SPI: An Open Interface Integrating Highly Interactive Electronic CAD Tools *

J. P. Schupp, J. Cockx †, L. Claesen, H. De Man ‡

IMEC, Interuniversity Micro Electronics Center,
VSDM division, Kapeldreef 75, 3030 Leuven, Belgium.
Phone 32-16-281201
Telefax 32-16-229400

Abstract

This paper describes the SPI Interface as an interface unifying the integration of electronic CAD tools. The goal of the interface is to provide a *direct communication* and *interactive feedback* between the primary design tools (schematics editors, symbolic layout editors, module generators etc.) and intelligent verification tools (electrical debugging, timing verification, simulation etc.).

The SPI Interface is not a framework but *complementary* to existing frameworks. It is the specification of a Structure Procedural Interface together with some utilities to standardise the *communication among* CAD Tools and to support their *integration*.

SPI offers a novel interface that supports, in an easy and efficient way, the integration of in-house as well as foreign CAD Tools. SPI will be made available and promoted in the European Electronic CAD Community.

1 Introduction

During the global design process of VLSI chips or VLSI modules, the verification of the correctness of the circuits takes a considerable amount of the design time. One bottleneck in the efficient application of this verification is the interactivity between the verification tools and the designer. The current design practice is that CAD tools are running *one at a time* and that *communication is via files*. This is extremely time consuming in a verification phase where the feedback between design definition and design verification is currently taking most of the designers time. Another disadvantage of the current CAD tools is that they often require different formats for representing design information, which necessitates the use of *cross-reference lists* and makes it harder for the designer to relate information from a verification tool to the original design information (such as schematics).

To allow for a much faster feedback between verification tools and the designer, the SPI Interface is developed. The overall methodology and design philosophy of the SPI concept is described in more detail in [Ram 87], [Cla 87] and [Sch 89].

The SPI Interface transfers primarly *structure* information. The structure is produced by primary design definition tools (called structure *producers*) even if that structure is not its main output (e.g. schematics editor, layout editor, module generation program). The structure is consumed by design verification tools (called structure *consumers*) to perform analysis on this data (e.g. verification tools, simulators). The structure in the SPI Interface is represented by *netlists*.

In section 2 of this paper the integration of Electronic CAD Tools will be discussed. The SPI Interface will be compared with other frameworks in general. Section 3 gives a feeling of what the objective of SPI is. In section 4 an introduction to the specification and the data model of the Structure Procedural Interface is given. Also an introduction to the SPI utilities will be given. Section 5 describes an existing integration.

*Research performed within the scope of the ESPRIT 1058 project: "Knowledge Based Design Assistant for Modular VLSI Design". Partners: IMEC Leuven Belgium, INESC Lisbon Portugal, Philips Eindhoven The Netherlands, Silvar-Lisco Leuven Belgium.

†Silvar-Lisco, Abdijstraat 34, B-3030 Leuven, Belgium, phone: +32-16-200016

‡Professor at K.U.Leuven

2 Integration of Electronic CAD Tools

In the area of CAD and Tool Integration in commercial applications and research of electronic design, a lot of effort is ongoing on DataBase Management Systems (data modelling, version-handling, maintenance, etc.) [Ram 87]. This could be considered to be *horizontal integration*, this is an *indirect interaction* among CAD Tools, see the horisontal double arrows in Figure 1. One can see that a tool only *interacts* with a user and a database. Interaction among tools must be performed along this database delaying the interaction with the designer.

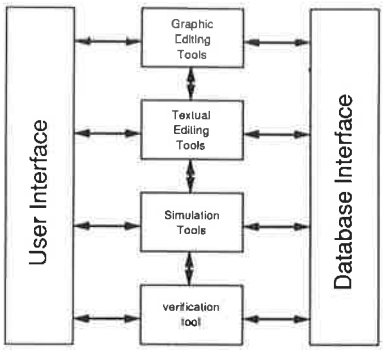

Figure 1: Horisontal and Vertical Integration of CAD Tools.

Instead of this horisontal integration the SPI Interface [Coc 89] concentrates on the *direct interaction* among CAD Tools. This could be pointed to as the *vertical integration*, see the vertical double arrows in Figure 1. Direct communication and interactive feedback is performed by the possibility of *highlighting*, *selection* and *backannotation*. The SPI Interface is not intended to be a DBMS to integrate horisontally, but to be complementary to existing frameworks and databases.

The SPI Interface poses very few constraints on existing CAD tools to integrate together. Each CAD tool can be developed independently and within its own environment (user interface and database) but the use of some uniform database (and DBMS) is desirable. For this reason of independency, together with the fact that different languages and different machines can be used, the SPI Interface is said to be *open*. Therefore making a coupling with SPI is not very complicated. The effort to couple (or to integrate) a structure consumer to SPI is about 1 week, and 3 weeks for a structure producer. It is important to notice that this integration work can be done *after* the design and implementation of a CAD tool.

