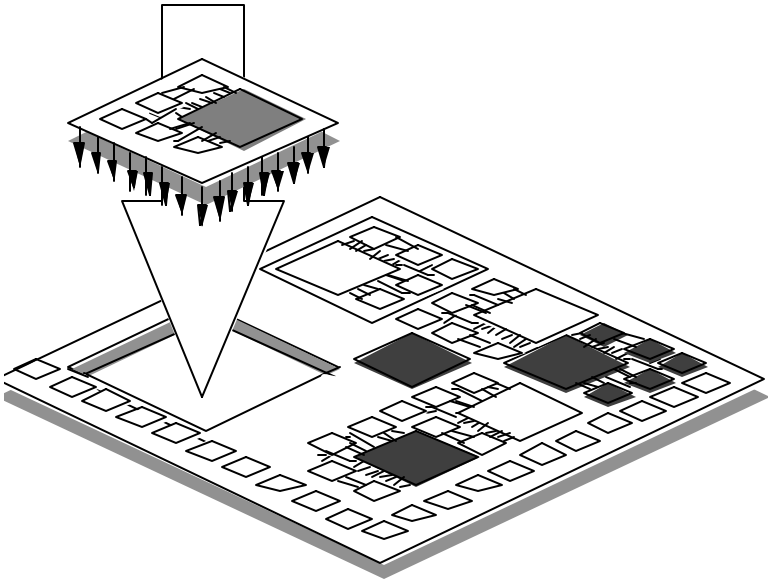
VSI Alliance™
Soft and Hard VC Structural, Performance and Physical Modeling SpecificationVersion 2.1

(I/V 1 2.1)

Implementation/Verification Development Working Group

January 2001



Dedication to Public Domain

VSI Alliance hereby dedicates all copyright that VSI Alliance holds in this _____ (the "Work") to the public domain, free of charge, and for the general benefit of the public at large.

VSI Alliance intends this dedication to be an overt act of relinquishment in perpetuity of all present and future rights that VSI Alliance may have in the Work under copyright law, whether vested or contingent, including without limitation, the right to prevent others from freely reproducing, distributing, transmitting, using, modifying, building upon or otherwise exploiting the Work for any purpose, commercial or non-commercial, or in any way.

VSI Alliance understands that such relinquishment includes the relinquishment of all rights to enforce (by lawsuit or otherwise) any copyrights that VSI Alliance may have in the Work.

IMPORTANT - NO WARRANTY. THE WORK IS PROVIDED "AS IS", "WHERE-IS", WITHOUT WARRANTY OF ANY KIND. WITHOUT LIMITING THE GENERALITY OF THE FOREGOING, VSI ALLIANCE EXPRESSLY DISCLAIMS ALL WARRANTIES WITH RESPECT TO THE WORK, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, WARRANTIES OF TITLE, NON-INFRINGEMENT OF INTELLECTUAL PROPERTY RIGHTS, AND IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Implementation/Verification Development Working Group
(I/V 1 2.1)

Members of the Development Working Group:

Ambit
Aristo Technology
ARM
Cadence Design Systems
Cypress Semiconductor
Fujitsu Microelectronics
LSI Logic
Mentor Graphics
Nortel
Synopsys

Active Contributors:

Andres Teene (Chair)..... LSI Logic
Mark Hahn (Co-Chair)..... Cadence Design Systems
Mitch Heins.....Ambit Design Systems
John Biggs..... ARM
Cedric Iwashina ..... Aristo Technology
Daniel Ryan ..... Cypress Semiconductor
Michael Berend.....Fujitsu Microeletronics
Patrick Offers..... Nortel
Karen Bartleson ..... Synopsys
Mike Andrews..... Mentor Graphics

Other Contributors:

Jin-Sheng Shyr.....Toshiba
Jerry Frenkil.....Sente

Technical Editor

Wilsa Schroers

Revision History

Revision 2.0 -Editorial Staff - Updated cover, legends to match current VSIA Standard Formatting 21Apr00

Revision 2.0 - Editorail Staff - Formatted for Education Revision 5Jun00

Revision 2.1 -Editorial Staff, H. Leeds - Updated document to comply with Deliverables rules 23Oct00

Table of Contents

List of Tables

List of Figures

1. Overview

1.1 Scope and Field of Use

This specification defines the VC data representation standards to support the hardware design flow from RTL design planning through final verification.

This specification includes the RTL source and performance model formats needed for both hard and soft VCs, summary of the design constraints requirements, endorsement of the OVI Design Constraints Working Group (DC-WG) for the development of the design constraints standard, append “API” to Open Library to give Open Library API, and the endorsement of the Open Library (OLA) standard for the performance modeling standard.

The specification includes the data formats that should be used for the structural netlist and a number of physical data types associated with hard VCs. The specification also includes a set of guidelines for reducing the likelihood of name-space collisions between different VCs, and between VCs and other system logic, as well as guidelines for handling any name-space collisions that do occur.

The “field of use” for VSIA standard data formats is defined as creating, defining, exchanging, and integrating descriptions of virtual components of integrated circuits.

1.2 Referenced IP

This specification references a number of data formats, including:

- Open Library API (OLA)
  - Owner: OLA task force
  - Status: under development
  - http://www.si2.org/ola
- Design Constraints Working Group (DC-WG)
  - Owner: OVI
  - Status: under development
  - http://www.vhdl.org/dcwg
- Verilog Synthesis Subset: IEEE 1364.1 Draft Standard Register Transfer Level Subset based on Verilog Hardware Description Language (the IEEE 1364.1 Draft specification is based on the OVI Verilog HDL Synthesis Subset specification)
  - Owner: IEEE
  - Status: IEEE Draft Standard
  - http://www.eda.org/vlog-synth
- VHDL Synthesis Subset: IEEE 1076.6 Draft Standard for VHDL Register Transfer Synthesis
  - Owner: IEEE
  - Status: IEEE Draft standard
  - http://standards.ieee.org/faqs/order.html
- Verilog: IEEE 1364-1995
  - Owner: IEEE
  - Status: Accredited standard
  - http://standards.ieee.org/faqs/order.html
- VHDL: IEEE 1076-1987
  - Owner: IEEE
  - Status: Accredited standard, older version
  - http://standards.ieee.org/faqs/order.html

- EDIF: EDIF 2 0 0
  - Owner: EIA
  - Status: Accredited standard, older version
  - http://www.edif.org/index.html
- Spice: VC Hspice 1.0a
  - Owner: Avant! Corporation
  - Status: Licensed through VSIA Technology Contribution Agreement
  - http://www.vsi.org
- SPEF: IEEE P1481, 1998
  - Owner: IEEE
  - Status: IEEE Ballot Standard
  - http://www.eda.org/dpc
- VC LEF: VC LEF 5.1
  - Owner: Cadence Design Systems, Inc.
  - Status: Licensed through VSIA Technology Contribution Agreement
  - http://www.vsi.org
- GDSII: GDSII Stream Format 6.0.0
  - Owner: Cadence Design Systems, Inc.
  - Status: Licensed through VSIA Technology Contribution Agreement
  - http://www.vsi.org

1.3 Definition of Terms

For clarity, the following terms are defined below. For a more detailed listing, see the Glossary.

RTL Source – Defines the VC source description and is the primary input for the implementation and verification of the VC within a system chip design.

Basic Delay Model – Defines timing specification of the VC.

Timing Analysis Model – Defines the static timing characteristics of the VC.

Power Model – Defines the power specification of the VC.

Peripheral Interconnect Model (PIN) – Specifies the interconnection RCs for the peripheral interconnect between the physical I/O ports and the internal gates of the VC.

Physical Blocks – A model of the physical implementation of the VC and the system chip.

VC Port – The pad or point of interconnection between the VC and the system chip.

Behavioral Model – Describes the component function and timing without describing the internal structure.

Cell Level Netlist – A structural interconnection of design objects ranging from simple logic gates to complex functions.

Circuit Level Netlist – A structural interconnection of semiconductor devices such as transistors, resistors, and capacitors.

1.4 Methodology Description

This specification defines the VC data representation standards to support the hardware design flow from RTL design planning through final verification. The details of the design flow and methodology are not part of the VSIA specification, but the standards selected by the DWG must support commonly used flows and methodologies.

Figure 1: VC Design Methodology illustrates the VC creation and integration portion of the VC Design Methodology that is supported by this specification.

The data representation standard requirements will be based on the commonly used implementation and verification tasks. Representative tasks include:

- Implementation Tasks
  - Design estimation
  - Logic design (RTL floorplanning, synthesis, detailed floorplanning, clock insertion, test insertion)
  - Physical design (module generation, datapath generation, place & route, IPO/ECO)
- Verification Tasks
  - Functional verification (event-driven simulation, cycle based simulation, emulation, FPGA emulation, formal verification)
  - Timing verification (delay calculation, parasitic extraction, timing simulation, timing analysis)
  - Power analysis, signal integrity analysis
  - Physical verification (LVS, DRC, ERC)

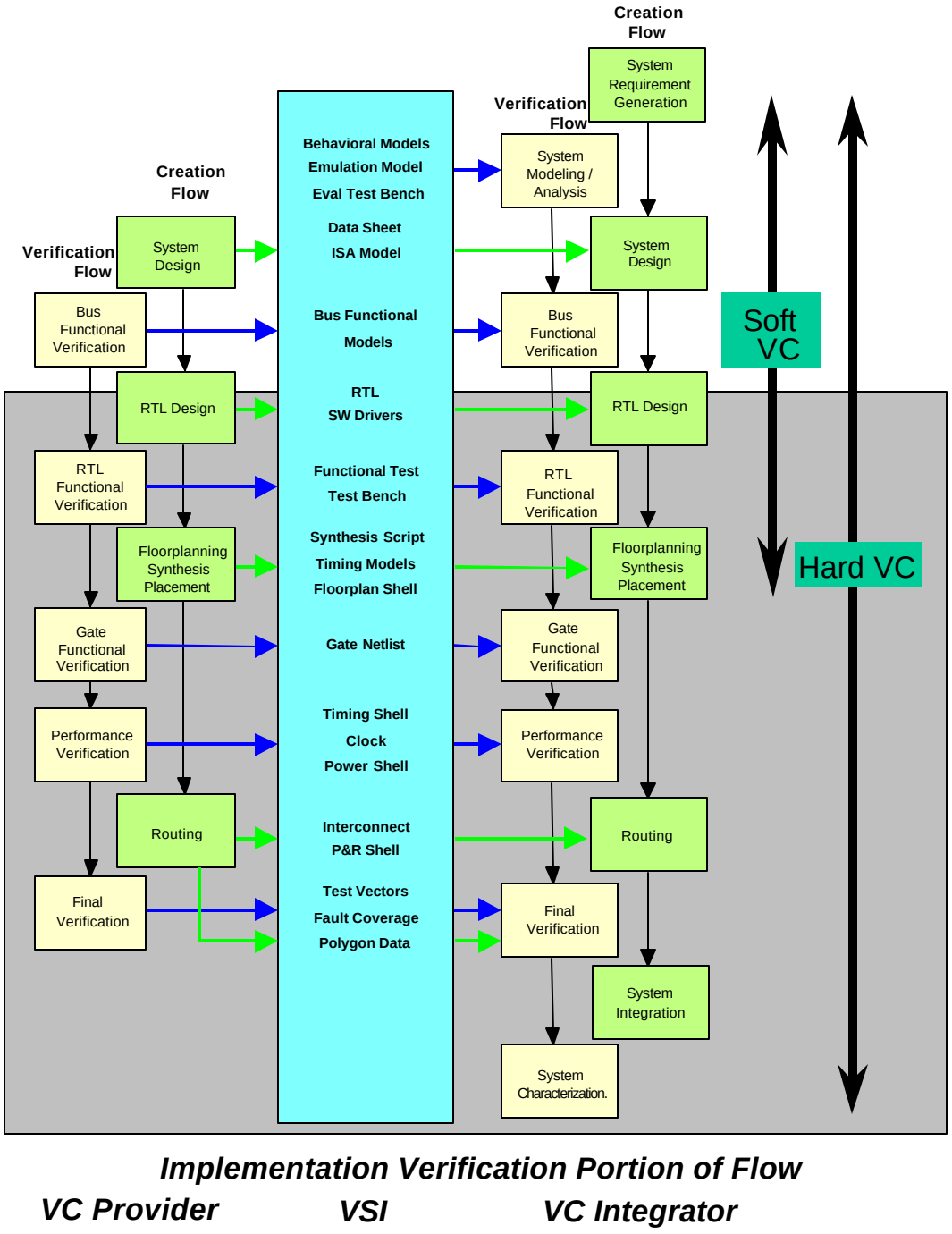


Figure 1: VC Design Methodology

1.5 Summary of Deliverables

These deliverables will include the definition of the data representation requirements, selection of the data representation formats, and the definition of appropriate VC guidelines as required for the implementation and verification of VCs and system chip designs.

