The Electronic Design Interchange Format EDIF
— Capabilities and Evaluation —

by

Klaus Quibeldey-Cirkel
Michael G. Wahl

University of Siegen
Institute of Data Processing
Hölderlinstraße 3
D-5900 Siegen
West Germany

Abstract

This article gives an introduction to the de-facto industry standard for the exchange of electronic design information: EDIF. First, the incentives for a common interchange data format are analyzed and — contrasting the flaws of previous formats — the evolution of the EDIF standardization effort is outlined. Currently, EDIF is mainly used as a flexible vehicle for information exchange between electronic circuit designers on the one side and both CAD/CAE and semiconductor industries on the other. In the areas of mask artwork, netlists and schematics, the format is already well established as a standard in a multivendor design environment. Intensive on-going work is focused on the domains of printed circuit design, behavioral description and test. With the help of some illustrative examples, a digest of the EDIF framework — basic concepts and syntax — is presented. The paper closes with an evaluation of the status quo and an outlook on the long-term objectives of the EDIF community.

1 Introduction

1.1 Motivation for a Standard Interchange Format

The rapidly widening CAx scenario of today places the electronic circuit designer in a predicament: On the one hand, highly sophisticated design and verification tools are readily at his disposal supporting all stages of the design process — from the abstract view of behavioral description down to the geometric patterns of mask layout. In principle, he is free to choose equipment and services best suited for his particular task from a growing number of silicon foundries and CAD/CAE system and workstation companies. On the other hand, however, the variety of feasible choices is strongly narrowed by the incompatibility of the diverse data formats actually in use.

Looked at in the short term, the problem of data exchange between in-house design systems and out-of-house equipment and services from ASIC, PCB and tester industries can only be solved by format translators tailored to the internal data structures of the individual tools. It goes without saying that this solution brings about a large overhead of software development. In the long run, solely a common interchange format for all facets of electronic design data will drastically diminish investment in format converting. Fig. 1.1 is to illustrate this. Given the number of senders and recipients by *n*, *n(n-1)* different formatters would be necessary for an unrestricted bidirectional information exchange. Compared with *2n* conversions using a common information interface, the strongly felt need for an international cooperative standardization effort is self-evident.

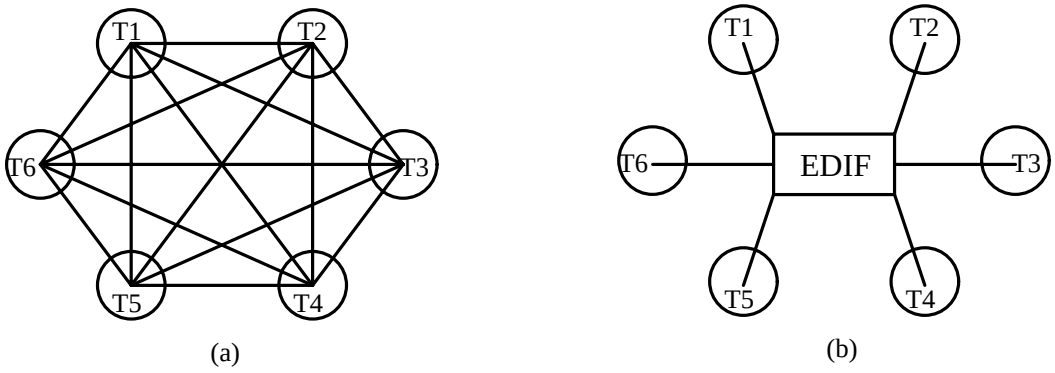

Fig. 1.1: Reducing the number of formatters from quadratic (a) to linear (b).

1.2 Shortcomings of Previous Data Formats

While a wide variety of interchange formats and hardware description languages have been developed over the past decade, none of them has turned out to be a suitable candidate for an industry standard because of one or more of the following drawbacks.

- *Narrow scope*

  Not all aspects of electronic design data (e.g. mask artwork, netlists, behavioral description) are adequately addressed by the same format.

- *Proprietary constraints*

  Many formats are owned or controlled by specific CAD/CAE companies and silicon foundries. Thus, access and proliferation are limited by licensing.

- *Difficult to implement and to extend*

  The development and maintenance of parser and generator software, especially if upward compatibility is an issue, are not always supported by a straightforwardly structured data format. As design and verification methods of integrated and printed circuits rapidly evolve, the spectrum of design data widens and requires easily manageable extensions to the actual version. A great deal of present formats misses the capability to meet future needs of the design process.

The public interchange format EDIF has successfully overcome most of these drawbacks. From the beginning, it has been designed to achieve compatibility in the multivendor CAD/CAE environments of today and tomorrow. Not by decree but by broad applicability, EDIF has become a de-facto industry standard.

