Tunable Embedded Software Development Platform

施文彬 (Win-Bin See)
經濟部漢翔航空工業公司
台中市福星北路 68 巷 111 號 16-7
winbinsee@ms.aidc.com.tw

陳少傑 (Sao-Jie Chen)
國立台灣大學電機系
台北市羅斯福路四段一號
csj@cc.ee.ntu.edu.tw

熊博安 (Pao-Ann Hsiung)
國立中正大學資訊工程系
嘉義縣民雄鄉三興村 160 號
pahsiung@cs.ccu.edu.tw

李宗演 (Trong-Yen Lee)
國防大學中正理工學院電機系
桃園縣大溪鎮員樹林三元一街 190 號
tylee@ccit.edu.tw

摘要

由於新的功能需求以及新硬體的加入使得嵌入式系統的需求快速增加。軟、硬體並行開發可以縮短系統開發時間並降低嵌入式系統開發成本。我們提出一個"可調整的嵌入式系統軟體發展平台 (TESDP) "來協助分離軟、硬體開發，同時維持開發平台以及嵌入式系統平台間的相似性。TEDSP 在嵌入式軟體發展用的電腦上，提供一個讓嵌入式系統軟體可以執行的環境。我們為嵌入式系統軟體 (ESS) 的即時控制核心提供一套監視以及驗證資訊蒐集機制。ESS 與可調適的模擬硬體框架 (SHF) 在 TESDP 上合作以達成伴隨嵌入式系統軟硬體更新與演進的系統開發工作。此一方法，讓嵌入式系統軟、硬體開發可以同步進行。 我們展示將此一方法應用於一個車用嵌入式系統：模組式行動通信終端機 (MMDT) 的開發。MMDT 應用於智慧型運輸系統領域，提供資料通訊及全球定位功能，提供行控中心對於所屬車隊車輛的追蹤、派遣管理及協助。 TESDP 協助我們在硬體構型演進的同時進行軟體開發。 嵌入式系統市場產品如行動電話、個人數位助理 (PDA)、數位相機等等，需求非常旺盛。我們相信此一方法對於盼望在旺盛的嵌入式系統市場擷取商機的電子產業會有很大的助益。

關鍵詞：嵌入式系統軟體，物件導向式框架，即時系統，軟體驗證。

1. Introduction

Following the advances in the design and fabrication techniques for semiconductor devices, various micro-controllers and peripheral control chips are proliferating with decreasing price and increasing performance. These technology advancements have also enabled the development of inexpensive embedded systems that provide dedicated and integrated services. Mobile phones, digital camera and personal digital assistance (PDA) are examples of emerging embedded system applications. On the other hands, these kinds of embedded systems are suffered from having short life cycle time that have been caused by the changing appetite of customers and the introduction of new products from competitors. Hence, embedded system providers have to keep on developing new products based on new hardware components and new user demands in functionality and interface improvement, the embedded software will be the glue to all hardware components. To take advantage of cost reduction from mass production, programmable micro-controllers are used in embedded system design. Embedded system software drives the micro-controller and associated hardware components to provide the system functionality required. Embedded system software can program the same micro-controller and cooperate with proper peripheral configuration for various applications per requirement specified. To cope with the demanding requests for new embedded system products, the industry needs good design methods and tools for embedded system software development.

In order to reduce the development time for the embedded system software, various techniques could be taken, such as adopting software reuse technique, seeking for the advancement in software synthesis and verification [1, 3, 4, 6, 7].

Embedded system software is usually developed on a hardware platform that is different from the final target environment. Cross compilers are used in the software development station to generate target code, and then downloaded into the RAM or programmable ROM resides in embedded hardware platform for execution. Accordingly, development methods that could enable the parallelism in software and hardware development will also be helpful for embedded system development. Object-Oriented programming is a paradigm that has been pledged to enable better software re-use, object-oriented frameworks have been worked prominently in this aspect [7, 13, 14]. In this article, we propose a development method that integrates the object-oriented paradigm to support the parallel development in embedded software and hardware. It also provides framework for execution information collection to support the system verification and tuning.

This article is organized as follows. Section 2 gives a brief overview about previous work in object-oriented software framework and tools that support embedded software development. Section 3 describes the proposed embedded software development method that based on a *Tunable Embedded Software Development Platform* (*TESDP*). Section 4 illustrates the feasibility of this development platform through the design and implementation of an embedded mobile data terminal for intelligent transportation system application. Section 5 concludes the article and gives directions for future work.