The light shaded entries in Table 1 are the entries modified by this specification.

Table 1: VSIA Data Deliverables

Section	Deliverable	VSIA Endorsed Formats	VSIA Specified Formats	Soft	Firm	Hard	Comments
2.1	Implementation Level Behavioral and Structural Models						
2.1.1	RTL Source		Verilog/VHDL Synthesis Subset	M	CM	CR	Mandatory (M) for Firm VCs if required for synthesis. Required (R) for Hard VCs if formal verification or functional simulation is met.
2.1.2	Cell Level and Circuit Level Netlist		Verilog, EDIF-netlist, VC Hspice, VHDL	--	CM	CM	Mandatory (M) for Firm VCs if they are described at the gate level. Mandatory (M) for Hard VCs if layout versus schematic checking is required for these VCs.
2.2	Implementation Level Performance Models						
2.2.1	Basic Delay Model	OLA		R	M	M	
2.2.2	Timing Analysis Model	OLA		R	M	M	
2.2.3	Power Model						
2.2.3.1	Black/Gray Box Power Model Requirements	OLA		R	M	M	
2.2.3.2	RTL Source Power Model Requirements		Verilog/VHDL Synthesis Subset	M	--	--	
2.2.3.3	Cell Level Power Model Requirements		Verilog, EDIF-netlist, VHDL, SPEF	--	CR	CR	Recommended (R) if a more accurate power model is required.

Table 1: VSIA Data Deliverables (Continued)

Section	Deliverable	VSIA Endorsed Formats	VSIA Specified Formats	Soft	Firm	Hard	Comments
2.2.3.4	Circuit Level Power Model Requirements		VC Hspice	--	--	CR	Recommended (R) if a more accurate power model is required.
2.2.4	Peripheral Interconnect Model		SPEF	--	CR	CM	Recommended (R) for Firm VCs and Mandatory (M) For Hard VCs if the peripheral interconnect needs to be included in the delay calculation.
2.2.5	Circuit Level Interface Model Requirements		VC Hspice	--	CM	CM	Mandatory (M) if the IO or interfaces need to be modeled at the circuit level.
2.3	Physical Modeling						
2.3.1	Detailed Physical Block Description		GDSII	--	CM	M	Mandatory (M) if Firm VCs contain physical data.
2.3.2	Pin List/Pin Placement		VC LEF	--	CM	M	Mandatory (M) if Firm VCs contain physical data.
2.3.3	Routing Obstructions		VC LEF	--	CM	M	Mandatory (M) if Firm VCs contain physical data.
2.3.4	Footprint		VC LEF	--	CM	M	Mandatory (M) if Firm VCs contain physical data.
2.3.5	Power and Ground		VC LEF	--	CR	M	Recommended (R) if Firm VCs contain physical data.
2.3.6	Signature		VC LEF	--	--	M	
2.4	Design Constraints						
2.4.1	Timing Constraint	DC-WG		M	M	--	
2.4.2	Clock Constraints	DC-WG		M	M	CM	Mandatory (M) if the clock signal requirements need to be provided to the Hard VC.
2.4.3	Logic Architecture Constraints	DC-WG		M	--	--	
2.4.4	Area Constraints	DC-WG		R	CM	--	Mandatory (M) if Synthesis or Floorplanning is required.

Table 1: VSIA Data Deliverables (Continued)

Section	Deliverable	VSIA Endorsed Formats	VSIA Specified Formats	Soft	Firm	Hard	Comments
2.4.5	Physical Implementation Constraints	DC-WG		CM	CM	--	Mandatory (M) if Floorplanning is required.
2.4.6	Power Constraints	DC-WG		CM	CM	--	Mandatory (M) if Synthesis or Floorplanning is required.
2.4.7	Test Constraints	DC-WG		M	M	M	
2.4.8	Environmental / Operating Constraints	DC-WG		M	M	M	

1.6 Endorsements

VSIA Implementation/Verification DWG will work with other standards organizations for the development of new or emerging standards required for implementation and verification of VCs and system chip integration. The I/V DWG will endorse the development of select standards provided the I/V DWG requirements are included in the development of the standard. The standard will be specified by the I/V DWG when the standard is completed, approved by the standard organization, and meets the I/V DWG requirements.

The purpose of the endorsements are to allow IP providers, system integrators, and EDA developers to prepare development plans needed to support the future adoption of these emerging standards.

The following outlines the I/V DWG specification and endorsement process:

- Define the VSIA requirements for the integration/verification of VCs.
- Identify potential defacto, accredited or emerging standards.
- Work with standards organization to endorse the emerging standard if defacto or accredited standards are not available or sufficient.
- Encourage participation from VSIA members and industry.
- Provide VSIA inputs and requirements necessary for the development of the standard.
- Endorse the standard development, if the VSIA requirements are included.
- Participate in the reviews of the draft standard.
- Specify the standard for the VSIA specification when the standard is completed and approved, provided it meets the VSIA requirements.
- Promote the use of the standard.

1.6.1 Design Constraints Endorsement

The I/V DWG endorses the OVI Design Constraints Working Group (DC-WG) development of a Design Constraints standard. The I/V DWG is currently working with the OVI DC-WG to define the group charter, scope, and requirements. The design constraint requirements are specified in section 2.4.

1.6.2 Performance Modeling Endorsement

The performance modeling and library attribute data requires advanced features and capabilities to support the emerging very deep sub-micron process and design technologies. The I/V DWG endorses the development of the Open Library API (OLA) standard to provide the advanced capabilities needed to support the emerging technologies. We strongly encourage the development of the OLA standard, OLA based EDA tools, and ASIC libraries to provide the infrastructure needed for the production deployment of OLA for VC deliverables, and system chip design, and verification.

Table 2 lists criteria to define the requirements and dependencies necessary for the production deployment of OLA. They are based on when VC providers and system chip designers can effectively use OLA for VC exchange and system chip design. The criteria define the specific VSIA trigger points and target timeframe for the adoption of OLA as a VSIA standard. The criteria will be monitored and the target timeframe will be updated accordingly.

Table 2: OLA Requirements and Dependencies

Criteria	Requirements	Summary
Technical Suitability	The OLA standard must support the requirements for the Performance Models defined in section 2.2. Trigger: Standard needs to meet the VSIA requirements.	The OLA is an API based standard and is a combination of DCL for delay calculation and ALF for function and attribute specification.
Process Technology	The advanced process technologies will be one of the primary drivers for the adoption of OLA. Trigger: Production availability of the advanced .25 and .15 um process technologies.	The modeling requirements for the advanced .25 and .15 um technologies will drive the need and use of OLA. It is expected the .5 and .35 um technologies will continue to use the existing formats.
ASIC Library availability	ASIC libraries supporting OLA is a key OLA infrastructure requirement for the VC and system chip integration. Trigger: Availability of a significant number of OLA ASIC libraries for a specific generation of process technology.	The initial set of production ASIC OLA libraries will be available by Q1 1999.
EDA tool support	Mainstream EDA tool support of OLA is also a key OLA infrastructure requirement for VC and system chip integration. Trigger: EDA tools that support a significant number of design flows.	The initial set of EDA tools supporting OLA is expected to be released by the end of 1998.
Standard availability	The specification of the OLA standard is required. Trigger: Completed specification that includes the VSIA requirements.	The initial OLA draft standard is expected to be available by the end of 1998. The initial version of the standard may not include all the necessary requirements.
DWG Sponsored Pilot	A DWG sponsored pilot will be required to validate that OLA can be utilized for VC and system chip integration. Trigger: Completion of DWG sponsored pilot.	The pilot needs to address applicability of OLA to complex IP functions, VC functions from multiple sources, integration of interconnect delay algorithm, and complex delivery and support mechanism.

The target timeframe for the VSIA specification of OLA for Performance Modeling and VC Attributes standard is late 1999 to early 2000. This is based on the expected initial introduction of OLA ASIC libraries and OLA based EDA tools in early 1999.

2. Specification of Deliverables

2.1 Implementation Level Behavioral and Structural Models

Behavioral models describe the component function and timing without describing the internal structure. Structural models represent a component in terms of the interconnection of the constituent components.

Detailed model definitions are specified in the “VSIA System Level Design Model Taxonomy” document. This document is located at web site: http://www.vsi.org/library.

2.1.1 RTL Source

The RTL source defines the VC source description and is the primary input for the implementation and verification of the VC within a system chip design. The RTL source is required for synthesis of soft VCs. The RTL source is also required for both VC and chip level verification using both simulation and formal verification. Refer to the “VSIA System Level Design Model Taxonomy” document for more information.

Formats

- Verilog Synthesis Subset: IEEE 1364.1 Draft Standard Register Transfer Level Subset based on Verilog Hardware Description Language
- VHDL Synthesis Subset: IEEE 1076.6 Draft Standard for VHDL Register Transfer Synthesis

General Usage

The RTL source file should be written in a standard RTL language. The RTL source should consist of both structural and synthesizable code. The RTL synthesizable code should conform to the synthesizable VHDL or Verilog HDL subset. The RTL source should be written in a modular form that clearly identifies:

- Memory functions to be instantiated through generators, or specified as pre-designed library elements.
- Structured functions where generators or pre-existing blocks are to be used for performance, area, or power — such as data flow structures.
- **Note:** Structures that are instantiated require the availability of the required generators, libraries, and appropriate supporting views.
- Synthesizable logic.

Generators are specialized functions or tools used to implement specific design functions. When utilized, the generators should be elaborated and the results instantiated in the RTL source. The results of the generators include instantiated modules or components, structural design descriptions, or synthesizable code.

The RTL source name space should follow the guidelines as described in the appendix. No assumptions are made to which language is used in describing the VC.

We recognize the requirement for optional encryption of the RTL source, however the VSIA has not yet specified standard mechanisms for encryption. The VSIA IP Protection DWG is investigating this.

Applicability

- Soft VCs: MANDATORY

  The RTL source is used for the synthesis of soft VCs.
- Firm VCs: CONDITIONALLY MANDATORY

  The RTL source is used for portions of firm VCs that are to be synthesized.
- Hard VCs: CONDITIONALLY RECOMMENDED

  The RTL source is the source description for the VC. The RTL source is needed if formal verification is used for functional verification or if functional simulation is used and no other simulation models are provided.

2.1.2 Cell Level and Circuit Level Netlist

For firm and hard VCs, a structural netlist of blocks, generic gates, library-specific gates or transistors is required or recommended in certain circumstances. Refer to the “VSIA System Level Design Model Taxonomy” document for more information.

Formats

- Verilog: IEEE 1364-1995
- VHDL: IEEE 1076-1987
- EDIF: 2 0 0
- Spice: VC Hspice

Relationship between VC Hspice and the Hspice Simulator

VSIA selected the Hspice language as the starting point in developing a standard for a device-level structural netlist format. VSIA worked with Avant! to define the “VC Hspice Structural Netlist Specification,” which is essentially a subset of the Hspice language.

The subset was carefully chosen, and several minor modifications were made to increase generality. Particular attention was given to ensuring that the specification does not include any references to simulator features, device models, or device parameters that are proprietary (including any that are proprietary to the Hspice simulator).

Therefore, VSIA believes there is nothing inherent in the standard itself that requires the use of the Hspice simulator.

General Usage

The structural netlist may be used as an input to synthesis, to floorplanning, or as a final technology-specific deliverable.

Circuit Level Interface Netlist Usage

For transistor level descriptions, the complete structural description of the interface includes:

- The portion of the circuit contained within the VC(s).
- The remainder of the circuit, which is part of the system chip.

