Project No. 1058

SPI: A Practical and Open Interface for Electronic CAD Tool Integration *

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

IMEC Lab.[§], *Leuven, Belgium, Phone: +32-16-281203*

Abstract

This paper describes the SPI Interface as a practical and open interface to integrate 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 the specification of a Structure Procedural Interface together with some utilities to standardise the *communication among* CAD Tools and to support their *integration*.

The data model of SPI is compatible with the ECIP [Eci 88a][Eci 88b] data model. The EDIF [Edi 87] terminology is used. Bindings do exist for C, Pascal, Lisp and Fortran, implementations do exist on a variety of Unix workstations.

The utilities are software *tools* and *libraries*. The software *tools* are a file dumping and a file restoring program. The software *libraries* include hierarchy and bus expansion, a browser, interprocess communication, merging of netlists from different producers into one netlist and tracing.

Because SPI offers an 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

*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

[§]Interuniversity Microelectronics Center

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 and cross-reference lists*. 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 information.

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] and [Cla 87].

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 horizontal 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.

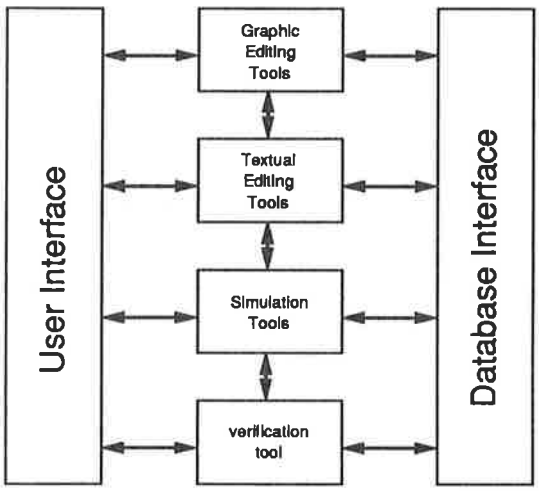

Figure 1: Horizontal and Vertical Integration of CAD Tools.

Instead of this horizontal 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 to *highlight*, *select* and *back-annotate* (see subsection 3.2). The SPI Interface is not intended to be a DBMS to integrate horizontally, but to be complementary to existing frameworks and databases. Each CAD tool may have its own database but the use of some uniform database (and DBMS) is desirable.

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

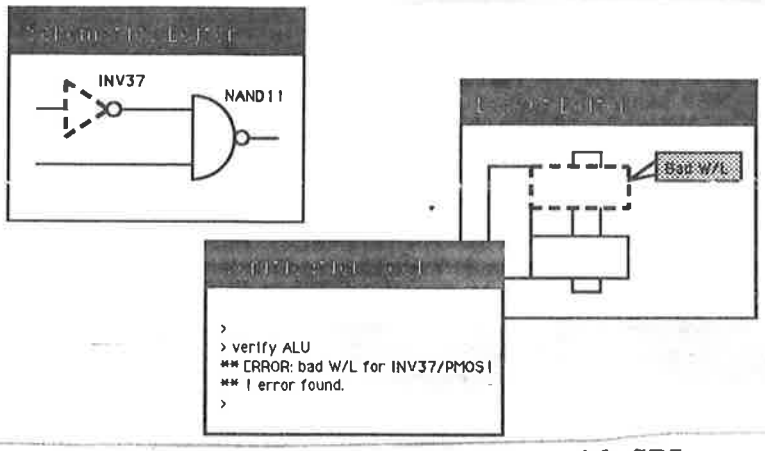

Figure 2: An example of tool integration with SPI.

third window. 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. With SPI it is possible to communicate interactively from the verification tool to these two editors. These two editors can communicate with each other such that they 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 netlists.

In the section 2 of this paper the Global System Architecture of the SPI Interface will be discussed. Section 3 is an introduction to the specification of the Structure Procedural Interface. In section 4 the integration of electronic CAD tools with the SPI Interface will be shown with an example and within the scope of the E-1058 project.

2 Global System Architecture

Figure 3 gives the major information flow (communication) in the architecture of the SPI Interface. The notational conventions used are indicated in the figure.

An implementation of the system can be done over one or more processes running on one or more machines. The functionality of the individual modules and the interfaces, depicted in Figure 3, will be explained in the following subsections.

