Basic Requirements for an Efficient Inter-Framework-Communication*

Arno Kunzmann Ralf Seepold

Forschungszentrum Informatik, FZI
Systementwurf in der Mikroelektronik (Prof. W. Rosenstiel)
Haid-und-Neu-Straße 10-14, D-76131 Karlsruhe, Germany
E-mail: kunzmann/seepold@fzi.de

Abstract

For the design of complex electronic CAD systems, several frameworks have been proposed and implemented offering a different set of basic services, and therefore, supporting specific and dedicated design activities. In this paper the results of a case-study are summarized where two almost complementary frameworks have been coupled in order combine their advantages: (1) the JESSI-COMMON-Framework with special emphasis on design management, and (2) FMCAD (representing a widespread design environment of the electronic CAD domain) that is tuned on close tool integration and efficient data handling. As a result of this case-study the basic requirements for an efficient inter-framework-communication are described to indicate some trends for future framework developments.

1 Introduction

Within the electronic CAD domain there exist several frameworks with a different set of services resulting in a specific support of dedicated design activities. For many applications a combination of these isolated environments could improve design efficiency. Therefore, the current work focuses on the coupling of two different, almost complementary frameworks: the JESSI-COMMON-Framework (JCF) release 3.0[‡] [17] and a widespread design environment of the electronic CAD world that will be called FMCAD in this paper.

As mentioned above, each framework has its own strengths. The JCF framework is a very powerful environment for supporting design management, configuration management and concurrent engineering. Therefore, JCF does not focus on a certain domain, but supports a set of services which are required in many different domains. In contrast to that, the function of FMCAD is concentrated on close tool integration and efficient interaction for specific prob-

* This research is supported by the ESPRIT project 7364.

‡ SNI provides a commercial version of JCF under the name SIFRAME.

lem-oriented electronic CAD environments. In this case, the guidance and support of a designer are only rudimentarily provided.

The ultimate goal is to combine the main advantages of both approaches to form an extendable and powerful environment for the development of electronic systems and to derive fundamental requirements for an efficient inter-framework-communication. Subsequently, the main differences of the isolated systems and the aiming advantages of a combined approach are discussed.

2 Presentation of the JESSI-COMMON-Framework and FMCAD

In this chapter the JESSI-COMMON-Framework (JCF) and the FMCAD framework are presented to give a brief overview of the main structuring (e.g. architecture). Additional information can be found in [6], [8], [9] and [16].

2.1 JESSI-COMMON-Framework

JCF release 3.0 was chosen for the for framework coupling case-study because this release was available at the beginning of this work. Essential parts of the JCF information architecture can be found in [9], the main basic concepts are summarized in this subsection.

JCF applies a strict distinction between *resources*, which can be used for different projects, and actual *project data*. At the *resources definition*, the JCF administrator defines users and teams that are allowed to use the framework. Furthermore, design flows [11] are precisely defined. For this purpose it is described which tools are part of the design flow, how these tools can be used, and which data are passed between the tools. Note that these resource data are stored as meta data in the framework, and are completely under control of the framework. This is realized in contrast to FMCAD which does not manage any resource data.

The kernel of the JCF *project organization* is formed by a product structure that is modelled via a *Cell* hierarchy (more precise: via *ComposedOf* relations between *Cell Versions*). Each *Cell Version* has a flow and a team attached to it. The philosophy is that only members of the team are allowed to work on such a *Cell Version*, and that they must follow the attached flow. A particular *Cell* can have several *Cell Versions*, with possibly different *teams* or *flows* attached.

When a user wants to work on a *Cell Version*, he must first *reserve* it in his *Workspace*. It is then isolated for his own use, and no other user can do any work with it. As soon as the work on the *Cell Version* is finished, the user *publishes* it. The *Cell Version* is then removed from the *Workspace*, and can be used by the other users of the *team*.

JCF stores all its data in the underlying object-oriented OMS database. As soon as data are needed, they are checked-out from OMS, and when work is finished, data are checked-in again. The fact that a real database is used for data storage is a significant difference to the FMCAD approach, where data are just stored in the UNIX file system. For more information, we refer to the JCF documentation (e.g. [6] and [12]).