The circuit level interface netlist supplied by the VC provider should describe the portion of the circuit contained within the VC. The VC integrator should create a structural description of the remainder of the circuit, and combine it with the circuit level interface netlist for each VC.

The models for the devices used in the VC portion of the circuit should be referenced by the circuit level interface netlist, but not defined by it. Non-proprietary model parameters for each device instance in the VC portion of the circuit should be specified in the circuit level interface netlist. Control information should be specified by the VC integrator.

Proprietary model parameters may either be included in the circuit level interface netlist or specified in a separate file that is not part of the VSIA standard. Using a separate file may improve interoperability by hiding the proprietary parameters from simulators that do not support them. However, the syntax for some proprietary model parameters requires that they be specified on the device instance, which means they must be included in the circuit level interface netlist.

Applicability

- Firm VCs: CONDITIONALLY MANDATORY

  As the definition of a firm VC can range from RTL with topological planning to a technology-specific placed netlist, the presence of a structural netlist is mandatory if the VC is described at the gate level (either in the target technology or in a generic library).
- Hard VCs: CONDITIONALLY MANDATORY

  Layout versus schematic checking is usually performed by the VC Integrator. The VC Integrator usually performs layout versus schematic checking. As this compares the structural netlist and the detailed physical block description, the structural netlist is mandatory for a hard VC when layout versus schematic checking is required.

- Soft VCs: NOT REQUIRED

2.2 Implementation Level Performance Models

These deliverables define the VC design data needed for the timing/power driven implementation and verification of the VC/System Chip design. These consist of the VC delay, timing analysis, and power models. These performance models are required for logic synthesis, static timing analysis, power analysis, and any other design task that requires delay and power calculation.

Applicability

The performance model applicability is consistent for the Basic Delay Model and Timing Analysis Model for VC deliverables.

- Soft VCs: RECOMMENDED

  This model is not required because the accuracy without the presence of a cell library or interconnect structure is limited. If the provider has verified the VC against a reference environment, a model may be provided that is usable to the VC user. However, this provides limited information on how the block will behave in the customer application — primarily due to variances in layout and interconnect delays.
- Firm VCs: MANDATORY

  The interconnect delay approximations are based on a topological layout description and are considered sufficiently predictive for preliminary system-level timing analysis. Therefore, the base timing model is required for firm VCs.
- Hard VCs: MANDATORY

  This model is required. A gate or transistor-level static timing analysis with characterization for temperature, process, and voltage variations and a curve fit should be performed. Accuracy will be limited by the tools used and should be backed up with a SPICE analysis of the critical paths. The process used for characterization of the Hard VC should be listed as part of the VC documentation.

Black and Gray Box Models

For the representation of the VC independent of the peripheral interconnect model, there are two modeling styles that may be used.

In the black box modeling style, none of the internal structure within the VC is exposed in the model. This style has traditionally been used for small SSI/MSI level standard cell functions, and it is widely supported by EDA tools. However, it has serious limitations for cases such as transparent latches on the boundary of the VC, internal generated, gated, or muxed clocks, and VCs that use multiple clocks.

In the gray box modeling style, some of the internal structure within the VC is exposed in the model, but the model hides most of the detail of combinational logic. Generally, a gray box model contains all of the sequential devices. For each pair of sequential devices connected by combinational logic, the model includes a representative timing arc, which consolidates the timing of all of the combinational logic paths into a single delay model. In certain cases, such as internally generated, gated, or muxed clocks, the gray box model may include additional details of the internal structure, so the tools using the model can more accurately analyze the effect of the environment around the VC on the timing within the VC. Gray box models are not yet widely supported by EDA tools.

Note: In the following sections, refer to the “VSIA System Level Design Model Taxonomy” document. This document can be found at web site: http://www.vsi.org/library.

2.2.1 Basic Delay Model

The basic delay model defines the timing specification of the VC. The basic delay model is required for delay calculation and is the basis for the timing analysis model.

Format

The I/V DWG endorses the development of the Open Library API (OLA) standard as the VC Basic Delay Model standard.

Currently Used Formats: Liberty, TLF, <ASIC vendor formats>

General Usage

The characterization of the embedded block in the time domain is an important element in delivering VCs that can be embedded. The typical methods for characterizing small SSI/MSI level standard cell functions have matured to provide a fairly accurate approximation of the actual cell behavior. This has typically been achieved through multiple SPICE simulations followed by curve fitting techniques. Unfortunately, this technique is limited by the practical bounds of SPICE simulation. For larger VCs, an alternative method is needed that combines static timing analysis with SPICE to provide a useful characterization. Furthermore, a method is needed to properly handle the I/O boundary conditions affecting the timing model and the true behavior over all possible clock domains.

The timing specification format remains relatively constant over the various types of VC. However, the accuracy of the characterization model improves as the design implementation becomes more technology specific. The format proposed consists of two fundamental representations, which are combined to provide a complete characterization. The first representation covers the VC independent of the peripheral interconnect, and the second (section 2.2.4) addresses the peripheral portion of the block recognizing the possibility that nets extending beyond the block boundaries should be modeled in the context of use at the system-level. A discussion of the utility for each model in light of the VC type is included.

The basic delay model includes:

- Path Delays: These are expressed as delay models associated with timing arcs, which model the maximum and minimum delays of timing paths in the block through various types of devices. These include delays through combinational logic, delays from sequential devices, delays through tristate devices, and delays of asynchronous clear or set paths through sequential devices. The delay is typically expressed as a function of output load and input slew, but other factors such as process, voltage, and temperature may be considered.
- Signal Slew Rates: These are expressed as slew models associated with timing arcs, which model the maximum and minimum slew rates of output signals. The slew is typically expressed as a function of output load and input slew.
- Timing Checks: These are expressed as timing check models associated with timing arcs, which provide the data used by the model consumer to perform a comprehensive set of timing checks such as those defined in Standard Delay Format (SDF). Refer to the SDF manual for a complete list and a detailed description of the various timing checks.

The delay calculation model should accurately model the dependence of the signal delays and slews on the environment of the block given the technology in which the block is implemented. For current technologies, the two dimensional table lookup approaches used in the industry seem to be sufficient. The model should have the flexibility to incorporate more sophisticated delay calculation approaches, which are likely to be required for new technologies. The recommended methodology for generating the model is through static timing analysis for all synchronous blocks. The model should include as a reference either:

- The technology characterization parameters used in the model generation process (temperature, process, voltage)

  or
- The reference environment used in the VC layout

The model format for the VC is executable at the next level of timing analysis and suitable for use by timing simulation or system-level static timing verification.

2.2.2 Timing Analysis Model

The timing analysis model describes the static timing characteristics of the VC. The timing analysis model includes the interface and timing arc attributes that are not included in the basic delay models, but are necessary for static-timing analysis.

Format

The I/V DWG endorses the development of the Open Library API (OLA) standard as the VC Timing Analysis Model standard.

Currently Used Formats: Liberty, TLF, <ASIC vendor formats>

General Usage

Basic Timing Analysis Requirements

The timing analysis models includes path delay, slew rate and timing checks as specified for the basic delay model (section 2.2.1).

- State-Dependent Timing and Modes of Operation: These include the ability to specify that a path delay or timing check is valid only when the values of input signals or internal state variables satisfy a given condition; the ability to specify that a path delay or timing check is valid only if the block is in a given mode of operation; the ability to specify that a mode of operation for the block will be selected if input signals or internal state variables satisfy a given condition; and the ability to specify timing checks and the maximum and minimum delays across a set of related state-dependent delays or checks.
- Clock Networks: The ability to specify the insertion delay and skew of clock networks within the block; the ability to specify the waveforms of clock signals generated within the block from another clock waveform provided to the block; and the ability to place restrictions on the clock waveforms and their relationships to ensure the validity of the model.
- Functional Information: The ability to specify the Boolean functions of some output or internal signals in terms of input signals and internal state.
- Support for Multiple Operating Conditions: The ability to specify the delays and slew rates at multiple operating conditions (process, voltage, and temperature). Also the ability to specify the range of intra-die variation in operating conditions for which the model is valid.
- Design Properties: The ability to specify the input and output port capacitance of the model. Also the ability to specify minimum and maximum limits on capacitance, transition or fanout for the model or its ports.
- Support for Multicycle and False Paths: The ability to represent multicycle paths in the model and the constraints on clock waveforms that guarantee they will be evaluated as multicycle paths. Also the ability to represent false and multicycle paths that exist at the interface of the block and may or may not become false or multicycle when the model is used in a design.
- Compression of Identical Timing: The ability to specify that a set of signals (for example, an address bus) has identical worst-case timing. A single timing behavior can be specified that applies to all members in the set.
- Level of Accuracy: The ability to specify the accuracy assumptions under which the model for an IP block is generated. Static timing applications can use this information to derive the worst case timing of the block.
- User Attributes: The ability to define user attributes and specify their values.

Requirements for Latch-Based Designs

- Support the specification of the timing behavior of latch-based designs including the correct modeling of time borrowing behavior across an arbitrary number of latches.

Requirements for Circuit Level Design

- Dynamic and Domino Logic: The ability to specify that a transition on an output signal or a setup check occurs only if the transition on the input signal is in a certain direction (falling or rising but not both).
- Pass Transistors: The ability to model pass transistors that are connected to input or output ports of the design as switches that affect the peripheral parasitics of the IP block.
- Requirements for Peripheral Interconnect
- Preserve the Parasitics of Peripheral Interconnect: The ability to preserve the detailed parasitic information of peripheral interconnects of an IP block including feedthrough interconnect. This information should be accessible by static timing analysis applications.
- Physical Connection Points: The ability to represent the possible physical connection points of peripheral nets. Using a different physical connection point results in a different interconnect delay.

2.2.3 Power Model

The power model defines the power specification of the VC. The power specification format does not necessarily remain the same for the various types of VCs. However, the accuracy of the model improves as the design implementation becomes more detailed. Special power access should be properly described and defined, as appropriate, in VC LEF format.

The VC power model will depend on at least one, and often many, atomic power models. An atomic power model is a model that contains no hierarchy and depends on no other models. An example of an atomic power model is a nand gate or a flip-flop, although atomic models are not necessarily limited to such low low-level functions.

Any power model should be capable of representing both dynamic power and static power, where the dynamic power may be due to capacitive loading or short-circuit currents, and the static power may be due to state-dependent static currents.

General Usage

The power model is required for all types of power analysis: average, peak, RMS, etc. The type of model delivered will affect the resulting accuracy for any analysis. In general, the lower the abstraction level, the more accurate the model is. However, the use of lower level representations will compromise analysis run-times and feasibility. For example, circuit level power analysis of multi-million transistor chips for sufficiently long vector sets is at best exceedingly slow and is usually impractical. Thus, the type or model utilized may vary during the design process depending on the phase of the design and the analysis requirements.

Basic Power Analysis Requirements

Any power analysis should include effects caused by the following conditions and events:

- Switching activity on input ports, output ports, and internal nodes
- State conditions on input ports, output ports, and optionally internal nodes
- Modes of operations
- Environmental conditions such as supply voltage and external capacitive or resistive loading.

The VC power model can be provided as atomic black/gray box model for power analysis of hard, firm, and soft VCs. Alternatively the VC power analysis can also be implemented with RTL, cell level, or transistor level models.

2.2.3.1 Black/Gray Box Power Model Requirements

The black/gray box power model is an atomic VC power model that can be used for RTL or structural cell level power analysis. This can provide the appropriate level of accuracy for most applications.

Format

The I/V DWG endorses the development of the Open Library API (OLA) standard as the VC power model standard for black/gray box power models.

Currently Used Formats: Liberty, Sente

Applicability

- Soft VCs: RECOMMENDED

  This model is not required because the accuracy is limited without the presence of a cell library or interconnect structure.
- Firm VCs: MANDATORY
- Hard VCs: MANDATORY

2.2.3.2 RTL Source Power Model Requirements

RTL source level power model is an alternative approach for RTL power estimation. The RTL source level model defines the functionality and structure of the design, which along with the technology information can be used to estimate RTL power.

Format

A synthesizable RTL source description, as defined in section 2.1.1, is used for a RTL source power model.

