TetraMAX® ATPG
Quick Reference

Version A-2007.12, December 2007

Comments?
Send comments on the documentation by going to http://solvnet.synopsys.com, then clicking "Enter a Call to the Support Center."

SYNOPSYS®

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

> "This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____."

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in the U.S.A.

Document Order Number: 37575-000 F1
TetraMAX ATPG Quick Reference, version Z-2007.03

Contents

Conventions

This quick reference uses the following conventions in the command syntax:

- Brackets ([ ]) denote optional choices. A command is still valid if you omit these choices.
- A vertical bar (|) separate different items from which you choose a single item.
- An asterisk (*) indicates the default setting in a list of choices.
- Angle brackets (< >) denote a grouping of related items.
- Ellipses (...) indicate that the previous item in the syntax can be repeated one or more times.
- *Italics* indicate a user-defined item for which you substitute a value or character string.
- Uppercase letters shown in the command syntax represent the minimum characters necessary to enter the command. For example, to enter the command **`REMove CApture Masks -All`**, it is sufficient to type **`REM CA M -A`**.

Online Help

TetraMAX provides online help in the following forms:

- Text-only help on TetraMAX commands, displayed in the transcript window. Use the help command in the command-line window.
- Window-based, hyperlinked help on commands, design flows, error messages, design rules, fault classes, and many other topics. Use the pull-down menu command Help > Help Topics, or use the **man** command and specify the topic of interest.
- The *TetraMAX ATPG User Guide*, available in online form. Use the pull-down menu command Help > User's Guide.
- The *TetraMAX Release Notes*, available in online form. Use the pull-down menu command Help > Release Notes.

For text-only help, use the **help** command. For a list of help topics available, type the following command in the command-line window:

```
> help
```

To display a list of all commands and a summary of their supported syntax, type the following command in the command-line window:

```
> help -usage
```

To display the syntax of a specific command, type the following command in the command-line window:

```
> help command_name
```

where *command_name* is the command whose syntax you want to view.

TetraMAX ATPG Commands

The TetraMAX ATPG commands are described in alphabetical order. An index to the commands is provided at the end of this manual.

Add ATPG Constraints

```
Add Atpg Constraints <name>  <0 | 1 | Z>  [-
Module <module_name>]  <atpg_gate_name |
gate_id |  module_pinname |  pin_pathname>
[-Drc | -Iddq]
```

This command defines constraints on nets that must be satisfied during pattern generation. In the command, specify a name to identify the constraint, the constraint value (0, 1, or Z), and the place in the design to apply the constraint.

Add ATPG Primitives

```
Add Atpg Primitives <name>  <And | Or | SEL1
| SEL01 | Equiv>  [-Module <module_name>]
[input_connection_list ]
```

This command creates a primitive that is added to the design and has its inputs connected to specified nets. When you constrain the output of the added primitive, it forces the pattern generation algorithm to conform to specified logical conditions at the connection points. In the command, specify a name for the added primitive, its logical function, and its input connections.

Add Capture Masks

```
Add CApture Masks <instance_names |
gate_ids>
```

This command masks off any potential value captured into a specified scan or nonscan sequential cell. The cell is treated as though the data input were connected to a TieX gate, which means that scan cells with capture masks might still have non-X values in a pattern.

Add Cell Constraints

```
Add CEll Constraints
<[0 | 1 | X | OX | XX]>
<[chain_name | instance_name> [-POSition
<cell_pos1 | SCI [cell_pos2 | SCI]> |
  -All]
```

This command specifies the load and unload values allowed for specified scan cells. The test pattern generator creates patterns that always satisfy these cell constraints. In the command, specify the type of constraint (end-of-load value or observe-X type constraint) and the cells where you want the constraint applied. You can specify one cell, a range of cells, or all cells in a scan chain; or a specific latch or flip-flop instance.

Add Clocks

```
Add CLocks <0 | 1>  <pin_names_list>
[-Timing <<period> <LE> <TE>
<measure_time>>]
[-Unit <ps | ns>]
[-Shift] [-Ref_Timing <<period> <LE> <TE>>]
[-PLLclock]  [-INTclock]  [-REFclock]
[-PLL_Source <node_name>]
[-Cycle <<cycle_id> <-Always <ON | OFf> |
int_node_name <0 | 1>>>]
```

This command defines a primary input port to be a clock. For ATPG purposes, a clock is a pulsed input port that can change the state of a flip-flop,

latch, or RAM. This includes any ordinary active-high or active-low clock, asynchronous set/reset signal, or RAM write signal.

In the command, specify the off state of the clock (0 or 1) and one or more port names. You can optionally specify the clock timing characteristics: period, leading-edge time, trailing-edge time, and measure time. You can also specify that the clock is a shift clock, which is used to shift data through a scan chain.

Add Delay Paths

```
Add DElay Paths filename
```

This command reads in a delay path list from a specified file.

Add Display Gates

```
Add DIsplay Gates
<pin_pathname | gate_id | -All>
```

This command adds gates associated with the pin_pathname, the gate ID, or all gates to the GSV.

Add Distributed Processors

```
Add Distributed Processors [-32bit]
[-Options <name>]  [-Nslaves <d>]
[-Min_slaves <d>]  <hostnames |  -Lsf
<bsub_exec_name> | -Grd <bsub_exec_name> |
-GEneric <generic_exec_name>>
```

This command identifies the slave processors sharing a distributed fault simulation or test generation task.

Add Equivalent Nofaults

```
Add Equivalent Nofaults
```

This command propagates the existing list of nofaults (created with the **`add nofaults`** command) to their equivalent fault sites and adds these equivalent sites to the nofault list. The

existing list includes user-defined nofaults as well as netlist modules that have nofault attributes on module pins or internal gate pins.

Add Faults

```
Add Faults
[instance | pin_pathname |  -Module <module
name> |  -All |  -CLocks | -SCan_enable | -
bridge_location <bridge_location1
bridge_location2> |  -Node_file <filename>]
[-LAunch <launch_clock>]  [-CApture
<capture_clock>] [-EXclusive]  [-SHared] [-
INTER_clock_domain] [-INTRA_clock_domain]
[-Stuck <0|1|01> |  -Slow <R|F|RF>] [-Bridge
<0 | 1 | 01>] [-AGgressor_node <First |
Second | Both>]  [-MAx_invalid_report <d>]
```

This command creates a list of faults for test generation or fault simulation. Potential fault sites include all top-level ports and all input and output pins of cells that have a netlist-defined pin name. You can add faults to the pins of a specified instance, to a single pin, to pins of all instances of a specified module, or to all potential fault sites in the design.

Add Net Connections

```
Add Net Connections <net_type>  [net_names |
pin_pathnames]  [-port <port_names>] [-
Module <module_names> |  -All]  [-Disconnect
|  -Remove] [-Buf]  [-Inv]
```

This command changes the behavior of specified nets in the netlist. TetraMAX considers these changes the same as netlist data when it performs fault simulation and test generation.

By default, these changes affect the in-memory image of the design and modules, thereby affecting patterns and netlists written by TetraMAX.

Add Nofaults

```
Add Nofaults [instance | pin_pathname | -
Module <list of module name and its pin
names>]  [-Stuck <0|1|01>] [-Instance <list
of intance name and its pin names>]
```

This command adds the nofault attribute to specified locations in the design, thus preventing placement of faults at those locations. In order to be effective, you must use this command before you use the **`add faults`** or **`read faults`** command.

You can add the nofault attribute to all pins associated with a specific instance, to a specific pin, or to one or more pins of every instance of a specified module.

Add PI Constraints

```
Add PI Constraints <0 | 1 | X | Z>
<pin_names |  -All |  -ALL_Bidis | -
ALL_BIDIS_Except_clocks |  -
ALL_Except_clocks>
```

This command constrains one or more primary input ports to a specified value (0, 1, X, or Z). The ATPG algorithm only generates patterns that satisfy the defined constraints on these ports. An alternative to using this command is to define the constraints in the STIL procedure file.

This command does not constrain the ports during scan chain loading or unloading. To control the port values at these times, set the desired values in the STIL procedure file.

Add PI Equivalences

```
Add PI Equivalences <<reference port name | constrained port name>>
[-Invert <name, <reference port name | constrained port name>> |
-Differential <name, <reference port name | constrained port name>> |
-Same_polarity <name, <reference port name | constrained port name>>]
```

This command defines an equivalence or inverse relationship between one or more pairs of primary inputs. The ATPG algorithm only generates patterns that satisfy the defined logical relationships between these pairs of input ports.

This command does not constrain the ports during scan chain loading or unloading. To control the port values at these times, set the desired values in the STIL procedure file.

Add PO Masks

```
Add PO Masks <-All |  pin_name>
```

This command specifies a primary output port (or bidirectional port) whose output is to be masked. The test generator and fault simulator do not give fault detection credit for these ports, and the expected value is set to X. Masking can be useful when there are circuit outputs that might not be observable in a particular tester or system environment, such as boundary scan.

The PO mask does not mask data from scan chain outputs during scan-shift cycles.

Add Scan Chains

```
Add Scan Chains
chain_name input_pin output_pin
```

This command defines a scan chain in the absence of a DRC file. You can use this command before the **write drc_file**

command to create a template STIL procedure file or before the **run drc** command to specify the scan chains.

Scan chains defined by this command are ignored if you specify a DRC file with the **run drc** command or **set drc** command.

Add Scan Enables

```
Add Scan Enables 0 | 1 | Z pin_name
```

This command defines a pin that must be initialized to a specific value for scan shifting. You can use this command before the **write drc_file** command to create a STIL procedure file, or before the **run drc** command to specify the conditions for scan shifting.

Add Slow Bidis

```
Add Slow Bidis <port_name | -All>
```

This command modifies the associated BUS primitives to output an X if any TSD or SW primitives are not driving a Z onto the BUS primitive.

Add Slow Cells

```
Add Slow Cells <instance_names | gate_id>
```

This command modifies the simulation behavior of scan DFF or DLAT cells such that any transition launched from those cells is simulated as an X.

Add Slow Path

```
Add SLow Path [-from <from,from_pin_name>]
[-through <pin_name>] [-to <to,to_pin_name>]
[-MAX_fails <d>]
```

The **add slow path** command defines timing exceptions in a way that is similar to the way that they are defined in PrimeTime.

You can use the **add slow path** command to define timing exceptions the way you define them in PrimeTime. Note: The **add slow path** command is supported for Basic Scan and Fast Sequential engines only. There is no Slow Path support in Full Sequential, including path delay ATPG.