2. Previous Work

Embedded system is a special purpose computer system that consists of controller and peripheral devices. Most embedded system needs to response to some external events with some timing constraints. To cope with proliferating demands in embedded system development, various methodologies and tools are developed for embedded real-time system development.

Object oriented frameworks [9] provide reusable domain specific software that can be applied with minor modification. Two recently proposed frameworks, *Object-Oriented Real-Time System Framework (OORTSF)* [13, 14] and *RTFrame* [5] are providing reusable real-time system frameworks. *VERTAF* [7] integrates verification capability into its framework.

Execution time information of functions in the embedded system provides base data for system design, analysis and verification. Classical scheduling policies [10, 11] use execution time information for schedulability check. Cortes *et al.* [4] introduce Petri Net based formal verification method that uses "transition delay" associated with transition to represent the execution time of the function. In the timed automata based formal verification method [7] for embedded system, mode predicates represent the information about execution time of function. It is desirable to have actual execution time collection mechanism as a baseline design for embedded system development. We introduce several objects into a kernel that is based on *OORTSF* to provide the actual execution time information collection.

Hardware development tool supports gate-level abstraction, this model would be too detailed to be suitable for the development of embedded system. Some embedded system development platform [16] provides higher level of abstractions include microprocessor, cache, memory, DMA, etc. Some embedded system development tools abstract the system into graph structure and use graph algorithm to explore the properties required by system specification [1, 3, 7, 15]. The abstractions used in the above tools are either too detailed or too high-level for the development of embedded system that need to address software and hardware at the same time.

It is suggesting in providing a development platform to support the abstractions of both the hardware and software that are manageable to the embedded system application designer. We propose a *Tunable Embedded Software Development Platform (TESDP)* that addresses this issue to support the parallel development of embedded software and hardware.

3. Tunable Embedded Software Development Platform

Typical embedded system consists of programmable micro-controller, memory, and peripherals. Embedded system reacts to its environment and needs to satisfy some kind of execution sequence and timing constraints. Accordingly, most embedded software is also real-time software. Embedded system exhibits its functionality through its input/output interconnections, and peripherals with respect to the environment. Software on micro-controller does the data computation and senses the environment data to generate the system output. Embedded system synchronizes and communicates with its environment via mechanisms such as hardware interrupts, port based input/output and memory mapped input/output.

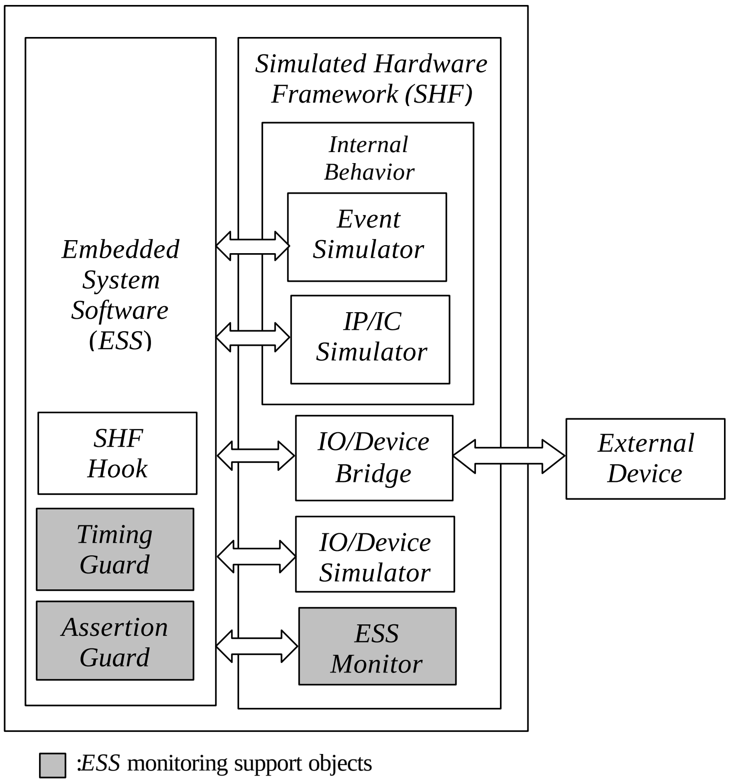

Figure 1. Tunable Embedded Software Development Platform (*TESDP*).

