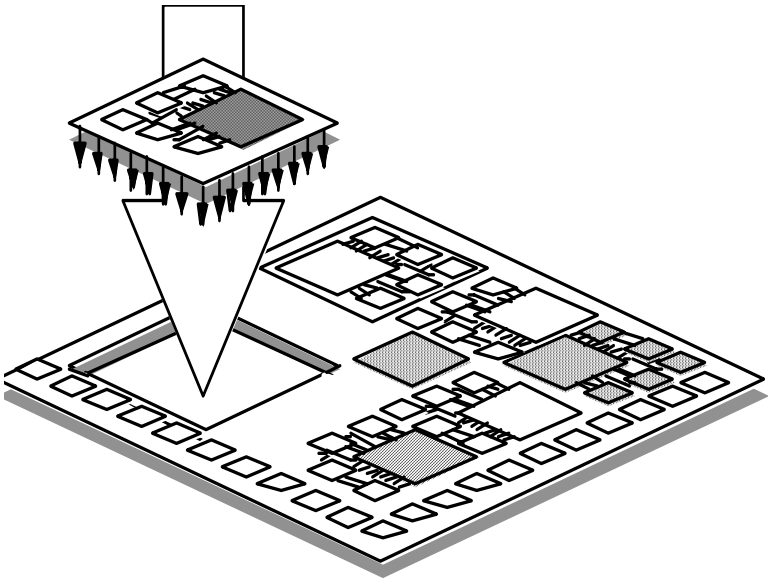
VSI Alliance™

Virtual Component Identification
Soft IP Tagging Standard
Version 2.0

(IPP 4 2.0)

Intellectual Property Protection
Development Working Group

Released September, 2006



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.

Virtual Component Identification
Soft IP Tagging Standard
Development Working Group
(IPP 4 2.0)

Members of the Development Working Group:

Agere Systems
ARC International PLC
ARM
Cadence Design Systems
Center of Harbin Microelectronics
ECSI
Freescale
GDA Technologies
Goya Technology
IBM
Intel
Intel
LSI Logic
Mentor Graphics
Oki Electric Industry
Philips Semiconductors
Sipac
Sonics
Synchronicity
Toshiba
VCX

Technical Editor/Author:

Gary Delp..... LSI Logic
Kathy Werner..... Freescale

Revision History

Version 0.1	May04	Kenneth Goodnow	Completed original draft for IPP DWG review.
Version 1.0	14Jun04	Ian Mackintosh	Edited and formatted draft for member review.
Version 1.0	27Aug04	Ian Mackintosh	Incorporated review comments, prepared final release.
Version 2.0	14Sep06	Kathy Werner	Updated to align with hard spec, IPP 1 3.0

Contents

Figures

Tables

1. Overview

1.1 Scope and Field of Use

This standard provides a way to track non-hardened or soft IP information throughout the design and development process. The design process can include editing, synthesis, timing, placement, wiring, and other steps leading to GDS II generation. Semiconductor foundries, providers of virtual components (IP), and manufacturers of design tools can use the methods described in this standard to track identification information throughout each level of the development process. At each level, tracking information is obtained from the previous level, and is transported to the next level using the appropriate output format.

Note that this standard specifically addresses the passing of information throughout the design process. It does not consider the protection of the intellectual property (IP) within the design. The passing methods described in this standard are not secure, so they are potentially susceptible to tampering. These methods are intended only to facilitate the passing, use, and sharing of information among honest IP users and IP providers, helping to ensure the quality of the final design. Nevertheless, the mere existence of these methods does afford a low-level form of security.

1.2 Compatibility with VSIA Physical Tagging Standard

This standard deals with the tagging of “Soft IP,” that is, a virtual component or intellectual property that has been delivered to a customer in RTL, gate level netlist, or script form. A companion document, the “Virtual Component Identification Physical Tagging Standard,” deals with the tagging of virtual components (or “Hard IP”) that are delivered in GDS II form. These two tagging standards are compatible and complementary, resulting in information being stored in the GDS II file as a series of similarly formatted text lines.

1.3 Benefits

Expanded use of Soft IP allows more flexible use of design libraries, foundry sources, and methodologies. However, this flexibility also creates the problem of a IP becoming absorbed into the overall chip design at some stage of the design process. For example, a Verilog file (as a IP) might be presented as source to a customer. This Verilog file could then be synthesized, timed, placed, and routed along with the customer’s own logic. When viewing the final netlist or GDS II files, there would be no means of discerning the source, version, date, or manufacturer of the IP.

This standard provides a means of retaining information about the IP all the way through the design process, from source to GDS II. It takes identification information from the source Verilog file and passes it along through each of the design steps. Either directly or through a tool, customers can then view information about the IPs included in the chip and the steps the IP went through in the development process. By this means, customers can determine if the correct version of a IP is contained within the chip.

1.4 Methodology

Providers of virtual components, CAD software manufacturers, or foundries can use this document to enhance their products by utilizing this information tracking standard.

Virtual component manufacturers can use this standard to include version, date, and source information that is useful for quality and business reasons.

CAD software manufacturers can use this standard to enhance their tools to transport the IP information from one stage of their toolset to the next, from an industry standard format into their toolset, in the output of an industry-standard format at the end of their tool chaining. CAD software manufacturers can also provide new software to collect, correlate, and display this information at various places in the tool chain.

Semiconductor foundries can use this standard to provide additional services to support a quality check prior to IC manufacturing.

1.5 Referenced Intellectual Property

N/A

1.6 Definition of Terms

CAD	Computer Aided Design
EDA	Electronic Design Automation
Foundry	Semiconductor Manufacturing facility that creates integrated circuits for other companies
GDS II - Stream	Format for physical implementation of design data the property of Cadence Design Systems, Inc.
IP	Intellectual Property, specifically electronic design or silicon
Physical Tagging Standard	Previously released VSIA tagging standard for IP that is shipped as GDS II.
Soft IP	Intellectual property for electronic design that is delivered without semiconductor device layout information.
Tag	Text data embedded in a file that provides information required for business reporting.
VSIA	Virtual Socket Interface Alliance, Incorporated

2. Soft IP Tag Specification

2.1 Background

The VSIA has previously issued a standard for IP tagging within the GDS II output format, “VSI Alliance Virtual Component Identification Physical Tagging Standard (IPP 1 3.0).” This standard is available from the VSIA Web site at www.vsi.org.

The VSIA Physical Tagging Standard states that information must be encoded into the physical description of a IP so the semiconductor foundry can produce a report of the parameters indicating the ownership of the IP in the design. The GDS II-Stream format is used for this purpose. This format’s encoding mechanism allows for an arbitrary number of fields. The encoding must have at least the fields indicated in Table 1. These fields provide anyone reading the tags with the basis to produce a report containing the minimum required information prescribed by this standard, that is, a report of Vendor, Product, (product) Version, and Metric.

The VSIA Physical Tagging Standard supports a text string that serves as a tag and resides within a specified level of the GDS II file. The format of the text string is a reserved ampersand (&) character followed by a space. The expected sequence following the space is keyword, space, value. An example of this format is:

& Vendor company_name
& Product product_name
& Version string1
& Metric string2

The keywords supported by the current version of the VSIA Physical Tagging Standard are indicated in the following table. The ampersand (&) character identifies the start of a keyword in the Hard IP tag.

Table 1: Currently Supported GDS II Hard IP Tagging Keywords

Keyword	Argument(s)	Example
Vendor	*String*	& Vendor CompanyZ
Product	*String*	& Product New_IP
Version	*String*	& Version 1.3a
Metric	*String*	& Metric 47.3
IP_Owner	*String*	& IP_Owner WMSG
Techno	*String*	& Techno CMOS090GP
Area	*Floating*	& Area 10.514
Celltype	*LIB or IP obligatory* *LEAF optional* *Empty also allowed*	& Celltype LEAF
Cell_Id	*String*	& Cell_Id nd2x1
Signature	*String*	& Signature <tbd>
Tag_Spec	*String*	& Tag_Spec 3.0
Date_Time	*String*	& Date_Time 20061014
	Delimiter	<space>

The GDS II-Stream specification limits the string length to a maximum of 512 characters. This limitation includes the keyword, spaces, and the special ampersand (&) character. The VSIA Physical Tagging Standard specifies only the information about the physical design step to be contained within the GDS II file. This Soft IP Tagging Standard is intended to be seamlessly integrated with the existing Physical Tagging Standard by adding information earlier in the design process.

