An Hierarchical Schematic Editor to WWW

Lisane B. de Brisolara, Leandro S. Indrusiak, Ricardo A. da Luz Reis
{lisane,lsi,reis}@inf.ufrgs.br

Universidade Federal do Rio Grande do Sul – UFRGS
Cx Postal 15064 – Av. Bento Gonçalves, 9500 – Campus do Vale – Bloco IV
Bairro Agronomia – Porto Alegre – RS – Brasil – 91501-970

Abstract

This paper presents the research and development of a hierarchical diagram editor, called BLADE (Block and Diagram Editor), accessible over internet-like networks and included into the Cave2 Framework [IND 98]. This tool will be used as a graphical interface between the user and the design environment, providing a visualization of the design in different abstraction levels hierarchies. This work also shows a new approach for the data structures behind the diagram editor, based in a separation between design data and visualization data.

1 Introduction

A schematic editor is a tool mainly used during the design of a VLSI system for system specification. The approach based in schematic capture was widely adopted by IC designers. However, nowadays designers are using HDLs (Hardware Description Language) or other high level approaches to specify a complex design [KUR 97]. There is a strong tendency that the circuit specification, which nowadays occurs in HDL in textual mode, will should be able to be done graphically. This tendency shows the possibility of utilization of a new kind of schematic edition tools, the diagram editor, in complex IC design. The diagram editor can be considered as an extended schematic editor that can handle different kind of diagrams and abstraction levels.

In the specification of a complex system it must be used the an hierarchical approach of functional blocks. This blocks can be seen as boxes that can be specified in a higher or lower abstraction level depending on the its complexity degree. A hierarchical diagram editor can allow a graphical description of complex system using functional blocks. Then, these blocks are described in a logic level or RTL level. In the current frameworks, the schematic editor main function is visualization. It allows a hierarchical view of a complex circuit lets a graphical vision of a circuit in different abstraction levels or using different graphical/textual representations. A schematic editor can supply a visualization of a circuit generated by synthesis tools. This graphical vision allows a control of the design flow, allowing the visualization of each refinement done during the design process.

Nowadays, system-on-chip designs are composed by custom blocks and by IPs (intellectual property), allowing the design of complex systems in a short time. The use of diagram editors in the design of systems using IPs is a necessity [BER 00]. This tendency is

one more motivation for research and development of new schematic editors that can handle different abstraction levels and its graphical views, facilitating the integration among them.

The BLADE (Block and Diagram Editor) tool is a diagram editor to be inserted in CAVE environment to allow the construction of the diagram representation of complex IC allowing visual specification. CAVE is an IC design framework based in World Wide Web. Its main objective is to integrate CAD tools and to automate a distributed IC design flow [IND 98]. In CAVE project, it is also being researched the insertion of functions for collaborative work, using JINI and JavaSpaces [SAW 01] [HER 01]. Blade is under development considering the specification of CAVE's data persistence and collaboration services. The Java language is being used in BLADE development. Some advantages on using Java are platform independence and applications integration facility in WWW, because the applications are developed as *applets* to be integrated to hyperdocuments and visualized with web browsers.

2 Schematic Editor Functions and Requisitions

A diagram editor allows a diagram construction using graphical basic primitives, as rectangle, circle, lines, etc. This kind of editor must have features like zoom, rotation and must allow global variable definition by the user such as colors, grid, etc. These features also must be implemented in a schematic editor.

A schematic editor usually allows the circuits edition in logic level, using logic gates stored in libraries or using functional blocks associated to hierarchical principle. Such tool works with some basic primitives as logic gates, connections, blocks, ports, etc. These basic primitives can be inserted, moved and removed in to/from schematic as in other kinds of diagram editors. However, there are some particularity related to this kind of diagram editor. The connections in a diagram must be strongly linked to every component, to make sure that when a component is moved, every linked connections to it will be also moved to insures schematic consistency.

After insertion and interconnection of blocks, it must be defined the correspondent semantics of each block in different abstraction levels.

When the graphical specification of a design is done, a *netlist* description must be generated in a default format order to be used in others tools, for example the EDIF (Electronic Design Interchange Format) format.

3 Blade Development

The first step on the development of Blade was the definition of its underlying software architecture, through the definition of a set of object classes. An important design decision was the division of the diagram data in two main classes hierarchies: design data and visualization data. The first, represent the needed data to generate the *netlist* descriptions and reference the application's specific data. The second, represent the data used to generate diagram views. This approach has as main objective to allow editor remote operation and the insertion of collaborative work which represent the one of the features of the CAVE Project. This practice is very used in software engineering because it allows the reusability and facilitates system maintenance and understanding.

This approach is to allow collective classes relations without losing reusability. Several toolkits used to construct of user graphical interfaces separate the interface presentation aspects from the application data. Then, the classes that define application and representation data can be re-used independently [GAM 00]. The separation of visualization and design data allows the modeling of graphical representation and functionality representation through different objects to allow different visualizations by different designers.

