90nm Process Advantage SRAM Generator User Guide

ARM®

June 2005, Revision 2005q3v1

ug_adv_90nm_sram_2005q3v1

ARM reserves the right to make changes to any products and services described herein, at any time without notice in order to make improvements in design, performance, or presentation and to provide the best possible products and services. Customers should obtain the latest specifications before referencing any information, product, or service described herein, except as expressly agreed in writing by an officer of ARM.

ARM does not assume any responsibility or liability arising out of the application or use of any products or services described herein, except as expressly agreed to in writing by an officer of ARM; nor does the purchase, lease, or use of a product or service from ARM convey a license under any patent rights, copy rights, trademark rights, or any other of the intellectual property rights of ARM or of third parties.

ARM Physical IP, Inc. 141 Caspian Court, Sunnyvale, CA 94089, USA

Table of Contents

Preface

Chapter 1

Chapter 2

Chapter 3

Preface

Revision History

The following table provides the revision history for this manual.

Part Number	Updates (to template)
ug_adv_90nm_2004q3v0	• Initial Release
ug_adv_90nm_2004q3v1	• Update selected CLI command entries; delete TetraMAX procedures from Chapter 4
ug_adv_90nm_2005q1v1	• Update redundancy text
ug_adv_90nm_sram_2005q2v1	• Added Advanced Options GUI, DC leakage note, scientific notation
ug_adv_90nm_sram_2005q2v2	• Updated dpccm and asvm text and behavior table
ug_adv_90nm_sram_2005q3v1	• Power ring text update

Customer Support

For general questions related to ARM’s Physical IP product availability and their licensing requirements, customers can go to www.artisan.com. Log in and click on the “New Business Request” link to enter a request.

Customers with active Support contracts can obtain support for ARM’s Physical IP products by going to www.artisan.com and clicking on “New Technical Request.” Support contract options are available for review at www.artisan.com/support/programs.html.

You may also contact ARM Physical IP by telephone or email, using the following information:

■ 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.
(Pointing Hand) ☞	Important information or explanation.
`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 (e.g., `-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

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 90nmAdvantage SRAM generators to create memory instances based on a variety of parameters, and the corresponding EDA tool support views. This manual provides information about single- and dual-port, high speed/high density SRAM generators.

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

Refer to the following sections for more detailed information about this generator.

- This chapter provides basic information about Advantage SRAM generators 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,” - Provides information about EDA tools support and provides instructions on generating specific views.

1.2 Advantage SRAM Generator Features

Advantage memory generators include the following features:

- Optimized for High Speed/High Density
- Aspect Ratio Control for Efficient Floor Planning
- Memory Operation and Retention at Low Frequency (Down to 0 MHz)
- Low Active Power and Leakage-Only Standby Power
- Timing and Power Models for Industry-Leading Design Tools
- Configurable Word-Write Mask Option
- Extra Margin Adjustment™ (EMA) Option
- Integrated BIST Mux Option
- Soft Error Repair (SER) Option
- Flex-Repair™ Redundancy Option (available with a separate Fuse Box generator)
- ArtiGrid™ Over-the-Cell Power Routing Option
- Maximum Static Power Dissipation Corner

A standard set of EDA support views can be generated from Advantage SRAM generators. These views are verified with the tools defined in the applicable EDA package; EDA tools and version specifics are detailed in the README file. Refer to Chapter 4, “Generator Views,” for details about tools and using the views. Optional support is available and can be added to most existing generators without installing a completely new generator.

Standard and optional tool support is listed below.

Standard Support

Verilog
VHDL

Flex-Repair & SER Verilog
Flex-Repair & SER VHDL

Synopsys Liberty

VCLEF Footprint

GDSII Layout
LVS netlist

PostScript Datasheet
ASCII Datatable

Optional Support

LogicVision IC Memory BIST
Mentor MBISTArchitect
FastScan

EDAPlus
TetraMAX
Hercules

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
- LINUX Redhat v7.2

☞ 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
```

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.

☞ 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.

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**
Advantage Single-Port SRAM	sram-sp-adv-v40	sram_sp_adv
Advantage Dual-Port SRAM	sram-dp-adv-v40	sram_dp_adv

2
Using the Generator

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 for use with industry standard 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*>.

Refer to the table in "Directory Structure and Executables" on page 1-8 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 instance with Verilog view, mux = 8, words = 256, type:

```
sram_sp_adv verilog -mux 8 -words 256
```

The table in "Directory Structure and Executables" on page 1-8 provides a list of standard generator executables. Refer to "Generating Views from the Command Line" on page 2-16 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. Refer to "Generating Views from the GUI" on page 2-13, "Generating Views from the Command Line" on page 2-16, and "Generator Options" on page 2-22 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

Standard Support	
View	**Output Files**
Verilog model	*<instance_name>*.v
VHDL model	*<instance_name>*.vhd
Flex-Repair & SER Verilog[1]	*<instance_name>*_rtl.v
Flex-Repair & SER VHDL[1]	*<instance_name>*_rtl.vhd
Synopsys (Liberty) for each corner[2, 3, 4]	*<instance_name>*_*<corner>*_syn.lib
VCLEF footprint	*<instance_name>*.vclef *<instance_name>*_ant.lef *<instance_name>*_ant.clf
GDSII layout file	*<instance_name>*.gds2
LVS netlist	*<instance_name>*.cdl
Optional Support	
View	**Output Files**
LogicVision IC Memory BIST	*<instance_name>*.memlib
Mentor MBISTArchitect[5]	*<instance_name>*.mbist
Mentor FastScan	*<instance_name>*.fastscan
Synopsys (Liberty) TetraMAX[6]	*<instance_name>*.tv
Hercules[6, 7]	N/A
Datasheets	
View	**Output Files**
PostScript datasheet	<*instance_name*>.ps
ASCII datatable	*<instance_name>*.dat

(1) The word-write mask option should be set to ‘off’ when SER is selected as either ‘1db1bc’ or ‘2bd1bc.’

(2) 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 only four corners at a time. The characterization corner name (i.e., slow, fast, fast@-40C, fast@125C, typical) is inserted into the output filename (e.g., sram_sp_adv_<corner>_syn.lib).

(3) 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 applicable corners.

(4) Synopsys models are generated with maximum alternate current (AC) values for each supported corner. 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. The static power corner models both AC and static power under this condition. 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.

(5) The word-write mask option is not supported in .mbist models.

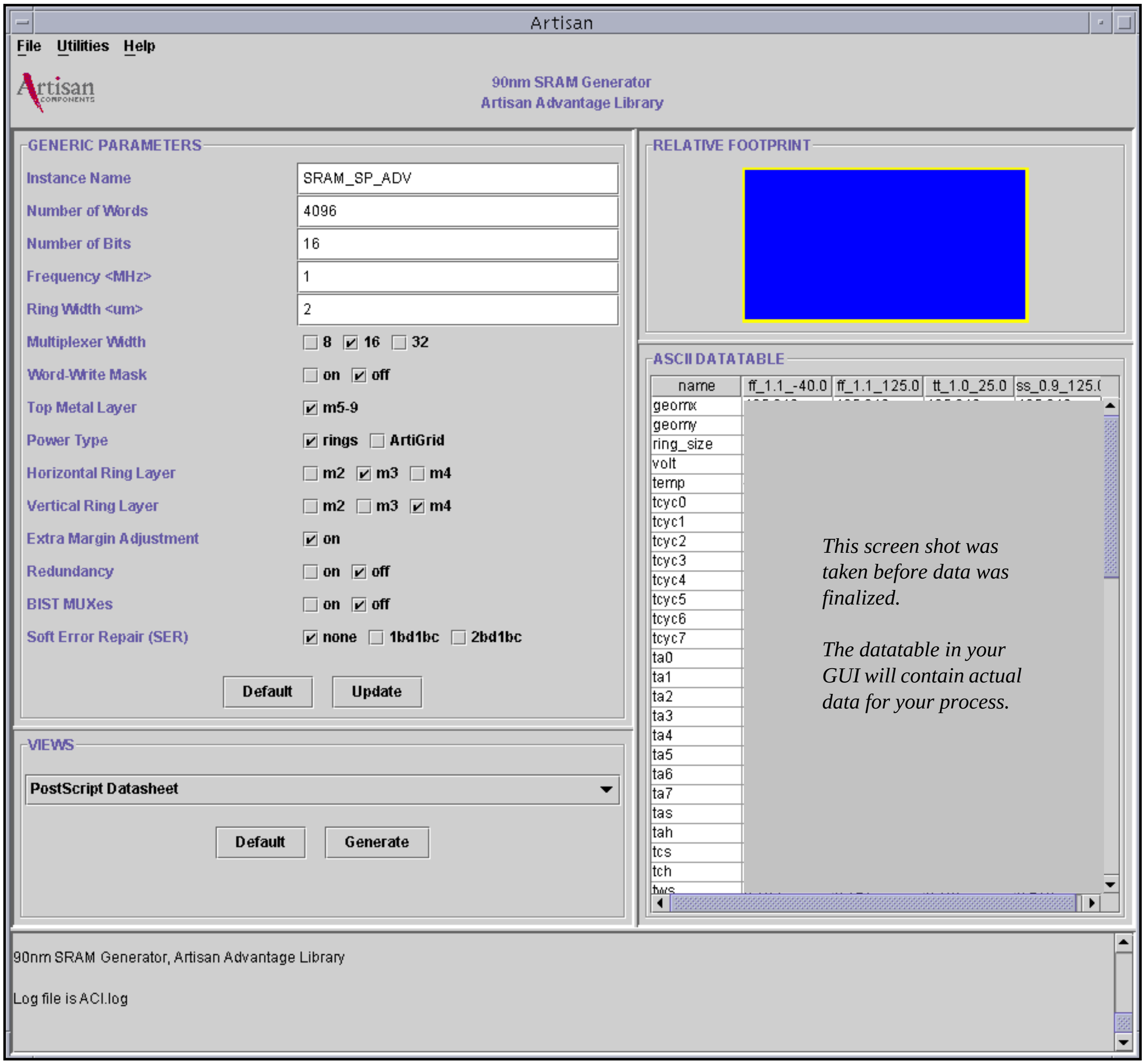
(6) This optional support is available for free to ARM’s Artisan Access (Free) Library Program licensees under ARM’s Artisan EDAPlus programs with EDA partners.

(7) Hercules support means LVS/DRE verification with Hercules if the rule decks are available from the foundry.

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 this sample. 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



90nm SRAM Generator
Artisan Advantage Library

GENERIC PARAMETERS

Instance Name	SRAM_SP_ADV
Number of Words	4096
Number of Bits	16
Frequency <MHz>	1
Ring Width <um>	2
Multiplexer Width	☐ 8 ☑ 16 ☐ 32
Word-Write Mask	☐ on ☑ off
Top Metal Layer	☑ m5-9
Power Type	☑ rings ☐ ArtiGrid
Horizontal Ring Layer	☐ m2 ☑ m3 ☐ m4
Vertical Ring Layer	☐ m2 ☐ m3 ☑ m4
Extra Margin Adjustment	☑ on
Redundancy	☐ on ☑ off
BIST MUXes	☐ on ☑ off
Soft Error Repair (SER)	☑ none ☐ 1bd1bc ☐ 2bd1bc

PostScript Datasheet

ASCII DATATABLE

name	ff_1.1_-40.0	ff_1.1_125.0	tt_1.0_25.0	ss_0.9_125.(
geomx				
geomy				
ring_size				
volt				
temp				
tcyc0				
tcyc1				
tcyc2				
tcyc3				
tcyc4				
tcyc5				
tcyc6				
tcyc7				
ta0				
ta1				
ta2				
ta3				
ta4				
ta5				
ta6				
ta7				
tas				
tah				
tcs				
tch				
tws				

This screen shot was taken before data was finalized.

The datatable in your GUI will contain actual data for your process.

```
90nm SRAM Generator, Artisan Advantage Library

Log file is ACI.log
```

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. Refer to Figure 2-1 "Example: SRAM Generator GUI" 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. Refer to Chapter 3 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 also 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 "Views and Output Files" on page 2-5. For detailed information about using these views refer to Chapter 4, "Generator Views".

You can generate multiple views at one time. Refer to "Generating Multiple Views" on page 2-14. You can also 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. Refer to Figure 2-1 "Example: SRAM Generator GUI" 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 darker color. The power ring is shown in lighter color.

2.3.4 ASCII Datatable Pane

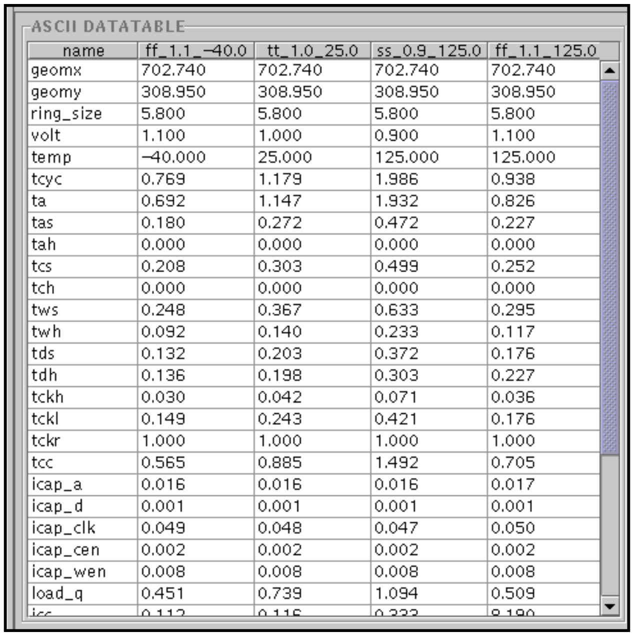
When you invoke the GUI, values for the default instance are displayed in the *ASCII Datatable* pane. When you change the generic values in the GUI, and click on the *Update* button in the *Generic Parameters* pane, the ASCII datatable automatically updates to reflect the new values.

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 and 2-3 show sample ASCII datatable panes. The corners and parameters shown may not be the same as those in your generator GUI. Information about minor deviations to the generators can be found in the README text file that is enclosed with your generator or in an addendum attached to this manual.

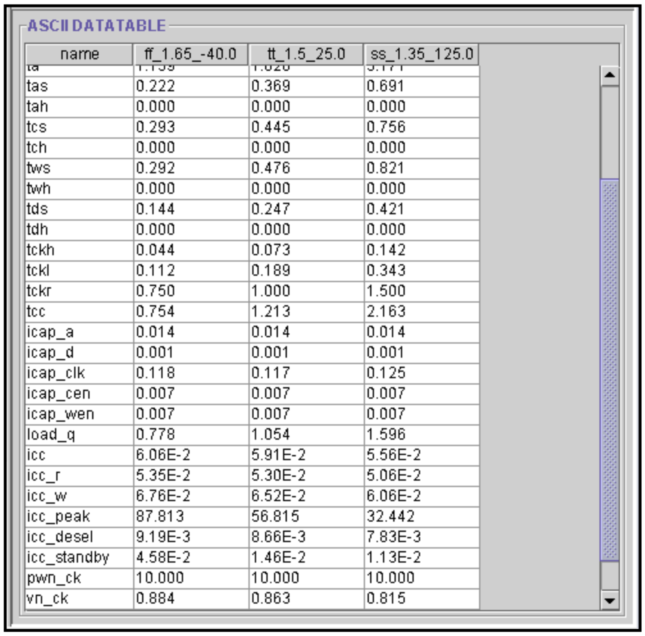
In the ASCII datable and postscript data sheets, non-power values are displayed in decimal format. Current/power values in datatables and datasheets may be shown in scientific notation or in decimal format.

Figure 2-2. Example: ASCII Datatable Pane



name	ff_1.1_–40.0	tt_1.0_25.0	ss_0.9_125.0	ff_1.1_125.0
geomx	702.740	702.740	702.740	702.740
geomy	308.950	308.950	308.950	308.950
ring_size	5.800	5.800	5.800	5.800
volt	1.100	1.000	0.900	1.100
temp	–40.000	25.000	125.000	125.000
tcyc	0.769	1.179	1.986	0.938
ta	0.692	1.147	1.932	0.826
tas	0.180	0.272	0.472	0.227
tah	0.000	0.000	0.000	0.000
tcs	0.208	0.303	0.499	0.252
tch	0.000	0.000	0.000	0.000
tws	0.248	0.367	0.633	0.295
twh	0.092	0.140	0.233	0.117
tds	0.132	0.203	0.372	0.176
tdh	0.136	0.198	0.303	0.227
tckh	0.030	0.042	0.071	0.036
tckl	0.149	0.243	0.421	0.176
tckr	1.000	1.000	1.000	1.000
tcc	0.565	0.885	1.492	0.705
icap_a	0.016	0.016	0.016	0.017
icap_d	0.001	0.001	0.001	0.001
icap_clk	0.049	0.048	0.047	0.050
icap_cen	0.002	0.002	0.002	0.002
icap_wen	0.008	0.008	0.008	0.008
load_q	0.451	0.739	1.094	0.509
icc	0.112	0.116	0.333	8.190

Figure 2-3. Example: ASCII Datatable Pane with Scientific Notation



name	ff_1.65_-40.0	tt_1.5_25.0	ss_1.35_125.0
ta	1.159	1.626	3.171
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
tdh	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 3. 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 Datatable Units

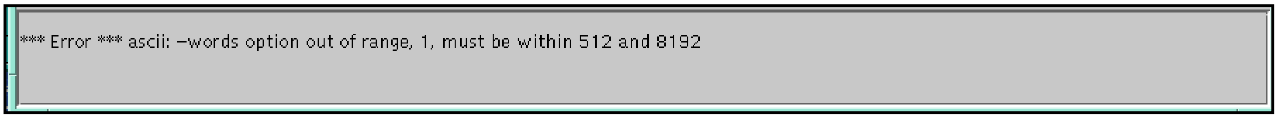
Parameters	Units
Geometry	Microns
Current-consumption	Milliamperes
Timing	Nanoseconds

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 2-4. The valid range, 512 to 8192 in this case, is also provided.

Figure 2-4. Example: Message Pane



