Conformal Low Power: Tcl Attributes and Usage

Product Version: Conformal 23.20
April 2024

Contents

Summary

Conformal Low Power provides Tcl attribute support. Users can explore the database for diagnosis, customize the report, and write rule filters by using the supported Tcl attributes. This guide explains about:

- How to know the supported attributes for any object.
- How to use the Tcl attributes to query the design and 1801 power intent information, and
- How to write the rule filters, which can filter out a portion of the reported violations.

Commands

Report all available Tcl attributes

```
report_tcl_attributes
[-RTL_OBJ] [-LIB_OBJ]
[-CPF_OBJ] [-1801_OBJ] [-PIA_OBJ]
[-SDC_OBJ]
[-LIB_COMPARE_OBJ] [-PIC_OBJ]
   [-RULE_OBJ] [-RULE]
   [<obj_name* ...>]
[-XML <filename>]
```

To retrieve the value of a specific attribute, use

```
get_attribute <object | object_list>
[<-all | attribute_name>] [<var_name>]
```

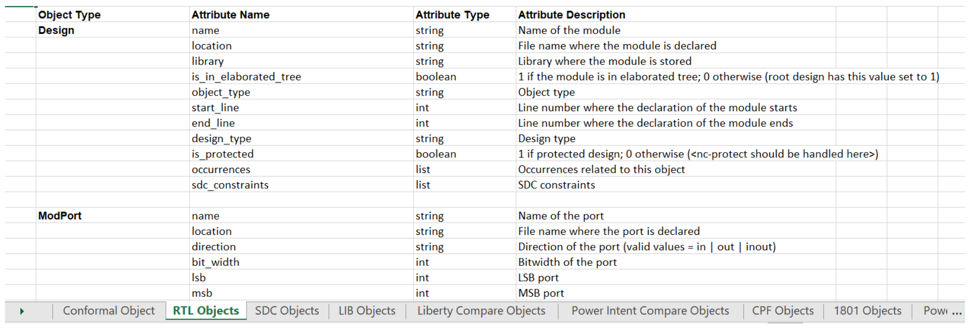
To review the 1801 checker occurrence attributes, add the **`-attributes`** option for the **`report_rule_check`** command.

```
report_rule_check -1801 -attributes
```

Report Tcl Attributes

Users can view the supported Tcl attributes by the **`report_tcl_attributes`** command. Adding the **`-xml`** option to the **`report_tcl_attributes -xml`** command can write out the attributes to an XML file which can be opened as an Excel spreadsheet.

```
TCL_SETUP> report_tcl_attributes -xml clp_tcl_attr.xml
```



Object Type	Attribute Name	Attribute Type	Attribute Description
Design	name	string	Name of the module
	location	string	File name where the module is declared
	library	string	Library where the module is stored
	is_in_elaborated_tree	boolean	1 if the module is in elaborated tree; 0 otherwise (root design has this value set to 1)
	object_type	string	Object type
	start_line	int	Line number where the declaration of the module starts
	end_line	int	Line number where the declaration of the module ends
	design_type	string	Design type
	is_protected	boolean	1 if protected design; 0 otherwise (<nc-protect should be handled here>)
	occurrences	list	Occurrences related to this object
	sdc_constraints	list	SDC constraints
ModPort	name	string	Name of the port
	location	string	File name where the port is declared
	direction	string	Direction of the port (valid values = in | out | inout)
	bit_width	int	Bitwidth of the port
	lsb	int	LSB port
	msb	int	MSB port

Conformal Object | RTL Objects | SDC Objects | LIB Objects | Liberty Compare Objects | Power Intent Compare Objects | CPF Objects | 1801 Objects | Pow...

