Leda
Release Notes

Version 2006.06
June 2006

Comments?
E-mail your comments about this manual to
leda-support@synopsys.com.

SYNOPSYS®

Contents

1 Release Notes - 2006.06

Introduction

These notes for the 2006.06 release of the Leda software are organized in the following sections:

New in Leda 2006.06

Attention
Starting with version 2006.06, Leda no longer support Solaris 5.8.

Leda Tcl Interface for Collections and Wildcard Support

A collection is a group of objects exported to the Tcl user interface. Leda provides the following set of commands that you can use to create and manipulate collections:

- add_to_collection
- append_to_collection
- compare_collections
- copy_collections
- filter_collections
- foreach_in_collection
- index_collection
- remove_from_collection
- sizeof_collection
- sort_collection
- get_object_name
- query_objects
- all_clocks
- all_inputs
- all_instance
- all_outputs
- all_registers
- get_clocks

- set_power_switch

You can use simple wildcard characters like * and ? and also complete regular expressions with the collection. For more information, see the *Leda User Guide*.

SDC Equivalency Checks

This release features the new SDC Equivalency Check rules. SDC equivalence checker compares two SDC files belonging to two different design phases and reports the discrepancies in the intent of the design constraints of the two SDC files. The SDC Equivalency Check rules are grouped under a new ruleset named Equivalency in the Constraints policy (see Table 1)

Note

You cannot use these rules in batch or GUI mode. They only work in Tcl shell mode.

For more information, see the *Leda Constraints Rules Guide.*

Table 1: SDC Policy Equivalency Ruleset

Label	Message
SDC_EQCLK01	Equivalency file clock constraint is inconsistent with reference file clock constraint: equivalency clock is %s
SDC_EQCLK02	Reference file clock constraint is inconsistent with equivalency file clock constraint: reference clock is %s
SDC_EQIDL01	Equivalency file input delay constraint is inconsistent with reference file delay constraint: input port is %s
SDC_EQIDL02	Reference file input delay constraint is inconsistent with equivalency file delay constraint: input port is %s
SDC_EQODL01	Equivalency file output delay constraint is inconsistent with reference file delay constraint: output port is %s
SDC_EQODL02	Reference file output delay constraint is inconsistent with equivalency file delay constraint: output port is %s
SDC_EQFLP01	Equivalency file false path constraint is inconsistent with reference file false path constraint.
SDC_EQFLP02	Reference file false path constraint is inconsistent with equivalency file false path constraint.
SDC_EQMCP01	Equivalency file multicycle path constraint is inconsistent with reference file multicycle path constraint.

Table 1: SDC Policy Equivalency Ruleset

Label	Message
SDC_EQMCP02	Reference file multicycle path constraint is inconsistent with equivalency file multicycle path constraint.
SDC_EQCMB01	Equivalency file max-delay path constraint is inconsistent with reference file max-delay path constraint.
SDC_EQCMB02	Equivalency file min-delay path constraint is inconsistent with reference file min-delay path constraint.
SDC_EQCMB03	Reference file max-delay path constraint is inconsistent with equivalency file max-delay path constraint.
SDC_EQCMB04	Reference file min-delay path constraint is inconsistent with equivalency file min-delay path constraint.

New SDC Policy Objects Ruleset

This release includes a new ruleset named Objects in the Constraints policy (see Table 2). These rules are used to identify prohibited objects for the SDC commands create_clock and create_generated_clock. For more information, see the *Leda Constraints Rules Guide.*

Table 2: New SDC Policy Objects Ruleset

Label	Message
SDC_OBJ01	Do not set create_clock to input pin of black box macro cells
SDC_OBJ02	Do not set create_clock to hierarchical pin
SDC_OBJ03	Do not set create_clock to net
SDC_OBJ04	Do not set create_clock to output pin of registers
SDC_OBJ05	Do not set create_clock to output pin of combinational cells in clock line
SDC_OBJ06	Do not set create_clock to primary output
SDC_OBJ07	Do not set create_clock to clock pin of registers
SDC_OBJ08	Do not set create_clock to data pin of registers
SDC_OBJ09	Do not set create_clock to input pin of combinational cells in clock line
SDC_OBJ12	Do not set create_generated_clock to primary input
SDC_OBJ13	Do not set create_generated_clock to input pin of Black Box macro cells
SDC_OBJ14	Do not set create_generated_clock to hierarchical pin
SDC_OBJ15	Do not set create_generated_clock to net
SDC_OBJ16	Do not set create_generated_clock to output pin of combinational cells in clock line
SDC_OBJ17	Do not set create_generated_clock to data pin of registers
SDC_OBJ18	Do not set create_generated_clock to input pin of combinational cells in clock line

New Prebuilt Configurations

This release includes a new set of five prebuilt configurations (see Table 3).

Table 3: New Prebuilt Configurations

Prebuilt Configurations	Description
SDC-quality-RTL	This configuration is a duplicate of the existing SDC-RTL prebuilt configuration.
SDC-quality-prelayout	This configuration is a duplicate of the existing SDC-prelayout prebuilt configuration.
SDC-quality-postlayout	This configuration is a duplicate of the existing SDC-postlayout prebuilt configuration.
SDC-top-versus-block	This configuration contains rules to verify consistency between block level and top level design constraint file.
SDC-equivalency	This configuration contains rules to verify equivalency between two SDC files of a design.

For more information, see the *Leda User Guide*.

New Formality Policy Rules

This release includes a new ruleset named SystemVerilog in the Formality policy (see Table 4).For more information, see the *Leda Formality Rules Guide.*

Table 4: New Formality Policy SystemVerilog Ruleset

Label	Message
FM_200	The intent of the always_latch construct is not verified by Formality.
FM_201	Declaration of types, tasks and functions at $root function is not supported.
FM_202	Reference to the $root instance path is not supported.
FM_203	Automatic interfaces are not supported.
FM_204	Const declarations are not supported.
FM_205	Structure literal assignments are not supported.
FM_206	Array literal assignments are not supported.

Table 4: New Formality Policy SystemVerilog Ruleset

Label	Message
FM_207	Generic interfaces are not supported.

Known Limitations

For array literals and struct literals, rules can be written to detect the usage of these constructs. The only exception is when (according to SV3.1a specifications) a array/struct literal is mentioned in the same syntax as a concatenation (without a “ ‘ ”). For example:

```
struct {
       int a;
       real b;
} STR;
STR c = {1,0.5};
```

In the above example, Leda considers it as a concatenation and so, Leda does not report rule FM_205 (structure literal used). The same applies to rule FM_206, when concatenation syntax is used for array literal.

New Xilinx Policy Rules

This release includes a new ruleset named Virtex4 in the Xilinx policy (see Table 5). For more information, see the *Leda Xilinx Rules Guide.*

Table 5: New Xilinx Policy Virtex4 Ruleset

Label	Message
XV4_1001	Internally generated synchronous reset detected (block level). Xilinx recommends that you avoid using internally generated synchronous reset.
XV4_1002	Internally generated asynchronous reset detected (block level). Xilinx recommends that you avoid using internally generated asynchronous reset.
XV4_1003	Multiple asynchronous resets detected in this unit/module.
XV4_1004	Multiple asynchronous resets detected in always block.
XV4_1005	Inferred register with asynchronous reset detected.
XV4_1006	Multiple synchronous resets detected in this unit/module.
XV4_1007	Multiple synchronous resets detected in always/process block.
XV4_1008	Reset line used both as synchronous and asynchronous reset is detected.
XV4_1009	Register with fixed value reset is detected.
XV4_1010	Detected an asynchronous and synchronous reset coded combinatorially.
XV4_1200	Multiple clocks detected in always/process block. Xilinx recommends that you specify only one clock signal in an always/process block.
XV4_1201	Multiple clocks detected in always/process block. Xilinx recommends that you specify only one clock signal in an always/process block.
XV4_1202	Clock gating detected. Xilinx warns that you avoid gating the clock and instead use the CE input of the sequential elements for controlling the clock.
XV4_1203	Internally generated clock detected. Xilinx recommends that you avoid using generating clocks in a module/unit and instead use the DCM to create the desired clock frequency if possible.
XV4_1204	Module/unit with more than one clock detected. Xilinx strongly recommends that you do not include logic associated with multiple clocking domains in a single module/unit.
XV4_1205	Clocks must not be used as data.
XV4_1206	Detected input DDR flip-flop.

