SiWare Automotive Grade 1 Single Port High Density Via 12 ROM 1M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process

User Manual

DesignWare® Embedded Memories
gf22nsd41p10s1dvl01ms

EDOC-1537
Version A03
December 2020

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
June 2019	A01	Document for A01 FE/LEF compiler release.
March 2020	A02	Document to support A02 Full GDS compiler release. Part number changed from gf22nsd41p10asdvl01ms to gf22nsd41p10s1dvl01ms. Updated operating characterization conditions information as per changes in PVT licensing. Updated information on LTT support to reflect XLTT. Removed characterization data to comply with revised documentation requirements.
December 2020	A03	Document to support A03 Full GDS compiler release. Updated VMIN voltages for RM modes. Updated operating characterization conditions information as per PVTs added in the compiler.

Contents

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:

Section	Describes
Chapter 1, "Product Line Overview"	The key features of the memory compiler
Chapter 2, "Compiler Profile"	The functional features and specifications of the compiler
Chapter 3, "Compiler Source and Output Files"	The directory structure of the compiler database, files, and templates
Chapter 4, "Using Front-End Views"	The files in the front-end views
Chapter 5, "Using Back-End Views"	The back end views files that support layout and fabrication
Chapter 6, "Configuring Files"	How to configure the custom global (GLB) file, the Extended Layer TranslationTable 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 p*n* where *n* is a release number such as 1, 2, or 3 (p1, p2, or p3). See 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.	Embed-It!® Integrator User Guide
Mentor Graphics FastScan	http://www.mentor.com/dft/fastscan_ds.pdf
Synopsys TetraMAX	http://www.synopsys.com/Tools/Implementation/RTLSynthesis/Test/Pages/TetraMAXATPG.aspx

Getting Technical Assistance

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

1 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 foundry approved custom bitcell to ensure high yield and reliability.
- Memory compilers with options for advanced power management modes, such as Light Sleep and Shut Down.
- Front-End and Back-End views for integration into EDA tools from leading tool suppliers.
- Memory compilers provide a provision for Back Biasing to boost the performance.

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.

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

- Fine-grained Power Gating
- Use of Long channel devices
- Integrated Power Gating

- Integrated Level Shifters

1.2 SiWare™ High-Density ROM Memory Compilers

The SiWare High-Density ROM memory compilers offer two configuration modes, one of which supports Test. 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 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 and Synchronous data flow with external clock.

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.

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

Table 1-1 SiWare High-Density ROM Memory Options

Configuration	Parameters Required

SiWare-LT
(default configuration)
	`bist_enable=FALSE`
	`pg_enable=FALSE`

SiWare-IT
(LT + Scan Chain + BIST Muxes)
	`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 and Figure 1-2 show a graphical representation of the above mentioned modes.

Figure 1-1 SiWare-LT

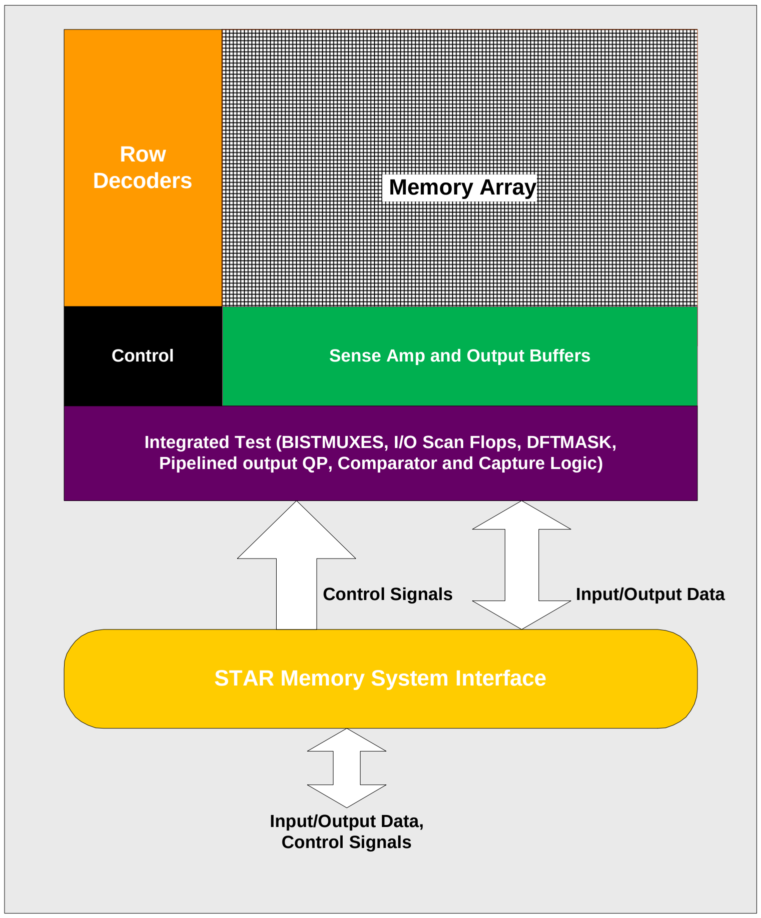
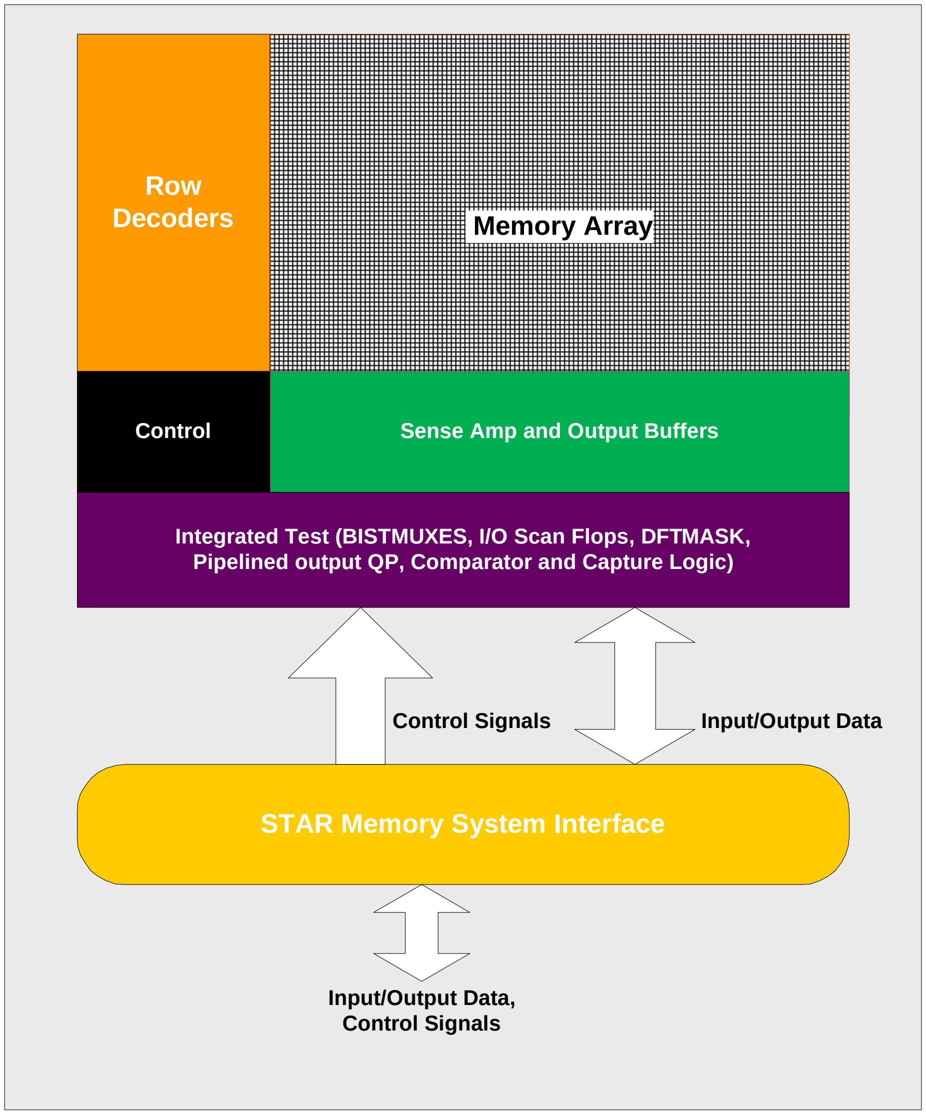

Row Decoders
Memory Array
Control
Sense Amp and Output Buffers
Integrated Test (BISTMUXES, I/O Scan Flops, DFTMASK, Pipelined output QP, Comparator and Capture Logic)
Control Signals
Input/Output Data
STAR Memory System Interface
Input/Output Data, Control Signals

Figure 1-2 SiWare-IT


Row Decoders
Memory Array
Control
Sense Amp and Output Buffers
Integrated Test (BISTMUXES, I/O Scan Flops, DFTMASK, Pipelined output QP, Comparator and Capture Logic)
Control Signals
Input/Output Data
STAR Memory System Interface
Input/Output Data, Control Signals

1.3 Compiler Features

The High-Density ROM compilers provide the following features:

- "Power Management" on page 13
- "BIST Interface" on page 14
- "Self Time Bypass" on page 14
- "Read Margin Control" on page 14
- "Enable Bitwise Corruption" on page 16
- "Back Bias" on page 16
- "Performance Boosters" on page 18
- "Optional Periphery Transistor Threshold Voltage Selection" on page 18

1.3.1 Power Management

The 22nm FD SOI ROM 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	**LS (Light Sleep)** - Provides leakage reduction with fine-grained power gating.
`TRUE`	Advanced (IPG)	**LS (Light Sleep)** - Provides leakage reduction with fine-grained power gating. **SD (Shut Down)** - When the SD pin is asserted, there is a complete shutdown and the memory outputs are held low.

Attention A wake-up (Ready for Operation=ROP) pin is also provided in the Shut Down mode 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.

1.3.2 BIST Interface

The SiWare-IT option 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.

Several pins are added to support BIST. Refer to Table 2-2, "Input Pin Description".

1.3.3 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.

1.3.4 Read Margin Control

The memory array bitlines are pre-charged prior to a memory cell access. After accessing the memory (Read cycle), bitline voltage changes depending upon the bitcell content. The bitline voltage is transferred through Column Mux pass transistor to the input of the sense amplifier to determine what data is stored in the bitcell. Since it takes time for the bitline to develop sufficient voltage difference compared to the precharged value. A small voltage difference at the sense amplifier input is susceptible to noise and sense amplifier input voltage offset. Higher input voltage delta 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 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 1-3.

