SPI:
A Procedural Interface for Electronic CAD Tool Integration*

P. De Worm, R. Severyns, L. Marent, A. Demarée,
J. Cockx†, Ph. Reynaert‡, L. Claesen§, H. De Man¶, P. Six

IMEC, Kapeldreef 75, B-3001 Leuven, Belgium

Abstract

This paper describes the SPI Structure Procedural Interface to integrate electronic CAD tools. SPI is intended to be a standard interface for the exchange of netlist information between CAD tools. SPI uses interprocess communication and provides a set of utilities for expanding circuit descriptions (hierarchy and busses) and making use of multiple editors. The interface fits within the global philosophy for an open system architecture and therefore considerably simplifies tool integration besides making it more effective. The SPI Interface is not a framework, it has no own database, no user interface nor a design management system, but it is *complementary* to a framework.

1 Introduction

The **S**tructure **P**rocedural **I**nterface **SPI** transfers *structural* information. The structure (hierachical interconnection of components) is produced by primary design definition tools, called structure *producers* (e.g., schematics editors, structure description languages, ...) and is consumed by design verification tools, called structure *consumers* (e.g., verification tools, simulators, ...) to perform analysis on this data.
The basis of the Structure Procedural Interface **SPI** [Cock89,Cock90,Schu90] has been developed in the ESPRIT 1058 project (partners: IMEC, Philips, EDC (formerly Silvar-Lisco), INESC) and has been adopted successfully in other projects such as the CAD Framework Initiative (CFI), Design Representation Procedural Interface working group.

*This research is sponsored by the JCF ESPRIT-5082 project of the EC.
†EDC, Abdijstraat 34, B-3001 Leuven
‡EDC, Abdijstraat 34, B-3001 Leuven
§Professor at K.U. Leuven
¶Professor at K.U. Leuven

The SPI concept has been demonstrated in the CFI Integration Project at the 1990 Design Automation Conference [CFI90a,CFI90b].
Further research is being performed in the Jessi-Common-Framework (JCF) project (ESPRIT 5082). SPI is the main integration mechanism for electronic design CAD tools, and is also used in other ESPRIT projects (SPRITE, JESSI, ...).
In section 2 of this paper the key characteristics of the SPI interface are indicated. Section 3 gives an introduction to the specification and the data model of the Structure Procedural Interface. Also the SPI utilities will be described. In section 4 recent enhancements and related interface work are explained. Section 5 describes some of the actual shortcomings. Finaly, ongoing work and conclusions are mentioned in section 6 and 7.

2 Key Characteristics

SPI is a standard interface, with the following key characteristics:

- Direct communication

  One of the main disadvantages of many current CAD tools is that communication is via files. The representation of design information requires often different formats and the need of cross-reference lists are necessary. 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.

  SPI avoids this problem using *direct communication* between the CAD tools. Figure 1 shows a typical design cycle without and with using SPI.

- Interactive feedback

  SPI provides a direct *interactive feedback*, with mechanisms for highlighting, selecting and backannotating of the structural information, which results in a fast design cycle and a tight integration.

- Environment independent

  SPI is *environment independent*; it runs on every machine that supports UNIX and TCP/IP. This allows CAD tools to run on different machines and in their original environments.

- Language independent

  In the specification of the SPI communication protocol, only *long integers* and *character strings* are used. This makes SPI *language independent*. Any language with these two data types can be used to implement the specifications (C, Pascal, Lisp, Fortran). CAD tools written in different languages can therefore be integrated together.

- Transparent

  A structure consumer does not need to know whether the netlist has been expanded, merged, or comes from another process.

