

Verilog model for the F48 technology, 64-Gbit (8 x 8 Gbits), 8 Chip Enable, NAND SLC very large page memory device

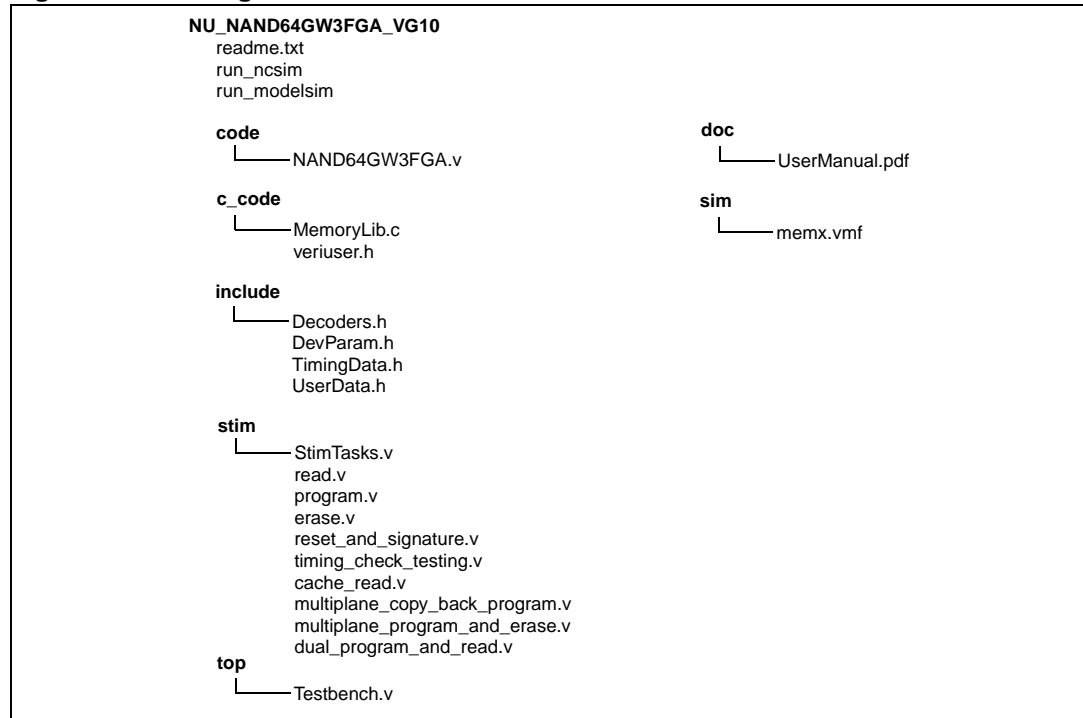
This user guide describes the Verilog behavioral model for the NAND64GW3FGA flash memory device.

Organization of the Verilog model delivery package

The Verilog model delivery package, `NU_NAND64GW3FGA_VG10.zip`, is organized into a main directory, named `NU_NAND64GW3FGA_VG10`, that contains the following seven subdirectories and their related files (see [Figure 1](#)):

- **code** subdirectory: contains the model source files
- **c_code** subdirectory: contains the model source files written using C language
- **include** subdirectory: contains the header files
- **doc** subdirectory: contains the model documentation
- **sim** subdirectory: contains the simulation initialization files
- **stim** subdirectory: contains the stimuli files used for simulation
- **top** subdirectory: contains the testbench file used for simulation

Figure 1. Package architecture



1. See the `readme.txt` file for the complete list of files contained in each folder.

Contents

- 1 Device description 3**

- 2 Verilog behavioral model 4**
 - 2.1 Verilog modules 4
 - 2.2 Header files 5
 - 2.3 C libraries 5
 - 2.4 Verilog testbench and stimuli files 6

- 3 Simulation guidelines 7**
 - 3.1 Launching a simulation 7
 - 3.2 Memory file format 8
 - 3.3 Customize model and simulation 8
 - 3.4 Simulation timings 9

- 4 Verilog types used in model ports 10**

- 5 Revision history 11**

1 Device description

The NAND64GW3FGA is part of the single level cell (SLC), 4224-byte page family of non-volatile NAND flash memories. The device has a density of 64 Gbits and combines eight 8-Gbit dice in a stacked device. Each 8-Gbit dice has its own Chip Enable and Ready/Busy pin so that it can be driven independently using the relative Chip Enable pin. The eight 8-Gbit dice are grouped by four (group A and group B) and each group of four 8-Gbit dice shares the same inputs/outputs bus and input signals. The device operates from a 3 V power supply.

