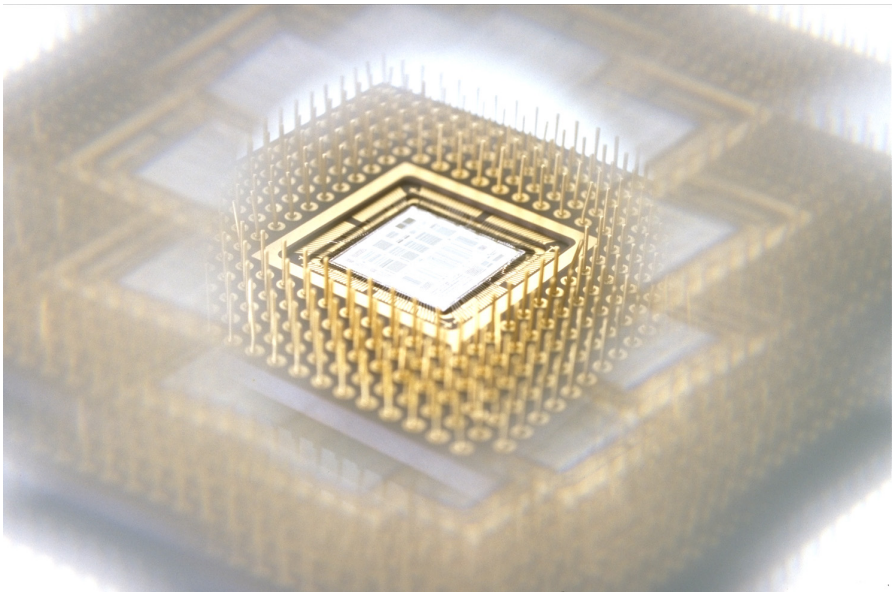
Avertec Tools

HITAS
Reference Guide



Software Release 3.4p5

June 7th, 2010

About this Document

This document explains:

- The input formats supported
- How to perform the timing analysis
- How to browse the timing database
- The timing database formats
- User constraints

Documentation issued and compliant with Avertec Tools Release 3.4p5.

Please contact support@avertec.com for comments relating to this manual.

Table of Contents

Chapter 1. Input Files

1.1. Technology File and Netlist

1.1.1. SPICE

The syntax of the SPICE subset supported by HITAS is given here in Backus-Naur Form. The meta-symbols of BNF are:

`::=`	meaning "is defined as"
`|`	meaning "or"
`<>`	angle brackets used to surround category names. The angle brackets distinguish syntax rules names (also called non-terminal symbols) from terminal symbols which are written exactly as they are to be represented.
`[]`	Enclose optional items
`{}`	Enclose repetitive items (zero or more times)

Expressions and Values

A value is refered to as `<val>`. A value can be associated with the following units, and is scaled accordingly:

`ff`	1e-15
`pf`	1e-12
`f`	1e-15
`p`	1e-12
`n`	1e-9
`u`	1e-6
`m`	1e-3
`k`	1e+3
`meg`	1e+6
`mi`	25.4e+6
`g`	1e+9

v	1
ns	1e-9
ps	1e-12
s	1

An expression is refered to as <expr>, and should appear enclosed in '', () or {}. Carriage returns are ignored within expressions and treated as white spaces, which means that an expression can be continued on subsequent lines without using the + sign.

The following mathematical functions supported within <expr> are valif, max, dmax, min, dmin, trunc, int, sqrt, exp, log, sin, cos, tan, asin, acos, atan, atan2, sinh, cosh, tanh, log10, ceil, floor, fabs, abs, pow and pwr

User-defined Functions

HITAS supports user-defined functions when specified through the .FUNC card. The example below illustrates the syntax supported:

```
.FUNC my_func(a,b) a*b+pow(2,a)
```

MOSFET

```
Mxx <ND> <NG> <NS> <NB> <MNAME> [L=<val>] [W=<val>]
+ [AD=<val>] [AS=<val>] [PD=<val>] [PS=<val>]
+ {[<param>=<val>|<expr>]} [$X=<val>] [$Y=<val>]
```

Parameters:

xx	MOS transistor name
<ND>	Drain node
<NG>	Gate node
<NS>	Source node
<NB>	Bulk node
<MNAME>	Model name, described in a .MODEL card

Optional parameters:

L=<val>	Channel length in meters (unless specified unit)
W=<val>	Channel width in meters (unless specified unit)
AD=<val>	Drain area in sq. meters (unless specified unit)
AS=<val>	Source area in sq. meters (unless specified unit)

PD=<val>	Drain Perimeter in meters (unless specified unit)
PS=<val>	Source Perimeter in meters (unless specified unit)
$X=<val>	X coordinate
$Y=<val>	Y coordinate
<param>=<val>|<expr>	Instantiation specific parameters, for example nrs, nrd, mulu0, delvt0, sa, sb, sd, nf, nfing, m

If AD, AS, PD or PS are not specified, they are calculated with the GEOMOD parameter (only BSIM4, otherwise value is 0).

MOSFET Models

```
.MODEL <MNAME> <nmostype>|<pmostype>
+ [(] [<param>=<val>|<expr>] [)]

<nmostype> ::= NMOS|NMOSBSIM3|NMOSBS32|NMOSBS4|
+ NMOSBS41|NMOSBS42|NMOSBS43|NMOSBS44|NMOSBS45|NMOSBS46

<pmostype> ::= PMOS|PMOSBSIM3|PMOSBS32|
+ PMOSBS4|PMOSBS41|PMOSBS42|PMOSBS43|PMOSBS44|PMOSBS45|PMOSBS46
```

MOS types

NMOS,PMOS	N-Channel, P-Channel MOSFET model
NMOSBSIM3,PMOSBSIM3	N-Channel, P-Channel BSIM3v3.0 Berkeley MOSFET model
NMOSBS32,PMOSBS32	N-Channel, P-Channel BSIM3v3.2.4 Berkeley MOSFET model
NMOSBS4,PMOSBS4	N-Channel, P-Channel BSIM4.0 Berkeley MOSFET model
NMOSBS41,PMOSBS41	N-Channel, P-Channel BSIM4.1 Berkeley MOSFET model
NMOSBS42,PMOSBS42	N-Channel, P-Channel BSIM4.2 Berkeley MOSFET model
NMOSBS43,PMOSBS43	N-Channel, P-Channel BSIM4.3 Berkeley MOSFET model
NMOSBS44,PMOSBS44	N-Channel, P-Channel BSIM4.4 Berkeley MOSFET model
NMOSBS45,PMOSBS45	N-Channel, P-Channel BSIM4.5 Berkeley MOSFET model
NMOSBS46,PMOSBS46	N-Channel, P-Channel BSIM4.6 Berkeley MOSFET model

Supported BSIM and PSP levels:

`LEVEL=8`	NGSPICE Berkeley BSIM3v3 model, up to BSIM3v3.2.4 (`VERSION=32`)
`LEVEL=49`	HSPICE BSIM3v3 model, up to BSIM3v3.2.4 (`VERSION=32`)
`LEVEL=53`	ELDO BSIM3v3 model, up to BSIM3v3.2.4 (`VERSION=32`)
`LEVEL=14`	NGSPICE Berkeley BSIM4 model, up to BSIM4.3 (`VERSION=43`)
`LEVEL=54`	HSPICE BSIM4 model, up to BSIM4.3 (`VERSION=43`)
`LEVEL=60`	ELDO BSIM4 model, up to BSIM4.3 (`VERSION=43`)

```
TOOL hspice
BSIM3V3  param level   49
BSIM3V3  param level   53
BSIM4    param level   54
PSP      param level 1020
PSPB     param level 1021

TOOL eldo
BSIM3V3  param level   49
BSIM3V3  param level   53
BSIM4    param level   60
PSP      param level 1020
PSPB     param level 1021

TOOL ngspice
BSIM3V3  param level    8
BSIM4    param level   14


TOOL titan
BSIM3V3  model BSM3 setdefault version 3.0
BSIM3V3  model BS32 setdefault version 3.24
BSIM4    model BS4  setdefault version 4.2
BSIM4    model BS41 setdefault version 4.1
BSIM4    model BS42 setdefault version 4.21
```

Different industry-standard electrical simulators have different interpretations of the parameters of .MODEL statement, which also deviate from the Berkeley model (see Berkeley's BSIM3v3.2.4 or BSIM4.3.0 MOSFET Model User's Manual). This can lead to significant differences in the results given by different simulators.

With no `simToolModel` variable set, HITAS uses the HSPICE model. Otherwise, HITAS interprets the parameters in the .MODEL statement with regard to the value of the `simToolModel` variable, and uses the model of the corresponding simulator.

The HSPICE BSIM3v3 model (`LEVEL=49`, `simToolModel = HSPICE`) used by HITAS deviates from the Berkeley BSIM3v3 model with regard to the following parameters (only if parameter `ACM=0-3`):

`CJSWG`	ignored, `CJGATE used instead`

MJSWG	ignored, MJSW used instead
PBSW	ignored, PHP used instead
PBSWG	ignored, PHP used instead
NF	the W of the is divided by NF to choose the appropriate model in the techno file

The ELDO BSIM4 (LEVEL=60, simToolModel = ELDO) model used by HITAS deviates from the Berkeley BSIM4 model with regard to the initialization of the binning parameters of LPEB (lateral non uniform doping on K1):

LLEPB=0	instead of LLEPB=LLPE0
WLEPB=0	instead of WLEPB=WLPE0
PLEPB=0	instead of PLEPB=PLPE0
NF	the W of the is not divided by NF to choose the appropriate model in the techno file

The TITAN BSIM models used by HITAS are fully compliant with Berkeley BSIM models. The only special behavior relates to NF

NF	the W of the is not divided by NF to choose the appropriate model in the techno file

JFET

```
Jxx <ND> <NG> <NS> <MNAME> {[<param>=<val>|<expr>]}
+ [$X=<val>] [$Y=<val>]
```

Parameters:

xx	JFET transistor name
<ND>	Drain node
<NG>	Gate node
<NS>	Source node
<MNAME>	Model name, described in a .MODEL card

Optional parameters:

$X=<val>	X coordinate
$Y=<val>	Y coordinate

`<param>=<val>|<expr>`	Instantiation specific parameters

Warning: JFETs are parsed but are not suuported as transistors. They can only be interpreted as resistances. See `avtSpiJFETisResistance`.

Junction Diode

```
Dxx NP NN MNAME [AREA=<val>] [PJ|PERI=<val>]
+ {[<param>=<val>|<expr>]} [$X=<val>] [$Y=<val>]
```

Parameters:

`xx`	Diode name
`<NP>`	Positive node
`<NN>`	Negative node
`<MNAME>`	Model name, described in a .MODEL card

Optional parameters:

`$X=<val>`	X coordinate
`$Y=<val>`	Y coordinate
`<param>=<val>|<expr>`	Instantiation specific parameters

Resistance

```
Rxx N1 N2 [R=]<val>|<expr> [TC1=<val>|<expr>]
+ [TC2=<val>|<expr>] {[<param>=<val>|<expr>]}
```

Parameters:

`xx`	Resistance name
`<N1>, <N2>`	Resistance nodes
`[R=]<val>|<expr>`	Value of resistance in Ohm

Optional parameters:

`TC1=<val>|<expr>`	Parsed but not supported
`TC2=<val>|<expr>`	Parsed but not supported
`<param>=<val>|<expr>`	Parsed but not supported

Capacitance

```
Cxx <N1> <N2> [C|VALUE=]<val>|<expr> [POLY=<val>|<expr>]
+ {[<param>=<val>|<expr>]}
```

Parameters:

xx	Capacitance name
<N1>, <N2>	Capacitance nodes
[C|VALUE=]<val>|<expr>	Value of capacitance in Farads

Optional parameters:

POLY=<val>|<expr>	Parsed but not supported
<param>=<val>|<expr>	Parsed but not supported

Subcircuit Instance

```
Xxx {<NN>} <MNAME> {[<param>=<val>|<expr>]}
+ [$X=<val>] [$Y=<val>] [$T=<Tx> <Ty> <R> <A>]
```

Parameters:

xx	Instance name
{<NN>}	list of nodes. Number must be the same as the subcircuit being instantiated
<MNAME>	Subcircuit being instantiated

Optional parameters:

$X=<val>	X coordinate
$Y=<val>	Y coordinate
$T=<Tx> <Ty> <R> <A>	Transform of the placement (X translation, Y translation, reflexion and rotation). Parsed but not supported
<param>=<val>|<expr>	Instantiation specific parameters, updating subcircuit parameters

Independant Voltage Source

```
Vxx <NP> <NN> DC [=]
+ <expr>|<pwl_function>|<pulse_function>

<pwl_function> ::= PWL (<TN> <VN> {<TN> <VN>} [TD] [SHIFT=<val>])

<pulse_function> ::= PULSE (<V0> <V1> <TD> <TR> <TF> <PW> <PER>)
```

Parameters:

xx	Voltage source name
<NP>	Positive node. The node may be hierarchical, up to one level of hierarchy
<NN>	Negative node. The node may be hierarchical, up to one level of hierarchy

Piece Wise Linear function parameters:

<TN>	Time Ti in seconds (unless specified unit)
<VN>	Value Vi of the source in volts at time Ti
<TD>	Negative node
SHIFT=<val>	Value added to all time values specified in the PWL card

Pulse function parameters:

<VO>	Initial value in volts of DC voltage
<V1>	Pulse magnitude in volts
<TD>	Delay time in seconds (unless specified unit)
<TR>	Rise time in seconds (unless specified unit)
<PW>	Pulse width in seconds (unless specified unit)
<PER>	Pulse period in seconds (unless specified unit)

The PWL and PULSE functions can be used to define clocks as an alternative to the INF or SDC constraint files. However, care should be taken to ensure that enough of the waveform is specified for the parser to be able to deduce the rise/fall clock instants and the period.

The DC function can be used to specify power supply values. If the specified negative node is the node 0, or a node for which a supply value has been associated, then the supply value given by the sum of the DC value and the negative node supply value is associated to the positive node. Fairly complex multi-voltage configurations are possible, since multiple Vcards are possible and they can be resolved in any order.

The DC function, especially in combination with the .GLOBAL directive is a powerful mechanism for specifying which nodes are power supplies. Supplies can be completely determined using these cards without using any configuration variables. A node for which the supply value is superior to avtVddVssThreshold is considered to be a VDD node, else the node is considered to be a VSS node.

Supported Voltage Sources: Scenario 1

```
.GLOBAL inh_VDD inh_GND
```

```
Vsup inh_VDD inh_GND 1.2V
Vgnd inh_GND 0 0V

.SUBCKT inv A B inh_VDD inh_GND
MP0 B A inh_VDD inh_VDD PCH
MN0 B A inh_GND inh_GND NCH
.ENDS
```

Supported Voltage Sources: Scenario 2

```
Vgnd GND 0 0V
Vsup VDD GND 1.2V

.SUBCKT INV A B inh_VDD inh_GND
MP0 B A inh_VDD inh_VDD PCH
MN0 B A inh_GND inh_GND NCH
.ENDS

Xinv0 A B VDD GND INV
```

Supported Voltage Sources: Scenario 3

```
Vgnd GND 0 0V

.SUBCKT INV A B inh_VDD inh_GND
Vsup inh_VDD inh_GND 1.2V
MP0 B A inh_VDD inh_VDD PCH
MN0 B A inh_GND inh_GND NCH
.ENDS

Xinv0 A B VDD GND INV
```

Supported Voltage Sources: Scenario 4

```
Vgnd GND 0 0V
Vsup12 VDD GND 1.2V

.SUBCKT INV A B inh_VDD inh_GND
Vsup14 inh_VDD inh_GND 1.4V
MP0 B A inh_VDD inh_VDD PCH
MN0 B A inh_GND inh_GND NCH
.ENDS

Xinv0 A B VDD GND INV
```

`Vsup14` is ignored

Supported Voltage Sources: Scenario 5

```
Vgnd GND 0 0V
Vsup12 VDD GND 1.2V
Vsup14 VDD GND 1.4V

.SUBCKT INV A B inh_VDD inh_GND
MP0 B A inh_VDD inh_VDD PCH
MN0 B A inh_GND inh_GND NCH
.ENDS

Xinv0 A B VDD GND INV
```

`Vsup12` is ignored

Supported Voltage Sources: Scenario 6

```
Vgnd Xinv0.inh_GND 0 0V
Vsup12 Xinv0.inh_VDD 0 1.2V

.SUBCKT INV A B
MP0 B A inh_VDD inh_VDD PCH
MN0 B A inh_GND inh_GND NCH
.ENDS

Xinv0 A B INV
```

The following syntax is not supported:

Vsup12 Xinv0.inh_VDD Xinv1.inh_GND 1.2V

File Inclusion

```
.LIB|.LIBRARY LNAME [<LIBTYPE>]
.INCLUDE <FILENAME>
```

HITAS has a limited support of relative paths: when the path is not absolute, the path is assumed to be relative to the working directory of HITAS (the directory where it has been invoked from). Contrary to other simulators, it is not assumed to be relative to the directory of the file which makes the inclusion. This limitation can be overwhelmed by the variable `avtLibraryDirs`

Subcircuit

```
.SUBCKT <NAME> <NN> {<NN>} [PARAM:]{[<param>=<val>|<expr>]}

{<component>}

.ENDS [<NAME>]

<component> ::= M|J|D|R|C|X|V|.SUBCKT|.LIB|.INCLUDE|.MODEL|.PARAM
```

Parameters:

`<NAME>`	Name of the subcircuit
`<NN>`	Node name. Nodes with the same name followed by period and number are considered to be on the same net, even if they are not connected in the subcircuit. For example, in `.SUBCKT nand2 in out out.1 out.2 vss vdd out.3` the nodes `out`, `out.1`, `out.2` and `out.3` are considered to be the same signal. See also `avtSpiMergeConnector` and `avtSpiConnectorSeparator`.
`<param>=<val>|<expr>`	Default parameters

HITAS supports the declaration of subcircuits within subcircuits. However, if subcircuit A is defined within a subcircuit, instantiations of subcircuit A must not occur before in the file. This is not true if subcircuit A is defined at top-level.

Parameters

```
.PARAM {<param>=<val>|<expr>}
```

Temperature

```
.TEMP <val>|<expr>
.OPTION TEMP <val>|<expr>
```

Scale Factor

```
.SCALE <val>|<expr>
```

Scales MOSFET parameters.

```
L=L*<val>

W=W*<val>

PD=PD*<val>

PS=PS*<val>

SA=SA*<val>

SB=SB*<val>

SD=SD*<val>

AD=AD*<val>*<val>

AS=AS*<val>*<val>
```

Global Nodes

```
.GLOBAL {node}
```

When using the TCL interface, one should take care that the validity of `.GLOBAL` statement is limited to the context of the `avt_LoadFile` function call. For example, let's suppose a `.GLOBAL` statement defined in `globals.spi`, and a netlist defined in `netlist.spi`:

```
avt_LoadFile globals.spi spice
avt_LoadFile netlist.spi spice
```

With such a script, the `.GLOBAL` statement will not be available in `netlist.spi`. If it is not the intended behavior, prefer `.INCLUDE globals.spi` in `netlist.spi`.

1.1.2. VERILOG

HITAS supports the structural subset of the Verilog Hardware Description Language. For more information see the IEEE P1364 standard

1.1.3. VHDL

HITAS supports the structural subset of the VHDL Hardware Description Language. For more information see the IEEE P1076 standard.

1.2. Parasitics

1.2.1. DSPF Used for Connectivity

If the DSPF is used for connectivity purpose (the SPICE netlist is not connected without the DSPF, connectivity is ensured by the R elements), use .INCLUDE parasitics.dspf inside .SUBCKT

1.2.2. DSPF Used for Back-Annotation

If there is no BUSBIT construct, only identifiers containing [] or < > are considered as vectors.

1.2.3. SPEF

HITAS supports the Standard Prasitic Exchange Format Language, used for parasitic back-annotation putpose. For more information see the IEEE 1481-1999 standard.

1.3. Timing Characterizations

1.3.1. Liberty

HITAS supports the Liberty version 2000.11 (.lib) Open Source standard for integrating external timing characterization.

Supported statements are:

- `bus`
- `cell`
- `cell_fall`
- `cell_rise`
- `fall_constraint`
- `fall_propagation`
- `fall_transition`
- `ff`
- `ff_bank`

- `latch`
- `latch_bank`
- `library`
- `lu_table_template`
- `pin`
- `rise_constraint`
- `rise_transition`
- `rise_propagation`
- `statetable`
- `timing`
- `type`
- `wire_load`
- `wire_load_table`

For more information see the Liberty User Guide.

1.3.2. TLF

HITAS supports the Cadence Timing Library Format versions 3 and 4, for integrating external timing characterization.

Supported TLF statements are:

- `cell` `celltype`
- `seq` `comb`
- `ignore`
- `incell` `outcell`
- `model`
- `spline` `const` `data`
- `input` `output` `bidir`
- `clock`
- `clock_slew_axis` `input_slew_axis`
- `slew_axis` `load_axis` `axis`
- `path` `path_extension`
- `fast` `slow`
- `delay` `slew`
- `t01` `t10` `t0z` `t1z` `tz0` `tz1` `negedge` `posedge`
- `setup`
- `hold`
- `pin` `pintype` `ground` `pindir` `pin_cap`

- timing_props
- for_bits
- vendor
- environment
- version tlf_version
- header date library technology
- control
- tristate
- asynch
- low high
- linear
- value
- table_input_threshold table_output_threshold
- table_transition_start table_transition_end
- net_cap net_res
- temperature
- voltage
- function
- enable
- register latch
- clear set
- clock slave_clock
- inverted_output

1.4. INF Design Specific Configuration

1.4.1. Description

The `.inf` file is an ASCII file with different sections. Blank lines, lines starting with '#' or any character between '/*' and '*/' are considered as comments. Most of the sections, except the header, are defined with a section name and the information related to this section are enclosed with `Begin` and `End;`. The syntax for each section depends on the section itself. The sections are not depending on each other so they can be declared out of order and several times.

The syntax is case insensitive for the keywords. The signal names can be given without quotes but to enable the use of signal names with special characters or unfortunately matching a keyword, they are required.

It is also possible to use units. For timing values with no specified unit, the Pico-second will be used. Valid units are: ps, ns, ms. For the capacitance values with no unit is specified, the Femto-farad will be used. Valid units are: pf, ff.

The information file gives information for the abstraction, database construction and static timing analysis tools. The information file name does not need to be related to a subcircuit name. In fact, the first token that should be set in the information file is "name" followed by the subcircuit name the information are given for. If this information is not given, the information file parser will guess the name of the subcircuit from the file name.

For a subcircuit, several information files can be given. They will be loaded in a particular order and the information will be merged. Actually, there are 2 different information files that can be generated by our tools and loaded, by default, with the following precedence:

SDC input file

The SDC command will be translated in the appropriate information file sections.

User-defined INF file

The order of loading and the files to load can be set by modifying the variable `avtReadInformationFile`. The files are separated by commas and the special character '$' will be replaced by the subcircuit name. Regular expressions can be used there to handle more complicated naming. Its default value is `$.spice.inf,$.sdc.inf,$.inf`. The '\' before the '.' simply indicates to the regular expression matching algorithm not to interpret '.' as any characters but as the character '.'. The names are from the less priority to the more priority. If the value is set to `no`, no information file will be loaded.

1.4.2. General

It is necessary to specify the design name. This is done by using the section "name". Optionally, a version for the information file can be given.

```
NAME mysubckt;
VERSION 1.2;
```

1.4.3. Disassembly Directives

IGNORE

If some components must be removed from the original netlist for any reason, it is possible to specify the component type and names in the IGNORE section. Th ere are 4 component types: Instances, Transistors, Resistances, Capacitances. Th e component names can be given using regular expressions.

```
IGNORE
BEGIN
  Instances: *fake, top.instancetoremove;
  Capacitances: toolowcapa*;
  ...
END;
```

At the moment, it is possible to remove resistances and capacitances only in the top level figure. To overcome this limitation, an information file can be wr itten for the sub-circuits.

CONSTRAINT

To perform analysis in a specific case, the user can apply static logic levels on the input connectors with the CONSTRAINT section. It contains internal or external signals constrained by "one" or "zero". The static logic levels are propagated through the netlist before databse construction.

```
CONSTRAINT
BEGIN
    sig1: 0;
    sig2: 1;
    sig3: 0;
    ...
END;
```

MUTEX

If some of the input connectors are mutually exclusive, this should be indicated in the MUTEX section using:

`muxUP{term1, term2, ...}`	to express that one port at most in the list is "one".
`muxDN{term1, term2, ...}`	to express that one port at most in the list is "zero".
`cmpUP{term1, term2, ...}`	to express that one and only one port in the list is "one".
`cmpDN{term1, term2, ...}`	to express that one and only one port in the list is "zero".

```
MUTEX
BEGIN
    muxUP{a,...,d};
    muxDN{m,...,p};
    cmpUP{i,...,l};
    cmpDN{x,...,z};
    ...
END;
```

INPUTS

User can specify connectors which should be considered as inputs. The disassembling process uses this information to inhibit the construction of cone branches from these connectors. This directive is essentially useful when dealing with RAMs, when the user wants to inhibit the construction of the reading bus.

```
INPUTS
BEGIN
    connector0;
    connector1;
    ...
END;
```

STOP

User can specify a list of signals which should be considered as stop points for the functional analysis phase of the disassembly. This means that any logic preceding the stop points will not be used in order to analyze the behavior of any gate following the stop point.

```
STOP
BEGIN
    sig1;
    sig2;
    sigs*;
    ...
END;
```

It is possible to use the wildcard '*' as for signal renaming.

DIROUT

In order to orient transistors, user can specify a list of signals, each one identifying the source or drain of a transistor. Transistors are then oriented towards these signals. Orientation of successive transistors is done by associating a level with each signal identifying a source or drain. Transistors are then oriented from higher level signals to lower level signals.

Transistor orientation is useful to avoid false branches construction, especially when dealing with pass-transistors.

```
DIROUT
BEGIN
    sig2: "1";
    sig3: "2";
    ...
END;
```

If no DIROUT level is specified, default level is `-1`.

It is possible to use the wildcard '*' as for signal renaming.

The DIROUT directive is equivalent to the NETOUTPUT directive in FCL. If a signal name is preceded by the '~' character then this signal will not be treated as an output, this is to deal with the case of signals whose names end in "_s" and must not be considered as output terminals.

DLATCH

Some internal tri-state nodes have to be considered to be dynamic latches for functional modeling and timing analysis purposes.

If only a few number of that tri-state nodes have to be taken into account, specify the list of corresponding signals into a DLATCH section. On the contrary, if the variable `yagleMarkTristateMemory` is set to `yes` in the configuration file and a few of tri-state nodes must not be taken into account, specify the list of corresponding signals into a DLATCH section, preceding each signal by a '~' character.

```
DLATCH
BEGIN
    sig1;
    ~ sig2;
```

```
    ...
END;
```

It is possible to use the wildcard '*' as for signal renaming.

CKLATCH

By default the latch recognition phase performed during the circuit disassembly does not require that external clocks be specified. Latches are identified either by structure (yagleStandardLatchDetection) or by Boolean analysis of combinatorial loop stability (yagleAutomaticLatchDetection).

However, it is sometimes necessary to constrain the latch recognition to identify only those latches for which the local clocks lie on a genuine clock path. To do this, specify the list of external connectors (or internal signals) in a CKLATCH section.

It is also possible to specify that an external connector (or internal signal) is definitely not a clock by preceding the name by a '~' character. In this case, any latch input at the end of a timing path originating from this connector (or signal) is considered to be a data input.

```
CKLATCH
BEGIN
    connector1;
    sig1;
    ~ sig2;
    ...
END;
```

It is possible to use the wildcard '*' as for signal renaming.

PRECHARGE

Signals whose names end in "_p" are considered to be precharged and therefore dealt with differently by the tool. If any other signals should be considered precharged these can be specified in the PRECHARGE section.

```
PRECHARGE
BEGIN
    sig1;
    ~ sig2;
    ...
END;
```

If a signal name is preceded by the '~' character then this signal will not be treated as a precharge, this is to deal with the case of non-precharge signals whose names end in "_p".

NOTLATCH

Disables the detection of latch on given signal.

```
NOTLATCH
BEGIN
    sig1;
    ...
END;
```

If a signal name is preceded by the '~' character then this signal will not be treated as a precharge, this is to deal with the case of non-precharge signals whose names end in "_p".

MARKSIG

Allows to set special markings on signals (nets). Especially useful for custom latch recognition. Available markings are:

LATCH	Signal corresponds to a latch memory-point.
FLIPFLOP	Signal corresponds to a flip-flop memory-point.
MASTER	Signal corresponds to the master memory-point of a flip-flop.
SLAVE	Signal corresponds to the slave memory-point of a flip-flop.
MEMSYM	Signal corresponds to one side of a symmetric memory.
RS	Signal corresponds to one side of an RS bistable.
VDD	Signal corresponds to an alimentation.
VSS	Signal corresponds to ground.
BLOCKER	No branch of a cone can go through the signal.
STOP	Cannot exploit logic beyond this point for functional analysis in the disassembler.
SENSITIVE	Marks the signal as a particularly sensitive signal. If a timed behavioral model of this signal is produced then the most precise (but cumbersome) model will be generated.

Example:

```
MARKSIG
BEGIN
    sig1: LATCH+MASTER
    sig2: STOP
    ...
END;
```

MARKTRANS

Allows to set special markings on transistors. Especially useful for custom latch recognition. Available markings are:

BLEEDER	Transistor corresponds to a bleeder.
FEEDBACK	Transistor corresponds to a feedback transistor of a memory-point.

COMMAND	Transistor corresponds to a command transistor of a memory-point, i.e driven by command signal.
NOT_FUNCTIONAL	Transistor should be ignored when calculating gate functionality.
BLOCKER	No branch of a cone can contain this transistor unless it is the first transistor of the branch.
UNUSED	No branch of a cone can contain this transistor.
SHORT	The transistor is considered short-circuited, the gate signal no longer contributes to the list of inputs.

Example:

```
MARKTARNS
BEGIN
    trans1: FEEDBACK+NOT_FUNCTIONAL
    trans2: COMMAND
    ...
END;
```

1.4.4. Timing Directives

BREAK

For the static timing analysis, verification of setup and hold constraints is only performed at reference points, i.e. external connectors, latch and precharge data input, latch and precharge command. However, it may be necessary to perform verifications on other points. These so-called break points are specified in the BREAK section.

```
BREAK
BEGIN
    sig1;
    sig2;
    ...
END;
```

The break point mechanism can also be used to handle internally generated clocks, since a signal can be declared a break point and then declared a clock in the static timing analysis file.

BYPASS

To eliminate some critical paths, a BYPASS section should be used.

```
BYPASS
BEGIN
    sig1;
    sig2 <;
    sig3 >;
    ...
END;
```

If a signal appears with no argument, the tool eliminates all paths which cross it. With the '>' argument, all paths which end on the signal will be suppressed and with the '<' argument, all paths beginning on the signal are suppressed. With the '!' argument, all paths passing through an input/output tri-state connector are suppressed.

FALSEPATH

During the static timing analysis, it is possible to specify that certain timing paths are false paths using the FALSEPATH section. Each line in this section describes a path to be ignored by the static timing analysis. A line is made up of the list of signals through which the path passes. The first signal in the list must be the start of a timing path and the last one must be the end of a timing path. It is possible to add the following tokens to the list in order to further constrain the path matching:

`<UP>`	must match rise transition on net given by preceding name.
`<DOWN>`	must match fall transition on net given by preceding name.
`<VOID>`	means signal given by preceding name must de directly followed by signal given by succeeding name.
`: <HZ>`	at the end of the list means must match a high impedance transition at the end of the path.
`: <NOTHZ>`	at the end of the list means must NOT match a high impedance transition at the end of the path.

```
FALSEPATH
BEGIN
    sig1 sig2 sig3;
    sig1 sig4 <UP> sig5 <DOWN>;
    sig1 sig2 sig6 <VOID> sig7;
    sig1 sig2 sig6 sig7 : <HZ>;
    ...
END;
```

INTER

In order to reduce the number of critical paths identified by the tool, it can sometimes be useful to specify that certain internal signals be considered as though they were reference points. The list of these intermediary points is specified in the INTER section. Correct choice of these points can lead to factorization of the critical paths, and hence a significant reduction in the size of the output file generated.

```
INTER
BEGIN
    sig1;
    sig2;
    ...
END;
```

PINSLOPE

The PINSLOPE section describes the slopes to be applied on input connectors. Rise is the rising slope and fall is the falling slope. A single value is the slope applied for rising and falling. Slopes are given in Pico-seconds.

```
PINSLOPE
BEGIN
    sig1: rise = 100 fall = 200;
    sig2: rise = 150 fall = 180;
    sig3: 200;
    default: rise = 120 fall = 140;
    ...
END;
```

PATH DELAY MARGIN

It is possible to add an uncertainty margin to the paths, using the PATH DELAY MARGIN section. The modification is done by applying a `factor` and a `delta` to the path delay. So the new delay value is old_delay*`factor`+`delta`. The path delay won't be affected directly but the path margin will affect the stability analysis results.

The selection of the paths is done with the name of the signal at the end of the path and the type of the path: "clockpath", "datapath", "min", "max", "rise" and "fall". Mixing those values is possible. If the path type is not specified, all the paths arriving on the signal will be affected. The path ending signal can also be defined by a type: "latch", "prech" (precharge), "break" (breakpoint), "con" (connector) or "any".

```
PATH DELAY MARGIN
BEGIN
    prech data_P: factor=1   delta=10ps;
    latch *: factor=1.1 delta=0    datapath max;
    any   *sig*: factor=1   delta=-7   clockpath;
    ...
END;
```

The factor and the delta are needed and must appear in this order.

NOFALLING - NORISING

In certain cases, it is necessary to prevent the propagation of timing events, for example from glitch generators in which we are only interested in either the up or down event. The list of signals for which to suppress the down event is given in the NOFALLING section. The list of signals for which to suppress the up event is given in the NORISING section.

```
NOFALLING
BEGIN
    sig1;
    sig2;
    ...
END

NORISING
BEGIN
    sig1;
    sig2;
```

```
    ...
END;
```

CONNECTOR DIRECTIONS

If it is necessary to modify connectors direction in the timing database (DTX and TTX), the CONNECTOR DIRECTIONS section can be used. Connector names can be regular expressions. Supported connector directions are: `Input`, `Output`, `InOut`, `Tristate`, `HZ`, `Unknown`.

```
CONNECTOR DIRECTIONS
BEGIN
input: sig1, sig2;
hz: sig3;
...
END;
```

RC - NORC

To enable or disable RC delay calculations for individual signals, use the RC and NORC sections.

```
RC
BEGIN
    sig1;
    sig2;
    ...
END

NORC
BEGIN
    sig1;
    sig2;
    ...
END;
```

NOCHECK

```
NOCHECK
BEGIN
    sig;
    sig ALL;
    sig SETUP;
    sig HOLD;
    ...
END;
```

DONOTCROSS

Disables any path crossing specified latch or command.

```
DONOTCROSS
BEGIN
    sig1;
    sig2;
    ...
END;
```

1.4.5. Timing Abstraction Directives

Parameterization of Lookup Tables

Lookup tables defined for timing paths depend on input slopes and output capacitances. The SLOPEIN and CAPAOUT sections provide various means for specifying the ranges of slopes and capacitances applied on external connectors. Ranges can be set for selected connectors, but a default range can also be given, using the keyword 'default'. Ranges can be set by enumeration, using the syntax:

```
signal: (slope_0, slope_1, ... , slope_n);
```

Ranges can also be set by specifying a lower bound, an upper bound, and a step, using the syntax:

```
signal: (lower_bound: upper_bound: step);
```

SLOPEIN

The SLOPEIN section describes the slopes to be applied on input connectors. Slopes are given in pico-seconds.

```
SLOPEIN
BEGIN
    sig1: (100, 400, 900, 1500, 2200, 3000);
    sig2: (100: 1000: 100);
    default: (100, 400, 900, 1500, 2200, 3000);
    ...
END;
```

CAPAOUT

The CAPAOUT section describes the capacitances to be applied on output connectors. To specify capacitances, a scaling factor is available. The range of capacitance can be given for specific length and width of transistors. The capacitances really applied on the connector are then scaled by the ratio between the given length and width, and the characteristic length and width of the transistors of the cell driving the pin. Capacitances are given in Femto-farads, lengths are given microns.

```
CAPAOUT
BEGIN
    sig1: (0.7, 80.5, 190.0, 300.5);
    sig2: (0: 250: 25);
    default: (0.3, 38.5, 147.0, 311.5) L=0.18 W=0.6;
    ...
END;
```

CLOCK CONNECTORS

Clocks must be declared in the information file in order to compute setup and hold constraints. These clocks are specified in the CLOCK CONNECTORS section.

```
CLOCK CONNECTORS
BEGIN
    sig1;
    sig2;
    ...
END;
```

1.4.6. STA Directives

The eight sections in the information file concerning the stability information are:

- General header
- Clock Specifications
- Clock domains
- Equivalent clocks
- Multiple Clock priorities
- Gated clock states
- Input connector stability specifications
- Output connector constraints to verify

A wildcard can be used for any of the specified node names. The character '*' matches any legal character string.

For example:

```
com* matches com1, com2, command, etc.
```

General Header

The format of the header is as follows:

```
PERIOD  <value>;
SETUP   <value>;
HOLD    <value>;
```

The name is the name of the circuit the information are given for. The period is the default clock period for clocks whose period is not specified. SETUP and HOLD are additional constraints on setup and hold slacks verifications.

It is possible to omit the default period specification so long as each clock has a period associated with its definition or via the domain definition.

CLOCK CONNECTORS

This section is used to define all the external clock signals. The waveforms at each external clock connector must be specified over a single period.

The syntax is as follows:

```
CLOCK CONNECTORS
BEGIN
    {[ideal] [virtual] [!] <ck1>:
        down (<min>:<max>)|<value>;
        up   (<min>:<max>)|<value>;
        [latency down (<min>:<max>)|<value>;]
        [latency up   (<min>:<max>)|<value>;]
        [period <value>;]}
END;
```

For each of the external clock connectors, four parameters must be given: the earliest and latest instant of the falling edge, and the earliest and latest instant of the rising edge. The order is irrelevant. Actual values themselves are used to order the clock phases within the period.

It is not necessary to specify which of the clock edges corresponds to the reference phase. This is automatically detected by HITAS.

The period can optionally be specified here individually for all the clocks. If it is not specified here the clock must have a period via either a domain specification or a default period in the header.

If '!' is added before the clock specification, the values set for the rising and falling slope time of the clock will be swapped.

If 'virtual' is added before the clock specification, that indicates the clock is not associated to a real circuit connector.

If 'ideal' is added before the clock specification, that indicates no propagated delay will be considered for this clock. Nevertheless, the user can specify latency for each clock edge.

A clock can be define from the characteristics of another clock 'refclock', multiplying its period and/or shifting its edges. The syntax is as follow:

```
CLOCK CONNECTORS
BEGIN
    { <ck2>:
        clock <refclock> [* <mult_factor>] [+|- <delta>];}
END;
```

ASYNCHRONOUS CLOCK GROUPS

This section allows the user to assign clock connectors to clock domains. Timing checks are only performed on paths which do not cross domain boundaries. Each domain must be given a name. However the name itself is purely informative.

The syntax is as follows:

```
ASYNCHRONOUS CLOCK GROUPS
BEGIN
    {<domain1>: <clock1> [,<clock2>,...,<clockn>];
         [period <value>;]}
END;
```

Each domain contains the list of clock connectors (defined in the clock specification section) which make up the domain.

The period of the clocks can be specified here, since clocks in the same domain must have identical periods.

EQUIVALENT CLOCK GROUPS

This section allows the user to indicate that separate clock connectors should be treated as having identical phases.

The syntax is as follows:

```
EQUIVALENT CLOCK GROUPS
BEGIN
    {<group1>: <clock1> [,<clock2>,...,<clockn>];}
END;
```

Each group contains the list of clock connectors (defined in the clock specification section) which make up the equivalent group.

MULTIPLE CLOCK PRIORITY

This section allows the user to specify which clock should be considered as having the priority in the case of clocked signals (latches, flip-flops or precharges) which depend on multiple clocks.

Defining priority clocks is useful in the case of multiple clocks due to multiple operating modes (e.g. test mode or functional mode).

The syntax is as follows:

```
MULTIPLE CLOCK PRIORITY
BEGIN
    {<clocked_signal>: <clock_connector>;}
END;
```

Each line associates a clocked signal with its highest priority clock connector.

SPECIFY INPUT CONNECTORS

This section is used to specify the stability intervals at the input terminals. If this section is omitted for any of the inputs, then they are assumed to become unstable at the earliest instant of the last phase and stabilized at the latest possible instant of the last phase.

The syntax is as follows:

```
SPECIFY INPUT CONNECTORS
BEGIN
    {[<input> [rising|falling] [from <phase>]:
        unstable <value> [after|before <phase>];
        stable   <value> [after|before <phase>];]}
    |
    default [rising|falling] [from <phase>]:
        unstable <value> [after|before <phase>];
        stable   <value> [after|before <phase>];
END;
```

For each input connector, the instants it becomes unstable and stable must be given. The phase of origin of the data can optionally be specified after the input connector name. A phase identifier has the format:

```
<ckname> [rising|falling]
```

where ckname refers to a specified external clock. In the case of a clock for which the rising and falling edge refer to different phases, it is necessary to specify the edge, in order to uniquely identify the phase. By default the phase of origin is assumed to be the last phase.

The 'default' stability is used for all inputs for which the stability has not been explicitly specified.

The instants of stability and instability are specified in picoseconds. The values can either be absolute or relative to a specified phase. If the values are absolute then they are considered to be the result of the phase of origin. The values are specified relative to a particular phase using the keywords after and before. In general, you use the keyword 'after' to express that the relative time is as a result of the specified clock phase.

If the keyword 'before' is used, then the relative time is assumed to be for the preceding clock cycle (i.e. 'for' rather than 'as a result of' the specified phase).

