	64 to 8192, increment = mux • 2
number of bits	mux = 4	2 to 128, increment = 1
	mux = 8	2 to 128, increment = 1
	mux = 16	2 to 64, increment = 1
frequency (MHz)	1 to $1/t_{cyc} \bullet 1000$ , increment = 1	
word partition size <sup>1</sup>	1 to min (36, bits-1) increment = 1	
top chip metal layer support	m4 to top metal layer supported by design process	
top generator metal layer	m4	
horizontal ring layer	m1, m2, m3, or m4	
vertical ring layer	m1, m2, m3, or m4	
ring width (μm)	2 to 10,000	
Block Parameters		
Parameter	Ranges	
total memory bits	32 to 524,288, total bits = words • bits	
rows in memory matrix	4 to 512, increment = 2, rows = words / mux	
columns in memory matrix	8 to 1024, increment = mux, columns = bits • mux	
address lines	mux = 4	4 to 11
	mux = 8	5 to 12
	mux = 16	6 to 13
output drive strength	See“Running the Generator from the Graphical User Interface (GUI)” on page 2-3.	

<sup>1</sup> The input pin capacitance for each pin of the write enable bus is proportional to the size of the word partition. For example, an instance with bits=32 and wp\_size=24 will have two partitions, one with 24 bits and one with 8 bits. The write enable pin for the 24 bit partition will have a significantly larger input pin capacitance than the write enable pin for the 8 bit partition. When modeling write enable timing, the write enable pin with the largest capacitance is used in the typical and slow corner timing models. The write enable pin with the smallest capacitance is used in the fast corner timing models. ARM recommends that the critical path, setup and hold analysis be performed for all corners.

### 3.3.5 Dual-Port SRAM Block Diagrams (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

The SRAM has two ports for the same memory locations. Both ports can be independently accessed for read or write operations. The two ports function identically. Figure 3-18 shows the block diagram for high-density or high-speed/density dual-port SRAMs.

**Figure 3-18. Dual-Port High-Speed/Density and High-Density SRAM Block Diagram**



**Notes:**

When the word-write mask (wwm) option is turned on, WEN is a bus. When wwm is turned off, WEN is a signal pin. Output enable (OEN) pins apply to some 180nm and older generators.

### 3.3.6 Dual-Port SRAM Core Address Maps (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

This section describes the core address diagrams for high-speed/density and high-density synchronous dual-port SRAMs. Core address maps are the same for high-speed/density and high-density dual-port SRAMs.

Figures 3-19 to 3-21 show the physical core mapping for port A of each mux value. The physical core mapping for port B is identical to port A. For example, addresses  $AA_x$  and  $AB_x$  will access the same physical core cell.

