UPF Commands Supported by VCS NLP

1 Introduction

This document describes the Unified Power Format (UPF) commands supported by VCS Native Low Power (NLP). These commands are used to specify the power intent for low power designs.

2 UPF Commands Supported by VCS NLP

The following section describe the IEEE 1801 UPF commands supported by VCS NLP. For more information about using the UPF commands, see the IEEE 1801 (UPF) specification:

- Basic Power Network Commands
- Isolation Commands
- Retention Commands
- Logic Editing Commands
- Query Commands
- Simulation/Verification Extension
- Utility Commands
- Other Supported Power Intent Commands

2.1 Basic Power Network Commands

The basic power commands define the power domains of the design and the supply ports, supply nets, and power switches of each domain. The following are the basic power network commands supported by VCS NLP:

- associate_supply_set
- create_power_domain
- create_power_switch
- create_supply_port
- create_supply_net
- connect_supply_net
- create_supply_set
- set_domain_supply_net

2.1.1 associate_supply_set

The `associate_supply_set` command associates a supply set handle to another supply set. The `supply_set_name` specifies the name of the supply set that must be associated with the handle specified with the `-handle` option.

Syntax

```
associate_supply_set supply_set_name_list
   [ -handle supply_set_handle ]
```

2.1.2 create_power_domain

The `create_power_domain` command defines a power supply distribution network at the current scope (hierarchical level) or at the scope of a specified hierarchical instance.

Syntax

```
create_power_domain domain_name
   [-elements element_list]
   [-include_scope]
   [-available_supplies supply_set_sef_list]
   [-supply {supply_set_handle [supply_set_ref]}*]
   [-scope instance_name]
   [-update]
```

* Indicates that the item in curly braces can be repeated in the command.

2.1.3 create_power_switch

The `create_power_switch` command creates an instance of a power switch in the scope of a power domain. The switch has at least one input supply port and one output supply port.

Syntax

```
create_power_switch switch_name
   -output_supply_port {port_name [supply_net_name]}
   {-input_supply_port {port_name [supply_net_name]}}*
   {-control_port {port_name [net_name]}}*
   {-on_state {state_name input_supply_port    {boolean_function}}}*
   [-off_state {state_name {boolean_function}}]*
   [-supply_set supply_set_name]
   [-on_partial_state {state_name input_supply_port    {boolean_function}}]*
   [-ack_port {port_name net_name [{boolean_function}]}]*
   [-ack_delay {port_name delay}]*
   [-error_state {state_name {boolean_function}}]*
   [-domain domain_name]
```

* Indicates that the item in curly braces can be repeated in the command.

2.1.4 create_supply_port

The `create_supply_port` command creates a supply port at the current scope or at the scope of a power domain specified with the -domain option.

Syntax

```
create_supply_port port_name
   [-domain domain_name]
   [-direction <in | out | inout>]
```

2.1.5 create_supply_net

The `create_supply_net` command creates a supply net to supply power or ground to a power domain.

Syntax

```
create_supply_net net_name
   [-domain domain_name] [-reuse]
   [-resolve <unresolved | one_hot | parallel>]
```

2.1.6 connect_supply_net

The `connect_supply_net` command specifies an explicit connection of a supply net to a list of supply ports, thereby overriding any implicit connections that might otherwise apply.

Syntax

```
connect_supply_net net_name
   [-ports list]
   [-vct vct_name]
```

2.1.7 create_supply_set

The `create_supply_set` command creates a supply set at the current level of logic hierarchy. A supply set is a collection of supply nets. A supply set consists of the following functions:

- Power
- Ground
- pwell
- nwell

Syntax

```
create_supply_set set_name
   [-function {func_name [net_name]}]*
   [-update]
```

* Indicates that the item in curly braces can be repeated in the command.

2.1.8 set_domain_supply_net

The `set_domain_supply_net` command specifies the primary power net and primary ground net for an existing power domain.