RTL Objects And Supported Attributes

To view the supported attributes of a specific object, specify the object at the report command. For example, to view the supported design object attributes, specify **–rtl_obj**.

```
TCL_SETUP> report_tcl_attributes -rtl_obj
=====================================================================
=                           RTL Objects
=====================================================================
|       Object Type
|         Attribute Name  Attribute Type  Attribute Description
|
|       Design
|          name           string          Name of the module
|          location       string          File name where the module is declared
|          library        string          Library where the module is stored
…
|
|       ModPort
|          name           string          Name of the port
|          location       string          File name where the port is declared
…
|       Instance
|          name           string          Name of the instance
|          full_name      string          Hierarchical name of the instance from
root
|          ref_name       string          Reference (design or libcell) full_name
|          location       string          File name where the instance is
declared
|          is_leaf_cell   boolean         1 if reference is library cell or
primitive;
0 otherwise
```

Users can also specify the object type in the command. For example, to view the supported attributes of type **instance**:

```
TCL_SETUP> report_tcl_attributes instance
=====================================================================
=                                RTL Objects                        =
=====================================================================
|       Object Type
|         Attribute Name  Attribute Type   Attribute Description
|
|       Instance
|         name            string          Name of the instance
|         full_name       string          Hierarchical name of the instance from
root
|         ref_name        string          Reference (design or libcell) full_name
|         location        string          File name where the instance is
declared
|         library         library         Library where instance is declared
|         type            string          Type of the instance Conformal
identifies,
|                                         others should be "UNKNOWN"
|         is_leaf_cell    boolean         1 if reference is library cell or
primitive;
|                                         0 otherwise
|         is_dont_touch   boolean         1 if the instance is inferred as a
|                                         dont_touch; 0 otherwise
|         is_dont_use     boolean         1 if the instance is inferred as a
dont_use;
|                                         0 otherwise
|         is_pad_cell     boolean         1 if the instance inferred as a
pad_cell;
|                                         0 otherwise
|         bbox_type       string          Type of the bbox (valid values = timing
|                                         undefined | empty | unsupported |
notranslate)
…
```

Query Attribute and Value Of a Design Object

Users can also find out the supported attribute names of an object type with the **`get_attribute`** command.

Example 1: In the below example, it returns the supported attribute names for an attribute of the type instance.

```
TCL_SETUP> get_attribute [find -instance {ur1/ud1/u_fp0000}]
name full_name ref_name location library type is_leaf_cell is_dont_touch
is_dont_use
is_pad_cell bbox_type object_type is_bbox design_type start_line end_line parent
pow
er_domain is_clock_gating_cell clock_gating_cell_type has_mod_instance
is_protected
is_in_elaborated_tree is_sequential is_test_cell is_physical_cell
lowpower_cell_type
 is_std_cell retention_type clamp_value async_clamp_value matching_strategies
target
ing_strategies matching_strategies_and_ports targeting_strategies_and_ports
```

```
is_virtu
al ref_object pim_object is_soft_macro pg_nwell_type pg_pwell_type is_cdm_cell
is_mu
ltiple_stage_lsh occurrences 1801_design_attributes 1801_port_attributes
is_tool_gen
erated_name
```

Example 2: When a pin object is specified as an option, the attribute names of a pin object are returned.

```
TCL_SETUP> get_attribute [find -pin {ur1/u_iso/Ib}]
name ref_name full_name location direction bit_width lsb msb fanout fanin
is_clock
is_set_reset is_leaf_pin object_type bus_idx bus_name design_type upper_net
lower_net
start_line end_line parent power_domain active_phase type is_in_elaborated_tree
library
libcell function timing_arc test_cell_signal_type is_pad p_pg_type
physical_power_domain
is_analog is_unconnected alive_during_power_up alive_during_partial_power_down
crossings driver_supply_set receiver_supply_set supply_driver root_supply_driver
is_domain_boundary matching_strategies targeting_strategies is_isolated
off_to_on_ok
is_virtual pg_type ref_object pim_object supply_object is_lp_ctrl
constraint_value
is_reconnected reconnected_net is_soft_macro_pin occurrences
1801_design_attributes
1801_port_attributes is_constant constant_value p_is_constant p_constant_value
tied_value
is_undriven is_floating is_tool_generated_name
```

Example 3: To report all attribute names with their values, use the **-all** option.

