VITAL: VHDL Initiative Toward ASIC Libraries

Technical Requirements Document
Version 1.0

October 8, 1992

1.0 Purpose

The purpose of VIT AL is to promote the rapid development and use of ASIC libraries for VHDL design. The purpose of this document is to outline the key elements of design data which need to be incorporated into a set of standard modeling practices in order to enable the rapid development of ASIC libraries.

The information presented in this document is the result of numerous discussions with ASIC vendors, CAE tool vendors, and ASIC designers to determine the requirements for effective design and fabrication of ASICs using VHDL.

VITAL's charter is to overcome the technical barriers which may exist and to promote the development and use of ASIC libraries for VHDL based design. This will be accomplished by developing an "engineered solution" for ASIC design using VHDL.

The outcome of this effort is expected to be:

- a common back-annotation file form
- a transformation methodology from this file format into VHDL
- a common timing representation for writing VHDL models
- a (potentially acceleratable) primitive set
- a standard interface specification for models
- example models to serve as a demonstration vehicle.

The initial effort will leverage V erilog modeling techniques for the benefit of the equivalent VHDL implementation. A second phase will enhance that methodology to provide increased capability .

2.0 Coordination

The VHDL Technology Group will act as the technical lead on the project with William Billowitch serving as the principal engineer. Steve Schulz, a member of the technical staff at the Defense Systems and Electronics Group of Texas Instruments is a co-coordinator in the U.S. Eric Huyskens, head of CAD and test for the VLSI group at Alcatel NV will serve as European coordinator.

EMAIL: An email reflector has been set up to allow for free communication among the members participating in the program. To be added to the list, or to participate, send your request or comments to:

vital@cadence.com

Standards Committees: It is the intent of this effort to share the efforts of this initiative with appropriate standards organizations in addition to soliciting input from knowledgable industry source not already affiliated with such organizations.

Call For Action:

Vital is an initiative on a fast track. If you have suggestions, solutions, or recommendations, please forward them to my attention immediately. The best way to influence the outcome of this effort is to become involved as an ACTIVE participant early in the development process.

Technical Survey

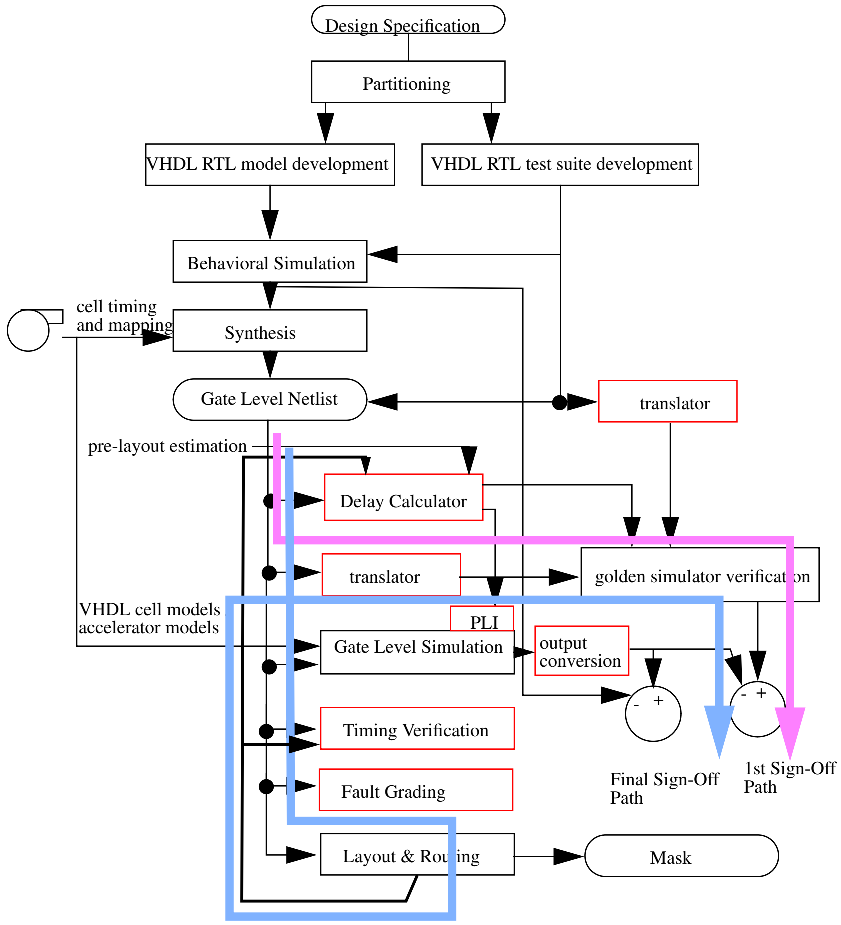
A series of meetings with ASIC foundries had been conducted to establish the requirements for implementing a sign-off path for VHDL-based ASIC designs. Discussion topics included their use of existing back-annotation solutions, timing verification, fault grading, modeling style, reasons for offering or not offering VHDL solutions, and wish lists for future implementations.