Table 5: New Xilinx Policy Virtex4 Ruleset

Label	Message
XV4_1400	Case statement without 'others' default clause detected.
XV4_1401	Incomplete case statement using full_case directive is not recommended (not supported by some simulation tools).
XV4_1402	Incomplete case statement detected without default clause
XV4_1403	'if' statement without 'else' clause in a combinational logic block detected. Xilinx recommends that you include 'else' clause as default in 'if' statements.
XV4_1404	Unequal length between case expression and case item condition in case, casex, or casez statement detected.
XV4_1405	Number of nested elements in if statement should be less than 7.
XV4_1406	When case items are duplicated (parallel), do not use parallel_case directive.
XV4_1407	The directive 'full_case' is being used with a case statement that is complete.
XV4_1408	The directive 'full_case' is being used with a case statement that is incomplete.
XV4_1600	Inferred latch detected. Xilinx recommends that you avoid inferring latches in your design.
XV4_1601	Inferred module/unit level tri-states detected. Xilinx recommends that you avoid using tri-state buffers and use other logic resources when available.
XV4_1602	VHDL/Verilog key word detected. Xilinx warns that you avoid using VHDL/Verilog reserved words as identifiers in your code.
XV4_1603	Instantiated shift register SRL16 detected. Xilinx recommends that you register the last bit of the shift register for best performance.
XV4_1604	Reversed bit numbering detected. Xilinx recommends that you designate the right-most bit as the LSB (aka little endian) for multibit buses in your design.
XV4_1605	Unequal length LHS and RHS in assignment detected.
XV4_1606	Invalid data type detected on port.
XV4_1607	Multiplier has been detected.
XV4_1608	Do not use tool specific pragma.

Table 5: New Xilinx Policy Virtex4 Ruleset

Label	Message
XV4_1609	Very long port/module/entity name detected.
XV4_1610	Detected two-dimensional array, could be opportunity for RAM inference.
XV4_1612	Invalid mode of BUFFER found.
XV4_1613	Detected a small counter.
XV4_1614	Black-box component not in Xilinx DB libraries is detected.
XV4_1615	Output double data rate register functionality is detected.
XV4_1616	Flip-flop likely to be merged in synthesis is detected.
XV4_1800	Unconnected ports detected. Xilinx requires that all ports be connected.
XV4_1801	Missing signal in sensitivity list of combinational block detected.
XV4_1802	Combinational feedback loop detected. Xilinx requires that all combinational feedback loops be eliminated if possible.
XV4_1803	Module/unit containing non-registered outputs detected. Xilinx recommends that you register the outputs of a module/unit in your design.
XV4_1804	Flip-flop inferred using blocking assignment to a reg detected. Xilinx strongly recommends that you use non-blocking assignments to assign registers intended to be inferred as flip-flops.
XV4_1805	Asynchronous logic detected. Xilinx strongly recommends that you document and isolate all asynchronous logic.
XV4_1806	High fanout net/port detected. Xilinx recommends register duplication to reduce the fanout.
XV4_1807	Unequal length port and connection in module instantiation detected.
XV4_1808	Redundant signals in sensitivity list detected.
XV4_1809	Positional association in port mapping of an instance detected.
XV4_2000	Non Moore style state machine is detected.
XV4_2001	No asynchronous logic detection will be performed in the module/unit of a finite state machine. Xilinx warns that you ensure proper coding for any asynchronous logic in the module/unit of a finite state machine.
XV4_2002	Use parameter declarations to define the state vector of a state machine.

Table 5: New Xilinx Policy Virtex4 Ruleset

Label	Message
XV4_2003	Xilinx recommends not using ‘define.
XV4_3000	Obsolete component detected. Xilinx recommends that you use the Architecture Wizard to create the IBUFG/DCM/BUFG.
XV4_3001	Obsolete component detected. Xilinx recommends that you use RAMB16_Sx or RAMB16_Sx_Sy.
XV4_3002	Obsolete component detected. Xilinx recommends that you use STARTUP_VIRTEX4.
XV4_3003	Obsolete component detected. Xilinx recommends that you use BSCAN_VIRTEX4.
XV4_3004	Obsolete component detected. Xilinx recommends that you use CAPTURE_VIRTEX4.
XV4_3005	Obsolete component detected. Xilinx recommends that you use IBUF(G)/IBUF(G)DS.
XV4_3006	Obsolete component detected. Xilinx recommends that you use IOBUF/IOBUFDS.
XV4_3007	Obsolete component detected. Xilinx recommends that you use OBUF/OBUFDS or OBUFT/OBUFTDS.
XV4_3008	Obsolete component detected. Xilinx recommends that you use IBUFG, BUFG or IBUFG, DCM, BUFG.
XV4_3009	Obsolete component GT* detected. Xilinx recommends that you use GT11.
XV4_3010	RAMB4* components are not supported in Virtex4. Xilinx recommends using the RAMB16 instead.
XV4_3011	Obsolete component IFDDR* detected. Xilinx recommends that you use IDDR.
XV4_3012	Obsolete component OFDDR* detected. Xilinx recommends that you use ODDR.

New Design Policy Rules

This release includes one new rule in the Design policy (see Table 6).For more information, see the *Leda Design Rules Guide.*

Table 6: New Design Policy Rules

Label	Message
NTL_STR77A	Prohibited cell found %s

Design Policy Rules Updated

This release includes a set of nine enhanced Design policy rules (see Table 7). For more information, see the *Leda Design Rules Guide*.

Table 7: Modified Rules in Design Policy

Label	Policy	Modifications

NTL_CON04	DESIGN	By default, this check is applied on the inout ports. You can set parameter APPLY_ONLY_ON_INPUTS to 1, to apply this check only on input ports (the default value is 0). For example:
		`leda> rule_set_parameter -rule NTL_CON04 \`
		`-parameter APPLY_ONLY_ON_INPUTS -value 1`

NTL_CON05	DESIGN	By default, this check is applied on the inout ports. You can set parameter APPLY_ONLY_ON_INPUTS to 1, to apply this check only on input ports (the default value is 0). For example:
		`leda> rule_set_parameter -rule NTL_CON04 \`
		`-parameter APPLY_ONLY_ON_INPUTS -value 1`

NTL_CON06	DESIGN	By default, this check is applied on the inout ports. You can set parameter APPLY_ONLY_ON_INPUTS to 1, to apply this check only on input ports (the default value is 0). For example:
		`leda> rule_set_parameter -rule NTL_CON06 \`
		`-parameter APPLY_ONLY_ON_INPUTS -value 1`
		If an inout port is derived with the "PAD" attribute from a DB cell, or if the inout port is owned by an analog DB cell (with the attribute "is_analog") yet the inout port is not a digital pin (without the attribute "is_digital_pin"), such an inout port will not be taken as an input to be checked by this rule even if the parameter APPLY_ONLY_ON_INPUTS is set to 0. Connection of such inout ports to supply is considered to be fine.

Table 7: Modified Rules in Design Policy

Label	Policy	Modifications

NTL_CON08	DESIGN	By default, this check is applied on the inout ports. You can set parameter APPLY_ONLY_ON_OUTPUTS to 1, to apply this check only on output ports (the default value is 0). For example:
		`leda>` **`rule_set_parameter -rule NTL_CON08 \`**
		`-parameter APPLY_ONLY_ON_OUTPUTS -value 1`
		If an inout port is derived with the "PAD" attribute from a DB cell, or if the inout port is owned by an analog DB cell (with the attribute "is_analog") yet the inout port is not a digital pin (without the attribute "is_digital_pin"), such an inout port will not be taken as an output to be checked by this rule even if the parameter APPLY_ONLY_ON_OUTPUTS is set to 0. Connection of such inout ports to supply is considered to be fine.

