TOTAL RECALL: WHAT TO LOOK FOR IN A MEMORY MODEL LIBRARY

MARK PERYER – VERIFICATION IP ARCHITECT, MENTOR GRAPHICS

WHITEPAPER

Mentor Graphics®

FUNCTIONAL VERIFICATION

www.mentor.com

INTRODUCTION

Almost all electronics systems use memory components, either for storing executable software or for storing data, therefore having accurate memory models available in proven, standards-based libraries is essential to the functional verification process. The models that make up the library should possess specific qualities, and the library itself should deliver a comprehensive solution that supports any type of simulation environment.

High fidelity memory models should include:

- Front door memory protocol interfaces
- Back door access
- Assertions
- Functional coverage monitors
- Memory protocol debug support
- Standards compliance
- Compatibility with all major simulators
- On-the-fly reconfiguration for second source evaluation

Mentor Graphics® now offers a new, comprehensive memory Verification IP (VIP) library that embodies all of these qualities and addresses the growing need for accurate memory simulation models.

MEMORY MODEL ESSENTIALS

For verification modelling purposes a memory device can be abstracted as a signal-level protocol interface to a storage array. The signal-level interface has to conform to the timing and behavior of the memory protocol, which may be specified in an industry standard, such as the JEDEC JES79-3F standard for DDR3 or, in a specific case, it may be described in a device manufacturer's datasheet. How the storage array is implemented is not directly visible to the user, but for simulation models it is generally implemented using either a SystemVerilog data structure or an optimized C data structure.

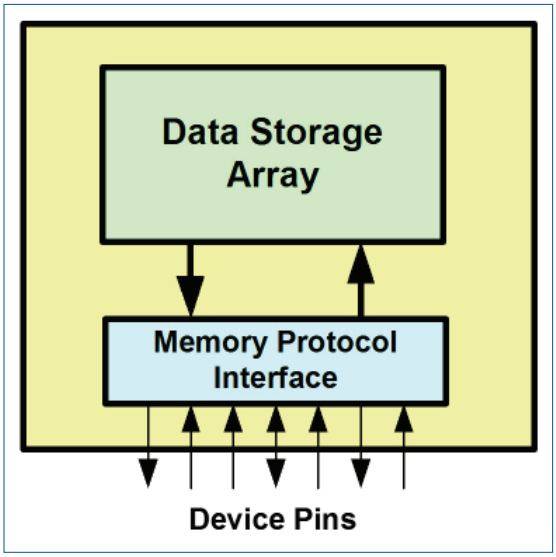

Figure 1: Generic abstraction of a memory device model

When a memory model is used in a testbench, it is instantiated as a component that is connected to a memory controller, which is either the design under test (DUT) or part of the DUT. Accesses to the memory take place using the signals of the memory model's protocol front end, and data is transferred in and out of the model's storage array. The complexity of the front-end protocol varies by memory type, but it can involve concurrent transfers interacting with a memory state space using control and status registers. Getting decent performance from some types of memory, such as DDR, relies on the controller recognizing certain types of data traffic and reorganizing the memory accesses to optimize the access rate in and out of memory. This level of sophistication requires a high fidelity memory model not only to reproduce the complex behavior and timing of a real memory device but also to determine whether the controller optimizations are effective.

The front door access to the memory over the protocol interface is mandatory for functional verification, but it does take time to transfer data in and out of the memory; therefore a backdoor interface is used to directly load or unload the memory storage array. The backdoor interface is typically used to load an executable software image in a memory model, or it may be used to load data content that is going to be manipulated by a hardware accelerator. The backdoor interface can also be used either to check memory content during a test or compare the memory data against a golden reference at the end of a test. A backdoor interface can be coupled to a memory debugger to allow memory content to be interactively viewed and changed.

As well as providing this useful front and back door functionality, other capabilities are required to enhance the usability of the memory model for verification. The memory model should provide a means of checking that the memory protocol is being followed and flagging errors as they occur. This is usually supported by the use of assertions that are fired when an error occurs, making it easier to get to the root cause of the problem. The memory model should provide a functional coverage monitor that tracks the different ways in which the protocol has been used. This can be used to check that a memory controller has been thoroughly verified, or to understand which modes of protocol operation have not been tested. Support for debugging the memory protocol is also important in order to efficiently trace the source of a bug.