```
*** Error *** ascii: -words option out of range, 1, must be within 512 and 8192
```

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.

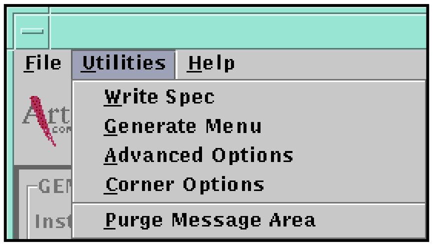
2.3.6 File Menu (Exiting the GUI)

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. Figure 2-5 shows a sample Utilities pull-down menu.

Figure 2-5. Example: Utilities Pull-Down Menu



Refer to "Using Specification Files" on page 2-19, "Generating Multiple Views" on page 2-14, "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.

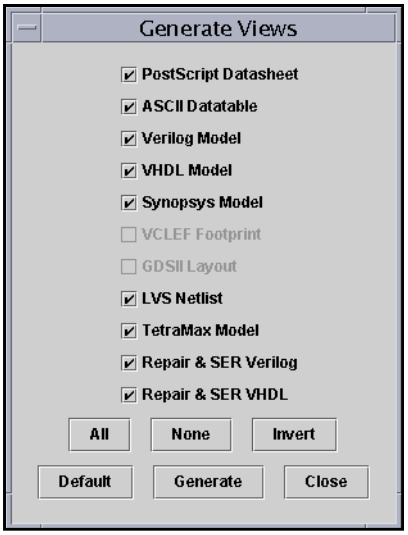
For instance, if you want to create datasheets for multiple instances, you can start by changing the generic parameters shown in the Generic Parameters section or in the Advanced Options pull-down menu of the GUI. Click on Update in the GUI. Select “Postscript Datasheet” from the Views pull-down menu 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 Figure 2-5 "Example: Utilities Pull-Down Menu" on page 2-12. Then select *Generate Menu*, such as the sample shown in Figure 2-6 "Example: Generate Menu." The *Generate Menu* window shows a list of views that you can generate.

Figure 2-6. Example: Generate Menu



☑ PostScript Datasheet
☑ ASCII Datatable
☑ Verilog Model
☑ VHDL Model
☑ Synopsys Model
□ VCLEF Footprint
□ GDSII Layout
☑ LVS Netlist
☑ TetraMax Model
☑ Repair & SER Verilog
☑ Repair & SER VHDL

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.

☞ You can also refer to "Advanced View-Specific Options" on page 2-33 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. Refer to 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

<table>
<tr><th colspan="2"><i>Standard Support</i></th></tr>
<tr><th>View</th><th>View-Command</th></tr>
<tr><td>Verilog model</td><td>verilog</td></tr>
<tr><td>VHDL model</td><td>vhdl</td></tr>
<tr><td>Flex-Repair and SER Verilog</td><td>verilog_rtl</td></tr>
<tr><td>Flex-Repair and SER VHDL</td><td>vhdl_rtl</td></tr>
<tr><td>Synopsys (Liberty)</td><td>synopsys</td></tr>
<tr><td>VCLEF footprint</td><td>vclef-fp</td></tr>
<tr><td>GDSII layout file</td><td>gds2</td></tr>
<tr><td>LVS netlist</td><td>lvs</td></tr>
<tr><th colspan="2"><i>Optional Support</i></th></tr>
<tr><th>View</th><th>View-Command</th></tr>
<tr><td>LogicVision IC Memory BIST</td><td>logicvision</td></tr>
<tr><td>Mentor MBISTArchitect</td><td>mbist</td></tr>
<tr><td>Mentor FastScan</td><td>fastscan</td></tr>
<tr><td>Synopsys TetraMAX</td><td>tmax</td></tr>
<tr><th colspan="2"><i>Datasheets</i></th></tr>
<tr><th>View</th><th>View-Command</th></tr>
<tr><td>PostScript datasheet</td><td>postscript</td></tr>
<tr><td>ASCII datatable</td><td>ascii</td></tr>
</table>

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. Refer to "Running the Generator from the Command Line" on page 2-4 for instructions. Refer to "Command-Line Syntax" on page 2-22 for details about writing commands to run the generator correctly.

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

For example:

```
    % sram_sp_adv 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:

```
    % sram_sp_adv vclef-fp gds2 synopsys -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 (Liberty) view.

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:

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

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

Commands (in sequence)	**Example**
`%<executable>`	`%sram_sp_adv`
`<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`

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 2-7 is an example of a simple specification file that includes a few options.

Figure 2-7. 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. Refer to the "Generator Options" section on page 2-22.

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 2-8 "Example: Parameter Information." 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 2-8. Example: Parameter Information

```
* name:       xxx Generator
*             Artisan xxnm 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 refer to the parameters tables in Chapter 3 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. Refer to "Directory Structure and Executables" on page 1-8, 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" section on page 2-19.

`-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. Refer to "View Commands" section on page 2-16 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 “ascii `-help`" to view the help information related to the ASCII datatable, such as generator parameters.

`-instname` *name*

The default instance name is the same as the executable name, in capital letters. For instance, if the executable is sram_sp_adv, the default instance name is SRAM_SP_ADV. Refer to "Directory Structure and Executables" on page 1-8, regarding the executable name for your generator.

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. Refer to Chapter 3 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. Refer to Chapter 3 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 µm, 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, refer to "Supply Connections to Power Rings" on page 3-61.

`-mux` *number*

The generator GUI shows the default mux value and any choices for this option. Refer to Chapter 3 for the mux values for specific generators.

`-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 companion 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.

`-top_layer` *m4|m5-9*

The generator GUI shows the default value for the top metal layer and any choices for this option. For more details, refer to "Top Metal Layer" on page 3-64.

`-power_type` *rings/ArtiGrid*

Power is supplied through a ring structure in standard generators (*rings* option) or through the ArtiGrid structure that is over-the-cell (OTC). The *rings* option is the default.

`-horiz` *string*

This parameter is active only when the *rings* option is selected as the `-power type` option.

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 but not the same layer (e.g., m1 and m2, but not m1 and m1). When a horizontal value is selected, the vertical ring layer is automatically set to a valid value.

`-vert` *string*

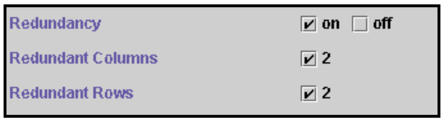
This parameter is active only when the *rings* option is selected as the `-power type`.

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 but not the same layer (e.g., m1 and m2, but not m1 and m1). When a vertical value is selected, the horizontal ring layer is automatically set to a valid value.

`-redundancy` *on|off*

When Flex-Repair or redundancy is turned on, the generator makes available two columns and two rows of integrated redundancy.
The extra rows and columns are shown in the GUI, as shown below, or from the command line.



Redundancy ☑ on □ off
Redundant Columns ☑ 2
Redundant Rows ☑ 2

The generator can also create associated RTL, which sets memory pins needed to access the redundant locations, depending on external inputs. For detailed instructions on how to use the Flex-Repair option, refer to the "90nm Advantage Memory Generator Test and Repair Features Application Note."

`-rcols` *2*

This option provides two redundant columns. See the "90nm Advantage Memory Generator Test and Repair Features Application Note" for more details.

`-rrows` *2*

This option provides two redundant rows. See the "90nm Advantage Memory Generator Test and Repair Features Application Note" for more details.

`-bmux` *on|off*

When this option is turned on, the generator adds MUXes at most input and output pins. The MUXes can be used by BIST engines to access the memory in test mode. The default value is off. For more details, see the "90nm Advantage Memory Generator Test and Repair Features Application Note."

`-ser` *none/1bd1bc/2bd1bc*

This option allows you to store error correcting codes by adding extra bits to each word of a memory. Along with the repair RTL created by the generator, this option prevents soft-errors due to external particles such as neutron or alpha particles.

By default, this option is disabled (set to “none”). You can set it to 1bd1bc to enable one bit error detection and one bit error correction. Setting it to 2bd1bc, provides two bits of error detection and one bit of error correction. This option is only available for memory instances when the word-write mask option is off (deselected).

2.7.3 Setting Advanced Options

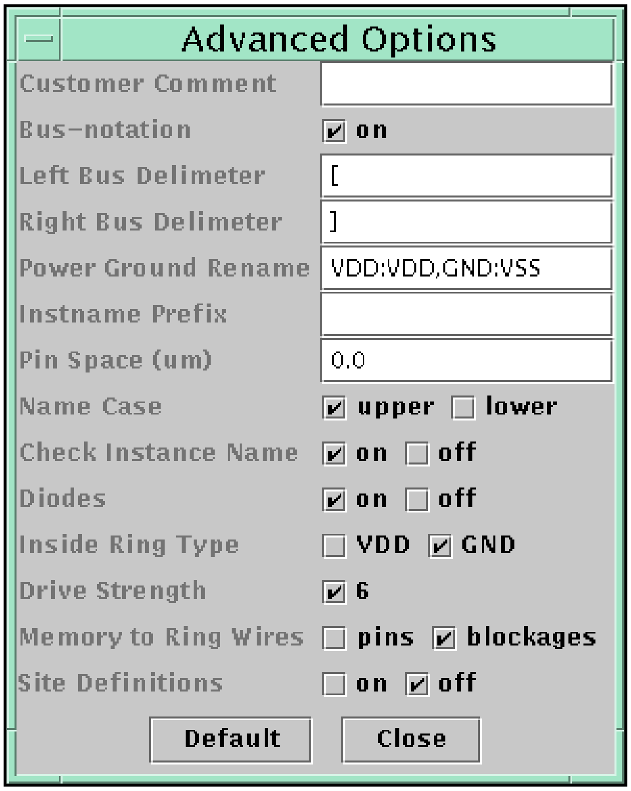
You can set advanced options from the generator GUI or by entering the options on the command line. This section demonstrates the methods for setting these options, including some options that apply only to specific views. Some of these options depend on the setting of the generic parameters in the main 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* window. This window displays the advanced options that you can specify, as shown in Figures 2-9 and 2-10. Enter a string or number in the fields, or select the check boxes as needed.

☞ The number of advanced options may vary and depends on your specific generator. or 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.

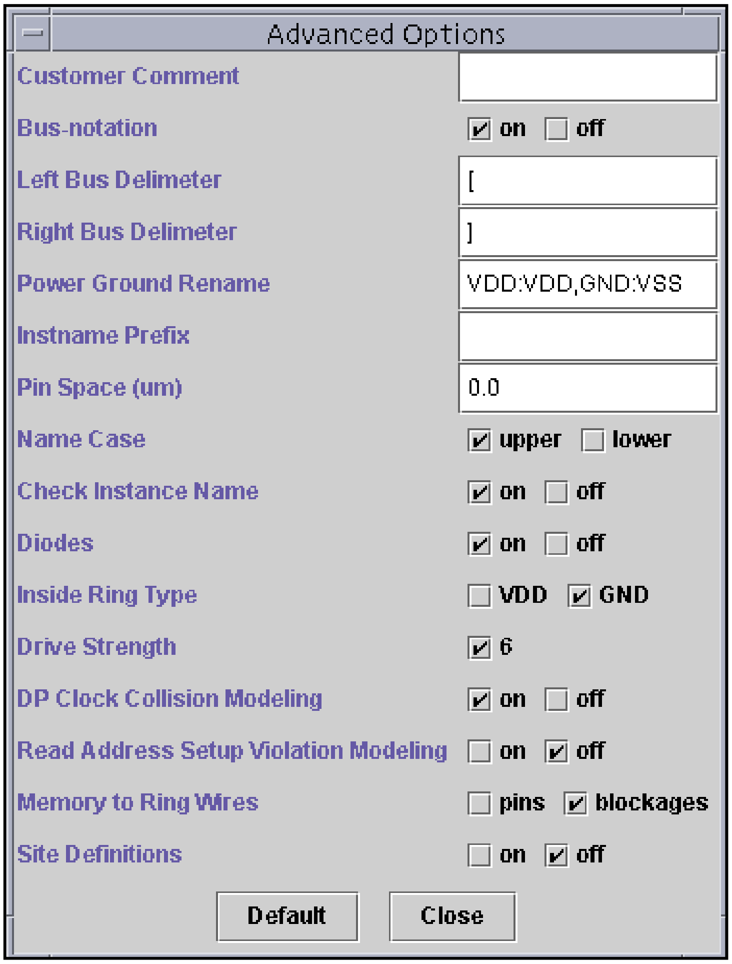
Figure 2-9. Example: Advanced Options Window



Advanced Options

Customer Comment	
Bus-notation	☑ on
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
Memory to Ring Wires	□ pins ☑ blockages
Site Definitions	□ on ☑ off

Figure 2-10. Example: Advanced Options Window with DPCCM and ASVM[1]



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

[1] 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. Refer to the "Command-Line Syntax" section on page 2-22 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 2-21.

`-bus_notation` *on*

By default, all bus pins are represented by bus notation in the interface of models (e.g., 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*

This parameter is active only when the *rings* option is selected as the `-power type` option.

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. Refer to the "Supply Connections to Power Rings" section on page 3-61 for more information about selecting the ring type.

`-se_immunity` *on|off* This option allows you to add deep NWELL implants and/or p-taps in the memory, providing better immunity to soft errors caused by external particles such as neutrons or alpha particles. The default value is off.

`-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.

`-drive` *number*

The Advanced Options menu shows the output drive strength.

`-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. Refer to 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. Refer to 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 the Synopsys model, the default library name is *userlib*. Use this option to specify your choice for `-libname`. This option applies only to the Synopsys model.

`-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*

This parameter is active only when the *rings* option is selected as the `-power type` option.

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.

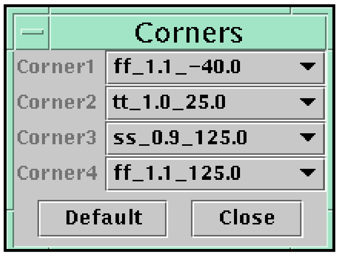
2.7.5 Selecting Characterization Corners

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, and so on. The number of corner options may vary per generator. Figure 2-11 shows a Corners dialog box with four fields.

Figure 2-11. Example: Corners Dialog Box



Corner1 ff_1.1_-40.0
Corner2 tt_1.0_25.0
Corner3 ss_0.9_125.0
Corner4 ff_1.1_125.0

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 1.1V, and the second value is -40^{o}C.

The Corners menu and the ASCII datatable in the GUI show the default corners. In addition, a Maximum Static Current Corner is available for use in electromigration calculations. You can select this additional corner from any of the Corner fields.

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 3-65.

2.7.5.2 Maximum Static Power Dissipation Corner

The maximum static power dissipation corner feature uses the maximum power value and minimum timing value.

2.7.5.3 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 3-65. 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

3.1 Overview

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

☞ Information about deviations to the Advantage 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

Generator	Product Name	Executable
Advantage Single-Port SRAM	sram-sp-adv-v40	sram_sp_adv
Advantage Dual-Port SRAM	sram-dp-adv-v40	sram_dp_adv

The following sections provide detailed information about your generator.

- "Synchronous Single-Port SRAM Architecture and Timing Specifications" on page 3-4
- "Synchronous Dual-Port SRAM Architecture and Timing Specifications" on page 3-30
- "SRAM Power Structure (sram_sp_adv, sram_dp_adv)" on page 3-58
- "SRAM Physical Characteristics (sram_sp_adv, sram_dp_adv)" on page 3-64

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 (sram_sp_adv)

Descriptions for the basic functionality of this generator are provided. In addition, descriptions of features you can use to enable the test and repair functions of your generator are provided. You can obtain further repair and test details and scenarios from the “90nm Advantage Memory Generator Test and Repair Features Application Note.”

3.2.1.1 Basic Functionality

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] is 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] is the result of a read operation.

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

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.

3.2.1.2 Test and Repair Functionality

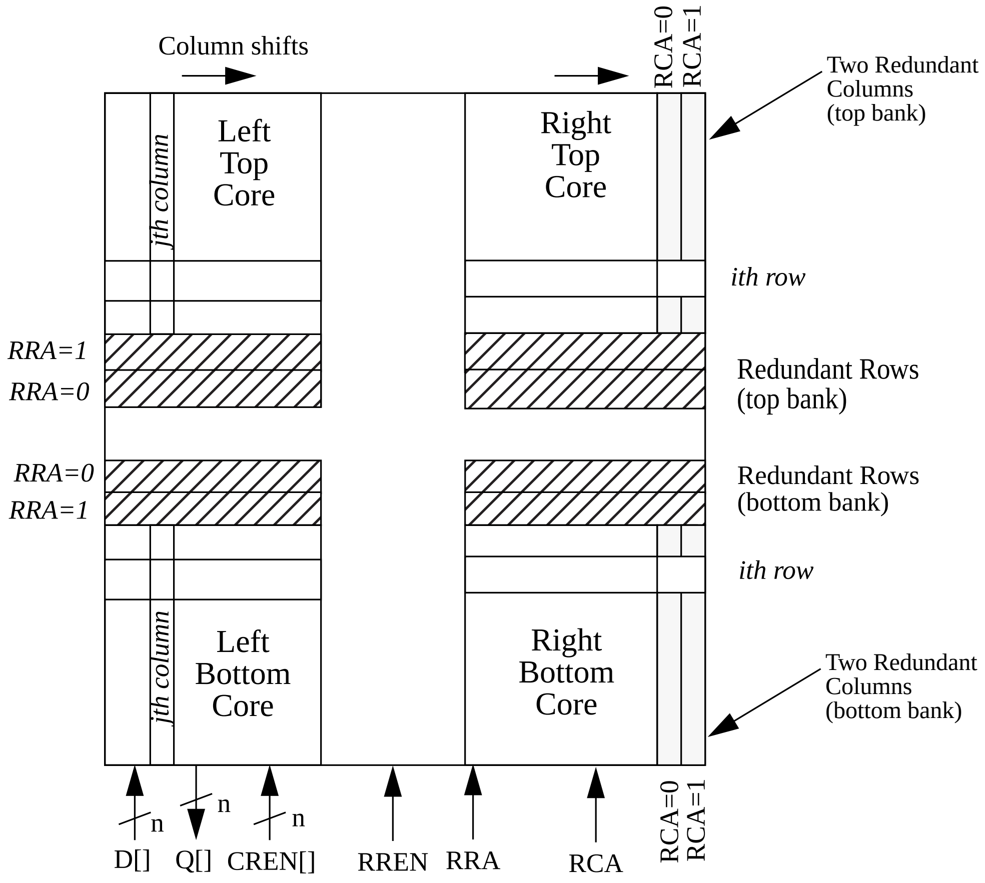
The SRAM generator includes features such as Flex-Repair/Redundancy, Soft Error Repair (SER), Built-In Self Test (BIST) MUXes, and Extra Margin Adjustment (EMA), as described in this section. Feature options are described in the “Generator Options” section on page 2-22. You can also obtain examples of how this functionality can be implemented in the “90nm Advantage Memory Generator Test and Repair Features Application Note.”

3.2.1.2.1 Flex-Repair/Redundancy

The Flex-Repair feature (also referred to as Redundancy) supports two rows and two columns of redundancy in the memory block. The memory generator can also create the RTL that is used to drive the memory signals to access the redundant locations. Figure 3-1 shows the architecture of a single-port SRAM with redundancy, with two rows and two columns.

☞ While only two redundant columns can be accessed, the width of the group of columns added for redundancy is set by the sense amp width.

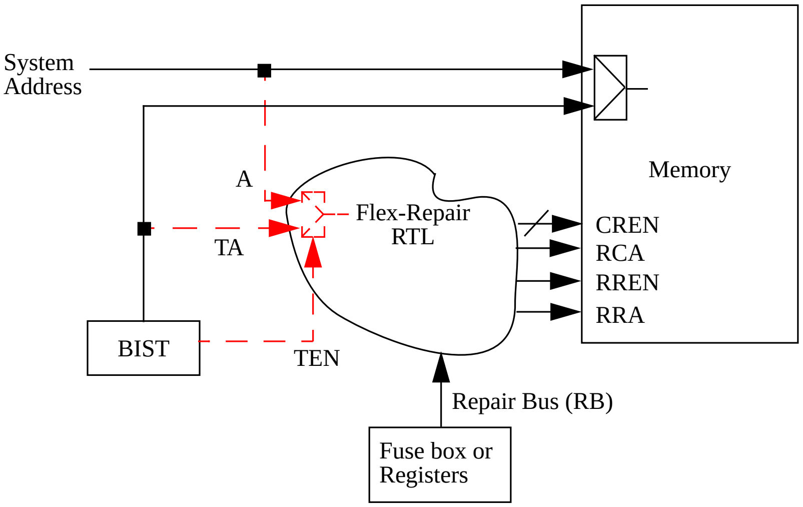
Figure 3-1. Architecture of a Typical Single-Port SRAM with Redundancy



RCA: present in all instances
RRA: present if at least one row of redundancy is enabled
CREN: present in all instances
RREN: present if at least one row of redundancy is enabled

Figure 3-2 shows the redundancy RTL created by the generator.

