Enhanced Functionality by Coupling the JESSI-COMMON-Framework with an ECAD Framework*

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

Within the electronic CAD domain there exist several frameworks each with a different set of services, which results in the specific support of dedicated design activities. One of the most innovative framework systems is the JESSI-COMMON-Framework (JCF). In contrast to JCF, a widespread ECAD framework (called FMCAD) has nearly complementary goals: while JCF offers strong support for working with consistent data concurrently, the basic functionality of FMCAD heavily supports the designer. In order to combine the advantages of both frameworks, a new hybrid framework was developed by combining JCF and FMCAD. As described in the paper, this ultimate goal could not yet be reached, but the achieved results and experience with the existing prototype emphasize that this integration and encapsulation could be very helpful for future frameworks.

1 Introduction

CAD frameworks [Ramm92] provide many different services to support the development of complex designs. These developments are typically based on teams working with a large number of different dedicated tools which require highly flexible design systems. In the domain of electronic CAD, very often, tools of different vendors have to be combined to form one common, efficient design environment ([Seep94a] and [Seep94b]). In addition, the combination of company-specific customization and modelling techniques form a highly complex system with many interrelationships. Frameworks can be regarded as a software infrastructure which support tool integration, concurrent engineering and modelling techniques to meet the goals of managing and handling these design environments. In order to demonstrate the broad range of offered framework services, some frameworks will be briefly described in the following.

Most of the existing frameworks offer different levels of integration. For instance, the JESSI-COMMON-Framework (JCF) ([Stei92] and [Over]) supports three integration levels, ranging from simple black-box integration up to very tight white-box integration. The main advantages of JCF are its combination with very powerful design management features and different versioning techniques [Lieb92]. Having a differing focus, the CADENCE™ Design Framework II (DFII) [Cade] provides more functionality at the integration and customization level, and offers a large number of integrated tools for IC design. Design management and concurrent engineering are only rudimentarily supported by the standard DFII and may cause consistency problems. An important representative framework of the academic world is the Nelsis framework [Bosc91]. Compared with JCF, Nelsis supports a more flexible design flow management, but Nelsis lacks the strong concurrent engineering and configuration possibilities. In contrast to the JCF release 3.0, Nelsis provides open interfaces for tight tool integration. The Ulysses framework [Pari94] mainly supports distributed environments and automatic execution of tools by dynamic network load balancing and task debugging. Specific designer assistance is offered by so-called *Design Consultants* like CADEC [KC92], the Bretschneider approach [Bret91] or Handicap [Stra94]. The MCC [Alle91] system focuses very strongly on process control and load balancing. The latter task is also supported by the FLOPPY [Quel94] and FLOW [Kash92] system.

The described subset of existing frameworks emphasize that each approach is a stand-alone approach, focused on certain domains and application profiles, which consequently results in limited capabilities. In order to combine the strengths of the available frameworks, a very important future task is to overcome the existing local boundaries and to form combined framework-based systems.

In this paper, two nearly complementary frameworks have been selected for inter-framework coupling: JCF and a widespread ECAD framework, called FMCAD. The main goal is to combine the powerful design management, concurrent engineering and configuration management with the FMCAD framework, which provides powerful integrated tools and a very flexible customization language. In order to closely couple these framework isles one framework has to be the master and the other must be

*) This research is supported by the ESPRIT project 7364.

the slave in the coupling scenario. Due to its functionality, JCF was chosen to be the master. Furthermore, it was decided that the hybrid framework prototype should contain a couple of different FMCAD tools (schematic entry, layout editor and a simulator). In order to couple the frameworks properly, the different data models have to be mapped onto each other. The following section introduces the JCF and the FMCAD framework models before the mapping and encapsulation procedures are described. The evaluation results and conclusions are reported in Sections 3 and 4 of this paper.

2 JCF and FMCAD Mapping

In order to build an efficient hybrid framework by coupling JCF and FMCAD, the individual information models have to be mapped onto each other. In most cases, notations and information models differ greatly. To introduce and motivate the selected mapping, both the JCF and the FMCAD information model architectures will be presented in this section. The current encapsulation of the FMCAD tools is described in the final sub-section.

2.1 JESSI-COMMON-Frame 3.0 Architecture

The main architecture components which are important for coupling the different models are presented in Figure 1.