2.1 UTM: Unix Tool Manager

The Unix Tool Manager is the main process controlling the global CAD activity. It is a very simple manager. It is no more than the possibility to execute additional commands under a Unix-shell. One can control the structure producers: e.g. to reset a process from a waiting state to an active state. Also from the shell one can

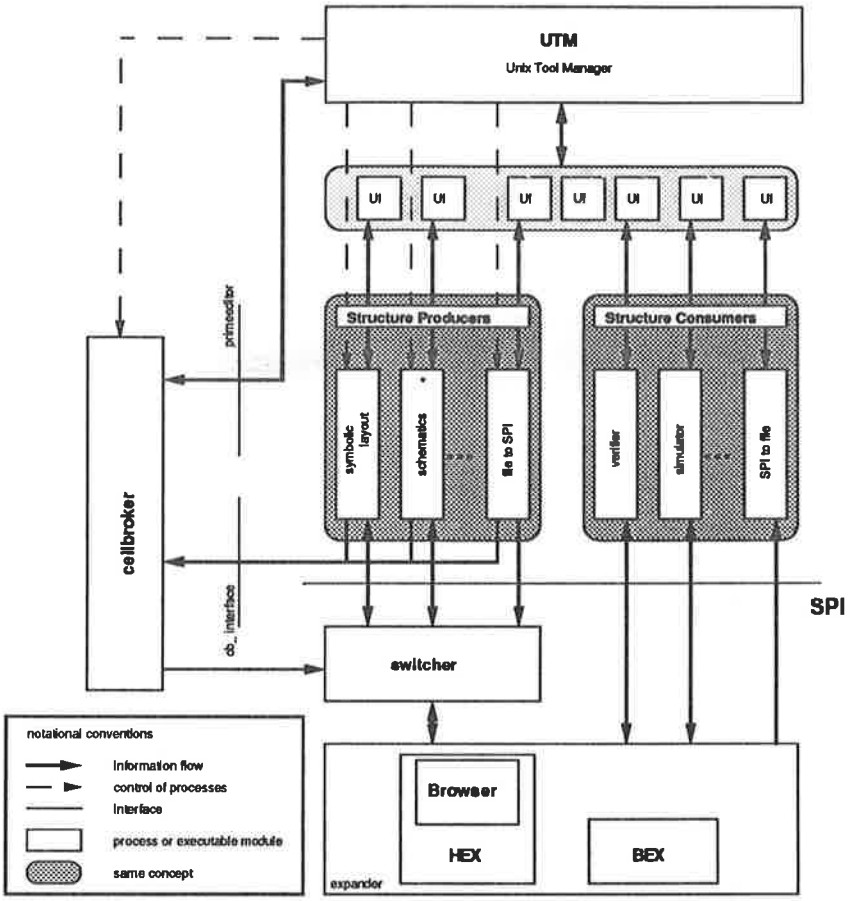

Figure 3: Global System Architecture.

start or delete the cellbroker or execute cellbroker commands. All the structure producers and consumers are started or ended in their own environment.

2.2 Application Programs

The application programs are the programs (the CAD Tools) which are to be integrated. The SPI Interface is built on the distinction between structure producers and structure consumers. A program can belong to the two classes. An example is a preprocessor program that takes structure and transforms it into another structure. With the SPI Interface the producers can run in *parallel*. The consumers must run *sequentially* unless a synchronising protocol is constructed among the consumers.

2.2.1 Structure Producers

The structure producer programs consist of application programs that have a direct graphical interaction with the user. These programs are for the most part editor-like and communicate in a graphical way (e.g. Schematics Editors). Sometimes a structure producer can be a textual editor with textual interaction (e.g. Struc-

ture Description Language Editors). The structure producer programs generate structural data that a structure consumer can read in via SPI calls.

2.2.2 Structure Consumers

The structure consumer programs are the programs that use structural data to perform new transformations on this data; for example a (timing) verification tool.

2.3 HEX: Hierarchy 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. With the use of the cellattribute *expansion_level* the depth of expansion can be controlled.

2.4 BEX: Bus Expander

The bus expander is similar to the HEX module. It is a utility that generates a circuit without busses (that is, all the busses are expanded) to a specific structure consumer.

2.5 Cellbroker

The cellbroker is a simple database that maintains and controls information about all cells defined in the different producer(s).

2.6 Switcher

The switcher can determine from the cellbroker where a certain cell is located and create a link from the consumer to the correct producer. It will also match and merge cells, instances and ports based on their (user)names.

2.7 Browser

This module is a software tool to browse through the design hierarchy during highlight and select actions.

2.8 UI: User Interface

Each CAD Tool may define its own user interface.