Syntax

```
set_domain_supply_net domain_name
   -primary_power_net supply_net_name
   -primary_ground_net supply_net_name
```

2.2 Isolation Commands

The isolation commands specify the strategy for inserting isolation cells at the outputs of switched (power-down) domains. Following are the isolation commands supported by VCS NLP:

- set_isolation
- set_isolation_control
- map_isolation_cell

2.2.1 set_isolation

The `set_isolation` command specifies the isolation strategy for a power domain and the elements in that domain on which the strategy is applied.

Syntax

```
set_isolation isolation_name
```

```
   -domain ref_domain_name
   [-elements element_list]
   [-exclude_elements exclude_list]
   -source source_supply_ref
   -sink sink_supply_ref
   [-diff_supply_only <TRUE | FALSE>]
   [-applies_to <inputs | outputs | both>]
   [-isolation_power_net net_name]
   [-isolation_ground_net net_name]
   [-no_isolation]
   [-location <automatic | self | fanout | parent >]
   [-clamp_value {<0 | 1 | Z | latch>*}]
   [-isolation_signal signal_list <>]
   [-isolation_sense <high | low | {<high | low>*}>]]
   [-isolation_supply_set supply_set_list]
   [-isolation_sense {<high | low>*}]
   [-update]
```

* Indicates that the item in curly braces can be repeated in the command.

2.2.2 set_isolation_control

The `set_isolation_control` command allows the specification of the isolation control signal and the logical sense of that signal.

Syntax

```
set_isolation_control isolation_name
   -domain domain_name
   -isolation_signal signal_name
   [-isolation_sense <high | low>]
   [-location <self | parent| fanout>]
```

2.2.3 map_isolation_cell

The `map_isolation_cell` command maps a particular isolation strategy to a library cell or range of library cells.

Syntax

```
map_isolation_cell isolation_name
   -domain domain_name
   [-lib_cell_type lib_cell_type]
   [-lib_model_name model_name {-port {port_name  net_name}}*]
```

* Indicates that the item in curly braces can be repeated in the command.

2.3 Retention Commands

The retention commands specify the strategy for inserting retention cells inside switched (power-down) domains. The following are the retention commands supported by VCS NLP:

- set_retention
- set_retention_control
- set_retention_elements

2.3.1 set_retention

The `set_retention` command specifies the registers in the domain that need to be implemented as retention registers and identifies the save and restore signals for the retention functionality.

Syntax

```
set_retention retention_name
   -domain domain_name
   [-elements element_list]
   [-retention_power_net net_name]
   [-retention_ground_net net_name]
   [-retention_supply_set ret_supply_set]
   [-no_retention]
    [-save_signal {logic_net <high | low | posedge | negedge>}
   -restore_signal {logic_net <high | low | posedge |    negedge>}]
   [-save_condition {{boolean_function}}]
   [-restore_condition {{boolean_function}}]
   [-retention_condition {{boolean_function}}]
   [-use_retention_as_primary]
   [-parameters {<RET_SUP_COR | NO_RET_SUP_COR | SAV_RES_COR | NO_SAV_RES_COR> *}]
   [-transitive <TRUE | FALSE>]
   [-update]
```

* Indicates that the item in curly braces can be repeated in the command.

2.3.2 set_retention_control

The `set_retention_control` command allows the specification of the retention control signal and the logical sense of that signal.

Note: The `set_retention_control` command is deprecated as per UPF 2.0 LRM (IEEE Std 1801-2013).

Syntax

```
set_retention_control retention_name
   -domain domain_name
   -save_signal {{net_name <high | low | posedge | negedge>}}
   -restore_signal {{net_name <high | low | posedge |   negedge>}}
   [-assert_r_mutex {{net_name <high | low | posedge |   negedge>}}]*
   [-assert_s_mutex {{net_name <high | low | posedge |   negedge>}}]*
   [-assert_rs_mutex {{net_name <high | low | posedge |  negedge>}}]*
```

* Indicates that the item in curly braces can be repeated in the command.

2.3.3 set_retention_elements

The `set_retention_elements` command defines a list of state elements whose collective state shall be maintained coherently if retention is applied to any of these elements in the list.

Syntax

```
set_retention_elements retention_list_name
   [-elements element_list]
```

2.4 Logic Editing Commands

The logic editing commands create logic nets and logic ports and make connections between these nets and ports, irrespective of the power network. The following are the logic editing commands supported by VCS NLP:

- create_logic_net
- create_logic_port
- connect_logic_net

2.4.1 create_logic_net

The `create_logic_net` command creates a new logic net in the active scope.