JCF makes a clear distinction between *resources* and *project data*. *Resources* are defined by the framework *administrator*. Each user becomes a member of the appropriate teams and these teams can be used to support projects. Similar to this definition of teams, each *design flow* has to be defined in advance, and therefore, it will become part of the *resources* and can be regarded as metadata. The metadata are completely under the control of the framework.

The *project* organization is described by *cells* and relationships between *cells*. The *cell* consists of the attached *flow* and *team*. The *version* mechanism enables an instantiation of the same *cell* which is called a *cell version*. Each *cell version* may contain a modified *flow* as well as a different *team*. Inside the *cell version* there exists a second versioning mechanism which is called a *variant*. The users have the ability to derive many different *variants* of the same *flow* in one *cell version* to store the modifications and to select the optimal design solution.

Flows are fixed and cannot be modified, i.e., the user must follow the flow constraints. Furthermore, the *workspace* concept of JCF allows only one user to work on a particular *cell version* if this cell version is reserved in his private *workspace*. Other users are only allowed to read the published parts of the design data. When the work is finished, the cell can be *published* and then be modified by other users. This *workspace* concept is the kernel of the JCF multi-user capabilities.

Metadata and design data are stored in the common object-oriented database OMS [Meck92]. In case of encapsulation, the required data are copied to and from the database via the UNIX file system. Direct access to the internal

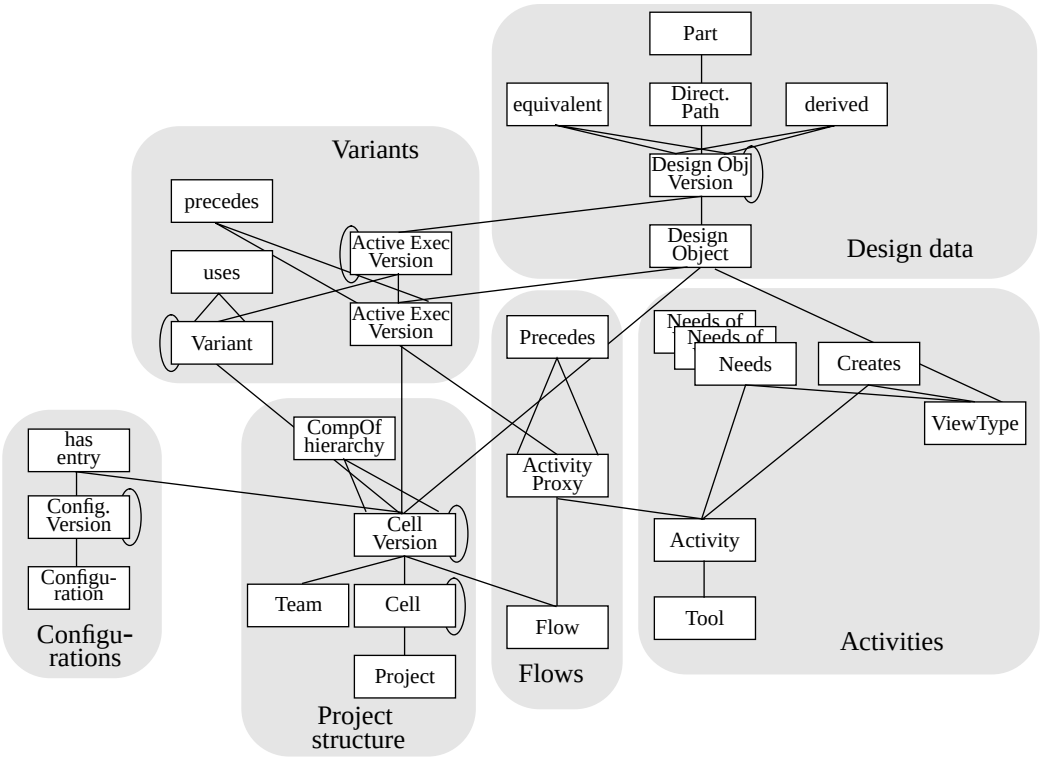

Figure 1. Information architecture of JCF 3.0 (in OTO-D format [Bekk92]) [vdH94]

structure of the stored data by an appropriate interface is not possible.

2.2 FMCAD Architecture

In contrast to JCF, the FMCAD stores the design data not in a common database, but in libraries. The library consists of a UNIX directory and the related *.meta*-file describes the contents of the directory (metadata). The logical data objects are named *cells*, *views*, *cellviews*, *cellview visions* and *configuration*s. Figure 2 gives a brief description of the architecture. 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 *cellview*s.