NTL_CON09	DESIGN	By default, this check is applied on the inout ports. You can set parameter APPLY_ONLY_ON_OUTPUTS to 1, to apply this check only on output ports (the default value is 0). For example:
		`leda>` **`rule_set_parameter -rule NTL_CON09 \`**
		`-parameter APPLY_ONLY_ON_OUTPUTS -value 0`

NTL_CON10	DESIGN	By default, this check is applied on the inout ports. You can set parameter APPLY_ONLY_ON_OUTPUTS to 1, to apply this check only on output ports (the default value is 0). For example:
		`leda>` **`rule_set_parameter -rule NTL_CON10 \`**
		`-parameter APPLY_ONLY_ON_OUTPUTS -value 1`
		If an inout port is derived with the "PAD" attribute from a DB cell, or if the inout port is owned by an analog DB cell (with the attribute "is_analog") yet the inout port is not a digital pin (without the attribute "is_digital_pin"), such an inout port will not be taken as an output to be checked by this rule even if the parameter APPLY_ONLY_ON_OUTPUTS is set to 0. Connection of such inout ports to supply is considered to be fine.

Table 7: Modified Rules in Design Policy

Label	Policy	Modifications

NTL_CON14	DESIGN	By default, this check is applied on the inout ports. You can set parameter APPLY_ONLY_ON_INPUTS to 1, to apply this check only on input ports (the default value is 0). For example:
		`leda> rule_set_parameter -rule NTL_CON14 \`
		`     -parameter APPLY_ONLY_ON_INPUTS -value “1”`
		If an inout port is derived with the "PAD" attribute from a DB cell, or if the inout port is owned by an analog DB cell (with the attribute "is_analog") yet the inout port is not a digital pin (without the attribute "is_digital_pin"), such an inout port will not be taken as an input to be checked by this rule even if the parameter APPLY_ONLY_ON_INPUTS is set to 0. Connection of such inout ports to supply is considered to be fine.

NTL_CON15	DESIGN	The message of this rule has been changed to “Power rails belonging to different supply types should not short %s1”. Arguments: • %s1 is the falsely connected supply pin. The source position of this rule is the line in which the pin is instantiated. A new parameter STRICTLY_CHECK, if set to TRUE, will check if the signal is connected to another signal of same polarity. For example, signal VDD connected to VDD, signal GND connected to GND and signal VDD(A) is connected to signal VDD(A).
NTL_STR21	DESIGN	The message of this rule has been changed to “The level of the design hierarchy should not exceed %d: %s”. Arguments: • %d is the value of the parameter “NB_MAX_HIERARCHY_LEVEL”. • %s is the hierarchy name of the reported instance.

NTL_STR77	DESIGN	By default this rule will report an error only once even if the prohibited cell is instantiated multiple times. To enable reporting violation for each instantiation of the prohibited cell, the following parameter should be set as follows:
		`leda> rule_set_parameter -rule NTL_STR77 \`
		`     -parameter MSG_ON_EACH_INSTANCE -value 1`

VCS Obsolete Rules

The following are the list of obsolete VCS rules that have been removed from the VCS policy documentation (see Table 8). For more information, see the *Leda VCS Rules Guide.*

Table 8: Obsolete VCS Policy Rules

Label	Policy	Message
VCS_10	VCS	Don't use implicit wire declaration.
VCS_11	VCS	Implicit net declaration is not supported.
VCS_2_2	VCS	Avoid using time declarations.
VCS_2_4	VCS	Avoid using trireg.
VCS_2_5	VCS	Avoid using ranges/array for integers.
VCS_3_1	VCS	Avoid using multiple n_input_gate.
VCS_3_2	VCS	Avoid using enable_gate.
VCS_3_3	VCS	Avoid using mos switches.
VCS_3_7	VCS	Avoid using pull gates.
VCS_4	VCS	Avoid strengths in cont assigns.
VCS_7	VCS	Avoid case statements in sequential always blocks.
VCS_7_1	VCS	Avoid using repeat in always block.
VCS_7_10	VCS	Avoid task enable in always blocks.
VCS_8	VCS	Avoid variable LHS bit-selects in always blocks.

Support of $power and $isolate System Calls

This version of Leda supports the system calls $power and $isolate. Leda infers power domains if $power system call is used in Verilog RTL code. Leda infers an isolation cell if $isolate system call is used in Verilog RTL code. For more information, see the *Leda Power Rules Guide.*

Warning
This feature is not supported in VHDL.

New Power Policy RTL Ruleset

This release includes a new ruleset named RTL in the power policy (see Table 9). These rules are used to check the semantics and usage of $power and $isolate system task calls. For more information, see the *Leda Power Rules Guide.*

Table 9: New Power Policy RTL Ruleset

Label	Message
RTLPOW00	Wrong number of arguments passed - at least 6 arguments are expected. Currrent $power statement will be ignored by Leda power checks.
RTLPOW01	First argument to $power (power domain name) must be a quoted string. Currrent $power statement will be ignored by Leda power checks.
RTLPOW02	Second argument to $power (power_on net) must be a valid signal name. Currrent $power statement will be ignored by Leda power checks.
RTLPOW03	Third argument to $power (on_sense expression) must be a valid expression. Currrent $power statement will be ignored by Leda power checks.
RTLPOW04	Fourth argument to $power (power_on ack net) must be a valid signal name. Currrent $power statement will be ignored by Leda power checks.
RTLPOW05	Fifth argument to $power statement (ack_sense expression) must be a valid expression. Currrent $power statement will be ignored by Leda power checks.
RTLPOW06	Argument must be a valid instance name. Currrent $power statement will be ignored by Leda power checks: Argument %d
RTLPOW07	$power must occur in an initial block
RTLPOW08	$power statement must not be nested within any control structure
RTLPOW09	$power statement must not be preceded by any timing control
RTLPOW10	Each instance in a design may be named in a power statement only once. Instance name will be ignored in enclosing statement: %s”
RTLPOW11	Power domains cannot be redefined at different hierarchical levels. Current statement will be ignored in power domain elaboration.
RTLPOW20	Signals passed as arguments to $power must be wires declared in the same module
RTLPOW21	Register initialized by a constant
RTLPOW22	Extra drivers detected for powerup_ack_net signal: %s
RTLPOW23	Non edge-sensitive logic detected on boundary of power-down region.

Table 9: New Power Policy RTL Ruleset

Label	Message
RTLPOW24	Nested non-contiguous power regions detected
RTLISO00	Wrong number of arguments passed - exactly 4 arguments are expected. Current $isolate statement will be ignored by Leda power checks
RTLISO01	First argument to $isolate (output net) must be a valid signal name. Currrent $isolate statement will be ignored by Leda power checks.
RTLISO02	Second argument to $isolate (enable net) must be a valid signal name. Currrent $isolate statement will be ignored by Leda power checks.
RTLISO03	Third argument to $isolate (input net) must be a valid signal name. Currrent $isolate statement will be ignored by Leda power checks.
RTLISO04	Output and input signals must be of the same type. Currrent $power statement will be ignored by Leda power checks.
RTLISO05	Fourth argument to $isolate statement must have the same type as the output signal (first argument). Currrent $isolate statement will be ignored by Leda power checks.

Rules that refer to valid signals refer to signals declared in the same module as the call.

Wild cards are not supported among instance name arguments. For example:

```
$power(temp1, in1[2], 1'b1, dummy_out, 1'b1, "foo", "bar, sub*");
```

No power domain will be inferred for the above command, because the instance name sub* cannot be resolved. However, Leda 2006.06 gives no warning (for example, rule RTLPOW06 does not flag) about this.

Known Limitations

- Rule RTLPOW06 has the following known limitations. If a power domain refers to instances (6th and subsequent arguments) that are not declared in the current module, the name given must be an absolute name, starting with the top module. In the example below, inst2/inst21 will not be found (must be top/inst2/inst21). However the power domain is inferred and this may cause some rules to give false violations.

```
initial

$power ("Power_Domain1", en, 1'b0, ack, 1’b0, "inst1", "top/inst2",
"inst2/inst21");

submod inst2 (.Pin(data_in2), .Pout(data_out));

submod inst3 (.Pin(data_in2), .Pout(data_out));
```

- Leda does not accept several instances within the same quotes.
- Leda accepts simple non-hierarchical instantiations only. For example, inst2, but not top.inst2. If you use top/inst2, Leda gives a syntax error.
- Leda infers a conditional multiplexer instead of a tristate for the command

```
$isolate(o,en,i,1'bz)
```

- Leda infers a conditional multiplexer instead of a latch for the command

```
$isolate(o,en,i,i)
```

PG-Netlist Support

This release of Leda has enhanced the Leda Power checking technology in order to be compliant with DC. Leda now supports netlists with explicit connectivity information of power and ground nets. These are called PG-Netlist.