VERIFY OUTPUT CONNECTORS

This section is used to specify arrival times to be verified at output connectors. If this section is omitted for any of the outputs, then setup and hold slacks will not be calculated.

The syntax is as follows:

```
VERIFY OUTPUT CONNECTORS
BEGIN
    {[<output> [rising|falling] [for <phase>]:
        unstable <value> [after|before <phase>];
        stable   <value> [after|before <phase>];]}
    |
    default [rising|falling] [for <phase>]:
        unstable <value> [after|before <phase>];
        stable   <value> [after|before <phase>];
END;
```

The syntax is similar to the input connector stability specifications. However the phase specifications identify the phase for which the data is destined.

The times can be absolute or relative, as for the input specification. Here, however, since it is a destination phase which is specified, it can often be convenient to use the keyword 'before' to state that a time is 'relative to' and 'for' the specified phase.

Example of Stability Specification File

Since this file must be provided by the user, an example is shown here for clarification.

```
name mycircuit;
version 1.00;
period 180000;
setuptime 100;
holdtime 200;

CLOCK CONNECTORS
BEGIN
  ck:
    up   (90000:90100);
    down (10000:10100);
END;

CONDITIONED COMMAND STATES
BEGIN
  com1: up;
```

```
END;

SPECIFY INPUT CONNECTORS
BEGIN
  out1:
    unstable 1000 after ck rising;
    stable   3000 after ck rising;
  default:
    unstable 500 after ck rising;
    stable   4000 after ck rising;
END;

VERIFY OUTPUT CONNECTORS
BEGIN
  out1:
    unstable 1000 after ck rising;
    stable   3000 after ck rising;
  default:
    unstable 500 after ck rising;
    stable   4000 after ck rising;
END;
```

1.4.7. Crosstalk Directives

CROSSTALK MUTEX

During the aggression detection stage, it is possible to specify that only one signal can switch at the same time among a set of signals. Theses sets are specified in CROSSTALK MUTEX section :

```
CROSSTALK MUTEX
BEGIN
    MUXUP {sig1, sig2};
    MUXDN {sig3, sig4, sig5};
    MUXUP {sig7, sig8, ..., sig12};
    ...;
END;
```

In each MUXUP (MUXDN) section, only one signal in the list can rise (fall) at the same time. A signal can be listed at most in one MUXUP and in one MUXDN section. If two or more signals of a same mutex appear in the aggressor list of a signal, the one with the bigger coupling capacitance between the victim and this aggressor is considered to be switching at the same time.

1.4.8. Correspondencies INF / Tcl

INF FILE	INF API	SDC
IGNORE	inf_DefineIgnore	
CONSTRAINT		set_case_analysis 0 | 1
MUTEX	inf_DefineMutex	
INPUTS	inf_DefineInputs	
STOP	inf_DefineStop	
DIROUT	inf_DefineDirout	
DIROUT ~	inf_DefineNotDirout	
DLATCH	inf_DefineDLatch	
DLATCH ~	inf_DefineNotDLatch	
CKLATCH	inf_DefineCkLatch	
CKLATCH ~	inf_DefineNotCkLatch	
PRECHARGE	inf_DefinePrecharge	
PRECHARGE ~	inf_DefineNotPrecharge	
NOTLATCH	inf_DefineNotLatch	
BREAK	inf_DefineBreak	
BYPASS		set_disable_timing / set_false_path -from | -through | -to
FALSEPATH		set_false_path -from -through -to
INTER	inf_DefineInter	
PINSLOPE / PINSLEW	inf_DefineSlope	set_input_transition
PINSLOPE / PINSLEW	inf_DefineConnectorSwing	
PATH DELAY MARGIN	inf_DefinePathDelayMargin	
NOFALLING		set_case_analysis fall / set_false_path -fall
NORISING		set_case_analysis rise / set_false_path -rise
CONNECTOR DIRECTIONS	inf_DefineConnectorDirections	
RC	inf_DefineRC	
NORC	inf_DefineNORC	
NOCHECK		set_false_path -setup | -hold
DONOTCROSS	inf_DefineDoNotCross	
SLOPEIN	inf_DefineSlopeRange	
CAPAOUT	inf_DefineCapacitanceRange	
CROSSTALK MUTEX	inf_DefineCrosstalkMutex	
SETUP	inf_SetSetupMargin	
HOLD	inf_SetHoldMargin	
CONDITIONED COMMAND STATES	inf_DefineConditionedCommandStates	
CLOCK CONNECTORS / ASYNCHRONOUS CLOCK GROUP	inf_DefineClock	create_clock / create_generated_clock / set_clock_latency
EQUIVALENT CLOCK GROUP	inf_DefineEquivalentClockGroup	
MULTIPLE CLOCK PRIORITY	inf_DefineClockPriority	
SPECIFY INPUT CONNECTORS		set_input_delay
VERIFY OUTPUT CONNECTORS		set_output_delay
MULTICYCLE PATH		set_multicycle_path

Chapter 2. Output Files

2.1. DTX - The Detailed Perfmodule Format

2.1.1. Description

The DTX format is used to report gate or interconnection delay details. The file has the extension .dtx. This format is also known as the 'detailed perfmodule'. This is a highly compact text format suitable for flat and hierarchical analysis. The content is best visualized with a Tcl interface, or with the XTAS GUI.

2.1.2. Units

Unless otherwise stated, all capacitances are given in Femto-farads, all times are given in picoseconds, and all resistances are given in ohms.

2.1.3. Comments

Any line starting with the '#' character is considered to be a comment.

2.1.4. Articles

The 'detailed perfmodule' contains several articles and is terminated by the 'G' article.

Information Header

General information is given in a single 'H' article at the beginning of the file :

```
H <tool> <vers> <name> <techno> <techno_vers> <inslope> <outcapa> <hierarchy_level>
(<day> <month> <year> <hour> <minute> <second>);
```

Instances

Instances in a hierarchical figure are given in 'X' articles :

```
X <figure_name> <instance_name>;
```

External Connectors

External connectors are given in 'C' articles :

```
C <type> <index> <name> <netname> <capa>;
```

The 'type' defines the connector direction and whether it is the command of a register or a precharge. It can be one of :

I for an input connector

O	for an output connector
B	for a bidirectional connector
T	for a transceiver connector
Z	for a high-impedance connector
X	for an unconnected connector
IQ	input connector used as command for register or precharge
BQ	output connector reused as input to command register or precharge

The electrical parameter given for each connector is :

capa	capacitance attached to connector

Internal Connectors

Internal connectors are given in 'N' articles :

```
N <type> <index> <name> <netname> <capa>;
```

The 'type' defines whether it is the command of a register or a precharge. It can be one of :

I	not used as command for register or precharge
Q	used as command for register or precharge

The electrical parameter is the same as those for external connectors.

Register or Precharge Commands

Commands of registers and precharges are given in 'Q' articles :

```
Q <type> <index> <name> <netname> <capa> (<cmd1> <cmd2> ... );
```

The 'type' can be one of the following :

C	for an external connector command (also an external connector)
N	for an internal connector command (also an internal connector)
E	for an external command (extremity of an external path)
I	for an internal command (not the extremity of an external path)

The list of command names in brackets is the list of commands lower down in the hierarchy which are internal or external commands and which have been replaced by this command at the current hierarchical level. Thus the command attribute of an arc can be defined at each level in the hierarchy.

Registers

Registers are given in 'L' articles :

```
L <type> <index> <name> <netname> <capa>;
```

The 'type'is XY where x can be L, F, R or S and Y can be C, N, E, I :

L	stand for latch
F	stand for flip-flop
R	stand for nand set reset latch
S	stand for nor set reset latch
C	an external connector register (also an external connector)
N	an internal connector register (also an internal connector)
E	an external register (extremity of an external path)
I	an internal register (not the extremity of an external path)

The list in brackets is the list of command events which allow the opening of a level-triggered register or the writing into an edge-triggered register. The 'index' represents the index of the command signal. The type can be one of :

U	for a rising-edge event
D	for a falling-edge event
Y	for 0 to HZ event
Z	for 1 to HZ event

Precharged Signals

Precharged signals are given in 'R' articles :

```
R <type> <index> <name> <netname> <capa>;
```

The 'type' can be one of :

C	for an external connector precharge (also an external connector)
N	for an internal connector precharge (also an internal connector)

E	for an external precharge
I	for an internal precharge

Break

Breaks are given in B articles:

```
B <type> <index> <name> <netname> <capa>;
```

The 'type' can be one of:

C	for an external connector precharge (also an external connector)
N	for an internal connector precharge (also an internal connector)
E	for an external precharge
I	for an internal precharge

Factorization Points

Factorization points are given in 'I' articles :

```
I <type> <index> <name> <netname> <capa>;
```

The 'type' can be one of :

E	for an external factorization signal
I	for an internal factorization signal

Other Signals

Any other signals are given in 'S' articles :

```
S <type> <index> <name> <netname> <capa>;
```

The 'type' can be one of :

E	for an external signal
I	for an internal signal

Elementary Delays

Delays represented by one of three 'delay_type' articles. These are :

F	for an elementary delay which is only part of paths internal to the figure.

E	for an elementary delay which is part of at least one path external to the figure.
D	for an elementary delay which has a connector as an extremity.

The general syntax of these articles is :

```
<delay_type> <starttype> <start> <endtype> <end> (
(<cmd> <cmdtype> (<trs> <type> <delay> <slope> <model(s)>))
( . . (. . . . .))
);
```

where the parameters represent :

starttype	type of the signal at the start of the path (C, N, Q, L, R, I, S).
start	index of the start signal.
endtype	type of the signal at the end of the path (C, N, Q, L, R, I, S).
end	index of the end signal.
cmd	index of command if end is register or precharge
cmdtype	type of event which activates command (U or D).
trs	transition type from event1 to event2, where an event can be one of the following: U for 0-to-1 event, D for a 1-to-0 event, Z for a 0-to-HZ event, Y for a 1-to-HZ event.
type	delay type (SMAX, SMIN, HMAX, HMIN, AMAX, AMIN, IMAX, IMIN, MAX or MIN). With SMAX and SMIN for set-up, HMAX and HMIN for hold, AMAX and AMIN for Access and MAX and MIN for delay, IMAX and IMIN for interconnect delay
delay	propagation delay.
slope	slope at end.
model(s)	one delay model (and one slope model).

2.2. STM - The Models Format

2.2.1. Description

The STM format is used by the timing analyzer HITAS to report delay and slope models. The file has the extension .stm.

2.2.2. Articles

The STM file contains a header and a list of models. These models are delay and slope models. Models refer to timing arcs and timing paths.

A model may be:

- a 1 or 2 dimension lookup table
- a Current Source Model (SCM) containing parameters enabling delay and slope computation

lookup table model:

```
model (
   name (modelname)
   vth (value) #measure threshold
   vdd (value) #power supply
   vt (value)  #vt transistor
   vf (value)  #final voltage
   spline (                          # 2D table
      input_slope_axis (value value value)
      load_axis (value value value)
      data (                      #        load
         (value value value)      #      --------->
         (value value value)      # slope|
         (value value value)      #      |
      )                           #      V
   )
)
```

"SCM" model:

```
model (
   name (modelname)
   vth (value) #measure threshold
   vdd (value) #power supply
   vt (value)  #vt transistor
   vf (value)  #final voltage
   scm_dual (
      dual (
         (<list_of_parameter_names>)
         (<list_of_parameter_values>)
      )
   )
   noise_scr(val) #noise parameter
)
```

pconf0, pconf1: conflict capacitance parameters

capai: intrinsic capacitance

irap: currant ratio

vddin: input voltage

vt: vt transistor

threshold: measure threshold

imax:	max currant
an, bn:	specifical parameters
vddmax:	power supply
rsat:	saturation resistance
rlin:	linear resistance
drc:	intrinsic RC delay

Four kinds of SCM models exist:

- scm_dual
- scm_good
- scm_false
- scm_path

scm_good model example:

```
scm_good (
   link_out (
      (ci cf k3 k4 k5)
      (val val val val val)
   )
   link_dual (
      (ci cf acti bcti)
      (val val val val)
   )
   dual (
      (<list_of_parameter_names>)
      (<list_of_parameter_values>)
   )
)
```

scm_false model example:

```
scm_false (
   link_out (
      (ci cf k3 k4 k5)
      (val val val val val)
   )
   false (
      (pconf0 pconf1 rtot kf vddmax)
      (val val val val val)
   )
)
```

scm_path model example:

```
scm_path (
   link_out (
      (ci cf k3 k4 k5)
      (val val val val val)
   )
```

```
      path (
         (pconf0 vddmax)
         (val val)
      )
)
```

2.2.3. Units

Unless otherwise stated, all capacitances are given in Femto-farads and all times are given in picoseconds.

2.3. STO - STA Output Format

The `.sto` file is an ASCII text file made up of six distinct sections. This file is generated by HITAS and is intended mainly for debugging purposes, since for most purposes, the setup and hold slacks given in the timing report are sufficient. The six sections are:

- General header
- Clock Specifications
- Conditioned clock states
- Input connector switching windows
- Output connector switching windows
- Memory Signal switching windows
- Internal Signal switching windows

Apart from the general header, some of the sections may be omitted if they are of no relevance.

2.3.1. Input Connector Switching Windows

This section gives the switching windows at the input connectors.

The syntax is as follows:

```
input connectors stability
begin
  <input1> [from <phase>]:
    unstable <value>;
    stable   <value>;
        |    |
        |    |
  <inputn> [from <phase>]:
    unstable <value>;
    stable   <value>;
end;
```

2.3.2. Memory Signals Switching Windows

This section specifies the switching windows at all memory signals.

The syntax is as follows:

```
memory nodes stability
begin
  <node1> [from <phase>]:
    unstable <value>;
    stable   <value>;
        |    |
        |    |
  <noden> [from <phase>]:
    unstable <value>;
    stable   <value>;
end;
```

This section gives the switching windows calculated for all memory signals.

2.3.3. Internal Signals Switching Windows

This section specifies the switching windows at all internal signals, except memory signals.

The syntax is as follows:

```
internal nodes stability
begin
  <node1> [from <phase>]:
    unstable <value>;
    stable   <value>;
        |    |
        |    |
  <noden> [from <phase>]:
    unstable <value>;
    stable   <value>;
end;
```

This section gives the switching windows calculated for all signals, except memory signals. If the analysis is performed on the timing path graph (the default), the section is empty. If the analysis is performed on the timing arcs graph, then the set includes all signals.

2.3.4. Output Connector Switching Windows

This section specifies the switching windows at the output connectors.

The syntax is as follows:

```
output connectors stability
begin
  <output1> [from <phase>]:
    unstable <value>;
    stable   <value>;
        |    |
        |    |
  <outputn> [from <phase>]:
    unstable <value>;
    stable   <value>;
end;
```

This section gives the switching windows calculated for outputs. This is the data which is compared with the output connector constraints specified in the '.inf' file or in the Tcl script, to calculate the setup and hold slacks. Every output connector is specified explicitly.

Note that, unlike the output specifications, it is the phase of origin which is given. The destination phase obviously cannot be deduced.

2.4. STR - STA Report Format

For most users, this is the most important file generated by HITAS. It has the suffix `.str` and a base name identical to that of the original subcircuit. It lists the setup and hold slacks calculated for all reference points. This includes:

- Output connectors
- Memory signals
- Conditioned memory commands
- Precharged signals

All memory and conditioned commands are specified with the details of propagated clock, since this is the reference for the setup and hold calculations.

A negative value for a setup or hold slack indicates a violation. In the event of a violation on a particular signal, all data sources resulting in a violation are listed for that signal, together with their individual setup and hold slacks.

2.5. CTK - Crosstalk Report Information

The `.ctk` file is an ASCII text file made up of three distinct sections. This file contains the following information:

- Delay changes due to crosstalk
- Detailed aggression reports for all analyzed nets
- Peak noise voltage for all analyzed nets

2.5.1. Delay Changes due to Crosstalk

This section reports all delay and slopes which have changed due to crosstalk. Thresholds for slope and delay changes can be set to avoid excessive reporting information. These thresholds are set using the `ctkDeltaDelayMin` and `ctkDeltaSlopeMin` configuration variables.

The syntax of this section is as follows:

```
BeginDelay
  UP|DW <start> UP|DW <end>
     delay|slope min|max <nominal> -> <aggressed>
         |      |
         |      |
     delay|slope min|max <nominal> -> <aggressed>
   |
   |
  UP|DW <start> UP|DW <end>
     delay|slope min|max <nominal> -> <aggressed>
         |      |
         |      |
```

```
     delay|slope min|max <nominal> -> <aggressed>
EndDelay
```

Fields <start> and <end> represent timing signal names. If the timing signal name is different from the net name, then the net name is given in brackets. Fields <nominal> and <aggressed> represent delay values in picoseconds without and with crosstalk respectively.

2.5.2. Detailed Aggression Report

This section gives detailed aggressor information for nets whose total coupling capacitance exceeds a certain threshold. The coupling capacitance due to each aggressor is given, as well as information as to whether the aggressor can modify minimum or maximum delays. Only nets for which the relative coupling capacitance is greater than the `ctkrcapamin` configuration variable are reported.

The syntax of this section is as follows:

```
BeginCrosstalk
  Node: DW|UP <signal>
    Ground capacitance: <value>
    Aggressor:
      [*] <signal> [B] [W] [R] [F] cc=<value>
             |         |
             |         |
      [*] <signal> [B] [W] [R] [F] cc=<value>
                                    ---------
                                    <total> (<relative>)
  |
  |
  Node: DW|UP <signal>
    Ground capacitance: <value>
    Aggressor:
      [*] <signal> [B] [W] [R] [F] cc=<value>
             |         |
             |         |
      [*] <signal> [B] [W] [R] [F] cc=<value>
                                    ---------
                                    <total> (<relative>)

EndCrosstalk
```

If a star (*) is present before an aggressor name, it mean that there is no corresponding timing signal. As no switching information is provided for this aggressor, the crosstalk engine assumes that this aggressor is always an active aggressor.

If the character 'B' or 'W' or both are present on a line, this means that the aggressor can modify minimum propagation delays (B = Best Case) or maximum propagation delays (W = Worst Case). If the character 'R' or 'F' or both are present on a line, this means that the aggressor has made a contribution to calculate the real rise (R) peak noise voltage value or the real fall (F) peak noise voltage value. These characters can appear in lower case ('b','w','r','f') when crosstalk mutex are used. This means the influence of the net is ignored because of the crosstalk mutex. The total value of the coupling capacitance and it relative contribution to the total net capacitance is also given.

2.5.3. Peak Noise Report

This section gives the results of a calculation of upper and lower peak voltages on a net as a result of its aggressors. This list is sorted according to the peak noise voltage value. Only nets for which the peak noise voltage is above the threshold given by the `ctknoisemin` configuration variable are reported.

The syntax of this section is as follows:

```
BeginNoise
  DW|UP <signal> <mod ovr> <max ovr> <real ovr> \\
                 <mod und> <max und> <real und>
  |
  DW|UP <signal> <mod ovr> <max ovr> <real ovr> \\
                 <mod und> <max und> <real und>
EndNoise
```

For each state (UP or DOWN) of a signal, the crosstalk engine gives the peak noise voltage (<max ovr> and <max und>) calculated with all aggressors considered active, and the "real" noise voltage (<real ovr> and <real und>) calculated considering possible switching configurations of aggressors. <mod ovr> and <mod und> are the electrical model used to evaluate peak noise voltage.

Currently, the noise voltage on the net is evaluated by replacing the net driver by a single equivalent constant-valued resistor. This value is either determined from the transistor netlist (model SCR) or, if this is not possible, defined by the `ctknoisedefaultresi` configuration variable (model CC). Both overshoot and undershoot values are calculated for each signal state.

2.6. CTX - Annotated Perfmodule Format

The CTX file (suffix `.ctx`) is an ASCII text file containing all the delays calculated with crosstalk effects of a complete design hierarchy. This file is associated with all of the original detailed perfmodule files describing the hierarchy. It is intended to be viewed using the timing browser Xtas. This file contains four parts:

- General header
- Top level delays
- Instance delays
- End of file

2.6.1. General Header

The CTX file header is the same as the corresponding perfmodule file of the top level of the design:

```
H <tool> <vers> <name> <techno>
<techno_vers> <inslope> <outcapa> <hierarchy_level>
(<day> <month> <year> <hour> <minute>
<second>);
```

2.6.2. Top Level Delays

Part of this file contains all the delays of the top level.

```
M <subcircuit name> (
  D <line index> (
     ((MAX <delay> <slope>))
     ((MIN <delay> <slope>))
    )
  |
  |
  D <line index> (
     ((MAX <delay> <slope>))
     ((MIN <delay> <slope>))
    )
)
```

The <line index> value is the index of the corresponding line in the detailed timing perfmodule.

2.6.3. Instance delays

This section gives all delays for all instances.

```
I <instance name> (
  D <line index> (
     ((MAX <delay> <slope>))
     ((MIN <delay> <slope>))
    )
  |       |
  |       |
  D <line index> (
     ((MAX <delay> <slope>))
     ((MIN <delay> <slope>))
    )
)
|
|
I <instance name> (
  D <line index> (
     ((MAX <delay> <slope>))
     ((MIN <delay> <slope>))
    )
  |       |
  |       |
  D <line index> (
     ((MAX <delay> <slope>))
     ((MIN <delay> <slope>))
    )
)
```

For each instance, the <line index> value is the index of the corresponding line in the detailed timing perfmodule.

2.6.4. The End of File

The end of the CTK file is given by the `G;` article.

Chapter 3. Log Files

3.1. REP - Report File

Each execution of HITAS results in the generation of a report file. This file is given the name <input_name>.rep. It contains a list of diagnostics (warnings and error messages) attributed to particular signals or transistors within the input net-list. Here we explain in more detail the particular messages which you may come across in this report file.

3.1.1. Warning Messages

"[WAR] Possible unconnected supply ?"
Means that an internal signal whose name contains `avtVddName` or `avtVssName` has been found. Verify if this signal should be connected to an external supply, or if `avtGlobalVddName` and `avtGlobalVssName` should be positioned.

"[WAR] Transistor used as a resistance"
Indicates that a transistor P-channel (resp. N-channel) with gate connected to the ground (resp. power supply) has been found in the circuit.

"[WAR] Transistor used as a diode"
Indicates that a transistor with drain (or source) connected to gate has been found in the circuit, and the signal connecting them is neither power supply nor ground.

"[WAR] Transistor is always off"
Indicates that a transistor P-channel (resp. N-channel) with gate connected to power supply (resp. ground) has been found in the circuit.

"[WAR] Transistor used as a capacitance"
Indicates that a transistor with drain and source connected together has been found in the circuit.

"[WAR] Gate of transistor is not connected"
Indicates that a transistor gate which is connected to nothing has been found in the circuit.

"[WAR] Drain of transistor is not connected"
Indicates that a transistor drain which is connected to nothing has been found in the circuit.

"[WAR] Source of transistor is not connected"
Indicates that a transistor source which is connected to nothing has been found in the circuit.

"[WAR] Transistors are not used in the circuit"
This means that these transistors are not used to pull up or pull down any transistor gate in the circuit, or any external connector. This occurs for example if the output of a gate does not drive anything: In this case HITAS considers the transistors of the gate to be unused.

"[WAR] Loop between 2 gates (bleeder found)"
This means that a loop corresponding to a bleeder has been found in the circuit.

"[WAR] Loop between 2 gates (latch found)"
This means that a loop corresponding to a latch has been found in the circuit.

"[WAR] Loop between 2 gates (bi-stable found)"
This means that a loop corresponding to a bi-stable has been found in the circuit.

"[WAR] Loop between 2 gates (nothing found)"
This means that a two gate loop which does not correspond to a latch, bleeder or bi-stable has been found in the circuit.

"[WAR] Conflict may occur on signal"
This means that the signal may be pulled-up and pulled-down simultaneously. This is a warning since this message may disappear with a greater depth for the functional analysis process. Or it may not be possible to resolve the conflict given the logic within the circuit.

"[WAR] HZ state may occur on signal"
This means that the signal is not pulled up or pulled down for any set of input stimuli on the cone entries. This is a warning for the same reason as a conflict.

"[WAR] Signal does not drive anything"
This means that the signal is not used as the input to any gate or used to drive any external connector.

"[WAR] Connector unused"
This means that the external connector is neither the input nor the output of any of the extracted transistor gates.

3.1.2. Error Messages

The presence of any of the following errors will disable the generation of the VHDL or Verilog description. If this behavior is not desired then HITAS must be executed with the `yagleNotStrict` variable.

"[ERR] Bad direction on connector"
Indicates that the orientation of an external connector after disassembly does not correspond to that specified in the input netlist.

"[ERR] Transistor gate signal is not driven"
Indicates that a transistor gate can not be pulled up or down.

3.1.3. Fatal Errors

The following error messages will not be found in the report file. These errors are fatal and will abruptly stop the execution of HITAS.

"[FATAL] No VDD/VSS connector in the circuit"
This means that HITAS did not find any external ports whose name is the name of the power supply in the circuit. Do `avtVddName` and `avtVssName` have the right value?

"[FATAL] Connector is power supply and ground"
This means that HITAS found a connector whose name includes `avtVddName` and `avtVssName`.

"[FATAL] No VDD/VSS signal in the circuit"
This means that HITAS did not find any signal whose name is the name of the power supply in the circuit.

"[FATAL] Several external connectors on signal"
This means that HITAS found several external connectors connected to the same equipotential, a configuration which HITAS considers illegal.

3.2. User-defined Log File

A global log file can be generated, logging the proccessing of all the components of the software. This file is customizable, and user can choose which component to log, and the level of log to apply.

Each line in the log file is beginning with the code related to the logged software component:

`FAC`	file access tracing
`MCH`	disk cache tracing (used for `.stm`, `.rcx` and `.spef` files)
`MCC`	MOSFET characterization
`RCN`	RC networks construction
`TRC`	RC networks characterization
`YAG`	transistor netlist disassembly
`TAS`	information related to delay calculation
`STM`	information related to delay models
`EFG`	spice deck generation

`GSP`	automatic stimuli generation
`TLF`	`.tlf` file generation
`LIB`	`.lib` file generation
`ERR`	error redirection in log file
`PRS`	statistics related to netlist parsing
`SPI`	detailed logging of the spice netlist and technology file parser

The `avtLogFile` variable activates the creation of the log file. The `avtLogEnable` variable selects the software components to log and the level of log. Please refer to the 'Configuration Variables' chapter for more details.

Chapter 4. Configuration Variables

4.1. License Server

avtLicenseServer

`<string>` Hostname of the machine running the license server

avtLicenseProject

`<string>` Project name. Used in license logging.

4.2. Environment

avtLibraryDirs

`<string>` The set of library directories which are scanned for required subcircuits.

avtBlackboxFile

`<string>` Name of the file containing the cells to exclude of analysis.

avtCatalogueName

`<string>` File containing a list of subcircuits to be considered as leaf cells when flattening a design. Each line in this file refers to a single subcircuit, with the format `<subcircuit> C`. The default value is `CATAL`.

4.3. Log File and Error Policy

avtErrorPolicy

`lenient` Default, the tool does not abort when encountering an error of level 1 (level 0 is WARNING, level 1 is ERROR, level 2 is FATAL ERROR).

`strict` the tool aborts when encountering an error of level 2.

avtLogFile

`<string>`	Name of the log file. If this variable is not set, no log file will be created (default behavior).

avtLogEnable

Enables the logging of the different steps of the timing analysis process. The syntax is as follow:

```
avtLogEnable "step:level"
```

the `step` parameter can take one or more of the following values:

`files`	info related to file access: `FAC` and `MCH` sections
`spice`	info related to spice netlist and technology file parser: `SPI` section
`mos_charac`	info related to MOSFET characterization: `MCC` section
`rc`	info related to RC networs processing: `RCN` and `TRC` sections
`disassembly`	info related to transistor netlist partitioning: `YAG` section
`delay_calc`	info related to delay calculation: `TAS` and `STM` sections
`spicedeck`	info related to spice deck generation: `EFG` and `GSP` sections
`timing_abs`	info related to timing abstraction: `LIB` and `TLF` sections
`error`	redirection of errors in the log file: `ERR` section
`stat`	info related to netlist parse: `PRS` section
`config`	info related to configuration `avttools.conf` and `information files`
`stability`	info related to stability: `STABILITY` section

The `level` value ranges from 0 to 9. Level 0 is stdout redirection, level 1 is customer info, levels 2 to 9 are debugging levels.

Example:

```
avtLogEnable "files:1 mos_charac:1
```

avtWarningFilter

`<string>`	Allows the filtering of warning messages on stdout and in the log file. Specify a list of error message identifiers for the messges to be filtered Example: `avtWarningFilter "SPF-003 SPF-015"`

4.4. Names

avtVddName

`<string>` Name of any signal or connector which is to be considered as power supply (a `*` in the name matches any string). Several names, separated by `:`, may be specified.

avtVssName

`<string>` Name of any signal or connector which is to be considered as ground (a `*` in the name matches any string). Several names, separated by `:`, may be specified.

avtGlobalVddName

`<string>` Name of an internal signal to be considered as power supply (a `*` in the name matches any string). Signals in different subcircuits of a hierarchical netlist with a name given here will be considered as equipotential and this name will be used in the flattened netlist. This is identical to the use of the .GLOBAL directive in a spice netlist. Several names, separated by `:`, may be specified.

avtGlobalVssName

`<string>` Name of an internal signal to be considered as ground (a `*` in the name matches any string). Signals in different subcircuits of a hierarchical netlist with a name given here will be considered as equipotential and this name will be used in the flattened netlist. This is identical to the use of the .GLOBAL directive in a spice netlist. Several names, separated by `:`, may be specified.

avtCaseSensitive

`yes` Upper and lower case characters are distinct

`no` Upper and lower case characters are seen as identical

`preserve` Default, upper and lower case characters are seen as identical but the original case is preserved

avtInstanceSeparator

`<char>` Character used to separate instance names in a hierarchical description. Default value is `.`

avtFlattenKeepsAllSignalNames

`yes` When flattening a netlist, each signal keeps all its names through the hierarchy.

`no` Default, only one name (the shortest) is kept per signal.

avtVectorize

Controls the internal representation of vector-signals.

`yes` Default, vector-signals are represented internally as vectors, as far as the vector indexation is one of [], <>, _. For example, if both `foo[1]`, `foo<1>` and `foo_1` appear in the source file, they will all be represented internally as `foo 1`

`no` Vector signals are represented internally as they appear in the source file.

`<string>` Explicits the vector-signals indexations that will be interpreted as vectors, and the represented internally as vectors. `string` is a comma-separated list of single or paired delimiters. For example, if `string` is set to `"[],_"`, only `foo[1]` and `foo_1` will be represented internally as `foo 1`.

Special attention should be paid to the Verilog case. Verilog only accepts `[]` as legal vector indexation. Legal verilog vectors are represented internally as vectors if `avtVectorize` is different to `no`.

Illegal Verilog vectors are supported and controlled by `avtVectorize` as far as they are escaped and `avtStructuralVerilogVectors` is set to `yes`. For example, `\foo<1>` is represented internally as a vector if `avtStructuralVerilogVectors` is set to `yes` and `avtVectorize` is set to `<>`.

4.5. Transistor Characterization

avtElpCapaLevel

Allows the user to compute differents kind of input capacitance.

`0` Input capacitance is the average between up and down capacitance

`1` Default, nominal up and nominal down capa are used to compute timing

`2` Same behavior as if set to 1 but also minimal and maximal capacitances are computed for both transitions (6 capacitances at all).

avtTechnoModelSeparator

`<char>`	Character that will be used as a separator between the model name and the model index. Default value is .

avtElpDriveFile

`yes`	A ELP file specified by `avtElpGenTechnoName` will be printed after transistor electrical characterization.
`no`	default

avtElpGenTechnoName

`<string>`	Name of the generated ELP file. Default is `techno.elp`.

4.6. Input Netlist and Parasitics

avtInputFilter

`<string>`	Shell command line used to decompress an input netlist

avtOutputFilter

`<string>`	Shell command used to compress an output file

avtFilterSuffix

`<string>`	Suffix of the compressed files

avtDisableCompression

`<string>`	Space separated filename list for which compression must be ignored. EXAMPLE: "*.rcx *.rep"

avtAnnotationKeepCards

`transistor`	`M` character is kept before the transistor name
`diode`	`D` character is kept before the diode name
`resistance`	`R` character is kept before the resistance name
`instance`	`X` character is kept before the instance name
`capacitance`	`C` character is kept before the capacitor name
`none`	No character is kept
`all`	`M`, `R`, `X`, `C`, `D` characters are kept

avtMaxCacheFile

`<int>`	If cache mechanisms are used, sets the maximum number of files that can be opened at the same time. Larger the value is, faster is the disk access. Default value is 128. Maximum value depend on your system (see UNIX command `limit`).

avtParasiticCacheSize

`<int>[Kb|Mb|Gb]`	Size (bytes) of the memory cache for all applications dealing with parasitics. Value represents the maximum amount of information stored in memory. Increase this value to lessen disk access and speed-up application. `avtParasiticCacheSize` cannot be used together with compressed files.
`10Mb`	Default
`0`	Disable cache and load all the parasitic information

avtFlattenForParasitic

`yes`	HITAS flattens a hierarchical netlist in order to annotate the netlist with SPEF or DSPF parasitics. To be used together with `avtCatalogueName`
`no`	default

avtVddVssThreshold

`<float>`	Value (in volts) defining the absolute voltage value level above which a node is considered to be a power supply node. Default value is 0.5.

4.7. SPICE Parser

avtSpiCreateTopFigure

`yes`	Default, parser automatically creates a top-level for all elements outside of SUBCKT definition. All equipotentials are made into external connectors. The name of the top-level is the same as the filename without the extension unless a subcircuit of this name exists, in which case the name is prefixed by `top_`
`no`	No top-level is created

avtSpiParseFirstLine

`yes`	First line of all SPICE files are taken into account, unlike the behavior in standard SPICE
`no`	First line of all SPICE files are ignored
`include`	Default, first line of the top-level SPICE file is ignored, but the first line of included files are parsed normally

avtSpiReplaceTensionInExpressions

`yes`	Avoids expression evaluation errors due to unhandled dynamic tensions in expression. The voltage is considered to be 0.
`no`	Default.

avtEnableMultipleConnectorsOnNet

`yes`	By default, there can only be one external connector per net after a netlist parse. If multiple connectors are found, they are merged into one. This can have a big drawback. Connectors required on the interface of a top level netlist can be missing. There can also be issues for ignoring instances with transparencies using hierarchical names as transparences are analysed to build nets prior to check ignored instance resistors. Setting this variable to yes allows multiple external connectors on nets so transparences are analysed during the resistor removal step without the nets being shorted already. This has an effect on ignored instances containing transparences. It affects HITAS behaviour and may make it not work in hierarchical mode.
`no`	Default.

avtSpiMergeConnector

`yes`	Default, connectors with the same radical, but different node indexes, will be merged (they are supposed to belong to the equipotential outside the subcircuit). The separator between the radical and the index is given by `avtSpiConnectorSeparator`.
`no`	Connectors are not merged

avtSpiConnectorSeparator

`<char>`	Character used to separate a connector radical name from its node index (`ck.1`, `ck.2` ... for example).

avtSpiKeepNames

`transistor`	Transistor name is kept in the database
`diode`	Diode name is kept in the database
`resistance`	Resistance name is kept in the database
`allnodes`	All node names are kept for signals in the database
`none`	No name is kept in the database
`all`	All names are kept in the database

avtSpiKeepCards

`transistor`	`M` character is kept before the transistor name
`diode`	`D` character is kept before the diode name
`resistance`	`R` character is kept before the resistance name
`instance`	`X` character is kept before the instance name
`capacitance`	`C` character is kept before the capacitor name
`none`	No character is kept
`all`	`M`, `R`, `X`, `C`, `D` characters are kept

avtSpiNameNodes

`yes`	Default, nodes names are used rather than the node numbers
`no`	Only node numbers are used

avtSpiNodeSeparator

`<char>`	Character that will be used as a separator between the node name and the node number. The default value is _

avtSpiInstanceMultiNode

`yes`	Default, allows two or more identical nodes to be declared in a subckt interface
`no`	Only the first node declared is taken into account

avtSpiIgnoreDiode

`yes`	Diodes are ignored by the SPICE parser.
`no`	Diodes are characterized.

avtSpiMergeDiodes

`yes`	Diodes are merged with neighboring transistors if the transistor is of the same type and area of the connected source or drain is 0.
`no`	Diodes are characterized independantly.

avtSpiIgnoreVoltage

`yes`	Voltage sources are ignored by the SPICE parser.
`no`	Voltage sources are not ignored.

avtSpiIgnoreModel

`yes`	Model directives are ignored by the SPICE parser.
`no`	Model directives are not ignored.

avtSpiIgnoreCrypt

`yes`	Encryption directives (used to indicate encrypted data) are ignored.
`no`	The default. Encryption directives must surround encrypted test obtained by `avt_EncryptSpice` function.

avtSpiJFETisResistance

`yes`	JFETs are considered to be resistances. Values are resolved by the SPICE parser.
`no`	

avtSpiShortCircuitZeroVolts

`yes`	Voltage sources with a value of 0 are modeled by the SPICE parser as resistances of 0 Ohms.
`no`	

avtSpiMaxResistance

`<float>`	If a resistance's value is greater than `float` (in Ohms), then the resistance is considered to be open circuit.

avtSpiMinResistance

`<float>`	If a resistance's value is less than `float` (in Ohms), then the resistance is considered to be short circuit.

avtSpiMinCapa

`<float>`	If a capacitance's value is less than `float` (in Ohms), then the capacitance is ignored

avtSpiOneNodeNoRc

`no`	Removes on all nets containing only one node all parasitics information at the end of the parse.
`yes`	

avtSpiOrderPinPower

`yes`	Uses the name (in the same manner as `avtSpiDspfBuildPower`) of the instance nodes to ensure a correct order for power supply connectors.
`no`	

avtSpiFlags

This configuration is used to control the behavior of the spice parser/driver. The values (flags) are added separated with commas.

`DriveInstanceParameters`	Enables the drive of the instances with all their parameters
`IgnoreGlobalParameters`	Works with `DriveInstanceParameters` and removes all the global parameters from the instance parameters to drive. Useful when the netlist has been flattened and the parameters inherited by the leaf instances.
`KeepBBOXContent`	Will keep the content of the figures set as blackboxes whereas by default only the interfaces are kept.
`TransfertTopLevelVcards`	Will transfert voltage sources connected to instances, who are defined out of a subckt in the spice file, in their corresponding circuit subckt so the Vcards can be taken into account when working on one of this instance circuit. This option is enabled by default. It can be unset by adding '!' in front of the option: '! TransfertTopLevelVcards'.
`ExplicitInstanceNames`	If enabled then instance names specified in the netlist are prefixed by the subckt name in order to create the internally used name.

avtSpiTolerance

This variable tunes the tolerance of the SPICE parser regarding unrecognized syntaxes for R (resistances) and C (capacitances) devices.

`low`	Parser exits when encountering unknown syntax
`medium`	Parser continues and tries to keep only the nominal value of the device, issuing a warning message
`high`	Same as in the `medium` configuration, but no warning message is issued

avtSpiHandleGlobalNodes

`yes`	Default, global nodes defined in spice netlist without resistances will be considered equipotential.
`no`	

4.8. SPICE Driver

avtSpiVector

`_`	Default, vectors are of the shape `foo_1` in output spice files
`[]`	Vectors are of the shape `foo[1]` in output spice files
`()`	Vectors are of the shape `foo(1)` in output spice files
`<>`	Vectors are of the shape `foo<1>` in output spice files

avtSpiDriveDefaultUnits

`<string>`	Its behavior is to indicate the parameter units to be used when instantiating a transistor. For instance, `avtSpiDriveDefaultUnits = W:1e-6;L:1` will set the spice driver to drive parameter W value in micron and parameter L in meter.

avtSpiUseUnits

`yes`	Allows the use of units in driven spice files. This is the default.
`no`	

avtSpiDriveParasitics

`yes` A SPEF file will be generated while parsing a SPICE file. The loaded file will be stripped of all resistors and capacitors. The SPEF file can be used as a parasitic cache file.

`no`

avtSpiDriveTrsInstanceParams

`no` Specifics instances parameters for the models of transistors will not be driven.

`yes`

avtSpiDriveCapaMini

`<float>` When driving a Spice netlist, doesn't drive capacitances below `float` (in Pico-farads). Default is 10-6 pF.

avtSpiDriveResiMini

`<float>` When driving a Spice netlist, fix minimum value for resistances to `float` (in Ohms). Default is 10e-3 Ohms.

avtSpiRCMemoryLimit

`<int>` Amount of memory in MB allowed to creating a .SPEF file from a spice file. This option influences `avtSpiDriveParasitics` speed. The default value is 100.

avtSpiFlags

This configuration is used to control the behavior of the spice parser/driver. The values (flags) are added separated with commas.

`DriveInstanceParameters`	Enables the drive of the instances with all their parameters
`IgnoreGlobalParameters`	Works with `DriveInstanceParameters` and removes all the global parameters from the instance parameters to drive. Useful when the netlist has been flattened and the parameters inherited by the leaf instances.
`KeepBBOXContent`	Will keep the content of the figures set as blackboxes whereas by default only the interfaces are kept.
`TransfertTopLevelVcards`	Will transfert voltage sources connected to instances, who are defined out of a subckt in the spice file, in their corresponding circuit subckt so the Vcards can be taken into account when working on one of this instance circuit. This option is enabled by default. It can be unset by adding '!' in front of the option: '! TransfertTopLevelVcards'.

4.9. VHDL Parser/Driver

avtVhdlMaxError

`<int>`	Maximum number of errors before the VHDL structural parser abandons.

avtStructuralVhdlConfigure

`yes`	VHDL structural driver generates the appropriate configuration statement to allow simulation.
`no`	Default

avtStructuralVhdlSuffix

`<string>`	Suffix of VHDL structural (netlist) file. The default is `vhd`

avtBehavioralVhdlSuffix

`<string>`	Suffix of VHDL behavioral file. The default is `vhd`

4.10. VERILOG Parser/Driver

avtVerilogKeepNames

`yes` When generating Verilog output, any internal names which are not legal verilog names are preceded by a double backslash.