Figure 3-2. RTL for Redundancy Created by the Generator



Refer to the “90nm Advantage Memory Generator Test and Repair Features Application Note” for more detailed information about RTL pins.

3.2.1.2.2 Soft-Error Repair (SER)

SER automatically adds extra bits per word into the memory block and creates an accompanying RTL logic needed to implement Error Correction Codes (ECC) for SER. A composite RTL is created if both RTL and SER are enabled.

SER has two options:

- One bit error detection and one bit error correction (1bd1bc)
- Two bit error detection and one bit error correction (2bd1bc)

Refer to the “90nm Advantage Memory Generator Test and Repair Features Application Note” for more detailed information about SER pins.

3.2.1.2.3 Extra Margin Adjustment (EMA)

EMA is always enabled. The delays are selected by programming values 000 through 111 on pins EMA [2:0]. The default value is 000; validated or baseline value is 000. Incremental values greater than 000 provide progressively slower timing pulses.

This delay provides extra time for memory read and write operations by slowing down the memory access. The extra time may result in yield increase in the case of unexpected manufacturing instability with respect to memory circuitry, especially the bit cells.

EMA margin increments can be defined on a project-specific basis but a default methodology, based on relative delay increments (versus absolute), is needed.

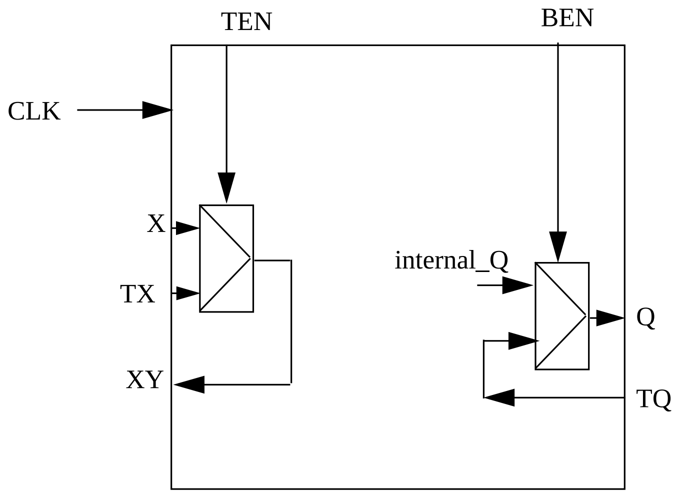
EMA[000] > BLM_DT
EMA[001] > BLM_DT + EMA_DT
EMA[010] > BLM_DT + 2*EMA_DT
EMA[011] > BLM_DT + 3*EMA_DT

3.2.1.2.4 Built-In Self Test (BIST) MUXes

BIST requires memory inputs that are driven by BIST logic and regular system signals. The SRAM generators have MUXes built into their timing critical memory pins with minimal effect on timing. The MUX outputs at input pins are available as separate pins that can be used to test the MUXes themselves.

Memory outputs should be easily controlled for better testability of logic in SRAMs. The SRAM generator has built-in bypass MUXes that can be driven by a scan chain in bypass mode. These MUXes have a very small effect on the memory access time and area. Figure 3-3 shows the BIST MUXes block diagram.

Figure 3-3. Single-Port SRAM BIST MUXes Block Diagram



X can be A, D, CEN and WEN.

3.2.2 Single-Port SRAM Pins (sram_sp_adv)

Figure 3-4 shows basic and optional test/repair pins for the single-port SRAM generator.

Figure 3-4. Single-Port SRAM Basic and Test/Repair Pins

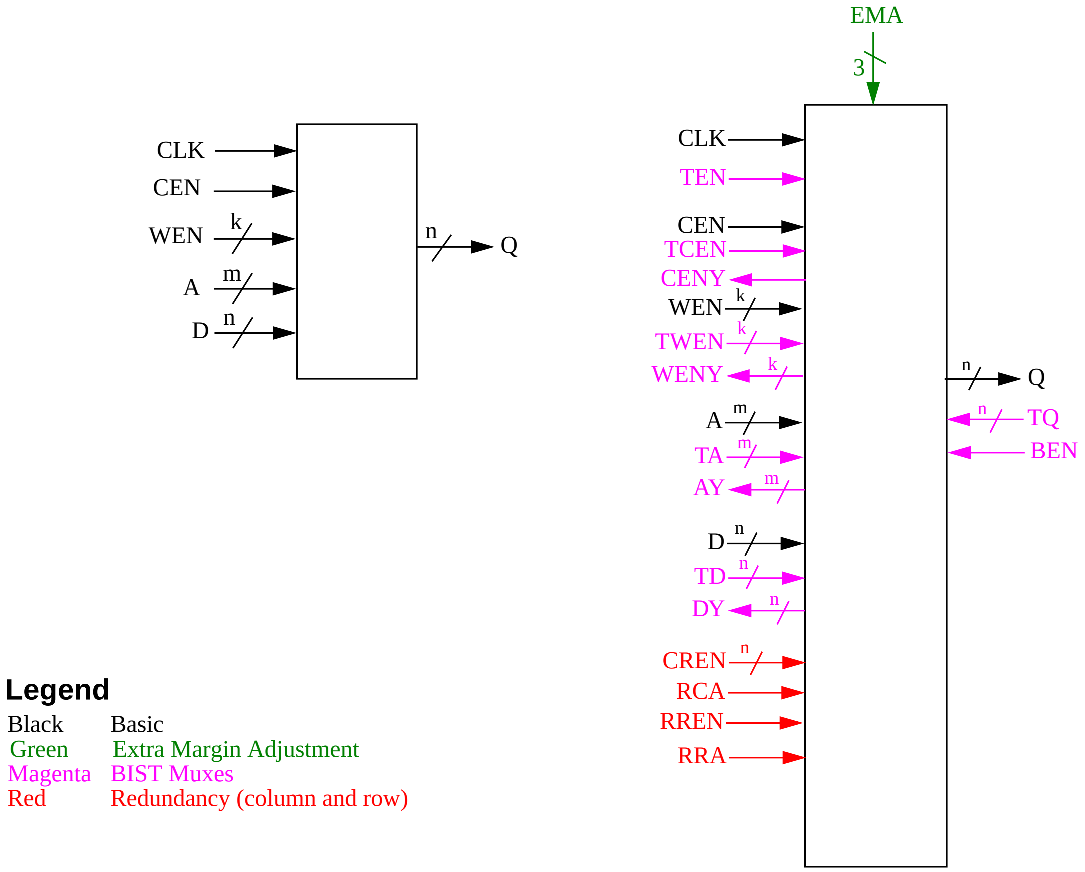


Table 3-2 provides the single-port SRAM generator pin descriptions.

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

Name	Type	Description
Basic Pins		
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)
Extra Margin Adjustment Pin		
EMA[2:0]	Input	Extra Margin Adjustment (EMA[0] = LSB)
Flex-Repair Pins		
CREN[n-1:0]	Input	Column Redundancy Enable, active low. Available when one or two column redundancy is enabled.
RCA	Input	Redundant Column Address. Available when two-column redundancy is enabled.
RREN	Input	Row Redundancy Enable, active low. Available when one or two row redundancy is enabled.
RRA	Input	Redundant Row Address, lowest row for 0. Available when two row redundancy is enabled.
BIST Mux Pins		
TEN	Input	Test Mode Enable, active low. 0=Test operation, 1=Normal operation
TA[m-1:0]	Input	Address Test Input (TA[0] = LSB)
AY[m-1:0]	Output	Address MUX output (AY[0] = LSB)
TD[n-1:0]	Input	Address Test inputs (TD[0] = LSB)
DY[n-1:0]	Output	Data MUX output (DY[0] = LSB)
TCEN	Input	Chip Enable Test Input, active low
CENY	Output	Chip Enable MUX output
TWEN [*]	Input	Write Enable Test inputs, active low. *If word-write mask is enabled, this becomes a bus.
WENY[*]	Output	Write Enable MUX output. *If word-write mask is enabled, this becomes a bus.
BEN	Input	Bypass Mode Enable, active low. 0=Bypass operation, 1=Normal operation
TQ[n-1:0]	Input	Bypass Q input, in write mode. If BEN=0, Q[n-1:0]=TQ[n-1:0].

3.2.3 Single-Port SRAM Logic Tables (sram_sp_adv)

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	**Word-write:** 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]. **Word-Write Mask:** 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]. For the rest of the data-bits that correspond to WEN[ ] high, the data is read from the memory location and driven to bus Q[ ].
L	H	SRAM data	Read	The memory location specified by the address bus, A[m-1:0], is read and driven onto the data output bus Q[n-1:0].

Table 3-4 describes the behavior of BIST MUXes at inputs to the memory.

Table 3-4. Single-Port SRAM BIST MUXes at Memory Input

TEN	Mode	Function
H	Regular Mode	In this mode, the basic pins, A[], D[], CEN, and WEN[] are used for the internal operations described in Table 3-3. These inputs are also available at the MUX outputs: AY[], DY[], CENY, and WENY[], respectively.
L	Test Mode	In this mode, the test pins, TA[], TD[], TCEN, and TWEN[] are used for the internal operations described in Table 3-3. These inputs are also available at the MUX outputs: AY[], DY[], CENY, and WENY[], respectively.

Table 3-5 describes the behavior of BIST MUXes at the output pins.

Table 3-5. Single-Port SRAM BIST MUXes at Memory Output

BEN	Mode	Q	Function
H	Regular Mode	Last Data Read or Written	In this mode, the data from the last read or write operation is available at Q.
L	Data Bypass	Data at bypass pin	Data at the bypass pin TQ is available at output Q. This operation is not clocked. All other operations described in Table 3-3, for basic features, work when BEN=0, with the exception that data does not appear at the output.

Redundancy is a user-selectable option and combinations are presented in Table 3-6. When the Repair RTL is generated, it is always generated for two rows and two columns redundancy. This RTL has a Repair Bus as an Input. This Repair Bus (RB) should be connected to the fuse box. The fuse box bits are used to program the RTL in order to generate signals that correct the defect in the memory.

You have the option to program this RB to get different redundancy configurations, for example, two rows and one column, if needed. The RB contains four control bits (RRE2, RRE1, CRE2, CRE1) that enable or disable a particular redundant row or column. By selective tie-off of these control bits to logic-0 you can disable a corresponding redundant row or column.

Corresponding to these control bits there are other signals generated when redundancy is enabled: (RRE2, FRA2), (RRE1, FRA1), (CRE2, FBA2, FCA2) and (CRE1, FBA1, FCA1). If a control bit is disabled by tie-off to logic-0, you should tie-off the rest of the bits in that group to logic-0 as well. After a given configuration is set, the synthesis is run with Boolean optimization and prunes the unneeded logic of the RTL. For example if the (RRE2, FRA2) bits are tied-off to logic-0 to get single row redundancy, then the result is RRA memory port being tied-off with logic-0. Use Table 3-6 to obtain the desired redundancy configuration; remaining pins should be connected as usual to the fuse box. This process is also described inside the Repair RTL.

Table 3-6. Single-Port Redundancy Configuration Scheme

To obtain these Redundancy Configurations	Perform these Tie-off to logic-0 operations
Two Column Redundancy, no Row Redundancy	RRE2, FRA2, RRE1, FRA1
Two Column Redundancy, one Row Redundancy	RRE2, FRA2
One Column Redundancy, no Row Redundancy	RRE2, FRA2, RRE1, FRA1, CRE2, FBA2, FCA2
One Column Redundancy, one Row Redundancy	RRE2, FRA2, CRE2, FBA2, FCA2
One Column Redundancy, Two Row Redundancy	CRE2, FBA2, FCA2
No Column Redundancy, one Row Redundancy	RRE2, FRA2, CRE2, FBA2, FCA2, CRE1, FBA1, FCA1
No Column Redundancy, Two Row Redundancy	CRE2, FBA2, FCA2, CRE1, FBA1, FCA1
No Column and No Row Redundancy	Turn-off the redundancy option and regenerate all models.
Two Column and Two Row Redundancy	No Need to do any tie-off’s.

Tables 3-7 and 3-8 show row (RREN) and column (CREN) redundancy logic tables, respectively.

Table 3-7. Single-Port SRAM Row Redundancy

RREN	Mode	Function
H	Read or write	Operations are performed as defined in Table 3-3, for basic memory operations. Main memory locations are accessed as specified by A[m-1:0].
L	Redundant row access	Instead of the locations in the regular rows, the redundant rows are accessed as specified by RRA and A[p-1:0]. For two redundant rows, RRA=0 accesses the lower row.

☞ For MUX values of 8, 16, and 32, the values for “p” are 3, 4, and 5, respectively.

Table 3-8. Single-Port SRAM Column Redundancy

CREN[i]	Mode	Function
H	Normal	The data is routed from pins (D[i], Q[i]) to the respective bit locations in the memory core.
L	Shift	The data is routed from pins (D[i], Q[i]) to the adjacent bits (i+1) in the memory core. The right-most bit is shifted to the redundant columns. For two redundant columns, the right-most bit is shifted to one of the two columns, depending on value of RCA.

3.2.4 Single-Port SRAM Parameters (sram_sp_adv)

The standard input and block parameters of a synchronous single-port SRAM are listed in Table 3-9. Refer to 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 3-9. Single-Port High Speed/Density 90nm SRAM (sram_sp_adv) Parameters

Input Parameters		
Parameter	**Ranges**	
number of words	mux = 8	256 to 4096, increment = $\text{mux} \bullet 4$
	mux = 16	512 to 8192, increment = $\text{mux} \bullet 4$
	mux = 32	1024 to 16384, increment = $\text{mux} \bullet 4$
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 $1/t_{cyc} \bullet 1000$, increment = 1	
word partition size[1]	1 to min (36, bits-1) increment = 1	
multiplexer width (µm)	8, 16, 32; default 16	
word-write mask	on, off; default off	
top chip metal layer support	m4 to top metal layer supported by design process	
top generator metal layer	m5 - 9	
power type	Rings, ArtiGrid; default rings	
horizontal ring layer	m2, m3, or m4; Ring power type must be selected; default m3	
vertical ring layer	m2, m3, or m4; Ring power type must be selected; default m4	
ring width (µm)	2 to 10,000; Ring power type must be selected	
extra margin adjustment	always on	
redundancy	on, off; default off	
redundant columns	2; redundancy must be selected	
redundant rows	2; redundancy must be selected	
BIST MUXes	on, off; default off	
soft error repair[2]	none, 1bd1bc, 2bd1bc; default none.	
Block Parameters		
Parameter	**Ranges**	
total memory bits	512 to 524,288, total bits = $\text{words} \bullet \text{bits}$	
rows in memory matrix	32 to 512, increment = 4, rows = $\text{words} / \text{mux}$	
columns in memory matrix	16 to 1024, increment = mux, columns = $\text{bits} \bullet \text{mux}$	
address lines	mux = 8	8 to 12
	mux = 16	9 to 13
	mux = 32	10 to 14
output drive strength, Q[ ]	Refer to the generator GUI or command line help instructions on page 2-3.	

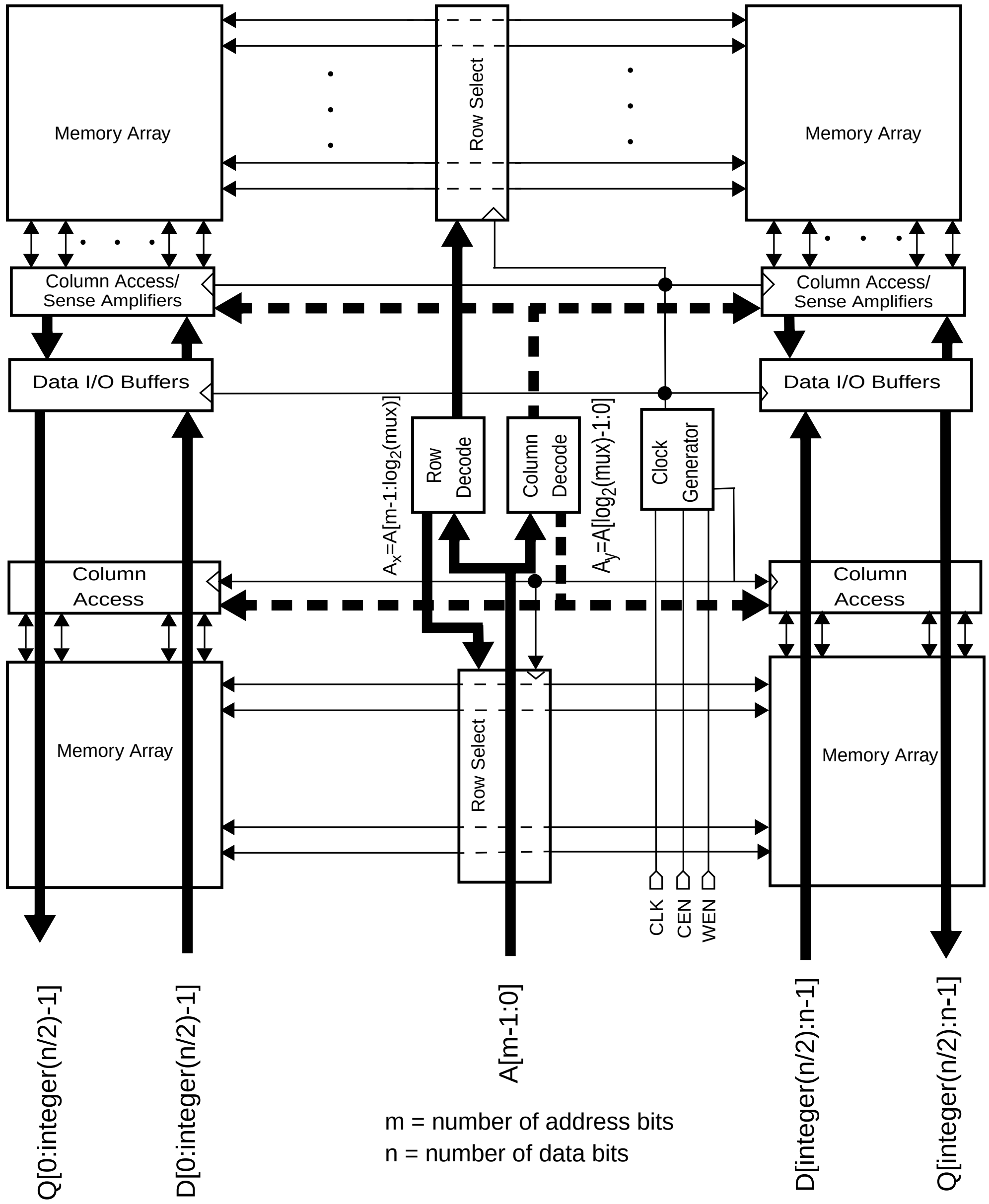
[1] The input pin capacitance for each pin of the write enable bus is proportional to the size of the word partition. For example, an instance with bits=32 and wp_size=24 will have two partitions, one with 24 bits and one with 8 bits. The write enable pin for the 24 bit partition will have a significantly larger input pin capacitance than the write enable pin for the 8 bit partition. When modelling 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.

[2] When the SER feature is enabled and any of the '1bd1bc' or '2bd1bc' options are selected, the actual memory generated will have more bits then specified in the GUI, these extra bits are used to store the error detection and correction code. For example if user specifies 120 bit memory in GUI and uses '1bd1bc' SER option then the generated memory will contain 127 bits. The extra 7 bits are needed for storing the error detection and correction code. So with SER option selected the range of bits specified in the GUI has to be lower then the maximum allowed for a given value of multiplexer. As an example the maximum number of bits allowed for multiplexer=8 with SER is 120 for a single port memory instead of 128 with SER option 'none'.