THE MENTOR MEMORY LIBRARY

The Mentor memory model VIP library contains twenty five of the most commonly used memory types. Through configuration, the library supports thousands of models based on memory devices, and users are able to create their own configurations, allowing an almost infinite number of models to be supported.

DDR Memory Type Support	Flash Memory Type Support
DDR2, DDR3, DDR4	SDCard
LPDRR2, LPDDR3, LPDDR4	eMMC
UDIMM (DDR2, DDR3, DDR4)	ONFI
RDIMM (DDR2, DDR3, DDR4)	Serial Flash
LRDIMM (DDR3, DDR4)	NAND Flash
DFI	NOR Flash
Wide IO, Wide IO2	UFS
HMC	
HBM	

Table 1: Memory model types currently supported by the Mentor memory library.

All of the models available in the library can be used either as a stand-alone memory model or as a UVM agent, supporting any type of simulation environment. The Mentor VIP models are qualified on all of the three main EDA simulation platforms — Questa® from Mentor Graphics, Incisive® from Cadence®, and VCS® from Synopsys®.

The memory models are packaged as SystemVerilog modules with a pin-out corresponding to the modelled memory type. This allows them to be instantiated as components in a design netlist or in the top level of a test harness. The models respond to the signal-level protocol for front-door accesses and provide a back-door API that is common across the library.

The models provide full functionality and timing accuracy for each type of memory model. This includes optional functional mode settings and support of advanced operations, such as training and levelling, which is used to fine-tune the response of high-speed protocol interfaces, such as DDR4.

The timing and behavior of the models are highly configurable allowing them to be tuned to take on the personality of real memory devices. Each memory module has a MANUFACTURER and a PART_NUMBER parameter that allows you to specify which device the model should behave like. These parameters are used at the beginning of a simulation to set up the appropriate configuration options within the model. The value of these part number parameters can be changed as a simulator command line option. This makes it possible to change the part modelled without having to recompile the design and testbench, which is useful for checking that a second source component will work in a system design.

The memory models are supported by the Mentor VIP configuration GUI, which allows you to either instantiate a specific memory component model in a testbench created by the GUI or create a configuration file that can be loaded by the memory component. The GUI gives you access to all of the configuration options available for a specific memory type, including timing parameters. This allows you to create your own variant of a memory model to explore specific corner cases. The path to the configuration file generated by the GUI is another parameter of the memory module and, if specified, overrides the MANUFACTURER and PART_NUMBER configurations.

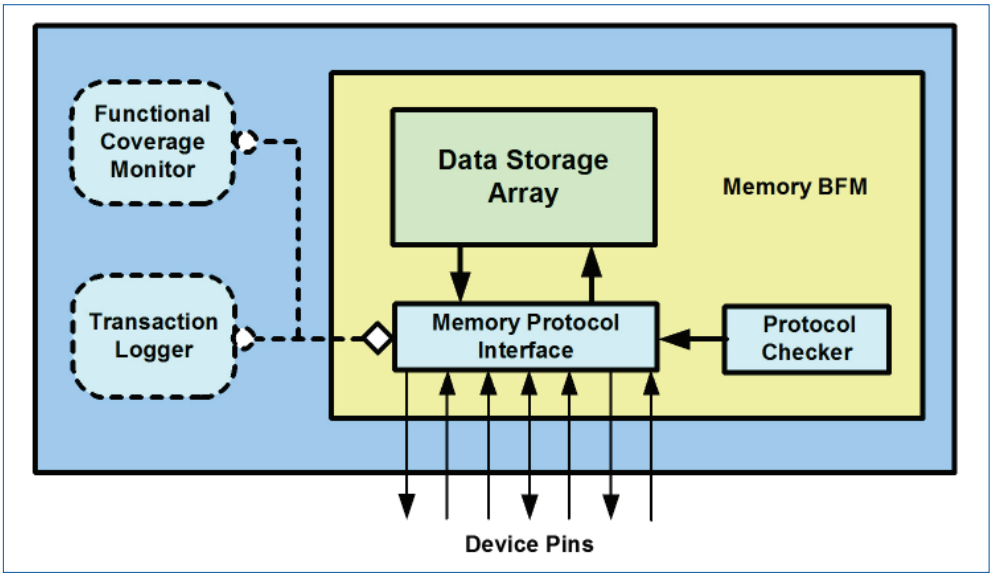

Figure 2: Mentor memory model block diagram.