Table 1-3 Timing Mode Options

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

The **`timing_mode`** GLB parameter enables the selection of process-sigma characterization and read 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[2:0] settings are user adjustable. The RM interface to the user is encoded so that all compilers have the same default RM[2:0] values. The user can select from the settings shown in Table 1-4 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[2:0] 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[2:0] value set during compile, and the external, user provided RM[2:0] 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. The RM[3] pin is used for test purposes. 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 1-4 Read Margin Control Pins

Signal	Description
RME	Read Margin Enable pin. Selects between internal and external RM[2:0]* settings. RME=0: Internal settings; RME=1: External settings
RM[3:0]	RM[2:0] are used to change Read Margin settings. RM[3] is for Synopsys' test purposes.

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).

1.3.5 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/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.

1.3.6 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.

This feature allows user to apply back bias to boost the performance by lowering the Vt of the devices. It is controlled by the **`backbias_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 th transistor.

The figure below illustrates the features:

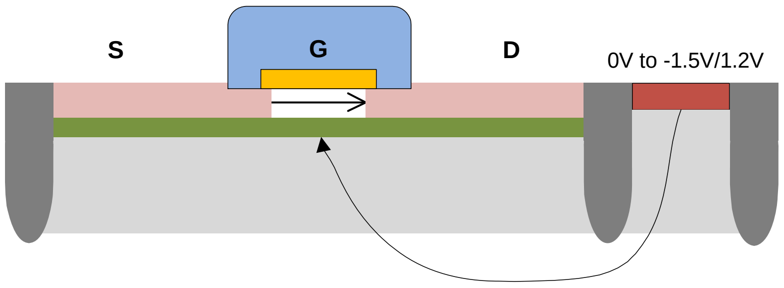


Table 1-5 VNW_N/VPW_P Bias Range for different Voltage Domains

Domain	Process	Voltage (V)	Temperature (C)	VNW_N (V)	VPW_P (V)	Extraction Corner
Deep Underdrive (0.5)	SSG	0.45	-40	1.20	-1.50	FuncCmax
	SSG	0.45	125	0.60	-0.85	FuncCmax
Underdrive (0.65)	SSG	0.59	-40	0.85	-1.50	FuncCmax
	SSG	0.59	125	0.60	-0.80	FuncCmax
Normal (0.8)	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.

1.3.7 Performance Boosters

There are several performance boosting features available within the 22nm FD SOI SiWare compilers. Table 1-6 provides the parameters used to enable the options, as well as the performance enhancement obtained with each option.

Table 1-6 Performance Booster Options

Feature	GLB Parameter	Description
Center Decode	`center_decode`	The Center Decode option allows for a trade off between area and performance. When `center_decode` is set to TRUE, it improves instance performance at the cost of area. If `center_decode=TRUE`, then there are two repair elements - one set of 4 columns on each side of the center decoder. If `center_decode=FALSE`, then the instance is generated with side decode and has one repair element.
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.

1.3.8 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 1-7. The list of devices used for different **`periphery_Vt`** options is presented in Table 1-8.

Table 1-7 Periphery Vt Options

		periphery_Vt Options	
Compiler	**Process**	**LOW**	**ULTRALOW**
High-Density Single Port ROM	FD SOI	`periphery_Vt=LOW`	`periphery_Vt=ULTRALOW` (DEFAULT)

Table 1-8 Periphery devices for different Vt Options

periphery_Vt	Periphery Devices
LOW	Majority LVT and few SLVT devices for speed critical path
ULTRALOW	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 Via 12 ROM 1M 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 gf22nsd41p10s1dvl01ms.

Table 2-1 Compiler Naming Convention

Name Segment	Value in gf22nsd41p10s1dvl01ms	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 *n*p, where *n* is the number of ports
Read/Write	10	Its format is *nm*, where *n* = number of read ports, *m* = number of write ports, therefore 10 = one read port and no (zero) write ports
Product family	s1	SiWare Automotive Grade 1
Product subfamily	dvl	High-Density Via 12 ROM Via 12 is the layer between Metal1 and Metal2 on which the programming occurs.
Size	01m	1M bits
Protocol	s	Synchronous clock

2.2 Compiler Features and Benefits

The *SiWare Automotive Grade 1 Single Port High Density Via 12 ROM 1M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process* provides:

- All memory compilers have leakage control techniques to reduce standby current.
- Two power management modes: LIGHT SLEEP and SHUT DOWN to reduce static power.
- Dynamic Voltage and Frequency Scaling support.
- 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 self-timed circuits and adjusting read margins.
- Provides programming support and intelligent ROM programming to average power.
- Anti-signature (ASE) is calculated by ROMTool.
- ASE is stored outside the address range of the memory to provide a continuous address space when using multiple memories.
- Three test mode options to support third party memBIST, Synopsys' STAR Memory System and Redundancy.
- Patented array architecture: Hybrid of NOR and NAND ROM to get the best area and performance.
- Unique leakage reduction scheme used in the bitcell array to limit quiescent current consumption to be close to zero when the ROM is not enabled.
- 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.
- Four aspect ratios for maximum area and performance optimization.
- Maximum instance size of 1280K bits.
- Zero Quiescent Current consumption when ROM is not in read.
- Flexibility in specifying the logical size of the ROM, including both word size and number of address locations, and column mux option.
- BIST interface: Integrated test logic to facilitate at-speed memory BIST & high-speed scan testing.

2.3 Specifications

This section describes the specifications of the *SiWare Automotive Grade 1 Single Port High Density Via 12 ROM 1M Sync Compiler For GLOBALFOUNDRIES 22nm FD SOI P-Optional Vt/Cell Std Vt Process*.

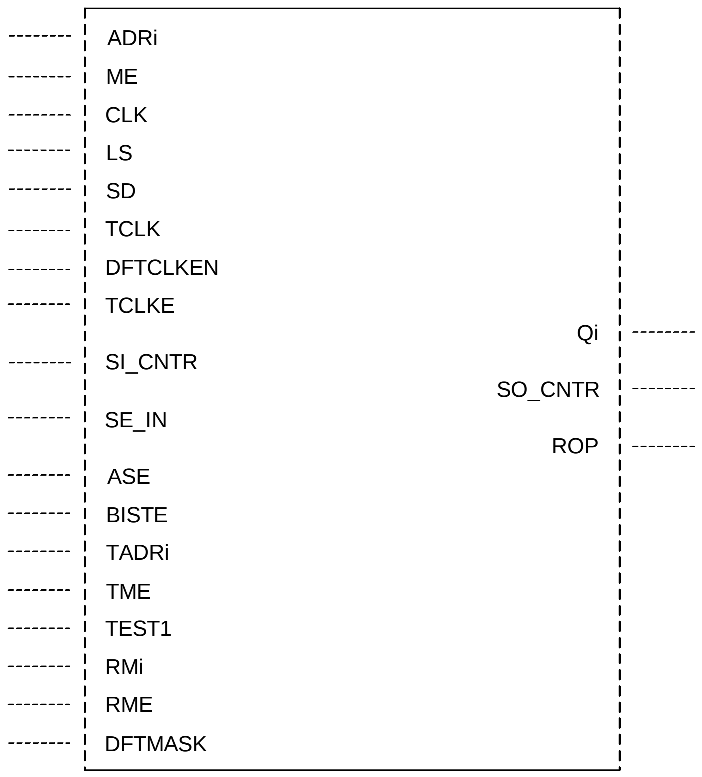
- "Memory Symbol" on page 22
- "Pin Description" on page 23
- "Block Diagram" on page 25
- "Functional Options" on page 27
- "Functional Descriptions" on page 27
- "Anti-Signature Mode" on page 28
- "ROMTool Features" on page 28
- "Compiler Range Information" on page 30
- "Metal Layer Usage" on page 31
- "Signal Pin and Power Routing" on page 32
- "IP Tagging" on page 35
- "Physical Grids" on page 35
- "Logic Truth Tables" on page 35
- "Hazard Information" on page 38
- "Waveform Diagrams" on page 39
- "Standard Operating Characterization Conditions" on page 45
- "Using Licensed Custom PVTs" on page 45

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 Via 12 ROM 1M 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. See Figure 2-3 on page 26 for illustrations of the different pinout options.

Figure 2-1 Memory Symbol



2.3.2 Pin Description

Table 2-2 and Table 2-3 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 2-2 Input Pin Description

Signal	Enabled	Description
ADR [13:0]	always	Address Input. This Address input port is used to address the location to be read during the Read cycle.
ME	always	Memory enable input. When the memory enable input is Logic High, the memory is enabled and read operations can be performed. When memory Enable input is Logic Low, the memory is deactivated.
CLK	always	Clock input. This is the external clock for the memory.
LS	always	Light Sleep Input. When this pin is active then memory goes into low leakage mode, there is no change in the output state.
SD	`pg_enable=TRUE`	Shut Down Input. This pin shuts down power to periphery and memory core.
TCLK	`bist_enable=TRUE`	BIST Clock Input. This is the external clock for the memory. This is used only when BIST is enabled.
DFTCLKEN	`bist_enable=TRUE`	This Input enable scan chain clocks for ATPG scan. Data flows synchronously with clock.
TCLKE	`bist_enable=TRUE`	Test external clock enable signal.
SI_CNTR	`bist_enable=TRUE`	Scan Chain Input for "Control Signals".
SE_IN	`bist_enable=TRUE`	Scan Enable input for "Control scan chain". Once made active, internal clock generation stops. READ operation is disabled.
ASE	`bist_enable=TRUE`	Anti-Signature mode control Input. When the ROM is being programmed by the ROMTool, it has a provision for storing the anti-signature. The anti-signature is calculated internally by the ROMTool software based on the contents being store in the ROM and it can be read out by asserting the ASE pin. This will facilitate testing of the ROM by Synopsys SMS. When ASE=1, the anti-signature will be available on the output pins independent of address being applied.
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.
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. This pin is used 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 operations can be performed. When the BIST Memory Enable is Logic Low, the memory is deactivated. This is used only when BIST is enabled.

Table 2-2 Input Pin Description

Signal	Enabled	Description
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. When this pin is high it enables the test mode for externally controlling read self timed clocks. This pin should be low for normal operation of the memory.
RM [3:0]	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 Margin Input. This input is used for setting the Read margin. It programs the sense amp differential setting and allows the trade-off between speed and robustness. RM[2:0] = 3'b000 is the slowest possible mode of operation for the memory. This setting is required for VDDMIN operation. RM[2:0] values control access time & cycle time of the memory. Refer to the timing table for more details. RM[3] pin is reserved for debug mode to disable the keeper. User should tie this pin to "logic 0" for functional 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 Margin Enable Input. This selects between the default Read margin setting (RME=0), and the external pin Read margin setting (RME=1).
DFTMASK	`bist_enable=TRUE`	This pin is used to enable the ATPG mode. Once made active, internal clock generation stops (READ operation is disabled).