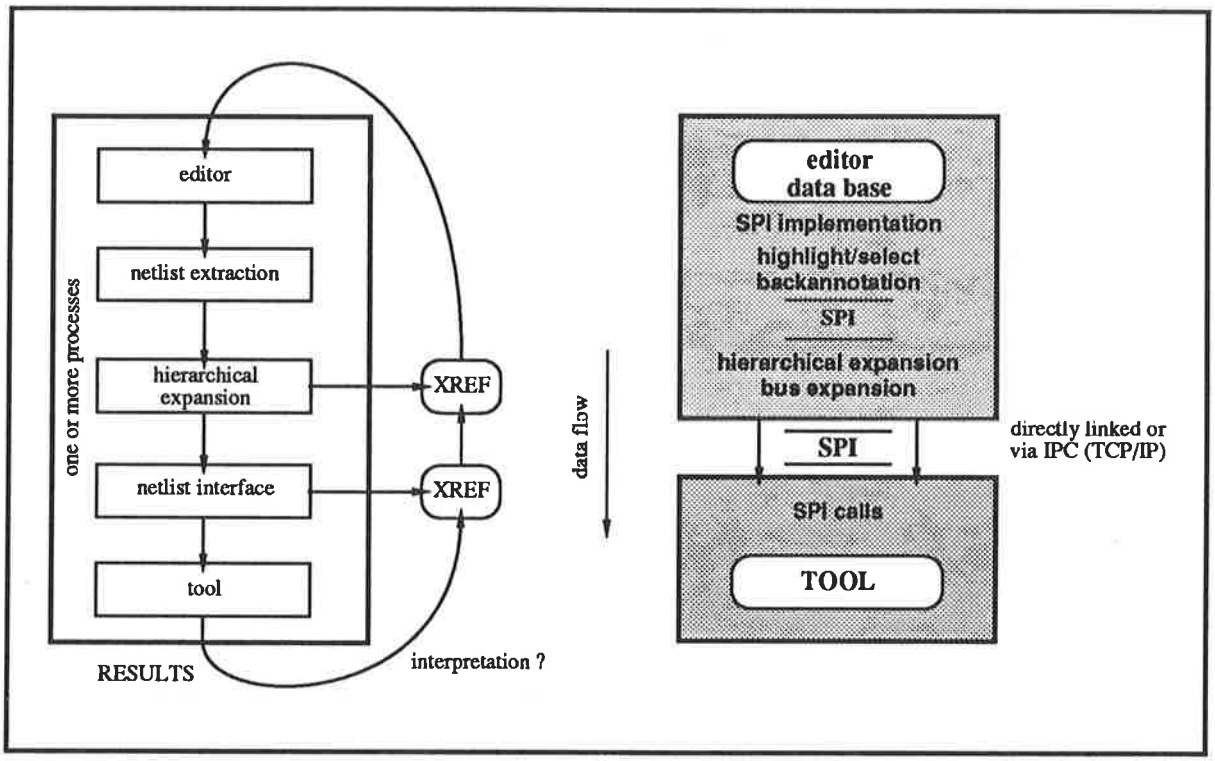

Figure 1: A typical design cycle without and with SPI.

Because of these characteristics, SPI poses few constraints on CAD tools. SPI supports the integration of in-house as well as third party design tools provided that the source code is available. Therefore, SPI can easily be integrated in existing CAD environments. The structure producers have to implement the specified procedures, while the structure consumers have to call these procedures. The effort to couple a structure consumer to SPI is about 1 week, and 3 weeks for a structure producer. The difference in time is because an editor has to implement all SPI functions, while a consumer has only to call the SPI routines. Once the SPI Interface is integrated in a CAD tool, a lot of other CAD tools which implemented SPI become accessible. Therefore, the SPI interface is said to be **open**.

3 Structure Procedural Interface

The specification of a **S**tructure **P**rocedural **I**nterface (including the data model) together with software utilities provide a standard way for *communication* of netlist structure among CAD tools and to support their *integration*.