The Leda analyzer parses any Verilog gate-level netlist descriptions with explicit power ports/nets representation, and it extracts the specific power semantics of these ports/nets. Therefore Leda can automatically infer the corresponding voltage domains (power domains). So, you need not define power domains (for voltage checks) in the case of PG-Netlist.

Leda supports the new Liberty syntax for DB cells added in Library Compiler 2006.06 and is also upwardly compatible with 2006.03 syntax. The backward compatibility support for existing library DBs compiled with old PG pin syntax is exactly the same as Library Compiler. When such DBs are read into memory, they are automatically translated to equivalent DBs as if they were compiled with the new PG pin syntax. In this way, you can combine new PG pin-based library DBs and old PG pin-based library DBs in the same flow. The translation process works as follows:

For each cell group, create a “primary power” pg-pin for each rail connection (<rc_name>, <voltage_name>) as follows:

```
pg_pin(<rc_name>)
{
    voltage_name : <voltage_name>;
    pg_type : primary_power;
}
```

And a “primary_ground” pg_pin for each cell group as follows:

```
          pg_pin(<pg_pin_name>)
          {
                voltage_name : <ground_rail_voltage_name>;
                pg_type : primary_ground;
          }
```

By default, <pg_pin_name> is “__VSS”. If the “__VSS” name has been reserved for the rail connection name <rc_name>, a unique name will be generated for <pg_pin_name> by trying __VSS1, __VSS2,…, VSSn until it does not conflict with a rail connection name.

Note that for libraries where a cell does not define a rail_connection group, the “primary_power” will be created as follows:

```
          pg_pin(<pg_pin_name>)
          {
                voltage_name : < voltage_map_name>;
                pg_type : primary_power;
          }
```

where, <pg_pin_name> is first value from the value set {“VDD”, "__VDD", "__VDD0", "__VDD1", ... } which does not cause the name conflicts and < voltage_map_name> is the name of the default power_rail in the power_supply group.

If a library cell contains additional pins that are intended to be used as pg-pins, Leda cannot identify them automatically. In this case, you need to use the set_power_pin command in the Leda shell to indicate the pins. The command is used as follows:

```
   leda> set_power_pin <cell name> <pin name> -power | -gnd
```

For example:

```
   leda> set_power_pin CELL VDD -power
   leda> set_power_pin "CELL" VSS -gnd
```

To apply on all cells, put "" as cell name.

```
   leda> set_power_pin "" VPWR -power
   leda> set_power_pin "" VGND -gnd
```

For level shifters and other multi-voltage cells, you need to qualify both power pins.

For more information, see the *Leda Power Rules Guide.*

VeRSL Additions

New VeRSL local attributes were added to the following templates (see Table 10).

Table 10: Attributes in Templates

Attribute	Template
multiply_blocking_assigned_signal	blocking_assignment
arg_decl_outside_stmt_scope	system_task_enable
is_assigned	reg_declaration
initialized_by_constant	reg_declaration
automatic	module_declaration
automatic	interface_declaration
root_ref	upward_ref
integer_literal_overflow	literal
special_percentile_handle	continuous_assign blocking_assignment non_blocking_assignment binary_operation procedural_continuous_assign procedural_continuous_force function_call module_instantiation

The following are the three new templates added to the VeRSL language:

- struct_literal
- array_literal
- upward_ref

For more information, see the *VeRSL Reference Guide.*

New Tcl Shell Mode Options

The following options were added in this release:

- The -for_equivalency option when used with read_constraints command can be used to access the SDC Equivalency checks.

- The command create_power_domain syntax has changed. Option -cells is changed to -object_list and -power_down is changed to -power_down_ctrl. Option -power_down is now a boolean expression.

For more information, see the *Leda Tcl Interface Guide.*

Changes in DQL Clock and Reset Inference

Leda now has an unified clock inferencing technique and has a consistent definition of the terms clock origins, gated clocks, internally generated clocks, etc. So, there is an impact on rules like NTL_CLK04 (internally generated clock) and NTL_CLK07 (gated clock).

The set of clocks returned by the DQL function DQ_get_all_clock_origins has changed to reflect the new definition. This may have an impact on some custom rules.

Hardware Inference

This section explains how Leda identifies entities like primary clock, generated clock, gated clock, clock origin, and clock domain. This can also be extended to resets.

Primary Clock

A primary clock is a primary input port used as a clock. Leda also considers inverters and buffers on the clock path. Some examples of primary clocks are shown in Figure 1, 2 and 3.

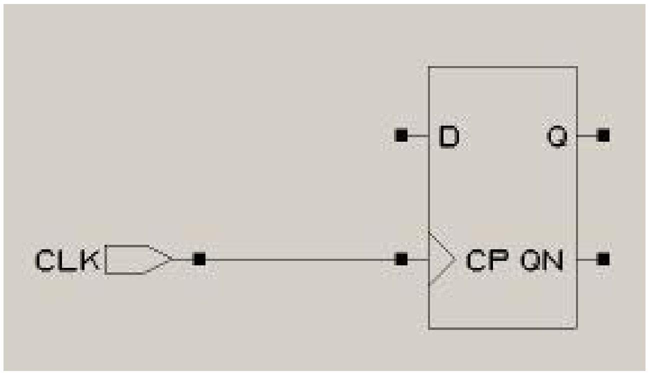

Figure 1: Signal CLK is a primary clock

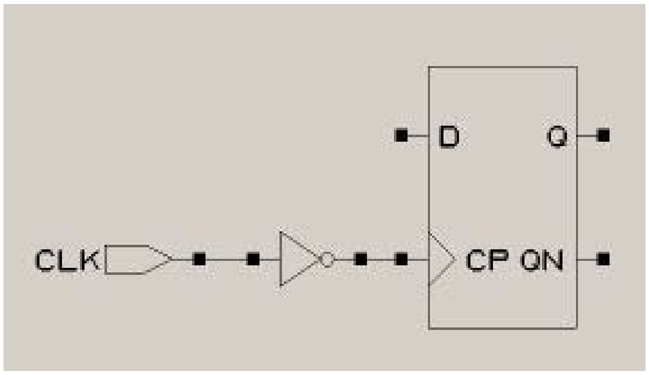

Figure 2: Signal CLK is a primary clock

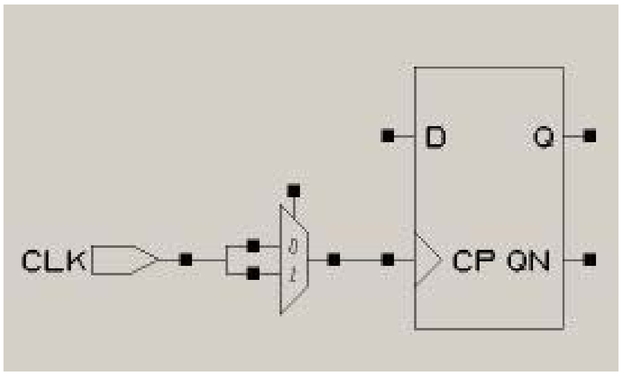

Figure 3: Signal CLK is also a primary clock

Internally Generated Clock

An internally generated clock is a clock derived from a primary clock or an internal signal driving the clock input of a flip-flop but not connected to a primary port.

In Figure 4, the internally generated clock has a synchronous relationship with its primary (or master) clock.