3.2.5 Single-Port SRAM Block Diagrams (sram_sp_adv)

The synchronous Advantage single-port SRAM instance block diagram is shown in Figures 3-5 and 3-6.

Figure 3-5. Single-Port Advantage SRAM Basic Block Diagram



m = number of address bits
n = number of data bits

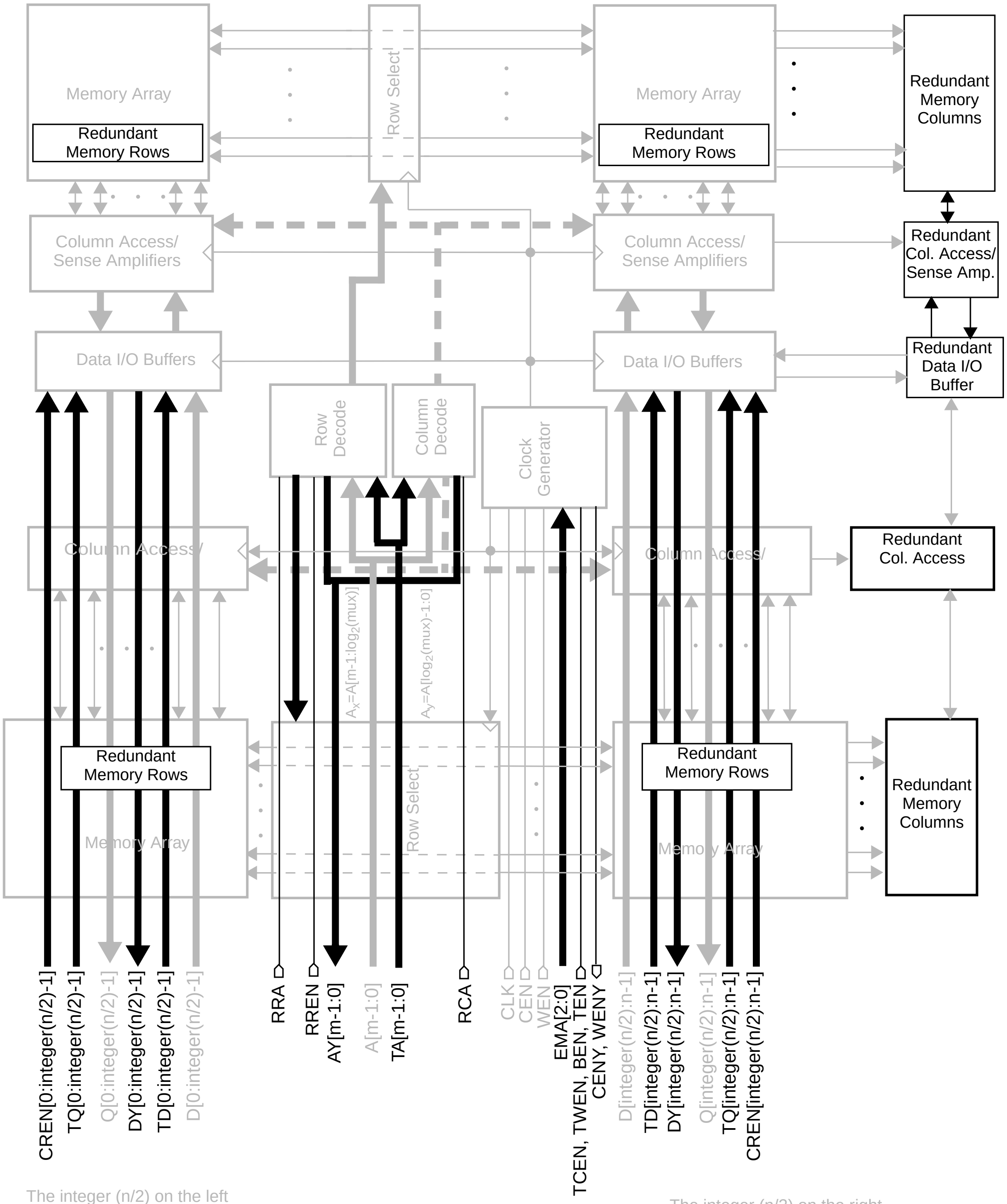
The integer (n/2) on the left side (above) is rounded down. Thus, when there is an odd number of bits, the left side will have one less bit than the right side.

The integer (n/2) on the right side (above) is rounded up. Thus, when there is an odd number of bits, the right side will have one more bit than the left side.

Notes:

Word-Write Mask
When the word-write mask option is turned on, the WEN pin is a bus signal and is located at the Data I/O Buffers (not shown in block diagram).
When the word-write mask option is turned off, the WEN pin is a signal pin and is located at the Clock Generator, as shown.

Figure 3-6. Single-Port Advantage SRAM Basic and Test/Repair Block Diagram



The integer (n/2) on the left side (above) is rounded down. Thus, when there is an odd number of bits, the left side will have one less bit than the right side.

The integer (n/2) on the right side (above) is rounded up. Thus, when there is an odd number of bits, the right side will have one more bit than the left side.

m = number of address bits
n = number of data bits

Notes:

Word-Write Mask
When the word-write mask option is turned on, the WEN pin is a bus signal and is located at the Data I/O Buffers, (not shown in block diagram).
When the word-write mask option is turned off, the WEN pin is a signal pin and is located at the Clock Generator, as shown.

EMA
The EMA option is always on; the EMA pin is an input bus signal.

BIST MUX
When the BIST MUX option is turned on, the TCEN, TEN, BEN, CENY, TD, TQ, TA, DY, and AY pins are available.
When the BIST MUX option is turned on and:

- the word-write mask option is turned on, the WEN, TWEN, and WENY pins are bus signals and are located at the Data I/O Buffers (not shown in block diagram).
- the word-write mask option is turned off, the WEN, TWEN, and WENY pins are signal pins and are located at the Clock Generator, as shown.

When the BIST MUX option is turned off, the TCEN, TEN, BEN, CENY, TD, TQ, TA, DY, AY, TWEN, and WENY pins are unavailable.

3.2.6 Single-Port SRAM Core Address Maps (sram_sp_adv)

An example of the standard physical core mapping for Advantage mux values is shown in Figures 3-7 through 3-9.

Figure 3-7. Single-Port Advantage SRAM Mux 8: Core Address Mapping

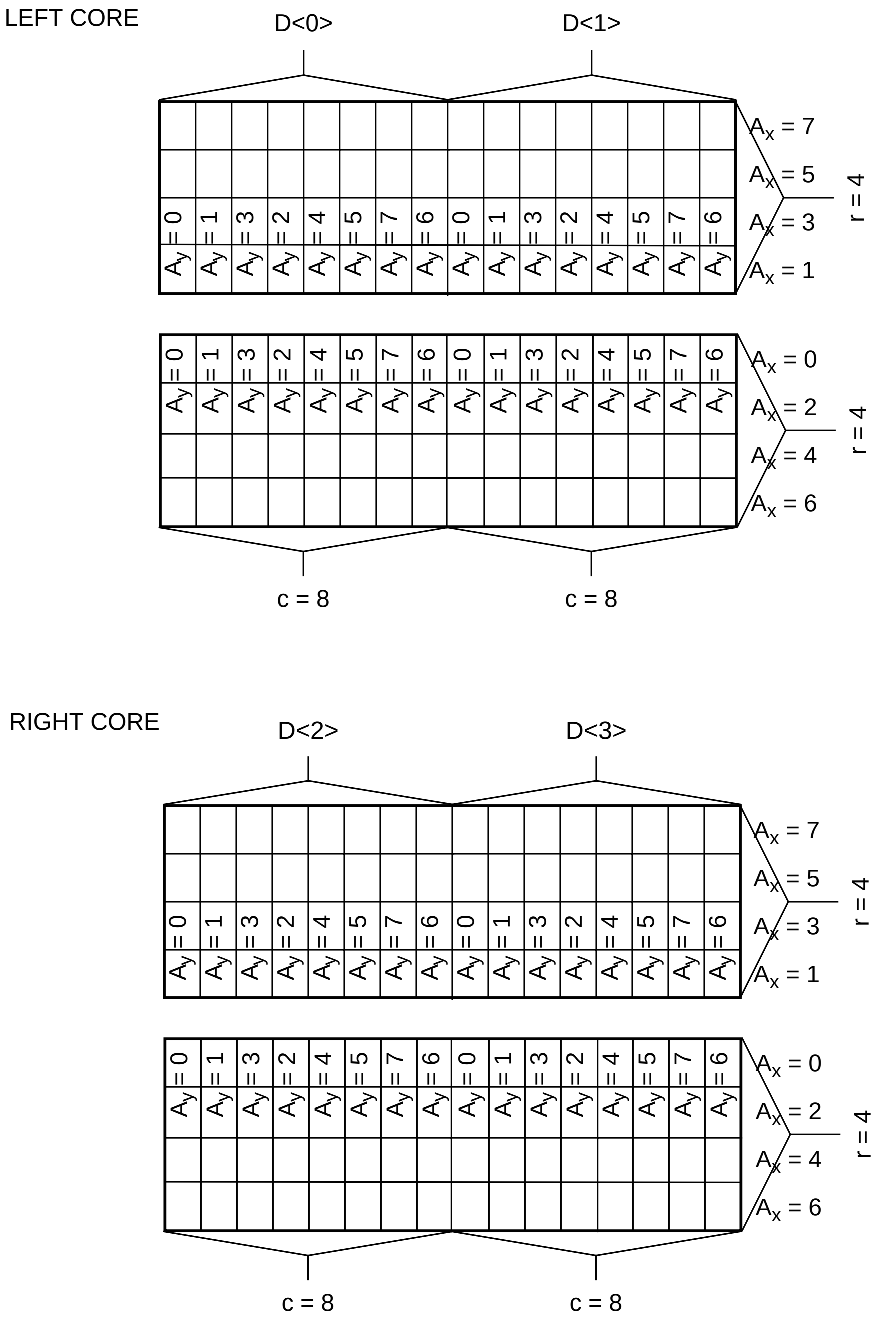


Figure 3-8. Single-Port Advantage SRAM Mux 16: Core Address Mapping

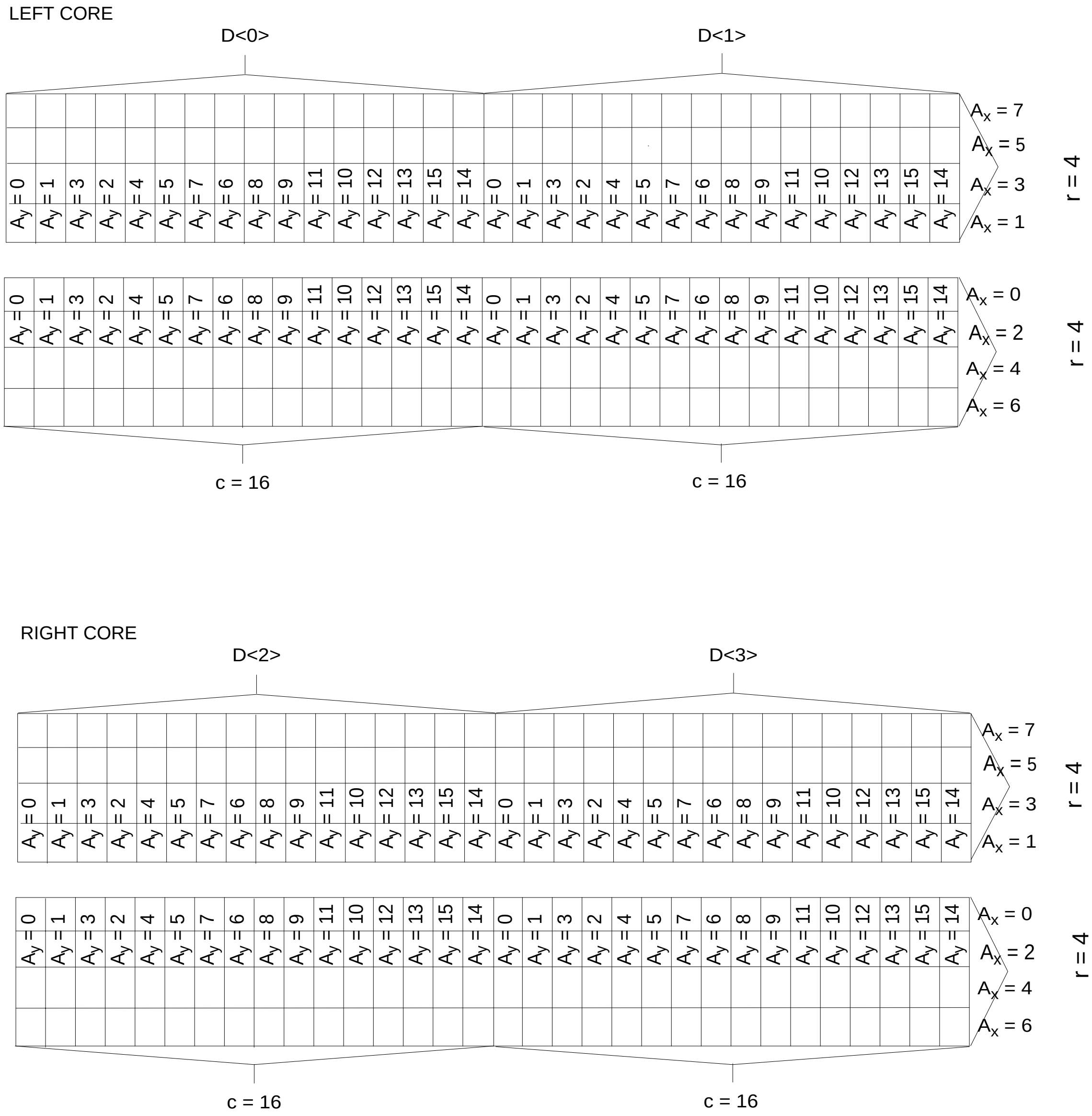


Figure 3-9. Single-Port Advantage SRAM Mux 32: Core Address Mapping (Part 1 of 2)

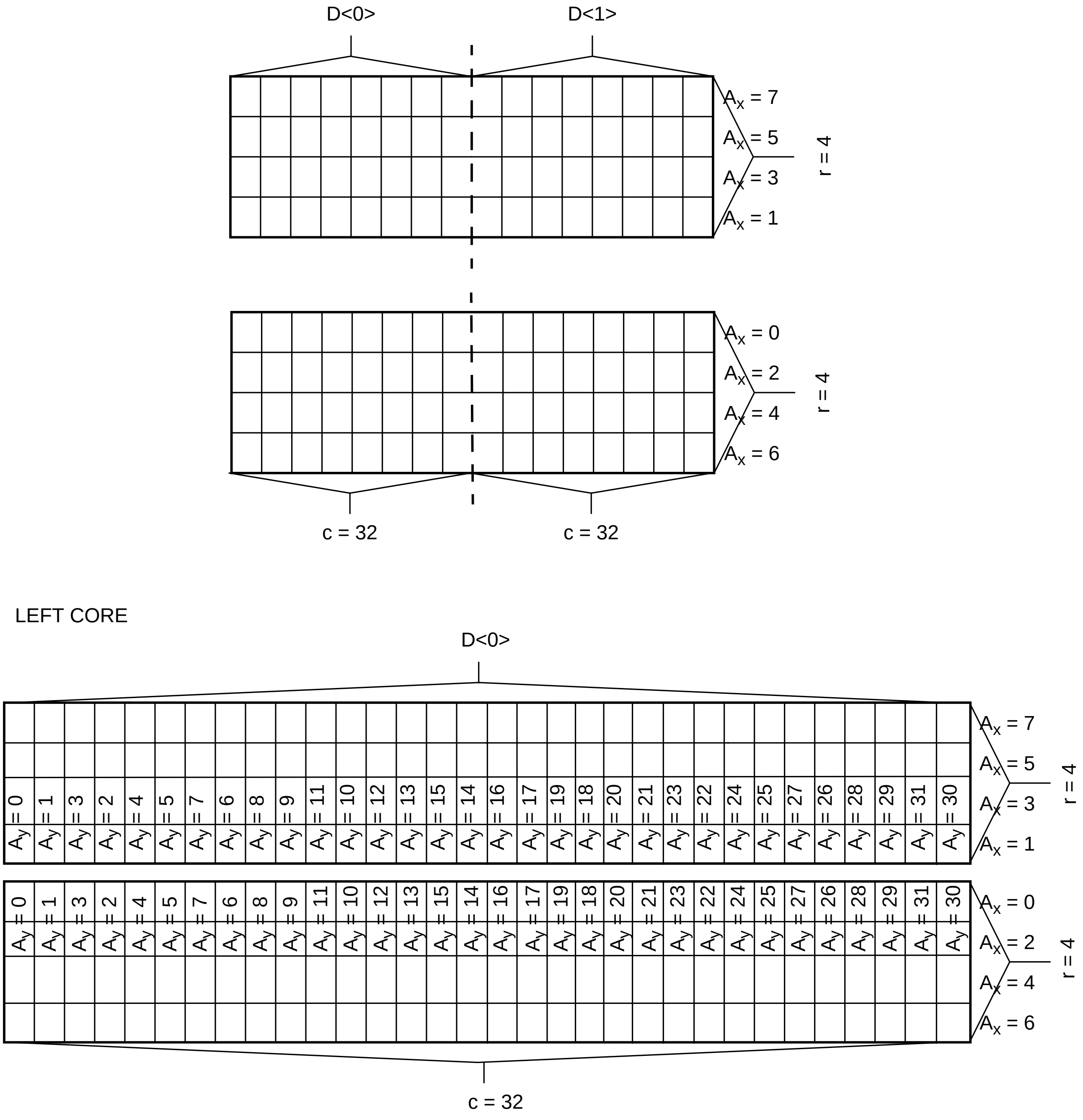
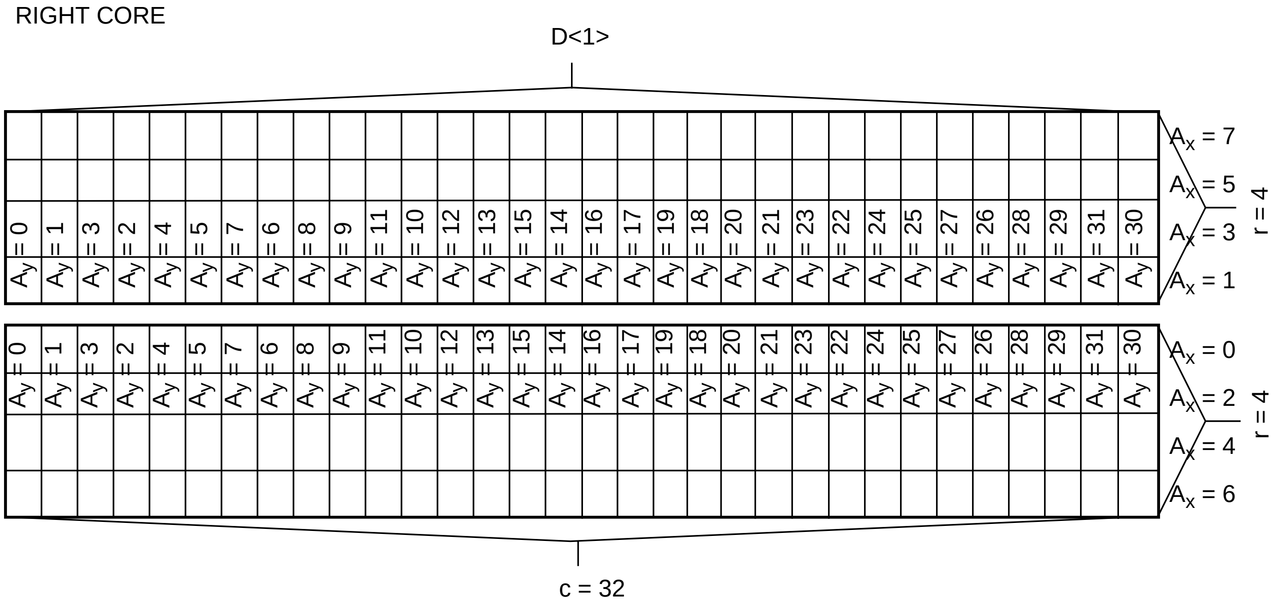


Figure 3-10. Single-Port Advantage SRAM Mux 32: Core Address Mapping (Part 2 of 2)



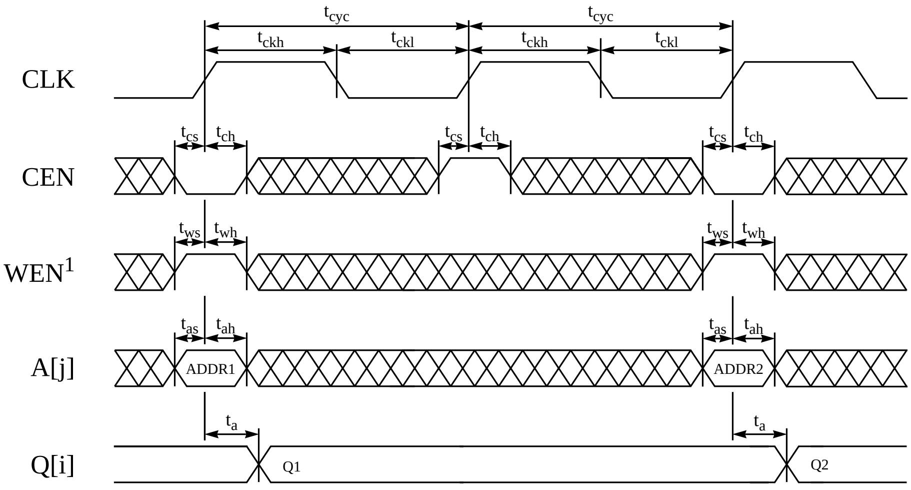
3.2.7 Single-Port SRAM Timing Specifications (sram_sp_adv)

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 "Pin Descriptions for Single-Port SRAM Generators" on page 3-11.