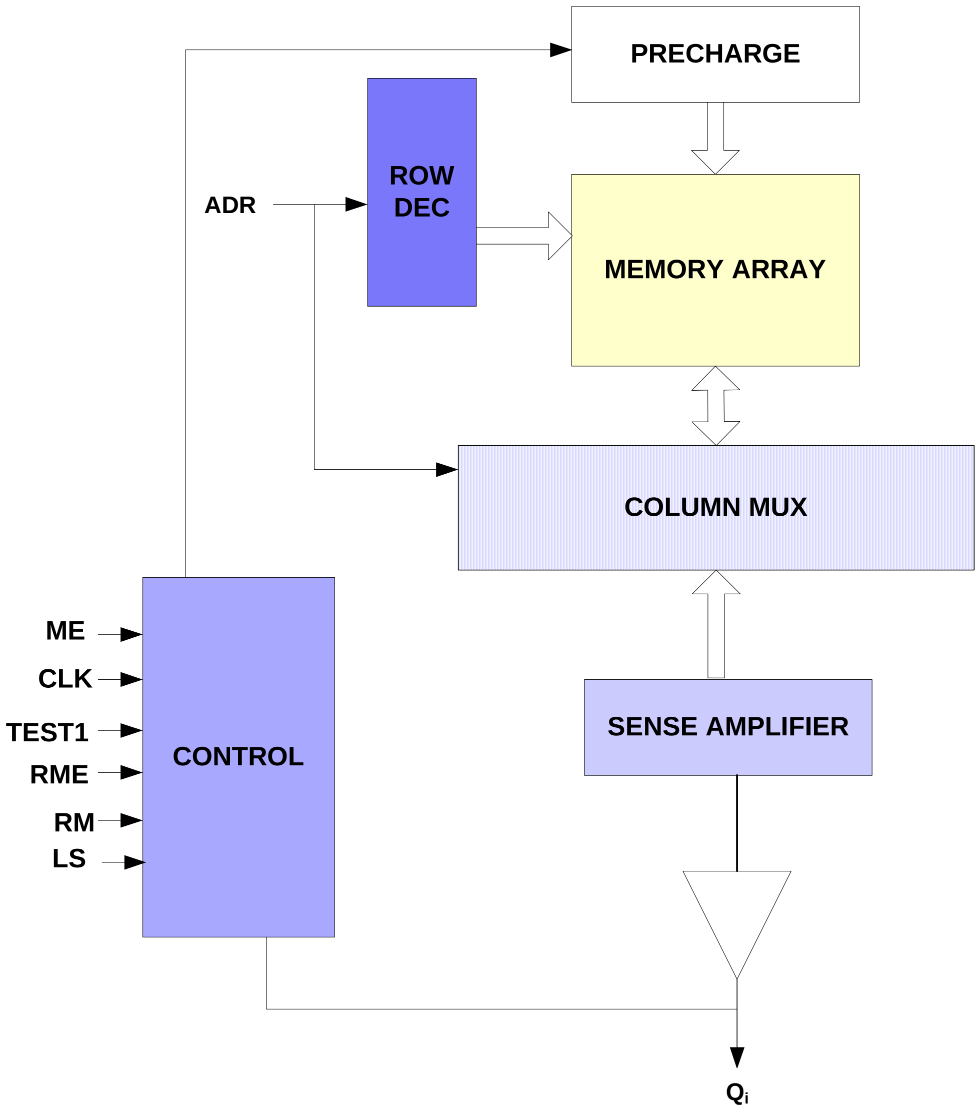
Table 2-3 Output Pin Description

Signal	Enabled	Description
Q [15:0]	always	Data Output bus. It outputs the contents of the memory location addressed by the Address Input signals.
ROP	`pg_enable=TRUE`	ROP is Ready for Operation pin. This pin should be high for any valid memory operation.
SO_CNTR	`bist_enable=TRUE`	Scan Chain Output for "Control Signals".

2.3.3 Block Diagram

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

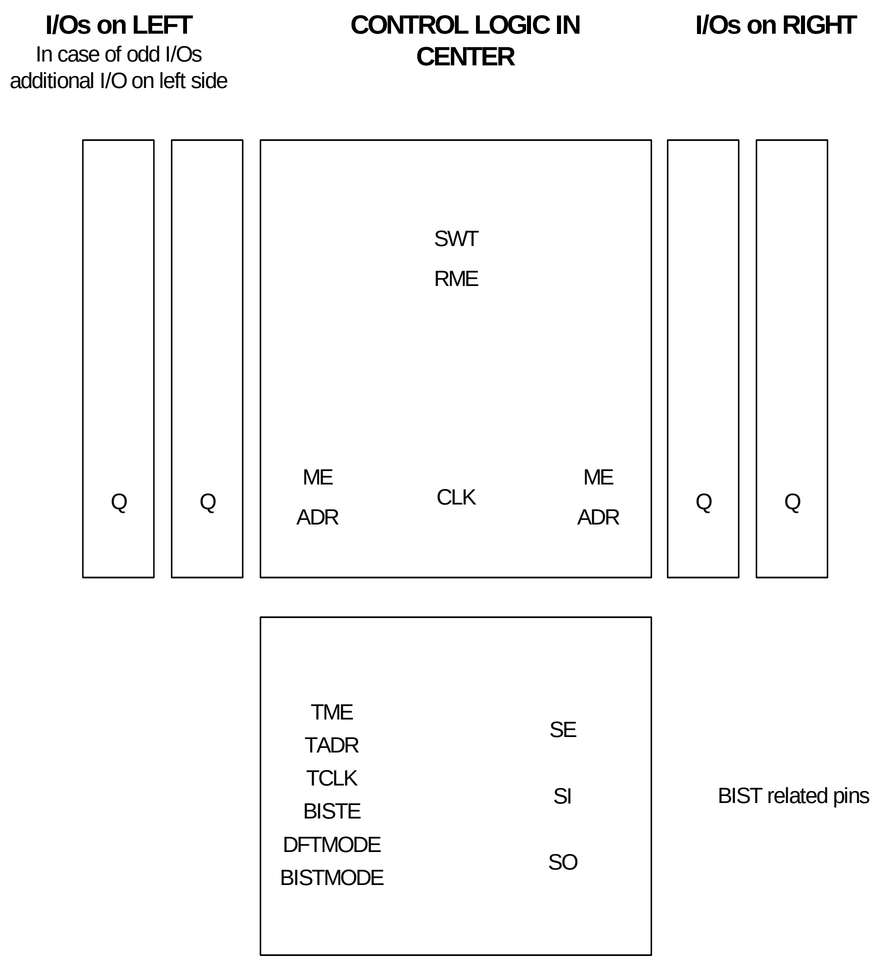
Figure 2-2 Functional Pin Block Diagram



2.3.3.1 Pin Position

Figure 2-3 shows the positions of various pins in the SiWare-LT and SiWare-IT compilers.

Figure 2-3 Pin Position (SiWare-LT/IT)



2.3.4 Functional Options

The *SiWare Automotive Grade 1 Single Port High Density Via 12 ROM 1M 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 2-4. The flags **`bist_enable`** and **`pg_enable`** determine the selection of the configuration that is to be used.

Table 2-4 SiWare High-Density Memory Options

Configuration	Parameters Required

SiWare-LT
(default configuration)
	`bist_enable=FALSE`
	`pg_enable=FALSE`

SiWare-IT
(LT + Scan Chain + BIST Muxes)
	`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 inputs are optional. These input pins will be generated only if BIST is enabled by setting **`bist_enable=TRUE`** in the custom global file.
- ME can be configured as active high or active low input.

2.3.5 Functional Descriptions

2.3.5.1 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.

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. They are used for DVFS to get Fmax by changing a voltage in different RM. The other pin, RM[3], is used for test purposes. There is also a Read Margin Enable (RME) pin that selects between the internal default RM[2:0] value set during compilation, and the external, user provided RM[2:0] 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 2-5 Read Margin Control Pins

Signal	< >_custom.glb setting	Description
RME	None, always enabled	Read 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	RM[2:0] are used to change Read Margin settings. RM[3] is for Synopsys' test purposes.

Note * RM3 is not controlled by RME.

2.3.6 Anti-Signature Mode

When the ROM is programmed by the ROMTool, it has a provision for storing the anti-signature. The anti-signature is calculated internally by the ROMTool software based on the contents being stored in the ROM and it can be read out by asserting the ASE pin.

This will help facilitate testing of the ROM by Synopsys' SMS. When ASE=1, the anti-signature will be available on the output pins independent of address being applied. Please note that ASE is a synchronous input and therefore proper setup/hold times have to be followed when using this pin.

2.3.7 ROMTool Features

2.3.7.1 Programming

The memory instances can be programmed while generating the macro or on a generated memory instance. The ROMTool software requires the programming code to be always specified in HEX format. This format has to be compatible with Verilog "$readmemh" statement. In case the code is available in a different format, Synopsys, Inc. supplies translators to convert it to the HEX format. Currently translators are available for Intel HEX format, Motorola S-records format and binary format.

To program a memory instance at the time of generation, the following flags should be set in the instance configuration file:

```
rom_programming = true
rom_code_file = <The code file in HEX format>
```

To program/re-program a generated memory instance, create a control file with the following flags:

```
File rom_prog.cnt
// Configuration File for Standalone Romtool
// Inputs
//Input Database
db_dump_file = <memory_name>.db
//Rom Code in Hexadecimal Number System
rom_code_file = <The code file in HEX format>

//Input GDS, Spice and Verilog
rom_input_gds_file = <memory_name>.gds
```

```
spice_file = <memory_name>.cir
verilog_file = <memory_name>_func.v
// Outputs
//Output directory
rom_out_dir = <output directory>

//Output GDS,Spice and Verilog
rom_gds_out_file = <output gds file>.gds
rom_spice_file = <output spice circuit file>.cir
rom_verilog_file = <output verilog structural netlist>.v
```

This file is used as an input to ROMTool to program/re-program a generated memory instance. This is done by the command:

```
romtool <romtool control file>
```

2.3.7.2 Power Saving Feature

The power saving feature has been enhanced in ROMTool software to control the programming for individual bit lines. This feature controls the number of 1s and 0s that have been programmed in the array. Since reading a programmed 1 consumes less power than reading a programmed 0, while programming, the software will count the number of 0's and 1's on every IO. If (number of 0's) > (number of 1's) then all the columns associated with the IO will be programmed with opposite data and an extra inversion will be added to the output circuitry.

