



# **SiWare Automotive Grade 1 Single Port High Density and Performance Leakage Control SRAM 2M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process**

## **User Manual**

---

*DesignWare® Embedded Memories  
gf22nsd41p11s1dcl02ms*

## Copyright Notice and Proprietary Information

© 2020 Synopsys, Inc. All rights reserved. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

### Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

### Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

### Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <https://www.synopsys.com/company/legal/trademarks-brands.html>

All other product or company names may be trademarks of their respective owners.

### Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

### Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.  
[www.synopsys.com](http://www.synopsys.com)

# Revision History



## Note

Links and references to section, table, figure, and page numbers in this table are only assured to be valid for the version in which the change is made.

Date	Document Version	Description
July 2019	A01	Document for A01 FE/LEF compiler release.
March 2020	A02	Document updated for template changes; supports A02 Full GDS compiler release. Part number changed from gf22nsd41p11sadcl02ms to gf22nsd41p11s1dcl02ms. Updated operating characterization conditions information as per PVTs updated in the compiler. Added information on Enable Bit Grouping. Updated information on LTT support to reflect XLTT. Removed characterization data to comply with revised documentation requirements.
July 2020	A03	Document to support A03 Full GDS compiler release. Updated RA and WA pin settings.
December 2020	A04	Document to support A04 Full GDS compiler release. Updated VMIN voltages for RM modes. Updated operating characterization conditions information as per PVTs added in the compiler.



# Contents

---

Revision History .....	3
About this Manual .....	1
Chapter 1	
Product Line Overview .....	3
1.1 SiWare™ High-Density Memory Compilers .....	3
1.1.1 Key Features .....	3
1.1.2 Compiler Modes .....	4
1.2 Compiler Features .....	12
1.2.1 Power Management .....	12
1.2.2 Dual Rail Functionality .....	13
1.2.3 BIST Interface .....	13
1.2.4 Redundancy .....	14
1.2.5 Read Pipeline .....	15
1.2.6 Self Time Bypass .....	15
1.2.7 Read Margin Control .....	15
1.2.8 Synchronous Write Through .....	17
1.2.9 Bit Write .....	17
1.2.10 Enable Bit Grouping .....	18
1.2.11 Enable Bitwise Corruption .....	18
1.2.12 Enable Bitwise Memory Corruption .....	18
1.2.13 ME Gating .....	18
1.2.14 Back Bias .....	19
1.2.15 Performance Boosters .....	20
1.2.16 Optional Periphery Transistor Threshold Voltage Selection .....	20
Chapter 2	
Compiler Profile .....	23
2.1 Compiler Naming Convention .....	23
2.2 Compiler Features and Benefits .....	24
2.3 Specifications .....	26
2.3.1 Memory Symbol .....	26
2.3.2 Pin Description .....	28
2.3.3 Block Diagram .....	34
2.3.4 Functional Options .....	35
2.3.5 Functional Descriptions .....	36
2.3.6 Compiler Range Information .....	37
2.3.7 Metal Layer Usage .....	38
2.3.8 Signal Pin and Power Routing .....	40

2.3.9 IP Tagging .....	43
2.3.10 Physical Grids .....	43
2.3.11 Logic Truth Tables .....	43
2.3.12 Hazard Information .....	47
2.3.13 Waveform Diagrams .....	48
2.3.14 Standard Operating Characterization Conditions .....	56
2.3.15 Using Licensed Custom PVTs .....	57
<b>Chapter 3</b>	
Compiler Source and Output Files .....	59
3.1 Using Compiler Library Files .....	59
3.2 Accessing Output Files .....	61
3.2.1 compout Directory .....	61
3.2.2 compout/views/instance_name/<pvt_name> Subdirectory Files .....	63
<b>Chapter 4</b>	
Using Front-End Files .....	65
4.1 Using Template Files .....	65
4.1.1 Using ATPG Tools .....	66
4.1.2 Using MemoryBIST Tools .....	66
4.1.3 Performing Power Analysis .....	67
4.1.4 Performing Simulations .....	67
4.1.5 Performing Static Timing Analysis .....	68
4.1.6 Using Switch-Level Verilog Models .....	69
4.1.7 Power Verilog Models .....	69
4.1.8 Low Power Formats .....	70
4.2 Setting Front-End Flags .....	70
<b>Chapter 5</b>	
Using Back-End Views .....	71
5.1 Using LEF Models .....	71
5.2 Using GDSII .....	72
5.2.1 Naming GDSII Libraries .....	72
5.2.2 Using the Extended Layer Translation Table File .....	72
5.3 Using SPICE Netlists .....	72
5.4 Reading the Bitmap (Scramble) Information .....	73
<b>Chapter 6</b>	
Configuring Files .....	75
6.1 Expanded SiWare™ Instance Naming Convention .....	75
6.2 Configuring the Custom GLB File .....	78
6.2.1 Setting Parameters .....	78
6.2.2 Generating Pins .....	79
6.3 Configuring the Extended Layer Translation Table File .....	79
6.4 Place and Route Considerations .....	79
<b>Index</b> .....	81

# About this Manual

---

This manual shows how to use Synopsys' DesignWare Embedded Memories. It describes an overview of Synopsys' memories, compiler profile, the compiler directory structure, Front-End files, Back-End files, and user-configurable files.

## Audience

This manual is intended for design engineers and customer support engineers assumed to be familiar with integrated circuit design technology.

## Using this Manual

This manual is organized into the following chapters:

Chapter	Describes
<a href="#">Chapter 1, "Product Line Overview"</a>	The key features of the memory compiler in this product line
<a href="#">Chapter 2, "Compiler Profile"</a>	The functional features and specifications of the compiler
<a href="#">Chapter 3, "Compiler Source and Output Files"</a>	The directory structure of the compiler database, files, and templates
<a href="#">Chapter 4, "Using Front-End Files"</a>	The files in the Front-End views
<a href="#">Chapter 5, "Using Back-End Views"</a>	The Back-End views files that support layout and fabrication
<a href="#">Chapter 6, "Configuring Files"</a>	How to configure the custom global (GLB) file, the Extended Layer Translation Table file (XLTT), and the Power Place and Route (PPR) file

## Minimum Operating System Requirements

- 1 gigabyte main memory
- 1 gigabyte of disk space per compiler to support installation of the compiler and generation of at least one instance.



### Note

Refer to the Release Notes for Embed-It!® for a list of hardware and software supported.

---

## Synopsys' Product Releases

Synopsys, Inc. releases a new version of pre-silicon products (version A) periodically. These products incorporate Synopsys Technical Action Request (STAR) that resolve bugs and provide product enhancements. To address high-priority change requests, Synopsys, Inc. also provides a patch release between quarterly updates when necessary. A patch release is denoted by the suffix  $pn$  where  $n$  is a release number such as 1, 2, or 3 (p1, p2, or p3). Refer to the **release.txt** file in the **complib** directory for a list of release dates.

**Note**

License files need not be updated for patch releases.

## Related Documentation

The following documents provide more information about software used with the memory compilers.

For information on	See
How to use Synopsys' Embed-It! Integrator software interface to generate memory components.	<a href="#">Embed-It!® Integrator User Guide</a>
Mentor Graphics FastScan	<a href="http://www.mentor.com/dft/fastscan_ds.pdf">http://www.mentor.com/dft/fastscan_ds.pdf</a>
Synopsys TetraMAX	<a href="http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Test/Pages/TetraMAXATPG.aspx">http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Test/Pages/TetraMAXATPG.aspx</a>

## Getting Technical Assistance

For technical support with Synopsys' products, please contact the Synopsys Support Team at <http://solvnetplus.synopsys.com>.



## Product Line Overview

---

This chapter provides an overview of Synopsys' DesignWare Embedded Memories and the key features of *22nm FD SOI SiWare™ High-Density Compilers*.

### 1.1 SiWare™ High-Density Memory Compilers

Synopsys' 22-nanometer (nm) FD SOI SiWare product family of memory compilers provides a powerful dashboard of options that enables System-on-Chip (SoC) designers to explore trade-offs between performance, area, power, and statistical yield to generate optimal memory configurations. This dashboard control capability is critical at 22nm FD SOI where design and process complexities require sophisticated management of the various trade-offs to effectively meet stringent end-product requirements and increasingly narrow time-to-market windows.

#### 1.1.1 Key Features

Synopsys' SiWare™ High-Density products provide ASIC vendors and designers with:

- Memory instances optimized for area, without compromising quality.
- Memory compilers that leverage the standard foundry delivered bit cells to ensure high yield and reliability.
- Memory compilers that provide an option of redundancy capabilities for repair purposes.
- Memory compilers with options for advanced power management modes, such as Light Sleep, Deep Sleep, and Shut Down.
- Memory compilers provide the provision for Back Biasing to boost performance.
- Front-End and Back-End views for integration into EDA tools from leading tool suppliers.

The SiWare High-Density memory compilers are optimized to generate memories with the absolute minimum area and power to enable designers to achieve aggressive critical path requirements. Detailed timing parameters can be found in instance datasheets.

SiWare SRAM compilers offer multiple levels of power management with Light Sleep, Deep Sleep, and Shut Down pins to enable biasing, partial periphery shutdown, full periphery shutdown with data retention, and a complete shutdown without data retention. An option for Dual Rail is also available to support Dynamic Voltage Frequency Scaling (DVFS).

Synopsys' STAR Memory System (SMS) enables cost-effective embedding, testing and repairing of SiWare memories. The integrated test and repair capability ensures higher yielding semiconductors, and can potentially save millions of dollars in recovered silicon, substantially reduce test costs, and enable faster time to volume production.

Synopsys, Inc. provides power saving design techniques to implement these features:

- Source Biasing
- Fine-grained Power Gating
- Use of Long channel devices
- Integrated Power Gating
- Integrated Level Shifters

### 1.1.2 Compiler Modes

The SiWare High-Density memory compilers offer six configuration modes, some of which provide features to support Test and Repair. Each of these modes may be combined with advanced power management. They are defined as below:

- SiWare Lite (SiWare-LT) mode without Built-in test muxes or repair.
- SiWare Lite Redundancy (SiWare-LR) mode that adds redundancy to SiWare-LT mode. This mode enables repair solutions at the end of customers. Signature registers are integrated in the memory macro and remain in ALWAYS\_ON state when in a power management mode.
- SiWare Integrated Test Lite (SiWare-IT-Lite) mode that incorporates fully scannable input and output signals, pipelined output QP and Synchronous data flow with external clock.
- SiWare Integrated Test (SiWare-IT) mode that incorporates built-in test muxes and other test circuitry to enable at-speed testing along with fully scannable input and output signals, pipelined output QP and Synchronous data flow with external clock.
- SiWare STAR Lite (SiWare-ST-Lite) mode that adds redundancy to SiWare-IT-Lite.
- SiWare STAR (SiWare-ST) mode that adds redundancy to SiWare-IT.

Table 1-1 shows the different modes with their respective flag settings.

**Table 1-1 SiWare High-Density Memory Options**

Configuration	Parameters Required
SiWare-LT (default configuration)	redundancy_enable=FALSE bist_enable=FALSE scan_enable=FALSE pg_enable=FALSE
SiWare-LR (LT + Redundancy)	redundancy_enable=TRUE row_redundancy=TRUE/FALSE bist_enable=FALSE scan_enable=FALSE pg_enable=FALSE
SiWare-IT-Lite (LT + Scan chains + SWT + Pipeline)	redundancy_enable=FALSE bist_enable=FALSE scan_enable=TRUE pg_enable=FALSE
SiWare-IT (IT-Lite + BIST Muxes + Comparator)	redundancy_enable=FALSE bist_enable=TRUE pg_enable=FALSE
SiWare-ST-Lite (LR + Scan chains + SWT + Pipeline)	redundancy_enable=TRUE row_redundancy=TRUE/FALSE bist_enable=FALSE scan_enable=TRUE pg_enable=FALSE
SiWare-ST (ST-Lite + BIST Muxes + Comparator)	redundancy_enable=TRUE row_redundancy=TRUE/FALSE bist_enable=TRUE pg_enable=FALSE



**Note**

All the modes described in the table above are also possible with Advance Power Management (pg\_enable=TRUE).

Figure 1-1 through Figure 1-6 show a graphical representation of the above mentioned modes.

**Figure 1-1 SiWare-LT**

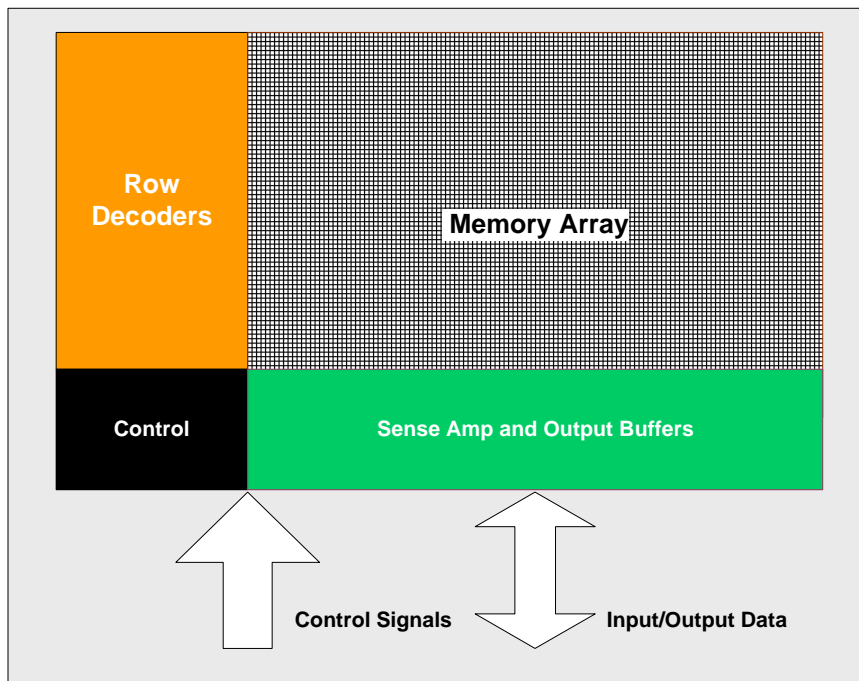
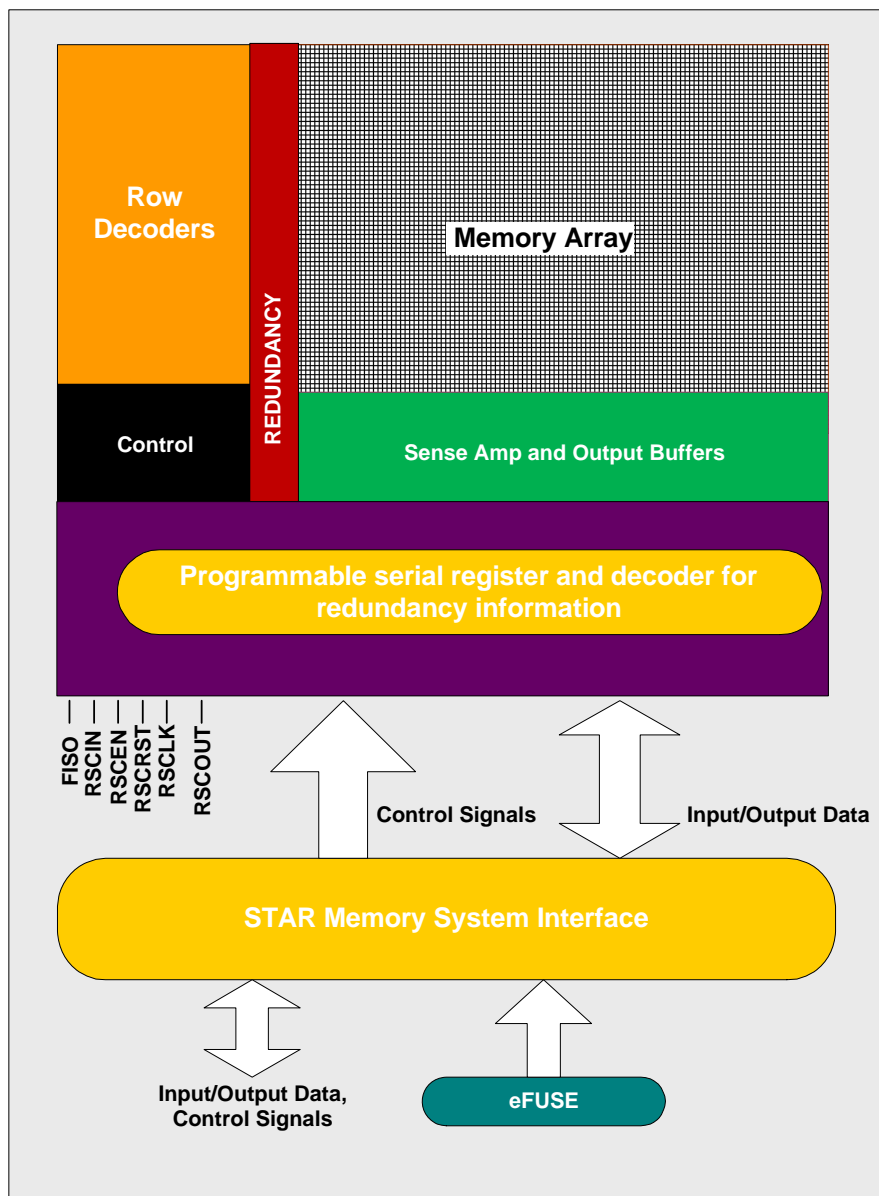
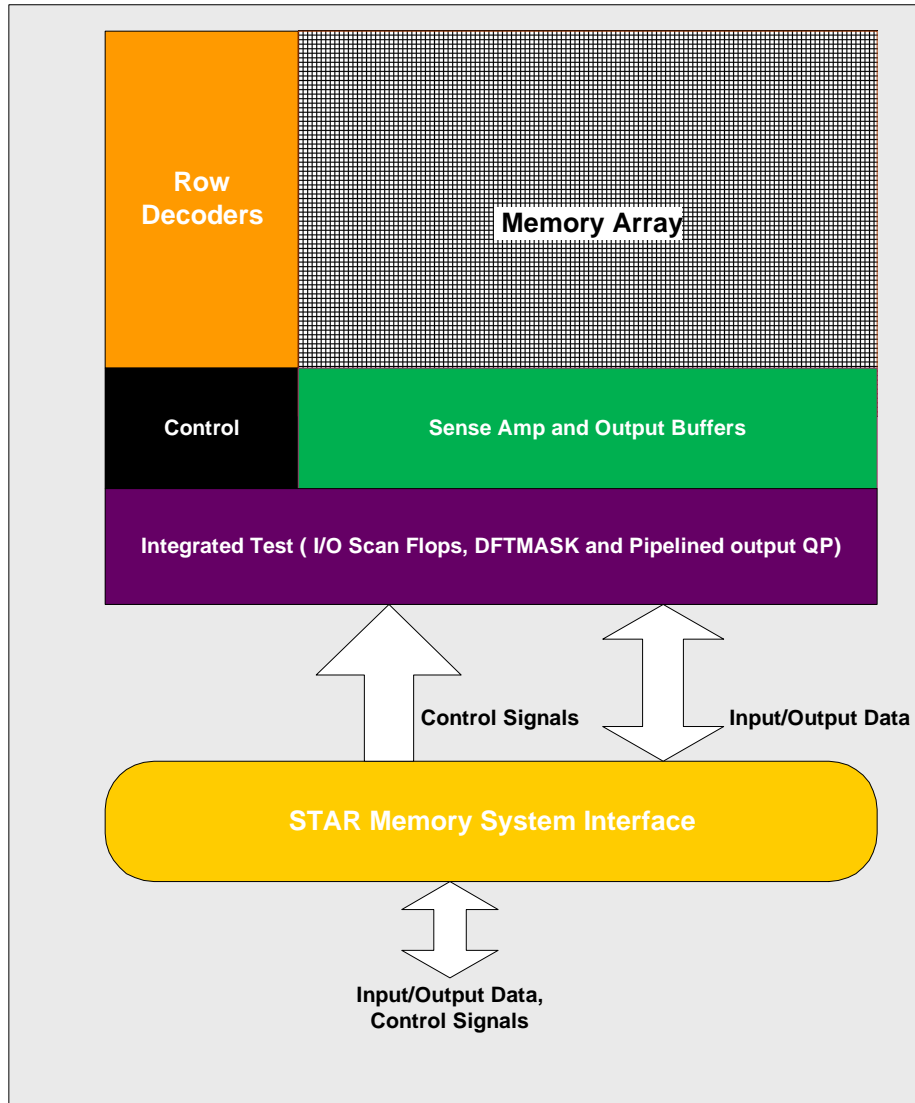
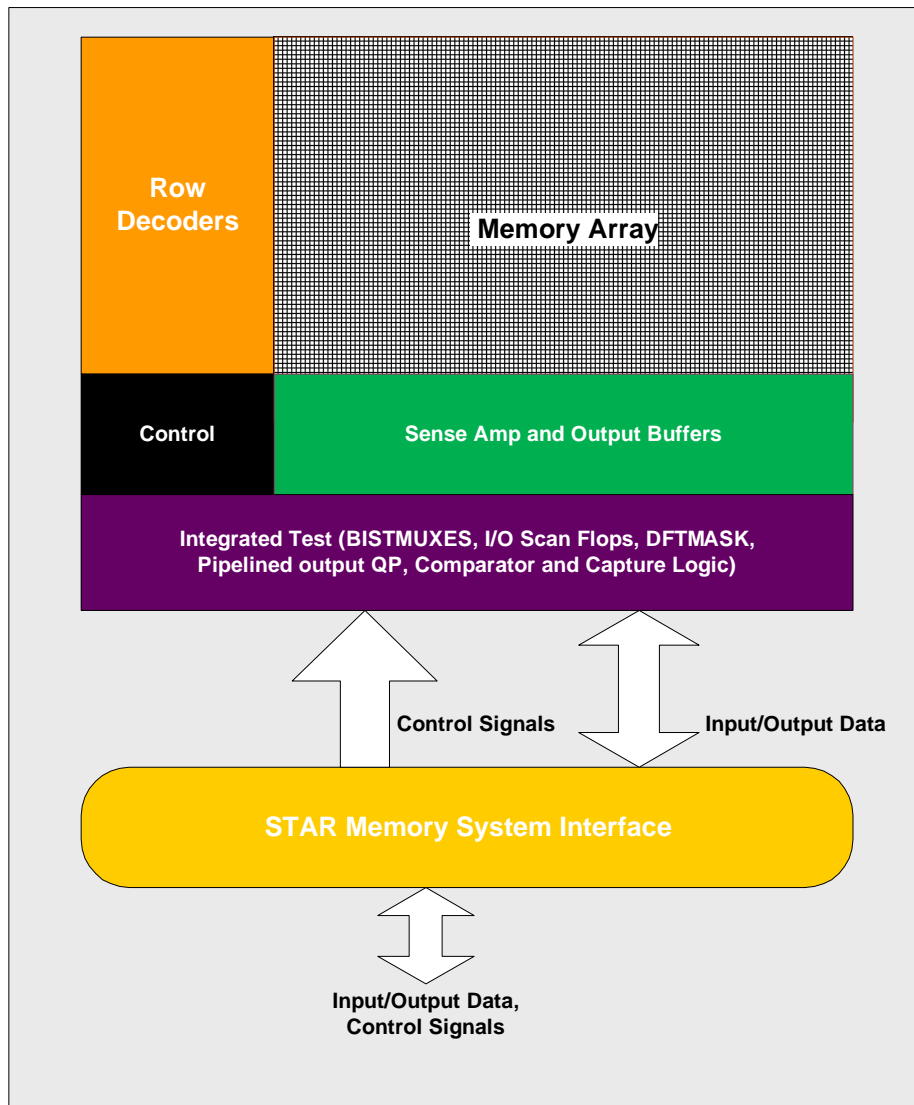