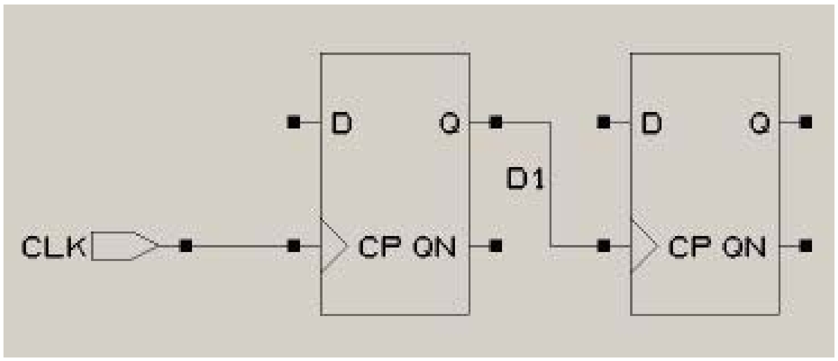

Figure 4: Signal D1 is a generated clock from primary clock CLK

In Figure 5, signal INT1 is a generated clock as there is no connection to any primary port.

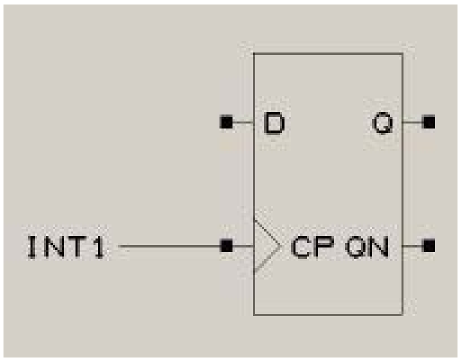

Figure 5: INT1 is a generated clock as no connection to a primary port

Figure 6 models a disconnected signal driving the clock input of a flip-flop. This implies that for any disconnection in the clock connectivity, Leda infers a new clock.

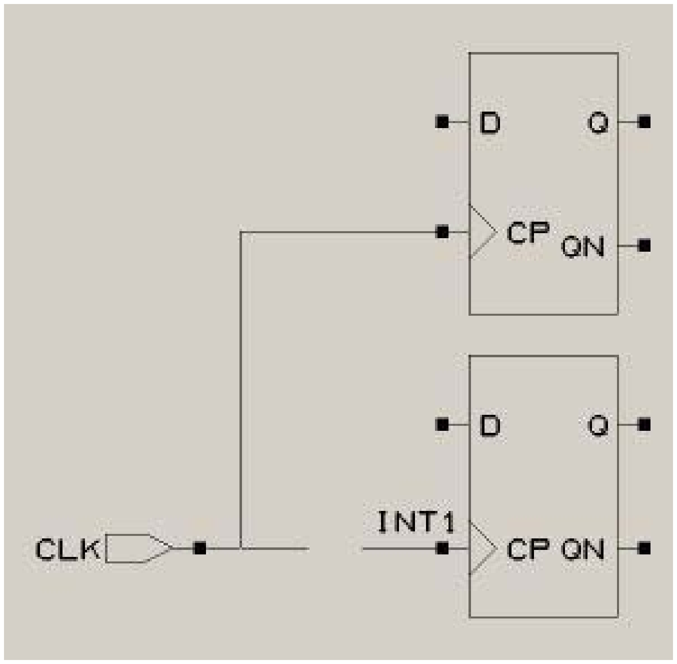

Figure 6: INT1 is a generated clock due to disconnection

Gated Clock

A gated clock is a signal on a clock path that is the output of a combinatorial complex block. A gated clock is modelled in both Figure 7 and Figure 8. In Figure 7, the clock origin is CLK and in Figure 8, the gated clock (output signal of the AND gate) is the clock origin.

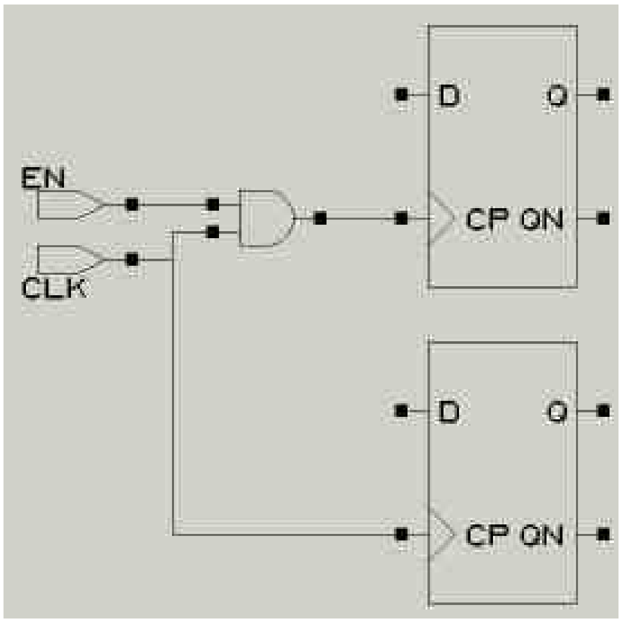

Figure 7: Gated clock

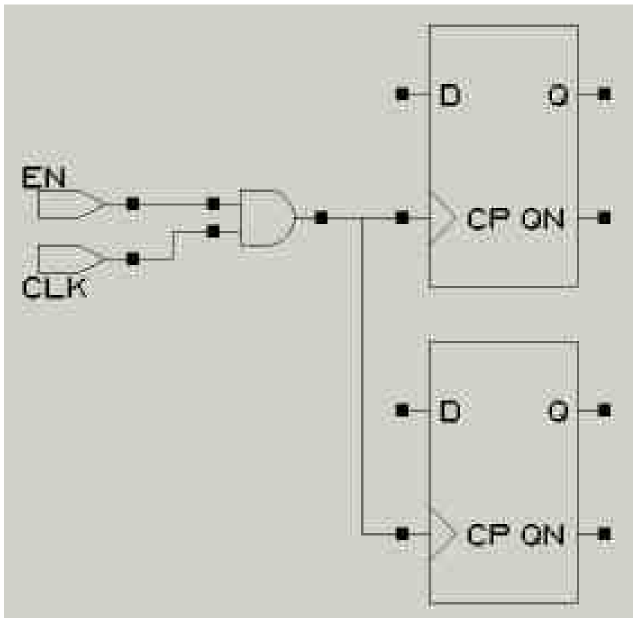

Figure 8: Gated clock

Attention

Figure 8 infers an internally generated clock as there are no identified clock origins (primary clock or internally generated clock) on the input cone to the gate.

The general rule for inferring gated clock is as follows:

Note

A gated clock is considered as a clock origin, if and only if, there is zero or more than one identified clock origins in its fan-in cone. Such gated clocks, that is also a clock origin are considered as a kind of internally generated clock. Other gated clocks having exactly one clock origin in its fan-in cone, are neither considered as clock origins nor as internally generated clocks.

Clock Origin

Clock origins are the set of all clocks comprising the primary clocks and internally generated clocks.

Note

If a clock or reset/set origin signal is tied to a constant value, then that signal no longer act as a clock or reset/set regarding its register. So, the signal is not stored in the control origin list.

Clock Domain

A clock domain is a set of flip-flops driven by the same clock origin or derivatives of this clock origin. A derivative is an internally generated clock whose master is the current clock origin or is a gated clock and one of the inputs to the gate is the current clock origin. The master clock origin of a clock domain is the highest clock in the hierarchy. All clocks in a clock domain are synchronous.

Rule Changes

To show the impact of the change in clock inference, we compare two “identical” rules NTL_CLK04 and C_1203 for internally generated clocks and see how the results will change in 2006.06.