2.2 Overview

This new Soft IP Tagging Standard takes the VSIA Physical Tagging Standard keywords, delimiters, and structure as an example. It then uses them at an earlier point during the design and development process to include additional IP information. Each level of the development process can add a tag. With an appropriate keyword, it can also identify the level at which the tag was created. In this way, the history of the design development of a piece of IP can be observed by examining the tag lines and keywords. The Soft IP tag could be contained in either design data or design construction scripts. The number of added tag structures is unlimited. At a given design level, several different processes can add tag information.

The Soft IP Tags described in this specification all have % as the delimiter all the way through, including the tool process step that generates GDSII. Hard Tags as described in the ref spec, all use & as the delimiter. The two different characters for tags will only be in the same files at the GDSII level because the hard tags are only defined for GDSII. Tag Readers may be configured to read only hard tags by reading only the text structures with & as a delimiter. Tag readers that wish to have the “audit trail” capability of the soft tags will be configured to read the text structures with % in them.

This standard details the implementation details of this new Soft IP tag. The details are described in the following three sections: “System Level Requirements,” “Tag Requirements,” and “Tool Requirements.”

2.3 System Level Requirements

These requirements encompass the general structure of the Soft IP Tagging Standard.

Req 2.3.1 The Format and Structure of Soft IP Tags Is Similar to That of Hard IP Tags.

Explanation: A Soft IP tag adds a set of tag information with content similar to that identified in the Hard IP tagging standard for GDS II. The format and structure is a reserved character, followed by a keyword and keyword information, as shown in the following physical tagging example:

& Vendor CompanyX & Product FunctionX & Version 1.4 & Metric 111703

The standard is “VSI Alliance Virtual Component Identification Physical Tagging Standard (IPP 1 1.0).” It is available from the VSIA Web site at www.vsi.org.

Req 2.3.2 The Soft IP Tags specific to technology should only be inserted when mapping the file to a technology.

Explanation: The techno and area tags are not applicable to HDL or RTL level designs. These tags must not be inserted until the design is physically mapped.

Req 2.3.3 All Tag Information Must Be Passed from the Input Files to the Output Files Unless Duplication Removal Is Used.

Explanation: At each stage of the IP/Chip design development process, all of the Soft IP tags from the previous stages are copied and added to that stage’s output. For example, a Verilog output of a chip level netlist could contain the following lines:

// {Soft IP Tag Identifier} % Vendor CompanyY % Product USBHOST2.0 % Version 1.4 % Metric 011702

// {Soft IP Tag Identifier} % Vendor CompanyY % Product USBHOST2.0 % Version 1.0 % Metric 090703

// {Soft IP Tag Identifier} % Vendor EDAZ % Product SynthesizerG % Version 4.3a % Metric 111703

These lines might have originated from the original source or from an intermediary step.

Any Soft IP tags that are generated from the current design step must also be passed to the output files. Generated tags are subject to the generated tag control requirements detailed in Section 2.5, “Tool Requirements.”

Duplication removal is discussed in Section 2.5, “Tool Requirements.”

Req 2.3.4 Every Soft IP Tag Must Be Delimited as a Separate Entity in the Output File.

Explanation: Each Soft IP tag must be considered a complete, individual entity. Soft IP tags cannot be combined and passed on as a new tag. The reader of the tags in the final GDS II output file is not able to determine the correctness or intent of combined tags.

Req 2.3.5 Specified Keywords Cannot Be Duplicated Within a Tag.

Explanation: The following keywords cannot be duplicated within a single tag:

- Vendor
- Company
- Version
- Metric
- IP_Owner
- Techno
- Area
- Celltype
- Cell_Id
- Signature
- Tag_Spec
- Date_Time
- Process_Step

The reader of the tags in the final GDS II output file is not able to determine the correctness or intent of combined tags. These keywords are described in Section 2.4, “Tag Requirements.”

Req 2.3.6 Soft IP Tags Must Be Written to the Identified Layer in the GDS II Output File as Specified in the VSIA Physical Tagging Standard.

Explanation: At the final step, these lines are added to the GDS II at the reserved layer specified by the VSIA Physical Tagging Standard. The information in this layer is the GDS II information, as required by that standard, and additional Soft IP tag lines that are copied from the previous stages.

This standard is “VSI Alliance Virtual Component Identification Physical Tagging Standard (IPP 1 3.0).” This standard is available from the VSIA Web site at www.vsi.org.

Req 2.3.7 Soft IP Tags Must Be Placed in the Top Most Hierarchical Level That Uses the IP.

Explanation: The Soft IP tags should be written into the output file at the topmost hierarchical level that uses the IP or script. If multiple files are generated from the design step, the Soft IP tags should be placed only in the files that contain the topmost hierarchical level that uses the IP or script. Users must follow this rule when generating a Soft IP tag for a source RTL file. For example, a Soft IP tag placed in a source Verilog file must be in the top-level Verilog module. It must be inside the module, not just in the file.

This requirement removes any ambiguity as to the placement or duplication of the Soft IP tags.

2.4 Tag Requirements

The following requirements describe the actual format of the Soft IP tag, the different elements of the tag, and the constraints upon those elements.

Req 2.4.1 Soft IP Tag Format

Explanation: The Soft IP tag has the following format:

{file format delimiter} {soft IP Tag delimiter} % {keyword} {data} % {keyword} {data} % …

This format follows the same general structure as the Hard IP tagging standard with the additional elements to allow identification of the Soft IP tag in different output file formats.

Req 2.4.2 The First Characters are File Format Delimiter Followed By a Space Character.

Explanation: The first character of a Soft IP tag within a tool output must be a file format delimiter. This delimiter identifies the Soft IP tag as a repository of information about the design, but not a design element itself. Each tool-specific format delimiter might be different. For example, comments in various programming languages might be delimited by any of the following characters: /*…..*/, #, or //. The delimiter must be followed by a space character to delineate the end of the delimiter and the beginning of the rest of the tag.

In order to make this tagging standard implementable in an expedient fashion, the tool-specific format delimiter should be an existing character set, if possible. For example, the tagging with a Verilog file uses the // comment characters as the Verilog file format delimiter. By using the comment delimiter, the Soft IP tag is seen as a non-design element. Use of these characters does not require any changes to the Verilog language to implement the Soft IP tag.

If all tools within the design development path support comments in their output formats, the delimiters should be the comment character or character sets.

Req 2.4.3 Verilog File Format Delimiter

Explanation: The file format delimiter for Verilog is two front slash characters (//), sequentially with no space between them, as with the Verilog comment delimiters. The delimiter characters should be the first characters on the Soft IP tag line. These characters appear as:

```
//
```

Req 2.4.4 VHDL File Format Delimiter

Explanation: The file format delimiter for VHDL is two dash characters (--), sequentially with no space between them, as with the VHDL comment delimiters. The delimiter characters should be the first characters on the Soft IP tag line. These characters appear as:

```
--
```

Req 2.4.5 Tcl File Format Delimiter

Explanation: The file format delimiter for Tcl and UNIX/Linux shell scripts (csh, tcsh, bash, and so on) is a pound sign character (#), as with the Tcl and UNIX/Linux shell script (csh, tcsh, bash, and so on) comment delimiters. It should be the first character on the Soft IP tag line. This character appears as:

```
#
```

Req 2.4.6 The Soft IP Identifier is the First Element of the Tag.