`no` Default. Illegal names are modified to create a legal name.

avtStructuralVerilogVectors

Affects the parsing of illegal Verilog vector-signals in a netlist, i.e. vector-signals that are not indexed using the `[]` characters. Illegal Verilog vector-signals are supported as long as they are preceded by `\`, otherwise the Verilog parser issues a syntax error. Legal Verilog vector-signals are controlled by `avtVectorize`.

`yes` Force illegal Verilog vector-signals to be represented as vectors in the internal database, with regard to the value of `avtVectorize`. For example, `\foo<1>` is represented internally as `foo 1` if `avtVectorize` is set to `<1>`

`no` Default, illegal Verilog vector-signals are represented in the internal database as they appear in the file. For exemple, `\foo<1>` is represented internally as `foo<1>`

avtStructuralVerilogSuffix

`<string>` Suffix of Verilog structural (netlist) file. The default is `v`

avtBehavioralVerilogSuffix

`<string>` Suffix of Verilog behavioral file. The default is `v`

avtVerilogMaxError

`<int>` Maximum number of errors before the Verilog parser abandons.

4.11. DSPF/SPEF Parser

avtAnnotationPreserveExistingParasitics

`yes` Existing parasitics on nets annotated in a DSPF/SPEF file won't be overridden by the parasitics in the DSPF/SPEF file. The DSPF/SPEF information will rather be added to the existing ones.

`no` Default

avtAnnotationDeviceConnectorSetting

`<string>` Overrides the internal tool known device connector names used in DSPF/SPEF annotation. The string must contain 10 items in the following order: transistor source name, transistor gate name, transistor drain name, transistor bulk name, resistor positive connector, resistor negative connector, capacitor positive connector, capacitor negative connector, diode positive connector, diode negative connector. By default, the tool knows of "s g d b 1 2 1 2 1 2" and "s g d b pos neg 1 2 1 2".
Example:

```
avt_config avtAnnotationDeviceConnectorSetting "src gate drn blk 1 2 1 2 1 2"
```

avtSpiDspfBuildPower

`yes` Only used for DSPF annotation. When creating a figure from DSPF information, use the avtGlobalVddName, avtGlobalVssName, avtVddName and avtVssName to detect power connections on the instance, so they are created on the boundary of it instead of being merged with all unknown connectors.

`no`

avtSpiDspfLinkExternal

`yes` Only used for DSPF annotation. When an external connector is not connected to anything, and if there is an internal signal with the same name, then the connector is assumed to be on this signal.

`no`

avtSpiPinDspfOrder

`yes` Only used for DSPF annotation. Order of connector of an instance is the one described in the DSPF instead of the one described for the instance interface.

`no`

4.12. General Configuration

tasHierarchicalMode

`yes`	Hierarchical analysis mode.
`no`	Default, flat transistor analysis mode.

tasBlackboxRequiresTimings

`yes`	HITAS Reads a hierarchical netlist where some instances are considered as blackboxes. The name of these instances should be specified using the `avt_SetBlackBoxes` command. The hierarchical netlist is then flattened to the transistor level, apart from the blackbox instances, to generate a hybrid transistor and instance netlist. The database construction is performed on this hybrid netlist and then HITAS will incorporate timing for the blackboxed instances from an external timing database for the model of the instances which must exist.
`no`	Default

tasTreatBlackboxHierarchically

`yes`	HITAS Reads a hierarchical netlist where some instances are considered as blackboxes. The name of these instances should be specified using the `avt_SetBlackBoxes` command. HITAS creates an intermediate subcircuit containing only the non-blackbox instances. A top-level is also created, and instantiates the intermediate subcircuit and the blackbox instances. Database construction is performed on the intermediate subcircuit.
`no`	Default

tasFigName

`<string>`	Name of the subcircuit to read, or to rename the database.

tasBefig

`yes`	Generates a behavioral description of the design
`no`	Default

tasFlatcells

`yes`	Flattens the timing views of all the models given in the catalogue file specified by `avtCatalogueName`.
`no`	Default

tasSilentMode

`yes` HITAS redirects `stdout` and `stderr` respectively to `.tou` and `.ter` files.

`no`

tasPathFactorisation

`yes` HITAS keeps paths starting and stopping at factorization points that are not reference points. This in order to decrease the number of path of the TTX.

`no` Default

4.13. Disassembly

4.13.1. Functional Analysis

yagAnalysisDepth

`<int>` Allows the user to set the depth for the functional analysis. This is the number of gates that will be taken into account for the functional analysis, so that HITAS can detect re-convergence in the circuit. Default is 7.

`0` Functional analysis process is disabled

yagHzAnalysis

`yes` Allows functional analysis through high impedance nodes.

yagMaxBranchLinks

`<int>` Maximum number of links in a cone branch.

yagRelaxationMaxBranchLinks

`<int>` Used to limit the maximum number of links for the difficult gates for which functional dependencies could not be resolved.

yagBddCeiling

`<int>` Limits the maximum number of BDD nodes which are allowed to be created for the resolution of any Boolean expression. If this limit is exceeded the operation is abandoned. Default is 10 000.

yagElectricalThreshold

`<float>` Used in electrical resolution of conflicts to determine the zones corresponding to the high, low and conflictual states. Default is 4, implying that the high and low states are represented by zones 1/4 of the zone Vss-Vdd.

yagUseStmSolver

`yes` Precise current calculations using technology files are used in electrical conflict resolution.

`no` Default, basic transistor dimensions are used in electrical conflict resolution.

yagRelaxationAnalysis

During the gate construction phase, HITAS attempts to resolve all functional dependencies before building a particular gate. However, in particular cases of looped dependencies, this may not be possible for all gates.

`yes` Functional dependencies are ignored to resolve these gates.

`no` Default, HITAS tries to use as much information as possible.

yagDetectGlitchers

`yes` A branch containing two transistor with mutually exclusive gate drivers and which cannot be part of another gate are assumed to exist dynamically. They are therefore not removed by the functional analysis. This is the default.

`no`

yagKeepRedundantBranches

For any CMOS dual cones extracted, if supplementary branches are added at a later stage of the disassembly and the gate remains non-conflictual, then these branches are considered to be functionally redundant.

`yes` The branches are kept.

`no` Default, the branches are removed.

yagPullupRatio

`<float>` Used in the detection of pull-up or pull-down resistance transistors. Default is 10, implying that a transistor is a pull-up if an estimation of its resistance is greater than 10 times the resistance of the most resistive current path to ground. Similarly for pull-pown resistances.

4.13.2. Transistor Orientation

yagSimpleOrientation

yes	Activates a simple transistor orientation heuristic. Can sometimes accelerate the disassembly, however, it is more robust to rely exclusively upon the functional analysis.
no	Default.

yagUseNameOrientation

yes	Exploits the _s naming convention for transistor orientation.
no	Default.

yagBlockBidirectional

yes	Bidirectional transistors are not allowed.
no	Default.

yagCapacitanceCones

yes	Default. Build cones on nodes with only capacitances. Necessary to calculate the out of path capacitance.
no	Disables construction of cones on only capacitance nodes.

yagTestTransistorDiodes

yes	Default. Any transistor with the gate shorted to a source or a drain is considered as a diode.
no	Disables diode detection.

yagMutexHelp

yes	A file with extension '.mutex' is generated containing help for MUTEX settings on external pins or memory nodes necessary to correctly orient the transistors.
no	Default.

4.13.3. Latch Recognition

yagSimpleLatchDetection

A simple structure based recognition algorithm which handles the various cases of double inverter loops. This approach is not usually required and is not guaranteed to be formally correct but can sometimes help in cases where the automatic approach is too CPU intensive. The following values can be given for this variable:

`memsym` Double inverter loops are also analyzed to see if they correspond to a simple symmetric bitcell. In this case the the command of the bitcell is the input of the pass transister or transfer gate connected directly to the loop.

`levelhold` Double inverter loops are considered to be level-hold or buskeeper structures (i.e. not latches).

`strictlevelhold` Double inverter loops are considered to be level-hold or buskeeper structures (i.e. not latches), but only if only one side of the inverter loop is connected.

`latch` Double invertor loops are treated as latches without any anlysis, unless a level-hold or memsym option is also activated and these forms match. Commands are guessed without analysis. This option helps if double inverter loops are used to latch the output of complex multiplexors.

The above options can be concatenated by separating the individual options with a '+' character. However the combinations "levelhold+strictlevelhold" and "levelhold +latch" make no sense. The search options are applied in the order specified above. By default all the options are disabled.

yagAutomaticLatchDetection

`yes` Advanced latch detection algorithm based on Boolean loop analysis is activated. Default.

`no` Advanced latch detection is disabled.

yagSetResetDetection

`yes` Only works with `yagAutomaticLatchDetection` set to `yes`. Asynchronous latch commands are marked asynchronous instead of being marked commands. False timing arcs (corresponding to the conditioning of data by an asyn or the conditioning of an async by a clock) are disabled.

`remove` Does the same as the `yes` mode. In addition marks as non-functional any branches corresponding to an asynchronous write.

`no` Default.

yagAutomaticRSDetection

`mark`	Default. Only works with `yagAutomaticLatchDetection` set to `yes`. Supplementary RS bistable detection algorithm is applied to automatically detected latches. Only NAND/NOR types are accepted. Any detected RS bistable loops will be marked and reported but not treated as latches.
`no`	Automatic RS detection is disabled. Recognition depends upon `yagAutomaticLatchDetection`
`mark+latch`	One of the gates of the loop is considered a latch. The latch is the gate with the largest number of outputs.
`mark+legal`	The algorithm assumes that an RS structure always remains in its legal states. Timing arcs are suppressed accordingly. For NOR-based RS, the following timing arcs are suppressed: S(f) to QB(r), R(f) to Q(r), QB(r) to Q(f) and Q(r) to QB(f). For NAND-based RS, the following timing arcs are suppressed: S(r) to QB(f), R(r) to Q(f), QB(f) to Q(r) and Q(f) to QB(r).
`mark+illegal`	The algorithm assumes that an RS structure may enter an illegal state. Less timing arcs are suppressed than when the tool assumes that an RS structure always remains in its legal states. For NOR-based RS, the following timing arcs are suppressed: Q(r) to QB(f) and QB(r) to Q(f). For NAND-based RS, the following timing arcs are suppressed: Q(f) to QB(r) and QB(f) to Q(r).

yagAutomaticMemsymDetection

`yes`	Only works with `yagAutomaticLatchDetection` set to `yes`. Supplementary symmetric memory detection algorithm is applied to automatically detected latches. Symmetric memories are memorizing elements such as bitcells for which data is written in both or either side of the memorizing loop. Both sides of the loop are marked as latches is order to verify all cases.
`no`	Default.

yagDetectDynamicLatch

`yes`	Internal tri-state nodes are considered to be dynamic latches for functional modeling and timing analysis purposes. A special algorithm, similar to that used in the automatic latch detection, is used to identify the latch commands and generate an accurate latch model.
`no`	Default.

yagDetectPrecharge

`yes` An algorithm designed to detect automatically most kinds of precharge nodes is activated. The algorithm is particularly designed for domino precharge style designs.

`no` Default.

yagBleederStrictness

A level between 0 and 2 defining the strictness of the bleeder detection algorithm. The value determines the kind of gate which can be tolerated in the bleeder loop.

`0` any CMOS gate is acceptable

`1` default, any CMOS dual gate is acceptable

`2` it must be an inverter

yagStandardLatchDetection

Deprecated. This structure based latch recognition technique is activated by default as a catch-all. It will probably be removed in a future version.

`yes` Default, standard latch detection algorithm is activated.

`no` Standard latch detection algorithm is disabled.

yagLatchesRequireClocks

`yes` Any latch which does not have a command which is at the end of a path from a specified clock is not considered to be a latch. If this option is used, then extreme care should be taken to specify the clocks to avoid problems in any subsequent analysis.

`no` Default.

yagDetectClockGating

`yes` If clocks are configured before the disassembly phase then reconvergence between clock and data will be detected, appropriate timing check and data filtering directives are automatically generated.

`check` Same as above except only the timing checks are added.

`filter` Same as above except only the data filtering directives are added.

`no` Default.

yagDetectDelayedRS

`yes` Detect special type of NAND/NOR bistable loop structure containing additional inverters to add delay in the loop. Results in the same handling as the `legal` setting for RS detection. This type of structure is commonly used to generate non-overlapping clocks.

`no` Default.

4.13.4. Pattern Matching

yagUseGenius

`yes` Extends the simple pattern recognition of FCL to allow the recognition of hierarchically defined structures of generic size.

`no` Default

yagUseOnlyGenius

`yes` Same as `yagUseGenius` but HITAS stops the execution after the hierarchical pattern recognition phase.

`no` Default

4.13.5. Cone Output Files

avtVerboseConeFile

`yes` Generating a `.cnv` cone file result in a more readable version but which is not suited for GUI vizualisation.

`no` Default.

avtNormalConeFile

`yes` A normal `.cns` cone file is produced.

`no` Default.

avtFullConeFile

`yes` `.cnv` or `.cns` cone files are generated with parasitic information.

`no` Default.

yagGenSignature

yes	Signatures are generated for each cone which are used to associate icons with the cones.
no	Default.

4.14. Timing DB Construction

4.14.1. Special Elements

tasTreatPrecharge

yes	Precharged signals are also considered as precharged if its name is suffixed by _p or if it is declared in the INF file. To be used only in the flat analysis mode.
no	Default

tasMemoryCharacterization

yes	Default, HITAS adds the latches' intrinsic setup, hold and access delays
no	

tasPreserveConnectorsDirection

yes	Preserve connectors direction if there are level constraints on connectors.
no	

4.14.2. Output Files

tasGenerateConeFile

yes	HITAS generates the .cns or .cnv files containing the cone view. To be used only in flat analysis mode.
no	Default

4.14.3. Delay Models

stmCacheSize

`<int>[K|M]` Size of the memory cache for all applications reading the STM file. The default unit is the Octet. If the value is followed by either `K` or `M` the unit is the kilo-Octet or the Mega-Octet.

`0` Default, the cache utilization is disabled and all the timing model information are loaded. `stmCacheSize` cannot be used together with compressed files.

stmShareModel

`yes` Identical cones will share the same timing model. There is a 1% tolerance in the matching of identical cones, regarding capacitance related parameters of the model. Extractors, especially when used in crosstalk mode, may show slight variations in the values of extracted capacitances for different instances of a same cell.

`no` Default

tasShortNamesForModels

`yes` HITAS will use model numbers instead of the model names

`no` Default

4.14.4. Delays

avtNewSwitchModel

`yes` HITAS will use an enhanced switch delay calculation model which accurately takes into account contributions from both transistor of the switch and also handles the difference between the opening times of the two transistors.

`no` Default

tasDelayPropagation

`yes` Default, HITAS will propagate through a gate the slope of this gate's input having the latest arrival time.

`no` HITAS will propagate through a gate the largest (or smallest) slope. Adds additional pessimism.

tasRefineDelays

`yes` HITAS will will perform an additional delay calculation phase after DTX generation using the slope propagation algorithm specified in `tasDelayPropagation`. This allows perfect delay value correlation with the result of delay recalculation after a slope or lod change.

`no` Default.

tasSwitchCapacitanceFactor

`<int>` Percentage of the 'out-of-path' capacitances associated with an input connector taken into account during a flat analysis. Default is 100. Affects only TTX and DTX file generation and does not change the delay computation.

tasPathCapacitanceFactor

`<int>` Percentage of the out-of-path capacitances taken into account. Default is 100.

tasPathCapacitanceDepth

`<int>` To be used together with `tasPathCapacitanceFactor`. Controls across how many transistor the out-of-path capacitance should be counted. Default is 1.

tasStrictPathCapacitance

To be used together with `tasPathCapacitanceFactor`.

`yes` Any out-of-path capacitance on internal nodes of any gate is ignored.

`no` Any out-of-path capacitance on internal nodes of a gate is considered on the gate output.

`latch` Default. Any out-of-path capacitance on internal nodes of a latch is ignored.

tasMaxPathCapacitanceFanout

`<int>` To be used together with `tasPathCapacitanceFactor`. If the number of gates contributing to an out-of-path capacitance for a given cone exceeds this value then only the largest contribution is taken into account. Effectively a mutual exclusion is assumed. Default is 15.

4.14.5. RC Networks

tasCalcRCDelays

`no`	Only the capacitances will be taken into account to compute the propagation delays
`yes`	Default

tasMergeRCAndGateDelays

`yes`	Merges the gate and RC delays in the flat analysis mode
`no`	Default

rcxMinRCSignal

`<int>`	If the sum of resistances multiplied by the sum of capacitances on an RC network is less than `int` (ps), then the RC network is ignored. Default is 1 ps.

rcxCapaLoadMaxRC

`<int>`	If the sum of resistances multiplied by the sum of capacitances on an RC network is less than `int` (ps), then the load of the driving gate is modeled by an equivalent substrate capacitance. Default is 5 ps.

rcxMaxDeltaLoad

`<int>`	If the `rcxCapaLoadMaxRc` test fails and the difference between an equivalent substrate capacitance model and a PI network model is less than `int` (ps), then the equivalent capacitance model is used, otherwise the PI model is used. Default is 1 ps.

tasRCDriverCalcMode

Controls the choice of which node or nodes of an RC network are driven by a preceding gate.

`all`	All possible input nodes of the RC are considered driven.
`all_direction`	All nodes of the RC which can be driven by the actual transition are assumed driven.
`select_direction`	One node of the RC which can be driven by the actual transition is assumed driven.
`all_gates`	The first three options apply to RC networks driven by any type of gate.
`parallel_gates`	The first three options apply to RC networks driven by gates containing parallel transistors or current paths.
`none`	One node of the RC is considered driven.
`auto`	Automatic mode which differentiates input transition and chooses one or all drivers based on gate type.

One of the first three options can be concatenated with one of the next two by separating the individual options with a '+' character. The default setting is "all_direction +parallel_gates".

rcxAweMatrix

`always`	Matrices are always used to compute the moments for RC delay with AWE algorithm
`ifneed`	Default, matrix is not used if RC network has one node driver and has a tree topology. In this case, a faster algorithm is used to compute the moments.
`never`	No matrix is used. RC networks which can't be handle by the faster algorithm are not computed.

rcxDelayCacheSize

`<int>[Kb|Mb|Gb]`	Cache size for gate output equivalent load and RC moment in AWE delay evaluation. When one of these values is computed, results are stored in the delay cache. If later re-computation is needed, results can be retreived immediatly. This cache is mainly used within the timing abstration engine. Default unit is the byte.
`0`	Disables the cache. Default value is 10Mb.

4.15. Path Browsing

avtPrecisionLevel

`0`	Default, slope propagation is done according to `tasDelayPropagation`
`1`	When computing propagation time for a path, the tool re-propagates the slope from start point, and computes delays along the path accordingly.

avtMaxPathPeriodDepth

`<int>`	When performing a path search, the tool needs a parameter to avoid loop search. This is the maximum number of period that a path can go through or the maximum number of latches at the same clock edge. Default value is 1.

avtTransparentPrecharge

`yes`	Sets `avtMaxPathPeriodDepth=1` only for the precharges so it's possible to report paths through domino precharges.
`no`	Default.
`unfiltered`	Same as `yes` but deactivates the precharge filtering effect so non existing path due to filtering will be kept.

4.16. SDC Support

sdcUnits

Sets the time unit and capacitance unit used in the Synopsys Design Constraint (SDC) files. Unit for time and unit for capacitance should appear both in a space-separated list (ex: `"1ps 1ff"`). Available values are:

`1ps`	Default
`10ps`	
`100ps`	
`1ns`	
`1ff`	
`10ff`	
`100ff`	
`1pf`	Default

4.17. Static Timing Analysis

stbDetailedAnalysis

`yes`	Enables multi-switching-windows analysis with the STA engine. The STA engine will use more memory and cpu.
`no`	Default

stbDetailedGraph

`yes`	The STA engine uses the `.dtx` file and computes switching windows for each point of the timing graph. It is required for crosstalk analysis.
`no`	Default

stbTraceMode

`yes`	The STA engine displays all intermediary values of the switching windows calculations on th standard output. Useful to see how the relaxation progresses.
`no`	Default

stbReportFile

`yes`	Default, the STA engine generates a `.str` timing report file containing details of all setup and hold slacks for reference points.
`no`	the STA engine does not generate the `.str` timing report file

stbOutFile

`yes`	The STA engine generates a `.sto` switching windows file containing details of the switching windows of reference points.
`no`	the STA engine does not generate the `.sto` switching windows file

stbSetupOnly

`yes`	Only errors due to setup time violations are reported.
`no`	Default

stbHoldOnly

`yes`	Only errors due to hold time violations are reported.
`no`	Default

stbEnableCommandCheck

`yes`	Commands will be checked for violations.
`no`	Default

stbMonoPhase

`flip_flop`	a latch clocked on the same phase than the latch generating its input data is assumed to be a flip-flop.
`transparent`	a latch clocked on the same phase than the latch generating its input data is always transparent.
`errors`	a latch clocked on the same phase than the latch generating is input data is not allowed, and an error is reported.

stbSaveErrors

`yes`	Errors are redirected to a separate `.ste` error file.
`no`	Default

stbSilentMode

`yes`	The STA engine displays neither errors nor warning on the standard output.
`no`	Default

stbTopLevelPath

`yes`	Violations are computed using only the paths at the top level (i.e. the interconnections at the top level).
`no`	Default

stbWorstCaseAnalysis

`yes`	Worst case analysis is performed by assuming that in the initial conditions, there is no latch transparency.
`no`	Default

stbCorrelatedSkewAnalysisDepth

`number`	Specifies the maximum search depth allowed to find common clock path when computing setup and hold values at each node. The value 'full' means unlimited.
`full`	Default

stbStabilityCorrection

`yes`	The false paths and false slacks are taken into account to correct the global stability time computed by STB. This corrects false data lags hence too pessimistic setup and hold computations.
`no`	Default

StbSuppressLag

`latch`	Transparency (data lag) will be ignored for latches.
`precharge`	Transparency (data lag) will be ignored for precharges.
`yes`	Transparency will be ignored for any of the above.

stbHelpForSetup

`yes`	Gives additional information to help setting the stability configuration. The information is driven in the log file.
`no`	Default

4.18. Statistical Analysis

spiActivateStatisticalFunctions

`yes` Enables montecarlo runs. The statitical functions will no more return the nominal values.

`no` Default

avtStatisticalDiscretisation

`number` Apply a discretisation to the values returned by random functions. The default value is 10. A high value means less discretisation but 0 deactivates the discretisation. This variable can have a great impact on HiTas speed.

4.19. Crosstalk Analysis

4.19.1. Running

stbCtkWorstBeginCondition

`yes` The SI engine performs a worst case static timing analysis with crosstalk. All aggression is assumed initially. Any false aggression is detected and removed.

`no` Default

stbCrosstalkMode

`yes` Enables crosstalk analysis.

`no` Default

stbCtkObservableMode

`yes` Default. The SI engine performs a best case static timing analysis with crosstalk. No aggression is assumed initially. Any aggression is detected and added. Uses a less pessimistic algorithm to detect observable aggression.

`no`

stbCtkNoInfoActif

`yes` The SI engine considers aggressors not defined in the timing view as active aggressors. No information on an aggressor are usually due to a coupling capacitance between usual net and an internal gate net.

`no` Default.

stbCtkMinOccurenceProbability

`value` [IN ALPHA DEVELOPMENT STAGE] Defines the probability value (from 0 to 1) under which a agression occurence probability is considered as not interesting during a crosstalk analysis. Default value is 1.

`no` Default.

stbCtkReportFile

`yes` The SI engine generates a `.ctk` crosstalk report file.

`no` Default.

4.19.2. Models

stbCtkMargin

`<float>` Minimum delay between two switching windows implying that there is no possibility of aggression. Default value is 0 picoseconds.

ctkNoiseDefaultResi

`<float>` Resistance in Ohms representing the driving strengh of input pins, in order to obtain pertinent peak noise values.

`10000` Default value

rcxCtkModel

Model used to compute elementary delay when taking into account crosstalk effect if an aggression is detected.

`MILLER_0C2C` Coupling capacitance is removed or replaced to a ground capacitance with a doubled value.

`MILLER_NOMINAL` Default, coupling capacitance is replaced to a ground capacitance with a value multiplied by 0.0 to 2.0 according to relative slope computed without coupling effect between victim and its aggressor.

rcxMinRelCtkFilter

`<float>` Number between 0 and 1. If the ratio between one coupling capacitance and the total capacitance is less than `float`, then the filtering of this coupling capacitance is controlled by `stbCtkNoInfoActif`. Default is 0.05

rcxMinRelCtkSignal

`<float>` Number between 0 and 1. If the ratio between the sum of coupling capacitance and the total capacitance is less than `float` then all the coupling capacitances are filtered. Default is 0.05

rcxCtkSlopeDelay

`SLOPE_DELAY_CTK` Default, basic slope is used during delay calculation. Gives pessimistic results, but is faster

`SLOPE_DELAY_ENHANCED` Removes the contribution of the victim on the slopes of its aggressors. More precise but requires more computation time

rcxCtkSlopeNoise

`SLOPE_CTK` Default, uses worst slopes with crosstalk to compute voltage noise

`SLOPE_NOMINAL` Uses slopes computed without aggression to compute voltage noise

`SLOPE_REAL` Removes the contribution of the victim on the slopes of its aggressors. Requires more computation time but gives better results

4.19.3. Convergence

stbCtkMaxIteration

`<int>` Defines the maximum number of iteration the SI can perform. Default value: 999

stbCtkminSlopeChange

`<int>` When no further aggression is detected or removed, this variable represents the minimum slope variation, in ps, that induces slope re-calculation. Default is 2

stbCtkMaxLastIter

`<int>` When no further aggression is detected or removed, this variable represents the maximum number of iterations of slope re-calculation. Default is 3

4.19.4. Reports

ctkDeltaDelayMin

`<int>` Minimum amount, in ps, of the propagation delay variation, versus crosstalk-free propagation delay. If the variation is greater than `int`, then it is reported in the `.ctk` file. Default is 0 ps.

ctkDeltaSlopeMin

`<int>` Minimum amount, in ps, of the slope variation, versus crosstalk-free slope. If the variation is greater than `int`, then it is reported in the `.ctk` file. Default is 0 ps.

ctkNoiseMin

`<int>` Minimum value, in mV, of the voltage noise to be reported in the `.ctk` file.

ctkCapaMin

`<int>` Number between 0 and 100. Minimum percentage of coupling capacitance. If the ratio of the coupling capacitance on a net is greater than `int` then it is reported in the `.ctk` file.

stbCtkMaxReportedSignals

`<number>` Number of signals to report at each crosstalk iteration. Setting the number to 0 disables the report generated at each iteration. Default is 0.

4.19.5. Scores

stbCtkCoefNoise

`<int>` Between 0 and 10. Specifies the noise voltage score part of the total score. Default is 4.

stbCtkMinNoise

`<int>` Minimum noise voltage score under which no report for this net is provided.

stbCtkCoefInterval

`<int>` Between 0 and 10. Specifies the interval score part of the total score. Default is 3.

stbCtkMinInterval

`<int>` Minimum interval score under which no report for this net is provided.

stbCtkCoefCtk

`<int>` Between 0 and 10. Specifies the crosstalk score part of the total score. Default is 2.

stbCtkMinCtk

`<int>` Minimum crosstalk score under which no report for this net is provided.

stbCtkCoefActivity

`<int>` Between 0 and 10. Specifies the activity score part of the total score. Default is 1.

stbCtkMinActivity

`<int>` Minimum activity score under which no report for this net is provided.

4.20. Timing Abstraction

4.20.1. Input Files

tmaLibraryFile

`<string>` Allows the user to specify an input library file containing the areas of the cells

tmaLibraryName

`<string>` Allows the user to specify the name which will appear in the `library()` staement

4.20.2. Output Files

tmaFunctionalityMode

`w` The timing abstraction engine generates a behavioral description

`r` The timing abstraction engine reads a behavioral description

`t` Default, the timing abstraction engine generates timing information only

tmaLibBusDelimiter

`[]` Default, vectors in the lib file are of the kind `foo[1]`

`_` Vectors in the lib file are of the kind `foo_1`

`-` Vectors in the lib file are of the kind `foo-1`

`{}` Vectors in the lib file are of the kind `foo{1}`

`()` Vectors in the lib file are of the kind `foo(1)`

`<>` Vectors are in the lib file are of the kind `foo<1>`

tmaDriveCapaout

`yes` Timing abstraction engine prints the output connectors capacitance in the `.lib` file

`no` Default

tmaLibSlewDerate

`<float>` Affects the `slew_derate_from_library` attribute of the Liberty file. Value should be comprised between 0 and 1.

tmaLibDriveTableIndex

`yes`	Enables indexes in LIB timing groups.
`no`	Default

avtPowerCalculation

`yes`	Enables leakage and switching power calculation.
`leakage`	Enables leakage power calculation.
`switching`	Enables switching power calculation.
`no`	Default

4.20.3. Units

tmaCapacitanceUnit

`pf`	Capacitance unit in pico-Farads
`ff`	Default, capacitance unit in femto-Farads

tmaTimeUnit

`ns`	Time unit in nanoseconds
`ps`	Default, time unit in picoseconds

tmaLeakagePowerUnit

`<string>`	Leakage power unit in LIB file. Valid values are 1mW, 100uW, 10uW, 1uW, 100nW, 10nW, 1nW, 100pW, 10pW and 1pW.
`1uW`	Default, leakage power unit.

4.21. Pattern Matching

fclLibraryName

`<string>`	Name of the file containing the list of cells in the user-defined cell library used. The default is `LIBRARY`.

fclLibraryDir

`<string>`	Access path to the directory containing the user-defined cell library used. Default is a directory `/cells` in `avtWorkDir`.

fclGenericNMOS

`<string>` A colon separated list of transistor model names which the FCL pattern-matching engine considers will match to any N-type transistor. If a pattern netlist contains non-generic N-channel transistors then these transistors will only match to transistors with an identical model. Default is `tn:TN`.

fclGenericPMOS

`<string>` A colon separated list of transistor model names which the FCL pattern-matching engine considers will match to any PMOS transistor. If a pattern netlist contains non-generic P-channel transistors then these transistors will only match to transistors with an identical model. Default is `tp:TP`.

fclWriteReport

`yes` A correspondence file is created if the -fcl option is used. This file details all the recognized instances.

`no` Default

fclAllowSharing

`yes` Matched cells are allowed to share transistors.

`no` Default

fclCutMatchedTransistors

`yes` Matched transistors are eliminated from the transistor netlist. Results in a strict partitioning of the cones and the matched cells.

`no` Default

fclMatchSizeTolerance

`<int>` Percentage tolerance for matching transistor sizes.

fclTraceLevel

`<int>` Number greater than 0. Trace information is displayed during the pattern-matching phase.

fclDebugMode

`<int>` Number greater than 0. Additional debugging information is displayed during the pattern-matching phase.

4.22. Hierarchical Pattern Matching

gnsLibraryName

`<string>` Name of the file (recognition library) containing the list of cells to recognize. Default is `LIBRARY`.

gnsLibraryDir

`<string>` Access path to the directory containing the recognition library. Default is directory `cells/` in `avtWorkDir`.

gnsKeepAllCells

`yes` All matched structures are extracted from the netlist.

`no` Default

gnsTemplateDir

`<string>` Directory where to find the GNS templates. Default is `$AVT_TOOLS_DIR/gns_templates`.

gnsTraceLevel

`<int>` From 0 to 6. Indicates the level of trace displayed during the recognition phase. Default is 0.

gnsTraceFile

`<string>` Name of the output trace file. Default is stdout.

gnsTraceModel

`<string>` When tracing the recognition, indicates the name of the recognized model to trace. If not specified, traces all models.

gnsFlags

This configuration controls the behavior of GNS. The values (flags) are added separated with commas. Available flags are:

`EnableCore` Enable the generation of a core file for a crash in a user compiled API.

`NoGns` Disables the generation of the `.gns` file

`VerboseGns` Produces a more readable `.gns` file.

`NoOrdering` Disables the top-level instance connectors reordering. Should not be set if using the BEG functions.

4.23. Simulator Linking

4.23.1. Simulation Tool Parameters

simToolModel

`Spice`	Technology files are interpreted in the same way as Berkeley Spice.
`Hspice`	Technology files are interpreted in the same way as Hspice. This is the default setting unless `simTool` is set.
`Eldo`	Technology files are interpreted in the same way as Eldo.
`Titan`	Technology files are interpreted in the same way as Titan.

simTool

`Spice`	Generates spicedeck and sets simulator linking options for Berkeley Spice.
`Hspice`	Generates spicedeck and sets simulator linking options for Hspice. This is the default setting unless `simToolModel` is set.
`Eldo`	Generates spicedeck and sets simulator linking options for Eldo.
`Titan`	Generates spicedeck and sets simulator linking options for Titan.
`Titanv7`	Generates spicedeck and sets simulator linking options for Titanv7.
`Mspice`	Generates spicedeck and sets simulator linking options for Mspice.
`Ltspice`	Generates spicedeck and sets simulator linking options for Ltspice.
`Ngspice`	Generates spicedeck and sets simulator linking options for Ngspice.

avtSpiceString

`<string> $`	Command line used to call the electrical simulator. The syntax is `string $`, where `string` is the full command line for the simulator.

simSpiceOptions

`<string>`	Simulator options written into the Spice Deck with a `.OPTION` statement

simTransistorAsInstance

`yes`	Transistors are written as instances (with an 'X') in the spice file
`smart`	Each transistors is written as instance or not depending on the its original setting
`no`	Default

avtSpiceOutFile

`$.<string>`	Output format generated by the simulator, containing the simulation results. Multiple formats may be concatened using the character `:`. `string` is the suffix of the output file.

avtSpiceStdoutFile

`$.<string>`	Output format generated by the simulator, for stdout redirection `string` is the suffix of the output file. Default is `$.out`

simAllowOverwriteFile

`yes`	Default, allows overwrite of existing files when running a simulation.
`no`	File overwriting not allowed.

simRemoveFiles

`yes`	Files written by HITAS for simulation are removed after execution
`no`	Default

simTechnologyName

`<string>`	Name of the technology file. Appears in the spice deck in a `.INCLUDE` statement. Multiple file names can be given using commas to separate them.

4.23.2. Extraction of the Simulation Results

simUsePrint

`yes`	Default, generates a spice deck with `.PRINT` statements printing simulation results as curves represented as tabular data. Delay and slope values are then computed by HITAS. CPU and disk expensive.
`no`	

simUseMeasure

`yes`	Generates a spice deck with `.MEASURE` statements directly extracting delay and slope values from the simulation results.
`no`	Default

simMeasCmd

To be used in combination with `SimUseMeasure`. Explicits the `.MEASURE` statements used in the spice deck.

- `%l` will be the label of the extraction.
- `%s<n>` will be the (trigger/target) node name.
- `%v<n>` will be the initial value of the (trigger/target) node.
- `%t<n>{<rise>,<fall>}` will be the (trigger/target) transition and the syntax of `<rise>` and `<fall>` for the simulator.
- `%n<n>{<last>}` will be the (trigger/target) number of transition and the syntax of `<last>` for the simulator.

Example: `.meas tran %l TRIG v(%s1) val=%v1 %t1{rise,fall}=%n1{LAST} TARG v(%s2) val=%v2 %t2{rise,fall}=%n2{LAST}`.

simExtractRule

Allows the user to specify the exact stimuli with metadata to be recognized to get result from simulation result file (used when SimUseMeasure is set to `yes`).

- First part: the type of pattern recognition. Value can be `line` or `table`.
- Second part: the mark-up of start and end for pattern recognition. Starting and ending mark-up are encapsulated by slash.
- Third part: the exact pattern containing metadata to be recognized. In case of table recognition, each metadata is considered as followed by a return.
- %l represent the extraction label.
- %i represent a string to be ignored.
- %v represent the value associated with the label.
- %0 represent a value to be ignored.

Example: `line:/EXTRACT INFORMATION/1****/:* %l = %v targ= %0 trig= %0` applied to

```
*************** EXTRACT INFORMATION ********************
*
* a_up_d_down_delay = 1.5e-12 targ= 5e-15 trig= 15e-8
* a_down_d_up_delay = 1.2e-12 targ= 5e-15 trig= 15e-9
*
1****
*
* a_up_d_down_slope = 1.5e-12 targ= 5e-15 trig= 15e-8
```

will match a_up_d_down_delay and a_down_d_up_delay.

simDriveAliasCorrespondence

`yes`	If `simSignalAlias` or `simTransistorAlias` is also set, correspondence between original name and their alias will be printed in the spice deck
`no`	Default

simSignalAlias

`<string>`	Signals names in the critical path will be replaced by `string` followed by a number.

simTransistorAlias

`<string>`	Transistor names in the critical path will be replaced by `string` followed by a number.

simAnalysisDepth

`<int>`	Maximum depth to analyze correlated signals. Default is 1. This variable can reveal false paths.
`0`	No correlated signals analysis

cpeMaxVariables

`<int>`	Maximum number of variables below which CPE will try an exhaustive search for the best path propagation solution. Default is 20.

cpePrechargedMemsym

`<yes>`	CPE will consider the symmetric memories to be precharge when computing propagation conditions.
`<no>`	Default.

4.23.3. Simulation Conditions Parameters

simOutLoad

`dynamic`	The gate/drain/source capacitances of out-of-path transistors are modeled in the spice deck as nominal capacitances. Recomended, need the electrical parameters of the transistors to be loaded before spice deck generation
`transistor`	Default, the gate/drain/source capacitances of out-of-path transistors are modeled in the spice deck as blocked transistors. It can increase significantly the size of the spice deck, and has very small impact on results accuracy.
`none`	No out-of-path capacitances. Debugging purpose.

simPowerSupply

`<float>`	Simulation voltage in Volts. Default is 5 Volts.

simTemperature

`<float>`	Simulation temperature in Celsius degrees. Default is 70C.

simTransientTime

`<float>`	Duration of the simulation in seconds. Default is 100e-9 seconds.

4.23.4. Simulation Transient Parameters

simTransientStep

`<float>`	Calculation step, in seconds, for the simulation and the internal data representation. Default is 1e-12 seconds.

simDcStep

`<float>`	Accuracy step, in Volts, of the static parameters evaluation when using HITAS, and step of the SPICE analysis in DC mode. Default is 1e-3 Volts.

4.23.5. Simulation Thresholds Parameters

simVthHigh

`<float>`	Between 0 and 1. High threshold of the slope as a percentage of the Vdd value. Default is 0.8

simVthLow

`<float>` Between 0 and 1. Low threshold of the slope as a percentage of the Vdd value. Default is 0.2

4.23.6. Simulation Input/Output Constraints Parameters

simInputStartTime

`<float>` Starting time of the input slope in seconds. Default is 0.

simSlope

`<float>` Transient time of the slope in seconds. Default is 200e-12 seconds.

4.24. API Specific

apiFlags

Controls the behavior of the GNS API. The values (flags) are added separated with commas. Available flags are:

`ttvUseInstanceMode` Sets the TTV functions to generate/use one timing view per instance of the same matched subcircuit.

`ttvDriveDTX` Enables the drive of the `.dtx` and `.stm` files for timing views created with the TTV functions

4.25. SSTA Analysis

avtSSTACacheSize

`<float>[unit]` Size of the memory allowed to read a set of SSTA stored files. The default value is 500Mb.

Chapter 5. Tcl Interface

5.1. Objects

5.1.1. Netlist

The Netlist object describes the electrical view of a Subcircuit. It may be either flat or hierarchical, and contains components of the files it originates from: mosfets, resistances, capacitances and instances.

5.1.2. TimingFigure

Property codes:

`TOP_LEVEL`	`TimingFigure*`. Top level TimingFigure the current TimingFigure belongs to.
`STABILITY_FIGURE`	`StabilityFigure*`. Stability figure built on the current TimingFigure.
`NAME`	`char*`. Name of the timing figure given as parameter.
`TEMP`	`double`. Reference temperature the TimingFigure has been built with.
`DEF_SUPPLY`	`double`. Reference power supply the TimingFigure has been built with.
`DEF_LOAD`	`double`. Default load for output connectors.
`DEF_SLOPE`	`double`. Default slope for input connectors.
`TH_LOW`	`double`. Low threshold in slope computing
`TH_HIGH`	`double`. High threshold in slope computing
`TECH_NAME`	`char*`. Technolgy name
`DATE`	`char*`. Date of generation
`TIME`	`char*`. Time of generation

Properties described above are available through function:

```
ttv_GetTimingFigureProperty <timing_figure> <code>
```

5.1.3. TimingSignal

Property codes:

`IS_CLOCK`	`char*`. Returns `yes` if the timing signal is defined as a clock. `no` otherwise.
`IS_ASYNCHRONOUS`	`char*`. Returns `yes` if the timing signal is defined as an asynchronous pin. `no` otherwise.
`DIR`	`char`. Direction of connector. Values can be `i` (input), `o` (output), `b` (bidirectional), `t` (tristate), `z` (HZ), `u` (unknown).
`CAPA`	`double`. Total capacitance on signal (in Farads).
`LOAD`	`double`. External load if the signal is an output connector (in Farads).
`LEFT_BOUND`	`int`. Left bound of connector range. If signal is not a vector, value is `-1`.
`RIGHT_BOUND`	`int`. Right bound of connector range. If signal is not a vector, value is `-1`.
`TOP_LEVEL`	`TimingFigure*`. Top level timing figure the timing signal belongs to.
`NET_NAME`	`char*`. Net name of timing signal.
`NAME`	`char*`. Name of timing signal.
`TYPE`	`char*`. Type of a timing signal. Possible values are `connector`, `command`, `latch`, `flip-flop`, `precharge`, `breakpoint`, `internal`, `signal` or an eventual concatenation of possible configurations.
`EVENT_UP`	`TimingEvent*`. Pointer to the rising TimingEvent related to the TimingSignal.
`EVENT_DOWN`	`TimingEvent*`. Pointer to the falling TimingEvent related to the TimingSignal.
`RISING_SLOPE`	`double`. Input slope set on TimingSignal as an external constraint (only valid for input connectors).
`FALLING_SLOPE`	`double`. Input slope set on TimingSignal as an external constraint (only valid for input connectors).
`LEAKAGE_UP_MAX`	`double`. Maximum leakage current in the high state (in Amps).

