set_dft_signal

NAME

set_dft_signal

Specifies the DFT signal types for DRC and DFT insertion.

SYNTAX

```
status set_dft_signal
     [-view existing_dft | spec]
     [-test_mode mode_name_list]
     -type signal_type
     [-port port_list]
     [-active_state active_state]
     [-timing timing]
     [-period period]
     [-hookup_pin hookup_pin]
     [-hookup_sense hookup_sense]
     [-internal_clocks none | single | multi]
     [-ctrl_bits ctrl_bits_list]
     [-pll_clock pll_clock]
     [-ate_clock ate_clock]
     [-differential_clock clock_port]
     [-connect_to object_list]
     [-connect_to_domain_rise clock_list]
     [-connect_to_domain_fall clock_list]
     [-usage use_type]
     [-associated_internal_clocks clock_pins_list]
     [-associated_clock associated_clock]
     [-exclude cell_list]
     [-codec decompressor_name]
```

Data Types

mode_name_list	string
signal_type	string
port_list	list
active_state	integer
timing	list
period	float
hookup_pin	list
hookup_sense	string
ctrl_bits_list	list
pll_clock	string
ate_clock	string
clock_port	string
object_list	list
clock_list	list
use_type	list
associated_clock	string
cell_list	list
decompressor_name	string

ARGUMENTS

-view existing_dft | spec

Indicates the view to which the specification applies. The following views are valid:

- **existing_dft** implies that the specification refers to the existing usage of a port. For example, when working with a design that is already DFT-inserted, the command to indicate that port SE is used as a scan-enable port is as follows:

  set_dft_signal -view existing_dft -port SE -type ScanEnable

- **spec** (the default) implies that the specification refers to ports that the tool must use during DFT insertion. For example, when preparing a design for DFT insertion, the command to specify that port SE is used as a scan-enable port for DFT insertion is as follows:

  set_dft_signal -view spec -port A -type ScanEnable

-test_mode ***mode_name_list***

Specifies the test mode(s) to which the specification applies. Test modes are created with the **define_test_mode** command. A value of **all** or **all_dft** causes the specification to apply to all test modes.

If the **-test_mode** option is not specified, the command applies to the current test mode, which is determined by the last **define_test_mode** or **current_test_mode** command executed. If no test modes have been defined, the command applies to the default test mode.

-type ***signal_type***

Specifies the signal type. The valid signal types are as follows:

- **Reset** is the asynchronous set or reset of the design.
- **Constant** is a continuously applied value to a port.
- **TestMode** is a continuously applied value to a test mode port that may have different values between test modes.
- **TestData** is the port used to apply scan data to a portion of the design that has had pre-DRC violations fixed by Autofix.
- **TestControl** is the port used to control the DFTMAX shift power groups feature.
- **ScanDataIn** is one of the scan inputs of the design.
- **ScanDataOut** is one of the scan outputs of the design.
- **ScanEnable** is the shift enable of a MUX-D scan design.
- **ScanClock** is a clock that performs both scan shift and capture in a MUX-D scan design. (This type is equivalent to defining the clock as both the MasterClock and ScanMasterClock types; there is no CTL signal type called ScanClock.)
- **ScanMasterClock** is the clock responsible for scan shift. In an LSSD scan style, the ScanMasterClock is the A clock, and in a clocked scan style it is the shift clock.
- **ScanSlaveClock** is the B clock in an LSSD scan style.
- **MasterClock** is the clock responsible for capture. In an LSSD scan style the MasterClock is the system clock, and in a clocked scan style it is the capture clock.

In general, the MasterClock is used for referring to the capture clock and ScanMasterClock for referring to the shift clock. In the LSSD scan style, the two shift clocks are specified as ScanMasterClock and ScanSlaveClock, and only the capture clock is specified as MasterClock.

In the MUX-D scan style, the same clock provides both shift and capture functionality. In this case, use the

ScanClock signal type, which simultaneously specifies both MasterClock and ScanMasterClock types. To create a shift-only clock in the MUX-D scan style, use the ScanClock type, then apply the **add_pi_constraint** command in TetraMAX.

To specify the sense for an internal hookup-only pin (no port is associated with the hookup pin), use the **-active_state** option.

