Insight Analyzer Version 4.90
Release Notes

4.90-10748
2020 January 10

Contents

Overview

This version of the Insight Analyzer has significant enhancements and changes over prior versions. There are new features, behavior changes, and migration aspects.

Updates & Changes

Setup

Save Settings

While the saved project settings script remains generally the same as in previous version, there are some slight modifications, as described below.

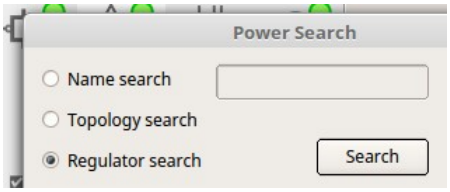
- The saved settings script no longer writes the legacy ‘devparams’ commands for versions older than v4.7.
- The save settings command has been updated for better interplay with optional UPF that may have been generated or loaded.

GUI

This is the last version series that will be using the v4 GUI. Some additions have been made, as described in this section.

Regulator Search

In the section for Power setup, the Search utility now has an option to use Regulators as a criteria. By default, the selected criteria is Topology. This new method can help to find smaller regulated rails that might otherwise go undetected during setup.

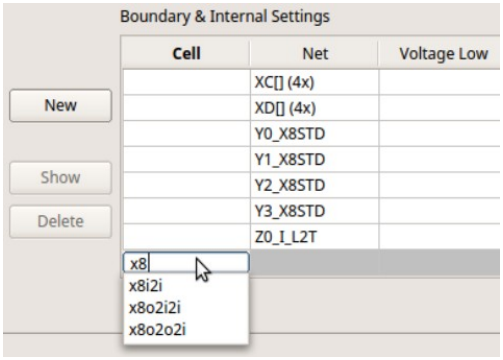


"Regulator search" in power setup

As with all power definitions, a regulator is defined by cell and net. If the defined regulator is a cell IO to the parent level, the effect of the regulator will propagate upward appropriately. Example: An LDO regulator can be defined as by the LDO output port, and the upper level net (above the definition) will effectively become the whole power supply.

IO & Internal Node Setup

This version adds support for definition of voltage values on internal nodes. This is generally reserved for special cases, such as locking a mode bit, setting a fixed reference value, etc.



Adding a cell / net value definition.

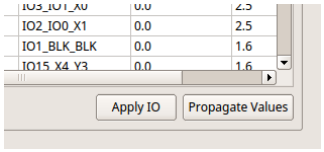
Definitions are based on cell plus net. In the GUI, you begin in the Boundary settings table. Press “New” to create a new row. In the Cell field, begin typing the name of the cell that holds the desired net. Command completion will automatically pop up a list of cell name matches. After choosing the cell, you would then enter the net name in the same way.

If using script commands, those are the same as prior versions. For example:

```
cdb net x82i-net16 setSpecVolts 0.3 1.1
```

Applying Settings

There are now two buttons in this window for applying settings.



Apply Settings

Apply IO: This button saves the assignments that have been made in the “Boundary & Internal Settings” table. It applies the commands

'cdb net _ setSpecVolts' and 'setSpecRails'. You must press this button for any of the table entries to take effect.

Propagate Values: This button forces a propagation of values into the circuit, and may take some time to complete. It calls the command 'cdb settleVoltValues'. Values are carried forward (input to output) for each stage where only a single output value could result. This is an optional step, and is usually not recommended.

The "Propagate Values" button above is generally used only in special cases. For example: An analog network of passgates and resistor dividers achieves constant values when controlled by two IO input pins. Normally, without using this function, the circuit values would be determined by reverse propagation (query internally, work outward to IO, yield min/max values). In contrast, with this function, forward propagation can be better at forcing single values where appropriate, such as resistor dividers and stuck logic.

Report Views

Waiver Comment Display

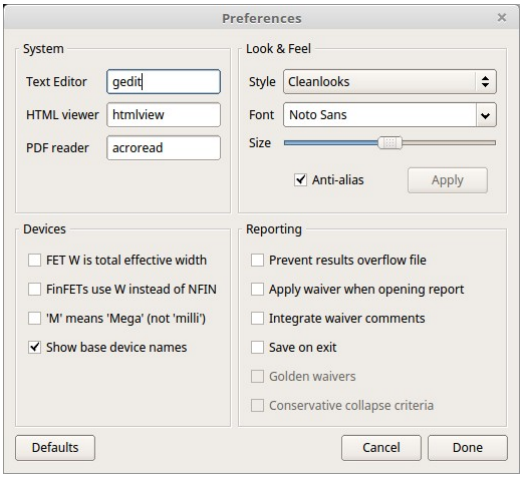
It is now possible to display waiver comments as part of the main report view. Please see the section on Waivers for more information.