Add Waveform Signals

```
Add Waveform Signals [pin_pathname | gate id
| net_names | -All]
```

This command adds waveforms associated with the pin_pathname, gate ID, or net_name, or all gates to the simulation waveform viewer.

If you enter a net name on the command line, the simulation waveform viewer keeps the net name as the signal name. If you pick a net from the GSV and add it to the simulation waveform viewer, the net name is the signal name. The **remove waveform signals** command must have the same options as the **add waveform signals** command.

Alias

```
ALias [alias_name [alias_text]
```

This command defines an alternative symbolic name for a command, along with potential command arguments and switches. The alias can then be used in place of the original command.

Analyze Buses

```
ANalyze Buses
gate_id ... | -All
[-Exclusive* [First* | All] |
 -Prevention | -Zstate ] [-Update]
```

This command analyzes a specified bus or all buses (**BUS** primitives) to determine whether each bus can be placed into a contention state, contention-free state, or Z state.

In the command, specify the type of bus analysis to perform: exclusive (a check for mutually exclusive control circuitry), prevention (simultaneous prevention of contention on all **BUS** devices), or Z state (a check for the possibility of a floating net). You can also specify whether to update the DRC results with the bus analysis.

Analyze Compressors

```
ANalyze COmpressors <-NUM_Chains <d>>  <-
NUM_Inputs <d>>  <-NUM_SCANOuts <d>> [-
NODiag] [-Verbose]  [-XTOLerance <Default |
High>]
```

This command enables you to run ATPG by creating a virtual compressor. You can select parameters for the compressor and the decompressor and run ATPG. Use the analyze compressors command to perform a what-if analysis and explore the benefit of scan-compression on your design. This command deletes the internal patterns after an analyze compressors run. You must run ATPG in scan mode separately and compare results to the scan mode.

The analyze compressors command prints out estimated data volume and test volume reductions you can achieve. These estimates assumes that you have exactly same coverage

and same pattern count in regular scan atpg. You also get an estimate of number of gates per chain required to create the compressor's you have chosen

You can run this analysis only after you run regular scan DRC. You cannot run it after scan-compression mode DRC.

Analyze Faults

```
ANalyze Faults <pin_pathname |
delay_pathname | -Class <list of
fault_classes>> [-stuck <0|1>] [-Observe
<d>] [-Display] [-Slow <R|F>]
[-FAUlt_simulation] [-FASt_sequential |
-FUll_sequential]  [-Max <d>]
[-TRAnsition_clocking]  [-Verbose]
[bridge_location]  [-Bridge <0 | 1>]
```

This command analyzes a specific fault, or one or more classes of faults, to help determine why the faults are not detected. The analysis is affected by the **`set atpg`**, **`set delay`**, and **`set contention`** command settings.

In the command, specify either a single fault at a specific location, an entire class of faults (such as NO or AN), or multiple classes of faults for analysis. For a single stuck-at fault, you can optionally specify a particular observe point and whether to display the results in the schematic viewer.

Analyze Feedback Path

```
ANalyze Feedback Path loop_id [-Verbose]
```

This command determines whether a specified feedback network has a sensitizable feedback path. If TetraMAX can create a pattern to satisfy a sensitization condition, you can display the simulation values for that pattern by using the **`set pindata pattern 0`** command.

To obtain a list of feedback path loop ID numbers, use the **report feedback paths -all** command.

Analyze Simulation Data

```
ANalyze Simulation Data [file_name]
[-FAst_sequential <d>]  [-FUll_sequential
<pat#1 pat#2>]
[-Top <top_instance_name>]
[-Sensitive | -INSensitive]
[-Bidi_events] [-MAX_Values_reported <d>]
[-MAX_Fails_reported <d>]  [-Diag]
```

This command compares the results of two different simulations or displays the results from a single simulation. In the command, specify either one or two of the following possible sources of simulation data: a VCD (Value Change Dump) simulation file, a set of patterns generated with the Fast-Sequential simulator, or a set of patterns generated with the Full-Sequential simulator.

If the two sources of simulation data are patterns from Fast-Sequential and Full-Sequential simulations, you can optionally perform an automated diagnosis of the simulation differences.

Analyze Violation

```
ANalyze Violation violation_id
```

This command analyzes the cause of a rule violation detected by the **run drc** command (for example, “Z1-4”) and generates a report on that violation. The gates associated with the violation are displayed in the schematic viewer, annotated with data that was used to perform rule checking.

Analyze Wires

```
ANalyze Wires <-All |  gate_id>  [-Update]
```

This command analyzes potential contention problems associated with WIRE primitives.

Build

```
BUILD [-Force]
```

This command manually returns a TetraMAX session to the BUILD command mode, allowing you to execute commands that operate only in that mode.

Cat

```
CAT filename [-Max n] [-Line_numbers]
```

This command displays the contents of a specified file, like the UNIX **cat** command. You can optionally limit the number of lines displayed or show line numbers along with the lines of text.

Cd

```
CD directory_path
```

This command changes the current working directory, like the UNIX **cd** command. It works only in the GUI mode of TetraMAX (not in the shell mode).

Clear

```
CLEAR
```

This command clears the text from the transcript window.

Cp

```
CP source_filename destination_filename
```

This command copies a file, like the UNIX “cp” command. It does not support wildcards (*) in the specified file names.

DRC

```
DRC [-Force]
```

This command manually returns a TetraMAX session to the DRC command mode, allowing you to execute commands that operate only in that mode.

Exit

```
EXIt [-Force]
```

This command ends the current TetraMAX session and returns you to the Windows or shell environment.

Get Licences

```
Get Licenses
```

```
<TEST-ANalysis | TEST-Faultsim | TEST-ATPG-Max | TEST-DIagnosis | TEST-Map | TEST-Fault-max | TEST-Accelerate-max> [-Wait <d>]
```

Use this command to manually check out selected licenses. Normally, this is not required because licenses are automatically checked out as needed.

The -Wait option helps the distributed ATPG to check out multiple copies of the ATPG license if distributed processing licenses are not available.This option tries to get a license multiple times until d seconds have passed.

d is an integer in seconds. It does not have a default value. If d < 60, the ATPG retries every 60 seconds until the time left is less than 60 seconds. It then retries every second before it times out.

Gui_Start

```
gui_start
```

This command displays the TetraMAX GUI in its current state in the shell mode.

Gui_Stop

```
gui_stop
```

This command stops the TetraMAX GUI session and reverts to the TetraMAX shell command prompt If you did not use the **gui_start** command to start the GUI, the **gui_start** command exits the TetraMAX application.

You can also use the **gui_start** command from the pulldown menu File > Exit GUI. If you use the **gui_stop** command before invoking TetraMAX using the **gui_start** command, TetraMAX displays an error message.

GSV GET Selected Cells

```
GSVGET SELected CEll s
```

When the GSV is active, this command returns the instance name of the gate(s) selected. If the selected gates are displayed in the design view, it outputs a single line that shows the instance names of the gates, separated by spaces. If any of the selected gates are at the primitive level, it displays the gateid in a second line. As a single library cell contains several primitives, the cell's instance name might be repeated for each primitive object. There is no correpondence between the order of instance names in the first line and the gateids in the second line.

This command is similar to the **get_cells -selected** feature in Tcl mode. You can use it to identify the gates selected in the GSV.

GSV Print

```
GSV Print
```

This command creates a grayscale PostScript file, which captures the schematic displayed in the Graphical Schematic Viewer (GSV).

You can add the GSV Print command into the TetraMAX scripts and capture schematic output automatically. The computman_writing host must have PostScript drivers installed (usually with lpr/lp). You can enclose the arguments in double quotes (“ “).

Help

```
Help [command_name | -All] [-Usage]
```

This command lists all of the available commands or reports the supported command switches and arguments for a specific command. For a restricted list of commands, use a wildcard character (for example, **help report***).

For a more detailed description of a specific command, use **man** *command_name*.

Ls

```
LS [-L] [pathname]
```

This command displays information about a specified file or directory, like the UNIX **ls** command. By default, it lists the names of the files in the current working directory.

Man

```
Man [command | rule_id | rule_violation_id |
  message_id | COMmands | GETting_started |
  FAUlt_classes | STIL_examples |
  RAM_examples]
```

This command invokes the TetraMAX online help and displays information on a specified command, rule ID, violation ID, message ID, or topic. For example: **man read netlist, man Z9, man C17-1, man fault_classes**. The **man** command by itself returns a list of all commands.

Mkdir

```
MKDIR pathname
```

This command creates a new directory, like the UNIX “mkdir” command.

Mv

```
MV source_filename destination_filename
```

This command moves or renames a file, like the UNIX **mv** command.

Pwd

```
PWD
```

This command displays the current working directory, like the UNIX **pwd** command.

Quit

```
QUIt [-Force]
```

This command terminates the current session and returns you to the Windows or shell environment. You can also use the **exit** command to terminate the current session.

Read DRC

```
Read Drc <filename>
```

This command, used in conjunction with the **update wft** and **update clock** commands, reads the specified SPF file, but will not further process it through DRC. It must be specified immediately before the **update wft** and **update clock** commands. Any changes made with the update commands will only be saved, and usable, if the **write drc** command is used immeditely after these commands, and then run drc is used with the newly generated file.

Read Faults

```
REAd Faults <file_name> [-Retain_code | -
Maintain_detection | -Force_retain_code] [-
Add | -Delete]
```

This command reads a list of faults from a file and adds those faults to (or removes them from) the current fault list. If you are adding new faults, you can optionally retain the fault classification codes provided in the fault file for each fault location.

Read Image

```
REAd Image file_name
```

This command reads and restores the TetraMAX database saved by a **`write image`** command.

Read Layout

```
REAd LAyout <lib_name>  <-Cell_name
<cell_name>>  [-Version <d>]
[-MAX_mismatch_report <d>]
```

This command reads the Milkyway database (Synopsys physical database) and displays the corresponding physical data for the resulting fault candidates in its diagnosis output.

Milkyway database consists of a library that holds all the cell views of the design. There is a cell in the library matching the top-level module in TetraMAX. The library cells in TetraMAX also have a corresponding cell in Milkyway. TetraMAX can identify the missing correspondences. Physical data for corresponding objects is stored for each TetraMAX primitive.

This command sends any messages from the Milkyway database parser to the standard output of the TetraMAX process. These messages are

related to the physical implementation and might not be useful for TetraMAX users. TetraMAX does not store the output in its log file.

Read Memory_file

```
REAd Memory_file
<gate_id | instance_name>
filename [-Binary | -Hex*]
[-Range first_address last_address]
```

This command loads a specified RAM or ROM instance with a memory image obtained from a data file in Verilog format. You can use this command to alter the RAM or ROM contents specified in the netlist model.