In addition, each 8-Gbit dice has its own maximum number of bad blocks and its own electronic signature code.

2 Verilog behavioral model

The NAND64GW3FGA Verilog behavioral model is located in the `NAND64GW3FGA.v` file of the `code` subdirectory. It includes a set of modules that implement all the device functions listed in the device datasheet (see [Section 2.1: Verilog modules](#)). The modules use a set of parameters defined in specific header files that are in the `include` subdirectory (see [Section 2.2: Header files](#)).

Some model functionalities are implemented using C language. In this case, a PLI library (see [Section 2.3: C libraries](#)) is used to transfer data between C code and Verilog code. The section of the model written in C language is placed in the `c_code` subdirectory.

The Verilog model was validated using a Cadence NC-SIM 5.7 simulator. The use of the model with other simulators is not guaranteed.

Refer to the `readme.txt` file in the main package directory for the reference datasheet used during model development and validation. Go to the Numonyx web site on www.numonyx.com or contact your local Numonyx sales office for the most recent version of the device datasheet.

2.1 Verilog modules

[Table 1](#) describes the modules of the NAND64GW3FGA Verilog model.

Table 1. Description of the NAND64GW3FGA Verilog model modules

Module	Description
NANDxF_quad	The 'core' of the quadruple stacked device model (one for each group of four 8-Gbit dice).
NANDxF	The 'core' of the single die model. It latches signals, data, addresses and commands, and uses all other modules.
UtilFunctions	Contains utility functions used in various parts of the model.
CUIdecoder	Decodes command sequences of the flash memory.
Memory	Implements the basic operations for the read and write memory matrix by updating the page index and data transfers between the page buffer and the flash memory.
Program	Implements program and erase operations.
Read	Models read operations.
OutputBusManager	Models the output bus where output data are written on the bus in accordance with expected timings.
LockManager	Implements features (locking features) to protect the device against program and erase operations.
StatusRegister	Models the status register of the flash memory.
Signature	Contains the electronic signature data.
TimingCheck	Checks the timing of input signals during simulation, to verify if related constraints are respected.

2.2 Header files

[Table 2](#) describes the header files used in the NAND64GW3FGA Verilog model.

Table 2. Description of the NAND64GW3FGA Verilog model header files

Header file	Description
Decoders.h	Used in the NANDxF module and contains all instances of the CUIdecoder module, where each instance recognizes a specific command sequence of flash memory.
UserData.h	Contains definitions (device specifications and timing check options) that can be changed by users. It is the only header file that can be modified by the user (see Section 3.3: Customize model and simulation).
DevParam.h	Contains definitions of constants related with memory characteristics used in various parts of the model.
TimingData.h	Contains the definitions of the timing constraints.

2.3 C libraries

The memory array data structure is implemented in C language as a dynamic list of pages to optimize simulation performance. Each node of the list contains an array that represents one page. Only the pages programmed by the user are present in the list.

A `MemoryLib.c` file contains definitions of the dynamic list and of the tasks that operate on this data structure. These C tasks are used as system tasks in the Verilog code.

A PLI library transfers data between C code and Verilog code. PLI is a standard library, defined in the file `veriusr.h` that is in the `c_code` directory.

The code `NAND64GW3FGA.v` Verilog file and the C library file must be compiled in the same order in the `run_ncsim` file.

2.4 Verilog testbench and stimuli files

The `top` subdirectory of the Verilog model delivery package contains a testbench file, `Testbench.v`, which allows the model to be simulated using various stimuli files.

Stimuli files written in Verilog format are available in the `stim` subdirectory. These files cover many operational conditions of the device, in particular the command user interface (CUI) commands. Stimuli files use specific Verilog tasks contained in the file `stim/StimTasks.v`.

3 Simulation guidelines

3.1 Launching a simulation

The `run_ncsim` file (located in the main directory) is an example of the script used to launch a simulation using a Cadence NC-SIM simulator. This script compiles and elaborates the Verilog model, C libraries, testbench, and stimuli files.

