1.0 Appendix C: VITAL-SDF Mapping

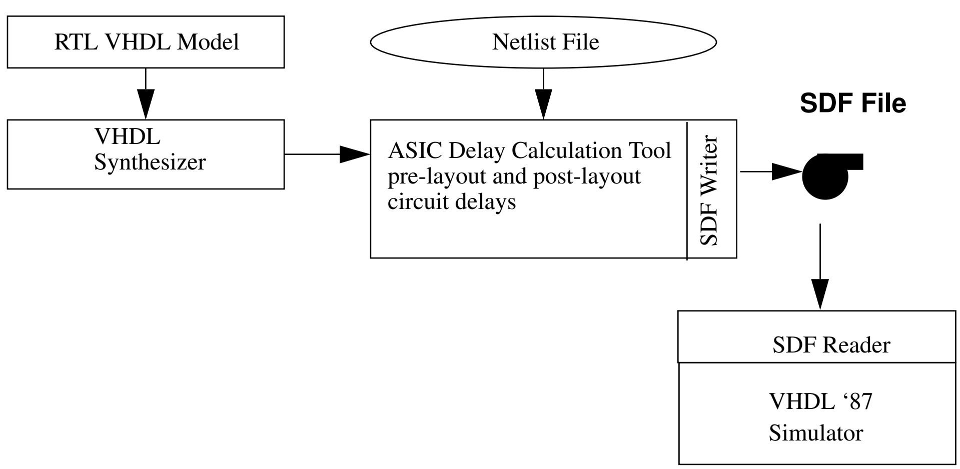
The naming conventions and types of VITAL generics provide the placeholders via which back-annotation of timing data is performed in VITAL compliant models. There are two primary data paths by which timing data can be annotated to model instances and input to simulation:

- By using the appropriate VHDL configuration declaration (generated from an SDF file or any other source)
- Direct-import of timing data from SDF file (possibly generated by a delay calculator). A VITAL compliant simulator, having implemented an SDF reader, directly imports timing data into the simulator.
- The backannotation via direct import of SDF will happen AFTER the elaboration of a VHDL model is done, but BEFORE the initialization phase of simulation starts. The resulting timing data values (viewed by the model during simulation) are the same regardless of the annotation path employed (this is ensured by the restrictions specified in Section 2.9.2 on the use of generics). For a given design, the user may user either of the above two approaches for backannotating any timing data, but not both..
- This specification is based upon version v2.1 of the Open Verilog International (OVI) Standard Delay File (SDF) specification and provides the mapping between the constructs in a SDF file and the corresponding generics in the VITAL compliant models.

Note, however, that other methods, such as reading VHDL source code via TEXTIO were considered, but rejected because of efficiency considerations. In addition, VHDL source code procedural interfaces to SDF were discounted as well because of their relatively poor performance.

1.1 Back-Annotation via SDF Direct Import

For high gate count designs, the compilation of a configuration declaration (generated from a SDF file) for a design containing possibly hundreds of thousands of lines of VHDL code would require a large compilation time. In such situations, a direct SDF import capability is required (see the diagram below). This document specifies the mapping between the SDF constructs and VHDL generics, so that a backannotator can directly modify the generic values in a VITAL compliant model.



1.2 Assumptions

It is assumed that SDF version v2.1 will be used for timing information.

It is also assumed that the SDF file will be created (say by a delay calculator) by using the information from the library data e.g. a VHDL model. This also means that, in general, the data in the SDF file will be consistent with that in the VITAL compliant models.

In constructing the generic names, underscores are used as separators. However, since underscores can also form a part of any VHDL name (say port name), there is always a potential for name conflict. It is assumed that the user would be careful to avoid such conflicts. Also, the delay calculators (or any other tools which writes out SDF file) must take this into account if they use the generic names in generating the SDF file.

Although SDF could specify delays as min:typ:max values, only one of these values will be used for backannotation. The selection of the specific delay values (min, typ or max) could be done by the backannotation program (under a user controlled option).

1.3 Overview

The SDF annotator will perform backannotation by reading delay data from the SDF file and annotating it onto generics in a VITAL compliant model.

- The annotator will handle both ABSOLUTE values and INCREMENTAL delay values from the SDF file. The INCREMENTAL delay values will be added/subtracted to the actual delay value (after elaboration) for the given generic and the new value will be backannotated.
- The type of generics in the VHDL model is predefined in the VITAL_Timing package. It can be of any of the types mentioned below. The annotator will check the type of the generic before annotation is performed and any mismatches would be handled as per the error handling scheme described later.

