

# A Proposed Back Annotation File Format for VHDL

**Victor Berman**  
**Cadence Design Systems**  
**2 Lowell Research Center**  
**Lowell, MA 01852**  
**berman@cadence.com**

**Cary Ussery**  
**Vantage Analysis Systems**  
**42808 Christy Street**  
**Fremont, CA 94538**  
**uunet!vntage!cau**

## 1.0 Introduction

This paper proposes a file format for timing information which can be back annotated into a VHDL design. The authors are in the process of establishing an IEEE PAR for, among other issues, standardizing such a format and this paper gives an overview of a strawman for that effort. The file format is based on the SDF format for Verilog with modifications to make it appropriate for both VHDL and Verilog models. The proposal targets the 1992 version of the language which is currently under development. The proposal covers back annotation of timing information appropriate for commonly used model characteristics such as pin to pin delays, wire delays.

## 2.0 An Overview of Back Annotation

Back annotation is the process of including externally derived data into a design or model. This is typically used to refine timing data in a model as the design passes through the stages from functional verification to pre- and post- layout analysis. This timing data is derived from several sources such as timing analyzers, delay calculators, and library specific data tables. This data is generally technology and process specific and the methods for performing delay calculation are generally proprietary and are different for each foundry. The advantage of using back annotation over directly computing delay information in the simulation models is that the process can be decoupled from the functional design, and tools which are specifically engineered for timing analysis can be used. In addition, if the data used for back annotation is in a neutral format, it can be

used as a bridge between different tools in the design process as well as between tools in different systems.

This type of design where the timing data forms the glue between the different aspects of the design process is becoming known as timing-driven design.

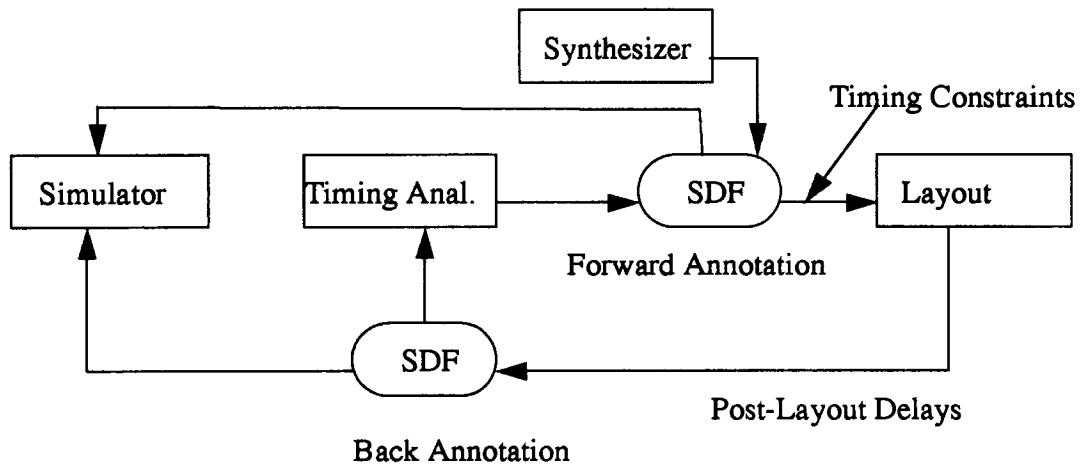
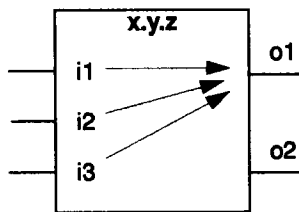


Fig 2-1. Design Flow for Timing-Driven Design

### 3.0 Parameters of Circuit Timing.

#### I/O Path Delays



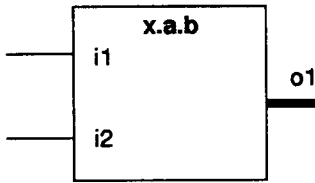
An I/O path delay represents the delays on a legal path from an input port to an output port of a device. Each delay value is associated with a unique input port/output port pair.