The report returns all the attribute names with their values in a list format: **{<attribute_name> <value> <attribute_name> <value> ... }**. The report shows '*is_bbox 0*' as one of the attributes, which means that the instance is not black boxed.

```
TCL_SETUP> get_attribute [find -instance ur1/ud1/u_fp0000] -all
name u_fp0000 full_name ur1/ud1/u_fp0000 ref_name CLP_Generic_L_std/SC_BUF
location ./ve
rilog.template/top.pg.1001.v library CLP_Generic_L_std type BUF is_leaf_cell 1
is_dont_t
ouch 0 is_dont_use 0 is_pad_cell 0 bbox_type {} object_type instance is_bbox 0
design_type
revised start_line 43 end_line {} parent ur1/ud1 power_domain TDD
is_clock_gating_cell 0
clock_gating_cell_type {} has_mod_instance 0 is_protected 0 is_in_elaborated_tree
1
is_sequential 0 is_test_cell 0 is_physical_cell 0 lowpower_cell_type {}
is_std_cell 1 re
tention_type {} clamp_value {} async_clamp_value {} matching_strategies {}
```

```
targeting_str
ategies {} matching_strategies_and_ports {} targeting_strategies_and_ports {}
is_virtual
 0 ref_object CLP_Generic_L_std/SC_BUF pim_object {} is_soft_macro 0
pg_nwell_type {} pg
_pwell_type {} is_cdm_cell 0 is_multiple_stage_lsh 0 occurrences
CROSSING_OFF_TO_ON_ISO/
1 1801_design_attributes {{top_ports_have_anon_supply true}} 1801_port_attributes
{} is_
tool_generated_name 0
```

To query a specific attribute value, the user can easily specify the attribute name.

Example 4: To find the file name where the instance is declared, use:

```
TCL_SETUP> get_attribute [find -instance ur1/ud1/u_fp0000] location
./verilog.template/top.pg.1001.v
```

Example 5: To check if the pin is tied to a constant.

```
TCL_SETUP> get_attribute [find -pin g0/Y] is_constant
0
```

Example 6: To check if the pin has propagated 0 or 1 constant value.

```
TCL_SETUP> get_attribute [find -pin g0/Y] p_is_constant
1
```

Example 7: To check the tied constant value of a pin. An empty list is reported when the pin does not have a tied value.

```
TCL_SETUP> get_attribute [find -pin ur1/ud1/u_fp0000/A] tied_value
TCL_SETUP> get_attribute [find -pin g0/Y] tied_value
1
```

Example 8: To check the matching strategy of an isolation instance.

```
TCL_SETUP> get_attribute [find -instance g_stack_0__U1/ISO2] matching_strategies
g_stack_0__U1/PD_vdd.iso_R1
```

Query Attributes of a 1801 Power Intent Object

Users can also query the 1801 power intent object with the **`get_attribute`** command.

Example 1: The example below returns the supported attributes for the 1801 **`set_isolation`** commands (can use **`-isolation`** or **`-set_isolation`**)

```
TCL_SETUP> get_attribute [find -set_isolation]
object_type design_type type_name command filename lineno start_line parent
occurrences
name strategy_name elements exclude_elements domain source sink applies_to
```

```
name_prefix
name_suffix instance use_equivalence applies_to_boundary location transitive
handles
no_isolation force_isolation applies_to_clamp applies_to_sink_off_clamp
applies_to_source_off_clamp isolation_power_net isolation_ground_net
isolation_supply_set
isolation_signal isolation_sense clamp_value async_clamp_value async_set_reset
sink_off_clamp source_off_clamp diff_supply_only strategy_elements
implementable_elements
use_functional_equivalence is_implicit 1801_design_attributes
1801_port_attributes
```

Example 2: To get all the isolation strategies

```
TCL_SETUP> get_attribute [find -set_isolation *] name
PD_SW1.iso_vddc2_to_vddc1 PD_LV.iso_vddc2_to_vddc PD_LV.iso_vddc1_to_vddc
```

Example 2: To get the implementable elements of a specific isolation strategy

```
TCL_SETUP> get_attribute [find -set_isolation PD_LV.iso_vddc2_to_vddc]
implementable_elements
out2 out3

# Empty list is returned when there is no implementable element
TCL_SETUP> get_attribute [find -set_isolation PD_LV.iso_vddc1_to_vddc]
implementable_elements
TCL_SETUP>
```

Query Attribute For A Rule Check

There are three ways to find out the supported attribute names of the 1801 rule check:

- **report_tcl_attributes**

```
TCL_SETUP> report_tcl_attributes CROSSING_OFF_TO_ON_NOLP
==============================================================================
=                             CROSSING Rules                                 =
==============================================================================
|       Object Type
|                     Attribute Name       Attribute Type   Attribute
Description
|
|       CROSSING_OFF_TO_ON_NOLP
|                     driver               object           Path segment start
point
|                     driver_supply_set    object           Path
…
```

- **get_attribute**

```
TCL_SETUP> get_attribute [find -ruleinst CROSSING_OFF_TO_ON_NOLP]
name full_name object_type desc full_desc location start_line status runtime
mem_usage
 filters version severity category analysis_style rule_type required_license
options
required_state check_proc report_rule_proc get_occr_msg_proc diagnose_proc
open_schematics_proc open_sdc_src_proc open_hdl_src_proc include_files
table_header help_file rule
set rulegrp rulesrc is_run is_virtual occrs occr_count occr_limit
continue_on_error
```

- **`report_rule_check -attributes`**

```
TCL_SETUP> report_rule_check CROSSING_OFF_TO_ON_NOLP/1 -attributes
CROSSING_OFF_TO_ON_NOLP: Invalid OFF to ON crossing
    Severity: Error
    1: Path segment from 'VDD1' (tied to '1') (supply set: 'supply_set_VDD1') to
'm1/in_
pin' (supply set: 'supply_set_VDD2_VSS' at 1V) fails OFF to ON check
(UPF/top.upf:67 Sta
te: 'PMC_GROUP.M2')

       Attributes:
            driver (pin)                                 : VDD1
            driver_supply_set (supply_set)               : supply_set_VDD1
            receiver (pin)                               : m1/in_pin
            receiver_supply_set (supply_set)             : supply_set_VDD2_VSS
            receiver_supply_set_voltage (double)         : 1.0
```

Applications

Tcl attribute support is available in the 1801 rule checks for reporting, diagnosis, and creating filters.

The attributes make the information to be available for users for rule occurrence filtering or analysis.

Later in this section, we have examples of how to view all the supported Tcl attributes for the violated rule and how to create filters to customize the report usage.

Customized Report Using TCL Attributes

Users can quickly get details about a rule occurrence and its attribute, using **`report_rule_check -attributes`**. The attributes reported are in the following format:

`<attribute_name> (<type>): <value>`

```
TCL_SETUP> report_rule_check CROSSING_OFF_TO_ON_ISO/1 -attributes
CROSSING_OFF_TO_ON_ISO: Isolation cell fails OFF to ON check
```

```
    Severity: Error      Occurrence: 1
    1: Path segment from 'u_iso/Y' (supply set: 'supply_set_VDP_VSS') to
'ur/u0/A' (supply
 set: 'supply_set_VDR_VSS' at 1V) fails OFF to ON check (top.upf:96 State:
'PMC_GROUP.M3')
       Attributes:
            driver (pin)                                  : u_iso/Y
            driver_supply_set (supply_set)                :
supply_set_VDP_VSS
            receiver (pin)                                : ur/u0/A
            receiver_supply_set (supply_set)              :
supply_set_VDR_VSS
            receiver_supply_set_voltage (double)          : 1.0
            receiver_supply_set_voltage_range (string)    : 1.0:1.0
            crossings (crossing list)                     : crossing_4
            has_off_to_on_crossings (bool)                : 1
            has_voltage_conflict_crossings (bool)         : 0
            power_state (object)                          : PMC_GROUP.M3
            is_driver_iso_cell (bool)                     : 1
            is_receiver_iso_cell (bool)                   : 0
```

Report Rule Occurrence And Sub-Attributes

The **`add_rule_attribute_display`** command is used to report a sub-attribute of a rule occurrence. For example, to report the power attribute of a **`supply_set`** the command is used as follows:

```
TCL_SETUP> add_rule_attribute_display -object_type supply_set power
0
TCL_SETUP> report_rule_check CROSSING_OFF_TO_ON_ISO -verbose -attributes
CROSSING_OFF_TO_ON_ISO: Isolation cell fails OFF to ON check
    Severity: Error      Occurrence: 1
    1: Path segment from 'u_iso/Y' (supply set: 'supply_set_VDP_VSS') to
'ur/u0/A' (supply
 set: 'supply_set_VDR_VSS' at 1V) fails OFF to ON check (top.upf:96 State:
'PMC_GROUP.M3')
       Attributes:
            driver (pin)                                  : u_iso/Y
            driver_supply_set (supply_set)                :
supply_set_VDP_VSS
               driver_supply_set.power (supply_net)       : VDP
            receiver (pin)                                : ur/u0/A
            receiver_supply_set (supply_set)              :
supply_set_VDR_VSS
               receiver_supply_set.power (supply_net)     : VDR
            receiver_supply_set_voltage (double)          : 1.0
            receiver_supply_set_voltage_range (string)    : 1.0:1.0
            crossings (crossing list)                     : crossing_4
            …
```