The following signal types are used in on-chip clocking (OCC) controller flows:

- **Oscillator** specifies the output pin of the on-chip clock generator or the output pins of the user-instantiated on-chip clock controller. In both cases, the **-hookup_pin** option must be used. This is a constantly running clock signal that never turns off.
- **RefClock** is a constantly running external reference clock. The clock can have timing which differs from the test default period.
- **pll_reset** is the asynchronous signal resetting the OCC logic.
- **pll_bypass** is the signal making the OCC logic transparent when the active value is applied.

The following signal types can be used in core-wrapping flows:

- **wrp_shift** specifies the scan shift signal for wrapper cells.
- **wrp_clock** specifies the clock signal for dedicated wrapper cells.
- **input_wrp_shift** specifies the scan shift signal for input wrapper cells in the maximized reuse flow. It can be used only with the **-connect_to** option.
- **output_wrp_shift** specifies the scan shift signal for output wrapper cells in the maximized reuse flow. It can be used only with the **-connect_to** option.
- **wrp_ded_capture_in** specifies the input capture signal in the maximized reuse flow. It is used for input dedicated wrapper cells when creating a wrapped core for hierarchical wrapping flows. It is also used for input shared wrapper cells in custom core wrapping flows.
- **wrp_ded_capture_out** specifies the output capture signal in the maximized reuse flow. It is used for output dedicated wrapper cells when creating a wrapped core for hierarchical wrapping flows. It is also used for output shared wrapper cells in custom core wrapping flows.
- **wrp_ded_capture_inout** specifies the input/output capture signal in the maximized reuse flow. It is used only for shared wrapper cells which are connected to both input and output ports in custom core wrapping flows.
- **wrp_safe_in** specifies the safe control signal for input wrapper cells in the maximized reuse flow. This signal is used when creating a wrapped core for hierarchical wrapping flows.
- **wrp_safe_out** specifies the safe control signal for output wrapper cells in the maximized reuse flow. This signal is used when creating a wrapped core for hierarchical wrapping flows.
- **wrp_td_test** specifies the signal used in core wrapping to aid transistion delay testing. The signal controls the hold state of the wrapper cell in capture. The cell can either hold the state or toggle in capture based on this signal. This signal is used only in custom core wrapping flows.

The following signal types can be used to define IEEE 1500 controller signals in IEEE 1500 flows:

- **WSI** specifies the scan-in signal.
- **WRSTN** specifies the reset signal.
- **WRCK** specifies the clock signal.
- **CaptureWR** specifies the capture-enable signal.
- **ShiftWR** specifies the shift-enable signal.

- **UpdateWR** specifies the update-enable signal.
- **SelectWIR** specifies the shift-path selection signal, which selects between the instruction register and data registers.
- **WSO** specifies the scan-out signal.

The following signal types can be used to define LogicBIST signals in LogicBIST self-test flows:

- **lbistEnable** specifies the signal that enables self-test mode. This also enables any wrapper or test-point logic associated with the self-test mode.
- **lbistStart** specifies the signal that begins self-test operation.
- **lbistStatus_1** and **lbistStatus_0** specify: 00 when idle; 01 when running; 10 when test complete and passed; and 11 when test complete and failed.
- **lbistSeedValue** specifies signals of a bus that provides the BIST seed value.
- **lbistSignatureValue** specifies signals of a bus that provides the BIST signature value.
- **lbistPatternCount** specifies signals of a bus that provides the pattern count used for BIST.
- **lbistShiftLength** specifies the signals of a bus that provide the number of shift cycles per pattern for each BIST pattern.
- **lbistBurnInEnable** specifies the signal that enables burn-in operation.
- **lbistBurnInStopOnFail** specifies whether burn-in operation should stop or continue if self-test fails.

The following signal type can be used in the shared codec I/O flow:

- **codec_enable** specifies a signal that enables a codec when the shared controls feature is used. It must be driven by a hookup pin; port-driven signals are not supported.

The following signal types can be used to define PCO wrapper chain signals in the power controller override (PCO) flow:

- **pco_shift_clk** specifies the shift clock signal.
- **pco_update_clk** specifies the update clock signal.
- **pco_override** specifies the override signal.

The following signal types can be used in BSD synthesis and integration:

- **tdi** specifies the TDI port or bsd reg tdi access pin.
- **tdo** specifies the TDO port or bsd reg tdo access pin.
- **tdo_en** specifies the TDO port enable pin.
- **tck** specifies the TCK port.
- **tms** specifies the TMS port.
- **trst** specifies the TRST port.
- **bsd_shift_en** specifies the register access pin to be hooked up to TAP shift_dr pin when the instruction that selects the register is active.
- **bsd_capture_en** specifies the register access pin to be hooked up to TAP sync_capture_en pin when the instruction that selects the register is active. This signal is also used in core integration.
- **bsd_capture_dr** specifies the register access pin to be hooked up to TAP Capture-DR state on the negative edge of TCK when the instruction that selects the register is active.