Syntax

```
create_logic_net net_name
```

2.4.2 create_logic_port

The `create_logic_port` command creates a new logic port in the active scope and specifies the port direction (`in` for input and `out` for output).

Syntax

```
create_logic_port port_name
   [-direction <in | out>]
```

2.4.3 connect_logic_net

The `connect_logic_net` command connects a specified logic net to one or more specified logic ports.

Syntax

```
connect_logic_net net_name
   -ports port_list
   [-reconnect]
```

2.5 Query Commands

Each query commands returns a list of objects in the design that match the criteria you specify in the command.

Note: Query commands are not supported in the UPF. You can specify query commands in a configuration file and pass it to the `vcs` command line using the `-lpa_bind <filename>` option.

The following are the query commands supported by VCS NLP:

- find_objects
- query_cell_instances
- query_cell_mapped
- query_pst
- query_pst_state
- query_power_switch
- query_design_attributes
- query_hdl2upf_vct
- query_isolation
- query_power_domain
- query_retention

- query_retention_control
- query_supply_net

2.5.1 find_objects

The `find_objects` command finds instances, nets, or ports defined in the logic hierarchy in a specified scope and returns a list of object names that match a given search pattern.

Syntax

```
find_objects scope
   -pattern search_ pattern
   [-object_type <inst | port | net>]
   [-direction <in | out | inout>]
   [-transitive [<TRUE | FALSE>]]
   [-regexp | -exact]
   [-ignore_case]
   [-non_leaf | -leaf_only]
```

2.5.2 query_cell_instances

The `query_cell_instances` command returns a list of instance names for all instances of a given reference cell in the current scope of the design. You can optionally restrict the query to a given power domain.

Syntax

```
query_cell_instances cell_name
   [-domain domain_name]
```

2.5.3 query_cell_mapped

The `query_cell_mapped` command returns the reference cell name of a given cell instance.

Syntax

```
query_cell_mapped instance_name
```

2.5.4 query_pst

The `query_pst` command returns information about previously defined power state tables in the active scope. The command `query_pst *` returns a list of the power state table names.

Syntax

```
query_pst table_name
   [-detailed]
```

2.5.5 query_pst_state

The `query_pst_state` command returns information about the states that have been previously defined for a specified power state table.

Syntax

```
query_pst_state state_name
   -pst table_name
   [-detailed]
```

2.5.6 query_power_switch

The `query_power_switch` command returns information about previously defined power switches.

Syntax

```
query_power_switch switch_name
   [-detailed]
```

2.5.7 query_design_attributes

The `query_design_attributes` command queries attribute information for a specified `element_name` or `model_name`.

Syntax

```
query_design_attributes
    <-element element_name | -model model_name>
   [-detailed]
```

2.5.8 query_hdl2upf_vct

The `query_hdl2upf_vct` command can list and query any previously defined value conversion table (VCT).

Syntax

```
query_hdl2upf_vct vct_name
   [-detailed]
```

2.5.9 query_isolation

The `query_isolation` command can list the previously defined isolation strategies for the specified power domain domain_name.

Syntax

```
query_isolation isolation_name
   -domain ref_domain_name
   [-detailed]
```

2.5.10 query_power_domain

The `query_power_domain` command queries the parameters of a power domain.

Syntax

```
query_power_domain domain_name
   [-detailed]
```

2.5.11 query_retention

The `query_retention` command lists the previously defined retention strategies for the specified power domain domain_name.

Syntax

```
query_retention retention_name
   -domain domain_name
   [-detailed]
```

2.5.12 query_retention_control

The `query_retention_control` command queries the retention control information for a retention strategy.

Syntax

```
query_retention_control retention_name
   -domain domain_name
   [-detailed]
```

2.5.13 query_supply_net

The `query_supply_net` command returns the information about a previously created supply net.