The main aspects of the SPI Interface supporting the task of integration are:

- **standardisation:** The standard protocol provided by the Structure Procedural Interface allows that the processes can communicate with each other. It also makes possible the implementation

of *transparent* utilities which can be shared among all the CAD Tools.

- **direct communication and interactive feedback:** transfer of netlists, highlight, selection and backannotation, which provide a *fast design cycle* and a *tight integration.*
- **netlist merging:** Different structure producers, together defining a hierarchical design, produce 1 netlist to the structure consumer.
- **interprocess communication:** Allows CAD tools to run in different processes and environments and on different computers.

3 The objective of the SPI Interface

The objective of the SPI Interface is briefly illustrated in Figure 2.

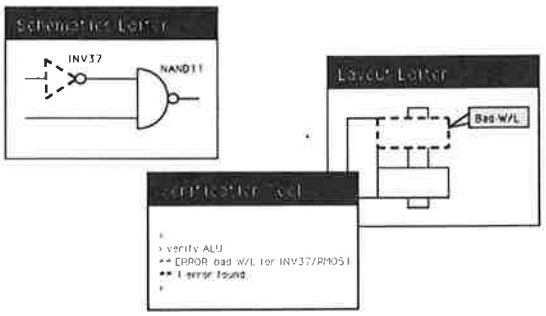

Figure 2: An example of tool integration with SPI.

A schematics editor contains a design called ALU with invertors and other gates. The invertor has been implemented as a set of rectangles in a layout editor. Both are visible in different windows, using different editors. A verification tool has been started in a third window without stopping the schematics and symbolic layout editor. It issues an error message for one of the transistors of the invertor in ALU and highlights this invertor in both editors (see dashed lines). The schematics editor is for example an in-house editor integrated in some database environment. At the other side, the layout editor is a foreign tool with its own database and user interface. With SPI it is possible to communicate interactively from the verification tool to these two editors. These two editors are called by a *transparent* utility (the netlistmerger) such that the two editors produce *one* netlist to the verification tool. If there was no *direct communication*, the two netlists should have been merged in some way together with the generation and use of cross-references. If there was no *interactive feedback*, the interpretation of the error should have been very painful because then one had to interpret it using the cross-references and the two different formats of netlists.

4 SPI: Structure Procedural Interface

The SPI Interface is a Structure Procedural Interface together with some utilities to standardise and to support:

- the *transfer* of structure *from* structure producers *to* structure consumers.
- the *transfer* of structure related information, called attributes, in a *bidirectional* way between structure producers and consumers using attribute- and backannotation operations.
- the *interactive communication* between structure producers and consumers using highlight and select operations.

Essentially the SPI Interface consists of two main concepts:

- The *(paper)* specification of the Structure Procedural Interface (including the data model). The specification is the definition of the standard protocol for transferring information between CAD tools.
- The utilities include hierarchy and bus expansion, interprocess communication, merging of netlists from different producers into one netlist, tracing, browser and file dumping and restoring; this is the only *software* of the SPI Interface.

In this section the SPI data model, specification and utilities will be discussed.

4.1 The SPI Data Model

The major foundation for CAD Tool Integration is *communication.* To facilitate successful communication a common *data model* is required. Together with this data model a *procedural interface* can be defined. The data on which the interface and the data model operate on, are *netlists.* Therefore *structure* producers and consumers are sometimes called *netlist* producers and consumers.

The SPI data model is illustrated in Figure 3. A netlist consists of cells. A cell can have ports, which define its external interface. A cell can contain instances of other cells and nets. When Cell NAND is instantiated in Cell FF, the ports of Cell NAND are also instantiated in Cell FF; these instantiated ports are called instance ports, see Figure 3. A net[1] connects zero or more instance ports and zero or more cell ports. The fact that an instance port is connected to a net is called an internal connection. The fact that a cell port is connected to a net is called an external connection.

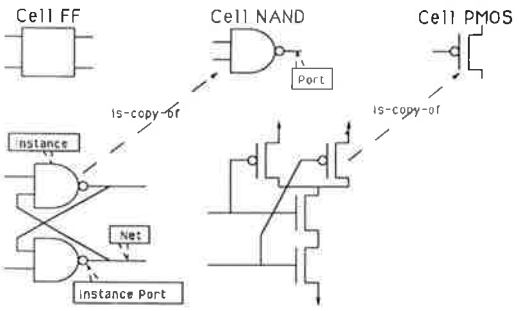

Figure 3: The SPI netlist data model: an example.

