MEMAKER (200901.2.1)

User Guide
Rev.: 200901.2.1v1.0
Issue Date: January 2010

REVISION HISTORY

Memaker User Guide

Date	Rev.	From	To
May 2004	1.0	-	Original
Jun. 2004	1.1	-	Changed pictures
Jun. 2004	1.2	-	Added the ROM description
Nov. 2004	2.0	-	Upgraded to version 200410.1.1
Jun. 2005	200410.3.01.21	-	• Removed the support for the HP platform • Added the support for the Mentor FastScan and MBIST models
Jul. 2005	200410.3.01.22	-	Added more information on license features and package release
Nov. 2005	200410.4.01.23	-	• Added the support for the Memaker rule file generation • Changed the extension name of the Mentor FastScan model from “_Matpg.lib” to “.fastscan” • Updated the error/warning message list • Updated the Memaker batching mode generation
Dec. 2005	200601.1.1.24	-	Changed the tool version from 200410.4.01 to 200601.1.1
Aug. 2006	200601.3.1v1.0	The TetraMAX ATPG models and Memory Definition Table (MDT) models were not supported.	Added the support for the TetraMAX ATPG models and the Memory Definition Table (MDT) models
Nov. 2007	200701.1.1v1.0	-	Updated for the 200701.1.1 release
Jan. 2008	200801.1.1v1.0	-	Updated for the 200801.1.1 release
Jun. 2008	200802.1.1v1.0	-	• Updated for the 200802.1.1 release • Added the support for the -fullspeed option and removed the -ecs and -pv options • Updated the FLEXlm installation • Updated the batch mode commands • Updated the message list

Date	Rev.	From	To
Sept. 2008	200802.2.1v1.0	-	• Added the support for terminating the model generation process by clicking the “Cancel” button • Changed the GUI style of the memory type selection from the radio button to the combo box • Added the support for the bus mode and the multiple ring types selection
May 2009	200901.1.1v1.0	-	• Updated to version 200901.1.1 • Updated the installation guide • Updated the error/warning message list
Jan. 2010	200901.2.1v1.0	-	• Updated to version 200901.2.1 • Supported the layer translation function of the UMC LEF model for some libraries

Printed in Taiwan 2010

Faraday Technology Corporation
No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Faraday's home page can be found at:
http://www.faraday-tech.com

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Installation Guide

This chapter contains the following sections:

1.1 System Requirements

1.1.1 Operating System

- Sun OS 5.8 or later
- Linux i686 with 32-bit core 2.4.21 or later
- Linux x86_64 (AMD64) with 64-bit core 2.4.21 or later

1.1.2 Hard Disk Space

- For each COT package release, a free hard disk space ranging from 100 MB to 280 MB is needed for installing the design kit.
- For each ASIC package release, a free hard disk space of 850 MB is required for installing the design kit.

1.1.3 Memory

- Minimum: 40 MB RAM
- Recommended: 128 MB RAM
- The actual amount of memory needed is proportional to the size of a design.

1.2 Installation Instructions

1.2.1 Installing Memaker

1. Go to the <install_path> directory
2. Extract the design kit from the *.tar.gz* file
3. The directory structure will be extracted as listed in the previous page

1.2.2 Setting the Environment

For the COT package:

1. Edit the environment file <install_path>/memaker.env
    a. Set the FTC environment to <install_path>
    b. Set the LM_LICENSE_FILE path to <your_license_file>

For the ASIC package:

1. Edit the environment file <install_path>/FTC.env
    a. Set the FTC environment to <install_path>
    b. Set the LM_LICENSE_FILE path to <your_license_file>

1.2.3 Installing FLEXlm License Manager Software (Only for COT Package Release)

1. Memaker is controlled by a floating license; select a workstation to function as the program server.
2. Obtain a license file from Faraday:
    a. Log on to the website, http://freelibrary.faraday-tech.com, with your AIP downloadable account
    b. Look for the words "Memaker License Download" (Design Kit License Download for ASIC customers) on the license download page
    c. Add the host ID of your machine into the database
    d. Click the "Download" button to send an email to your inbox

3. Install the FLEXlm software to the license server:
   a. Assume that the license server is running on SunOS with a hostname, "licsun01," and set the license daemon port number to the default port number, "99993"
   b. Once the design kit is extracted, a directory named "flexlm" is created under <$FTC> with the following contents:

```
$FTC/flexlm
|bin
   |solaris     FLEXlm utility for SUN platform
      |ftclmd   Vendor daemon
      |lmgrd    FLEXlm daemon manager
      |lmutil   License Administration Utility
   |linux       FLEXlm utility for Linux(32-bit) platform
      |ftclmd   Vendor daemon
      |lmgrd    FLEXlm daemon manager
      |lmutil   License Administration Utility
   |linux64     FLEXlm utility for Linux(64-bit) platform
      |ftclmd   Vendor daemon
      |lmgrd    FLEXlm daemon manager
      |lmutil   License Administration Utility
```

   c. Copy the generated license file (As shown above) into the $FTC/flexlm/license/ directory
   d. Set the license path by using one of the following four methods. Method 4 is mostly recommended.

- Method 1:

  %> setenv LM_LICENSE_FILE $FTC/flexlm/license/memaker.dat
- Method 2:

  %> setenv LM_LICENSE_FILE 99993@licsun01
- Method 3:

  %> setenv FTCLMD_LICENSE_FILE $FTC/flexlm/license/memaker.dat
- Method 4:

  %> setenv FTCLMD_LICENSE_FILE 99993@licsun01

Notes:

1. "FTCLMD" is the daemon name of the Faraday vendor.
2. The communication port of the Faraday vendor daemon is set to "99993".

e. Modify the license file as follows:

The following is an example of the license file that can be downloaded from the Faraday website.

```
#============================================================
#          FARADAY MEMAKER LICENSE
#============================================================
# Date : Thursday 05th of June 2008 10:41:41 AM
#
# Please refer to license installation documentation for
# details on installing your license authorization file.
#============================================================

SERVER licsun01 aaaaa 99993
VENDOR ftclmd /path_to_the_ftclmd/ftclmd
FEATURE MEMAKER ftclmd 2008.01 05-dec-2008 10 SIGN="0032 502B 5935 \
  E641 C932 0824 25DF 4100 D26F 23E9 75CE 8B3C A730 9368 8A6F"
```

f. Replace "server_hostname" with the hostname of your license server. Please refer to Section 1.5.1 for information on how to determine the hostname of your license server.

`aaaaa is the host ID that added from Faraday free library license download page.`

g. Modify "path_to_ftclmd" to the path of the vendor daemon (ftclmd).

h. The default "port" of the license file is "99993". Users may need to modify this number if it is not available on the license server.

1.3 Launching the License Daemon

1. Start the license daemon

   %> $FTC/flexlm/bin/solaris/lmgrd -c $FTC/flexlm/license/memaker.dat -l logfile

   -c: Used to assign the location of the license file

   -l: Used to assign the log file
2. Check the daemon status

   %> $FTC/flexlm/bin/solaris/lmutil lmstat -a -c $FTC/flexlm/license/memaker.dat

   or

   %> $FTC/flexlm/bin/solaris/lmutil lmstat -a -c 99993@licsun01
3. Once users have launched the license daemon, and the log file does not contain any error message, users are ready to run the design kit program.