I/O path delays are represented in the SDF by the keyword *IOPATH*, as shown in the following example SDF entry:

```
(INSTANCE /x.y.z)
(DELAY (ABSOLUTE
  (IOPATH (posedge i1) o1 (2:3:4) (4:5:6)
  (3:5:6))
  (IOPATH i2 o1 (2:4:5) (5:6:7) (4:6:7))
  (COND i1 (IOPATH i3 o1 (2:4:5) (4:5:6)
  (4:5:7)))
)
```

#### Device Delays

Device delays are used to associate delays with a module, gate, or a gate's output port. If a module has more than one output, delays can be attached to each output port using the optional port specification. If no port is specified it is assumed

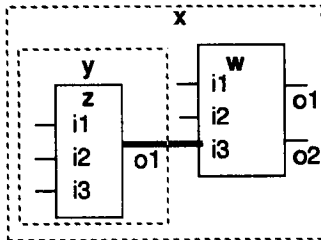


that all output ports have the same delay value associated with them.

The device delay is represented in the SDF by the keyword *DEVICE*, as shown in the following example SDF entry:

```
(INSTANCE x.a.b)
(DELAY (ABSOLUTE
        (DEVICE o1 (6:7:8) (4:6:7) (5:8:9)
                  (4:7:8) (4:5:7) (7:8:9))
))
```

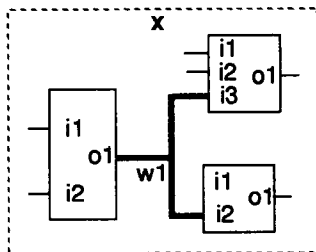
## Interconnect Delays



Interconnect delays represent actual or estimated delays in the wire paths between devices. Conceptually, the delay is associated with the input port of the destination device. In the case of multiple sources fanning into the same device input port, an array of delays is associated with the input port, each one representing a unique interconnect path from each source. Interconnect delays are represented in the SDF by the keyword *INTERCONNECT*, as shown in the following example SDF entry:

```
(INSTANCE x)
(DELAY (ABSOLUTE
        (INTERCONNECT y.z.o1 w.i3 (5:6:7)
                    (5.5:6:6.5))
))
```

## Net Delays



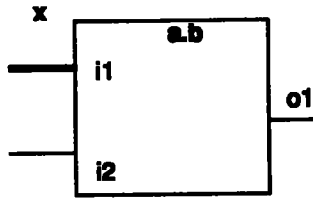
A net delay represents a delay for a complete net, where delays from all the source port(s) on the net to all destination port(s) have the same value. It is a short form of interconnect delay.

Net delays are represented in the SDF by the keyword *NETDELAY*, as shown in the following example SDF entry:

```
(INSTANCE x)
(DELAY (ABSOLUTE
        (NETDELAY w1 (2.5:3.0:3.5) (2.9:3.4:4.2)
                    (6.3:8:9.9))
))
```

In this example, the net is identified by name. A net may also be identified by the output port driving it (if it has only one source).

## Port Delays



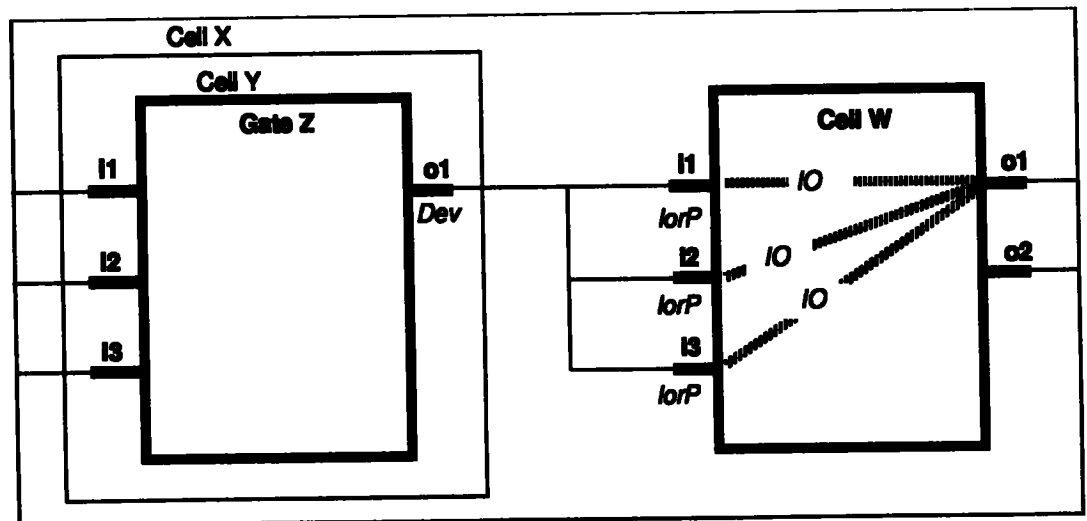
Port delays allow estimated interconnect signal delays to be placed at input ports without having to specify a start point for the delay path. Interconnect delays override port delays if both are specified for the same input port.