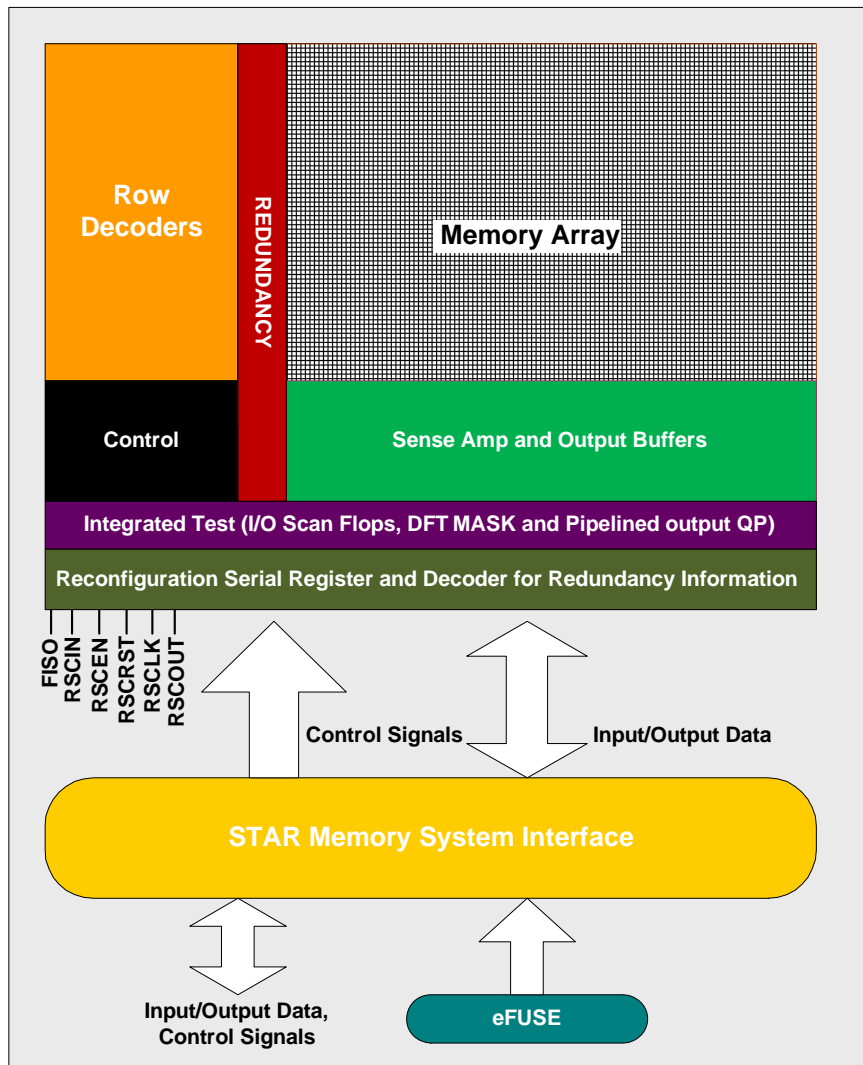
Figure 1-2 SiWare-LR



**Figure 1-3 SiWare-IT-Lite**

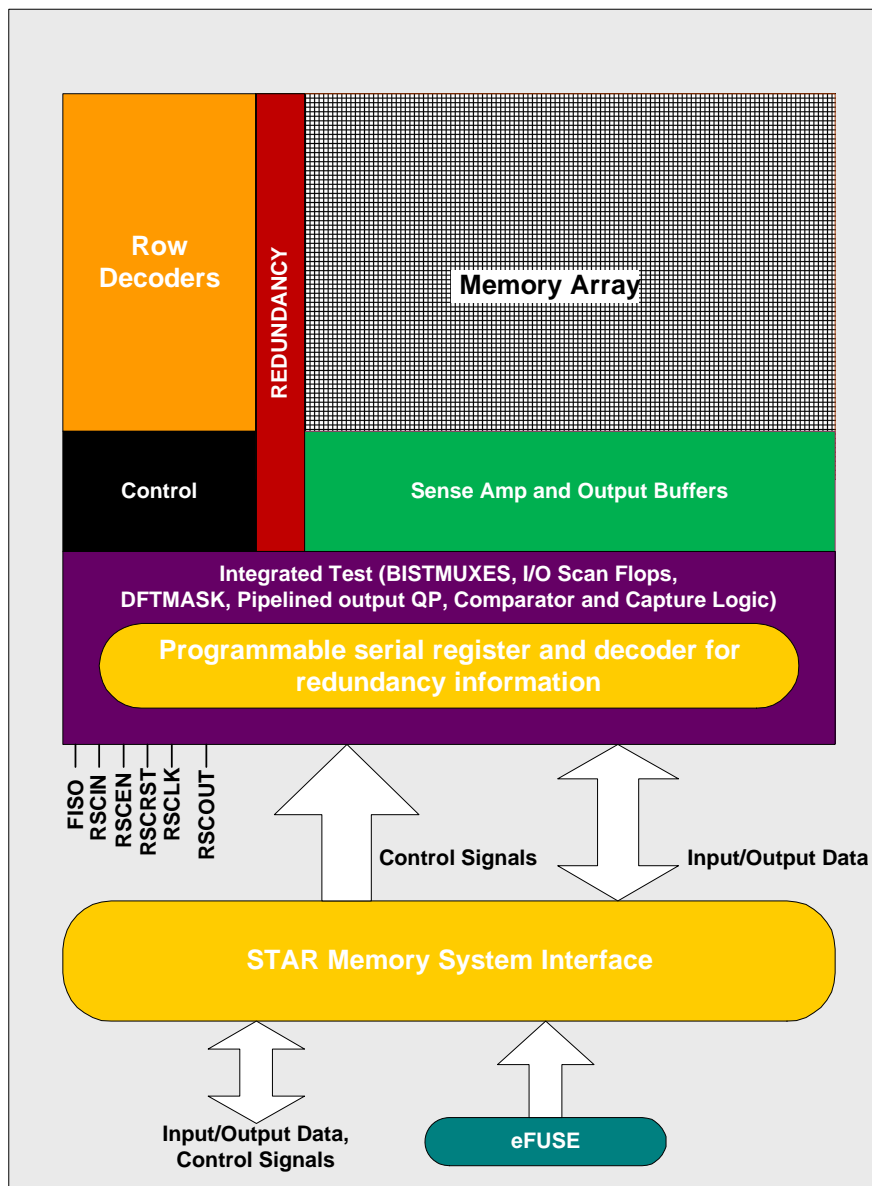
**Figure 1-4 SiWare-IT**



**Figure 1-5 SiWare-ST-Lite**



**Figure 1-6 SiWare-ST**



## 1.2 Compiler Features

The High-Density compilers provide the following features:

- “Power Management” on page 12
- “Dual Rail Functionality” on page 13
- “BIST Interface” on page 13
- “Redundancy” on page 14
- “Read Pipeline” on page 15
- “Self Time Bypass” on page 15
- “Read Margin Control” on page 15
- “Synchronous Write Through” on page 17
- “Bit Write” on page 17
- “Enable Bitwise Corruption” on page 18
- “Enable Bitwise Memory Corruption” on page 18
- “ME Gating” on page 18
- “Back Bias” on page 19
- “Performance Boosters” on page 20
- “Optional Periphery Transistor Threshold Voltage Selection” on page 20

### 1.2.1 Power Management

The 22nm FD SOI compilers support instance generation in the following power management modes:

- Standard - default mode
- Advanced - memory with integrated power gating (IPG)

The Standard and Advanced power management options are controlled by the `pg_enable` flag. When power gating is enabled, the memory compilers will support power downs with data retention, and all the outputs will be held low. See [Table 1-2](#) for further details.

**Table 1-2 Power Management Modes**

Power Management Compiler Setting (pg_enable)	Product Mode	Power Management Plus
FALSE (default value)	Standard	<b>LS (Light Sleep)</b> - Provides leakage reduction with fine-grained power gating and source biasing. Data Retention is valid for voltages greater than Vnom-10%.

**Table 1-2 Power Management Modes**

Power Management Compiler Setting (pg_enable)	Product Mode	Power Management Plus
TRUE	Advanced (IPG)	<p><b>LS (Light Sleep)</b> - Provides leakage reduction with fine-grained power gating and source biasing. Data Retention is valid for voltages greater than Vnom-10%.</p> <p><b>DS (Deep Sleep)</b> - When the DS pin is asserted, integrated periphery power gating with data retention available and the memory outputs are held low. Data Retention is valid for voltages greater than Vnom-10%.</p> <p><b>SD (Shut Down)</b> - When the SD pin is asserted, there is a complete shutdown (both the periphery and array are power gated), with no data retention, and the memory outputs are held low.</p>



**Attention**

A wake-up (Ready for Operation=ROP) pin is also provided in the Deep Sleep and Shut Down modes to manage and control the in-rush peak current of the SoC. It enables memories to be chained together in this mode and allows them to be woken up serially.

## 1.2.2 Dual Rail Functionality

Separate voltage rails for the array and the periphery may be enabled at the instance level, with level shifters in the periphery. This option is controlled by setting the `vdda_enable` GLB parameter. See “[Standard Operating Characterization Conditions](#)” on page 56 for additional information. This option enables the array to be held at a safe operating voltage of  $VDD_{nom} \pm 10\%$ .

Additional pins of POFF[1:0] will also be visible with dual rail option. These pins can be used to put memory in power down mode which can allow periphery supply to turn off completely.

All Power management features are available for dual rail option as well.

## 1.2.3 BIST Interface

The SiWare-IT (without redundancy) and SiWare-ST (with redundancy) options create memory instances that include all of the necessary logic to facilitate at-speed Built In Self Test (BIST). BIST is enabled by setting the `bist_enable` GLB parameter.

When the `bist_enable` option is enabled, the generated memory instance includes multiplexers (muxes) for all address, control and data signals as well as comparators and capture logic. All output signals are fully scannable and the data flows synchronous with the external clock. Incorporating this logic into the memory instance reduces the critical path when BIST is enabled and reduces the number of wires that are required to route between the memory instance and the BIST engine.

The integrated logic will also enable high performance testing of functional logic surrounding the memory in the designs, using ATPG scan tools.

Synchronous Write-through is available when these options are enabled. This allows input data to flow to output pins synchronously with the clock.

Several pins are added to support BIST. Refer to [Table 2-3, “Input Pin Description”](#).

## 1.2.4 Redundancy

The SiWare-ST option enables the memory compiler to generate memory instances that include redundancy for repair. Redundancy provides additional memory to the instance to be used when BIST diagnostics determine that a repair is necessary.

Redundancy is enabled with the `redundancy_enable` GLB parameter.

The SiWare compilers contain two types of redundancy: row and column. When `redundancy_enable=TRUE`, column redundancy is enabled. When `row_redundancy=TRUE`, both row and column redundancy resources are available.



### Note

The `redundancy_enable` parameter is represented as 'Column Redundancy' in the Integrator GUI for SRAM compilers.

The number and configuration of repair elements is dependent on the compiler as described in [Table 1-3](#).

**Table 1-3 Redundancy Repair Elements**

Compiler Architecture	High-Density Single Port SRAM	Ultra High-Density Single Port SRAM	Ultra High-Density 2-Port Register File	1-Port Register File
<b>Column Redundancy Resources</b> ( <code>redundancy_enable=TRUE</code> , <code>row_redundancy=FALSE</code> )	Single element of 4 columns on one side of the center block	Single element of 4 columns on one side of the center block	Single element of 4 columns on one side of the center block	Single element of 4 columns on one side of the center block
<b>Row Redundancy Resources</b> ( <code>redundancy_enable=TRUE</code> , <code>row_redundancy=TRUE</code> )	4 redundant rows independent of banking	4 redundant rows independent of banking	N/A	N/A

Additionally, the compilers allow optimization of reconfiguration register bits resulting in variable length of these bits. This helps in reducing the number of Efuse bits and repair time required by memories. The reconfiguration register length is controlled by GLB parameter `variable_reconfig_reg_len`.

If this flag is set to `TRUE`, instances generated will have variable reconfiguration register length as per instance configuration.

There also happens a change in the MASIS view, and the respective reconfiguration registers count gets reduced there. Information on this section of MASIS can be found in the MASIS v1.2 Memory and SMS Interface Standard document 'masis1.2.pdf.'

If the `variable_reconfig_reg_len` is set to `FALSE`, instances generated will have fixed maximum required reconfiguration register length as original.



#### Note

Users must ensure to keep the same "variable\_reconfig\_reg\_len" setting when generating the instance multiple times such as FEV and BEV to avoid inconsistent SMS and MASIS implementation.

### 1.2.5 Read Pipeline



#### Note

This feature is only available when BIST is enabled.

The DFTCLKEN pin is used for ATPG testing of the memories and is therefore valid only in IT and ST modes. If DFTCLKEN=1, input to flop will be D/TD XOR WEM/TWEM and clock will be applied to scan chains independent of PIPEME. If DFTCLKEN=0, input to flop is memory output, and clock to flop is controlled by PIPEME.

In all cases, in the pipeline mode, the output pin QP will have the pipeline output instead of the output pin Q.

### 1.2.6 Self Time Bypass

In the Self Time bypass mode is controlled by toggling the TEST1 pin. In this mode (TEST1=1), the memory self time circuitry is bypassed. The memory timing is controlled by the external clock signal (CLK). Self Time bypass mode is initiated after the rising edge of CLK and is terminated by the falling edge. The Self Time bypass mode may be used to determine the margin of the internal self-timed circuitry.

### 1.2.7 Read Margin Control

The memory array bitlines are pre-charged prior to a memory cell access. After accessing the memory (Read cycle), a differential signal develops between the bitlines (bitline and bitline bar). This differential signal is fed to the input of the bitline sense amplifier to determine what data is stored in the bitcell. Since it takes time for the differential signal on the bitlines to develop, the greater the time delay prior to strobing the sense amplifier, the greater will be the differential signal at the input to the sense amplifier. A low sense amplifier differential signal is susceptible to noise and sense amplifier input voltage offset. Higher input differential voltage results in greater reliability of the sensed data. However, delaying the time when the sense amplifier is strobed results in a longer cycle time, reducing maximum operating speed and increasing access time (Tcc and Tcq increase). Hence the trade off of memory speed verses yield/reliability.

The longer is the wait, the easier it is for the sense amplifier to determine what was stored in the memory cell. Thus the term *Robustness*. The longer is the wait, the longer it takes to access the cell (i.e., access time). Thus, the term *Speed Tradeoff*.

The **timing\_mode** GLB parameter in the **<memory\_compiler\_name>\_custom.glb** file allows the user to tune the memory instance timing during compile. This parameter enables the selection between high yield and high performance READ/WRITE margin settings. All memories are characterized with six **timing\_mode** settings that control the compile time options available with the compiler. These are described in [Table 1-4](#) and [Table 1-5](#).

**Table 1-4 Timing Mode Options - Single Rail**

Timing Mode	RME pin during chip Operation	RM[3:0]	VDD <sub>nom</sub> (V)	VDD <sub>max</sub> (V)	VDD <sub>min</sub> (V)	Description
RM5	1	4'b0111/4'b0110/4'b0101	0.9	0.945	0.72	Overdrive
RM4	1	4'b0100	0.9	0.945	0.72	Overdrive
RM3	1	4'b0011	0.8	0.945	0.72	Nominal Voltage Operation
RM2	1	4'b0010	0.8	0.945	0.72	Nominal Voltage Operation
RM1	1	4'b0001	0.8	0.945	0.72	Nominal Voltage Operation
RM0	1	4'b0000	0.8	0.945	0.72	Nominal Voltage Operation

**Table 1-5 Timing Mode Options - Dual Rail**

Timing Mode	RME pin during chip Operation	RM[3:0]	VDDA <sub>max</sub> (V)	VDDA <sub>nom</sub> (V)	VDDA <sub>min</sub> (V)	VDD <sub>max</sub> (V)	VDD <sub>min</sub> (V)	Description
RM5	1	4'b0111/4'b0110/4'b0101	0.945	0.9	0.72	0.945	0.59	Overdrive
RM4	1	4'b0100	0.945	0.9	0.72	0.945	0.59	Overdrive
RM3	1	4'b0011	0.945	0.8	0.72	0.945	0.59	Nominal Voltage Operation
RM2	1	4'b0010	0.945	0.8	0.72	0.945	0.59	Nominal Voltage Operation
RM1	1	4'b0001	0.945	0.8	0.72	0.945	0.59	Nominal Voltage Operation
RM0	1	4'b0000	0.945	0.8	0.72	0.945	0.59	Nominal Voltage Operation

The **timing\_mode** GLB parameter enables the selection of process-sigma characterization and read-write margin settings for use during instance generation. All memories are characterized for RM[3:0]=4'b0000 to 4'b0111. The default RM setting for all memories will be RM[3:0]=4'b0011.

The Read Margin RM[3:0] settings are user adjustable. The RM interface to the user is encoded so that all compilers have the same default RM[3:0] values. The user can select from the settings shown in [Table 1-6](#) based on their application requirements. The memory compiler generated RM values are listed in the output file **<memory\_instance\_name>\_params.tcl** located in the **compout/views/<memory\_instance\_name>** directory.

**Note**

Changes to RM[3:0] settings will be reflected in the memory instance timing.

Besides the `timing_mode` GLB parameter, there is an additional GLB parameter as mentioned below:

- `enable_timing_mode_pwr_ds` - When this flag is set to TRUE, it enables publishing of Typical and Worst Power data for each timing mode in instance datasheets.

Any of the static compile time options can be overridden dynamically by using the Read Margin Enable (RME) pin. The RME pin selects between the internal default RM[3:0] value set during compile, and the external, user provided RM[3:0] settings.

There are four Read Margin (RM[3:0]) pins available in Synopsys' DesignWare embedded memories. They are used for DVFS to get Fmax by changing a voltage in different RM. It is required that external access to all RM pins, including RME, is provided for debug and yield analysis purposes. Registers can be used to access the RM pins for debug. If Synopsys' STAR Memory System is used, the RM pins will be automatically registered during compilation.

**Table 1-6 Read Margin Control Pins**

Signal	Description
RME	Read-Write Margin Enable pin. Selects between internal and external RM[2:0]* settings. RME=0: Internal settings; RME=1: External settings.
RM[3:0]	Used to change Read-Write Margin settings.



#### Note

\* RM values specified in the datasheet are the values used in characterization and recommended to set for RM pins. Default RM setting is also hard-coded inside memory and selectable by RME pin (set to low).

## 1.2.8 Synchronous Write Through

Synchronous write-through is used for test coverage improvement. It lets signals travel across the memory and enables the internal memory logic to be visible to the external test circuit. This eliminates the hidden embedded circuit effect of the internal memory logic. The test circuit can be a scan circuit or any other type of circuit. Synchronous write-through will be controlled with the DFTMASK pin. When DFTMASK=1, the output latch will follow the input write data D. When DFTMASK=0 the output will remain in the previous state. Synchronous write-through muxes will be placed before the output latches. The clock going to the output latches must be capable of going to Logic-1, during ATPG mode.

## 1.2.9 Bit Write

The Bit Write feature enables users to independently write to any subset of the memory word. Each bit may be written separately, while retaining the value of other bits before the write operation. The Bit Write feature provides a Write-Enable Mask (WEM) pin for each I/O pin.