  - Delay Group: Type for specifying the number of delays (1, 2 or 6) for a scalar generic

  - DelayTypeXX: For 1 delay value
  - TransitionArrayType and its subtypes: For handling an array of TIME values
    - DelayType01: For 2 delay values
    - DelayType01Z: For 6 delay values

  - Array of Delay Group: Type for specifying generics which are array of above delay types

  - DelayArrayTypeXX, DelayArrayType01, DelayArrayType01Z: All these are array of objects defined by the "Delay Group Type". These would be typically used for specifying delays between a pair of array ports.
- For SDF constructs which use a hierarchical name, the annotator will perform the backannotation on the generics using the appropriate scope.
- Wildcards in instance names are expanded and the annotation will take place for all instances in or below the specified scope.

- A generic name (for backannotable timing parameter) in a VITAL compliant model can be of any of the following forms:

(a) <VITAL_prefix>_<port1_name>

For example, tipd_DATA

(b) <VITAL_prefix>_<port1_name>_<port2_name>

For example, tpd_CLK_Q

(c) <VITAL_prefix>_<port1_name>_<edge>

For example, tpw_CLK_posedge

(d) <VITAL_prefix>_<port1_name>_<port2_name>_<edge1>_<edge2>

For example, tsetup_CLK_DATA_posedge_posedge

(e) <VITAL_prefix>_<port1_name>_<inst_name>

For example, tdevice_Q_I1

(f) <VITAL_prefix>_<port1_name>_<port2_name>_<condname>

For example, tpd_CLK_Q_RESET_EQ_1

(g) <VITAL_prefix>_<port1_name>_<port2_name>_<condname>_<edge1>

For example, tsetup_CLK_DATA_RESET_EQ_1_posedge

(h) <VITAL_prefix>_<port1_name>_<port2_name>_<condname>_<edge1>_<edge2>

For example, tsetup_CLK_DATA_RESET_EQ_1_posedge_posedge

The <VITAL_prefix> could be any of the following:

tpd : For Iopath delays

tdevice : For Device delays

tipd : For Port/Interconnect delays

tsetup : For Setup Timing check

thold : For Hold Timing check

trecovery : For Recovery Timing check

tpulse : For Pathpulse delays

tpw : For Pulse width check

tperiod : For Period check

tskewconstraint : For Skewconstraint

1.4 SDF to VHDL Error Reporting Rules

In situations where there is a difference in the amount (and kind) of information between the SDF file and the corresponding VHDL model, a tool which implements SDF to VHDL translation or direct SDF importation will behave in the following manner:

1.4.1 Error: An error message is given and no backannotation is performed

- It is an error if the generic name that the backannotator constructs from the SDF file is not present in the VHDL model. This document describes all the SDF constructs that are supported.
- It is an error if the model has a generic that holds fewer delay values than specified in SDF e.g., the model has a generic of type DelayTypeXX and the SDF has 6 delay values.
- It is an error if the result of adding INCREMENTAL delay value to the actual value of the generic results in a delay which is less than 0. The only exception to this will be cases when negative values of timing constraints are specified.
- It is an error if a generic of type DelayArrayTypeXX or DelayArrayType01 or DelayArrayType01Z is specified in the model and the corresponding ports in the SDF file do not have an index specification.
- It is an error if the size of the generic of type DelayArrayTypeXX or DelayArrayType01 or DelayArrayType01Z does not match the required size (see 1.5.1.2 for how the size of generic array is to be computed).
- It is an error if the model has a generic of type DelayArrayTypeXX or DelayArrayType01 or DelayArrayType01Z and the array index of the generic element corresponding to a particular SDF entry is out of range (see Section 1.5.1.2 for rules for index computation).
- It is an error is the generic is of type DelayArrayTypeXX or DelayArrayType01 or DelayArrayType01Z and the corresponding ports are scalars.