3.1 Data Model

Communication between CAD tools requires a common *data model*. Together with this data model a *procedural interface* can be defined. The data on which the interface operates are *netlists*. The SPI data model is illustrated in figure 2.
The SPI data model is compatible with the ECIP data model [Ecip88a,Ecip88b]. The terminology used to describe netlists has been borrowed from the EDIF standard [Edif87]. A *cell* is a building block representing a part of the circuit. A cell communicates with the outside world through its *ports*. A cell can either be a *leaf cell* or be composed of *instances* of other cells. The ports of these instances and of the cell itself can be connected. A set of connected ports is called a *net*. Connections of instance ports are called *internal* connections. Connections of ports of the cell itself with instance ports are called *external* connections. Nets and ports can have a width and thus become busses (cables), respectivily bus ports.

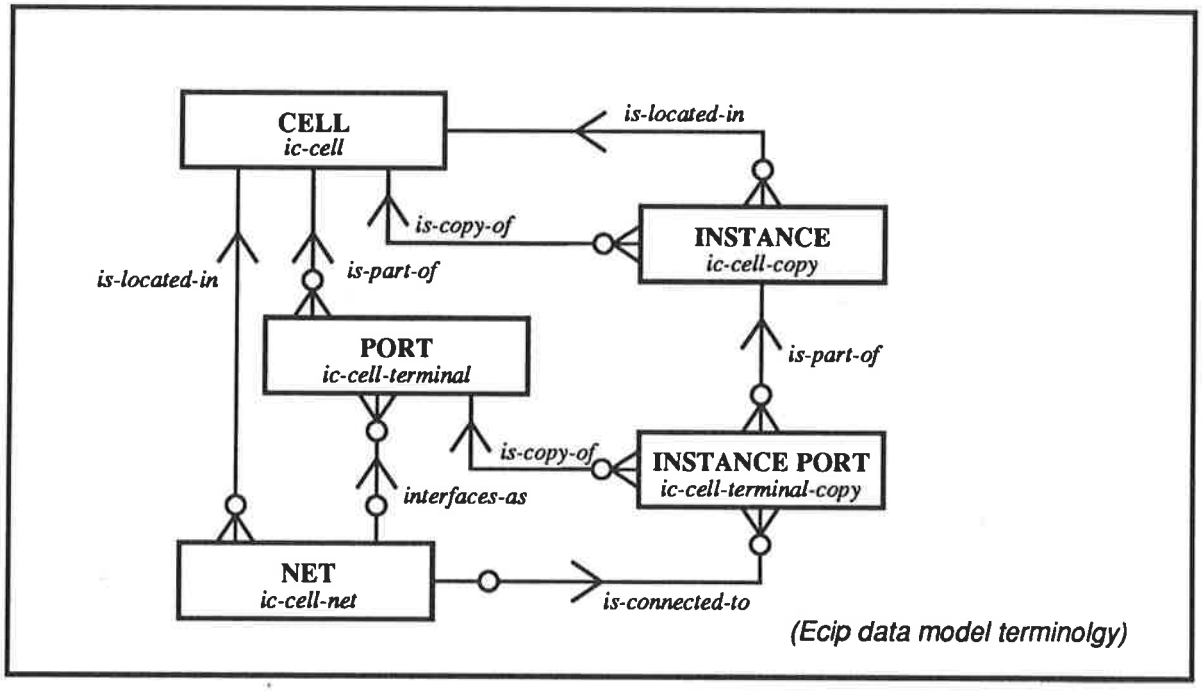

Figure 2: The SPI netlist data model.

The data model contains five objects (cells, instances, nets, ports and instance ports), and is known as the "five box model".

3.2 The Specification of the Structure Procedural Interface

SPI references to all structure objects by non-zero *long integer* numbers instead of *names*. The names of the objects are only useful for the user. The specification of the Structure Procedural Interface is written using C syntax and contains *procedures*

- to control: for setting up the data structure and initialization of the structure producer(s);
- to request structure information: to get structure information from the producer;
- to request structure related information (*attributes*): to get attributes about objects from the producer;
- for backannotation of structure related information: to add or modify structure attributes in the structure producer from a structure consumer;
- to highlight structure objects: to highlight objects in graphical representation (e.g., schematics editor) or in textual form (e.g., textual editor);
- to request selection of a structure object: selection of an object by the user by pointing to it, or by referencing it by name;
- to ask for usernames of structure objects: to pass the username of an object.