Assuming active high control:

- If the WEM pin for an I/O is high, the data on the input pin (D) is written to the corresponding bit cell in the addressed location.



- If the WEM pin for an I/O is low, the data on the input pin (D) is not written to the corresponding bit cell in the addressed location.

There is no loss of power or speed when the Bit Write feature is enabled. However, power will be reduced by disabling writes using these pins. See “[Functional Descriptions](#)” on page 36 for more information about enabling this option.

### 1.2.10 Enable Bit Grouping

The Bit Grouping feature divides the memory IO's in number of sections to optimize separate Data and WEM Setup/Hold timings.

This feature is controlled by the `enable_bit_group` GLB parameter. When this flag is set to TRUE, group of the data and WEM bits will have separate Setup/Hold parameters. The number of group of data and WEM will depend upon the value of number of Memory IO's.

Separate timings will be reported in the instance datasheets for setup and hold parameters which will be grouped in range of NBs according to the sections divided.

### 1.2.11 Enable Bitwise Corruption

The Bitwise Corruption feature is controlled by the `enable_bitwise_corr` GLB parameter. The default value of this flag is set to TRUE.

When this flag is set to TRUE, bitwise corruption in scan chain is enabled in case of ADR/D/WEM/CD setup/hold violations and verilog model supports bit-wise corruption.

If there is a memory with NW=64, that is 6 addresses, and there is a violation on ADR[5], then there will be corruption of scan flops which is driven from ADR[5] in control scan chain, and no other address flop will get corrupted.

If the `enable_bitwise_corr` flag is FALSE, on any ADR bit violation, verilog model will corrupt all the address flops in scan chain. For example, here on ADR[5] violation, all the address flops in the control scan chain will be corrupted.

### 1.2.12 Enable Bitwise Memory Corruption

The Bitwise Memory Corruption feature is controlled by the `enable_bitwise_mem_corr` GLB parameter. The default value of this flag is set to TRUE.

When this flag is set to TRUE, bitwise corruption of memory word is enabled in case of D/WEM/TD setup/hold violations.

If this flag is set to FALSE, violation on single bit of D/WEM/TD would lead to corruption of memory word based on WEM bits CCLWEM.

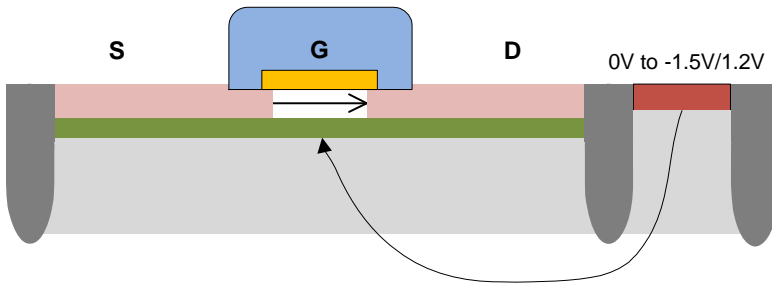
### 1.2.13 ME Gating

ME Gating is controlled by the `me_gating` GLB parameter. When set to TRUE, it helps to reduce the pin toggling powers of ADR/D/WEM/WE pins. ME setup time may increase in this case.



## 1.2.14 Back Bias

FD SOI allows efficient transistor control. This technology enables control of the behavior of transistors not only through the gate, but also by polarizing the substrate underneath the device. The figure below illustrates the feature.



This feature allows user to apply back bias to boost the performance by lowering the  $V_t$  of the devices. It is controlled by the `back_bias_enable` GLB parameter.

Due to transistor construction in FD SOI and its ultra-thin insulator layer, basing is much more efficient. Also, the presence of buried oxide allows the application of higher biasing voltages, resulting in break-through dynamic control of the transistor. The table below shows VNW\_N and VPW\_P Bias range for different voltage domains.

**Table 1-7 VNW\_N and VPW\_P Bias Range for different Voltage Domains**

Domain	Process	Voltage (V)	Temperature (C)	VNW_N (V)	VPW_P (V)	Extraction Corner
Deep Underdrive (0.5V)	SSG	0.45	-40	1.20	-1.50	FuncCmax
	SSG	0.45	125	0.60	-0.85	FuncCmax
Underdrive (0.65V)	SSG	0.59	-40	0.85	-1.50	FuncCmax
	SSG	0.59	125	0.60	-0.80	FuncCmax
Normal (0.8V)	SSG	0.72	-40	0.70	-1.50	FuncCmax
	SSG	0.72	125	0.60	-1.00	FuncCmax
	SSG	0.72	150	0.60	-0.85	FuncCmax
Overdrive (0.9V)	SSG	0.81	-40	0.60	-1.50	FuncCmax
	SSG	0.81	125	0.60	-0.90	FuncCmax



### Note

VNW\_N: Supply name for Nwell back bias for Forward Body Bias (FBB) Schemes  
VPW\_P: Supply name for Pwell back bias for Forward Body Bias (FBB) Schemes.

## 1.2.15 Performance Boosters

There are several performance boosting features available within the 22nm FD SOI SiWare compilers. [Table 1-8](#) provides the parameters used to enable the options, as well as the performance enhancement obtained with each option.

**Table 1-8 Performance Booster Options**

Feature	GLB Parameter	Description
Center Decode	<code>center_decode</code>	<p>The Center Decode option allows for a trade off between area and performance. When <code>center_decode</code> is set to TRUE, it improves instance performance at the cost of area.</p> <p>If <code>center_decode</code>=TRUE, then there are one repair elements.</p> <p>If <code>center_decode</code>=FALSE, then the instance is generated with side decode and has one repair element.</p>
Banks	BK	Setting the number of banks provides a compile time option to split the memory into more than one bank. Memory banking is efficient for large instances. It improves performance and active power at the cost of area.
Column Mux	CM	Allows user to change the aspect ratio of the instance for chip floorplan for a trade-off between area and performance.

## 1.2.16 Optional Periphery Transistor Threshold Voltage Selection

A compile-time option, `periphery_vt`, is offered to select the periphery transistor threshold voltage implant (Vt). The array transistor threshold implants remain unchanged by this compile-time option. The default setting and available options for `periphery_vt` are described in [Table 1-9](#). The list of devices used for different `periphery_vt` options is presented in [Table 1-10](#).

**Table 1-9 Periphery Vt Options**

Compiler	Process	periphery_Vt Options	
		LOW	ULTRALOW
Ultra High-Density and Performance Two Port Register File	FD SOI	<code>periphery_vt</code> =LOW	<code>periphery_vt</code> =ULTRALOW (DEFAULT)
Ultra High-Density Single Port SRAM	FD SOI	<code>periphery_vt</code> =LOW	<code>periphery_vt</code> =ULTRALOW (DEFAULT)
High Density and Performance Single Port SRAM	FD SOI	<code>periphery_vt</code> =LOW	<code>periphery_vt</code> =ULTRALOW (DEFAULT)
High Density and Performance Single Port Register File	FD SOI	<code>periphery_vt</code> =LOW	<code>periphery_vt</code> =ULTRALOW (DEFAULT)

**Table 1-10    Periphery devices for different Vt Options**

periphery_Vt	Periphery Devices
LOW	Majority LVT and few SLVT devices for speed critical path
UTRALOW	Majority SLVT and few LVT devices for leakage saving



# 2

## Compiler Profile

This chapter describes the functional features, specifications, and timing characterization of the *SiWare Automotive Grade 1 Single Port High Density and Performance Leakage Control SRAM 2M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process*.

### 2.1 Compiler Naming Convention

Synopsys, Inc. uses the same name for a compiler and for the directory that contains the compiler library files. Compiler names contain segments of lowercase, alphanumeric characters with each segment denoting a characteristic of that specific compiler. For example, [Table 2-1](#) explains the structure of the name for the current compiler gf22nsd41p11s1dcl02ms.

**Table 2-1 Compiler Naming Convention**

Name Segment	Value in gf22nsd41p11s1dcl02ms	Description
Foundry	gf	GLOBALFOUNDRIES
Technology	22n	22 nanometers
Process	sd4	FD SOI P-Optional Vt/Cell Std Vt
Ports	1p	Single Port The format is $np$ , where $n$ is the number of ports.
Read/Write	11	One read and one write port The format is $nm$ , where $n$ is the number of read ports and $m$ is the number of write ports.
Product family	s1	SiWare Automotive Grade 1
Product subfamily	dcl	High-Density and Performance Leakage Control SRAM
Size	02m	2M bits
Protocol	s	Synchronous clock

## 2.2 Compiler Features and Benefits

The *SiWare Automotive Grade 1 Single Port High Density and Performance Leakage Control SRAM 2M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process* provides:

- All memory compilers have leakage optimization to reduce standby current.
- Three power management modes: LIGHT SLEEP, DEEP SLEEP, and SHUT DOWN to reduce static power.
- Dual Rail: Optional dual power supply using separate periphery and array power supplies, with level shifters in the periphery.
- Dynamic Voltage and Frequency Scaling support with periphery voltage reduction foundry specified VDDMIN supported (0.59V).
- Six characterized timing modes (RM0, RM1, RM2, RM3, RM4, RM5) to support speed binning, yield enhancement, and robust low voltage operation to VDDMIN.
- Special test modes enable externally bypassing read and write self-timed circuits and adjusting read and write margins.
- Pipeline option available with BIST option.
- ECC compatible layout.
- Three test mode options to support third party memBIST, Synopsys' STAR Memory System and Redundancy.
- Compile-time option to enable column redundancy for optimal yield.
- Separate output (Q) and input (D) pins used for optimum data bus timing.
- Optional bit-write feature that enables you to selectively write to each I/O. The subword can be as small as one bit. A maskable Write Enable (WEM) signal is provided for each I/O for maximum flexibility.
- All unused pins are tied to internally generated logic 0/1 instead of directly tied to power supply 0/1.
- Bank options of (1, 2, 4, and 8) are available. Bank=1 achieves the highest density, while Bank=8 gives higher speed and lower power.
- Three aspect ratios for maximum area and performance optimization.
- Maximum instance size of 2560K bits.
- Zero Quiescent Current consumption when all Register File inputs including clocks which are stable.
- Always active outputs.
- Control signals are configured active high.
- Built in BIST interface, wrapper logic, and comparators allow easy connections to Synopsys' STAR Memory System BIST solution with redundancy.
- Includes a sub-word feature, which allows each bit in a word to be independently selected by a bit in the write-enable-mask for write.

- Provides an option to have single cycle clock read (flow-through) or pipelined read (data valid after 1 read clock latency) for BIST. The write operation is always in a single clock cycle.
- Flexibility in specifying the logical size of the RAM, including both word size and number of address locations, and column mux.
- Built in Self Test Wrapper logic.
- Built in repair technique.
- At-speed capture and compare support by build-in scan chains at inputs and outputs.
- Separate memory enable for pipeline output (BIST only).
- Periphery Off feature: Supports periphery off (VDD=0 or floating) during periphery off mode. This feature is controlled by the `vdda_enable` GLB parameter. When `vdda_enable=TRUE`, POFF pins will appear and users can shut off complete periphery power supply.
- Read Assist pin is supported in the compiler. This pin is controlled by the flag `read_assist`. The possible values are 0 and 1. It is recommended to keep `read_assist=TRUE`.
- Since Bitcell used in this compiler would always need Write Assist, hence WA and WPULSE pins are always present.

**Table 2-2 Settings for RA, WA, and WPULSE Pins**  
(`read_assist=TRUE`)

Timing Mode	RA	WA	WPULSE
RM5	00	01	000
RM4	00	01	000
RM3	00	10	000
RM2	00	10	000
RM1	00	11	000
RM0	00	11	000

## 2.3 Specifications

This section describes the specifications of the *SiWare Automotive Grade 1 Single Port High Density and Performance Leakage Control SRAM 2M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process*.

- [“Memory Symbol”](#) on page 26
- [“Pin Description”](#) on page 28
- [“Block Diagram”](#) on page 34
- [“Functional Options”](#) on page 35
- [“Functional Descriptions”](#) on page 36
- [“Compiler Range Information”](#) on page 37
- [“Metal Layer Usage”](#) on page 38
- [“Signal Pin and Power Routing”](#) on page 40
- [“IP Tagging”](#) on page 43
- [“Physical Grids”](#) on page 43
- [“Logic Truth Tables”](#) on page 43
- [“Hazard Information”](#) on page 47
- [“Waveform Diagrams”](#) on page 48
- [“Standard Operating Characterization Conditions”](#) on page 56
- [“Using Licensed Custom PVTs”](#) on page 57

### 2.3.1 Memory Symbol

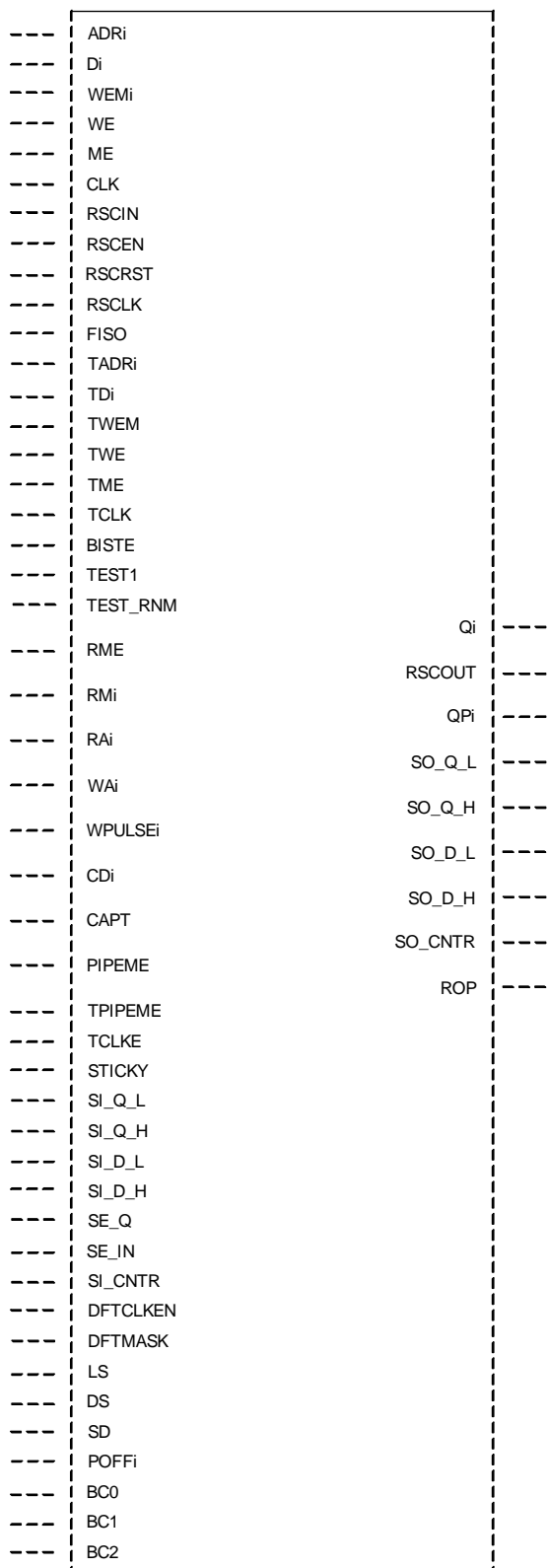
[Figure 2-1](#) shows the circuit symbol for a memory instance generated for the *SiWare Automotive Grade 1 Single Port High Density and Performance Leakage Control SRAM 2M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process*.

**Note**

[Figure 2-1](#) contains all possible pins. Not all pins are present on all instances.



**Figure 2-1 Memory Symbol**



## 2.3.2 Pin Description

Table 2-3 and Table 2-4 provide a brief description of the input and output pins. The number of bits against the pin names in following tables is instance specific. Values may differ for different instances.

**Table 2-3 Input Pin Description**

Signal	Enabled	Description
ADR [13:0]	always	Address input. This Address input port is used to address the location to be written during the Write cycle and read during the Read cycle.
D [31:0]	always	Data input. The data input bus is used to write the data into the memory location specified by the address input port during the Write cycle.
WEM [31:0]	wem_enable=TRUE SW=1	Maskable Write Enable. It includes the Bit Write feature where selective write to individual I/O's can be done using the Maskable Write Enable signals. When the memory is in Write cycle, one can write selectively on some I/O's.
WE	always	Write Enable input. When the Write Enable input is Logic High, the memory is in the Write cycle. When the Write Enable input is Logic Low, the memory is in the Read cycle.
ME	always	Memory Enable input. When the Memory Enable input is Logic High, the memory is enabled and read/write operations can be performed. When the Memory Enable input is Logic Low, the memory is deactivated.
CLK	always	Clock input. This is the external clock for the memory.
RSCIN	redundancy_enable=TRUE	Reconfiguration Register Scan Input. This is the Input pin is used to input the Redundancy Scan bit.
RSCEN	redundancy_enable=TRUE	Reconfiguration Register Scan Enable Input. This is the Input pin used to enable the Redundancy Scanning.
RSCRST	redundancy_enable=TRUE	Reconfiguration Register Scan Asynchronous Reset Input. This pin is used for clearing the scan register. Negedge(1->0) of RSCRST(FISO=0) is necessary prior to start of normal memory operation (read/write).
RSCLK	redundancy_enable=TRUE	Scan Clock for Reconfiguration Register.
FISO	redundancy_enable=TRUE	Reconfiguration register isolation. This pin is used to isolate the reconfiguration register and preserve the contents during advanced power management modes when the power supply to the periphery is shutdown.
TADR [13:0]	bist_enable=TRUE	Address inputs for BIST. This Address input port is used to address the location to be read during the Read cycle and written during the Write cycle. This pin is used only when BIST is enabled.

**Table 2-3 Input Pin Description**

Signal	Enabled	Description
TD [3:0]	bist_enable=TRUE	Data Inputs for BIST. This data input bus is used to write the data into the memory location specified by the address input port during the Write cycle. This is used when BIST is enabled.
TWEM	bist_enable=TRUE wem_enable=TRUE SW=1	Maskable Write Enable for BIST. It includes the Bit Write feature where, using the Maskable Write Enable signals, the selective write to individual I/O's can be done. When the memory is in Write cycle, you can write selectively on some I/O's. This pin is used only when BIST is enabled.
TWE	bist_enable=TRUE	Write Enable for BIST. When the BIST Write Enable input is Logic High, the memory is in the Write cycle. When the BIST Write Enable is Logic Low, the memory is in the Read cycle. This pin is used only when BIST is enabled.
TME	bist_enable=TRUE	Memory Enable for BIST. When the BIST Memory Enable is Logic High, the memory is enabled and read/write operations can be performed. When the BIST Memory Enable is Logic Low, the memory is deactivated. This is used only when BIST is enabled.
TCLK	bist_enable=TRUE	BIST Clock Input. This is the external clock for the memory. This is used only when BIST is enabled.
BISTE	bist_enable=TRUE	BIST Enable. It controls the BIST test mode. When enabled, it can be used for testing memory without modifying the main circuit.
TEST1	always	IT IS STRONGLY RECOMMENDED TO HAVE THESE PINS CONTROLLABLE IN SILICON BY SOFTWARE/FIRMWARE! IT CAN BE VERY BENEFICIAL FOR OVERALL PRODUCTION OR TEST.  TEST1 input. Test pin to bypass self-timed circuit. The external clock controls the read and write control signals.
TEST_RNM	test_rnm_enable=TRUE	IT IS STRONGLY RECOMMENDED TO HAVE THESE PINS CONTROLLABLE IN SILICON BY SOFTWARE/FIRMWARE! IT CAN BE VERY BENEFICIAL FOR OVERALL PRODUCTION OR TEST.  When this pin is high Memory will go in idle state and bit-lines are pre-charged high. ATPG mode should be turned off in this mode.
RME	always	IT IS STRONGLY RECOMMENDED TO HAVE THESE PINS CONTROLLABLE IN SILICON BY SOFTWARE/FIRMWARE! IT CAN BE VERY BENEFICIAL FOR OVERALL PRODUCTION OR TEST.  Read-Write Margin Enable Input. This input selects between the default Read-Write margin setting (RME=0), and the external pin Read-Write margin setting (RME=1).

**Table 2-3 Input Pin Description**

Signal	Enabled	Description
RM [3:0]	always	<p>IT IS STRONGLY RECOMMENDED TO HAVE THESE PINS CONTROLLABLE IN SILICON BY SOFTWARE/FIRMWARE! IT CAN BE VERY BENEFICIAL FOR OVERALL PRODUCTION OR TEST.</p> <p>Read-Write margin Input. This input is used for setting the Read-Write margin. It programs the sense amp differential setting and allows the trade off between speed and robustness.</p> <p>RM[3:0] = 4'b0000 is the slowest possible mode of operation for the memory. This setting is required for RM0 operation.</p> <p>RM[3:0] values control access time &amp; cycle time of the memory. Refer to the timing table in instance datasheets for more details.</p>
RA [1:0]	read_assist=TRUE	<p>IT IS STRONGLY RECOMMENDED TO HAVE THESE PINS CONTROLLABLE IN SILICON BY SOFTWARE/FIRMWARE! IT CAN BE VERY BENEFICIAL FOR OVERALL PRODUCTION OR TEST.</p> <p>Read Assist Pins to control WL under-drive.</p>
WA [1:0]	always	<p>IT IS STRONGLY RECOMMENDED TO HAVE THESE PINS CONTROLLABLE IN SILICON BY SOFTWARE/FIRMWARE! IT CAN BE VERY BENEFICIAL FOR OVERALL PRODUCTION OR TEST.</p> <p>WA[1] Write assist enable pin (Active High). WA[0:0] Write Assist pin to control negative voltage on SRAM bitline.</p>
WPULSE [2:0]	always	<p>IT IS STRONGLY RECOMMENDED TO HAVE THESE PINS CONTROLLABLE IN SILICON BY SOFTWARE/FIRMWARE! IT CAN BE VERY BENEFICIAL FOR OVERALL PRODUCTION OR TEST.</p> <p>Write Assist Pulse to control pulse width of negative voltage on SRAM bitline.</p>
CD [3:0]	bist_enable=TRUE	Control Data Input. Control bits to generate write compare data.
CAPT	bist_enable=TRUE	Capture Error Input. The CD bus is compared with the data read from the memory. When CAPT is logic 0, the output from the comparator is stored in the comparison register inside the memory.
PIPEME	scan_enable=TRUE or bist_enable=TRUE	Pipeline Memory enable input for the read port. When the PIPEME Input is Logic High, the pipeline output of the memory will switch at the rising edge of the clock and will follow the regular output with a latency of one clock cycle. When it is Logic Low, the pipeline output will retain its previous value.
TPIPEME	bist_enable=TRUE	BIST Pipeline Memory enable input for the read port. When the TPIPEME Input is Logic High, the pipeline output of the memory will switch at the rising edge of the clock and will follow the regular output with a latency of one clock cycle. When it is Logic Low, the pipeline output will retain its previous value.