1.3 The Evolution of the EDIF Effort

In late 1983, EDIF was launched as a cooperative initiative of major CAD system and semiconductor corporations: Daisy Systems, Mentor Graphics, Motorola, National Semiconductor, Tektronix and Texas Instruments. The initial approach was to extract the best features from previous formats to develop a powerful unified industry standard. A steering committee was formed to coordinate the development activities with recognized institutes of standardization. The first version of the Electronic Design Interchange Format (1985) already covered the necessary subsets of data to be exchanged between ASIC designers and silicon foundries. From the beginning, ASIC-oriented information, such as macro libraries, technology specifications, netlists, schematics, mask layout and stimulus/response vectors, could be expressed in EDIF. In order to investigate special problem areas of the design process, the steering committee impaneled several subcommittees of experts to address the issues of printed circuit design, test specification, device modelling, behavioral description, physical design rules, process and device, reliability and quality. With upward compatibility in mind, the solutions found are

proposed for inclusion within the evolving EDIF framework. Fig. 1.2 illustrates the scheme of the EDIF organization as a whole.

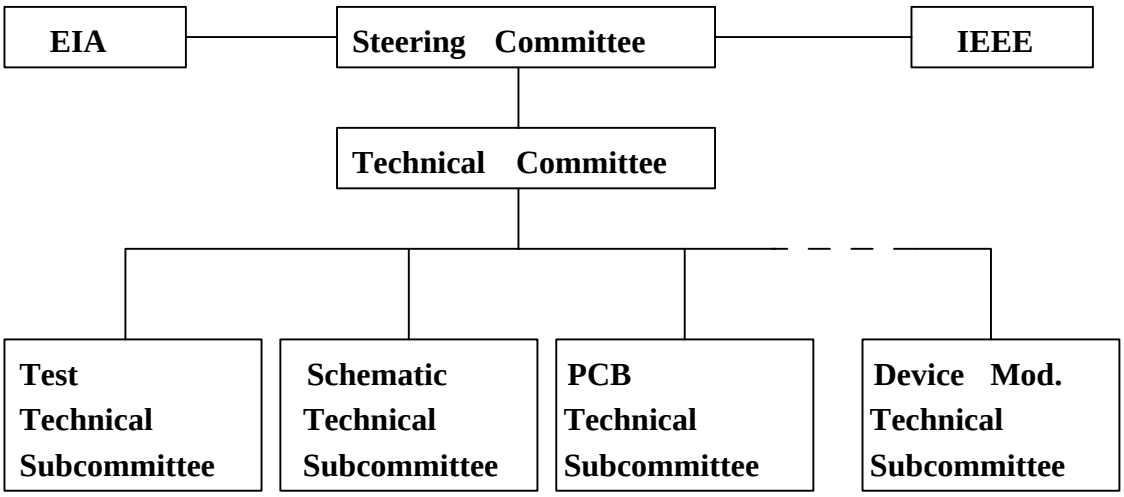

Fig. 1.2: EDIF organization scheme

2 EDIF Digest

2.1 Basic Concepts

Modern design methodology is mainly based on two powerful principles of structuring: *abstraction* and *instantiation*.

Different levels of abstraction, such as behavioral, logical and physical, help to structure the mass of data compiled during the entire design process of electronic circuits. The top-down approach involves the method of gradual refinement as it is well known in software engineering. The bottom-up approach facilitates the composition of complex structures from less complex ones.

The instantiation principle is essential for VLSI design. By referencing previously defined and verified design units — called cells —, an instantiation hierarchy avoids the time and storage space consuming efforts of replicating cell contents. At a given level of abstraction, it further supports structuring and, thus, clarity of information.

The basic concepts of EDIF strictly observe these fundamental principles of structuring mass data. The following is to demonstrate this.

2.2 The Structure of an EDIF Description

Fig. 2.1 illustrates the EDIF hierarchical approach of representing design data. At the highest level, the description contains all aspects of information in a single logical file. The file may contain several libraries defining cells and technologies. A library represents a grouping of cells with a set of technological features in common. Libraries may be defined either locally, i.e. within the current EDIF file, or externally. In the former case, design information should be complete. This is not to say that the design to be shipped has to be complete, too. The format allows shipment of partial information, such as macro libraries and technology specifications from ASIC manufacturers, the only condition being that the subset of information is self-contained.

As most design and verification tools only support a narrow scope of application — a design rule checker, for instance, will mainly work on mask layouts while a simulator, by contrast, will do so on netlists, device models and testvectors —, EDIF provides several perspectives of the information content of a cell. A cell definition can be partitioned into one or more different representations, called *views*. Currently, ten *view types* are provided. While eight views are directly concerned with design information, two are for general purposes.

- *netlist*

  This view is used to specify connectivity. It holds the details of instantiated cells and the connectivity between them.