Attributes are additional information associated with one of the five objects of the data model. Attributes have a *name* and a *value*. The name is represented as a character string. The value can be a character string, or an integer or floating point number.

3.3 Utilities

The SPI utilities are software modules that call and/or implement the SPI functions and that can be linked with an SPI editor and/or tool, and provide additional features.
The utilities include hierarchy and bus expansion, interprocess communication, merging of netlists from different producers into one netlist, a database that maintains information about all cells defined in the different producers and a browser for highlighting and selecting in hierarchical designs.

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.

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) from a netlist producer supporting busses to a structure consumer that only accepts simple nets.

Cellbroker
The cellbroker is a simple database that maintains and controls information about all cells defined in the different producer(s). It can also be used to select the editor for each cell if multiple versions of these cells are available in different editors. The cellbroker is a server process accessible via remote procedure calls. See figure 3.

Switcher
The switcher can determine from the cellbroker in which editor a certain cell is located and create an IPC link from the consumer to the correct producer. Figure 3 shows that it is possible for a tool to communicate with different editors via the switcher.

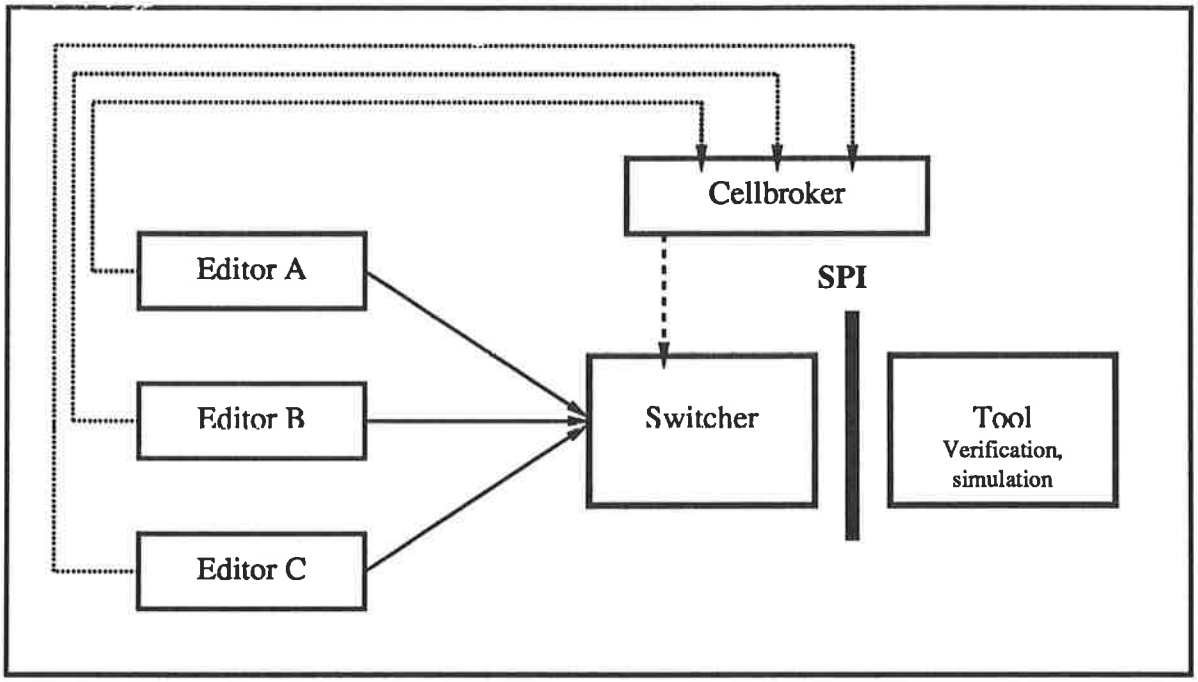

Figure 3: Using multiple editors with the SPI Cellbroker and Switcher.