Explanation: The Soft IP tag must be identified as a special element of a design or be considered as a comment. The tool must be able to identify this set of characters to pass onto the next design tool. For this specification, the Soft IP tag identifier is the following string:

```
VSIA_Soft_IP_Tag
```

An example of this is a Soft IP tag implemented in Verilog. The beginning of the tag appears as follows:

```
// VSIA_Soft_IP_Tag %
```

The percent character (%) is the delimiter between elements of a tag.

Req 2.4.7 Space and Percent Characters Must Be Placed Between Each Element.

Explanation: A space character followed by a percent character and another space character (%) must be placed between each element of a tag. An element is either the Soft IP tag identifier or a keyword and its data. For example, the percent characters and spaces in the following example are used correctly:

// VSIA_Soft_IP_Tag % Vendor *ProviderC* % Product *XXY* % Version *1.1* % Metric *0304*

The items shown in italic are the data for each of the keywords.

Req 2.4.8 The First Keywords are Required Elements of Every Tag.

Explanation: The following keywords are required for every tag for soft (RTL) IP. This supports the Hard IP tagging standard structure.

Table 2: Required Keywords 1

Keyword	Argument(s)	Definition	Example
Vendor	*String*	Complete legal name of IP Vendor	% Vendor CompanyD
Product	*String*	Complete name of product	% Product New_IP
Version	*String*	Version of product	% Version 1.3a
Metric	*String*	Floating point number	% Metric 47.3
IP_Owner	*String*	Organization name owner of IP, Library or memory generator	% IP_Ownder WMSG
Celltype	*LIB or IP obligatory* *LEAF optional* *Empty also allowed*	LIB = Library cell or memory generator IP = reusable high level block or memory LEAF = leafcell. Smallest entity, could not be found alone, only in an IP or a LIB Note : for memory generator, both IP or LIB are authorized	% Celltype LEAF
Cell_Id	*String*	Unique identifier of Cell: - More specific IP name - Cell name for Library cell - Generator name followed by all parameters values used for cut generation	% Cell_Id nd2x1
Signature	*String*	Signature to check consistency layout	% Signature <tbd>
Tag_Spec	*String*	Corresponds to the Revision of VSI Physical Standard Tagging document	% Tag_Spec 3.0
Date_Time	*String*	Tagging date YYYYMMDD[_HHMMSS]	% Date_Time 20061014_231542

Req 2.4.9 Required Keywords for physically mapped IP

Explanation: The following keywords are required for every tag for physically mapped (GDSII) IP. This supports the Hard IP tagging standard structure. These tags are specific to technology and must be included during the mapping to a specific technology. They are not required for structural, non-technology mapped IP.

Table 3: Required Keywords 2

Keyword	Argument(s)	Definition	Example
Techno	*String*	Defines the process technology used. Vendor technology name in upper case.	% Techno CMOS090GP
Area	*Floating*	Product or cell area in mm²	% Area 10.514

Req 2.4.10 Character Constraints for Data Information

Explanation: The data information after a keyword can contain only the following characters:

- Alphanumeric characters: a-z A-Z 0-9
- Space character:
- Punctuation characters: . , ? ! " ' ` : ;
- Special characters: ( ) { } [ ] \ / | _ - = + @ # $ ^ * < > ~

The ampersand (&) and percent (%) characters are forbidden in the data information after a keyword. These characters are not allowed because they mark the beginning of the next keyword for the Hard IP and Soft IP tags, respectively.

The total length of reserved character, keyword, and keyword data cannot exceed 512 characters.

Req 2.4.11 Additional Keywords For Soft IP Tags

Explanation: Additional keywords can be added to the tag, as detailed in the VSIA Physical Tagging standard. We propose the addition of some specific keywords for Soft IP. These keywords would be the following:

Table 4: Additional Keywords

Keyword	Argument(s)	Definition	Example
Process_Step	*String*	The design step from which the tag is obtained.	% Process_Step Source
Hier	*String*	Hierarchical level identifier used when flattening a hierarchy	% Hier Subgroup1

Req 2.4.12 Process_Step Keyword Use

Explanation: This keyword identifies the step within the development process that generated the Soft IP tag. These steps are identified as:

- Source
- Netlist
- Synthesis
- Timing
- Layout
- Wiring
- Checking
- {_User Defined}

_User Defined - A user of the Soft IP tag may elect to create an additional processing step identifier. This identifier must start with an underscore character. This underscore character allows automated report generation tools to skip or ignore the user-defined identifier. For example:

> % Process_Step _Formal_verification

Req 2.4.13 Hier Keyword Use

Explanation: This keyword is used when a hierarchical design is flattened. This flattening can cause an inadvertent duplication of tags. To preserve the correspondence between the number of tags that are in the final GDS II file and the number of IP elements, the Soft IP tag must be made unique when flattening a level. This keyword allows the Soft IP tag to be unique.

This keyword should be added as the last keyword at the end of the tag.

For example, a design with two hierarchical blocks (subblock1 and subblock2) might be flattened. Each hierarchical block would contain the following Soft IP tag:

> // VSIA_Soft_IP Tag % Vendor Green,_Inc. % Product CoreZ % Version 1.4 % Metric 02 % Process_Step Source % Date_Time 20040829_130400

After flattening the design, two identical tags would be present in the flattened output file. Redundancy removal (as described in Section 2.5, “Tool Requirement”) would eliminate one of these tags and cause an inaccurate count of the number of IP uses. The following example shows two identical tags delineated by the use of the Hier keyword:

> // VSIA_Soft_IP Tag % Vendor Green,_Inc. % Product CoreZ % Version 1.4 % Metric 02 % Process_Step Source % Date_Time 20040829_130400 % Hier subblock1
>
> // VSIA_Soft_IP Tag % Vendor Green,_Inc. % Product CoreZ % Version 1.4 % Metric 02 % Process_Step Source % Date_Time 20040829_130400 % Hier subblock2

Req 2.4.14 User-defined Keywords

Explanation: User-defined keywords are allowed within the Soft IP tag. However, the keywords must be delineated by the percent (%) character, and the actual keyword must start with an underscore (_) character. The only characters allowed within the keyword are alphanumeric and underscore characters. Spaces are not allowed.

Allowed characters: a-z A-Z 0-9 _

For example:

> // VSIA_Soft_IP Tag % Vendor Green,_Inc. % Product CoreZ % Version 1.4 % Metric 011702 % _USER_KEYWORD user1

Req 2.4.15 All Soft IP Tags Must End with Characters that Delimit the Tag as a Separate Line or Entry in the Output File.

Explanation: The Soft IP tag must end with whatever characters are required to separate the Soft IP tag as an individual element of the output file. In many cases, this requirement reduces to the use of a carriage return or line feed and carriage return. The ending characters, like the file format delimiter described previously, are dependent on the format of the output file.

2.5 Tool Requirements

The following requirements describe the functions needed within an EDA tool that processes or handles Soft IP tags.

Req 2.5.1 Tag Identification and Preservation

Explanation: As a tool processes either the design data or a control script, it must check for the presence of legal Soft IP tags within the file. If any legal Soft IP tags are present, the tags must be passed to the output file using the format delimiter for that particular type of file format.

All output files must contain the Soft IP tag information.

Req 2.5.2 Tag Ordering Within Output Files Must Duplicate the Input File Ordering of the Tags.

Explanation: As the Soft IP tags are passed from one tool to the next, the ordering of the tags implies the history of the design development process for that tag. Therefore, tools that pass the Soft IP tags to the next stage must maintain the ordering of the Soft IP tags from the input file.

Tags added to the list of Soft IP tags should be added to the end of the list.

Req 2.5.3 Additional Tags Should Be Placed in the Output File at the End of the List.

Explanation: Additional tags added to the list of Soft IP tags should be added to the end of the list in the order in which they are obtained and generated. For example:

A synthesis tool takes in a source file with the following tag:

> // VSIA_Soft_IP Tag % Vendor Green, Inc. % Product CoreZ % Version 1.4 % Metric 02 % Process_Step Source % Date_Time 20020829_130400

The synthesis tool then generates a tag that identifies the version of the compiler that is used to compile the design.