1.4 Running Memaker

%> source memaker.env

%> memaker

1.5 Lisecnse Server/File Q & A

1.5.1 How to Determine the Host ID

FLEXlm uses different identification methods as the host IDs of different machine architectures. To determine the host ID of a machine, please refer to the following table:

Machine Platform	Command
SUN	hostid
Linux (32-bit and 64-bit)	lmhostid

1.5.2 How to Determine the Hostname

To retrieve the hostname of a machine, please refer to the following table:

Platform	Command
SUN	hostname
Linux (32-bit and 64-bit)	hostname

1.5.3 How to Update the License File

To update or reload the license file, please issue the following command:

%> lmreread -c $FTC/flexlm/license/memaker.dat

1.5.4 Shut Down the License

To shut down the license file or server, please run one of the following commands:

%> $FTC/flexlm/bin/solaris/lmdown -c $FTC/flexlm/license/memaker.dat

or

%> $FTC/flexlm/bin/solaris/lmdown -c 99993@licsun01

Chapter 2

Introduction to Memaker

This chapter contains the following sections:

2.1 Overview

Memaker, the Faraday proprietary memory compiler, is capable of automatically generating the data sheet, Verilog simulation model, VHDL simulation model, Synopsys synthesis model, Mentor FastScan and MBIST models, GDSII file, LEF file, and SPICE netlist file of a specific memory configuration. During the execution, the Graphic User Interface (GUI) of Memaker will guide users on how to choose a proper size of the memory with the required performance and dimension.

2.2 Memaker Functions

Memaker provides the following deliverables:

Package	Deliverable Name	EDA View
Design kit	Synthesis model	Synopsys liberty model
	Simulation models	Verilog model
		VHDL model
	ATPG model	Mentor Graphic FastScan model
	TetraMax model	Synopsys TetraMAX model
	MBIST model	Mentor Graphic Architect MBIST model
	MDT model	Novas Verdi MDT behavior models
	P & R physical model	Cadence LEF P & R model
Tape-out kit	Netlist	LVS netlist
	Physical layout	GDSII

Note: The tape-out kit includes all items in the design kit.

2.3 Memaker Input/Output Relationship

Figure 2-1 shows the relationship between the inputs and outputs.

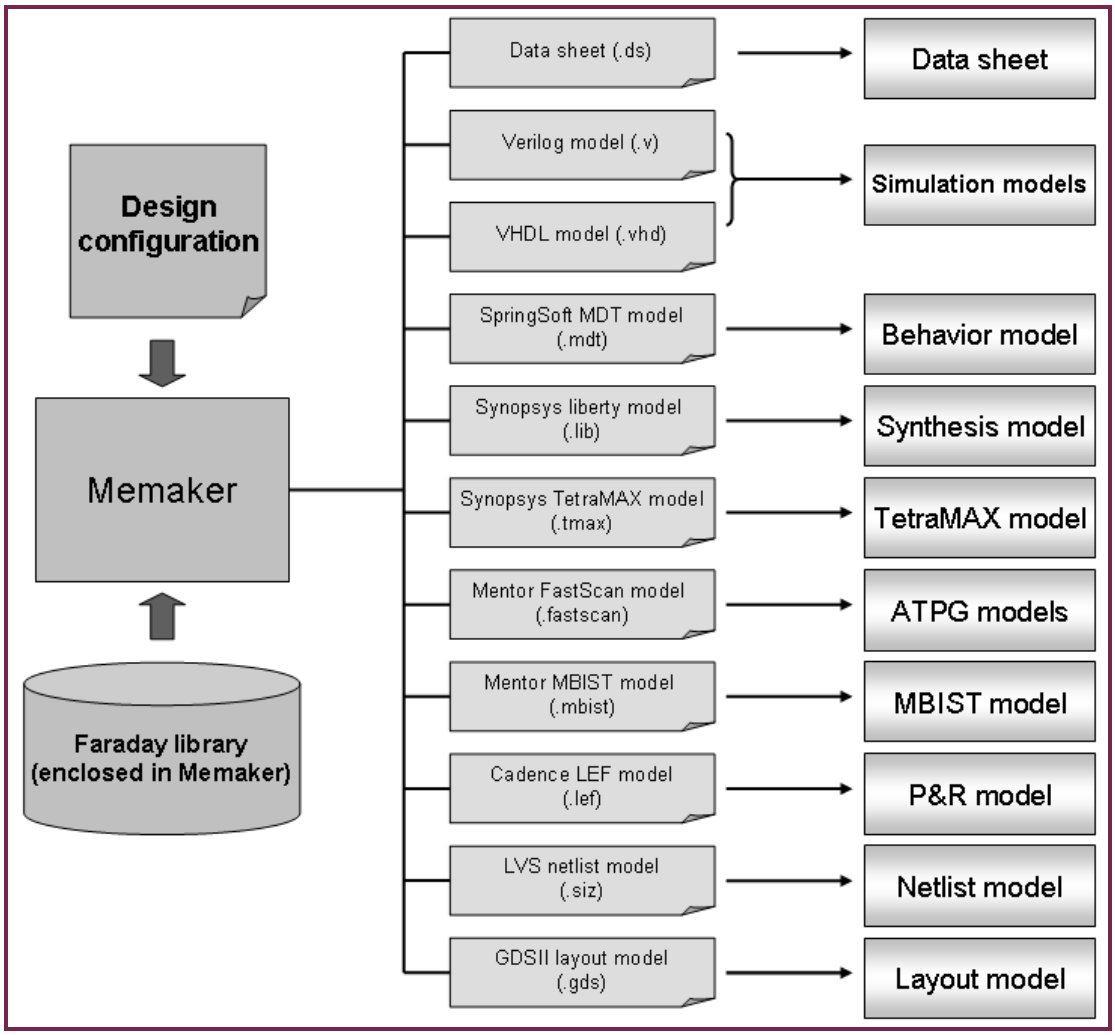

Figure 2-1. Memaker Input/Output Relationship

Users should provide the design information that describes the memory configuration and the external environment conditions.[1] With the embedded technology library, the specific memory models can be generated and used for synthesis, simulation, P & R, and layout verification.

[1] Users can assign a configuration file, an instance name list file, or a Memaker rule file as the input, or the configurations in the Memaker GUI. Please refer to Sections 3.5 for details.

Chapter 3

Running Memaker

This chapter contains the following sections:

3.1 System Requirements

Memaker supports the following OS platforms:

- Solaris 5.8 or above
- Linux i686 with 32-bit core 2.4.21 or above
- Linux x86_64 (AMD64) with 64-bit core 2.4.21 or above

Approximately 40 MB memory space is required to run Memaker.

3.2 License Requirements

The following license features are required to run Memaker.

Table 3-1. License Features of Memaker

License Feature	Memaker Option	Note
MEMAKER	GUI-Memaker, UI-Batch mode	Common features
WISESILICON	WiseSilicon engine (COT package only)	
SU180_GDS	GDS file generation license	Depends on the available compilers and licenses
SU180_LEF	LEF file generation license	
SU180_SIZ	Size file generation license	

Note: The GDS, SIZ, and LEF features are not available in the ASIC package.

3.3 Setting Environment Variables

Please refer to the installation guide to set up the environment.

3.4 Memaker Command Syntax

To run Memaker in the GUI mode, the syntax is as follows:

<Syntax>: memaker

Due to the large volume of pre-characterization data that the system must compute, users may need to wait until the program is fully initialized. During the initialization, the message, "Please wait while loading memaker package…", will appear on the terminal.

3.4.1 Memaker GUI

In this section, the main page window and different ways of using the Memaker software are addressed.

3.4.2 Memaker Main Page

Figure 3-1 shows the main page window of Memaker. The usages of each entry are listed below:

- Menu bar
  The "Menu" bar provides two pull-down menus, the mode selection menu and the help menu, which contains the online release notes and version information, as shown in "Help".
- Mode
  The "Mode" bar allows users to switch between the express mode and the smart mode. The express mode is a faster and more efficient entry manner. The smart mode provides more detailed data comparison, which is similar to a decision system. New users are recommended to use the smart mode to help them make the most appropriate selections.
- Compiler
  The "Compiler" bar allows users to switch between the compilers that Memaker has already ported. It includes two combo boxes: The memory type and the optional feature selections. Users can select the preferred memory type, and then select the optional features, such as the BTI interface and the row redundancy repair function, from the combo box. The warning message will appear in red color if this memory type (Feature) is not free of charge.