**Table 2-3 Input Pin Description**

Signal	Enabled	Description
TCLKE	bist_enable=TRUE	Test external clock enable signal.
STICKY	bist_enable=TRUE	Sticky Input. When STICKY is Logic level 0, the captured data in the comparison register is overwritten with the latest comparison data. when STICKY is Logic level 1, the captured data in the comparison register is OR'ed with the old value of the comparison register and stored back in the comparison register.
SI_Q_L	scan_enable=TRUE or bist_enable=TRUE center_decode=TRUE	Scan Chain Input for lower bits of QP.
SI_Q_H	scan_enable=TRUE or bist_enable=TRUE center_decode=TRUE	Scan Chain Input for higher bits of QP.
SI_D_L	scan_enable=TRUE or bist_enable=TRUE center_decode=TRUE	Scan Chain Input for lower bits of data.
SI_D_H	scan_enable=TRUE or bist_enable=TRUE center_decode=TRUE	Scan Chain Input for higher bits of data.
SE_Q	scan_enable=TRUE or bist_enable=TRUE	Scan Chain Enable Input for pipe chain.
SE_IN	scan_enable=TRUE or bist_enable=TRUE	Scan Enable input for "Data and Control scan chains". Once made active, internal clock generation stops (READ/WRITE operation is disabled).
SI_CNTR	scan_enable=TRUE or bist_enable=TRUE	Scan Chain Input for "Control Signals".
DFTCLKEN	scan_enable=TRUE or bist_enable=TRUE	This Input enable scan chain clocks for ATPG scan. Data flows synchronously with clock.
DFTMASK	scan_enable=TRUE or bist_enable=TRUE	This pin is used to enable the ATPG mode. When DFTMASK is asserted, output bus follows data input bus synchronously. It allows data inputs ("D" bus) to be written to output ("Q" bus) after a CLK rising edge. DFTCLKEN should be enabled for this operation. When DFTMASK is asserted, the internal clock of the memory is generated though no read and write operation is done. The internal clock generation allows CLK rise to Q delay when DFTMASK=1 to be close to CLK rise to Q delay when DFTMASK=0 on respective timing_mode.
LS	always	Light Sleep Input. When the memory is disabled and the pin is active, then the memory goes into low leakage mode. There is no change in the output state.
DS	pg_enable=TRUE	Deep Sleep Input. This pin shuts down power to periphery and maintain memory contents. The outputs of the memory are pulled low.

**Table 2-3 Input Pin Description**

Signal	Enabled	Description
SD	pg_enable=TRUE	Shut Down Input. This pin shuts down power to periphery and memory core, no memory data retention.
POFF [1:0]	vdda_enable=TRUE	<p>THESE SIGNALS MUST BE DRIVEN FROM VDDA ARRAY SUPPLY-POWER DOMAIN, NOT PERIPHERY SUPPLY!</p> <p>Periphery Off mode inputs. These inputs are used to put memory in power down modes in which VDD is allowed to collapse.</p> <p>POFF[1:0] = 2'b00 - No Periphery Off Mode.</p> <p>POFF[1:0] = 2'b01 - Standard Periphery Off Mode: It shuts down power to periphery and maintain memory contents (VDD=X), Outputs X.</p> <p>POFF[1:0] = 2'b10 - BC0 Periphery Off Mode: Periphery Off mode with bypass for Array Source Bias.</p> <p>POFF[1:0] = 2'b11 - SD Periphery Off Mode: It shuts down power to periphery and memory core, no memory data retention (VDD=X), Outputs X.</p>
BC0	always	Biasing bypass input pin, setting BC0 to a 'logic 1' bypasses the array bias in Light and Deep sleep modes.
BC1	always	Biasing Level Adjust Input. Its setting gives a smaller value of rise on vss-core voltage to give enough head room for retention. This setting is recommended for Vnom -10% to Vmax.
BC2	always	Biasing Level Adjust Input. Its setting gives a higher value of rise on vss-core voltage to give enough head room for retention. This setting is recommended for Vnom to Vmax.

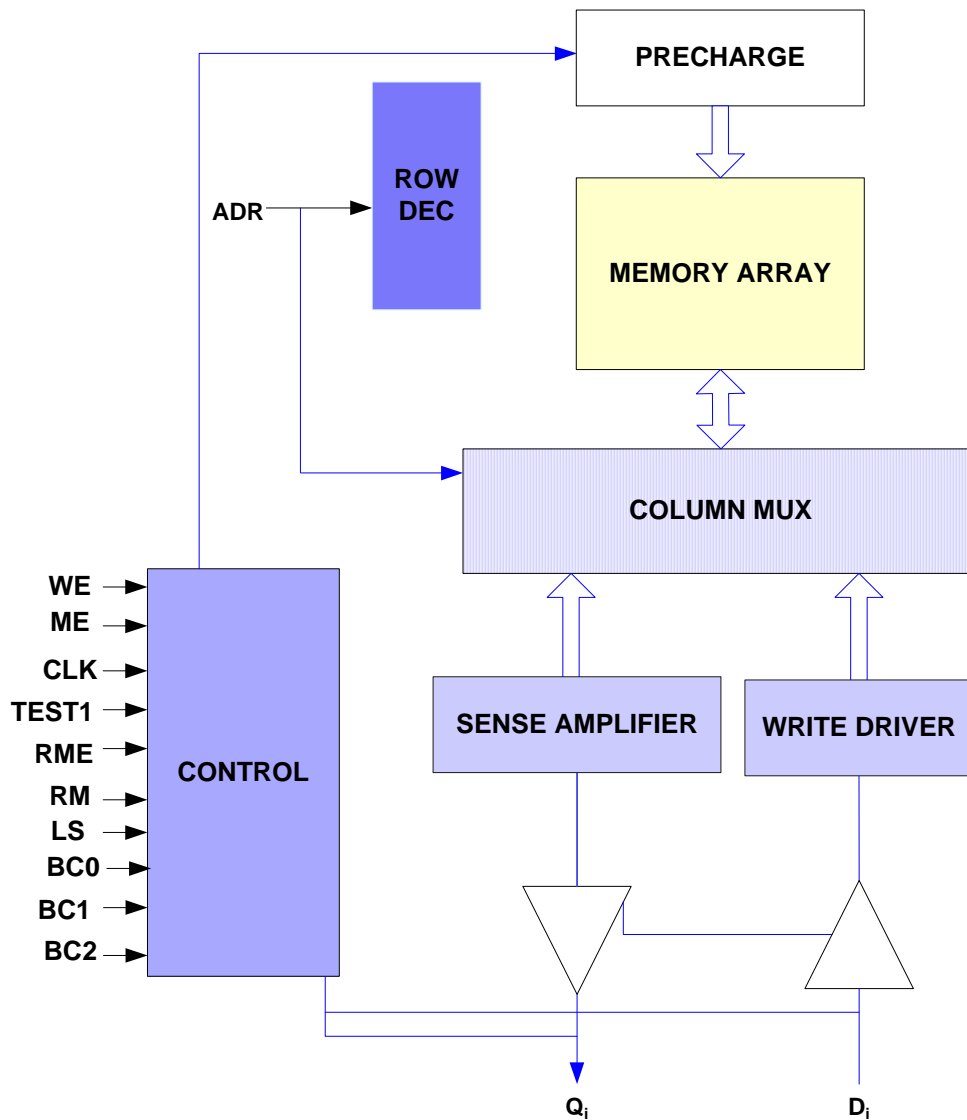
**Table 2-4 Output Pin Description**

Signal	Enabled	Description
Q [31:0]	always	Data output bus. It outputs the contents of the memory location addressed by the Address Input signals.
RSCOUT	redundancy_enable=TRUE	Reconfiguration Register Scan Output. This is the pin used to output the Reconfiguration Register Scan bit.
QP [31:0]	bist_enable=TRUE	Data Output during pipeline mode.
SO_Q_L	scan_enable=TRUE or bist_enable=TRUE center_decode=TRUE	Scan Chain Output for lower bits of QP.
SO_Q_H	scan_enable=TRUE or bist_enable=TRUE center_decode=TRUE	Scan Chain Output for higher bits of QP.
SO_D_L	scan_enable=TRUE or bist_enable=TRUE center_decode=TRUE	Scan Chain Output for lower bits of data.
SO_D_H	scan_enable=TRUE or bist_enable=TRUE center_decode=TRUE	Scan Chain Output for higher bits of data.
SO_CNTR	scan_enable=TRUE or bist_enable=TRUE	Scan Chain Output for "Control Signals".
ROP	pg_enable=TRUE	ROP is Ready for Operation pin. This pin should be high for any valid memory operation.

### 2.3.3 Block Diagram

Figure 2-2 shows the functional pin block diagram for the compiler.

Figure 2-2 Functional Pin Block Diagram





## 2.3.4 Functional Options

The *SiWare Automotive Grade 1 Single Port High Density and Performance Leakage Control SRAM 2M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process* provides the following functional options:

- This compiler supports the configurations shown in [Table 2-5](#). The flags **redundancy\_enable**, **row\_redundancy**, **bist\_enable**, **scan\_enable**, and **pg\_enable** determine the selection of the configuration that is to be used.

**Table 2-5 SiWare High-Density Memory Options**

Configuration	Parameters Required
SiWare-LT (default configuration)	redundancy_enable=FALSE bist_enable=FALSE scan_enable=FALSE pg_enable=FALSE
SiWare-LR (LT + Redundancy)	redundancy_enable=TRUE row_redundancy=TRUE/FALSE bist_enable=FALSE scan_enable=FALSE pg_enable=FALSE
SiWare-IT-Lite (LT + Scan chains + SWT + Pipeline)	redundancy_enable=FALSE bist_enable=FALSE scan_enable=TRUE pg_enable=FALSE
SiWare-IT (IT-Lite + BIST Muxes + Comparator)	redundancy_enable=FALSE bist_enable=TRUE pg_enable=FALSE
SiWare-ST-Lite (LR + Scan chains + SWT + Pipeline)	redundancy_enable=TRUE row_redundancy=TRUE/FALSE bist_enable=FALSE scan_enable=TRUE pg_enable=FALSE
SiWare-ST (ST-Lite + BIST Muxes + Comparator)	redundancy_enable=TRUE row_redundancy=TRUE/FALSE bist_enable=TRUE pg_enable=FALSE



### Note

All the modes described in the table above are also possible with Advance Power Management (**pg\_enable=TRUE**).

- BISTE and TEST input pins are optional. These input pins will be generated only if BIST is enabled by setting **bist\_enable=TRUE** in the custom global file.

## 2.3.5 Functional Descriptions

### 2.3.5.1 Bit Write Mask

This memory includes the Sub Word feature where selective writes to individual I/O's can be done using the maskable write enable signals (WEM). Each I/O has its own WEM control signal that allows completely flexible masking capability. The WEM signals need to be tied together externally to support different sub-word sizes. TWEM is 1 bit and controlled by the SMS interface.

The SW option in the GLB file can remove the WEM/TWEM signals.

### 2.3.5.2 Test Modes, RM[3:0]

The memory function, when any of the test modes are enabled is explained below:

**TEST1:** TEST1 disables the memory read self-timed clock, and controls the memory with the external clock. Q will appear on the output after the negative edge of external clock when TEST1=TRUE. Read starts with the positive edge and terminates with the negative edge of the CLK. Write also starts with the positive edge and terminates on the negative edge of the CLK. Both read and write could not be done in one cycle for TEST1.

**TEST\_RNM:** When this pin is high, memory will go into idle state. The purpose of this mode is to help identify marginal bits by uncovering read-upset failures. This can be crucial in identifying root causes of yield fallout. In this mode bitlines stay at high during a dummy read access.



**Note** ATPG mode should be turned off in TEST\_RNM mode.

**RME:** Read Margin Enable selects between the default RM setting, and the external pin RM setting. If RME is set low the default RM values will be applied to the memory.

**RM[3:0]:** There are four Read Margin RM[3:0] pins available in Synopsys' DesignWare embedded memories. These are used to trade-off between speed and robustness (yield). There is also a Read Margin Enable (RME) pin that selects between the internal default RM[3:0] value set during compilation, and the external, user provided RM[3:0] settings. It is required that external access to all RM pins, including RME, is provided for debug and yield analysis purposes. Registers can be used to access the RM pins for debug. If Synopsys' STAR Memory System is used, the RM pins will be automatically registered during compilation.

**Table 2-6 Read Margin Control Pins**

Signal	< >_custom.glb setting	Description
RME	None, always enabled	Read-Write Margin Enable pin. Selects between internal and external RM[2:0]* settings. RME=0: Internal settings; RME=1: External settings.
RM[3:0]	None, always enabled	Used to change Read-Write Margin settings.



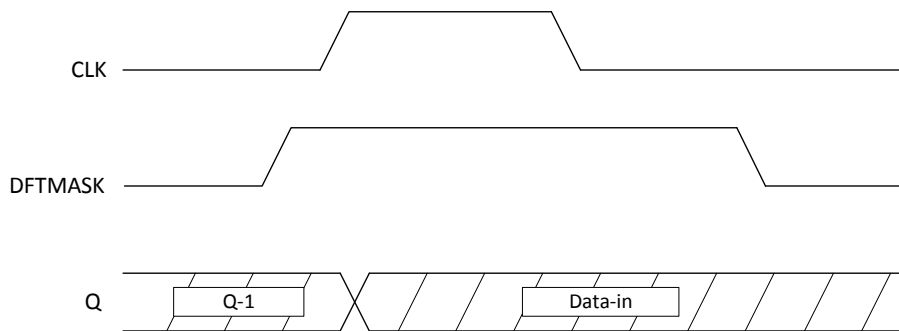
## Note

\* RM3 is not controlled by RME.

### 2.3.5.3 Synchronous Write-Through

Synchronous write-through will be controlled with the DFTMASK pin. When DFTMASK=1, the output latch will follow data-in. When DFTMASK=0 the output will remain in the previous state. For details related to SWT functionality, refer to [Table 2-9](#) on page 43.

**Figure 2-3 DFTMASK Waveform**



### 2.3.6 Compiler Range Information

[Table 2-7](#) shows the ranges for the address, Number of Words (**NW**), Number of Bits (**NB**), Column Mux (**CM**) options, and bit size ranges for this compiler, including the impact of the `center_decode` parameter value.

**Table 2-7 Compiler Range Values**

CM	Address Inputs Min	Address Inputs Max	NW Increment	BK	NW Min	NW Max	NB Increment	NB Min	center_decode=TRUE		center_decode=FALSE	
									NB Max	Maximum Number of Bits	NB Max	Maximum Number of Bits
4	5	10	16	1	32	1024	1	8	320	327680	160	163840
	6	11	32	2	64	2048	1	8	320	655360	160	327680
	7	12	64	4	128	4096	1	8	320	1310720	160	655360
	8	13	128	8	256	8192	1	8	320	2621440	160	1310720
8	6	11	32	1	64	2048	1	8	160	327680	80	163840
	7	12	64	2	128	4096	1	8	160	655360	80	327680
	8	13	128	4	256	8192	1	8	160	1310720	80	655360
	9	14	256	8	512	16384	1	8	160	2621440	80	1310720

**Table 2-7 Compiler Range Values**

CM	Address Inputs Min	Address Inputs Max	NW Increment	BK	NW Min	NW Max	NB Increment	NB Min	center_decode=TRUE		center_decode=FALSE	
									NB Max	Maximum Number of Bits	NB Max	Maximum Number of Bits
16	7	12	64	1	128	4096	1	8	80	327680	40	163840
	8	13	128	2	256	8192	1	8	80	655360	40	327680
	9	14	256	4	512	16384	1	8	80	1310720	40	655360
	10	15	512	8	1024	32768	1	8	80	2621440	40	1310720

The Column Mux is an option that helps change the physical aspect ratio of the instance generated by the compiler. When you lower the Column Mux value, the number of selection levels (multiplexing) of instance I/O used is reduced.

If the instance has a high number of bits per word and a low number of words, use the lowest possible value for the Column Mux option to obtain a square aspect ratio. If the instance has a high number of words and a low number of bits per word, use the highest possible value Column Mux.

### 2.3.7 Metal Layer Usage

Figure 2-4 shows the metal layer usage in the compiler. The metal directions are with respect to vertical poly. Table 2-8 explains the routing instructions for various metal layers.

- M1, M2, C1, and C2 layers are completely blocked.
- M2 is Bitline. C1 is for Wordline.
- C2 is orthogonal to SRAM poly.
- C3/C4 and above are available for routing over the instance. C3/C4 are mainly used for power routing and can also be used for signal routing. For the usage of C3 and C4 as a power routing refer [“Signal Pin and Power Routing”](#) on page 40.

Figure 2-4 Metal Layer Usage

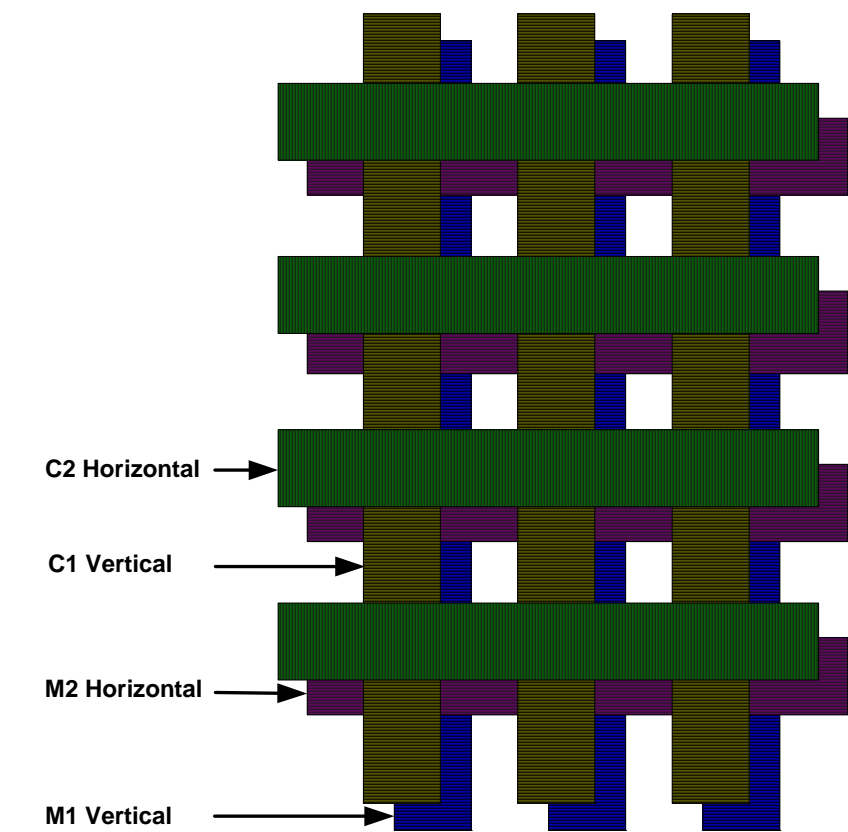


Table 2-8 Metal Layer Routing

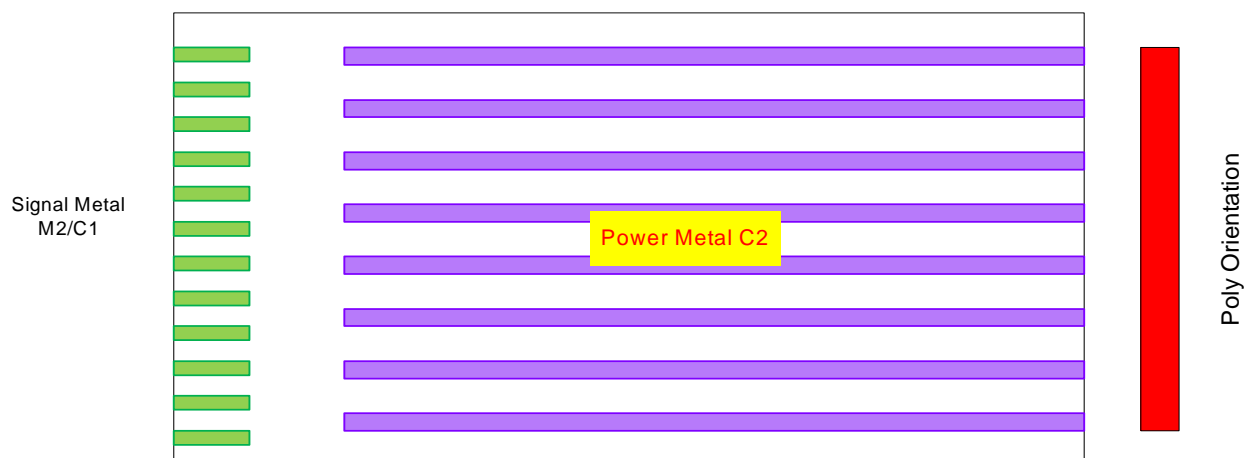
Metal Layer	Routing Instructions
Metal layer availability	M1, M2, C1, and C2 are completely blocked.
C2 routing porosity	Not supported
Routing porosity for C3/C4 and above (including the power mesh)	100%, freely

## 2.3.8 Signal Pin and Power Routing

Signal and power routing of the memory compiler depends on the standard cell usage. A standard cell with 9 track uses VHV scheme and 6.75 track uses HVH scheme, where the memory instance will have M2 and C1 as the signal metal, respectively. M2/C1 signal pins are in horizontal direction (orthogonal to Poly direction) as shown in the [Figure 2-5](#).

