Design and Realization of Simulation Environment of Embedded Software and Hardware Intergration Based on GEF

Bo Geng, Qinghua Cao
Department of Computer Sciences, Beihang University
Beijing, 100191, China
E-mail: gengbo8@gmail.com, caoqinghua@buaa.edu.cn

***Abstract*—Embedded software and hardware integration simulation platform is developed for simulating the embedded systems design process in current engineering system, which can facilitate finding various problems in system design process. For example, in the system scheme phase, the scheme and design verification are untimely and inadequate. In the early prototype phase, software development lags behind result in deferral of the overall progress of the system. And in the late prototype stage, the problem is lacking configuration item test environment. Embedded software and hardware integration simulation platform can provide verification of hardware and software integration and test development environment. Therefore, the quality of software development in embedded systems can be significantly improved and development cycle can be remarkably shortened by using this simulation platform.**

Keywords-component; Embedded Software; Simulation; Graphical Editor Framework ; Software and Hardware Combination

I. INTRODUCTION

This paper aims to introduce a set of embedded hardware and software integration simulation platform which can validate, compare and determine a variety of design plans of the subsystem. This platform also provides a development environment for parallel development of hardware and software in the subsystem developing process. On the whole, the simulation platform we introduce can supports both the verification of embedded software configuration item testing and extending to larger scale system simulation[1] .

The current domestic and international research in this area is mainly focused on the hardware simulation or software simulation separately. I have combined both of them. This paper consists of four parts. The second part is about related work. The third part is about the system design and implementation. The last part is the experimental results and summary.

II. RELATED WORK

The key technology consists of three parts, the plug-in development, SWT (Standard Widget Toolkit), JFACE and GEF (Graphical Editor Framework). We will introduce the three parts in detail respectively in the following sections.

A. The Plug-in Development

The unique plug-in architecture of Eclipse is the most attractive aspect to developers. This architecture allows developers to add functionality to the platform by adding the predefined extension points. Extension points are just the interface the plug-in providing. The ECLIPSE platform's workbench components contain some extensions. For example, it allows the plug-in to extend eclipse user interface, making the user interface include menu choices and toolbar buttons, request notification of different events, and create a new view. Workspace component includes extension points allowing users and resources (containing the project and file) to interact with each other. Figure 3 shows the eclipse workbench and workspace's architecture.

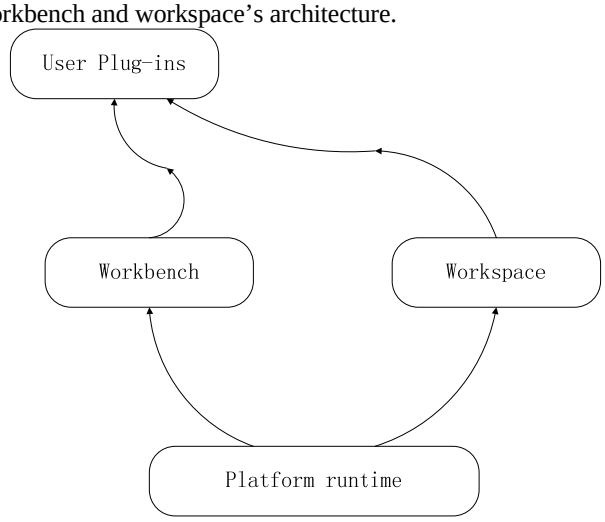

Figure 1. Eclipse workbench and workspace's architecture.

The plug-in is a kind of structured component which uses OSGI manifest file (MANIFEST.MF) and plug-in manifest file (plugin.xml) to describe itself to the system. The platform is responsible for the maintenance of the registry of installed plug-ins and the functions they provide.

B. SWT/JFACE

There are three main graphics libraries in JAVA-based desktop application development. They are AWT (Abstract Window Toolkit), Swing, and SWT [2]. Desktop applications written in the first two libraries are unaesthetic, inefficient in implementation, and low response speed. SWT can overcome the shortcomings of AWT and SWING mentioned above. UI written by SWT is far beyond AWT and Swing of SUN in terms of response speed and beautiful interface. Its rich components help programmers develop fully functional UI program. This is mainly because AWT simply simulates the local operating system's window

Published by Atlantis Press, Paris, France.

components. Nevertheless, SWT maximizes the graphical components API of the operating system which means that as long as the operating system provides corresponding graphical components, SWT can call them using the Java Native interface (JNI). SWT will simulate implementation only when the operating system does not provide corresponding components. The advantage of SWT using JNI technology is that SWT is closely connected with local operating system. Therefore, UI written with SWT have almost no difference with the local operating system window. At the same time, SWT library reflects the local operating system's basic widgets. In many circumstances, the level of this approach is much lower than our expectation. Hence, we use JFACE library as SWT's enhanced library. JFACE is the expansion of SWT. JFACE library's components will not interact with the local operating system. The relationship among operating system, JNI, SWT and JFACE is shown in Figure 4.