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

4 Recent enhancements

A number of extension and enhancements were added to the current version of SPI.

4.1 Naming conventions in SPI

CAD applications exchanging structural information must also be able to exchange names of design entities (cells, instances, nets, ports, attributes), mainly for backannotation to the user. However, not all tools use the same naming scheme; some tools use very simple names, while other tools allows almost any character in names. Problems arise when some editor generate names that can not be accepted by other editors or tools. Therefore, a filter is needed to convert complicated names back and forth to simple names. A special

utility ("spinacon": SPI Name Conversion) [Seve89] is developped to decode and encode complex names.

4.2 Controlling Netlist Expansion

Until now, it was not possible for a tool to decide at run-time what kind of netlist expansion (HEX, BEX, HEX & BEX) should be applied by the SPI system. The tool developer had to decide at the linking step of the tool which SPI library to link. Three libraries are available, one that does no expansion, one that does the hierarchical expansion and one that does bus expansion in addition to hierarchy expansion.
A new function has been provided to control the netlist expansion:

void SPIcontrol(mode)

The parameter *mode* can have one of the following values:

- *SPI_NOEX* : perform no hierarchy expansion;
- *SPI_HEX* : perform hierarchy expansion;
- *SPI_BEX* : perform bus expansion;
- *SPI_HEXBEX* : perform bus expansion in addition to hierarchy expansion.

The default mode, when the function *SPIcontrol()* is not used, is *SPI_NOEX*, i.e., no hierarchy expansion. This routine serves as a switch that guides all subsequent SPI-function calls to their respective 'no-expansion'-, 'hex'-, 'bex'- or 'hexbex'- implementations.
The result is a new library that contains all the possible expansion methods. The implementation of the run-time controllable expansion is realised in such a way that it is fully compatible with the existing SPI definitions. This means that the programmer does not have to change anything to existing source code if he/she does not want to use this new feature. It is now possible for a tool to decide at run-time what kind of netlist expansion must be applied. This is especially useful when selective expansion is needed [Dema90].

4.3 Exchange Layout information

The need to enable tools to obtain layout data and supply layout results in a manner that it will be independent of the CAD system in which they will be used resulted in the definition of a Layout Procedural Interface (LPI). The interface contains routines to transfer *technology data* and *layout information*, a set of *selection* routines and *operation* routines. This interface is general enough so that many layout tools can use it [Rijn89, Mar90a]. Figure 4 shows a design cycle with interprocess communication using SPI and LPI.