- **bsd_update_en** specifies the register access pin to be hooked up to TAP sync_update_dr pin when the instruction that selects the register is active. This signal is also used in core integration.
- **bsd_update_dr** specifies the register access pin to be hooked up to TAP Update-DR state on the negative edge of TCK when the instruction that selects the register is active.
- **capture_clk** specifies the register access pin to be hooked up to TCK/clock_dr pin when the instruction that selects the register is active.
- **update_clk** specifies the register access pin to be hooked up to TCK/update_dr pin when the instruction that selects the register is active.
- **inst_enable** specifies the register access pin to be held active when the instruction that selects the register is active.
- **bist_enable** specifies the register access pin to be held active when the instruction that selects the register is active and TAP is in Run-Test-Idle state.
- **bist_clk** specifies the register access pin to be hooked up to TCK when the instruction that selects the register is active and TAP is in Shift-DR state. This signal is also used in core integration.
- **bsd_reset** specifies the register access pin to be hooked up to the Test-Logic-Reset FSM state signal when the instruction that selects the register is active. When the bsd_reset signal is not part of any instruction's test data register definition, it is hooked up directly to the Test-Logic-Reset state signal.
- **bsd_test_logic_reset** specifies the Test-Logic-Reset TAP FSM state signal.
- **bsd_run_test_idle** specifies the Run-Test-Idle TAP FSM state signal.
- **bsd_select_dr_scan** specifies the Select-DR TAP FSM state signal.
- **bsd_capture_dr** specifies the Capture-DR TAP FSM state signal.
- **bsd_shift_dr** specifies the Shift-DR TAP FSM state signal.
- **bsd_exit1_dr** specifies the Exit1-DR TAP FSM state signal.
- **bsd_pause_dr** specifies the Pause-DR TAP FSM state signal.
- **bsd_exit2_dr** specifies the Exit2-DR TAP FSM state signal.
- **bsd_update_dr** specifies the Update-DR TAP FSM state signal.
- **bsd_select_ir_scan** specifies the Select-IR TAP FSM state signal.
- **bsd_capture_ir** specifies the Capture-IR TAP FSM state signal.
- **bsd_shift_ir** specifies the Shift-IR TAP FSM state signal.
- **bsd_exit1_ir** specifies the Exit1-IR TAP FSM state signal.
- **bsd_pause_ir** specifies the Pause-IR TAP FSM state signal.
- **bsd_exit2_ir** specifies the Exit2-IR TAP FSM state signal.
- **bsd_update_ir** specifies the Update-IR TAP FSM state signal.
- **bsd_mode_in** is the mode signal for input BSR cells.
- **bsd_mode_out** is the mode signal for output BSR cells.
- **bsd_mode1_inout** is the mode signal for mode1 of BC_7 BSR cells.
- **bsd_mode2_inout** is the mode signal for mode2 of BC_7 BSR cells.

-port ***port_list***

Indicates the list of ports on which to apply the specifications.

-active_state *active_state*

Specifies the active states for the following signal types:

```
ScanEnable
Reset
Constant
TestMode
pll_reset
pll_bypass
```

The active state can be 0 or 1 and it specifies the active sense of the port (high or low) or an internal hookup only pin (no port is associated with the hookup pin).

Wildcards and collections are supported.

-timing *timing*

Specifies the rise time and fall time for clocks. For example:

```
set_dft_signal -view existing_dft -type ScanClock \
  -port CLK -timing [list 45 55]
```

-period *period*

Specifies the period of the on-chip clocking (OCC) reference clock. The period value is a floating point number in nanoseconds. When **-period** is used, the values specified after **-timing** indicate the leading edge and the trailing edge of the reference clock. The **-period** option is valid only on signals of the RefClock type.

-hookup_pin *hookup_pin*

Specifies the pin to which signals are to be connected. By default, the **insert_dft** command connects wires to the core side of identified signal ports, jumping pads and buffers as needed. The **-hookup_pin** argument overrides this behavior and instructs **insert_dft** to connect wires to the specified pin. The pin can be either a leaf pin or hierarchical pin. Validation ensures that the pin direction (driver or load) is consistent with the signal type. Verify that a specific access pin is associated with only one design port.