// VSIA_Soft_IP Tag % Vendor Blue, Inc. % Product Synth_Compiler % Version 4.4 % Metric 02 % Process_Step Synthesis % Date_Time 20030407_110500

The sequence of Soft IP tags in the output file format now looks like this:

// VSIA_Soft_IP Tag % Vendor Green, Inc. % Product CoreZ % Version 1.4 % Metric 02 % Process_Step Source % Date_Time 20020829_130400

// VSIA_Soft_IP Tag % Vendor Blue, Inc. % Product Synth_Compiler % Version 4.4 % Metric 02 % Process_Step Synthesis % Date_Time 20030407_110500

The final tag comes from a synthesis script that identifies the script used for netlist creation. The tag looks like this:

// VSIA_Soft_IP Tag % Vendor Green, Inc. % Product CoreZ_synth_script % Version 1.2 % Metric 01 % Process_Step Synthesis % Date_Time 20030407_111000

The final output format looks like this:

// VSIA_Soft_IP Tag % Vendor Green, Inc. % Product CoreZ % Version 1.4 % Metric 02 % Process_Step Source % Date_Time 20020829_130400

// VSIA_Soft_IP Tag % Vendor Blue, Inc. % Product Synth_Compiler % Version 4.4 % Metric 02 % Process_Step Synthesis % Date_Time 20030407_110500

// VSIA_Soft_IP Tag % Vendor Green, Inc. % Product CoreZ_synth_script % Version 1.2 % Metric 01 % Process_Step Synthesis % Date_Time 20030407_111000

Note that the ordering of the second and third tags is time dependent on what is accomplished first, the creation of the tag or the reading of the synthesis script in the synthesis tool. This example assumes that the tool tag is generated first.

Req 2.5.4 Ordering Priority of Multiple Tag Input Sources in the Output File

Explanation: If several Soft IP tag streams or groups are combined at a single tool, the groups may be in any order, but the ordering within the groups must be maintained. Any additional tags added at this step could be added individually to each group or appended on the end of the concatenation of all of the groups.

Req 2.5.5 Adding Tool-Specific Tags

Explanation: The tool may add a Soft IP tag within the output file that contains information about the tool used in the design process.

Tags created by the tool may be modified, removed, or reordered. The only exception to this rule is the addition of the Hier keyword to a previously created tag. This addition, modification, removal, or reordering of the current tool-generated tags should fall under the verboseness controls, as specified in the following requirement.

Req 2.5.6 Verboseness Controls of Tool-Generated Tags

Explanation: An EDA tool that allows adding tool-specific tags to the output format of the tool must also provide controls that allow users to control the verboseness of the tags. This allows users to create a Soft IP tag and to disallow the tag creation. The verboseness control must provide the following controls:

- Creation of tag
- No creation of tag

The EDA tool provider in this verboseness control must also handle iterative use of the tool during the design process. The verboseness control must additionally provide the following controls:

- Creation of tag during first pass only
- Creation of tag for every pass through the tool
- Update of tag during an iteration (for example, Date_Time)
- Creation of tag during first pass and last pass
- Update of tag that uses a _User_Defined keyword for number of passes

For example, if a synthesis optimization tool is rerun twenty times to produce the correct output, the verboseness controls must allow the user to control the way the Soft IP tag is handled. In this case, several possibilities exist, including the following:

- There could be twenty tags.
- The tag could be updated after each pass.
- The tag could contain the first pass information.
- There could be a first-pass tag and a last-pass tag.
- The tag could contain the number of iterations through the tool.

The verboseness controls must also handle the ability to allow user definition of the position of the tag within a hierarchical design. Some possibilities include the following:

- The tag could be entered at the top of the hierarchy.
- The tag could be added to each previous tag stream at the top of the hierarchy.
- The tag could be entered a single time in each of the lower hierarchies.
- The tag could be entered in each of the tag streams in the lower hierarchy.
- The tag could be entered in some combination of the above.

The actual implementation of the verboseness control is left to the tool designer.

Req 2.5.7 Verboseness Controls of Tags Within Control Scripts

Explanation: An EDA tool that allows adding script-specific tags to the output format of the tool must also provide controls that allow users to control the verboseness of the tags. This allows users to create a Soft IP tag and to disallow the tag creation. The verboseness control must provide the following controls:

- Creation of tag
- No creation of tag

The EDA tool provider in this verboseness control must also handle iterative use of the script during the design process. The verboseness control must additionally provide the following controls:

- Creation of tag during first pass only
- Creation of tag for every pass through the tool
- Update of tag during an iteration (for example, Date_Time)
- Creation of tag during first pass and last pass
- Update of tag that uses a _User_Defined keyword for number of passes

For example, if a synthesis optimization tool is rerun twenty times to produce the correct output, the verboseness controls must allow users to control the way the Soft IP tag within the script is handled. In this case, several possibilities exist:

- There could be twenty tags.
- The tag could be updated after each pass.
- The tag could contain the first pass information.
- There could be a first pass tag and a last pass tag.
- The tag could contain the number of iterations through the tool.

The verboseness controls must also handle the ability to allow user definition of the position of the tag within a hierarchical design. Some possibilities include the following:

- The tag could be entered at the top of the hierarchy.
- The tag could be added to each previous tag stream at the top of the hierarchy.
- The tag could be entered a single time in each of the lower hierarchies.
- The tag could be entered in each of the tag streams in the lower hierarchy.
- The tag could be entered in some combination of the above.

The actual implementation of the verboseness control is left to the tool designer.

Req 2.5.8 Tags Must Be Made Unique When Duplicate Tags are Found While Flattening a Hierarchical Design.

Explanation: Flattening can cause an inadvertent duplication of tags. To preserve the correspondence between the number of tags that are in the final GDS II file and the number of IP elements, the Soft IP tag must be made unique when flattening a level. The Hier keyword can be used to make the Soft IP tag unique.

When a tool flattens a hierarchical design, the tool should examine all Soft IP tags found within each hierarchy. If a duplicate Soft IP tag is found, those Soft IP tags should be made unique by adding a Hier keyword and an appropriate name in the data section of the keyword to the tag. This keyword should be added to the end of the tag as the last keyword.

The data section of the Hier keyword should be the sub-hierarchy name.

Req 2.5.9 Duplicate Tag Removal Should Be Allowed.

Explanation: Some tools can produce more than one output file for use in the following design steps. Previous sections of this document require that all output files must contain the Soft IP tags obtained from earlier and present design steps. Therefore, all output files from a design step should have the Soft IP tags.

While processing input files, if duplicate tags are found in the same hierarchical design, users should be able to remove the duplicated tags. This allows an accurate correspondence between the number of tags in the final GDS II file and the number of IP elements in the design.

The tool should allow or cause the duplicate tag not to be carried to the output of that design step. This allows an accurate correspondence between the number of tags in the final GDS II file and the number of IP elements in the design.

Req 2.5.10 Structure of Soft IP Tag Within the GDS II Output File.

Explanation: The Soft IP tag should be represented as a single “text” line per keyword in the GDS II output file. In other words, the Soft IP tag is broken up into a set of lines, with one keyword and its associated data per line.

The “VSIA_Soft_IP_Tag” delimiter is dropped when the tag is converted to GDS II format.

Each text string should start with the percent (%) character. This differentiates Soft IP tag keywords from Hard IP tag keywords.

The X and Y coordinates should be the same value for all keywords within a single Soft IP tag. This allows coherency of the keywords for the entire Soft IP tag.

The following example shows the correct GDS II structure for a Soft IP tag.

```
struct ip_uart
[...]
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Vendor Company2"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Product IPX"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Version 01.01"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Metric 0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Date_Time 20030312"
```

Note that the X and Y coordinates are the same for all of the keywords.