`LEAKAGE_UP_MIN`	`double`. Minimum leakage current in the high state (in Amps)e.
`LEAKAGE_DN_MAX`	`double`. Maximum leakage current in the low state (in Amps).
`LEAKAGE_DN_MIN`	`double`. Minimum leakage current in the low state (in Amps).
`GATE_TYPE`	`char*`. Returns the name of the gate type driving the timing signal if it is known.
`VOLTAGE_SWING`	`char*`. Returns the voltage difference between the vdd and vss sypply nodes on the signal.

Properties described above are available through function:

```
ttv_GetTimingSignalProperty <sig> <code>
```

5.1.4. TimingEvent

Property codes:

`SIGNAL`	`TimingSignal`. Signal of the timing event.
`TRANS`	`char`. Transition direction of the timing event. Value can be `u` (rising) or `d` (falling).
`STABILITY`	`list of list`. The list of instability gaps of the timing event available when STB has been run on the timing figure. A range is a list of 2 values. eg: `{{0 10} {20 40}}`.

Properties described above are available through function:

```
ttv_GetTimingEventProperty <timing_event> <code>
```

5.1.5. TimingPath

Property codes:

`DELAY`	`double`. Delay attached to the timing path. If a crosstalk analysis has been run on the timing figure, this delay take into account the crosstalk effect.
`REF_DELAY`	`double`. Delay attached to the timing path. Crosstalk effect is not taken into account.
`SIM_DELAY`	`double`. Simulated delay attached to the timing path. Returns -1 if simulation had failed.
`DATA_LAG`	`double`. Only for clock-to-q access paths: waiting delay for the data to be available on the latch's input, after the latch opening (value is present only after stability analysis).
`SLOPE`	`double`. Output slope attached to the timing path. If a crosstalk analysis has been run on the timing figure, this slope take into account the crosstalk effect.
`REF_SLOPE`	`double`. Output slope attached to the timing path. Crosstalk effect is not taken into account.
`SIM_SLOPE`	`double`. Simulated output slope attached to the timing path. Returns `-1` if simulation had failed.
`START_TIME`	`double`. Start time of the timing path.
`START_SLOPE`	`double`. Start slope of the timing path.
`START_SIG`	`TimingSignal*`. Starting node of the timing path.
`END_SIG`	`TimingSignal*`. Ending node of the timing path.
`START_EVENT`	`TimingEvent*`. Starting timing event of the timing path.
`END_EVENT`	`TimingEvent*`. Ending timing event of the timing path.
`COMMAND`	`TimingEvent*`. Timing event corresponding to the command of the timing path terminating on a register. In the case of a flip-flop this event is the triggering edge, in the case of a latch it is the event which opens the latch for writing.
`ACCESS_COMMAND`	`TimingEvent*`. Timing event corresponding to the command of the latch or precharge crossed by the timing path when the path is an access. The result is the event which opens the latch for writing.

ACCESS_LATCH	TimingEvent*. Timing event corresponding to the latch or precharge crossed by the timing path when the path is an access.
START_TRAN	char. Transition direction on the starting node of the timing path. The value can be u (rising) or d (falling).
END_TRAN	char. Transition direction on the ending node of the timing path. The value can be u (rising) or d (falling).
TYPE	char*. Returns either min or max.
IS_HZ	char*. Returns "yes" if the timing path is a HZ path. "no" otherwise.
PATH_MARGIN_FACTOR	double. Path margin factor defined for this timing path.
PATH_MARGIN_DELTA	double. Path margin delta defined for this timing path.

Properties described above are available through function:

```
ttv_GetTimingPathProperty <timing_path> <code>
```

5.1.6. TimingLine

Property codes:

`DELAY_MAX`	`double`. Max delay attached to the timing line. If a crosstalk analysis has been run on the timing figure, this delay take into account the crosstalk effect.
`REF_DELAY_MAX`	`double`. Max reference delay attached to the timing path. Crosstalk effect is not taken into account.
`DELAY_MIN`	`double`. Min delay attached to the timing line. If a crosstalk analysis has been run on the timing figure, this delay take into account the crosstalk effect.
`REF_DELAY_MIN`	`double`. Min reference delay attached to the timing path. Crosstalk effect is not taken into account.
`SLOPE_MAX`	`double`. Max slope attached to the timing line. If a crosstalk analysis has been run on the timing figure, this slope take into account the crosstalk effect.
`REF_SLOPE_MAX`	`double`. Max reference slope attached to the timing path. Crosstalk effect is not taken into account.
`SLOPE_MIN`	`double`. Min slope attached to the timing line. If a crosstalk analysis has been run on the timing figure, this slope take into account the crosstalk effect.
`REF_SLOPE_MIN`	`double`. Min reference slope attached to the timing path. Crosstalk effect is not taken into account.
`TYPE`	`char`. Line type: setup, hold, access, hz, rc, prech, eval or data.
`START_EVENT`	`TimingEvent*`. Start event of the timing line.
`END_EVENT`	`TimingEvent*`. End event of the timing line.
`COMMAND`	`TimingEvent*`. Command event of the timing line. Returns `NULL` if the line is not commanded.

Properties described above are available through function:

```
ttv_GetTimingLineProperty <timing_line> <code>
```

5.1.7. TimingDetail

Property codes:

`HZ`	`int`. Indicates if the detail is an HZ state. If true, the signal direction indicates the state of the output before being HZ. 0 if false, 1 if true.
`EVENT`	`TimingEvent*`. Pointer to the related TimingEvent
`NODE_NAME`	`char*`. The name of the node in timing detail
`NET_NAME`	`char*`. The name of the net in timing detail
`SIGNAL_TYPE`	`char*`. Type of the timing signal. Possible values are `connector`, `command`, `latch`, `flip-flop`, `precharge`, `breakpoint`, `internal`, `signal` or an eventual concatenation of possible configurations.
`DELAY`	`double`. The delay (in seconds) of the path detail. If a crosstalk analysis has been run on the timing figure, this delay take into account the crosstalk effect.
`SLOPE`	`double`. The slope (in seconds) of the path detail. If a crosstalk analysis has been run on the timing figure, this delay take into account the crosstalk effect.
`REF_DELAY`	`double`. The delay (in seconds) of the path detail
`REF_SLOPE`	`double`. The slope (in seconds) of the path detail
`SIM_DELAY`	`double`. The simulated delay (in seconds) of the path detail. `-1` if no simulation has been done or an error occurred.
`SIM_SLOPE`	`double`. The simulated slope (in seconds) of the path detail. `-1` if no simulation has been done or an error occurred.
`DATA_LAG`	`double`. Only for clock-to-q access paths: waiting delay for the data to be available on the latch's input, after the latch opening (value is present only after stability analysis).
`TYPE`	`char*`. A string identifying whether the path detail is a gate or RC delay.
`TRANS`	`char`. The transition direction of the node. `u` (rising) or `d` (falling).

Properties described above are available through function:

```
ttv_GetTimingDetailProperty <timing_detail> <code>
```

5.1.8. StabilityFigure

The StabilityFigure object is a back-annotation of the TimingFigure object. It contains the switching windows associated to the TimingSignal objects of the TimingFigure object. The switching window information of the StabilityFigure object is obtained by functions accessing directly the TimingSignal objects.

Property codes:

`TIMING_FIGURE`	`TimingFigure*`. The TimingFigure the StabilityFigure has be created from.

Properties described above are available through function:

```
stb_GetStabilityFigureProperty <slack_object> <code>
```

5.1.9. StabilitySlack

Property codes:

`TYPE`	`char*`. Slack type: `setup` or `hold`.
`VALUE`	`double`. Slack value
`DATA_VALID`	`double`. Arrival time of the DATA VALID event.
`DATA_REQUIRED`	`double`. Arrival time of the DATA REQUIRED event.
`DATA_VALID_PATH`	`TimingPath*`. Path related to the DATA VALID event.
`DATA_REQUIRED_PATH`	`TimingPath*`. Path related to the DATA REQUIRED event. Can return `NULL` if the data required time is a user specified one set up with SDC command `set_output_delay`.
`START_EVENT`	`TimingEvent*`. Starting event of the DATA VALID path.
`END_EVENT`	`TimingEvent*`. Ending event of the DATA VALID path.
`THRU_EVENT`	`TimingEvent*`. In case DATA VALID is an access, the event of the access latch.
`THRU_COMMAND_EVENT`	`TimingEvent*`. In case DATA VALID is an access, the command event enabling the data thru the latch.
`IS_HZ`	`char*`. Returns "yes" if the DATA VALID path is a HZ path. "no" otherwise.
`DATA_VALID_PATH_MARGIN`	`double`. Computed margin to be added to the DATA VALID path delay.

DATA_REQUIRED_PATH_MARGIN	double. Computed margin to be added to the DATA REQUIRED path delay.
INTRINSIC_CONSTRAINT	double. Setup or hold constraint at the DATA VALID node.
SKEW_COMPENSATION	double. Skew compensation value to be added to the DATA REQUIRED path delay. Skew compensation is deduced from common clock tree between DATA REQUIRED and DATA VALID paths.
UNCERTAINTY	double. Clock uncertainty to be added to the DATA REQUIRED path delay. Clock uncertainty is a user given value.

Properties described above are available through function:

```
stb_GetSlackProperty <slack_object> <code>
```

5.1.10. TimingConstraint

Property codes:

TYPE	char*. Constraint type: setup or hold.
VALUE	double. Constraint value.
DATA_PATH	TimingPath*. Path related to the data of the constraint. If the constraint is a directive, the data path can be a clock path if specified in the directive.
CLOCK_PATH	TimingPath*. Path related to the clock of the constraint. If the constraint is a directive, the clock path can be a data path if specified in the directive.
INTRINSIC_MARGIN	double. Intrinsic setup or hold margin for a constraint on a latch node or user defined margin in case of a directive.
INTRINSIC_MARGIN_MODEL	char*. Intrinsic setup or hold model name (most likely a .lib table model name). Can return NULL if not no model is associated with the intrisic margin.
DATA_PATH_MARGIN	double. Computed margin to be added to the DATA_PATH delay.
CLOCK_PATH_MARGIN	double. Computed margin to be added to the CLOCK_PATH delay.
MASTER_CLOCK	TimingEvent*. TimingEvent of the clock at the interface of the netlist if the CLOCK_PATH comes from a generated clock.

	If it is not the case, the `MASTER_CLOCK` is the starting event of the `CLOCK_PATH`.
`MASTER_CLOCK_LATENCY`	`double`. Value of the delay from the master clock to the generated clock.

Properties described above are available through function:

```
ttv_GetTimingConstraintProperty <constraint_object> <code>
```

5.1.11. BehavioralFigure

The BehavioralFigure structure describes the functionality of a circuit.

5.1.12. Node Crosstalk Statistics

Properties:

`SCORE_NOISE`	The noise score.
`SCORE_ACTIVITY`	The activity score.
`SCORE_CTK`	The crosstalk score.
`SCORE_INTERVAL`	The interval score.
`SCORE`	The total score.
`VOLTAGE_THRESHOLD`	The worst gate threshold among all of the fanout gates.
`NOISE_RISE_MAX`	The maximum rise peak, considering active all aggressors
`NOISE_FALL_MAX`	The maximum fall peak, considering active all aggressors
`NOISE_RISE_REAL`	The real rise peak, considering active overlapping aggressors only
`NOISE_FALL_REAL`	The real fall peak, considering active overlapping aggressors only
`CAPA_GROUND`	The ground interconnect capacitance.
`CAPA_CTK`	The crosstalk interconnect capacitance.
`CAPA_CGATE`	The load of all gate input driver.

Properties described above are available through function:

```
ctk_GetStatNodeProperty <stability_figure> <index> <property>
```

5.1.13. Delay Crosstalk Statistics

Properties.

`START_NODE`	The starting event of delay.
`END_NODE`	The ending event of delay.
`DELAY_MIN_STA`	The minimum delay without crosstalk.
`DELAY_MAX_STA`	The maximum delay without crosstalk.
`SLOPE_MIN_STA`	The minimum slope without crosstalk.
`SLOPE_MAX_STA`	The maximum slope without crosstalk.
`DELAY_MIN_CTK`	The minimum delay with crosstalk.
`DELAY_MAX_CTK`	The maximum delay with crosstalk.
`SLOPE_MIN_CTK`	The minimum slope with crosstalk.
`SLOPE_MAX_CTK`	The maximum slope with crosstalk.

Properties described above are available through function:

```
ctk_GetStatLineProperty <stability_figure> <index> <property>
```

5.1.14. Crosstalk Aggressor

Properties.

`SIGNAL`	The timing signal (optionnal).
`NETNAME`	The netname.
`CAPA_CTK`	The coupling capacitance.
`DELAYBESTAGR`	Return "no", "yes" or "excluded" if this aggressor is quiet, active or if it has not been taken into account due to switching mutual exclusion.
`DELAYWORSTAGR`	idem.
`NOISERISE`	idem.
`NOISEFALL`	idem.

Properties described above are available through function:

```
ctk_GetAggressorProperty <stability_figure> <property>
```

5.2. Units

5.2.1. Input Values

All values given as inputs to Tcl functions of the HITAS platform need to be specified in the International Units System.

Values given as inputs to SDC functions are not given within the International Units System. Units are controlled by the variable `sdcUnits`

5.2.2. Returned Values

All values returned by Tcl functions are given in the in the International Units System, except for timing reports, which are controlled by the `ttv_SetupReport` function.

5.3. General

5.3.1. Configuration

avt_Config

```
void avt_Config(char *var, char *val)
```

Main way to configure the tool. Affects a value to one of the variables listed in the Configuration Variables section

`var`	Configuration variable to be set
`val`	New value
EXAMPLE	`avt_Config tasGenerateConeFile yes`

---

avt_GetConfig

```
char *avt_GetConfig(char *var)
```

returns the configurated value for configuration variable `var`

`var`	Configuration variable to be set
EXAMPLE	`set cone_cfg [avt_GetConfig tasGenerateConeFile]`

5.3.2. File Loading

avt_SetBlackBoxes

```
void avt_SetBlackBoxes(List *list)
```

Allows the user to blackbox subcircuits. Blackboxed subcircuits will not be analyzed. Instead, the tool will let a hole. Whether this hole should be filled up or not by a timing description depends on configuration variables `tasIgnoreBlackbox` *and* `tasTreatBlackboxHierarchically`. *If a blackbox name is prefixed with "unused:", no hole will be created but instead all transistors in the blackbox will be marked as unused. Those blackboxes can still be retreived with GNS if the recognition rule uses the same transistor names as in the blackbox. This command is equivalent to and overrides the creation of a* `BLACKBOX` *file.*

`list`	List of subcircuits to be blackboxed. All intended blackboxed subcircuits should present as only one `avt_SetBlackBoxes` command is allowed.
EXAMPLE	`avt_SetBlackBoxes [list "sense_amp"]`

---

avt_LoadBehavior

```
BehavioralFigure *avt_LoadBehavior(char *filename, char *format)
```

Loads behavioral descriptions and construct internal representation according to the file format

`filename`	File to be loaded
`format`	Available formats are `vhdl` and `verilog`
EXAMPLE	`avt_LoadFile model.v verilog`

---

avt_DriveBehavior

```
void avt_DriveBehavior(BehavioralFigure *befig, char *format)
```

Drives a behavioral description according to the file format from the given internal representation

`befig`	Behavior to be driven
`format`	Available formats are `vhdl` and `verilog`
EXAMPLE	`avt_DriveBehavior $befig output.v verilog`

---

avt_LoadFile

```
void avt_LoadFile(char *filename, char *format)
```

Loads files and construct internal representation according to the file format

`filename`	File to be loaded
`format`	Available formats are `spice`, `tlf4`, `tlf3`, `lib`, `verilog`, `vhdl`, `spf`, `dspf`, `inf`, `spef` and `ttv`
EXAMPLE	`avt_LoadFile design.hsp spice`

avt_EncryptSpice

```
void avt_EncryptSpice(char *inputname, char *outputname)
```

Encrypts all sections of a Spice file (netlist or technology file) which are encapsulated by the `.protect` *and* `.unprotect` *spice cards.*

`inputname`	File to be encrypted
`outputname`	Destination for encrypted output
EXAMPLE	`avt_EncryptSpice techno.hsp techno.hsp.enc`

avt_SetCatalog

```
void avt_SetCatalog(List *argv)
```

Sets the leaves when flattening a netlist to catal level; equivalent to create a CATAL file

`argv`	List of subcircuits that will be used as leaves
EXAMPLE	`avt_SetCatalog [list "nand2" "inv" ]`

avt_GetCatalog

```
StringList *avt_GetCatalog()
```

Returns the current list of cells set as leaves for a catal-level flatten

EXAMPLE	`set catal [avt_GetCatalog]`

avt_CheckTechno

```
void avt_CheckTechno(char *label, char *tn, char *tp)
```

Runs a set of benchs to findout possible technology errors

`label`	A prefix label for the output result files
`tn`	NMOS transistor characteristics. It's a space separated string with coming first the NMOS transistor name followed by the parameters. Authorized parameters are: l, w, delvt0, mulu0, sa, sb, sd, nf, nrs, nrd, sc, sca, scb, scc.
`tp`	same as `tn` for PMOS transistor.
`EXAMPLE`	avt_CheckTechno check1 "nmos l=0.4u w=0.8u" "pmos l=0.4u w=1.6u"

5.3.3. Netlist Modification

avt_GetNetlist

```
Netlist *avt_GetNetlist(char *name)
```

Retrieves a netlist from memory and returns its pointer

`name`	Name of the netlist to get in the program's memory
EXAMPLE	`set netlist [avt_GetNetlist "my_design"]`

---

avt_FlattenNetlist

```
void avt_FlattenNetlist(Netlist *lf, char *level)
```

Flattens a netlist to a given level.

`lf`	Pointer on the netlist to be flattened
`level`	Hierarchical level (coming from top-level) the nelist will be flattened to. Available levels are `trs`, `catal` or `bbox` (transistor, catalog or blackbox). If none of those levels are used, `level` will be considered an instance name, to which the netlist will be flattened.
EXAMPLE	`avt_FlattenNetlist $netlist trs`

---

avt_DriveNetlist

```
void avt_DriveNetlist(Netlist *lf, char *filename, char *format)
```

Saves the netlist on disk according to the given format

`lf`	Pointer on the netlist to be saved
`filename`	Name of the file to be created
`format`	Available formats are `spice`, `verilog`, `vhdl` and `spef`
EXAMPLE	`avt_DriveNetlist $netlist design.spi spice`

---

avt_DisplayNetlistHierarchy

```
void avt_DisplayNetlistHierarchy(FILE *f, char *netlistname, int
maxdepth)
```

Displays hierarchy information of a given netlist, and other info such as number of transistors

`f`	Pointer on the file where to save information, for standard output set `stdout`
`netlistname`	Pointer on the netlist
`maxdepth`	Maximum hierarchical depth coming from top level; can be set to 0 for infinite depth
`EXAMPLE`	`avt_DisplayNetlistHierarchy stdout "my_design" 3`

avt_DisplayResistivePath

```
void avt_DisplayResistivePath(FILE *f, Netlist *lf, char *connector1,
char *connector2)
```

Displays one resistive path between two connectors at the interface of a netlist.

`f`	Pointer on the file where to save information, for standard output set `stdout`
`lf`	Pointer on the netlist
`connector1`	first connector name
`connector2`	second connector name
`EXAMPLE`	`avt_DisplayResistivePath stdout [avt_GetNetlist "mynetlistname"] vdd_0 vdd_1`

avt_RemoveResistances

```
void avt_RemoveResistances(Netlist *lf, char *nameregex)
```

Removes all resistances on signals matching a regular expression

`lf`	Pointer on the netlist where to remove resistances
`nameregex`	Regular expression to be matched, for all signals use `*`
`EXAMPLE`	`avt_RemoveResistances $netlist "cpu.*.sig3*"`

avt_RemoveCapacitances

```
void avt_RemoveCapacitances(Netlist *lf, char *nameregex)
```

Removes all capacitances on signals matching a regular expression

`lf`	Pointer on the netlist where to remove capacitances
`nameregx`	Regular expression to be matched, for all signals use `*`
EXAMPLE	`avt_RemoveCapacitances $netlist "cpu.*.sig3*"`

5.3.4. Statistics

avt_StartWatch

```
void avt_StartWatch(char *name)
```

Starts a timer; if the timer already exixts it'll be reset to 0.

name	Timer name
EXAMPLE	`avt_StartWatch "CPU_TIME"`

---

avt_StopWatch

```
void avt_StopWatch(char *name)
```

Stops a timer; the timer must be started for the function to work

name	Name of the timer to stop
EXAMPLE	`avt_StopWatch "CPU_TIME"`

---

avt_PrintWatch

```
char *avt_PrintWatch(char *name)
```

Returns a string with the value of a timer; the timer must have been started

name	Name of the timer to print
EXAMPLE	`avt_PrintWatch "CPU_TIME"`

---

avt_GetMemoryUsage

```
long avt_GetMemoryUsage()
```

Returns an integer with the memory usage of the program in bytes

EXAMPLE	`set memory [avt_GetMemoryUsage]`

---

avt_RegexIsMatching

```
int avt_RegexIsMatching(char *nametocheck, char *template)
```

	Returs 1 if `nametocheck` *matches the regular expression* `template`*, 0 otherwise.*
`nametocheck`	name to check.
`template`	regular expression to use.
EXAMPLE	`set match [avt_RegexIsMatching tatoo5 *too*]`

5.4. INF Configuration

Signal naming: only disassembly directives accept all hierarchical names as inputs. Other directives take as input the hierarchical name of the highest level.

5.4.1. General

inf_SetFigureName

```
void inf_SetFigureName(char *name)
```

Sets the target figure on which to apply the INF functions

`name`	Name of the target figure
`EXAMPLE`	`inf_SetFigureName cpu`

---

inf_AddFile

```
void inf_AddFile(char *filename, char *figname)
```

Loads an INF file and applies included statements on a figure (this function does not invoke `inf_SetFigureName`*).*

`filename`	INF file to load
`figname`	Figure on which to apply INF statements. Those statements will be added to the ones that my be already present.
`EXAMPLE`	`inf_AddFile cpu.inf cpu`

---

inf_Drive

```
void inf_Drive(char *outputname)
```

Saves applied INF statements on disk

`outputname`	File where to save INF statements (the `.inf` suffix is not automatically added)
`EXAMPLE`	`inf_Drive cpu.inf`

---

inf_ExportSections

```
void inf_ExportSections(char *outputname, char *section)
```

Saves on disk applied INF statements related to specific INF sections

`outputname`	File where to save INF statements
`section`	`OperatingCondition`, `PinSlew`, `Rename`, `Stop`, `Sensitive`, `Suppress`, `Inputs`, `NotLatch`, `CkLatch`, `Ckprech`, `Precharge`, `Dirout`, `Mutex`, `CrosstalkMutex`, `Constraint`, `ConnectorDirections`, `PathIN`, `PathOUT`, `PathDelayMargin`, `MulticyclePath`, `Ignore`, `NoCheck`, `Bypass`, `NoRising`, `NoFalling`, `Break`, `Inter`, `Asynchron`, `DoNotCross`, `Transparent`, `RC`, `NORC`, `SIGLIST`, `Falsepath`, `Delay`, `Dlatch`, `FlipFlop`, `Slopein`, `Capaout`, `OutputCapacitance`, `SwitchingProbability`, `Directives`, `Stb` and `Stuck`.
`EXAMPLE`	`inf_ExportSections cpu.inf "Dirout CrosstalkMutex"`

inf_CleanFigure

```
void inf_CleanFigure()
```

Removes all INF statements on current figure

5.4.2. Disassembly Directives

inf_DefineIgnore

```
void inf_DefineIgnore(char *type, List *list)
```

The tool ignores specified components. Equivalent to commenting out elements in a SPICE netlist.

`list` Pointer on the list of components to ignore. An component name can be a regular expression.

`type` Supported types are `Instances`, `Transistors`, `Resistances`, `Capacitances`, `Diodes`, `Parasitics` and `SignalNames`. `Parasitics` affects only DSPF files. `SignalNames` affects only the flattening of a hierarchical netlist, by ignoring the given name if several hierarchical names are available for one net.

EXAMPLE
```
inf_DefineIgnore Transistors *.M23*
```

---

inf_DefineMutex

```
void inf_DefineMutex(char *type, List *list)
```

Adds mutual exclusion constraints on signals, in order to help the disassembly process. May be especially usefull when dealing with shifters or multiplexors, in case mutual exclusion constraints can not be directly derived from internal combinational circuitry (if the mutual exclusions constraints come from latched values or come from constraints on external pins).

`type` Mutual exclusion constraints, legal values for are `muxup`, `muxdn`, `cmpup` and `cmpdn` (see INF file description)

`list` List of signals mutual exclusions constraints should be applied on

EXAMPLE
```
inf_DefineMutex cmpup [list a_0 a_1 a_2 a_3]
```

---

inf_DefineInputs

```
void inf_DefineInputs(char *name)
```

Sets a signal as a circuit input, in order to help the disassembly process.

`name` Signal's name

inf_DefineDirout

```
void inf_DefineDirout(char *name, int level)
```

Defines the level of a signal for transistor orientation, in order to help the disassembly process.

`name` Signal's name

`level` Signal's level; transistors are oriented (the sense of the current is) from high-level to low-level signals.

inf_DefineDLatch

```
void inf_DefineDLatch(char *name)
```

Sets a signal as a dynamic latch. Works only if the surrounding circuitry permits a HZ state on the signal. Commands are then identified automatically.

`name` Signal's name

inf_DefineNotDLatch

```
void inf_DefineNotDLatch(char *name)
```

Disables a dynamic latch directive on a signal. To be used together with `yagMarkTristateMemory`

`name` Signal's name

inf_DefineNotLatch

```
void inf_DefineNotLatch(char *name)
```

Disables the identification of a latch on a signal

`name` Signal's name

inf_DefineKeepTristateBehaviour

```
void inf_DefineKeepTristateBehaviour(char *name)
```

Disables the transformation of bus into register when configurations 'avtVerilogTristateIsMemory' or 'yagleTristateIsMemory' is used to drive a behavioural model.

`name` Signal's name

---

inf_DefinePrecharge

```
void inf_DefinePrecharge(char *name)
```

Sets a signal as a precharge.

`name` Signal's name

---

inf_DefineNotPrecharge

```
void inf_DefineNotPrecharge(char *name)
```

Disables the identification of a precharge on a signal

`name` Signal's name

---

inf_DefineModelLatchLoop

```
void inf_DefineModelLatchLoop(char *name)
```

Feedback loop is explicitly modeled in behavioural model if signal is a static latch.

`name` Signal's name

---

inf_DefineMemsym

```
void inf_DefineMemsym(char *name0, char *name1)
```

Sets a pair of signals to be a symmetric memory so long as there is a loop between the two signals.

`name0` name of first memsym signal.

`name1` name of second memsym signal.

`EXAMPLE` `inf_DefineMemsym memsym0 memsym1`

---

inf_DefineRS

```
void inf_DefineRS(char *name, char *type)
```

Allows control of how individual RS are handled. Overrides the global setting in yagAutomaticRSDetection.

`name`	Signal's name, either the set or the reset one is enough.
`type`	LEGAL, ILLEGAL or MARK_ONLY.
`EXAMPLE`	`inf_DefineRS rsnode "LEGAL"`

---

inf_MarkSignal

```
void inf_MarkSignal(char *name, char *marks)
```

Allows application of special signal markings, such as latch identification.

`name`	Signal's name
`marks`	For a complete list of markings please refer to the INF section of this manual, MARKSIG subsection.
`EXAMPLE`	`inf_MarkSignal dff_m "LATCH+MASTER"`

---

inf_MarkTransistor

```
void inf_MarkTransistor(char *name, char *marks)
```

Allows application of special transistor markings, such as latch commands identification.

`name`	Signal's name
`marks`	Legal markings are "Bleeder", "Feedback", "Command", "NonFunctional", "Blocker", "Short", "Unused". Types may be concatenated with the '+' character and are case-insensitive. For a description of the types please refer to the INF section of this manual, MARKTRANS subsection.
`EXAMPLE`	`inf_MarkTrans m0 "FEEDBACK+NOT_FUNCTIONAL"`

5.4.3. Timing Directives

inf_DefineConnectorSwing

```
void inf_DefineConnectorSwing(char *name, double lowlevel, double
highlevel)
```

Sets the switching voltage magnitude for a connector. This is usefull for multivoltage analysis.

`name`	Connector's name
`lowlevel`	Lower-bound voltage level, in Volts
`highlevel`	Upper-bound voltage level, in Volts

---

inf_DefinePathDelayMargin

```
void inf_DefinePathDelayMargin(char *type, char *name, double factor,
double delta, char *pathtype)
```

Defines a derating to be applied on a path delay, with the following formula: `new_delay = real_delay * factor + delta`. *This derating is only used for the computation of setup and hold slacks by the by slack report functions.*

`type`	Arrival point type of the path on which to apply the derating. Valid values are `any`, `latch`, `break`, `prech` and `con` (connector)
`name`	Arrival point name of the path on which to apply the derating. Wildcards can be used.
`factor`	Integer value
`delta`	Integer value
`pathtype`	String with tokens separated by spaces or _. Valid token values for pathtype are `clockpath`, `datapath`, `min`, `max`, `rise` and `fall`. An empty string means that all the tokens are used.

---

inf_DefineConnectorDirections

```
void inf_DefineConnectorDirections(char *type, List *list)
```

Modifies the direction of connectors

`type`	Affected direction, can be `Input`, `Output`, `InOut`, `Tristate`, `HZ` or `Unknown`
`list`	List of signals a new direction will be affected to

inf_DefineNORC

```
void inf_DefineNORC(char *name)
```

Disables RC delay computation on a signal

`name`	Signal's name

inf_DefineDoNotCross

```
void inf_DefineDoNotCross(char *name)
```

Disables the transparency of a latch or precharge, so that no path can traverse it. Effective only when static timing analysis has been run (`stb`*). To be used together with* `avtMaxPathPeriodDepth`*.*

`name`	Signal's name

inf_DefineTransparent

```
void inf_DefineTransparent(char *name)
```

Force transparency of a latch or precharge, so that path can traverse it.

`name`	Signal's name

inf_DisableTimingArc

```
void inf_DisableTimingArc(char *input, char *output, char *direction)
```

Disables the construction of timing arcs between timing signals.

`input`	Source timing signal of the timing arc to disable
`output`	Sink timing signal node of the timing arc to disable
`direction`	Transition to disable: `du`, `ud`, `dd`, `uu`, `?u`...
`EXAMPLE`	`inf_DisableTimingArc in out ud`

inf_DefineFalsePath

```
void inf_DefineFalsePath(List *arglist)
```

Defines a false path. The parameters use the same syntax as in the .inf file except for the signal direction specification which does not allow spaces. Valid values `<up>` *and* `<down>`.

EXAMPLE `inf_DefineFalsePath [list sig1 <up> sig2 <down> sig3 sig4].`

5.4.4. Timing Abstraction Directives

inf_DefineSlopeRange

```
void inf_DefineSlopeRange(char *name, List *argv, char *type)
```

Sets a range for input slopes in lookup tables construction (.lib file purpose), values in Seconds. This function must be called before database construction (`hitas` *function) for ranges to be taken into account.*

`name`	Connector the defined range will be applied on, `default` is for all connectors.
`type`	`custom` or `linear`.
`argv`	If `type` is `custom`, list of values defining the slope range. If `type` is `linear`, 3-uple `(first_slope, last_slope, step)`.
EXAMPLE	`inf_DefineSlopeRange default [list 50e-12 120e-12 240e-12] custom`

---

inf_DefineCapacitanceRange

```
void inf_DefineCapacitanceRange(char *name, List *argv, char *type)
```

Sets a range for ouput capacitances in lookup tables construction (.lib file purpose), values in Fahrads. This function must be called before database construction (`hitas` *function) for ranges to be taken into account.*

`name`	Connector the defined range will be applied on, `default` is for all connectors.
`type`	`custom` or `linear`.
`argv`	If `type` is `custom`, list of values defining the capacitance range. If `type` is `linear`, 3-uple `(first_capa, last_capa, step)`.
EXAMPLE	`inf_DefineCapacitanceRange default [list 100e-15 200e-15 300e-15] custom`

5.4.5. STA Directives

inf_DefineStrictSetup

```
void inf_DefineStrictSetup(char *name)
```

Defines an latch whose setup must be check on the command opening event rather than the closing event.

`name`	Name of the latch
`EXAMPLE`	`inf_DefineStrictSetup mylatch`

inf_DefineAsynchron

```
void inf_DefineAsynchron(char *name)
```

Defines an asynchronous pin, i.e. reset pin. Necessary to obtain recovery/ removal timing groups in .lib characterization.

`name`	Name of the reset pin
`EXAMPLE`	`inf_DefineAsynchron reset`

inf_DefineEquivalentClockGroup

```
void inf_DefineEquivalentClockGroup(char *domain, List *list)
```

Sets clocks as belonging to the same timing domain.

`domain`	Timing domain name
`list`	Set of clocks belonging to the domain
`EXAMPLE`	`inf_DefineEquivalentClockGroup domain1 [list ck1 ck2]`

inf_DefineClockPriority

```
void inf_DefineClockPriority(char *name, char *clock)
```

Defines the preferred clock for a signal in case of multiple clock possibility.

`name`	Signal's name
`clock`	Priority clock

inf_DefineDirective

```
inf_DefineDirective <mode> [type1] <signal1> [dir1] <operation> [type2]
<signal2> [dir2] [margin]
```

Adds a custom timing check between any two nodes. This timing check will be taken into account in STA and slack reports.

`mode` Type of operation: `check` or `filter`.

`type1` Type of propagated information on `signal1` to check: `clock` or `data`. Default is `data`.

`signal1` Net name of the first signal.

`dir1` Edge or value of `signal1` to consider: `rising` or `falling`. Default is both.

`operation` `before`, `after` or `with`. Operation `with` checks that `signal1` is stable when `signal2`'s state is up or down. `with` cannot be used with `signal1` as a clock or `signal2` as a data, and `dir2` as falling or rising. Operations `before` and `after` compare arrival times between clock/data and clock/data. `dir1` and `dir2` can be `rising` or `falling` only.

`type2` Type of propagated information on `signal2` to check: `clock` or `data`. Default is `data`.

`signal2` Net name of the second signal.

`dir2` Edge or value of `signal2` to consider: `rising`, `falling`, `up` or `down`. Default is rising and falling.

`margin` Margin to add when computing slack or constraint.

`EXAMPLE` `inf_DefineDirective check "data" with clock "data" up`. Checks that "data" is stable when the propagated clock state on data is high.

---

inf_DefineFalseSlack

```
inf_DefineFalseSlack [<restriction>:] <startclock> [<startclock_dir>]
<startnode> [<startnode_dir>] <endnode> [<endnode_dir>] [<endnode_hz_state>]
<endclock>
```

Defines a slack as being invalid.

`restriction` a '-' separated list of checks/signal types. Possible values: `setup`, `hold`, `latch`, `prech`. `latch` and `prech` define `endnode` signal type. Default is "setup-hold".

`startclock`	name of the clock generating the `startnode` data
`startclock_dir`	transition of `startclock` : <up>, <down>, <rise> or <fall>. Default is both.
`startnode`	name of the generated data node
`startnode_dir`	transition of `startnode` : <up>, <down>, <rise> or <fall>. Default is both.
`endnode`	name of the data arrival node
`endnode_dir`	transition of `endnodestartnode` : <up>, <down>, <rise> or <fall>. Default is both.
`endnode_hz_state`	specify whether the end node transition goes to hz state or not: <hz> or <nothz>. Default is both.
`endclock`	name of the clock controlling the arrival node
`EXAMPLE`	`inf_DefineFalseSlack setup: "ck" <rise> "data" <rise> "arrival" "ck2"`.

inf_DefineSwitchingProbability

`inf_DefineSwitchingProbability signal switching probability`

[IN ALPHA DEVELOPMENT STAGE] Associates a switching probability to `signal`. *This probability can be used by CTK to remove non interesting agressions (see also variable: stbCtkMinOccurenceProbability).*

`signal`	signal to associate the probability to.
`switching probability`	probability value from 0 to 1.
`EXAMPLE`	`inf_DefineSwitchingProbability "enable" 0.25`. Signal "enable" can switch 1 time each 4 clock cycles.

5.4.6. Crosstalk Directives

inf_DefineCrosstalkMutex

```
void inf_DefineCrosstalkMutex(char *type, List *list)
```

Sets mutually exclusive signals with regard to crosstalk analysis.

`type`	Legal values are `muxup` and `muxdn`.
`list`	List of signals on which to apply mutual exclusion constraints

5.5. SDC Support

Timing constraints can also be defined with SDC commands. Use the `sdcUnits` configuration variable to define the time unit for all commands. Only the commands that follow are supported as API functions.

5.5.1. Object Access Commands

`all_clocks`	Applies a command to all clocks.
`all_inputs`	Applies a command to all inputs. No options are supported.
`all_outputs`	Applies a command to all outputs. No options are supported.
`get_clocks`	Returns only the patterns given in argument. The name is verified to be a clock name by the calling command.
`get_pins`	Returns only the patterns given in argument.
`get_ports`	Returns only the patterns given in argument.

5.5.2. set_case_analysis

SYNTAX

```
set_case_analysis 0 | zero | 1 | one
                  | rise | rising | fall | falling
                  signal_list
```

ARGUMENTS

`0 | zero`	Sets logical constraint 0 on specified signals (connectors or internal signals)
`1 | one`	Sets logical constraint 1 on specified signals (connectors or internal signals)
`rise | rising`	Only rising transitions are propagated on specified signals (connectors or internal signals)
`fall | falling`	Only falling transitions are propagated on specified signals (connectors or internal signals)

INF EQUIVALENT

`set_case_analysis 0 | 1` translates to the `CONSTRAINT` section

`set_case_analysis rise | fall` translates to the `NORISING` and `NOFALLING` sections

5.5.3. set_disable_timing

SYNTAX

```
set_disable_timing signal_list
```

INF EQUIVALENT

`set_disable_timing` translates to the `BYPASS` section

5.5.4. set_false_path

SYNTAX

```
set_false_path [-setup | -hold] [-rise] [-fall]
               [-from fr_list]
               [-through through_list]
               [-to to_list]
```

ARGUMENTS

`-rise`	Restricts false paths to those ending on a rising edge
`-fall`	Restricts false paths to those ending on a falling edge
`-from`	Specifies a list of false paths starting points
`-through`	Specifies a list of intermediary points the false paths must go through. Multiple groups of intermediary points can be specified using multiple `-through` options
`-to`	Specifies a list of false paths ending points
`-setup`	Disables setup verifications on signals specified in the `to_list`. The `-setup` option is ignored if `to_list` is not present. `-rise` and `-fall` options do not apply on setup verification disabling. `from_list` and `through_list` are not taken into account.
`-hold`	Disables hold verifications on signals specified in the `to_list`. The `-hold` option is ignored if `to_list` is not present. `-rise` and `-fall` options do not apply on hold verification disabling. `from_list` and `through_list` are not taken into account.

INF EQUIVALENT

`set_false_path -from | -through | -to` translates to the `BYPASS` section

`set_false_path -from -through -to` translates to the `FALSEPATH` section

`set_false_path -rise | -fall` translates to the `NORISING` and `NOFALLING` sections

`set_false_path -setup | -hold` translates to the `NOCHECK` section

5.5.5. create_clock

SYNTAX

```
create_clock [-name clock_name] -period period_value
             [-waveform edge_list] connector_list
```

ARGUMENTS

`-name`	Specifies the name of the clock to be created.
`-period`	Sets the clock period.
`-waveform`	Sets the rising and falling edge times of the clock waveform.
`connector_list`	Connectors to apply the waveform on. If no connector is specified, the clock is assumed to be virtual

INF EQUIVALENT

`create_clock` translates to the `CLOCK CONNECTORS` and `ASYNCHRONOUS CLOCK GROUPS` section

5.5.6. create_generated_clock

SYNTAX

```
create_generated_clock [-name clock_name] -source source_connector
                       [-divide_by factor | -multiply_by factor]
                       [-duty_cycle duty_cycle_value]
                       [-invert]
                       [-edges edge_list]
                       [-edge_shift shift_list] connector_list
```

ARGUMENTS

`-name`	Name of the clock to be created.
`-source`	Clock source from which the clock is generated.
`-divide_by`	Divides the clock source frequency by `factor`. `factor` must be a power of 2.
`-multiply_by`	Multiplicates the clock source frequency by `factor`. `factor` must be a power of 2.
`-duty_cycle`	Sets the duty cycle in percents.
`-invert`	Inverts the waveform of the generated clock.
`-edges`	Specifies the number of clock source edges to form the edges of the generated clock.

`-edge_shift`	For each edge specified in the `edge_list`, sets the shift value to be applied to.

INF EQUIVALENT

`create_generated_clock` translates to the `CLOCK CONNECTORS` and `ASYNCHRONOUS CLOCK GROUPS` section

5.5.7. set_clock_latency

SYNTAX

```
set_clock_latency [-source] [-rise] [-fall]
                  [-late] [-early]
                  delay_value clock_list
```

ARGUMENTS

`-source`	Specifies clock source latency. If not set, ideal clock latency is assumed.
`-rise`	Sets the delay for the clock's rising edge.
`-fall`	Sets the delay for the clock's falling edge.
`-late`	The delay affects only the max time of the clock's corresponding edge. Only for source latency.
`-early`	The delay affects only the min time of the clock's corresponding edge. Only for source latency.
`-max`	The delay affects only the max time of the clock's corresponding edge. Only for ideal clock latency.
`-min`	The delay affects only the min time of the clock's corresponding edge. Only for ideal clock latency.