When the **-type** is a scan clock and **-view** is set to **existing_dft**, the **-hookup_pin** argument allows you to specify internal pins for the scan clocks along with their associated top-level clock port specified with **-port**. The resulting protocol is accurate and complete with real top-level clocks.

If hookup pins are specified, only one port can be specified in the *port_list*.

However, if the internal pins flow is enabled (**set_dft_drc_configuration -internal_pins**), a hookup pin can be specified without specifying a port. The hookup pin specified should be either on a black box or should have a proper global driver. If a proper global driver is not found, **insert_dft** will not stitch to the specified hookup pin.

-hookup_sense *hookup_sense*

Specifies the hookup sense for the hookup pin. This option is valid only when a single hookup pin or a port is specified. The valid values are **inverted** and **non_inverted**. The default is **non_inverted**.

-internal_clocks none | single | multi

Specifies the setting for an internal clock. An internal clock is defined as an internal signal driven by a multiplexer (or multiple input gate) output pin. This option applies only to the multiplexed flip-flop scan style and it is ignored for other scan styles.

Note that the output of clock gating cells that are introduced by the Synopsys Power Compiler will not be treated as internal clocks, even if the **-internal_clocks** option is enabled.

The **-internal_clocks** option can be set to the following values:

- **single** specifies that **insert_dft** treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **single** value instructs the tool to stop at the first buffer or inverter driving the flip-flops clock.
- **none** (the default) specifies that **insert_dft** does not treat internal clocks as separate clocks.
- **multi** specifies that **insert_dft** treats any internal clocks in the design as separate clocks for the purpose of scan chain architecting. The **multi** value instructs the tool to jump over buffers and inverters, stopping at the first multi-input gate driving the flip-flops clock.

-ctrl_bits ***ctrl_bits_list***

Lists triplets that specify the sequence of control bits needed to enable the propagation of the clock generator outputs. This option is used in the on-chip clocking (OCC) flow. The first element of each triplet is the cycle number (integer) indicating when the clock signal will be propagated. The second element is the pin name of the control bit (a valid design hierarchical pin name). The third element is the active value (0 or 1) of the control bit. For example:

```
set_dft_signal -type Oscillator -hookup_pin pll_controller/clk \
 -pll_clock pll_i/pll_clk -ate_clock ate_clk \
 -ctrl_bits [list \
   0 clk_chain_i/clk_ctrl_data[0] 1 \
   1 clk_chain_i/clk_ctrl_data[1] 1 \
   2 clk_chain_i/clk_ctrl_data[2] 1] \
 -view existing_dft
```

-pll_clock ***pll_clock***

Specifies the port name of the pll clock. This option is used in the on-chip clocking (OCC) flow.

-ate_clock ***ate_clock***

Specifies the port name of the ATE clock. This option is used in the on-chip clocking (OCC) flow.

-differential_clock ***clock_port***

Specifies the pin name of a differential clock. For example, if clk_m is the minus side of clk_p:

```
set_dft_signal -view existing_dft -type ScanClock \
 -port clk_m -differential {clk_p}
```

Note that the minus side will track all of the properties of the plus side with an inverted polarity. A complete example for clk_m and clk_p is as follows:

```
set_dft_signal -view existing_dft -type Oscillator \
  -port clk_p
set_dft_signal -view existing_dft -type ScanClock \
  -port clk_p -timing [list 45 55]
set_dft_signal -view existing_dft -type ScanClock \
 -port clk_m -differential {clk_p}
```

-connect_to ***object_list***

Specifies that the DFT signal should be connected from the port or hookup pin to only the specified design objects.

You can use the **-connect_to** option to define ScanEnable, TestMode, or wrp_shift signals that should only be connected to a specific set of design objects. These are known as object-specific signal definitions. Depending on the signal type, only certain design object types can be specified and certain signal usages are supported. For the different signal types and usages defined with the **-type** and **-usage** options, the allowed object types for the **-connect_to** option are:

- **-type ScanDataIn**
  - **pin** (ScanDataIn pins of CTL-modeled cores)
- **-type ScanDataOut**