Read Netlist

```
REAd NEtlist
filename
[-Format Edif | VErilog | VHdl]
[-Library] [-Master_modules]
[-Sensitive | -INSensitive]
[-Delete] [-Noabort]
[-Verbose]
```

This command reads a design netlist. TetraMAX can automatically identify and accept any combination of netlists in Verilog, EDIF, and VHDL formats; and in either ASCII or compressed (GZIP or binary) form.

Typically, you only need to specify the file name to successfully read in a netlist file. However, options are available to control the interpretation of the netlist data. You can read in multiple files with a single command by using a wildcard character (for example, **`read netlist mux1*`**).

Read Nofaults

```
REAd NOfaults filename
```

This command adds the nofault attribute to the list of locations specified in a file, thus preventing the placement of faults at those locations.

read_sdc

```
read_sdc file_name[-echo][-syntax_only]
-version sdc_version]
```

This command reads in setup timing exceptions directly from an SDC (Synopsys Design Constraints) file. You must be in Tcl and DRC mode to use this command (after you successfully run the **run_build** command, but before running **run_drc**).

Read Slow Path

```
REAd Slow Path <file_name>  -delete
```

This command reads a file of **add slow path** commands and processes them.

Read Timing

```
Read Timing <filename>  [-Delete]
```

This command causes TetraMAX to read a file of data that will be interpreted as the minimum slacks (timing margins) for each faultable pin in the design. This minimum slack information can be printed from PrimeTime using a procedure defined in the pt2tmax Tcl script provided within the TetraMAX release.

The format for the input file is a single line for each faultable pin. The line contains three fields: the pinpathname for the pin, the minimum slack for a rising transition through that pin, and the minimum slack for a falling transition through the pin.

Refresh Schematic

```
REFresh Schematic
```

This command redraws the diagram in the graphical schematic viewer (GSV). You can use this command to update the schematic display labels after you use the **set pindata** command.

Remove ATPG Constraints

```
REMove Atpg Constraints
atpg_constraint_name | -All
```

This command removes a specified ATPG constraint, or all ATPG constraints, previously added with the **add atpg constraints** command.

Remove ATPG Primitives

```
REMove Atpg Primitives
atpg_primitive_name | atpg_primitive_id |
-All
```

This command removes a specified ATPG primitive, or all ATPG primitives, previously added with the **add atpg primitives** command.

Remove Capture Masks

```
REMove CApture Masks
instance_names | gate_id | -All
```

This command removes a specified scan or nonscan cell capture mask, or all such capture masks, previously added with the **add capture masks** command.

Remove Cell Constraints

```
REMove CEll Constraints <chain_name | instance_name | -All> [-POSition <cell_pos1 | SCI [cell_pos2 | SCI]>]
```

This command removes a specified cell constraint, or all cell constraints, previously added with the **add cell constraints** command.

Remove Clocks

```
REMove CLocks <pin_names | -All>
```

This command removes one or more clock declarations, or all clock declarations, previously added with the **add clocks** command or implicitly defined in the STIL procedure file and implemented with the **run drc** command.

Remove Compressors

```
REMove COmpressors <compressor_name | -Load | -Unload | -All>
```

This command removes compressors from the defined compresors list. Note that this command is valid in DRC mode only.

Remove Delay Paths

```
REMove DElay Paths [-All | path_name] [-MIN_Cycle <d> | -MAX_Cycle <d>] [-MIN_Slack <d> | -MAX_Slack <d>]
```

This command removes delay path definitions from TetraMAX memory, thus excluding them from being used for fault seeding and ATPG.

Remove Display Gates

```
REMove Display Gates <pin_pathname | gate_id | -All>
```

This command removes gates associated with the pin_pathname, the gate ID, or all gates from the GSV.

Remove Distributed Processors

```
REMove Distributed Processors
<hostnames | -Lsf | -Grd | -All>
```

This command removes machines from the current list of slave machines sharing a distributed task.

Remove Faults

```
REMove Faults
[pin_pathname | instance_name | -Module
<module_name> | -bridge_location
<bridge_location1 bridge_location2> | -CLass
<fault_class> | -All | -CLOcks | -
SCan_enable | -Retain_sample <sample_number>
| -Slow <R|F|RF>]
[-LAunch <launch_clock>]
[-CApture <capture_clock>]
[-EXclusive] [-SHared]
[-INTER_clock_domain]
[-INTRA_clock_domain] [-Bridge <0 | 1 | 01>]
[-AGgressor_node <First | Second | Both>]
[-COMBINATIonal_feedback]
[-NON_Strength_sensitive]
```

This command removes faults from the current fault list, which means that the pattern generator will not attempt to detect the removed faults. You can remove faults associated with a specific pin, specific instance, or all instances of a named module; or remove a random sampling of all faults while retaining a statistical percentage; or remove all faults.

Remove Licenses

```
REMove Licenses <TEST-ANalysis | TEST-
Faultsim | TEST-ATPG-Max | TEST-DIagnosis |
TEST-Map | TEST-Fault-max | TEST-Accelerate-
max>
```

This command manually releases and checks in one or more specified licenses so that others may use them. Before you remove a Test-Delay

or Test-Iddq license, you must first change the fault model to “stuck-at” using **set faults -model stuck**.

Remove Net Connections

```
REMove Net Connections <net_names |  -All>
```

This command removes net connections previously added with the **add net connections** command. You can remove net connections from one or more specified nets or from all nets.

Remove Nofaults

```
REMove Nofaults <instance_name |
pin_pathname | -All | -Module <module_name>>
[-Stuck <0|1|01>] [pin_names]
```

This command removes some or all of the nofault attributes in the design. Nofault attributes can exist because they were defined in the netlist, added with the **add nofaults** command, or added with the **add equivalent nofaults** command.

You can use the **remove nofaults** command only when the current fault list does not contain any faults.

Remove PI Constraints

```
REMove PI Constraints <pin_names | -All>
```

This command removes one or more specified primary input constraints, or all such constraints, previously added with the **add pi constraints** command or implicitly defined in the test protocol file.

Remove PI Equivalences

```
REMove PI Equivalences <-All | pin_names>
```

This command removes one or more specified primary input equivalences, or all such equivalences, previously added with the **add pi equivalences** command.

Remove PO Masks

```
REMove PO Masks <pin_name | -All>
```

This command removes one or more specified primary output masks, or all such masks, previously added with the **add po masks** command.

Remove Scan Chains

```
REMove Scan Chains <-All | chain_names>
```

This command removes a specified scan chain or all scan chains previously defined by the **add scan chains** command. It does not affect any scan chains defined by the test protocol file and implemented with the **run drc** command.

Remove Scan Enables

```
REMove Scan Enables <pin_names | -All>
```

This command removes a specified scan enable initialization definition, or all such definitions, previously defined by the **add scan enables** command. It does not affect any scan enable initialization defined by the test protocol file and implemented with the **run drc** command.

Remove Slow Bidis

```
REMove SLow Bidis <port_name | -All>
```

This command removes the modified simulation behavior of the bus associated with a specified bidi pin, or of the buses associated with all bidi pins previously defined as slow bidis by the **add slow bidis** command.

Remove Slow Cells

```
REMove SLow Cells <instance_name | gate_id |
-All>
```

This command removes the modified simulation behavior of a specified scan or nonscan DFF or DLAT cell, or of all such cells previously defined as slow cells by the **add slow cells** command.

Remove Slow Path

```
REMove SLow Path
[-All] [-from <from,from_pin_name>]
[-through <pin_name>] [-to <to,to_pin_name>]
```

This command removes slow paths that have been defined by the **add slow path** command.

Remove Waveform Signals

```
Remove Waveform Signals [pin_pathname | gate
id | net_names | -All]
```

This command removes gates associated with the pin pathname, gate ID, net names, or all gates from the GSV.

Report ATPG Constraints

```
REPort Atpg Constraints [-Verbose | -
NOVerbose | -Summary | -All] [-Max <d>]
```

This command generates a report on the ATPG constraints added with the **add atpg constraints** command.

You can choose to get a summary report or a complete report. A summary report tells you how many ATPG constraints exist and how many are monitored during DRC analysis. A complete report shows the name, constraint value, DRC monitoring status, and location of each ATPG constraint.

Report ATPG Primitives

```
REPort Atpg Primitives <name | -Summary | -
All> [-Max <d>] [-Verbose | -NOVerbose]
```

This command generates a report on a specified ATPG primitive or all ATPG primitives added with the **add atpg primitives** command.

You can choose to get a summary report or a complete report. A summary report tells you how many ATPG primitives exist for each type (**AND**, **OR**, **SEL1**, **SEL01**, and **EQUIV**). A complete report shows the name, gate ID number, gate type, and input connections for each ATPG primitive in the design.

Report Bist

```
REPort BIst [-Lfsr_chain | -
Prpg_shadow_chains]
```

This command reports puesdo random pattern generator (PRPG) and multiple input signature register (MISR) cell data.

Report Buses

```
REPort Buses <gate_id | -BEhavior <Buf | Inv
| And | Or | Xor | Mux | TIE0 | TIE1 | TIEZ>
| -BIdis | -CLock | -Contention <Fail | Pass
| Abort | Bidi | Ignore> | -Keepers | -Pull |
-Weak | -Zstate <Fail | Pass | Abort | Bidi |
Ignore> | -Summary | -All> [-Max <d>] [-
Verbose] [-NOVerbose]
```

This command generates a report on the contention status of **BUS** primitives in the design. You can choose to get a report on a specific **BUS** primitive, on **BUS** primitives that fall into a specific category, or on all **BUS** primitives in the design. You can choose a summary report, which lists the number of **BUS** primitives in each category; or a complete report, which shows the gate ID number, contention status, Z-state status, number of strong/weak drivers, and learned behavior category for each **BUS** primitive.

Report Capture Masks

```
REPort CApture Masks
```

This command generates a report on any capture masks added with the **add capture masks** command. The report shows the gate ID number, sequential cell type, and instance name associated with each capture mask.

Report Cell Constraints

```
REPort CEll Constraints
```

This command generates a report on any cell constraints added with the **add cell constraints** command. The report shows the type of constraint, chain name, position in the chain, and instance name associated with each constraint.

Report Clocks

```
REPort CLocks
[-Command_report | -Verbose | -Matrix | -
Dominant | -INTClocks | -PLLClocks]
[-by_id]
```

This command generates a report on all pins defined to be clocks, either explicitly by the **add clocks** command or implicitly by the test protocol file. These clock pins include active-high clocks, active-low clocks, asynchronous set/reset pins defined to be clocks, and memory-write control lines. You can optionally have the clocks reported in the form of **add clock** commands

Report Commands

```
REPort COMmands [command_name | -All] [-
Usage] [-History | -History_depth <d>] [-
depth]
```

This command provides information on a specified command or all commands, like the **help** command; or displays a history of commands executed in the current session.

