TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.3, March 2014, pp. 1799 ~ 1803
DOI: http://dx.doi.org/10.11591/telkomnika.v12i3.4468

MVx-modeling: A Modeling Environment for Embedded Software Design of Heterogeneous MPSoC

Zhai Wen-zheng*[1,2], Hu Yue-li[1]
[1]Shanghai University, School of Mechatronic Engineering and Automation, Shanghai, 200072, China
[2]Taizhou University, School of Mathematics & Information Engineering, Taizhou, 317000, China
*Corresponding author, e-mail: zhwenzheng@shu.edu.cn

Abstract

Owing to the complexity and flexibility in heterogeneous multiprocessor System-on-Chip (MPSoC) architecture, parallel programming has become a tough and urgent problem to solve. Model-based development methodology improves the software production efficiency and the reliability. Some model development tools such as Simulink and Rhapsody are widely applied in the development of embedded applications. But they do not satisfy the characteristics of embedded heterogeneous multi-core software. Thus developing a general modeling environment for model-based software design is necessary and meaningful. This paper designed and implemented a graphical modeling platform MVx-modeling by using the object-oriented programming technology and Visual C#.NET development language. Practical application shows MVx-modeling platform easy-to-use and efficient. It speeds embedded application development and lowers software costs for heterogeneous MPSoC.

Keywords*: modeling environment, model-based development, embedded software, heterogeneous MPSoC*

1. Introduction

Along with the advancement of the semiconductor manufacturing and the in-depth research of the processor architecture, the heterogeneous MPSoC has been widely used in the embedded system, typically as Sony-Toshiba-IBM Cell Broadband Engine (Cell BE) [1] for consumer electronic products and TI OMAP [2] for cellular phones.

Owing to the complexity and flexibility in heterogeneous multi-core architecture, multi-core programming has become a tough and urgent problem to solve. The traditional parallel development method is based on the use of language extensions such as OpenMP, MPI [3] parallel libraries, parallel programming language and parallel compiler, which have been proven inappropriate in embedded software development [4].

MDA methodology makes developers to focus modeling instead of the traditional coding concerning many platform details [5], improves the software production efficiency and the reliability of the software in embedded software design. Some commercial model development tools such as MATLAB/Simulink[6], LabVIEW [7] and Rhapsody are applied within the detailed design and coding phase of embedded applications. But they do not satisfy the characteristics of embedded heterogeneous multi-core software, the reason is neither they generate parallel programs based on customized embedded Real-Time Operating System (RTOS) nor check the schedulability of the generated software[8][9]. And they do not consider the resource constraints at the stage of code generation and provide verification capability of the generated parallel software [10].

Thus developing a general modeling environment for model-based software design of embedded heterogeneous MPSoC is necessary and meaningful. This paper designed and implemented a graphical modeling tool MVx-modeling by using the object-oriented programming technology and Visual C#.NET development language.

The remainder of this article is organized as follows: Section 2 presents the graphic modeling system structure. Section 3 outlines some key development technologies. Section 4 presents MVx-modeling environment prototype. Section 5 gives the conclusions with some future work.

Received August 25, 2013; Revised September 30, 2013; Accepted October 9, 2013

2. Module Partition

MVx-Modeling environment is a graphical productivity tool for engineers developing embedded applications based on the μc/os-Ⅱ RTOS. It provides a visual abstraction and design assistance in the model-based software development life cycle.