2.9 The Primeeditor Interface

With the so-called primeeditor interface a user can tell the cellbroker that a certain cell must be taken from a certain producer; e.g. when two producers define the same cell, a user must be able to indicate the producer that contains the *primary*

data[1]. To allow the user to specify this, the *cell* command has been implemented. The *cell* command allows the user to set and list the prime producer for one or more cells.

2.10 The Cb_ Interface

With this interface one can tell the cellbroker to add or delete a structure producer or to add or delete a cell.

3 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 Utilities including hierarchy and bus expansion, interprocess communication, merging of netlists from different producers into one netlist, tracing, browser[2] and file dumping and restoring; this is the only *software* of the SPI Interface. The utilities are designed and implemented such that they are *transparant* for the structure producers and consumers.

The SPI data model and the specification of the Structure Procedural Interface will be described in this section.

3.1 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 4. A netlist consists of cells. A cell can have ports, which define its external interface. A cell can contain instances of

[1]The *primary data* corresponds to the information that the designer wants to enter as the specification of the design.

[2]The browser is under development.

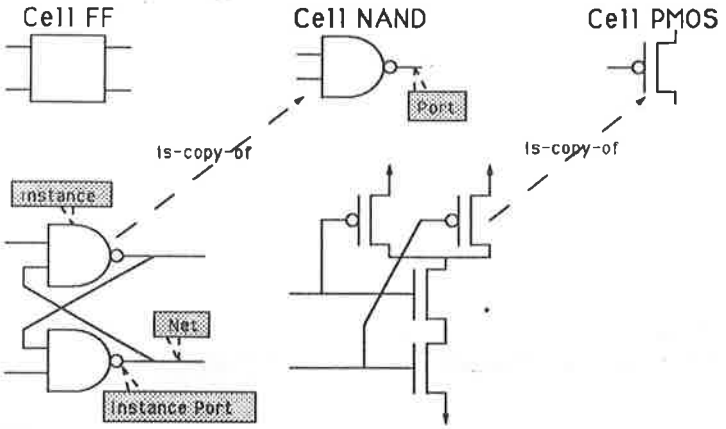

Figure 4: The SPI netlist data model: an example.

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 4. A net[3] 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.

Cell, ports, nets, instances and instance ports can have attributes, which 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 5[4]. Note that the terminology used in EDIF and ECIP differs. SPI uses

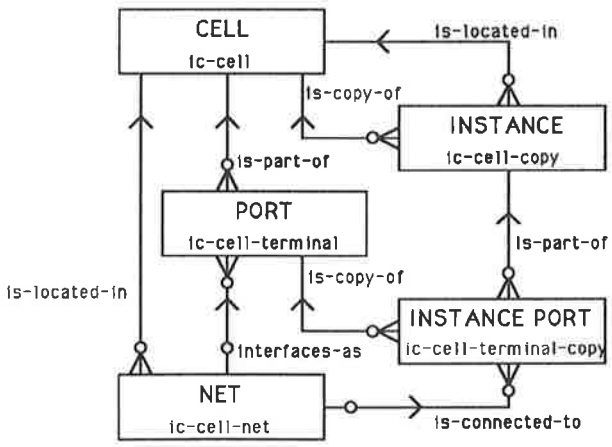

Figure 5: The SPI netlist conceptual model.

[3]One could also handle *busses* in the SPI Interface, but this will not be explained in this paper.
[4]The upper-case notations are SPI notations, the lower case ECIP.

the EDIF terminology, see the names between brackets. All the other names and notations are the ECIP conventions.

3.2 The Specification of the Structure Procedural Interface

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
- 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 (e.g. C, Lisp, Pascal, Fortran) can implement these specifications; CAD tools written in different languages can therefore be integrated together.

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.

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

4 Integration with SPI

The SPI Interface is supporting the task of integration. Figure 6 gives an example. The main aspects of SPI Interface helping the task of integration are

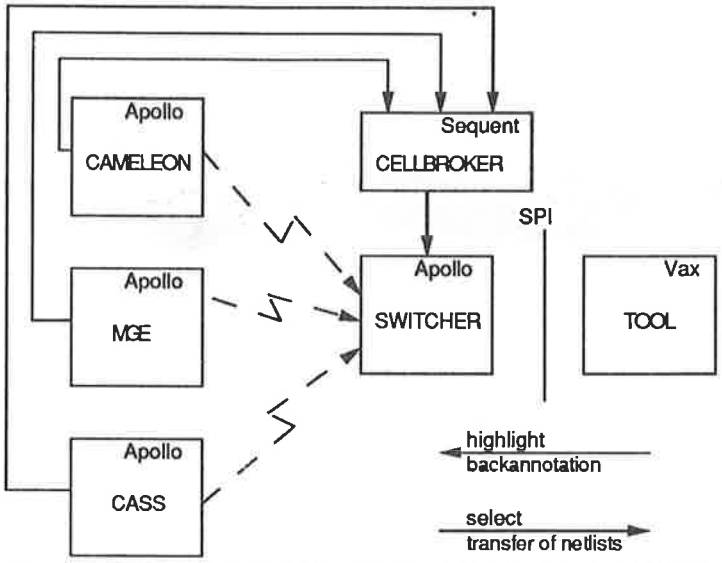

Figure 6: A possible session on different machines.