3.2.7.1 Single-Port SRAM Timing Diagrams (sram_sp_adv)

Figures 3-11 and 3-12 show timing diagrams for Advantage 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.

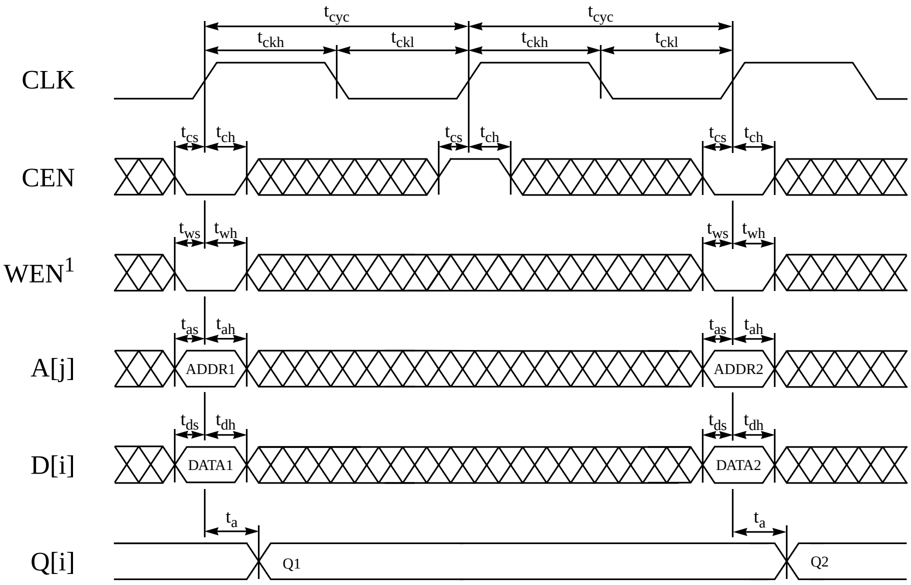
Figure 3-11. Single-Port SRAM Read-Cycle Timing



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.

[1] 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.

Figure 3-12. 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.

[1] 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.

3.2.7.2 Single-Port SRAM Timing Parameters (sram_sp_adv)

The GUI generated ASCII datatable contains timing parameters listed in Table 3-10.

Table 3-10. Single-Port SRAM Timing Parameters

Parameter	Symbol
Cycle time	t_{cyc}
Access time[1,2]	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}

Table 3-10. Single-Port SRAM Timing Parameters (Continued)

Parameter	**Symbol**
Data hold	t_{dh}
Clock high (minimum pulse width)	t_{ckh}
Clock low (minimum pulse width)	t_{ckl}
Clock rise slew (maximum transition time)	t_{ckr}
Access time and EMA enabled: eight numbers for eight values of EMA[1,2]	$t_{a[0-7]}$
Extra margin enable pin setup	t_{emas}
Extra margin enable pin hold	t_{emah}
Address setup, test pin	t_{tas}
Address hold, test pin	t_{tah}
Chip enable setup, test pin	t_{tcs}
Chip enable hold, test pin	t_{tch}
Write enable setup, test pin	t_{tws}
Write enable hold, test pin	t_{twh}
Data setup, test pin	t_{tds}
Data hold, test pin	t_{tdh}
Test enable setup	t_{tens}
Test enable hold	t_{tenh}
Bypass enable setup	t_{bens}
Bypass enable hold	t_{benh}
Column redundancy enable setup	t_{crens}
Column redundancy enable hold	t_{crenh}
Redundant column address setup	t_{rcas}
Redundant column address hold	t_{rcah}
Row redundancy enable setup	t_{rrens}
Row redundancy enable hold	t_{rrenh}
Row redundancy address setup	t_{rras}
Row redundancy address hold	t_{rrah}
Load dependence factor on data output (ns/pF)	load_q

Table 3-10. Single-Port SRAM Timing Parameters (Continued)

Parameter	**Symbol**
Load dependence factor on chip enable MUX output (ns/pF)	load_ceny
Load dependence factor on write enable MUX output (ns/pF)	load_weny
Load dependence factor on address MUX output (ns/pF)	load_ay
Load dependence factor on data MUX output (ns/pF)	load_dy

[1] The ASCII datatable shows fixed delay values. These parameters have a load dependence (K_{load}), which is used to calculate: $\mathrm{TotalDelay} = \mathrm{FixedDelay} + (\mathrm{Kload} \times \mathrm{Cload})$, for timing views.

[2] 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.

3.2.7.3 Single-Port SRAM Power Parameters (sram_sp_adv)

The GUI contains an ASCII datatable that provides characterization values for each corner. These values are also available in a generated postscript datasheet. Table 3-11 shows the single-port SRAM power parameters.

Table 3-11. Single-Port SRAM Power Parameters

Parameter	Symbol
AC Current[1, 4]	i_{cc}
Read AC Current[4]	i_{cc_r}
Write AC Current[4]	i_{cc_w}
Peak Current[4]	i_{cc_peak}
Deselected Current[2]	i_{cc_desel}
Standby Current[3]	$i_{cc\_standby}$
AC Current: eight numbers for eight values of EMA	$i_{cc[0:7]}$
Read AC Current: eight numbers for eight values of EMA	$i_{cc\_r[0:7]}$
Write AC Current: eight numbers for eight values of EMA	$i_{cc\_w[0:7]}$

[1] 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.

[2] 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.

[3] Value is independent of frequency and assumes all inputs and outputs are stable.

[4] For most generators, value shows dynamic current without leakage (standby) component. Refer to 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. Refer to “Current Calculations” on page 3-58 for instructions on recalculating the current for a specific design. Refer to “Noise Limits” on page 3-60 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 (sram_dp_adv)

Descriptions for the basic functionality of this generator are provided. In addition, descriptions of features you can use to enable the test and repair functions of your generator are provided. You can obtain further repair and test details and scenarios from the "90nm Advantage Memory Generator Test and Repair Features Application Note".

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

3.3.1.1 Basic Functionality

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] is 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] is the result of a read operation.

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.

3.3.1.2 Test and Repair Functionality

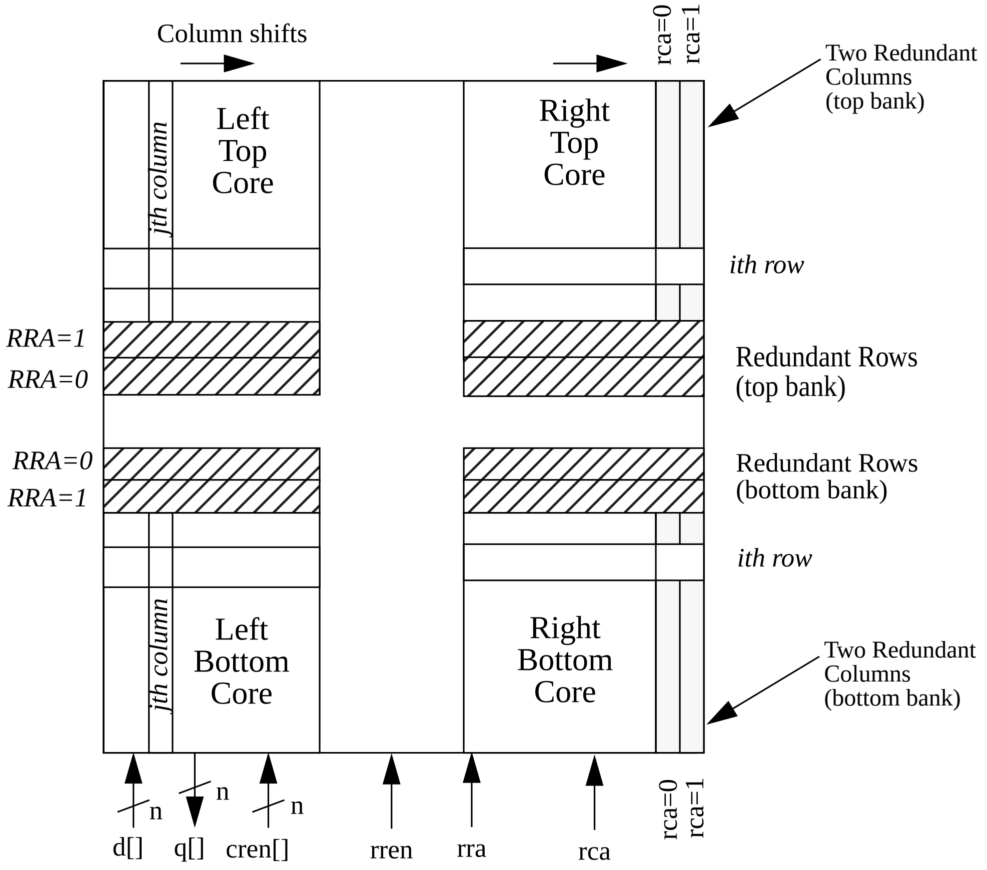
The SRAM generator includes features such as Flex-Repair Redundancy, Soft Error Repair (SER), Built-In Self Test (BIST) MUXes, and Extra Margin Adjustment (EMA), as described in this section. Feature options are described in the “Generator Options” section on page 2-22. You can also obtain examples of how this functionality can be implemented in the “90nm Advantage Memory Generator Test and Repair Features Application Note.”

3.3.1.2.1 Flex-Repair/Redundancy

The Flex-Repair feature (also referred to as Redundancy) supports two rows and two columns of redundancy in the memory block. The memory generator can also create the RTL that is used to drive the memory signals to access the redundant locations. Figure 3-13 shows the architecture of a dual-port SRAM with redundancy, with two rows and two columns.

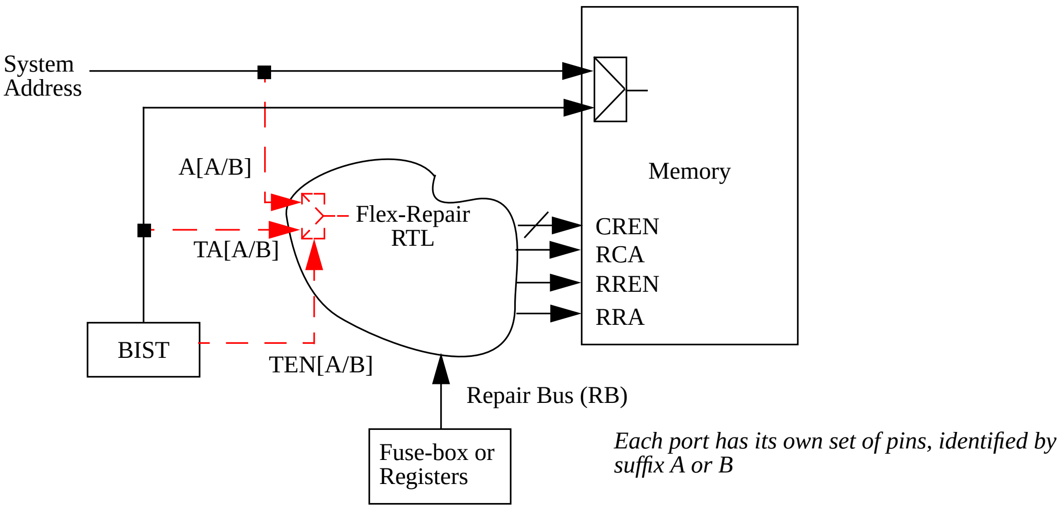
☞ While only two redundant columns can be accessed, the width of the group of columns added for redundancy is set by the sense amp width.

Figure 3-13. Architecture of a Typical Dual-Port SRAM with Redundancy



rca: RCAA, RCAB present only when two column redundancy is enabled
rra: RRAA, RRAB present only when two row redundancy is enabled
cren: CRENA, CRENB present only when column redundancy is enabled
rren: RRENA, RRENAB present only when row redundancy is enabled

Figure 3-14. RTL for Redundancy Created by the Generator



Each port has its own set of pins, identified by suffix A or B

Refer to the “90nm Advantage Memory Generator Test and Repair Features Application Note” for more detailed information about RTL pins.

3.3.1.2.2 Soft-Error Repair (SER)

SER adds extra bits per word into the memory block automatically and creates accompanying RTL logic needed to implement Error Correction Codes (ECC) for SER. A composite RTL is created if both RTL and SER are enabled.

SER has two options:

- Single bit error correction and single bit error detection
- Single bit error correction and double bit error detection

Refer to the “90nm Advantage Memory Generator Test and Repair Features Application Note” for more detailed information about SER pins.

3.3.1.2.3 Extra Margin Adjustment (EMA)

EMA is always enabled. The delays are selected by programming values 000 through 111 on pins EMA [2:0]. The validated or baseline value is 000 as a setting of 000. Incremental values greater than 000 provide progressively slower timing pulses.

This delay provides extra time for memory read and write operations by slowing down the memory access. The extra time may result in yield increase in the case of unexpected manufacturing instability with respect to memory circuitry, especially the bit cells.

EMA margin increments can be defined on a project-specific basis but a default methodology, based on relative delay increments (versus absolute), is needed.

EMA[000] > BLM_DT
EMA[001] > BLM_DT + EMA_DT
EMA[010] > BLM_DT + 2*EMA_DT
EMA[011] > BLM_DT + 3*EMA_DT

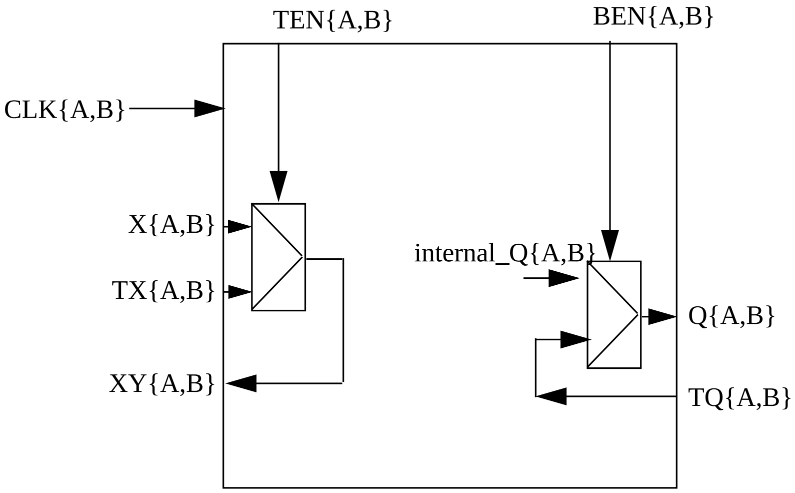
3.3.1.2.4 Built-In Self Test (BIST) MUXes

BIST requires memory inputs that are driven by BIST logic and regular system signals. The SRAM generators have MUXes built into their timing critical memory pins with minimal effect on timing. The MUX outputs at input pins are available as separate pins that can be used to test the MUXes themselves.

Memory outputs should be easily controlled for better testability of logic in SRAMs. The SRAM generator has built-in bypass MUXes that can be driven by a scan chain in bypass mode. These MUXes have a very small effect on the memory access time and area.

Figure 3-15 shows the BIST MUXes block diagram, where each pin applies to ports A and B.

Figure 3-15. Dual-Port SRAM BIST MUXes Block Diagram



X can be A, D, CEN and WEN.

3.3.2 Dual-Port SRAM Pins (sram_sp_adv, sram_dp_adv)

Figure 3-16 shows basic and optional test/repair pins for the dual-port SRAM generator.