INF EQUIVALENT

`set_clock_latency` translates to the `CLOCK CONNECTORS` and `ASYNCHRONOUS CLOCK GROUPS` section

5.5.8. set_clock_uncertainty

SYNTAX

```
set_clock_uncertainty [-from from_clock] [-to to_clock]
                      [-rise_from rise_from_clock]
                      [-fall_from fall_from_clock]
                      [-rise_to rise_to_clock]
                      [-fall_to fall_to_clock]
                      [-rise] [-fall]
                      [clock_pin_list]
                      [-setup] [-hold] uncertainty
```

DESCRIPTION

Specifies the clock uncertainty (skew characteristics) of specified clock networks. This command can specify either interclock uncertainty or simple uncertainty.

ARGUMENTS

`-from`	Specifies the source clock pins for interclock uncertainty.
`-to`	Specifies the destination clock pins for interclock uncertainty.
`-rise_from`	Same as -from, but uncertainty only applies to rising edge of source clock.
`-fall_from`	Same as -from, but uncertainty only applies to falling edge of source clock.
`-rise_to`	Same as -to, but uncertainty only applies to rising edge of destination clock.
`-fall_to`	Same as -to, but uncertainty only applies to falling edge of destination clock.
`-rise`	Deprecated option. Same as -rise_to.
`-fall`	Deprecated option. Same as -fall_to.
`clock_pin_list`	Specifies the destination clock pins for simple uncertainty.
`-setup`	Indicates that the uncertainty only applies to setup checks.
`-hold`	Indicates that the uncertainty only applies to hold checks.
`uncertainty`	A floating-point number (usually positive) that specifies the uncertainty value.

EXAMPLE

```
set_clock_uncertainty CLK1 100
set_clock_uncertainty -from CLK1 -to CLK2 200
```

5.5.9. set_input_transition

SYNTAX

```
set_input_transition [-rise] [-fall]
                [-max] [-min] transition port_list
```

ARGUMENTS

`-rise`	Sets rising transition only

`-fall`	Sets falling transition only
`-min`	Not supported
`-max`	Not supported
`transition`	Transition value
`port_list`	List of input pins

EXAMPLE

```
set_input_transition -rise 100 di
set_input_transition -fall 120 di
```

INF EQUIVALENT

`set_input_delay` translates to the PINSLEW section

5.5.10. set_load

SYNTAX

```
set_load [-min] [-max] [-substract_pin_load]
                [-pin_load] [-wire_load] value objects
```

DESCRIPTION

Sets the capacitance to a specified value on specified ports and nets. This function should be called before database construction or database loading. In order to operate correctly, this function should be used together with the configuration variable `avtPrecisionLevel` set to `1`.

ARGUMENTS

`-min`	Not supported
`-max`	Not supported
`-substract_pin_load`	Not supported
`-pin_load`	Not supported
`-wire_load`	Not supported
`value`	Capacitance value
`objects`	List of output pins

EXAMPLE

```
avt_config avtPrecisionLevel 1
inf_SetFigureName my_design
set_load 0.31 [get_ports [all_outputs]]
```

```
set fig [ttv_LoadSpecifiedTimingFigure my_design]
set clist [ttv_GetPaths $fig * * uu 5 critic path max]
```

INF EQUIVALENT

`set_load` translates to the OUTPUT CAPACITANCE section

5.5.11. set_input_delay

SYNTAX

```
set_input_delay [-clock clock_name] [-clock_fall]
                [-rise] [-fall] [-max] [-min]
                delay_value connector_list
```

ARGUMENTS

`-clock`	Relative clock.
`-clock_fall`	Reference edge is the falling edge of a clock. Affects the `From` and `After` statements. If not specified reference edge is rising edge.
`-rise`	Delay for input rising edge only.
`-fall`	Delay for input falling edge only.
`-min`	Minimum arrival time on input. Corresponds to the `Unstable` subsection in the `.inf` file.
`-max`	Maximum arrival time on input. Corresponds to the `Stable` subsection in the `.inf` file.

EXAMPLE

```
set_input_delay -clock ck -clock_fall -rise -min 200 di
set_input_delay -clock ck -clock_fall -rise -max 300 di
```

INF EQUIVALENT

`set_input_delay` translates to

```
SPECIFY INPUT CONNECTORS
BEGIN
    "di" Rising From "ck"  Falling:
      Unstable 200 After "ck"  Falling;
      Stable 300 After "ck"  Falling;
END;
```

5.5.12. set_output_delay

SYNTAX

```
set_output_delay [-clock clock_name] [-clock_fall]
```

```
                [-rise] [-fall] [-max] [-min]
                delay_value connector_list
```

ARGUMENTS

`-clock`	Relative clock.
`-clock_fall`	Reference edge is the falling edge of a clock. Affects the `For` and `After` statements. If not specified reference edge is rising edge.
`-rise`	Delay for output rising edge only.
`-fall`	Delay for output falling edge only.
`-min`	Minimum departure time on output. Corresponds to the `Unstable` subsection in the `.inf` file.
`-max`	Maximum departure time on output. Corresponds to the `Stable` subsection in the `.inf` file.

INF EQUIVALENT

`set_input_delay` translates to `VERIFY OUTPUT CONNECTORS` section.

5.5.13. set_multicycle_path

SYNTAX

```
set_multicycle_path [-setup] [-hold]
                    [-rise] [-fall] [-start] [-end]
                    [-from from_list] [-to to_list] path_multiplier
```

ARGUMENTS

`-setup`	Uses `path_multiplier` for setup calculation only.
`-hold`	Uses `path_multiplier` for hold calculation only.
`-rise`	Affects rising path delays only.
`-fall`	Affects falling path delays only.
`-start`	Calculation is relative to the clock's period at path startpoint.
`-end`	Calculation is relative to the clock's period at path endpoint.
`-from`	Specify a list of timing path startpoints. Valid startpoints are latches, input pins, or inout pins.
`-to`	Specify a list of timing path endpoints. Valid endpoints are latches, output pins, or inout pins.

INF EQUIVALENT

`set_multicycle_path` translates to `MULTICYCLE PATH` section.

5.5.14. set_max_delay

SYNTAX

```
set_max_delay [-rise] [-fall]
              [-from from_list]
              [-rise_from from_list] [-fall_from from_list]
              [-to to_list]
              [-rise_to to_list] [-fall_to to_list]
              delay_value
```

DESCRIPTION

Specifies a required maximum delay for the specified timing paths. Appears as a setup check in slack reports.

ARGUMENTS

`-rise`	Only rising paths are constrained.
`-fall`	Only falling paths are constrained.
`-from`	Specifies the timing path source.
`-rise_from`	Same as -from, but constraint only applies to rising edge of source.
`-fall_from`	Same as -from, but constraint only applies to falling edge of source.
`-to`	Specifies the timing path destination.
`-rise_to`	Same as -to, but constraint only applies to rising edge of destination.
`-fall_to`	Same as -to, but constraint only applies to falling edge of destination.
`delay_value`	A floating-point number (usually positive) that specifies the delay value.

EXAMPLE

```
set_max_delay -from CLK1  -to LATCH1 500
```

5.5.15. set_min_delay

SYNTAX

```
set_min_delay [-rise] [-fall]
```

```
[-from from_list]
[-rise_from from_list] [-fall_from from_list]
[-to to_list]
[-rise_to to_list] [-fall_to to_list]
delay_value
```

DESCRIPTION

Specifies a required minimum delay for the specified timing paths. Appears as a hold check in slack reports.

ARGUMENTS

`-rise`	Only rising paths are constrained.
`-fall`	Only falling paths are constrained.
`-from`	Specifies the timing path source.
`-rise_from`	Same as -from, but constraint only applies to rising edge of source.
`-fall_from`	Same as -from, but constraint only applies to falling edge of source.
`-to`	Specifies the timing path destination.
`-rise_to`	Same as -to, but constraint only applies to rising edge of destination.
`-fall_to`	Same as -to, but constraint only applies to falling edge of destination.
`delay_value`	A floating-point number that specifies the delay value.

EXAMPLE

```
set_max_delay -from CLK1  -to LATCH1 500
```

5.6. Timing DB Generation

5.6.1. Automatic Generation

hitas

```
<TimingFigure *> hitas <figname> [-annotatefromcns] [-startfromcns]
```

Generates a timing figure from a pre- or post-layout transistor netlist. It is assumed that an internal representation of the netlist exists in the program's memory, i.e., that the related files (including MOS models) have already been loaded. See `avt_LoadFile` *function.*

`figname`	Name of the subcircuit the timing figure is to be derived
`-annotatefromcns`	[experimental] If a `$filename`.cns file exist, disassembling circuit stage is replaced by annotate the circuit from cns file information.
`-startfromcns`	[experimental] Same as `-annotatefromcns` expect that the netlist saved in the cns file is used, not the user loaded one.
EXAMPLE	`set fig [hitas my_design]`

---

hitas_pvt_count

```
int hitas_pvt_count()
```

Returns the number of PVT errors encountered in the last hitas run.

EXAMPLE	`set pvterrors [hitas_pvt_count]`

---

ttv_LoadSDF

```
TimingFigure *ttv_LoadSDF(Netlist *fig, char *sdf_file)
```

Constructs a timing figure from a netlist and an SDF file. The netlist must have been constructed before (related files loaded)

`fig`	Pointer on a netlist
`sdf_file`	Name of the corresponding SDF file

5.6.2. Manual Generation

ttv_CreateTimingFigure

```
void ttv_CreateTimingFigure(Netlist *lf)
```

Initiates the creation of a timing figure. Computes the connectors capacitances assuming the netlist provided is a flat-transistor one.

`lf` Netlist the timing figure is based on.

EXAMPLE `ttv_CreateTimingFigure $netlist`

---

ttv_EditTimingFigure

```
void ttv_EditTimingFigure(TimingFigure *tvf)
```

Edit an existing timing timing figure. `ttv_FinishTimingFigure` *must be called after the modifications to make the timing figure usable.*

`tvf` TimingFigure to edit.

EXAMPLE `ttv_EditTimingFigure $fig`

---

ttv_AddConnector

```
TimingSignal *ttv_AddConnector(char *name, char dir)
```

Creates a connector timing signal.

`name` Name of the newly created command. If this timing signal already exists, no new timing signal is created. In the last case, incompatible timing signal types will raise a warning and the timing signal type will be overridden

`dir` Direction of the connector: i=in, o=out, t=tristate, z=hz, x=unknown, b=inout (output reused internaly)

EXAMPLE `set con [ttv_AddCommand input i]`

---

ttv_AddCommand

```
TimingSignal *ttv_AddCommand(char *name)
```

Creates a command timing signal.

name	Name of the newly created command. If this timing signal already exists, no new timing signal is created. In the last case, incompatible timing signal types will raise a warning and the timing signal type will be overridden
EXAMPLE	`set com [ttv_AddCommand com0]`

ttv_AddLatch

```
TimingSignal *ttv_AddLatch(char *name)
```

Creates a latch timing signal.

name	Name of the newly created latch. If this timing signal already exists, no new timing signal is created. In the last case, incompatible timing signal types will raise a warning and the timing signal type will be overridden
EXAMPLE	`set latch [ttv_AddLatch lt0]`

ttv_AddPrecharge

```
TimingSignal *ttv_AddPrecharge(char *name)
```

Creates a precharge timing signal.

name	Name of the newly created precharge. If this timing signal already exists, no new timing signal is created. In the last case, incompatible timing signal types will raise a warning and the timing signal type will be overridden

ttv_AddBreakpoint

```
TimingSignal *ttv_AddBreakpoint(char *name)
```

Creates a breakpoint timing signal.

name	Name of the newly created breakpoint. If this timing signal already exists, no new timing signal is created. In the last case, incompatible timing signal types will raise a warning and the timing signal type will be overridden

ttv_AddTiming

```
TimingLine *ttv_AddTiming(char *input, char *output, double max_delay,
double max_slop, double min_delay, double min_slope, char *dir)
```

Creates a data delay line.

`input`	Delay line start point
`output`	Delay line end point
`max_slope`	Maximum slope associated with delay line
`min_slope`	Minimum slope associated with delay line
`max_delay`	Maximum propagation delay associated with delay line
`min_delay`	Minimum propagation delay associated with delay line
`dir`	Transitions on start and end points: `uu` for input rising/output rising, `ud` for input rising/output falling, `dd` for input falling/output falling, `du` for input falling/output rising.

ttv_AddHZTiming

```
TimingLine *ttv_AddHZTiming(char *input, char *output, double max_delay,
double min_delay, char *dir)
```

Creates a HZ data delay line.

`input`	Delay line start point
`output`	Delay line end point
`max_slope`	Maximum slope associated with delay line
`max_delay`	Maximum propagation delay associated with delay line
`dir`	Transitions on start and end points: `uu` for input rising/output rising, `ud` for input rising/output falling, `dd` for input falling/output falling, `du` for input falling/output rising.

ttv_AddSetup

```
TimingLine *ttv_AddSetup(char *input, char *output, double delay, char
*dir)
```

Creates a setup line.

`input`	Data point
`output`	Clock point
`delay`	Constraint associated with the setup line

`dir`	Transitions on data and clock points: `uu` for input rising/output rising, `ud` for input rising/output falling, `dd` for input falling/output falling, `du` for input falling/output rising.

ttv_AddAccess

```
TimingLine *ttv_AddAccess(char *input, char *output, double max_delay,
double max_slop, double min_delay, double min_slope, char *dir)
```

Creates a data access delay line

`input`	Data access line start point
`output`	Data access line end point
`max_slope`	Maximum slope associated with delay line
`min_slope`	Minimum slope associated with delay line
`max_delay`	Maximum propagation delay associated with delay line
`min_delay`	Minimum propagation delay associated with delay line
`dir`	Transitions on start and end timing signals: `uu`, `ud`, `dd` or `du`.

ttv_AddHZAccess

```
TimingLine *ttv_AddHZAccess(char *input, char *output, double max_delay,
double min_delay, char *dir)
```

Creates a HZ data access line.

`input`	Delay line start point
`output`	Delay line end point
`max_slope`	Maximum slope associated with delay line
`max_delay`	Maximum propagation delay associated with delay line
`dir`	Transitions on start and end points: `uu` for input rising/output rising, `ud` for input rising/output falling, `dd` for input falling/output falling, `du` for input falling/output rising.

ttv_AddHold

```
TimingLine *ttv_AddHold(char *input, char *output, double delay, char
*dir)
```

Creates a hold constraint line between a data pin and a clock pin.

`input`	Data pin
`output`	Clock pin
`delay`	Nominal constraint value associated with the line
`dir`	Transitions on the data and clock pins: `uu` for input rising/output rising, `ud` for input rising/output falling, `dd` for input falling/output falling, `du` for input falling/output rising.

---

ttv_SetLineCommand

```
void ttv_SetLineCommand(TimingLine *tl, char *max_command, char
*min_command, char *dir)
```

Associates commands to a line. The command is the signal that enables reading or writing in a latch point.

`max_command`	Command generating the maximum delay
`min_command`	Command generating the minimum delay
`dir`	First character is the edge of the max command and the second of the min command.

---

ttv_SetLineModel

```
void ttv_SetLineModel(TimingLine *tl, char *modelname, char *where)
```

Associates a model name to a line.

`tl`	Timing Line to set the model name
`modelname`	name of the model to associate.
`where`	value to associate model to: "delay max", "delay min", "slope max" or "slope min".

---

ttv_FinishTimingFigure

```
TimingFigure *ttv_FinishTimingFigure()
```

Returns the finished timing figure currently in creation. No more modification will be permitted on the timing figure after this step.

---

ttv_DriveTimingFigure

```
void ttv_DriveTimingFigure(TimingFigure *tvf, char *filename, char
*format)
```

Drives a timing figure on disk

`tvf`	Figure to be driven
`filename`	Full filename
`format`	File format, `dtx` or `ttx`

---

ttv_CreateTimingTableModel

```
void ttv_CreateTimingTableModel(TimingFigure *tvf, char *name,
DoubleListTimeValue *axis1, DoubleListCapaValue *axis2, DoubleTable
*values, char *type)
```

Creates a timing table model using the given axis and table values. For GNS usage, set `tvf` *to NULL*

`tvf`	Timing figure where table model will be created
`name`	Name of the table model
`axis1`	Y-axis values of the table
`axis2`	X-axis values of the table
`values`	Values to set into the table. It's a 2D array.
`type`	defines the type of the table axis: "slope-slope", "slope-capa" or "slope-ckslope".
EXAMPLE	`ttv_CreateTimingTableModel $fig mymodel {1e-12 3e-12} {1e-15 2e-15} {{1e-12 2e-12}{3e-12 4e-12}} "slope-capa"`

---

ttv_CreateEnergyTableModel

```
void ttv_CreateEnergyTableModel(TimingFigure *tvf, char *name,
DoubleListTimeValue *axis1, DoubleListCapaValue *axis2, DoubleTable
*values, char *type)
```

Associate an energy table model using the given axis and table values. For GNS usage, set `tvf` *to NULL*

`tvf`	Timing figure where table model will be created
`name`	Name of the table model
`axis1`	Y-axis values of the table
`axis2`	X-axis values of the table
`values`	Values to set into the table. It's a 2D array.
`type`	defines the type of the table axis: "slope-slope", "slope-capa" or "slope-ckslope".
`EXAMPLE`	`ttv_CreateTimingTableModel $fig mymodel {1e-12 3e-12} {1e-15 2e-15} {{1e-12 2e-12}{3e-12 4e-12}} "slope-capa"`

---

ttv_SetTimingFigureName

```
void ttv_SetTimingFigureName(TimingFigure *tvf, char *name)
```

Change the name of a timing figure.

`tvf`	Figure to be changed
`name`	New figure name

5.7. Timing DB Browsing

5.7.1. TimingFigure Access

ttv_GetTimingFigure

```
<TimingPathList *> ttv_GetTimingFigure <name>
Obsolete but still working:
<TimingPathList *> ttv_GetTimingFigure <name> <temperature> <supply
value>
```

Gets a timing figure from memory, assuming it has already been loaded. Only usefull when a timing figure comes from a `.lib` *of* `.tlf` *file.*

`name`	Timing figure's name
`EXAMPLE`	`set fig [ttv_GetTimingFigure my_design]`

---

ttv_LoadSpecifiedTimingFigure

```
TimingFigure *ttv_LoadSpecifiedTimingFigure(char *name)
```

Reads a DTX timing figure from disk.

`name`	Timing figure's name. The name of the timing figure should be the same as its file name, without the `dtx` suffix.
`EXAMPLE`	`set fig [ttv_LoadSpecifiedTimingFigure "my_design"]`

---

ttv_LoadSpecifiedTimingPathFigure

```
TimingFigure *ttv_LoadSpecifiedTimingPathFigure(char *name)
```

Reads a TTX timing path figure from disk.

`name`	Timing figure's name. The name of the timing figure should be the same as its file name, without the `ttx` suffix.
`EXAMPLE`	`set fig [ttv_LoadSpecifiedTimingPathFigure "my_design"]`

---

ttv_LoadCrosstalkFile

```
int ttv_LoadCrosstalkFile(TimingFigure *tvf)
```

Loads a CTX file, result of a previous crosstalk analysis, and annotates a timing figure with crosstalk information such as delta-delays. The CTX file should have the same name of the timing figure, with the `ctx` *extension. Returns 1 on failure, 0 on success.*

`tvf`	Pointer on the timing figure to annotate
EXAMPLE	`ttv_LoadCrosstalkFile $fig`

ttv_RecomputeDelays

```
void ttv_RecomputeDelays(TimingFigure *tvf)
```

Recomputes delays and slopes in a timing figure according to new input slopes and output capacitances.

`tvf`	Pointer on the timing figure
EXAMPLE	`ttv_RecomputeDelays $fig`

ttv_GetTimingFigureProperty

```
Property *ttv_GetTimingFigureProperty(TimingFigure *fig, char *code)
```

Returns a property of a timing figure. A Property is a polymorphic type, the returned type depends on the property code.

`fig`	Pointer on the timing figure
`code`	Property code; for available property codes, see the `TimingFigure` object section

5.7.2. TimingPath Access

ttv_GetPaths

```
<TimingPathList *> ttv_GetPaths [<TimingFigue *tf>] [-from
<startnodelist>] [-to <endnodelist>] [-thru <accessnodelist>] [-dir
<dir>] [-nb <nb>] [-critic|-all] [-path|-access] [-max|-min]
Obsolete but still working:
TimingPathList *ttv_GetPaths(TimingFigure *tf, char *clock, char *start,
char *end, char *dir, long number, char *all, char *path, char *minmax);
```

Gets a selection of paths/accesss, depending upon configuration.
If `tf` *is not specified then* `startnodelist`, `endnodelist` *and* `accessnodelist` *are considered* `TimingSignal` *lists. They are node name lists otherwise.*
The `accessnodelist` *is used only if* `-access` *is specified.*

`tf`	Related timing figure to use.
`-from <startnodelist>`	Path start node list (clock node if `-access` is used). If not specified, all possible start nodes are considered.
`-thru <accessnodelist>`	Path access node list (latchs, precharges, breakpoints). Used only with `-access`. If not specified, all possible access nodes are considered.
`-to <endnodelist>`	Path end node list (latchs, precharges, breakpoints, connectors). If not specified, all possible end nodes are considered.
`-dir <dir>`	Path transitions: `uu`, `dd`, `ud`, `du`; `u` or `d` can be replaced by the wildcard `?`. `z` or `/` can be added to the usual direction to respectively retreive only HZ path or only non-HZ path. Default is `??`.
`-nb <nb>`	Maximum number of paths that will appear in the returned list. If `<nb>` is negative or zero, no path number limit will be considered. Default is `-1`.
`-critic`	Returns only critical paths. This is the default.
`-all`	Returns critical and parallel paths.
`-path`	Searches for paths. This is the default.
`-access`	Searches for accesses.
`-max`	Uses maximum delays for the search. This is the default.
`-min`	Uses minimum delays for the search.

`Obsolete options:`

start	Path starting point (clock signal for an access path)
end	Path destination point
dir	Path transitions: uu, dd, ud, du; u or d can be replaced by the wildcard ?. z or / can be added to the usual direction to respectively retreive only HZ path or only non-HZ path.
number	Number of paths/access that will appear in the returned list. If number is negative, no path number limit will be considered
all	all or critic: all paths appear in the returned list or only critical paths disregarding parallel paths
path	Type of paths to return, either path or access
minmax	Maximum (max) or minimum (min) paths are returned

EXAMPLE	`set p_list [ttv_GetPaths $fig -to {*outsig* out} -dir ?r -nb 10 -critic -path -min]`
	or with Timing Signals myclock1 and myclock2:
	`set p_list [ttv_GetPaths -from {$myclock1 $myclock2} -access]`
	Obsolete use:
	`set p_list [ttv_GetPaths $fig * * ?? 10 critic path max]`

ttv_CharacPaths

```
TimingPathList *ttv_CharacPaths(TimingFigure *tf, double slopein, char
*start, char *end, char *dir, long number, char *all, char *path, char
*minmax, double capaout, int propagate)
```

Search for timing paths. Path delay is computed with regard to input slope and output load.

tf	Related timing figure to be driven
slopein	Path input slope to be propagated, if 0, default slope is taken into account.
start	Path starting point (clock signal for an access path)
end	Path destination point.
dir	Path transitions: uu, dd, ud, du; u or d can be replaced by the wildcard ?
number	Number of paths/access that will appear in the returned list. If number is negative, no path number limit will be considered

`all`	`all` or `critic`: all paths appear in the returned list or only critical paths disregarding parallel paths
`path`	Type of paths to return, either `path` or `access`
`minmax`	Maximum (`max`) or minimum (`min`) paths are returned
`capaout`	Load to add to output connectors (on-path bidir connectors affected only in full propagation, see below). If negative load, default is taken into account.
`propagate`	Type of slope propagation, can take values `0` (no propagation), `1` (full propagation) or `2` (1-stage propagation)
EXAMPLE	`set p_list [ttv_CharacPaths $fig 100e-12 a b ud 1 critic path max 10e-15 1]`

ttv_GetParallelPaths

```
TimingPathList *ttv_GetParallelPaths(TimingPath *ta, long num)
```

Returns a list of paths parallel to a given path, i.e. starting end ending on the same nodes, with same transitions, but crossing different intermediary nodes.

`ta`	Pointer to a timing path
`num`	Max number of parallel paths to report
EXAMPLE	`set pr_list [ttv_GetParallelPaths [lindex $p_list 1] 10]`

ttv_ProbeDelay

```
TimingPathList *ttv_ProbeDelay <TimingFigure *tf> <StringList
*nodenamelist> [flags]
Obsolete but still working:
TimingPathList *ttv_ProbeDelay(TimingFigure *tf, double inputslope,
StringList *nodenamelist, char *dir, int nbpaths, char *path_or_access,
char *minmax)
```

Retreives the paths by specifying a list of nodes on the paths without the need for intermediate nodes to be forcibly path stop nodes (latches, precharges, connectors, ...).
The type of the crossed nodes can be specified as wildcards can be used to indicate a set of nodes.

`tf`	Timing figure

`nodenamelist`	Nodes that the path must cross. Must be present at least the start node, all memory/intermediate nodes on the path in the right order and the end node. The types of the nodes can be specified by adding the prefixes: "`-any=`", "`-latch=`", "`-prech=`" and "`-con=`". Respectively for any node, latch, precharge and connector. If wildcards are used for the intermediate nodes without precising the node types, "-latch=" is assumed. Using "-any=" for intermediate nodes can be very time consuming because all the matching nodes combinations will be considered. Adding `-access` before one of this option (eg. -access-latch) will indicate a probe thru the access timing line to the specified node.
`flags:`	
`-dir <dir>`	Requested start-to-end transition. Default is `??` (all transitions).
`-slope <inputslope>`	Value of the input slope to propagate through the path. If negative, the slope computed during database creation will be propagated.
`-nb <maxpath>`	If positive maximum number of paths returned. Default, is `-1` (unlimited).
`-min`	Search for minimum-delay paths
`-max`	Search for maximum-delay paths. This is the default.
`-nosync`	Disable synchornization to the latch opening when crossing latches if stb has been run on the timing figure.
`-noprop`	Disable slope propagation thru the found path gates. `<inputslope>` is ignored.
`EXAMPLE`	To probe a delay from node "net23" to node "net045" thru node "lt1": `set p_list [ttv_ProbeDelay $fig {net23 lt1 net045} -slope 100e-12 -dir ud -nbpaths 1]` To probe paths from all connectors to node "net045" thru any latch: `set p_list [ttv_ProbeDelay $fig {-con=* -latch=* net045} -slope 100e-12]`. To probe the minimum accesses from connector "ck" thru any latch to node "endnode": `set p_list [ttv_ProbeDelay $fig {-con=ck -access-latch=* endnode} -min]`.

ttv_FreePathList

```
void ttv_FreePathList(TimingPathList *lst)
```

Frees the paths in a list

`lst` Pointer on the head of a path list

EXAMPLE `ttv_FreePathList $p_list`

---

ttv_SearchExcludeNodeType

```
void ttv_SearchExcludeNodeType(char *conf)
```

Allows the exclusion of nodes of a given type for path searching

`conf` List of node types separated by spaces. Valid types are `Command`, `Precharge`, `Latch`, `Connector` and `Breakpoint`.

EXAMPLE `ttv_SearchExcludeNodeType "Command Precharge"`

---

ttv_CharacPathTables

```
ListOfDoubleTable *ttv_CharacPathTables(TimingPath *pth,
DoubleListTimeValue *slopes, DoubleListCapaValue *capas, int propagate)
```

Recompute path delay and output slope with all the combinations of slopes and capacitances. Returns a list of 2 matrices (list of lists). The first list contains the delays and the second the slopes. Each line correspond to an input slope and each element of a line to a capacitance.

`pth` Timing path to recompute

`slopes` List of slope values to use

`capas` List of capacitance values to use

`propagate` type of slope propagation, can take values `0` (no propagation), `1` (full propagation))

EXAMPLE `set res [ttv_CharacPathTables $mypath {1e-12 3e-12} {1e-15 2e-15} 1]`

---

ttv_GetTimingPathProperty

```
Property *ttv_GetTimingPathProperty(TimingPath *fig, char *code)
```

Returns a property of a timing path. A Property is a polymorphic type, the returned type depends on the property code.

`fig`	Pointer on the timing path
`code`	Property code; for available property codes, see the `TimingPath` object section

ttv_DetectFalseClockPath

```
void ttv_DetectFalseClockPath(TimingFigure *tf)
```

Checks the evidence of signal propagation of clock paths arriving on command signals (and not on other signals), in order to eliminate possible false paths. Detected false paths are added to the INF file.

`tf`	Pointer on a timing figure

ttv_DetectFalsePath

```
void ttv_DetectFalsePath(TimingFigure *tvf, char *start, char *end)
```

Checks the evidence of signal propagation between two signals, in order to eliminate possible false paths. Detected false paths are added to the INF file.

`tf`	Pointer on a timing figure
`start`	Name of the starting point
`end`	Name of the ending point

ttv_GetGeneratedClockPaths

```
TimingPathList *ttv_GetGeneratedClockPaths(TimingFigure *tvf, TimingEvent
*tve, char *minmax)
```

Gets the path list from a top level clock to the specified generated clock timing event.

`tvf`	Timing figure
`tve`	Generated clock timing event
`minmax`	Maximum (`max`) or minimum (`min`) paths are returned
`EXAMPLE`	`set p_list [ttv_GetGeneratedClockPaths $fig $clockevent max]`

5.7.3. TimingLine Access

ttv_GetLines

```
TimingLineList *ttv_GetLines(TimingFigure *tvf, char *start, char *end,
char *dir, char *linetype)
```

Gets a selection of lines, depending upon configuration.

`tf`	Related timing figure
`start`	Line starting point
`end`	Line destination point
`dir`	Line transitions: `uu`, `dd`, `ud`, `du`; `u` or `d` can be replaced by the wildcard `?`
`type`	Line type filters: `setup`, `hold`, `access`, `hz`, `rc`, `prech`, `eval`, `data` or `all`
EXAMPLE	`set l_list [ttv_GetLines $fig * * ?? "access setup hold"]`

ttv_ComputeLineDelay

```
double ttv_ComputeLineDelay(TimingLine *tl, double slope_in, double
output_capa, char *delayslope, char *maxmin)
```

Recomputes a line delay or slope depending on an input slope and an output capacitance.

`tl`	Related timing line
`slope_in`	desired input slope
`output_capa`	desired output capacitance
`delayslope`	type of delay to recompute `delay` or `slope`
`maxmin`	type of delay to recompute `max` or `min`
EXAMPLE	`set val [ttv_ComputeLineDelay $line 1.5e-12 0.3e-15 delay max]`

ttv_SetTimingLineDelay

```
void ttv_SetTimingLineDelay(TimingLine *tl, char *delayslope, char
*maxmin, double value)
```

Forces the value of the propagation delay or output slope of a line. The line's propagation delay or output slope will never be reevaluated.

`tl` Related timing line

`delayslope` type of delay to set `delay` or `slope`

`maxmin` type of delay to set `max`, `min` or `all`

`value` new value

EXAMPLE `ttv_SetTimingLineDelay $line delay max 1e-9]`

---

ttv_CharacTimingLineModel

```
DoubleTable *ttv_CharacTimingLineModel(TimingFigure *tf, char *name,
DoubleListTimeValue *input_slope, DoubleListCapaValue *output_capa, char
*type)
```

Compute delays or slopes using a given timing line model name. Returns a matrix of values. Each line correspond to an input slope and each element of a line to a capacitance.

`tvf` Timing figure with the timing model

`name` Name of the timing model

`slopes` List of slope values to use

`capas` List of capacitance/slope values to use

`type` defines the type of the table axis: "slope-slope", "slope-capa:delay" or "slope-capa:slope".
"`slope-slope`" (`capas` contains slopes values) returns delay values
"`slope-capas:delays`" returns delay values
"`slope-capa:slope`" returns slope values

EXAMPLE `set res [ttv_CharacTimingLineModel $fig "hold__EN/CP_01_1" {1e-12 3e-12} {1e-15 2e-15} slope-capa:slope]`

---

ttv_GetTimingLineProperty

```
Property *ttv_GetTimingLineProperty(TimingLine *tl, char *code)
```

Returns a property of a timing line. A Property is a polymorphic type, the returned type depends on the property code.

`tl`	Pointer on the timing line
`code`	Property code; for available property codes, see the `TimingLine` object section

5.7.4. TimingPath Reports

ttv_DisplayPathList

```
void ttv_DisplayPathList(FILE *f, TimingPathList *tpl)
```

Prints summary information about paths given in a path list

`tpl`	Pointer on a path list
`f`	File where to save the report, `stdout` for standard output

ttv_DisplayPathListDetail

```
void ttv_DisplayPathListDetail(FILE *f, TimingPathList *tpl)
```

Prints detailed information about paths given in a path list

`tpl`	Pointer on a path list
`f`	File where to save the report, `stdout` for standard output

ttv_DisplayPathDetail

```
void ttv_DisplayPathDetail(FILE *f, int num, TimingPath *tp)
```

Prints detailed information about a given path

`tp`	Pointer on a path
`f`	File where to save the report, `stdout` for standard output
`num`	Number to identify the path

ttv_DisplayPathDetailShowColumn

```
void ttv_DisplayPathDetailShowColumn(char *conf)
```

Selects the columns to be displayed in the reports provided by `ttv_DisplayPath`*. By default, all columns are displayed when available information. Wild cards can be used, eg.* `dt.*slope`*. Prefix* `pl.` *refers to path reports,* `.dt` *to path detail reports,* `c2l.` *to connector to latch and* `stab` *to stability reports.*

conf	list of column IDs separated by spaces. Valid IDs are: pl.index, pl.starttime, pl.startslope, pl.pathdelay, pl.totaldelay, pl.datalag, pl.endslope, pl.startnode, pl.endnode, dt.simacc, dt.simdelta, dt.simslope, dt.simerror, dt.refacc, dt.refdelta, dt.refslope, dt.reflagacc, dt.reflagdelta, dt.ctkacc, dt.ctkdelta, dt.ctkslope, dt.ctklagacc, dt.ctklagdelta, dt.capa, dt.nodetype, dt.nodename, dt.netname, dt.linetype, dt.transistors, dt.clockinfo, stab.from, stab.thru, c2l.absdata or all.

ttv_DisplayPathDetailHideColumn

```
void ttv_DisplayPathDetailHideColumn(char *conf)
```

Selects the columns not to be displayed in the reports provided by ttv_DisplayPath. *By default, all columns are displayed when available information. Wild cards can be used, eg.* dt.*slope. *Prefix* pl. *refers to path reports,* dt. *to path detail reports,* c2l. *to connector to latch and* stab *to stability reports. Hidding* dt.linetype *activates the merge of RC delays with gate delays.*

conf	list of column IDs separated by spaces. Valid IDs are: pl.index, pl.starttime, pl.startslope, pl.pathdelay, pl.totaldelay, pl.datalag, pl.endslope, pl.startnode, pl.endnode, dt.simacc, dt.simdelta, dt.simslope, dt.simerror, dt.refacc, dt.refdelta, dt.refslope, dt.reflagacc, dt.reflagdelta, dt.ctkacc, dt.ctkdelta, dt.ctkslope, dt.ctklagacc, dt.ctklagdelta, dt.capa, dt.nodetype, dt.nodename, dt.netname, dt.linetype, dt.transistors, dt.clockinfo, stab.from, stab.thru, c2l.absdata or all.

ttv_DisplayClockPathReport

```
void ttv_DisplayClockPathReport(FILE *f, TimingFigure *tf, char *clock,
char *minmax, int number)
```

Prints a detailed report of timing paths originating from a clock.

f	File where to save the report, stdout for standard output
tf	Pointer on the timing figure to consider
clock	Starting point name; can be a wildcard name
minmax	min or max; type of path to be searched

number	Maximum number of paths to report

ttv_DisplayConnectorToLatchMargin

```
void ttv_DisplayConnectorToLatchMargin(FILE *f, TimingFigure *tf, char
*inputconnector, char *mode)
```

Prints a report displaying setup and hold constraints on latch points, originating from input connectors (both from rising and falling transitions).

f	File where to save the report, `stdout` for standard output
tf	Pointer on the timing figure to consider
inputconnector	Names of the connectors to consider, can be wildcards.
mode	Controls the amount of information displayed. Valid values are `summary` or `all` (display path detail) associated with `split`. Using `split` displays the report connector one connector at a time. This option can also be used to reduce memory usage on huge UTDs. `margins` will show the computed contsraints for each path in the summary. `pathcomp` permits will display the spice total error as total spice delay versus total tas delay + path margin. By default, the path margin is not included.
EXAMPLE	`ttv_DisplayConnectorToLatchMargin $ofile $fig "*" "summary split"`

ttv_PlotPathDetail

```
void ttv_PlotPathDetail(FILE *f, TimingPath *tp)
```

Plots the waveforms of nodes on a given path.

tp	Pointer on a path
f	File where to save the report, `stdout` for standard output

ttv_SetupReport

```
void ttv_SetupReport(char *conf)
```

Changes units and format used in path reports. Default units are `ns` *and* `pf`*. Formats affect the way most of the values are displayed in the report columns. By default, values are justified right, node and net names are*

displayed together, and delay deviations are given relatively to the path delay.

`conf` — List of names separated by spaces. Valid units are: `ns`, `ps`, `pf`, `ff` and #digits (which defines the precision for delay values, # ranging from 1 to 9). Valid formats are: `JustifyRight`, `JustifyLeft`, `OnlyNetName`, `NodeName`, `HideRC`, `ShowRC`, `LocalError` (which gives local delay deviations), `PathComp` (compares simulated delay versus hitas delay+path margin).

---

ttv_DumpHeader

```
void ttv_DumpHeader(FILE *f, TimingFigure *tvf)
```

Prints information about the given timing figure and the running script

`f` — File where to save the report, `stdout` for standard output

`tvf` — Pointer on a timing figure

---

ttv_DumpFigure

```
ttv_DumpFigure <FILE *f> <TimingFigure *tvf> [type]
```

Prints a table with the Timing Lines of a TimingFigure.

`f` — File where to save the report, `stdout` for standard output

`tvf` — Pointer on a timing figure

`type` — Same type as ttv_GetLines API to filter output. Valid values: `setup`, `hold`, `access`, `hz`, `rc`, `prech`, `eval`, `data` or `all`.

5.7.5. TimingDetail Access

ttv_GetPathDetail

```
TimingDetailList *ttv_GetPathDetail(TimingPath *ta)
```

Returns the list of elementary timing arcs (lines) making up a critical path

`ta`	Pointer on the timing path to consider
EXAMPLE	`set detail [ttv_GetPathDetail $path]`

---

ttv_GetTimingDetailProperty

```
Property *ttv_GetTimingDetailProperty(TimingDetail *det, char *property)
```

Returns a property of a timing detail. A Property is a polymorphic type, the returned type depends on the property code.

`ev`	Pointer on the timing detail
`code`	Property code; for available property codes, see the `TimingDetail` object section

---

5.7.6. TimingDetail Simulation

ttv_SimulatePath

```
ttv_SimulatePath <TimingFigure *tvf> <TimingPath *ta> [-force] [-mc
<number>] [-plot]
```

Constructs and saves on disk the spice netlist and stimuli required for simulation, relative to the given path. This requires all configuration variables related to simulation to be set correctly. Returns 0 on success, 1 on error.

`tvf`	Pointer on a timing figure
`path`	Pointer on the timing path the spice deck is to be extracted from
`-force`	The path will be simulated even if it has already been
`-mc <number>`	The number of monte-carlo simulations to be performed
`-plot`	Generated a waveform file for viewing with gnuplot or gwave. Note that `simUsePrint` must be set.

No description

---

ttv_SimulateCharacPathTables

```
TimingPathTable *ttv_SimulateCharacPathTables(TimingFigure *tvf,
TimingPath *tp, DoubleListTimeValue *slopes, DoubleListCapaValue *capas,
int maxsim)
```

Simulates path with all the combinations of slopes and capacitances values. Returns a matrix (list of lists) of paths. Each line corresponds to an input slope and each element of a line to a capacitance. Each path in the matrix is a replica of the user given path except for the simulated slopes and delays.

`tvf`	Timing figure of the path
`tp`	Timing path to simulate
`slopes`	List of slope values to use
`capas`	List of capacitance values to use
`maxsim`	maximum number of concurent simulation authorized.

EXAMPLE

```
set res [ttv_SimulateCharacPathTables $myfig $mypath {1e-12
3e-12} {1e-15 2e-15} 3]
```

ttv_DriveSpiceDeck

```
int ttv_DriveSpiceDeck(TimingFigure *tvf, TimingPath *path, char
*filename)
```

Constructs and saves on disk the spice netlist and stimuli required for simulation, relative to a given path. This requires all configuration variables related to simulation to be set correctly. Returns 0 on success, 1 on error.

`tvf` Pointer on a timing figure

`path` Pointer on the timing path the spice deck is to be extracted from

`filename` File where to save the report

ttv_DriveSetupHoldSpiceDeck

```
int ttv_DriveSetupHoldSpiceDeck(TimingFigure *tvf, TimingPathList
*datapath, TimingPathList *clockpathlist, char *filename)
```

Constructs and saves on disk the spice netlist and stimuli required for simulation, relative to the computation of a setup or hold value. This requires all configuration variables related to simulation to be set correctly. Returns 0 on success, 1 on error.

`tvf` Pointer on a timing figure

`datapath` Pointer on the data path the spice deck is to be extracted from

`clockpathlist` Pointer on the clock path the spice deck is to be extracted from

`filename` File where to save the report

ttv_DisplayActivateSimulation

```
void ttv_DisplayActivateSimulation(char mode)
```

Enables the construction and simulation of the spice deck of a given path, when a reporting request occurs for this path. The simulation columns in the report appear automatically except explicitly disabled with `ttv_DisplayPathDetailHideColumn`.