Port delays are represented in the SDF by the keyword *PORT*, as shown in the following example SDF entry:

```
(INSTANCE x)
(DELAY (INCREMENT
(PORT a.b.i1 (-2:0:2))
))
```

In this example, the port delay is specified as incremental, which means that the values increase or decrease existing delay data rather than replacing it.

## 4.0 A File Format for Back Annotated Timing SDF Abstract Delay Models



- Dev* = DEVICE delays (Intrinsic delays)
- lorP* = INTERCONNECT delays or PORT delays
- IO* = IOPATH delays (Cell path delays)

---

## DEVICE Delays

```
(INSTANCE /x/y/z)
(DELAY
  (INCREMENT
    (DEVICE <delay_spec>)
  )
)
```

Note: <delay\_spec> represents delay details that have been omitted.

The DEVICE delay reflects the intrinsic delay of a cell or gate. Intrinsic delay is specific to the type of the object and so has the same value for every instance of that cell or gate. Conceptually, this represents all path delays through the object, independent of loading or input slope. At the gate level the delay is associated with the output. If a cell has more than one output, inertial delays can be attached to each output port by using the optional port specification. If no port is specified, then it is assumed that all ports have the same delay value associated with it.

---

## IOPATH Delays

```
(INSTANCE /x/y/z)
(DELAY
  (ABSOLUTE
    (IOPATH (posedge i1) o1 <delay_spec>)
    (IOPATH i2 o1 <delay_spec>)
    (IOPATH i3 o1 <delay_spec>)
  )
)
```

Note: <delay\_spec> represents delay details that have been omitted.

IOPATH delays represent delays on legal paths from each input port to all or some of the output ports of a device. The place holder (delay box) for these delay values are placed between the input and output port pairs.

---

## INTERCONNECT Delays

```
(INSTANCE x)
(DELAY
  (INCREMENT
    (INTERCONNECT y.z.o1 w.i1 <delay_spec>)
  )
)
```

Note: <delay\_spec> represents delay details that have been omitted.

Interconnect delays represent actual or estimated delays in the wire paths between devices. Conceptually, this delay box is situated at the input port of the destination device. In case of multiple sources fanning into a device input port, an array of delay boxes is associated with the port, each box representing a unique interconnect path from each source.

NETDELAY is a special case for interconnect delay, where all delays from the source port on the net to all the destination ports are of the same value.

---

### **PORT Delays**

The PORT delay command allows placement of estimated interconnect delay values in the input delay box, without requiring any path begin-points to be specified.

INTERCONNECT delays override PORT delays if both are specified for the same input of a cell.

## **Standard Delay File Descriptions**

This section presents the lexical conventions, syntax, escaping rules, and semantics of the Standard Delay File format.

---

### **Lexical Conventions**

Keywords, identifiers, characters, and numbers must be delimited by either syntax characters or by white space. Syntax characters are any non-alphanumeric characters required by the syntax. Alphanumeric characters include upper and lower case alphabetic characters, all the numbers, and underscore. White space may be used to separate lexical tokens, except for hierarchy separators in path names where no white space is allowed.

<item>	is a syntax item.
<item>*	is zero or more items.
<item>?	is one or more items.
{<item>}	is an optional item.
<qstring>	is alphanumeric text enclosed between double quotes.