Report Compressors

```
REPort COMPressors <compressor_name | -Load
| -Unload | -All> [-Verbose]
```

This command generates a report containing data for the specified compressors. Note that this command is valid in both DRC and TEST modes.

Report Delay Paths

```
REPort DElay Paths <-All | path_name> [-
Verbose] [-Display] [-Pindata]
```

This command generates individual paths for viewing with the GSV and exports a delay path list.

Report Display Gates

```
REPort Display Gates
```

This command generates a report information about the gates currently being displayed in the GSV.

Report Distributed Processors

```
REPort Distributed Processors
```

This command gets information on the distributed processors performing a distributed fault simulation task.

Report Faults

```
REPort FAults [instance_name | pin_pathname]
[-CLass <fault_class_list>]
[-Unsuccessful | -SUmmary | -Profile | -All]
[-Stuck <0|1|01> | -Slow <R|F|RF>]
[-COllapsed | -UNCollapsed] [-Max <d>]
[-Verbose] [-PER_clock_domain]
[-Level <depth min_count>] [-Pattern_id <d>]
[bridge_location] [-Bridge <0 | 1 | 01>]
[-AGgressor_node <First | Second | Both>]
[-BRIDGE_Feedback] [-BRIDGE_Strong]
[-NDetects <Histogram | Faults <d>>]
[-Slack <Faults <d1> <d2> | Sdql | TMgn |
TDet | DElta | EFfectiveness>]
```

Allowed *fault_class* codes are:

DS - detected_by_simulation
DI - detected_by_implication
DR - detected_robustly
NP - not_analyzed-pos_detected
NC - not-controlled
NO - not-observed
UU - undetectable-unused
UT - undetectable-tied
UB - undetectable-blocked
UR - undetectable-redundant
AN - atpg_untestable-not_detected
AP - atpg_untestable-pos_detected
DT - detected (DS + DR + DI)
PT - possibly detected (NP + AP)
UD - undetectable (UU + UT + UB + UR)
ND - not detected (NC + NO)
AU - ATPG untestable (AN)

This command generates a report on the faults contained in the current fault list.

You can specify the type of information to be reported: faults associated with a specific instance or pin in the design, faults falling into specific classes (such as DS and UU), and all unsuccessful faults (those not identified as detected during fault simulation), a summary report that lists the numbers of faults in each category, or a complete report on all faults.

Report Feedback Paths

```
REPort FEedback Paths [path_id | -All | -
Gate <gate_id | pin_pathname |
instance_name>] [-Verbose] [-Summary]
```

This command generates a report on combinational feedback path networks in the design. A summary report just shows the number of feedback paths. A full report shows the loop ID number, number of gates, number of sources,

and sensitization status of each feedback path. To get a report on a specific feedback path, you can specify its loop ID number or a gate contained in the feedback path.

Report Instances

```
REPort INstances
instance_name [-Module name] [-Netnames]
```

This command generates a report on a specified instance. You specify either a hierarchical instance name or the name of an instance within a specified module.

Report Layout

```
REPort LAyout <layout | Tmax object_name>
[-Instance | -Cell_type | -Tmax_cell_type |
-Net]
```

This command extracts physical data and finds corresponding objects (instances, nets) between the layout and the TetraMAX design.

Report Licenses

```
REPort Licenses
```

This command lists all licenses currently checked out by your TetraMAX session.

Report Memory

```
REPort MEmory <instance name | Gate Id | -
All | -Summary> [-Contents <address | All>]
[-Max <d>] [-Verbose]
```

This command generates a summary report on all **MEMORY** primitives in the design, or a detailed report on one or all such primitives. A summary report shows the total number of RAMs and ROMs having certain stability and off-state characteristics. A standard or verbose report shows information on each memory primitive, including the type, gate ID number, instance path, and associated memory file.

Report Modules

```
REPort MOdules [name | -UNReferenced | -
UNDefined | -UNSpecified | -Behavioral | -
Errors | -Summary | -All] [-Verbose] [-udp]
```

This command generates a report on one or more netlist modules in the design. You can get a report on a specific module, all modules falling into one category, or all modules in the design.

A summary report lists the number of modules that have been specified by various means (structural Verilog, behavioral Verilog, and so on). A standard report lists the module names and provides information on each module's usage and I/O pins.

Report Net Connections

```
REPort NEt Connections
```

This command generates a report on all net connections added with the **`add net connections`** command.

Report Nets

```
REPort NEts net_name [-Module name]
```

This command generates a report on a specified net. You specify either a hierarchical net name or the name of an net within a specified module. The report lists the module input and output pins connected to the net.

Report Nofaults

```
REPort NOFaults
instance_name |
pin_pathname [-Stuck<0|1|01>] |
-Summary | -All
[-Max <d>]
```

This command generates a report on the nofault attributes in the design. Nofault attributes can exist because they were defined in the netlist,

added with the **add nofaults** command, or added with the **add equivalent nofaults** command.

Report Nonscan Cells

```
REPort NONscan Cells
<-Summary | -All |
  C0 | C1 | CU | L0 | L1 | TLA | LE | TE |
  LS | Unstable_set_resets>
[-Max <d> ] [Ram_out] [-Verbose]
```

This command generates a report on the nonscan flip-flops and latches contained in the design. You can get a summary report, which shows the number of cells falling into different categories or a detailed report on all nonscan cells falling into a specific category. A detailed report shows the nonscan cell category, behavior data, ID number, instance name, and instance type of each cell.

Report Patterns

```
REPort PAtterns <pat_specification | -All |
-Summary> [-Internal | -External] [-Chain
<name>] [-Types] [-Path_delay] [-SLack] [-
CHAIN_Mapping]
```

This command generates a report on the pattern data in the internal or external pattern buffer. A summary report shows the number of patterns of each type (Basic-Scan, Fast-Sequential, and so on). A standard report shows the pattern data for the specified range of patterns.

Report PI Constraints

```
REPort PI Constraints [-Command_report]
```

This command generates a report on all primary input and bidirectional ports that are constrained explicitly by the **add pi constraints** command or implicitly by the test protocol file.

Report PI Equivalences

```
REPort PI Equivalences [-Command_report]
```

This command generates a report on all sets of primary input and bidirectional ports that are constrained to be equivalent by the **add pi equivalences** command.

Report PO Masks

```
REPort PO Masks
```

This command generates a list of all primary output ports that are constrained to be masked by the **add po masks** command.

Report Primitives

```
REPort PRimitives [primitive ID number | instance_name | net_name | pin_pathname | -PORts | -PIS | -POS | -PIOs | -Type <type> | -Summary] [-Max <d>] [-pins] [-all]
```

This command generates a report on primitives contained in the design built by TetraMAX. A summary report shows the number of primitives of various types: primary inputs, primary outputs, **DFF**s, **NAND**s, and so on. To get a detailed report on one or more primitives, specify the ID number, instance name, net name, pin path name, primitive category, or primitive type. You can request multiple reports in a single command.

Report Rules

```
REPort RUles [rule type | rule ID | -All] [-Fail] [-verbose]
```

This command generates a report on the number of rule violations detected by TetraMAX. The report shows the rule ID number, severity level, number of failures, and a brief description of each rule.

In the command, specify the rules you want reported, such as **c4** for “clock unable to capture,” **c** for all clock rules, or **-all** for all rules. You can optionally show only the rules for which failures occurred.

Report Scan Ability

```
REPort SCan Ability [-Max <d>]
```

This command generates a report on nonscan cells that have been selected to behave as scan cells with the **set scan ability** command.

Report Scan Cells

```
REPort SCan CElls [chain_name [position] |
-Shadows | -Unstable_set_resets | -All] [-
MAX <d>] [-MASter_only] [-Reverse_order] [-
Verbose] [-SHIft_clocks | -Clocks] [-Pins]
[-IGNORE_constraints]
```

This command generates a report on scan cells in the design. The report shows the name of the scan chain, the position of the cell within the scan chain, the type of behavior identified for the cell (master, slave, shadow, and so on), the inversion status, the primitive ID of the cell, and the instance name of the cell.

In the command, specify the scan cells to report. You can specify one or all cells of a particular scan chain, all shadow gates of scan cells, all scan cells with unstable set or reset controls, or all scan cells.

Report Scan Chains

```
REPort SCan CHains
[-Command_report] [-Shift_clocks] [-Verbose]
```

This command generates a report on the scan chains in the design. For each scan chain, the report shows the scan chain name, load/unload group, length, input port name, and output port name.

Report Scan Enables

```
REPort SCan Enables
```

This command generates a report on the scan enable attributes previously defined with the **add scan enables** command.

report_sdc

```
report_sdc [-clocks][-to_cells][-groups]
[-case_analysis][-false_paths]
[-multicycle_paths][-case_paths]
[-disable_paths][-all_paths]
```

This command reports timing exceptions read in from a SDC (Synopsys Design Constraints) file. It is used only in TEST mode, and must be used in conjunciton with the **read_sdc** command, which reads in an SDC file. Note that this command is valid only Tcl mode.

Report Scan Path

```
REPort SCan Path <chain_name> <SCO |
cell_position SCI | cell_position> [-
Reverse] [-Verbose]
```

This command generates a detailed report on the gates contained in a specified scan chain. In the command, specify the name of the scan chain and the numeric range of scan cells for the report. Use **sco** to represent the scan chain output and **sci** to represent the scan chain input.

Report Settings

```
REPort SEttings [command_type | -All] [-
Command_report]
```

This command generates a report on the option settings controlled by the **set** commands: **set atpg**, **set build**, and so on. To restrict the scope of the report, specify the type of settings you want reported (for example, **report settings atpg**).

Report Slow Bidis

```
REPort Slow Bidis
```

This command generates a report on the bidi pins defined as slow bidis by the the **add slow bidis** command.

Report Slow Cells

```
REPort Slow Cells
```

This command ggenerates a report on the scan or nonscan DFF or DLAT cells previously defined with the **add slow cells** command.

Report Slow Path

```
REPort SLow Path [-All] [-from
<from,from_pin_name>] [-through <pin_name>]
[-to <to,to_pin_name>]
```

This command reports slow paths that have been defined by the **add slow path** command.