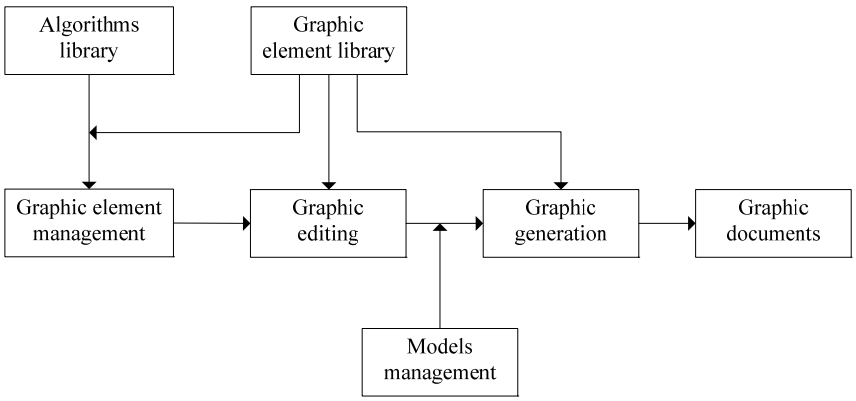

Figure 1. Graphic Modeling System Structure

Figure 1 shows that the modeling system structure is composed of graphic element management subsystem, models management subsystem, graphic editing subsystem and graphic documents subsystem. Graphic element is the fundamental unit of graphical modeling, is a graphical representation of the algorithm.

3. Key Technologies

MVx-Modeling represents the application and the architecture in graphical form with visual icons allowing user to model and build software system at a higher, intuitive level. Some key development technologies are put forward.

3.1. Object-oriented Graphics Class Structure Design

Figure 2 depicts the proposed object-oriented graphics class structure. In this section, base class BaseObj defines graphical elements' public attributes and virtual function operations such as copy, paste, delete, move etc. In each of the derived classes TaskObj, FunObj, LineObj, SemaphObj, the specific attributes and operating methods are defined in detailed.

Base class DrawTool defines mouse events virtual functions of click, move operation. Depending on the different graphic elements in the inheritance function, the corresponding graphic operation functions in a class are called to complete create, move and delete operation.

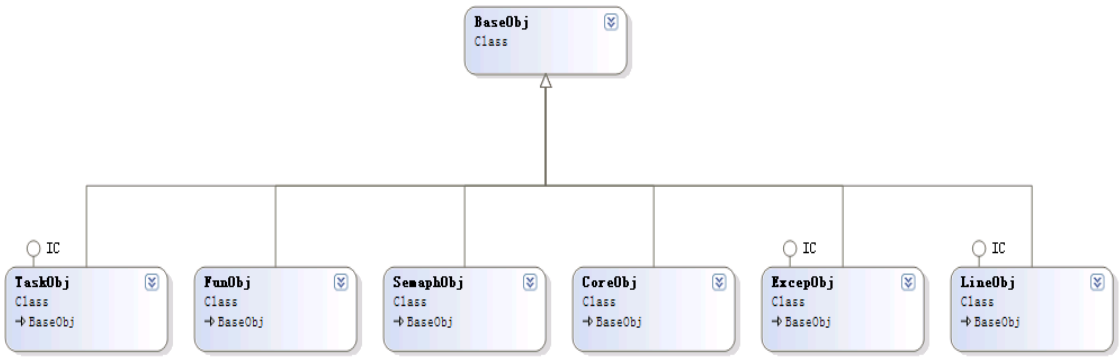

Figure 2. Object-oriented graphics class structure

3.2. Graphic Element Management

Graphic element is a graphical representation of the algorithm. A definition of graphic element aims at to establish the mapping relationship between graphic element and algorithm, the corresponding relationship between pins of graphic element with input/output in algorithm, to specify symbol file and to assign properties of graphic element etc.

In the application modeling, when the Task graphic element is newly created, the definition dialog box is activated to display as Figure 3. At the same time, 2 files named Task*.h and Task*.c are generated automatically in the project folder. Some properties such as name, description, task executors path, initial priority, type, size, cost are valued in detail.