To build the simulation script (for Cadence NCSIM, or Mentor Modelsim simulators), the following steps must be respected:

1. Compile the C code and build the dynamic library.

```
[Linux and Solaris / all simulators]:  
gcc -c <C_code_file_name.c> -o <object_file_name.o>  
ld -G <object_file_name.o> -o <dynamic_library_name.so>
```

2. Compile the Verilog code.

```
[NCSIM]:  
ncvlog -cdslib <cds.lib path> -hdlvar <hdl.var path>  
<file_to_be_compiled.v>  
.....
```

```
[Modelsim]:  
vlog -work <work_dir path> <file_to_be_compiled.v>  
.....
```

3. Elaborate the Verilog code (NCSIM only).

```
ncelab -cdslib <cds.lib path> -hdlvar <hdl.var path> -loadpli1  
<dynamic_library_path>:bootstrap_fun  
      <snapshot_name>
```

4. Launch the simulation.

```
[NCSIM]:  
ncsim -cdslib <cds.lib path> -hdlvar <hdl.var path>  
<snapshot_name> -gui
```

```
[Modelsim]:  
vsim <top_level_module> -pli <dynamic_library_path>
```

3.2 Memory file format

To simplify testing of the model functionalities, the memory array must be loaded with specific data at power-up.

The format of the memory file is:

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

Thus, `hex_data` is memorized at location `hex_address`, `hex_data_1` is located at the location `hex_address + 1`, and so on.

For example:

```
@07FFFF
4B
9A
.....
```

Comments are allowed in the memory file lines using the notation:

```
/ / comment
```

The model is delivered with a template memory file called `memx.vmf`, which is located in the **sim** subdirectory (see [Figure 1](#)).

For the NAND64GW3FGA stacked device four memory files (for each group of four 8-Gbit dice) have to be defined as parameter. In this case, in the stimuli file, the following syntax can be used:

```
defparam testbench.DUT.memory_file1 = "memory_file_name";
defparam testbench.DUT.memory_file2 = "memory_file_name";
defparam testbench.DUT.memory_file3 = "memory_file_name";
defparam testbench.DUT.memory_file4 = "memory_file_name";
```

3.3 Customize model and simulation

Some features and characteristics of the model and simulation process can be customized by the user. Parameters that can be modified are contained in the `UserData.h` header file.

They include:

1. Device name definition

This is the first definition contained in `UserData.h`. It specifies which device is described by the model (each device is characterized by memory size, bus width, supply voltage and timing constraints).
2. TimingCheck

If this string is set to 'on', all timing checks are performed during the simulation. If it is set to 'off', no timing checks are made.

3.4 Simulation timings

The user can reduce some of the latency time values to reduce simulation time. These values can be fixed by setting variables in the `TimingData.h` file. Variables that can be fixed by the user are listed in [Table 3](#).

For example, if the user wants to change the program delay to 100 ns, the program delay constant can be redefined as follows:

```
parameter program_delay = 100;
```

Table 3. Customizable simulation timings

Timing
read_delay
cache_read_delay
program_delay
erase_delay
mult_read_delay
multCacheRead_delay
multProgBusy_delay
multEraseBusy_delay
busy_delay

4 Verilog types used in model ports

Ports of the NAND64GW3FGA module are used to connect the Verilog model with external devices. These ports and related port types are listed in [Table 4](#).

Table 4. Verilog behavioral model ports

Port	Type	Description
V _{DD}	[31:0] input wire ⁽¹⁾	Supply voltage
WP _{A_N}	Input wire	Write Protect
WP _{B_N}	Input wire	Write Protect
AL _A	Input wire	Address Latch Enable
AL _B	Input wire	Address Latch Enable
CL _A	Input wire	Command Latch Enable
CL _B	Input wire	Command Latch Enable
E _{A1_N} to E _{A4_N}	Input wire	Chip Enable
E _{B1_N} to E _{B4_N}	Input wire	Chip Enable
R _{A_N}	Input wire	Read Enable
R _{B_N}	Input wire	Read Enable
W _{A_N}	Input wire	Write Enable
W _{B_N}	Input wire	Write Enable
IO _A	[busDim-1:0] output wire ⁽²⁾	Input output bus
IO _B	[busDim-1:0] output wire ⁽²⁾	Input output bus
RB _{A1_N} to RB _{A4_N}	Output wire	Ready/Busy signal
RB _{B1_N} to RB _{B4_N}	Output wire	Ready/Busy signal
dump ⁽³⁾	Input wire	Dump memory

1. Voltage signal is represented with a 32-bit binary array which corresponds to a voltage value in millivolts
2. busDim = 8.
3. The dump feature is an additional input of the Verilog model (not present in the device) that is used for dumping memory content in a file. It is not implemented in the current version and must be left unconnected (high impedance).

5 Revision history

Table 5. Document revision history

Date	Revision	Changes
02-Jul-2009	1	Initial release.

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 Numonyx's 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.

Copyright © 11/5/7, Numonyx, B.V., All Rights Reserved.