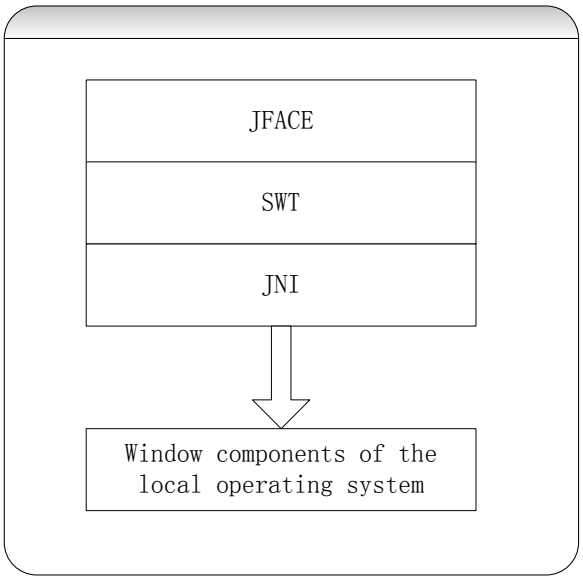

Figure 2. The relationship among operating system, JNI, SWT and JFACE

C. GEF

GEF (Graphical Editor Framework) is a graphical editor framework, which allows developers to graphically display and edit the model for enhancing the user experience [3]. There are many similar applications, such as UML class diagram editor, graphical XML editor, interface design tool, graphical database structure design tools, and so on. GEF has the following common features in the graphical editor aspect.

- Provide an editing area and a tool bar. Users select a tool in the tool bar. Then place the node or connection in the editing area by dragging or clicking them.
- Node can contain child nodes.
- Users can view and modify most of the properties of a node or a connection.
- Connection endpoint is fixed in the node.
- Provide context menu and keyboard commands.
- Provide zooming function of the diagram.
- Provide outline view to display the thumbnails of the editing area, or tree model structure.
- Support for undo and redo function.

As a conclusion, showing model based on RCP (Rich Client Platform) separately is enough already. If we want to make graphics have connectivity and editing features, GEF is a very suitable choice.

III. System Design and Realization

The main contribution of this paper is setting up an embedded simulation system combining hardware and software based on existing technology. The whole framework of our design is, shown in Figure 1. As shown in this figure, the software part of the embedded hardware and software integration simulation platform is divided into client software and simulation control software, running on the client computer and simulation unit control computer respectively [4]. This paper mainly focuses on client software development. In our system, a PC or workstation is used as clients whose operating system is Windows XP. It implements configuration management, simulation process control, data processing and other functions.

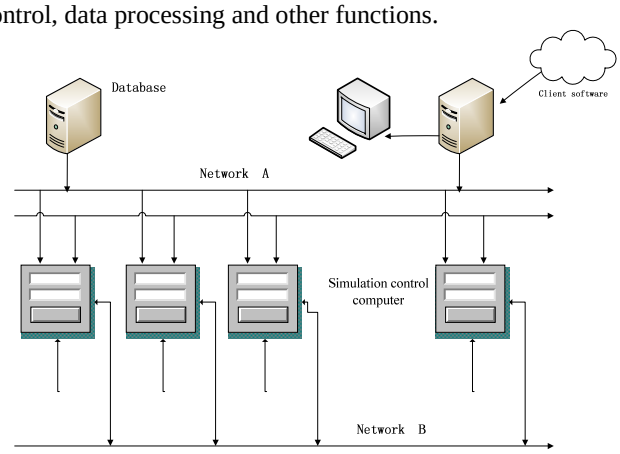

Figure 3. System architecture

As shown in Figure 1, our system consists of a client computer, several simulation control computers, and a database server. All the nodes in the system connect to each other by two Ethernet.

In the system configuration process, client computer sends configuration data to each simulation control computer through network A. When a simulation runs, all simulation control computers accept simulation commands and collect state data generated in the simulation process. In the data processing stage, the master software reads various data generated in the simulation process from the database by network A. In the same time, simulation control software collects generated data through network A. All the data collected is saved in the database. In a simulation process, communication equipment simulation module of the simulation control software will sent packets from the target processor to the destination node after grouping these

packages to IP packets. When simulation control software receives the communication simulation data by network B, the corresponding communication device emulation module sends the data to the soft core of the device in the simulation unit after unpacking the simulation data. Eventually, the target processor will read the data [5].

A. *Client Software Design*

The structure of the client software is shown in Figure 2, which mainly consists of five modules: simulation design module, model library management module, compilation module, simulation execution module, and integrated information management module.