Summaries of each discussion area are presented below.

2.1 Design Flow

While the issue of developing VHDL model libraries was initially a key concern, it became quite evident from conversations with ASIC foundries that the issues involved in providing a total VHDL solution far overshadowed simply the model library issues.

Figure 2.1:



By observing the design flow graph above, it is easy to see that VHDL models comprise only a small portion of the tools required to bring a total VHDL solution to the marketplace. Yet it is the information contained in the library which drives the remaining tools and translators. Therefore, the development of a VHDL ASIC modeling methodology will enable the development of the ancillary tools.

The key issues identified by participants in this process have been articulated below:

- Implementing a total VHDL solution involves the commitment of approximately $750k - $1M per model library, in terms of manpower support, tools, and effort. In a cost conscious environment, delaying this cost until absolutely essential is desirable.
- Currently, VHDL customers have not pressed hard enough (i.e. voted their dollars elsewhere) to convince the foundry to budget the expense.

- There are self-feeding misconceptions that VHDL cannot handle timing information in a robust manner necessary for ASIC development (Editor's note: This Vital initiative is designed to change that perception).
- There is widespread concern that major CAE vendors do not have a robust VHDL solution and that the ASIC foundries will have to build a different library to support each vendor.

2.2 Back-Annotation

From observation of the design flow, it is clear that the need for a robust pre-layout and post-layout back-annotation methodology in VHDL is paramount to the success of this endeavor. Therefore, the Vital initiative will focus considerable efforts to develop a back-annotation solution which works and can be enhanced later to accommodate more advanced timing models and relationships.

An overwhelming number of ASIC foundries expressed an interest in using SDF (Standard Delay File) as a back-annotation file format. Each was willing to expend the resources to develop SDF interfaces. However, all ASIC foundries reiterated their preference for a single standard in contrast to supporting multiple standards in this area for the economic reasons expressed previously.

2.3 Model Interface Issues

Every ASIC foundry expressed an interest in standardizing upon an interface methodology for models. In fact, such a standardization is essential for the development of a working back-annotation solution. Furthermore, by defining an interface specification, model interface styles have the potential to be developed in a vendor independent style which would allow users and tool vendors to benefit from knowing that certain information regarding a model will be expressed in a particular fashion and consequently, tools can be developed to exploit this capability.

2.3.1 Interface Value Systems

Of the ASIC foundries who participated in the survey, all expressed that an X01Z value set was a necessary set of values in modeling ASIC cells for non-ECL technologies. ECL technologies require additional strength information in order to interoperate.

From the conversations, each vendor was in favor of supporting the STD_LOGIC_1164 system which incorporates the X01Z subtype. In addition, certain accelerator products have been optimized for a subtype of STD_LOGIC_1164. All ASIC foundries emphasized the importance of having acceleratable gate level primitives to provide adequate throughput for verification.

This potential restriction does not impact compliance with STD_-LOGIC_1 164 since X01Z is a defined subtype under the standard.

2.4 Timing Information

It is largely recognized that a need exists to standardize the way timing is represented for ASIC modeling while at the same time, allowing leading-edge vendors the flexibility they may require to extend the specific methods by which timing information is derived or handled within a model.

Therefore considerable efforts will be expended to define the modeling interface for timing.

2.4.1 Naming Conventions

Users prefer a standard naming convention for timing information which would allow their tools to provide or extract information in a VHDL modeling context.

2.4.2 Pin-to-Pin Timing

A number of pin-to-pin timing issues which require consideration include:

- State dependent delays
- Load dependent delays
- Process dependent delays
- Temperature dependent delays
- Voltage dependent delays
- Transition time dependent delays
- Dynamic delay effects

2.4.3 Timing Verification

Certain foundries rely heavily upon static timing verification, particularly for their synchronous design technique customers. Therefore, an additional requirement will be to support static timing information tools while addressing other timing information issues.

2.4.4 Synthesis Timing Information

By and large, synthesizers require the same types of timing information as do static timing analyzers. Therefore, it is expected that a solution for the former will also suffice for the latter.

2.5 Accelerated Simulation

Next to timing accuracy, is an urgent need for accelerated gate-level simulation. Existing solutions (i.e. Ikos, Zycad, Cadence) utilize a special set of highly optimized primitives coupled with a somewhat restricted logic value system. The combination allows the vendor to offer an accelerated solution via table lookups, specialized hardware or a combination of the two.

Given the insatiable demand for simulation speed, it is essential that this effort define a set of acceleratable primitives.