A library of atomic models and technology information from which power characteristics can be extracted is also needed for the power analysis; this may be obtained from the VC provider or silicon vendor.

Applicability

- Soft VCs: MANDATORY
- Firm VCs: not required
- Hard VCs: not required

2.2.3.3 Cell Level Power Model Requirements

The structural gate/cell level power model can provide a more complete power model with a higher level of accuracy than the black/gray box power model. However, this does require that more detailed information be provided with the model.

Format

A structural netlist, Verilog, VHDL, or EDIF netlist, as defined in section 2.1.2, and a set of (either estimated or actual) interconnect parasitics in SPEF format, as defined in section 2.2.4, are used for the structural power model.

A library of atomic models is also needed for the power analysis. This library information can be obtained from the VC provider or silicon vendor.

Applicability

- Soft VCs: not applicable
- Firm VCs: CONDITIONALLY RECOMMENDED

  The cell level power model is conditionally recommended when a more accurate power model is required.
- Hard VCs: CONDITIONALLY RECOMMENDED

  The cell level power model is conditionally recommended when a more accurate power model is required.

2.2.3.4 Circuit Level Power Model Requirements

The circuit level power analysis provides the most accurate power analysis, however it is the most time consuming.

Format

A circuit level netlist in VC Hspice format, as defined in section 2.2.5, is used for accurate transistor level power analysis. The descriptions should include all interconnect parasitics.

Currently Used Formats: SPICE

Applicability

- Soft VCs: not applicable
- Firm VCs: not applicable
- Hard VCs: CONDITIONALLY RECOMMENDED

The circuit level power model is conditionally recommended when the most accurate power model is required.

2.2.4 Peripheral Interconnect Model

The peripheral interconnect model (PIM) represents the interface interconnect network shell around the internal VC delay model as shown in Figure 2: Peripheral Interconnect Model. The peripheral interconnect model specifies the interconnect RCs for the peripheral interconnect between the physical I/O ports and the internal gates of the VC. The PIM should not include any gates or device loading; these should be included in the internal VC delay model. This provides a separation of the peripheral interconnect RCs from the VC intrinsic delays as expressed by the basic delay model. By preserving the peripheral interconnect model, delay calculation at the next hierarchy level can be performed using the actual interconnect rather than an inaccurate approximation of loading and interconnect.

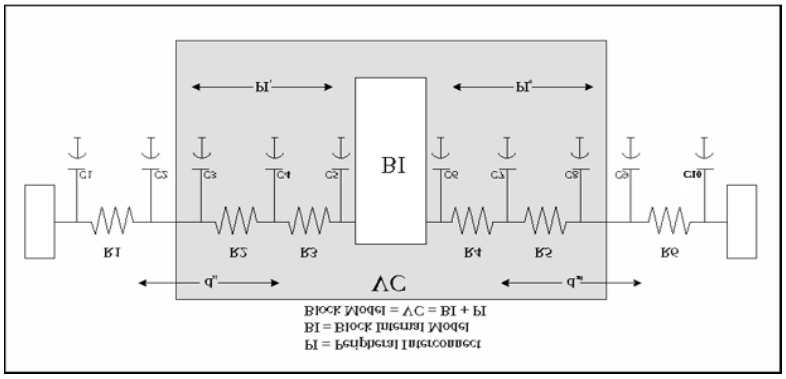


Figure 2: Peripheral Interconnect Model

Format

The fully distributed RC tree representation in SPEF IEEE P1481, 1998, is used for the Peripheral Interconnect Model.

Usage

The peripheral interconnect model (PIM) is used to accurately calculate the interconnect delays and output cell delays associated with the VC. The delay calculation system requires an accurate and complete interconnect model to accurately determine effective output loads, cell delay, and interconnect delay. The PIM describes the accurate interconnect model for the input and output ports of the VC.

The PIM for each VC port is merged with the extracted interconnect model for the routed net at the next level of hierarchy to generate a complete interconnect model for the specific net. Because of this, the fully distributed RC tree representation of SPEF should be used, rather than a lumped or reduced representation. The complete interconnect model is used by the delay calculation system along with the delay model to calculate the output cell delay for the driver of the net and the interconnect delay for the net. These delays can be represented in the SDF format.

Applicability

Hard VCs: CONDITIONALLY MANDATORY

- The peripheral interconnect model is required for hard VCs where the peripheral interconnect between the I/O ports and the internal gates of the VC is clearly understood (see Figure 2: Peripheral Interconnect Model).
- The peripheral interconnect model is not needed for hard VCs that do not have any appreciable interconnect between the I/O ports and the internal gates of the VC. In this case, the delay calculation system would calculate the interconnect and cell delay using the basic delay model for the VC with the routed interconnect model.

Soft VCs: not required

Firm VCs: CONDITIONALLY RECOMMENDED

- The peripheral interconnect model is recommended for firm VCs where the peripheral interconnect between the I/O Ports and the internal gates of the VC is clearly understood (see Figure 2: Peripheral Interconnect Model).

Related Guidelines

The capacitance associated with the interconnect described by the peripheral interconnect model may be affected by cross coupling from system chip level routing. The peripheral interconnect model should reflect the worst case values for capacitance which might occur in any usage of the VC. It may be possible to minimize the cross-coupling effect by creating obstructions around the peripheral interconnect.

2.2.5 Circuit Level Interface Model Requirements

A circuit level netlist describing the electrical characteristics of the VC interface is recommended for certain cases where the normal delay models do not apply.

Format

VC Hspice is used for this information.

Currently Used Formats: SPICE

Usage

The circuit level interface netlist is used to do delay calculation and signal integrity analysis on the interface nets for these special cases. It is not intended to describe the internals of the VC, as this would duplicate the information in the structural netlist and the basic delay model.

Applicability

Hard VCs: CONDITIONALLY MANDATORY

The circuit level interface netlist is mandatory for hard VCs in the following cases. The following list illustrates the types of special cases that may occur. Other special cases may also exist.

- When the VC contains I/O buffers which drive off-chip.
- When the VC interface includes analog pins.
- When the VC interface includes pins which use non-standard voltage swings.
- When the VC interface includes pins that operate at a high-speed frequency, where the normal delay models do not apply.

Soft VCs: not required

Firm VCs: CONDITIONALLY MANDATORY

- See Hard VCs below

2.3 Physical Modeling

This specification addresses only the physical block implementation of hard VCs. A future specification will address the physical block implementation of soft and firm VCs.

The physical block description for Hard VCs consists of two models of the physical implementation.

The first model is a detailed description of the physical implementation of the VC at the polygon level. The preferred data format for this model is GDSII-Stream 6.0.0, and the use of this model is described in more detail in section 2.3.1.

The second model is an abstraction that contains enough information to enable floorplanning, placement, and routing of the system level chip containing the VC. The preferred data format for this model is the MACRO section of VC LEF 5.1. The VC LEF model contains the following items, which are described in more detail below:

- Footprint
- Interface pin/port list, shape(s), and usage
- Routing obstructions within the VC
- Power and ground connections
- Signature

2.3.1 Detailed Physical Block Description

The detailed physical block description is a model of the physical implementation of the VC at the polygon level.

Format

GDSII-Stream 6.0.0 is used for this information.

Currently Used Formats: GDSII, LEF/DEF, SPICE

Usage

The GDSII file for a VC is required for:

- Accurate interconnect parasitic extraction for routing over the VC.
- Flat physical design rule verification as part of the overall chip assembly task.
- Mask generation.

A GDSII "donut" describes only the area of the VC close to the boundary, as shown by the gray area in Figure 3: GDSII Donut.

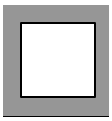


Figure 3: GDSII Donut

A VC provider may optionally create a GDSII donut to allow certain tasks to be performed by an integrator who is not given access to the complete GDSII. This approach may not be suitable if the pins of the VC are not located on the boundary.

Applicability

Soft VCs: not required

Firm VCs: CONDITIONALLY MANDATORY

- Required for those portions of Firm VCs that contain physical data.

Hard VCs: MANDATORY

Related Guidelines

A VC provider may consider the block description for a Hard VC to be particularly valuable intellectual property. If the VC provider chooses not to make the block description available to the VC integrator, it is important to carefully consider how this affects several tasks that might otherwise be done by the integrator:

- When system chip level routing is allowed over portions of the VC (which is controlled by creating or not creating routing obstructions), the block description for the VC is required for accurate extraction of the parasitics associated with the chip level routes. It is also required to ensure that the chip level routes do not cause physical design rule violations or signal integrity/noise problems within the VC.
- Physical verification of the connections to the VC.

Flat physical design rule verification requires merging the block description for all of the VCs in the system chip. VC providers should work with manufacturers to ensure consistency between the VC block descriptions and the GDSII descriptions for other system chip logic.

2.3.2 Pin List/Pin Placement

A physical description of the interface pins of the VC is required, including size, location, shape, connection layers, and equivalent pins.

Format

VC LEF is used for this information.

Currently Used Formats: GDSII, LEF/DEF, SPICE

Usage

The pin description is required for placement of the VC within a system chip, routing, and parasitic extraction, among other tasks.

The pin shapes should intersect an appropriate grid structure for placement and routing.

Pins may not be located within routing obstructions, except for a special case: when a pin is located within an obstruction and an edge of the pin is one grid away from the edge of the obstruction. In this case, the router is expected to connect to the pin despite the obstruction.

Applicability

Soft VCs: not required

Firm VCs: CONDITIONALLY MANDATORY

- Required for those portions of Firm VCs that contain physical data.

Hard VCs: MANDATORY

2.3.3 Routing Obstructions

A description of the obstructions within the VC is required, including size, shape, and affected layers. This information is required for system chip level routing over the VC. The obstructions should be defined in a way that ensures that routing within the non-obstructed areas will not cause design rule violations, crosstalk, or other noise-related issues.

Format

VC LEF is used for this information.

Related Guidelines

It is often desirable to simplify the actual obstructions within a hard VC to reduce routing time and protect the intellectual property associated with the VC. This is commonly done by merging small obstructions together when creating a VC LEF description from GDSII-Stream.

2.3.4 Footprint

A description of the dimensions of the VC is required, including the location of the origin, any symmetry, and legal rotations. This information is used for placing the block within a system chip on the appropriate grid structure for place, route, and parasitic extraction.

Format

VC LEF is used for this information.

Currently Used Formats: LEF

Usage

Normally, footprint descriptions in VC LEF are rectangular. However, it is possible to describe rectilinear footprints by setting the SIZE of the macro as a whole to a rectangular bounding box, then defining obstructions on a special OVERLAP layer within the bounding box.

The obstructions on the OVERLAP layer indicate areas within the bounding box, which no other macro should overlap. Thus, the obstructions should completely cover the rectilinear shape of the VC.

Applicability

Soft VCs: not required

Firm VCs: CONDITIONALLY MANDATORY

- Required for those portions of Firm VCs that contain physical data.

Hard VCs: MANDATORY

2.3.5 Power and Ground

A description of the power and ground connections to the VC is required.

Format

VC LEF is used for this information. The USE of each power or ground pin should be set to POWER or GROUND.

Currently Used Formats: LEF/DEF, ASCII

Applicability

Soft VCs: not required

Firm VCs: CONDITIONALLY RECOMMENDED

- Required for those portions of Firm VCs that contain physical data.

Hard VCs: MANDATORY

Usage

Typically, the connections can be defined using any of the following methods:

- A set of interface pins/ports (Figure 4: Power Interfaces: Interface Pins)
- A contiguous ring structure modeled as a pin/port (Figure 5: Power Interfaces: Correct Ring Structure)
- A rectangular plane with the shape modeled as a pin/port (Figure 7: Power Interfaces: Power Plane Structure).

Enough power and ground connections should be provided to ensure voltage drop, noise, and simultaneous switching requirements are met under worst case conditions.

Power Interface Pins

Interface pins/ports should follow the same rules as normal signal pins/ports and ensure that at least one routing grid intersects the pin/port shape.

When there are multiple power pins, it is not necessary to explicitly specify the MUSTJOIN property on each pin; all power pins are to be connected together automatically.