Info Column Display

It is now possible to add informational text in a new report column. Please see the section on Reporting for more information.

Preference Settings

New preference settings have been added to GUI controls, under "Setup > Preferences".



Setup > Preferences

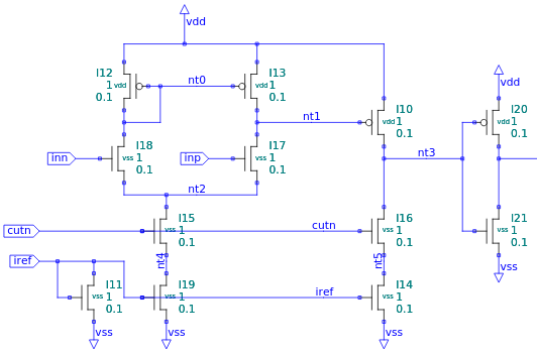
- (Prevent results overflow file: Legacy option is not described here.)
- Apply waivers when opening report: To prevent long delays in re-loading large reports with many waivers, this option can be cleared. Please see section on Reloading Reports for more information.
- Integrate waiver comments: When viewing reports, the waiver comments are displayed as an additional column. Please see section on Integrate Waiver Comments for more information.
- (Save on exit: Legacy option is not described here.)

- Golden waivers: This control is read-only, showing the current state of this option. Please see section on Golden Waiver Format for more information.
- Conservative collapse criteria: This control is read-only, showing the current state of this option. Please see User Manual for description of ‘project preference report_collapseAddSubscripts’.

Plugins & Applications

Floating Nodes

The Floating Nodes application has been substantially updated, to deliver improved discrimination, better handling of analog topologies, and more extensive support for digital receiver and power state combinations.



Amplifier floats at input to next stage (missing a common cutoff).

Building Block Topologies

Prior versions were based on a generic solving approach, with transistors taken individually and limited consideration of the overall circuit context. This version adds recognition of common circuit structures, the building blocks that make up almost all analog circuits. This enables a focused approach in the process of chasing hypothetical floats.

For more information on the analog building blocks, please see the section “API: Circuit Building Blocks”.

Analog Cases

Impact on handling of analog cases includes the following examples:

Constant current: When a stack of MOSFETs are configured as diodes, current is constantly flowing, and float violations are avoided. These topologies are identified ahead of time, and can include resistors and BJT transistors as well. Violations can be reduced in reference generators and other self-biasing loops.

Current Mirrors: There is a significant reduction in false positives in current mirrors. A current mirror with a single follower and common current sink or source is no longer reported as a conditional float. However, a current mirror with multiple followers (and hence, multiple stacks), is fully tested for conditional float and will yield a violation if appropriate.

Amplifiers: An amplifier with floating output is a particular risk that is now more accurately pinpointed, as the topology is identified ahead of time. If the receiving MOSFET is still powered at the time of the float, then a violation is issued.

Common cutoff: In the example of an amplifier (above), if the receiving MOSFET has power cut at the time of a float, then a violation is avoided. This can result in significant reduction of false positives over prior versions. The same reduction is applied to other cases as well (not limited to amplifiers). A common cutoff control will isolate victim MOSFETs, avoiding the float violations

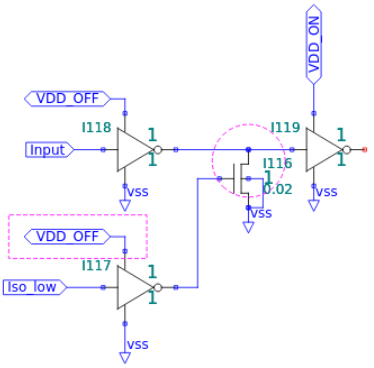
of prior versions.

Receivers and Isolation

IO definitions: Minor updates have been made with respect to the definition of incoming signals (external drivers) and supply names and states. Prior functionality is not changed, but some corner cases have been made to work better than in the past.

Clamps and isolation: In this version, single transistors are more completely supported as forms of isolation. A transistor that receives a valid isolation control will be recognized as a clamp, and prevent a float violation as appropriate. For completeness, here is a listing of the supported protections for digital receivers:

- Single MOSFET, gated by valid control signal in the state appropriate for the N / P transistor type.
- NAND or NOR stack, gated by valid control signal in the appropriate state for the given topology.
- Isolation cell, gated by valid control signal in agreement with cell definitions (UPF).