Power pins in C2 for VDD/VSS/VDDA/VDDF/VNW\_N/VPW\_P works with both standard cells schemes.

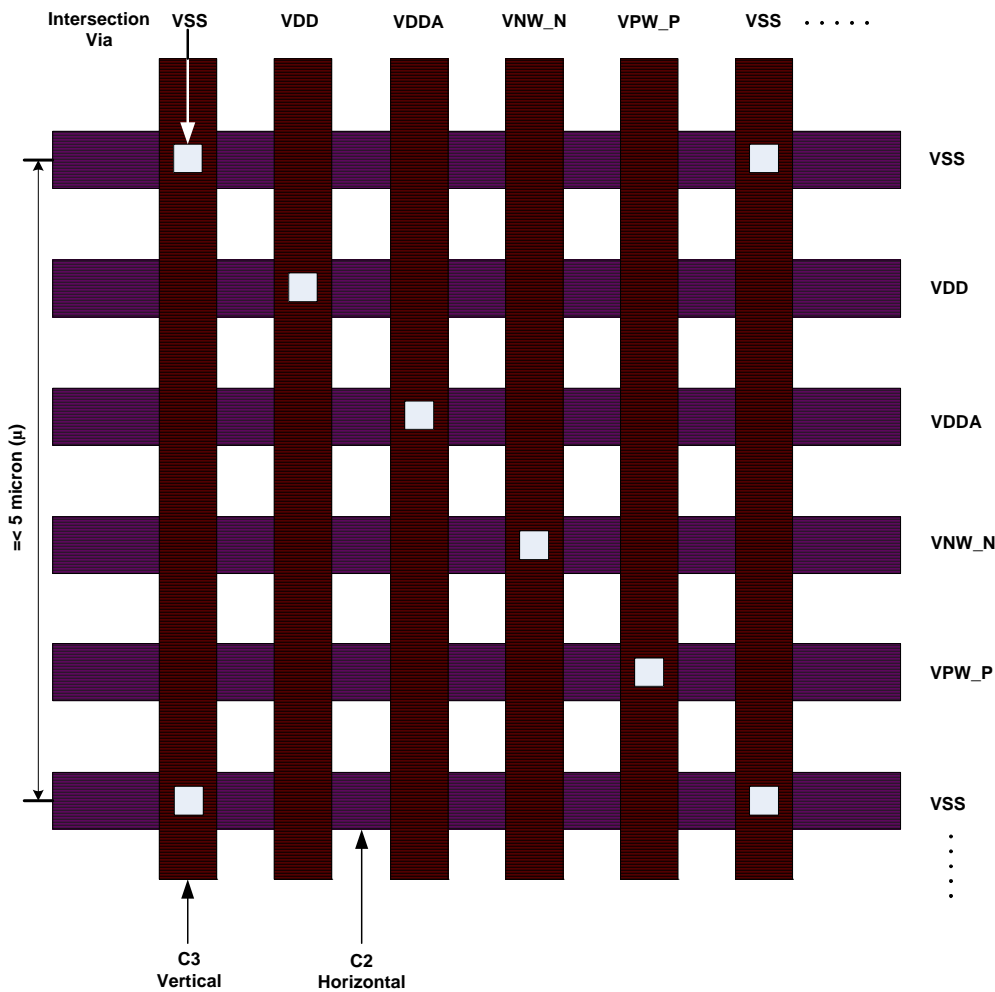
**Figure 2-5 Signal and Power Metal Orientation**



### 2.3.8.1 9 Track Standard Cell Library Compatible Scheme (VHV Scheme)

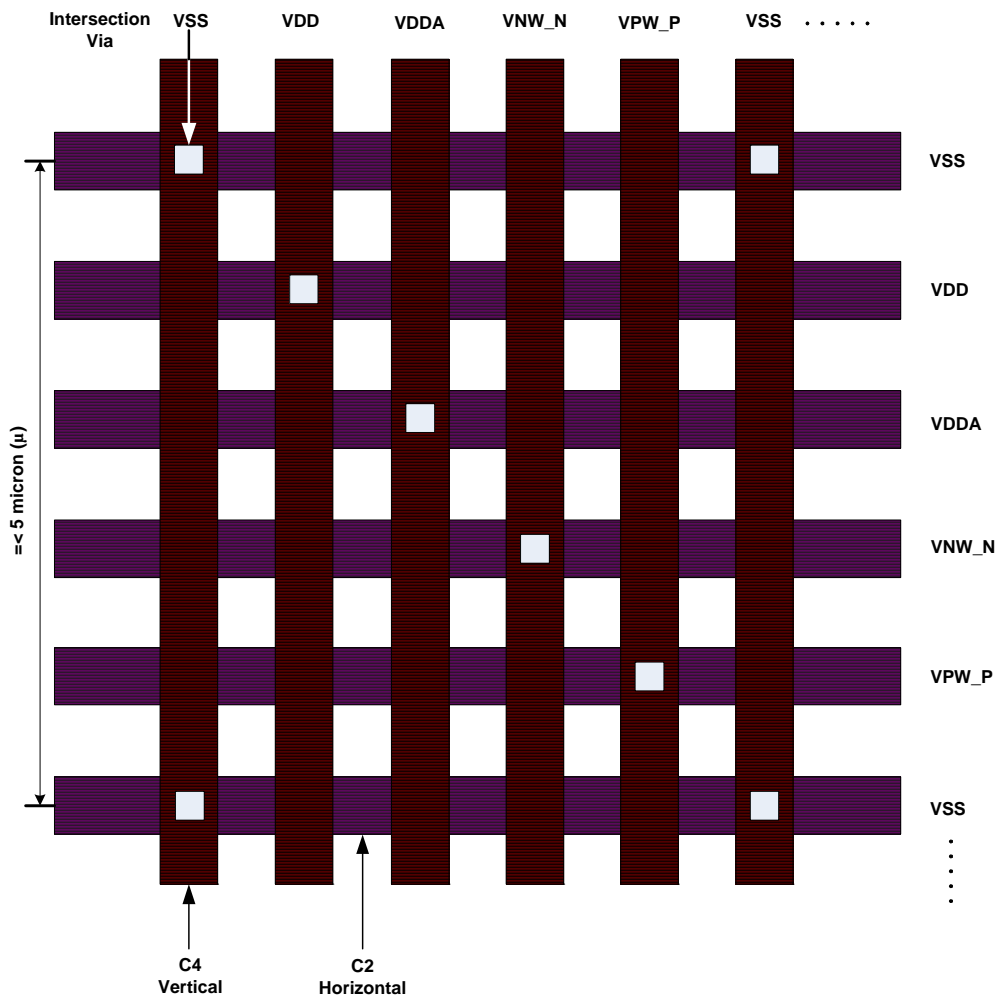
- Memory instance Power Ground (PG) pins are in C2.
- Chip PG C3 rails should be connected to memory PG pins at every intersection.
- Chip PG C3 rails must be connected to memory PG pins at every  $5\mu$  for all PG supplies as shown in the [Figure 2-6](#). For example VDD, VSS, VDDA, VDDE, VNW\_N, and VPW\_P.
- Populate all the C3/C2 crossovers with maximum number of vias possible.
- Memory instance signal pins are in M2 and can be accessed in M2 or above.

**Figure 2-6 Power Mesh Connections**



**2.3.8.2 6.75 Track Standard Cell Library Compatible Scheme (HVH Scheme)**

- Memory instance Power Ground (PG) pins are in C2.
- Chip PG C4 rails should be connected to memory PG pins at every intersection.
- Chip PG C4 rails must be connected to memory PG pins at every  $5\mu$  for all PG supplies as shown in the [Figure 2-7](#). For example VDD, VSS, VDDA, VDDE, VNW\_N, and VPW\_P.
- Populate all the C4/C2 crossovers with maximum number of vias possible.
- Memory instance signal pins are in C1 and can be accessed in C1 or above.

**Figure 2-7 Power Mesh Connections**



## 2.3.9 IP Tagging

Synopsys, Inc. labels its Intellectual Property (IP) with either the use of the IP marking layer or a VSIA tag layer, noted in the **file\_xlft.csv** file. This GDSII layer identification is required to correctly label Synopsys' IP at the foundry.

It is imperative that customers include this layer, in the form **layer:datatype**, when ordering masks.

## 2.3.10 Physical Grids

The layout grid is 0.001 $\mu$ m.

## 2.3.11 Logic Truth Tables

### 2.3.11.1 Memory Function Truth Table



#### Note

Negedge(1->0) of RSCRST(FISO=0) is necessary prior to start of normal memory operation (read/write).

NC: No change in Q output, when DFTMASK transitions from H -> L.

Q-1: Remain the same state from previous cycle.

Q: Output with no CLK latency.

Table 2-9 shows the memory function truth table for the compiler.

**Table 2-9 Memory Function Truth Table**

Function	CLK	ME	WE	WEM	D	DFTMASK	SE_IN	DFTCLKEN	BISTE	Q
Idle	L	X	X	X	X	L	X	L	L	Q-1
Disabled	H	L	X	X	X	L	L	L	L	Q-1
Read	H	H	L	X	X	L	L	L	L	Q
Write	H	H	H	H	Data-in	L	L	L	L	Q-1
Mask	H	H	H	L	X	L	L	L	L	Q-1
SWT	H	X	X	H	Data-in	H	L	H	L	Data-in
	H			L	X	H	L	H		H
	L			X	X	L	X	X		NC



#### Note

The status of test signals for the above truth table is TEST1=L.

**2.3.11.2 Power Mode Truth Table****Note**

\* Applicable only when `pg_enable=TRUE`.

QP-1 = Pipe output remains the same state from previous cycle.

Q-1 = Memory output remains the same state from previous cycle.

ROP-1 = Ready for Operation output remains the same state from previous cycle.

SO-1 = Scan output remains the same state from previous cycle.

SO = SCAN CHAIN OUTPUTS: SO\_Q\_L, SO\_Q\_H, SO\_D\_L, SO\_D\_H, SO\_CNTR

In Periphery Off modes, Periphery supply VDD is allowed to be powered down.

Table 2-10 shows the truth table for Power mode.

**Table 2-10 Power Mode Truth Table**

Mode	ME	LS	DS*	SD*	POFF[1:0]	BC0/ BC1/ BC2	ROP	All Other Input Pins	Output (Q)	QP	SO	Comments
Functional	0/1	0	0	0	00	0/1	ROP-1	Normal input mode	Q	QP	SO	Normal operation
Light Sleep	0/1	1	0	0	00	0/1	ROP-1	Cannot float, must be driven	Q (hold last state)	QP (hold last state)	SO (hold last state)	Partial Periphery shutdown, Array source bias
Deep Sleep	X	X	0->1	0	00	0/1	0	X	0	0	0	Periphery shutdown, Array source bias
	X	X	1->0	0	00	0/1	1	X	Q (retains 0)	X	X	Periphery shutdown, Array source bias
Shut Down	X	X	X	0->1	00	X	0	X	0	0	0	Periphery shutdown, Array shutdown
	X	X	X	1->0	00	X	1	X	Q (retains 0)	X	X	Periphery shutdown, Array shutdown
Standard Periphery Off	X	X	X	X	01	X	X	X	X	X	X	VDD can be removed; Array Source Bias
BC0 Periphery Off	X	X	X	X	10	X	X	X	X	X	X	VDD can be removed; no source biasing on Array.
Shut Down Periphery Off	X	X	X	X	11	X	X	X	X	X	X	VDD can be removed; Array shutdown

### 2.3.11.3 BIST Mode Truth Table

Table 2-11 shows the truth table for BIST mode.

**Table 2-11 BIST Mode Truth Table**

Function	BISTE	Active Pins					
Normal Mode	L	ME	WE	ADR	D	WEM	CLK
BIST Mode	H	TME	TWE	TADR	TD	TWEM	TCLK



**Note**

TCLKE is used as select pin for CLK/TCLK.

### 2.3.11.4 Redundancy Scan Mode Truth Table

Table 2-12 shows the truth table for the Redundancy Scan mode.

**Table 2-12 Redundancy Scan Mode Truth Table**

Function	RSCLK	RSCEN	RSCIN	ME	RSCRST	FISO
Normal	X	X	X	H	X	H
Normal	X*	X	X	H	L	L
Normal	X	L	X	H	L	L
Scan	L->H	H	Fuse_Data	X	L	L
Reset	X	X	X	X	H	L



**Note**

\* RSCLK is not running and is held at either H or L.

### 2.3.11.5 Bias Control Truth Table



**Note** \* Applicable only when `pg_enable=TRUE`.

Table 2-13 shows the bias control truth table.

**Table 2-13 Bias Control Truth Table**

Mode	BC0	BC1	BC2	SD*	DS*	LS
Normal Mode	0/1	0/1	0/1	0	0	0
Non recommended, Debug mode	0	0	0	0	0	1
LS with Array Bias (recommended for Vnom -10% to Vmax)	0	1	0	0	0	1
LS with Array Bias (recommended for Vnom to Vmax)	0	0	1	0	0	1
Non recommended, Debug mode	0	1	1	0	0	1
LS without Array Bias	1	0/1	0/1	0	0	1
Non recommended. Debug mode	0	0	0	0	1	0
DS with Array Bias (recommended for Vnom-10% to Vmax)	0	1	0	0	1	0
DS with Array Bias (recommended for Vnom to Vmax)	0	0	1	0	1	0
Non recommended. Debug mode	0	1	1	0	1	0
DS without Array Bias	1	0/1	0/1	0	1	0
Shutdown mode without array retention	0/1	0/1	0/1	1	0	0

### 2.3.11.6 Minimum Voltage Truth Table

Table 2-14 shows the truth table for minimum voltage allowed for different RM modes.

**Table 2-14 Minimum Voltage for different RM Modes Truth Table**

Mode	RM[3:0]	VMIN (V)	RA Setting	WA Setting
RM5	0111/0110/0101	0.81	00	01
RM4	0100	0.81	00	01
RM3	0011	0.72	00	10
RM2	0010	0.675	00	10
RM1	0001	0.63	00	11
RM0	0000	0.59	00	11



### Note

In the above table, RME=1 for all RM modes.

## 2.3.12 Hazard Information



### Note

ME - Pin ME  
WE - Pin WE  
DI - Pin D  
ADR - Pin ADR  
Q - Pin Q  
NC - No Change  
CCL - Corrupt Current Location  
CEM - Corrupt Entire Memory  
WE is High during Write Mode.  
WE is Low during Read Mode.

### 2.3.12.1 Hazard Information for Read/Write Cycle

Table 2-15 shows the Read/Write cycle hazard information for the compiler.

**Table 2-15 Hazard Information for Read/Write Cycle**

Mode	ME	WE	ADR	DI	Memory	Q
Read	High	-	-	-	NC	Valid
Write	High	-	-	-	Data-In	Valid
Read	High	-	-	Violation	NC	Valid
Write	High	-	-	Violation	CCL	Valid
Read	High	-	Violation	-	CEM	X
Write	High	-	Violation	-	CEM	Valid
Read	High	-	Violation	Violation	CEM	X
Write	High	-	Violation	Violation	CEM	Valid
Read/Write	High	Violation	-	-	CCL	X
Read/Write	High	Violation	-	Violation	CCL	X
Read/Write	High	Violation	Violation	-	CEM	X
Read/Write	High	Violation	Violation	Violation	CEM	X
Read	Violation	-	-	-	CEM	X
Write	Violation	-	-	-	CEM	Valid
Read/Write	Low	-	-	-	NC	Valid

2.3.12.2 Hazard Information for Clock Violations


Table 2-16 shows the clock violations hazard information for the compiler.

Table 2-16 Hazard Information for Clock Violations

Mode	Violation	ME	Memory	Q
Read/Write	Clock Cycle Width	Low/High	CEM	X
Read/Write	Clock High Pulse Width	Low/High	CEM	X
Read/Write	Clock Low Pulse Width	Low/High	CEM	X

2.3.13 Waveform Diagrams

This section shows the various timing waveforms.

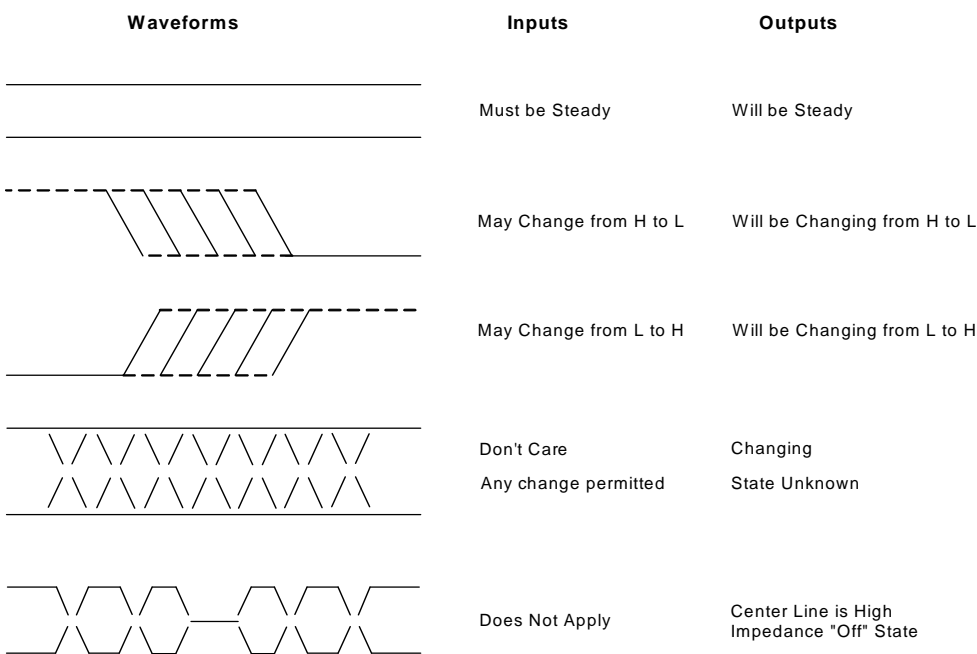
 **Note**

If ME is High, it is necessary to ensure that the timing spec of each signal is kept with respect to the clock.

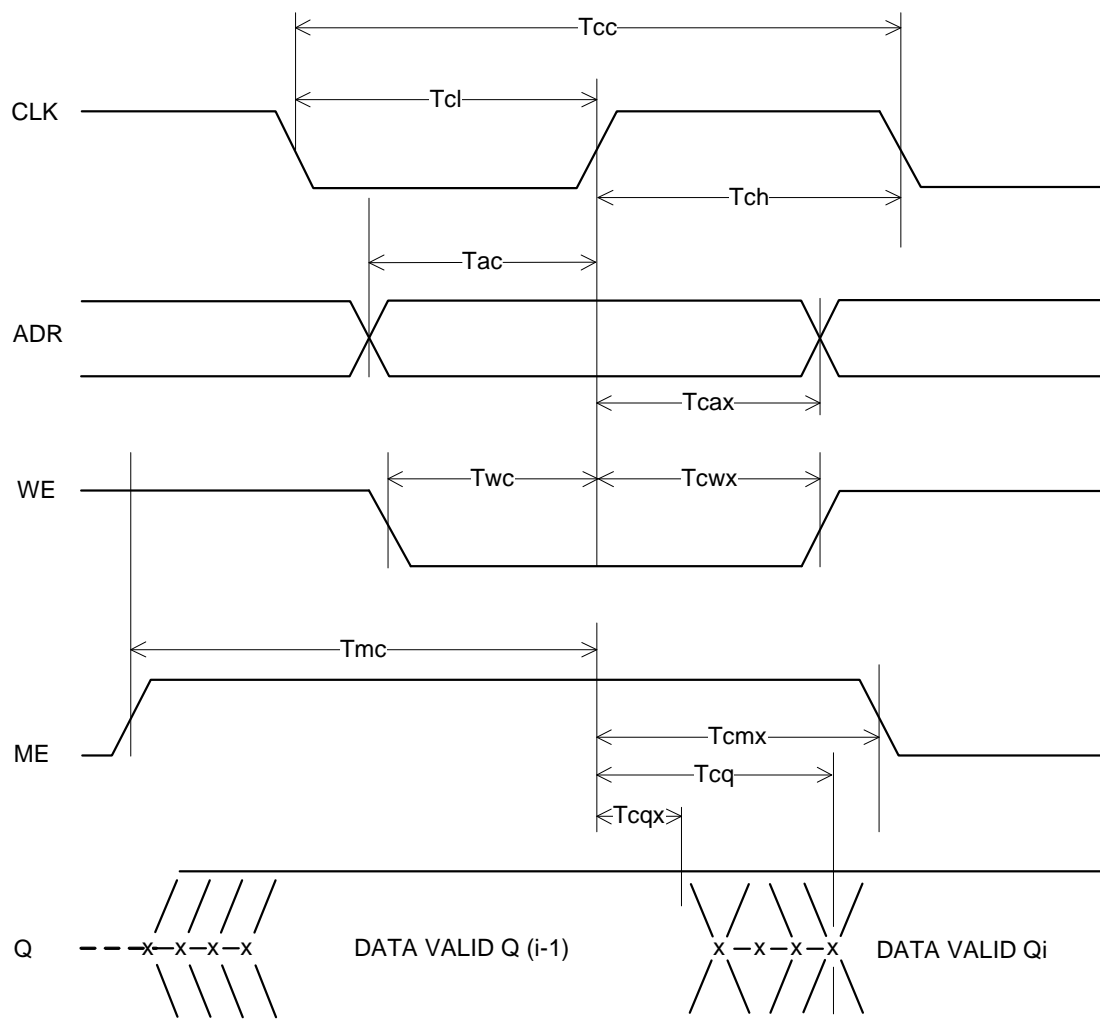
If ME is Low, it is not necessary to meet the timing spec of each signal. This applies to power management pins like LS as well. When ME=0, LS can be asserted and de-asserted asynchronously.