2.3.7.3 Physical Bitmap

ROMTool can generate the physical bitmap of the memory. This can be done using the flag

```
physical_map_file = true
```

The output file <memory_name>.physical_map contains the HEX address and bitnum, decimal location in microns and programmed data. If the programmed data is inverted due to the power saving feature, an asterisk "*" appears along with the programmed data.

Table 2-6 shows an example of a physical map file from ROMTool.

Table 2-6 Physical Bitmap

Address	Bitnum	Cell X	Cell Y	Data
0	0	85.040	65.100	1*
0	1	86.140	65.100	1*
0	2	104.180	65.100	1
0	3	105.280	65.100	1*

2.3.7.4 Anti-Signature

ROMTool has a feature for testing ROMs with the Synopsys' BIST processor.

While programming the ROM, ROMTool also calculates the anti-signature (which depends on the code being programmed in the ROM). This anti-signature is stored in the memory and is available on the output bus by asserting ASE pin. The anti-signature is stored outside the address range of the memory to provide a continuous address space when using multiple memories to create a bigger one.

2.3.8 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	center_decode=TRUE			center_decode=FALSE		
								NB Min	NB Max	Maximum Number of Bits	NB Min	NB Max	Maximum Number of Bits
8	6	11	32	1	64	2048	2	8	160	327680	4	80	163840
	7	12	64	2	128	4096	2	8	160	655360	4	80	327680
	8	13	128	4	256	8192	2	8	160	1310720	4	80	655360
	9	13	256	8	512	8192	2	8	160	1310720	4	80	655360
16	7	12	64	1	128	4096	2	4	80	327680	4	40	163840
	8	13	128	2	256	8192	2	4	80	655360	4	40	327680
	9	14	256	4	512	16384	2	4	80	1310720	4	40	655360
	10	14	512	8	1024	16384	2	4	80	1310720	4	40	655360
32	8	13	128	1	256	8192	2	4	40	327680	4	20	163840
	9	14	256	2	512	16384	2	4	40	655360	4	20	327680
	10	15	512	4	1024	32768	2	4	40	1310720	4	20	655360
	11	15	1024	8	2048	32768	2	4	40	1310720	4	20	655360
64	9	14	256	1	512	16384	2	4	20	327680	4	10	163840
	10	15	512	2	1024	32768	2	4	20	655360	4	10	327680
	11	16	1024	4	2048	65536	2	4	20	1310720	4	10	655360
	12	16	2048	8	4096	65536	2	4	20	1310720	4	10	655360

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.

2.3.9 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 32.

Figure 2-4 Metal Layer Usage

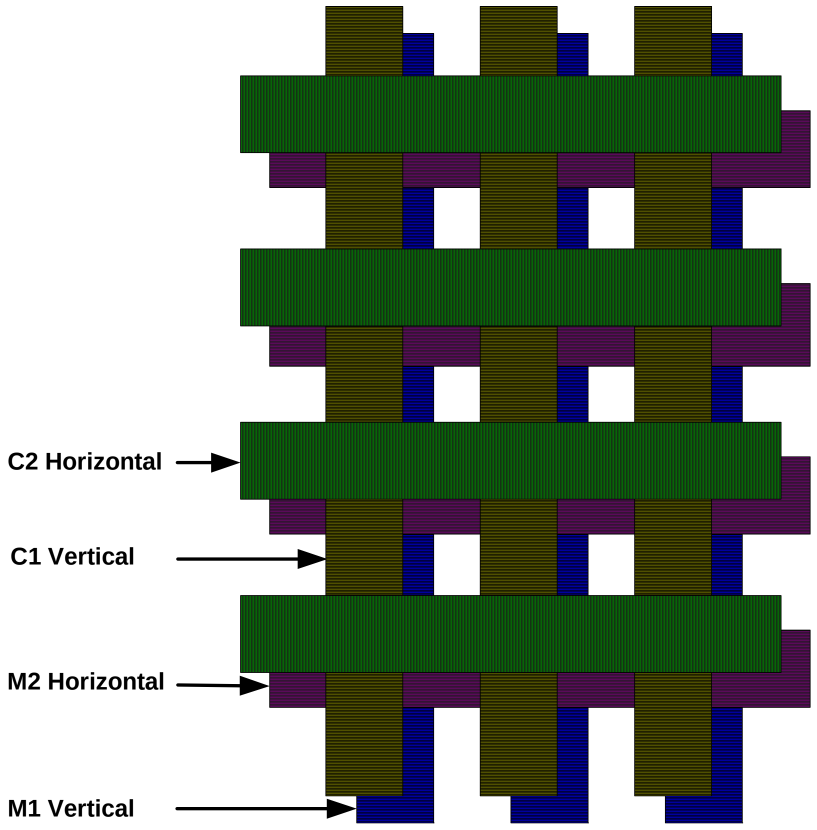


Table 2-8 Metal Layer Routing

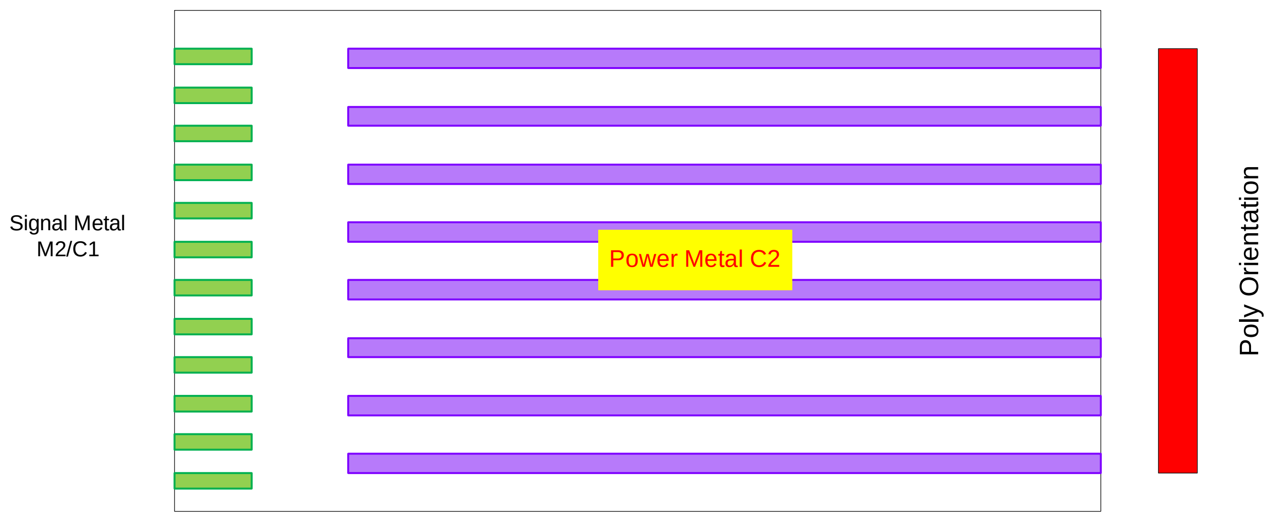
Metal Layer	Routing Instructions
Metal layer availability	M1, M2, C,1and C2 are completely blocked.
C2 routing porosity	Not supported
Routing porosity for C3/C4 and above (including the power mesh)	100%, freely

2.3.10 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 Figure 2-5.

Power pins in C2 for VDD/VSS/VNW_N/VPW_P works with both standard cells schemes.

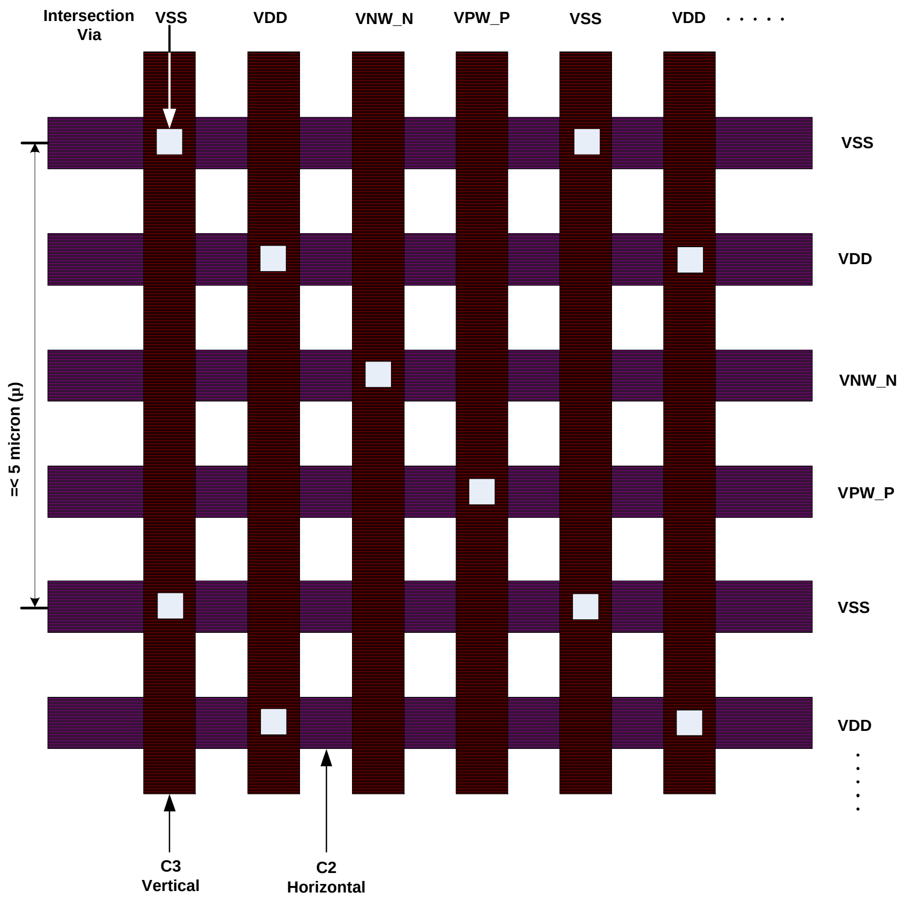
Figure 2-5 Signal and Power Metal Orientation



2.3.10.1 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 5µ for all PG supplies as shown in the Figure 2-6. For example VDD, VSS, 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.