2.2 FMCAD

FMCAD stores design data in libraries. A library consists of a directory in the Unix file system and a library catalog file. The catalog file stores the logical (meta) data objects. The design data files are stored in the Unix directory. The basic logical data objects in a FMCAD library are *cells*, *views*, *cellviews*, *cellview versions* and *configurations*.

A *cell* is the basic design object. It forms an individual building block of a chip or system. It is a logical, rather than a physical design object. Each *cell* has one or more *cellviews*.

A *view* is one type of representation, such as schematic or layout. A *view* has a name and is of one specific *viewtype*. The *viewtype* associates the *view* with a certain FMCAD tool, for example the layout editor. A *view* is also a logical design object. A *cellview* is a virtual data file created in association with a *cell* and a *view*. It is a logical, rather than a physical design object. A *cellview version* is the data file of a *cellview* at a particular time. *Versions* are created by check-out/check-in operations. A *cellview* can have more than one *version*. The *version* models the link to the design file in the file system. The checked-out *version* is the *version* which a user is working on. Only one *version* of a *cellview* can be checked out at a time. A *configuration* is a collection of *cellview versions* that are related. For each *cellview* at maximum one *version* can be part of the *configuration*.

2.3 Basic Results

The combined JCF-FMCAD framework approach should sum up the advantages of both approaches to form a new and more powerful environment for chip design. Of course, it is not possible to combine only advantages without reducing the high flexibility of FMCAD, but as described in this paper an efficient design support can be enabled. A very essential task of the framework coupling is the mapping of the different data models. In order to support the individual characteristics of the frameworks, the master-slave approach was selected. Based on the very powerful management features of JCF, it was decided that JCF is the master framework. A more detailed description of the data mapping is given in [9].

In the following, the main achievements including the basic problems of the combined JCF-FMCAD framework approach are evaluated and summarized. A short overview is provided in Table 1 and a detailed description of the results is stated by the following bullet points:

- **Multi-User Design and Concurrency Control**

One achievement was to use the multi-user capabilities of JCF 3.0 also in the JCF-FMCAD environment. Based on the JCF *workspace* concept, individual design elements can be reserved and isolated from the access of other users. While in FMCAD, parallel work on different versions of the same *cellview* is not possible, the JCF-FMCAD environment provides this feature.

- **Design Management and Data Consistency**

Compared with FMCAD, the combined framework provides a more flexible design management system. For instance, JCF-FMCAD offers the versioning of both *cells* and *design objects*. In addition, the representation of hierarchy information by meta data allows a more powerful consistency check.

- **Handling of Design Hierarchies**

This was one of the most difficult tasks during the JCF-FMCAD integration. Compared with FMCAD some flexibility in handling design hierarchies was lost, since the existing JCF-FMCAD prototype requires that all hierarchical manipulations must be done manually via the JCF desktop. Another limitation is caused by non-isomorphic hierarchies that are not yet supported by JCF release 3.0 but will be supported by future releases.

- **User-interface**

In the JCF-FMCAD prototype system, the designer has to work with both the FMCAD and JCF user-interface. In spite the fact that JCF is conform with the major user-interface standards (using X-Windows and Motif), the user has to cope with an additional user-interface.

- **Flow Management and Derivation Relations**

In contrast to standard FMCAD, JCF-FMCAD provides flow management capabilities, but in a slightly restricted way, due to the available handling of non-isomorphic hierarchies in the current JCF release 3.0. Especially to support the archiving and retrieving of design data, derivation relations and *what-belongs-to-what* information are very essential features that are supported by the JCF-FMCAD framework system.

- **Performance**

This aspect is of lower importance since the main aspect of this inter-framework coupling is functionality. The performance of meta data operations in the JCF-FMCAD environment is sufficiently high.

Criteria	JCF 3.0	FMCAD	JCF-FMCAD
Multi-User Design	+	-	+
Concurrency Control	+	o	+
Design Management	+	o	+
Data Consistency	+	-	+
Design Hierarchies	-	+	o
User Interface	+	+	o
Design Flow Management	+	-	+
Derivation Relation	+	o	+
Performance	o	+	o

+ ≡ good support
o ≡ medium support
- ≡ weak support

Table 1: Criteria Table

3 Requirements for an Inter-Framework Communication

Based on the performed case-study, a major task of future developments for inter-framework-communication is to close the gap between isolated and stand-alone frameworks. Existing framework isles have been claimed for industrial standards and are very important for long-term company decisions. The following requirements are derived from the described experiences that are based on a fundamental mid-term study and several consultations with major chip developers. As a result, Table 2 summarizes the main requirements that can be regarded as general requirements and should be taken into account for future developments of open frameworks.