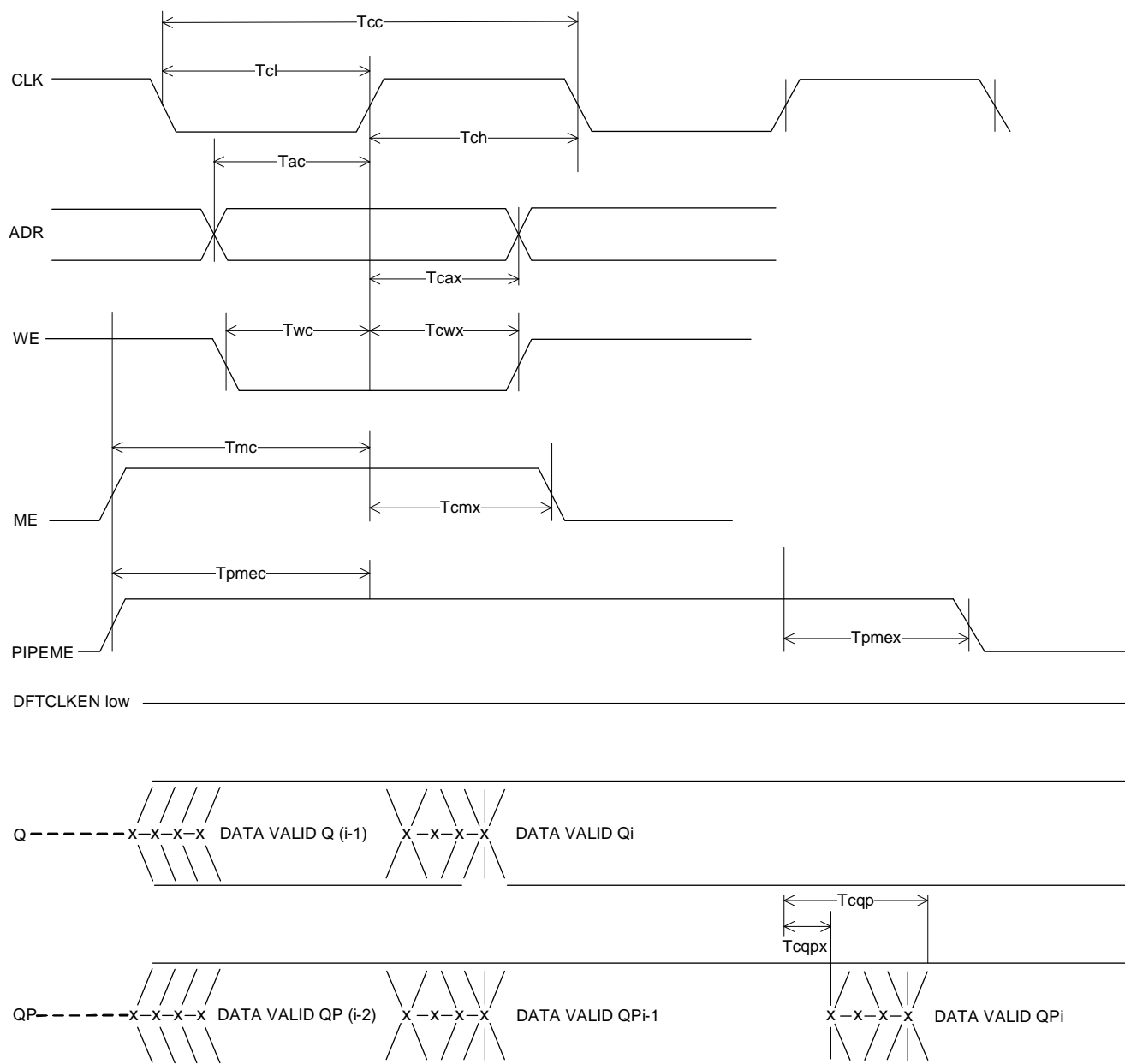
Figure 2-8 shows the conventions used in the waveform diagrams.

Figure 2-8 Waveform Conventions



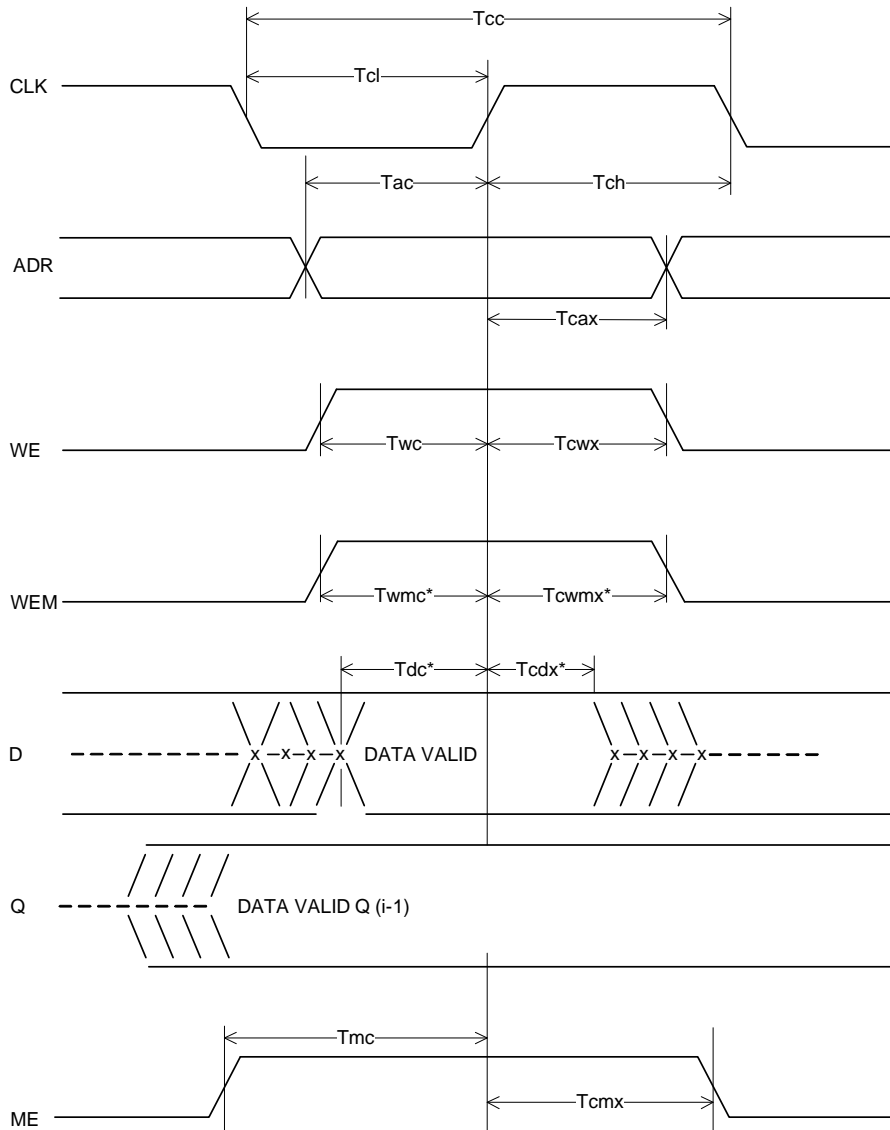
**Figure 2-9 Read Cycle Timing**



**Figure 2-10 Read Cycle Timing: Pipeline Mode**



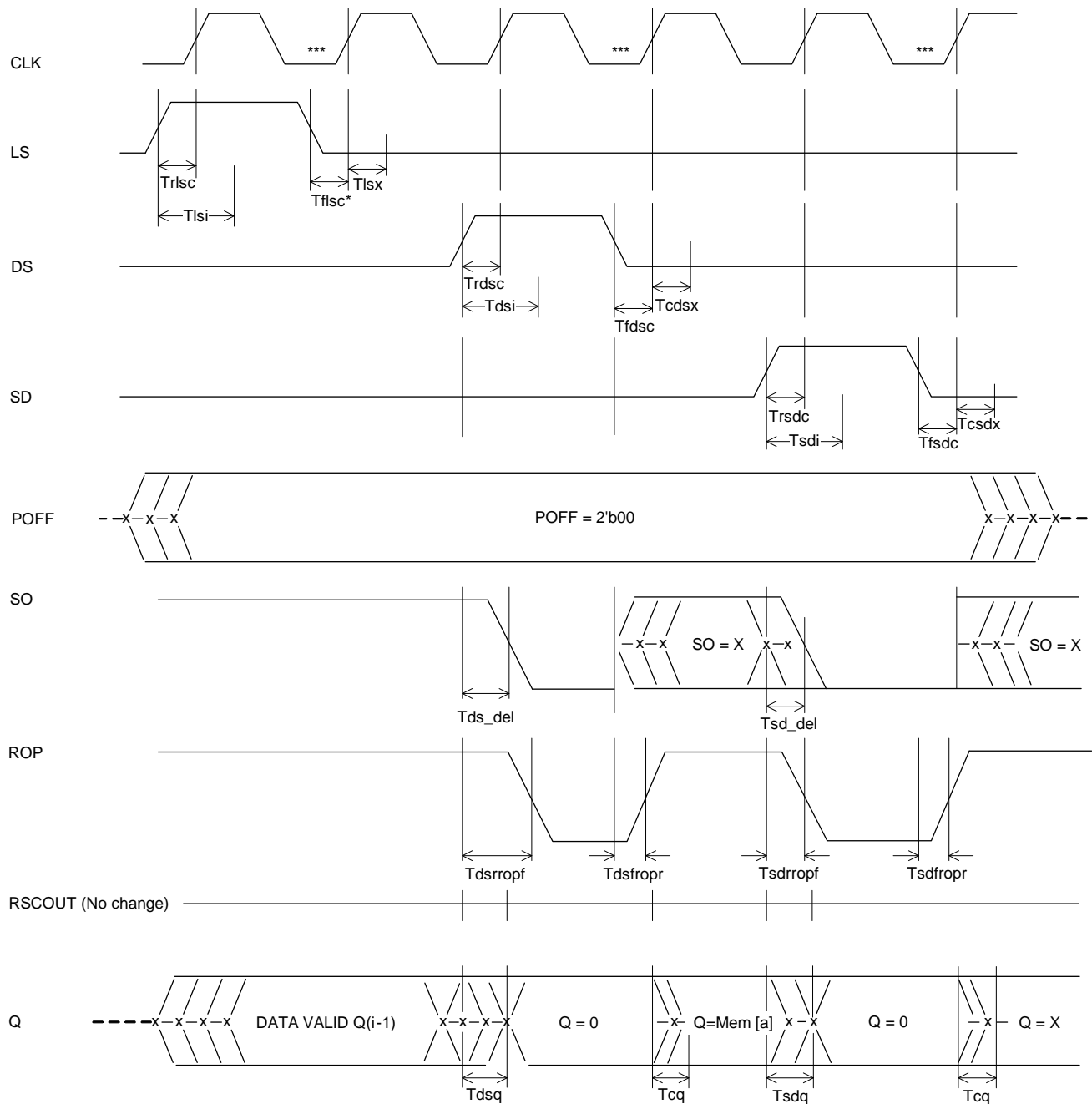
**Figure 2-11 Write Cycle Timing**



**Note**

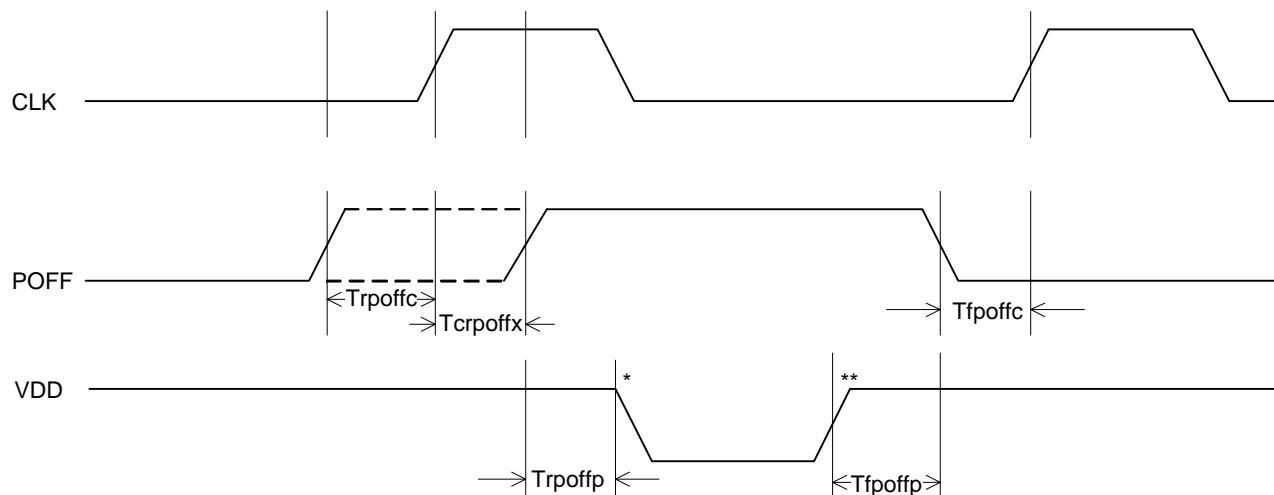
In the above waveform,

- $T_{dc}^* = T_{d1c\_l}/T_{d2c\_l}/T_{d3c\_l}/T_{d4c\_l}/T_{d5c\_l}/T_{d6c\_l}/T_{d7c\_l}/T_{d8c\_l}/T_{d1c\_h}/T_{d2c\_h}/T_{d3c\_h}/T_{d4c\_h}/T_{d5c\_h}/T_{d6c\_h}/T_{d7c\_h}/T_{d8c\_h}$
- $T_{cdx}^* = T_{cd1x\_l}/T_{cd2x\_l}/T_{cd3x\_l}/T_{cd4x\_l}/T_{cd5x\_l}/T_{cd6x\_l}/T_{cd7x\_l}/T_{cd8x\_l}/T_{cd1x\_h}/T_{cd2x\_h}/T_{cd3x\_h}/T_{cd4x\_h}/T_{cd5x\_h}/T_{cd6x\_h}/T_{cd7x\_h}/T_{cd8x\_h}$
- $T_{wmc}^* = T_{wm1c\_l}/T_{wm2c\_l}/T_{wm3c\_l}/T_{wm4c\_l}/T_{wm5c\_l}/T_{wm6c\_l}/T_{wm7c\_l}/T_{wm8c\_l}/T_{wm1c\_h}/T_{wm2c\_h}/T_{wm3c\_h}/T_{wm4c\_h}/T_{wm5c\_h}/T_{wm6c\_h}/T_{wm7c\_h}/T_{wm8c\_h}$
- $T_{cwmx}^* = T_{cwm1x\_l}/T_{cwm2x\_l}/T_{cwm3x\_l}/T_{cwm4x\_l}/T_{cwm5x\_l}/T_{cwm6x\_l}/T_{cwm7x\_l}/T_{cwm8x\_l}/T_{cwm1x\_h}/T_{cwm2x\_h}/T_{cwm3x\_h}/T_{cwm4x\_h}/T_{cwm5x\_h}/T_{cwm6x\_h}/T_{cwm7x\_h}/T_{cwm8x\_h}$

**Figure 2-12 Power Down Mode Timing****Note**

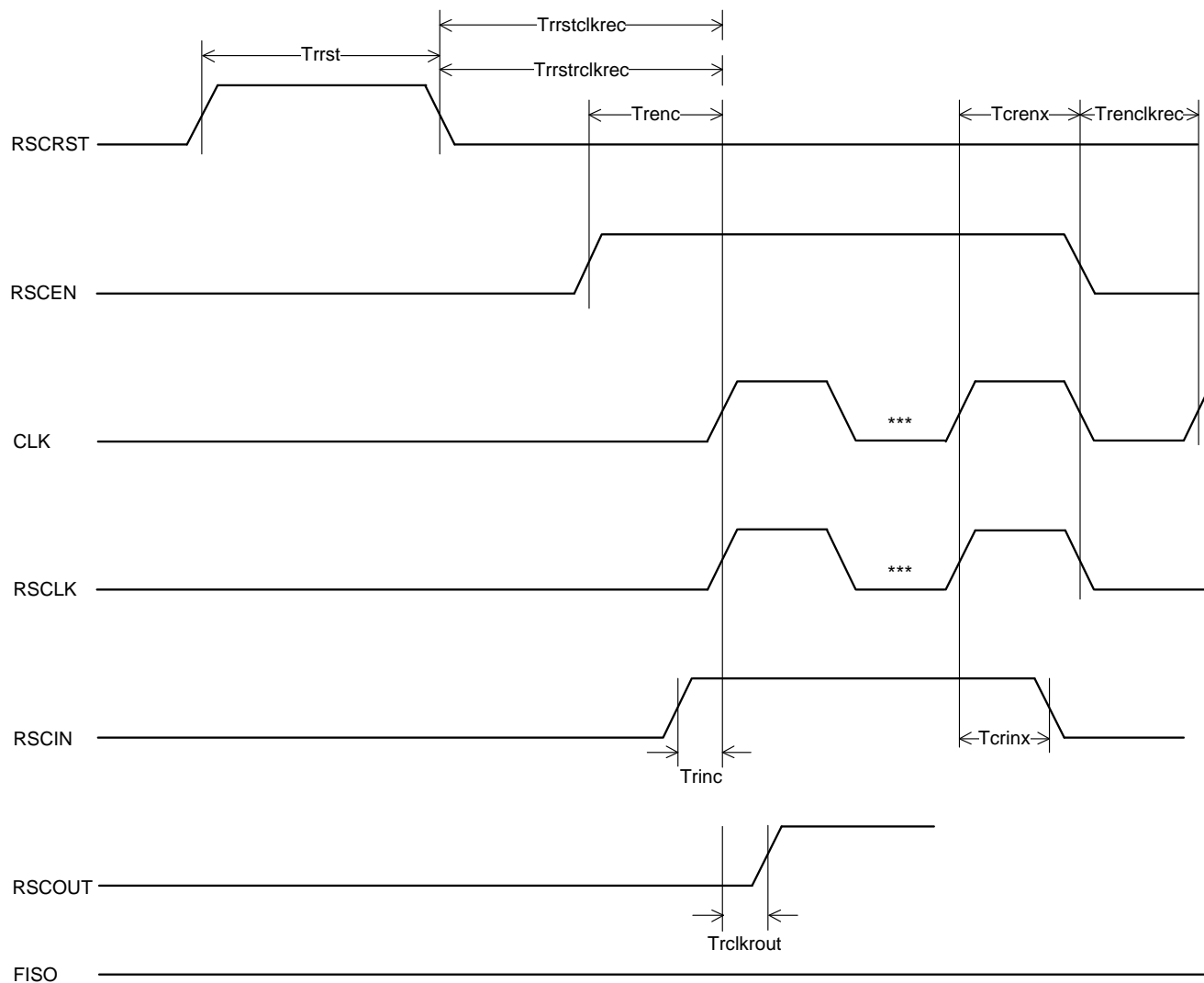
- An alternate solution to avoid memory access during wake-up is to set the MEA/MEB pin to inactive state for at least " $T_{lsc}^*/T_{fsc}/T_{fsc}$ " time interval after the "LS/DS/SD" falling edge.
- $T_{ds\_del}/T_{sd\_del} \rightarrow T_{dssd\_l/h}$ ,  $T_{dssq\_l/h}$ ,  $T_{dssc}$  &  $T_{dsd\_l/h}$ ,  $T_{dsdq\_l/h}$ ,  $T_{dsdc}$
- $T_{lsc}^* = T_{fls0c}$  (When  $BC0=0$ ) /  $T_{fls1c}$  (When  $BC0=1$ )

**Figure 2-13 Periphery Off mode**

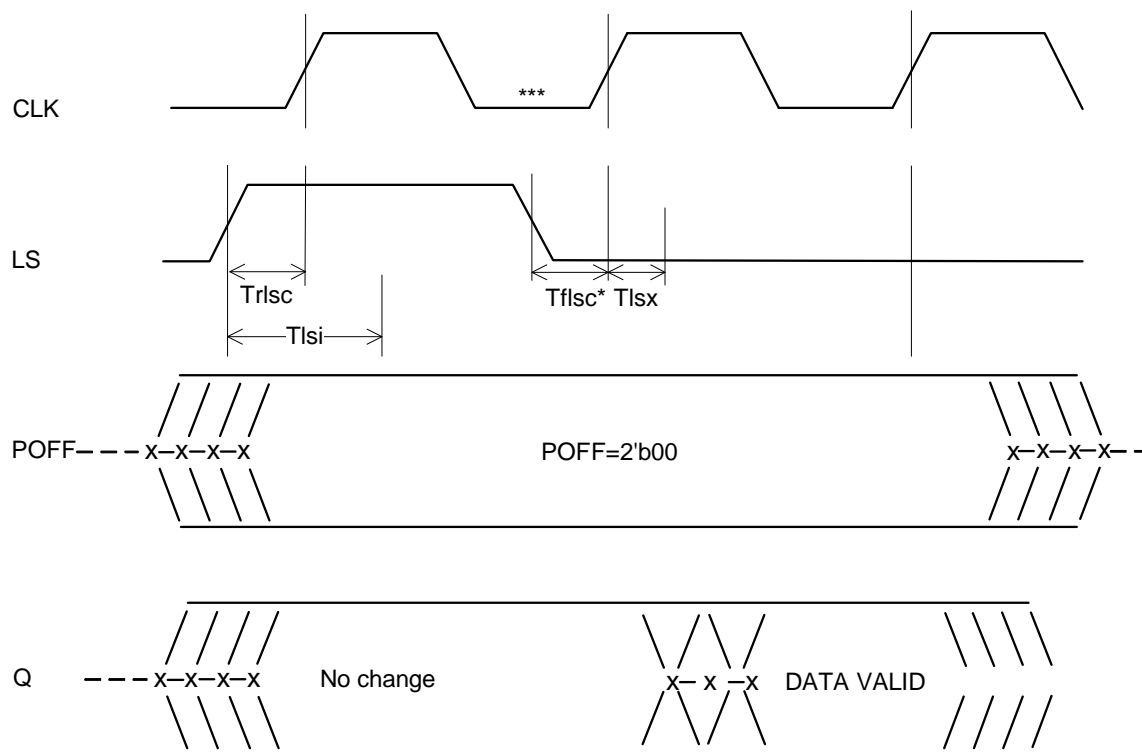


**Note**

- \* VDD starts to fall, or goes floating.
- \*\* VDD has reached at 90% level after wake-up.

