This User manual describes the VERILOG behavioral model for the N25Q128xxxxx Serial Flash Memory device.

Organization of the VERILOG Model Delivery package

The VERILOG model delivery package, is organized into a main directory, named *NU_N25Q128xxxxx_VGx.x.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.

Figure 1. Package architecture

readme.txt
run_ncsim
run_modelsim

code
N25Q128xxxxx.v

include
Decoders.h
DevParam.h
TimingData.h

doc
N25Q128xxxxx_UserManual.pdf

stim
various stimuli files written in Verilog language

top
Testbench.v
StimTasks.v
ClockGenerator.v
StimGen_interface.h

sim
mem_Q128_xxx.vmf

Note: See the readme.txt file for the complete list of the files contained in each folder.

1. Verilog behavioral model

The *N25Q128_xxxxx.v* file of the code subdirectory contains the N25Q128xxxxx 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.

Section 1.1 and *Section 1.2* describe the model's Verilog modules and the header files.

Note: The model has been validated using a Cadence NC-SIM 5.7 simulator. 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 Numonyx web site or contact your local Numonyx sales office, for the most recent version of the device datasheet.

1.1 Verilog modules

This section describes the N25Q128xxxxx Verilog modules.

N25Qxxx
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

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

CUIdecoder
This module decodes the command sequences of the Flash memory.

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

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

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

Read
This module implements certain algorithms used in read operations.

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

StatusRegister

This module models the status register of the Flash memory.

FlagStatusRegister
This module models the flag status register.

NonVolatileConfigurationRegister
This module models the Non Volatile Configuration register.

VolatileConfigurationRegister
This module models the Volatile Configuration register.

VolatileEnhancedConfigurationRegister
This module models the Volatile Enhanced Configuration register.

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

DualQuadOps
This file runs the algorithms controlling the Dual and Quad read and program operations.

1.2 Header files

This section describes the header files used in the model.

Decoders.h
This file is used in N25Q128xxxxx module. It contains all the instances of the CUI decoder module
(each instance recognizes a specific command sequence of the Flash memory).

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

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 which provides stimuli to simulate a specific device operation. The various versions of stimuli (such as read.v and program.v) are contained in the **stim** directory. The port interface of all stimuli files is defined in the header file *top/StimGen_interface.h*.

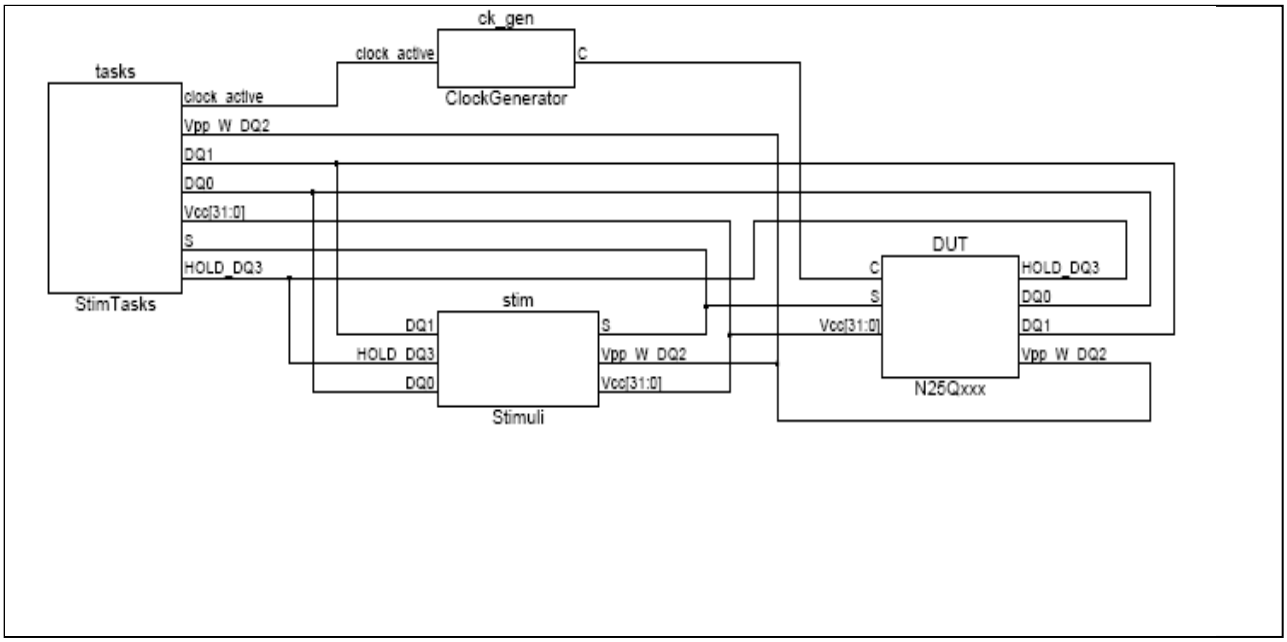
The user can choose the operation to simulate by compiling a specific version of the stimuli file. For example, if the *stim/read.v* file is compiled, then the read operations are simulated.