Complete checking of off clamps, controls, and state combinations.

Control propagation: In this version, extensive support has been added for the many combinations of power states, buffer types, high / low state consideration. Example: An active-high isolation control can effectively pass through an inverting buffer, yielding an active-low output, even if the inverting buffer is powered by an off power supply. This will avoid violations that may have been issued by prior versions. Correspondingly, if the same off buffer is used for a control of the opposite state, that would be invalid, and a violation would still be issued. These aspects also impact the plugin “Isolation Network”.

Floating Nodes | Floating Nodes

	Violation	Priority	Topology	xell	Net	Key Nodes
1	Float, conditional	1	Mirror	3xNDGxP_90N...	/XBxNDGxP/X...	xXT_BGxNB, SOx_MODx
2	Float, conditional	1	Mirror	TxDxT_BIxS	/XTxDxT/XBIx...	MxSTxRPU, PBIxSIPP, PxxSIPP, Tx_THR_DIV2
3	Float, conditional	1	Mirror	/xLxMP	/XVxLxMP-NB	
4	Float, conditional	1	Logic, Loop	5xR_LxTxH	/XSxRxDx/Xx...	BNM2
5	Float, conditional	1	Logic	TxDxT_BIxS	/XTxDxT/XBIx...	MxSTxRPU, PBIxSIPP, PxxSIPP
6	Float, conditional	1	Logic	TSxN	/XTSxN-BNM	N_6, PBIxSIPP, PxxSIPP
7	Float, constant	1		5xR_PRxxMP1	/XSxRxDx/Xx...	
8	Float, constant	1		5xR_LxTxH	/XSxRxDx/Xx...	
9	Float, constant	1		5xR_PRxxMP2	/XSxRxDx/Xx...	

New columns in Floating Nodes report.

Reporting

In this version, two new columns have been added to violations table view. This makes simple ‘diff’ of reports invalid. However, waiver keys are fully compatible with the prior version.

Topology: The following keywords indicate topology characteristics where applicable. These keywords can be sorted or filtered as desired:

- Amplifier: The node is part of an amplifier core.
- FwdStack: The node is a link within a diode / resistor stack from power to ground. This is rare in float cases, but could be displayed if the node connects to gates in other stacks, and the forward stack can be cut by a series transistor.
- Mirror: The node is the gate of the diode-tied transistor forming a current mirror. Mirror nodes can be reported as violations where the current mirror has additional slaves beyond the first follower stack.
- Logic: The node is an input to some logic function (not necessarily a CMOS stack).
- Loop: The node is part of a loop, such as a feedback path in an amplifier, or a latch.
- Latch: The node is the storage point of a recognized latch.

Key Nodes: The input nodes to a particular violation are displayed in this dedicated column. An "input" is a node where the circuit exploration stopped, such as an external IO pad, a dangling node, a hard tie, resistor divider, or a flip flop output. By reviewing the names in this column, you might recognize a broad pattern that can be fixed in setup, such as an external "power down" input that should be defined. Or, it might be a way to identify violations due to a common known issue that could be waived all at once.

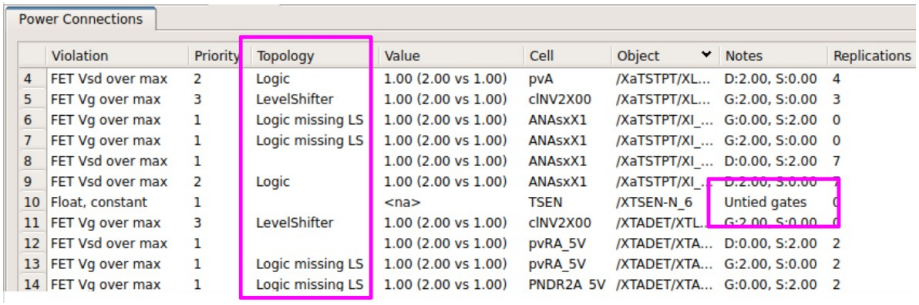
Settings

Setting options have changed for this application. The slider control has been removed, and new flags have been added. Legacy run scripts are still supported (legacy settings are read-only).