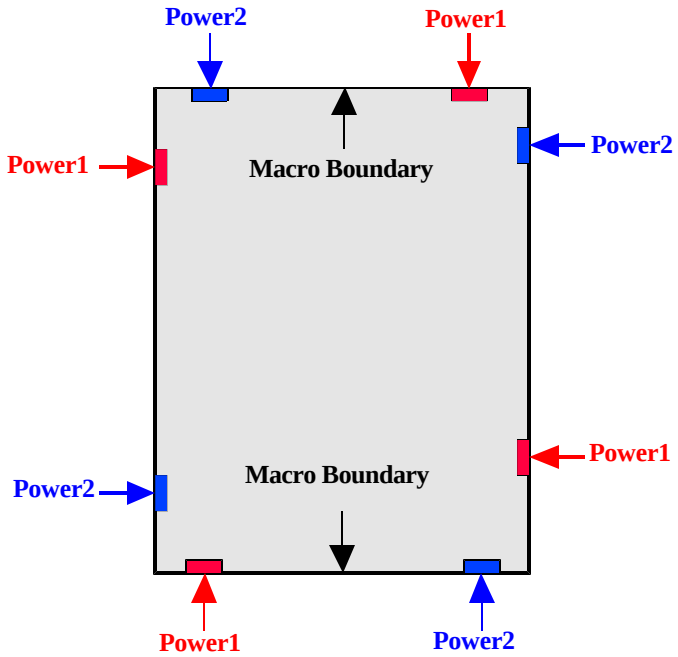


Figure 4: Power Interfaces: Interface Pins

Power Rings

Ring structures should be contained completely within the boundary of the block as shown in Figure 5: Power Interfaces: Correct Ring Structure. Ring structures should not be defined outside the block boundary as shown in Figure 6: Power Interfaces: Incorrect Ring Structure.

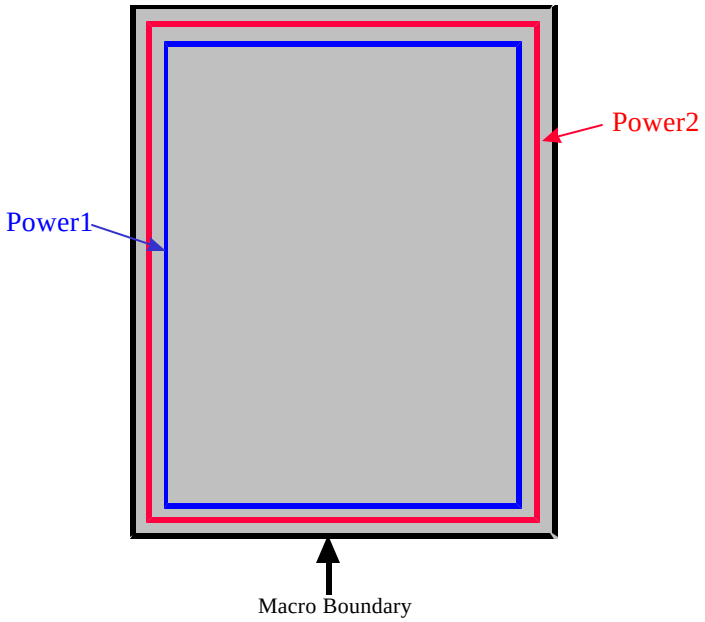


Figure 5: Power Interfaces: Correct Ring Structure

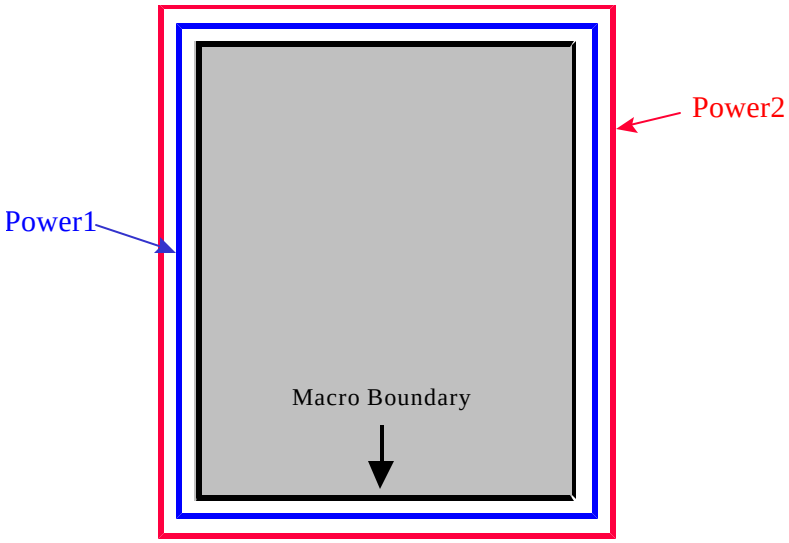


Figure 6: Power Interfaces: Incorrect Ring Structure

Power Planes

Power edges should be coincident with the routing grid for common layers; connections are allowed anywhere within the defined rectangular pin/port.

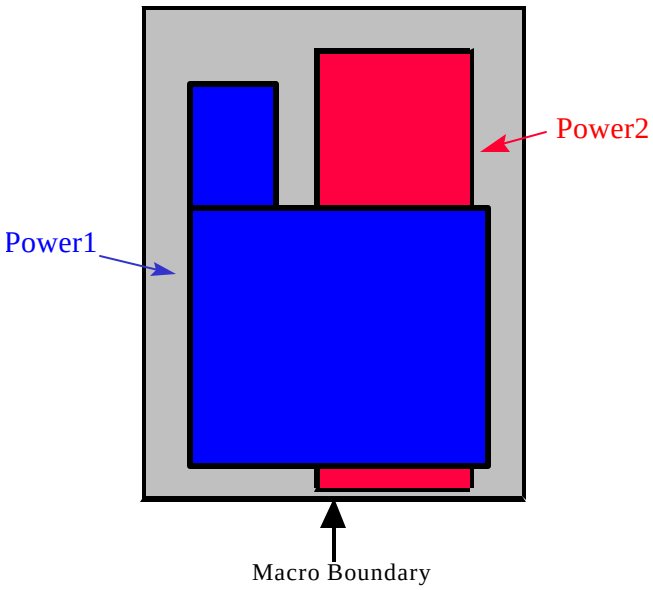


Figure 7: Power Interfaces: Power Plane Structure

2.3.6 Signature

A signature section has been defined by VSIA as a user extension (a VC LEF 5.1 feature) to the VC LEF MACRO section. The signature section is required, and it identifies at least the VC provider, the revision of the VC, and the revision of the underlying technology.

A digital signature based on public-key encryption may optionally be included in the signature section; this can be used in authenticating the VC LEF file.

The signature is not intended to replace a general IP protection capability, nor is it intended to provide a complete "technology fingerprint" of the VC. It is simply a mechanism for embedding some identifying information about the VC in the VC LEF file itself, to enable some types of consistency checking and authentication.

Applicability

Soft VCs: not required

Firm VCs: not required

Hard VCs: MANDATORY

2.4 Design Constraints

Design Constraints describe restrictions on the usage of a VC within a system chip design, or on the implementation of the VC itself.

Interface constraints describe requirements for the interfaces to the VC. Interface constraints apply to the system chip design surrounding the VC, and they ensure that the VC will work properly in the system chip context. Interface constraints are required for soft, firm, and hard VCs.

Internal constraints describe requirements for the internal implementation of the VC. They apply to the VC itself, and they are used for soft and firm VCs to ensure that the VC implementation (which is done by the VC integrator) meets the specifications for the VC. A VC will generally contain hierarchy, and internal constraints may be used to describe requirements for lower levels of hierarchy within the VC, as well as requirements for the VC as a whole.

Format

The I/V DWG endorses the OVI Design Constraints Working Group (DC-WG) development of a Design Constraints Standard.

2.4.1 Timing Constraints

Timing constraints are related to the basic delay model in the sense that the timing constraints describe what should be implemented, while the basic delay model describes what has already been implemented (or at least estimated).

Detailed interface timing constraints on non-clock signals are usually included in the basic delay model.

Interface Constraints

- Setup, hold, removal, recovery, pulse width, nochange

Internal Constraints

- Arrival times at inputs
- Required times at outputs
- Internal and external combinational delay (for purely combinational paths)
- Multi-cycle paths
- False paths
- Data signal transition times
- Transparent latch borrowing limits

Format

Currently Used Formats: Synopsys DC Shell, Ambit Constraints, GCF, SDF

Applicability

Soft VCs: MANDATORY

Firm VCs: MANDATORY

Hard VCs: not required

2.4.2 Clock Constraints

Clock constraints describe requirements for the clock signals provided to the VC, and (for soft and firm VCs) the required implementation of the clock distribution network within the VC.

Interface Constraints

- Clock frequency
- Duty cycle
- Jitter
- Relationship between multiple clocks provided to the VC

Internal Constraints

- Insertion delay
- Skew
- Transition times
- Pulse width
- Gating
- Type of distribution to be used and restrictions on its structure (such as tree instead of trunk, how many levels, what kind of buffers and routing style)
- Generated clock waveforms

Format

Currently Used Formats: Synopsys DC Shell, Ambit Constraints, GCF, SDF

Applicability

Soft VCs: MANDATORY

Firm VCs: MANDATORY

Hard VCs: CONDITIONALLY MANDATORY

- Required if the clock signal requirements need to be provided to the Hard VC.

2.4.3 Logic Architecture Constraints

Logic architecture constraints describe the designer's intent for aspects of the logical implementation of the design that cannot be expressed directly in the RTL description.

Logic architecture constraints only apply to soft VCs, and there are only internal constraints.

Format

Currently Used Formats: Synopsys and Ambit RTL Parameters.

Applicability

Soft VCs: MANDATORY

Firm VCs: not required

Hard VCs: not required

Internal Constraints

- Hierarchy manipulations (flattening or preserving sections of the logical hierarchy),
- State machine encoding
- Resource sharing restrictions
- Mapping/binding restrictions for certain operators or design objects

2.4.4 Area Constraints

Area constraints describe requirements for how much area may be consumed by a portion of the design. For synthesis, the constraints may describe just the area consumed by the primitives themselves, while in floorplanning and place and route the constraints may account for various types of physical overhead such as power and clock distribution.

Area constraints only apply to soft and firm VCs, and there are only internal constraints.

Format

Currently Used Formats: Synopsys DC Shell, Ambit Constraints

Applicability

Soft VCs: RECOMMENDED

Firm VCs: CONDITIONALLY MANDATORY

- Required for synthesis or floorplanning.

Hard VCs: not required

Internal Constraints

- Primitive area
- Power distribution area
- Clock distribution area
- Test logic area

2.4.5 Physical Implementation Constraints

Physical implementation constraints describe requirements for the topology and routing of the internals of a firm VC.

Internal Constraints

- Regions
- Placement groups
- Datapath topology
- Wide wire specifications
- Power distribution structures

I/O Port Placement Constraints

Format

Currently Used Formats: PDEF, DEF

Applicability

Soft VCs: CONDITIONALLY MANDATORY

- Required for floorplanning

Firm VCs: CONDITIONALLY MANDATORY

- Required for floorplanning.

Hard VCs: not required

2.4.6 Power Constraints

Power constraints describe requirements for the power consumption of a VC.

Interface Constraints

- Current requirements for power pins

Internal Constraints

- Average, peak dynamic power
- Static power
- Leakage power
- Specified for the VC as a whole, for hierarchical blocks within the VC, or for particular signals within the VC

Format

Currently Used Formats: Synopsys DC Shell

Applicability

Soft VCs: CONDITIONALLY MANDATORY

- Required for synthesis or floorplanning

Firm VCs: CONDITIONALLY MANDATORY

- Required for synthesis or floorplanning

Hard VCs: not required

2.4.7 Test Constraints

Test constraints describe requirements for the test logic within a soft or firm VC. These constraints are used in inserting the test logic and verifying that the test logic functions correctly.