`mode`	`yes` or `no` to activate or deactivate spice deck construction and simulation

ttv_Simulate_AddDelayToVT

```
void ttv_Simulate_AddDelayToVT(TimingFigure *tvf, TimingPath *pth,
TimingEvent *latch)
```

Applyed on the path, indicate to CPE to add a measure to retreive the delay to VT of the ending node of `pth` *which is a command generating the event* `latch` *on the latch.*
The value can be retreived using the corresponding TimingPath property after the simulation is done.

`tvf`	Pointer on a timing figure.
`pth`	Pointer on the timing path.
`latch`	Event of the latch generated by the ending node of `pth`.

ttv_Simulate_FoundSolutions

```
int ttv_Simulate_FoundSolutions()
```

Returns the number of path activation solutions found by CPE.
`0` *means the path can not be activated hence it is a false path.*
More than `1` *solution, means CPE tried to find the best solution depending on whether the path was a max or a min path between those solutions.*

5.7.7. TimingSignal Access

ttv_GetTimingSignalListByNet

```
TimingSignalList *ttv_GetTimingSignalListByNet(TimingFigure *tvf, char
*name)
```

Retreives a timing signal list from a net name

`tvf`	Pointer on a timing figure
`name`	Name of the timing signal to retrieve. Should repect the naming conventions defined with `avtCaseSensitive` and `avtVectorize` configuration variables.
EXAMPLE	`set siglist [ttv_GetTimingSignalListByNet $fig net045]`

---

ttv_GetTimingSignal

```
TimingSignal *ttv_GetTimingSignal(TimingFigure *tvf, char *name)
```

Retreives a timing signal from the timing figure; returns NULL on failure.

`tvf`	Pointer on a timing figure
`name`	Name of the timing signal to retrieve. Should repect the naming conventions defined with `avtCaseSensitive` and `avtVectorize` configuration variables.
EXAMPLE	`set sig [ttv_GetTimingSignal $fig net045]`

---

ttv_GetTimingSignalList

```
TimingSignalList *ttv_GetTimingSignalList(TimingFigure *tvf, char *type,
char *location)
```

Retreives a list of timing signals of a given type and location

`tvf`	Pointer on a timing figure
`type`	`connector`, `precharge`, `latch`, `command` or `breakpoint`
`location`	`interface`, `internal` or `all`
EXAMPLE	`set ext_latches [ttv_GetTimingSignalList $fig latch interface]`

ttv_GetClockList

```
TimingSignalList *ttv_GetClockList(TimingFigure *tvf)
```

Returns the list of signals marked as clock in a given timing figure, with a `create_clock` *command for example. Returns an empty list if the timing figure is NULL, or if there are no clock connectors.*

`tvf`	Pointer on a timing figure
EXAMPLE	`set clock_list [ttv_GetClockList $fig]`

ttv_GetTimingSignalProperty

```
Property *ttv_GetTimingSignalProperty(TimingSignal *fig, char *code)
```

Returns a property of a timing signal. A Property is a polymorphic type, the returned type depends on the property code.

`fig`	Pointer on the timing signal
`code`	Property code; for available property codes, see the `TimingSignal` object section

ttv_GetLatchAccess

```
double ttv_GetLatchAccess(TimingFigure *tvf, TimingSignal *latch, char
dir, TimingSignal *command, char *minmax)
```

Returns a latch's intrinsic access delay, from command to latch output

`tvf`	Pointer on a timing figure
`latch`	Pointer on the latch to consider
`dir`	Indicates that the access delay is given for the `falling` or `rising` edge of the latch node. Use `both` to be able to choose between min and max values.
`command`	Associated command node; if NULL, all commands will be considered
`minmax`	`min` or `max`; defines if min or max value is to be chosen when there are multiple choices.
EXAMPLE	`set intrinsic [ttv_GetLatchAccess $fig $lt1 falling $com1 min]`

ttv_GetLatchSetup

```
double ttv_GetLatchSetup(TimingFigure *tvf, TimingSignal *latch, char
dir, TimingSignal *command)
```

Returns a latch's intrinsic setup constraint, from latch input to command

`tvf`	Pointer on a timing figure
`latch`	Pointer on the latch to consider
`dir`	Indicates that the setup constraint is given for the `falling` or `rising` edge of the latch node. Use `both` to be able to choose between min and max values.
`command`	Associated command node; if NULL, all commands will be considered
`minmax`	`min` or `max`; defines if min or max value is to be chosen when there are multiple choices.
EXAMPLE	`set intrinsic [ttv_GetLatchSetup $fig $lt1 falling $com1]`

ttv_GetLatchHold

```
double ttv_GetLatchHold(TimingFigure *tvf, TimingSignal *latch, char dir,
TimingSignal *command)
```

Returns a latch's intrinsic hold constraint, from latch input to command

`tvf`	Pointer on a timing figure
`latch`	Pointer on the latch to consider
`dir`	Indicates that the hold constraint is given for the `falling` or `rising` edge of the latch node. Use `both` to be able to choose between min and max values.
`command`	Associated command node; if NULL, all commands will be considered
`minmax`	`min` or `max`; defines if min or max value is to be chosen when there are multiple choices.
EXAMPLE	`set intrinsic [ttv_GetLatchHold $fig $lt1 falling $com1]`

ttv_GetSignalCapaList

```
DoubleList *ttv_GetSignalCapaList(TimingSignal *tvs)
```

Returns the list of capacitance values (in Farads) for a given timing signal. Values are given in the order: nominal, min and max for rise transition then nominal, min and max for fall transition.

`tvs` Pointer on the timing signal to consider

`EXAMPLE` `set capa_list [ttv_GetSignalCapaList]`

---

ttv_GetFullSignalNetName

```
char *ttv_GetFullSignalNetName(TimingFigure *tvf, TimingSignal *tvs)
```

Returns the full hierarchical net name of a given timing signal. Returned name is to be pasted for further use. Up to 16 full names are kept in order to reduce memory consumption.

`tvf` Pointer on a timing figure

`tvs` Pointer on the timing signal to consider

`EXAMPLE` `set name [ttv_GetFullSignalNetName $fig $sig]`

---

ttv_GetFullSignalName

```
char *ttv_GetFullSignalName(TimingFigure *tvf, TimingSignal *tvs)
```

Returns the full hierarchical name of a given timing signal. Returned name is to be pasted for further use. Up to 16 full names are kept in order to reduce memory consumption.

`tvf` Pointer on a timing figure

`tvs` Pointer on the timing signal to consider

`EXAMPLE` `set name [ttv_GetFullSignalName $fig $sig]`

5.7.8. TimingEvent Access

ttv_GetTimingEventProperty

```
Property *ttv_GetTimingEventProperty(TimingEvent *ev, char *code)
```

Returns a property of a timing event. A Property is a polymorphic type, the returned type depends on the property code.

`ev`	Pointer on the timing event
`code`	Property code; for available property codes, see the `TimingEvent` object section

---

ttv_GetLatchCommands

```
TimingEventList *ttv_GetLatchCommands(TimingSignal *tvs)
```

Returns the list of commands enabling writing in a given latch

`ts`	Pointer on the latch to consider
`EXAMPLE`	`set com_list [ttv_GetLatchCommands $lt1]`

---

ttv_GetLatchEventCommands

```
TimingEventList *ttv_GetLatchEventCommands(TimingSignal *tvs, char dir)
```

Returns the list of commands enabling writing of a specific transition in a given latch

`tvs`	Pointer on the timing signal to consider
`dir`	`u` or `d`; transition to be written
`EXAMPLE`	`set com_list [ttv_GetLatchCommands $lt1 u]`

5.8. Static Timing Analysis

5.8.1. Stability Back-Annotation

stb

```
StabilityFigure *stb(TimingFigure *tf)
```

Launches the static timing analysis, based upon SDC and INF constraints. The analysis can be tuned with configuration variables through `avt_Config ()`. *The function returns a stability figure, which is a mapping of stabilty information (switching windows) on the timing figure.*

`tf`	Pointer on the timing figure the analysis is to be run on

---

stb_LoadSwitchingWindows

```
StabilityFigure *stb_LoadSwitchingWindows(TimingFigure *tvf, char
*filename)
```

Reads an STO file (`.sto` *extension), result of a static timing analysis (*`stb` *funcion). Back-annotates (maps) a timing figure with stability information. Warning: it is not yet compatible with false paths configuration.*

`tvf`	Pointer on the timing figure to back-annotate
`filename`	Name of the STO file to load

---

stb_FreeStabilityFigure

```
void stb_FreeStabilityFigure(StabilityFigure *sf)
```

Deletes a stability figure from memory

`sf`	Pointer on the stability figure to be freed

---

stb_GetStabilityFigureProperty

```
Property *stb_GetStabilityFigureProperty(StabilityFigure *sf, char
*property)
```

Returns a property of a stability figure. A Property is a polymorphic type, the returned type depends on the property code.

`sf`	Pointer on the stability figure.
`code`	Property code; for available property codes, see the `StabilityFigure` object section.

5.8.2. Violations

stb_DisplayErrorList

```
void stb_DisplayErrorList(FILE *f, StabilityFigure *stbfig, double
margin, int nberror)
```

Prints a report of signals presenting timing violations (setup and hold)

`f`	File where to save the report, `stdout` for standard output
`stbfig`	Pointer on a stability figure
`margin`	Value below which a setup/hold slack is reported as a violation. For example, if the margin is 100ps (100e-12 second), a signal with a setup or a hold slack below 100ps will be reported as a violation.
`nberror`	Maximum number of reported violations

---

stb_GetErrorList

```
TimingSignalList *stb_GetErrorList(StabilityFigure *sf, double margin,
int nberror)
```

Returns a sorted list of the signals having the worst violations.

`sf`	Pointer on a stability figure
`margin`	Value below which a setup/hold slack is reported as a violation. For example, if the margin is 100ps (100e-12 second), a signal with a setup or a hold slack below 100ps will be reported as a violation.
`nberror`	Maximum number of reported violations

---

stb_GetSetupSlack

```
double stb_GetSetupSlack(StabilityFigure *fig, char *signame, char dir)
```

Returns the setup slack value of a given signal. If there is no setup slack value, -1 is returned.

`fig`	Pointer on a stability figure
`signame`	Name of the signal to consider
`dir`	`u`, `d` or `?`; reports slack value for a given transition

stb_GetHoldSlack

```
double stb_GetHoldSlack(StabilityFigure *fig, char *signame, char dir)
```

Returns the hold slack value of a given signal. If there is no hold slack value, -1 is returned.

`fig`	Pointer on a stability figure
`signame`	Name of the signal to consider
`dir`	`u`, `d` or `?`; reports slack value for a given transition

stb_DisplaySlackReport

```
stb_DisplaySlackReport <FILE *file> <StabilityFigure *stbfig> [-from
<nodename>] [-to <nodename>] [-thru_node <nodename>] [-dir <dir>] [-
nbslacks <nb>] [-margin <marginval>] [-noprechlag] [common-options]
or
stb_DisplaySlackReport <FILE *file> -slacks <stabilityslackslist> [-
nbslacks <nb>] [common-options]
common-options:
[-setuponly] [-holdonly] [-simple] [-summary] [-displaythru] [-
displaymargins]
Obsolete but still working:
stb_DisplaySlackReport(FILE *f, StabilityFigure *stbfig, char *start,
char *end, char *dir, int number, char *mode, TimeValue margin)
```

Retreives and prints or only prints a slack report.

`file`	File where to save the report, `stdout` for standard output.
`stbfig`	Stability figure.
`-from <nodename>`	DATA VALID start point (clock or input pin).
`-to <nodename>`	DATA VALID end point (latch, precharge, output or directive node).
`-thru_node nodename`	DATA VALID access node when the DATA VALID is an acces path.
`-dir <dir>`	Start and end points transitions, `u`, `d`, `?`. `z` or `/` can be added to the usual direction to respectively display only HZ slacks or only non-HZ slacks. Default is `??`.
`-nbslacks <nb>`	Maximum number of reported setup and hold slacks. Default is unlimited (`<nb>`<=0).

`-margin <marginval>`	Value below which a setup/hold slack is reported as a violation. For example, if the margin is 100ps (100e-12 second), a signal with a setup or a hold slack below 100ps will be reported as a violation. Default is `0`.
`-noprechlag`	Removes slacks whose DATA VALID passes thru transparent precharges when DATA VALID is an access.
`common-options:`	
`-setuponly`	Displays only setup slacks.
`-holdonly`	Displays only hold slacks.
`-summary`	Displays only the summary of the slacks.
`-displaythru`	Displays access node in the summary if the DATA VALID path is an access.
`-displaymargins`	Displays the path margins and the intrinsic margin in the summary.
`-simple`	Displays less detailed slack details by hiding period change operations.

`EXAMPLE`	To display an already obtained StabilitySlack list:
	`stb_DisplaySlackReport $myfile -slacks $myslacklist -displaythru -simple -summary -displaymargins`
	To display a slack report:
	`stb_DisplaySlackReport $myfile $sf -margin 100e-9 -from myclock -to mylatch -dir ?f -simple`

`obsolete options:`	
`mode`	Selects the displayed slack types: `setup`, `hold`, `all`, `margins`, `summary setup`, `summary hold` or `summary all`. `all` means `setup` and `hold`. `margins` will show in the summary the internal margins used for the slack computation. `thru` shows in the summary the latch through which the data arrives as well as the latch command which enables this data.

stb_DisplayCoverage

```
stb_DisplayCoverage <FILE *file> <StabilityFigure *stbfig> [-detail]
```

Prints a coverage report concerning all the signals checked or not checked in the design.

`file`	File where to save the report, `stdout` for standard output.
`stbfig`	Stability figure.

`-detail`	Output the list of the signals not checked.
EXAMPLE	`stb_DisplayCoverage stdout $sfig -detail`

stb_GetSlacks

```
<StabilitySlackList *> stb_GetSlacks <StabilityFigure *stbfig> [-from
<nodename>] [-to <nodename>] [-thru_node <nodename>] [-dir <dir>] [-
nbslacks <nb>] [-margin <marginval>] [-noprechlag] [-setuponly] [-
holdonly]
Obsolete but still working:
StabilitySlackList *stb_GetSlacks(StabilityFigure *stbfig, char *start,
char *end, char *dir, int number, char *mode, TimeValue margin)
```

Returns a list of stability slack objects.

`stbfig`	Stability figure.
`-from <nodename>`	DATA VALID start point (clock or input pin).
`-to <nodename>`	DATA VALID end point (latch, precharge, output or directive node).
`-thru_node nodename`	DATA VALID access node when the DATA VALID is an acces path.
`-dir <dir>`	Start and end points transitions, `u`, `d`, `?`. `z` or `/` can be added to the usual direction to respectively display only HZ slacks or only non-HZ slacks. Default is `??`.
`-nbslacks <nb>`	Maximum number of retreived setup and hold slacks. Default is unlimited (`<nb><=0`).
`-margin <marginval>`	Value below which a setup/hold slack is reported as a violation. For example, if the margin is 100ps (100e-12 second), a signal with a setup or a hold slack below 100ps will be reported as a violation. Default is `0`.
`-noprechlag`	Removes slacks whose DATA VALID passes thru transparent precharges when DATA VALID is an access.
`-setuponly`	Retreives only setup slacks.
`-holdonly`	Retreives only hold slacks.

EXAMPLE	`set sl [stb_GetSlacks $sf -from ck -to end* -setuponly -nbslacks 10]`
	Obsolete:
	`set sl [stb_GetSlacks $sf * * ?? 0 setup 100e-9]`

stb_FreeSlackList

```
void stb_FreeSlackList(StabilitySlackList *cl)
```

Frees a list of stability slack objects.

`cl` slack object list

---

stb_GetSlackProperty

```
Property *stb_GetSlackProperty(StabilitySlack *so, char *property)
```

Returns a property of a slack object. A Property is a polymorphic type, the returned type depends on the property code.

`so` Pointer on the slack object

`code` Property code; for available property codes, see the `StabilitySlack` object section

---

stb_SortSlacks

```
<StabilitySlackList *> stb_SortSlacks [<StabilitySlackList *> l1]
[<StabilitySlackList *> l2] ...
```

Concatenates the given slack lists and sorts the resulting list from the worst slack to the best.

`l1, l2, ...` StabilitySlack list to merge and sort.

`EXAMPLE` `set result [stb_SortSlacks $slacklist1 $slacklist2 $slacklist3]`

---

stb_ComputeSlacks

```
<StabilitySlackList *> stb_ComputeSlacks <StabilityFigure *sf> <char
*datavalid> <operation> <char *datarequired> [options]
or
<StabilitySlackList *> stb_ComputeSlacks <TimingPathList *datavalid>
<operation> <TimingPathList *datarequired> [options]
options:
[-nextperiod] [-margin <val>] [-nosync]
```

Computes slacks for a combination of data valid path list and required path list.

The data valid path and data required path can be given as a specification or directly as a path list.
A specification is a string containing the arguments to be given to the ttv_ProbeDelay API without mentioning the Timing Figure argument which is deduced from the Stability Figure. If "-min" or "-max" is not specified in a specification, "-max" for data valid and "-min" for data required is assumed for `before` *operation and the opposite for* `after` *operation.*

`sf` Stability figure.

`datavalid` DATA VALID specification or path list.

`datarequired` DATA REQUIRED specification or path list.

`operation` Operation to do between the two lists. Value can be `before` or `after`. `before` corresponds to a setup and `after` to a hold.

`options:`

`-margin <val>` Adds an additionnal margin to the calculus. The higher the margin the worse is the slack values. If a negative value is given and the datariquired path ends on the command of the dala valid path end latch, the UTD latch to command intrinsic margin is will be used.

`-nextperiod` If the data required path and the data valid path comes from the same clock edge, tells the API that the data required is generated by the next cycle.

`-nosync` Disables all kind of guessed data required period changes done automatically by the API hence the given paths are compared directly.

`EXAMPLE` By using the specifications:

```
set slacks [stb_ComputeSlacks $sf "{-con=ck* -access-latch=convlat convsig}" before "{ck1 convck convsig} -dir ? d" -margin 1ps]
```

By using the path lists:

```
set slacks [stb_ComputeSlacks $data_paths_min after $clock_path_max -margin 2ps -nextperiod]
```

stb_FindLagPaths

```
stb_FindLagPaths <file> <slacklist> [-margin <value>] [-lagmargin <value>] [-maxdepth <number>] [-closingpath]
```

Reports the lag paths for each given slack. This API can help find possible false paths generating data lag hence creating wrong negative slacks.

`file`	File where to save the report, `stdout` for standard output.
`slacklist`	List of slacks to analyse.
`-margin <value>`	Only analyse slacks whose value is lower than or equal to `value`. Default is 1s (i.e. all given slacks).
`-lagmargin <value>`	Only analyse slacks where the data lag value is greater than `value`. Default is 0.
`-maxdepth <number>`	Stop backtracking after a maximum of `number` paths for each slack. Default is 3.
`-closingpath`	Displays the closing clock path in case of huge lags where the latch closing is responsible for the lag maximum value.
`EXAMPLE`	stb_FindLagPaths stdout $myslacks -lagmargin 100e-12

stb_DriveReport

```
void stb_DriveReport(StabilityFigure *stbfig, char *filename)
```

Drives a report file corresponding to a stability figure

`stbfig`	Stability figure
`filename`	File where to save the report

5.9. Statistical Runs

5.9.1. MonteCarlo Runs

avt_SetMainSeed

```
void avt_SetMainSeed(int value)
```

Sets the main seed to use when evaluating random mathematical function with monte-carlo runs enabled. The main seed is used for anything but model parameters in a ".mcparam" spice section concerning transistor model alterations. This function goal is to enable the reproducibility of past results.

`value`	seed value.
EXAMPLE	`avt_SetMainSeed 123456`

---

avt_GetMainSeed

```
int avt_GetMainSeed()
```

Returns the main seed used to evaluate random mathematical function with monte-carlo runs enabled. The main seed is used for anything but model parameters in a ".mcparam" spice section concerning transistor model alterations. This function goal is to enable the reproducibility of past results.
This function must be called after HiTas execution.

EXAMPLE	`puts [avt_GetMainSeed]`

---

avt_SetGlobalSeed

```
void avt_SetGlobalSeed(int value)
```

Sets the seed to use when evaluating random mathematical function with monte-carlo runs enabled. Global seed is used only for parameters in a ".mcparam" spice section concerning transistor model alterations. This function goal is to enable the reproducibility of past results.

`value`	seed value.
EXAMPLE	`avt_SetGlobalSeed 654321`

---

avt_GetGlobalSeed

```
int avt_GetGlobalSeed()
```

Returns the main seed used to evaluate random mathematical function with monte-carlo runs enabled. Global seed is used only for parameters in a ".mcparam" spice section concerning transistor model alterations. This function goal is to enable the reproducibility of past results.
This function must be called after HiTas execution.

EXAMPLE

```
puts [avt_GetGlobalSeed]
```

---

runStatHiTas

```
runStatHiTas <numberofruns> [-slavescript <name>] [-jobscript <name>] [-datafile <name>] [-incremental | -override]
```

Initiates monte-carlo statistical timing runs through the call of several independent hitas (slave) runs.
This command creates a master process which will centralize all the different monte-carlo results.
The TCL variable `ssta_MasterPath` *is available in the slave runs and indicates the directory where the master script is run.*

`numberofruns` Scheduling description of the jobs define as `<total number of runs>:<maximum number of parallel runs>` to run. Or simply the number of run desired if no parallelism is wanted.

`-jobscript <name>` Script file name. This script is called to spawn each slave runs. By default the slave scripts run on the same machine in separate sub-directories.

`-slavescript <name>` The slave monte-carlo script name which run a single monte carlo job. This argument can be used if the slave script is not the one using the API runStatHiTas. By default it's the current script.

`-datafile <name>` Result file name. If the file already exists, a number is added to the filename if neither `-incremental` nor `-override` are specified. By default the name is set to "ssta_data.log".

`-incremental` Force all the results to be added at the end of the data file preserving previous results.

`-override` Overrides existing datafile.

EXAMPLE

```
runStatHiTas 500:2 -slavescript slave-script.tcl -datafile monte-data.log
```

Runs `500 slave-script.tcl` (`2` max in parallel) with the result file `monte-data.log`.

---

avt_McPostData

```
void avt_McPostData(char *msg)
```

Used by a slave monte-carlo run, tells the master to write the given message (result) to the result file.
This function exits the slave script.

`msg` Message to send.

`EXAMPLE` `avt_McPostData [list $myworstsetupslack $myworstholdslack]`

---

avt_McInfo

```
void avt_McInfo(char *msg)
```

Used by a slave monte-carlo run, tells the master to display/log the given message.
The message does not go to the result file.

`msg` Message to send.

`EXAMPLE` `avt_McInfo "Running into STA stage"`

5.9.2. MonteCarlo Analysis

ssta_ToolBox

```
ssta_ToolBox [-parsedatafile] [-getfield <filednum>] [-getdistrib]
[options]
options:
[-values <valuelist>] [-filename <fname>]
[-parsefunction <funcname>] [-fixedrange <time>] [-nbrange <number>]
```

Handles various operations related to statistical timing analysis results.

`-parsedatafile` Reads a SSTA data file and returns its content as a list (one entry per SSTA run) of 2 element lists with the SSTA run number and the line corresponding to the SSTA run from the file: {{<ssta run number> <file line>} {...} ...}.

`-getdistrib`	Returns a distribution from the list of values. The distribution is a list of list with 5 elements: {{<low range value> <high range value> <number of occurence> <cumulative number of occurence> <cumulative % of occurence>} {...} ...}.
`-getfield <filednum>`	Reads and returns the `<filednum>`'th field of each line of a file: {<line1 field value> <line2 field value> ...}.

`options:`

`-values <valuelist>`	List of values to use. Works with `-getdistrib`.
`-filename <fname>`	Filename of the file to read from or to drive to. Works with `-getfield`, `-parsedatafile`.
`-parsefunction <funcname>`	Function name to run while parsing result file. Each line parsed is feed to the function as one argument which is a list with 2 values: {<ssta run number> <file line>}. When `-parsefunction` is used, the API returns nothing. Works with `-parsedatafile`.
`-nbrange <number>`	Specifies the number of range to use when building a distribution. The default value is `20`. Works with `-getdistrib`.
`-fixedrange <time>`	Specifies a range length rather than a range number to build a distribution. Works with `-getdistrib`.
`EXAMPLE`	ssta_ToolBox -getdistrib -values {1 3 5 1 8}

ssta_PathReport

```
Usage 1: ssta_PathReport -senddata <path list> <methodname>
Usage 2: ssta_PathReport -display <ssta result file> <filedescriptor>
```

Sends a predefined monte-carlo result from a given path list to the master ssta script or drives a report from predefined sent data file.

`Usage 1:`

`-senddata <path list> <methodname>`	Sends a predefined result extracted from the given `<path list>`. `<path list>` is standard path list obtained thru ttv_GetPaths for instance. Only one method (kind of results to output) exists for now called "simple".

`Usage 2:`

`-display <ssta result file> <filedescriptor>`	Uses the predefined data writen into result file `<ssta result file>` to output a specific report concerning the path monte-carlo characteristics. `<filedescriptor>` if the result of a "fopen" command call in with the report will be written.
`EXAMPLE`	ssta_ToolBox -senddata $mypathlist simple

ssta_SlackReport

```
Usage 1: ssta_SlackReport -senddata <stability figure> <methodname> Usage 2: ssta_SlackReport -display <ssta result file> <filedescriptor> Usage 3: ssta_SlackReport -plot <ssta result file> <output filename>
```

Sends a predefined monte-carlo result from a given path list to the master ssta script or drives a report from predefined sent data file.

`Usage 1:`

`-senddata <stability figure> <methodname>`	Sends a predefined result extracted from the given `<stability figure>`. Only one method (kind of results to output) exists for now called "simple".

`Usage 2:`

`-display <ssta result file> <filedescriptor>`	Uses the predefined data writen into result file `<ssta result file>` to output a specific report concerning the path monte-carlo characteristics. `<filedescriptor>` if the result of a "fopen" command call in with the report will be written.

`Usage 3:`

`-plot <ssta result file> <output filename>`	Uses the predefined data writen into result file `<ssta result file>` to output a gnuplot view of the distributions of worst setup and hold slacks. The gnuplot file output are prefixed with `<output filename>`.
`EXAMPLE`	ssta_ToolBox -senddata $stbfig simple or ssta_SlackReport -plot "ssta_data.log" "distrib"

5.10. Timing Abstraction

5.10.1. Blackbox Modelling

tma_GetConnectorAxis

```
DoubleList *tma_GetConnectorAxis(TimingFigure *tvf, char *type, char
*name)
```

Returns the defined connector axis set for a connector/signal.
If no axis have been defined for the signal or the signal does not exist, the returned list contains only one element with value `-1`.
There is an exeption if the signal is an internal clock and no axis is defined for it. In this case, the slope max of the clock is returned.

`fig`	Pointer on the timing figure.
`type`	Type of axis: `slope` or `capacitance`
`name`	Name of the signal.
`EXAMPLE`	`set slopeaxis [tma_GetConnectorAxis $fig slope ck]`

---

tmabs

```
TimingFigure *tmabs <TimingFigure *fig> <BehavioralFigure *befig>
<clocks> <inputs> <outputs> [flags]
or
TimingFigure *tmabs <TimingFigure *fig> \[-behaviour <bef>] \[-clocks
<clocklist>] \[-inputs <inputlist>]
\[-outputs <outputlist>] \[-internal <internallist>] [flags]
```

Generates a characterization from a timing database and associates functionality if provided.
It returns a blackbox timing figure with constraints at the interface of the circuit.
The constraints generated can be filter by input, output and/or clock connectors.
Generated clocks are handled and simulations can also be used thru CPE to get spice precise results for the circuit characterisation.

`fig`	Pointer on the timing figure to be blackboxed.
`befig`	Pointer on the behavioral figure to associate with the blackbox. Set to `NULL` if no info is available.

`clocks`	list of clock names to consider in the timing figure.
`inputs`	list of input names to consider in the timing figure.
`outputs`	list of output names to consider in the timing figure.
`flags:`	
`-simulate <list>`	Indicates what to simulate. `<list>` is a list of path description. A path description is a list of 1 or 2 pin names. The list looks like `{pin_name related_pin_name}` following .lib convention. if the related_pin_name is omited it is considered to be *.
`-maxsim <number>`	Maximum number of parallel simulations to launch to speed up characterisations done with simulations.
`-enablecache`	Enables the use of a cache file with already simulated path delays to continue a previously stopped run.
`-scalevalues <list>`	Gives a list of factors to use to computed default slope and capacitance axis in case they are not specified in the information file.
`-verbose`	Enables verbose mode.
`-detailfile <filename>`	Drives in the file `<filename>` the detail of the paths used for each characterisation.
`-minonly`	Computes only min delays.
`-maxonly`	Computes only max delays.
`-setuponly`	Computes only setups.
`-holdonly`	Computes only holds.
`-dumpdtx`	Drive the computed blackbox at the end of the process.
`-exitonerror`	Exit on any error. Most likely simulation errors.
`-ignoremargins`	Dot not add defined path margins to the computed/simulated delays.
`-detectfalsepath`	Run a falsepath detection prior to begin the blackbox creation.
`-addpins <pinlist>`	Add non existing connectors into the blackbox. Direction of pins can be defined in the list. eg. `input0 input1 i output o` . See `ttv_AddConnector` for available pin directions.

EXAMPLE `set bbox [tmabs $tf NULL * * * -maxsim 4 -simulate * -verbose -enablecache]`

tma_SetMaxCapacitance

```
void tma_SetMaxCapacitance(TimingFigure *bbox, char *name, CapaValue value)
```

Sets the max capacitance to report in .lib file.

`bbox` Blackbox timing figure.

`name` Name of the connector (can be a regular expression).

`value` Capacitance value

EXAMPLE `set slopeaxis [tma_GetConnectorAxis $fig slope ck]`

lib_DriveFile

```
void lib_DriveFile(List *fig_list, List *befig_list, char *file, char *delaytype)
```

Saves a list of blackboxed timing figures on disk, in a single `.lib` *file*

`fig_list` List of timing figures to print in the `.lib` file. For a single timing figure, use a list of one element (example given below)

`befig_list` List of behavioral figures to associate with the timing figures. There should be a one-to-one correspondence between the elements in `befig_list` and `fig_list`, and there should be the same number of elements if the two lists. Use `NULL` to fill the gaps in `befig_list`.

`file` Name of the `.lib` file to create

`delaytype` Defines if the `cell_rise` and `cell_fall` timing groups will have only maximum delays (`max`), minimum delays (`min`) or both (`both`).

EXAMPLE `lib_DriveFile [list $bbox] NULL cpu.lib max`

lib_DriveHeader

```
void lib_DriveHeader(TimingFigure *fig, FILE *file, char *libname)
```

Prints a `.lib` *header in a file, regarding to the information present in the given timing figure*

`fig`	Pointer on the timing figure to consider
`file`	Pointer on the file where to print the `.lib` header
`libname`	Name to be put in the `library` statement of the `.lib` header

lib_CanonicPinName

```
char *lib_CanonicPinName(char *name)
```

Adapts pin names according to the `.lib` *syntax (typically bus delimiter are replaced by* _)

`name`	Pin's name

5.10.2. Greybox Modelling

ttv_GetConstraints

```
TimingConstraintList *ttv_GetConstraints(TimingFigure *tf, char
*inputconnector, char *towhat)
```

Computes the constraints at the interface of a timing figure. Returns a list of TimingConstraint.

`tf`	Timing figure
`inputconnector`	Connector whose constraints must be computed
`towhat`	Ending path node types, can be a mix of `latch`, `precharge`, `clockgating`, `precharge` or `all`
`EXAMPLE`	`set cl [ttv_GetConstraints $fig * all`

ttv_FreeConstraints

```
ttv_FreeConstraints <allobj>
```

Frees the constraints.

`allobj`	Constraint objects to free
`EXAMPLE`	`ttv_FreeConstraints $myconstraints`

ttv_GetTimingConstraintProperty

```
Property *ttv_GetTimingConstraintProperty(TimingConstraint *co, char
*property)
```

Returns a property of a timing constraint. A Property is a polymorphic type, the returned type depends on the property code.

`co`	Pointer on the timing constraint
`property`	Property code; for available property codes, see the `TimingConstraint` object section

5.11. Crosstalk DB Browsing

5.11.1. Building statistics

ctk_LoadAggressionFile

```
void ctk_LoadAggressionFile(StabilityFigure *stbfig)
```

Loads the crosstalk database (generated with `stb` *in crosstalk mode), and back-annotates a stability figure*

`stbfig`	Pointer on the stability figure to back-annotate

---

ctk_LoadCrosstalkResults

```
ctk_LoadCrosstalkResults <timingfigure>
```

Puts the UTD in the same state as after the crosstalk analysis was done. The corresponding StabilityFigure is returned. Functions 'ttv_LoadCrosstalkFile', 'stb_LoadSwitchingWindows' then 'ctk_LoadAggressionFile' are called to do this action.

`timingfigure`	TimingFigure to put into post-crosstalk analysis state
`EXAMPLE`	`set stabfig [ctk_LoadCrosstalkResults $myUTD].`

---

ctk_DriveStatCtk

```
void ctk_DriveStatCtk(StabilityFigure *stbfig)
```

Drives a `.ctk` *crosstalk analysis report file*

`stbfig`	Pointer on the stability figure associated

---

ctk_BuildCtkStat

```
void ctk_BuildCtkStat(StabilityFigure *stbfig)
```

Builds the internal crosstalk table of statistics for events and delays

`stbfig`	Pointer on the stability figure associated

5.11.2. Browsing statistics

ctk_GetStatNodeProperty

```
Property *ctk_GetStatNodeProperty(StabilityFigure *stbfig, int index,
char *property)
```

Returns a property of a crosstalk node. A Property is a polymorphic type, the returned type depends on the property code.

`stbfig`	Pointer on a stability figure
`index`	Number between 1 and the value returned by `ctk_GetNumberOfCtkStatNode`
`property`	Property code; for available property codes, see the `StatNode` object section

---

ctk_GetNumberOfCtkStatNode

```
int ctk_GetNumberOfCtkStatNode(StabilityFigure *stbfig)
```

Returns the number crosstalk node statistics

`stbfig`	Pointer on a stability figure

---

ctk_GetCtkStatNodeFromEvent

```
int ctk_GetCtkStatNodeFromEvent(StabilityFigure *stbfig, TimingEvent
*event)
```

Returns the crosstalk statistic index of a given timing event

`stbfig`	Pointer on a stability figure
`event`	Pointer on a timing event

---

ctk_GetNumberOfCtkStatLine

```
int ctk_GetNumberOfCtkStatLine(StabilityFigure *stbfig)
```

Returns the number crosstalk delay statistics

`stbfig`	Pointer on a stability figure

---

ctk_GetStatLineProperty

```
Property *ctk_GetStatLineProperty(StabilityFigure *stbfig, int index,
char *property)
```

Returns a property of a crosstalk delay line. A Property is a polymorphic type, the returned type depends on the property code.

`stbfig`	Pointer on a stability figure
`index`	Number between 1 and the value returned by `ctk_GetNumberOfCtkStatLine`
`property`	Property code; for available property codes, see the `StatLine` object section

5.11.3. Sorting statistics

ctk_SortCtkStatLine

```
void ctk_SortCtkStatLine(StabilityFigure *stbfig, char *criterion)
```

Sorts crosstalk delay statistics according to various criteria

`stbfig`	Pointer on a stability figure
`criterion`	`ABSOLUTE_DELAY`, `ABSOLUTE_MAX_DELAY`, `ABSOLUTE_MIN_DELAY`, `RELATIVE_DELAY`, `RELATIVE_MAX_DELAY`, `RELATIVE_MIN_DELAY`, `ABSOLUTE_SLOPE`, `ABSOLUTE_MAX_SLOPE`, `ABSOLUTE_MIN_SLOPE`, `RELATIVE_SLOPE`, `RELATIVE_MAX_SLOPE` or `RELATIVE_MIN_SLOPE`.

5.11.4. Getting aggressors

ctk_GetAggressorList

```
AggressorList *ctk_GetAggressorList(StabilityFigure *stbfig, TimingEvent
*event)
```

Returns the list of all the aggressors of an event. This list must be freed with the `ctk_FreeAggressorList` *command.*

`stbfig`	Pointer on a stability figure
`event`	Pointer on a timing event

ctk_GetAggressorProperty

```
Property *ctk_GetAggressorProperty(Aggressor *aggressor, char *property)
```

Returns a property of a aggressor node. A Property is a polymorphic type, the returned type depends on the property code.

`aggressor`	Pointer on a aggressor
`property`	Property code; for available property codes, see the `Aggressor` object section

ctk_FreeAggressorList

```
void ctk_FreeAggressorList(AggressorList *list)
```

Frees the list given by `ctk_GetAggressorList`

`list`	Pointer on the aggressor list to be freed

5.12. CPE (path simulation)

5.12.1. Pattern Generation

cpe_DefineCorrelation

```
cpe_DefineCorrelation <net1> <relation> <net2>
```

Defines the relation between 2 signals to help computing propagation condition for simulations.

`net1`	First net name.
`relation`	Value '=' means `net1=net2` Value '`!=`' means `net1`=not(`net2`)
`net2`	Second net name.
`EXAMPLE`	`cpe_DefineCorrelation clk != clkb`

Chapter 6. Error Codes

6.1. API

API-001	Error executing <string>_AtLoad_Initialize(): <string>
API-002	Could not open dynamic library '<string>' reason: <string>
API-003	Cannot read file <string>
API-004	Internal error #<decimal>
API-005	<string>:<decimal>: '<string>' not defined
API-006	<string>:<decimal>: '<string>' already used, primary declaration line <decimal>, file <string>
API-007	<string>:<decimal>: Only int*, void*, double *, char* and FILE* are accepted
API-008	<string>:<decimal>: illegal format string '%%<character>'
API-009	<string>:<decimal>: not enough arguments for format
API-010	<string>:<decimal>: only const char* accepted
API-011	<string>:<decimal>: too many arguments for format
API-012	<string>:<decimal>: exclude can't be in a conditional block
API-013	<string>:<decimal>: '*' doesn't match with type
API-014	<string>:<decimal>: function type FILE* doesn't match
API-015	<string>:<decimal>: type of variable '<string>' doesn't match
API-016	<string>:<decimal>: '<string>' might be used uninitialized
API-017	<string>:<decimal>: digit '<decimal>' doesn't match
API-018	<string>:<decimal>: string '<string>' doesn't match
API-019	<string>:<decimal>: flow '<string>' doesn't match
API-020	<string>:<decimal>: undefined type '<string>'

API-021	<string>:<decimal>: sizeof(<string>) is unknown
API-022	<string>:<decimal>: assignment from <string><string> to <string><string> without a cast
API-023	<string>:<decimal>: incompatible type assignment <string><string> != <string><string>
API-024	<string>:<decimal>: variable '<string>' can not be indexed
API-025	<string>:<decimal>: forbiden operation on this variable type
API-026	<string>:<decimal>: '<string>' has an unexpected type
API-027	<string>:<decimal>: '<string>' should not be pointers
API-028	<string>:<decimal>: type of '<string>' and '<string>' mismatch
API-029	<string>:<decimal>: '<string>' and '<string>' should not be pointers
API-030	<string>:<decimal>: '<string>' has an unexpected type
API-031	<string>:<decimal>: unauthorized test on type '<string>'
API-032	<string>:<decimal>: interpreter can not cast <string> to <string>
API-033	<string>:<decimal>: too few arguments for function 'malloc'
API-034	<string>:<decimal>: too many arguments in function 'malloc'
API-035	<string>:<decimal>: too few arguments for function 'callfunc'
API-036	<string>:<decimal>: callfunc: only 'char *' pointer type can be used in function call
API-037	<string>:<decimal>: can not assign <string><string> to <string><string>
API-038	<string>:<decimal>: unknown variable '<string>'
API-039	<string>:<decimal>: unautorized operation on pointers
API-040	<string>:<decimal>: unautorized operation on type <string><string>
API-041	<string>:<decimal>: division by zero
API-042	<string>:<decimal>: unautorized operation on type '<string>'
API-043	<string>:<decimal>: can not make the requested dereference of '<string><string>'

API-044	<string>:<decimal>: can not make the dereference of '<string>'
API-045	<string>:<decimal>: Can not call functions with more then <decimal> arguments
API-046	<string>:<decimal>: function '<string>' can't be found in the dynamic libraries
API-047	<string>:<decimal>: <string>
API-048	error happens at <string>:<decimal>
API-049	<string>:<decimal>: variable '<string>' already declared in this scope
API-050	<string>:<decimal>: type '<string>' must be used as pointer
API-051	<string>:<decimal>: type of '<string>' must be 'FILE *'
API-052	<string>:<decimal>: Fatal error while executing program
API-053	<decimal> errors. Cannot execute
API-054	<string>:<decimal>: parameter '<string>' is uninitialised
API-055	<string>:<decimal>: conflicting type for parameter '<string>' : '<string><string>'!='<string><string>'
API-056	<string>:<decimal>: return value for void function
API-057	<string>:<decimal>: return value for void function
API-058	<string>:<decimal>: conflicting type for return value : '<string><string>'!='<string><string>'
API-059	Function '<string>' used in action was not found Location(s):
API-060	<string>:<decimal>: too many arguments in function '<string>'
API-061	<string>:<decimal>: too few arguments for function '<string>'
API-062	Somewhere: function '<string>' can't neither be found in interpreter nor in the dynamic libraries
API-063	Somewhere executing <string>
API-064	Error happens at somewhere when executing '<string>'
API-065	Error executing TCL function '<string>'

6.2. API AVT

AVTAPI-001	could not find netlist '<string>' in memory
AVTAPI-002	unknown file format '<string>' for '<string>'
AVTAPI-003	could not create file '<string>'
AVTAPI-004	could not find connector '<string>'
AVTAPI-005	could not find node for connector '<string>'
AVTAPI-006	Unable to encrypt file '<string>'

6.3. API BEG

BEGAPI-001	name '<string>' does not exist in figure '<string>'
BEGAPI-002	could not find correspondance for instance '<string>'
BEGAPI-003	delay variable '<string>' does not exist
BEGAPI-004	could not export behaviour '<string>'. Current instance behaviour is not created yet
BEGAPI-005	behaviour '<string>' already exists
BEGAPI-006	could not import behaviour '<string>'. It does not exist

6.4. API CNS

CNSAPI-001	cone figure <string> already loaded
CNSAPI-002	transistor Netlist <string> already loaded
CNSAPI-003	no Cone figure was loaded for '<string>'

6.5. API CTK

CTKAPI-001	No crosstalk statistics available.
CTKAPI-002	No crosstalk statistics available for this event.