Hint: To know the supported attribute of an object type, specify it at the **report_tcl_attributes** command:

```
TCL_SETUP> report_tcl_attributes supply_set
==============================================================================
=                               1801 Objects                                 =
==============================================================================
|       Object Type
|                               Attribute Name     Attribute Type   Attribute
Description
|
|       supply_set
|                               object_type       string           Type of the
Tcl object
...
|                               name              string           Object name
|                               function          string           Option '-
function'
|                               power             object           Power net
|                               ground            object           Ground net
|                               nwell             object           Nwell net
...
```

Get Attribute And Applications

A given rule occurrence can also be accessed in Tcl using the **get_attribute** command.

```
TCL_SETUP> set occr [find -occr CROSSING_OFF_TO_ON_ISO/1]
CROSSING_OFF_TO_ON_ISO/1

TCL_SETUP> get_attribute $occr driver
u_iso/Y

TCL_SETUP> get_attribute $occr driver_supply_set
supply_set_VDP_VSS

TCL_SETUP> get_attribute $occr load
ur/u0/A
```

Similarly, to access sub-attributes, syntax is <attribute_name>.<sub_attribute_name>

```
TCL_SETUP> get_attribute $occr driver_supply_set.power
VDP
```

Rule Filter Commands

Options as **`-ATTRibutes`** and
`-FILTER <expression> [-REGEXp]` are provided with the commands
`report_rule_check` and **`add_rule_filter`** to let the user customize the report.

Rule filter command settings:

- **`add_rule_filter`**
- **`delete_rule_filter`**
- **`report_rule_filter`**

Filter <expression>

Allows queries based on attributes. The *<expression>* must use the following format *<attribute_name><operator><value>*.The following operators are supported. Multiple conditions can be combined with the logical operators !, &&, ||, and parentheses ().

- ==!= String and value comparison
- =~ !~ Pattern matching

Tcl List Query Operators

For the attributes that are 'list' types, the Tcl list operators provided below offer greater convenience and flexibility.

- {>=} Every element of RHS is in LHS
  e.g., attr {>=} {1 2} is a superset or equal to set {1 2}
- {>} Every element of RHS is in LHS, and LHS has some elements not in RHS
  e.g., attr {>} {1 2} is a proper superset of set {1 2}
- {<=} Every element of LHS is in RHS
  e.g., attr {<=} {1 2} is subset of, or equal to set {1 2}
- {<} Every element of LHS is in RHS, and RHS has some elements not in LHS
  e.g., attr {<} {1 2} is proper subset of set {1 2}
- {><} There is no element in both LHS and RHS, and none of them is empty
  e.g., attr {><} {1 2} is disjoint with the set {1 2}

- {<>} There is some element in both LHS and RHS
  e.g., attr {<>} {1 2} has nonempty intersection with the set {1 2}

- {==} Every element in LHS is in RHS, AND vice versa
  e.g., attr {==} {1 2} is equal to the set {1 2}

- {!=} Some element in LHS is not in RHS, OR vice versa
  e.g., attr {!=} {1 2} is not equal to the set {1 2}

Tcl List Query Operators (Wildcard)

For the filter expression which does not need to specify every element in a list query. In this case, a wildcard can be used to specify the query, and the following operators support wildcards:

- attr {<~} {pattern ...}
  Every element of LHS matches some pattern in RHS

- attr {>~} {pattern ...}
  Some elements of LHS match some patterns in RHS

- attr {!<~} {pattern ...}
  Some elements of LHS do not match any pattern in RHS

- attr {!>~} {pattern ...}
  Every element of LHS does not match any pattern in RHS