Report Summaries

```
REPort SUmmaries [PRimitives] [Faults]
[PAtterns] [Library_cells] [Memory_usage]
[Optimizations] [Sequential_depths]
[Cpu_usage] [-launch <launch_clock>] [-
capture <capture_clock>] [-exclusive] [-
shared] [-inter_clock_domain] [-
intra_clock_domain] [-PER_clock_domain]
```

This command generates a summary report on one or more of the following topics: primitives, faults, patterns, library cells, memory usage, build optimization, control/observe sequential depth, or CPU usage.

Report TRC

```
REPort Trc
```

This command reports the currently selected tester rules check.

Report Version

```
REPort VErsion [-Full]  [-Short]  [-Banner]
[-Verbose]  [-Address]  [-Release]
```

This command reports the TetraMAX version number or virtual address space.

Report Violations

```
REPort VIolations <rule_type | -All> [-max
<d>]
```

This command generates a report on the rule violations detected by TetraMAX. The report shows the severity, description, location, and rule ID number of each violation.

In the command, specify which violations you want reported: the violation ID number, the rule ID number (such as `c4`) for violations of that rule, the rule type (such as `c`) for all violations of that type, or `-all` for all violations.

Report Wires

```
REPort Wires [gate_id | -Summary | -All] [-
Contention <Fail | Pass | Abort>] [-Max <d>]
[-Verbose]
```

This command generates a report on the wire gates in the design. A summary report shows the number of wire gates falling into different contention status categories. A full report shows the gate ID number, contention status, Z-state status, number of strong and weak drivers, and behavior data for each gate.

Reset AU Faults

```
RESet Au Faults
```

This command changes the detection status of all AU (ATPG untestable) faults to ND (not detected).

Reset State

```
RESet State
```

This command resets the session state in preparation for restarting fault simulation or test generation using the current fault list. Any existing internal patterns are deleted and faults are set to their initial classification, giving credit for any current redundancy and ATPG untestable conditions.

Rm

```
RM filename
```

This command removes an existing file, like the UNIX **rm** command. It does not support the use of wildcard (*) characters.

Run ATPG

```
RUn Atpg [Basic_scan_only |
FAst_sequential_only | FUll_sequential_only]
[-Auto_compression] [-Random] [-Distributed]
[-Observe_file <file_name>] [-NDEtects <d>]
[-NOdisturb_clock_grouping]
```

This command generates test patterns for the current set of faults using the pattern source set by the **set patterns** command. To specify the parameters for ATPG, use the **set atpg** command.

Run Build_model

```
RUn Build_model [top_module]  [-Weakgates]
[-all]  [-remove_pio_pull]  [-prt < >]
```

This command builds the in-memory simulation model from the design modules that have been read into TetraMAX. To control the build process, use the **set build** command. After the build process is completed, TetraMAX performs a learning process, which you can control with the **set learning** command.

Run Diagnosis

```
RUn DIagnosis <file_name> [-Truncate <d>]
[-Chain_failure] [-Display] [-
Only_report_failures] [-Verbose] [-
Write_fault_list <file_name>] [-Replace] [-
COMPress <Gzip | Binary>] [-RANk_faults]
```

This command performs a device failure diagnosis, given a file that identifies its failing tester measures. This diagnosis assumes the test patterns were originally generated by TetraMAX and that these patterns are available. Note that you use the **`set diagnosis`** command to make specifications relevant to this command.

The pattern set must be external, as specified by the **`set patterns`** command. For a selected fault, you can use the **`run simulation`** command to create a representative failure file to use for a diagnosis.

In the **`run diagnosis`** command, you can optionally specify a truncation point for the test patterns, request a chain load/unload diagnosis instead of a the standard functional diagnosis, and request a graphical schematic display of the results.

Run DRC

```
RUn DRc [file_name] [-Test | -Delete] [-
prescan] [-DFt] [-PATTERNEXEC <exec_name>]
[-APpend]
```

This command performs design rule checking. TetraMAX checks the scan structure of the design and determines how to use this structure for test generation and fault simulation. The DRC process is affected by the **`set drc`** and **`set atpg`** commands.

You can optionally specify the name of a DRC file (the STIL procedure file) to be used for design rule checking. This file can specify DRC

information such as clock definitions, tester cycle periods, output measure values, scan chains, and scan shift initialization. You can also specify this file name using the **set drc** command.

Run Fault_sim

```
RUn FAult_sim [-First_pattern <d>] [-
Last_pattern <d>] [-STore] [-CHeckpoint
<fault_filename>] [-NODrop_faults | -
SEquential | -SEquential_nodrop_faults] [-
Distributed] [-DETected_pattern_storage] [-
STRong_bridge] [-NDEtects <d>]
```

This command performs fault simulation for the current fault list and pattern source, allowing you to determine the test coverage obtained by an externally generated test pattern.

You can optionally specify a truncation point (last pattern number), request a sequential simulation rather than a scan-based combinational simulation, or store the external patterns that detect faults in the internal pattern buffer.

Run Justification

```
RUn Justification [-Set <<gate_id |
pin_pathname> <0 | 1 | Z | R | F>>] [-
Verbose] [-STore] [-Previous] [-
NOConstraints] [-NOPrevention] [-
FUll_sequential] [-Replace_pattern <d>]
```

This command creates a pattern that satisfies a given set of conditions placed on specified internal or external circuit nodes. If this process is successful, you can optionally place the resulting pattern into the internal pattern buffer and then write it out to a file. The justification process uses ATPG techniques, as defined by the **set atpg** command.

To specify the conditions for justification, use the **-set** option and specify one or more circuit locations and the desired value (0, 1, Z, R, or F) at each location.

Run Mapping

```
RUn Mapping
<instance_name | -Module <module_name>>
<-Pattern_file <file_name>>
[-DELete] [-DEPendent_pattern_mapping]
[-NOMAp_pin <pin_name>]
[-Sensitive | -INSensitive] [-Verbose]
[-CLock_merge] [-MAX_Errors <d>]
[-Analyze <d>]
[-STORE_Debug_data | -NOSTORE_Debug_data] [-
Merge | -MIXed_event_merge | -NOMErge]
[-REAd_write_overlapping | -
NOREAd_write_overlapping]
[-Reload_after_unload | -
NOReload_after_unload]
[-STRobe_offset <<d> FS|PS|NS|US|MS|S>]
[-STRobe_period <<d> FS|PS|NS|US|MS|S>]
[-STRobe_rising <port_name>]
[-STRobe_falling <port_name>]
[-STRobe_comments <comments>]
[-STRobe_event < >]
[-VCd_clock <<0|1> <port>>]
```

This command maps functional test patterns for an embedded submodule into new test patterns for the whole device under test. Based on the connectivity of the embedded submodule, TetraMAX determines how to control the inputs and observe the outputs of the submodule, and then constructs its own patterns that apply the pattern data to the submodule within the larger design.

Run Observe Analysis

```
RUn Observe Analysis <-Observe_file <file
name>> [-Replace] [-MAX_Observe_points <d>]
[-Num_observe_points_per_cell <d>]
```

This command performs an observability analysis of the design based on controllability and observability SCOAP numbers, and then

suggests the best places in the design to modify for improved test coverage. TetraMAX lists the identified locations in the transcript in rank order as the analysis progresses.

Run Pattern_compression

```
RUn Pattern_compression [num_passes]
[-RESet_au_faults] [-MIn_eliminated_pats
<d>] [-MAx_useless_passes <OFf | d>]
[-Order <RAndom | REverse>]
[-Verbose | -NOVerbose] [-RETain_order] [-
NDetects <d>]
```

This command attempts to reduce the number of patterns by eliminating unnecessary patterns from the current pattern list. TetraMAX identifies the unnecessary patterns by performing fault simulation of the current patterns, retaining only those patterns that are required for detection of additional faults. Then, for a multiple-pass compression, it reorders the patterns and repeats the same elimination process. To avoid repeating the same pattern order, the reordering algorithm alternates between reversing the current order and rearranging the order randomly for each pass.

Run Simulation

```
RUn Simulation [-SEquential]
[-SEQUENTIAL_Update] [-STore] [-NOCompare]
[-Nox_difference] [-pin <<pathname> <0 | 1>>
| -Chain <<chain_name> <position>>]
[-Max_fails <d>] [-Failure_file <file_name>]
[-Replace] [-Last_pattern <d>]
[-Bridge <WAnd | WOr | Adom | Bdom> <nodea>
<nodeb>>] [-SLow <R | F | RF>]
[-FASt <R | F | RF>] [-STuck <0 | 1>]
[-Update]
```

This command performs a simulation using the current pattern source, as determined by the **`set patterns`** command, and reports any differences between the simulated and expected values.

You can run this simulation in the presence of a single specified fault, which shows all simulation failures resulting from that fault. You can optionally save the failure data in a file and use that file as input to the **`run diagnosis`** command.

Run Testpoint Analysis

```
RUn TestPoint Analysis
<-Test_point_file <file name>> [-Replace]
[-MAX_Test_points <d>]
[-Num_observe_points_per_cell <d>]
[-Class <list of fault_classes>]
[-Dont_touch <module name>]
```

This command controls test-point insertion based on undetected fault topology. Note that this command can only be used in the TEST command mode.

Run TRC

```
RUn TRc
```

This command performs design tester rule checking.

Set ATPG

```
Set Atpg
[-ADJacent_fill_range <d>]
[-ALlow_clockon_measures | -
NOALlow_clockon_measures]
[-ANalyze_untestable_faults | -
NOANalyze_untestable_faults]
[-Abort_limit <d>]
[-BASIC_Min_detects_per_pattern <d [d]>]
[-CApture_cycles <d>]
[-CHAin_test <OFf | 0011 | 0101 | 1000 |
0111 | <bit_string> <R | C>>]
[-CHeckpoint <<interval_time>
```

```
<fault_filename> <pattern_filename>> | -
NOCHeckpoint] [-COverage <f>] [-DEcision
<Random | NORandom>] [-DI_analysis |  -
NODI_analysis] [-
FAST_Min_detects_per_pattern <<d> [d]>]
[-FIll <Random | 0 | 1 | X | Adjacent>]
[-FULL_Min_detects_per_pattern <<d> [d]>]
[-FULL_SEQ_ABort_limit <d>]
[-FULL_SEQ_ATpg | -NOFULL_SEQ_ATpg]
[-FULL_SEQ_Merge <Off | Low | Medium | High
| <d>>]
[-LETE_fastseq | -NOLETE_fastseq]
[-FULL_SEQ_Time <max_secs_per_fault
[max_secs_per_run]>]
[-MErge <Off | Low | Medium | High | <d> [Low
| Medium | High | <d>]>]
[-NEW_Capture | -NONEW_Capture]
[-PATterns <d>] [-PRevention <Random |
NORandom>]
[-POST_Capture_contention_prevention | -
NOPOST_Capture_contention_prevention]
[-RESET_Bidis | -NORESET_Bidis]
[-SINGle_load_per_pattern | -
NOSINGle_load_per_pattern]
[-STore | -NOSTore] [-SUmmary | -NOSUmmary]
[-Time <<max_secs_per_fault>
[max_secs_per_run]>]
[-Verbose | -NOVerbose]
[-OPTIMIZE_Bridge_strengths | -
NOOPTIMIZE_Bridge_strengths]
[-min_detects_per_pattern <d>]
[-MIN_PATterns_threshold <d>]
[-MIN_Ateclock_cycles <d>]
```

This command sets the parameters that control the ATPG process. It also affects all analyses that use the test pattern generation algorithm, including DRC and justification. You can display the current settings with the **`report settings`** command.