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 application, 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 more a logical than a physical design object.

A *cellview version* is the data file of a *cellview* at a particular time. *Version*s are created by checkout/checkin operations. A *cellview* can have more than one *version*. The *version* models the link to the design file in the file system.

A *configuration* is a collection of *cellview version*s that are related. For each *cellview*, at maximum one *version* can be part of the *configuration*.

The FMCAD tools run on top of the framework and each part of the system can be modified by an extension language. FMCAD provides all necessary interfaces and inter-tool communication (ITC), e.g., cross-probing between the schematic editor and layout editor. The viewtype concept is very flexible and it allows viewtypes to be easily switched with the same tool.

In FMCAD it is possible to have multi-user access to a library. However, the concurrent access to a *cellview* object is controlled by a checkin/checkout model. 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. This means that only one user can change a *cellview* at a time. It is not possible for two users to work on two different *version*s of a *cellview* in parallel. The refreshment of the metadata objects is not performed automatically, and therefore, it is the responsibility of the designer to keep his design up to date. Of course, this aspect may cause severe locking problems during the design process.

FMCAD supports non-isomorphic hierarchies because the hierarchies depend on the *viewtypes*. The dynamic hierarchy binding without storing what belongs to what relationships (by always using the default version of a *cellview*) is difficult to handle because the history of the development is not stored. In contrast to JCF, the metadata (hierarchy relationship) are not stored separately, and are part of the design data.

2.3 Data Model Mapping

After the presentation of the different framework information architectures, the mapping of the data model has to be performed. The starting point is the decision which framework should be the master and which should be the slave in the coupling environment.

Due to the fact that the JCF framework offers much more efficient concepts in the area of design management and multi-user support, and FMCAD efficiently supports the incorporated design tools, it was decided that JCF was to be the master.

To summarize the possible mapping of the information models, Table 1 shows the current mapping strategy. For example, the FMCAD's *Library* object can be mapped on the JCF *Project* object.

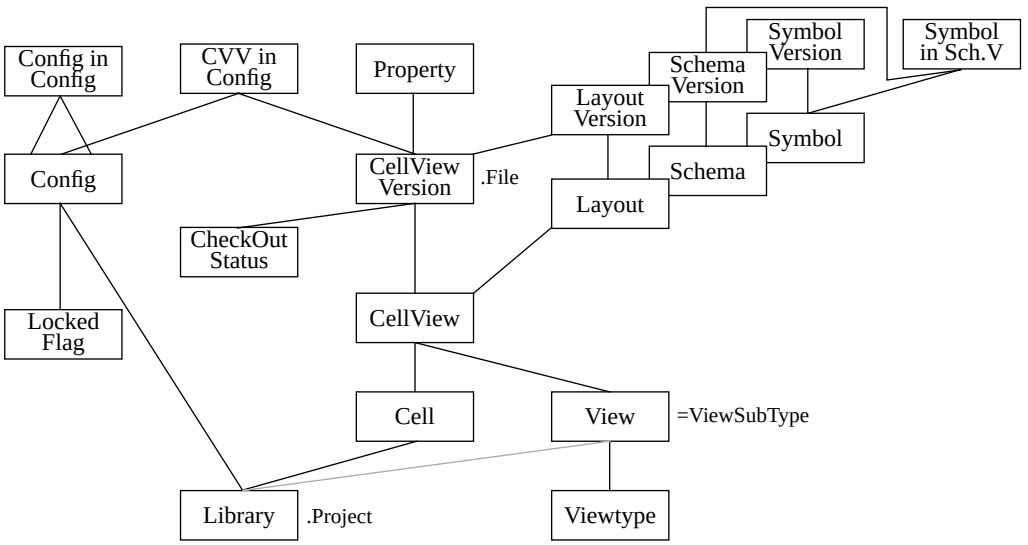

Figure 2. Information architecture of FMCAD [vdH94]

Hierarchy is a key feature in VLSI design. Here, the two frameworks offer conflicting approaches: while in FMCAD design hierarchies are specified within the design files, JCF uses separated metadata that must be submitted manually via the JCF desktop. For the common JCF-FMCAD environment, this implies that first the complete design hierarchy information has to be defined and passed to JCF. In this case, JCF can completely control the data consistency of versioned hierarchical designs.