- W/B write

  The “W/B write” bar allows users to choose between the word write and byte write in the RAM types. This bar will reveal the “ROM Type” selection and then hide the “Word Write/Byte Write” in the ROM type memory.
- Design configuration

  This design configuration allows users to enter the design-related configurations, which include the words, bits, bytes (If the byte-write is selected), Mux, ROM code version, ROM code file (If the ROM type memory is selected), output loading, data slew rate, clock slew rate, power ring width, instance name, and specified output files directory, if necessary.
- Scene window

  The scene window will show users how the aspect ratio of the instance changes as the words, bits, bytes, and Mux parameters are modified.
- Model button

  The “Model” button allows users to select among two categories: The front-end and back-end models. In the COT package release, if users are not licensed for the GDSII or SPICE netlist features, the respective check box will be grayed. In the ASIC release, users will not see the check boxes of LEF, GDSII, and SPICE netlist.
- Operation panel

  The operation panel provides users with three convenient options: “Select all,” “Unselect all,” and “Invert”. The “Select all” option allows users to select all models. The “Unselect all” option allows users to unselect all models. The “Invert” option allows users to invert the current selection so that all models, other than the one selected currently, will be selected. Once the model is selected, users should click the “Generate” button to generate the models.
- Bus mode generation

  The bus mode generation check box allows users to generate the bus mode models when the “Generate” button is clicked.
- Model generation

  The “Generate” button allows users to generate the memory models according to the design configurations, selecting models, and bus mode generation box.
- Message window

  The status messages will be displayed in this window during the execution of Memaker.

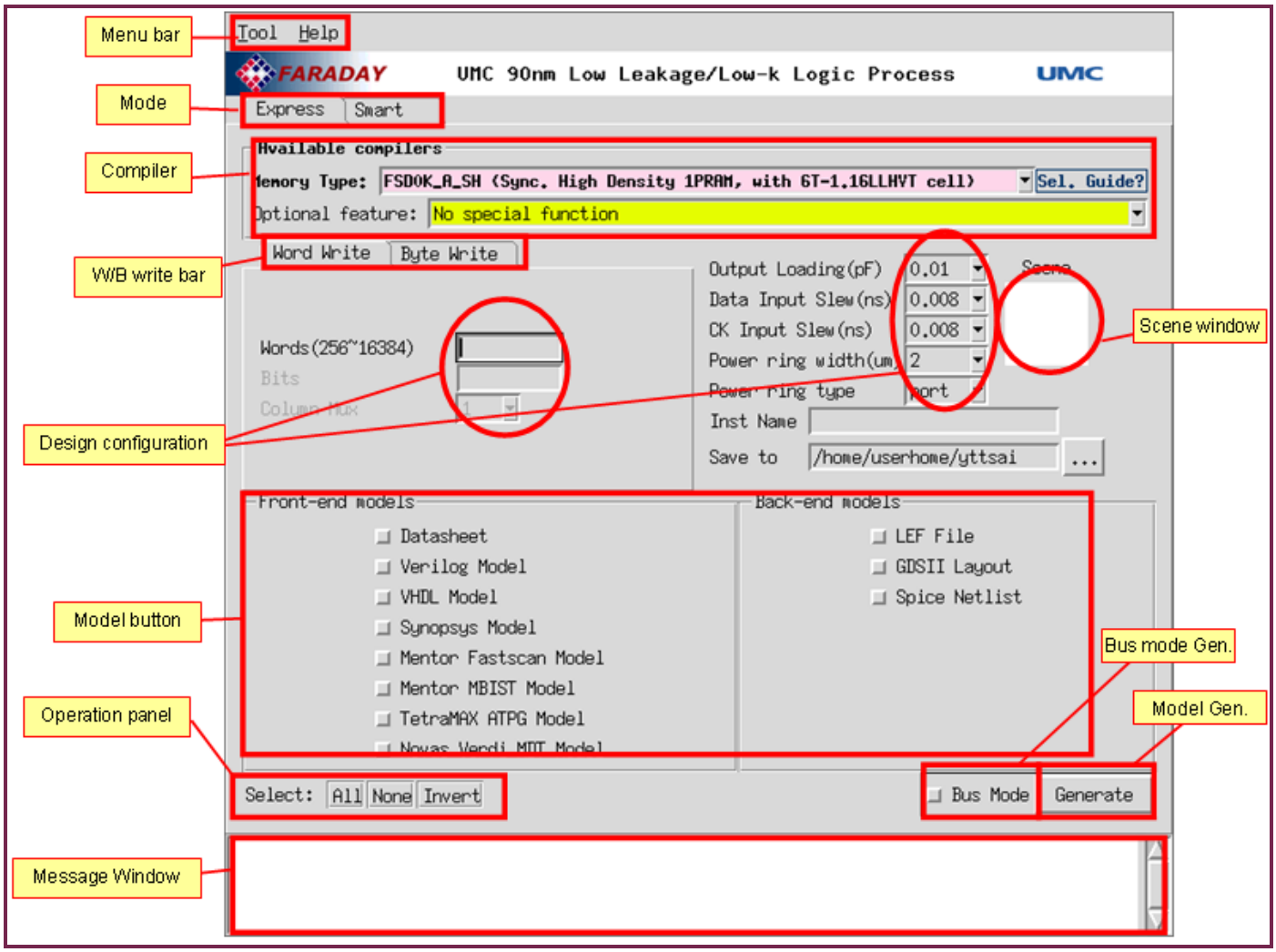

Figure 3-1. Memaker Main Page Window

3.4.3 Express Mode

The express mode provides users with a faster approach to generate the required models. It is the default choice of Memaker. To use the express mode, users should make sure that the "Express" tab page is active. If not, users should click the "Express" page title to activate the "Express" tab page. Click the "Word Write" page title once to enter the desired values in the words, bits, and Mux fields. Click the "Byte Write" page title (Please refer to Figure 3-2) and enter the desired words, bit/byte, bytes/word, and Mux fields.

After entering the values into the appropriate fields, users then click the "Generate" button to process the entries. If the model is successfully generated, a message window, as shown in Figure 3-3, will appear. At the same time, an output file will be generated in the current directory or the specified directory.