**Figure 3-19. Dual-Port SRAM Mux 4: Core Address Mapping**

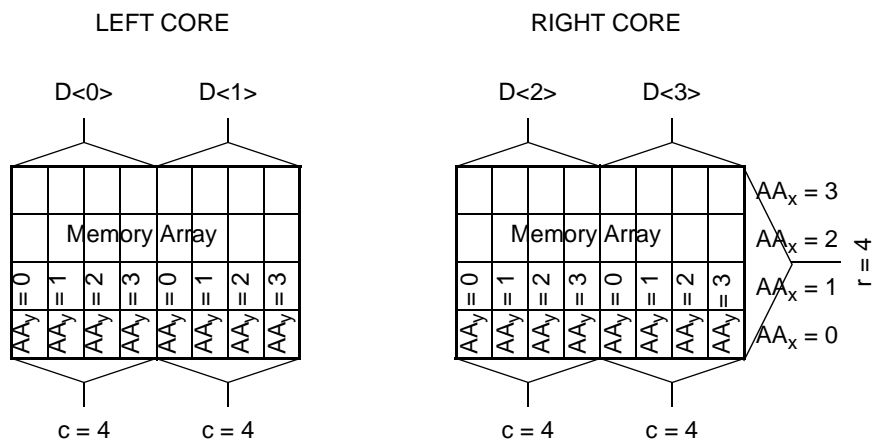


Figure 3-20. Dual-Port SRAM Mux 8: Core Address Mapping

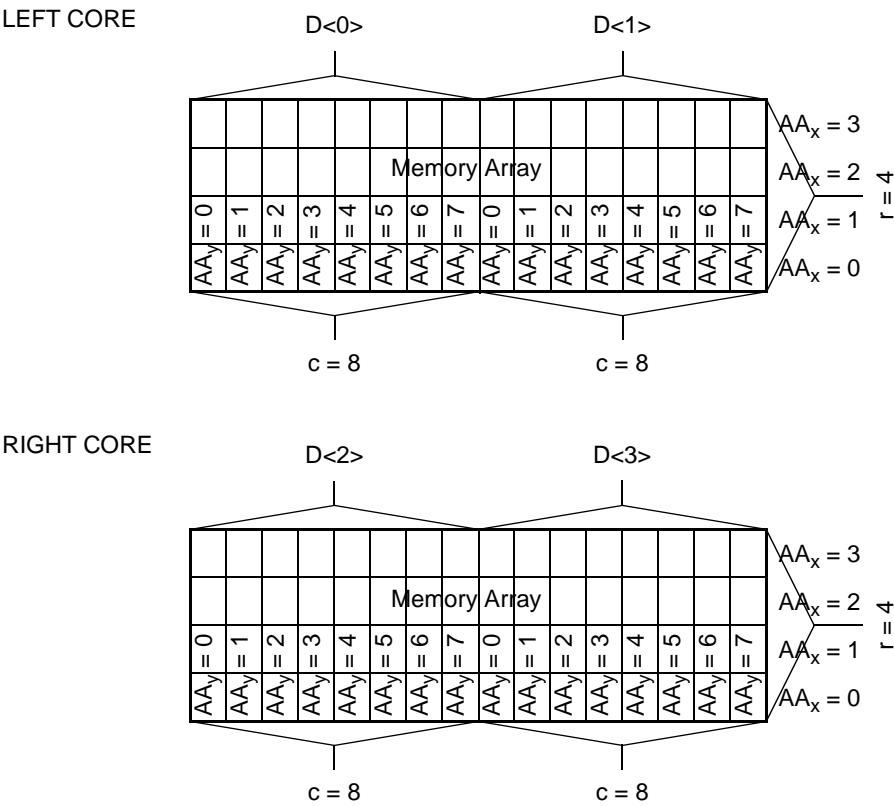
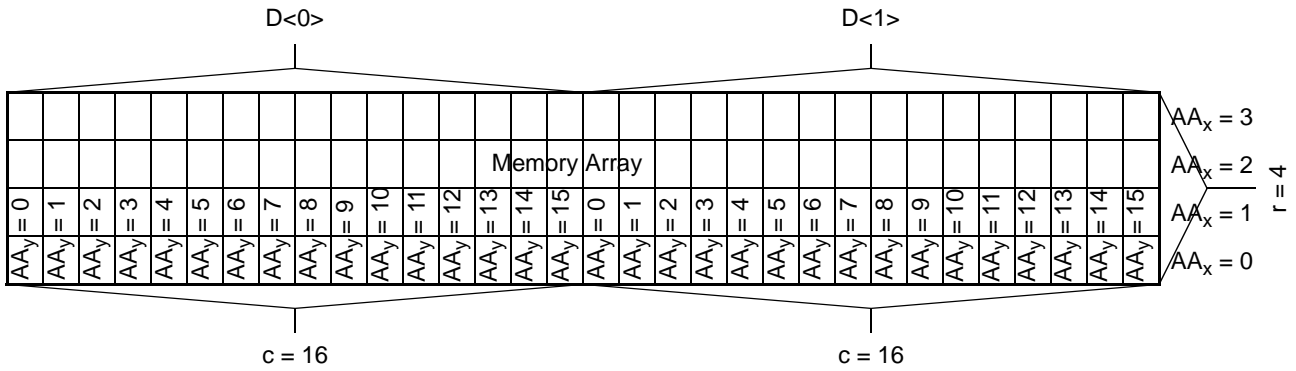
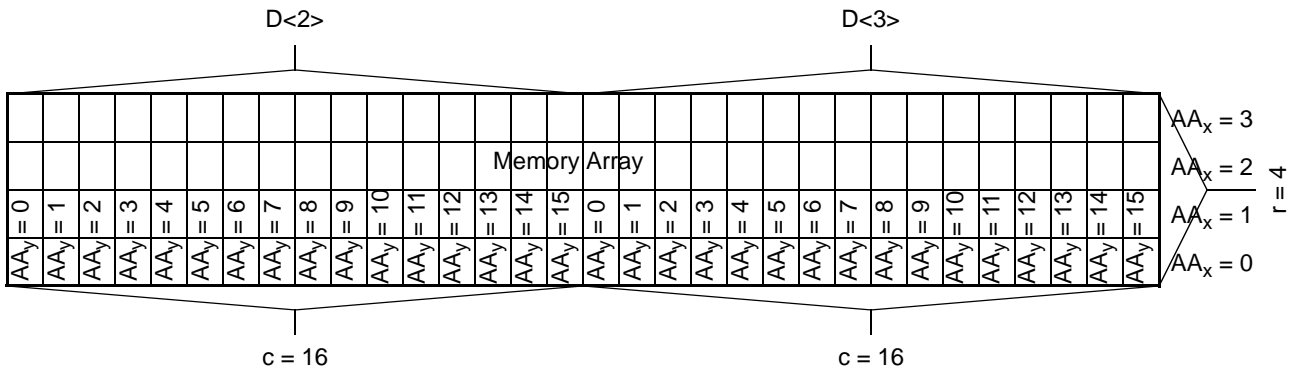


Figure 3-21. Dual-Port SRAM Mux 16: Core Address Mapping

LEFT CORE



RIGHT CORE



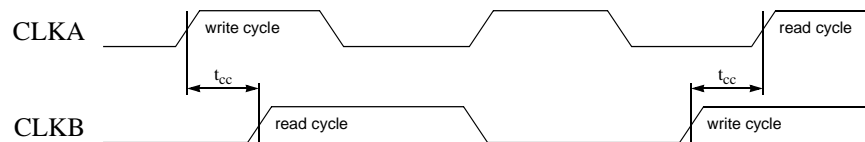
### 3.3.7 Dual-Port SRAM Timing Specifications (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

This section contains timing diagrams, timing parameters and power parameters for the synchronous dual-port high speed/density and high density SRAMs.

#### 3.3.7.1 Dual-Port SRAM Clock Timing Diagrams (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

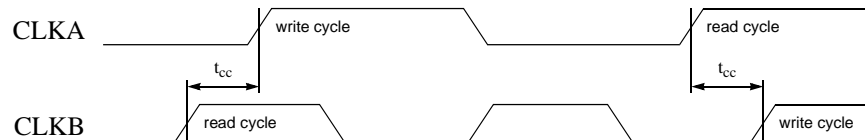
Figures 3-22 to 3-24 show clock timing diagrams for high-speed/density and high-density synchronous dual-port SRAMs. Standard rising/falling delays and slews percentages are shown in these diagrams. Some generators may be designed with different percentages. Check a GUI generated postscript datasheet to verify the delay and slew values for a particular instance.

**Figure 3-22. Dual-Port SRAM Write-Read Clock Timing (Accessing Same Address)**



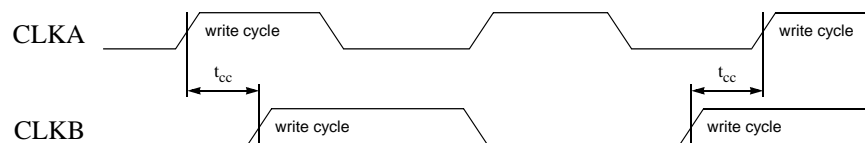
Rising delays are measured at 50% VDD and falling delays are measured at 50% VDD.  
Rising and falling slews are measured from 10% VDD to 90% VDD.

**Figure 3-23. Dual-Port SRAM Read-Write Clock Timing (Accessing Same Address)**



Rising delays are measured at 50% VDD and falling delays are measured at 50% VDD.  
Rising and falling slews are measured from 10% VDD to 90% VDD.

**Figure 3-24. Dual-Port SRAM Write-Write Clock Timing (Accessing Same Address)**



Rising delays are measured at 50% VDD and falling delays are measured at 50% VDD.  
Rising and falling slews are measured from 10% VDD to 90% VDD.



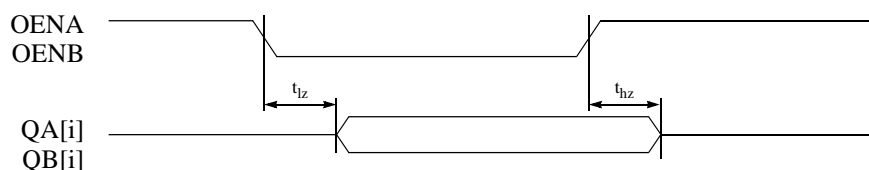
Table 3-12 illustrates read and write behavior during clock contention, when both ports access the same address.