Setting	*Description*
-analogScenarios	**New**: Generic check for circuits aside from digital receiver isolation. Formerly effort >= 2.
-checkDomainIsolation	**New**: Check digital receivers for domain isolation. Formerly was done at all effort levels.
-checkLatchNodes	**New**: Include recognized latches in checking (more strict)
-checkTopInputs	Report top IO gate-only nodes as float violations.
-scanReplicationFactor	**New**: Increases scan repetition with value.
-replicateFlatViolations	**New**: Formerly effort == 4.
-reportEverything	Add violations for unresolved cases.
-effort	Read-only: Legacy setting.
-domainCrossing	Unused flag
-railCrossing	Unused flag

Power Connections

This application has received minor updates. Essential functionality has not changed. Report files can not be compared against those of past versions by simple 'diff'. Waivers remain fully compatible.


Power Connections

	Violation	Priority	Topology	Value	Cell	Object	Notes	Replications
4	FET Vsd over max	2	Logic	1.00 (2.00 vs 1.00)	pvA	/XaTSTPT/XL...	D:2.00, S:0.00	4
5	FET Vg over max	3	LevelShifter	1.00 (2.00 vs 1.00)	clNV2X00	/XaTSTPT/XL...	G:2.00, S:0.00	3
6	FET Vg over max	1	Logic missing LS	1.00 (2.00 vs 1.00)	ANAsxX1	/XaTSTPT/XI_...	G:0.00, S:2.00	0
7	FET Vg over max	1	Logic missing LS	1.00 (2.00 vs 1.00)	ANAsxX1	/XaTSTPT/XI_...	G:2.00, S:0.00	0
8	FET Vsd over max	1		1.00 (2.00 vs 1.00)	ANAsxX1	/XaTSTPT/XI_...	D:0.00, S:2.00	7
9	FET Vsd over max	2	Logic	1.00 (2.00 vs 1.00)	ANAsxX1	/XaTSTPT/XI_...	D:2.00, S:0.00	7
10	Float, constant	1		<na>	TSEN	/XTSEN-N_6	Untied gates	0
11	FET Vg over max	3	LevelShifter	1.00 (2.00 vs 1.00)	clNV2X00	/XTADET/XTL...	G:2.00, S:0.00	0
12	FET Vsd over max	1		1.00 (2.00 vs 1.00)	pvRA_5V	/XTADET/XTA...	D:0.00, S:2.00	2
13	FET Vg over max	1	Logic missing LS	1.00 (2.00 vs 1.00)	pvRA_5V	/XTADET/XTA...	G:2.00, S:0.00	2
14	FET Vg over max	1	Logic missing LS	1.00 (2.00 vs 1.00)	PNDR2A_5V	/XTADET/XTA...	G:0.00, S:2.00	2

New column and indicators in Power Connections.

Topology Indicators

When a violation is issued, additional topology info is now included in a new column. This helps a user to identify which violations may be more important than others.

The following keywords indicate topology characteristics where applicable. These keywords can be sorted or filtered as desired:

- Logic: The given device instance is part of a recognized logic or CMOS structure. The former practice of embedding this as part of the "Priority" column has been dropped.
- MirrorSlave: The instance is a transistor that is gated by a current mirror. This can often indicate that the violation is safe, but still must be manually reviewed.
- FwdStack: The instance is a resistor, diode, or transistor in the DC path of a constant current stack.
- LevelShifter: The instance part of a recognized level shifter pattern or a defined level shifter cell.
- IsolationCell: The instance is part of a defined isolation cell.

Notes Column

In contrast to prior versions, the "Notes" column now shows additional helpful information.

- Missing <block name>: For a float violation, it can be determined that the cause of float is an empty or missing block cell. In this case, the net is tied to MOSFET gates only, and also connects to an instance of a cell that either has no contents or is missing from the netlist. This can commonly happen when a place-and-route block is omitted from the workflow.
- Untied gates: Similar to the "missing cell" case above, the net connects MOSFET gates only, but does not have any other destination.

Avoiding Cases

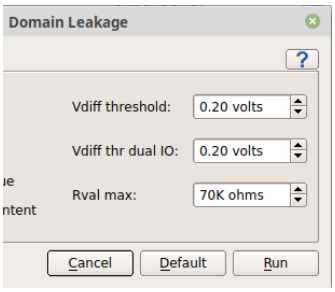
New reductions are applied in the following area: Transistor over voltage, as part of an analog bias network, where it can reasonably be assumed that the transistors in question are in a safe steady state. Such transistors can be parts of current mirrors or diode tied stacks. In many cases, the reduction results in the outright removal of a violation.

In some cases, a borderline transistor will be reported, but will be accompanied by a topology mark such as “MirrorSlave” in the results view. A user can now more easily find these borderline cases among other results.

Domain Leakage

The Domain Leakage application has received some updates in this version, as described in the following sections.