JCF object	FMCAD object
Project	Library
CellVersion	Cell
ViewType	View
DesignObject	Cellview
DesignObjectVersion	Cellview Version

Table 1: JCF - FMCAD mapping

Unfortunately, the usage of hierarchies is limited by the problem of non-isomorphic hierarchies, since, for example, the hierarchy of the *viewtype* schematic can differ from the hierarchy of the *viewtype* layout. Since, in contrast to FMCAD, JCF 3.0 does not yet support non-isomorphic hierarchies, these kinds of hierarchies are not supported by the current hybrid JCF-FMCAD environment.

2.4 Encapsulation

The current JCF-FMCAD encapsulation scenario contains three different FMCAD tools:

- Schematic entry tool
- Layout entry tool
- Digital simulator tool.

Since each tool is modelled by one JCF activity, JCF records all derivation relationships between schematic and layout versions. In the new JCF-FMCAD environment, the flow management features of JCF are available to support the designer's work. Due to the closed interfaces of JCF, FMCAD's ITC could not be used normally. Special wrappers and additional software helped to reduce potential drawbacks in the current environment and enabled activity execution when its predecessor was not yet finished and guaranteed consistency by additional windows which appear in the FMCAD tool environment. The customization of the encapsulation was extended by several extension language procedures to trigger functions and lock menu points in order to prevent data inconsistency.

3 Results

In order to discuss the main achievements of the hybrid JCF-FMCAD framework approach, the evaluation results are summarized below. The basic criteria cover technical criteria, design management services and performance aspects.

3.1 Multi-User Design and Concurrency Control

If IC designs are composed of several JCF cells, the standard multi user capabilities of JCF can also be used in the JCF-FMCAD environment. Using the JCF workspace concept, individual cells can be reserved and isolated from access by other users. Standard FMCAD does not provide this flexible method of controlled data access: there is only one *.meta* file per project, and the access of all designers working concurrently in this project has to be explicitly coordinated. This may cause severe locking problems. While in FMCAD parallel work on different versions of the same *cellview* is not possible, the JCF-FMCAD framework provides this feature. Not yet possible in JCF or in the combined framework is data sharing between projects. It would be helpful to also provide access to cells of other projects.

3.2 Design Management and Data Consistency

Compared with FMCAD, JCF-FMCAD provides a more flexible design management functionality. For instance, FMCAD offers a rather simple versioning mechanism, while JCF-FMCAD provides a two-level versioning approach: versioning of *cells*, and versioning of *design objects* (within a cell). A second example addresses hierarchy information that is stored in JCF metadata, while in FMCAD this information is hidden in the design files. This results in a more powerful data consistency check in JCF-FMCAD. Another essential new design management feature would be the possibility to control the entities: *users*, *teams*, *tools* and *flows* and their relationships. These entities cannot be distinguished within FMCAD.

3.3 Handling of Design Hierarchies

Handling of design hierarchies was one of the most difficult tasks during the JCF-FMCAD encapsulation. It is important to note that FMCAD is very flexible in design hierarchy manipulations, but this feature is compensated by poor consistency control of versioned hierarchical designs. Here, JCF provides much better support. The existing JCF-FMCAD prototype requires that all hierarchical manipulations must be done manually via the JCF desktop before the design is started. In the future, this drawback could be overcome by a JCF procedural interface which might be used by the design tools to pass the hierarchy information to JCF. However, JCF release 3.0 does not support this feature. Another major problem is caused by non-isomorphic hierarchies, i.e., functional and physical hierarchies differ. This feature will be supported in future releases of JCF.

3.4 User Interface

In the existing JCF-FMCAD prototype system, the designer has to work with both the FMCAD and JCF user interface. In spite of the fact that JCF conforms with the major user interface standards (using X-Windows and Motif) and some kind of intuitive *look&feel*, the user has to cope with an extra user interface.

3.5 Flow Management and Derivation Relations

Standard FMCAD does not support flow management capabilities. Since the FMCAD user can invoke all design tools in a very flexible manner, related data management cannot be provided, and therefore, neither derivation relations nor the what-belongs-to-what information is available. Since JCF provides all these features, the combined JCF-FMCAD framework provides a solution, but in a slightly restricted way, since a designer is forced to use design-specific flows. These flows can only be defined and changed by the project manager, and the specified order in which tools can be executed is prescribed and fixed for the designer. This may lead to some acceptance problems on the one hand, but on the other hand forced design flows can be used to ensure quality aspects by forcing the successful execution of the required tools.