Set Bist

```
Set BIst
[-CHAin_test | -NOCHAin_test]
[-DBist | -NODBist] [-DEbug <d> | -NODEbug]
[-DUmp <interval_id | None>] [-Hex | -NOHex]
[-MAX_Intervals <d>] [-MAX_Routing <d>]
[-MAX_Seed_patterns <d>]
[-NUM_Dbist_patterns_per_interval <d>]
[-NUM_Patterns_per_interval <d>]
[-Randomize_pis | -NORandomize_pis]
[-USE_CEll_constraints | -
NOUSE_CEll_constraints]
[-USE_COnstant_value_cells | -
NOUSE_COnstant_value_cells]
[-OPTimize_fault_sim | -
NOOPTimize_fault_sim]
[-Verbose | -NOVerbose]
[-FORCE_Xdbist_scalable_selector |
[-HEX_Diag_data | -NOHEX_Diag_data]
[-force_lfsr_load] [-min_l < >]
[-noforce_lfsr_load] [-PO_assumed_scan]
[-NOPO_assumed_scan] [-ASSUME_Bist_setup]
[-NOASSUME_Bist_setup]
[-Trailing_edge_checking]
[-NOTrailing_edge_checking]
```

This command is the primary interface for controlling BIST related ATPG pattern generation options, such as patterns per interval, and maximum seeds

Set Build

```
Set BUIld
[-DELete_unused_gates |  -
NODelete_unused_gates]
[-Merge <Equivalent_dlat_dff |
EQUIVALENT_Initialized_dlat_dff |
NOEquivalent_dlat_dff | FLipflop_from_dlat |
FLIPFLOP_Cell_from_dlat |
NOFLipflop_from_dlat | Dlat_from_flipflop |
NODlat_from_flipflop |  Wire_to_buffer |
NOWire_to_buffer |  Mux_from_gates |
MUXPins_from_gates | MUXX_from_gates |
NOMux_from_gates |  Xor_from_gates |
XORPins_from_gates | NOXor_from_gates |
Cascaded_gates_with_pin_loss |
NOCascaded_gates_with_pin_loss |
Tied_gates_with_pin_loss |
NOTied_gates_with_pin_loss |
Global_tie_propagate |
```

```
NOGlobal_tie_propagate | Bus_keepers |
NOBus_keepers | FEedback_paths |
NOFEedback_paths >]
[-Undriven_bidi <PIO | PI | PO >]
[-Hierarchical_delimiter <c>]
[-Coerce_port_directions | -
NOCoerce_port_directions]
[-Fault_boundary <Lowest | Hierarchical>]
[-Add_buffer | -NOAdd_buffer]
[-Limit_fanout <d>]
[-Black_box <module_name> | -Empty_box
<module_name> | -DESign_box <module_name> |
-Portfault_box <module_name> | -NOBox
<module_name> | -Reset_boxes]
[-INStance_modify <<instance_name>
<gate_type>> | -NOINStance_modify
<instance_name> | -NOINStance_modify_all]
[-Net_connections_change_netlist | -
NONet_connections_change_netlist]
```

This command sets the parameters that control the building of simulation models with the **`run build_model`** command. You can designate black box modules and control optimization features such as gate merging and unused gate deletion.

Changes to the build settings only affect future use of the **`run build_model`** command; they do not affect the flattened image already created by a prior build operation.

Set Buses

```
Set BUSes
[-External_z <X | 0 | 1 | Z>]
[-Fault_contention <And | Or | X>]
[-Contention_status <<Ignore | NOIgnore>
<gate_id |pin_pathname | All>]
[-Zstate_status <<Ignore | NOIgnore>
<gate_id | pin_pathname | All>>]
```

This command controls the behavior of bus gates. It sets the behavior of the Z state for external pins and the fault machine contention value for buses with contending values.

Set Colors

```
Set COLors <Error | Warning | User_commands
| File_commands> <red_number>
<green_number>  <blue_number>
```

This command sets the text display color in the transcript for messages of a specified type: errors, warnings, user command messages, or file command messages. You specify the intensity of each color as a integer value from 0 to 255.

Set Commands

```
Set COMmands [History | NOHistory]  [NOAbort
| Abort | Exit]
```

This command determines whether TetraMAX maintains a command history and the action taken when an error occurs during command file execution (abort, continue, or exit).

Set Compressor Connections

```
Set COMPressor Connections
<compressor_name>  [-Mode <d>]
[-Internal <<input_id> <output_id>...>]
[-LOAD_input <<input_id> <port_name>>]
[-LOAD_Output <<output_id> <chain_name>>]
[-UNLOAD_Input <<input_id> <chain_name>>]
[-UNLOAD_Output <<output_id> <port_name>>]
```

Set Contention

```
Set CONtention [ABus | NOABus] [BIdi |
NOBIdi] [Bus | NOBus]
[Dff_dlat | NODff_dlat] [Float | NOFloat]
[Ram | NORam]
[WIre | NOWIre]
[-Atpg | -NOAtpg] [-Capture | -NOCapture]
[-Multiple_on | -NOMultiple_on]
[-REtain_bidi | -NOREtain_bidi]
[-Severity <Error | WArning | Ignore>]
[-Preclock | -NOPreclock]
[-Verbose | -NOVerbose]
```

This command specifies the manner in which TetraMAX performs contention checking, the types of design elements checked (bidirectional ports, buses, RAMs, and so on), and the action taken when a contention condition occurs.

Set Delay

```
Set DElay
[-PI_changes | -NOPI_changes]
[-PO_measures | -NOPO_measures]
[-ALLOW_Reconverging_paths | -
NOALLOW_Reconverging_paths]
[-DIagnostic_propagation | -
NODIagnostic_propagation]
[-Mask_nontarget_paths | -
NOMask_nontarget_paths] [-RElative_edge | -
NORElative_edge]
[-RObust_fill | -NORObust_fill]
[-Launch_cycle <Last_shift | System_clock |
Any>]
[-DAta <path_name>]
[-Simulate_hazards | -NOSimulate_hazards]
[-COmmon_launch_capture_clock | -
NOCOmmon_launch_capture_clock]
[-TWO_clock_transition_optimization | -
NOTWO_clock_transition_optimization]
[-ALLOW_Multiple_common_clocks | -
NOALLOW_Multiple_common_clocks]
[-SLow_equivalence | -NOSLow_equivalence]
[-DISturb_clock_grouping]
[-NODISturb_clock_grouping]
[-TRAnsition_clocking_checks]
[-NOTRAnsition_clocking_checks]
[-MAX_Delta_per_fault <value>]
[-SLACKDATA_FOR_Atpg]
[-NOSLACKDATA_FOR_Atpg]
[-SLACKDATA_FOR_Faultsim]
[-NOSLACKDATA_FOR_Faultsim]
[-Clock_period <f>]
[-SDQL_coefficient <{n1 n2}>]
[-MAX_Tmgn <f>] [-MUlticycle_length <d>]
```

This command sets options for use in transition fault and path delay fault test generation and fault simulation.

Set Diagnosis

```
SEt DIagnosis
[-Time_limit <d> | -NOTime_limit]
[-Verbose | -NOVerbose]
[-CHEck_expected_data | -
NOCHEck_expected_data]
[-LAyout_data |  -NOLAyout_data]
[-Sample <<d> <d>>]
[-INcomplete_failures | -
NOINcomplete_failures]
[-REPOrt_net_data <d> | -NOREPOrt_net_data]
[-Max_report_failures <d>]
[-CYcle_offset <d>]
[-Pt_credit | -NOPt_credit]
[-COMposite | -NOCOMposite]
```

This command specifies the parameters associated with performing a diagnosis of a failing device (specified by the **`run diagnosis`** command).

Set Distributed

```
Set DIstributed [-SHELL_Timeout <d>]
[-SLave_setup_timeout <d>]
[-PRint_stats_timeout <d>]
[-SCript <user defined tmax script>]
[-Work_dir <work directory>]
[-SHELL <type of shell: rsh, ssh, remsh>]
[-Verbose | -NOVerbose]
```

This command controls the amount of time the system is given to perform various distributed processor-related communications, and identifies the location of a common working directory that the machines performing a distributed fault simulation task will share to exchange information. Every machine sharing this task must have read/write/execute rights to this directory.

Set DRC

```
Set Drc
[file_name | -NOFile] [-Oscillation <d>]
[-TRace | -Chain_trace <chain_name> | -
NOTRace]
[-clock_gating_init_cycles_integer <d>]
[-SHadows | -NOSHadows | -
SErial_shadows_only]
[-TLas | -NOTLas]
[-STORE_SEtup | -NOSTORE_SEtup]
[-STORE_STability_patterns | -
NOSTORE_STability_patterns]
[-Bidi_control_pin <0 | 1 <name>> | -
Bidi_control_none]
[-SKew <d>] [-Remove_false_clocks | -
NORemove_false_clocks]
[-Initialize_dff_dlat <0 | 1 | X | Random>]
[-Allow_unstable_set_resets | -
NOAllow_unstable_set_resets]
[-CLock <pin_name> | -Any | -Dynamic | -
One_hot | -Seq_capture]
[-MUlti_captures_per_load | -
NOMulti_captures_per_load]
[-DISturb_clock_grouping | -
NODISturb_clock_grouping]
[-COntroller_clock <pin_name> | -
NOCOntroller_clock]
[-Z_Check_with_all_constraints | -
NOZ_Check_with_all_constraints]
[-Unstable_lsrams | -NOUnstable_lsrams]
[-Dslave_remodel | -NODslave_remodel]
[-OBServe_procedure <Master | Any>]
[-Pipeline | -NOPipeline] [-optimize]
[-NUM_Pll_cycles <d>]
[-MAX_Pll_simulation_passes <d>]
[-PLL_Simulate_test_setup | -
NOPLL_Simulate_test_setup]
[-USE_Cell_constraints | -
NOUSE_Cell_constraints]
[-COMPressor_debug_data]
[-NOCOMPressor_debug_data]
[-report_x_sink]  [-noreport_x_sink]
[-STORE_UNload_mode_data]
[-NOSTORE_UNload_mode_data]
[-PLL_launch_on_shift | -
NOPLL_launch_on_shift]
[-CONStraints | -NOCONStraints]
[-SDC_environment <Tmax_drc |
Sdc_case_analysis | None>]
```

This command sets various parameters that control the DRC process carried out by the **run drc** command. You can control features such as the use of a DRC file (STIL procedure file), the use of unstable set/reset inputs on sequential devices, clock characteristics, sequential device initialization, false clock removal, shadow register detection, transparent latch detection, and scan chain tracing.