Cell, ports, nets, instances and instance ports can have attributes, that consist of names and values.

The terminology used to describe netlists here reflects a similar definition from EDIF [Edi 87]. The data model is compatible with the ECIP [Eci 88a][Eci 88b] data model. An ECIP-style conceptual diagram of the SPI data model is given in Figure 4. SPI uses the EDIF terminology, see the names in upper-case. All the other names and notations are the ECIP conventions.

4.2 The Specification of SPI

The specification of the Structure Procedural Interface [Coc 89] is written using C syntax and contains the following sorts of *procedures*:

- control: for setting up the datastructure, initialization of the structure producer(s)
- request structure information: to get structure information from the producer
- request structure related information (*attributes*): to get attributes from the producer

[1] One could also handle *busses* in the SPI Interface, but this will not be explained in this paper.

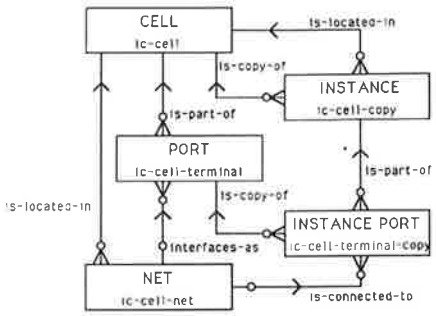

Figure 4: The SPI netlist conceptual model.

- backannotation of structure related information: to create or modify structure attributes in the structure producer from a structure consumer.
- highlight structure objects: to highlight objects in graphical representation (e.g. schematics editor) or in textual form (e.g. textual editor)
- request selection of a structure object by the user: selection of an object by the user by pointing to it (e.g. with a mouse), or by referencing it by name (e.g. type name at keyboard)
- ask for usernames of structure objects: to pass the username of an object.

In the specification, only *long integers* and *character strings* are used. Therefore each language which has these two types can be used to implement these specifications; CAD tools written in different languages can therefore be integrated together. Currently language bindings do exist for C, Pascal, Lisp and Fortran.

An example of a specification of a procedure for the request of structure information is

```
long SPIgetCell (name)
char *name;
```

parameters:	`char *name`	The name of a cell of which the netlist consumer would like to know the object handle.
returns:	The object handle of the cell with that name, or zero.	
usage:	The netlist producer returns the object handle of the cell with the requested name, or zero if it does not know such a cell.	

It is important to notice that the structure producers have to *implement* these procedures such that the implementation fulfil the specification. The structure consumers *can* only *call* this procedures. One can compare this with the concept of *procedural abstraction* where the structure producers are the implementors of the (abstraction by) specification. A key advantage of the abstraction by specification is that the implementation is irrelevant, and one can change to another implementation (read structure producer) without affecting any program (read structure consumer) that uses the abstraction [Lisko 86].

Remark: The (user)names of objects and the attributes (and values) are also standardised in the SPI Project [Sev 89][Coc 89].

4.3 The Utilities

The utilities are the only software of the SPI Interface and makes of the SPI Interface a good vehicle for the integration of CAD tools. Due to the standard protocol, defined by the specification, the utilities are implemented in such a way that they are *transparent* for the implementors of CAD tools. The most important utilities are

- **hierarchy and bus expander:** The hierarchy expander is a utility that provides a *flattened* circuit to a specific structure consumer from hierarchical structure producers using information obtained via SPI calls. The depth of expansion can be controlled. The bus expander is similar to the hierarchy expander. It is a utility that generates a circuit without busses (that is, all the busses are expanded) to a specific structure consumer.
- **netlist merger:** The netlists of different structure producers, together defining a hierarchical design, will be presented to the consumers as *one* netlist.
- **interprocess communication:** Due to this utility the CAD tools can run in different processes and on different machines in a transparent way.

Other utilities like **tracing** (to debug ones usage of the SPI Interface), **browser** (to browse through design hierarchy during highlight and select calls), **file dumping and restoring** are also part of the set of utilities, a set which can be extended arbitrarily.

5 Description of an existing integration

Figure 5 displays a real session of the SLOOP-II Timing Verification of a VLSI module taken from the CATHEDRAL-II module library using the SPI Interface. Three editors (a module generator, a layout and a schematics editor) are started in three different processes within their own environment. The three editors together defines a hierarchical design of a module (in this example an ALU). Then the timing verifier starts the verification by first selecting the ALU in the editor defining the toplevel of the ALU by using the browser-utility. After selection the whole netlist is read in; the hierarchy expander and the netlist merger are activated. The timing verifier performs a longest path analysis on this netlist and highlights the longest path in the three editors using the browser-utility. In the last step a timing view of the ALU is generated and some hierarchical information is backannotated to the original definition of the ALU.