**Table 3-12. Dual-Port SRAM Read and Write Behavior When Accessing Same Address**

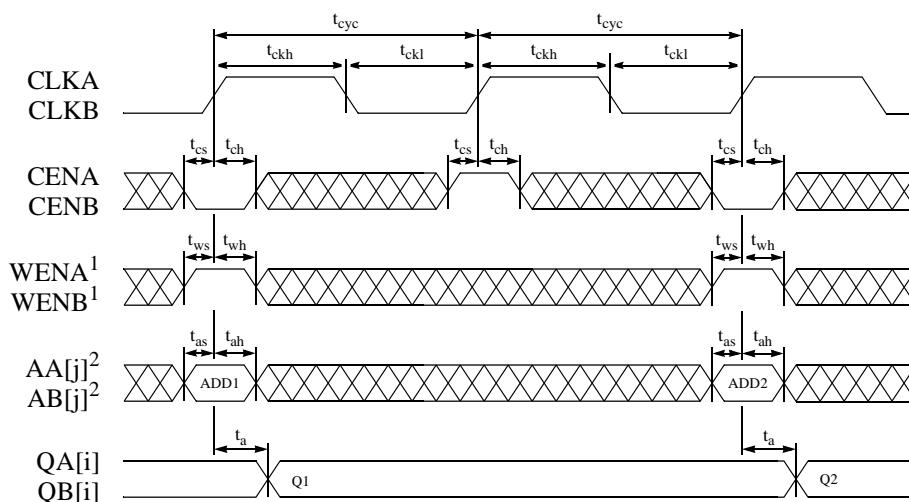
Action	Condition	Behavior
write from one port then read from the other port	$t_{cc}$ is satisfied (see Figure 3-22)	write OK D-to-Q write through OK read (new data) OK
	$t_{cc}$ is not satisfied (see Figure 3-22)	If DPCCM = ON write succeeds D-to-Q write through OK for write port read port produces an X at the output  If DPCCM = OFF write fails, an X is placed in the memory location indicated by the address on the two ports  D-to-Q write through OK for write port  read port produces an X at the output
read from one port, followed by write from the other port	$t_{cc}$ is satisfied (see Figure 3-23)	write OK D-to-Q write through OK read (old data) OK
	$t_{cc}$ is not satisfied (see Figure 3-23)	If DPCCM = ON write succeeds D-to-Q write through OK for write port read port produces an X at the output  If DPCCM = OFF write fails, an X is placed in the memory location indicated by the address on the two ports  D-to-Q write through OK for write port  read port produces an X at the output
write from one port then write from the other port	$t_{cc}$ is satisfied (see Figure 3-24)	both writes OK (second write overwrites first write) D-to-Q write throughs OK
	$t_{cc}$ is not satisfied (see Figure 3-24)	both writes fail, an X is placed in the memory location indicated by the address on the two ports  D-to-Q write throughs OK
read from one port then read from the other port	no restriction	both reads OK

Some 180nm high-speed dual-port SRAM generators have an OEN pin. High-density dual-port SRAM generators do not contain OEN pins.

### Figure 3-25. Dual-Port SRAM Output-Enable Timing

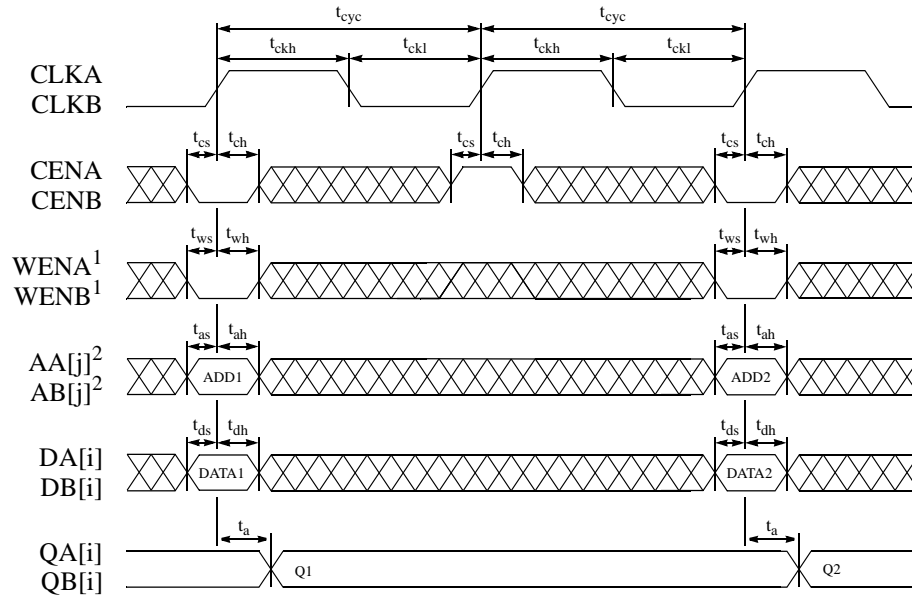


### Figure 3-26. Dual-Port SRAM Read-Cycle Timing



<sup>2</sup> For 130nm ra1shd/ra2shd processes, ADD1 and ADD2 are changed to ADDR1 and ADDR2, respectively.

Figure 3-27. Dual-Port SRAM Write-Cycle Timing



Rising delays are measured at 50% VDD and falling delays are measured at 50% VDD.  
Rising and falling slews are measured from 10% VDD to 90% VDD.

<sup>1</sup> When word-write mask is turned off, WEN is a signal pin as shown in this diagram.  
When word-write mask is turned on, WEN is a bus.

<sup>2</sup> For 130nm ra1shd/ra2shd processes, ADD1 and ADD2 are changed to ADDR1 and ADDR2, respectively.

### 3.3.7.3 Dual-Port SRAM Timing Parameters (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

The GUI generated postscript datasheets and the ASCII datatable contain timing parameters listed in Table 3-13. Some 180nm high-speed dual-port SRAM generators may have the OEN parameter. High-density dual-port SRAM generators do not have the OEN parameter.

**Table 3-13. Dual-Port SRAM Timing Parameters**

Parameter	Symbol
Cycle time	$t_{cyc}$
Access time <sup>1,2</sup>	$t_a$
Address setup	$t_{as}$
Address hold	$t_{ah}$
Chip enable setup	$t_{cs}$
Chip enable hold	$t_{ch}$
Write enable setup	$t_{ws}$
Write enable hold	$t_{wh}$
Data setup	$t_{ds}$
Data hold	$t_{dh}$
Output enable to hi-Z	$t_{hz}$
Output enable active <sup>1</sup>	$t_{lz}$
Clock high (minimum pulse width)	$t_{ckh}$
Clock low (minimum pulse width)	$t_{ckl}$
Clock rise slew (maximum transition time)	$t_{ckr}$
Clock collision	$t_{cc}$

<sup>1</sup> The ASCII datatable and postscript datasheet show fixed delay values. These parameters have a load dependence ( $K_{load}$ ), which is used to calculate:

TotalDelay = FixedDelay + ( $K_{load} \times C_{load}$ ), for timing views.

<sup>2</sup> Access time is defined as the slowest possible output transition for the typical and slow corners, and the fastest possible output transition for the fast corner.

Typical and slow timing models are generated with maximum delays, and the fast model is generated with minimum delays. ARM recommends that critical path, setup and hold analysis be performed for all corners.