Threshold for IO Cases

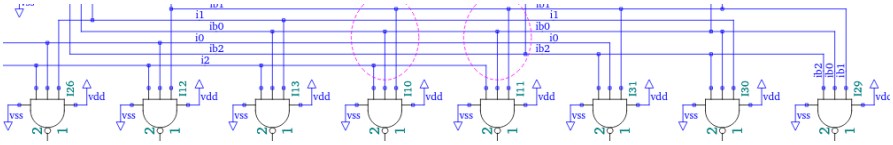


Threshold setting for dual IO

In cases where power is provided externally to both sides of a MOSFET, risk may be increased due to off chip noise or other uncontrollable variations. This has been demonstrated to pose an actual risk of activation of parasitic SCR in silicon. During checks, voltage definitions were “clean” and did not fall within the normal threshold of failure provided by -vDiffThreshold.

In this version, a new threshold setting is provided for situations where both sides of a MOSFET are exposed to external IO. In the GUI, the setting appears with the standard threshold setting. The default value is 0.2v. For strict handling of these dual IO cases, the threshold can be set as low as 0.0v.

Setting	*Description*
-vDiffDualIOthreshold	**New**: Threshold for cases where both source and drain of MOSFET are exposed to external values from IO.

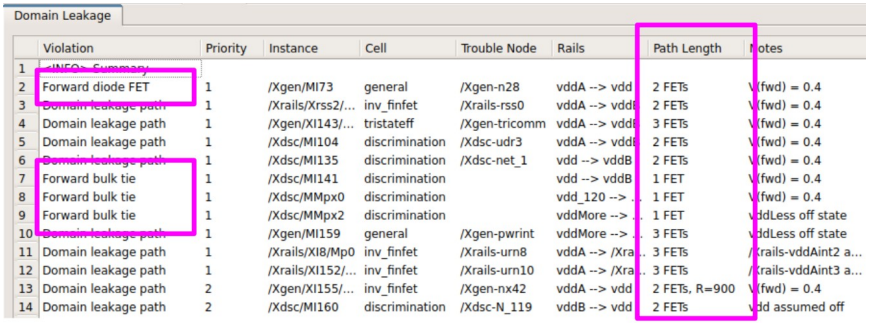

Multiplexer decode logic

One-Hot Multiplexers

Test multiplexers are a common source of false positives in any tool that traces DC paths between power domains. A test multiplexer will expose theoretical paths between different power supplies, as various output values can be sampled through passgate networks. In theory, these DC paths could cross conduct between supplies, but the actual decode logic normally prevents this. Classic DC tracing tools are not able to consider decode logic, and hence would normally issue false positives.

Insight Analyzer addresses this classic problem. This version adds state-based awareness to the decode logic that drives passgate arrays. For multiplexer structures with topologically obvious decoding within the circuit, this version is able to eliminate false positives in the Domain Leakage application.

For related information, please see the section in API under State Based Commands.



New wording and column in Domain Leakage report.

Reporting

The report format is changed in this version. If your workflow depends on the report format remaining unchanged, you can disable the change with this setting:

```
runenv setHook domLeak_v4p9reporting 0; # maintain v4.8 domain leakage reporting
```

Assuming you do not make the above setting, the following changes appear with this version:

A new column has been added in the results, to show indications of DC path length. This column is named “Path Length”. The presence of this column means simple ‘diff’ against prior version report files will be invalid. There is no impact on saved waivers. The waiver workflow is fully compatible.

Violations from this application are described with updated wording.

Here is an overview of the wording changes:

	v4.8	*v4.9 (more granularity)*
Violation column	“Domain leakage” *(all cases)*	“Domain leakage path” *(most cases)*
		“Forward bulk tie” *(single device path)*
		“Forward substrate” *(single device path)*
		“Forward diode FET” *(sink device is wired as diode)*
		“IO leakage path” *(source or sink at external IO)*
Notes column	“Off (assumed)”	“vLDO assumed off”
Details pop-up: Various wording improvements, such as shown here	“Current path (high V)”	“Current path (high, 2v)”
	“Forward bias leakage in MOSFET” *(wording for most cases)*	“MOSFET configured as diode may be sinking current.” *(where appropriate)*
		“DC path exists from supply rail to a current sink.” *(where appropriate)*
		“MOSFET forward conducting from S/D

		rail to Bulk tie.” *(where appropriate)*

Fanout

While the Fanout application remains functionally the same as in the previous version, the settings for this application have changed, with greater control over the basic internal scanning for subject nets.