Figure 3-16. Dual-Port SRAM Basic and Test/Repair Pins

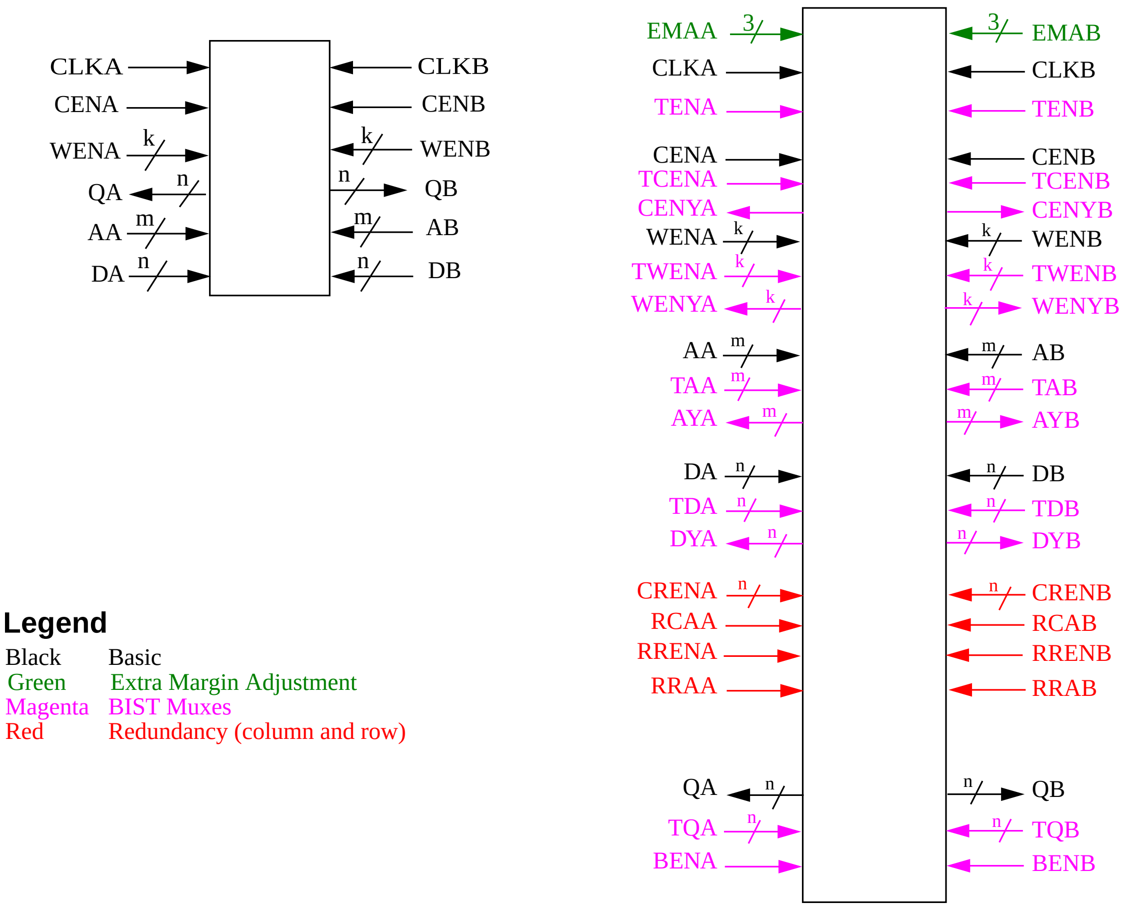


Table 3-12 provides the dual-port (ports A and B) SRAM generator pin descriptions.

Table 3-12. Pin Descriptions for Dual-Port SRAM Generators

Name	Type	Description
		Basic Pins
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)
		Extra Margin Adjustment Pins
EMAA[2:0], EMAB[2:0]	Input	Extra Margin Adjustment (EMAA[0] = LSB), (EMAB[0] = LSB)
		Flex-Repair Pins
CRENA[n-1:0], CRENB[n-1:0]	Input	Column Redundancy Enables, active low (CRENA[0] = LSB), (CRENB[0] = LSB). Available when one or two column redundancy is enabled.
RCAA, RCAB	Input	Redundant Column Addresses. Available when two-column redundancy is enabled.
RRENA, RRENB	Input	Row Redundancy Enables, active low. Available when one or two row redundancy is enabled.
RRAA, RRAB	Input	Redundant Row Addresses, lowest row for 0. Available when two row redundancy is enabled.
		BIST Mux Pins
TENA, TENB	Input	Test mode Enables, active low: 0=Test operation, 1=Normal operation
TAA[m-1:0], TAB[m-1:0]	Input	Address Test Inputs (TAA[0] = LSB), (TAB[0] = LSB)
AYA[m-1:0], AYB[m-1:0]	Output	Address MUX outputs (AYA[0] = LSB), (AYB[0] = LSB)
TDA[n-1:0], TDB[n-1:0]	Input	Data Test inputs (TDA[0] = LSB), (TDB[0] = LSB)
DYA[n-1:0], DYB[n-1:0]	Output	Data MUX outputs (DYA[0] = LSB), (DYB[0] = LSB)
TCENA, TCENB	Input	Test Mode Chip Enable (TCEN) Test inputs, active low
CENYA, CENYB	Output	Chip Enable (CEN) MUX outputs
TWENA [*], TWENB [*]	Input	Test Mode Write Enable (TWEN) Test inputs, active low. (TWENA[0] = LSB), (TWENB[0] = LSB) *If word-write mask is enabled, this becomes a bus.
WENYA[*], WENYB[*]	Output	Write Enable MUX outputs. (WENA[0] = LSB), (WENB[0] = LSB) *If word-write mask is enabled, this becomes a bus.
BENA, BENB	Input	Bypass Mode Enables, active low. 0=Bypass operation, 1=Normal operation
TQA[n-1:0], TQB[n-1:0]	Input	Bypass Q inputs in write mode (if BEN=0, Q[n-1:0]=TQ[n-1:0])

3.3.3 Dual-Port SRAM Logic Tables (sram_dp_adv)

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. wen is WEN when word-mask is on and WEN[ ] when it is off.

Table 3-13 shows the logic functions for basic dual-port SRAM generator functions.

Table 3-13. 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	**Word-write:** 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] **Word Write Mask:** The corresponding data partition is selected by the write enables, WEN[k-1:0]; 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 and driven from the memory location specified on the address bus A[m-1:0].

Table 3-14 describes the behavior of BIST MUXes at inputs to the memory.

Table 3-14. Dual-Port SRAM BIST MUXes at Memory Input

TEN	Mode	Function
H	Regular Mode	In this mode, the basic pins, A[], D[], CEN, and WEN are used for the internal operations described for basic features, Table 3-13. These inputs are also available at the MUX outputs: AY[ ], DY[ ],CENY, and WENY, respectively.
L	Test Mode	In this mode, the test pins, TA[], TD[], TCEN, and TWEN are used for the internal operations described for basic features, Table 3-13. These inputs are also available at the MUX outputs: AY[ ], DY[ ],CENY, and WENY, respectively.

Table 3-15 describes the behavior of BIST MUXes at the output pins.

Table 3-15. Dual-Port SRAM BIST MUXes at Memory Output

BEN	Mode	Q	Function
H	Regular Mode	Last Data Read or Written	In this mode, the data from the last read or write operation is available at Q.
L	Test Mode	Data at bypass pin	Data at the bypass pin TQ is available at output Q. This operation is not clocked. All other operations described in Table 3-13, for basic features, work when BEN=0, with the exception that data does not appear at the output.

Redundancy is a user-selectable option and combinations are presented in Table 3-16. When the Repair RTL is generated, it is always generated for two rows and two columns redundancy. This RTL has a Repair Bus as an Input. This Repair Bus (RB) should be connected to the fuse box. The fuse box bits are used to program the RTL in order to generate signals that correct the defect in the memory.

You have the option to program this RB to get different redundancy configurations, for example, two rows and one column, if needed. The RB contains four control bits (RRE2, RRE1, CRE2, CRE1) that enable or disable a particular redundant row or column. By selective tie-off of these control bits to logic-0 you can disable a corresponding redundant row or column.

Corresponding to these control bits there are other signals generated when redundancy is enabled: (RRE2, FRA2), (RRE1, FRA1), (CRE2, FBA2, FCA2) and (CRE1, FBA1, FCA1). If a control bit is disabled by tie-off to logic-0, you should tie-off the rest of the bits in that group to logic-0 as well. After a given configuration is set, the synthesis is run with Boolean optimization and prunes the unneeded logic of the RTL. For example if the (RRE2, FRA2) bits are tied-off to logic-0 to get single row redundancy, then the result is RRA memory port being tied-off with logic-0. Use Table 3-16 to obtain the desired redundancy configuration; remaining pins should be connected as usual to the fuse box. This process is also described inside the Repair RTL.

Table 3-16. Dual-Port Redundancy Configuration Scheme

To obtain these Redundancy Configurations	Perform these Tie-off to logic-0 operations
Two Column Redundancy, no Row Redundancy	RRE2, FRA2, RRE1, FRA1
Two Column Redundancy, one Row Redundancy	RRE2, FRA2
One Column Redundancy, no Row Redundancy	RRE2, FRA2, RRE1, FRA1, CRE2, FBA2, FCA2
One Column Redundancy, one Row Redundancy	RRE2, FRA2, CRE2, FBA2, FCA2
One Column Redundancy, Two Row Redundancy	CRE2, FBA2, FCA2
No Column Redundancy, one Row Redundancy	RRE2, FRA2, CRE2, FBA2, FCA2, CRE1, FBA1, FCA1
No Column Redundancy, Two Row Redundancy	CRE2, FBA2, FCA2, CRE1, FBA1, FCA1
No Column and No Row Redundancy	Turn-off the redundancy option and regenerate all models.
Two Column and Two Row Redundancy	No Need to do any tie-off’s.

Tables 3-17 and 3-18 show row (RREN) and column (CREN) redundancy logic tables, respectively.

Row redundancy is available only as an incremental option to column redundancy; there is no row redundancy without column redundancy. This is reflected in the GUI.

Table 3-17. Dual-Port SRAM Row Redundancy

RRENA	Mode	Function
H	Regular operation	Operations are performed as defined in Table 3-13, for basic memory operations. Main memory locations are accessed as specified by A[m-1:0].
L	Redundant row access	Instead of the locations in the basic rows, the redundant rows are accessed as specified by RAA and A[p-1:0]. For two row redundancy, RRA=0 selects the lower row.

☞ For MUX values of 4, 8, and 16, the values for “p” are 2, 3, and 4, respectively.

Table 3-18. Dual-Port SRAM Column Redundancy

CRENA[i]	Mode	Function
H	Normal	The data is routed from pins (D[i], Q[i]) to the respective bit locations in the memory core.
L	Shift	The data is routed from pins (D[i], Q[i]) to the adjacent bits (i+1) in the memory core. The right-most bit is shifted to one the redundant columns. For two redundant columns, the right-most bit is shifted to one of the two columns, depending on value of RCA.

3.3.4 Dual-Port SRAM Parameters (sram_dp_adv)

The standard input and block parameters of a synchronous dual-port SRAM are described in Table3-19. You can also refer to 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 3-19. Dual-Port High Speed/Density 90nm SRAM (sram_dp_adv) Parameters

Input Parameters		
Parameter	**Ranges**	
number of words	mux = 4	128 to 2048, increment = $\text{mux} \bullet 4$
	mux = 8	256 to 4096, increment = $\text{mux} \bullet 4$
	mux = 16	512 to 8192, increment = $\text{mux} \bullet 4$
number of bits[1]	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 $1/t_{cyc} \bullet 1000$, increment = 1	
word partition size[2]	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	m5 - 9	
power type	Rings, ArtiGrid; default rings	
horizontal ring layer	m2, m3, or m4; Ring power type must be selected; default m3	
vertical ring layer	m2, m3, or m4; Ring power type must be selected; default m4	
ring width (µm)	2 to 10,000	
extra margin adjustment	always on	
redundancy	on, off; default off	
redundant columns	0, 1, or 2; redundancy must be On; default = 2	
redundant rows	0, 1, or 2; redundancy must be On; default = 0	
BIST MUXes	on, off; default off	
soft error repair[3]	none, 1bd1bc, 2bd1bc; default none.	
Block Parameters		
Parameter	**Ranges**	
total memory bits	256 to 262,144, total bits = $\text{words} \bullet \text{bits}$	
rows in memory matrix	32 to 512, increment = 2, rows = $\text{words} / \text{mux}$	
columns in memory matrix	8 to 512, increment = mux, columns = $\text{bits} \bullet \text{mux}$	
address lines	mux = 4	7 to 11
	mux = 8	8 to 12
	mux = 16	9 to 13
output drive strength, Q[ ]	Refer to the generator GUI or command line help instructions on page 2-3.	

[1] Enabling the SER feature for soft-error detection and correction increases the limits beyond the specified number of bits.

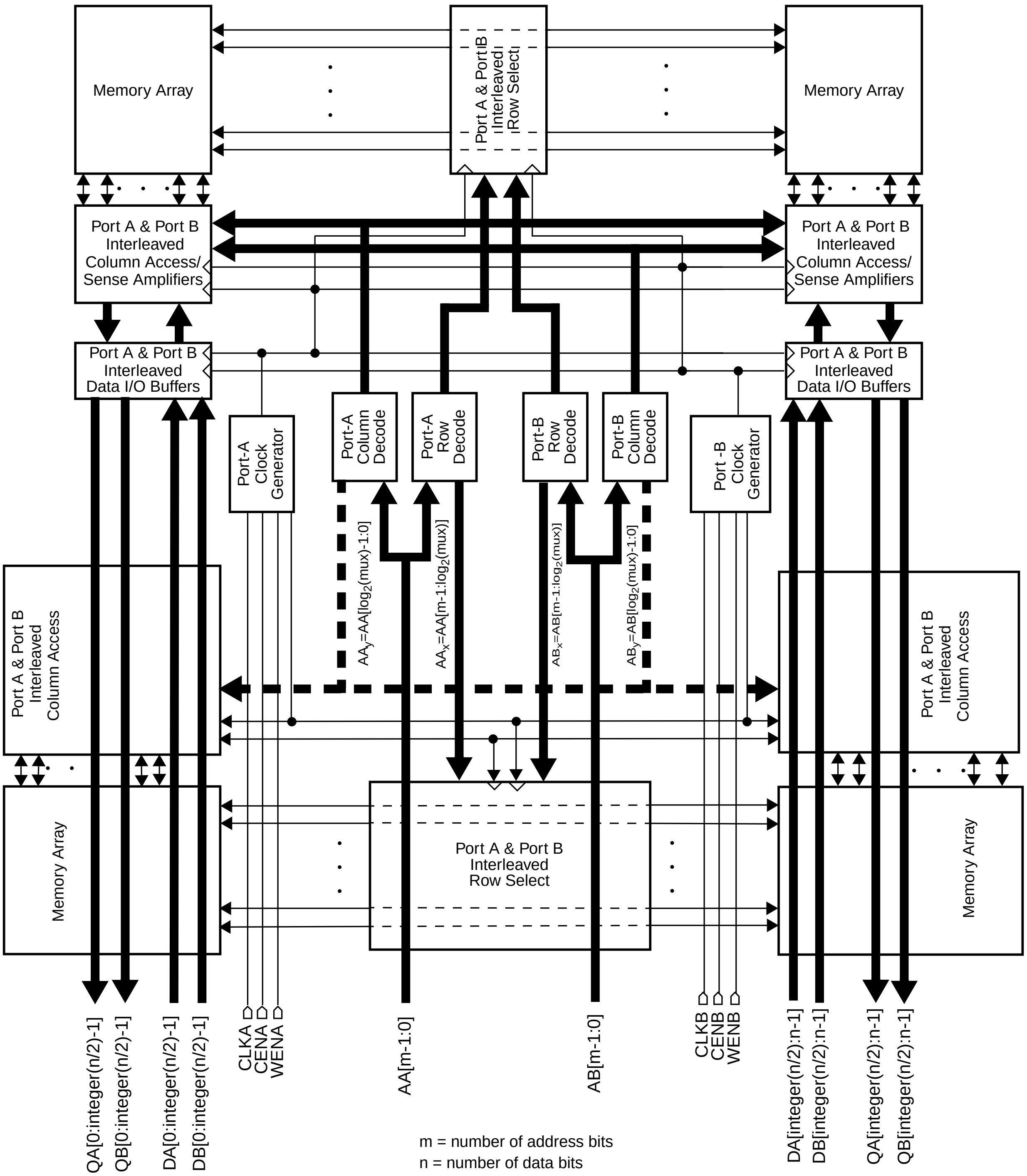
[2] The input pin capacitance for each pin of the write enable bus is proportional to the size of the word partition. For example, an instance with bits=32 and wp_size=24 will have two partitions, one with 24 bits and one with 8 bits. The write enable pin for the 24 bit partition will have a significantly larger input pin capacitance than the write enable pin for the 8 bit partition. When modelling 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] When the SER feature is enabled and any of the '1bd1bc' or '2bd1bc' options are selected, the actual memory generated will have more bits then specified in the GUI, these extra bits are used to store the error detection and correction code. For example if user specifies 120 bit memory in GUI and uses '1bd1bc' SER option then the generated memory will contain 127 bits. The extra 7 bits are needed for storing the error detection and correction code. So with SER option selected the range of bits specified in the GUI has to be lower then the maximum allowed for a given value of multiplexer. As an example the maximum number of bits allowed for multiplexer=8 with SER is 120 for a single port memory instead of 128 with SER option 'none'.

3.3.5 Dual-Port SRAM Block Diagrams (sram_dp_adv)

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. Figures 3-17 and 3-18 show the block diagram for Advantage dual-port SRAMs.

Figure 3-17. Dual-Port Advantage SRAM Basic Block Diagram


Memory Array
Port A & Port B Interleaved Row Select
Memory Array
Port A & Port B Interleaved Column Access/ Sense Amplifiers
Port A & Port B Interleaved Column Access/ Sense Amplifiers
Port A & Port B Interleaved Data I/O Buffers
Port A & Port B Interleaved Data I/O Buffers
Port-A Clock Generator
Port-A Column Decode
Port-A Row Decode
Port-B Row Decode
Port-B Column Decode
Port -B Clock Generator
AA_y=AA[log$_2$(mux)-1:0]
AA_x=AA[m-1:log$_2$(mux)]
AB_x=AB[m-1:log$_2$(mux)]
AB_y=AB[log$_2$(mux)-1:0]
Port A & Port B Interleaved Column Access
Port A & Port B Interleaved Column Access
Memory Array
Port A & Port B Interleaved Row Select
Memory Array
QA[0:integer(n/2)-1]
QB[0:integer(n/2)-1]
DA[0:integer(n/2)-1]
DB[0:integer(n/2)-1]
CLKA
CENA
WENA
AA[m-1:0]
AB[m-1:0]
CLKB
CENB
WENB
DA[integer(n/2):n-1]
DB[integer(n/2):n-1]
QA[integer(n/2):n-1]
QB[integer(n/2):n-1]

m = number of address bits
n = number of data bits

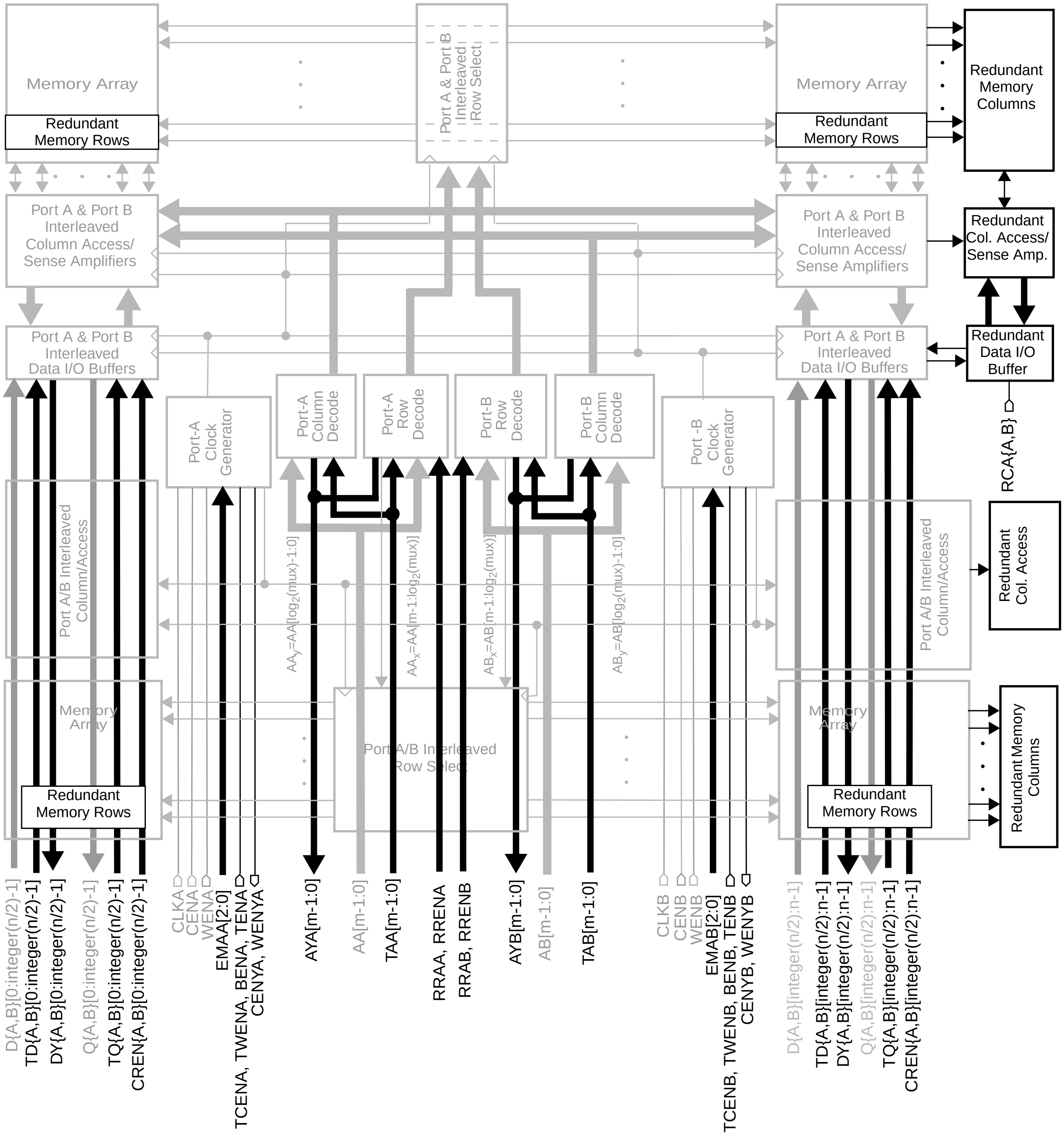
The integer (n/2) on the left side (above) is rounded down. Thus, when there is an odd number of bits, the left side will have one less bit than the right side.