In 2006.03, rule NTL_CLK04 behavior is incorrect with respect to the definition of internally generated clock. Figure 9 illustrates this:

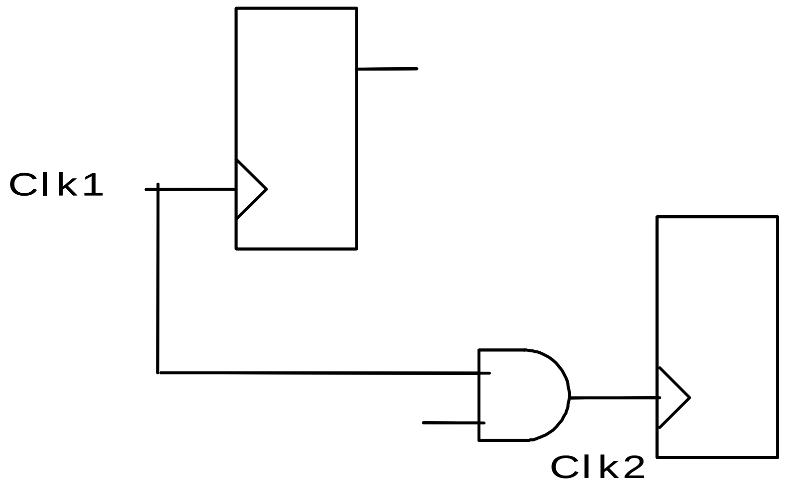

Figure 9: A sample schematic

For figure 9, in version 2006.03, rule NTL_CLK04 flags once and rule C_1203 doesn't flag. C_1203 has the correct behavior as clk2 is a gated clock and not an internally generated clock. In version 2006.06, rule NTL_CLK04 will not flag.

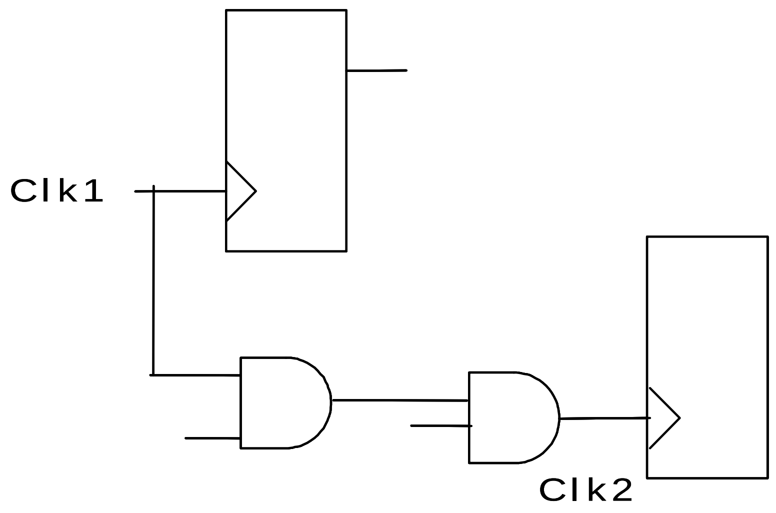

Figure 10: Another sample schematic

In version 2006.03, rule C_1203 doesn't flag and rule NTL_CLK04 flags once for figure 10. Again NTL_CLK04 will no longer flag in 2006.06. However, NTL_CLK07 (gated clock) will now flag twice, once for each gate, whereas in 2006.03 it only flagged once.

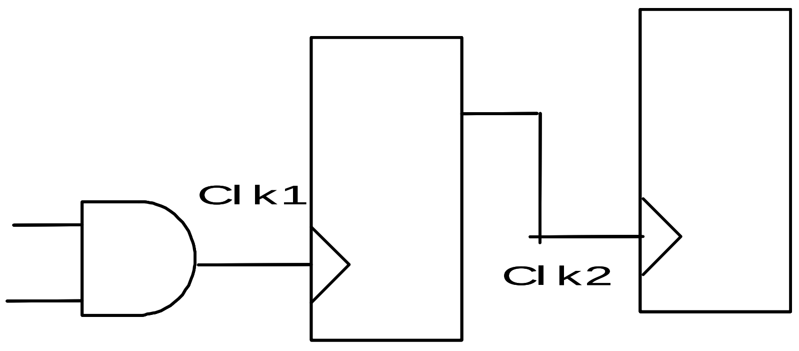

Figure 11: Another sample schematic

For figure 11, rule NTL_CLK04 flags twice, whereas rule C_1203 flags only once. Here, the behavior of rule NTL_CLK04 is correct as Clk1 is also an internally generated clock because, none of the signals in its input cone are clock origins. In version 2006.06, rule C_1203 will also flag twice.

DQL Function Updated

The function get_all_clock_origins returns a different list in this version 2006.06. This change is due to the implementation of unified clock inference. However, if you want to regenerate the original list as in previous versions, you will have to do the following:

```
for each clock origin CLK_ORG derived with current unified clock
enhancement {
  get all gates from CLK_ORG to GATE_LIST;
  if GATE_LIST is not null {
    get last gate in GATE_LIST to CLK_GATE;
    take output of CLK_GATE as REAL_CLK_ORG;
  } else {
    take CLK_ORG as REAL_CLK_ORG;
  }
  // Do further check/analysis on REAL_CLK_ORG;
}
```

License Feature

The leda_power feature applies to all leda_power commands (create_power_domain, infer_power_domain, use of $power, $isolate). However, the following message will no longer appear:

Warning: power commands are in beta

The leda_sdc feature is introduced for top vs block and equivalency checking. It is checked out when either:

read_constraints –block or read_constraints –for_equivalency are used. If the feature exists, Leda also print out one of the message:

Warning: SDC top vs block checks are in beta

Roadrunner Ruleset Name Changed

The Roadrunner ruleset of VCS policy is renamed as VCS PERFORMANCE ruleset.

Fixed STARS

Following are the STARs that were fixed since the last release (2006.03):

- 9000017744
- 9000038060
- 9000045025
- 9000062386
- 9000072256
- 9000077587
- 9000078608
- 9000083542
- 9000083863
- 9000084959
- 9000085402
- 9000086192
- 9000088123
- 9000090380
- 9000090814
- 9000091280
- 9000091310
- 9000091973
- 9000092185
- 9000092474
- 9000092980
- 9000094963
- 9000095280
- 9000095887
- 9000096089
- 9000096345
- 9000096498

- 9000096930
- 9000097690
- 9000098578
- 9000098660
- 9000098987
- 9000100490
- 9000100637
- 9000100639
- 9000101202
- 9000102849
- 9000102870
- 9000103135
- 9000103557
- 9000103669
- 9000103896
- 9000109494
- 9000110958
- 9000111316
- 9000111333
- 9000111336
- 9000111340
- 9000111354
- 9000111646
- 9000111778
- 9000112398
- 9000112910
- 9000113045
- 9000113125
- 9000114251

- 9000114355
- 9000114461
- 9000115802
- 9000115901
- 9000116710
- 9000117575
- 9000117576
- 9000119169
- 9000119236
- 9000119290
- 9000119420
- 9000119851
- 9000119861
- 9000120163
- 9000120302
- 9000120465
- 9000120631
- 9000121614
- 9000121736
- STS0143216

Supported Operating Systems

This release of Leda supports the following operating systems:

- Sun Solaris 32-bit
- Linux RedHat Enterprise 3.0 32-bit
- Suse 32-bit running on Intel Xeon - EM64T processor
- Suse 64-bit running on Intel Xeon - EM64T processor
- IBM-AIX 32-bit
- HP-UX 32-bit
- Linux RHEL 3.0 Opteron AMD 64-bit

Upgrading to Leda 2006.06

This release is incompatible with all previous versions of Leda. When you upgrade to the new Leda software, execute the checker in a clean directory that contains no rules, resources, libraries, or projects compiled with previous versions of the software.

Caution

Before you remove your old Leda installation tree, be sure to save any custom rule configurations specified in configuration files (for example, config.tcl), and any custom rule source code specified in .rl or .sl files, in a safe location, so that you can reuse your rule configurations and recompile your custom rules with the new software.

Installation and Documentation

For detailed installation, licensing, and configuration information, see the *Leda Installation Guide*. Note that you can find this book, and all other manuals in the Leda documentation set, in the $LEDA_PATH/doc directory after you install the software. For an overview of the Leda documentation that includes hyperlinks to all books in the set, see the *Leda Document Navigator* (navigator.pdf).

Software Packaging

This version of Leda is available as a compressed tar file leda_`version_architecture`.tar.Z where, *version* is a 6-digit number (concatenation of year and month, for example: 200606) for the tool version and *architecture is gccsparcOS5/ linux/ amd64/ rs6000/ hp32/suse32/sus364*.