<hchar>	is any ASCII character except "\".
<number>	is a positive real number.
<rnumber>	is a real number.
<dnumber>	is a number containing a set of the following characters, optionally preceded by + or - 0123456789.
<identifier>	Identifiers may be up to 1024 characters long. No white space is permitted in an identifier.
<path>	is a hierarchical <identifier> or a special character "*". No white space is permitted in a <path>.
<fpath>	is a full path name for a file.

---

## Delay File Syntax

This section presents the syntax for the major items in a Standard Delay File.

<delay_file>	::= ( DELAYFILE <design_name> <date> <vendor> <program_name> <version> <hierarchy_divider> <voltage> <process> <temperature> <time_scale> <cell>? )
<design_name>	::= ( DESIGN { <qstring> } )
<date>	::= ( DATE { <qstring> } )
<vendor>	::= ( VENDOR { <qstring> } )
<program_name>	::= ( PROGRAM { <qstring> } )
<version>	::= ( VERSION { <qstring> } )
<hierarchy_divider>	::= ( DIVIDER { <hchar> } )
<voltage>	::= ( VOLTAGE { <expression> } )
<process>	::= ( PROCESS { <qstring> } )
<temperature>	::= ( TEMPERATURE { <rexpression> } )

```

<time_scale> ::= ( TIMESCALE <tsvalue> fs )
              ||= ( TIMESCALE <tsvalue> ps )
              ||= ( TIMESCALE <tsvalue> ns )
              ||= ( TIMESCALE <tsvalue> us )
              ||= ( TIMESCALE <tsvalue> ms )
              ||= ( TIMESCALE )

<cell> ::= ( CELL <celltype> <instancename> <timing_spec>? )

<celltype> ::= ( CELLTYPE <qstring> )

<instancename> ::= <instance>?

<instance> ::= ( INSTANCE { <path> } )

<timing_spec> ::= ( DELAY <delttype>? )
                 ||= ( TIMINGCHECK <tcdef>? )
                 ||= ( INCLUDE <fpath> )

```

## 5.0 Reading Timing Information into VHDL Description

---

### SDF Access by Tools

To access data in the SDF, EDA tools must contain function calls to access routines in the SDF Interface. The function calls may be embedded in the tool itself, or be in external access utilities.

A timing analysis or synthesis tool must use:

- n write function calls to pass timing constraints forward to layout tools
- n read function calls to access pre- and post-layout timing information for accurate circuit analysis

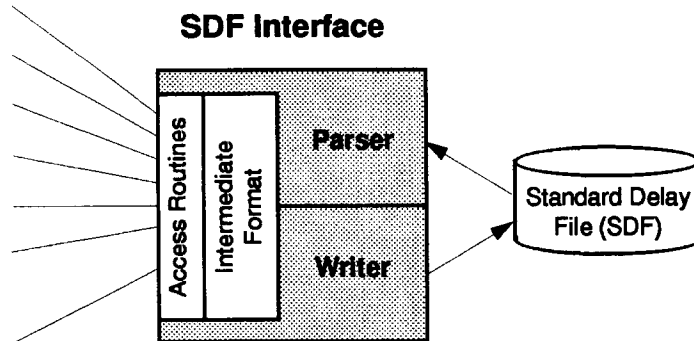
A logic simulation tool must use:

- n read function calls to access pre- and post-layout timing information for accurate circuit analysis

An IC or PCB layout tool must use:

- n read function calls to import design constraints into the tool
- n write function calls to pass parasitic information about the design to a delay calculator, which then passes pre- and post-layout timing information back to the simulation and timing analysis tools

An Automatic Test Pattern Generator (ATPG) tool must use:  
n read function calls to import critical path information into  
the ATPG



## 6.0 Summary

The formats and timing models introduced in this paper are quite general and fit within the framework of standard practises of commercial ASIC and standard component models. The specific methodology for applying these concepts to VHDL will be the work of the Special Interest Group on Timing which is being organized as a working group of the IEEE DASS. This group will also be investigating the mapping of timing information into the VHDL description and how that mapping interoperates with the timing information contained in the VHDL description.