The integer (n/2) on the right side (above) is rounded up. Thus, when there is an odd number of bits, the right side will have one more bit than the left side.

Notes:

Word-Write Mask
When the word-write mask option is turned on, the WEN{A,B} pin is a bus signal and is located at the Data I/O Buffers (not shown in block diagram).
When the word-write mask option is turned off, the WEN{A,B} pin is a signal pin and is located at the Clock Generator, as shown.

Figure 3-18. Dual-Port Advantage SRAM Basic and Test/Repair Block Diagram


Memory Array
Redundant Memory Rows
Port A & Port B Interleaved Row Select
Memory Array
Redundant Memory Rows
Redundant Memory Columns
Port A & Port B Interleaved Column Access/ Sense Amplifiers
Port A & Port B Interleaved Column Access/ Sense Amplifiers
Redundant Col. Access/ Sense Amp.
Port A & Port B Interleaved Data I/O Buffers
Port A & Port B Interleaved Data I/O Buffers
Redundant Data I/O Buffer
RCA{A,B}
Port-A Clock Generator
Port-A Column Decode
Port-A Row Decode
Port-B Row Decode
Port-B Column Decode
Port -B Clock Generator
Port A/B Interleaved Column/Access
Port A/B Interleaved Column/Access
Redundant Col. Access
AA$_y$=AA[log$_2$(mux)-1:0]
AA$_x$=AA[m-1:log$_2$(mux)]
AB$_x$=AB[m-1:log$_2$(mux)]
AB$_y$=AB[log$_2$(mux)-1:0]
Memory Array
Redundant Memory Rows
Port A/B Interleaved Row Select
Memory Array
Redundant Memory Rows
Redundant Memory Columns
D{A,B}[0:integer(n/2)-1]
TD{A,B}[0:integer(n/2)-1]
DY{A,B}[0:integer(n/2)-1]
Q{A,B}[0:integer(n/2)-1]
TQ{A,B}[0:integer(n/2)-1]
CREN{A,B}[0:integer(n/2)-1]
CLKA
CENA
WENA
EMAA[2:0]
TCENA, TWENA, BENA, TENA
CENYA, WENYA
AYA[m-1:0]
AA[m-1:0]
TAA[m-1:0]
RRAA, RRENA
RRAB, RRENB
AYB[m-1:0]
AB[m-1:0]
TAB[m-1:0]
CLKB
CENB
WENB
EMAB[2:0]
TCENB, TWENB, BENB, TENB
CENYB, WENYB
D{A,B}[integer(n/2):n-1]
TD{A,B}[integer(n/2):n-1]
DY{A,B}[integer(n/2):n-1]
Q{A,B}[integer(n/2):n-1]
TQ{A,B}[integer(n/2):n-1]
CREN{A,B}[integer(n/2):n-1]

The integer (n/2) on the left side (above) is rounded down. Thus, when there is an odd number of bits, the left side will have one less bit than the right side.

m = number of address bits
n = number of data bits

The integer (n/2) on the right side (above) is rounded up. Thus, when there is an odd number of bits, the right side will have one more bit than the left side.

Notes:

Word-Write Mask
When the word-write mask option is turned on, the WEN{A,B} pin is a bus signal and is located at the Data I/O Buffers (not shown in block diagram).
When the word-write mask option is turned off, the WEN{A,B} pin is a signal pin and is located at the Clock Generator, as shown.

Redundancy
When the redundancy option is turned on and:

- the RCOLS option "1" is selected, the CREN{A,B} pin is an input bus and the RCA{A,B} pin is unavailable.
- the RCOLS option "2" is selected, the CREN{A,B} pin is an input bus and the RCA{A,B} pin is an input pin.
- the RROWS option "1" is selected, the RREN{A,B} pin is an input pin and the RRA{A,B} pin is unavailable.
- the RROWS option "2" is selected, the RREN{A,B} pin is an input pin and the RRA{A,B} pin is an input pin.

When the redundancy option is turned off, the CREN{A,B}, RCA{A,B}, RRA{A,B}, and RREN{A,B} pins are unavailable.

EMA
The EMA option is always turned on; the EMA{A,B} pin is an input bus signal.

BIST MUX
When the BIST MUX option is turned on, the TCEN{A,B}, TEN{A,B}, BEN{A,B}, CENY{A,B}, TD{A,B}, TQ{A,B}, TA{A,B}, DY{A,B}, and AY{A,B} pins are available.

When the BIST MUX option is turned on and:

- the word-write mask option is turned on, the WEN{A,B}, TWEN{A,B}, and WENY{A,B} pins are bus signals and are located at the Data I/O Buffers (not shown in block diagram).
- the word-write mask option is turned off, the WEN{A,B}, TWEN{A,B}, and WENY{A,B} pins are signal pins and are located at the Clock Generator, as shown.

When the BIST MUX option is turned off, the TCEN{A,B}, TEN{A,B}, BEN{A,B}, CENY{A,B}, TD{A,B}, TQ{A,B}, TA{A,B}, DY{A,B}, AY{A,B}, TWEN{A,B}, and WENY{A,B} pins are unavailable.

3.3.6 Dual-Port SRAM Core Address Maps (sram_dp_adv)

This section describes the core address diagrams for Advantage synchronous dual-port SRAMs.

Figures 3-19 through 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

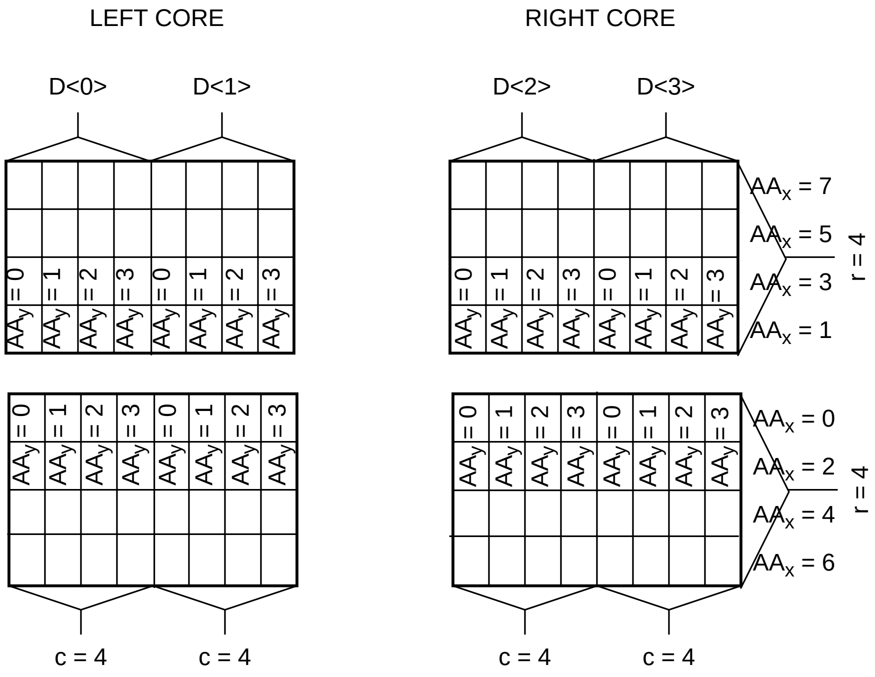


Figure 3-20. Dual-Port SRAM Mux 8: Core Address Mapping

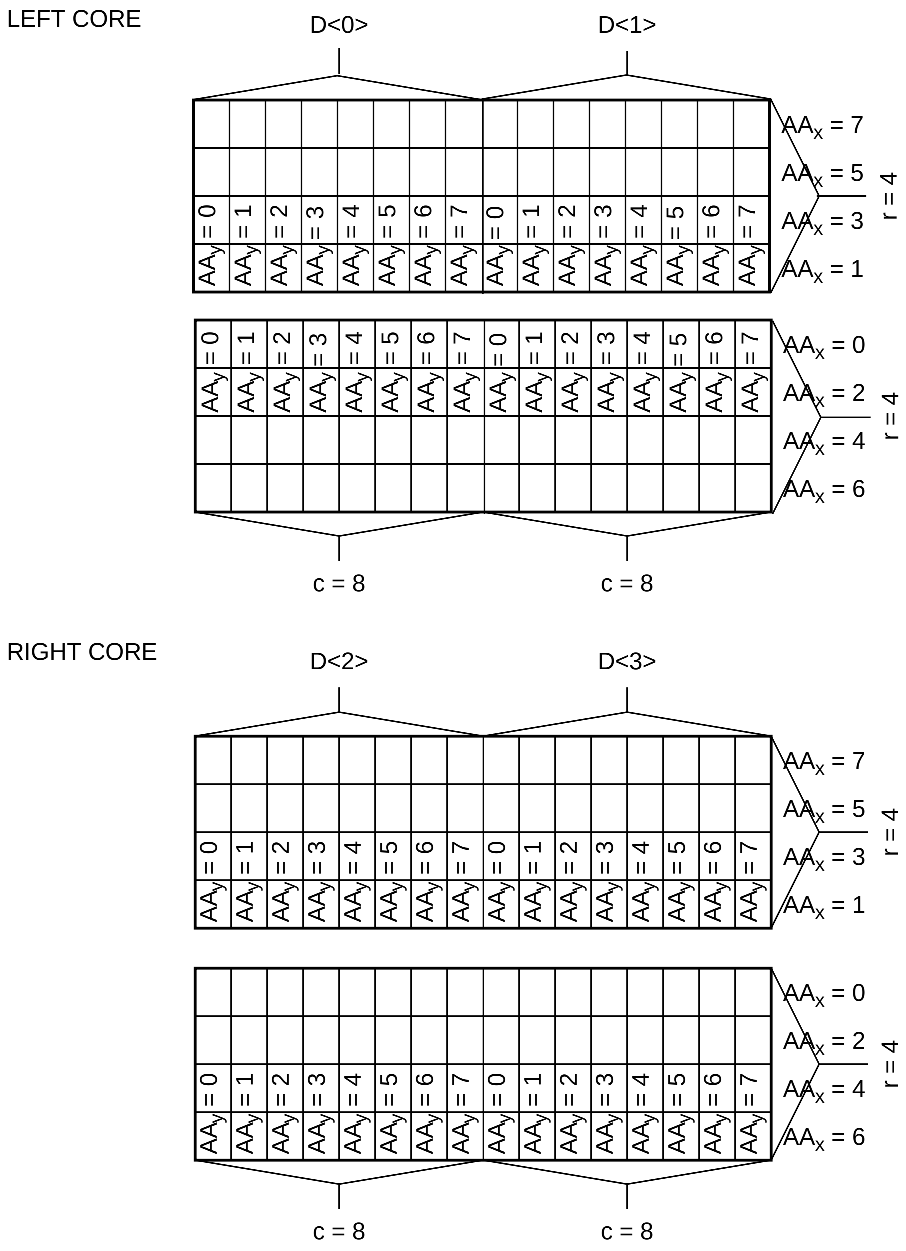
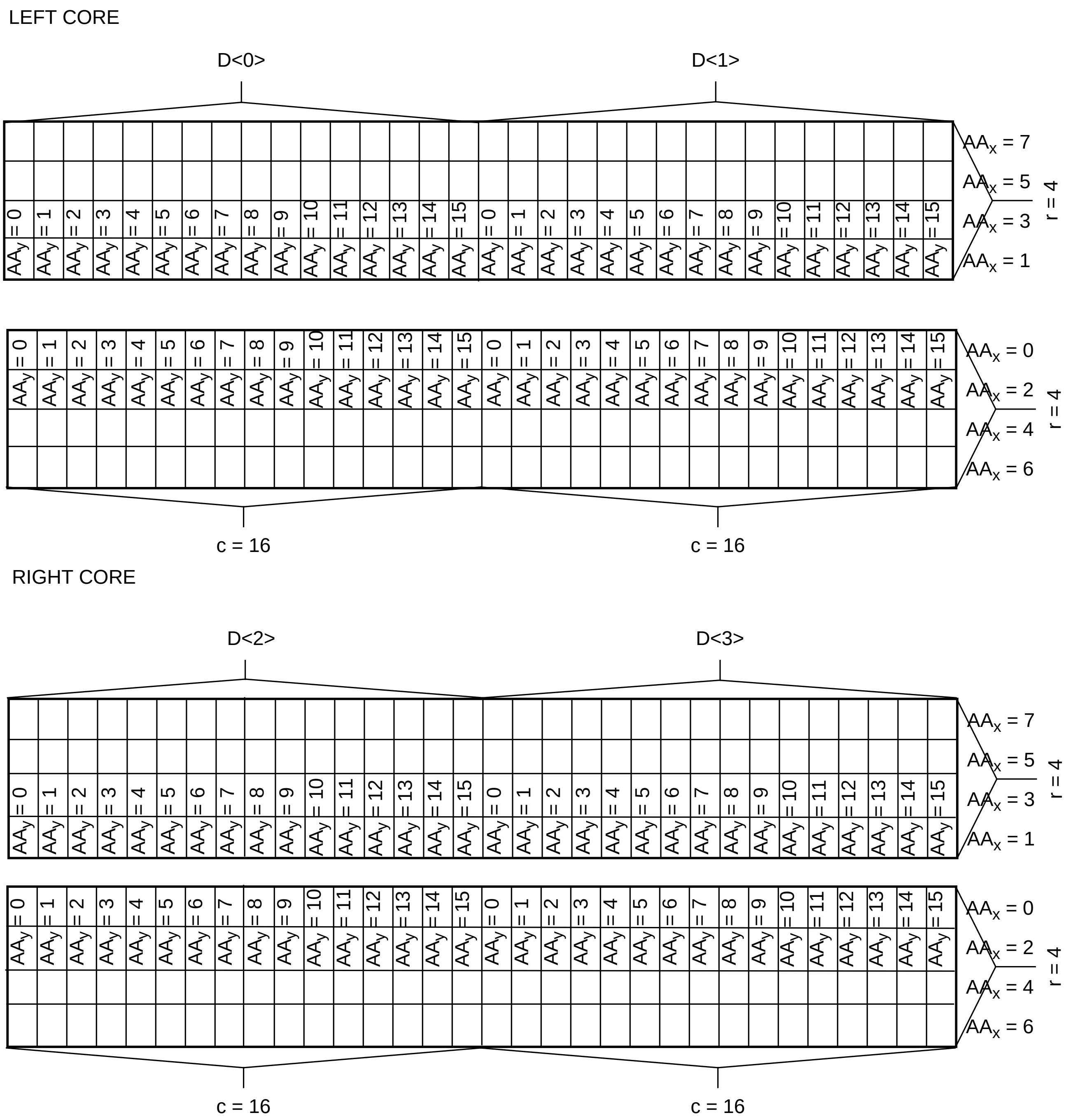


Figure 3-21. Dual-Port SRAM Mux 16: Core Address Mapping



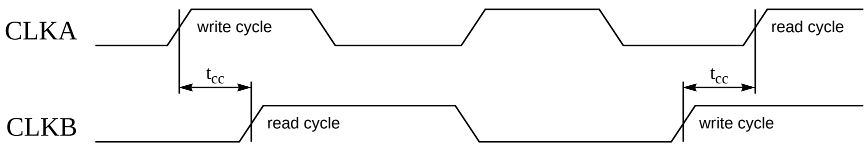
3.3.7 Dual-Port SRAM Timing Specifications (sram_dp_adv)

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

3.3.7.1 Dual-Port SRAM Clock Timing Diagrams (sram_dp_adv)

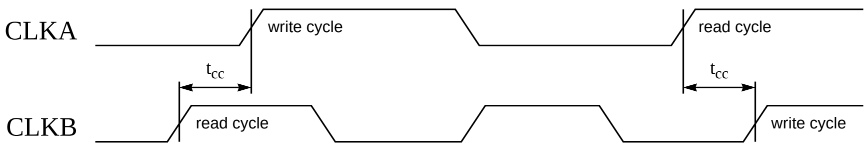
Figures 3-22 through 3-24 show clock timing diagrams for Advantagesynchronous dual-port SRAMs. Standard rising/falling delays and slews percentages are shown in these diagrams. Some generators may be designed with different percentages. Check a GUI generated postscript datasheet to verify the delay and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

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



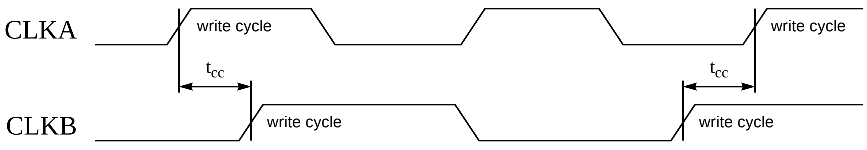
Rising delays are measured at 50% VDD and falling delays are measured at 50% VDD.
Rising and falling slews are measured from 10% VDD to 90% VDD.

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 3-20 illustrates read and write behavior during clock contention, when both ports access the same address.