Setting	*Description*
-fastScanning	**Legacy setting**: Now made read-only. If set to “1”, scan only the first appearance of each hierarchy branch. If set to “0”, scan all hierarchy branches (all replicated instance patterns).
-scanReplicationFactor	**New setting**: If set to “1”, scan only the first appearance of each hierarchy branch. If set to “0”, scan all hierarchy branches (all replicated instance patterns). *If set to values above 1, scan increased numbers of hierarchy branch replications.* This setting has no effect if -fastScanning is set (legacy support). This setting corresponds to ‘cscan … -limitRepBranches’.

Contention

This checking application is based on the same underlying API as the Floating Nodes application. The underlying API has been re-implemented, and does impact this application.

In this version, results of the Contention application may vary from prior versions, due to differences in the handling of borderline cases such as latch feedback circuits.

Settings for this application have changed slightly, with greater control over the basic internal scanning for subject nets.

Setting	*Description*
-fastScanning	**Legacy setting**: Now made read-only. If set to “1”, scan only the first 4 appearances of each hierarchy branch. If set to “0”, scan all hierarchy branches (all replicated instance patterns).
-scanReplicationFactor	**New setting**: If set to “1”, scan only the first appearance of each hierarchy branch. If set to “0”, scan all hierarchy branches (all replicated instance patterns). *If set to values above 1, scan increased numbers of hierarchy branch replications.* This setting has no effect if -fastScanning is set (legacy support). This setting corresponds to ‘cscan … -limitRepBranches’.

Isolation Network

This checking application has been updated for more complete handling of various corner cases. In general, it is still recommended to run these checks in conjunction with Floating Nodes, for chasing cross domain float risks.

Level Shifters

This auditing application has been updated for better reporting of various corner cases, such as the way input signals are listed when they don't have any power.

In addition, updates in the base API that detects level shifters and isolation cells may impact the results of this audit.

Rail Definitions

This check / audit application has been added to the general package. It is used for sanity checks on power definitions, prior to running detail checks such as Power Connections.

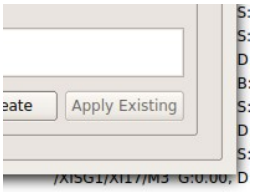
Reporting & Waiving

Reloading Large Reports

There are situations where re-loading a large report can take time in the GUI. This can happen when thousands of violations and waivers are combined. The GUI may pause with "Applying waivers" and a progress bar. (The use of redundant, overlapping, and wildcarded waivers can cause this situation.)

To sidestep this problem, a user preference option is provided under Setup > Preferences: "Apply waivers when opening report". If this option is disabled, the application of waivers is postponed until the user explicitly chooses to apply the waivers. The internal command is:

```
project preference applyWaiversOnOpen 1/0
```



Apply waivers button

In the Waivers dialog window, a new button is provided to "Apply waivers". To apply pre-existing waivers to a loaded report, press this button.

This is only used if the above preference is set to disabled, and a loaded report file was originally saved prior to the existence of new waivers.

New Columns

Reports for Floating Nodes, Domain Leakage, and Power Connections have new columns. Saved report files can not be compared by simple 'diff'. There is no impact on waiving or saved waivers,

which remain fully compatible. See the respective plugins for more details.

Adding Information Column

Starting in this version, a new column can be added to any report. In this column, you can place any text as desired.

This works through the report line item qualifier callback. Previously, this callback was used to modify or suppress reported line items. Now, the callback may also inject text into a new column.

The new column can be added to reports as they are generated (during a run), and after they are generated (when re-loading a saved report).

The process works as follows:

1. You provide a script callback that intercepts data rows just as they are being sent to a report (saved file or GUI view). This is '::reportQualifier_' followed by the name of the application.
2. Before sending any rows to your report (before running an application or loading a report file), use the command 'rptif setComments 1'
3. Start an application scan (real time data), or load a report file loaded (saved data). If loading, it is done with the command 'rptif openFile'
4. In the script callback, per each line, you add a Tcl 'dict' field for Comment. The new column will appear in the GUI and/or report file, as appropriate.

For more information, please see the working files under
examples / internal / reporting / comments.

Integrate Waiver Comments

This version adds the ability to view waiver comments in the report view, adjacent to each violation or line item. In this mode, a column appears on the right side of a report in the GUI. For any report line that is waived, the waiver's comments are displayed in this column. The displayed text comes from the waiver database only, and is not individually addressable as a part of the line item data.