Requirements	Priority	Advantage
efficient data access	high	enhanced performance
support of standards (e.g. CFI)	high	easy integration and customization
support of user-interface customization	medium	provide a common look & feel
modularity	high	adequate and flexible customization
support of project management	high	reduced costs by efficient use of resources
design management functionality	high	flexible modelling
shared libraries	medium	data consistency
support of integration	high	fast and tight tool integration
support of ITC	high	efficient use of standard capabilities

Table 2: Requirements for Inter-Framework Communication

The coupling of different frameworks requires the mapping of different data models. Furthermore, the exchange of meta and design data is necessary to communicate between the framework sites. An efficient data access influences both the performance and acceptance of the coupled system. The priority of this requirement is very high because the end user has to be convinced to use the system. Limitations exist in case of closed interfaces or import/export functions that are based on file-formats. Future framework releases have to take into account that the mentioned aspect is of high priority.

The coupling of very different systems is hardly manageable if existing standards are ignored or missing. CFI standards should be supported as soon as they are available to guarantee easy and efficient integration of environments.

Due to the fact that each user prefers a customized environment to enhanced his own work profile, a modern user-interface should be very flexible and easy to configure. Of course, this requirement is of medium importance.

Regarding the different existing environments in IC companies, each major company mixes in-house developed tools with commercial tools or frameworks. In many cases, the optimal solution is not to integrate the whole tool set, but some specific modules, since a full integration requires too much time and effort. Furthermore, a module concept provides interfaces to customize and configure a framework that fulfils exactly the requirements of the company. If standards (e.g. CFI) are missing other preliminary interfaces have to be offered. The final transfer should be supported by migration tools.

The deep integration of design tools is a very complex task. But the management of a complex project is much more important than a deep integration. Wrong decisions on this level will waste time and money and may cause the stop of a project. Efficient project management facilities which support the project manager through all phases of the development are completely missing. Additionally, new concepts have to be developed to close the gap between existing systems in the areas of Operation Research and the existing ECAD frameworks.

Several different levels of designer support can be distinguished (e.g. knowledge based systems or design flows). The acceptance of the system by the designer, who is a specialist in using a subset of the design tools, will be influenced by the flexibility of the design management. The design management must be flexible to avoid strictly forced design flows were possible, and it must be applicable to control the project state and ensure a specific quality level. On the one hand a bypassing of flows should be impossible, and on the other hand the flow should offer more flexibility than pure logical AND connections.

The use of libraries is a central concept in chip developments. It is very difficult to keep the library consistent and continue work during the design while changes are taking place. Concurrent engineering should be offered for the designer's work as well as for system administrators to ensure consistency at all levels.

It is important to notice that there will be different levels of integration for external tools. Only a minor set will be deeply integrated and the majority of tools can be adapted by a loosely coupling. Nevertheless, integration facilities must be available, very efficient, supported by standard tools and conform to existing standards. Many commercial systems support this facility by a powerful extension language.

Concerning the modularity of software and software products, inter-tool-communication (ITC) is a future-oriented feature to offer very flexible and powerful mechanisms for tool coupling. This coupling is necessary for tools which communicate among others inside the same framework and also between different frameworks. This service should be offered as a basic tool for integration and communication.

4 Conclusion and Future Work

With the described JCF-FMCAD framework prototype it could be demonstrated that a combination of frameworks with different basic architectures and individual design services can provide enhanced functionality. The study provides basic results which can be used to enhance the state-of-the-art in inter-framework-communication and push future framework development.

In some cases, the coupling offers not the advantages of both isolated frameworks but this fact is mainly influenced by the existing software releases, i.e., limitations in available interfaces. The coupling of frameworks depends on the experience of the integrator and is not supported by existing tools, therefore, the data model mapping remains a key aspect for successful integration.