2.5.1 Acceleratable Primitives

A number of ASIC vendors develop their cell libraries using Verilog primitives. And a number of vendors have expressed an interest in developing an equivalent VHDL solution.

As an example, the Verilog Primitives are shown below:

AND	BUF	NMOS	TRAN	PULLUP
NAND	NOT	PMOS	TRANIF0	PULLDOWN
NOR	BUFIF0	CMOS	TRANIF1	
OR	BUFIF1	RNMOS	RTRAN	
XOR	NOTIF0	RPMOS	RTRANIF0	
XNOR	NOTIF1	RCMOS	RTRANIF1	

2.5.2 Impact on Logic Systems

The goal is to develop a solution which preserves compatibility with the STD_LOGIC_1 164 package while continuing to allow vendors to utilize their proprietary acceleration algorithms.

In the design of 1 164, these issues had been considered. Consequently , subtypes had been or can be declared which provide type compatibility at the port level.

Table 1: Acceleration Value System Mappings

1164	VHDL-XL	Sim1	Sim2	others
U	X	X	X	
X	X	X	X	
0	0	0	0	
1	1	1	1	
Z	Z	Z	Z	
W	X	X	W	
L	0	0	L	
H	1	1	H	
-	X	X	X	

2.6 Fault Grading

Fault simulation endeavors to locate physical flaws of semiconductor manufacturing by devising a comprehensive test suite whose purpose is to detect failed transistors or gates. The classical techniques involve detecting “stuck-at-0” and “stuck-at-1” faults using numerous methods.

No specific requirements were raised during the discussions pertaining to fault simulation. Therefore, these issues will be postponed until a later time.

2.7 Suggested Methods for Developing Libraries

Two methods were suggested for assisting ASIC foundries in developing models of their libraries. The first suggestion was to develop and offer a Verilog to VHDL translator robust enough to handle ASIC modeling styles. The other involved the development of a template library which foundries could use and modify to develop their own libraries.

After conversation with a number of ASIC foundries the following conclusions were drawn:

- General concern over the robustness of translation and the efficiency of the generated VHDL code.
- A number of vendors developed their libraries by translation from an internal "golden" format to Verilog and going from Verilog to VHDL would be yet another step removed from the original golden environment.
- Model templates are not required since there is a concern that the templates will not be specific enough for the vendor.
- A corresponding test suite to accompany the library is necessary and vendors prefer to develop such a test suite themselves for verification reasons.
- Hence the template library only solves a small percentage of problems and is not required.

Consequently, neither of these suggestions would serve the needs of the marketplace from the perspective of ASIC library development.

2.8 General Issues

A number of general issues were addressed in addition to the ones already mentioned. A brief outline of issues follows:

- Among the avid Verilog users, there is a general preference to preserve a Verilog look and feel to the extent possible in VHDL.
- Strong desire for timing accuracy and fast (accelerated) simulation
- A number of vendors expressed a desire to have more features built into VHDL rather than available through packages.
- Most have a modeling style which separates timing information and handling from the functional aspects of the model.
- Strong interest in supporting Verilog's User Definable Primitives (UDP's)

3.0 Technical Requirements

This section sets forth a set of requirements and proposes certain solutions based upon existing methodologies which have been known to work in an acceptable manner . Emphasis during the implementation phase will be on using existing methodologies if they apply .

3.1 Back-Annotation Issues

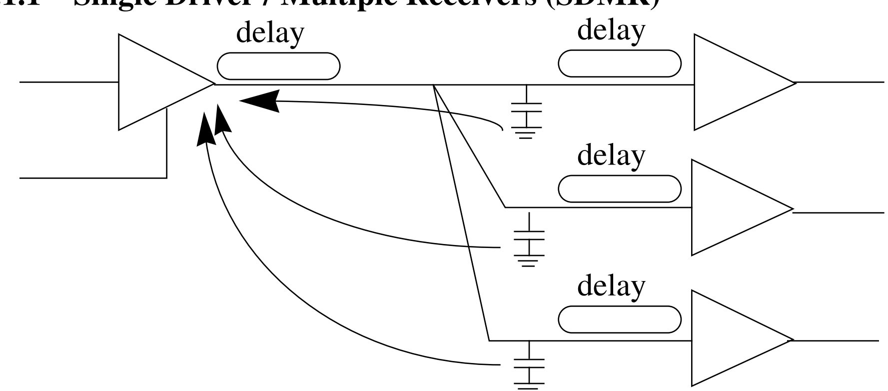
The key issues regarding back-annotation are (a) how do we represent hierarchical back-annotation and pin-to-pin timing parameters and (b) how does that information map into the particular instance of a given model.

3.1.1 Interconnection T opologies

There are a number of commonly used interconnection topologies found in typical logic designs. The impact of these topologies reflects upon the design of (a) a back-annotation scheme and (b) an interconnection modeling approach.