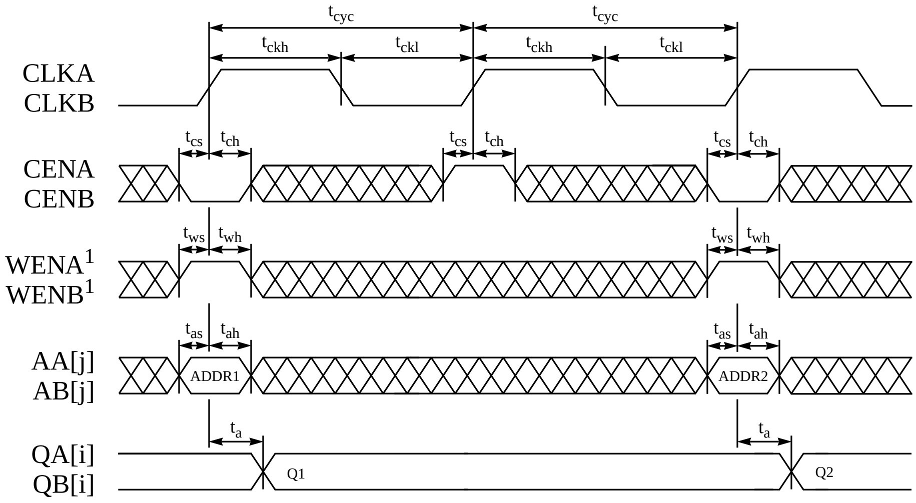
Table 3-20. Dual-Port SRAM Read and Write Behavior When Accessing Same Address

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

3.3.7.2 Dual-Port SRAM Timing Diagrams (sram_dp_adv)

Figures 3-25 and 3-26 show timing diagrams for Advantage 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.

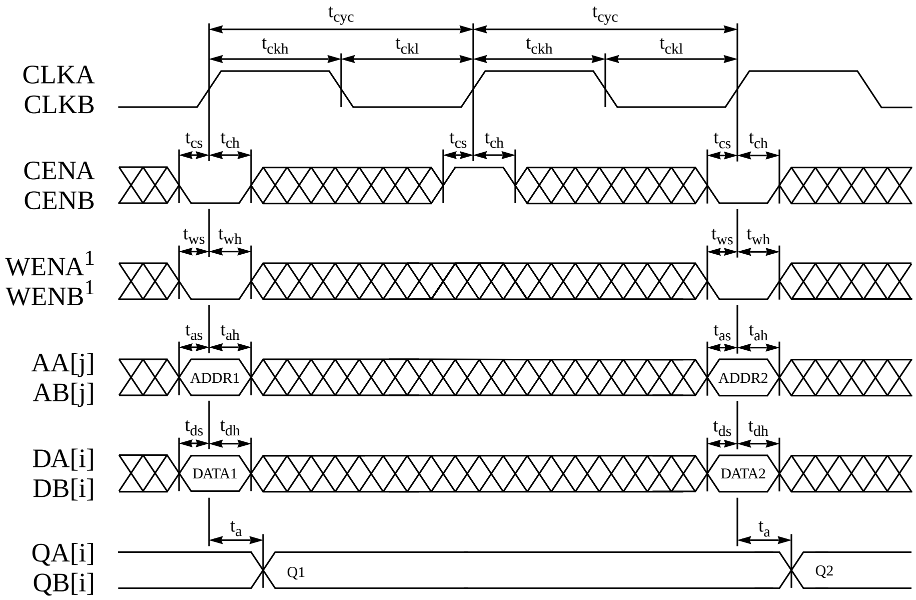
Figure 3-25. Dual-Port SRAM Read-Cycle Timing



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.

[1] 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.

Figure 3-26. 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.

[1] 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.

3.3.7.3 Dual-Port SRAM Timing Parameters (sram_dp_adv)

The GUI generated postscript datasheets and the ASCII datatable contain timing parameters listed in Table 3-21.

Table 3-21. Dual-Port SRAM Timing Parameters

Parameter	Symbol
Cycle time	t_{cyc}
Access time[1,2]	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}
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}
Access time, EMA is enabled: eight numbers for eight values of EMA[1,2]	$t_{a[0-7]}$
Address setup, test pin	t_{tas}
Address hold, test pin	t_{tah}
Chip enable setup, test pin	t_{tcs}
Chip enable hold, test pin	t_{tch}
Write enable setup, test pin	t_{tws}
Write enable hold, test pin	t_{twh}
Data setup, test pin	t_{tds}
Data hold, test pin	t_{tdh}
Test enable setup	t_{tens}

Table 3-21. Dual-Port SRAM Timing Parameters (Continued)

Parameter	Symbol
Test enable hold	t_{tenh}
Bypass enable setup	t_{bens}
Bypass enable hold	t_{benh}
Extra margin enable pin setup	t_{emas}
Extra margin enable pin hold	t_{emah}
Column redundancy enable setup	t_{crens}
Column redundancy enable hold	t_{crenh}
Redundant column address setup	t_{rcas}
Redundant column address hold	t_{rcah}
Row redundancy enable setup	t_{rrens}
Row redundancy enable hold	t_{rrenh}
Row redundancy address setup	t_{rras}
Row redundancy address hold	t_{rrah}
Load dependence factor on data output (ns/pF)	load_q
Load dependence factor on chip enable MUX output (ns/pF)	load_ceny
Load dependence factor on write enable MUX output (ns/pF)	load_weny
Load dependence factor on address MUX output (ns/pF)	load_ay
Load dependence factor on data MUX output (ns/pF)	load_dy

[1] The ASCII datatable and postscript datasheet shows fixed delay values. These parameters have a load dependence (K_{load}), which is used to calculate: $\mathrm{TotalDelay} = \mathrm{FixedDelay} + (\mathrm{Kload} \times \mathrm{Cload})$, for timing views.

[2] 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 (sram_dp_adv)

The GUI contains an ASCII datatable that provides characterization values for each corner. Table 3-22 shows the dual-port SRAM power parameters.

Table 3-22. Dual-Port SRAM Power Parameters (changes pending)

Parameter	Symbol
AC Current[1, 4]	i_{cc}
Read AC Current[4]	i_{cc_r}
Write AC Current[4]	i_{cc_w}
Peak Current[4]	i_{cc_peak}
Deselected Current[2]	i_{cc_desel}
Standby Current[3]	$i_{cc\_standby}$
AC Current: eight numbers for eight values of EMA	$i_{cc[0:7]}$
Read AC Current: eight numbers for eight values of EMA	$i_{cc\_r[0:7]}$
Write AC Current: eight numbers for eight values of EMA	$i_{cc\_w[0:7]}$

[1] 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.

[2] 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.

[3] Value is independent of frequency and assumes all inputs and outputs are stable.

[4] For most generators, value shows dynamic current without leakage (standby) component. Refer to 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. Refer to “Current Calculations” on page 3-58 for instructions on recalculating the current for a specific design. Refer to “Noise Limits” on page 3-60 for more information related to power considerations.

3.4 SRAM Power Structure (sram_sp_adv, sram_dp_adv)

The following sections explain the power structure options available with single- and dual-port, Advantage SRAM generators for both the Ring and ArtiGrid options.

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,

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

From the datasheet,

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

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: 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,

$$I_p = \mathrm{icc\_peak}$$

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 suppling 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 (µm);

W_{iravg} = connection width based on average voltage (µm);

W_{irp} = connection width based on peak voltage (µm);

C = current density rule constant (mA/µm);

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

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

ΔV_{iravg} = allowable average voltage drop within the power wires on the chip (mV);

ΔV_{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 (µm);

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

W = connection width (µm);

we have:

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

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

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

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

☞ 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 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 Ring Power Structure Options (sram_sp_adv, sram_dp_adv)

Ring-based power routing is the default power methodology for the SRAM.

3.5.1 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 (μm);

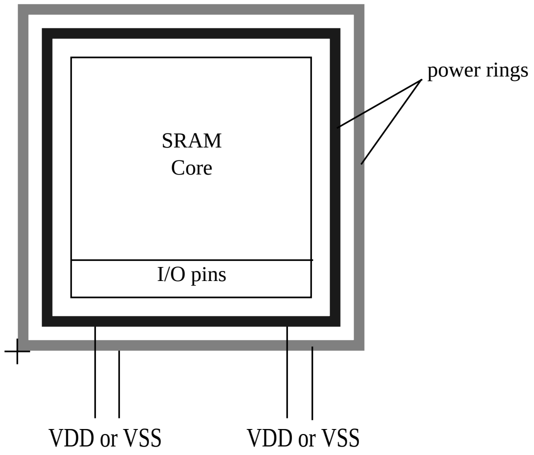
n = number of equal-width-wire power connections at the I/O pins;

we have:

W_m = ring width with n connections (μm) = $1 / n \bullet W_r$

Figure 3-27 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. The outside ring will then have the opposite polarity.

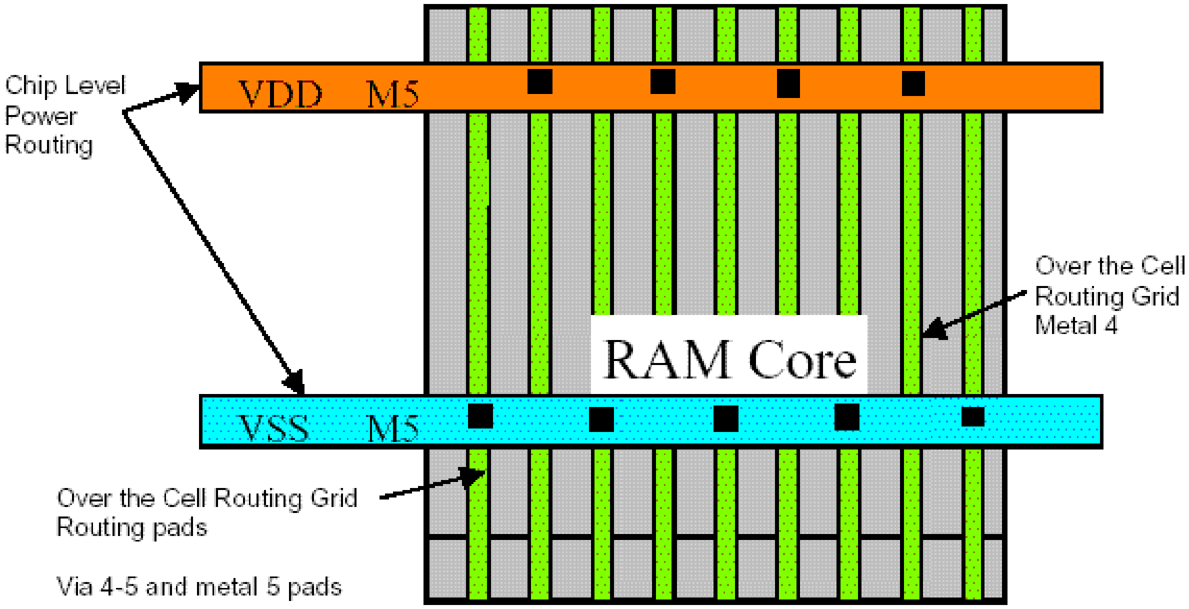
Figure 3-27. Multiple Power and Ground Connections



3.6 ArtiGrid Power Structure Options (sram_sp_adv, sram_dp_adv)

When the ArtiGrid option is selected from the GUI or command line, vertical Metal4 straps of VDD and GND is visible to the chip level routing tools.

Figure 3-28. ArtiGrid Schematic



Industry standard routing tools can be used to connect the vertical Metal4 straps to Metal5 or higher chip level power grid. ArtiGrid for SRAMs will have the vertical M4 straps over the instance.

You must route chip level VSS and VDD to the memory instance and drop vias down to the Metal4 straps. Take care to connect to every observed Metal4 power and ground strap.

3.7 SRAM Physical Characteristics (sram_sp_adv, sram_dp_adv)

This section provides physical design characteristics for single- and dual-port, Advantage SRAM generators. Information such as top metal layer usage, I/O pin connections, ArtiGrid/Over-the-Cell (OTC) routing, and characterization environments is also provided in this section. All information applies to both ring power and ArtiGrid power structured SRAM generators.

3.7.1 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.

3.7.2 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.

EMA pins are exception to the usual pin placement locations. These pins are located at the top edge of the memory.

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.

In some configurations it may not be possible to fit all the pins on metals1-4 using wide sizes. In these cases, wide pins are allowed on metals1-3. If neither of these cases are possible due to pin-density, the thin pins are created on metals1-3 with at least one routing pitch spacing between the pins.

3.7.3 Characterization Environments

By default, the generator is characterized as fast (ff), typical (tt), and slow (ss) operating conditions or corners, and also at a high leakage corner. 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.

3.8 SRAM Timing Derating (sram_sp_adv, sram_dp_adv)

Derating factors are coefficients that the characterization data is multiplied by in order to arrive at timing data that reflects different operating conditions. Standard 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, 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

4.1 Overview

This chapter lists EDA tools supported by the generator and various tools used for verification. These details apply to most standard 90nm Advantage SRAM generators unless otherwise noted.

4.2 Tool Verification

The Advantage SRAM generators produce a set of views and models that are verified with the tools defined in ARM's Artisan EDA Packages. The views/models as well as the tools used for verification are listed in Table 4-1. As needed, refer to the README file in your generator to determine the EDA tool version(s) used to verify your generator.

Table 4-1. Tools Used for Verificatioin .

View and provided file name	Tool	Vendor
Standard Support		
Verilog (.v)	NC-Verilog	Cadence
	VCS	Synopsys
	ModelSim (Verilog)	MTI/Mentor
VHDL (.vhd)	NC-SIM (VHDL)	Cadence
	ModelSim (VHDL)	MTI/Mentor
Repair and SER Verilog (_rtl.v)	NC-Verilog	Cadence
	Design Compiler	Synopsys
Repair and SER VHDL (_rtl.vhd)	ModelSim (VHDL)	MTI/Mentor
	Design Compiler	Synopsys
Synopsys (Liberty) (.lib)	Design Compiler	Synopsys
VCLEF footprint (.vclef)	SOC Encounter	Cadence
GDSII and LVS netlist	Calibre	Mentor
Optional Support		
MBISTArchitect (.mbist)	MBISTArchitect	Mentor
IC Memory BIST(.memlib)	IC Memory BIST	LogicVision
TetraMAX (.tv)	TetraMAX Model	Synopsys
FastScan (.fastscan)[1]	FastScan Model	Mentor
GDSII and LVS netlist	Hercules	Synopsys
GDSII and LVS netlist	Assura	Cadence

[1] Tetramax and Hercules support is available for free to ARM's Artisan Access (Free) Library Program licensees under ARM's Artisan EDAPlus programs with EDA partners.

4.3 Using the Generator Views

This section provides information about simulating design modules that use views provided by the 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:

```
vcs <name>.v

simv
```

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:

```
vcs <test-bench>.v

simv > verilog.log
```

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 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 *Modelsim* 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 *Modelsim* window, select *Signals*.

To view the waveforms of the signals in the design, select the *Wave* pull-down menu in the *Modelsim Signals* pop-up window, and then select *Signals in Design*. Next, select the *View* pull-down menu in the *Modelsim* window, and select *Waves*. The *Modelsim 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 *Modelsim Signals* pop-up window, and then select *Signals in Design*. Next, select the *View* pull-down menu in the *Modelsim* window, and select *List*. The *Modelsim 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 Repair and SER Verilog/VHDL Models

The repair RTL consists of a hierarchical design that incorporates user-selected SER and FR functionalities.The top-most hierarchy consists of the SER design module that, in turn, encloses a subsequent hierarchythat contains FR design modules. The Repair RTL has a Repair Bus (RB) that needs to be connected to FUSE box. The details of this can be obtained from the "90nm Advantage Memory Generator Test and Repair Features Application Note," and the header of the Repair RTL. The Repair RTL has a pin "TEST_MODE_ATPG[A/B]" that, when set to logic-1, causes the CREN[A/B] bus to go to logic-1. This should be set to logic-1 when you run the ATPG tool; for example, Tetramax.

4.3.3.1 Row Redundancy

- RRE1 (RRE2): Row redundancy enable for 1st (2nd) redundant row, if set to logic 1 then row redundancy is ON for 1st (2nd) row.
- FRA1 (FRA2): Faulty row address for 1st (2nd) redundant row. This has the logical address of the row.

4.3.3.2 Column Redundancy

- CRE1 (CRE2): Column redundancy enable, if set to logic 1, then 1st (2nd) column redundancy is ON.
- FBA1 (FBA2): Faulty Bit Address for 1st (2nd) column, this bus encodes the index of the BIT in the faulty word.
- FCA1 (FCA2): Faulty Column Address for 1st (2nd) column, this bus encodes the logical address of the faulty column in the faulty bit.

4.3.3.3 Repair Verilog/VHDL Simulation

You must link the Verilog/VHDL models for the memory generator, fuse box, and standard cells with the Repair Verilog/VHDL model for simulation. The Repair bus can be driven with different logical values to test different fuse values. Similarly, other design inputs can be driven to test for design functionality.

4.3.3.4 Repair Verilog Synthesis

You must link the .lib and .db models for the memory generator, fuse box, and standard cells with the Repair Verilog/VHDL model. Set the constraints for the synthesis tool to meet your design needs and synthesize the design.

4.3.4 Using the Synopsys (Liberty) Model to Generate SDF

After generating the Synopsys (Liberty) .lib model, you can generate SDF using these steps.

Invoke Synopsys tools:

```
     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 file, <`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 file 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 SRAM. In SDF, if the hold time is a negative number, it is set to zero.

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

4.3.5 Loading the VCLEF Description into SOC Encounter

After generating VCLEF, load it into SOC Encounter using these steps.

Invoke SOC Encounter.

Bring up the “Design Import” window.

Perform either task “a” or “b” described below.

a. Enter the VCLEF filename and the LEF technology header filename along with other required inputs, such as Verilog netlist of the design.

b. Prepare a config file as shown in the following example and type

```
load config <config_file> 1
```

in the command window.

Example config file:

```
global_rda_Input
set rda_Input(ui_netlist) “testRoute.v”
set rda_Input(ui_netlisttype) {Verilog}
set rda_Input(ui_settop) {1}
set rda_Input(ui_topcell) {testRoute}
set rda_Input(ui_leffile) “<tech>.lef <name>.vclef”
set rda_Input(ui_pwrnet) {VDD}
set rda_Input(ui_gndnet) {VSS}
set rda_Input(ui_pg_connections) [list {PIN:VDD:} {PIN:VSS:}]
set rda_Input(PIN:VDD:) {VDD}
set rda_Input(PIN:VSS:) {VSS}
```

This creates the SOC Encounter database necessary for floor planning, placement, and routing; `<tech>` is the technology LEF and `<name>` is the memory instance name.

4.3.6 Using Astro with ARM's Artisan Generators

In order to use ARM's Artisan generator with the Synopsys Astro 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 "`read_lef`" 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 by running "`read_lef`" to generate a FRAM view
3. Import the GDSII into Milkyway

It is important to run "`read_lef`" before importing the GDSII.

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

4.3.6.1 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>`.

Start Milkyway. On the command line for Milkyway, use the Synopsys LEF reader `read_lef` or, from the menu, select `Cell Library > Lef In...`

A form is displayed. Fill this form with the appropriate entries for library name, .lef file name, etc., then click on OK to create the CEL view and FRAM view.

4.3.6.2 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. On the command line, use `auStreamIn` or, 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.

4.3.7 Loading the GDSII Layout into a DFII Library

After generating the GDSII layout, you can load it into a SRAM 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 SRAM 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.

4.3.8 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 Assura, Mentor Graphics Calibre, or Synopsys 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, (i.e., there is no short- or open-circuit caused by a place-and-route or other tool).

4.3.8.1 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:

```
                 Layout                        Schematic
                   ↓                               ↓
<instance_name>             <instance_name>
<instance_name><cellname>  <instance_name><cellname>
```

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 the Mentor Graphics web site.

Index