The following example shows a GDS II structure with a Hard IP Tag and two Soft IP tags:

```
struct ip_uart
[...]
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,1 "& Vendor Company1"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,1 "& Product UART"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,1 "& Version 02.01"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 5,1 "& Metric 0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 6,1 "& IP_Owner Dsgn2"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 7,1 "& Techno CMOS90GP"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 8,1 "& Area 10.512"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 9,1 "& Celltype IP"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 10,1 "& Cell_Id F8x8"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 11,1 "& Tag_Spec 3.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 14,14 "& Date_Time 20051015"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 13,13 "% Vendor Company2"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 13,13 "% Product IPX"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 13,13 "% Version 01.01"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 13,13 "% Metric 0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 13,13 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 13,13 "% Date_Time 20030312"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 14,14 "% Vendor Company3"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 14,14 "% Product ZZZ"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 14,14 "% Version 03.03"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 14,14 "% Metric 0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 14,14 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 14,14 "% Date_Time 20030415_150530"
```

Req 2.5.11 The Order of Soft IP Tags Within the GDS II Output File and Use of X,Y Coordinates

Explanation: The ordering of the tags within the GDS II file should reflect the ordering as received prior to or during the GDS II generation. Verboseness controls are the same as previously described in this standard.

This ordering of the tags requires that the text line utilizes the same X and Y coordinates to explicitly describe the ordering. The X and Y coordinates value should increment with every Soft IP tag in the order of reception of the tags.

The same X and Y coordinate value is used for every keyword value for a particular Soft IP tag. The X and Y coordinates are both required because, in flattening a hierarchical GDS II file, Soft IP tags from sub-hierarchal GDS II files may end up with the same X or Y coordinate as other Soft IP tags. However, by definition, both coordinates cannot be given the same coordinate set during the flattening process.

The value for the X or Y coordinate used is not recommended to be beyond the macro or chip boundary.

In the following example, the X and Y coordinates are incremented between text lines to differentiate the order. The Soft IP tag with a vendor of Company2 has been placed in the tag stream at an earlier time than the second tag with vendor of Company3.

```
struct ip_uart
[...]
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Vendor Company2"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Product IPX"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Version 01.01"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Metric 0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Date_Time 20030312"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Vendor Company3"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Product ZZZ"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Version 03.03"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Metric 0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Date_Time 20030415"
```

3. Recommended Usage

This section of the standard details the suggested usage and manipulation of the Soft IP tag. This section contains information for IP developers, EDA tool vendors, and chip developers.

3.1 IP Developers

For the IP developer, the Soft IP tag creates the ability to mark the source and deliverable files for a customer with product-specific information that is not lost in the design process and can aid in communicating IP enhancements, problems, and changes to a customer.

The IP is marked with a comment that is placed in the Verilog or VHDL. An example of this in Verilog is:

> // VSIA_Soft_IP_Tag % Vendor Yellow_inc % Product HOST2.0 % Version 2.3 % Metric 011702 % Process_Step Source % Date_Time 20031121_125534

This tag shows the producer of the IP, the product, and the version of the IP. This version information is important for establishing which IP version a customer is using. The customer can examine the GDS II or some other intermediate output format for the product and version information, and then use that information when dealing with communications from the IP developer.

After this Soft IP tag is placed in the source or netlist deliverable, the tag is carried throughout the rest of the development process and is eventually deposited in the GDS II for the chip.

3.1.1 Soft IP Tag Location and Frequency of Use in Source Code

The Soft IP tag for a synthesizable IP should be captured a single time in the main module for the design. The Soft IP tag should be contained in the first few lines of the module.

Because correct coding style dictates the division of separate modules into separate files, the natural inclination is to place a Soft IP tag within each module. However, it is not recommended to place a Soft IP tag within each sub-module in the source files, since this could lead to excessive amounts of information from the design process in the GDS II level. For example, consider a chip that contains 20 pieces of IP. If each IP has 100 sub-modules and each sub-module contains a Soft IP tag, the final chip GDS II would have 2,000 Soft IP tags from the source files alone.

The main module should contain the Soft IP tag, capturing the version of the IP within that tag for all of the modules released with that main module.

For example, a piece of Soft IP in Verilog might appear as the following three module definitions:

```
Module TOPIP ( inputs....., outputs.....);
// VSIA_Soft_IP Tag % Vendor Green, Inc. % Product CoreZ % Version 1.4 % Metric 02 %
Process_Step Source % Date_Time 20020829_130400
Inputs ..............
Outputs ............
Submodule1 sub1 ( ... );
Submodule2 sub2 ( ... );
endmodule
Module Submodule1 ( inputs....., outputs.....);
Endmodule
Module Submodule2 ( inputs....., outputs.....);
endmodule
```

The submodule declarations would not have the Soft IP tag.

3.1.2 Soft IP Tag Location and Frequency of Use in Control Scripts

The Soft IP tag should be contained within the first few lines of the main control script. It is not recommended to place a Soft IP tag within each referenced file in the control script, since this could lead to excessive amounts of information from the design process in the GDS II level. The main file should contain the Soft IP tag, capturing the version of the control script within that tag for all of the control files released with that main control script.

3.1.3 User-defined Keyword Usage

This standard allows for the use of user-defined keywords. These keywords must be prefaced with an underscore (_) character. The use of additional keywords can be valuable in the identification and quality IP process. However, care must be taken to ensure that these keywords are not used arbitrarily. The following points are some suggested guidelines for user-defined keyword usage:

- Have a real use for the keyword.
- Use only predefined keywords that are common throughout your organization.
- Keep the number of characters to a minimum.
- Be aware that software may be used to extract reports from the final GDS II file.
- If appropriate, publish in product literature the keyword and the definition of its use.

3.2 EDA Developers

EDA developers are concerned with two aspects of the Soft IP tag. These aspects are passing previous tag information to the output and adding tool specific tag information, if required.

3.2.1 Passing Previous Tag Information

The EDA tool provider must ensure that any Soft IP tag that is present in the input file must also appear in an appropriate manner in the output file. This tag information does not have to be combined with other Soft IP tags in a single section, but this may be a good way of collecting and storing the tag information.

The standard requires that the spatial order be maintained when passing the tag information into the output file. If multiple files are read into the tools, the order of the reading of the files should ascertain the output ordering of the Soft IP tags.

3.2.2 Adding Tool-Specific Soft IP Tags

The EDA tool provider has the ability to add Soft IP tags to the output file. These tags can contain the time of operation and the version of the tool that was used. These pieces of information can be useful, but care must be taken to avoid erroneous or excessive use of Soft IP tags.

The tool should not create a tag similar to an output log. The Soft IP tag is not intended as an internal log file. Rather, the tool-specific Soft IP tag should be used only if there is some type of tool-specific information that the customer may need after chip development is completed. The following example illustrates a scenario where there is a need for tool-specific information.

If the Soft IP tag is added by a tool, the tag contains date of use and tool version information. At a later date, a problem might be identified with a tool. All chips that are manufactured with output of this tool are then checked, revealing that the problem only appears with only a single version of the tool. The customer can then examine the GDS II record and check all the Soft IP tags to determine if that version of the tool was used during development of the chip.

3.2.3 Verboseness Control for Adding Tool-Specific Soft IP Tags

The EDA tool provider must provide verboseness settings to allow customers to adjust the output of Soft IP tags into the output files. These verboseness controls should prevent an inordinate amount of tool-specific IP tags to be released into the output of the tool. This could be caused by an iterative use of the tool, where every time the tool is used, a Soft IP tag is appended. The EDA tool provider could allow different settings or control of the way an iterative tag is handled. The tag could be updated or replaced, or only the original could be kept. The decision as to what actual output is generated should be left to the user.

The verboseness tools should not normally prevent Soft IP tags from the input file from being passed to the output file.