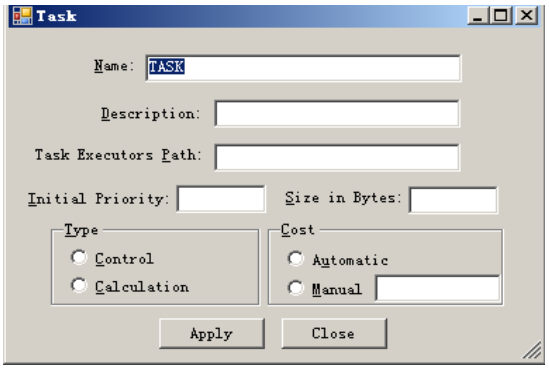

Figure 3. Task Graphic Element Definition

```
<?xml version="1.0" encoding="utf-8"?>
<MPSoC>
  <Cores>
   <processor name="arm926ej-s0">
          <index>0</index>
          <localMem name="lmad0">
                <addr>0x0</addr>
                <size>0x10000</size>
          </localMem>
          <shareMem name="shmad0">
                <addr>0x10000</addr>
                <size>0x40000</size>
          </sharedMem>
          <OS>
              <support>TRUE</support>
          </OS>
   </ processor >
   ……
  </Cores>
</MPSoC>
```

(a) Architecture part

```
<MPSoC>
  <Objs>
      <task name="SensorReader">
      <coreMap>0</ coreMap>
       <priority>9</ priority >
      <fileName>SensorReader_arm926ejs0.c</
fileName >
      </task>
  </Objs>
  <Objs>
      <semaph>
      <name>Semph_sensor</name>
      <src> SensorReader </src>
      <dst> LcdOutput</dst>
      </semaph>
  </Objs>
   ……
</MPSoC>
```

(b) Application part

Figure 4. An Example of the Model Files Design

As soon as double-click or right click the task graphic element, the dialog box is displayed to facilitate further modification. Other graphic elements are similar to the operation of the Task graphic element.

When the Semaphore graphic element is newly created, the system call functions of relevant event are produced and inserted into the main application file.

3.3. Design of Model Files

As is seen in Figure 4, the “architecture” section of the model files contains the hardware architecture information that is necessary to describe heterogeneous multi-core platform. The “application” section specifies kernel objects tasks and events used in the communication and synchronization requirements among tasks or exceptions.

The name of root node is MPSoC, the first level child node Cores, Objs are used to record all graphic element information in the application model and architecture model.

When the user operates graphic element in modeling area, the correspond nodes are linked certain list in the computer memory. When the file save command is executed, the list are saved as xml-style files at last

3.4. Picking Graphics

Picking up graph is the prime work of graphic element operation and properties editing and the accuracy and efficiency of picking up will directly affect the function of the graphics modeling environment.

When the user clicks on a graphic module, the corresponding processing procedure is activated to track mouse events. As is seen in Figure 5, firstly the picking graphic algorithm traverse each node in the linked list orderly to identify the clicked graphic element just before. Then highlight the graphic module by showing color outline border to declare "selected" state.

```
for (int i = 0; i < mlObj.Count; i++)
 {
   if ((mlObj[i].isHit(e.Location) == true))
   { mnObjSelectedIndex = i;
     break;
    }
}
```

Figure 5. Picking Graphics Algorithm

4. Implementation

As is seen in Figure 6, MVx-Modeling environment is divided into five distinct areas:

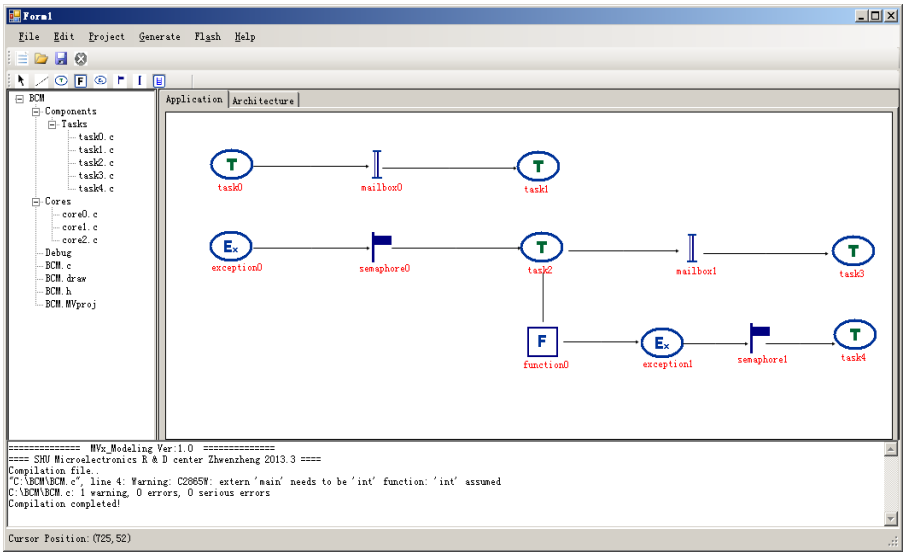

Figure 6. MVx-Modeling Environment

a) Main Menu: handles project files and text documents; edits layer diagrams; generates the parallel source code.