### 3.3.7.4 Dual-Port SRAM Power Parameters (ra2sh, ra2shd, ra2shd\_rd, ra2sd)

The GUI contains an ASCII datatable that provides characterization values for each corner, per instance. These values are also available in a generated postscript datasheet. Table 3-14 shows the dual-port SRAM power parameters.

**Table 3-14. Dual-Port SRAM Power Parameters**

Parameter	Symbol
AC Current <sup>1, 4</sup>	$i_{cc}$
Read AC Current <sup>4</sup>	$i_{cc\_r}$
Write AC Current <sup>4</sup>	$i_{cc\_w}$
Peak Current <sup>4</sup>	$i_{cc\_peak}$
Deselected Current <sup>2, 4</sup>	$i_{cc\_desel}$
Standby Current <sup>3</sup>	$i_{cc\_standby}$

<sup>1</sup> Value assumes 50% read and write operations, where all addresses and 50% of input and output pins switch. This value is an average of the read and write current ( $i_{cc\_r}$ ,  $i_{cc\_w}$ ) values.

<sup>2</sup> Value assumes SRAM is deselected, all addresses switch, and 50% of data input pins switch. The logic-switching component of deselected power becomes negligibly small if the input pins are held stable by externally controlling these signals with chip select.

<sup>3</sup> Value is independent of frequency and assumes all inputs and outputs are stable.

<sup>4</sup> For most generators, value shows dynamic current without leakage (standby) component. See the Power table in your generator's postscript datasheet to determine if your generator includes a DC leakage component.

The current values shown in datasheets and datatables are based on certain assumptions. See "Current Calculations" on page 3-41 for instructions on recalculating the current for a specific design. See "Noise Limits" on page 3-45 for more information related to power considerations.

## 3.4 SRAM Power Structure (ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2shd\_rd, ra2sd)

The following sections explain the power structure options available with single- and dual-port, high-speed/density and high-density SRAM generators. It also explains the power structure options for the single-port low-power SRAM generator.

### 3.4.1 Current Calculations

The average current,  $I_{avg}$ , in mA, for the SRAM instance may be calculated from data reported in the ASCII datatable, as well as the datasheet. The average current reported in the datasheet assumes 50% read and write operations where all addresses and 50% of input and output pins [unloaded] switch. You may choose to recalculate this number based on the percentage of reads and writes in a given design. This value is used to calculate the size of the power bus.

Given:

$c$  = average capacitance of output port (pF);

$n$  = number of ports;

From the datatable,

$$I_{avg} = \text{AC Current} + \left( \frac{1}{2} \cdot cvf \cdot \text{bits} \right) \cdot n$$

From the datasheet,

$$I_{avg} = \text{AC Current} + \left( \frac{1}{2} \cdot cvf \cdot \text{bits} \right) \cdot n$$

The peak current,  $I_p$ , in mA, for the instance is reported in the datasheet and ASCII datatable. This peak current is measured during read/write HSPICE simulations and reflects the maximum simulated value. The amplitude of the peak may be large, but the duration is very short due to ideal circuit behavior.

If the memory is deselected and only the clock switches, then the current is the same as standby current. Standby current assumes no switching, and normal reverse-bias leakage.

———— **Note** ————

NOTE: When the SRAM is deselected, all addresses switch, and 50% of data input pins switch, then current consumption may be up to 30-40% of *icc* because the input latches are open, and the internal logic can switch. The logic-switching component of deselected power becomes small if the address, data, and write enable pins are held stable by externally controlling these signals with chip select.

---

From the datatable,

$$I_p = icc\_peak$$

From the datasheet,

$$I_p = \text{Peak Current}$$

### 3.4.2 Power Distribution Methodology

Your chip-level power distribution must ensure that the wire widths supplying power to the SRAM satisfy electromigration guidelines and limit the average and peak voltage drop in the power wires to an acceptable value. To ensure memory timing accuracy, the voltage supplied to the memory must be the same as the characterized voltage. The SRAM minimum supply wire widths are calculated as follows.

For example, given:

$W_{em}$  = connection width based on electromigration ( $\mu\text{m}$ );

$W_{iravg}$  = connection width based on average voltage ( $\mu\text{m}$ );

$W_{irp}$  = connection width based on peak voltage ( $\mu\text{m}$ );

$C$  = current density rule constant ( $\text{mA}/\mu\text{m}$ );

$I_{avg}$  = average current consumed by the SRAM ( $\text{mA}$ );

$I_p$  = peak current consumed by the SRAM (mA);

$\Delta V_{iravg}$  = allowable average voltage drop within the power wires on the chip (mV);

$\Delta V_{irp}$  = allowable peak voltage drop within the power wires on the chip (mV);

$L_{eff}$  = effective wire length of power connection from power pad to the SRAM ( $\mu m$ );

$R_m$  = resistance of metal wire (Ohms/square);

$W$  = connection width ( $\mu m$ );

we have:

$$W_{em} = \frac{I_{avg}}{C},$$

$$W_{iravg} = \frac{L_{eff} \cdot R_m}{\Delta V_{iravg}} \cdot I_{avg}$$

$$W_{irp} = \frac{L_{eff} \cdot R_m}{\Delta V_{irp}} \cdot I_p$$

$$W = \max(W_{em}, W_{iravg}, W_{irp})$$

———— **Note** ————

These sample calculations do not take into account the other components on the chip that may be supplied by the same wire. You must adjust wire width accordingly. The  $L_{eff}$  parameter can also be adjusted to account for the varying width of the power wires.

---



### 3.4.3 Supply Connections to Power Rings

The generator has the capability of generating power rings around the SRAM. You must properly size these rings. The size of the rings depends on the chip-level power distribution methodology, the number, width, and placement of supply wire connections to the power rings, and current consumption.

ARM recommends that the current be evenly supplied from the edge of the instance where the I/O pins are located.

In the case of one supply wire connection to the SRAM power rings, the ring width must be at least half the width of the power bus connection. This is possible if the current is approximately equally distributed on either side of the connection into the ring.

In case of multiple connections to the ring, the width may be reduced only if the connections are evenly distributed along the edge where the I/O pins are located. The ring width must remain equal to or greater than half the width of the widest power connection.