Interface Constraints

- Type of test socket provided
- Test clock waveforms
- Other constraints to be defined by the Test DWG

Internal Constraints

- Type of test structure to be inserted

- Scan chain ordering
- Test coverage
- Other constraints to be defined by the Test DWG

Format

Currently Used Formats: <tool specific>

Applicability

Soft VCs: MANDATORY

Firm VCs: MANDATORY

Hard VCs: MANDATORY

2.4.8 Environmental/Operating Conditions

Environmental/operating conditions describe the range of operating conditions in which the VC is expected to function properly. A hard VC may have tighter restrictions than the underlying technology.

Interface Constraints

- Process, voltage, and temperature ranges
- Required driver strengths
- Required driver types
- Input slew rates
- External capacitance

Internal Constraints

- Driver strength
- Driver type
- Input slew rate
- External capacitance

Format

Currently Used Formats: Synopsys DC shell, Ambit Constraints, GCF

Applicability

Soft VCs: MANDATORY

Firm VCs: MANDATORY

Hard VCs: MANDATORY

3. Virtual Component Guidelines

3.1 Chip-to-VC Hierarchical Integration

Designs at the VC-level and chip-level often use similar methods but at different levels of design hierarchy. However, where differences in methodology occur, it becomes necessary to provide an approach to “bridge” the two levels of hierarchy together. This can be done by providing guidelines at both the logical and physical integration levels.

3.1.1 Logic Design Integration Guidelines

Logic design integration guidelines are those guidelines affecting the logic at the boundary of the VC interfacing to the system chip.

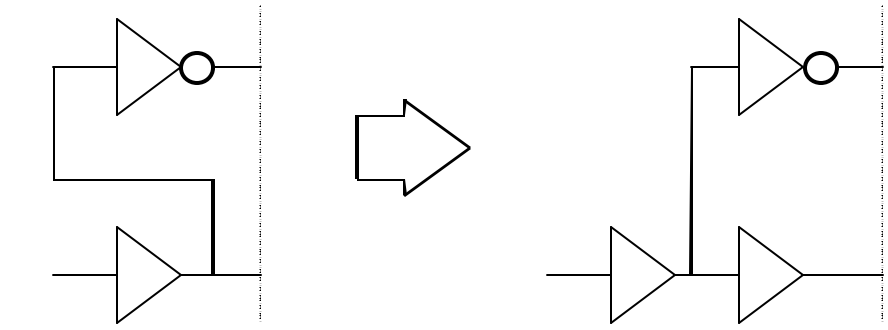
3.1.1.1 Logical I/O Ports

The I/O ports provide the chip-level interface to the VC. These ports should provide the interface between two distinct levels of hierarchy: the VC and the system chip. It is recommended that these ports be carefully designed so that:

- The system chip integration can be accomplished with as much flexibility as possible without affecting the functionality or performance of the VC; and,
- The VC operations do not affect the overall operation of the chip in any unexpected or unplanned ways.

The goal of good I/O port design should be to isolate the two levels of hierarchy from undesirable affects on each other. The following are recommendations:

- Re-entrant outputs should be avoided.



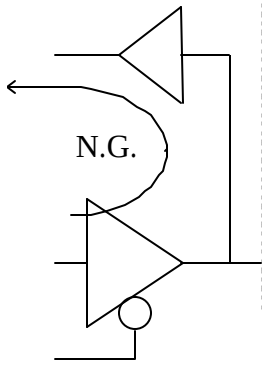


Figure 8: Re-entrant Outputs

- Through nets should be avoided.

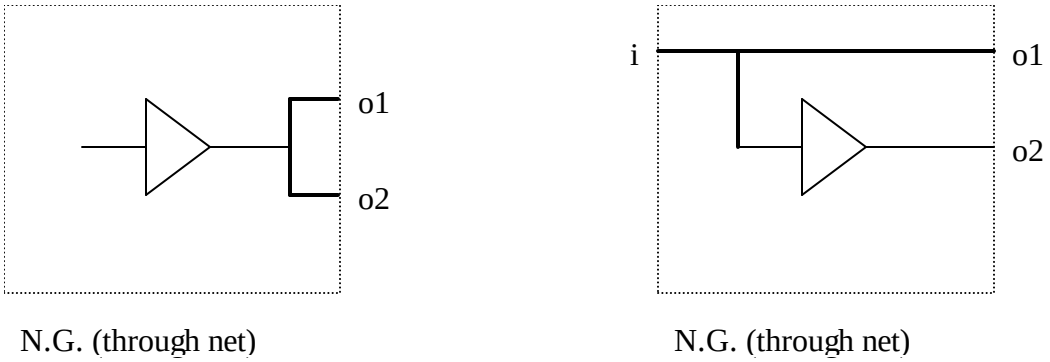


Figure 9: Through Net Rule

- When tri-state capable output or bi-directional ports are used, tri-state enable/bi-directional control should be available at the VC boundary.

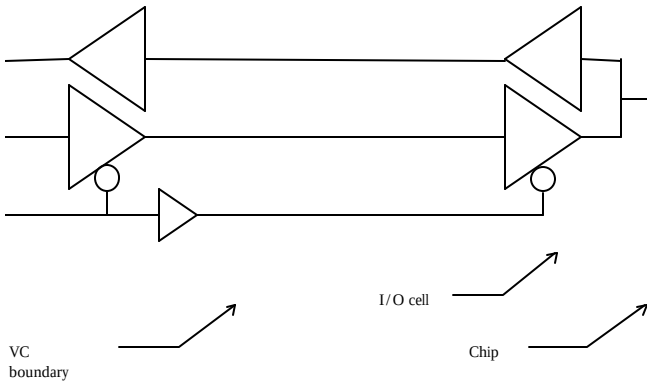


Figure 10: Tri-State Control Available at the VC Boundary

- The VC user should provide hold cells for the chip level busses.

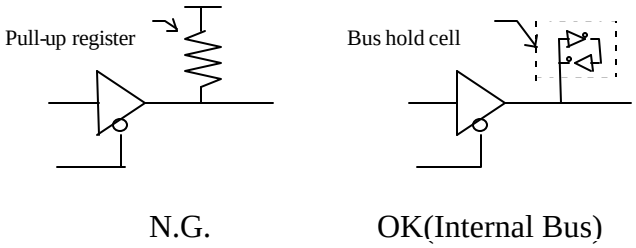


Figure 11: Bus Hold Cell

- Bi-directional I/O cells should be divided as separate input and output cells, if possible, as timing calculation becomes more difficult.

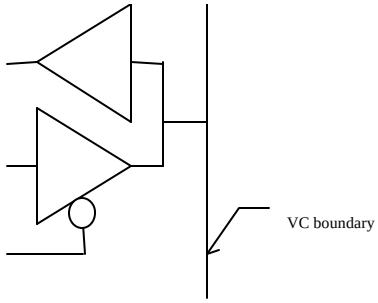


figure (1)

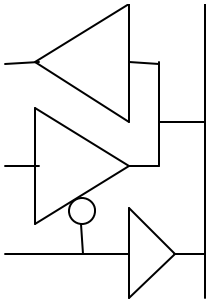


figure (2)

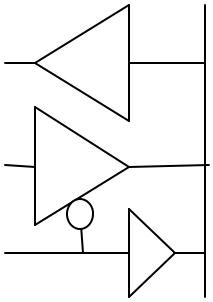


figure (3)

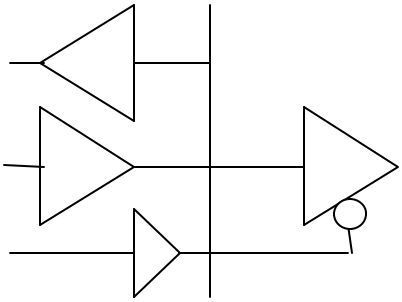


figure (4)

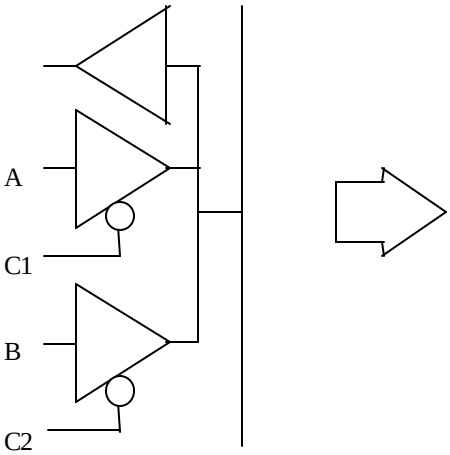


figure (5)

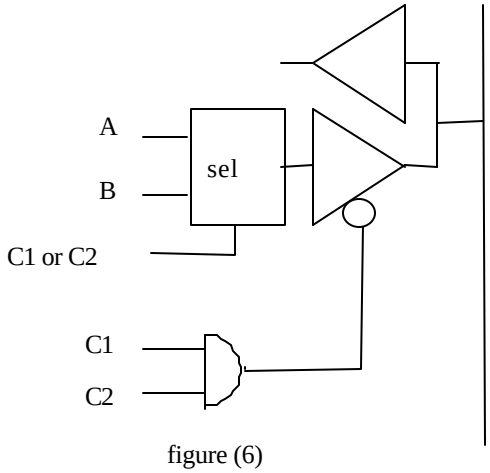


figure (6)

Figure 12: Dividing Bi-directional

- All reset and preset signals (synchronous and asynchronous) should be active low at the periphery of the macro.

3.1.1.2 Clocking Guidelines

The system chip should provide synchronized clocking at the system level. It is recommended that the system chip clock designer provide the clocking signal(s) to the VC for this purpose. The VC designer should provide clock access to the VC and pass along to the system designer the appropriate VC clock characteristics to enable overall chip clock synchronization.

Recommendations to accomplish the previous are:

- Synchronous designs are preferred.
- Single clock, single-phase clock architectures are preferred.
- Clock port access should be properly identified by the VC provider.
- Clock characterization of the VC should be provided to the system chip designer. This should describe the valid conditions for VC clock operation, such as valid frequencies, skew requirements, duty cycle, and pulse width.
- The VC designer should provide information about any special clocking requirements.
- When asynchronous signals are mixed with synchronous signals, the internal block design structure should separate synchronous and asynchronous domains. The interface relationships between the blocks should be documented.

Clocking Guidelines — Soft VC

- It is recommended that no clock distribution circuits be included as part of a soft VC.
- It is a general recommendation that all soft VCs be “characterized” against a reference implementation library. When this is done, a description of the clock distribution tree, the clock skew achieved, and the clock delays observed in the reference implementation should be included.

Clocking Guidelines — Hard and Firm VCs

- It is recommended that hard and firm VCs include the clock distribution network as part of the deliverable. Furthermore, if additional clock frequencies or phases are being developed in the VC from the clock I/O on the VC, they should be clearly defined and described in detail. An alternative is to provide the additional clock circuitry as a companion VC to provide the unique frequency or conditions.
- The internal clocking distribution should be described. At a minimum this should include internal clock delay and skew.
- The tradeoff between clock delay and clock skew is the responsibility of the hard VC designer. Caution should be exercised to assure that a proper balance is achieved, and also that on-chip variations are considered.
- Many different clocking schemes can be utilized to meet aggressive performance or power objectives in a VC. When methods such as gated clocks, multiple frequency, multiple pulse width, multi-phase or asynchronous circuits are employed, the VC provider should provide detailed documentation on the operation of the clock circuitry and the legitimate frequencies at which the design can be run.

3.1.1.3 Timing

Internal timing of the VC is the responsibility of the VC designer, and the chip-level timing of the system chip is the responsibility of the system chip designer. Recommendations for the timing of the interface between the system chip and the VC are:

- Static timing analysis is the preferred characterization method for VCs. When there are “don’t care” paths or conditions in the circuit, these should be described (in terms of clocking conditions and path) to permit successful static timing analysis.
- It is recommended that the VC designer either follow strict rules in the design and placement of the I/O port or a peripheral timing model be provided with the VC.
- Gray box models are not yet widely supported by EDA tools, so a black box model should be provided for every VC as a baseline description. The black box model should be sufficiently pessimistic to account for inaccuracies due to the inherent limitations of the black box modeling style.
- When a VC contains level-sensitive latches on the boundary of the VC, internal clock generators, internal clock gating or muxing, or uses multiple clocks, a gray box model should be provided to accurately model the timing behavior of the VC.