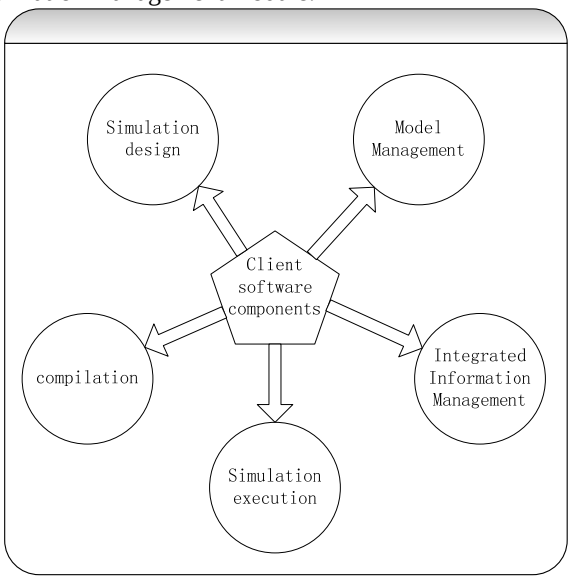

Figure 4. Client architecture

The simulation design module includes three parts: hardware resources access, four level design and multi-project simulation supporting. After starting the client software, the user will first rotationally query its own network segment .Then the client software will get the hardware resources shown in the interface. The model library management module includes two parts, the system model library management and custom model library management. Custom model library can add, delete and modify the properties and appearance of the self-defined model. System model library is very similar to custom model library except for deleting function. The compilation module calls a batch consisting of QUARTUS commands to execute the compilation process, which can generate the configuration files. Simulation execution module performs dynamic simulation process by changing the color and the state of the component. Integrated information management module includes user management and simulation playback. The simulation playback displays useful information by calculating simulation data queried from database.

B. *Code Realization*

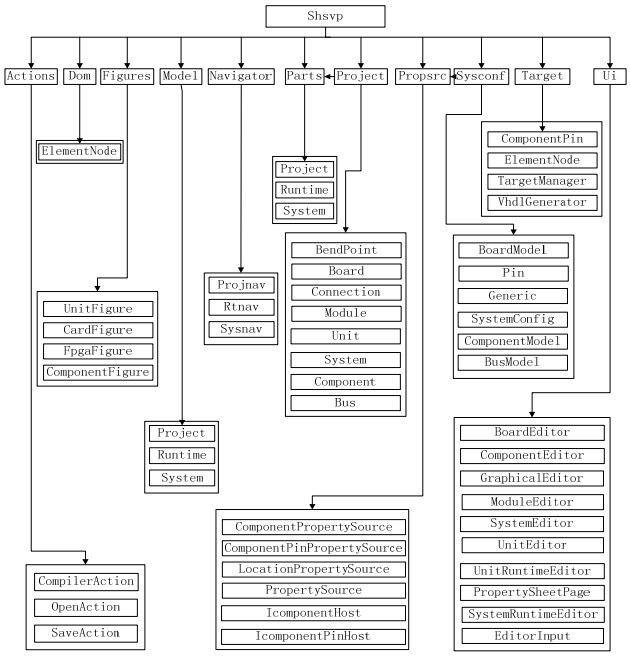

Figure 5. The architecture of the code implementation

The system uses Eclipse as the development IDE and Java as the programming language [6]. From the point of view of code implementation, the client software can be divided into 11 modules as shown in the Figure 5. Actions module mainly consists of Compile-Action, Open-Action and Save-Action. They constitute the implementation of the corresponding option in the menu bar and toolbar, such as compiling, opening the project and saving the project.

Dom module only has an ElementNode class, which is primarily responsible for maintaining reading and writing of the project file. At the same time, the class maintenances a hash table about property. Only when users perform saving action, ElmentNode will synchronize the contents of hash table to the project xml file [7].

Figures, Model, Parts constitute the three components of the MVC pattern of GEF. Figure module mainly designs the appearance and colour of the device and unit. Model module mainly maintains the properties of the model by ElementNode. It is the sources as the list of attributes, will be registered as a listener to the list of attribute modify listener. Parts module is the core module of MVC, which is the bond of the model and the view. All the operations to views by user will be synchronized to model attributes through the controller [8]. At the same time, the modification of the model attributes will inform the controller. Then the controller changes the appearance and shapes corresponding view.

The Navigator module is responsible for the navigation function. It contains three parts, as shown in the Figure 5. The Project part is responsible for the navigation of the

current project [9]. Runtime provides a navigation view of simulation state after the start of the simulation. System is the navigation of model library management. Users can add the properties of the model and configure the appearance or shape the model by right-click the menu.