3.2.4 Tool-specific User-Defined Keyword Usage

This standard allows for the use of user-defined keywords. These keywords must be prefaced with an underscore (_) character. The use of these additional keywords can be valuable in the identification and quality IP process. However, care must be taken to ensure that these keywords are not used arbitrarily. The following points are some suggested guidelines for user-defined keyword usage:

- Have a real use for the keyword.
- Only use predefined keywords common throughout your organization.
- Keep the number of characters to a minimum.
- Be aware that software may be used to extract reports from the final GDS II file.
- If appropriate, publish in product literature the keyword and the definition of its use.

3.3 Chip Developers

A chip developer uses the Soft IP tag by examining the outputs during the design process or in the GDS II file. In these files, the customer can determine the source, version, and date of the IP, and perhaps also the processing of the IP through the design process.

Before committing the design to silicon, customers can determine if the version of the IP on the chip is current, and if it contains any problems that affect their design.

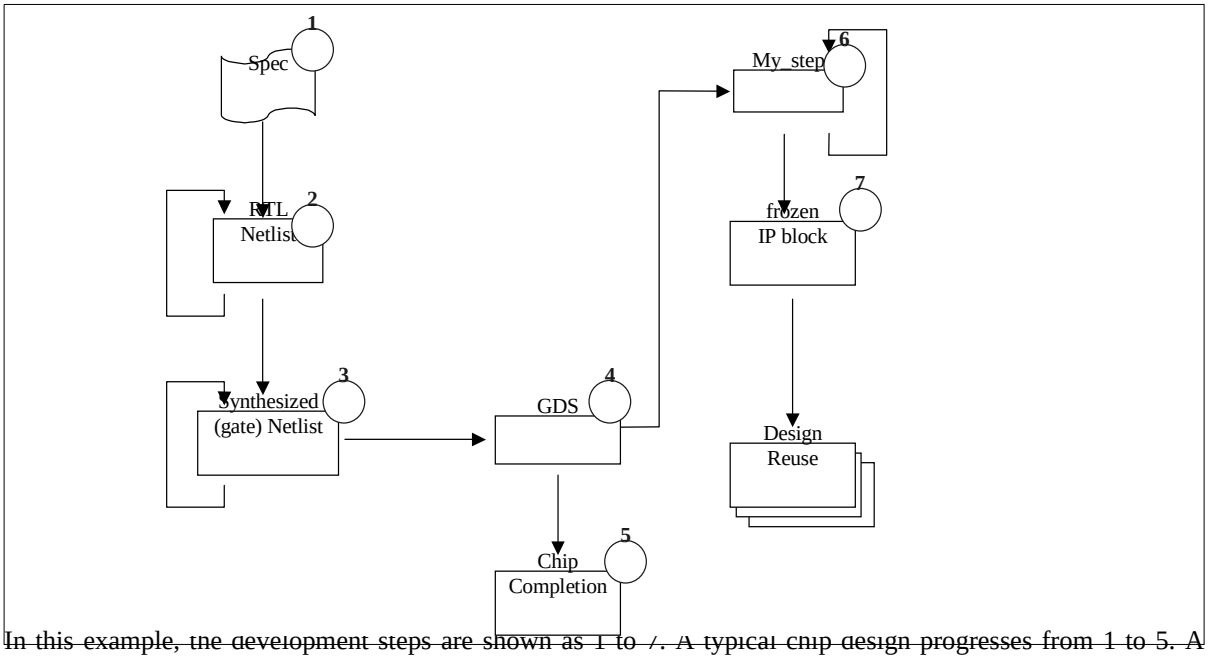
As a general strategy, the chip developer must decide the level of verboseness to allow during the design process, as well as the use of the information ultimately contained within the GDS II file.

4. Example Implementations

4.1 Example 1

The following diagram shows the possible steps in the design process.

Figure 1: Example Design Process



In this example, the development steps are shown as 1 to 7. A typical chip design progresses from 1 to 5. A physically designed IP block progresses from 1 to 4, and then from 6 to 7.

The following list shows the IP tag at the different steps of the development process. The tool-specific format delimiter is shown as ?? because it is not known at this time.

1. // VSIA_Soft_IP_Tag % Vendor Blue_LLC % Product ASIC1.3 % Version 12.0 % Metric 103003 % IP_Owner Dsgn1 % Celltype IP % Cell_Id Function1.1 % Tag_Spec 3.0 % Process_Step Specification % Date_Time 20030223_130501

2. // VSIA_Soft_IP_Tag % Vendor Blue_LLC % Product ASIC1.3 % Version 12.0 % Metric 103003 % IP_Owner Dsgn1 % Celltype IP % Cell_Id Function1.1 % Tag_Spec 3.0 % Process_Step Specification % Date_Time 20030223_130501

   // VSIA_Soft_IP_Tag % Vendor Blue_LLC % Product ASIC 12.0 % Version 1.3 % Metric 103003 % IP_Owner Dsgn2 % Celltype IP % Cell_Id Function2.0 % Tag_Spec 3.0 % Process_Step Source % Date_Time 20031102

3. ?? VSIA_Soft_IP_Tag % Vendor Blue_LLC % Product ASIC1.3 % Version 12.0 % Metric 103003 % IP_Owner Dsgn1 % Celltype IP % Cell_Id Function1.1 % Tag_Spec 3.0 % Process_Step Specification % Date_Time 20030223_130501

   ?? VSIA_Soft_IP_Tag % Vendor Blue_LLC % Product ASIC 12.0 % Version 1.3 % Metric 103003 % IP_Owner Dsgn2 % Celltype IP % Cell_Id Function2.0 % Tag_Spec 3.0 % Process_Step Source % Date_Time 20031102

   ?? VSIA_Soft_IP_Tag % Vendor Green_inc % Product synth_compiler % Version 10.1a % Metric 103003 % IP_Owner Grp1 % Celltype IP % Cell_Id Fn3 % Tag_Spec 3.0 % Process_Step Synthesis %

Date_Time 20040103_070000

4. text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Vendor Blue_LLC"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Product ASIC1.3"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Version 12.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Metric 103003"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% IP_Owner Dsgn1"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Celltype IP"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Cell_Id Function1.1"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Tag_Spec 3.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Process_Step Specification"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 1,1 "% Date_Time 20030223_130501"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Vendor Blue_LLC"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Product ASIC12.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Version 1.3"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Metric 103003"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% IP_Owner Dsgn2"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Celltype IP"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Cell_Id Function2.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Tag_Spec 3.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 2,2 "% Date_Time 20031102"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Vendor Green_inc"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Product synth_compiler"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Version 10.1a"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Metric 103003"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% IP_Owner Grp1"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Celltype IP"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Cell_Id Fn3.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Tag_Spec 3.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Process_Step Synthesis"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 3,3 "% Date_Time 20040103_070000"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Vendor Red_inc"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Product Chipdesign"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Version 5.00"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Metric 030403100304"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% IP_Owner Dsgn2"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Techno CMOS90GP"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Area 10.512"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Celltype IP"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Cell_Id Function2.0"

text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Tag_Spec 3.0"

text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Process_Step Layout"

text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY 4,4 "% Date_Time 20040213_130501"

The 42 text lines in the final GDS II output (in Step 4) represent the entire set of Soft IP tags from the previous design steps, as well as a tag that was created when the GDS II file was created. It contains the entire audit trail of the development process. A user can ascertain where the information for a particular piece of IP is derived by the Process_Step keyword and the ordering information contained within the X and Y coordinate.

4.2 Example 2

In the following example, a chip is to be designed with two pieces of IP, one in Verilog and the other in VHDL. The Verilog IP is synthesized using a synthesis script supplied by the IP provider. The following sequence shows the trail of the Soft IP tags.