3.6 Performance

This aspect is of less importance since the main aspect of the inter-framework coupling is functionality. The performance of metadata operations in the JCF-FMCAD environment is sufficiently high. These operations are based on the JCF desktop methods. For design data manipulations the performance is strongly dependent on the amount of data: While the time delay for small designs is acceptable, more complex and realistic designs may cause problems, mainly due to the fact that design data have to be copied to and from the JCF database even in the case of read only accesses.

4 Conclusion and Future Work

With the described hybrid JCF-FMCAD framework prototype it could be demonstrated that a combination of frameworks with different basic architectures and individual design support services can provide enhanced functionality. The most important resulting features are the support of concurrent engineering and design management, basic services of JCF, and the flexible handling of design hierarchies, one of the main advantages of the standard FMCAD environment. In the JCF-FMCAD system, the latter feature could only be basically implemented due to the restricted JCF hierarchy management. The other stated goals could be reached, and therefore, the described connection between the JCF and the FMCAD framework improved the functionality of the stand-alone and isolated systems.

Acknowledgement

The authors would like to thank Hans Kok and Kees Lepoeter (Philips Research, NL) for their constructive discussions and important contributions to this work and the prototype implementation.

References

[Alle91] W. Allen, D. Rosenthal and K. Fiduk “The MCC CAD Framework Methodology Management System” *28th DAC*, pages 694–698, 1991.

[Bekk92] J.H.ter Bekke *Semantic Data Modelling* Prentice Hall, Englewood Cliffs, 1992.

[Bosc91] K.O.ten Bosch and et al. “Design Flow Management in the Nelsis CAD Framework” *28th DAC*, pages 711–716, 1991.

[Bret91] F. Bretschneider, Ch. Kopf and M. Zolg “Process Control for Large Complex Systems” *Winter Simulation Conference*, 1991.

[Cade] Cadence release 4.2.2 documentation *Cadence Design Framework II Reference*. CADENCE.

[Kash92] Y. Kashai “Flow - A Concurrent Methodology Manager” *EDAC*, pages 20–24, 1992.

[KC92] E. Kwee-Christoph, B. Eschermann, O. Haberl, R. Kumar and A. Kunzmann “The CADEC VLSI Design Support Methodology” *CompEuro*, 1992.

[Lieb92] D. Liebisch and A. Jain “JESSI-COMMON-FRAMEWORK Design Management – The Means to Configuration and Execution of the Design Process” *Euro-DAC*, pages 552–557, 1992.

[Meck92] A. Meckenstock and D. Zimmer “Concept and Architecture of a Distributed Object-Oriented Database Kernel” Technical Report Cadlab Report 13/92, CADLAB Paderborn, 1992.

[Over] JCF 3.0 System Overview. *Documentation belonging to JCF release 3.0*. ESPRIT Project 7364.

[Pari94] S. Parikh, D. Sarnoff and M.L. Bushnell et al. “Distributed Computing, Automatic Design and Error Recovery in the ULYSSES II Framework” *EDAC*, pages 610–617, 1994.

[Quel94] T. Quellenberg, J. Schubert and G.v. Bueltzingsloewen “FLOPPY - A Flow Management for On-Line Processes in Production Systems” *IMSE European Workshop on Integrated Manufacturing Systems Engineering*, 1994.

[Ramm92] F. J. Rammig and B. Steinmüller “Frameworks und Entwurfsumgebungen” *Informatik-Spektrum (1992) 15*, pages 33–43, 1992.

[Seep94a] R. Seepold and A. Kunzmann “Basic Requirements for an Efficient Inter-Framework-Communication” *4th International Conference IFIP WG 10.2 Working Conference on Electronic Design Automation Frameworks*, Nov 1994.

[Seep94b] R. Seepold, J. Schubert, 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 and FOF*, pages 680–685, Aug 1994.

[Stei92] B. Steinmüller and et al. “JCF-Start-up Phase Final Version” Technical Report JCF/CADLAB/053-02/01-June-92, Internal Report, ESPRIT Project 7364, Jun 1992.

[Stra94] M. Straube, W. Wilkes and G. Schlageter “HANDICAP - A System for Design Consulting” *DAC*, 1994.

[vdH94] P. van den Hamer, H. Kok and K. Lepoeter “Private Communication” Internal Discussion, 1994.