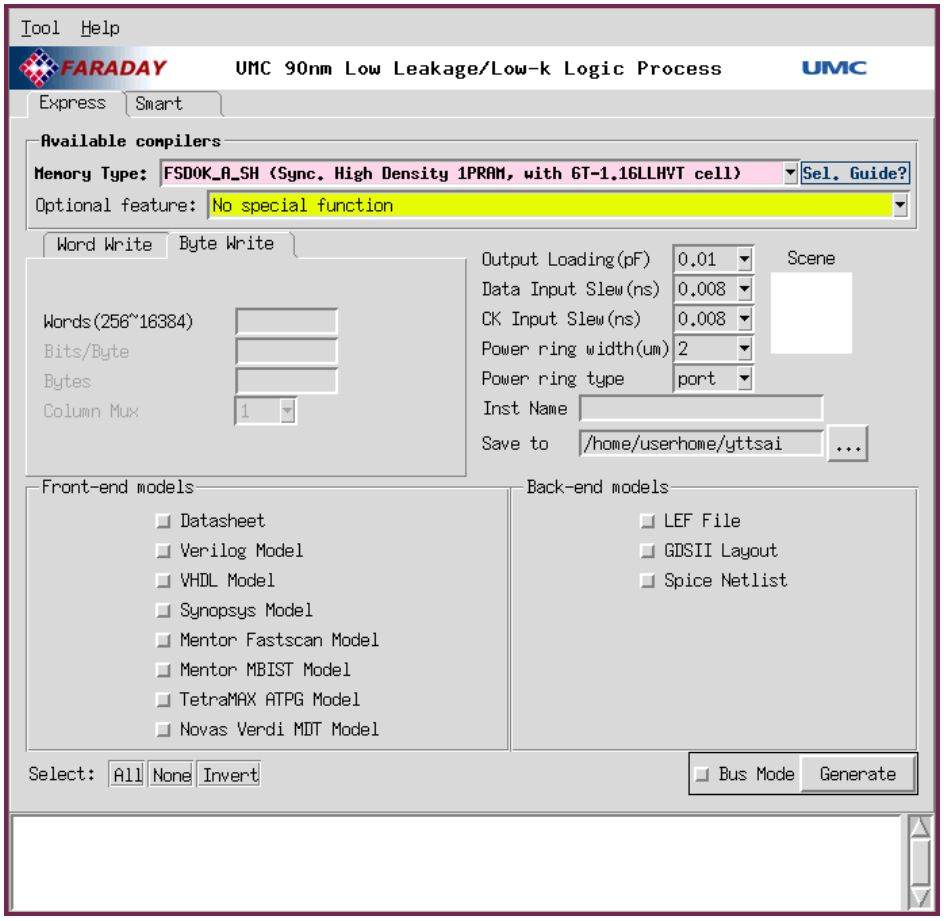

Figure 3-2. Memaker Express Mode in Byte Write

Figure 3-3. Completion Window

3.4.3.1 Express Mode in ROM Type

To enter the express mode in the ROM type, users should make sure that the ROM compiler of the "Memory Type" combo box is active. Click on the desired values in the words, bits, and Mux fields. Next, click on the "Code version" tab, the code version is one or two alphabets from "A" to "Z," or "A0" to "AF," "B0"to "BF,"…"Z0" to "ZF." This is to differentiate the ROM codes with the same configuration.

Click the selected "ROM Code" (The "ROM Code" panel allows users to upload the binary (Hexadecimal) ROM code file, or provides a dummy ROM code filled by '0'). If users choose the binary (Hexadecimal) ROM code, and upload the binary (Hexadecimal) ROM code from the selected directory, clicking the "Generate" button will generate the front-end and back-end models. If users select "filled by all 0," only the front-end models and the LEF file will be generated; the GDSII layout and the SPICE netlist will be ignored.

- ROM code file format

  The ROM code file is a text file format. The memory words are addressed from '0' to (Number of words) '-1', continuously (Please refer to Example 1), and the number of consecutive words cannot be more or less than the specified number of the words. The code file to be read must contain only the following:

  - **White space** (Spaces, new lines, tab, and form-feeds)
  - **Comments** (Both types of comments are allowed: //, /* */)
  - **Binary number** ('0', '1') (For binary ROM code format)
  - **Hexadecimal number** (0-9, A-F, and a-f) (For hexadecimal ROM code format)

  The underscore (_) can be used for readability purpose. For the data bit width less than the bit number, a '0' is filled towards MSB (Please refer to Example 2).

Example 1:

Code file of 128 words x 8 bits

```
/* DO7...DO0 */
11001111 // address0
10110011 // address1
11010111 // address2
00000000 // address3
    .
    .
00000000 // address125
00000000 // address126
10000100 // address127
```

Example 2:

```
1001011              1001011
100_111              0100111
0110011              0110011
1_111        =       0001111
1011111              1011111
111                  0000111
0101                 0000101
```

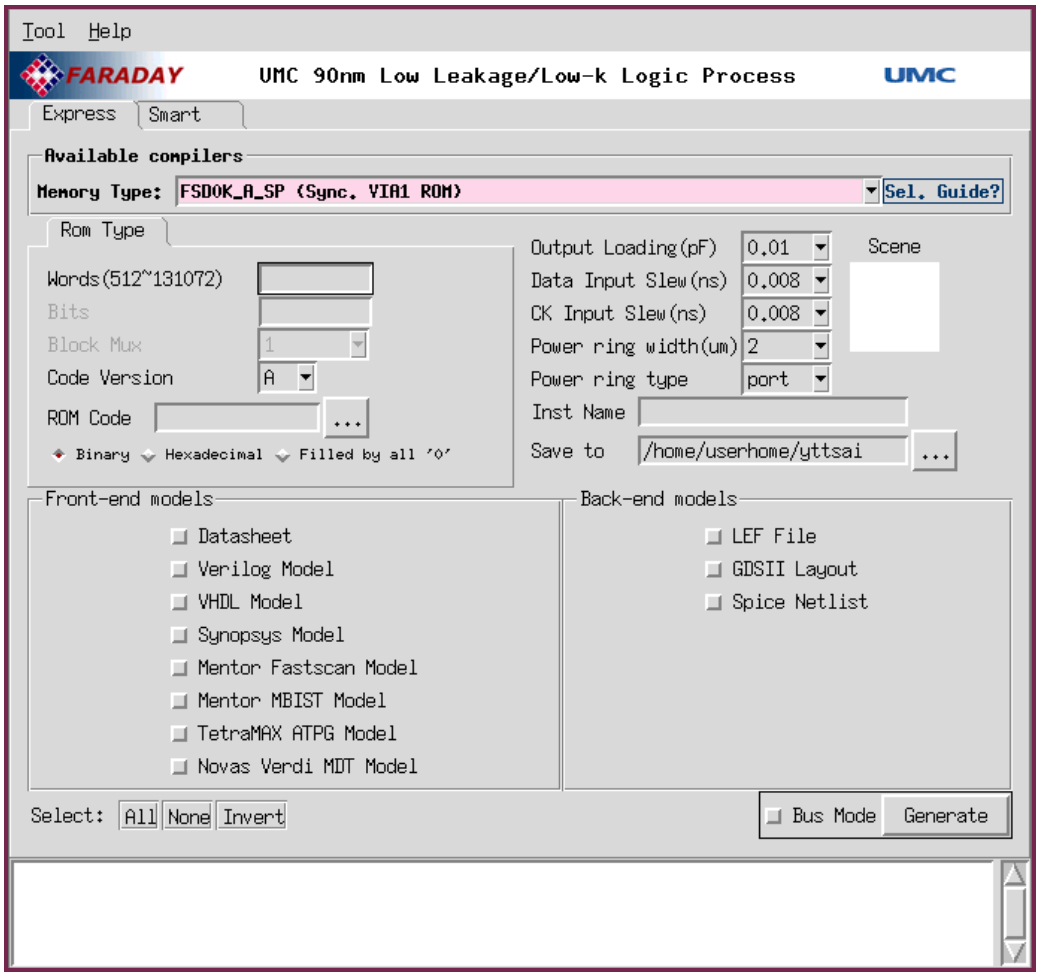

Figure 3-4. Memaker Express Mode in ROM Type

3.4.4 Smart Mode

An alternative way to enter the design data is to use the smart mode. This mode provides the comprehensive, step-by-step, instructions to enter the information. Most of the GUIs are the same as the ones in the express mode.

- Compiler types
  The "Compiler types" bar allows users to switch between the compilers that has been ported by Memaker.
- W/B write
  The "W/B write" bar allows users to choose between the word-write and byte-write memory types. The bar will reveal the "ROM Type" selection and hide "Word Write/Byte Write" in the ROM type memory.
- Design configuration
  The design configuration allows users to enter the design related configuration. It includes the words, bits, and bytes (If the byte-write is selected), and real-time prompts will guide users through the data entry procedures. Users may encounter three prompts: "Next >>", "<< Back", and "Go to Preview." The "Next" button will take users to the next field. The "<< Back" button allows you to re-enter the previous field. The "Go to Preview" button will only appear when users reach the end of the prompt. Clicking this button will take users to the preview window. For a detailed description of the preview window, please refer to Section 3.4.4.2.
- Operation panel
  The "Operation panel" provides users with the "Clear" button. The "Clear" button allows users to clear the values entered in the current field.
- Console window
  The status messages will be displayed on the window during the execution of Memaker.