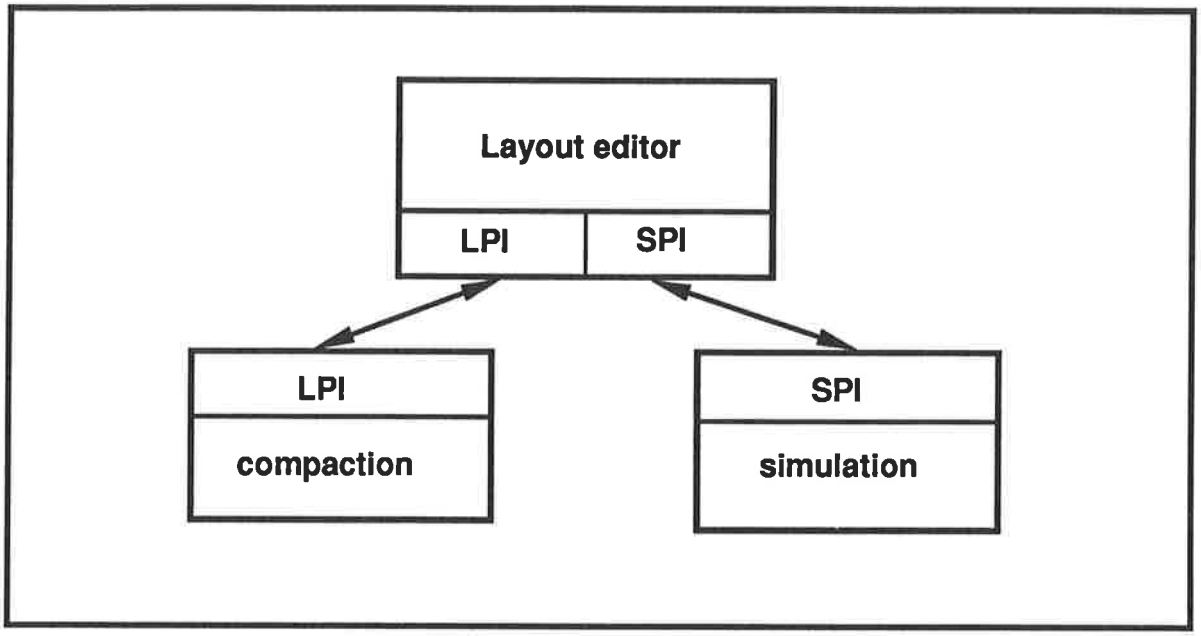

Figure 4: Interprocess communication using SPI and LPI.

4.4 Exchange Waveform information

Simulation tools with adequate functionality and high performance are required to meet the increasing complexity of electronic systems. To achieve this goal, different simulator tools can be integrated onto a simulation platform. Many simulators do exist, each tuned for a specific application field using different data types (e.g., boolean, integer, real, analog and digital values, ...) for representing the signal values. To interface between different simulators, it must be possible to exchange *simulation data*, *status information* and *command* routines. The routines to exchange simulation data must be general enough to support all the different data types and formats. Analogous with SPI, a Waveform Procedural Interface (WPI) is being defined. The interface contains routines to transfer values for the signals, to start the simulator, and to get the resulting values of the signals back to the controlling tool. The use of attributes to transfer the signal data will allow that different simulation tools can be coupled using WPI [Clae90].

5 Shortcomings

The SPI interface is used intensively in IMEC to couple different simulation tools, synthesis tools, module generation tools, ... SPI has proven to be very effective, however a number of shortcomings did appear [Schu89]. A number of these shortcomings will be solved in the near future. A preliminary study of the deficiencies and enhancements has been made. Solving these shortcomings, the new version of SPI must be upward compatible.
The major shortcomings are discussed in the following subsections.

5.1 One way communication

The transfer of structure data is *unidirectional*, netlist information flows from editors to tools. A tool cannot be both a 'consumer' and a 'producer' at the same time. A bi-directional communication (*read/write*) would be needed for that. An example is the following: a tool receives (*reads*) the structure information with SPI-calls and does some manipulations on this data. Then the tool sends (*writes*) the structure information to another tool. Figure 5 shows such a bi-directional communication. Only the exchange of structure related information, called attributes, goes in a *bi-directional* way between structure producers and consumers using backannotation operations.