The descriptions provided within this section 4.1 shall pertain to schematics of component interconnections as well as to hierarchical component interconnections as well.

3.1.1.1 Single Driver / Multiple Receivers (SDMR)

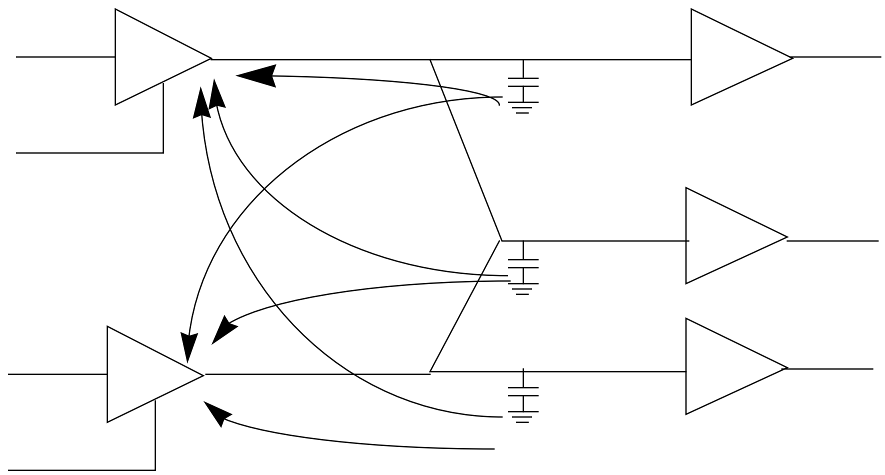


Data type/resolution function impact: The single driver / multiple receiver case also requires no more than X-0-1-Z data types and does not require a resolution function.

Timing impact: In this case, there are four subnets. The first from the tristate driver to the first interconnection of the receivers, the second, third and fourth from the intersection to each of the receivers. In this case, all of the lumped capacitive load is reflected back to the driver . Each of the interconnection paths are assigned their respective delay subsegments. Notice that a portion of the interconnect path is reflected to the driver , while the path delay from the interconnection to the receivers is reflected to each of the receivers respectively .

SDF Impact: In this case, all of the transmission delay can be reflected to the inputs and the SDF INTERCONNECT syntax will suffice.

3.1.1.2 Multiple Drivers / Multiple Receivers (MDMR)



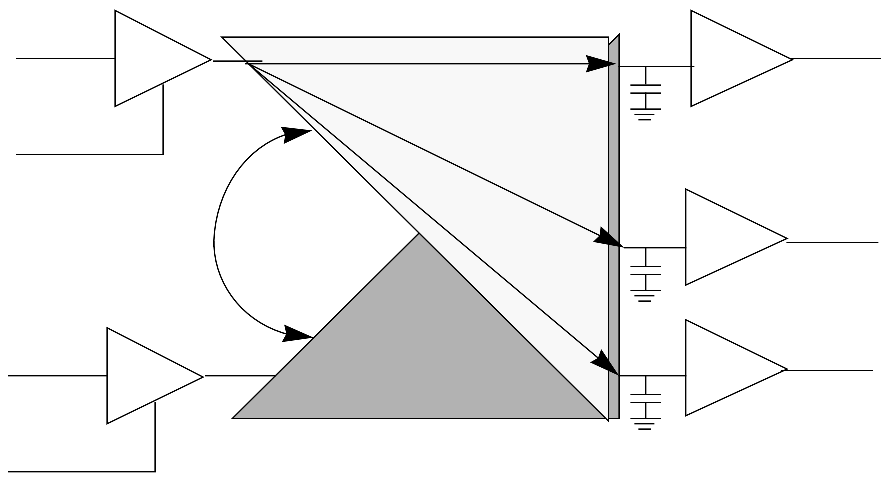
Data type/resolution function impact: The multiple driver / multiple receiver case also requires no more than X-0-1-Z data types but does require a resolution function.

Timing impact: This situation is perhaps one of the more complicated interconnection topologies and timing problems to solve.

- Assume for a moment that both tri-state drivers are in the active state (i.e. driving a 0 or 1). Under this circumstance, we could reflect a proportionate share of the accumulated capacitive load to each of the drivers. For example, if each receiver has a 2pf load, then the total load on the net would be 6 pf and between two equivalent drivers, each would share a 3 pf load. Under this condition, a simple timing model could be used where the propagation delay is derated based upon the load. In addition, the only skew between a signal received by the top receiver and any other receiver would simply be due to the difference in path delays.
- In the case where one of the tri-state drivers is disabled (i.e. driving Z) and the other is driving an active value, a different series of events must be considered. Namely ,

the load is no longer shared by both drivers since the disabled driver is not contributing to charge or discharge the lumped capacitance. In this case, all of the load must be reflected to those drivers which are actively driving. This dynamic load scenario is one of the most difficult to handle in an efficient manner in any digital simulation environment and to the author's knowledge has never been addressed as part of a built-in simulator solution. (Note: This problem is solvable in VHDL, but at great simulation runtime expense)

.