• 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 N25Q128xxxxx module is the model of the serial Flash memory (device under test).

Figure 2 illustrates the connections between the different modules described above.

Figure 2. Testbench



The testbench illustrated here only provides an example for driving the N25Q128xxxxx memory model. However, users can simulate the model using their 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 M25Pxxx 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 N25Q128xxxxx 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.

Moreover, the *run_modelsim* script file is provided for launching a simulation using a Mentor Modelsim simulator.

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 *N25Q128xxxxx* module. It can be specified in the stimuli file using the following syntax:

`defparam Testbench.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 N25Q128xxxxx Verilog HDL 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 N25Q128xxxxx Verilog HDL model is not properly driven; that is, the provided stimuli sequence does not comply with N25Q128xxxxx 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.

3.4 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.

Table 1 lists the variables that can be redefined by the user.

Table 1. User-customizable simulation timings

Timing
program_delay
erase_delay
erase_ss_delay (erase subsector)
erase_bulk_delay
write_SR_delay (write status register)
clear_FSR_delay (clear Flag Status Register)
write_NVCR_delay (write Non Volatile Configuration register)
write_VCR_delay (write Volatile Configuration register)
write_VECR_delay (write Volatile Enhanced Configuration register)
write_access_power_up_delay (tDTW)
read_access_power_up_delay (tVTR)
full_access_power_up_delay (tVTW)

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

parameter program_delay = 100;

4 Model ports

The ports of M25Pxxx module connect the models to external devices. *Table 2* list these ports and related types.

Table 2. Model Ports for N25Q128xxxxx devices

Port	Type	Description
S	input wire	Chip Select
C	input wire	Serial Clock
HOLD DQ3	inout wire *	Hold /additional data I/O
DQ0	inout wire *	Serial data input
DQ1	inout wire *	Serial data output
Vcc	[31 : 0] input wire **	Supply voltage
W/Vpp DQ2	inout wire **	Write Protect /Enhanced Program supply voltage*** /additional data I/O

* these ports are of “inout” type because in the dual and quad protocol DQ0,DQ1, W/Vpp 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 .

*** Enhanced Program supply voltage not implemented.

Revision history

Date	Version	Revision
23-3-2009	1.0	First Issue.

Legal Disclaimer

Please Read Carefully:

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH NUMONYX™ PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN NUMONYX'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, NUMONYX ASSUMES NO LIABILITY WHATSOEVER, AND NUMONYX DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF NUMONYX PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.
Numonyx products are not intended for use in medical, life saving, life sustaining, critical control or safety systems, or in nuclear facility applications.
Numonyx may make changes to specifications and product descriptions at any time, without notice.
Numonyx, B.V. may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.
Designers must not rely on the absence or characteristics of any features or instructions marked “reserved” or “undefined.” Numonyx reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.
Contact your local Numonyx sales office or your distributor to obtain the latest specifications and before placing your product order.
Copies of documents which have an order number and are referenced in this document, or other Numonyx literature may be obtained by visiting the Numonyx website at http://www.numonyx.com.
Numonyx StrataFlash is a trademark or registered trademark of Numonyx or its subsidiaries in the United States and other countries.
*Other names and brands may be claimed as the property of others.