Rule Check Example

In the below example, there are 54 **`CROSSING_OFF_TO_ON_NOLP`** violation occurrences:

```
CROSSING_OFF_TO_ON_NOLP: Invalid OFF to ON crossing
    Severity: Error      Occurrence: 54
    1: Path segment from 'in[0]' (supply set: 'SS_TOP') to 'u1/u0/A' (supply
 set: 'SS1' at 1V) fails OFF to ON check (Power_Intent/top.upf:52 State:
'PD_TOP.STD1')
```

Users can view each occurrence with its attributes as :

```
TCL_SETUP> report_rule_check CROSSING_OFF_TO_ON_NOLP/1 -attributes
CROSSING_OFF_TO_ON_NOLP: Invalid OFF to ON crossing
    Severity: Error
    1: Path segment from 'in[0]' (supply set: 'SS_TOP') to 'u1/u0/A' (supply
 set: 'SS1' at 1V) fails OFF to ON check (Power_Intent/top.upf:52 State:
```

```
'PD_TOP.STD1')

       Attributes:
            driver (pin)                                         : in[0]
            driver_supply_set (supply_set)                       : SS_TOP
            receiver (pin)                                       : u1/u0/A
            receiver_supply_set (supply_set)                     : SS1
            receiver_supply_set_voltage (double)                 : 1.0
            receiver_supply_set_voltage_range (string)           : 1.0:1.0
            crossings (crossing list)                            : crossing_1
            has_off_to_on_crossings (bool)                       : 1
            has_voltage_conflict_crossings (bool)                : 1
            power_state (object)                                 : PD_TOP.STD1
            is_driver_iso_cell (bool)                            : 0
            is_receiver_iso_cell (bool)                          : 0
```

Rule Filter Example

Example 1: To filter the violation by a specific design object reported at some checkers, for example, the occurrences reported at **CROSSING_OFF_TO_ON_*** from a specific driver, *u4/u5/Y*, users can use the below filter

```
TCL_SETUP> add_rule_filter F1_CROSSING_OFF_TO_ON –rule CROSSING_OFF_TO_ON_* \
              –filter {driver == “u4/u5/Y”}
```

To further constrain the receiver supply set is *SS1*, use

```
TCL_SETUP> add_rule_filter F1_CROSSING_OFF_TO_ON \
       –rule CROSSING_OFF_TO_ON_* \
       –filter {(driver == “u4/u5/Y”) && (receiver_supply_set == “SS1”)}
```

Example 2: To filter the driver supply set from *SS1* and the receiver supply set is *SS2*, use

```
TCL_SETUP> add_rule_filter F2_CROSSING_OFF_TO_ON –rule CROSSING_OFF_TO_ON_* \
   –filter {(driver_supply_set == “SS1”) && (receiver_supply_set == “SS2”)}
```

Effects Of Rule Filters

To report all filter settings and effects, use **report_rule_filter**. For example,

```
TCL_SETUP> report_rule_filter *

==========================================================================
=                                Rule Filters                            =
==========================================================================
|  Filter 'F12'
|    Filter: (driver_supply_set == "SS1") && (receiver_supply_set == "SS2")
|    Applied to:
|      - Rule instance 'CROSSING_OFF_TO_ON_NOLP'
|      - Rule instance 'CROSSING_OFF_TO_ON_ISO'
|      - Rule instance 'CROSSING_OFF_TO_ON_PSW'
|      - Rule instance 'CROSSING_OFF_TO_ON_LSH'
|      - Rule instance 'CROSSING_OFF_TO_ON_RET'
|      - Rule instance 'CROSSING_OFF_TO_ON_AON'
|      - Rule instance 'CROSSING_OFF_TO_ON_ALIVE_DURING_POWER_UP'
|    Status: Filtering out 12 occurrences
| -----------------------------------------------------------------------------
|  Filter 'F23'
|    Filter: (driver_supply_set == "SS2") && (receiver_supply_set == "SS3")
|    Applied to:
|      - Rule instance 'CROSSING_OFF_TO_ON_NOLP'
|      - Rule instance 'CROSSING_OFF_TO_ON_ISO'
|      - Rule instance 'CROSSING_OFF_TO_ON_PSW'
|      - Rule instance 'CROSSING_OFF_TO_ON_LSH'
|      - Rule instance 'CROSSING_OFF_TO_ON_RET'
|      - Rule instance 'CROSSING_OFF_TO_ON_AON'
|      - Rule instance 'CROSSING_OFF_TO_ON_ALIVE_DURING_POWER_UP'
|    Status: Filtering out 12 occurrences
===============================================================================
```