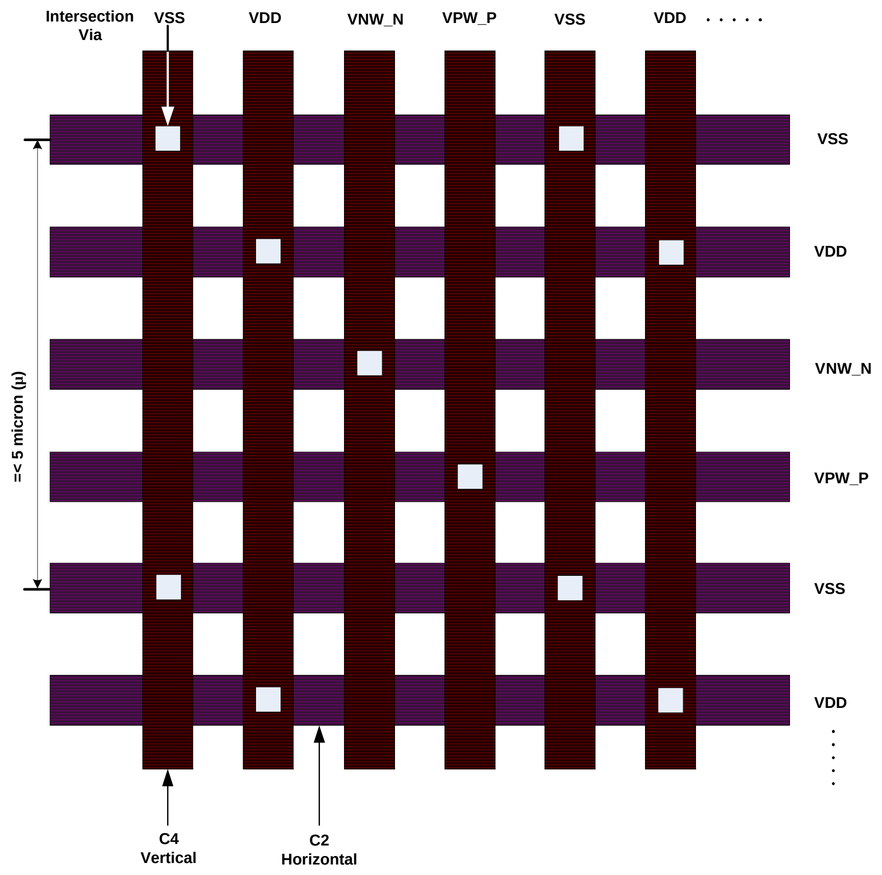
Figure 2-6 Power Mesh Connections



2.3.10.2 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 5µ for all PG supplies as shown in the Figure 2-7. For example VDD, VSS, 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



2.3.11 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_xltt.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.

2.3.12 Physical Grids

The layout grid is 0.001 μm.

2.3.13 Logic Truth Tables

2.3.13.1 Memory Function Truth Table

Note 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	BISTE	ASE	SE_IN	DFTMASK	DFTCLKEN	Q
Idle	L	X	L	L	X	L	L/H	Q-1
Disabled	H	L	L	L	L	L	L/H	Q-1
Read	H	H	L	L	L	L	L/H	Q
No Operation	H	X	X	X	L	H	L/H	Q-1

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

2.3.13.2 ASE Mode Truth Table in Functional Mode

Table 2-10 shows the ASE mode truth table in functional mode.

Table 2-10 ASE Mode Truth Table in Functional Mode

Function	CLK	ME	SD	BISTE	ASE	DFTCLKEN	Q
ASE	L->H	H	L	L	H	L/H	Anti-signature

2.3.13.3 ASE Mode Truth Table in BIST Mode

Table 2-11 shows the ASE mode truth table in BIST mode.

Table 2-11 ASE Mode Truth Table in BIST Mode

Function	TCLK	TME	SD	BISTE	ASE	DFTCLKEN	Q
ASE	L->H	H	L	H	H	L/H	Anti-signature

2.3.13.4 BIST Mode Truth Table

Table 2-12 shows the truth table for BIST mode.

Table 2-12 BIST Mode Truth Table

Function	BISTE	Active Pins		
Normal Mode	L	ME	ADR	CLK
BIST Mode	H	TME	TADR	TCLK

Note TCLKE is used as select pin for CLK/TCLK.

2.3.13.5 TEST1 Mode Truth Table

The TEST1 input is not latched by clock. It needs to remain asserted from one setup time before the clock rising edge to one hold time after the clock falling edge. Table 2-13 shows the truth table for TEST1 mode.

Table 2-13 TEST1 Mode Truth Table

Function	TEST1
TEST1 Mode	H

2.3.13.6 Power Mode Truth Table

Note

* RM3 should be set to "0" in functional mode.
** Applicable only when `pg_enable=TRUE`.
ASG = Anti-signature
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_CNTR

Table 2-14 shows the truth table for Power mode.

Table 2-14 Power Mode Truth Table

Mode	ME	ASE	LS	SD**	RM3	ROP	Q	SO	Comments
Normal	0/1	0	0	0	0*	ROP-1	Q-1	SO-1	Normal mode
Light Sleep	0/1	X	1	0	0/1	ROP-1	Q-1	SO-1	Xdec power gate
Anti-Signature	H	0->1	0	0	0/1	ROP-1	ASG	SO-1	Anti-signature on output
	X	1->0	0	0	0/1	ROP-1	Q-1	SO-1	
Shutdown	X	X	X	0->1	X	0	0	0	Periphery shutdown, array shutdown
	X	X	X	1->0	X	1	0	X	

2.3.13.7 RM3 Pin Truth Table

Table 2-15 shows the truth table for RM3 pin.

Table 2-15 RM3 Pin Truth Table

Mode	RM3
Functional	0*
Light Sleep (LS)	0/1
Shutdown (SD)	X

Note

* RM3 should be set to zero in Functional mode.

2.3.13.8 Minimum Voltage Truth Table

Table 2-16 shows the truth table for minimum voltage allowed for different RM modes.

Table 2-16 Minimum Voltage for different RM Modes Truth Table

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

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

2.3.14 Hazard Information

Note NC - No Change
ME - Pin ME
ADR - Pin ADR
Q - Pin Q

2.3.14.1 Hazard Information for Read Cycle

Table 2-17 shows the Read cycle hazard information for the compiler.

Table 2-17 Hazard Information for Read Cycle

Mode	ME	ADR	Memory	Q
Read	High	-	NC	Valid
Read	High	Violation	NC	X
Read	Violation	-	NC	X
Read	Low	-	NC	Valid

2.3.14.2 Hazard Information for Clock Violations

Table 2-18 shows the clock violations hazard information for the compiler.