- In the case shown above, the delays from the bottom driver and the top driver cannot be reflected to the inputs of the receivers. Hence, a network model must be inserted between the drivers and receivers such that the pin-to-pin delays of the interconnection can be properly attributed to the path by which the signal had arrived.
- In addition, by the introduction of a net model, a signal can be passed between the net models to provide dynamic drive indications in order that the net model can be designed to properly handle dynamic loads.

SDF Impact: SDF cannot currently handle this timing scenario. However, with tool sets which can automatically generate an interconnect model as shown above, SDF will be able to represent the timing information through and about that interconnect model. Therefore no changes are required of SDF in order to accommodate this scheme provided that ancillary tools are intelligent enough to generate the net model.

3.2 Back-Annotation File Format Proposal

Proposal to standardize on SDF: We are proposing that the SDF be adopted as a strawman standard for back-annotation data exchange. SDF was designed for this purpose and is being widely adopted by tool vendors as well as ASIC suppliers and it seems well suited to meeting the requirements. It has been adopted as a standard for this purpose by OVI (Open Verilog International) and is in the public domain.

The SDF is a repository for timing data generated by EDA tools for use at any stage in the design process. The data in the SDF file is represented in a tool-independent way and can include:

- Delays: module path, device, interconnect, and port
- Timing checks: setup, hold, recovery, skew, width, and period
- Timing constraints: path and skew
- Incremental and absolute delays
- Conditional and unconditional module path delays and timing checks
- Design/instance-specific or type/library-specific data
- Scaling, environmental, technology, and user-defined parameters

The SDF is a medium by which timing and delay data are passed between EDA tools for use at different stages in the design process. The SDF file may be accessed through a procedural interface by which EDA tools are assured of consistent, accurate, and up-to-date data. This means that EDA tools can use data created by other tools as input to their own processes. Sharing data in this way, layout tools can use design constraints identified during timing analysis, and simulation tools can use the post-layout delay data.

3.2.1 Open Issues

While SDF as it is currently defined, provides a solution for the majority of situations which arise while interconnecting components, there are a number of techniques which are not currently accommodated in the SDF v1.0 design.

One of those issues involves the handling of Multiple Driver / Multiple Receiver. However, with the introduction of net models, SDF will be able to accommodate more advanced interconnection methodologies.

Final analysis of SDF's capabilities may reveal additional areas requiring work. However, as of this writing, SDF seems adequate to handle ASIC timing information.

3.3 Model Interfaces

Each model shall have an interface which adhere's to the specifications detailed below:

3.3.1 Logic System

Based upon VHDL's subtyping mechanisms and the overriding issue of assuring interconnect compatibility between model vendors and users, the following requirements have been developed.

- Ports shall use the STD_LOGIC_1164 data type or a subtype thereof as its interconnect port data type.
- Scalar ports shall be specified as std_logic data type or a subtype thereof.
- Array ports shall be specified as std_logic_vector data type. (Note: No reasonable subtype could be declared such as X01Z_vector which would ever be type compatible with std_logic_vector , therefore the requirement for std_logic_vector must apply).
- Models may convert the 9-state data type into a four state (i.e. X01Z) for their internal modeling requirements and acceleration.

3.3.2 Generic Parameters

With the interest of maintaining consistency in modeling style between one model and another, the following requirements for generic parameters have been developed.

- SysVoltage: There shall be a parameter to specify the voltage of the chip.
- SysProcess: There shall be a parameter to specify the process used in fabricating the chip.
- SysTemp: There shall be a parameter to specify the ambient temperature surrounding the chip.

3.3.3 Timing Parameters

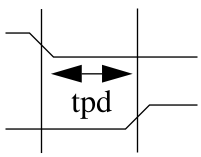
Timing for ASIC models can take a number of forms. The most basic is Intrinsic delay where a timing parameter specifies the non-loaded, nominal delay or timing parameter of the device. This delay value is usually affected by a number of external factors including:

- temperature
- voltage
- process

- input rise time
- cross-products between input rise time and other environmental factors

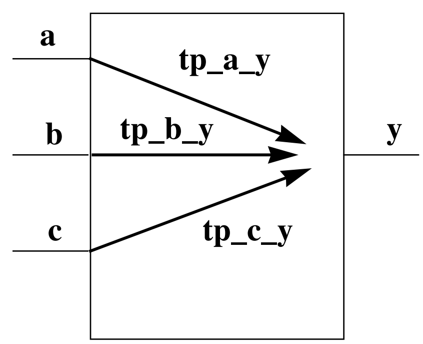
Therefore, with the diversity of styles with which ASIC foundries represent timing parameter derating equations, an arbitrarily general approach to derating must be implemented.