**Figure 2-14 Normal Power-up Redundancy Scan Timing (redundancy\_enable=TRUE)**

**Figure 2-15 Light Sleep Mode Timing**



**Note**

- An alternate solution to avoid memory access during wake-up is to set the ME pin to inactive state for at least " $T_{flsc}^*$ " time interval after the "LS" falling edge.
- $T_{flsc}^* = T_{fls0c}$  (When  $BC0 = 0$ ) /  $T_{fls1c}$  (When  $BC0 = 1$ )

## 2.3.14 Standard Operating Characterization Conditions



### Note

- VDD: Periphery Voltage
- VDDA: Array Voltage
- VNW\_N: Nwell back bias Voltage for Forward Body Bias (FBB) Schemes.
- VPW\_P: Pwell back bias Voltage for FBB Schemes.

Table 2-17 shows the typical, best, and worst operating characterization conditions supported by this compiler. Also, additional characterization is performed to support the Dual Rail option (`vdda_enable=TRUE`). This characterization is performed with separate array and periphery voltages and the specified temperature and process conditions as described in Table 2-18.

**Table 2-17 Standard Operating Conditions - Single Rail**

PVT Corner	Process	VDD (V)	Temperature (C)	VDDA (V)	VNW_N (V)	VPW_P (V)	RC Extraction
tt0p8v25c_0p8v_0v_0v	TT	0.8	25	0.8	0	0	Nominal
ssg_fcmax0p72v125c_0p72v_0v_0v	SSG_FCMA	0.72	125	0.72	0	0	FuncCmax
ssg_fcmax0p72vn40c_0p72v_0v_0v	SSG_FCMA	0.72	-40	0.72	0	0	FuncCmax
ssg_fcmax0p72vn40c_0p72v_0p7v_n1p5v	SSG_FCMA	0.72	-40	0.72	0.7	-1.5	FuncCmax
ssg_fcmax0p72v125c_0p72v_0p6v_n1p0v	SSG_FCMA	0.72	125	0.72	0.6	-1.0	FuncCmax
tt0p8v85c_0p8v_0v_0v	TT	0.8	85	0.8	0	0	Nominal

**Table 2-18 Standard Operating Conditions - Dual Rail**

PVT Corner	Process	VDD (V)	Temperature (C)	VDDA (V)	VNW_N (V)	VPW_P (V)
tt0p65v25c_0p8v_0v_0v	TT	0.65	25	0.8	0	0
tt0p65v85c_0p8v_0v_0v	TT	0.65	85	0.8	0	0

## 2.3.15 Using Licensed Custom PVTs

Customers may license additional characterized PVTs, beyond the standard PVTs listed in [“Standard Operating Characterization Conditions”](#) on page 56, for the compiler.

The complete list of PVTs and the corresponding licenses required to generate them can be found in the custom GLB (<compiler\_name>\_custom.glb) file shipped with the compiler. In order for instances to be generated with these PVT, the custom GLB file will need to be edited to set the appropriate values for the variable "pvt\_enable". For example, `pvt_enable = {1 2 3 4 5}`.

PVTs specific to dual Rail can be enabled by setting `vdda_enable=TRUE` in the custom GLB or selecting Dual Power Supply = TRUE in the GUI during instance generation.

Refer to Chapter 6 of the Integrator Manual for steps to edit the custom GLB file.





# 3

## Compiler Source and Output Files

This chapter describes the directory structure of the compiler database, files, and templates.

### 3.1 Using Compiler Library Files

The files that make up the compiler are located in the **<compiler>** directory. Each compiler directory is located in a compiler library. See the Embed-It! Integrator User Guide for details on specifying the compiler library path information. [Table 3-1](#) provides a description of these files.



#### Note

The files described in the figures and tables in this chapter represent a complete list of the files that may be included in a Synopsys compiler. However, these files are independent of the technology/process and some of them may not be included and/or supported by the current compiler. Contact Synopsys support for additional assistance.

**Table 3-1** Files in the **<compiler>** Library

Filename	Contains
<b>User-editable files</b>	
<Compiler_name>_custom.glb	Customer-specific overrides to flags in the compiler global parameters (glb) file.
design2<foundry>_xltd.csv	GDSII layer mapping details - the design layer numbers are mapped to foundry layers.
<b>Read-only files</b>	
release.txt	Information about the changes made to the compiler in each patch release of the compiler such as patch notes and release notes.
<b>Information files</b>	
check.tcl	Tcl file to check range of the values for NW, NB, CM and SW for the compiler.
gdsview.cfg	Layer display and window position options for the gdsview tool.
<Compiler_name>.cir	Netlist file that represents the circuit design. It is used for LVS.

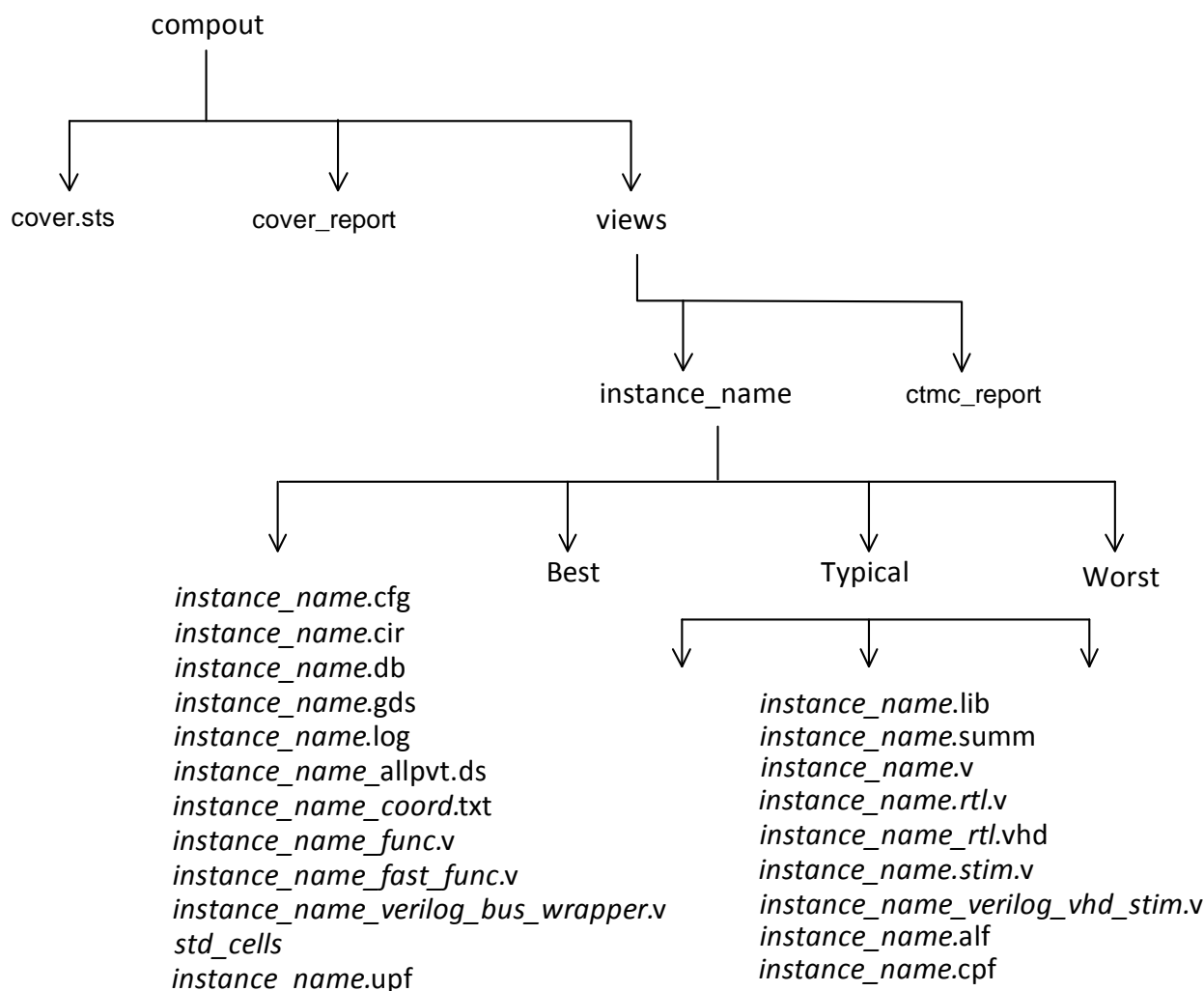
**Table 3-1 Files in the <compiler> Library**

Filename	Contains
<Compiler_name>_cells_info.txt	The file that represents the port information for the leaf cells used in building a memory instance.
<Compiler_name>oa/compiler_lib	Open Access database directory containing the leaf cells layout used in building a memory instance.
<Compiler_name>_caf.py	Memory instance structure information placement file. It contains the layout tiling and netlist generation instructions to build an instance in the compiler range.
<Compiler_name>.v	Switch-level representation of all cells used to build instances. The macros defined in this library are called by the switch-level netlist generated by the placement file <b>&lt;Compiler_name&gt;.pf</b> .
<Compiler_name>.glb	Global parameters file to set up compiler specific instance generation options. The other compiler-specific files are referenced in this file.
<Compiler_name>.prm	Parameter definitions file for the compiler.
<Compiler_name>.cpj	This is a database file for AutoChar enabled compilers. It contains all the information that is required for characterization of the compiler.
*.tpl	Front-End Template files (encrypted).

## 3.2 Accessing Output Files

The compiler generates output files and directories and stores them in the compout directory, which is the root output directory. [Figure 3-1](#) shows an example of a compout directory structure.

**Figure 3-1 compout Directory Structure Example**



### 3.2.1 compout Directory

[Table 3-2](#) lists and describes the contents of the compout directory.

**Table 3-2 compout Directory**

Filename	Description
cover.sts	Shows the error and warning statistics during instance generation.

**Table 3-2 compout Directory**

Filename	Description
cover_report	Reports the number of errors and warnings after the COVER file completes execution.
views/	Directory that contains the ctmc_report and the instance directory. See <a href="#">compout/views Subdirectory</a> .

**3.2.1.1 compout/views Subdirectory**

[Table 3-3](#) lists and describes the compout/ views directory files, ctmc\_report and *instance\_name*. The *instance\_name* is substituted with the name of the memory instance generated.

**Table 3-3 compout/views Subdirectory**

Filename	Description
ctmc_report	Reports the number of errors and warnings after the instance is generated.
<i>instance_name</i> /	Instance directory. See <a href="#">compout/views/instance_name Subdirectory</a> .

**compout/views/instance\_name Subdirectory**

[Table 3-4](#) lists and describes the compout/ views/instance\_name subdirectory.

**Table 3-4 compout/views/instance\_name Subdirectory**

Filename	Description
Best/	Subdirectory containing Front-End views for the best operating point.
Typical/	Subdirectory containing Front-End views for the typical operating point.
Worst/	Subdirectory containing Front-End views for the worst operating point.
<i>instance_name</i> .cfg	Instance configuration file
<i>instance_name</i> .cir	Instance SPICE netlist for LVS
<i>instance_name</i> .db	Synopsys' internal database that records all options set for this instance.
<i>instance_name</i> .gds	Instance GDSII view
<i>instance_name</i> .log	Instance log file
<i>instance_name</i> .plef	Physical LEF view of the ring structure. Includes instance blockages.
<i>instance_name</i> _allpvt.ds	All PVT datasheet that provides pin descriptions, logic truth table, timing characterization, Read and Write cycle timing, and power dissipation.
<i>instance_name</i> _coord.txt	File that contains the x and y coordinates of the power ring edges and memory core in the instance.

**Table 3-4 compout/views/instance\_name Subdirectory**

Filename	Description
<i>instance_name_func.v</i>	A switch-level Verilog netlist file. The same switch-level or transistor-level netlist that the compiler generates in SPICE format can also be generated in Verilog. The difference is that the Verilog netlist can be used for logic simulation and not for LVS.
<i>instance_name_fast_func.v</i>	Verilog model with limited functionality to speed up the simulation time. It is useful during initial phase of implementation and should not be mistaken for a signed-off quality Verilog model. This model supports only basic read/write operations (without BIST muxes), LS, DS, SD and PIPEME functionality. It supports X-handling and "virage_ignore_read_addx" option. The <code>\$readmemh</code> task can be used to load the memory.
<i>instance_name_verilog_bus_wrapper.v</i>	Verilog file used to test the basic functionality of the memory without consideration for BIST or ATPG.
<i>std_cells.v</i>	Verilog model for the dummy standard cells used in Verilog netlist for ATPG.
<i>instance_name_hcell.txt</i>	Hcell equivalence file for hierarchical LVS matching with Calibre.
<i>instance_name.clp</i>	Verilog Model used in conformal Low Power tool. This model captures power management information for the memory core.
<i>instance_name.ctl</i>	Core Test Language Model that contains scan chain information for the memory core. This information is used for testing at the system level.
<i>instance_name.upf</i>	IP UPF view for the memory instance. 'Unified Power Format', IEEE standard low power format to model the power intent.

### 3.2.2 compout/views/instance\_name/<pvt\_name> Subdirectory Files

Table 3-5 shows the files in the compout/views/instance\_name/<pvt\_name> Subdirectory. The directory representing each process condition (best, typical, and worst) contains a list of these files.

**Table 3-5 compout/views/instance\_name/<pvt\_name> Subdirectory Files**

Filename	Description
<i>instance_name.lib</i>	Synopsys timing model
<i>instance_name.v</i>	Instance Verilog model
<i>instance_name_rtl.vhd</i>	VHDL RTL core model
<i>instance_name_stim.v</i>	Verilog behavioral model testbench
<i>instance_name_verilog_vhd_stim.v</i>	Verilog behavioral model and VHDL behavioral model testbench
<i>instance_name.cpf</i>	CPF view for the memory instance. 'Common Power Format', Si2 low power format to model the power intent.



# 4

## Using Front-End Files

---

This chapter describes the Front-End files, which support documentation, simulation and testing of memory instances. The Front-End files are the raw data file, the delay equation file, and the template files. They provide the Front-End views of the memory instance. The following sections describe these files:

- [“Using Template Files”](#) on page 75
- [“Setting Front-End Flags”](#) on page 80



### Note

For detailed description of all output views, refer to the “Working with Output Views” chapter of the Embed-It!® Integrator User Manual.

---

### 4.1 Using Template Files

The compiler library contains a template for each Front-End view. The compiler software uses these templates to generate the corresponding Front-End files.

The following sections describe the Front-End views that Synopsys, Inc. supports.

- [“Using ATPG Tools”](#) on page 76
- [“Using MemoryBIST Tools”](#) on page 76
- [“Performing Power Analysis”](#) on page 77
- [“Performing Simulations”](#) on page 77
- [“Performing Static Timing Analysis”](#) on page 78
- [“Using Switch-Level Verilog Models”](#) on page 79
- [“Power Verilog Models”](#) on page 79
- [“Low Power Formats”](#) on page 80

### 4.1.1 Using ATPG Tools

Synopsys, Inc. supports the ATPG tools TetraMAX and FastScan.

The Synopsys compilers use the **mc\_spf.tpl** template file as the initialization file for Tetramax. The **mc\_fastscan.tpl** template file generates the FastScan (**.fs\_lib**) model and the **mc\_atpg\_netlist.tpl** generates a Verilog Netlist for ATPG (**\_atpg\_netlist.v**). The template provides both FastScan and mBIST solutions based on how you set the Front-End flag **mbist\_on**.

- If **mbist\_on=0**, the **.fs\_lib** model provides only FastScan solution.
- If **mbist\_on=1**, the **.fs\_lib** model provides both FastScan and mBIST solutions.
- Use the **mc\_tpf.tpl** template file to obtain test procedures for FastScan

### 4.1.2 Using MemoryBIST Tools



#### Note

This section is not applicable for **bist\_enable=TRUE** or **redundancy\_enable=TRUE**. When the **bist\_enable** option is enabled, the generated memory instance includes multiplexers (muxes) for all address, control and data signals as well as comparators and capture logic. Third party memory BIST will not yet support this additional logic.

Synopsys, Inc. provides both BIST and Scan solutions to verify memory. The time taken to scan in inputs and scan out results increases the total test time. The BIST models generate synthesis scripts for the BIST controller and test cases based on standard algorithms.

Synopsys, Inc. supports the STAR Memory System and third party MemoryBIST tools. The STAR Memory System is more advantageous when integrated test logic is included when compiling the memory instances. The integrated test logic enables high speed test and reduces the routing congestion between the memory instance and the wrapper.

Synopsys' memory compilers support third party Tessent MemoryBIST and BoundaryScan solutions with views generated by the **mc\_fastscan.tpl** template file.

- If **mbist\_on=0**, the **.fs\_lib** model provides only BoundaryScan solution.
- If **mbist\_on=1**, the **.fs\_lib** model provides both BoundaryScan and MemoryBIST solutions.

The MemoryBIST view is generated using the **mc\_membist.tpl** template file. The **.membist** file is the memBISTGenerate configuration file. It contains the:

- MemoryController wrapper file - Describes each memory instance
- MemoryCollar template wrapper file - Describes the sequence of memory tests. This file references a MemoryTemplate in the memory library file **.lvlib**.

The **.lvlib** library file contains the MemoryTemplate wrapper, which describes the ports and behavior of the memory.



## 4.1.3 Performing Power Analysis

Synopsys, Inc. supports the power analysis model Advanced Library Format (ALF).

### 4.1.3.1 Using ALF

Use the **mc\_alf.tpl** template file to generate the ALF view. The view consists of library and cell sections. The cell section contains the pin descriptions and input pin constraints. It also contains the power information for the memory similar to the Synopsys model. Use the ALF format to perform power analysis with the help of Powertheatre from Sequence Design.



#### Note

The Synopsys ALF format denotes the Advanced Library Format. There are other products such as the Cadence Ambit tool format Ambit Library Format, which are different from the Synopsys ALF.

## 4.1.4 Performing Simulations

The following sections describe the simulation models that Synopsys, Inc. supports:

- [“Using Verilog Behavioral Model”](#) on page 77
- [“Using Verilog RTL Model”](#) on page 78
- [“Using VHDL RTL Model”](#) on page 78

### 4.1.4.1 Using Verilog Behavioral Model

The simulation model that Synopsys, Inc. supports is Verilog Behavioral Model.

Use the **mc\_verilog.tpl** template file to generate the **.v** Verilog behavioral model. This model describes the logic in different modes with every possible transition of every pin along with X handling and timing violations. The timing values include all path delays, input pins, constraint timing, and recovery.

To help perform simulations with parallel testbench from ATPG tools like Mentor fastscan & Synopsys tetramax, the Verilog behavioral model contains dummy standard cells with exactly same hierarchy as ATPG netlist.

The Compiler contains the **std\_cells.v** file. This file is a Verilog model for the dummy standard cells used in Verilog netlist for ATPG.

To perform verilog simulations, use verilog **<mem>.v -v std\_cells.v**.

When Verilog Behavioral model is used with command line arguments **+define+VIRAGE\_FAST\_VERILOG**, the model will act as the RTL model describing only memory functions in all modes. It does not contain timing information.

## Verilog Compiler Directives

The verilog behavioral model supports following compiler directives to provide flexibility and extra features to the customers.