Syntax

```
query_supply_net net_name
   [-domain domain_name]
   [-detailed]
```

2.6 Simulation/Verification Extension

The simulation and verification extension commands support low-power checking by simulation and verification tools. The following are the simulation and verification extension commands supported by VCS NLP:

- set_design_top
- set_partial_on_translation

2.6.1 set_design_top

The `set_design_top` command specifies the root of the design.

Syntax

```
set_design_top root
```

2.6.2 set_partial_on_translation

The `set_partial_on_translation` command specifies the translation of the `PARTIAL_ON` state to either `FULL_ON` or `FULL_OFF` to evaluate the power state of supply sets and power domains.

Syntax

```
set_partial_on_translation
   [FULL_ON | OFF]
```

2.7 Utility Commands

These are the commands to load and execute UPF commands from a file, to write a set of UPF commands to a file, and to set the hierarchical scope for subsequent UPF commands. The following are the utility commands supported by VCS NLP:

- load_upf
- set_scope
- set_design_attributes
- set_port_attributes

2.7.1 load_upf

The `load_upf` command executes the UPF commands in a specified file.

Syntax

```
load_upf upf_file_name
   [-scope instance_name]
```

2.7.2 set_scope

The `set_scope` command specifies the hierarchical scope of subsequent commands, including UPF commands.

Syntax

```
set_scope [instance]
```

2.7.3 set_design_attributes

The `set_design_attributes` command sets the attributes of one or more cells. The `-models` option specifies a list of design models on which the attributes are set.

Syntax

```
set_design_attributes
   [-models model_list | -elements element_list]
   {-attribute name value}*
```

* Indicates that the item in curly braces can be repeated in the command.

2.7.4 set_port_attributes

The `set_port_attributes` command specifies a collection of ports where the attributes must be set for the source or sink, for the interface of the power domains, when used with the set_isolation command.

Syntax

```
set_port_attributes
   [-model name]
   [-elements element_list]
   [-applies_to <inputs | outputs | both>]
   [-attribute {name value}]*
   [-clamp_value <0 | 1 | Z | latch>]
   [-driver_supply supply_set_ref]
   [-receiver_supply supply_set_ref]
   [-pg_type pg_type_value]
   [-related_power_port supply_port_name]
   [-related_ground_port supply_port_name]
   [-related_bias_ports supply_port_name_list]
   [-repeater_supply supply_set_ref]
   [-feedthrough]
   [-unconnected]
```

* Indicates that the item in curly braces can be repeated in the command.

2.8 Other Supported Power Intent Commands

Following are the other UPF commands supported by VCS NLP to specify power intent:

- bind_checker
- describe_state_transition
- set_design_top
- set_equivalent
- set_simstate_behavior

2.8.1 bind_checker

The `bind_checker` command inserts checker modules into a design without modifying the design code or introducing functional changes.

Syntax

```
bind_checker instance_name
   -module checker_name
   [-elements element_list]
   [-ports {{port_name net_name}*}]
   [-parameters {{param_name param_value}*}]
```

* Indicates that the item in curly braces can be repeated in the command.

2.8.2 describe_state_transition

The `describe_state_transition` command specifies the legality of a transition from one object's named power state to another.

Syntax

```
describe_state_transition transition_name
   -object object_name
   -from {from_list} -to {to_list} -paired {{from_state   to_state}*}}
   [-legal | -illegal]
```

* Indicates that the item in curly braces can be repeated in the command

2.8.3 set_design_top

The `set_design_top` command specifies the root of the design.

Syntax

```
set_design_top root
```

2.8.4 set_equivalent

The `set_equivalent` command declares that the specified supplies are electrically or functionally equivalent.

Syntax

```
set_equivalent
   [-function_only]
   [-nets supply_net_name_list]
   [-sets supply_set_name_list]]
```

2.8.5 set_simstate_behavior

The `set_simstate_behavior` command specifies the simstate behavior for models or instances.

Syntax

```
set_simstate_behavior <ENABLE | DISABLE>
   [-models list]
   [-elements element_list]
   [-exclude_elements exclude_list]
```

Copyright Notice and Proprietary Information

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.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/Company/Pages/Trademarks.aspx.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
690 E. Middlefield Road
Mountain View, CA 94043
www.synopsys.com