Table 2-18 Hazard Information for Clock Violations

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

2.3.15 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

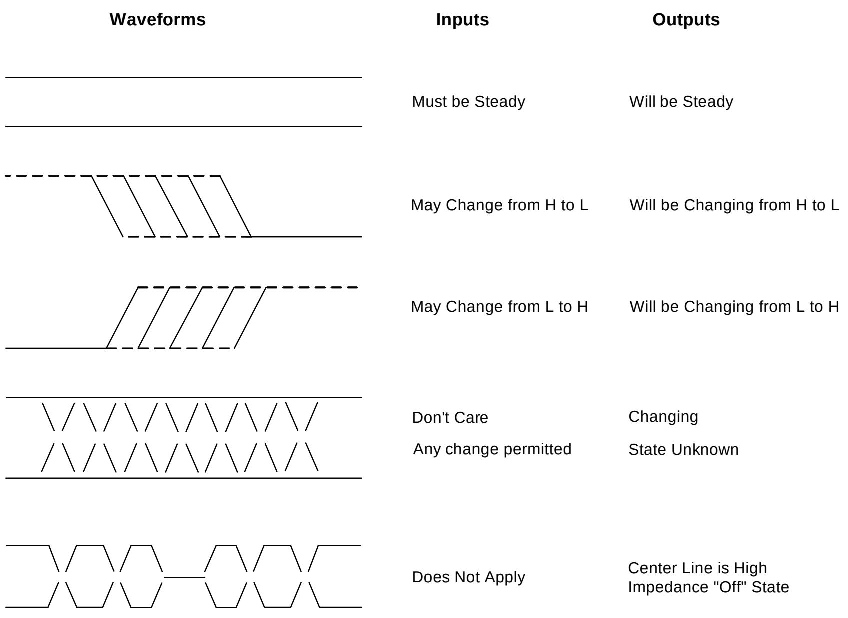


Figure 2-9 Read Cycle Timing

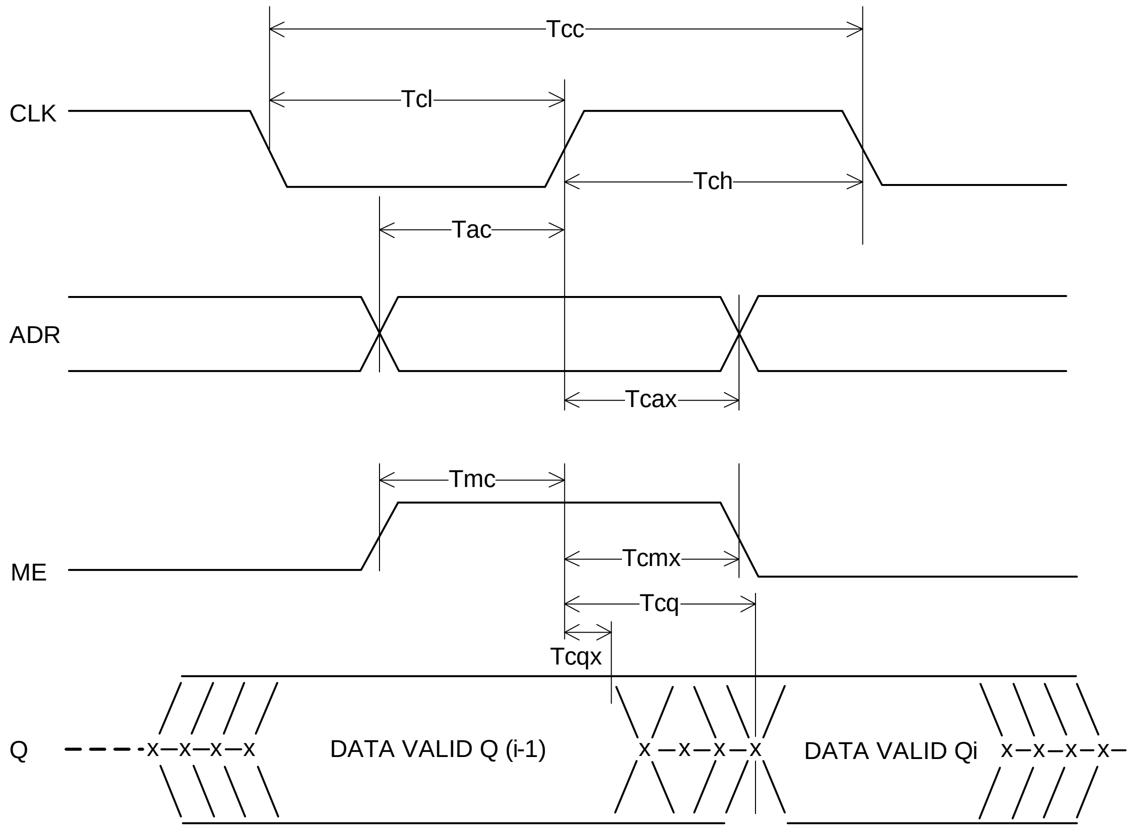


Figure 2-10 Anti-Signature Cycle Timing

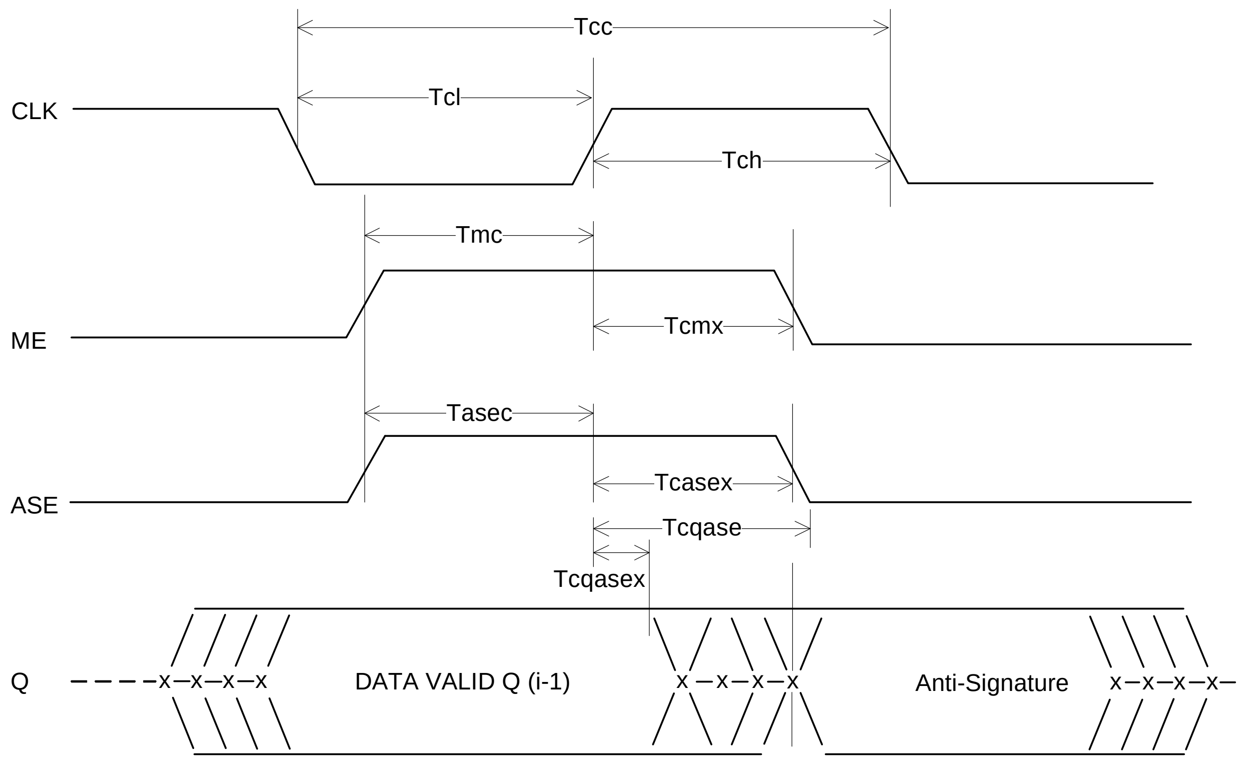
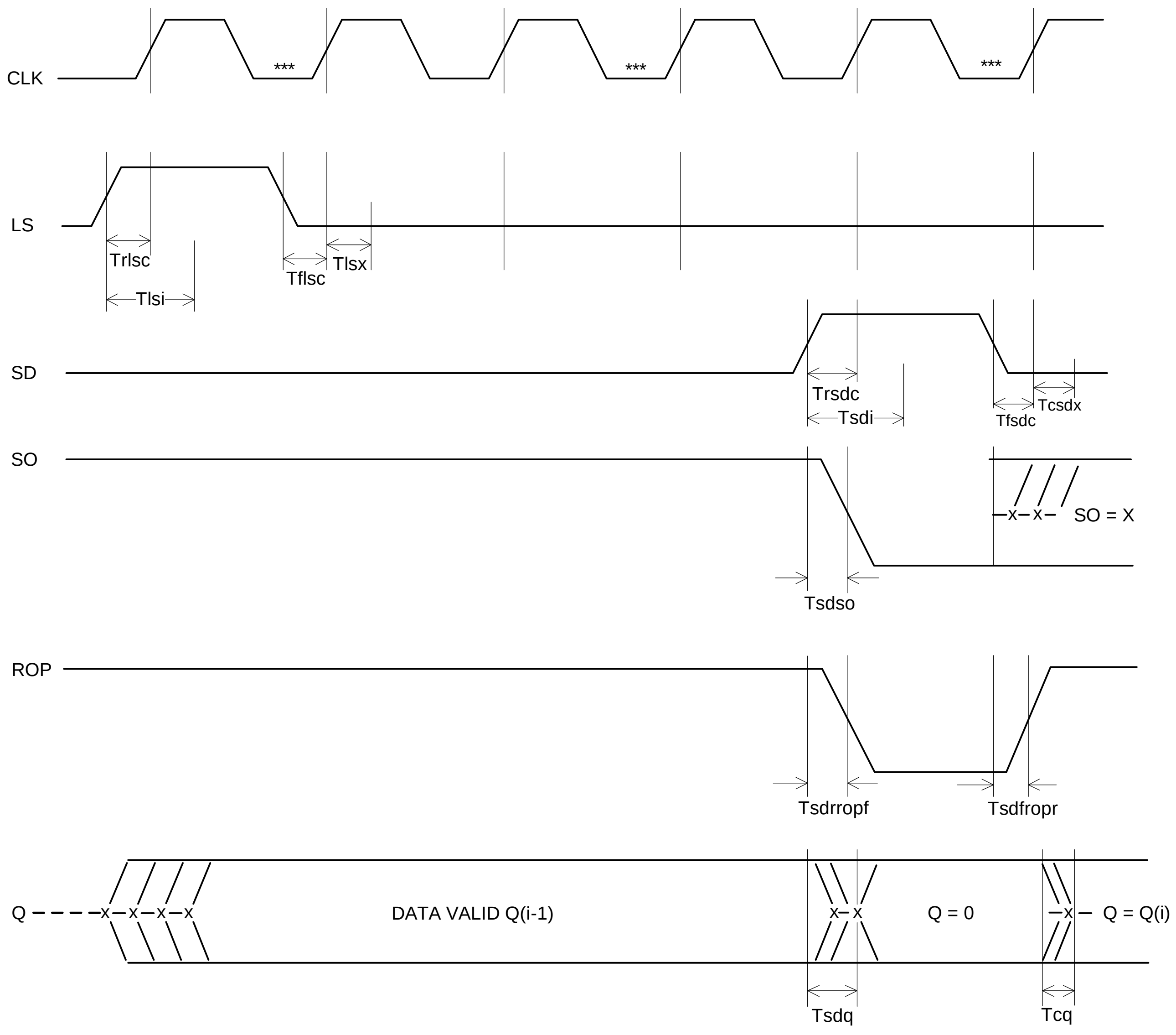


Figure 2-11 Power Down Mode Timing



Note An alternate solution to avoid memory access during wake-up is to set the ME pin to inactive state for at least "Tflsc/Tfsdc" time interval after the "LS/SD" falling edge.

Figure 2-12 Test Mode Timing

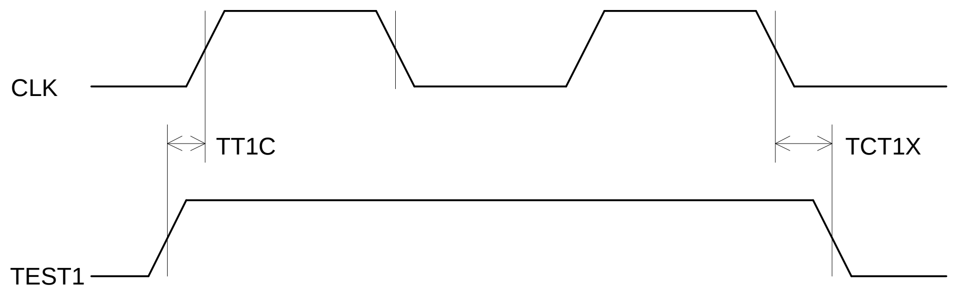
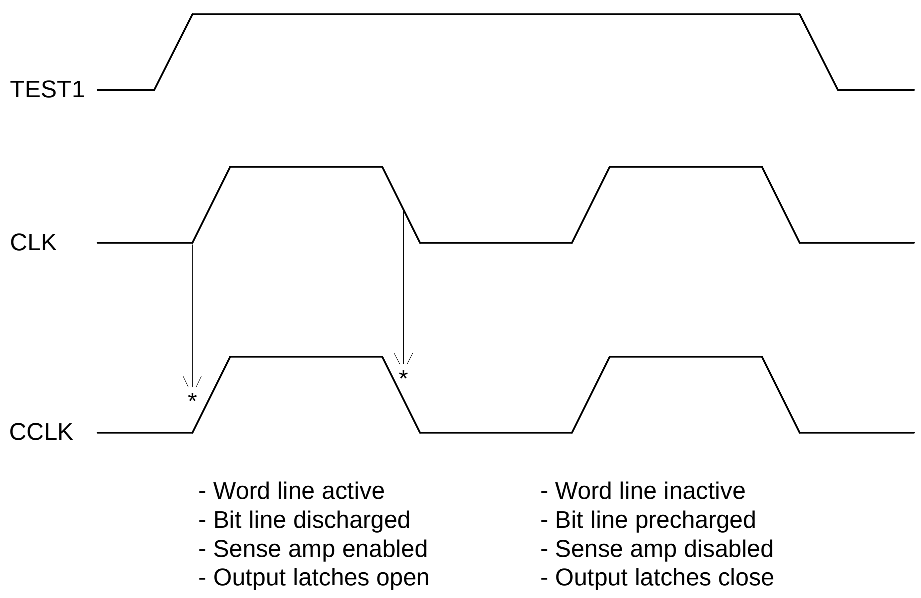
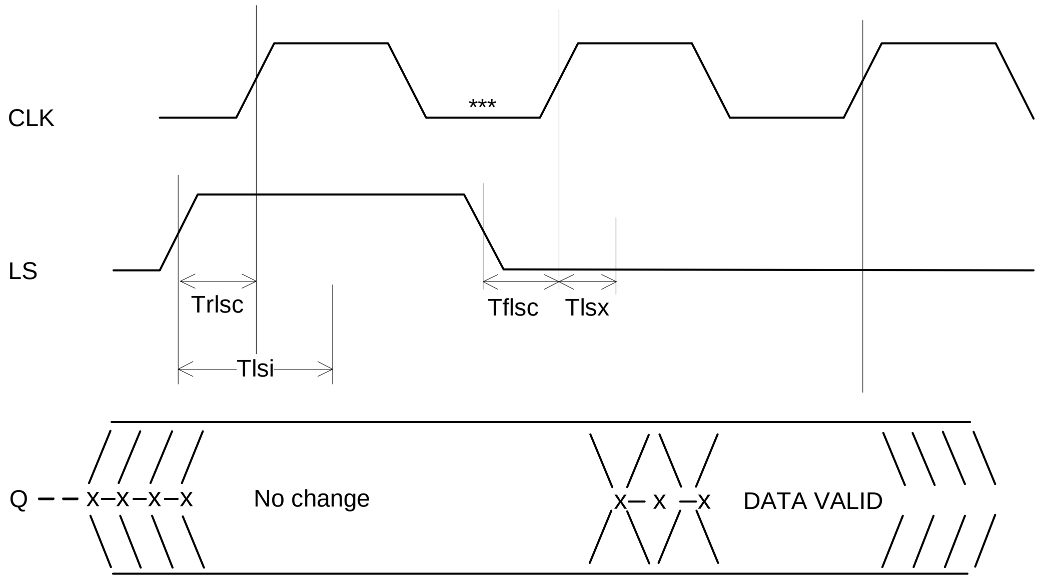