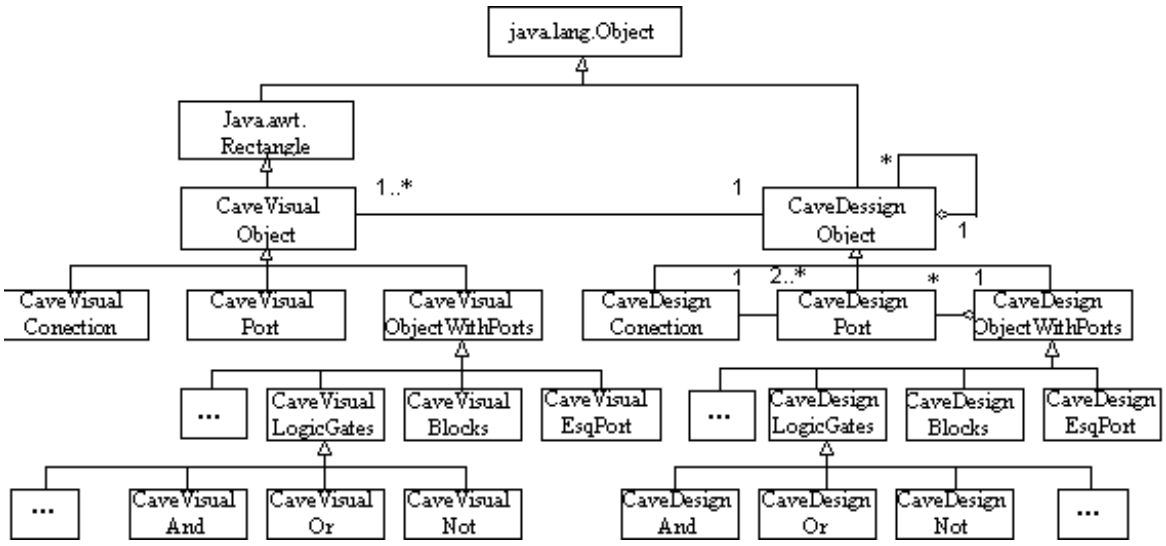

Figure 1 - Blade classes hierarchy.

In the CAVE environment, other graphical editors were already implemented using Java, such as JALE/JALE3D [GRA 99] [OST 01]. Some of the building blocks of the Cave Framework were modified to allow the development of this diagram editor tool. Classes from JALE are used, specially those which define graphical functions and data structures of layout editor, minimizing BLADE's developing time. The figure 1 presents the BLADE´s main classes. The *CaveVisualObject* class is utilized in JALE tool and was modified to be used in BLADE. This class extends *java.awt.Rectangle* to every derived objects circumscribed in rectangle area, turning consistent these objects. There are methods responsible for drawing, rotation, selection, adjust of envelop and other functions to allow diagram edition and visualization in *CaveVisualObject* class. Every edition operation occur through the *CaveVisualObject* objects and this objects makes the interface between users and *CaveDesignObject* objects. The *CaveDesignObject's* sub-classes stores information associated to circuit functionality, as gates, blocks and connections that compose the circuit design, as such as information associated to blocks functionality. There is an association between *CaveDesignObject* and *CaveVisualObject* that let to work with several views of a design. A *CaveDesignObject* object can be composed of some *CaveDesignObject* objects, this relationship gives the possibility to work in a design with different abstraction levels and hierarchy. The *CaveDesignObjectWithPorts* objects has one or more *CaveDesignObjectPort* objects, turning possible to work with functional blocks and logic gates with variable ports number. *CaveDesignConection* object is composed of interconnections between

CaveDesignPort objects. This is the structure under the BLADE tool which will be able to connect objects graphically and to generate a description *netlist* of diagrams and to manage the use of different views and abstraction levels in a design.

4 Conclusions

This paper presented a brief report on the research and development of an extended schematic editor called BLADE (Block and Diagram Editor). This diagram editor allows hierarchical schematic edition and visualization, and the extension of those representations to others abstractions levels. The main advantages offered by this tool is platform independence, collaborative work and the use of remote operation.

An object-oriented approach and software design patterns are being used in BLADE development, which allow the reusability of classes and code. It is used an approach to allow the separation of design semantic and its graphical representation and making easier the insertion of collaborative work.

Firstly, the BLADE tool will work with logic gate schematics, but the objective is to generate a generic diagram editor tool which can be extensible to handle other diagrams, representing different levels of abstraction.

5 References

[BER 00] BERGAMASCHI, R. A. & LEE, W. R. **Designing System-on-Chip Using Cores**. In: Design Automation Conference, 37th. Los Angeles, California, 2000.

[GAM 00] GAMMA, E. et al. **Padrões de Projeto: Soluções Reutilizáveis de Software Orientado a Objetos**. Bookman, Porto Alegre, 2000.

[GRA 99] GRALEWSKI, D.; WINCKLER, M.A.A.; INDRUSIAK, L.S; REIS, R.A.L. **JALE Layout Editor**. In: Proceedings of the XIV Microeletronics Seminar. July. 09 -10, 1999, p. 51-54.

[HER 01] HERNANDEZ, E.; SAWICKI, S.; INDRUSIAK, L.; REIS, R. **Homero – Um Editor VHDL Cooperativo via Web**. In: Workshop Iberchip, VII. Montevideo, Uruguay, March-2001.

[IND 98] INDRUSIAK, L & REIS, R. **Ambiente de Apoio ao projeto de Circuitos Integrados baseado na *World Wide Web***. Master Thesis. UFRGS, Porto Alegre, 1998.

[KUR 97] KURUP, P. & ABBASI, S. **Logic Synthesis using Synopsys.** Kluwer Publishers, 1997.

[OST 01] OST, L.C.; MAINARDI, M.; INDRUSIAK, L.S.; REIS, R.A.L. **Jale3D - Platform-independent IC/MEMS Layout Edition Tool.** In: XIV SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 14., 2001, Pirenopolis. Proceedings... Los Alamitos: IEEE Computer Society Press, 2001. (to appear)

[SAW 01] SAWICKI, S.; REIS, R. **Uma Proposta de Trabalho Cooperativo baseado em Web para Concepção de Circuitos Integrados**. In: Semana Acadêmica. UFRGS, April, 2001.