The models have an API so you can reconfigure them during a simulation. The API allows either a new device part number to be specified or a different configuration file to be used. Almost any change can be made to the model using this approach. However, you must make sure that any change in behavior occurs at a reasonable place in the test, and you must be aware that the model may require re-initialization or training, depending on the extent of the reconfiguration.

Memory models have other module-level parameters that provide optional verification features. A functional coverage monitor for the memory protocol is activated using the ENABLE_FUNC_COV parameter. A memory transaction logger is turned on using the ENABLE _TXN_LOG parameter. The memory transaction logger either writes to the simulator transcript or to a specified log file, with the output being useful for tracing memory-level simulation activity.

Since the memory models are built on top of the Mentor Verification IP architecture, they have a built-in transactional debug facility that allows protocol activity to be viewed in a waveform window along with other design-level signals. This capability abstracts the memory accesses to high-level transactions, making it easy to understand what is going on in the memory protocol at any point in time.

ON THE FLY MODEL RECONFIGURATION

The general use case for the Mentor memory models is as stand-alone models that represent memory devices at the system level. In this case, they model a device's behavior for the duration of a test case and are not reconfigured. However, there are some memory controller verification scenarios where it is useful to change the timing or behavior of the model during the course of a simulation. This is supported by the Mentor memory models in one of three ways. Major changes to the configuration of the model to change the part number or to reconfigure it to a custom configuration can be made using a run-time API call — either specifying a new part number or the path to a new custom configuration file. Smaller run-time changes can be made by directly setting configuration variables in the model itself.

All of the available configuration variables for each model are documented in the comprehensive on-line documentation available with the library. For instance, making a subtle timing change to check that a memory controller can cope with a late timing response can be done by changing the appropriate timing variable on the fly.

Parameter	Purpose
MANUFACTURER	Identify part manufacturer
PART_NUMBER	Identifies device part number
CONFIG_FILE	The path to a custom configuration file created by the configuration GUI
ENABLE_FUNC_COV	Enables the functional coverage monitor
ENABLE_TXN_LOG	Enables the transaction logger for debug
TXN_LOG_FILE	File path for the transaction logger output
IF_NAME	Used with uvm_config_db to identify the models virtual interface handle

Table 2: Mentor memory model module parameters.

Changes to the configuration variables can be made using either hierarchical references in Verilog or VHDL testbenches or via a virtual interface handle in a SystemVerilog UVM testbench. The model places a reference to its virtual interface handle into the UVM `uvm_config_db` data structure, and components in the UVM testbench can reference the model's configuration parameters this way. Module-level parameters are provided to customize the naming of the path and key strings that are used for storing and retrieving the models virtual interface handle from the `uvm_config_db`.

CONCLUSION

The Mentor memory model library provides a comprehensive memory modelling solution and comes with an extensive range of configuration options that allow specific device parts or custom parts to be modelled very easily. The models come with built-in advanced verification features, can be used in any form of simulation-based verification environment, and are qualified to run on the Questa, Incisive, and VCS simulators.

Corporate Headquarters
Mentor Graphics Corporation
8005 SW Boeckman Road
Wilsonville, OR 97070-7777
Phone: 503.685.7000
Fax: 503.685.1204

Sales and Product Information
Phone: 800.547.3000
sales_info@mentor.com

Silicon Valley
Mentor Graphics Corporation
46871 Bayside Parkway
Fremont, CA 94538 USA
Phone: 510.354.7400
Fax: 510.354.7467

North American Support Center
Phone: 800.547.4303

Europe
Mentor Graphics
Deutschland GmbH
Arnulfstrasse 201
80634 Munich
Germany
Phone: +49.89.57096.0
Fax: +49.89.57096.400

Pacific Rim
Mentor Graphics (Taiwan)
11F, No. 120, Section 2,
Gongdao 5th Road
HsinChu City 300,
Taiwan, ROC
Phone: 886.3.513.1000
Fax: 886.3.573.4734

Japan
Mentor Graphics Japan Co., Ltd.
Gotenyama Garden
7-35, Kita-Shinagawa 4-chome
Shinagawa-Ku, Tokyo 140-0001
Japan
Phone: +81.3.5488.3033
Fax: +81.3.5488.3004

MGC 02-16 TECH14020-w