Project and Sysconf are both related to storage. There will be an ElementNode corresponding to every xml node in the project file. Then all operation of the project will be reflected to the ElementNode. The changing will be written to the project xml file only when the save button is clicked. Sysconf module maintains another separate xml file SystemConfig.xml. The file stores all resources in use, including bus model, device model and so on. Client can use Sysconf module to load the configuration file, which will determine what resources are available.

Propsrc module provides management functions of custom model. ComponentPinPropertySource defines the interface of the device pins, including the size and location information of the pins. ComponentPropertySource provides defined interfaces of device's attributes, including the size, shape, type, icon, and the corresponding VHDL source files of the device. The remaining classes are some interfaces and abstract classes assisting users to design the custom model.

Target is the compilation function module, which mainly consists of two classes. VhdlGenerator is responsible for writing the statement and main part of VHDL by directly manipulating the VHDL source file. TargetManager is responsible for automatically generating required VHDL code based on the connection relationship of the models [10].

The Ui module is mainly constituted by the editing areas. The editing areas involved in simulation design stage include the system editing area, unit editing area, board editing area and module editing area. And the editing areas involved in simulation execution stage include SystemRuntimeEditor and UnitRuntimeEditor. PropertySheetPage is the box to show the list of all properties of models.

IV. Experimental Results

Simulation platform already has the whole function from the simulation design to the simulation results review. This platform has passed the evaluation of a third-party evaluation centre, which reaches the standard of acceptance by the relevant departments. Below is the effect figure of the client software. Figure 6 shows the chip-level design. A CPU can be seen in the editing area associated with a 1553B and CAN model. In the editing area, there is a drawer on the left, which has system models and tools to provide connection function. We can connect the components in editing area by dragging the appropriate model. Outside the editing area, on the left is a navigation tree, which displays the nodes from subsystems-level to model-level. On the right is the property box, users can edit selected model properties. On the bottom is an output area, which can output the message of the compilation process. On the top are menu bar and toolbar, which provide many options and operation.

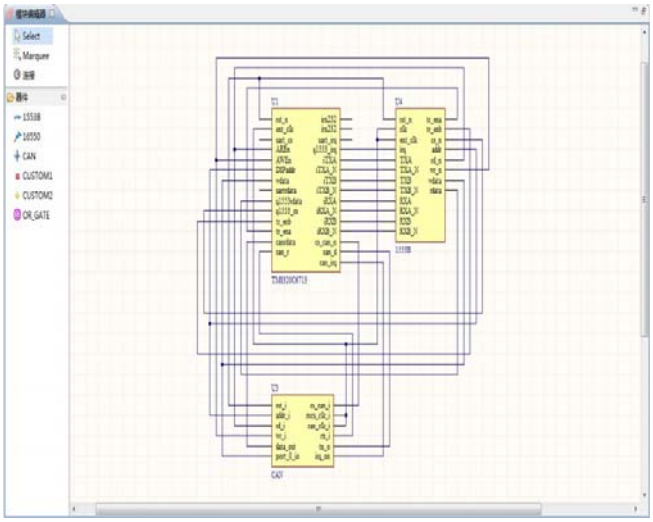

Figure 6. The interface of chip-level design

V. Conclusion

In this paper, a Graphical Editor Framework successfully combining platform of hardware and software simulation is developed. By this platform, the relevant departments can do the research and development of software when hardware conditions are not met. It greatly shortens the software development cycle, and overcomes the drawbacks of the software development lagging behind the hardware environment. What's more, a new direction of embedded software development is pointed out.

References

[1] Zhang Liang, Liu Bin, Lu Min-Yan.Embedded software test development environment framework design [J].Joumal of Beijing University of Aeronautics and Astronautics, 2005, 31 (3): 336-340.

[2] SWT. http://www.eclipse.org/swt/

[3] Gef. http://www.eclipse.org/gef/

[4] Y. Yi, D. Kim, and S. Ha, "Fast and accurate Cosimulation of MPSoC Using Trace-Driven Virtual Synchronization", IEEE Trans.Computer-Aided Design of Integrated Circuits and Systems, vol. 26, no.12, pp. 2186-2200, Dec. 2007.

[5] Byeongdo Kang, Young-Jik K won, A Design and Test Technique for Embedded Software, Software Engineering Research, Management and Applications, 2007 Fifth ACIS international Conference on. 160-165.

[6] Eclipse. http://www.eclipse.org/

[7] Java. http://www.java.com/zh_CN/

[8] Sung-Kwan Kim, Jongmoo Choi, Donghee Lee.Virtual Framework for Testing the Reliability of System Software on Embedded Systems.

[9] Wang Yichen, Liu Bin.Embedded software simulation testing environment architecture design. Computer Engineering and Applications, 2005.

[10] Byeongdo Kang, Young-Jik K won, A Design and Test Technique for Embedded Software, Software Engineering Research, Management and Applications, 2007 Fifth ACIS international Conference on. 160-165.