- **standardisation:** The standard procedural interface allows that the processes understand (can communicate with) each other. It also makes possible the implementation of utilities which can be shared among all the CAD Tools.
- **interprocess communication:** Allows editors to run on different Apollo workstations, cellbroker on a Sequent-machine, a verification tool on a VAX-Ultrix-machine.
- **netlistmerging:** The three editors, together defining a hierarchical design, produce 1 netlist to the consumer.
- **direct communication** and **interactive feedback:** transfer of netlists, highlight, selection and backannotation.

4.1 What do CAD Tools need to do to integrate with SPI?

Each CAD Tool can be developed independently and within its own environment (user interface and database). For this reason, 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. What a structure producer and consumer have to do is described in the next 2 subsections. It is important to notice that this integration work can be done *after* the design and implementation of a CAD Tool.

4.1.1 The structure producers

A producer *has*

- to implement all the SPI Procedures as specified in [Coc 89]
- to tell the cellbroker (by calling cb_ interface routines) which cells it has in its own database.

The communication and usage of the SPI Utility (interprocess communication) is established by simple *linking* with the utility library.

4.1.2 The structure consumers

A consumer *can*

- call SPI Procedures.

The communication and usage of the SPI Utilities (interprocess communication, netlistmerging, hierarchy or bus expander, browser, tracing) are established by simple *linking* with the utility libraries.

Remark: The file dumping and restoring utilities are stand alone utilities which need not to be linked with the CAD Tools. Utility libraries only have to be *linked* because they are *transparant* software utilities (due to the standardisation).

4.2 How does the result of the integration of CAD Tools appear?

Assume that a design of the `ALU` cell has been made in MGE (a Module Generation Environment) using instances of the cells `GFB & CARRY BYPASS`. These cells are designed in both Cass (a Schematics Editor) and Cameleon (a Symbolic Layout and Compaction System). Suppose that the designer wants to do interactive timing verification using the cells as viewed in Figure 7.

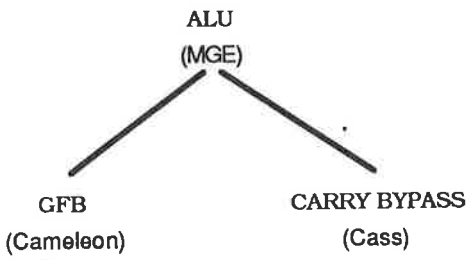

Figure 7: A view of the design of an `ALU`.

Description of a (artificial) session:

1. Take Figure 6 as the working environment. The user starts the 3 editors on 3 different Apollo workstations and the cellbroker on the Sequent-machine. The editors will tell the cellbroker which cells they have. In this case the cellbroker will known that MGE contains an `ALU`, Cameleon *and* Cass a `GFB` and a `CARRY BYPASS`.

2. To use only the cells as viewed in Figure 7 the user tells the cellbroker which cells are the primary cells. Here, the user indicate that the `GFB` cell has its primary data in Cameleon and the `CARRY BYPASS` cell in Cass.

3. Now, the user can start the timing verification tool on the VAX.

4. The tool wants to know which cell has to be verified. Therefore the select procedure is called. With the browser the selection is possible in all the editors. In this case the selection will be done in MGE.

5. After selection of the `ALU` in MGE the tool reads in the netlist. Because the timing verifier must have a flattened netlist without busses, it will use the hierarchy and bus expander. With the knowledge of the cellbroker, the switcher, in co-operation with the expander, can merge the netlist of the 3 cells in the 3 editors (`ALU`, `GFB`, `CARRY BYPASS`) into *one* flattened netlist.

6. The tool computes the longest path in the `ALU` and highlights this path with the highlight procedures. With the browser the user can browse through all the editors and cells to highlight the hierarchical path of his choice.

7. Then the tool and/or user will change the dimension attributes of one or more transistors in the longest path to speed up this path. The new attribute values will be backannotated in the right editor and cell by the use of the expander, the switcher and the knowledge of the cellbroker.