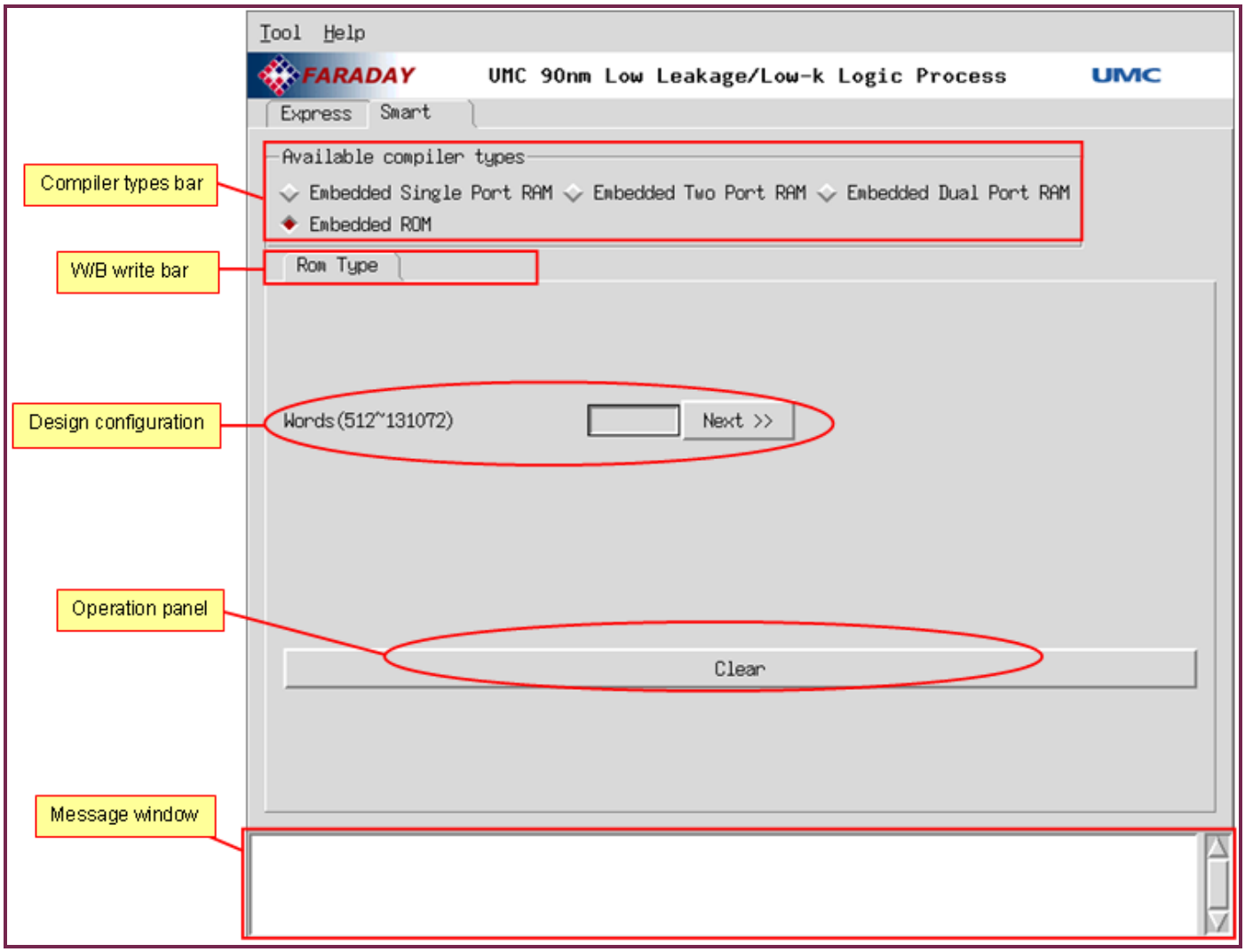

Figure 3-5. Memaker Smart Mode

3.4.4.1 Using Smart Mode

To use the smart mode, make sure that the "Smart" tab is active. If not, click the "Smart" page title to activate it. Users must select whether the memory is word-write or byte-write via the "W/B write bar." The following demonstration assumes that the word-write scenario has been chosen. When users enter a value in the "Design configuration" section and click the "Next" button, the program will verify the value and prompt the legal parameter range of the next entry. At the end of the entry, a "Go to Preview" button will appear. By clicking the button, a new configuration preview window will be displayed. Please refer to Figure 3-6 for the operation details.

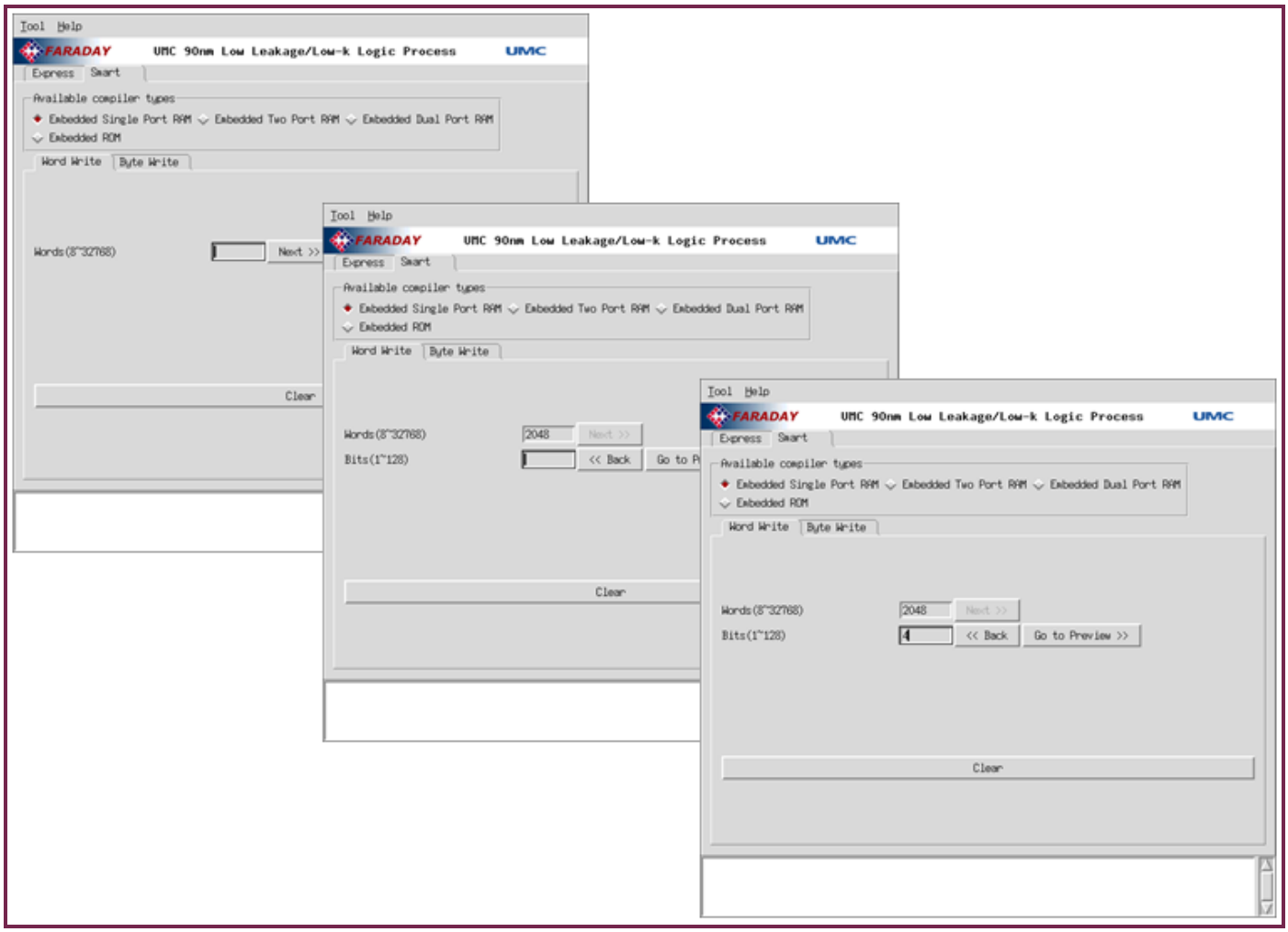

Figure 3-6. Word Write of Smart Mode

3.4.4.2 Preview GUI

Please refer to Figure 3-7 in regard to the following explanations:

- Memory type

  The "Memory type" shows all the candidate memory types that satisfy the initial configuration. It also provides the instance name specified for the preference.