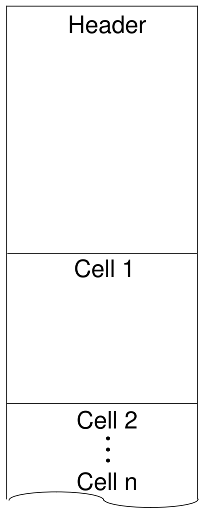
1.4.2 No message: The backannotator program continues without issuing any messages

- If the model has a generic of type DelayType01 or DelayType01Z and the SDF has 1 or 2 delay values the backannotator will fill-in the remaining delays in the generic by following the rules specified by the VitalExtendToFillDelay procedure in the VITAL timing package.
- The backannotator will ignore any timing information contained in the SDF file which is not supported by VITAL compliant models (e.g., constructs like CORRELATION, SUM, NOCHANGE etc.).
- If more than one SDF constructs result in the same generic name, the last value encountered by the annotator will overwrite the previously annotated delay values. This will also be applicable for situations arising because of the fact that SDF is case-sensitive but VHDL is not (the user should be careful to avoid such cases). If INCREMENTAL option is used for the subsequent SDF entry, this would mean that the previously annotated delay value will be considered as the base value w.r.t which the new delay value will be calculated.
- If the SDF fails to provide a value for a generic in a VITAL compliant model, then the associated default value (determined after elaboration) for that generic is used, as per the normal VHDL rules.

1.5 Correspondence of SDF Information to a VHDL Design

The SDF information, containing the timing and constraint data for a design, is tied to the design through cell and path names. Cell names correspond to component instance labels and/or hierarchical pathnames. An instance name of “*” indicates that all instances take on the same timing values.

.




```
(DELAYFILE
      (DESIGN      "my_design")
      (DATE        "Jun 12, 1992 09:46")
      (VENDOR      "Southwestern ASIC")
      (PROGRAM     "Fast program")
      (VERSION     "1.2a")
      (DIVIDER     /)
      (VOLTAGE     4.5:5.0:5.5)
      (PROCESS     "best")
      (TEMPERATURE 27:55:100)
      (TIMESCALE   100 ps)

      (CELL (CELLTYPE "DFF")
      (INSTANCE a.b.c)
                  (DELAY
                  (IOPATH (posedge clk) q (2:3:4) (5:6:7))
                          (PORT clr (2:3:4) (5:6:7))
                  )
      )
```

1.5.1 Path delays

1.5.1.1 Simple path delays

In case Input and Output ports are scalar or if they are arrays and the delay values between all input and output bits is same, the mapping is

IOPATH Input Output   tpd_Input_Output

Where tpd_Input_Output is a generic of any Delay group type i.e. DelayTypeXX, DelayType01 or DelayType01Z. For example,

IOPATH A   Y   tpd_A_Y

For cases when there is an <edge> specification along with Input name the mapping is as follows:

IOPATH (posedge A) Y   tpd_A_Y_posedge

1.5.1.2 Path delays with Input/Output name having index/range specification

For Input or Output ports in SDF which have an index/range specification (i.e. cases when delays between various bits of input/output ports can be different), the intent is to have a scheme which:

- reduces the number of generics and their size, whenever possible

- makes the generic name always determinant

- allows ease of use of indexed generic names in generate/loop statements in a VITAL model

This scheme outlined in this section is applicable to all SDF constructs which can have index/range specification in the port names.

For array ports, where the delay from each bit could be potentially different, the generics in a VITAL compliant models must be of array or delay groups (as defined earlier). For a pair of ports (A, Y), the following combinations can arise in a model:

(a) Port1 Vector	Port2 Vector
(b) Port1 Scalar	Port2 Vector
(c) Port1 Vector	Port2 Scalar
(d) Port1 Scalar	Port2 Scalar (This case has already been addressed in Section 1.5.1.1 above)

For all the above cases, in his VITAL model, the user must have a generic by the name

tpd_A_Y

which must be of an array of any Delay group type (DelayArrayTypeXX, DelayArrayType01 or DelayArrayType01Z). The size of this generic is computed as shown below:

- If only one of the ports is an array, the size of the generic array must be equal to sizeof(port).

- When both the ports are arrays, the size of the generic array must be equal to sizeof(port1) * sizeof(port2)

By using the above scheme, the user would be creating a delay slot for each path that can exist between the bits of Input and Output ports. The backannotator will replace an element of the above generic array by using a consistent scheme (using "row-dominance") which is shown below:

- Case1: If only 1 of the ports is an array

If the ports in the VHDL model are:	A Y(j1 to/downto j2)
If the corresponding generic is:	tpd_A_Y(g1 to/downto g2)
For any SDF entry:	IOPATH A Y[c]
The backannotator will replace the value of:	tpd_A_Y(x)

Where:

x = g2 + abs(c-j2)*(g1 - g2)/abs(g1 - g2)

For generic having descending range (g1 downto g2) this would reduce to

x = g2 + abs(c - j2)

For generic having ascending range (g1 to g2) this would reduce to

x = g2 - abs(c - j2)