Figure 2-13 Test1 mode: Bypassing Read Self-timed Circuits



Note If the TEST1 signal is active, internal SROM clocks will be controlled by the external clock instead of self timed circuits. In TEST1 mode, Tch requirement is that Tch must be less than 4*Tcq. Tcl requirement is that Tcl >= Tcc.

Figure 2-14 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 "Tflsc" time interval after the "LS" falling edge.

2.3.16 Standard Operating Characterization Conditions

Table 2-19 shows the typical, best, and worst operating characterization conditions supported by this compiler.

Table 2-19 Standard Operating Conditions

PVT Corner	Process	VDD (V)	Temperature (C)	VNW_N (V)	VPW_P (V)	RC Extraction
tt0p8v25c_0p8v_0v_0v	TT	0.8	25	0	0	Nominal
ssg_fcmax0p72v125c_0p72v_0v_0v	SSG_FCMAX	0.72	125	0	0	FuncCmax
ssg_fcmax0p72vn40c_0p72v_0v_0v	SSG_FCMAX	0.72	-40	0	0	FuncCmax
ssg_fcmax0p72vn40c_0p72v_0p7v_n1p5v	SSG_FCMAX	0.72	-40	0.7	-1.5	FuncCmax
ssg_fcmax0p72v125c_0p72v_0p6v_n1p0v	SSG_FCMAX	0.72	125	0.6	-1.0	FuncCmax
tt0p8v85c_0p8v_0v_0v	TT	0.8	85	0	0	Nominal
tt0p65v25c_0p65v_0v_0v	TT	0.65	25	0	0	Nominal
tt0p65v85c_0p65v_0v_0v	TT	0.65	85	0	0	Nominal

Note
VDD: Periphery Voltage
VNW_N: Nwell back bias Voltage for Forward Body Bias (FBB) Schemes.
VPW_P: Pwell back bias Voltage for FBB Schemes.

2.3.17 Using Licensed Custom PVTs

Customers may license additional characterized PVTs, beyond the standard PVTs listed in "Standard Operating Characterization Conditions" on page 45, 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 = {2 3 4 9 10}`**.

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
User-editable files	
<Compiler_name>_custom.glb	Customer-specific overrides to flags in the compiler global parameters (glb) file.
design2<foundry>_xltt.csv	GDSII layer mapping details- the design layer numbers are mapped to foundry layers.
Read-only files	
release.txt	Information about the changes made to the compiler in each patch release of the compiler such as patch notes and release notes.
Information files	
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.

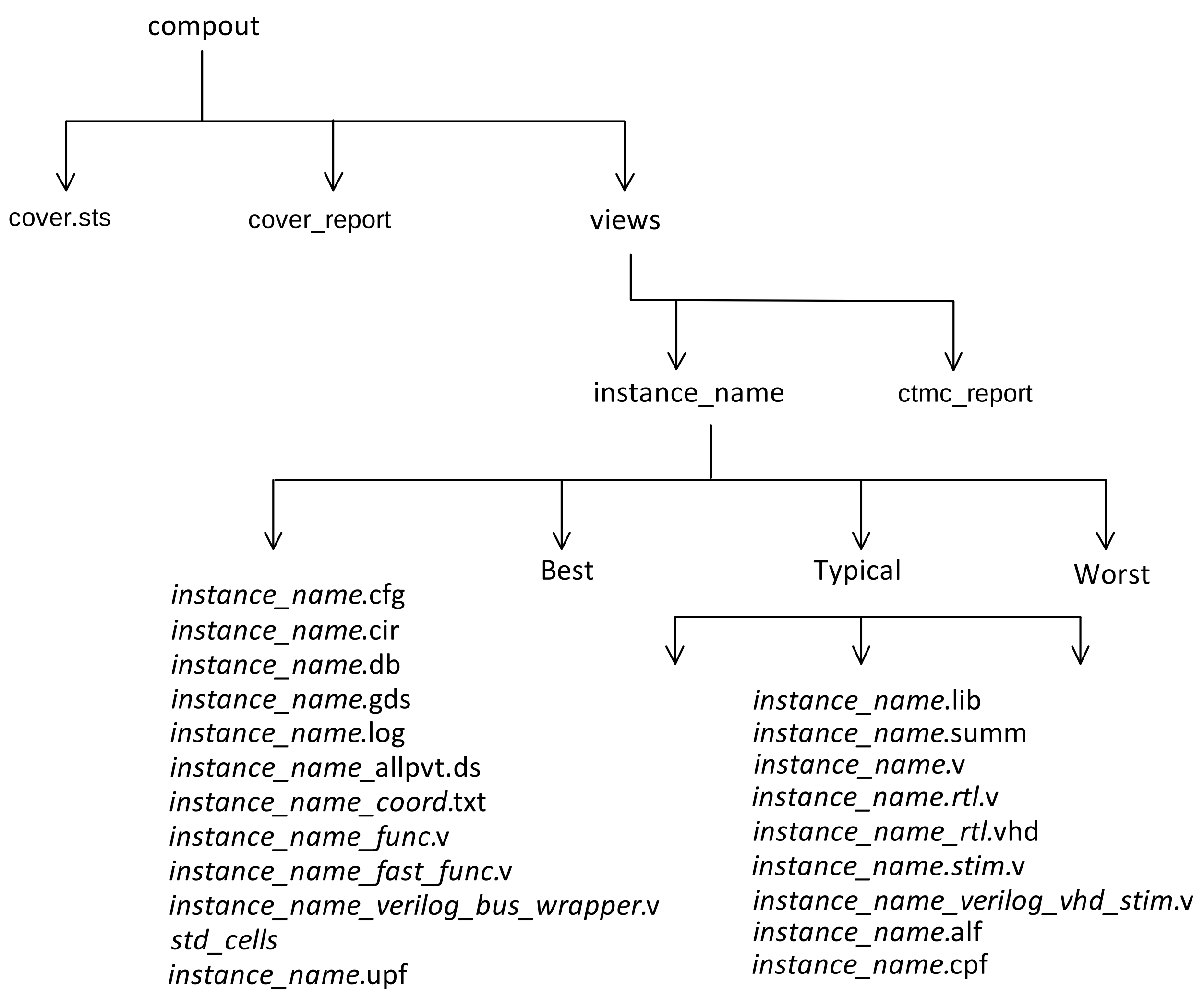
Table 3-1 Files in the <compiler> Library

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 <Compiler_name>.pf.
<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 *compout/views Subdirectory.*

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.
instance_name/	Instance directory. See compout/views/instance_name Subdirectory.

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.
instance_name.cfg	Instance configuration file.
instance_name.cir	Instance SPICE netlist for LVS.
instance_name.db	Synopsys' internal database that records all options set for this instance.
instance_name.gds	Instance GDSII view
instance_name.log	Instance log file
instance_name.plef	Physical LEF view of the ring structure. Includes instance blockages.
*instance_name*_allpvt.ds	All PVT datasheet that provides pin descriptions, logic truth table, timing characterization, Read and Write cycle timing, and power dissipation.
*instance_name*_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
*instance_name*_func.v	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.
*instance_name*_fast_func.v	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 **`$readmemh`** task can be used to load the memory.
*instance_name*_verilog_bus_wrapper.v	Verilog file used to test the basic functionality of the memory without consideration for BIST or ATPG.
std_cells.v	Verilog model for the dummy standard cells used in Verilog netlist for ATPG.
*instance_name*_hcell.txt	Hcell equivalence file for hierarchical LVS matching with Calibre.
instance_name.clp	Verilog Model used in conformal Low Power tool. This model captures power management information for the memory core.
instance_name.ctl	Core Test Language Model that contains scan chain information for the memory core. This information is used for testing at the system level.
instance_name.upf	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
instance_name.lib	Synopsys timing model.
instance_name.v	Instance Verilog model.
*instance_name*_rtl.vhd	VHDL RTL core model.
*instance_name*_stim.v	Verilog behavioral model testbench.
*instance_name*_verilog_vhd_stim.v	Verilog behavioral model and VHDL behavioral model testbench.
instance_name.cpf	CPF view for the memory instance. 'Common Power Format', Si2 low power format to model the power intent.

4 Using Front-End Views

This chapter describes various Front-End EDA views of the memory instances. It also describes how to set Front-End flags.

- "Using Template Files" on page 53
- "Setting Front-End Flags" on page 59

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 views.