Caution

Do not install Linux and UNIX versions of the Leda software in the same directory. They are not compatible. It is OK to have different UNIX platforms installed in the same directory (Solaris1, HP-UX, and AIX).

2 Release Notes - 2006.03

Introduction

These notes for the 2006.03 release of the Leda software are organized in the following sections:

New in Leda 2006.03

New Version Numbering

Leda has now moved to the Synopsys standard version numbering. So, the version number of this release is 2006.03.

New Leda Banner

Leda banner is now changed to reflect the change in version number. It now looks as shown below:

Leda (TM) Design & Constraint Checker

Version 2006.03 for sparcOS5 -- March 20, 2006

Copyright (c) 2000-2006 by Synopsys, Inc.

ALL RIGHTS RESERVED

This software and the associated documentation are confidential and proprietary to Synopsys, Inc. Your use or disclosure of this software is subject to the terms and conditions of a written license agreement between you, or your company, and Synopsys, Inc.

New platform Support

This version of Leda supports Suse32 and Suse64 on SUSE LINUX Enterprise Server 9 operating systems running on Intel Xeon - EM64T processors. This will not work on Suse running on other processors nor will it work on Suse versions older than 9.

New Command-Line Options

The following command-line options were added in this release:

- The -clock_file <file_name> option is used to specify the synchronous clocks in the design through set_clock_groups command. The checker uses this information for doing chip-level, netlist and SDC checks.

- The -sverilog option makes Leda parse and check language compliance for SystemVerilog. This option is added for compatibility with the VCS command line. This command line option now replaces the +sv and +sysvcs option available in Leda earlier.
- The -full_inf option is used to enable the display of deactivated rules and the violation summary in the .inf file.
- The -config_summary option is used to print the configuration summary on the console. The summary is displayed after the rules checking is done and is saved to $PWD/leda_config.log. This option is not available in the GUI mode.

For more information, see the *Leda User Guide*.

New Tcl Shell Mode Options

The following options were added in this release:

- The -block option, when used with the read_constraints command, reads the constraint file of a block.
- The -full_inf option when used with run and check command enables the display of deactivated rules and the violation summary in the .inf file.
- The print_config_summary command is used to print the configuration summary on the console. The summary is displayed after the rules checking is done and is saved to $PWD/leda_config.log.

For more information, see the *Leda User Guide*.

VeRSL Addition

New VeRSL local attributes were added to the following templates (see Table 11).

Table 11: Attributes in Templates

Template	Attribute
case_statement	case_exp_size_exceeding
casex_statement	casexz_exp_size_exceeding
casez_statement	casexz_exp_size_exceeding
binary_operation	in_assertion
blocking_assignment	lr_sign_match
non_blocking_assignment	lr_sign_match

Table 11: Attributes in Templates

Template	Attribute
procedural_continous_assign	lr_sign_match
procedural_continous_force	lr_sign_match
flipflop	synchronous_reset_signal
flipflop	synchronous_set_signal
flipflop	synchronous_load_signal
flipflop	asynchronous_reset_signal
flipflop	asynchronous_set_signal
flipflop	asynchronous_load_signal
always_construct	clock_in_condition
function_declaration	recursion_type
module_instantiation	keep_bits_order

For more information, see the *VeRSL Reference Guide.*

Enhanced Power Checking Technology

This release of Leda has enhanced the Leda Power checking technology in order to be compliant with the DC power domain and power net concepts, and to support Power/Ground netlists. This release includes a set of 35 new Tcl Power commands:

Attention
Power commands and checks are still in beta with release 2006.03. Some minor changes may be made to power-related definitions and commands of Leda to keep them in line with other tools power commands (e.g. DC, PT, Astro).

Attention
The Power Policy is currently in Limited Customer Availability (LCA) status. To use this policy, you need a separate Leda-power license feature. For more information, contact your sales rep.

- create_power_domain

- connect_power_domain
- get_power_domains
- remove_power_domain
- report_power_domain
- create_power_net_info
- remove_power_net_info
- report_power_net_info
- connect_power_net_info
- create_operating_conditions
- delete_operating_conditions
- report_operating_conditions
- set_operating_conditions
- get_cells
- get_nets
- get_pins
- get_port
- get_power_down
- get_power_down_ack
- get_power_cells
- getn_power_net
- get_nth_power_net
- get_power_net_type
- get_power_net_min_voltage
- get_power_net_max_voltage
- get_power_net_source_port
- enable_isolation_cell_recognition
- disable_isolation_cell_recognition
- reset_isolation_cell_recognition
- set_power_pin

- report_power_pins

The following Tcl commands are removed from this release:

- remove_voltage_domain
- report_voltage_domains
- set_power_domain
- set_voltage_domain

Two Tcl commands are modified in this release as shown below (see Table 12).

Table 12: Modified Tcl commands

Tcl commands	Modifications
report_power_domains	This command is now modified to report_power_domain.
remove_power_domain	The syntax and the arguments of this command is now changed to: `remove_power_domain <power_domain> | all`

For more information, see the *Leda User Guide*.

SystemVerilog Interface Support

This release includes support for full SystemVerilog interfaces.

Top vs. Block SDC Consistency Rules

This release features the new Top vs. Block Synopsys Design Constraints (SDC) consistency rules. You can use these rules to check for consistency in the usage of constraints between the top block and other sub blocks. These set of rules are grouped under a new ruleset named TVB in the Constraints policy (see Table 13). The documents are updated with the new rule descriptions and examples.

Note

These rules can be used only in the Tcl shell mode and cannot be used in the batch mode and GUI mode.

For more information, see the *Leda Constraints Rules Guide.*

Table 13: SDC Policy Consistency Ruleset

Label	Policy	Messages/Descriptions
SDC_TOP01	SDC	Block level clock constraint is inconsistent with top level clock constraint.
SDC_TOP02	SDC	Block level I/O delay constraint is inconsistent with top level I/O delay constraint.
SDC_TOP03	SDC	Block level false path constraint is inconsistent with top level false path constraint.
SDC_TOP04	SDC	Block level multicycle path constraint is inconsistent with top level multicycle path constraint.
SDC_TOP20	SDC	Block level max/min delay constraint is inconsistent with top level max/min delay constraint.

New Design Policy Rules

This release includes one new rules in the Design policy (see Table 14).For more information, see the *Leda Design Rules Guide.*

Table 14: New Design Policy Rules

Label	Message
NTL_STR97	Net connecting an analog and a digital pin detected: %s

Rule Messages Updated

This release includes a set of 15 enhanced rules in various policies (see Table 15). The messages and examples for these rules have been changed in the Leda documentation set. For more information, see the respective policy guides.

Table 15: Modified Rules in Various Policies

Label	Policy	Modifications
NTL_CLK17	DESIGN	The message of this rule has been changed to "Reconvergent path on clock tree detected: %s". The argument %s is the clock origin name. A new parameter ALLOW_CLOCK_ GATING_INTEGRATED_CELL that is set to 1 by default will allow this rule not to fire, when the clock path goes through a db cell having clock_gating_integrated_cell attribute and when a clock path goes through a pin having clock_gate_out_pin attribute.
NTL_CLK05	DESIGN	The secondary messages of this rule are: --- Target Clock: %s1 --- Source Register: %s2 --- Source Clock: %s3 --- Gate on path: %s4 --- Combinatorial path between registers: %s5 --- %s6 ... --- %sN Arguments: • %s1 is the hierarchy name of the clock origin of target flip-flop • %s2 is the hierarchy name of the source flip-flop • %s3 is the hierarchy name of the clock origin of source flip-flop • %s4 is the hierarchy name of the combinational gate (only one for example) on the data path, if there is any • %s5 - %sN is the hierarchy name of elements on the data path

Table 15: Modified Rules in Various Policies

Label	Policy	Modifications
NTL_CON01	DESIGN	Unconnected top-level input port %s1 %s1 is the hierarchy name of the primary input in top level. The source position is the line where the primary input is defined.
NTL_CON05	DESIGN	The message is now changed to: Some input pins tied together %s --- Input pin %s1 --- Tied pin %s11 ... --- Tied pin %s1n ... --- Input pin %sn --- Tied pin %sn1 ... --- Tied pin %snn ... Arguments: • %s is the hierarchy name of instance • %s1 is the hierarchy name of the first input pin that is tied • %s11 is the hierarchy name of the first input pin that is tied to %s1 • %s1n is the hierarchy name of the nth input pin that is tied to %s1 • %sn is the hierarchy name of the nth input pin that is tied • %sn1 is the hierarchy name of the first input pin that is tied to %sn • %snn is the hierarchy name of the nth input pin that is tied to %sn