The model timing interface shall be capable of supporting the following parameters:



- Intrinsic Timing Data: The base timing data value of a given timing specification at a nominal temperature, voltage, process and loading. This parameter may be modified by factors based upon the environment in which the device is placed. The timing parameter from the 50% value transition point of an input stimulus to the corresponding 50% transition point of an affected output signal.
- Wire Delay: The additional delay, placed either prior to an input port, or after an output port which modifies the time at which a signal is seen to change value. Provisions should be made to support state dependent wire delays.
- Rise / Fall Time: The delay between the 10% and 90% value of a signal undergoing a transition from one state to another.
- TimeMode: There shall be a generic parameter which selects whether to use t_minimum, t_typical, or t_maximum time values.
- MinTypMax: There shall be a data type which provides for minimum, typical and maximum time values.
- Defaults: Default parameters shall exist for each of the generic parameters specified.
- Capacitive Load: There shall be a specification for the capacitance of a port as seen from the perspective of the driver of that port. In addition, there shall be a means of specifying a default capacitive load for every port in addition to the capacitive load for a given port.
(i.e. Cload_portname, and Cload_default)
- RiseDriveFactor: There shall be a parameter definable for the rise resistance from each input pin to each output pin or alternatively from any input pin to all output pins or Alternatively you can specify a default factor. If you fail to specify a default, the default will be set to 0. This value is a min:typ:max type.
(i.e. RiseDriveFactor and RiseDriveFactorDefault)

- FallDriveFactor: as above.
- RiseSlewRate: There shall be a parameter definable for the incremental delay on an input pin used to model input slew rate. Provisions shall be made to specify slew from each input to a given output or alternatively from any input to a given output. This value is a min:typ:max type.
- FallSlewRate: as above.
- RiseT ime: There shall be a parameter definable for the input rise time on a port.
- FallT ime: as above.
- Specifications for composite type ports shall apply to each of the individual subelements of that composite type.
- Path Dependent delays: A mechanism shall be developed which supports separate path delay values from more than one input to a single common output.

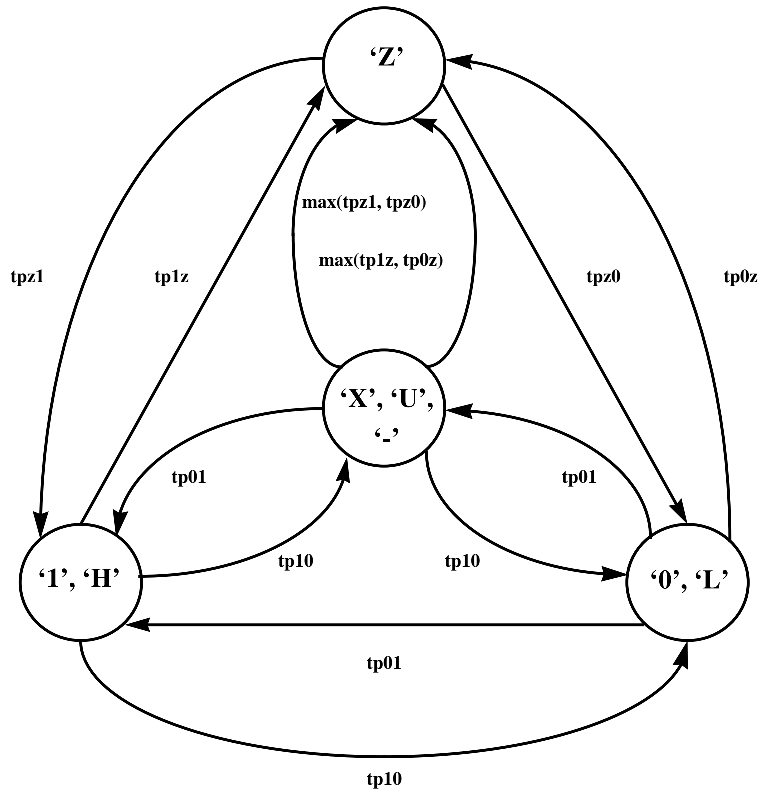


Timing delay information is typically described between input/output signal pairs. For example, a flip-flop may have a propagation delay from the rising edge of the clock to the time it takes for the Q output to change to a ‘0’. This timing specification is typically described by the identifier “tphl_clk_Q”.