3.1.2 Physical Design Integration

Predictability and controllability are required for the success of system chip physical designs. Performance factors, such as size, timing, and power, should be properly considered for system chip success. These attributes should be properly accounted for at system chip integration with guidelines written to bridge the level of hierarchy between the system chip and the VC. The three different types of VC require different types of guidelines. Hard VCs should have been designed to be consistent with system chip integration and be accompanied by proper documentation and specifications. Soft and firm VCs should provide pre-analysis results and physical design control factors.

3.1.2.1 Physical I/O Ports

The VC port is the pad or point of interconnection between the VC and the system chip. The location and dimensions are defined in the physical abstract.

- It is recommended that the VC port and I/O buffer be located near the edge of the VC, with access layers defined to simplify timing between the VC and the chip, and for accuracy and repeatability.
- There should be direct access to the pin, at least on the layer defined, but preferably in the direction the routing normally occurs on that layer as well.
- To avoid additional routing channels, all ports should be positioned so they can be wired to the intended grid. This means that no more than three adjacent pins can be spaced less than the external grid they are intended to connect to.
- The supplier of a hard VC should specify ports that connect to the external chip I/O pad and the appropriate electrostatic discharge protection devices, taps, and other reliability and manufacturing structures used.
- Any inputs that have diode or substrate connections on first metal should be listed separately. This will inform the chip integrator that the integrator need not be concerned about antenna problems with these inputs.
- GDSII labels with pin names are required on all I/O ports to adjust them to the grid.

3.1.2.2 Porosity and Blockage

A blockage file provided with the VC designates those regions where system chip level routing is allowed. This file should include all areas where there are wires and the possibility of error due to crosstalk.

- Timing models should include the assumed capacitive effects of feed through non-blockage points. In general, non-blocked areas should be in the same form and direction as the preferred routing direction for that layer.
- Blockages should be included for all possible layers.
- Sufficient porosity should be created for the design. One logical minimum is the minimum number of feed-throughs likely to be required if the VC is placed on an edge or corner of the die. More feed-throughs will probably be needed for VC placement anywhere else in the system chip, but the amount is a function of the size of the resulting system chip and the location of the core within the system chip.
- It is recommended that this file ensure the blockage of chip level interconnect routing from critical areas, such as noise or timing, since they could affect the implementation of the VC.

3.1.2.3 Physical Structure (Hard VC)

- Hard VCs are recommended to be rectangular.
- All process layers should be specified in the GDSII, even blank layers since blockages are as important as wires in the usage of the core.
- Hard VCs should be as self-contained as possible. This means all substrate taps required for the power usage of the VC should be included within the VC, or any field isolation for different power supplies should be wholly contained within the VC boundary.

3.1.2.4 Power Access

VCs will have various current requirements and will be designed with specific voltage drop expectations. In addition, they may have multiple voltage requirements or special power requirements.

- It is required that the VC provider carefully describes power requirements of VCs.
- On flip chips, where solder ball grids are to be used for the chip, the VC should provide the appropriate power pads on grid for access to power.
- The VC should provide power access contacts sufficient to meet both current and voltage drop requirements of the VC. It is recommended that special requirements be appropriately described for the system chip integrator.
- Special power access should be properly described and defined, as appropriate, in LEF format.
- The maximum power requirement should be stated (at least be defined for the specific process the GDSII was originally targeted for).
- The resulting operating voltage de-rating from nominal resulting from IR drop should be noted to the user of the VC for proper timing design.
- The power and ground connections should be on the edge of the VC with layers specified.
- The connection locations should be specified along with the width of available connection on that edge. There should be a suggested minimum total width of power and ground connections required for each VC, and the VC should have sufficient margin for integration.
- For VCs that have multiple power supplies, each supply's access points should be clearly labeled and differentiated from one another.

3.1.2.5 Shapes

Today's chip technologies have finely tuned photolithography processes. This is especially true for the more advanced (higher performance, high density) technologies. This results in rules for critical spacing and geometry shapes. Most shape rules require rectilinear geometries with possibly some conditional exceptions, and most critical spacing/sizing rules are consistent across a chip with possibly some conditional exceptions.

It is recommended that VC providers use only rectilinear geometries and generally consistent critical spacing rules. Only when a VC design is intended to be used for one process should technology specific rules be considered.

3.1.2.6 Grid

Today's design systems and routers can use the layout grid in various ways. It is expected that physical VCs will come in a variety of grid styles.

It is recommended that the system chip level integration design systems be able to accommodate grided, gridless, or a mixture. The physical VC supplier should provide a boundary description in physical dimensions (such as microns rather than grids).

3.2 Naming Guidelines

This describes the conventions to follow for VC component naming, pin naming, and modeling language interoperability.

The appendix attached to this document is in support of naming definitions. The appendix identifies keywords that VC providers should avoid in naming. It also proposes a set of conventions that EDA vendors should follow to facilitate inter-operability. Names and naming consistencies/ transformations between Verilog, VHDL, C, C++, and the file system are defined in detail.

3.2.1 VC Naming

- The top-level VC name should be meaningful and reflect the function of the VC (such as MPEG2 core, vc_mpeg2).
- Any known standard, implied or otherwise, in the component name should be documented in the claims and verification section of the VSIA document.
- Different cases should not be mixed (all lowercase or all uppercase, no mixed case naming).
- Lowercase is recommended.
- The previous conventions should also be applied hierarchically to all lower level module names that are seen by the VC user.

3.2.2 VC Pin Naming

- Pin names should comply with any existing naming conventions for the formats being used (such as VHDL, Verilog naming requirements, reserved key words).
- Different cases should not be mixed (all lowercase or all uppercase, no mix case naming).
- Lowercase is recommended.
- A consistent and coherent naming convention should be used for all common or global signals, such as clock, reset, etc.
- Active high and active low signals should be labeled to differentiate between the two types (such as sig1x, sig2x, where x denotes active low).
- All pin names should be documented in the user guide for the VC. This should detail the various signal attributes that may be required by the VC user (such as clock, reset, in/out/ bi-directional).
- All busses should start at zero.
- It is recommended that all busses be designated with the most significant bit on the left (i.e., D[7:0], so that D[7] refers to 27).

3.2.3 VC File Naming

- Known keywords should be avoided — see the appendix.
- A consistent naming convention should be used for all files within the VC and among all the VCs. Recommended example naming formats are:
  - C files vcname.c
  - ++ files vcname.cpp
  - Verilog vcname.v
  - VHDL vcname.vhd
  - SDF data vcname.sdf
  - test, etc.
- Directories should be used to allow the separation of different data types, such as gate-level and RTL VHDL.
  - rtl/vcname.vhd
  - rtl/vcsubmodule.vhd
  - gate/vcname.vhd
  - gate/vcsubmodule.vhd
- Whichever naming convention used should be reflected in the user guide. This will enable the revision control to be closely linked with the filenames.
- Where multi-platform support is required, the VC provider is responsible for ensuring a consistent naming format, not the VC user. The VC provider should document which platforms are supported for a particular VC.
- Filenames should not differ by case alone. This will reduce any problems with multi-platform support.

Appendix

VC Reserved Words

For Virtual Component design, it is recommended that the author refrain from use of the following sets of keywords.

Verilog

All reserved keywords as described in IEEE 1364-1995 Verilog Language Reference Manual.

VHDL

All reserved keywords as described in IEEE1076-1987 VHDL Language Reference Manual.

Windows NT

- CON
- AUX
- COM1
- COM2
- COM3
- COM4
- LPT1
- LPT2
- LPT3
- PRN
- NUL
- .
- ..
- \
- Non-printable and control characters (0x1F-0x7F, del)

UNIX

- .
- ..
- /
- Non-printable and control characters (0x1F-0x7F, del)

Name Space Descriptions and Transformation Rules

The following paragraphs propose a set of definitions for name spaces for EDA vendors to support, as well as transformation rules for mapping between name spaces. Implementing these name space transformation capabilities will support ease of integration of Virtual Component information produced by tools from different vendors.

Name Space Descriptions

- A raw string of characters (represented with uchar or ushort [UNICODE])
- String is terminated with the NULL character (value 0)
- Case sensitive
- No character restrictions
- Allowed characters are currently ASCII values 1 to 255
- In the future, to support UNICODE, character values will be 1 to 2^16-1

File System

- UNIX is case sensitive.
- *** NT is case INSENSITIVE but case preserving .***
- Forward slash and back slash are not allowed (back slash because of NT).
- White space is not allowed.

- Characters other than alphanumeric and underscore are discouraged (we are allowing dash and dollar sign to be used in names).
- NT does not allow filenames that match DOS devices, and thus the following names are reserved in NT: CON, AUX, COM1, COM2, COM3, COM4, LPT1, LPT2, LPT3, PRN, NUL.
- Let N_Filesystem be the set of characters that will be escaped in file system names. N_file system includes all characters that are not a letter, digit, underscore, dash, and dollar sign. All characters greater than 127 are in N_file system.
- N_file system includes the pound sign and percent character.
- **Note:** We want UNIX and NT to share the same name space since we want to be able to tar a library on UNIX, untar it on NT (or vice-versa), and have the library work in the new environment. This requirement leads us to treat the combined file system name space as case INSENSITIVE, but it also requires that we NOT use case preservation due to UNIX's case sensitivity.

VHDL

- Normal and escaped identifiers
- Normal identifiers are case insensitive.
- Escaped identifiers are case sensitive.
- Escaped identifiers can contain any graphic character and spaces.
- Escaped identifiers begin and end with \ (backslash).
- To embed \ (backslash) in an escaped identifier, use \\.
- Equivalent normal and escaped identifiers denote different objects.
- VHDL keywords must be escaped.
- Normal identifiers start with alpha and contain alphanums or underscores.
- All objects share the same name space (signals, instances, etc.).
- Let N_VHDL be the set of characters that will be escaped VHDL characters. N_VHDL includes all characters that are not in the graphic character set defined in chapter 13 of the VHDL 93 LRM. N_VHDL includes all characters greater than 255.

Verilog/Sensitive

- Normal and escaped identifiers
- Normal identifiers are case sensitive.
- Escaped identifiers are case sensitive.
- Escaped identifiers can contain any graphic character, but no spaces.
- Escaped identifiers begin with \ and are terminated by white space.
- Verilog keywords must be escaped.
- Normal identifiers contain letters, digits, dollar signs, and underscores.
- Normal identifiers cannot start with a digit or dollar sign.
- Equivalent normal and escaped identifiers denote the same objects.
- All objects share the same name space (signals, instances, etc.).
- Let N_Verilog be the set of characters that will be escaped Verilog characters. N_Verilog includes non-graphic characters, white space characters, and characters greater than 127.

Verilog/Insensitive (Verilog -u option)

Same as Verilog/Sensitive except that normal and escaped identifiers are case insensitive.

Name Space Transformation Rules

The following paragraphs outline the name space transformation rules for mapping between the name spaces defined above.
Note: In the following rules:
- #XX is used to map an "objectionable" character in the ASCII range 0 to 255.
- ##XXXX is used to map an "objectionable" character in the UNICODE range 256 to (2^16-1).
- X will be one of the characters 0-9, a-f; a-f will always be lowercase. The #XX and ##XXXX character sequences are referred to as "pound-hex groups" in the following rules.

Raw -> Verilog/Sensitive

If the raw identifier is a legal Verilog normal identifier, it is represented as such. Else, if the raw identifier contains any characters in N_Verilog, these characters are replaced with pound-hex groups. The resulting

string is represented as a Verilog escaped identifier (that is, it is prefixed with a backslash).

Verilog/Sensitive ->Raw

If the Verilog identifier is not escaped, the identifier is converted as-is into the raw identifier. If the identifier is escaped, the leading \ is removed, any pound-hex groups which in fact represent characters in N_Verilog are replaced by their single character equivalents, any pound-hex groups that do not represent characters in N_Verilog are left as-is, and the resulting string is the raw identifier.

Raw ->Verilog/Insensitive

Let ESC_VLOGUP be the character ^ (caret).
Uppercase characters are preceded with the ESC_VLOGUP prefix. If the resulting string is a legal Verilog normal identifier, it is represented as such. Else, if the resulting string contains any characters in N_Verilog, these characters are replaced with pound-hex groups. The resulting string is represented as a Verilog escaped identifier (that is, it is prefixed with \).