- **VIRAGE\_FAST\_VERILOG:** When this flag is used, the model will act as the RTL model describing only memory functions in all modes. It does not contain timing information. However, verilog modeling this mode is faster during simulations by 20 - 40% (the range depends on the Simulator and the instance size), when compared to regular behavioral mode with full timing.
- **MEM\_CHECK\_OFF:** This flag turns off most display messages from verilog model. These display messages are mostly warnings about hazardous conditions.
- **virage\_ignore\_read\_addr:** By default, memory is corrupted when address is unknown during Read or Write cycle. However, if you think it is extra conservative to corrupt memory during read, you can use this switch to make sure memory is not corrupted during unknown address in a Read cycle.
- **VIRAGE\_IGNORE\_RESET:** This flag ignores the initialization required by the memory verilog model.

**Note**

An application note is available for a detailed description of Verilog simulation. Contact the Synopsys Support Team at <http://solvnetplus.synopsys.com>.

#### 4.1.4.2 Using Verilog RTL Model

Use the **mc\_verilog\_rtl.tpl** template file to generate the **.v** model and use the “**VIRAGE\_FAST\_VERILOG**” define option to simulate this model in Verilog RTL. The Verilog RTL model is the main RTL functional model that describes all functions in different modes. It does not contain timing information.

#### 4.1.4.3 Using VHDL RTL Model

Use the **mc\_vhd\_rtl.tpl** template file to generate the VHDL RTL **model\_rtl.vhd**. The VHDL RTL model is the core RTL model that describes the memory functions in all modes. It does not contain timing information.

### 4.1.5 Performing Static Timing Analysis

The timing library models contain:

- Memory characterization data in a specific format
- Path delays with transition values for corresponding to slew rates and output loads
- Timing values for input pin constraints such as setup/hold, period, width, and recovery

Use the timing model files to generate SDF data and back-annotate the SDF data into the Verilog/Vital models and verify functionality and timing.

The following section describes the timing analysis tools that Synopsys, Inc. supports:

- [Using Synopsys .lib Models](#)

#### 4.1.5.1 Using Synopsys .lib Models

Use the **mc\_syn.tpl** template file to generate the Synopsys **.lib** model.

The Synopsys **.lib** model contains:

- Delay model - Provides table lookup information for delay calculations

- Environment conditions - Provides various operating conditions and k-factors for circuit timing evaluation.
- Lookup table - Describes timing information for:
  - Describing delays
  - Specifying constraints

The Synopsys **.lib** model:

- Enables Synopsys tools with library information including cell timing function, physical, power, test data for optimized tools performance while meeting the technology requirements
- Makes the library development process faster through simplified data entry
- Provides low cost library maintenance by using a single library source while minimizing library generation and verification costs
- Offers comprehensive modeling capabilities for design flows
- Develops high quality library that produces better designs
- Generates VHDL test-benches, and eases transition to the VITAL '95 standard by leveraging existing synthesis source library
- Offers a reliable, and low-cost, library development solution that is available and ready to use today
- Produces protected platform independent binary library file for delivery to customers

#### 4.1.6 Using Switch-Level Verilog Models

The same switch-level or transistor-level netlist that the compiler generates in SPICE format can also be generated in Verilog. The changes in the Verilog format include replacing the bit cell with a behavioral model, adding delays for simulation, and changing gates to Verilog primitives. The Verilog netlist is generated from the same source that generates the SPICE netlist both for the compiler libraries and instance-specific modules. The output filename is **<instance>\_func.v**. The difference is that you can use the Verilog netlist for logic simulation and not for LVS.

#### 4.1.7 Power Verilog Models

The Power Verilog Model, **\*.mv**, is similar to existing Verilog Behavioral Model. It contains power pins in the port list of memory as well as includes their functionality in the model. There is no role of power pins in the timing checks. The power pins are only included in the functional part of the view.

The following items are modelled with respect to the power pins:

If the power pins are not valid (VDD/=1 or VSS/=0) then the memory is corrupt (not in case of ROM) and outputs, and no memory operation is allowed.

If the power pins are valid (VDD=1 & VSS=0) then the power model will behave similar to Verilog Behavioral Model, that is, the memory will operate normally.

## 4.1.8 Low Power Formats

There are two views provided to support Low Power Formats, IP UPF - "<instance\_name.upf>" and CPF - "<instance\_name.cpf>" These views model power intent for the memory instance.

The views include power domains, power state transitions and isolations.

## 4.2 Setting Front-End Flags

This section describes the Front-End flags that you can set for your compiler.

You can set or change the Front-End flags in three ways:

- When using the GUI, you can set or change the Front-End flags to affect all instances generated by the compiler by defining them in the custom global file.
- When using the GUI or when running ISH, you can change the Front-End flags to affect all instances generated in the project.
- When using VMC, you can set or change the Front-End flags for a particular instance by defining them in the VMC configuration file (**filename.cfg**).

In the **custom.glb** file or the VMC configuration file, you can set parameters with commands in the format:

```
parameter=value
```

In the ISH script, you can set parameters with commands in the format:

```
component <obj> set_parameter <string:parameter_name> <string:parameter_value>
```

In all cases, each parameter setting is a separate line in the file, and they can appear in any order.

See the "Global Parameter (GLB) Reference" chapter of the Embed-It!® Integrator User Manual for a complete list of the Front-End flags and their descriptions.

Also refer the "Using Embed-It! Integrator Shell (ish)" chapter for additional information on the use of ISH commands.



The parameters that are supported by the current compiler are described in the compiler's custom global file. For further information please contact the Synopsys Support Team at <http://solvnetplus.synopsys.com>.

# 5

## Using Back-End Views

---

This chapter describes the Back-End views, which contain the files that support layout and fabrication. The compiler supports the following Back-End views:

- “Using LEF Models” on page 81
- “Using GDSII” on page 82
- “Using SPICE Netlists” on page 82
- “Reading the Bitmap (Scramble) Information” on page 83

### 5.1 Using LEF Models

Layout Exchange Format (LEF) provides an abstract for the memory. This is the minimum information required by the place and route tools to handle the memory macro generated by the compiler. The abstract defines the size of the macro, pin names and locations, pin layers, pin capacitance, and (optionally) diode/diffusion area information. It also includes all of the physical segments generated by the Power and Pin Router (PPR), which can be associated either with power routing or signal routing. You can also define obstruction regions (regions that the router must avoid during global or detail routing) in LEF. The macro core is marked by obstruction to avoid capacitance coupling crosstalk with critical internal signals such as bit or bitbar, se or seb.

The LEF view contains only the MACRO and SITE definition. You must provide technology definitions (layer names, layer properties, and viagens) that should be loaded into the router before reading the LEF view. LEF layer names are set up in the global file to match the layer names from the technology definition file.

Please contact Synopsys Support Team at <http://solvnetplus.synopsys.com> or refer to the Embed-It! software documentation for further information to change layer names, change LEF version, or to add other optional LEF information.

## 5.2 Using GDSII

The GDSII file contains layouts of the leaf cells in the GDSII format. It contains the mask data used to create the macro. It is also used to extract the physical layout netlist for LVS verification.

To change the GDSII layers used, you can modify the **file\_xltd.csv** file in the compiler directory or reference another layer map file using the **compiler\_xltd\_file** variable. For example:

```
set cfg(compiler_xltd_file) <path>/file_xltd.csv
```

**Note**

Refer to the “GDSII Layers” chapter of the Embed-It!® Integrator User Manual for details on how to use XLTT.

### 5.2.1 Naming GDSII Libraries

Synopsys’ DesignWare memory GDSII library names start with a prefix of the form **Mnnnxxn\_**. The prefix denotes the internal compiler design project name and the library revision. It is applied to every cell in the GDSII library and ensures that there are no conflicts in cellnames when you combine instances from different compilers or different versions of a compiler into a design. The maximum length of a cell name in the compiler library is 32 characters so that they are compatible with industry-standard layout tools.

Apart from compiler-specific leaf cells, an instance also contains compiler-generated, instance-specific cells. The instance-specific cellnames start with a user-defined prefix of an instance name. Compiler-generated block names can have a maximum length of 14 characters because if an instance name is 17 characters long, the total length of any generated cell name is a maximum of 32 characters.

### 5.2.2 Using the Extended Layer Translation Table File

The Extended Layer Translation Table (XLTT) file maps the Design layers to the foundry layers. Layers that are not needed for production are deleted (marked with a D in the XLTT file).

## 5.3 Using SPICE Netlists

Embed-It! Integrator generates the SPICE netlist (transistor-level netlist) for the macro. You can use this SPICE netlist only for LVS verification.

The compiler SPICE netlist contains the subcircuit representations of leaf cells used for building an instance. The compiler library of subcircuits use the same prefix as the GDSII library. So, you can combine netlists when combining instances from different compilers on to one design. The SPICE subcircuits match with the GDSII leaf cells. Cells that match GDSII cell contents (for hierarchical LVS) are defined in the **<instance>\_hcell.txt** output file produced for each instance.

The SPICE netlist generated for an instance uses the compiler subcircuit library as the base, and adds the subcircuit definitions for all generated blocks to it.

## 5.4 Reading the Bitmap (Scramble) Information

The bitmap file **<instance>.bitmap** is created during memory generation. This file contains the logical to physical scrambling information related to the memory instance being generated, taking into consideration the memory type, pin names and properties, memory port information and physical organization of the memory.

Refer to the Embed-It!® Integrator User Guide for further information on reading the bit map file.





# 6

## Configuring Files

This chapter shows how to configure the following files:

- [“Expanded SiWare™ Instance Naming Convention”](#) on page 85
- [“Configuring the Custom GLB File”](#) on page 88 - The custom GLB file contains parameters that influence the memory generation tool.
- [“Configuring the Extended Layer Translation Table File”](#) on page 89 - The Extended Layer Translation Table (or XLTT) file has GDS layer mapping details and can be used to convert a compiler from one technology to another.
- [“Place and Route Considerations”](#) on page 89.

### 6.1 Expanded SiWare™ Instance Naming Convention

The automatic memory name of the instance is generated by combining information from the compiler name itself with values entered for some of the instance parameters for the given component. For example, the default name calculation generates an instance name that looks like:

s1drlssd4u2p1024x128m4b4w0c1p1d1r1rm3sd rw11

This breaks down into these segments, and further described in [Table 6-1](#).

s1	drl	s	sd4	u	2p	1024	x	128	m4	b4	w0	c1	p1	d1	r1	rm3	sd	rw11
<s1>	<drl>	<s>	<sd4>	<u>	<2p>	<1024>	x	<128>	m<4>	b<4>	w<0>	c<1>	p<1>	d<1>	<r1>	rm<3>	<sd>	rw<11>
1	2	3	4	5	6	7		8	9	10	11	12	13	14	15	16	17	18



**Note**

The base compiler name contains eight segments (aabbccdddeeffggh) as described in [Table 2-1, “Compiler Naming Convention”](#), some of which are translated into the automatic instance name.

**Table 6-1 Automatic Name Calculation Breakdown**

Auto-Name Segment	Description	Taken From:
1	product family	Base compiler name (ee)
2	product sub_family (architecture)	Base compiler name (ff)
3	protocol/clock	Base compiler name (h)
4	process	Base compiler name (ccc)
5	periphery_Vt <ull>	u - periphery_Vt=ULTRALOW l - periphery_Vt=LOW
6	number of ports	Base compiler name (dd)
7	NW value	User specified value for Number of Words (NW)
8	NB value	User specified value for Number of Bits (NB)
9	m<n>	User specified value for Column Mux (CM)
10	b<0 1>	User specified value for Number of Banks (BK) during component configuration, if use_bk=TRUE for the component.
11	w<0 1>	User specified value for Bit Write (SW) during component configuration, if use_sw=TRUE for the component. Valid values are: TRUE(1) and FALSE(0).
12	c<0 1>	User specified value for center decode (center_decode). Valid values are: TRUE(1) and FALSE(0).
13	p<0 1>	User specified value for power gating (pg_enable). Valid values are: TRUE(1) and FALSE(0).
14	d<0 1>	User specified value for dual rail (vdda_enable). Valid values are: TRUE(1) and FALSE(0).

**Table 6-1 Automatic Name Calculation Breakdown**

Auto-Name Segment	Description	Taken From:
15	<l0l1l2lr0lr1lr2ls0ls1ls2>	<p>User specified values for redundancy_enable, bist_enable, and scan_enable combinations.</p> <p>l0 - bist_enable=FALSE, row_redundancy=FALSE, redundancy_enable=FALSE, scan_enable=FALSE</p> <p>l1 - bist_enable=FALSE, row_redundancy=FALSE, redundancy_enable=TRUE, scan_enable=FALSE</p> <p>l2 - bist_enable=FALSE, row_redundancy=TRUE, redundancy_enable=TRUE, scan_enable=FALSE</p> <p>r0 - bist_enable=TRUE, row_redundancy=FALSE, redundancy_enable=FALSE, scan_enable=FALSE</p> <p>r1 - bist_enable=TRUE, row_redundancy=FALSE, redundancy_enable=TRUE, scan_enable=FALSE</p> <p>r2 - bist_enable=TRUE, row_redundancy=TRUE, redundancy_enable=TRUE, scan_enable=FALSE</p> <p>s0 - bist_enable=FALSE, row_redundancy=FALSE, redundancy_enable=FALSE, scan_enable=TRUE</p> <p>s1 - bist_enable=FALSE, row_redundancy=FALSE, redundancy_enable=TRUE, scan_enable=TRUE</p> <p>s2 - bist_enable=FALSE, row_redundancy=TRUE, redundancy_enable=TRUE, scan_enable=TRUE</p>
16	rm<0l1l2l3l4l5>	<p>Read margin timing modes. User specified value for "timing_mode"</p> <p>rm0 - timing_mode=RM0</p> <p>rm1 - timing_mode=RM1</p> <p>rm2 - timing_mode=RM2</p> <p>rm3 - timing_mode=RM3</p> <p>rm4 - timing_mode=RM4</p> <p>rm5 - timing_mode=RM5</p>
17	<sd hd>	<p>User specified value for "output_drive"</p> <p>sd - output_drive=STANDARD</p> <p>hd - output_drive=HIGH</p>
18	rw<00l0l1l0l11>	<p>User specified values for "read_assist" and "write_assist".</p> <p>rw00 - read_assist=FALSE, write_assist=FALSE</p> <p>rw01 - read_assist=FALSE, write_assist=TRUE</p> <p>rw10 - read_assist=TRUE, write_assist=FALSE</p> <p>rw11 - read_assist=TRUE, write_assist=TRUE</p>

Users can control the auto name segments and the order in which they are to be included in the generated memory name, by using `glb` parameter.

If the generated default name results in the duplication of an component name in the project, then the generated name would be appended with a numeric index. For example:

```
s1drlssd4u2p1024x128m4b4w0c1p1d1r1rm3sdrw11
s1drlssd4u2p1024x128m4b4w0c1p1d1r1rm3sdrw11_1
s1drlssd4u2p1024x128m4b4w0c1p1d1r1rm3sdrw11_2
s1drlssd4u2p1024x128m4b4w0c1p1d1r1rm3sdrw11_n
```

This situation would happen when different components are configured with the same values for the parameters described in [Table 6-1](#) and different values for other parameters not identified in this table.

## 6.2 Configuring the Custom GLB File

The custom GLB file (`<compiler>_custom.glb`) is located under the compiler directory in the compiler library.

### 6.2.1 Setting Parameters

This section shows how to set parameters in the custom global file and the COVER control file.

- Parameter names are case sensitive and must appear as entered in the custom GLB file.
- In the custom GLB file, the values that appear on the right sides of equal signs are only sample values. Instead of directly using these values, users need to enter values that correspond to their processes and designs in their custom GLB files.
- To disable a parameter setting in the custom GLB file (and not override it with a new value), assign it the value `FALSE` or `TRUE`, where `FALSE` or `0` stands for disable and `TRUE` or `1` stands for enable. For example, the setting `bist_enable=TRUE` makes the BIST and TEST pins visible in the design and allows these features to be used.
- To set a parameter in the custom GLB file, enter the following command in the file:

```
parameter=value
```

- To set a parameter in the COVER control file, enter the following command in the file:

```
set cfg(parameter) value
```

- The parameter setting is a separate line in the file. The setting can be in any order.
- Parameter values must be specified using the same syntax as presented in the custom GLB. Parameter values can be surrounded by curly braces `{ }` or double-quotes `" "`. Double quotes are required if a string value contains spaces, or if the string looks like a number that can be incorrectly reformatted such as `"0123"`. Double quotes are used in the custom GLB and configuration files for parameters whose values are character strings other than Yes or No.
- Comments in the custom GLB file must be enclosed within the symbols `/*` and `*/`. Comments in the COVER control file must be preceded by the symbol `#`.



**Note**

The parameters that are supported by the current compiler are described in the compiler's custom global file. For further information please contact the Synopsys Support Team at <http://solvnetplus.synopsys.com>.

## 6.2.2 Generating Pins

The pin table is based on pin names already defined and should not require customization. If additional customization is required, contact Synopsys, Inc. for assistance.

## 6.3 Configuring the Extended Layer Translation Table File

The XLTT file defines the GDS layer mapping. It maps the Design layer numbers to foundry layers. For further details, refer to the "GDSII Layers" chapter of the Embed-It!® Integrator User Manual.

The `compiler_xltd_file` parameter defined in the compiler GLB file specifies the name of the layer mapping file. Use the `compiler_xltd_file` to convert from one layer to another. The filename signifies the layers mapped.

For example, to represent the Design-to-<foundry> layer translation table set the following parameter in the custom GLB file:

```
compiler_xltd_file="design2<foundry>_xltd.csv"
```



**Note**

While the XLTT file is open for user modifications, it is imperative that either the VSIA tag layer or IP Marker Layer are not removed.

## 6.4 Place and Route Considerations

This family of memory compilers implements a power mesh configuration. Power mesh is created by interlaced Metal3 mesh straps of VDD and VSS over the memory. These straps run orthogonal to the Metal2 layers below and are connected to their respective VDD and VSS straps. At the chip level, users must make connections to the Metal3 straps from a higher metal.

The orientation of an instance can be adjusted using the `instance_orientation` parameter. This parameter adjusts the orientation of the entire instance at the end of instance generation. Possible values are NULL or <orientation>.

For example,

```
instance_orientation=MXR270i,  
instance_orientation=MXR90.
```



**Note**

Refer to the Integrator Manual for valid values. Consult foundry for required orientation. R270 is the compiler default and is compliant to the design rule requirements.



# Index

---

## A

Address [37](#)

ATPG [66](#)

## B

Back end views [71](#)

Best operating point [62](#)

bitmap file [73](#)

Block diagram [34](#)

Built-In Self Test (BIST) [13](#)

## C

check.tcl [59](#)

Column Mux range [37](#)

Compiler directories, using [59](#)

Compiler directory structure [59](#)

Compiler features [12](#)

compiler library files [59](#), [60](#)

    .cir [59](#)

    .gds [60](#)

    .glb [60](#)

    .pf [60](#)

    .prm [60](#)

    .v [60](#)

    custom.glb [59](#)

compout/views directory [62](#)

compout/views output files

    \_allpvt.ds [62](#)

    \_coord.txt [62](#)

    \_fast\_func.v [63](#)

    \_func.v [63](#)

    \_hcell.txt [63](#)

    \_rtl.vhd [63](#)

    \_stim.v [63](#)

    \_verilog\_bus\_wrapper.v [63](#)

    \_verilog\_vhd\_stim.v [63](#)

    .cfg [62](#)

    .cir [62](#)

    .clp [63](#)

    .cpf [63](#)

    .ctl [63](#)

    .db [62](#)

    .gds [62](#)

    .lib [63](#)

    .log [62](#)

    .plef [62](#)

    .upf [63](#)

    .v [63](#)

    std\_cells.v [63](#)

compout/views/instance\_name/Best (or) Typical  
(or) Worst directory [63](#)

cover\_report [62](#)

cover.sts [61](#)

cpj [60](#)

ctmc\_report [62](#)

Custom global file (custom.glb)

    configuring [78](#)

    setting parameters [78](#)

custom.glb [59](#)

## D

Directory structure [59](#)

dual rail [13](#)

## E

Extended Layer Translation Table (XLTT) file [72](#)

Extended Layer Translation Table file

    configuring [79](#)

## F

Front-end files [65](#)

Front-end flags, setting [70](#)

## G

GDSII file [72](#)

GDSII library, naming [72](#)

gdsview.cfg [59](#)

Generating pins [79](#)

## I

Input pin. [17](#), [18](#)

Instance directory [62](#)

## L

Layout Exchange Format models [71](#)

## M

mBIST [66](#)

memBIST [66](#)

MemoryBIST [66](#)

## N

Naming

    GDSII libraries [72](#)

## O

oa [60](#)

output files [61](#)

## P

pg\_enable [12](#)

Pins, generating [79](#)

Power analysis [67](#)

power gating [12](#)

## R

Read-only files [59](#)

release.txt [59](#)

Revision history [3](#)

## S

scramble information [73](#)

Self Time bypass [15](#)

Setting

    custom.glb file parameters [78](#)

Specifications [26](#)

SPICE netlist [72](#)

Static timing analysis [68](#)

Subword or Bit Write [17](#)

## T

Table [79](#)

Timing library models [68](#)

Typical operating point [62](#)

## U

User-editable files [59](#)

## V

vdda\_enable [13](#)

views/ [62](#)

## W

WEM.

Word [37](#)

Worst operating point [62](#)

Write [17](#)

Write-Enable Mask (WEM) pin [17](#)

## X

xltt [59](#)