In modeling the timing of a component, it is important to keep in mind that a given path from input to output may have more than one timing parameter associated with the path. There may be a number of parameters such as:

- tphl: propagation delay for the OUTPUT to go from ‘1’ to ‘0’.
- tplh: propagation delay for the OUTPUT to go from ‘0’ to ‘1’.
- tphz: propagation delay for the OUTPUT to go from ‘1’ to ‘Z’.
- tplz: propagation delay for the OUTPUT to go from ‘0’ to ‘Z’

- tpzl: propagation delay for the OUTPUT to go from ‘Z’ to ‘0’.
- tpzh: propagation delay for the OUTPUT to go from ‘Z’ to ‘1’.
- tr: rise time
- tf: fall time



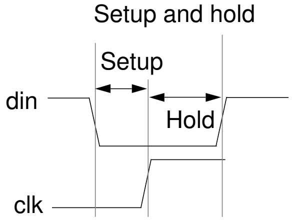
3.3.4 Value dependent delays

There shall be a mechanism for assigning a delay to an output signal with a delay which is dependent upon the new value of the output signal.

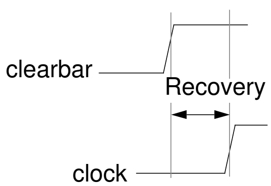
Optional: There shall be a mechanism for assigning delays which are dependent upon the input path state with possible relationship to the output state.

3.3.5 Timing Check Routines

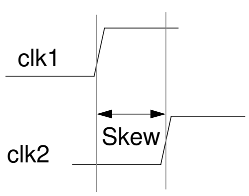
A number of timing checks must be supported in the modeling methodology:



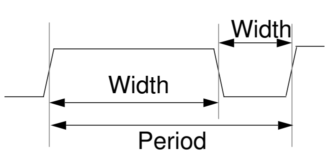
- The **SETUP , HOLD and SETUPHOLD** timing check represents limits on timing of a signal relative to a reference signal. The basic definition for setup and hold requirements is that the data signal must remain stable for a specified minimum time interval in order for a transition of the clock signal to store the data successfully. Setup time defines the minimum interval before the clock transition; hold time defines the minimum interval after the clock transition. Any change to the data signal within these intervals results in a timing violation. To shift the violation intervals with respect to the clock transition, either the setup time or hold time can be negative; however, their sum must always be greater than zero.



- The **RECOVERY** timing check is a constraint between a change in an asynchronous control signal and the next clock pulse, for example between clearbar and the clock for a flip-flop. If the clock signal violates the constraint, the output value is unknown—it can be the value set by the clearbar, or it can be the value clocked into the device.



- The **SKEW** timing check represents the limits for signal skew timing checks. A signal skew limit is the maximum allowable delay between two signals, which if exceeded causes devices to behave unreliably. For example, in a chain of devices such as counters that are connected to the same clock signal, the clock ideally changes at exactly the same time at all counters. However, there are some delay between the change at one device and the change at another. If this delay exceeds the signal skew limit, the data passed along the chain is unreliable.The skew limit between signals are identified as a design constraint by timing analysis tools.



- The **PULSEWIDTH** timing check is the maximum allowable time for the positive or negative phase of each cycle. If a signal has unequal phases, you can specify a separate width check for each phase.
- The **PERIOD** timing check represents the limits for period timing checks.The period timing check is the maximum allowable time for one complete cycle of the signal.

3.3.6 Naming Conventions

Parameter naming conventions shall be consistent, intuitive and unique.

Table 2: Transition Dependencies

Input Port	Output Port	prefix
‘0‘	‘1‘	tplh
‘0‘	‘Z‘	tplz
‘1’	‘0’	tphl
‘1’	‘Z’	tphz
‘Z’	‘0’	tpzl
‘Z’	‘1’	tpzh
‘0’ -> ‘1’	‘1’	tpph
‘1‘ -> ‘0‘	‘1’	tpnh
‘0’ -> ‘1’	‘0’	tppl
‘1’ -> ‘0’	‘0’	tpnl
any -> ‘1’	‘1’	tprh
any -> ‘1’	‘0’	tprl
any -> ‘0’	‘1’	tpfh
any -> ‘0’	‘0’	tpfl

3.3.7 Specifications (A TTRIBUTES)

Specifications contain information which is pertinent to a model, but is not information which is loaded or passed into the model either statically or during elaboration.

3.3.7.1 Synthesis Specifications

The specifications stated in this section are preliminary and are subject to considerable expansion in future versions of this document.

- **Area** : There shall be a specification for AREA in units of sq. microns.

- **GateCount** : A specification of gate count should be provided.
- **Power** : A specification for the device power dissipation expressed as a real number microwatts per Mhz shall be provided.
- **ConstraintMaxDelay** : Specifies the maximum allowed delay to a given output or from a given input to a given output.
- **ConstraintMaxCload** : Specifies the maximum allowed capacitive loading on a given output port.