To review the occurrences filtered by a specific filter, specify the filter name at the **`-filtered_by`** option at **`report_rule_check`**. For example,

```
TCL_SETUP> report_rule_check -filtered_by F12 -verbose

CROSSING_OFF_TO_ON_NOLP: Invalid OFF to ON crossing
    Severity: Error      Occurrence: 12
   13: Path segment from 'u1/u0/Y' (supply set: 'SS1') to 'u2/u0/A' (supply set:
'SS2' at 1V) fails OFF to ON check (Power_Intent/top.upf:52 State: 'PD_TOP.STD2')
   14: Path segment from 'u1/u0/Y' (supply set: 'SS1') to 'u2/u2/B' (supply set:
'SS2' at 1V) fails OFF to ON check (Power_Intent/top.upf:52 State: 'PD_TOP.STD2')
   15: Path segment from 'u1/u1/Y' (supply set: 'SS1') to 'u2/u0/B' (supply set:
'SS2' at 1V) fails OFF to ON check (Power_Intent/top.upf:52 State: 'PD_TOP.STD2')
   16: Path segment from 'u1/u1/Y' (supply set: 'SS1') to 'u2/u1/A' (supply set:
'SS2' at 1V) fails OFF to ON check (Power_Intent/top.upf:52 State: 'PD_TOP.STD2')
   17: Path segment from 'u1/u1/Y' (supply set: 'SS1') to 'u2/u3/B' (supply set:
'SS2' at 1V) fails OFF to ON check (Power_Intent/top.upf:52 State: 'PD_TOP.STD2')
   18: Path segment from 'u1/u2/Y' (supply set: 'SS1') to 'u2/u2/A' (supply set:
'SS2' at 1V) fails OFF to ON check (Power_Intent/top.upf:52 State: 'PD_TOP.STD2’)
…
```

Filter with List-type Attribute

This example shows how to categorize the report for the 'list-type' attribute using the string pattern match operator and list query operator.

```
CROSSING_FUNC_PATH_BUF_OFF: Supply sets of a buffer chain between two\
 functions are not compatible with source or receiving function supply sets
    Severity: Warning    Occurrence: 3
    1: In the path from 'in[1]' (supply set: 'ss_top') to 'u0/u0/B' \
(supply set: 'ss_top', path_type: 'Other')
```

```
       Instance 'buf1' (supply set: ss_vsw, power domain: PD_TOP) and \
the next 1 instances in the path are off (Power_Intent/buf.upf:50 State:
'PMC_GROUP.M2')
       Attributes:
            driver (pin)                                       : in[1]
            driver_supply_set (supply_set)                     : ss_top
            receiver (pin)                                     : u0/u0/B
            receiver_supply_set (supply_set)                   : ss_top
            pins (pin list)                                    : buf1/Y buf2/Y
            supply_set (supply_set)                            : ss_vsw
            power_state (simple_state)                         : PMC_GROUP.M2
            path_type (string)                                 : Other
            domain (power_domain)                              : PD_TOP
            crossings (crossing list)                          : crossing_2
            has_off_to_on (bool)                               : 1
            has_voltage_conflict (bool)                        : 0
            is_same_driver_receiver_supply_set (bool)          : 1
```

In this example, to only report the **`CROSSING_FUNC_PATH_BUF_OFF`** occurrences have *buf2/Y* in the paths. The report can be filtered through the filter expression using the string compare operator (need to match the reported name for string compare):

```
report_rule_check CROSSING_FUNC_PATH_BUF_OFF -filter {(pins =~ "*buf2/Y*")} -
verbose
```

Or using the list query operator to get the list element and compare:

```
report_rule_check CROSSING_FUNC_PATH_BUF_OFF -filter {(pins {>=} "buf2/Y")} -
verbose
```

Using the list operators is recommended for the list object query, which provides a more flexible usage than the string comparison.