- Case2: If both the ports are arrays

If the ports in the VHDL model are:	A(i1 to/downto i2) Y(j1 to/downto j2)
If the corresponding generic is:	tpd_A_Y(g1 to/downto g2)
For any SDF entry:	IOPATH A[r] Y[c]
The backannotator will replace the value of:	tpd_A_Y(x)

Where:

x = g2 + (abs(c - j2) + abs(r - i2)*(abs(j1 - j2) + 1))*(g1 - g2)/abs(g1 - g2)

For generic having descending range (g1 downto g2) this would reduce to

x = g2 + abs(c - j2) + abs(r - i2) * (abs(j1 - j2) + 1)

For generic having ascending range (g1 to g2) this would reduce to x = g2 - abs(c - j2) - abs(r - i2) * (abs(j1 - j2) + 1)

If the user has the following SDF entry (e.g., in situations where the delays between all the Inputs and Outputs were equal):

IOPATH A[0:1] Y[0:1] delay

In this situation, the annotator will annotate the same delay in the corresponding elements of the generic array (4 elements in this example), using the indexing scheme shown above.

1.5.1.3 Conditional Path Delays

In order to allow backannotation of conditional path delays, we need to map the condition expression in SDF to a generic name in the VHDL model. We need to define a scheme to take a condition expression and create a unique generic name corresponding to it. Since the VHDL identifiers can only be made up of letters, digits or underscores, we have to map the operator symbols appearing in a conditional expression to one of the above. The following SDF construct

COND <condition> IOPATH Input Output (delay)

would result in the generic name

tpd_Input_Output_<condname>

where <condname> is a name constructed from the condition expression using the following rules:

- separate each part of the condition with an underscore
- replace operator symbols as follows:

==, !=	by EQ, NE
<, <=, >, >=	by LT, LE, GT, GE
+, -, *, /, &, **	by PL, MI, MU, DI, AM, EX
(,)	by OP, CP
&&, ||	by AN, OR
- replace ranges like 1 to 2	by 1TO2
- replace tick (')	by TI
- replace [,]	by nothing (since VHDL names should be using (,) and not [,])

For example, using the above scheme,

COND RESET == \'1\' && CLK\'EVENT && CLK == \'1\' IOPATH A Y (delay)

will be mapped to

tpd_A_Y_RESET_EQ_TI_1_TI_AN_CLK_TI_EVENT_AN_CLK_EQ_TI_1_TI

The above scheme is applicable to any SDF construct which has a COND specification.

If both <edge> and COND appear in an SDF construct, the corresponding generic will be of the form:

tpd_Input_Output_<condname>_<edge>

1.5.2 Device delays

INSTANCE Instance DEVICE Output tdevice_Instance_Output

The instance on whose Output the device delay is specified should be a label for a concurrent procedure call.

1.5.3 Timing Checks

1.5.3.1 Simple timing checks

SETUP Input1 Input2	tsetup_Input1_Input2
HOLD Input1 Input2	thold_Input2_Input1
RECOVERY Input1 Input2	trecovery_Input1_Input2

The SETUPHOLD construct would be mapped to two separate generics- one each for SETUP and HOLD, using the convention described above. For Pulsewidth/Period, the SDF file only specifies the MINIMUM delay values. As such the user will only have one generic for pulsewidth/period.

WIDTH Input	tpw_Input
PERIOD Input	tperiod_Input

The above convention will also be used for cases when there is an <edge> specification along with Input. For example,

WIDTH posedge Input	tpw_Input_posedge
PERIOD negedge Input	tperiod_Input_negedge

1.5.3.2 Conditional Timing checks

Conditional timing check delays will be mapped using the same scheme as that for conditional path delays as has been outlined above.

COND <cond> SETUP Input1 Input2

will be mapped to

tsetup_Input1_Input2_<condname>

If both <edge> and COND appear in an SDF construct, the corresponding generic will be of the form:

tsetup_Input1_Input2_<condname>_<edge>_<edge>

1.5.4 Input Port Delays and Interconnect Delays

PORT Input	tipd_Input

For the multiple driver topology of INTERCONNECT delays, if the delay values from all drivers to a receiver are equal, then they may be modeled as an input port delay for that port. However, if these delays are not equal, then the last one appearing in the SDF file will be used as the value of the generic and a warning will be issued.

INTERCONNECT Output Input	tipd_Input

1.5.5 Skew Constraint

SKEWCONSTRAINT Port	tskewconstraint_Port