Select and Enable

You may decide to set or clear this optional behavior at any time during a session. New reports that are created after changing this setting will either have or not have the additional display column, as appropriate. However, changing this setting while reports are already viewed in the same GUI session can result in undefined behavior.

The option can be set by either of the following:

- In the GUI, with checkbox: “Setup” > “Preferences” > “Reporting : Integrate waiver comments”.
- In script, with command: ‘project preference report_showWaiverComments 1/0’

Create Waivers

The creation of waivers is independent of having this extra column displayed. You may create waivers as normally done. If the column is visible, comments that you attach to a waiver will be shown when you apply the given waiver.

Save Reports

Saved reports do not contain this extra column (although a saved report can include the waiver and comments as part of the report data).

Golden Waiver Format

Starting in this version, an optional new workflow can be used for waivers. The workflow provides a “golden” centralized, read-only source of waivers.

In contrast to the legacy method of working with waiver data:

- File format supports editing, sorting, and version control, as waivers are stored line by line.
- All waivers are stored in a single file, rather than separated by application name and cell name.
- A user’s waivers are merged into the golden source by an approval process, after a GUI session.

```
#Application   RootCell UserName TimeStamp   Comment   Key
Power Connections top    Evren Oct 8 2019   created   PwrGate,1.00(2.00vs1.00),*=top/=
Power Connections top    Evren Oct 8 2019   created   PwrMax,1.00(2.00vs1.00),*=top/=
Power Connections top    Evren Oct 8 2019   two end   PwrMax,1.00(2.00vs1.00),*=top/=
Power Connections top    Evren Oct 8 2019   two end   PwrMax,1.00(2.00vs1.00),*=top/=
Power Connections top    Evren Oct 8 2019   two end   PwrMax,1.00(2.00vs1.00),*=top/=
Level_Shifters top   Evren Oct 8 2019   level shifter waivers   OK:LevelShifterbytop
Level_Shifters top   Evren Oct 8 2019   level shifter waivers   OK:LevelShifterbytop
```

Format of saved golden waivers.

Select and Enable

This workflow should be enabled prior to working with reports and waivers. The following command will enable this workflow:

```
project preference waiver_goldenFormat 1
```

Create Waivers

Users create waivers as usual, in the GUI. From a user’s perspective, this part is apparently the same as with the legacy workflow.

User’s waivers are stored in the legacy waiver directory, but in a single file named UserLocal.filters. The format is line by line, delimited by tabs, as shown. Each line begins with the application name (such as “Power Connections”), and the top cell name. Following line data completes the waiver with user name, date, comments, and collapsing key.

Integrate Waivers

Golden waivers exist in a single file named Golden.filters. The user’s waivers are integrated with golden waivers by simple file manipulation, outside of Insight Analyzer. The lines may be freely added, deleted, sorted using common text manipulation methods.

A user will see the effect of golden waivers, but will not be able to delete golden waivers from a GUI session. The user may delete only their own waivers that have not yet been integrated into the golden set.

API

Circuit Building Blocks

Insight Analyzer has included recognition of subgraph isomorphism, or “pattern matching”, for many years. This has been based around the idea of a user text file that can be updated to define new patterns of interest. Now, in addition to the user defined pattern matching, Insight Analyzer adds built in recognition of common analog circuit structures. These circuit “building blocks” are present in almost every analog circuit. Their recognition is key to the contemporary enhancements in Power Connections and Floating Nodes applications.

Following are the analog building blocks available through the API:

	Description	*Members*
FETdiode	Single MOSFET wired as diode	inst, anode net, cathode net
FETcap	Single MOSFET wired as capacitor	instance, gate net
LongFET	Stack of homogeneous MOSFETs operating together as one.	series instances, source pin, drain pin
LongFETdiode	Diode formed of a LongFET	lead instance, series instances
CurrentMirror	MOSFET diode with at least one follower MOSFET	diode instance, follower instance, clamp off instance, additional slave follower instances
ForwardStack	Stack of MOSFETs configured to sink continuous current	anode net, cathode net, gate net segments
AmplifierCore	Central core of differential amplifier	CurrentMirror, master side input FET, follower side input FET, common current node, common current FET instance, additional current cutoff instances

Access

The building blocks are accessed through script commands as follows:

cdb cell _ childBldgBlocks
From the given cell, read out a list of each building block contained within. These are building blocks existing as children of the given cell, but not descendants further below. An example of this would be:

```
join [cdb cell COMP03 childBldgBlocks] \n

-type AmplifierCore -mirror {-type CurrentMirror -subType enclosed -masterInstance MI_16 -
```

instances {MI_15 MI_16} -gateNet VTOOLNB} -masterInputGate MP7-G -slaveInputGate MP8-G -refStackGateNets {COMP03-ICOMP} -refStackType PFET -masterCurrentLeg VTOOLNB -slaveCurrentLeg TOOLOW_1ST