- *schematic*

  Besides connectivity, associated diagrammatic information for schematic capture is described.

- *symbolic*

  The *symbolic* view is used to transfer high-level topological information of cells. It is placed at an intermediary stage between *schematic* and *maskLayout*.

- *maskLayout*

  This view describes geometric figures on mask layers and the information of how to interpret and represent them. The description is restricted to the corresponding cell boundaries. No information of connectivity is given.

- *pcbLayout*

  The *pcbLayout* view represents a class of information similar to the one of *maskLayout* except for the top level cell being a board.

- *behavior*

  (still incomplete in the present version)

- *logicModel*

  This view is used to describe the logic simulation model of a cell. It includes support of stimulus and response data.

- *graphic*

  Graphical information shared by several cells, like symbols, logos, pageborders, etc., can be expressed by this representation.

- *document*

  The *document* view holds textual information, diagrams and symbols in a printable form.

- *stranger*

  In all cases where the main EDIF views do not suffice, this one permits the use of any constructs to define special sets of data. However, no guarantee is given that the semantics of this view is transferable to or interpretable by another CAD system.

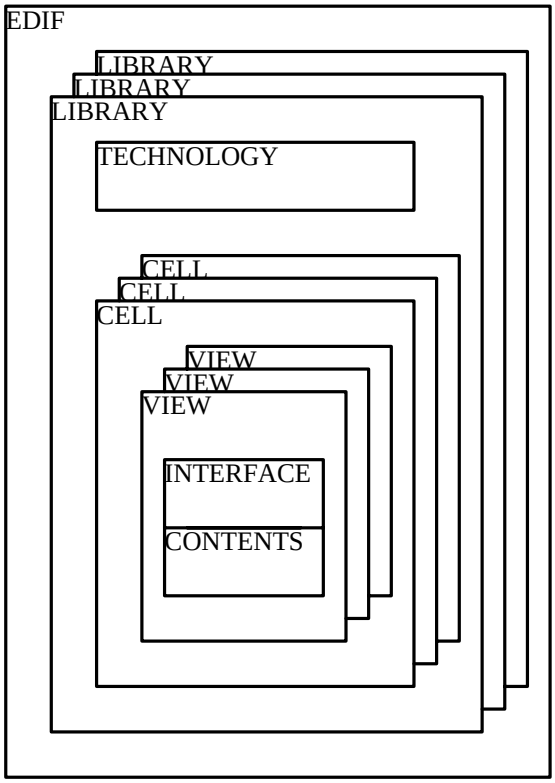

Fig. 2.1: Hierarchical structure of an EDIF file

At a given view, a cell is further structured into two blocks: *interface* and *contents*. The *interface* block can be regarded as the abstraction of the cell. It gives the set of information necessary for an instantiating cell to communicate with the instantiated one, e.g. via port references.

The detailed implementation of a cell at a particular *view* is described in the *contents* block. The description involves either the actual cell definition at the lowest level or

instantiations of simpler cells together with their connectivity. In the latter case, EDIF requires a *define-before-use* approach to make sure that the cell definitions are completed before the cells are instantiated, i.e. referenced. This concept facilitates the development of single-pass parsers and ensures design consistency.

2.3 Basic Syntax and Semantics

First and foremost, it should be stressed what EDIF does not stand for. It is neither a programming language nor a database system for electronic designs. In spite of its resemblance to the *Lisp* programming language, it is not executable. The intention solely lies on the transmission of design data from one database to another. Furthermore, the format does not specify any medium of transport, nor does it determine any physical compaction or encryption techniques. Issues of data access and storage efficiency are not addressed.

The syntax of EDIF shows a *Lisp*-like nested structure. The "atoms" are primitive data like identifiers and tokens. Complex structures are composed of parenthetical lists containing either primitive data or other lists as elements. Typically, the first element of a list is a reserved keyword which gives meaning to the following items.

Thus, the general form of an EDIF statement is as follows:

(keyword
    { *integerToken* | *stringToken* | *identifier* | *form* }
)

The curly braces {} indicate that an item may be repeated zero or more times. A vertical bar | symbolizes a choice. The *Lisp*-like syntax is obvious: A form is always put in parentheses (). It begins with a keyword followed by a list of items. As items may be other forms, an EDIF statement is typically given by a nested structure.

As the scope of the description may range from geometric primitives in the *maskLayout* view to procedural description of a microprocessor in the *behavioral* view, the language definition supports three different levels of complexity which are upward compatible. At *level 0* the user can exclusively describe non-parametric designs, i.e. only simple constants are permitted. *Level 1* allows the use of variables and expressions. This implies that parametric cells, e.g. a register with word length as parameter, can be expressed. If complex data types and procedural description are an issue, *level 2* provides the necessary constructs as they are known from higher programming languages.