Set Environment GUI

```
Set Environment Gui [-Max_history_length
<d>] [-ERror_messages <Transcript |
Message_box>] [-TOolbar <Text |
BItmap_only>] [-ENd_scroll | -NOEnd_scroll]
[-TITLE_Prefix <Window banner prefix>] [-
CMDbar | -NOCMDbar]
```

This command provides an alternative method for setting up the graphical user interface (GUI). Instead of using the **Edit > Environment** command in the GUI and interactively setting up the environment, you can use this command (possibly in a command file) to accomplish the same setup tasks.

Set Environment Info

```
Set Environment Info [-Always_on_top | -
NOAlways_on_top] [-Save_dimensions | -
NOSave_dimensions]
```

This command provides an alternative to the **Edit > Environment** command for controlling the overlap of the gate information window over other windows and the session-to-session window dimensions.

Set Environment Reports

```
Set Environment Reports
[-MAX_lines <d>] [-Line_length <d>]
[-Color_text <red_number green_number
blue_number>]
[-Color_background <red_number green_number
blue_number>]
[-Xpos <d>] [-Ypos <d>]
[-Width <d>]  [-Height <d>] [-Font_size <d>]
```

This command provides an alternative to the **Edit > Environment** command for controlling the text display characteristics resulting from various **report** commands. You can control the text color, text size, initial window location/dimensions, and so on.

Set Environment Transcript

```
Set ENVironment Transcript
[-Font_size <d> ]
[-Line_length <d> ]
[-MAX_lines <d> ]
```

This command provides an alternative to the **Edit > Environment** command for controlling the transcript window characteristics: text size, line length, line buffer size, and left margin size.

Set Environment Viewer

```
Set Environment Viewer
[-Buf_inv | -NOBuf_inv]
[-DEsign_symbols <Block | Primitives>]
[-DFf_dlat_symbol <MODE1|MODE2|MODE3>]
[-Font <System | Hershey>]
[-Instance_names | -NOInstance_names]
[-Left_justify_pis | -NOLeft_justify_pis]
[-Max_gates <d>] [-Primitive | -NOPrimitive]
[-Redraw <Always | If_necessary>]
[-MAX_Pindata_length <d>]
```

```
[-Scheme <Default | Black_on_white |
Synopsys>] [-TIme_out <d>]
[-TRuncate | -NOTRuncate]
[-Color <BAckground | BIdi | BLock |
BUfinv_removed | EDge_connector | ID
|INstance_name | HIghlight | NEt |
PINData_text | PINName_text | RUbberband |
TIE | TITLE_Background | TITLE_Text |
TYpe_name_text <red_number> <green_number>
<blue_number>>]
```

This command provides an alternative to the **Edit > Environment** command for controlling the display characteristics of the graphical schematic viewer. You can control the colors of objects, the D flip-flop display mode, the text font, the presence of instance name labels, the redraw parameters, and so on.

Set Faults

```
Set Faults
[-Model <Stuck | Iddq | Transition |
Path_delay | Bridging>]
[-Report <Collapsed | Uncollapsed>]
[-Pt_credit <d>] [-AU_credit <d>]
[-ATpg_effectiveness | -
NOAtpg_effectiveness]
[-Fault_coverage | -NOFault_coverage]
[-Equiv_code <code_name> | -NOEquiv_code]
[-Summary <Verbose | NOVerbose>]
[-BRIDGE_Inputs | -NOBRIDGE_Inputs]
```

This command specifies the type of fault model used for pattern generation and the characteristics of the fault report.

The three primary fault models are the stuck-at model, used for standard test pattern generation; the IDDQ model, used to generate quiescent states for IDDQ testing; and the transition delay fault model, used to detect slow-to-rise and slow-to-fall faults.

You can control fault reporting characteristics such as collapsed versus uncollapsed faults, credit for potentially detected (PT) and ATPG

untestable (AU) faults, and coverage types reported (test coverage, fault coverage, and ATPG effectiveness).

Set IDDQ

```
Set Iddq
[Float | NOFloat] [Strong | NOStrong]
[WEak | NOWEak] [WRite | NOWRite]
[-Atpg | -NOAtpg] [-Toggle | -NOToggle]
[-Exclude_ports <None | Bidis | All>]
```

This command specifies the conditions considered during IDDQ test pattern generation. The command settings are valid only if the IDDQ fault model has been selected with the **set faults** command.

In the **set iddq** command, an assertive keyword such as **float** means that the condition is considered and therefore avoided for IDDQ pattern generation. A negative keyword such as **nofloat** means that the condition is not considered and therefore allowed for IDDQ pattern generation.

Set Learning

```
Set Learning
[-Atpg_equivalence | -NOAtpg_equivalence]
[-Module_learning | -NOModule_learning | -
Common_input | -NOCommon_input]
[-Equivalent_latches | -
NOEquivalent_latches]
[-Implication <Low | Medium | High> | -
NOImplication]
[-MAX_FEedback_sources <d>]
[-Sim_passes <d>]  [-Test_passes <d>]
[-Verbose | -NOVerbose] [-nomodule]
[-DISable_time_limit]
[-NODISable_time_limit]
```

This command sets the parameters for the learning process that TetraMAX performs immediately after it builds a model. During this process, each gate can be assigned a learned

behavior characteristic (blocked, constrained, equivalent, and so on). TetraMAX uses this information during test pattern generation.

With this command, you can specify whether to perform various types of analysis (ATPG equivalence, common input, equivalent latch, and implication), and you can specify certain parameters for the learning analysis.

Set Messages

```
Set Messages
[-Display | -NODisplay]
[-Double_slash | -NODouble_slash]
[-NOLog | -Log <filename>] [-Replace | -
Append]
[-Level <Expert | Standard>]
[-Transcript_comments |
-NOTranscript_comments]
```

This command sets the parameters for the generation and display of messages in the transcript window.

Set Netlist

```
Set Netlist [-Pin_assign | -NOPin_assign]
[-EScape <Cond | All | None>] [-SCalar_net |
-NOSCalar_net]
[-Max_errors <d>]
[-REDefined_module <First | Last>]
[-Dominance_detection <ON | OFf | Boolean>]
[-CElldefine | -NOCElldefine]
[-ENable_portfaults | -NOENable_portfaults]
[-SUppress_faults | -NOSUppress_faults]
[-Xmodeling | -NOXmodeling]
[-SEquential_modeling | -
NOSEquential_modeling]
[-CHeck_only_used_udps | -
NOCHeck_only_used_udps]
[-COnservative_mux <None | Combinational_udp
| All>]
```

This command sets the parameters that control the reading of netlists with the **read netlist** command. It only affects the subsequent reading of netlists, not netlists that have been read in already.

Set Patterns

```
Set PAtterns [Internal|Random|Bist|External
<filename>]
-Delete
[-SEnsitive | -INSensitive]
[-STROBE_Comments <comments>]
[-STROBE_Period <<d> <FS|PS|NS|US|MS|S>>]
[-STROBE_Rising <port_name>]
[-STROBE_Falling <port_name>]
[-STROBE_Event < >] [-STROBE_Offset < >]
[-Histogram_summary | -NOHistogram_summary]
[-Load_summary | -NOLoad_summary]
[-NOVCD_clock | -VCD_clock <<Auto> | <0|1>
<port>>]
[-VERIlog_last_scan | -NOVERIlog_last_scan]
[-translate_dbist_patterns]
[-EXPAND_xdbist_patterns]
[-Cycle_count | -NOCycle_count]
[-Netlist_independent]
[-NONetlist_independent] [-APPend]
[-Measure_forced_bidis | -
NOMeasure_forced_bidis]
[-external <filename>]
```

This command specifies the source of patterns to use for simulation, fault simulation, or test generation. You can select internally generated patterns, random patterns (as defined by the **set random_patterns** command), or an external pattern file (the method used to read ATPG or functional patterns).

External pattern files can be in any of the following formats: binary, extended VCD (VCDE), STIL, Verilog, VHDL, and WGL. For patterns in VCDE format, you can optionally specify the strobe trigger conditions for measuring expected values.

Set Pindata

```
Set PIndata
[-None | -CLOCK_Cone <pin_name> | -CLOCK_ON
<pin_name> | -CLOCK_OFf | -Constrain_data |
-Debug_sim_data | -DELAY_data | -Error_data
| -FAULT_Sim_results <<pin_pathname | gate
pin_number> <0|1> <pattern>> | -Fault_data |
-Good_sim_results <d> | -Load | -
Master_observe | -Pattern <<d> | All |
Fast_sequential> | -FULL_SEQ_Scoap_data | -
FULL_SEQ_Tg_data | -SCoap_data | -
SEQ_Sim_data | -SHAdow_observe | -SHIft | -
BIST_Data | -BIST_Setup | -
STability_patterns | -TEst_setup | -TIe_data
| -Bidi_control_value]
[-Refresh] [-Shift_character <<c>>]
[-Constrain_character <<c>>]
[-UNLOAD_mode]
[-DRCdata < UNLOAD_Mode <d> >]
[-SDC_case_analysis]
```

This command specifies the type of pin data displayed by the **`report primitives`** command and by the graphical schematic viewer. You select any one of several pin data types: clock cone data, clock-off data, constraint data, fault data, SCOAP data, simulation event data, test setup data, and so on. You can optionally have the schematic display refreshed (redrawn with the new pin data) immediately.

Set Primitive_report

```
Set PRimitive_report
[-Max_fanout <d>]
[-Interval <d>]
[-Time <Clock | PReclock | POstclock | Lete
| All>]
[-Verbose | -FAST_lookup |  -FAST_NOLCS |  -
NOFAST_lookup]
```

This command specifies the type of information displayed by the **`report primitives`** command, including the pin data spacing interval, the number of fanout connections, and the time with respect to the capture clock at which to display the data.