    - **pin** (ScanDataOut pins of CTL-modeled cores)
- **-type ScanEnable -usage scan**
    - **cell** (leaf scan cells)
    - **cell** (hierarchical cell containing scan cells)
    - **design** (designs containing scan cells)
    - **clock** (scan cells clocked by specified clocks)
    - **pin** (ScanEnable pins of CTL-modeled cores)
- **-type ScanEnable | TestMode -usage clock_gating**
    - **cell** (clock-gating cells)
    - **cell** (hierarchical cell containing clock-gating cells)
    - **design** (designs containing clock-gating cells)
    - **clock** (test clocks gated by clock-gating cells)
    - **pin** (ScanEnable or TestMode pins of CTL-modeled cores)
- **-type wrp_shift | input_wrp_shift | output_wrp_shift**
    - **cell** (leaf wrapper cells) - highest priority
    - **pin** (specified wrp_shift pins of CTL-modeled cores)
    - **cell** (wrp_shift pins for all wrapper segments in CTL-modeled cores)
    - **port** (wrapper cells associated with specified ports)
    - **clock** (wrapper cells clocked by specified clocks) - lowest priority

For the **clock_gating** usage, you can specify ScanEnable or TestMode signal definitions, depending on the type of clock-gating control signal (scan-enable or test-mode) used by the specified objects.

Pins of CTL-modeled cores are not supported when performing simple HSS integration of standard scan cores.

When you specify a clock-domain-based signal specification by specifying clocks, they must be defined as valid test clocks with the **set_dft_signal** command. The propagation of the test clock determines the scope of the specification; the propagation of any underlying functional clocks at the clock source is not considered.

When using clock-domain-based signal specifications with CTL-modeled cores, the following requirements must be observed during core creation to ensure that the CTL models contain clock-domain information:

- Core-level ScanEnable signals must be defined using the **-usage** option of the **set_dft_signal** command. This ensures that the core's internal scan-enable signal is not used outside its intended usage.
- Core-level ScanEnable signals defined with a usage of **clock_gating** must also be defined as domain-specific signals using the **-connect_to***clock_list* option of the **set_dft_signal** command. This ensures that clock-specific clock-gating annotations are included in the CTL model.

If the cores do not contain clock-domain information, you can directly specify the ScanEnable pins of the CTL-modeled core instead.

The following example defines two clock-domain-based ScanEnable signals:

```
set_dft_signal -view exist -type ScanClock -timing {45 55} \
  -port {CLK1 CLK2}
```

```
set_dft_signal -type ScanEnable -port SE1 -usage scan \
  -connect_to CLK1
set_dft_signal -type ScanEnable -port SE2 -usage scan \
  -connect_to CLK2
```

The following example defines a ScanEnable signal to be used only as the clock-gating control signal for a specific block:

```
set_dft_signal -type ScanEnable -port SE_CG_IP -usage clock_gating \
  -connect_to UIP_CORE
```

The **-connect_to** option is also used in the hierarchical on-chip clocking (OCC) flow to specify the correspondence between the top-level ATE clock port and the core-level ATE clock pins. For existing top-level connections, use the **-view existing** option. For connections to be made by DFT insertion, use the **-view spec** option. For example,

```
set_dft_signal -view spec -port TOP_CLK -type MasterClock \
  -connect_to {CORE1/ATECLK CORE2/ATECLK} -timing {45 55}
```

Clock-domain-based wrapper shift signals are supported only in the maximized reuse flow.

-connect_to_domain_rise *clock_list*

Specifies a clock-domain-based ScanEnable specification that applies only to rising-edge scan cells.

This option works the same as specifying a test clock with the **-connect_to** option, except that it applies only to scan cells clocked by the rising edge of the specified test clocks. It is valid only with **-type ScanEnable** and **-usage scan** options.

This option is not supported for wrapper shift signals.

-connect_to_domain_fall *clock_list*

Specifies a clock-domain-based ScanEnable specification that applies only to falling-edge scan cells.

This option works the same as specifying a test clock with the **-connect_to** option, except that it applies only to scan cells clocked by the falling edge of the specified test clocks. It is valid only with **-type ScanEnable** and **-usage scan** options.

This option is not supported for wrapper shift signals.

-usage *use_type*

Specifies the usage of the signal. You must also specify **-view spec** with this option. This option specifies that **insert_dft** is to use the specified signal for that purpose only.

The signal must be defined with suitable valid values when using it to connect design elements using the **set_dft_connect** command. The valid values for *use_type* are as follows:

- **clock_gating** specifies that the signal is to be connected to the test pin of identified clock gating cells during **insert_dft**. This usage is valid only for **-type ScanEnable** and **-type TestMode**.

  If clock-gating test pin connection is disabled by setting the **-connect_clock_gating** option of the **set_dft_configuration** command to **disable**, this signal is not used.