- Design configuration

  The "design configuration" allows users to enter the design-related configurations, which include the output loading, data slew rate, power ring width, clock slew rate, ROM code version, upload and output files directory specified by the users. When the entry values are changed, the design attributes, such as the gate count and T_{aa}, will also be changed.

- Model selection

  The "model selection" lists a series of models that users can select for final outputs. It is divided into three categories: The front-end models, back-end models, and symbol models. In the COT release, if users are not licensed for the GDSII or SPICE netlist features, the respective check button will be grayed out. In the ASIC release, users will not see the check buttons of LEF, GDSII, and SPICE netlist.

- Operation panel

  The "operation panel" provides users with three convenient options: "Select all," "Unselect all," and "Invert". The "Select all" option allows users to select all models. The "Unselect all" option allows users to unselect all models. The "Invert" option allows users to invert the current selection so that all the models other than the one selected currently will be selected. Once the model is selected, users should click the "Generate" button to generate the models.

- Model generation panel

  The "model generation panel" provides users to generate (Bus mode) models by clicking the "Generate" button or cancel it by clicking the "Cancel" button.

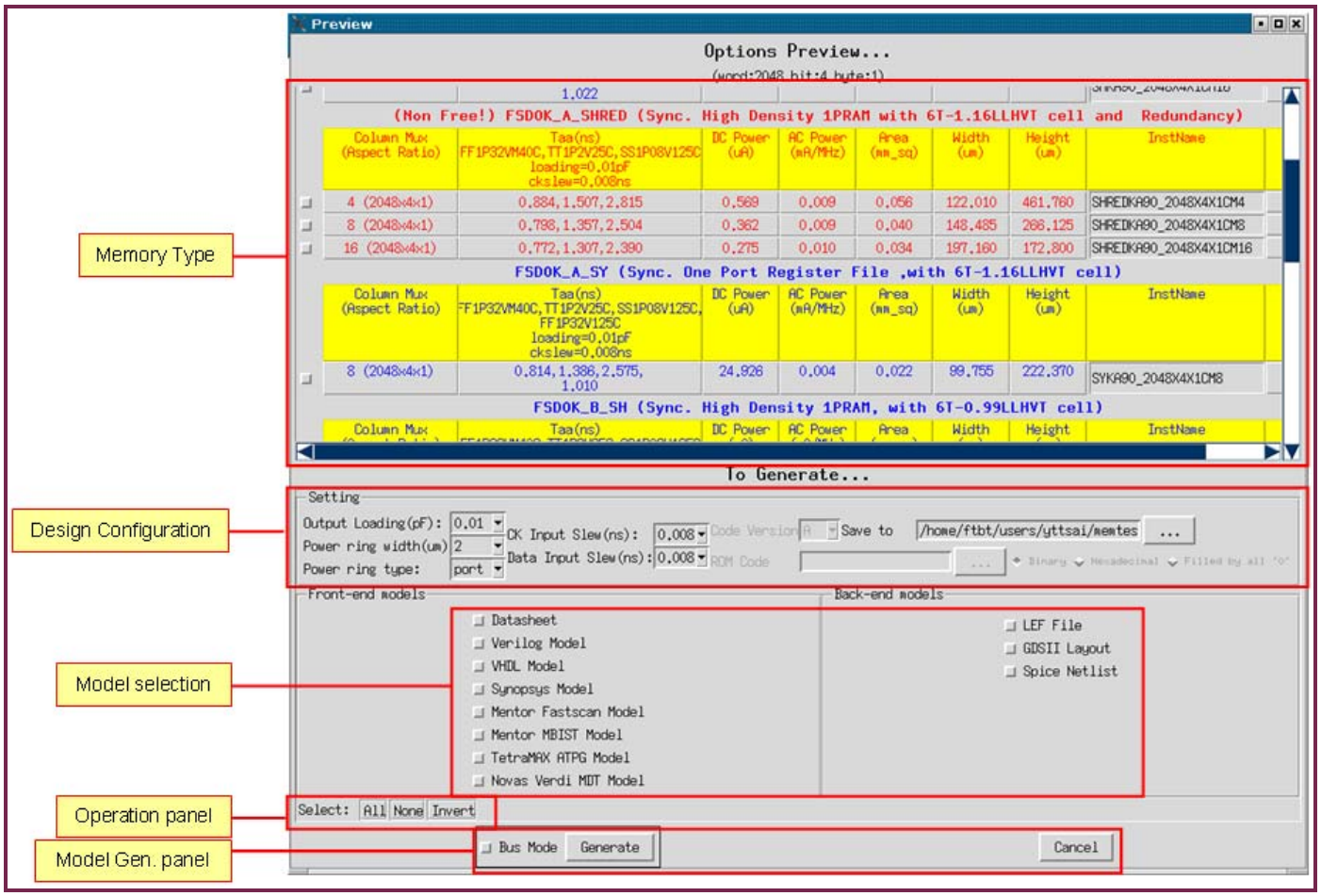

Figure 3-7. Preview Window

If the model generation is successful, a pop-up window, as shown in Figure 3-8, will appear. At the same time, an output file will also be generated in the current directory.

Figure 3-8. Completion Window

3.5 Memaker Batch Mode

Memaker provides the batch mode to generate models without executing GUI. This function is more efficient for advanced users. It includes the data sheet, Verilog model, VHDL model, Synopsys model, Mentor FastScan model, Mentor MBIST model, Synopsys TetraMAX model, Novas Verdi MDT model, GDSII layout, LEF file, and SPICE netlist. The usage is explained below:

- %memaker [arguments]
- For the COT package, the arguments are:

-s	tech	<Specify the technology>
-type	typename	<Memory type>
-mux	integer	<Specify the column/block multiplexer> (Aspect ratio)
-words	integer	<Number of words>
-bits	integer	<Number of bits>
-bytes	integer	<Number of bytes>
-load	real(pf)	<Output loading>
-datasr	real(ns)	<Input slew rate>
-cksr	real(ns)	<Clock slew rate>
-ring	(2|5|10)	<Specify the ring width of the instance>
-romcode	(file|All_0_patt)	<Specify the romcode file>
-codever	codever	<Specify the romcode version>
-instname	name	<Specify the instance name>
-dir	dirpath	<Assign the output directory>
-memlist	file	<Configuration list file> Debug option
-ful	file	<Instance name list file with the Faraday naming rule>
-rul		<Generate the instances by the Memaker rule file, memaker.rul>
-all		<Generate all supporting models>
-ds		<Data sheet model>
-ver		<Verilog behavior model>
-vhd		<VHDL behavior model>
-lib		<Synopsys Liberty model>
-mbist		<Mentor MBIST model>
-fscan		<Mentor FastScan model>

-tmax <TetraMAX ATPG model>
-mdt < Novas Verdi MDT model>
-siz <SPICE netlist>
-gds <GDSII layout>
-lef <LEF model>
-fullspeed (1|0) <Assign the full-speed mode in the MBIST model if possible>
(Default: 1)
-busmode <Generate models with bus format>
-lefmap <LEF layer mapping function>
-help <Display the help message>