Given:

$W_r$  = ring width with one power connection ( $\mu\text{m}$ );

$n$  = number of equal-width-wire power connections at the I/O pins;

we have:

$W_m$  = ring width with  $n$  connections ( $\mu\text{m}$ ) =  $1/n \bullet W_r$

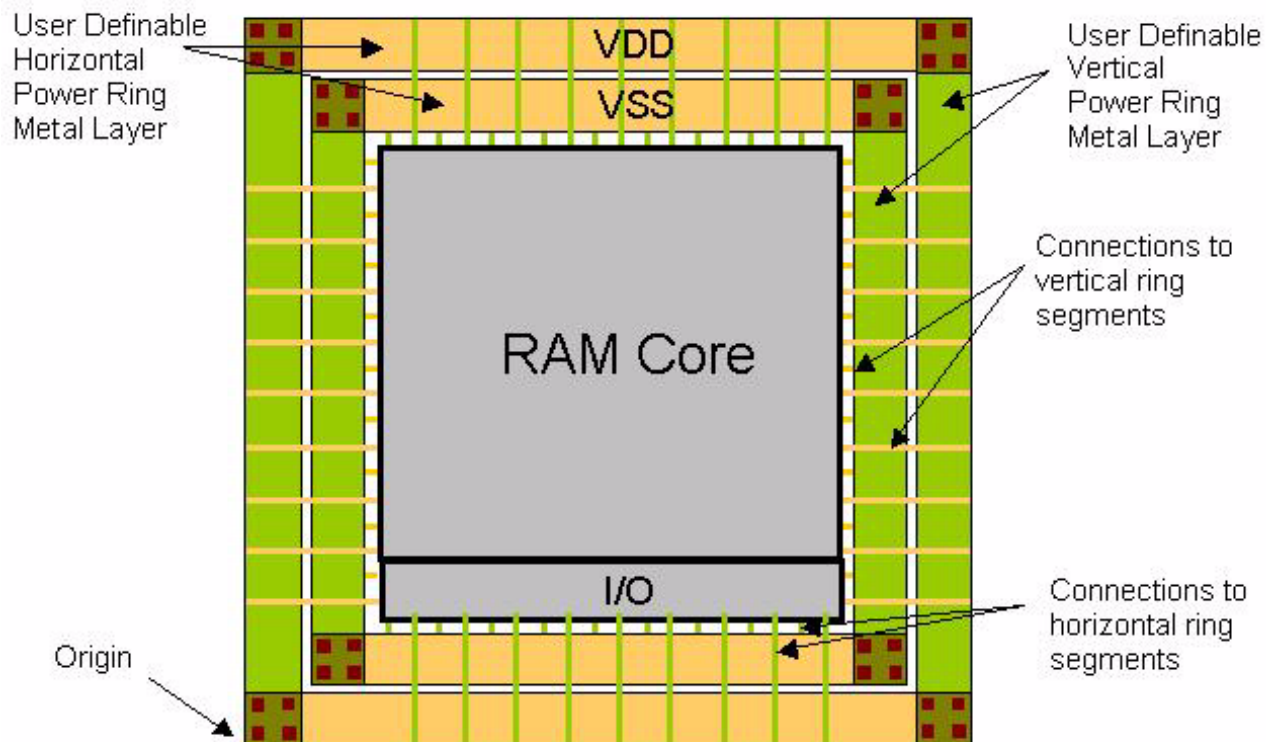
Figure 3-28 shows I/O connections on the bottom edge of the SRAM block. You can use the Inside Ring Type option in the Utilities > Advanced Options menu of the generator GUI to select whether the inside ring is VDD or VSS.

————— **Note** —————

The outside ring power type must be of opposite polarity to the inside ring, so that one ring is VDD and the other ring is VSS. The generator will not function correctly if you have two VDD rings or two VSS rings.

---

**Figure 3-28. Multiple Power and Ground Connections**



### 3.4.4 Noise Limits

The characterized clock noise limit,  $vn\_ck$ , is the maximum CLK voltage allowable for the indicated pulse width without causing a spurious memory cycle or other memory failure. For most generators, the standard pulse width,  $pwn\_ck$ , used in characterizing this limit is 10ns.

The power and ground noise limits,  $vn\_pwr$  and  $vn\_gnd$  respectively, are the maximum supply or ground voltage transition allowable without causing a memory failure. Power and ground noise limits are assured at 10% of the characterized voltage.

## 3.5 SRAM Physical Characteristics (**ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2shd\_rd, ra2sd**)

This section provides physical design characteristics for single- and dual-port, high-speed/density and high-density SRAM generators. It also provides information for single-port low-power SRAM generators. Information such as top metal layer usage, I/O pin connections, and characterization environments are also provided in this section.

### 3.5.1 Top Metal Layer

The generator has the capability of supporting different top-most metal layer designs. For example, a process may support a maximum of eight layers but you may elect to use only five layers for a given chip design. The layout size is the same for all top metal layer options.

All metal layers including metal4 and below are used in the design, and therefore, are blocked for routing. All metal layers above metal4 can be routed over the memory.

### 3.5.2 I/O Connections

Input/output (I/O) pins are located along the bottom edge of the memory block on any of the metal layers. The I/O pins are large enough to accommodate a pre-determined on-grid width wire connection. The pins are designed to be on the grid even when the memory is rotated or placed off the grid. Depending on the chip-level placement of a memory instance, a pin geometry may enclose multiple grid points but, as a worst case, only one is valid. A valid grid point is enclosed by the pin geometry by at least half of a wire width.

The router must access the pin by way of the routing track that corresponds to a valid grid point and is perpendicular to the cell edge. If the router approaches the pin off-grid and then bends the wire underneath the obstruction layer to connect to the valid grid point, a metal spacing or short circuit error may result.

For added flexibility, the pins can be accessed on several different routing layers. In most cases, the pin access layer is determined by the horizontal power ring layer, the memory orientation, and the chip-level routing methodology. The I/O wires must route across the horizontal power ring layer and therefore cannot be the same layer. The horizontal ring layers are displayed in the generator GUI.

### 3.5.3 Characterization Environments

By default, the generator is characterized to fast (ff), typical (tt), and slow (ss) environments, or corners. Your generator may have more corners, or a different set of corners. Only four corners are visible in the ASCII datatable of the generator GUI at one time. Similarly, only four corners are available at one time in the postscript datasheet.

You can determine the characterization corners from the ASCII datatable in the generator GUI. Move your mouse pointer (arrow) over the data columns to view the temperature and voltage corners for each column.

ARM recommends that critical path, setup and hold analyses be performed for all applicable corners.

### 3.6 SRAM Timing Derating (**ra1sh, ra1shd, ra1sd, ra1sl, ra2sh, ra2shd, ra2sd**)

Derating factors are coefficients that the characterization data is multiplied by to arrive at timing data that reflects different operating conditions. Standard Artisan memories do not support a timing derating methodology. By default, timing is provided for three characterization environments: fast, typical, and slow. Some generators may contain more environments.

Several delay calculators and the associated timing views, such as Synopsys and TLF, include a simplistic derating ability and a specified derating factor. The derating methodology supported by these delay calculators and the specified derating factor is not sufficient to accurately model the timing behavior of the memory. There is no derating for the models provided with these memories.

Relying on timing results using derating may lead to memory timing constraint violations and may cause a non-working part.