CTKAPI-003	unknown property <string>
CTKAPI-004	<string>: The CTK_NETLIST is not set, function ignored
CTKAPI-005	<string>: Error occured when calling function
CTKAPI-006	<string>: Unknown direction '<character>' using 'u'
CTKAPI-007	<string>: Unknown delay type '<character>' using 'w'

6.6. API FCL

FCLAPI-001	fclMarkSignal: Illegal marking '<string>'
FCLAPI-002	fclMarkTransistor: transistor '<string>' does not exist
FCLAPI-003	fclOrientSignal: multiple orientation on '<string>', resolved to %u

6.7. API GNS

GNSAPI-001	no corresponding transistor to <string>
GNSAPI-002	'<string>' has an RC network!
GNSAPI-003	connectors discrepancy between `<string>' and `<string>'
GNSAPI-004	number of connectors differs between `<string>' and `<string>'
GNSAPI-005	gns_AddExternalTransistors, wrong parameter
GNSAPI-006	can't reconnect transistor `<string>' (`<string>') to signal `<string>' (`<string>')
GNSAPI-007	connector `<string>' is on the interface in model `<string>', but the signal seem to be internal
GNSAPI-008	no instance `<string>': netlist `<string>' is flat
GNSAPI-009	Internal error #<decimal>
GNSAPI-010	<string>: no ground signal can be found in netlist
GNSAPI-011	no connector `<string>' in netlist `<string>'
GNSAPI-012	gns_KeepBestInstanceDelayAWE : no model correspondance

GNSAPI-013	gns_KeepBestInstanceDelayAWE : no instance founded
GNSAPI-014	gns_KeepWorstInstanceDelayAWE : no model correspondance
GNSAPI-015	gns_KeepWorstInstanceDelayAWE : no instance founded
GNSAPI-016	connectors <string> and <string> don't come from the same signal

6.8. API INF

INFAPI-001	you must first use 'inf_SetFigureName' prior to use other functions
INFAPI-002	figure '<string>' could not be found
INFAPI-003	invalid command state given: '<string>'
INFAPI-004	missing clock parameter
INFAPI-005	clock '<string>' not defined yet
INFAPI-006	no period specified for clock '<string>' or no default period
INFAPI-007	missing clock parameter
INFAPI-008	too many parameters starting at '<string>'
INFAPI-009	invalid bypass location given: '<string>'
INFAPI-010	unknown mutex type given: '<string>'
INFAPI-011	unknown crosstalk mutex type given: '<string>'
INFAPI-012	no signal elements in falsepath
INFAPI-013	invalid signal type '<string>'
INFAPI-014	unrecognized token '<string>'
INFAPI-015	unknown section '<string>'
INFAPI-016	'<string>' is not a number
INFAPI-017	too much values for slope or capa generation
INFAPI-018	unknown type given: '<string>'
INFAPI-019	inf_DisableTimingArc: invalid direction '<string>'

INFAPI-020	unknown characteristic '<string>'
INFAPI-021	bad value '<string>' for 'stbUnits'
INFAPI-022	Clock '<string>' is already defined.
INFAPI-023	inf_DefineStability command ignored
INFAPI-024	missing parameter
INFAPI-025	invalid parameter '<string>'
INFAPI-026	unsupported operation requested
INFAPI-027	incompatible tokens given for directive Using "... with sig rising|falling" or "... before|after sig up|down" is forbidden.
INFAPI-028	invalid type '<string>', ignoring this entry The given type does not apply to this section and can't be handled.
INFAPI-029	out of range probability value: %g The probability value must be a value ranging from 0 to 1.

6.9. API SIM

SIMAPI-001	could not find signal '<string>' in netlist
SIMAPI-002	could not find delay of node '<string>'
SIMAPI-003	could not find slope of node '<string>'
SIMAPI-004	authorized type for delay type are SIM_MAX and SIM_MIN
SIMAPI-005	sim_ComputeSetup: sens_d must be 'U' or 'D'
SIMAPI-006	sim_ComputeHold: sens_d must be 'U' or 'D'
SIMAPI-007	sim_ComputeAccess: sens_c must be 'U' or 'D'
SIMAPI-008	Connector '<string>' could not be found in netlist.
SIMAPI-009	'<string>' command file is generated with default wrapper
SIMAPI-010	sim_SetNoiseAnanlyseType : noise_type must be SIM_MIN or SIM_MAX
SIMAPI-011	AddStuckLevelVector: bad value 0x<string> for node <string>

SIMAPI-012	AddStuckLevel: bad value '<character>' for node <string>
SIMAPI-013	AddInputSlope param 4: allowed values are SIM_RISE and SIM_FALL
SIMAPI-014	AddInitLevel param 2: allowed values are 0 and 1
SIMAPI-015	Unknown simulator '<string>'. Using HSPICE.

6.10. API STB

STBAPI-001	unknown direction '<character>'
STBAPI-002	unknown mode '<string>'
STBAPI-003	NULL stbfig used
STBAPI-004	unknown or incomplete mode '<string>'. Required: 'setup', 'hold' or 'all'
STBAPI-005	valid values for type is 's' (setup) or 'h' (hold)
STBAPI-006	unknown property <string>
STBAPI-007	Function '<string>' is obsolete, no more maintained and will be removed soon, please use functions related to StabilitySlack objects

6.11. API TMA

TMAAPI-001	Unable to get behavioral figure for '<string>'
TMAAPI-002	Name '<string>' matched no connector in blackbox
TMAAPI-003	Invalid direction '<character>'

6.12. API TTV

TTVAPI-001	invalid direction for input signal '<character>' a direction should be 'u' or 'd'.
TTVAPI-002	no clock signal defined, no false <string> paths check done
TTVAPI-003	conflicting node type old:'<character>' new:'<character>'
TTVAPI-004	type '<character>' is unknown

TTVAPI-005	overriding type '<character>' with '<character>'
TTVAPI-006	subtype '<character>' is unknown
TTVAPI-007	<string>: invalid node type '<character>', ttv_AddCustomSignal ignored
TTVAPI-008	nul or negative <string>slope set to %gps for line <string>-><string> <string>
TTVAPI-009	<string>: invalid direction '<string>', timing line creation ignored
TTVAPI-010	unknown file format '<string>' for timing file '<string>'
TTVAPI-011	bad value '<string>' in 'ttv_ExcludeNodeType'
TTVAPI-012	bad value '<string>' in 'ttv_GetMatchingSignal'
TTVAPI-013	could not open current directory
TTVAPI-014	could not load timing figure '<string>'
TTVAPI-015	could not load crosstalk figure
TTVAPI-016	timing figure '<string>' already loaded
TTVAPI-017	could not find timing figure '<string>' with temperature=%g° and voltage=%gV
TTVAPI-018	invalid direction '<string>' a direction should look like \"uu\" (start signal rising, end signal rising) or \"ud \" (start signal rising, end signal falling) for instance.
TTVAPI-019	could not find signal '<string>' in figure
TTVAPI-020	no <string> path was found for path <string> -> <string> <string>
TTVAPI-021	.dtx file for '<string>' is needed to run 'all path' search
TTVAPI-022	available values for parameter 'minmax' are 'min' or 'max'
TTVAPI-023	available values for parameter 'all' are 'critic' or 'all'
TTVAPI-024	available values for parameter 'path' are 'path' or 'access'
TTVAPI-025	function '<string>' is obsolete, use ttv_GetPaths
TTVAPI-026	bad value for parameter 'order'.

TTVAPI-027	bad value for function 'ttv_AutomaticDetailBuild'. Valid values are 'on' and 'off'.
TTVAPI-028	bad value '<string>' in 'ttv_GetLines'
TTVAPI-029	unrecognized property '<string>':'<string>'
TTVAPI-030	unrecognized property '<string>'
TTVAPI-031	unknown parameter value '<string>'
TTVAPI-032	ttv_SetTimingLineDelay: negative slope given (%g), set to 0.1e-12
TTVAPI-033	stability error found at node <string>: data too late by %.1fps
TTVAPI-034	ttv_ProbeDelay: could not find node '<string>'
TTVAPI-035	at least <decimal> node names are required for the function to run
TTVAPI-036	could not open file '<string>' for writing
TTVAPI-037	ttv_SetReportUnit: error, setting digits to <decimal>
TTVAPI-038	ttv_SetReportUnit: unknown unit/setup '<string>'
TTVAPI-039	ttv_SetReportUnit is obsolete, use ttv_SetupReport
TTVAPI-040	ttv_DisplaySimulation: valid values for mode are 'y' or 'n'
TTVAPI-041	ttv_DisplayConnectorToLatchMargin: unknown mode '<string>'
TTVAPI-042	no defined clocks
TTVAPI-043	unknown property <string>
TTVAPI-044	could not load cone file '<string>.cns'
TTVAPI-045	Changing <string> to %g according to UTD
TTVAPI-046	Changing thresholds according to UTD
TTVAPI-047	ttv_ProbeDelay: '<string>' matched <decimal> node names, function run can be very long
TTVAPI-048	unknown node type '<string>'
TTVAPI-049	inconsistancies in given array and axis: <decimal> elements in array vs <decimal> in axis
TTVAPI-050	invalid axis type '<string>'. Only slope-slope, slope-ckslope and slope-capa are available

TTVAPI-051	unknown timing figure flag '<string>'
TTVAPI-052	could not find loaded timing figure '<string>'
TTVAPI-053	Wrong order list format. Should be a list of globalseed-localseed pair values.
TTVAPI-054	unknown connector direction '<character>'

6.13. AVT

AVT-000	Usage: avtdeltoken toolname servername hostname hostid hosttoken pid When using the AVT license server (avtld), avtdeltoken allows to delete a token. Be careful, deleting a token under use will crash the corresponding process.
AVT-001	Character out of [0-9A-Za-z] range in avertecbanner An illegal value has been used within an internal function. Please, report internal errors to Avertec Support.
AVT-002	Resulting size bigger than <decimal> columns not allowed in avertecbanner The formatting size of an object or a text does not fit the avertec banner dimensions. Please, report internal errors to Avertec Support.
AVT-003	'<string>' is not a valid variable Invalid variable in avttools.conf file. Check variable name or suppress wrong variable definition.
AVT-004	File avttools.conf, syntax error line <decimal> File avttools.conf cannot be printed. Correct synthax error (see documentation)
AVT-005	'<string> = <string>' with no effect. Value '<string>' already set with '<string>' A variable has been set twice within avttools.conf, suppress one if possible.
AVT-006	File avttools.conf, multiple declaration of '<string>' A variable has been set several times. Conflicts may occur. Check the declarations of the variable within avttools.conf file and keep only one.
AVT-007	Word <string> too long The string provided to the license server is too long. Use avtinfo to ensure valid value or license file are provided.
AVT-008	String too long The line or message provided to the license server is too long.

AVT-009	`Impossible to delete this token` The token cannot be deleted. Check token name and user rights.
AVT-010	`Usage: avtgenkey toolname vendor server hostid date license type` The license server avt (avtld) failed to handle the license key of avtlicense file.
AVT-011	`Usage: avtinfo tool_name` avtinfo displays information for the avt license server (avt). Ensure a valid tool_name is used.
AVT-012	`Bad Format For Date` The date in the license file is not valid. The license file may be invalid. Check the license file in use.
AVT-013	`Bad Token: <string>` An invalid token has been requested by the license server. Check license file is valid and up to date.
AVT-014	`Option -<character> requires an operand` The specified option is not properly used. Check tool's usage.
AVT-015	`Unrecognised option: -<character>` The specified option is unknown. Check tool's usage.
AVT-016	`Unrecognized Command line` Command line synthax invalid. Check tool's usage.
AVT-017	`Error in opening file <string>` The specified file cannot be open. Check user's rights, and file path.
AVT-018	`Usage: avtreserve tool_name minutes [nb_token]` Display the usage of binary avtreserve. avtreserve is used by the avt license server (avtld).
AVT-019	`Unable to reserve token` Token reservation failed. Check token name and reservation command synthax.
AVT-020	`Usage: avttool tools_name_list` Display usage for monitoring tokens information.
AVT-021	`Run failed` The binary ends with an error. Other messages should bring more information.
AVT-022	`Impossible to give token` The license server was unable to get a pid for the token process.
AVT-023	`License server <string> not responding` The license server is unreachable. Check a valid license server is running.

AVT-024	`Environment variable <string> is not defined` A variable needs to be set in the environment. Avertec documentation supplies the user with the variables legal values.
AVT-025	`...try with server <string>` Another license server has been found and will be tryed. This can occur if the original license server specified by the user was not found or returned an error.
AVT-026	`Bad license server <string>` The license server is not valid or was not responding. Check a valid license server is running.
AVT-027	`<string>` The running tool encountered an error. Specific information supplied by the tool is displayed.
AVT-028	`Token never taken` No token information available for the license server. If a binary (avtinfo, avtdeltoken, or other avt license server utility) has been used, please check the relevant usage.
AVT-029	`Bad token description` The command line synthax used is not valid. Check for relevant avt license server utility usage.
AVT-030	`Bad file name, check autorisation for <string>` The utility failed to open the specified license key or log file.
AVT-031	`Missing part of log file` The avt license server returns an error with the license log file. Usually not a critical error for the running process if avtld is used rather than flexnet.

AVT-033	`AVT-_LICENSE_FILE not set and $AVT_TOOLS_DIR/etc/avtlicense.lic missing.`
	`If errors occur, please check your Flexlm license paths.`
	The flex license file was not found. Use the flexnet license utilities to get information about flexnet license status.

AVT-034	`Flexlm returned error '<decimal>' when <string>.` A generic message displaying error numbers returned by flexnet. Flexnet (or FlexLM) documentation provide some clue to handle surch error number.
AVT-035	`Unable to find Flexlm job corresponding to feature <string>` The specified feature is not valid or no job use it. Flexnet license utilities can provide information about the feature, and the jobs running.
AVT-036	`Please, resolve Flexlm errors before running Avertec programs.`

	Flexnet encounters an error. As long as Flexnet license will not runs properly no licensed Avertec tool can be run. Dealling with license errors is a priority. Check if license file path and license daemon are valid and reachable.
AVT-037	`Log level of variable '<string>' is not a number. It is '<string>'.` Please refer to the configuration of log report.
AVT-038	`Log level of variable '<string>' must be a positive number less or equal to 9. It is '<string>'.` Please refer to the configuration of log report.
AVT-039	`Log variable '<string>' is unknown.` Please refer to the configuration of log report.
AVT-040	`Invalid value '<string>' for configuration variable '<string>', should be 'yes' or 'no'` Please refer to the HiTas Reference Guide.
AVT-041	`Invalid value '<string>' for configuration variable '<string>', please refer to the Reference Guide for correct values` Please refer to the HiTas Reference Guide.
AVT-042	`Invalid file format '<string>' for '<string>', please refer to the Reference Guide for valid formats` Valid formats are 'spice', 'lib', 'tlf3', 'tlf4', 'ttv', 'vhdl', 'verilog', 'dspf', 'spef' and 'inf'.
AVT-043	`Could not find a file matching '<string>'` The specified filename filter does not match any files in accessible directories.
AVT-044	`Multiple settings for configuration variable '<string>': '<string>' is overwritten by '<string>'`
AVT-045	`could not open file '<string>'`
AVT-046	`could not create file '<string>'`
AVT-047	`'avtWarningFilter' has been set to '<string>'`
AVT-048	`Subsequent call to 'avt_LoadFile' will not retain previous definitions of global parameters` The scope of .SCALE, .GLOBAL or other global parameters is limited to the files loaded by a single call of 'avt_LoadFile'. The values of those global parameters are not retained for subsequent calls of 'avt_LoadFile'.
AVT-049	`Incorrect unit specified in '<string>'`
AVT-050	`Incorrect number of jobs <string>`
AVT-051	`received answer from job <decimal>, but is not running !`

AVT-052	job <decimal> is not in state STAT_EXEC_WAIT when received begin packet
AVT-053	job <decimal> is not in state STAT_EXEC_RUN when received data
AVT-054	job <decimal> is not in state STAT_EXEC_RUN when received end
AVT-055	Job <decimal> aborted [<string>]
AVT-056	A system error occured when runing a new job
AVT-057	Statistical result file '<string>' already exist. Using '<string>'.
AVT-058	Error in <string> table. Entry <decimal> correspond to index <decimal>.
AVT-059	Received unhandled command '<string>' from job <decimal>
AVT-061	Invalid binary transfer
AVT-062	Failed to create directory '<string>'
AVT-063	Invalid value '<string>' for configuration variable '<string>' Valid value are '1mW', '100uW', '10uW', '1uW', '100nW', '10nW', '1nW', '100pW', '10pW' and '1pW'.

6.14. BEF

BEF-000	Behavior out format <string> is not a legal format! A Verilog file will be dumped. To describe the format for the generated behavior, you must set 'avtOutputBehaviorFormat' either to 'vhd' for VHDL or to 'vlg' for Verilog.\n
BEF-001	Behavior in format <string> is not a legal format! To describe the format of input behavior, you must set 'avtInputBehaviorFormat' either to 'vhd' for VHDL or to 'vlg' for Verilog.\n

6.15. BEG

BEG-001	<string> Attempt to merge bit vector and single bit <string>
BEG-002	<string> Internal error <string>
BEG-003	<string> Connector declared in, used as out <string>

BEG-004	<string> Null condition <string>
BEG-005	<string> Conflicting vector <string>
BEG-006	<string> Expression and variable has different size <string>
BEG-007	<string> Parser Failure <string>
BEG-008	<string> Conflicting declaration and use <string>
BEG-009	<string> Conflicting bus use <string>
BEG-010	<string> Direction of declaration conflicts with use <string>
BEG-011	<string> Selected signal, vector expression not allowed <string>
BEG-012	<string> Attempt to insert a signal into a defined signal <string>
BEG-013	<string> Vector incompletely defined, made external <string>
BEG-014	<string> Connector declared out, used as in <string>
BEG-015	<string> Bad value for 'BEG_USER_WAY', accepted 'to' or 'downto' <string>
BEG-016	Unknown
BEG-017	<string> Trace: <string>
BEG-018	<string> Convert an In to Out <string>

6.16. BEH

BEH-000	syntax error
BEH-001	combinatory loop: `<string>`
BEH-002	cannot make bdd of empty expression
BEH-003	cannot find terminal `<string>`
BEH-004	illegal use of STABLE attribute
BEH-005	cannot simplify internal signals
BEH-006	cannot make derivatives of expressions
BEH-040	signal `<string>` never assigned

BEH-041	`<string>` has not an empty architecture
BEH-068	port `<string>` has unknwon type
BEH-069	port `<string>` has unknwon mode
BEH-070	unknown time unit
BEH-100	cannot find `<string>`
BEH-107	cannot open result file
BEH-101	<string>: unknown operator
BEH-102	<string>: cannot create empty atom
BEH-103	<string>: cannot build NOT of empty expression
BEH-104	<string>: cannot combine empty expressions
BEH-105	<string>: cannot find terminal
BEH-110	<string>: decompiler called on empty figure
BEH-115	<string>: illegal bit string value : `<character>`
BEH-116	<string>: the same expression cannot be used twice
BEH-119	<string>: empty guard expression: `<string>`
BEH-120	<string>: empty waveform expression: `<string>`
BEH-200	<string>: illegal use of attribute STABLE
BEH-201	<string>: unknown terminal operand `<string>`
BEH-202	<string>: unknown operator `<decimal>`
BEH-203	<string>: empty expression
BEH-199	<string>: Please contact Avertec support
BEH-300	beaux `<string>` not empty
BEH-301	bebus `<string>` not empty
BEH-302	bebux `<string>` not empty
BEH-303	beder not empty

BEH-304	befig `<string>` not empty
BEH-305	begen `<string>` not empty
BEH-306	bemsg `<string>` not empty
BEH-307	beout `<string>` not empty
BEH-308	bequad not empty
BEH-309	bereg `<string>` not empty
BEH-310	biabl not empty
BEH-311	binode not empty
BEH-312	%20s -> <string>
BEH-313	%23s <string>
BEH-315	bevectaux `<string>` not empty
BEH-316	bevectout `<string>` not empty
BEH-317	bevectbux `<string>` not empty
BEH-318	bevectbus `<string>` not empty
BEH-319	bevectreg `<string>` not empty
BEH-320	vectbiabl not empty

6.17. BHL

BHL-000	Internal error <string>

6.18. BGL

BGL-000	Internal error <string>
BGL-001	`<string>` line <decimal> :`<string>` is incompatible with the entity name
BGL-002	`<string>` line <decimal> :bad entity declaration

BGL-003	`<string>` line <decimal> :bad port clause declaration
BGL-004	`<string>` line <decimal> :port `<string>` already declared
BGL-005	`<string>` line <decimal> :illegal port declaration `<string>` (mode, type, guard mark)
BGL-006	`<string>` line <decimal> :bad port declaration
BGL-007	`<string>` line <decimal> :`<string>` is incompatible with the architecture name
BGL-008	`<string>` line <decimal> :bad architecture declaration
BGL-009	`<string>` line <decimal> :illegal declaration
BGL-010	`<string>` line <decimal> :signal `<string>` already declared
BGL-011	`<string>` line <decimal> :illegal signal declaration `<string>` (type, guard mark)
BGL-012	`<string>` line <decimal> :component `<string>` already declared
BGL-013	`<string>` line <decimal> :instance `<string>` already declared
BGL-014	`<string>` line <decimal> :`<string>` unknown component
BGL-015	`<string>` line <decimal> :illegal usage of implicit port map description
BGL-016	`<string>` line <decimal> :`<string>` unknown local port
BGL-017	`<string>` line <decimal> :`<string>` unknown port or signal
BGL-018	`<string>` line <decimal> :illegal concurrent statement
BGL-019	`<string>` line <decimal> :bad signal association
BGL-020	`<string>` line <decimal> :null array not supported
BGL-021	`<string>` line <decimal> :illegal constraint in declaration of type
BGL-022	`<string>` line <decimal> :signal `<string>` used out of declared range
BGL-023	`<string>` line <decimal> :width or/and type mismatch
BGL-024	`<string>` line <decimal> :port `<string>` connected to more than one signal

BGL-025	`<string>` line <decimal> :can only assign to/from an external connector
BGL-026	`<string>` line <decimal> :instance <string> mismatch with the model
BGL-027	`<string>` line <decimal> :Unhandled feature
BGL-028	`<string>` line <decimal> :<string>
BGL-029	Cannot open result file
BGL-030	Cannot continue further more.
BGL-031	`<string>` line <decimal> :Syntax error
BGL-032	Too many errors. Cannot continue further more
BGL-033	File does not exist : <string>
BGL-034	Abnormal parsing for : <string>
BGL-035	Connection missing on port `<string>`
BGL-036	Consistency checks will be disabled
BGL-038	Internal error <string> while executing <string>
BGL-037	`<string>` line <decimal> :<string>

6.19. BVL

BVL-000	Internal error <string>
BVL-001	`<string>` line <decimal> `<string>` is incompatible with the entity name
BVL-002	`<string>` line <decimal> Bad entity declaration
BVL-003	`<string>` line <decimal> Bad port clause declaration
BVL-004	`<string>` line <decimal> Port `<string>` already declared
BVL-005	`<string>` line <decimal> Illegal port declaration (mode, type, kind)
BVL-006	`<string>` line <decimal> Bad port declaration
BVL-007	`<string>` line <decimal> `<string>` is incompatible with the architecture name

BVL-008	`<string>` line <decimal> Bad architecture declaration
BVL-009	`<string>` line <decimal> Illegal declaration
BVL-010	`<string>` line <decimal> Signal `<string>` already declared
BVL-011	`<string>` line <decimal> Illegal signal declaration (type, kind)
BVL-012	`<string>` line <decimal> `<string>` unknown port or signal
BVL-013	`<string>` line <decimal> Illegal concurrent statement
BVL-014	`<string>` line <decimal> Label `<string>` already declared
BVL-015	`<string>` line <decimal> `<string>` is incompatible with the block's label
BVL-016	`<string>` line <decimal> Input port `<string>` cannot be assigned
BVL-017	`<string>` line <decimal> Illegal unguarded signal assignment for `<string>`
BVL-018	`<string>` line <decimal> Illegal guarded signal assignment `<string>`
BVL-019	`<string>` line <decimal> Some choices missing in the selected signal assignment
BVL-020	`<string>` line <decimal> Output port `<string>` cannot be read
BVL-021	`<string>` line <decimal> Duplicate choice in selected signal assignment
BVL-022	`<string>` line <decimal> Illegal use of OTHERS in selected signal assignment
BVL-023	`<string>` line <decimal> Null array not supported
BVL-024	`<string>` line <decimal> Incompatible constraint and type
BVL-025	`<string>` line <decimal> Illegal assignment of `<string>` (widths mismatch)
BVL-026	`<string>` line <decimal> Signal `<string>` used out of declared range
BVL-027	`<string>` line <decimal> Width or/and type mismatch
BVL-028	`<string>` line <decimal> Signal `<string>` assigned more than once

BVL-029	`<string>` line <decimal> Signal `<string>` never assigned
BVL-030	`<string>` line <decimal> Illegal condition on signal `<string>`
BVL-031	`<string>` line <decimal> BEPOR type is unknown
BVL-032	`<string>` line <decimal> `<string>` is not a bit string litteral
BVL-033	`<string>` line <decimal> Bad generic declaration
BVL-034	`<string>` line <decimal> Bad generic element
BVL-035	`<string>` line <decimal> `<string>`: when expression must be a constant
BVL-036	`<string>` line <decimal> Illegal generic declaration (type, kind)
BVL-037	`<string>` line <decimal> Illegal constant declaration (type, kind)
BVL-038	`<string>` line <decimal> Illegal use of attribute STABLE on `<string>`
BVL-039	`<string>` line <decimal> Different delays not supported on waveforms
BVL-040	`<string>` line <decimal> Syntax error
BVL-041	Too many errors. Cannot continue further more
BVL-042	`<string>` Error line <decimal> : <string>

6.20. CBH

CBH-000	Internal Error <string>
CBH-001	<string>: Possible cause library not charged
CBH-002	<string> needs a file as argument
CBH-003	<string>

6.21. CGV

CGV-001	Internal error #<decimal>
CGV-002	Internal warning #<decimal>

CGV-003	<string>:%ld: unknown internal error <decimal>
CGV-004	could not open file '<string>'
CGV-005	could not parse file '<string>'

6.22. CNS

Error messages description not available yet.

6.23. EFG

EFG-001	Internal error: could not find node '<string>'
EFG-002	could not find a VDD node in generated spicedeck
EFG-003	could not find a GROUND node in generated spicedeck
EFG-004	Fatal error #<decimal>
EFG-005	Internal: spisig is NULL
EFG-006	Internal: signal <string> is not in figure <string>
EFG-007	signal <string> appear with different slopes
EFG-008	Internal: can't add global alim for sig <string>

6.24. EQT

EQT-001	Unknown unit '<string>': assuming factor 1
EQT-002	can't evaluate '<string>'
EQT-003	too much user defined functions
EQT-004	<string> can't be done yet on ternary or .func operators
EQT-005	Internal: empty node can't make ABL
EQT-006	Internal: integer value in ABL to make
EQT-007	Internal: node is NULL
EQT-008	Internal: Error in function <string>

EQT-009	Mismatching number of parameters for function '<string>'
EQT-010	Undefined function '<string>'

6.25. GNS

GNS-001	instance <string> already exist in figure <string>
GNS-002	instance model is the figure <string> itself
GNS-003	connector number discrepancy between figure <string> and instance <string> in figure <string>
GNS-004	Internal error <decimal>
GNS-005	Internal warning <decimal>
GNS-006	<string>: can't find transistor '<string>' in model
GNS-007	Invalid symmetry detected for instance '<string>' (<string>) on connector '<string>' and ? (<string> and <string>)
GNS-008	no FCL match for `<string>' (instance:'<string>' model:`<string>')
GNS-009	Vector ordering failed on instance '<string>' (<string>) who might be used as \"exclude\" . Try to add 'NoOrdering' in the 'GnsFlags' variable
GNS-010	Found an alim linked to a 'not' alim : <string> and <string> in <string>
GNS-011	can't retrieve blackbox instance `<string>' (`<string>')
GNS-012	can't retrieve blackbox connector `<string>.<string>' (`<string>')
GNS-013	can not find generic variable '<string>', assumed value 0
GNS-014	could not find black box '<string>' in circuit
GNS-015	single connector '<string>' (instance '<string>') is linked to a vector connector <string>
GNS-016	vector connector '<string>' (instance '<string>') is linked to a single connector <string>
GNS-017	could not find correspondance for transistor '<string>'
GNS-018	found a transistor with no name

GNS-019	could not find correspondance for signal '<string>'
GNS-020	can't drive '<string>' type in function call
GNS-021	can't drive pointer type in function call
GNS-022	can't find figure '<string>'
GNS-023	Could not write file <string>.gns
GNS-024	Cannot create file <string>.gen
GNS-025	Can't redirect GENIUS output to '/dev/null'
GNS-026	<string>:<decimal>: division by zero
GNS-027	forbidden operators 'mod', 'rem', '**'
GNS-028	variable <string> not found
GNS-029	<string>:<decimal>: IF forbidden for GNS
GNS-030	a variable name was expected for instance '<string>', found a number
GNS-031	generic variable '<string>' not define
GNS-032	Value of '<string>' for instance '<string>' must be <decimal>, actually %ld
GNS-033	There should be at least one instance of model '<string>' with <string>=<decimal>
GNS-034	More than 1 unknown generic variable
GNS-035	variable '<string>' is not defined yet
GNS-036	<string>: can't go thru '<string>'
GNS-037	<string>: can't find instance '<string>' in model
GNS-038	no corresponding transistor for <string>
GNS-039	*** <decimal> error(s) detected, I can't get farther!! ***
GNS-040	for model instance '<string>' can not evaluate left or right bound for connector '<string>' l=<decimal> r=<decimal>
GNS-041	no correspondance found for signal '<string>(<decimal>)

GNS-042	Error: <string>
GNS-043	No model file in library
GNS-044	Cannot open model file <string>
GNS-045	no model <string> found
GNS-046	unknown connector (<string>) declared in symmetric connector list
GNS-047	invalid mix of vector and bit
GNS-048	unknown connector (<string>) declared in coupled connector list
GNS-049	Could not find subfigure '<string>' in file '<string>'
GNS-050	Spice file <string> should be a flat transistor netlist
GNS-060	other errors follow...
GNS-061	in model '<string>', connector '<string>' of unexistant instance '<string>' must not be linked the model interface
GNS-062	in model '<string>', if connector '<string>' of unexistant instance '<string>' is not used, it must be linked to a supply
GNS-063	in model '<string>', connector '<string>' of unexistant instance '<string>' must not be linked to another unexistant instance connector
GNS-064	Inconsistancies found for instance '<string>' of model '<string>'
GNS-065	Connector '<string>' of instance mismatched with connector '<string>' of model
GNS-066	Inconsistancies found for connector '<string>' of instance '<string>' with model '<string>'
GNS-067	Connector number mismatched
GNS-068	While parsing correspondance tables, line <decimal>, transistor out of context
GNS-069	While parsing correspondance tables, line <decimal>, could not find transistor '<string>' in original netlist
GNS-070	While parsing correspondance tables, line <decimal>, signal out of context
GNS-071	While parsing correspondance tables, line <decimal>, could not find signal '<string>' in original netlist

GNS-072	While parsing correspondance tables, line <decimal>, instance out of context
GNS-073	While parsing correspondance tables, line <decimal>, could not find instance correspondance '<string>' table
GNS-074	While parsing correspondance tables, line <decimal>, could not find instance correspondance table
GNS-075	While parsing correspondance tables, line <decimal>, variables out of context
GNS-076	While parsing correspondance tables, line <decimal>, dictionary entry without dictionary mode
GNS-077	While parsing correspondance tables, line <decimal>, too many entries in dictionary
GNS-078	While parsing correspondance tables, line <decimal> ignored
GNS-079	Could not find instance in model to start search with There should be at least one real instance in the model
GNS-080	could not find instance '<string>' in GNS toplevel instance
GNS-081	out of bounds with <string> and <string> started from <string>
GNS-082	computing error for index=%ld end=<decimal>
GNS-083	parameter discrepancy between <string> and <string>
GNS-084	infinite loop on <string>
GNS-085	transistor in loop is forbidden
GNS-086	<string>:<decimal>: GNS ignored expansed 'FOR' driven by variable '<string>'
GNS-087	several signals connected to connector <string> of instance '<string>'
GNS-088	vector connector <string> connected to single signal <string>
GNS-089	connector '<string>' with several signals
GNS-090	can not compute destination connector index for '<string>', the high bound of signal '<string>' is not known yet
GNS-091	vector connector '<string>' connected to one bit signal

GNS-092	bit number <decimal> is out of bounds for signal <string>
GNS-093	bit number <decimal> is out of bounds for connector <string>
GNS-094	transistor type/parameter discrepancy (<string>)
GNS-095	<string>:<decimal>: too many parameters for transistor '<string>'
GNS-096	<string>:<decimal>: a number was expected for instance '<string>', found a variable name
GNS-097	<string>:<decimal>: A positive non nul number was expected for instance '<string>'
GNS-098	<string>:<decimal>: unknown transistor parameter '<string>'
GNS-099	instance type/parameter discrepancy (<string>)
GNS-100	too many connectors in instance '<string>'
GNS-101	connectors <string> and <string> mismatch for instances <string> and <string>
GNS-102	not enough connectors in instance '<string>'
GNS-103	parameter discrepancy between instances <string> and <string>
GNS-104	<string>:<decimal>: a variable name was expected for instance '<string>', found a number
GNS-105	could not find generic variable '<string>' in entity variable list
GNS-106	several signals connected to connector <string> of instance <string>
GNS-107	no search done on connector '<string>' signal '<string>', model must be a connexe graph
GNS-108	no search done on signal '<string>(%ld)', model must be a connexe graph
GNS-109	can't compute connector bound for connector '<string>' certainly while building a fake instance or transistor
GNS-110	width mismatch between connector '<string>(<decimal>..<decimal>)' and signal '<string>(<decimal>..<decimal>)'
GNS-111	Index <decimal> is out of range for signal <string> (<decimal>..<decimal>)
GNS-112	Index <decimal> is out of range for signal <string>

GNS-113	<string>:<decimal>: negative vector bound computed for expression, values are <decimal> and <decimal>
GNS-114	no search done on connector '<string>(%ld)', model '<string>' must be a connexe graph
GNS-115	connector <string>.<string> is in coupled list but has no symetric
GNS-116	no symmetry found for connector <string> in coupled connector list
GNS-117	coupling won't work with vectors ... yet...
GNS-118	can not find coupled connector for '<string>'
GNS-119	could not find connector '<string>(<decimal>)' for instance '<string>'
GNS-120	while swapping <string> and <string>, one of the connector did not have coupled connector list while the other has
GNS-121	Internal limitation. too much symmetric informations. Actual limit is <decimal>
GNS-122	same signal in different symmetry list
GNS-123	same signal in different coupled list
GNS-124	<string>:<decimal>: array of signal '<string>' out of bounds with model line <decimal>
GNS-125	<string>:<decimal>: array doesn't match for '<string>' (line model <decimal>)
GNS-126	<string>:<decimal>: connector '<string>' is missing in left side of instance
GNS-127	<string>:<decimal>: too many connections in Port Map. Component line <decimal>
GNS-128	<string>:<decimal>: not enough connections in Port Map. Component line <decimal>
GNS-129	<string>:<decimal>: <string> already excluded
GNS-130	<string>:<decimal>: instance <string> doesn't exist in architecture <string> of <string>
GNS-131	<string>:<decimal>: INPUT '<string>' cannot be connected with OUTPUT '<string>'

GNS-132	<string>:<decimal>: OUTPUT '<string>' cannot be connected with INPUT '<string>'
GNS-133	<string>:<decimal>: OUTPUT '<string>' cannot be connected with INPUT '<string>'
GNS-134	<string>:<decimal>: only one variable authorized in a 'for' expression. Use Hierarchy!
GNS-135	<string>:<decimal>: forbidden operator '<string>' on variable
GNS-136	<string>:<decimal>: division by zero could appear
GNS-137	<string>:<decimal>: a generic isn't needed by model line <decimal>
GNS-138	<string>:<decimal>: a generic is needed by model line <decimal>
GNS-139	<string>:<decimal>: not enough variables in component. model ends at line <decimal> with '<string>'
GNS-140	<string>:<decimal>: type '<string>' doesn't match with model line <decimal>
GNS-141	<string>:<decimal>: too many variables in component. model ends at line <decimal> with '<string>'
GNS-142	<string>:<decimal>: a port isn't needed by model line <decimal>
GNS-143	<string>:<decimal>: a port is needed by model line <decimal>
GNS-144	<string>:<decimal>: a bit is expected for '<string>' line <decimal> of model
GNS-145	<string>:<decimal>: size of '<string>' mismatches with model line <decimal>
GNS-146	<string>:<decimal>: a vector is expected for '<string>' line <decimal> of model
GNS-147	<string>:<decimal>: predefined rule '<string>': IN Grid, INOUT Source, INOUT Drain, IN Bulk
GNS-148	<string>:<decimal>: There is no vector in predefined rule '<string>'
GNS-149	<string>:<decimal>: The predefined entity '<string>' don't have a generic
GNS-150	<string>:<decimal>: '<string>' must be an external connector
GNS-151	<string>:<decimal>: predefined rule '<string>' impossible to use as a transistor name as an entity name

GNS-152	<string>:<decimal>: more than one action defined for entity '<string>'
GNS-153	<string>:<decimal>: no architecture defined for entity '<string>'
GNS-154	<string>:<decimal>: too many (><decimal>) transistors in architecture '<string>' to start FCL
GNS-155	<string>:<decimal>: Generate forbidden with transistor in model <string>
GNS-156	<string>:<decimal>: Blackbox with generic variables not implemented yet
GNS-157	<string>:<decimal>: GNS can't start on a pure transistor netlist '<string>'
GNS-158	you must explicitly specify the vector range for '<string>'
GNS-159	<string>:<decimal>: negative vector index
GNS-160	<string>:<decimal>: vector direction error
GNS-161	no figure model <string> found
GNS-162	<string>:<decimal>: <string> instance '<string>' can't be found
GNS-163	<string>:<decimal>: '<string>' used several times, primary use line <decimal> in file <string>
GNS-164	<string>:<decimal>: variable '<string>' not defined
GNS-165	<string>:<decimal>: <string>
GNS-166	Cannot read file <string>
GNS-167	Error executing TCL funcion '<string>': <string>
GNS-168	internal: unknown transistor model '<string>'

6.26. GSP

GSP-001	can't get up expression for cone <string>
GSP-002	can't get down expression for cone <string>
GSP-003	Internal: unknown transition for out signal <string>

GSP-004	can't propagate thru cone <string>
GSP-005	can't find global path transfer
GSP-006	Internal: can't get sup expr, cone list is NULL
GSP-007	Internal: can't get sup expr, cone is NULL
GSP-008	Internal: can't get sup expr for cone <string>
GSP-009	Internal: can't get input/ouput connector to analyse instance <string>

6.27. INF

INF-001	<string>L and W not required The parameters L and W won't be use because they have no meaning in this declaration.
INF-002	<string>unrecognized token '<string>', should be <string> The expected token was not found.
INF-003	<string>unrecognized token '<string>' The token is unknown and can't be handled.
INF-004	<string>invalid value '<string>', should be <string> An invalid value has been encountred.
INF-005	<string>unknown section '<string>', ignoring section The section is unknown and can't be handled.
INF-006	<string>unknown direction '<string>', ignoring section The connector direction is unknown and can't be handled.
INF-007	<string>clock '<string>' is not delared yet A clock signal is referenced but has not been declared yet.
INF-008	<string>no period specified for clock '<string>' or no default period It's impossible to find a period to associate with the clock. Either the default period or the clock period should be defined.
INF-009	<string>:<decimal>: syntax error near '<string>' A syntax error occured when parsing file. Either a signal name matchs a syntax token or the token is unknown. In the first case, consider enclosing the signal name with quote.

INF-010	<string>:<decimal>: syntax error in regular expression An invalid regular expression have been detected.
INF-011	no figure name given, guessing it is '<string>' The information file should begin with the figure name associated with the informations. If the name is not specified, a figure name will be guessed from the file name. This could lead to errors if the guessed name is wrong.
INF-012	can not open '<string>' information file An error occured when trying to read the information file.
INF-013	<string>information on signal <string> already read -- ignored
INF-014	<string>information on signal '<string>' has already been set elsewhere -- overriding with inf values A default value has been set by the netlist spice deck and will be shadowed by a new value who has more priority.
INF-015	<string>information '<string>' for '<string>' has already been set to '<string>' elsewhere -- overriding with inf value '<string>' A default value has been set by the netlist spice deck and will be shadowed by a new value who has more priority.
INF-016	<string>information '<string>' for '<string>' has already been set elsewhere -- overriding with inf values A default value has been set by the netlist spice deck and will be shadowed by a new value who has more priority.
INF-017	<string>information '<string>' for '<string>' has already been set to <decimal> elsewhere -- overriding with inf value <decimal> A default value has been set by the netlist spice deck and will be shadowed by a new value who has more priority.
INF-018	could not create file '<string>' An error occured when trying the open the information file in write mode.
INF-019	<string>slope defined for pin '<string>' <string> is too low, set to <string>
INF-020	<string>invalid check type ('<string>') for 'NoCheck' section
INF-021	unknown inf section '<string>'
INF-022	<string>unknown characteristic '<string>', ignoring this entry The signal characteristic is unknown and can't be handled.
INF-023	name '<string>' doesn't match any <string> in circuit
INF-024	<string>incompatible tokens given for 'directives' section

	Using "... with sig rising|falling" or "... before|after sig up|down" is forbidden.
INF-025	<string>invalid type '<string>', ignoring this entry The given type does not apply to this section and can't be handled.
INF-026	<string>out of range probability value: %g The probability value must be a value ranging from 0 to 1.
INF-027	invalid marking <string>

6.28. LOG

LOG-000	Internal Error <string>
LOG-001	Maximum number of variables for BDDs reached
LOG-002	BDD's system not enough memory...