- **scan** specifies that the signal is to be connected to the enable of scan elements during **insert_dft**. This usage is valid only for **-type ScanEnable**.

Multiple usages can be specified for a signal by specifying them as a list. For example, a ScanEnable can be used to connect to test pins of clock gating cells as well as enabling scan elements.

For example,

```
set_dft_signal -view spec -port SE \
 -type ScanEnable -usage {scan clock_gating}
```

-associated_internal_clocks *clock_pins_list*

Specifies a list of internal pins associated to be associated with a clock source port.

The tool assumes that the port-driven clock signal is also driven at these internal pins, with the same timing waveform and polarity. The fanout from each internal clock pin is treated as a separate clock domain when architecting scan chains, and the pin names are shown as the clock names by the **preview_dft** and **insert_dft** commands.

Only leaf pins can be specified; hierarchical pins are not supported.

For example,

```
set_dft_signal -type ScanClock -port CLK \
 -associated_internal_clocks [list U1/int_clk U2/int_clk1 U1/int_clk2] \
 -view existing
```

During pre-DFT DRC, the tool does not check for logical connectivity between the port and internal clock pins. This provides a method to bypass black boxes or other logic with unknown functionality in the clock network.

The connectivity is checked in post-DFT DRC.

-associated_clock ***associated_clock***

Specifies the pin or port name of a single clock that you want to associate with a specified scan enable signal.

For example,

```
set_dft_signal -view spec -type ScanEnable \
  -port SE1 -associated_clock CLK1
```

The port or pin must be previously defined using the **-port** or **-hookup_pin** option of the **set_dft_signal** command, respectively.

This option requires that the **-domain_based_scan_enable** option of the **set_scan_configuration** command be set to **true**.

-exclude ***cell_list***

Specifies a list of objects to be excluded from an object-specific DFT signal definition made with the **-connect_to**, **-connect_to_domain_rise**, or **-connect_to_domain_fall** option.

In some cases, an object-specific connection specification might require certain objects to be excluded, such as excluding specific leaf cells from a hierarchical block. In this case, you can specify the objects to exclude by using the **-exclude** option. The allowed object types for the **-exclude** option are:

- **-type ScanEnable -usage scan**
  - **cell** (leaf scan cells)
  - **cell** (hierarchical cell containing scan cells)
  - **design** (designs containing scan cells)
- **-type ScanEnable | TestMode -usage clock_gating**
  - **cell** (clock-gating cells)
  - **cell** (hierarchical cell containing clock-gating cells)
  - **design** (designs containing clock-gating cells)

This option excludes the specified objects from the current signal's specification, but it does not prevent those objects from being connected to other signals.

This option can only be used together with the **-connect_to**, **-connect_to_domain_rise**, or **-connect_to_domain_fall** option.

For example:

```
set_dft_signal -type ScanEnable -port SE1 -usage scan \
-connect_to {CLK1} -exclude UIP_BLOCK
```

-codec ***decompressor_name***

Specifies the codec to be controlled by a signal defined with the **-type codec_enable** option. Reference the codec's decompressor using the *cell_name*:*decompressor_name* syntax.

By default, the tool chooses a codec for the signal to control.

DESCRIPTION

The **set_dft_signal** command can be used to specify one or more primary input or output ports as DFT signals. It can be used either to describe the existing usage or to prescribe the usage during implementation.

A DFT signal has a port and a signal type. You can specify multiple DFT signals, but you can specify only one signal type. Port directions must be consistent with the signal type.

You can use the **set_dft_signal** command to declare different DFT signals for different test modes. For example, each test mode can have different scan-in and scan-out ports, or different constant, set, or reset signals. However, all test modes share the same scan-enable signals. You cannot specify different scan-enable signals for different test modes.

The **set_dft_signal** commands are not incremental. A **set_dft_signal** command that identifies a port overwrites any previous **set_dft_signal** command that identifies the same port.

To review your DFT signal specifications, use the **report_dft_signal** command. To remove a specification, use the **remove_dft_signal** command. To implement the DFT signal, use the **insert_dft** command.

Descriptive specification (using **-view existing_dft**) will invalidate test protocol and details about DFT structures.

SEE ALSO

current_design(2)
dft_drc(2)
insert_dft(2)
preview_dft(2)
remove_dft_signal(2)
report_dft_signal(2)
set_dft_configuration(2)
set_dft_drc_configuration(2)
write(2)