Table 15: Modified Rules in Various Policies

Label	Policy	Modifications
NTL_CON15	DESIGN	The message of this rule has been changed to “Power rails belonging to different supply types should not short %s1”. Arguments: • %s1 is the falsely connected supply pin. The source position of this rule is the line in which the pin is instantiated. A new parameter STRICTLY_CHECK, if set to TRUE, will check if the signal is connected to another signal of same polarity. For example, signal VDD connected to VDD, signal GND connected to GND and signal VDD(A) is connected to signal VDD(A).
NTL_CON18	DESIGN	The message of this rule has been changed to “There is output port, but it is not connected internally %s1”. Arguments: • %s1 s1 is the hierarchy name of the output port in the instance in which the port is not connected internally. The source position of this rule is the line in which the output port is defined. The secondary message of this rule is “Output port connectivity path: %s2”. Arguments: • %s2 is the hierarchy name of the other tied output port.
NTL_RST03	DESIGN	The message of this rule has been changed to “All registers must be asynchronously set or reset %s1”. Arguments: • %s1 is the non-asynchronous flip-flop The position is the line in which the non-asynchronous flip-flop is instantiated. The secondary message of this rule is “Other instance %s2”. Arguments: • %s2 is another non-asynchronous flip-flop which is instanced the same instance %s1.

Table 15: Modified Rules in Various Policies

Label	Policy	Modifications
NTL_RST04	DESIGN	The message of this rule has been changed to "Reset/set must not be used as data %s" ---Location where the signal is used: %s0 ---Element on path: %s1 .... ---Element on path: %sN ---Location where the signal is used as reset/set: %sN+1 Arguments: • %s is the hierarchy name of the set/reset signal. • %s0 is the hierarchy name of output of the sequential element that uses the set/reset signal as data. • %s1 is the hierarchy name of No. 1 element on the path from set/reset signal to the sequential that uses it as data. • %sN is the hierarchy name of No. N element on path from set/reset signal to the sequential that uses it as data. • %sN+1 is the hierarchy name of output of the sequential element that uses the signal as set/reset.
NTL_RST06	DESIGN	The message of this rule has been changed to "Avoid internally generated resets %s1". Arguments: • %s1 is the hierarchical name of the reset signal. A new parameter CHECK_ASYNCHRONOUS_RESET_ONLY if set to 1, can be used to check internally generated asynchronous reset signals only.

Table 15: Modified Rules in Various Policies

Label	Policy	Modifications
NTL_STR08	DESIGN	The message of this rule has been changed to “The number of gates instantiation should not exceed <NB_MAX_GATES> in instance: %s1” Arguments: • %s1 is the hierarchy name of the instance. The position is the line where the instance is used. Secondary messages: • The number of instantiated gates in the instance is %d1: %s2. • The instantiated gate: %s3. Arguments: • %d1 is the total number of instantiated gates in the instance. • %s2 is the hierarchy name of the instance. • %s3 is the hierarchy name of the detected gate.
SDC_IDL06	SDC	The message is now changed to “Input constrained versus wrong (real) clock: Port: %s1, clock (wrong): %s2”. The secondary messages of this rule could be one of the following: • Right clock is %s3 • Right clock source is (this does not have a corresponding SDC clock definition) %s4 • This port drives only combinatorial paths %s5
SDC_ODL06	SDC	The message is now changed to “Output constrained versus wrong (real) clock: Port: %s1, clock (wrong): %s2”. The secondary messages of this rule could be one of the following: • Right clock is %s3 • Right clock source is (this does not have a corresponding SDC clock definition) %s4

- Rule G_551_1_A is removed from the RTL and Leda-classic prepackaged configurations.

For more information, see the respective policy guides.

Obsolete Rules

The following are the list of formality rules that are obsolete and so they are removed from the formality policy. For more information, see the *Leda Formality Rules Guide.*

Table 16: Obsolete Formality Policy Rules

Label	Policy	Messages/Descriptions
FM_2_34A	Formality	Do not use ‘ifdef.
FM_100	Formality	Do not use generate.
FM_103	Formality	Do not use multi dimensional array
FM_104	Formality	Bit and part selects within array not allowed.
FM_109	Formality	Do not use combinational logic sensitivity token @(*).
FM_110	Formality	Do not use comma separated sensitivity list.
FM_115	Formality	Do not use combined port and data type declarations.
FM_116	Formality	Do not use ANSI C style port declaration.
FM_118	Formality	Implicit net declaration is not supported.

Fixed STARS

Following are the STARs that were fixed since the last release (4.2.1):

- STS0177581
- 9000025374
- 9000038582
- 9000040231
- 9000046630
- 9000047169
- 9000049864
- 9000052407
- 9000053642
- 9000054403

- 9000054849
- 9000055360
- 9000056608
- 9000060454
- 9000064476
- 9000069264
- 9000071021
- 9000072289
- 9000073637
- 9000076986
- 9000077221
- 9000078575
- 9000078663
- 9000079521
- 9000080449
- 9000083864
- 9000083866
- 9000088165
- 9000089422
- 9000089466
- 9000089991
- 9000090171
- 9000090753
- 9000090974
- 9000091804
- 9000092444
- 9000092447
- 9000092981
- 9000093081

- 9000093325
- 9000093949
- 9000094123
- 9000094185
- 9000094222
- 9000094897
- 9000095676
- 9000096767
- 9000096885
- 9000096887
- 9000096957
- 9000097280
- 9000098544
- 9000098830
- 9000099216
- 9000099682
- 9000099885
- 9000100140
- 9000100149
- 9000100242
- 9000100653
- 9000100954
- 9000101431
- 9000101598
- 9000101801
- 9000102111
- 9000102539
- 9000102763
- 9000103324

- 9000103447
- 9000104519
- 9000105088
- 9000106221
- 9000106949
- 9000107848

Compiler Versions

For compiling C-based custom SDC or Netlist checker rules, you must use the same compiler version that was used to build Leda. The following compiler version must be used with the 2006.03 release for the following operating system.

Table 17: Compiler Versions for Operating Systems

Operating System	Compiler Version
Sun Solaris	GNU 2.95
Linux RedHat Enterprise 3.0	GNU 2.9.6
Linux RHEL 3.0 Opteron AMD 64-bit	g++ 3.2.3
SUSE	gcc3.3.3
HP-UX	A.03.33

For more information, see the *Leda C Interface Guide.*

Upgrading to Leda 2006.03

This release is incompatible with all previous versions of Leda. When you upgrade to the new Leda software, execute the Checker in a clean directory that contains no rules, resources, libraries, or projects compiled with previous versions of the software.

Caution

Before you remove your old Leda installation tree, be sure to save any custom rule configurations specified in configuration files (for example, config.tcl), and any custom rule source code specified in .rl or .sl files, in a safe location, so that you can reuse your rule configurations and recompile your custom rules with the new software.

Installation and Documentation

For detailed installation, licensing, and configuration information, see the *Leda Installation Guide*. Note that you can find this book, and all other manuals in the Leda documentation set, in the $LEDA_PATH/doc directory after you install the software. For an overview of the Leda documentation that includes hyperlinks to all books in the set, see the *Leda Document Navigator* (navigator.pdf).

Software Packaging

This version of Leda is available as a compressed tar file leda_*`version_architecture`*.tar.Z where, *version* is a 6-digit number (concatenation of year and month, for example: 200603) for the tool version and *architecture is gccsparcOS5/ linux/ amd64/ rs6000/ hp32/suse*.

Caution

Do not install Linux and UNIX versions of the Leda software in the same directory. They are not compatible. It is OK to have different UNIX platforms installed in the same directory (Solaris1, HP-UX, and AIX).