Verilog Source for Core1

```
module Core1 ( );
// VSIA_Soft_IP_Tag % Vendor XXX % Product  Core1 % Version 5.00 % Metric
030403100304  %  Process_step Source
endmodule Core1;
```

VHDL Source for Core2

```
entity Core2 is
       port ();
-- VSIA_Soft_IP_Tag % Vendor YYY % Product  Core2 % Version 1.00 % Metric
03080902100304  %  Process_step Source
architecture rtl of Core2 is
begin
end;
```

Synthesis script for Core1

```
# VSIA_Soft_IP_Tag % Vendor XXX % Product Core1_synthesis % Version 2.00 % Metric
030403100304  %  Process_step Source
source "Core1.v"
# end script
```

The following steps are accomplished and the output of each stage is written out to Verilog format. The tags are shown as follows:

1) Synthesize Core1

Using synthesis tool, the core verilog, and associated synthesis script, a gate level netlist is generated. This is then output in Verilog format.

```
// VSIA_Soft_IP_Tag % Vendor XXX % Product  Core1 % Version 5.00 % Metric
030403100304  %  Process_step Source
// VSIA_Soft_IP_Tag % Vendor XXX % Product Core1_synthesis % Version 2.00 % Metric
030403100304  %  Process_step Source
// VSIA_Soft_IP_Tag % Vendor ZZZ % Product  Synthesis tool % Version 5.00 % Metric
030403100304  %  Process_step Synthesis
```

2) The next step is to combine the synthesized Core1 with Core 2 in the same netlist. This is then output in Verilog format.

```
// VSIA_Soft_IP_Tag % Vendor XXX % Product  Core1 % Version 5.00 % Metric
030403100304  %  Process_step Source
// VSIA_Soft_IP_Tag % Vendor XXX % Product Core1_synthesis % Version 2.00 % Metric
030403100304  %  Process_step Source
// VSIA_Soft_IP_Tag % Vendor ZZZ % Product  Synthesis tool % Version 5.00 % Metric
030403100304  %  Process_step Synthesis
// VSIA_Soft_IP_Tag % Vendor YYY % Product  Core2 % Version 1.00 % Metric
03080902100304  %  Process_step Source
```

3) The final step is layout and wiring with the results put into GDS II form.

```
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .1,.1 "% Vendor XXX"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .1,.1 "% Product  Core1"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .1,.1 "% Version 5.00"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .1,.1 "% Metric 030403100304"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .1,.1 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .2,.2 "% Vendor XXX"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .2,.2 "% Product  Core1_synthesis"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .2,.2 "% Version 2.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .2,.2 "% Metric 030403100304"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .2,.2 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .3,.3 "% Vendor ZZZ"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .3,.3 "% Product  Synthesis tool"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .3,.3 "% Version 5.00"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .3,.3 "% Metric 030403100304"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .3,.3 "% Process_Step Synthesis"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .4,.4 "% Vendor YYY"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .4,.4 "% Product  Core2"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .4,.4 "% Version 1.00"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .4,.4 "% Metric 03080902100304"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .4,.4 "% Process_Step Source"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Vendor GGG"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Product  Layout_tool"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Version 1.40"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Metric 030403100304"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% IP_Owner G1"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Techno CMOS65"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Area 15.002"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Celltype IP"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Cell_Id GFunc"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Tag_Spec 3.0"
text 63 TT 63 PR 0,2,0 MAG 0.001 R 90 XY .5,.5 "% Process_Step Layout"
```

Note the use of decimal numbering within the X and Y ordering.

5. Prototype Description

5.1 Scope and Use of Prototype

While developing this standard, a prototype was developed, in order to validate and debug the standard and provide further insight into possible ad-hoc methods of implementation. This section briefly discusses the prototype, and provides guidance for users who want to develop methods to implement this standard within their own design flows prior to the availability of support directly within EDA tools.

5.2 Summary of Prototype

The prototype effort may be summarized as follows:

- Using scripts only (without direct tool support), implement a Soft IP tagging flow from RTL (Verilog) to GDS II.
- Implement the above modifications on an existing flow consisting of the following:
  - Design Compiler
  - JupiterXT
  - Physical Compiler
  - Astro
- Pass an existing, simple test design through this flow with the addition of Soft IP tags, including RTL tags and tool tags.
- Provide this information as a section in the standard.

5.3 Prototype Implementation

5.3.1 Brief Technical Description

Passing IP tags within a design can be as simple or complex as desired. In the case of this prototype flow, the choice was made for simplicity. This simplicity of approach should make the prototype information useful for the broadest set of design practices and flows.

The prototype flow assumes that the IP tags have already been inserted into the RTL (in this case, Verilog) in compliance with the standard. The following excerpt shows a Verilog module in which a simple VSIA Soft IP tag has been added.

Figure 2: Sample Verilog with VSIA Tag

```
//                ********** SAMPLE VERILOG W/ VSIA TAG ************
. . .

module core (
              clk, rst_n, bi, vi, vo
);

// VSIA_Soft_IP_Tag % Vendor Synopsys % Product IPcore % Version 1.0 % Metric 1234

input                           clk;

. . .
```

An additional assumption that was made is that a list of all the RTL files required to build the design of interest is provided in file form. This file is referred to herein as a "manifest file." A file of this type often exists in order to simulate or build the design. In this case, it is used as a list of files to process in order to locate IP tags. The following example shows a manifest file:

Figure 3: Manifest File Example

```
######################### Manifest File Example ################################

$DESIGN_ROOT/hdl/sample_design/core.v
$DESIGN_ROOT/hdl/sample_design/vio.v
$DESIGN_ROOT/hdl/sample_design/module_fifo.v
$DESIGN_ROOT/hdl/sample_design/module_fifo_core.v
$DESIGN_ROOT/hdl/sample_design/module_datapath.v
$DESIGN_ROOT/hdl/sample_design/module_datapath_core.v

##EOF##
```

In the previous two example code segments, note that while several files are required in order to build the design "core," only the "core.v" file contains tag information.

One last point of simplification is how the tag data is stored while being processed. Instead of trying to understand multiple tools, databases, and so on, it was decided to maintain all the tag information in text files separate from the tool databases. This allows for a simple set of essentially three scripts to be used for all the tag processing.

The simple processing flow through the prototype may be described in the following steps:

1) The first script processes the files listed in the manifest file, extracts the Soft IP tags from them, and places the information in the text file (the "Tagfile") in a tagging area for further processing.

2) During each data processing step within the flow, a second script is executed that appends the tool, and if desired, the script tag information to the Tagfile.

3) The third script optionally re-inserts tags into the design database just prior to writing the GDS II for the design. This point in the flow was chosen for the prototype. It would also be possible to continue to process the design until final GDS II is written and then process the Tagfile and add the tags to the final GDS II, similar to the process for Hard IP tagging. This second method has the advantage of being able to add post-GDS II tool tags (such as DRC/LVS) to the GDS II, using the same method that is used for the other tags.

The following example shows the Tagfile after several processing steps have occurred in this process. (Note that script tags and date stamping were not implemented in the prototype.)

Figure 4: Tagfile Example

```
VSIA_Soft_IP_Tag % Vendor Synopsys % Product IPcore % Version 1.0 % Metric 1234
VSIA_Soft_IP_Tag % Vendor Synopsys % Product Design_Compiler % Version syn/2003.12
VSIA_Soft_IP_Tag % Vendor Synopsys % Product Jupiter % Version jupiter/2003.09-SP1
VSIA_Soft_IP_Tag % Vendor Synopsys % Product Astro % Version astro/2003.09-SP1
VSIA_Soft_IP_Tag % Vendor Synopsys % Product Physical_Compiler % Version syn/2003.12
```

Without direct tool support, the tag manipulation that occurs is driven primarily through scripting external to the tools. This allows the information provided here to be applicable to tools and flows other than those used for the prototype work (Synopsys-centric flow). The languages, scripts, and tools utilized are discussed in the following sections in more detail.