- For the ASIC package, the arguments are:

  (a) For FG70A_A-FSA0F_A:

  -s fg70a_a-fsa0f_a <Specify the technology package, fg70a_a-fsa0f_a>
  -tech technology <Technology name>
  -type typename <Memory type>
  -mux integer <Specify the column/block multiplexer> (Aspect ratio)
  -words integer <Number of words>
  -bits integer <Number of bits>
  -bytes integer <Number of bytes>
  -load real(pf) <Output loading>
  -ring (0|10) <Power ring width, unit: μm>
  -romcode romcode <Romcode file>
  -topmetal integer <Assign the top metal>
  -name instName <Memory instance naming translator>
  -ful filename <Instance name list file with the Faraday naming rule>
  -rul <Generate the instances by the Memaker rule file, *memaker.rul*>
  -all <Generate all supporting models>
  -ds <Data sheet model>
  -ver <Verilog behavior model>
  -vhd <VHDL behavior model>
  -lib <Complete Synopsys Liberty model>
  -mbist <Mentor MBIST model>

-fscan <Mentor FastScan model>
-tmax <TetraMAX ATPG model>
-mdt < Novas Verdi MDT model>
-fullspeed (1|0) <Assign the full-speed mode in the MBIST model if possible>
(Default: 1)
-nofaraday <Do not add Faraday attributes into Liberty file>

(b) For others:

-s tech <Specify the technology>
-type typename <Memory type>
-mux integer <Specify the column/block multiplexer> (Aspect ratio)
-words integer <Number of words>
-bits integer <Number of bits>
-bytes integer <Number of bytes>
-load real(pf) <Output loading>
-datasr real(ns) <Input slew rate>
-cksr real(ns) <Clock slew rate>
-ring (2|5|10) <Specify the ring width of the instance>
-romcode (file|All_0_patt) <Specify the romcode file>
-codever codever <Specify the romcode version>
-dir dirpath <Assign the output directory>
-memlist file <Configuration list file> (Debug option)
-ful file <Instance name list file with the Faraday naming rule>
-rul <Generate the instances by the Memaker rule file, *memaker.rul*>
-all <Generate all supporting models>
-ds <Data sheet model>
-ver <Verilog behavior model>
-vhd <VHDL behavior model>
-lib <Synopsys Liberty model>
-mbist <Mentor MBIST model>
-fscan <Mentor FastScan model>
-tmax <TetraMAX ATPG model>
-mdt <Novas Verdi MDT model>

-fullspeed (1|0) <Assign the full-speed mode in the MBIST model if possible> (Default: 1)

-busmode <Generate models with bus format>

-help <Display the help message>

-nofaraday <Don't add Faraday attributes into Liberty file>

Example 1 (Generate a RAM model):

- %> memaker -s fsc0h_d -type SH -mux 2 -words 1024 -bits 10 -bytes 2 -load 0.04 -datasr 0.1 -cksr 0.1 -ver -vhd
- %> memaker -s fg70a_a-fsa0f_a -tech FSA0A_A -type SR -mux 4 -words 1024 -bits 10 -bytes 2 -loading 0.04 -ds -ver -vhd -lib

Example 2 (Generate a ROM model with all zero patterns):

- %> memaker -s fsb0g_a -type SC -words 512 -bits 8 -bytes 1 -mux 2 -romcode All_0_patt -ds

Example 3 (Generate a ROM model with a ROM code file):

- %> memaker -s fsb0g_a -type SC -words 512 -bits 8 -bytes 1 -mux 2 -romcode ./romCode –ds
- %> memaker -s fg70a_a-fsa0f_a -tech FSA0A_A -type SO -mux 2 -words 1024 -bits 10 -loading 0.04 -romcode ./romCode2048 -ds -ver -vhd -lib

Example 4 (Generate memory instances listed in the memlist file):

- %> memaker -s fsc0h_d -type SH -memlist SH.list -load 0.04 -datasr 0.1 -cksr 0.1 -all
- where the SH.list file format is as follows:

```
word <word> bit <bit> [bc]m <mux> byte <byte> [romcode <romcodeFile>] [rformat (bin|hex)]
```

"romcode" and "rformat" are used for ROM type only. If "rformat" is not specified, memaker would use "bin" as the default.

```
word 54016 bit 8 bm 8 byte 1 romcode ./rc54106x8
word 54272 bit 8 bm 16 byte 1 romcode ./rc54272x8 rformat hex
word 84480 bit 4 bm 16 byte 1
word 117248 bit 4 bm 16 byte 1
```

Example 5 (Generate memory instances by the naming translator):

- %> memaker -s fg70a_a-fsa0f_a -tech FSA0A_A -name SU00800AN02 -loading 0.04 -ds -ver -vhd -lib

Example 6 (Generate memory instances by following the Faraday naming rule):

- %> memaker -s fsc0h_d -ful name.ful -load 0.04 -datasr 0.1 -cksr 0.1 -ds -ver -vhd -syn

  where name.ful file format is as follows:

```
SH180_54016X8X1BM8
SH180_54016X8X1BM8
SP180_1024X8X1BM1
SP180_54016X8X1BM8 romcode code1
```

Example 7 (Generate memory instances by following the Memaker rule file):

- %> memaker -rul

  If the memaker.rul file is not found in the current working directory, a template file, memaker.rul, will be created first to invoke "%> memaker –rul". Users can then modify this file to generate the memory instances. If memaker.rul is in the current working directory, it will read memaker.rul and generate the specified memory instances

The memaker.rul file format is as follows:

```
# common line
set RUL(<keyword1>) “string”
set RUL(<keyword2>) digits
set RUL(<keyword3>) {listEntry1 listEntry2 …}
```

The first line is a common line. The other lines are command lines, which are the descriptions of <keyword1>, <keyword2>, and <keyword3>:

<keyword1>	Optional	Description
tech	No	Technology name, such as FSD0A_A or FSC0H_D
dir	Yes	Destination directory It can be any writable directory path.
ringType	Yes	Power ring type The available options are port and ppr.
<keyword2>	**Optional**	**Description**
datasr	Yes	Data input slew rate, depending on the memory type
cksr	Yes	Clock slew rate, depending on the memory type
load	Yes	Output loading, depending on the memory type
ringWidth	Yes	Power ring width The available options are 2, 5, and 10.
libFaraday	Yes	***ASIC package*:** Append the Faraday power-config attributes in the Synopsys Liberty file. The available options are '0' and '1.' If set to '0,' there is no Faraday power-config attributes in the Liberty file.
lefMap	No	***COT package*:** Translate all layer names used in the LEF models into the UMC layer naming format, such as metal1, metal2, via, and via2. These layer names are changed to ME1, ME2, VI1, and VI2.
bus	Yes	Bus mode model generation. The available options are 0 and 1.
<keyword3>	**Optional**	**Description**
instList	No	***COT package*:** **inst** **{inst -code romCodeFile}** **{inst -code romCodeFile -codever codeVer}** **{inst -name instName}** **{inst -name instName -code romCodeFile}** **{inst -name instName -code romCodeFile -codever codeVer}** ***ASIC package*:** **inst** **{inst -code romCodeFile}** **{inst -code romCodeFile -codever codeVer}** * The format of **inst** should follow the Faraday naming rule, such as SH180_54016X8X1BM8 * **romCodeFile** should be a readable file path * The format of **codeVer** is [A-Z] or [0-9][A-Z].,Such as A, B, 1A, or 2C * **instName** is the user-defined module name

<keyword1>	Optional	Description
modelList	No	**{ALL}** **{item1 item2 item3 …}** * **keyword** will instruct Memaker to generate all available models ***COT package***: * **Item** should be ds, ver, vhd, lib, fscan, mbist, tmax, mdt, gds, lef, or siz ***ASIC package***: * **Item** should be ds, ver, vhd, lib, fscan, mbist, tmax, or mdt

Users can obtain the detailed usage information by using the "memaker -help" command.