Verilog/Insensitive ->Raw

If the Verilog identifier is not escaped, the identifier is copied to a temp string. Else, the leading \ is removed, any pound-hex groups which in fact represent characters in N_Verilog are replaced by their single character equivalents, any pound-hex groups that do not represent characters in N_Verilog are left as-is, and the resulting string is copied to a temp string.
Next, all letters A-Z in the temp string are lowercased unless they are preceded with the ESC_VLOGUP prefix, in which case the ESC_VLOGUP is removed and the letter is uppercased. The identifier then is represented as-is as a raw identifier.

Raw ->VHDL

Let ESC_VHDL be the six character sequence ESC, V, H, D, L, ESC, where ESC represents the escape character (ASCII hex value 1B).
If a raw identifier is prefixed with ESC_VHDL,
{
Then the ESC_VHDL sequence is removed, backslashes are doubled, and the remaining characters are represented as-is as a VHDL escaped identifier,
}
else If (the raw identifier contains no uppercase letters, begins with alpha, contains only alphanums and underscores, and does not contain a trailing or duplicate underscore and is not a VHDL keyword)
{
Then it is represented as a normal VHDL identifier
}
else
{
Any characters in N_VHDL are replaced with pound-hex groups, backslashes are doubled, and the resulting identifier is represented as an escaped VHDL identifier in its original case. (That is, it begins and ends with backslash.).
}

VHDL ->Raw

If a VHDL identifier is represented escaped {
Unescape the identifier, turn double backslashes into single backslashes. Replace any pound-hex groups which in fact represent characters in N_VHDL with their single character equivalents. Leave any pound-hex groups which represent characters not in N_VHDL as-is.
If the resulting string need not be escaped according to the rules of VHDL
{
Prefix the identifier with ESC_VHDL
}
return the resulting string
}
else
{
lowercase the identifier and return it

}

Raw ->File System

Any characters in N_file system are replaced with a pound-hex group. Every uppercase character is preceded with % and is left in uppercase. If the resulting name is a NT/DOS reserved device name (such as CON, AUX, COM1, COM2, COM3, COM4, LPT1, LPT2, LPT3, PRN, NUL) then % is appended to the name.

Example Raw	**file system mappings:**
•foo	foo
•Foo	%Foo
•Hi There%	Hi#20%There
•con	con%
•nec$f04	nec$f04

Note: These rules imply that “Foo” is not a legal file system name.

File System ->Raw

Replace every pound hex group that in fact represents a character in N_file system with the corresponding character. Leave any pound hex group that does not represent a character in N_file system as-is. Replace every occurrence of % followed by a letter with the same uppercase letter. Any trailing % is removed. All other characters are translated in lowercase. The resulting string is the raw identifier.

Case Preservation

When mapping between two case insensitive name spaces, it is often desirable but never strictly the case of identifiers. Note that case preservation is not an issue when either name space is case sensitive.
To support case preservation, the name mapping routines invisibly associate an additional piece of data with a RAW identifier -- this additional piece of data is termed the “case preservation string.” The case preservation string is a normal null terminated string that has the same length as the RAW identifier. The characters of the case preservation string are one of U, L, and ?, indicating that the original case of the corresponding character in the RAW identifier is either upper, lower, or of unknown case.
The case preservation capability provides only aesthetic benefits. The fact that case preservation occurs has no semantic effect.
An example of case preservation follows. Assume an application needs to map the identifier FooBar from the VHDL name space to the Verilog/Insensitive name space. The following transformations would occur:

- VHDL: FooBar
- Raw: foobar -- identifier
- ULLULL -- case preservation string
- Verilog/Insensitive: FooBar

Notes:

- An application which calls the name mapping routines will have no knowledge of the use of the case preservation string since its use is completely internal to the name mapping routines.
- The case preservation capability is intentionally turned off when mapping to or from the file system namespace. Because we want to treat the file system name space as a single unified name space, and because UNIX is in fact case sensitive while NT is case insensitive, it is required that case preservation not occur in this mapping.

Examples

- Convert an Verilog/Sensitive identifier “buffer” into the VHDL name space
  - Verilog/Sensitive -> Raw
  - buffer -> buffer
  - Raw -> VHDL
  - buffer -> \buffer\ -- since “buffer” is a VHDL keyword
- Convert VHDL identifier FOO into Verilog/Sensitive name space
  - VHDL -> Raw
  - FOO -> foo

- Raw -> Verilog/Sensitive
- foo -> foo

- Find the file system cell directory for the Verilog module named “Foo”
  - Verilog -> Raw
  - Foo -> Foo
  - Raw -> file system
  - Foo -> %Foo

Cross-name space references work sensibly. For example, if a VHDL entity has ports named “FOO” and “WIRE,” and if this entity is instantiated in a Verilog module, then the ports can be referenced in Verilog by the names “foo” and “\wire.” No additional information is required. As another example, if a Verilog/Sensitive name space path to an instance is a.b.c.A where “a” is the root instance and “a” and “A” are different modules due to Verilog’s case sensitivity, then the corresponding path expressed in the VHDL name space would be \A\.
Again, no additional information is required to indicate that a particular identifier exists in one name space or another. Note that this path may represent a mixed design hierarchy containing both Verilog and VHDL instances.

Name Space Collisions

When integrating two or more independent VCs there is the possibility of a name space collision, especially if they are described hierarchically.
For example, two independent GDSII layout descriptions could both define a sub-block called “adder” which would cause a name space collision when instantiated together in the same design.
This is a generic problem for many hierarchical VC description languages, such as HSPICE, GDSII, Verilog, etc. although it is not a problem for VHDL since it supports name space scoping.
To minimize the risk of unexpected name space collisions, VC providers should take reasonable steps to ensure their names will remain unique (for example, by using a VC specific prefix).
In addition to this VC specific prefix, it is also recommended that VC providers endeavor to make their whole name space unique. (for example, by prefixing with a company specific string such as JTAG ID, NASDAQ code, etc.).
However, adhering to these guidelines alone will not guarantee universal uniqueness of the VC name space, so collisions will have to be processed by exception.
In the event of a collision there are two possible scenarios:

- The VC integrator has modification rights.

  In this case the collision could be resolved either manually or automatically by simply altering one or more of the names in question.

- The VC provider does not have modification rights.

  In this case it will be necessary to negotiate resolution of the collision with the VC providers in question, especially if the VC has been “finger printed.”

Name space problems can also be addressed at the tool level in several ways:

- Provide mechanisms that allow separate libraries for each VC, so sub-blocks with conflicting names become unique by referring to them using a combination of the library name and the cell name.
- Provide mechanisms for detecting conflicts and doing name mapping when necessary. The name mapping can either be automatic, following an arbitrary convention to ensure uniqueness, or manual, where the integrator provides a file describing the mapping from each conflicting name to a unique name.
  A common convention for automatically ensuring uniqueness is to truncate names, then append a numeric suffix; the truncation ensures that the name does not exceed any length limitations.

The first approach is suitable for tools and techniques that are hierarchical. The second approach is required when flattening a design or when combining hierarchical VCs into a single library.

Glossary

Basic Delay Model	Defines timing specification of the VC.
Behavioral Model	A description of the function and timing of a component without describing a specific implementation. A behavioral model can exist at any level of abstraction. Abstraction depends on the precision of implementation details. For example, a behavior model can be a model that describes the bulk time and functionality of a processor that executes an abstract algorithm, or it can be a model of the processor at the less abstract instructions set level. The precision of internal and external data values depends on the model's abstraction level.
Black Box	An implementation of a hard, firm, or soft VC that is hidden from the designer. It is only observable as a bus functional model at the I/O ports**.**
Cell Level Netlist	A structural interconnection of design objects ranging from simple logic gates to complex functions.
Circuit Level Netlist	A structural interconnection of semiconductor devices such as transistors
Firm VC	VCs that have been structurally and topologically optimized for performance and area through floorplanning and/or placement using a generic technology library. The level of detail ranges from region placement of RTL sub-blocks, to relatively placed datapaths, to parametrized generators, to a fully placed netlist. A combination of these approaches is often used to meet the design goals. Firm VCs offer a compromise between soft VCs and hard VCs. They are more flexible and portable than hard VCs, yet more predictive of performance and area than soft VCs. Firm VCs include a combination of synthesizable RTL, reference technology library, detailed floorplan, and a full or partial netlist. When a full netlist is present, it is expected that the test logic has been inserted and that the test lists will accompany the design. Firm VCs do not include routing. Protection risk is equivalent to soft.
Gray Box	Includes additional details of the internal structure so tools using the model can more accurately analyze the effect of the environment around the VC.
Hard VC	VCs that have been optimized for power, size, or performance, and mapped to a specific technology. Examples include netlists fully placed, routed, and optimized for a specific technology library, a custom physical layout, or a combination of the two. Hard VCs are process/vendor specific and generally expressed in GDSII format. They have the advantage of being much more predictable, but consequently are less flexible and portable due to process dependencies. Hard VCs require, at a minimum, a high-level behavior model, a test list, full physical and timing models along with GDSII data. The ability to legally protect hard VCs is much better because there is no requirement for RTL.
Interoperability	Model interoperability designates the degree to which one model may be connected to other models, and have them function properly, with a modicum of effort. Model interoperability requires agreement in interface structure, data format, timing, protocol, and the information content/semantics of exchanged signals.
Performance Model	Performance is a collection of the measures of quality of a design that relates to the timely response of the system in reacting to stimuli. Measures associated with performance include response time, throughput, and utilization. A performance model may be written at any level of abstraction. In general, a performance model may describe the time required to perform simple tasks, such as memory access of a single CPU. However, in the content of VSIA, the typical abstraction level for performance models is most often at the multiprocessor network level.
Peripheral Interconnect Model (PIN)	Specifies the interconnection RCs for the peripheral interconnect between the physical I/O ports and the internal gates of the VC.
Physical Blocks	A model of the physical implementation of the VC and the system chip.

Power Model	Defines the power specification of the VC.
Register Transfer Level Model (RTL)	An RTL model describes a system in terms of registers, combinational circuitry, low-level buses, and control circuits, usually implemented as finite state machines. Some internal structural implementation formation is implied by the register transformations, but this information is not explicitly described. The primary purpose of RTL models is for developing and testing the internal architecture and control logic within an IC component so the design satisfies the required functionality and timing constraints of the IC. The RTL model is also used for specifying the design in a process neutral format that is retargetable to specific technologies or process lines through automatic synthesis. It is often used for the generating detailed test vectors, gathering timing measurements to increase the accuracy of more abstract models, investigating interactions with closely connected components, and it unambiguously documents the design solution.
RTL Source	Defines the VC source description and is the primary input for the implementation and verification of the VC within a system chip design.
Soft VC	VCs that are delivered in the form of synthesizable HDL code. The advantage is the flexibility of the source code so it can be retargeted to multiple manufacturing processes. The disadvantage is the difficulty in performance prediction (such as timing, area, power). Soft VCs typically have higher intellectual property protection risks because RTL source code is required by the integrator.
Structural Model	A representation of a component or system in terms of the interconnections of its constituent components. A structural model follows the physical hierarchy of the system. The hierarchy reflects the physical organization of a specific implementation. A structural model describes the physical structure of a specific implementation by specifying the components and their topological interconnections. These components can be described structurally, functionally, or behaviorally. Simulation of a structural model requires all models in the lowest (leaf) branches of the hierarchy to be behavioral or functional models. Therefore, the effective temporal, data value, and functional precisions depend on the leaf models. A structural model can exist at any level of abstraction. Structural precision depends on the granularity of the structural blocks.
System Chip	A term used to describe highly integrated devices. It is also known as system on silicon, system-on-a-chip, system-LSI, system-ASIC, and a System-Level Integration (SLI) device.
Timing Analysis Model	Defines the static timing characteristics of the VC**.**
VC Port	The pad or point of interconnection between the VC and the system chip.
Virtual Component (VC)	A block that meets the VSIA specification and is used as a component in the virtual socket design environment. Virtual Components can be of the forms: soft, firm, or hard. A pre-implemented, reusable module of intellectual property that can be quickly inserted and verified to create a single-chip system. VCs are also called megacells or cores.