b) Graphic element libraries: drag-and-drop placement and editing of kernel objects and services such as task, exception, function, semaphore, mailbox, queue.

c) Workspace Window: offers a hierarchical view of the layers that compose the OS-based application. Each layer describes one portion of the application running in a certain processor of heterogeneous MPSoC.

d) Model editor area: presents kernel objects and their relationships. The nodes of the application model graph represent the kernel objects, whereas the connection lines indicate calls to the kernel services that handle these objects.

e) Output window: displays status message, such as the status of ongoing operations or the generation of the source files.

The example of application modeling In Figure 6 specifies five tasks, two exceptions, two semaphores and one function instance, the producer task (task3) which signals a semaphore (semaphore0), and a consumer task (task5) which consumes the signals. After properties description configuration, the application model file can be saved to support further model-based development process.

5. Conclusion

Embedded software development is a very challenging task in heterogeneous MPSoC architecture. Model-based development method improves the software development efficiency and maintainability, shorten the development cycle. In this paper, we research and develop a new modeling environment MVx-Modeling, easy-to-use graphical design tool, that speeds embedded application development and lowers software costs for heterogeneous MPSoC.

The next step is to generate optimized parallel software codes for each processor from the MVx-Modeling environment. The ultimate goal of our effort is to create a model-based embedded heterogeneous MPSoC software development environment consisting of modeling, simulation, verification, code generation and testing process.

References

[1] Blagojevic Filip, Feng Xizhou, Cameron Kirk W, et al. *Modeling multigrain parallelism on heterogeneous multi-core processors: A case study of the cell BE*. High Performance Embedded Architectures and Compilers - Third International Conference, HiPEAC. Goteborg, Sweden. 2008: 38-52.

[2] Texas Instruments. *OMAP5910 Dual-Core Processor Technical Reference Manual*. TI Technical Document SPRU602B. 2003.

[3] Zhang JingJun, Liu WenJuan, Liu Guang Yuan. Parallel genetic algorithm based on the MPI environment. *TELKOMNIKA Indonesian Journal of Electrical Engineering.* 2012; 10(7): 1708-1715.

[4] Wayne Wolf, Ahmed Amine Jerraya, Grant Martin. Multiprocessor System-on-Chip (MPSoC) Technology. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*: 2008; 27(10): 1701-1713.

[5] Soonhoi Ha. *Model-based Programming Environment of Embedded Software for MPSoC*. Proceedings of the 2007 Asia and South Pacific Design Automation Conference. 2007: 330-334.

[6] Tatsuya Kamiyama, Masayoshi Tamura, Takahiro Soeda, etc. An Embedded Control Software Development Environment with Simulink Models and UML Models. *IAENG International Journal of Computer Science*. 2012; 261-268.

[7] Gary W Johnson. LabVIEW Graphical Programming: Practical Applications in Instrumentation and Control. McGraw-Hill School Education Group. 1997.

[8] Wenzheng Zhai, Yueli Hu. A model-based development method of embedded software for energy HMPSoC. *Energy Education Science and Technology Part A: Energy Science and Research*. 2013; 31(2): 1129-1134.

[9] Wenzheng Zhai, Yueli Hu, et al. User guided programming methodology for embedded heterogeneous multi-core processor. China. 201210151037.5. 2012; 10.

[10] Zhang Zhiyong, Liu Jie, Zhang Xinhui. UML modeling for dynamic logistics system based on DEVS. *TELKOMNIKA Indonesian Journal of Electrical Engineering.* 2013; 4(11): 2168-2173.