The new JCF release 4.0 will offer more open interfaces and enhanced functionality. The integration of tools and the coupling of other framework isles should be easier and more efficient than in the existing system. A planned completely new integration and coupling will show, if the demanded requirements are fulfilled. Furthermore, the level of support for inter-framework-communication (e.g. ,work in progress' or ,archive') has to be defined with respect to technical and praxis-oriented requirements for the next generation of design environments. This objective can be regarded as the next step towards a new prototype.

Acknowledgements

We would like to thank Hans Kok, Kees Lepoeter (Philips Research, Eindhoven) and Thomas Harriehausen (Siemens HL, Munich) for their contributions, support during the study and the fruitful discussions which enabled this publication.

References

- [1] W. Allen, D. Rosenthal and K. Fiduk, *'The MCC CAD Framework Methodology Management System'*, Proc. 28th DAC, pp.694-698, 1991.
- [2] K.O. ten Bosch et al., *'Design Flow Management in the Nelsis CAD Framework'*, Proc. of 28th DAC, pp. 711-716, 1991.
- [3] F. Bretschneider, Ch. Kopf and M. Zolg, *'Process Control for Large Complex Systems'*, Winter Simulation Conference, 1991.
- [4] Cadence Design Framework II Reference; Cadence release 4.2.2 documentation.
- [5] Framework Criteria Definitions, Authors: SP4 Team, Editor: H. Dorfmeister, Internal Report, ESPRIT Project 7364, JCF/BOSCH/001-02/25-September-1992, 1992.
- [6] JCF 3.0 System Overview; Documentation JCF release 3.0.
- [7] Y. Kashai, *'Flow - A Concurrent Methodology Manager'*, Proc. EDAC, pp. 20-24, Paris, 1992.
- [8] H. Kok, K. Lepoeter and R. Seepold, *'Report on the JCF-Cadence encapsulation project'*, ESPRIT Project 7364, Internal Report, JCF/FZI-Philips/032-1/18-May-94, 1994.
- [9] A. Kunzmann and R. Seepold, *'Enhanced Functionality by Coupling the JESSI-COMMON-Framework with an ECAD framework'*, Proc. EDAC, Paris, 1995.
- [10] E. Kwee-Christoph, B. Eschermann, O. Haberl, R. Kumar and A. Kunzmann, *'The CADEC VLSI Design Support Methodology'*, CompEuro, The Hague, The Netherlands, 1992.

- [11] D. Liebisch and A. Jain, '*JESSI COMMON FRAMEWORK Design Management -- The means to Configuration and Execution of the Design Process*', Proc. Euro-DAC, pp. 552-557, Hamburg, 1992.
- [12] A. Meckenstock and D. Zimmer, '*Concept and Architecture of a Distributed Object-Oriented Database Kernel*', Cadlab Report 13/92, CADLAB Paderborn.
- [13] S. Parikh, D. Sarnoff, M.L. Bushnell et al., '*Distributed Computing, Automatic Design, and Error Recovery in the ULYSSES II Framework*', Proc. EDAC, pp. 610-617, Paris, 1994.
- [14] T. Quellenberg, J. Schubert and G. v. Bultzingsloewen, '*FLOPPY - A Flow Management for On-line Processes in Production Systems*', submitted to IMSE European Workshop on Integrated Manufacturing Systems Engineering, Grenoble, 1994.
- [15] F. J. Rammig and B. Steinmüller, '*Frameworks und Entwurfsumgebungen*', Informatik-Spektrum (1992) 15, pp.33-43, 1992.
- [16] J. Schubert, R. Seepold, A. Kunzmann and U. Weinmann, '*Modelling a FPGA Design Flow in the JESSI-COMMON-Framework*', The 10th ISPE/IFAC International Conference on CAD/CAM, Robotics and Factories of the Future CARs&FOF'94, pp. 680-685, Ottawa, August 1994.
- [17] B. Steinmüller et al., '*JCF-Start-up Phase Final Version*', Internal Report, ESPRIT Project 7364, JCF/CADLAB/053-02/01-June-92, 1992.
- [18] M. Straube, W. Wilkes and G. Schlageter, '*HANDICAP - A System for Design Consulting*', Proc. EDAC, pp. 600-604, Paris, 1994.