8. Then the user might restart the iteration (in point 5.) to reach a global optimum.

4.3 Integration in the Esprit-1058 Project

In the E-1058 the integration of CAD Tools has been achieved with the SPI Interface. Figure 8 shows the structure producers (above the SPI-line) and consumers (down the SPI-line) of the E-1058 project that have been integrated.

- The structure producers are
  - **Cass**, a Schematics Editor
  - **MGE**, a Module Generation Environment
  - **Cameleon-1**, a Symbolic Layout and Compaction System
  - **Hilarics-2**, a Structure Description Language
  - **SPICEtoSPI**, a SPICE Netlist to SPI Netlist Translator
- The structure consumers are

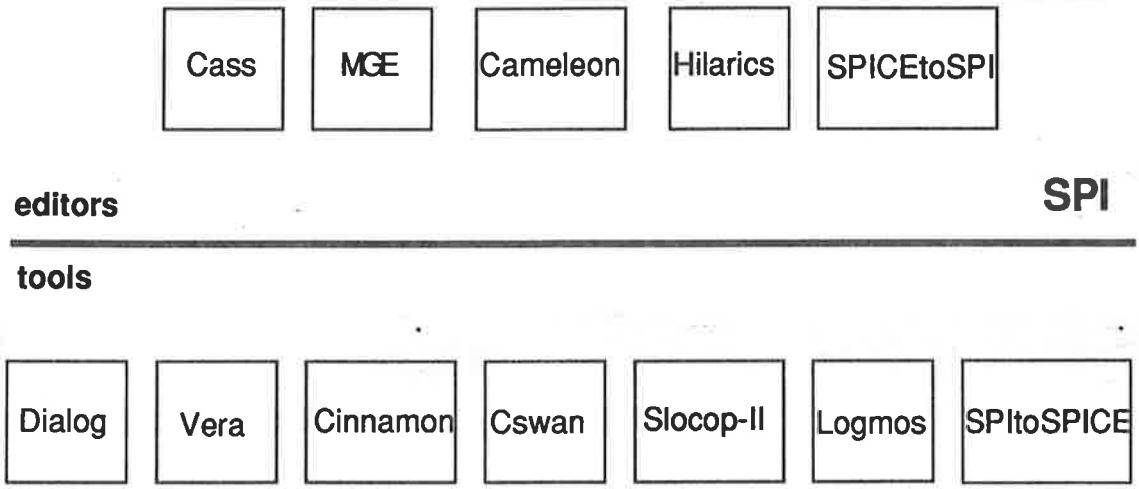

Figure 8: A very simple overview of the integration of the CAD-tools in E-1085.

- **Dialog**, a Knowledge Based Verification System
- **Vera**, a Knowledge Based Verification System
- **Cinnamon**, a Circuit Simulator
- **Cswan**, a (Parallel) Circuit Simulator
- **Slocop-II**, a New Timing Verification Tool
- **Logmos**, a Registor Transfer Simulator
- **SPItoSPICE**, a SPICE Netlist producer for SPICE-simulations

For the above CAD Tools the effort to couple a structure consumer to SPI is about 1 week, and 3 weeks for a structure producer. The implementors of these tools are satisfied with the *simple* and *easy* SPI Interface; *simple* and *easy* because SPI only handles structures (netlists). Therefore the SPI Interface is a *practical interface.*

5 Conclusion

In this paper, a *practical* and *open* Interface for interactive CAD tool integration has been presented. In addition to facilities provided by design management systems and DBMS, SPI provides a *direct communication* between structure producers and structure consumers. In this way the design cycle is shortened and verification tools can communicate more closely with the designer. An example is a timing verifier that can directly indicate the longest path in the schematics or symbolic layout by immediate highlighting without stopping either the timing verifier or the schematics or symbolic layout editor.

The SPI Interface is *open* because as few as possible constraints have been put on the tools themselves to be able to integrate already existing tools. Once a CAD Tool is integrated, a lot of other CAD Tools becomes available. It is also *practical* due

to its *simplicity* and *ease of use* and because its scope is limited to communication of netlist structures only.

The reason for the development of the SPI Interface is to standardise the integration of CAD tools. Therefore, to promote the use of the standard, efforts will be made in order to make SPI and its utilities available to the European Electronic CAD Community.

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, K. Croes, 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, L. Rijnders, G. Schrooten, R. Severyns, P. Six, E. Vanden Meersch, E. Willems, for many discussions and for their continuing efforts in the integration with SPI.

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.

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

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