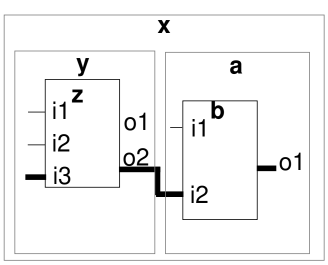
- The **SKEWCONSTRAINT** represents the same skew limit for multiple signal pairs.The signal in the *SKEWCONSTRAINT* construct is the reference signal (that is, the clock signal) against which all associated port signals are constrained.

3.3.7.2 Leaf-Level Specifications (Not related to synthesis)

- **MaxFanout** : Specifies the maximum fanout of a component or a given output port of that component
- **PinEquivalence** : Specifies multiple devices in the same package.

3.3.7.3 Design Constraints

The specifications stated herein are preliminary and are subject to change. This section pertains to constraints which may be specified for a design as a whole rather than within a leaf-level model



- .The **PATHCONSTRAINT** s are the critical paths in a design identified during timing analysis. Layout tools can use these constraints to direct the physical design. The constraint specifies the maximum allowable delay for a path, which is typically identified by the two ports at each end of the path. You can also specify intermediate ports to uniquely identify path(s).

- The **SUM** timing constraint limits the sum of the delays represented by two or more paths in a design to a specified maximum.

- The **DIFF** timing constraint limits the difference between delays represented by two paths in a design to a specified maximum.

3.3.8 Open Issues

A number of issues remain open for further discussion and needs analysis. These include:

- **Simultaneous event handling:** Event driven simulators, in general, have difficulties handling conditions where events and their corresponding pin-to-pin delays occur in such a time sequence that glitch conditions would potentially occur in an actual semiconductor device, but do not appear during simulation due to the handling of events.

3.4 Processing Model Timing Information

In order to support ASIC timing, the following set of requirements have been established:

3.4.1 Delays as a function of rise/fall time

- There shall be a mechanism for determining the propagation and output rise/fall delays as a function of the input rise/fall delay .
- The equations used to process this information must be able to be expressed in the most general of cases.

3.4.2 Derating Systems

There shall be a mechanism for derating the intrinsic delays and timing parameters by environmental factors such as

- capacitive loading,
- temperature
- voltage
- process variation.

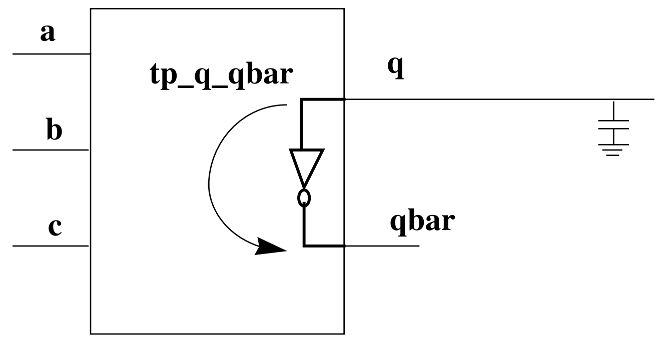
3.4.3 Handling X transitions

- In the case where an X represents an unknown case and the transition to that state is not otherwise specified (RAM and BUS timing models may be the exception), then delays should be specified on a pessimistic basis.

3.4.4 Requirement for the Propagation of Rise/Fall Data

There shall be a mechanism for statically propagating the value of output rise/fall times through a circuit‘s topology .

3.4.4.1 Q to Qbar loading effects



A number of devices have internal architectures where certain output ports are derived from other output ports. Under these circumstances, any loading which modifies the delay of the source output port will consequently modify the delay on the dependent output port as is shown in the graphic above.

3.5 Acceleratable Primitives

As a requirement to support accelerated gate-level simulation, a minimum of the following primitives must be supported.

3.5.1 Predefined Primitives

3.5.1.1 N-input Logical Gates

Any # of input ports, one output port

AND, NAND, OR, NOR, XOR, XNOR

3.5.1.2 N-fanout Buffers

Non-inverting and Inverting, single input, multiple output

BUF, NOT

3.5.1.3 Three-State Drivers

Non-inverting and Inverting three-state drivers with active high or active low enables

BUFIF0, BUFIF1, NOTIF0, NOTIF1

3.5.1.4 PULLUP , PULLDOWN

3.5.1.5 D-type Flip-Flop

3.5.1.6 JK T ype Flip Flop

3.5.1.7 Latches

3.5.2 Table Driven Primitives

A mechanism for representing boolean truth and state tables in an efficient coding format is highly desirable. V erilog implements this capability in its User Defined Primitive (UDP) construct. The requirements for such a construct are straight forward. Essentially , the primitive should implement Mealy and Moore machine state table representations as well as boolean truth tables. Furthermore, the construct should be able to represent edge-sensitive transitions