6.29. MBK

MBK-000	connector <string> not found in instance <string>
MBK-001	connector number mismatch beetwen instance <string> and figure <string>
MBK-002	can't evaluate '<string>', assuming 0
MBK-003	transistor length is null
MBK-004	transistor width is null
MBK-005	conflicting power supply on node '<string>' keeping %gv (other is %gv)
MBK-006	Can't flatten figure <string> because RC cache is active
MBK-007	lofigchain is missing on lofig <string>
MBK-008	Null CTC on signal %ld (1) in instance <string>
MBK-009	flattenlofig: connector <string> exists only in instance <string>
MBK-010	flattenlofig: connector <string> in instance <string>: number of physical nodes differ
MBK-011	Null CTC on signal %ld (2) in instance <string>

MBK-012	figure <string> not empty (type=%ld)
MBK-013	unflat error: no supply ground
MBK-014	unflattenlofig: connector number inconsistency between model '<string>' and instance '<string>'
MBK-015	duplosig impossible: signal %ld already exist
MBK-016	the radical <string> is already used in a vector
MBK-017	the radical <string> has a spurious vectorized value <string> (<string>)
MBK-018	figure '<string>': transistor '<string>' appears several times
MBK-019	can't open file <string>
MBK-020	addlofig impossible: figure <string> already exists
MBK-021	addlomodel impossible: model <string> already exists
MBK-022	illegal transistor type: %ld
MBK-023	addloins impossible: instance <string> already exist in figure <string>
MBK-024	addloins impossible: instance model is the figure <string> itself
MBK-025	addloins impossible: connector number discrepancy between figure <string> and instance <string> in figure <string>
MBK-026	addlocon impossible: connector <string> already exists in figure <string>
MBK-027	addlocon impossible: bad direction <character> in figure <string>
MBK-028	addlosig impossible: signal %ld already exist in figure <string>
MBK-029	getloins impossible: instance <string> doesn't exist in figure <string>
MBK-030	getlotrs impossible: transistor <string> doesn't exist in figure <string>
MBK-031	getlocon impossible: connector <string> doesn't exist in figure <string>
MBK-032	getlosig impossible: signal %ld doesn't exist in figure <string>

MBK-033	viewlo: empty list of figure
MBK-034	setsigsize() impossible: BKSIG not NULL
MBK-035	Conflict power supply on signal: <string>
MBK-036	u is not a square matrix
MBK-037	u and l matrix size mismatch
MBK-038	could not find a pivot
MBK-039	singular matrix given
MBK-040	matrix a and b size can not allow '*' operation
MBK-041	matrix a and b size can not allow '-' operation
MBK-042	matrix solve order has not been computed yet
MBK-043	matrix a and sol size can not allow solve operation
MBK-044	unreducable matrix given
MBK-045	fatal mbkalloc error: not enough memory when trying to allocate %lu bytes, top= %luKb
MBK-046	fatal mbkrealloc error: not enough memory
MBK-047	Can't open file <string> because too big
MBK-048	mbksysfopen: bad value for access
MBK-049	file <string> opened, file <string> ignored
MBK-050	file <string> opened for writting, file <string> is deleted
MBK-051	Cannot evaluate parameter <string>='<string>'<string>
MBK-052	Cannot evaluate expression '<string>=<string>' in figure '<string>': variable<string> <string> <string> unknown, assuming 0
MBK-053	Undefined <string> parameter '<string>' in subcircuit '<string>'
MBK-054	Error #<decimal> in avt communication protocol (<string>)
MBK-055	IP port <decimal> already in use. Waiting...
MBK-056	Can't activate master process in a slave process

MBK-057	not enought communication slot to handle new input connection !
MBK-058	Non slot to handle new input connection !
MBK-060	some communication slot are still active, but all process are finished !
MBK-061	a lofig is not allowed in the slave process !
MBK-062	a cnsfig is not allowed in the slave process !
MBK-063	a ttvfig is not allowed in the slave process !
MBK-064	a abnormal end-of-file was encountered !
MBK-065	error '<string>' while reading file '<string>'
MBK-066	error '<string>' while writing to file '<string>'
MBK-067	missing end of encryption marker in file '<string>'
MBK-068	Failed to transfer file '<string>'
MBK-069	Environment variable '<string>' is not set
MBK-070	Expression '<string>' returned negative value for <string>
MBK-071	Diode model evaluation returned <string> capacitance value
MBK-072	Resistive paths found between power supplies. If resistor names do not appear, consider using 'avt_config avtSpiKeepNames resistance'.

6.30. MCC

MCC-000	Unknown transition in mcc_generatesimgate
MCC-001	Default values assumed for vbs
MCC-002	Can't generate params for <string> (L = %ldn, W = %ldn)
MCC-003	Can't find vds for a degraded transistor
MCC-004	Hspice bsim3v3 model used with invalid ACM value (<decimal>)
MCC-005	Level = 53, default ACM=10 used!!!
MCC-006	Level = 49, default ACM=0 used!!!
MCC-007	Invalid value of parameter MOBMOD, default value used!

MCC-008	Computation of Vdsat failed! Default computation of Vdsat used!
MCC-009	Negative Leff for <string> (L = %ldn, W = %ldn)! default value assumed: Leff = %ldn
MCC-010	Negative Weff for <string> (L = %ldn, W = %ldn)! default value assumed : Weff = %ldn
MCC-011	Computation of Vdsat failed! Bad value for RDSMOD = 0 assumed
MCC-012	Technofile <string> to get doesn't exist!
MCC-013	Technofile <string> to delete doesn't exist
MCC-014	No model <string> in technofile <string>, can't get index!
MCC-015	No model <string> in technofile <string>, can't get model name!
MCC-016	No model <string> in technofile <string>, can't get XL!
MCC-017	No model <string> in technofile <string>, can't get XW!
MCC-018	Can't initialise diode model '<string>' parameters, unknown diode's level: <decimal>
MCC-019	Computation of VDDdeg for model <string> which is a PMOS transistor!
MCC-020	Issue occured while computing VDDdeg for model <string> (L=%g,W=%g), default value 'VDD (%g) - VTH (%g)' assumed (%g)
MCC-021	Computation of VSSdeg for model <string> which is a PMOS transistor!
MCC-022	Issue occured while computing VSSdeg for model <string> (L=%g,W=%g), default value 'VTH' assumed (%g)
MCC-023	mcc_dio_calcCDEP invalid value for TLEV! TLEV = 0 assumed
MCC-024	Issue occured while computing VTI_nmos
MCC-025	Issue occured while computing VTi_pmos
MCC-026	Technofile <string> doesn't exist, can't addmodel!
MCC-027	Model <string> doesn't exist!
MCC-028	No model <string> (<string> case L = %ldn, W = %ldn) for <string> The technology file does not contain the model or the device size is out of the model ranges. It may also be necessary to load the techonology file before using CPE.

MCC-029	No model for diode model <string>!
MCC-030	Closest model assumed <string> (lmin = %ldn, lmax = %ldn, wmin = %ldn, wmax = %ldn)
MCC-031	Positive vbs (%g) for transistor <string> (model <string> L=%gu W= %gu)
MCC-032	Negative vbs (%g) for transistor <string> (model <string> L=%gu W= %gu)
MCC-033	<string> is used for best corner!
MCC-034	<string> is used for worst corner!
MCC-035	Could not characterize transistor model '<string>' with given PVT: vdd=%gv temp=%g Check if the PVT suits your transistor model
MCC-036	Unknown transistor instance specific parameter '<string>'
MCC-037	Failed to evaluate <string>=<string>
MCC-038	Cannot evaluate model parameter <string>='<string>'<string>
MCC-039	Extra transistor instance specific parameter <string>=%g found in subckt '<string>', found value used
MCC-040	Extra transistor instance specific parameter M=<decimal> : technology check won't work properly
MCC-041	<string> found on transistor <string> in technology file subckt '<string>'
MCC-042	Can't determine transistor model type for '<string>' using simToolModel setting '<string>'. Guessing '<string>'. Please set correct simToolModel ('<string>'?).
MCC-043	Can't open transistor model definition file <string>.
MCC-044	Parse error line <decimal> file <string>.
MCC-045	Can't parse transistor definition file. Search in <string> : <string>

6.31. MGL

MGL-001	syntax error line <decimal>
MGL-002	illegal vector range specification at line <decimal>

The bounds specified for a bussed signal are not numerical values.

MGL-003	`port '<string>' already declared at line <decimal>` An external connector with the same name has already been declared in the module.
MGL-004	`net '<string>' already declared at line <decimal>` A net with the same name has already been declared in the module.
MGL-005	`net '<string>' used out of declared range at line <decimal>` A bit a range of bits is refered to for the given net despite being beyond the bounds specified in the net declaration.
MGL-006	`width or/and type mismatch at line <decimal>` A port connection in an instantiation does not match the port specification of the module being instantiated.
MGL-007	`escaped vector '<string>' not declared in strict bit ascending/descending order at line <decimal>` Only occurs if avtStructuralVerilogVector is set to yes. Escaped vectors declared as individual bits can be handled correctly only if the bits are declared in a strict ascending or descending order without any gaps.
MGL-008	`escaped vector '<string>' not declared with all bits together at line <decimal>` Only occurs if avtStructuralVerilogVector is set to yes. Escaped vectors declared as individual bits can be handled correctly only if the bits are declared together.
MGL-009	`Missing external connector for external signal '<string>' driving verilog netlist '<string>'` Occurs when driving an incoherent verilog netlist.
MGL-010	`Cannot open file to drive verilog netlist '<string>'` File I/O error, check disk space and privileges.

6.32. SDC

SDC-000	`Command '<string>' not supported`
SDC-001	`Line <decimal>: Ambiguous list of connectors of the option \"-through\":` `Consider an AND between the first and the second connectors.`
SDC-002	`Line <decimal>: clock reference '<string>' is unknown.`
SDC-003	`Line <decimal>: Reference clock is virtual.`
SDC-004	`set_load options -min, -max, -substract_pin_load, -pin_load, -wire_load not supported`

6.33. SIM

SIM-001	can not open file <string>
SIM-002	can not close file <string>
SIM-003	can not execute <string> or <string> exit with non zero value <decimal>
SIM-004	read error for net '<string>' in file <string>
SIM-005	previous read error
SIM-006	simToolModel '<string>' is unknown

6.34. SLIB

Error messages description not available yet.

6.35. SPF

SPF-001	Invalid pin type '<string>' at line <decimal> The pintype field does not contain a valid [IiOoSsBbJjXx] character.
SPF-002	Syntax error at line <decimal>, token '<string>' The given line does not conform to legal SPF syntax. If token is 'CR', there are probably missing elements on the line.
SPF-003	Undeclared node '<string>' (line <decimal>) The given node is not declared in the current net.
SPF-004	Net '<string>' with mismatching total capacitance (total=%gpf, sum=%gpf) The total capacitance already associated with a net in a design does not match the value given for the total capacitance for the net in the SPF file.
SPF-005	Unsupported divider, using '/' (line <decimal>) A missing or invalid character used to specify the hierarchy divider.
SPF-006	Unsupported delimiter, using ':' (line <decimal>) A missing or invalid character used to specify the name delimiter.
SPF-007	Unknown syntax for BUSBIT section (line <decimal>) Illegal syntax used to specify bus delimiters in the BUSBIT section.

SPF-008	`Undefined design entity name` The name of the design to be annotated is not specified in the SPF file.
SPF-009	`Entity '<string>' does not exist` The design entity specified for annotation does not exist. Either the netlist has not been loaded yet or the names do not match.
SPF-010	`Unknown capacitance unit '<string>', assuming 'pf' (line <decimal>)` Illegal specification of capacitance unit. Legal values are ff, pf, nf, uf, mf, kf. The final 'f' is optional and case is irrelevant.
SPF-011	`Unknown resistance unit '<string>', assuming none (line <decimal>)` Illegal specification of resistance unit. Legal values are f, p, n, u, m, k. The case is irrelevant.
SPF-012	`Device or instance '<string>' cannot be found (line <decimal>)` The object specified for connection to the parasitic network does not exist.
SPF-013	`Connector of transistor '<string>' named '<string>' cannot be found (line <decimal>)` The name used to specify the transistor connector is invalid. Legal names are g, d, s, b.
SPF-014	`Connector of transistor '<string>' named '<string>' might be connected to the wrong signal (line <decimal>)` There may be an incoherance in the parasitic file.
SPF-015	`Instance '<string>' cannot be found (line <decimal>)` The instance specified for connection to the parasitic network does not exist.
SPF-016	`Connector of instance '<string>' named '<string>' cannot be found (line <decimal>)` The name used to specify the instance connector is invalid.
SPF-017	`Connector '<string>' cannot be found (line <decimal>)` The external pin specified for connection to the parasitic network does not exist.
SPF-018	`Connector named '<string>' is on the wrong equipotential (line <decimal>)` There may be an incoherance in the parasitic file.
SPF-019	`Negative capacitance found (line <decimal>)` A negative value was used to specify a capacitance, this is ignored.
SPF-020	`Negative resistance found (line <decimal>)` A negative value was used to specify a resistance, this is ignored.
SPF-021	`<decimal> signal(s) missing in the netlist`

	Signals specified in the parasitic file do not exist in the netlist.
SPF-022	`Don't know how to handle connector '<string>' for resistor annotation (line <decimal>)` The handled names for resistor connectors are 'pos', 'neg' , '1' and '2'.
SPF-023	`Signal '<string>' missing in the netlist (line <decimal>)` Signal can not be found in netlist to annotate.
SPF-024	`Failed to open file '<string>'` The dspf file was not found.
SPF-025	`RC Network declaration for signal '<string>' continued at line <decimal>` Multiple consecutive declarations for the signal have been detected. They will be merged.
SPF-026	`Don't know how to handle capacitance/diode connector '<string>' for capacitance/diode annotation (line <decimal>)` The handled names for capacitance and diode connectors are '1' and '2'.
SPF-027	`No declaration of crosstalk capacitance node '<string>' (line <decimal>), associated with net '<string>'`
SPF-028	`Node '<string>' (line <decimal>) could not be created, coupling capacitance to this node will be ignored` A coupling capacitance to a non existant signal in the netlist will be ignored.

6.36. SPE

SPEF-001	`can't use RC cache with compressed file.`
SPEF-002	`could not find figure '<string>' specified into spef file '<string>'`
SPEF-003	`<string>:<decimal>: <string>`
SPEF-004	`<string>:<decimal>: Syntax Error near '<string>'`
SPEF-005	`can't get current file position.`
SPEF-006	`can't set current file position.`
SPEF-007	`only one spef cache can be used` `figure '<string>' won't be cached`
SPEF-008	`can't open file '<string>'`
SPEF-009	`<string>:<decimal>: signal '<string>' does not exist in netlist`

SPEF-010	<string>:<decimal>: transistor connector '<string>' of '<string>' can not be found
SPEF-011	<string>:<decimal>: object named '<string>' can not be found in transistors, connectors or instances
SPEF-012	<string>:<decimal>: resistor connector '<string>' of '<string>' can not be found
SPEF-013	<string>:<decimal>: capacitance connector '<string>' of '<string>' can not be found
SPEF-014	<string>:<decimal>: diode connector '<string>' of '<string>' can not be found
SPEF-015	<string>:<decimal>: delimiter '<character>' not found in name '<string>'
SPEF-016	<string>:<decimal>: Signal '<string>' not found in the netlist
SPEF-017	<string>:<decimal>: Connector of instance '<string>' named '<string>' can not be found
SPEF-018	<string>:<decimal>: Transistor, instance, resistor, capacitance or diode '<string>' can not be found
SPEF-019	<string>:<decimal>: Connector of transistor '<string>' named '<string>' can not be found
SPEF-020	<string>:<decimal>: Connector of resistor '<string>' named '<string>' can not be found
SPEF-021	<string>:<decimal>: Connector of <string> '<string>' named '<string>' can not be found
SPEF-022	<string>:<decimal>: Resistor found between two different nets '<string>' and '<string>'
SPEF-023	<string>:<decimal>: Capacitance node '<string>' not found in the netlist
SPEF-024	<string>:<decimal>: Could not find circuit connector '<string>'

6.37. SPI

SPI-002	File '<string>' line <decimal>: .INCLUDE within .SUBCKT not supported
SPI-005	Netlist inconsistency

SPI-007	Variable MBK_SPI_SEPAR must be a single character
SPI-008	Can't read design with mode '<character>'
SPI-009	Can't open file <string>
SPI-011	File <string> line <decimal>: Parameter '<string>' not supported
SPI-013	File '<string>' line <decimal>: Incomplete line
SPI-014	File '<string>' line <decimal>: Too many elements on line
SPI-015	File '<string>' line <decimal>: .SUBCKT within .SUBCKT not supported for SPEF
SPI-016	File '<string>' line <decimal>: No name for subckt
SPI-022	File '<string>' line <decimal>: .ENDS without .SUBCKT
SPI-024	File '<string>' line <decimal>: name discrepancy between .SUBCKT and .ENDS
SPI-025	File '<string>' line <decimal>: No name for transistor
SPI-026	File '<string>' line <decimal>: Transistor '<string>' is already defined
SPI-027	File '<string>' line <decimal>: Transistor has no DRAIN node
SPI-028	File '<string>' line <decimal>: Transistor has no GRID node
SPI-029	File '<string>' line <decimal>: Transistor has no SOURCE node
SPI-030	File '<string>' line <decimal>: Transistor has no BULK node
SPI-032	No model for transistor '<string>' (<string>)
SPI-033	File '<string>' line <decimal>: Incomplete parameter for transistor
SPI-034	File '<string>' line <decimal>: Bad value
SPI-036	File '<string>' line <decimal>: no name for resistor
SPI-037	File '<string>' line <decimal>: resistor '<string>' already defined in subckt
SPI-038	File '<string>' line <decimal>: no name for capacitance
SPI-039	File '<string>' line <decimal>: capacitance already defined in subckt

SPI-040	File '<string>' line <decimal>: No name for subckt
SPI-041	File '<string>' line <decimal>: Subckt '<string>' already defined
SPI-042	Subckt '<string>' already defined
SPI-043	Model '<string>' not found
SPI-044	In circuit '<string>': node number discrepancy between model '<string>' and instance '<string>'
SPI-045	In circuit '<string>': VDD and VSS nodes on the same net VSS nodes: <string> VDD nodes: <string>
SPI-046	In circuit '<string>': Many signals are VSS
SPI-047	In circuit '<string>': Many signals are VDD
SPI-048	File '<string>' line <decimal>: Line too long
SPI-049	File '<string>' line <decimal>: Illegal PWL definition
SPI-050	File '<string>' line <decimal>: Inconsistency in slope #<decimal> of PWL definition
SPI-051	File '<string>' line <decimal>: Node '<string>' already has a name
SPI-052	File '<string>' line <decimal>: Could not evaluate expression '<string>'
SPI-053	File '<string>' line <decimal>: Incomplete pulse definition
SPI-054	File '<string>' line <decimal>: Malformed parameter for '<string>' option
SPI-055	File '<string>' line <decimal>: Multiple parameters for '<string>' option, first one will be used
SPI-056	File '<string>' line <decimal>: No slopes found in PWL for node '<string>'
SPI-057	File '<string>' line <decimal>: No name for diode
SPI-058	File '<string>' line <decimal>: Diode '<string>' already defined
SPI-059	File '<string>' line <decimal>: Bad diode declaration
SPI-060	No model for diode '<string>' (<string>)

SPI-061	File '<string>' line <decimal>: Parameter incomplete for diode
SPI-064	File '<string>' line <decimal>: Unsupported syntax for capacitance
SPI-065	File '<string>' line <decimal>: Unsupported syntax for resistance
SPI-066	File '<string>' line <decimal>: Incomplete or unsupported parameter
SPI-067	File '<string>' line <decimal>: Bad model definition
SPI-068	Conflicting power supply at top level on node '<string>' keeping %gv (other is %gv)
SPI-069	File '<string>' line <decimal>: Incorrect .INCLUDE syntax
SPI-070	Conflicting power supply in circuit '<string>' on node '<string>' keeping %gv (other is %gv)
SPI-071	Unsolved vcard plus: '<string>' minus: '<string>' value: %g in subcircuit '<string>'
SPI-072	Unsolved vcard plus: '<string>' minus: '<string>' value: %g outside subcircuits
SPI-073	Multiple connectors on the same net in subcircuit '<string>', choosing name '<string>', ignoring {<string>}
SPI-074	File '<string>' line <decimal>: Redefinition of function '<string>'
SPI-075	File '<string>' line <decimal>: Malformed function definition
SPI-076	File '<string>' line <decimal>: Cannot evaluate model parameter <string>='<string>'<string>
SPI-077	Technology file loaded with unset simToolModel. Using default setting: simToolModel='<string>'
SPI-078	File '<string>' line <decimal>: No data read between .UNPROTECT and preceding .PROTECT statement. Check your password
SPI-079	File '<string>' line <decimal>: Unable to initialise decryption process
SPI-080	File '<string>' line <decimal>: Evaluate of expression '<string>'<string>
SPI-081	No transistor model names defined for Spice output. Did you set avtSpi[Tn|Tp]ModelName?

6.38. STB

STB-001	Multiple clocks for command '<string>'.
STB-002	Multiple phases for command '<string>'.
STB-003	Memory '<string>' without command ignored.
STB-004	No clock path to command <string>.
STB-005	No data path to latch <string>.
STB-006	No <string> edge on <string> '<string>' propagated from any clock.
STB-008	Bad clock specification for figure '<string>'.
STB-009	Clock connector '<string>' does not exist (line <decimal>).
STB-010	Connector '<string>' does not exist (line <decimal>).
STB-011	Command '<string>' does not exist (line <decimal>).
STB-012	No index on node '<string>'.
STB-013	Multiple commands on node '<string>' with different clocks.
STB-014	Monophase path for latch '<string>'.
STB-015	Clock '<string>' in multiple asynchronous groups.
STB-016	Clock '<string>' in multiple equivalent groups.
STB-017	Equivalent clock '<string>' in different asynchronous group.
STB-018	Multiple clock paths to '<string>'.
STB-019	Multiple clock constraint for signal '<string>'.
STB-020	Crossing clock domains at '<string>'.
STB-021	Undefined period for clock '<string>'.
STB-022	Bad period for clock '<string>'.
STB-023	Cannot find stability information.
STB-024	Cannot close stb file.
STB-025	Stability Figure does not exist.

STB-026	Cannot run '<string>'.
STB-027	Syntax error in stb file (line <decimal>).
STB-028	Unknown or ambiguous phase for '<string>' (line <decimal>).
STB-029	Unmatched stability intervals for '<string>' (line <decimal>).
STB-030	No clock for phase <decimal> on connector '<string>'.
STB-031	No corresponding output file suffix.
STB-032	Only hold constraint for signal '<string>'.
STB-033	Only setup constraint for signal '<string>'.
STB-034	<string>
STB-035	Could not find node '<string>'.
STB-036	Could not find latch node '<string>'.
STB-037	Unknown error.
STB-038	Overflow value %g detected for item <string> of <string>.
STB-039	Configuration mismatch when loading sto file. Check <string> section.
STB-040	Configuration mismatch when loading sto file. Check config variable <string>, current value : <string>
STB-042	Could not match false slack node '<string>' in UTD.
STB-043	Constraint or stability set on connector '<string>' with unknown direction.
STB-044	Bad or undefined clock waveform specification for clock '<string>'.
STB-045	Could not match node '<string>' in switching probability.
STB-046	Access line without command detected on node '<string>'
STB-047	Latch '<string>' has no commands This can lead to infinite loop in STA if there is a loop between 2 latches with no command
STB-048	Possibility of infinite loop: more than <decimal> iterations already done to stabilise the circuit

This is due to a very slow stability convergence or infinite loop

6.39. STM

STM-000	Array bound write in model '<string>'
STM-001	Parse error file '<string>' line <decimal>
STM-002	solvepi (): non convergence
STM-003	Negative capacitance found on signal <string>
STM-004	Deleting more models than allocted
STM-005	stm_cell_delmodel: null cell
STM-006	stm_cell_delmodel: null model
STM-007	Can't open file '<string>'
STM-008	stm_addht impossible: hash table size is '0'
STM-009	stm_addhtitem impossible: value is STM_EMPTYHT or STM_DELETEHT
STM-010	Constraint calculation meaningless in SCM model
STM-011	Cannot resolve pi tree with tables: taking c1 + c2
STM-012	Cannot resolve pi tree with polynoms: taking c1 + c2
STM-013	Load parameter not yet implemented for polynoms
STM-014	Clock slew parameter meaningless in SCM models
STM-015	Clock slew parameter not yet implemented for polynoms
STM-016	Data slew parameter meaningless in SCM models
STM-017	Data slew parameter not yet implemented for polynoms
STM-018	imax not yet implemented for tables
STM-019	imax not yet implemented for polynoms
STM-020	Slew parameter not yet implemented for polynoms
STM-021	Merge of polynom models not yet implemented

STM-022	Reduction of polynoms not yet implemented
STM-023	Shift of polynoms not yet implemented
STM-024	Negation of polynoms not yet implemented
STM-025	No model type for signature
STM-026	STM cannot compute imax from constant model
STM-027	STM cannot compute vth from constant model
STM-028	STM cannot compute slope from constant model
STM-029	NULL model
STM-030	Lost memory consistency
STM-031	Shift of SCM models meaningless
STM-032	Negation of SCM models meaningless
STM-033	Table extrapolation
STM-034	Noise created twice
STM-035	Cannot get current file position
STM-036	Cannot set file position
STM-037	Bad unit for STM_CACHESIZE
STM-038	invth created twice
STM-039	Default capacitance range used for function model
STM-040	Could not find timing model file for figure '<string>' The .stm file might not be accessible to the tool.
STM-041	Constant and table cannot be both valid
STM-042	CALL_SIMULATION called out of context
STM-043	CALL_SIMULATION can't find association for %p '<string>'
STM-044	CALL_SIMULATION_ENV called out of context
STM-045	CALL_SIMULATION_ENV can't find association for %p '<string>'
STM-046	CALL_CTK_ENV called out of context

STM-047	CALL_CTK_ENV can't find association for '<string>'
STM-048	GNS information not loaded, can't compute
STM-049	Could not retreive instance <string>
STM-050	Round overflow
STM-051	Parse error while reading TLF file line <decimal>
STM-052	Imprecision risk for delay '<string>(<character>)' to '<string>(<character>)', slope = <decimal>ps, load = <decimal>fF.

6.40. TAS

TAS-001	slope computing error : circuit not extracted or wrong extraction
TAS-002	netlist error: the connector <string> is not an input
TAS-003	netlist error: the connector <string> is not an output
TAS-004	the connector <string> is not used
TAS-005	the connector <string> can not reach the high level
TAS-006	the connector <string> can not reach the low level
TAS-007	can not force the level of connector <string>
TAS-008	can not open the file <string>
TAS-009	the signal <string> doesn't exist in the circuit or is not a transistor gate or is already declared
TAS-010	can not close the file <string>
TAS-011	no control signal in the latch %ld (<string>)
TAS-012	incomplete input list for the cone %ld (<string>)
TAS-013	the latch <string> is not a differential latch
TAS-014	error in technology file <string>
TAS-015	block without diffusion capacitance. can not measure the drain or source capacitance

TAS-016	block without diffusion capacitance drain and source capacitance will be measured with ACM=1 SPICE method
TAS-017	floating exception
TAS-018	no value or edge can be set for signal <string>. set to 0
TAS-019	can not find the slopes of the cone %ld (<string>)
TAS-020	no active branch found in the cone %ld (<string>)
TAS-021	error in the truth table %ld (<string>)
TAS-022	unknown state <string>
TAS-023	unknown column type <string>
TAS-024	non external branch containing connector cone %ld (<string>)
TAS-025	no slope on the node <string>
TAS-026	usage of the new delay switch model requiere to set the TAS_DELAY_SWITCH to NO.
TAS-027	tasalloc() error: not enough memory. only the critical path will be retained. freeing memory space
TAS-028	fatal error: not enough memory
TAS-029	unknown event %ld
TAS-030	no inout or output connector
TAS-031	loop detected: see the <string>.loop file for more informations
TAS-032	monolatch chain %ld (<string>)
TAS-033	the cone %ld (<string>) does not contain feedback pathes
TAS-034	non-external branch in the cone %ld (<string>)
TAS-035	latch on output connector <string>
TAS-036	precharge on output connector <string>
TAS-037	negative or zero delay between <string>
TAS-038	transistor link losed
TAS-039	too small resistance for signal <string>. set to 1

TAS-041	the precharge processing must be used with the 'ttv' file. the '-n' option will be set
TAS-042	there is no extractible path in the figure <string>
TAS-043	bad environment variable <string>
TAS-044	the cell characterization option is incompatible with the 'deb' file
TAS-045	errors detected in the netlist. see <string>.rep file for more details
TAS-046	latch on input connector <string> ignored
TAS-047	precharge on input connector <string> ignored
TAS-048	memory on input connector <string> ignored
TAS-050	transistors detected in figure <string> impossible in hierarchical mode or instances do not have timing files
TAS-051	impossible to find in figure <string>
TAS-052	bad connector direction on signal <string>
TAS-053	internal connector <string> not found in RCX view
TAS-054	only one connector on internal signal <string>
TAS-055	only input connectors or constante connectors on internal signal <string>
TAS-056	no input on internal signal <string>
TAS-057	conflict detected on internal signal <string>
TAS-058	no rc delay on signal <string>. may be the rc tree are not connected or there is no slope on the driver
TAS-059	more than one latch on signal <string>
TAS-060	more than one precharge on signal <string>
TAS-061	fatal error check your input files
TAS-062	figure <string> already exists
TAS-063	no rc delay on signal <string>. may be loop detected
TAS-064	no or bad rcx file for figure <string>

TAS-065	name <string> already exists
TAS-066	It is better to simulate the signal <string>
TAS-067	No timing model for instance <string>: flattened
TAS-068	No execution context (missing tas_setenv)
TAS-069	Conflicting temperature: <string>
TAS-070	PVT conditions prevent signal propagation on cone <string>, timing arc suppressed Vin under static threshold (Increasing power supply may solves)
TAS-071	PVT conditions lead to out of bound parameters on cone <string>, timing arc suppressed Delay and slope calculations failed on pass-gate (Increasing power supply may solves)
TAS-072	Signal '<string>' connected on power connector '<string>' of instance '<string>'
TAS-073	Could not find connector '<string>' in circuit '<string>'
TAS-074	can't extract path to simulate
TAS-075	can't get patterns to simulate the path

6.41. TRC

TRC-000	[AWE:<decimal>] Internal error : net <string> driver <string> receiver <string> The internal data representation for parasitic element is not consistent. Please contact Avertec Support.
TRC-001	[AWE] Negative delay computed on net <string> Parasitics on the net present some particularities, leading to an error when computing delays with AWE algorithm.
TRC-002	[AWE] All order fail when computing delay on net <string> Parasitics on the net present some particularities, leading to an error when computing delays with AWE algorithm.
TRC-003	Can't open file <string> for direct access. Cache is not used.

	The .rcx file is compressed, so it can't be opened for direct access. The cache is not used, the file is entirely loaded in memory, leading to more memory consumption.
TRC-004	`avtElpCapaLevel mismatch #<decimal>.` The value of the avtElpCapaLevel variable is not consistent with previous setting.
TRC-005	`Internal error #<decimal> on net <string>.` This is an internal error. Please contact Avertec Support
TRC-006	`Internal error #<decimal>.` This is an internal error. Please contact Avertec Support
TRC-007	`Negative ground capacitance found on signal <string>. code=<decimal>` The total capacitance on a net is negative. If it's a power supply net, it should be identified, and no operation on this signal would occur. Check configuration file.
TRC-008	`Can't determine equivalent load on signal <string>.` The equivalent gate output load can't be computed. The gate output is probably not connected to anything.
TRC-009	`The pi load for the equivalent gate load on signal <string> can't be determined.` The equivalent gate output load can't be computed. The gate output is probably not connected to anything, or parasitics on the net present some others particularities.
TRC-010	`The equivalent gate output load gives a negative capacitance for signal <string>. 0 value is retained` This problem occurs when there is a very strong coupling between nets.
TRC-011	`Can't open file <string>.<string> for writting.` Check usual Unix specification for file access: permission mode on file or on directory, disk space,...
TRC-012	`An error occurs when writting in file <string>.` Check usual Unix specification for file access: permission mode on file or on directory, disk space,...
TRC-013	`Try to load an RCX file version #<decimal>. Only 4th version is supported.` The RCX file is too old to be supported. It must be created with newer version of the tool.
TRC-014	`File parse error : Line too long.` The length of a line in the RCX file is too long. This case occurs if there are very long signal or instance names, too many nodes on a connector, or if the file is corrupted. The last line of an RCX file is the word END.
TRC-015	`File parse error line <decimal> in file <string>.<string>.`

An element is not recognized in the RCX file. The file may be corrupted. The last line of an RCX file is the word END. If this is not the case, there has been a problem during the creation of the RCX file.

TRC-016	`Error while getting environment variable : <string>. It is <string>` Check the Avertec Documentation to set proper values.
TRC-017	`could not modify connector '<string>' origin group` This is an internal error in rc delay calculation with GNS.
TRC-018	`rcx timing function does not exist` This is an internal error in rc delay calculation with GNS.
TRC-019	`could not get connector '<string>' origin group` This is an internal error in rc delay calculation with GNS.
TRC-020	`origin index of connector '<string>' is out of range` This is an internal error in rc delay calculation with GNS.
TRC-021	`could not get connector '<string>' destination group` This is an internal error in rc delay calculation with GNS.
TRC-022	`destination index of connector '<string>' is out of range` This is an internal error in rc delay calculation with GNS.
TRC-023	`rcx timing function does not exist` This is an internal error in rc delay calculation with GNS.
TRC-024	`rcx timing function already exist` This is an internal error in rc delay calculation with GNS.
TRC-025	`missing rcx timing origin or destination group` This is an internal error in rc delay calculation with GNS.
TRC-026	`could not find timings \"<character><character>\" between connectors '<string>' and '<string>' for signal '<string>'` This is an internal error in rc delay calculation with GNS.
TRC-027	`could not find timings \"<character><character>\" for an input slope of %gps between connectors '<string>' and '<string>' for signal '<string>'` This is an internal error in rc delay calculation with GNS.
TRC-028	`can't compute awe delay because a connector is missing : start='<string>' end='<string>'` This is an internal error in rc delay calculation.
TRC-029	`can't compute awe delay because input slope is negative : start='<string>' end='<string>'`

	The input slope for rc delay calculation is negative. There is a problem with the gate model. Check technological parameters and power supply.
TRC-030	`can't compute awe delay because power supply is less than vt : start='<string>' end='<string>'` Check power supply and technological parameter of the driver.
TRC-031	`can't compute awe delay because threshold is greater than power supply : start='<string>' end='<string>'` The voltage on the rc net can't reach the threshold delay measure.
TRC-032	`can't compute awe delay because because there is no rcx view or rcx view is typed RCXERROR : start='<string>' end='<string>'` This is an internal error in rc delay calculation.
TRC-033	`can't compute effective load on signal <string> because there is not a valid rcx view.` This is an internal error in rc delay calculation.
TRC-034	`can't get the file position : <string>` When using rc cache, can't get the file offset. Try to disable file compression if needed, or remove rc cache. Check unix file so.
TRC-035	`can't set the file position : <string>` When using rc cache, can't change the position in file. Try to disable file compression if needed, or remove rc cache. Check unix file access so.
TRC-036	`an error occured when reading file : <string>` This rcx file may be corrupted or truncated. The last line of the rcx file is EOF. Check disk space.
TRC-037	`an inconsistant line has been read : <string>` The rcx file is corrupted or truncated. The last line of the rcx file is EOF. Check disk space.
TRC-038	`No aggressor <string> found : <string>` The rcx file is corrupted or truncated. The last line of the rcx file is EOF. Check disk space.
TRC-039	`Signal <string> is defined more than one time : <string>` The rcx file is corrupted.
TRC-040	`Signal <string> is not defined : <string>` The rcx file is corrupted.
TRC-041	`Problem in cache mechanism : the signal <string> can't be released`
TRC-042	`[AWE] can't compute the switching instant of the input for rc net <string>`

	There is a problem when computing the rc input slope on a net. In this case, the rc delay is equal to the switching instant of the rc output.
TRC-043	[AWE] can't compute the switching instant of the output for rc net <string> There is a problem when computing the rc output slope on a net. In this case, the rc delay is equal to 0 and output rc slope is equal to input rc slope.
TRC-044	[AWE] can't determine the shape of the input for rc net <string> The representation for the rc driver can't be determined. In this case, the rc delay is equal to 0 and output rc slope is equal to input rc slope.
TRC-045	internal problem in rcdelay cache : signal <string> is yet in cache There is a proble in rcdelay cache mechanism. Desactivate rcdelay cache.
TRC-046	internal problem in rcdelay cache : no signal There is a proble in rcdelay cache mechanism. Desactivate rcdelay cache.
TRC-047	internal problem in rcdelay cache : signal <string> is not in cache There is a proble in rcdelay cache mechanism. Desactivate rcdelay cache.
TRC-048	internal problem in rcdelay cache : unknown data structure in cache for signal <string> There is a proble in rcdelay cache mechanism. Desactivate rcdelay cache.
TRC-049	internal problem in rcdelay cache : no awe data for signal <string> There is a problem in rcdelay cache mechanism. Desactivate rcdelay cache.
TRC-050	Net <string> is not connex The rc description of a net doesn't connect all connectors\n
TRC-051	Disabling compression for <string>.rcx file as RC cache is active. There is a request to write an .rcx file with the simultaneous uses of output filter and rc cache functionality activated. Since this last functionality requiers direct access on file, the output filter is not used for this file.
TRC-051	Net <string> is not connex The rc description of a net doesn't connect all connectors\n
TRC-052	Can't solve the LU hybrid matrix on net <string> from locon <string> There is a problem in awe computation with matrix.\n
TRC-053	Can't handle UTD for circuit '<string>' generated with avtEnableMultipleConnectorsOnNet=yes in hierarchical mode avtEnableMultipleConnectorsOnNet=yes is supported only for transistor level netlists.\n

6.42. TTV

TTV-010	parse <string>
TTV-011	file <string> does not exist
TTV-020	can not open the file <string>
TTV-021	can not close the file <string>
TTV-022	can not drive the file <string> the figure is not fully in memory
TTV-030	monolatch chain in figure <string>
TTV-031	figure <string> is not ready to built paths
TTV-040	the figure load before and the file <string> do not come from the same timing analyser run
TTV-041	<string> the dates of files must correspond
TTV-042	<string> do not generate with the same technology
TTV-043	<string> do not generate with the same version of technology
TTV-044	<string> do not generate with the same tool
TTV-045	<string> do not generate with the same version of tool
TTV-046	<string> level
TTV-050	not enough memory to allocate <string>
TTV-051	loop detected: see the <string>.loop file for more informations
TTV-052	character <character> in name <string> replaced by _
TTV-053	can not read <string>.rcx file
TTV-054	Infinite loop crossing latches found when searching path <string> Increase your period or set 'avtMaxPathPeriodDepth' to 0
TTV-055	Internal error: <string>
TTV-056	Could not find any path to a clock for precharge or latch '<string>' This message indicates that there is no path from a defined clock to the actual command. This generaly means that a clock declaration is missing for the circuit. Often, missing clocks are clocks that are generated from the interface clock into a latch node. This latch node might be the one to also set as a clock.

TTV-057	Node '<string>' activates writing in '<string>' but is not typed command TTV does not support latch as direct command of other latchs. This can also come from a setup error
TTV-058	Could not match false path <string> node '<string>' in UTD.
TTV-059	Value %gff in capacitance range is lower than connector '<string>' capacitance (%gff).
TTV-060	Could not match directive node '<string>' in UTD.
TTV-061	Error reading SSTA headers in file '<string>': <string>
TTV-062	Error reading SSTA store values in file '<string>'
TTV-063	No stored file found for run <decimal> (global seed=%u, main seed=%u)

6.43. VAL

Error messages description not available yet.

6.44. X2V

Error messages description not available yet.

6.45. YAG

YAG-001	Circuit not found Attempt to disassemble a circuit entity which does not exist.
YAG-002	RC cache must be deactivated in this mode Disassembly modes which involve manipulation of hierarchy require that the RC be deactivated.
YAG-003	Connector '<string>' is power supply AND ground The given external connector has been detected as both Vdd and Vss. This error is fatal.
YAG-004	No VDD signal in the circuit No Vdd power supply has been identified in the circuit to disassemble. This error is not fatal, however, it is recommended to check the power supply configuration.
YAG-005	No VSS signal in the circuit

	No Vss power supply has been identified in the circuit to disassemble. This error is not fatal, however, it is recommended to check the power supply configuration.
YAG-006	`Can't open file '<string>'` The specified file cannot be opened. Check disk space and access privileges. This error may be fatal.
YAG-007	`Can't close file '<string>'` The specified file cannot be closed. Check disk space and access privileges.
YAG-008	`Unrecognized contraint '<string>' in the circuit` The specified mutual exclusion constraint has no correspondance in the circuit to disassemble.
YAG-009	`Can't name toplevel <string> in Blackbox mode` Attempt to rename the top-level figure using a name which is already in use.
YAG-010	`Figure '<string>' has no transistors and no instances` Circuit contains no elements to disassemble.
YAG-011	`Transistors in non-leaf figure '<string>'` In hierarchical disassembly mode, if a no-leaf figure contains a mixture of transistors and instances then the result may not be as expected.
YAG-012	`Signal '<string>' is not driven in behavioural figure`

6.46. ZEN

Error messages description not available yet.

Index