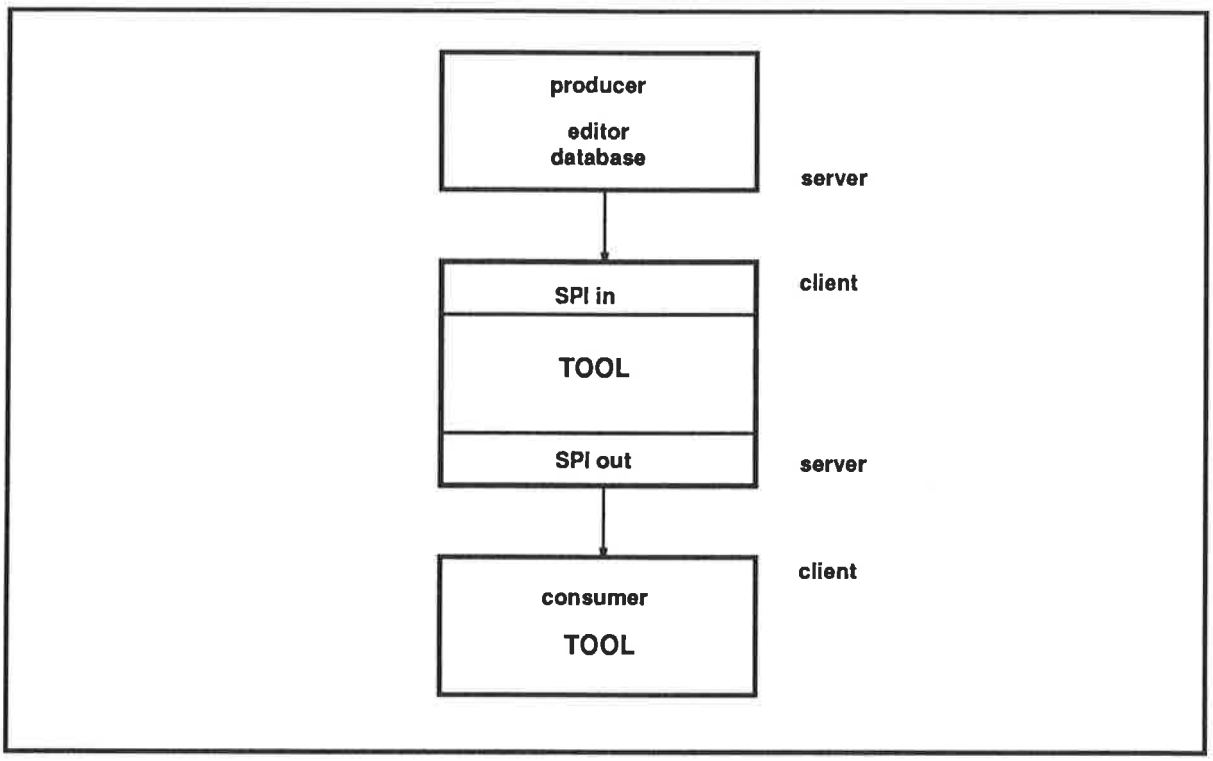

Figure 5: Bi-directional communication with SPI.

5.2 Synchronous communication

The implementation of the SPI routines does not allow *asynchronous* communication. In the new version of the SPI interface, processes must provide asynchronous communication. This means that one tool, after sending a request to another tool, must not wait until the other tool has completed the task, but can execute other commands from the designer.

5.3 No buffering of data

SPI transfers the data one element at a time. It should be possible to exchange groups of objects. Requests for information must therefore be powerful to get all information of an object. This will speed up the transfer of large designs in case of interprocess communication. Although requests to get information about one element should still be supported.

5.4 High number of communication routines

The current SPI routines are at a *low level*. New higher level functions that are more generic have to be defined. However, this implementation must still be compatible with the existing specification.

5.5 Fault handling and error return status

Fault handling and error return status are not available. In the current implementation, the processes can hang if a problem occurs.

5.6 Exchange of other views

There is a need for the exchange of other information views than netlists. The future implementation shall include additional views as listed below.

- Layout information.
- Waveform information.
- Signal Flow Graph (SFG) information (for high level synthesis).
- Parameterised netlist information: structural parameters are not allowed in SPI. For the moment, producers (editors) must do the parameter expansion themselves.

The Layout Procedural Interface and the Waveform Procedural Interface are already defined and will be implemented as part of the activities in the JCF project.

6 Work going on

The work on procedural interfaces is continuing in the JESSI-Common-Framework project (ESPRIT project 5082).
Future work will first concentrate upon research on improved mechanisms for interprocess communication [Sarm90,Mar90b,Mar90c]. Topics like bi-directional communication, synchronous/asynchronous interprocess communication, buffering and the enhancement of the performance of RPC mechanism will be taken into account.
Additional work will be done on the integration of new views in SPI.