# 4

## **Generator Views**

This chapter contains the following sections:

- “Overview” on page 4-3
- “Tool Verification” on page 4-4
- “Using the Generator Views” on page 4-5

## **4.1 Overview**

This chapter lists the tools used in designing the generator and describes using various tools to generate instances and views. These details apply to most standard 250nm to 130nm ASL SRAM generators unless otherwise noted.



## 4.2 Tool Verification

The ASL SRAM generators produce standard views and models, that have been verified with the tools defined in the EDA Packages. See Table 4-1. As needed, refer to the README file in your generator to determine the EDA or tool version(s) for your generator.

**Table 4-1. Tools Used for Verification**

View and provided file name	Tool	Vendor
<b>Standard Views</b>		
Verilog (.v)	NC-Verilog	Cadence
VHDL (.vhd)	ModelSim (VHDL)	MTI/Mentor
Repair Verilog (.v)*	NC-Verilog	Cadence
	Design Compiler	Synopsys
Repair VHDL (.vhd)*	ModelSim (VHDL)	MTI/Mentor
	Design Compiler	Synopsys
Synopsys (.lib)	Design Compiler	Synopsys
PrimeTime - STAMP (.data)	PrimeTime	Synopsys
TLF 4.1 (.tlf)	Pearl	Cadence
VCLEF Footprint (.vclef)	Silicon Ensemble	Cadence
Notes: * The Repair Verilog and Repair VHDL tools are supported by Artisan 130nm and 180nm synchronous single-port SRAM, and 130nm synchronous dual-port SRAM, with Flex-Repair, generators. ** LEF support includes an additional antenna LEF file with antenna calculation support per LEF 5.4 definitions.		
<b>Optional Views</b>		
FastScan (.fastscan)	FastScan	Mentor
MBISTArchitect (.mbist)	MBISTArchitect	Mentor
LogicVision (.memlib)	IC Memory BIST	LogicVision
TetraMAX (.tv)	TetraMAX	Synopsys

## 4.3 Using the Generator Views

This section provides information about simulating design modules from views available with standard generators.

### 4.3.1 Using the Verilog Model

After generating the Verilog model, simulate the design module using these steps.

Check the syntax of the Verilog model:

```
verilog <name>.v
```

where *<name>.v* is the Verilog model.

Use the SDF annotator to back-annotate the SDF timing files to the verilog model. An example command is:

```
$sdf_annotate(<sdf_file_name>, <instance>)
```

Run the simulation:

```
verilog <test-bench>.v
```

The simulation output is written to a file, *verilog.log*, which is placed in the current directory.

### 4.3.2 Using the VHDL Model

After generating the VHDL model, simulate the design module using these steps.

The *work* library is necessary for the compilation of any VHDL model. If it does not already exist, create the *work* library:

```
vlib work
```

Compile the library file (only once):

```
vcom <install_dir>/aci/executable/lib/vhdl_lib/  
artisan_lib.vhd
```

Compile the VHDL design:

```
vcom <name>.vhd
```

where *<name>.vhd* is the VHDL model.

Define a test bench in which the design unit is instantiated. It can be referenced through *<name>\_pkgs* by using the clause:

```
use work.<name>_pkgs.all;
```

in the test bench file, where *<name>\_pkgs* is a package included in the generated VHDL model.

Compile your test bench:

```
vcom <test-bench>.vhd
```

After compilation of the test bench, view the simulation results:

```
vsim
```

In the *Startup* pop-up window, highlight the module and the architecture bound to the design entity.

Click on the *Load* button. After loading the design entity, a *VSIM* pop-up window appears. From the *Run* pull-down menu, select *Time High* to run simulations for the maximum time specified in the test bench.

The simulation results can be viewed as waveforms or ascii text. From the *View* pull-down menu in the *VSIM* window, select *Signals*.

To view the waveforms of the signals in the design, select the *Wave* pull-down menu in the *VSIM Signals* pop-up window, and then select *Signals in Design*. Next, select the *View* pull-down menu in the *VSIM* window, and select *Waves*. The *VSIM Wave* window shows the waveforms. To save the configuration, select the *File* pull-down menu and then select *Save Configuration*. By default, the configuration is saved into the *wave.doc* file which is placed in the current directory. To print the waveforms, select the *File* pull-down menu and then select *Write Postscript*.

To view the simulation of the design as ascii text, select the *List* pull-down menu in the *VSIM Signals* pop-up window, and then select *Signals in Design*. Next, select the *View* pull-down menu in the *VSIM* window, and select *List*. The *VSIM List* window shows the ascii text. To save to a file, select the *File* pull-down menu and then select *Write to File*. By default, the list is saved into the *vsim.ls* file which is placed in the current directory.

The VHDL model can be back annotated into SDF:

```
vcom -sdf<corner> <name>.sdf
```

where *<name>.sdf* is the output SDF file and *<corner>* is *min*, *typ*, or *max*.

### 4.3.3 Using the Synopsys Model to Generate SDF

After generating the Synopsys model, you can generate SDF using these steps.

Invoke Synopsys:

```
dc_shell
```

Once inside `dc_shell`, execute the following Synopsys commands:

```
read_lib <name>.lib
write_lib <userlib>
link_library=<userlib>.db
target_library=<userlib>.db
read -f verilog <file>.v
write_timing -context verilog -f sdf-v2.1 -o <out>
```

where `<userlib>` is the name of your Synopsys library, `<name>.lib` is the Synopsys model, `<file>.v` is the top-level netlist, and `<out>` is the output file name.

If you are using multiple Synopsys libraries, you can read the `.lib` files into the Synopsys model and include each file in your `link_library` and `target_library` paths. You can write a script or manually remove the `lu_table_template` descriptions, `power_lut_template` descriptions, `type` descriptions, and `cell()` block for each `.lib` file. You should include all the descriptions and block into a single `.lib` file, and append the global library information found at the beginning of the generated `.lib` files to the beginning of your consolidated `.lib` file. Attach a closing bracket `"}"` to the end of the new `.lib` file.

The typical and slow Synopsys models are generated with maximum delays, and the fast model is generated with minimum delays. ARM recommends that critical path, setup, and hold analysis be performed for all corners.

In the Synopsys model, data from SPICE characterization is used for setup and hold times; no negative setup is used in the . In SDF, if the hold time is a negative number, it is set to zero.

——— **Note** ———

ARM recommends that you avoid using implicit netlisting because it is an error prone methodology.

---

### 4.3.4 Using the Pearl Model to Generate SDF

After generating the Pearl model, you can generate SDF using these steps.

Invoke the Pearl Timing Analysis tool:

```
pearlcell
```

Once in Pearl, execute the following commands:

```
ReadTechnology <your_technology_file>
Read TLF <name>.tlf
Read Verilog <your_verilog_file>
TopLevelCell <your_toplevel>
InputSlew <toplevel_input_pin_name>
           <value> <value> <value> <value> <value>
.
.
.
SetNodeCapacitance <toplevel_output_pin_name> + <value>
.
.
.
WriteSDFDelays -delimiter . -version2.1 -ns -precision 3
               "<your_sdf_file>"
quit
```

The typical and slow TLF models are generated with maximum delays, and the fast model is generated with minimum delays. ARM recommends that critical path, setup, and hold analysis be performed for all corners.