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

- "Simulation Models" on page 54
- "ATPG Models" on page 55
- "MemoryBIST Models" on page 57
- "Timing Library Model" on page 57
- "Power Analysis Model" on page 58

4.1.1 Simulation Models

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

- "Verilog Behavioral Model" on page 54
- "VHDL RTL Model" on page 55
- "Power Verilog Model" on page 55
- "Fast Functional Verilog Model" on page 55
- "Switch-Level Verilog Model" on page 55

4.1.1.1 Verilog Behavioral Model

Use the **`mc_verilog.tpl`** template file to generate the **`<instance_name>.v`** Verilog behavioral model. This model describes the logic in different modes with every possible transition of every pin along with X handling and full timing. Full timing modeling includes all path delays and input pin constraints like setup, hold, recovery, width, and period.

To help perform simulations with parallel testbench from ATPG tools like Synopsys Tetramax and Mentor Fastscan, 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 and Verilog memory model.

To perform Verilog simulations, use Verilog file **`<instance_name>.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 model **`<instance_name>.v`** 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. Verilog RTL model is faster, depending on the Simulator and the instance size, when compared to regular behavioral model 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.

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

4.1.1.2 VHDL RTL Model

Use the **`mc_vhd_rtl.tpl`** template file to generate the VHDL RTL **`<instance_name>_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.

4.1.1.3 Power Verilog Model

Use **`mc_power_verilog.tpl`** template file to generate the Power Verilog model **`<instance_name>.mv`**. The Power Verilog Model, **`<instance_name>.mv`**, is similar to Verilog behavioral model but 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 entire memory is corrupt (not in case of ROM) and outputs, and no memory operation is allowed.
- If the power pins are valid (VDD = 1 and VSS = 0) then the power model will behave similar to Verilog Behavioral Model, that is, the memory will operate normally.

4.1.1.4 Fast Functional Verilog Model

Verilog model with limited functionality to speed up the simulation time. This model supports limited functionality only and does not contain any timing information.

4.1.1.5 Switch-Level Verilog Model

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_name>_func.v`**. The difference is that you can use the Verilog netlist for logic simulation and not for LVS.

4.1.2 ATPG Models

Synopsys, Inc. supports the ATPG tools:

- "Using TetraMAX" on page 56
- "Using FastScan" on page 56

The **`mc_tetramax.tpl`** template file generates Tetramax (**`<instance_name>.max`**) model and mc_fastscan.tpl template file generates the FastScan (**`.fs_lib`**) model.

The **`mc_atpg_netlist.tpl`** template generates Verilog Netlist for ATPG (**`<instance_name>_atpg_netlist.v`**).

Tetramax and Fastscan models contain memory core description and Verilog netlist for ATPG contains the remaining logic surrounding the memory core.

The **`mc_spf.tpl`** template file generates initialization file for Tetramax and **`mc_tpf.tpl`** template file generates test procedure file for Fastscan.

The template **`mc_fastscan.tpl`** 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.

4.1.2.1 Using TetraMAX

TetraMAX ATPG automatically generates high quality manufacturing test vectors. TetraMAX is the only ATPG solution optimized for a wide range of test methodologies that is integrated with the Synopsys DFT compiler test synthesis tool. TetraMAX enables RTL designers to create efficient and compact tests for even the most complex designs.

Key Benefits

- Increases product quality with generated test vectors for high defect detection
- Reduces testing costs through the use of advanced vector compaction techniques
- Increases designer productivity by leveraging integration with Synopsys DFT compiler
- Creates tests for complex and multi-million gate designs

Features

- Extremely high capacity and performance
- Integrated graphical user interface
- Integrated context-sensitive online help
- Comprehensive scan design rule checking
- Utilizes existing Verilog simulation libraries
- Integrated fault simulator for functional vectors

4.1.2.2 Using FastScan

The Mentor Graphics FastScan ATPG tool suite produces manufacturing test vectors for deep submicron IC and ASIC designs.

FastScan provides:

- High capacity full scan and structured partial scan ATPG
- Support for a wide range of fault models: stuck-at and transition
- Extensive simulation based DRC
- Highly effective static and dynamic pattern compression
- A CPA option that supports at-speed path delay test pattern generation
- A MacroTest option that creates scan tests for small embedded cores and memories
- Diagnostics feature that analyzes failing patterns to isolate defects

4.1.3 MemoryBIST Models

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 view. This view is generated using the **`mc_lvlib.tpl`** template file.

4.1.4 Timing Library Model

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 models and verify timing arcs consistency.

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

4.1.4.1 Using Synopsys .lib Models

Use the **`mc_syn.tpl`** template file to generate the Synopsys **`<instance_name>.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.
- 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.

4.1.5 Power Analysis Model

Synopsys, Inc. supports the power analysis model Apache Virtual Model (AVM).

4.1.5.1 Using AVM

Use the **`mc_avmcfg.tpl`** template to generate the AVM view (**`*_avm.cfg`**). The view describes electrical properties of a memory instance and is used by the AVM utility of Redhawk tool to generate current and capacitance profile.

Use the AVM view to perform power analysis with the help of Redhawk tool from Ansys.

4.1.6 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 instances.

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

4.2 Setting Front-End Flags

This section describes how to set or change Front-End flags for your compiler. This can be done in the following ways:

- 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, change the Front-End flags to affect all instances generated in the project.
- When using VMC, 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 in the Integrator 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.

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.

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 61
- "Using GDSII" on page 62
- "Using SPICE Netlists" on page 62
- "Using Switch-Level Verilog Models" on page 63
- "Reading the ROM Physical Map File" on page 63

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 Customer Support 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_xltt.csv** file in the compiler directory or reference another layer map file using the **`compiler_xltt_file`** variable. For example:

```
set cfg(compiler_xltt_file) <path>/file_xltt.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 compiler GDSII library names start with a prefix of the form **`Mnnnxn_`**. 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`** file for the compiler.

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 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.

5.5 Reading the ROM Physical Map File

The physical map file **`<instance>.physical_map`** is created during memory generation when the generate_physical_map parameter is set. This file contains the physical map information related to the ROM instance being generated.

Figure 5-1 shows a section from a physical map file.

Figure 5-1 Physical Map File Example

```
 Address   Bitnum     Cell X     Cell Y  Data

     0        0       85.040     65.100   0
     0        1       86.140     65.100   1
     0        2      104.180     65.100   1
     0        3      105.280     65.100   1
     1        0       86.160     65.100   1
     1        1       87.260     65.100   0
     1        2      105.300     65.100   1*
     1        3      106.400     65.100   0
     2        0       87.280     65.100   0
     2        1       88.380     65.100   1
     2        2      106.420     65.100   0
     2        3      107.520     65.100   0
     3        0       88.400     65.100   0*
     3        1       89.500     65.100   0*
     3        2      107.540     65.100   0
     3        3      108.640     65.100   1
     4        0       89.520     65.100   0
     4        1       90.620     65.100   0*
     4        2      108.660     65.100   0*
     4        3      109.760     65.100   0
     5        0       90.640     65.100   0*
     5        1       91.740     65.100   1
     5        2      109.780     65.100   1*
     5        3      110.880     65.100   1
     6        0       91.760     65.100   1*
     6        1       92.860     65.100   0
     .
     .
     .
```

The columns in the file represent the following data:

- **Address** - 5 digit hex code representing the address of the bit
- **Bitnum** - 8 digit hex code representing the bit number
- **Cellx/Celly** - Floating point numbers that identify the x, y coordinates
- **Data** - Boolean value for data of a particular address.
- The ***** in the last column means, that this bit was inverted for power saving reasons. The array cell will have the "data" value, but the **Q** output will be inverted from this value.

6 Configuring Files

This chapter shows how to configure the following files:

- "Expanded SiWare™ Instance Naming Convention" on page 65
- "Configuring the Custom GLB File" on page 67 - The custom GLB file contains parameters that influence the memory generation tool.
- "Configuring the Extended Layer Translation Table File" on page 68 - 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 68.

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:

```
s1dvlssd4u1p64x4m8b1c0p0l0rm3sd
```

This breaks down into these segments, and further described in Table 6-1.

s1	dvl	s	sd4	u	1p	64	x	4	m8	b1	c0	p0	l0	rm3	sd
<s1>	<dvl>	<s>	<sd4>	<u>	<1p>	<64>	x	<4>	m<8>	b<1>	c<0>	p<0>	<l0>	rm<3>	<sd>
1	2	3	4	5	6	7		8	9	10	11	12	13	14	15

Note The base compiler name contains eight segments (aabbbcccddddeeffgggh) 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` `<u|l>`	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
8	`NB` value	User specified value for Number of Bits
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	c<0|1>	User specified value for center decode (`center_decode`). Valid values are: TRUE (1) and FALSE (0).
12	p<0|1>	User specified value for power gating (`pg_enable`). Valid values are: TRUE (1) and FALSE (0).
13	<l0|r0>	User specified values for `bist_enable`. l0 - `bist_enable=FALSE` r0 - `bist_enable=TRUE`
14	rm<0|1|2|3|4|5>	Read margin timing modes. User specified value for "`timing_mode`" rm0 - `timing_mode=RM0` rm1 - `timing_mode=RM1` rm2 - `timing_mode=RM2` rm3 - `timing_mode=RM3` rm4 - `timing_mode=RM4` rm5 - `timing_mode=RM5`
15	<sd|hd>	User specified value for "`output_drive`" sd - `output_drive=STANDARD` hd - `output_drive=HIGH`

Note Auto-Name Segment 15 '<sd|hd>' will be added only if the compiler supports user configurable `output_drive` option.
Supported output drive strengths are `STANDARD` and `HIGH` (4X of `STANDARD`).

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:

```
s1dvlssd4u1p64x4m8b1c0p0l0rm3sd
s1dvlssd4u1p64x4m8b1c0p0l0rm3sd_1
s1dvlssd4u1p64x4m8b1c0p0l0rm3sd_2
s1dvlssd4u1p64x4m8b1c0p0l0rm3sd_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_xltt_file`** parameter defined in the compiler GLB file specifies the name of the layer mapping file. Use the **`compiler_xltt_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_xltt_file="design2<foundry>_xltt.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

Attention The information presented in this section is based on the preliminary FE LEF compiler release. It will get updated in subsequent releases of the compiler.

This family of memory compilers implements a power mesh configuration. Power mesh is created by interlaced metal4 mesh straps of VDDP and VSS over the memory. These straps run orthogonal to the metal3 layers below and are connected to their respective VDDP and VSS straps. At the chip level, users must make connections to the metal4 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 = MXR270,
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