Set Random_patterns

```
Set RAndom_patterns
[-Clock <pinname> | -Clock_none]
[-Length <d>]
[-Observe <Master | SLave | SHadow>]
```

This command sets the parameters that control the application of random patterns, including the capture clock (if any), the number of patterns to apply, and the cell observe points.

Set Rules

```
Set RUles
[rule_id] [Error | Warning | Ignore]
[-Mask | -NOMask]
[-AUtofix | -NOAUtofix] [-reset]
```

This command sets the rule severity (error, warning, or ignore) for a specified rule (for example, C1 or N2). For certain DRC rules, you can optionally mask the control or observe functionality of the affected sequential elements to improve the ability to create patterns that pass in simulation, at the cost of lowered test coverage.

Set Scan Ability

```
Set SCan Ability
ON | OFf
nonscan_cell_gate | -DFf | -Dlat | -All
```

This command temporarily converts one or more specified nonscan cells into scan cells (or changes them back to nonscan cells), allowing you to explore the effects of these changes on test coverage.

set_sdc

```
set_sdc [-Verbose | -NOVerbose]
[-CAse_paths | -NOCase_paths]
[-Instance <instance_name>]
[-Environment <Tmax_drc | Sdc_case_analysis
| None>]
```

This command controls how TetraMAX interprets an SDC file used for specifying setup timing exceptions. It will only work if you specify it before the **read_sdc** command. You must be in DRC mode to use this command (after you successfully run the **run_build** command, but before running **run_drc**). Note that the **set_sdc** command is valid only Tcl mode.

Set Simulation

```
Set Simulation
[-Bidi_fill | -NOBidi_fill | -
STRong_bidi_fill | -WEAk_bidi_fill]
[-Data <<first_pattern> <last_pattern>>]
[-Measure <Sim | Pat>]
[-Oscillation <<num_passes>
<num_additional_fault_passes>>]
[-Words_per_pass <d>] [-Verbose | -
NOVerbose]  [-BAsic_scan | -NOBAsic_scan]
[-STOre_memory_contents <pattern_id | LAst>
| -NOSTOre_memory_contents]
[-XClock_gives_xout | -NOXClock_gives_xout]
[-XFill_out_of_range_write | -
NOXFill_out_of_range_write]
```

This command sets the parameters for simulations run with the **run simulation** and **run fault_sim** commands. You can control the simulation algorithm (Basic-Scan or sequential), the bidirectional port driving mode, the source of expected data for fault simulation, and so on.

Set TRC

```
Set TRc
filename [label...] |
-NOFile < [label]... |
-All >
```

This command selects the Tester Rule Checks to be performed after DRC or with the **run trc** command. Note: The syntax for the -nofile option requires either a label or the **-All** option. The need for the labels is different for the filename and the **-nofile** versions of this command.

For filename, the labels are optional; if labels are not specified, the first label in the tester rule check file is read. However, one or more labels (or **-All**) are required when specifying the **-nofile** option.

Set WGL

```
Set WGl
[-Bidi_map <<ab> <cd>>]
[-Scan_map <Dash | Bidi | Keep | None >]
[-Macro_usage | -NOMacro_usage]
[-PAd | -NOPad] [-Last_scan | -NOLast_scan]
[-Group_bidis | -NOGroup_bidis]
[-Tester_ready | -NOTester_ready | -
PRe_measured]
[-INversion_reference <Master | Cell | Omit
>]
[-FOrces_during_load <Allow_x |
Previous_values | No_x>]
[-Chain_list <All | Shift>]
[-0] [-1] [-x] [-z] [-identical_bidi_map]
```

This command sets the parameters for WGL pattern generation. You can specify whether to force unspecified inputs during load, whether to generate scan or parallel vectors for the last patterns, whether to use macros, whether to pad the shorter scan chains, and so on.

Set Workspace Sizes

```
Set WOrkspace Sizes
[-Atpg_gates <d>] [-CONnectors <d>]
[-Decisions <d>] [-DRC_buffer_size <d>]
[-Line <d>] [-String <d>]
[-COMMAND_Line <d>] [-COMMAND_Words <d>]
```

This command specifies the maximum number of items of a specific type allowed in the current TetraMAX environment: user-defined ATPG gates, fanout connections for the graphical schematic viewer, active decisions, state element candidates, line numbers per input file, or characters per string.

Source

`Source <file_name> [echo]`

Use this command for executing UNIX shell commands. The command string is passed to the default shell and executed. The command string may be enclosed in double quotes. When you use the echo option during TCL Mode, it echoes the commands in the **file_name** and displays them them in the GUI transcript or shell xterm.

System

`SYstem command_string`

This command executes UNIX shell commands. The command string is passed to the default shell and executed. The command string may be enclosed in double quotes.

NOTE: The standard input and standard output of this command is not available in TetraMAX. Progress or status from this command is not shown in the transcript.

Test

`TEST`

This command manually returns a TetraMAX session to the TEST command mode, allowing you to execute commands that operate only in that mode.

Unalias

`UNALias <alias_name | -All>`

This command removes a specified alias definition, or all definitions, created by the **alias** command.

Update Clock

```
Update Clock <-Wft <wft_name>>
<-Clock <clock_name>>
[-Pulse <<d>,leading and failing edge>]
[-Unit <ps | ns>]
```

This command changes the pulse timing of a specific clock in a specific named WaveformTable. It can be used only after the **`read drc`** command. All optional parameters not otherwise specified in this command will remain at existing values. The time values specified by this command, scaled based the specified unit value, will be modified to fit within the current scale of the WaveformTable being changed.

Update Wft

```
Update Wft <-Wft <wft_name>> [-Period <d>]
[-Force <d>] [-Strobe <d>] [-Unit <ps | ns>]
```

This command changes any of the period, force-event timing, or measure-event timing parameters within a specific named WaveformTable. It can be used only after the **`read drc`** command.

Write Drc_file

```
Write Drc_file
filename
[-Replace] [-Compress <Gzip | Binary> ]
[-Scancells] [-Generic_captures]
```

This command writes a test protocol file in STIL format based on the current clocks, PI constraints, PI equivalences, pin timing, and known STIL procedure files or Synopsys protocol files.

Write Faults

```
Write Faults
<filename> [instance_name | pin_pathname]
[-CLass <fault_class_list> | -SUmmary | -
All] [-Stuck <0|1|01> | -Slow <R|F|RF>]
[-COMpress <Gzip | Binary>]
[-COllapsed | -UNCollapsed]
[-Max <d>] [-Replace] [-Profile]
```

This command writes fault data to an external file. In the command, specify the name of the external file and the scope of the fault list: the faults associated with a specified instance or pin, the faults belonging to a specified class or classes of faults, all faults, or a summary report. You can optionally specify a collapsed or uncollapsed list, thus overriding the **`set faults`** command setting.

Write Image

```
Write Image <file_name> [-COMpress <Gzip |
Binary | Off>] [-Replace] [-Violations] [-
NEtlist_data]
```

This command after a successful DRC run writes the TetraMAX database to a file.

Note: This command does not write patterns out to the image file.

Write Netlist

```
Write Netlist <filename> [-Replace] [-Top
<top_name>] [-Format <Edif [External] |
VErilog>] [-Compress <Gzip | Binary>] [-STOp
<Edif | VErilog | VHdl | Design_level>]
```

This command writes netlist data to an external netlist file in a specified format. You can use compression to minimize the size of the file.

Note: Net connections added with the **`add net connections`** command are not reflected in the netlist written.

WARNING: Do not consider the write netlist command a tool for converting netlists between Verilog, EDIF, and VHDL formats, or to produce legal Verilog syntax for some other tool. The netlist written is intended for ATPG tool use and does not contain many of the items which can have been in the original netlist or module definitions.

For example, any behavioral module, with the exception of recognized RAM/ROM definitions, is converted to black box modules. Any timing information is stripped away. Any "specify" blocks may be stripped away. Any undefined modules will become black box definitions. Verilog UDPs are converted to gate-level functional netlists using ATPG primitives.

Write Patterns

```
Write Patterns
<filename> [-CELlnames <Internal | Verilog>]
[-COMpress <Gzip | Binary>]
[-EXClude <Setup | Repeat_setup | Patterns |
All>] [-FIrst <d>] [-Last <d>]
[-FOrmat <Binary | Stil | STIL99 |
VErilog_tables | VERILOG_Single_file | VHdl
| VHDL93 | Wgl| TStl2 | Ftdl | TDl91 |
WGL_Flat>]
[-CONfig_file <cfilename>] [-Internal | -
EXTernal] [-Measure_forced_bidis]
[-NOCompaction] [-NOOverlap_load]
[-Order_pins] [-PAD_character <0 | 1 | X >]
[-Replace] [-SErial | -PARallel |
num_shifts]
[-SOrted | -REOrder <file_name> | -Type
<Basic_scan | Clock_on_measure |
FAst_sequential | FUll_sequential>]
[-SPlit <d>]
[-Verilog_testbench_name <<name>>] [-STil]
[-NOEXPAND_VEctor | -EXPAND_VEctor]  [-VCS]
[-MTI] [-XL] [-NC]
```

This command writes ATPG patterns to a file in one of the supported formats: FTDL, STIL, TDL91, TSTL2, Verilog, VHDL, VHDL93, WGL,

WGL_FLAT, and the ATPG tool's proprietary binary. Optionally, the file may be written using one of two forms of compression: GZIP or binary.

Using FTDL, TDL91, TSTL2, or WGL_FLAT format causes TetraMAX to invoke a separate translation process, which opens a new window in which the translation process runs independently. The pattern formats FTDL, TDL91, and TSTL2 are write only; they cannot be read back in. This is important if you want to use the diagnostics capability of TetraMAX to assist in diagnosing device failures at the tester. If supporting diagnostics is part of your flow, you should also save the same patterns in a format that may be read back in, such as STIL or binary.

Write Simtrace

```
Write SImtrace <filename> [-Replace] [-Scan
<scanchainref>] [-Gate <gateidref>] [-Length
<d>] [-Verilog | -Dpv]
```

This command assists in debugging ATPG pattern miscompares found during Verilog simulation. It facilitates the creation of a Verilog module by adding simulation data to be used for comparison with TetraMAX.

Write Slow Path

```
Write SLow Path
<filename> [-Replace] [-Compress <Gzip |
Binary>]
```

This command writes a file of **`add_slow_path`** commands. This file can be used to reinstate the set of slow paths that are currently defined at the time the file is written.

Write TRC

```
Write Trc_file
<file_name> <label... | -All> [-Replace]
```

This command writes a tester rule file in CTL format based on the defined defaults.

Index to Commands