### 4.3.5 Using the PrimeTime Model to Generate SDF

After generating a Synopsys .db file, you can generate SDF using these steps.

Invoke PrimeTime:

```
pt_shell
```

Once inside pt\_shell, execute the following PrimeTime commands:

```
read_db <name>.db
set link_path=<userlib>.db
read_verliog <file>.v
write_sdf -version 2.1 -o <out>
```

where <userlib> is the name of your Synopsys library, <name>.lib is the Synopsys model, <file>.v is the top-level netlist, and <out> is the output file name.

### 4.3.6 Loading the VCLEF Description into Silicon Ensemble

After generating VCLEF, load it into Silicon Ensemble using these steps.

Invoke Silicon Ensemble.

In the Command window, type:

```
INPUT LEF FILENAME "<path>/<name>.vclef";
```

where <path> is the path to the VCLEF, and <name> is the SRAM instance name.

This creates the Silicon Ensemble database necessary for routing.

### 4.3.7 Using Apollo with ARM's Artisan Generators

In order to use the generator with the Avant! tool suite you need to import the SRAM into the Milkyway database. Before you can place or route a design, you need to create a FRAM view for any memories in the design. This can be done by importing the VCLEF and running "Blockage, Pin & Via" (BPV) to create the FRAM. If you wish to stream out a full GDSII database you also need to import the GDSII into the Milkyway database to create a CEL view.

ARM does not recommend trying to create a FRAM view directly from the GDSII. You must use the following sequence when importing the SRAM into the Milkyway database to avoid creating a FRAM view from the GDSII.

1. Generate the VCLEF and GDSII
2. Import the VCLEF into Milkyway
3. Run BPV to generate a FRAM view
4. Import the GDSII into Milkyway

It is important to run BPV before importing the GDSII.

Details on creating and importing VCLEF and GDSII are provided in the following sections. Consult the Avant! documentation for more details on the commands mentioned below. In particular, you should be familiar with Chapter Four of the Avant! *Milkyway Data Preparation User Guide*.

#### 4.3.7.1 Loading the VCLEF Description into the Milkyway Database

Create a LEF or VCLEF file from the generator.

Example:

```
<install_dir>/aci/<executable>/bin/<executable> vclef-fp  
-words 256 -bits 16 -mux 8 -instname <name>
```

This command creates a file called <name>.vclef. The instance name for this file is <name>.

Edit <name>.vclef and comment the line that contains "VERSION 5.1". Commented lines in LEF begin with the pound sign (#). Create a file named lef2Arcs.map using the same syntax as the gds2Arcs.map file.



Example:

```
gdsMacroCell  
<name>
```

Start Milkyway. On the command line for Milkyway, use the Avant! undocumented LEF reader `auLefIn`.

A form displays. Fill this form with the appropriate entries for library name, `.lef` file name, and `lef2Arcs.map` file, then click on OK to create the CEL view.

Now run Blockage, Pin & View (BPV) to create the FRAM view.

#### 4.3.7.2 Loading the GDSII Layout into the Milkyway Database

Create a GDSII file from the generator.

Example:

```
<install-dir>/<executable>/bin/<executable> gds2  
-words 256 -bits 16 -mux 8 -instname <name>
```

This command creates a file called `<name>.gds2`. The instance name for this file is `<name>`.

Create a file named `gds2Arcs.map`.

Example:

```
gdsMacroCell  
<name>
```

Start Milkyway. The VCLEF should have already been read into the library created in the previous section. Read in the GDSII. This process overwrites the CEL view with the actual layout. From the menus, select `Cell Library > Stream in...` You need to update the tech file to support all memory gds2 layers. Fill in the `Stream File Name` and the `Library Name`. Depending on your flow, the other fields may or may not need to be updated.

### 4.3.8 Loading the GDSII Layout into a DFII Library

After generating the GDSII layout, you can load it into a DFII library using these steps.

Invoke DFII.

From the DFII CIW, select the *File* pull-down menu, then select the *Import* pull-down sub menu and click on *Stream*.

In the *Stream* pop-up window, type the name of the GDSII layout file in the *Input File* field, and type the instance name in the *Top Cell Name* field. Type the name of the library in the *Library Name* field. Click on *User-Defined Data*.

In the *User-Defined Data* pop-up window, type the path to the metal layer table file in the *Layer Map Table* field, and type the path to the text font file in the *Text Font Table* field.

### 4.3.9 Using the LVS Netlist

After generating the LVS netlist, it may be used in conjunction with the GDSII file for verification.

Typically, you use a tool like Cadence Dracula, Mentor Graphics Calibre, or Avant! Hercules, to read a GDSII file and compare it with the LVS netlist. This test compares the layout and schematic to ensure there is no short- or open-circuit in the layout.

The LVS netlist is then added onto the chip level LVS netlist, and the same test is run when the chip is fully assembled. This process ensures that the chip is correctly assembled, (that is, there is no short- or open-circuit caused by a place-and-route or other tool).

### 4.3.10 Using Hierarchical LVS

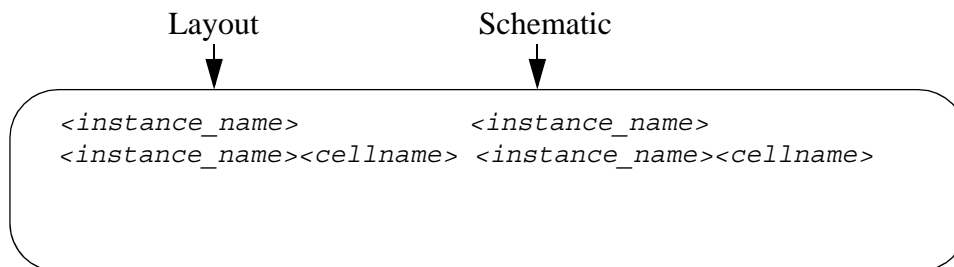
LVS (Layout vs. Schematic) performs an equivalence check between two different representations of a design. In this case, the physical (GDSII) and schematic (SPICE netlist) are compared to each other. The CALIBRE tool reports any discrepancies.

Executing LVS in a hierarchical mode is faster than using LVS in the flat mode. The blocks are checked only once, as opposed to multiple passes for various instantiations of the same block in the design.

To run calibre LVS on memory instances in full hierarchical mode, execute the following steps:

1. Invoke the generator GUI.
2. Generate the GDSII and LVS netlists to create the `<instance_name>.gds2` and `<instance_name>.cdl`.
3. Create the `.hcell` file.