Also feasibility studies to integrate SPI in the existing frameworks like Nelsis [Nels90], the NMP-CADLAB Framework [CADL90], or newly developed frameworks will be investigated.

7 Conclusion

In this paper, a *practical* and *open* interface for interactive CAD tool integration has been presented.
A set of existing CAD tools has been successfully integrated with SPI within the context of the ESPRIT-1058 project. Thanks to the integration, the CAD tools provide a true design assistant for the development of flexible VLSI modules.
SPI is an open system 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.
The SPI concept also proves to be a valuable input for the CFI Procedural Interface and for JESSI-CAD-Frame.
To promote the standard, SPI is put in *public* domain.

8 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-NatLab (The Netherlands) and EDC (formerly Silvar-Lisco Belgium). Many people have contributed to the success of the project, and we thank in particular I. Bolsens, T. Claes, K. Croes, P. Das, W. De Rammelaere, P. Johannes, T. Kostelijk, P. Lauwers, B. Lynch, G. Mole, H. Neto, P. Odent, P. Petroni, J. Raposo, L. Rijnders, G. Schrooten, J. P. Schupp, E. Vanden Meersch, E. Willems, for many discussions and for their continuing efforts in the integration with SPI.

References

[Cock89] J. Cockx, *SPI version 2.4*, EDC (Silvar Lisco), 18 September 1990.

[Cock90] J. Cockx, *RPC Compiler 1.7*, EDC (Silvar Lisco), 2 May 1990.

[Schu90] J. P. Schupp, J. Cockx, L. Claesen, H. De Man, *SPI: An Open Interface Integrating Highly Interactive Electronic CAD Tools*, IMEC vzw, VSDM division, EDAC-90, January 1990.

[CFI90a] *The CFI DAC '90 Scalar Netlist Programming Interface Specification*, CFI, 30 January 1990.

[CFI90b] *Draft Proposal Inter-Tool Communication Procedural Interface, Version 0.14, Document Number 52*, CFI, 8 October 1990.

[Ecip88a] The ECIP Conceptual Modelling Working Group, *ECIP Conceptual Model of Electronic Products*, November 7th, 1988.

[Ecip88b] D. Chalmers, F. Meys, *Successful CAD Integration Needs A Standard Conceptual Model*, Proceedings of the 5th Annual ESPRIT Conference, Brussels, November 14-17, 1988, pp. 170-185.

[Edif87] Electronic Industries Association, *EDIF - Electronic Design Interchange Format Version 2 0 0*, Washington D.C., May 1987.

[Schu89] J. P. Schupp, *An Evaluation of the SPI Interface*, IMEC vzw, VSDM division, June 1989.

[Seve89] R. Severyns, *Naming Conventions in SPI and in Tools Communicating with SPI*, Imec Document, October 1989.

[Dema90] A. Demarée, *Controlling Netlist Expansion in SPI*, Imec Document, December 10, 1990

[Rijn89] L. Rijnders, *Procedural Interface for Transferring Layout Information*, Imec Document, September 1990.

[Mar90a] L. Marent, *Layout Procedural Interface, User's Manual*, Imec Document, September 1990.

[Clae90] T. Claes, *Ontwikkeling van een Logmos Preprocessor XLV*, KIHL, Diepenbeek, 1990.

[Sarm90] H. Sarmento, *Ghost/Spook User Interface in the PACE framework*, IEEE International Conference on Computer-Aided Design, 1990.

[Mar90b] L. Marent, *Principles and Requirements of Interprocess Communication for CAD tools*, Imec Document, July 1990.

[Mar90c] L. Marent, *A Test Vehicle for Interprocess Communication*, Imec Document, October 1990.

[Nels90] *The Nelsis Framework, Release 4*, TU Delft, DIMES Design and Test Centre, May 1990.

[CADL90] *NMP – CADLAB Framework Release 1.1*, CADLAB, 1990.