-type CurrentMirror -subType enclosed -masterInstance MI_16 -instances {MI_15 MI_16} -gateNet VTOOLNB

-type FETdiode -coreInstance MI_11 -gateNet ICOMP

-type LongFET -instances {MP1 MP3} -gateNet VREF03

cdb net _ isCurrentMirror
Indicate whether the given net is the diode / gate of a current mirror.

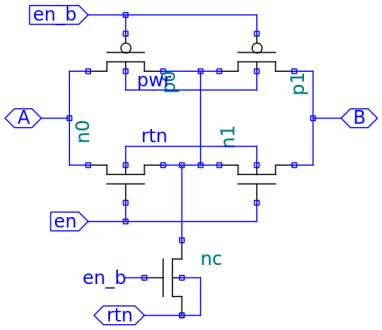
cdb net _ isForwardStack
Indicate whether the given net is part of the chain of segment nets in a forward conducting stack.

cdb inst _ isCurrentMirrorSlave, isCurrentMirrorMaster
Indicate whether the given instance is a MOSFET in the role of current mirror follower device or master diode device.

Recognition

Support for recognizing variations on passgates (analog transfer switches) have been added in this version. The new passgate types include those having multiple MOSFETs and clamping, as shown. These passgate types are often used in test structures.

To support these passgates, the file config/SubgraphTemplates.config has been expanded. If you edit or add to these definitions, the existing index numbering for MOSFET instances must be maintained. Clamp MOSFETs must be numbered above 4, as can be seen in the updated template file.



New passgate types

State Commands

This version adds a new API command for state-based applications. For complete information on the ‘states’ command, please see the documentation under Help > Programming Reference.

cdb states logicLowTogether _ _
cdb states logicHighTogether _ _
Determine whether the given two nets can possibly ever be low (or high) together at the same time.

cdb states passgatesOnTogether _ _
cdb states passgatesOffTogether _ _
Determine whether the given two passgate instances can ever be enabled (or disabled) simultaneously. In particular, for the case of disablement, please note that at present there is support only for a group of two via the Tcl interface.

cdb states pathThroughOffDevice _ _ ..._
Examine all devices in the given path and determine whether the path is guaranteed to be off at all times. This is a complex determination with many input factors (endpoint rails, values, etc). Please see the documentation for details.

cdb states readPathCache
Following an application or scan that invokes 'cdb states pathThroughOffDevice', this command will dump any net pairs that were found to have one-hot behavior (multiplexer decoded controls).

cdb states LHSlowerThanRHS _ _
cdb states LHSoffWhileRHSon _ _
These are re-packaging of existing net based commands, provided here under 'states' for convenience.

cdb states floatPossible _
Determine whether a conditional float is possible on the given net.

cdb states contentionPossible _
Determine whether the given net can be subject of logic contention (DC current such as zener diode bias stack may not trigger a result).

Other New Commands

The following commands are specialized, for particular applications only, but are listed here for completeness:

cdb cell _ testAsIsolationLS
Test the given block cell to determine if it is an isolation level shifter.

cdb cell _ testAsPlainLS
Test the given block cell to determine if it is a plain level shifter.

cdb inst _ testLevelShifter
Characterize and test a subgraph recognized level shifter for various big picture issues.

cdb inst _ printCCG
Print members of a Channel Connected Region.

cdb net _ isFeedbackPart
The given net is part of a recognized feedback path such as latch.

cdb net _ getClockInputs
Return the gating term(s) that form clock input(s) for the given net, such as latch.

cdb net _ printCCRExp
Print the gating term(s) of a Channel Connected Region.

Migration

Generally, this version of Insight Analyzer provides an easy migration path from the former version, with only minor impact.

Run Scripts

Plugin Configure Options

The Floating Nodes application uses some new flags, and drops others. Compatibility with legacy scripts is maintained. Please see Floating Nodes for more details.

The Domain Leakage application adds one new flag, which is not required to be explicitly set. Compatibility with legacy scripts is maintained.

The Fanout and Contention applications have changed -fastScanning flag (legacy) to -scanReplicationFactor. The legacy flag is read-only and will have the same effect as previously.

For all other applications, the plugin configure flags are compatible with run scripts of the prior version.