3.6 Memaker Input/Output Files

3.6.1 Input Files

No input file is required to exclude the Memaker rule generation; users only need to prepare the memory configuration information. On the other hand, to include the Memaker rule generation, an input file, *memaker.rul*, is needed.

3.6.2 Output Files

- *<cell_name>.ds*
  Data sheet file: This file contains information on the delay timing parameters, process metal options, input capacitance, output capacitance, power consumption, and area of a specific memory configuration.
- *<cell_name>.v*
  Verilog model file: This file will be generated when the "Verilog Model" in the "Model Button" section is marked.
- *<cell_name>.vhd*
  VHDL model file: This file is created when the "VHDL Model" check box is selected.
- *<cell_name>.lib*
  Synopsys dotlib model file: This file is created when the "Synopsys Model" checkbox is selected.

- *<cell_name>.mbist*

  Mentor Graphic MBIST model file: This file is created when the "Mentor MBIST Model" checkbox is selected.
- *<cell_name>.fastscan*

  Mentor Graphic FastScan model file: This file is created when the "Mentor Fastscan Model" checkbox is selected.
- *<cell_name>.tmax*

  Synopsys TetraMAX model file: This file is created when the "TetraMAX ATPG model" checkbox is selected.
- *<cell_name>.mdt*

  Novas Verdi model file: This is the MDT (Memory Definition Table) model file. This file is created when the "Novas Verdi MDT model" checkbox is selected.
- *<cell_name>.gds*

  GDSII file: This file is created when the "GDSII Layout" checkbox is selected.
- *<cell_name>.lef*

  LEF file: This file is created when the "LEF File" checkbox is selected.
- *<cell_name>.siz*

  SPICE netlist file: This file is created when the "Spice Netlist" checkbox is selected.

3.7 Memaker Error/Warning Message List

Error messages:

- (memItrl-1) No argument given in the Max function
- (memItrl-2) No argument given in the Min function
- (memItrl-3) Illegal format: %s and %s
- (memItrl-4) Bad Configuration in mux:%s, conf:%s.
- (memCPL-1) Unable to identify %s family.
- (memCPL-2) Illegal keyword %s at the line %s of the relation file, %s
- (memCPL-3) Encounter some errors in equation file %s. Message:\n%s\n
- (memCPL-4) Unable to find %s in equation file %s. Set to 0
- (memCPL-5) Unable to find %s timing. Please check!
- (memCPL-6) Unable to find any configure information in %s series memory.
- (memCPL-8) Doesn't support the Synopsys Liberty header in %s

- (memCPL-9) Can't find some corner settings. There are only: %s
- (memCPL-10) Can't find variable, %s. This is a fatal error.
- (memCPL-11) Can't find variable, %s. User should not generate the layout model by Memaker!
- (memCPL-12) Double booking: Define %s and %s at the same time!
- (memCPL-13) Can't find width %s setting in %s
- (memCPL-14) Can't find %s in memory database.
- (memCPL-15) %s Area definition is not equal to area data.
- (memCPL-16) Encounter some errors while trying to evaluate the relation file. Message:\n%s\n
- (memCPL-17) Encounter some errors while trying to get corner data:%s
- (memTPL-1) Encounter some errors in %s template file %s. Message:\n%s\n
- (memGNL-1) Sorry! Generation is not done!
- (memGNL-2) The specified directory, %s, doesn't have permission to write.
- (memGNL-3) %s doesn't exist!
- (memGNL-4) Wrong memory list: %s
- (memGNL-5) Mux is out of range!
- (memGNL-6) Word is out of range!
- (memGNL-7) Bit is out of range!
- (memGNL-8) Byte is out of range!
- (memGNL-9) Unable to find the ROM code file: %s
- (memGNL-10) The file size of the ROM code file is 0: %s
- (memGNL-11) Encounter some errors while parsing the ROM code file. Message:\n%s
- (memGNL-12) Instance generation is fail: %s
- (memGNL-13) Instance name, %s, is illegal!
- (memGNL-14) No such memory type, %s, in this technology!
- (memGNL-15) The prefix, %s, is wrong.
- (memGNL-16) Mux type, %s, is wrong.
- (memGNL-17) Output loading, %s, is out of range!
- (memGNL-18) Input slew rate, %s, is out of range!
- (memGNL-19) Clock slew rate, %s, is out of range!
- (memGNL-20) Power ring width, %s, is out of range!
- (memGNL-21) Please select memory models.
- (memGNL-22) Please specify memory type!
- (memGNL-23) Code version must \[A-Z\] or \[A-Z\]\[A-Z0-9\].

- (memGNL-24) Please specify words!
- (memGNL-25) Please specify bits!
- (memGNL-26) Please specify mux!
- (memGNL-27) Please specify bytes!
- (memGNL-28) Please specify romcode file!
- (memGNL-29) Can't find RUL(tech) in memaker.rul
- (memGNL-30) Can't find RUL(modelList) in memaker.rul.
- (memGNL-31) %s should be %s.
- (memGNL-32) ASIC package doesn't support renaming feature.
- (memGNL-33) %s is illegal option.
- (memGNL-34) Unable to find any available memory compilers.
- (memGNL-35) Word, Bit or Byte is out of range!
- (memGNL-36) Please select memory instances.
- (memGNL-37) Encounter some errors while generating symbol file. Message:\n%s\n
- (memGNL-38) Can not find IP tag information.
- (memGNL-39) Encounter some errors while generating %s model. Message:\n%s\n
- (memGNL-40) Can't parse the metric item from watermark information.
- (memGNL-41) Can't parse the vendor item from watermark information
- (memGNL-42) There is no any H-cell information in memory database.
- (memGNL-43) Encounter some errors while querying MOSCAP information. Message:\n%s\n
- (memGNL-44) Unable to create directory, %s
- (memGNL-45) %s is not a directory.
- (memGNL-46) Unable to find the acroread utility in the environment!
- (memGNL-47) Unable to find %s in the installing directory.
- (memGNL-48) The output loading of %s, %s pF, is out of range.
- (memGNL-49) Illegal argument %s
- (memGNL-50) The metal option %s and the top ring type %s are conflict.
- (memGNL-51) The pyramidal ring model doesn't support the specified type, %s.
- (memGNL-52) This option, %s, is not supported anymore. Please specify the option, %s
- (memGNL-53) Doesn't support the ring type %s.
- (memGNL-56) %s doesn't support the bus mode.
- (memGNL-57) Doesn't support the ROM code format %s.
- (memGNL-58) RUL(bus) should be 0 or 1.

- (memABT-1) Can't find %s definition. This is a fatal error. Please contact Faraday to get a hotfix version!

Warning messages:

- (memWarn-1) All 0 patterns don't support the GDSII layout and the SPICE netlist.
- (memWarn-2) Doesn't support the ring width %s. Set the power ring width to %s
- (memWarn-3) Doesn't specify loading. Using the default value: %s
- (memWarn-4) Doesn't specify data slew rate. Using default value: %s
- (memWarn-5) Doesn't specify the clock slew rate. Using the default value: %s
- (memWarn-6) Doesn't specify the ring width. Using the default value: %s
- (memWarn-7) ASIC package doesn't support the back-end models.
- (memWarn-8) You need to specify a romcode file before generating the LEF model, layout, or SPICE netlist file.
- (memWarn-11) Current is set to 0 in %s file.
- (memWarn-12) Doesn't support the ring type %s. Set the power ring type to %s
- (memWarn-13) %s doesn't support the TetraMAX ATPG Model
- (memWarn-14) %s doesn't support the Novas Verdi MDT Model
- (memWarn-15) Doesn't specify the bus mode. Using the default value: %s
- (memWarn-16) Doesn't support the full-speed mode of the MBIST model