This User manual describes the VERILOG behavioral model for the FM25Q08A Serial Flash Memory device simply. For more information, please refer to the FM25Q08A_ds_eng.pdf.

Organization of the VERILOG Model Delivery package

The VERILOG model delivery package, is organized into a main directory, named *FM25Q08A.zip* containing six subdirectories with their related files (see the following list):

- **code** subdirectory: contains model source files;
- **include** subdirectory: contains parameters and constants definitions files;
- **sim** subdirectory: simulation initialization files;
- **stim** subdirectory: stimuli files used for simulation;
- **top** subdirectory: others file used for simulation.
- **Doc** subdirectory: user guide and datasheet.

Figure1.Package architecture

run_ncsim run_vcs verdirun verilog.flist

code

FM_TOP.v

include

FM_Decoders.h FM_DevParam.h FM_TimingData.h

doc

FM25Q08A_UserGuide.pdf   FM25Q08A_ds_eng.pdf

stim

read.v program_erase.v

powerdown.v securitysec.v

lockmanager.v statusreg.v

top

Testbench.v StimTasks.v ClockGenerator.v

StimGen_interf

1. Verilog behavioral model

The FM_TOP.*v* file of the code subdirectory contains the FM25Q08A behavioral model. It includes a set of modules that implement all the device functions listed in the datasheet.

These modules use a set of parameters defined in specific header files contained in the include subdirectory.

Section1.1 and *Section1.2* describe the model's Verilog modules and the header files.

Note: The model has been validated using the Cadence NC-SIM 9.2 simulator and the Synopsys VCS-2010.06 simulator with wave file dumped by the Synopsys Verdi-2010.07. The use with other simulators is not guaranteed.

Please refer to readme.txt file, for the reference datasheet used during model development and validation. Check the FMSH website or contact your local FMSH sales office, for the most recent version of the device datasheet.

1.1 Verilog modules

This section describes the FM25Q08A Verilog modules.

FM25Q08A

This is the “core” of the model, which is used to:
- Latch interface signals, data, addresses, and commands
- Execute read operations
- Organize and control the operations of all other modules

FM_UtilFunctions

This module contains utility functions used in various parts of the model.

FM_CUIdecoder

This module decodes the command sequences of the Flash memory.

FM_Memory

This module defines:
- The data structure used for representing the memory array
- The tasks that operate on this data structure to read, write, or erase elements of the array

FM_SecuritySectors

This module defines the data structure and the tasks for modeling OTP Security Sectors memory area.

FM_Program

This module implements the algorithms that control program and erase operations.

FM_Read

This module implements certain algorithms used in read operations.

FM_LockManager

This module implements the algorithms used to protect the device against program and erase operations (locking features).

FM_StatusRegister1

This module models the status register1 of the Flash memory.

FM_StatusRegister2

This module models the status register2 of the Flash memory.

FM_TimingCheck

During the simulation, this module controls the timing of the input signals, to check if the related constraints are respected.

1.2 Header files

This section describes the header files used in the model.

FM_Decoders.h

This file is used in FM25Q08A module. It contains all the instances of the CUI decoder module (each instance recognizes a specific command sequence of the Flash memory).

FM_DevParam.h

This file contains the definitions of constants related with memory characteristics. These constants are used in various parts of the model.

FM_TimingData.h

This file contains the definitions of memory's AC timing parameters.

1.3 Testbench and Stimuli files

To provide an example of a complete Verilog HDL project that the user can simulate, other Verilog HDL files are offered in addition to the Flash memory model.

The **top** subdirectory of the delivery package contains a testbench file as well as other additional files that can be used to simulate the model with various stimuli files.

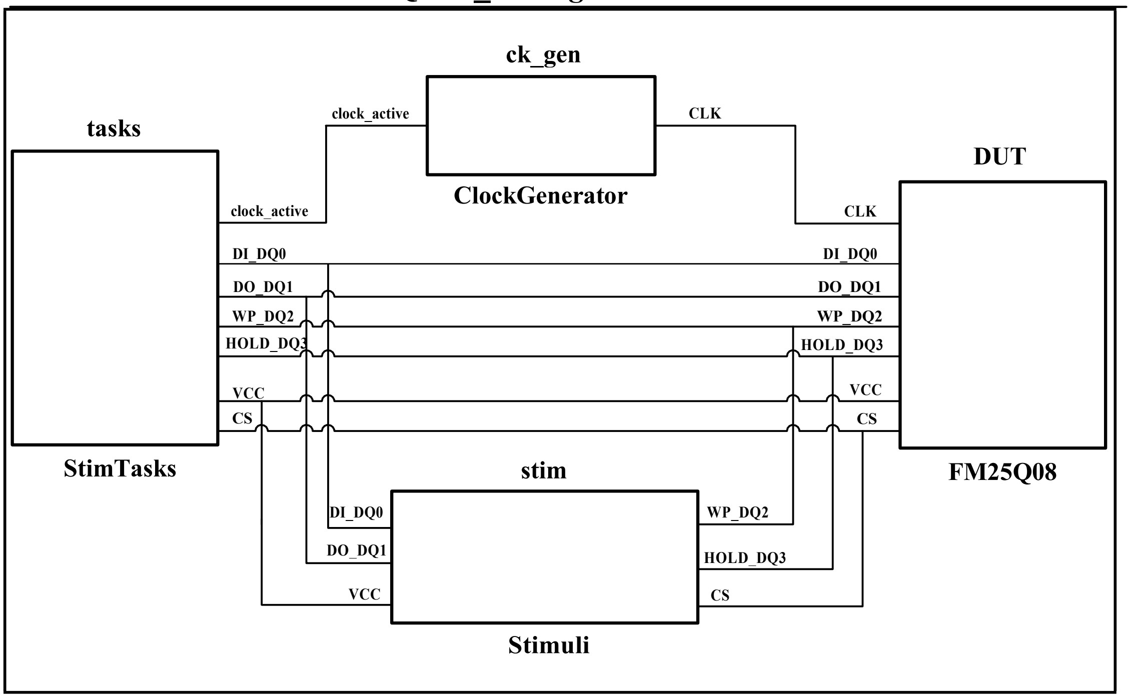
Stimuli files written in Verilog language are available in the **stim** subdirectory. These stimuli files cover many operational conditions of the device and, in particular, the CUI (command user interface) commands.