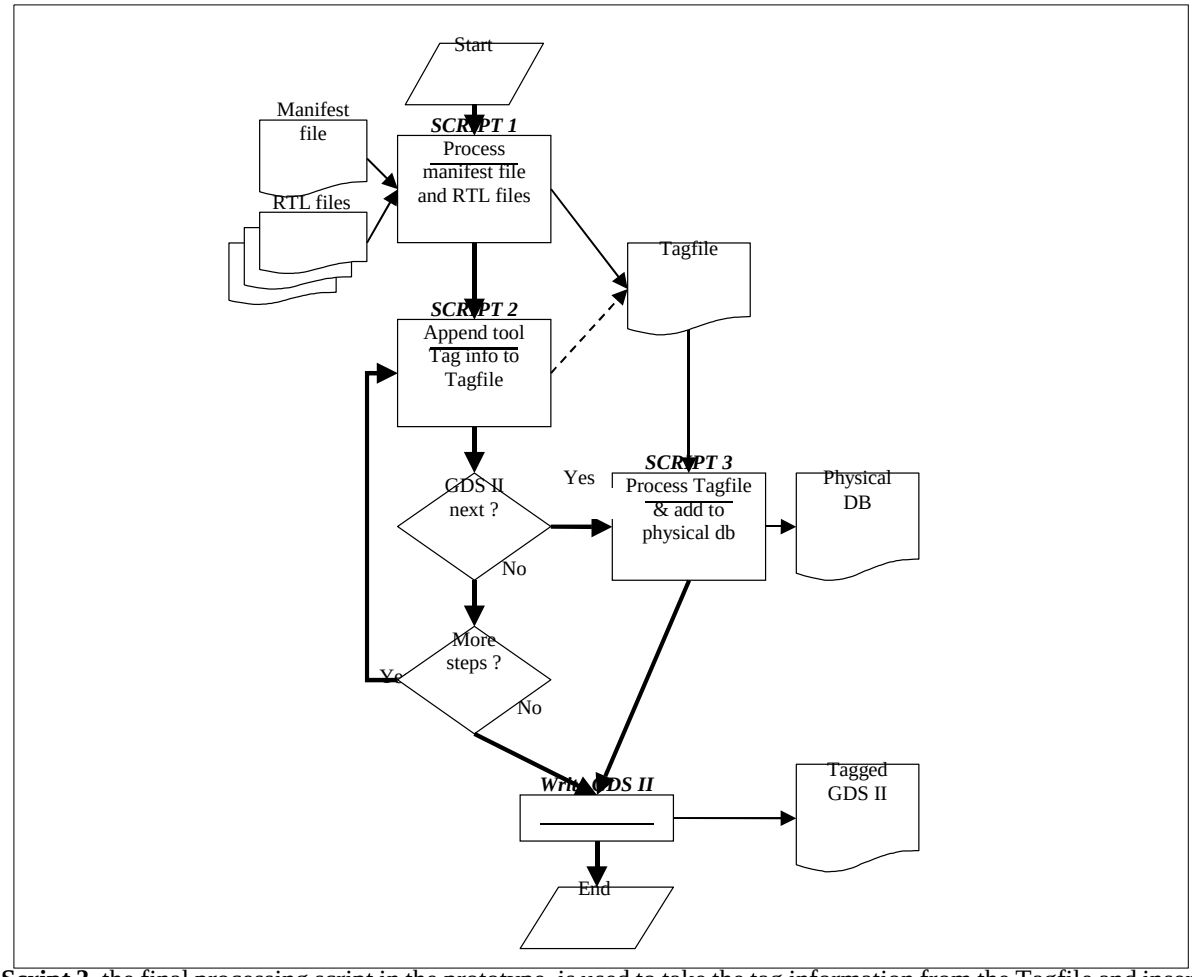
5.3.2 Flow Implementation

The flowchart provided in Figure 5, "Basic Tagging Prototype," gives an overview of how the basic tagging prototype can be implemented.

Script 1 in the flow is the initial tag processing script. It reads the previously discussed manifest file to determine which files should be processed in the search of Soft IP tagging information. It then parses through these files to find the tags and builds the initial Tagfile database. In the prototype, the level of hierarchy of the tags was not considered (see Section 5.3.3, "Known Limitations"), as can be seen in the simple Tagfile excerpt shown previously. This is a fairly simple file-parsing script, which can be implemented in whatever language is preferred by the developer. The script for the prototype was implemented using a combination of Perl and c-shell.

Script 2 is very closely coupled to whatever mechanisms are being used within the flow to control the tool and script launching. This level must have a knowledge of the tools and the versions being used, as well as a mechanism to parse through the tool's scripting language to extract script tags (not implemented in this prototype). It also must be at a point in the flow where access to all the data processing tasks is available. For the internal flow onto which the prototype development was layered, the mechanism was gmake. Makefiles are used to set up and control all the tools, their versions, the scripts, and the environment that the tools and scripts are launched from into an LSF system. For this simple implementation, the tool-launching target in the Makefile was modified to append an additional tool tag to the Tagfile on the initial run of that tool. This was sufficient to provide proof of the mechanics of the process.

Figure 5: Basic Tagging Prototype



Script 3, the final processing script in the prototype, is used to take the tag information from the Tagfile and insert it into the Milkyway database of Astro, the routing tool utilized in the prototype flow. This particular

script was implemented in Scheme. It is recommended to write it in the native scripting language of the tool utilized in the final database preparation steps prior to GDS II.

Finally, the GDS II is produced using the mechanisms provided within the tools.

The resultant GDS II can be scanned with simple modifications to existing Hard IP tag reporting scripts, due to similarities in syntax. A simple, internal-only, reporting script was used to report on the prototype flow tags.

The following example shows an excerpt of the resultant report.

Figure 6: Excerpt of Report

```
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:0  0
  STRING: "% Vendor Synopsys"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:0  0
  STRING: "% Product IPcore"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:0  0
  STRING: "% Version 1.0"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:0  0
  STRING: "% Metric 1234"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:1  1
  STRING: "% Vendor Synopsys"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:1  1
  STRING: "% Product Design_Compiler"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:1  1
  STRING: "% Version syn/2003.12"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:2  2
  STRING: "% Vendor Synopsys"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:2  2
  STRING: "% Product Jupiter"
TEXT:                   LAYER: 63  TEXTTYPE: 63  PRES: 0x8  STRAN: 0x0
  MAGNIFICATION: 0.002000  ANGLE: 0.000000  XY:2  2
  STRING: "% Version jupiter/2003.09-SP1"

. . .
```

5.3.3 Known Limitations

Due to constraints in time and complexity, and because the prototype was developed concurrently with the standard, several known limitations should be pointed out:

- The prototype did not handle levels of hierarchy. It was implemented in a simple “core hardening” type

of flow, where the top of the block of interest was the only level to contain a Soft IP tag. The scripts would need to be enhanced in order to track which block the tags are associated with. This would not be a difficult extension to make.

- No error checking was implemented on the syntax or the actual processing of the tags.
- The verboseness control was very crude, consisting simply of attaching a tool tag to the IP on the initial invocation of a given tool in the flow.
- Due to late changes in the standard, the actual prototype scripting no longer matches the standard completely. Thus, the scripting is not provided in its entirety, as this might lead to confusion.

5.4 Conclusions

This standard has presented a script-based prototype implementation of the Soft IP tagging standard with an existing design flow. It is believed that the prototype shows the feasibility of implementing the standard in a useful form through ad-hoc techniques, and that sufficient information has been provided to assist readers in performing their own implementation prior to direct tool support being available. The prototyping effort took a simple approach of maintaining tagging information in a separate, text-based file to keep it as generic as possible.

As a part of the standard development effort, the prototyping effort has provided key insights into potential implementation difficulties, and has thus improved the overall quality of the initial standard.

6. References and Acknowledgements

6.1 References

"GDSII Description Stream Format Standard." VSIA Contributed Technical Specification. Available from the VSIA Web site at www.vsi.org.

"VSI Alliance Virtual Component Identification Physical Tagging Standard (IPP 1 1.0, 2.0 & 3.0). Available from the VSIA Web site at www.vsi.org.

6.2 Acknowledgements

Special thanks to Carl Dobbs and Synopsys, Inc. for providing the resources and time to create the prototype system identified in Section 5, "Prototype Description."