The authoritative source of the language definition is given by the *"EDIF Reference Manual"*. Fig. 2.2 gives a simplified example of the syntax of an EDIF description. A flipflop is described from different views. Reserved keywords are set in bold type.

```
(edif SimpleExample
  (edifVersion 2 0 0)
  (edifLevel 0)
  (keywordmap (keywordlevel 0))
  (status
    (written
      (timeStamp 1990 09 13 12 45)
      (dataOrigin "University of Siegen") ...
    )
  )
  (library SimpleMacros
    (edifLevel 0)
    (technology ... )

    (cell NAND ... )

    (cell RSflipflop (cellType generic)

      (view LogicalNetlist (viewType netlist)
        (interface
          (port R (direction input))
          (port S (direction input))
          (port Q (direction output))
          (port NQ (direction output))
        )
        (contents
          (instance G1 (viewRef NANDNetlist (cellRef NAND)))
          (instance G2 (viewRef NANDNetlist (cellRef NAND)))
          (net FlipFlopNetlist
            (joined (portRef IN1 (instanceRef G1)) (portRef R))
            (joined (portRef IN2 (instanceRef G2)) (portRef S))
            (joined (portRef IN2 (instanceRef G1))(portRef OUT (instance G2))(portRef NQ))
            (joined (portRef IN1 (instanceRef G2))(portRef OUT (instance G1))(portRef Q))
          )
        )
      )

      (view SchematicDiagram (viewType schematic)
        (interface ... )
        (contents ... )
      )

      (view MaskArtwork (viewType maskLayout)
        (interface ... )
        (contents ... )
      )

      ...

    )
  )

  (design (cellRef RSflipflop))

)
```

Fig. 2.2: File extract to demonstrate the syntax of EDIF

3 Evaluation and Outlook

3.1 Status Quo

EDIF has become *"Recommended Standard ANSI/EIA—548"* which contains the official definition of the format. The present EDIF version 2 0 0 dating from April 1987 enjoys broad acceptance among both industry and science.

In the last few years, a great deal of CAD/CAE system and workstation companies in the U.S.A., Europe and Japan has developed EDIF-oriented design interfaces. Numerous software implementations based on the open standard are available on the market. It is widely used as an industry standard for the exchange of design data in the process of designing, testing and manufacturing electronic circuits with different CAx tools and on different sites.

The format has been subject of several international scientific conferences on VLSI design, testing and manufacturing. The EDIF standardization effort is a permanent focus of interest within international projects like ECIP *(European CAD Integration Project)*, JESSI *(Joint European Submicron Silicon)*, EIS (*Engineering Information System, U.S.A.)*, to name but a few. Many universities and research centers are deeply involved in EDIF development activities.

In addition to the initial organization scheme abroad, a similar one was formed in Europe (*European EDIF Technical Subcommittee* and subordinate bodies). Several EDIF User Groups emerged as a national forum for users who want to participate in the evolving standardization process. All EDIF bodies pursue an open policy of information and are open to any user from both industry and research.

3.2 Future Directions

Future releases of EDIF are planned to be incremental, meaning that extensions to the present standard will be targeted at particular domains. Later versions will include extensions to schematics, PCB and test. Currently, a great deal of research work is concentrated on data modelling aspects. The general objective is to build a consistent universal data model that will help to ensure semantic conformity. The authors of this paper are involved in the definition of such a data model for the area of testing. At the University of Siegen, intensive research activities are dedicated to the information link between CAE and ATE, that is to say between simulation and test of electronic circuits.

In case of further questions on EDIF organization and standard, feel free to contact the authors at their address.

References

- EDIF Electronic Design Interchange Format Version 2 0 0. EDIF Steering Committee, Electronic Industries Association (EIA), Washington D.C., April 1987
- Introduction to EDIF. EDIF Monograph Series, Vol. 1, EIA, Sept. 1988
- EDIF Connectivity. EDIF Monograph Series, Vol. 2, EIA, June 1989
- Using EDIF 2 0 0 for Schematic Transfer. EDIF Schematic Technical Subcommittee, EIA, July 1989
- Proceedings of the First European EDIF Forum. Brussels Airport, Belgium, Sept. 1987
- Proceedings of the Second European EDIF Forum. Amsterdam, The Netherlands, Oct. 1988
- Proceedings of the Third European EDIF Forum. Bonn, Germany, Oct. 1989

Abbreviations

CAD: Computer Aided Design

CAE: Computer Aided Engineering

ATE: Automatic Test Equipment

PCB: Printed Circuit Board

ASIC: Application Specific Integrated Circuit

VLSI: Very Large Scale Integration

LISP: List Processing Language

ANSI: American National Standards Institute

EIA: Electronic Industries Association

IEEE: Institute of Electrical and Electronic Engineers