The following modules are instantiated in the *Testbench.v* file of the **top** directory:

- The StimTasks module (described in *top/StimTasks.v*) contains specific Verilog tasks invoked by stimuli generator module.
- The Stimuli module generates stimuli for the Flash memory by using the tasks provided by the StimTasks module. This module is implemented in various versions, each of which provide stimuli to simulate a specific device operation. The various versions of Stimuli (such as read.v and program_erase.v) are contained in the **stim** directory. The port interface of all stimuli files is defined in the header file *top/StimGen_interface.h*.
  The user can choose the operation to simulate by compiling a specific version of the stimuli file. For example, if the *stim/read.v* file is compiled, then the read operations are simulated.
- The ClockGenerator module generates clock signals that are connected as inputs to the Flash memory. Clock generation is controlled by the clock_active signal driven to ClockGenerator by the StimTasks procedures. (These procedures are activated by the Stimuli module.)
- The FM25Q08A module is the model of the serial Flash memory (device under test).

Figure2 illustrates the connections between the different modules described above.

Figure2. Testbench



The testbench illustrated here only provides an example for driving the FM25Q08A memory model. However, users can simulate the model using his own drivers and providing specific stimuli. In this case, they must define a new *Testbench.v* file to link the new user's drivers to the FM25Q08A memory model.

2. Simulation guidelines

2.1 Launching a simulation

The *run_ncsim* file (located in the main directory) is an example of script used to launch a simulation using a Cadence NC-SIM simulator. This script compiles and elaborates:

- The Verilog FM25Q08A model (the "include" files also are considered)
- The *StimTasks* and *ClockGenerator*
- One of the stimuli files contained in the **stim** directory. The user can change the operations to simulate, by compiling one of the stimuli files of the **stim** directory.

2.2 Memory file

To simplify the testing of the model functions, the memory array can been loaded with specific data at power-up. The format of the *memory file* must be as follows:

```
@hex_address
hex_data
hex_data_1
..........
```

hex_data is memorized at location *hex_address*, *hex_data_1* at the location *hex_address +1*, and so on. As an example:

@07F

4B

9A

....

Moreover, comments are allowed in the memory file lines using the notation:// *comment* The model is delivered with a template memory file called *mem.vmf* in the **sim** subdirectory. The name of memory file is defined as a parameter of the FM25Q08A module. It can be specified in the stimuli file using the following syntax:

defparamTestbench.DUT.memory_file="memory_file_name";

If the user does not provide the initialization file (memory_file:=“”), all the memory bits are set to ’1’.

2.3 Messages when running a simulation

When running a simulation, the FM25Q08A VerilogHDL model sends messages through the simulator console to prompt the status of the model. The following kinds of messages are provided:

- INFO: is normal information about the device status.
- WARNING: informs the user that the result of a command sent to the memory may not be what the user expects. For example, a warning message is provided when a program operation is aborted because the page to be programmed is locked.
- ERROR: informs the user that the FM25Q08A VerilogHDL model is not properly driven; that is, the provided stimuli sequence does not comply with FM25Q08A specifications.
- TIMING ERROR: this message is displayed when one of the input signals of the memory (driven by the stimuli file) does not respect the AC timing constraints of the memory device.

3. Simulation timings

To reduce simulation time, the values of certain latency times can be redefined. These values can be defined by setting variables in the *TimingData.h* file.

Table1 lists the variables that can be redefined by the user.

Table1. User-customizable simulation timings

Timing
program_delay
erase_sector_delay
erase_subblock_delay
erase_block_delay
erase_chip_delay
write_SR_delay (write Non Volatile Status Registers)
write_VSR_delay (write Volatile Status Registers)
full_access_power_up_delay
read_access_power_up_delay
SUS latency
RST latency
deep_power_down_delay
release_power_down_delay_1
release_power_down_delay_2

For instance if the user want change the *program_delay* to 100ns, he can redefine *program_delay* constant as follows:

parameter program_delay =100;

4. Model ports

The ports of FM25Q08A module connect the models to external devices. *Table2* list these ports and related types.

Table2. Model Ports for FM25Q08A devices

Port	Type	Description
CS	Input wire	Chip Select
CLK	Input wire	Serial Clock
HOLD_DQ3	Inout wire*	Hold /additional data I/O
DI_DQ0	Inout wire*	Serial data input
DO_DQ1	Inout wire*	Serial data output
VCC	[31:0] input wire **	Supply voltage
WP_DQ2	Inout wire*	Write Protect /additional data

* these ports are of “inout” type because in the dual or quad SPI protocol, and QPI protocol DI_DQ0, DO_DQ1, WP_DQ2 and HOLD_DQ3 pins act as an input/output.

** voltage signal is represented with 32 bit binary array, which decimal value corresponds to voltage value in millivolts.

Revision history

Date	**Version**	**Revision**
2017-11-15	1.0	FirstIssue.