The Esprit-1058 Project, which partly concentrated on the development of the SPI Interface, integrates together 5 structure producers and 7 structure consumers.

6 Conclusion

In this paper, the SPI Interface for interactive CAD tool integration has been presented. The SPI Interface is in no sense a framework, design management system or a DBMS, but can complement existing frameworks in that it provides a *direct communication* between structure producers and structure consumers in an *interactive* way. In this way the design cycle is shortened and verification tools can communicate more closely with the designer.

The reason for the development of the SPI Interface is to unify the integration of CAD tools. Therefore, to promote the use of the standard protocol, efforts will be made in order to make SPI and its utilities available to the European Electronic CAD Community. More information is available from the authors upon request.

Acknowledgements

The work described in this paper has been performed in the scope of the ESPRIT Project 1058 with the following participants: IMEC (Belgium, Prime Contractor), INESC (Portugal), Philips (The Netherlands) and Silvar-Lisco (Belgium). Many people have contributed to the success of the project, and we thank in particular I. Bolsens, P. Das, A. Demaree, W. De Rammelaere, P. De Worm, P. Johannes, T. Kostelijk, P. Lauwers, B. Lynch, L. Marent, G. Mole, H. Neto, P. Odent, P. Petroni, J. Raposo, Ph. Reynaert, G. Schrooten, R. Severyns, P. Six, E. Vanden Meersch, E. Willems, for many discussions and for their continuing efforts in the integration with SPI.

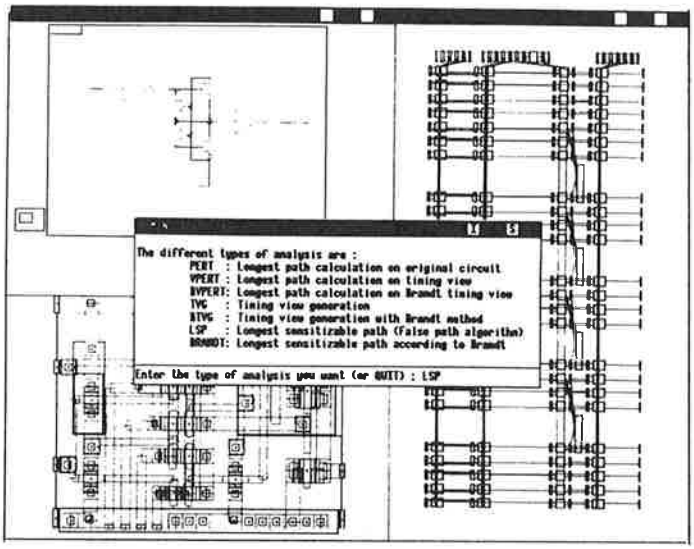

Figure 5: A Timing Verification session.

References

[Cla 87] L. Claesen, Ph. Reynaert, G. Schrooten, J. Cockx, I. Bolsens, H. De Man, R. Severeyns, P. Six, E. Vanden Meersch, *Open System Architecture of an Interactive CAD Environment for Parameterized VLSI Modules*, Proceedings of the 4th Annual ESPRIT Conference, Brussels, sept. 28-29, 1987, pp. 251-270.

[Coc 89] Cockx, J., *SPI version 2.3 Revision 2302*, Silvar-Lisco, Belgium, 6 February 1989.

[Eci 88a] The ECIP Conceptual Modelling Working Group, *ECIP Conceptual Model of Electronic Products*, November 7th, 1988.

[Eci 88b] Chalmers, D., Meys, F., *Successful CAD Integration Needs A Standard Conceptual Model*, Proceedings of the 5th Annual ESPRIT Conference, Brussels, November 14-17, 1988, pp. 170-185.

[Edi 87] Electronic Industries Association, *EDIF - Electronic Design Interchange Format Version 2 0 0*, Washington D.C., May 1987.

[Lisko 86] Liskov, B., Guttag, J., *Abstraction and Specification in Program Development*, The MIT Press, Cambridge, 1986.

[Ram 87] Rammig, F., J., *Tool Integration and Design Environments*, Proceedings of the IFIP WG 10.2 Workshop, Paderborn, FRG, November 26-27, 1987.

[Sch 89] Schupp, J., P., Cockx, J., Claesen, L., De Man, H., *SPI: A Practical and Open Interface for Electronic CAD Tool Integration*, Proceedings of the 6th Annual ESPRIT Conference, Brussels, November 27 - December 1, 1989.

[Sev 89] Severyns, R., *Naming Conventions in SPI and in Tools Communicating with SPI*, Imec, Belgium, April 4th, 1989.