Sample Hcell file format:



The left column corresponds to the layout instances and the right column corresponds to schematic instances where:

`<instance_name>` is the instance name you specified when the `.gds2` and `.cdl` views were created.

`<cellname>` is a hierarchical cell within the memory

4. Modify the rules file.

The rules file specifies the location and format of the following items:

```
LAYOUT PATH "<instname>.gds2"  
LAYOUT PRIMARY <instname>  
SOURCE PATH "<instname>.cdl"  
SOURCE PRIMARY <instname>  
LVS REPORT "<instname>.lvs"
```

5. Execute CALIBRE:

```
calibre -lvs -hier -spice <instname>.spc  
-hcell <instname>.hcell rulesfile
```

where the *.spc* file is output from calibre.

6. Examine the LVS report.

## **Troubleshooting Notes:**

More information about hierarchical LVS can be found in Mentor Graphics' *Physical Extraction and Verification Application Note #21: What to look for in the Calibre LVS transcript*.

Specifically, review the section on the command "LVS SHOW SEED PROMOTIONS YES"

This information is available online at Mentor Graphics' web site.

### **4.3.11 Using the Repair Verilog Model**

Currently, the Repair Verilog model is supported by 130nm and 180nm single-port, and 130nm dual-port SRAM, with Flex-Repair, generators.

The Repair RTL is available in two configurations. In one configuration type the fuse box is instantiated inside the top level module. This module contains an instantiation of the memory generator, the fuse box, and the repair RTL for the control box. In the second configuration type the top level module does not include an instantiated fuse box. You must instantiate and connect the fuse box manually if you are using this type of configuration. This may be a useful method if you are sharing fuse boxes.

The Repair Verilog model can be used for two purposes; synthesis and simulation. In a synthesis flow you can use the Repair Verilog with synthesis models such as .db or .lib, for the memory generator, fuse box, and standard cells. Use the Repair Verilog in simulations with Verilog models for the memory generator, fuse box generator, and standard cells.

#### **4.3.11.1 Repair Verilog Simulation**

You must link the Verilog models for the memory generator, fuse box, and standard cells with the Repair Verilog model for simulation. The default fuse box VHDL model has been created so that none of the fuses are blown. To test for the situation when the fuses are blown, the fuse box outputs have to be explicitly set to your defined values in simulation decks.

#### **4.3.11.2 Repair Verilog Synthesis**

You must link the .lib and .db models for the memory generator, fuse box, and standard cells with the Repair Verilog model. Set the constraints for the synthesis tool to meet your design needs and synthesize the design. You may create different repair RTL styles by rerunning the memory

generator, such as the multiplexer-based or 3-state control box architectures, to determine the best options for your design.

Typically, the constraints should be set to optimize the path from the data input of the RTL wrapper to the data input of the memory, so as to get the smallest setup time. You can provide similar optimization for the WEN inputs, if you are using the word write-mask option. The path from the memory output to the RTL wrapper output pin should also be optimized for the best performance, as it affects the access time of the memory generator.

A sample Synopsys synthesis script appears as follows:

```
/* Script for a 32 bit redundant memory with the D[32] as the
redundant bit. */
/* Set the technology libraries and search paths */
...

/* Read the top level design and set the "ralshr_top" as top design
*/
...
current_design ralshr_top

/* Set the output load */
set_load 0.050 all_outputs()

/* Optimize for best timing from redundant memory data inputs to
the redundant bit data input (DR) */
set_max_delay 0.0 -to I0/D* -from D*

/* Optimize for best timing from redundant bit data output (QR) to
the redundant memory data output*/
set_max_delay 0.0 -to Q* -from I0/Q*

/* Uniquify the instances for individual optimization */
uniquify

/* Ungroup and optimize the circuit to meet best timing */
set_ungroup ralshr_top
compile -ungroup_all -map_effort high

/* Do the reports for the design rule violations, area and timing.
*/
report_constraints -all_violators
report_cell
report_timing -to I0/D* -from D*
report_timing -to Q* -from I0/Q*

/* Save the optimized design */
```

### **4.3.12 Using the Repair VHDL Model**

Currently, the Repair VHDL model is supported by 130nm and 180nm single-port, and 130nm dual-port SRAM, with Flex-Repair, generators.

The Repair RTL is available in two configurations. In one configuration type the fuse box is instantiated inside the top level module. This module contains an instantiation of the memory generator, the fuse box, and the repair RTL for the control box. In the second configuration type the top level module does not include an instantiated fuse box. You must instantiate and connect the fuse box manually if you are using this type of configuration. This may be a useful method if you are sharing fuse boxes.

The Repair VHDL model can be used for two purposes; synthesis and simulation. In a synthesis flow you can use the Repair VHDL with synthesis models such as .db or .lib, for the memory generator, fuse box, and standard cells. Use the Repair VHDL in simulations with VHDL models for the memory generator, fuse box generator, and standard cells.

#### **4.3.12.1 Repair VHDL Simulation**

You must link the VHDL models for the memory generator, fuse box, and standard cells with the Repair VHDL model for simulation. The default fuse box VHDL model has been created so that none of the fuses are blown. To test for the situation when the fuses are blown, the fuse box outputs have to be explicitly set to your defined values in simulation decks.

#### **4.3.12.2 Repair VHDL Synthesis**

You must link the .lib and .db models for the memory generator, fuse box, and standard cells with the Repair VHDL model. Set the constraints for the synthesis tool to meet your design needs and synthesize the design. You may create different repair RTL styles by rerunning the memory generator, such as the multiplexer-based or 3-state control box architectures, to determine the best options for your design.

Typically, the constraints should be set to optimize the path from the data input of the RTL wrapper to the data input of the memory, so as to get the smallest setup time. You can provide similar optimization for the WEN inputs, if you are using the word write-mask option. The path from the memory output to the RTL wrapper output pin should also be optimized for the best performance, as it affects the access time of the memory generator.

A sample Synopsys synthesis script appears as follows:

```
/* Script for a 32 bit redundant memory with the D[32] as the
redundant bit. */
/* Set the technology libraries and search paths */
...

/* Read the top level design and set the "ralshr_top" as top design
*/
...
current_design ralshr_top

/* Set the output load */
set_load 0.050 all_outputs()

/* Optimize for best timing from redundant memory data inputs to
the redundant bit data input (DR) */
set_max_delay 0.0 -to I0/D* -from D*

/* Optimize for best timing from redundant bit data output (QR) to
the redundant memory data output*/
set_max_delay 0.0 -to Q* -from I0/Q*

/* Uniquify the instances for individual optimization */
uniquify

/* Ungroup and optimize the circuit to meet best timing */
set_ungroup ralshr_top
compile -ungroup_all -map_effort high

/* Do the reports for the design rule violations, area and timing.
*/
report_constraints -all_violators
report_cell
report_timing -to I0/D* -from D*
report_timing -to Q* -from I0/Q*

/* Save the optimized design */
```