Since C/C++ programming language is a very popular programming language used by most of the micro-controllers and the systems running under a Windows or a Unix OS platform. Most of the micro-controller uses Windows and Unix OS platform as its cross development platform. From the C/C++ programming point of view, target platform and the cross development platform provide the same abstraction in using the same high-level language. From system behavior point of view, the major difference between target platform and the development platform will be the differences in hardware interrupts, input/output port, the peripherals that exhibit system functionality. After keen arrangement, we can make up an illusion of the target embedded system that can execute the C/C++ programs of the target systems, and exhibits the behavior of the peripherals of the target system on the development platform. We call this embedded software development system as *Tunable Embedded Software Development Platform (TESDP)*. Figure 1 shows the software architecture of *TESDP*, which consists of two major parts: *Embedded System Software (ESS)* and *Simulated Hardware Framework (SHF)*. The task scheduling and control of *ESS* is based on the design of *OORTSF* [13, 14] to integrate the required application functionality. *TESDP* further adaptes an *ESS* from an original embedded system by compiling and linking it to the *SH*F for execution on the development platform.

We add three object classes into the *Embedded System Software (ESS)* to support verification data extraction and provide monitoring function of *TESDP*. We use *Timing Guard object* to collect the execution time information of function. The *Assertion Guard* object is used to provide run-time status information to the *ESS Monitor* in *SHF*.

- *SHF Hook:* This is an object-oriented class that supports the insertion and replacement of the operations in *ESS* operations for *SHF*. *SHF Hook* provides systematic and documented insertion of the mechanism to match *ESS* into the environment of our *TESDP* platform.

- *Timing Guard*: The execution time of a task depends on the speed of the processor used. We introduce a *timing guard object* to gauge the elapsed time of a function and to introduce the execution time offset as required for different platforms. Using *Timing Guard*, a user can mitigate the timing gap between *TESDP* and actual embedded system operation. *Timing Guard* supports the extraction of actual execution time information for a function. This information can be collected and refurbish to the system tuning process.

- *Assertion Guard*: System properties have to be formulated for formal verification[7]. Some programming paradigms focused on introducing pre-condition/post-conditions and invariants into the to-be-verified software in a structured manner[12]. System requirement compliance check can also be built into the software. For example, we can present condition of interest and its run-time verification using *Assert Guard* objects. Embedded system operation does not have rich user interface with outside world as general-purpose computer system does. Cooperating an *Assertion Guard* with the *ESS Monitor* in *SHF* can provide report to the system developer.

We choose C/C++ language to write programs executed in both the target micro-controller and development platform. To let the embedded software program executes on the *TESDP*, we need to find out a layer of separation that can derive an efficient *SHF* design. We categorized the hardware abstractions first and provide mechanisms to support the hardware abstraction using combinations of the software and hardware on the development platform and from the external devices connected to the *TESDP* software development platform. The layer of separation we choose for the micro-controller core and peripherals consists of three major abstractions described as follows.

- *Internal behavior*: It represents CPU interrupts, memory mapped I/O and special internal Integrated Circuit (IC) and/or Intellectual Proprietary (IP) components like programmable logic arrays. *The Event Simulator and IC/IP Simulator* are used to simulate the *internal behavior* of components.

- *IO/Device Bridge*: It represents external communication interfaces like RS232 and parallel port. *IO/Device Bridge* simulates those IO interconnections. *IO/Device Bridge* uses physical interface to connect the external devices.

- *IO/Device Simulator*: It represents IO devices, such as LCM display and keyboard/keypads. *IO/Device Simulator* simulates the IO/Device inside or externally connected to the embedded system.

- *ESS Monitor:* It reads *ESS* execution status data from *ESS Assert Guard* Object. *ESS Monitor* provides a means for the report of system and application function execution status to the system developer. Execution status information is very useful for system verification and tuning.

Using these abstraction techniques, the *SHF* can be tuned to adapt to the changes in the target embedded configuration. *SHF* is an object-oriented software framework that provides an execution support environment for the *ESS*. We use the Windows-98 platform on a Personal Computer for this *TESDP* design and implementation. Table 1 shows the *SHF* classes, *SHF* implementation and associated example embedded components. The object-oriented framework is a software reuse technique that provides half-done software to facilitate the reuse in both design and code. The *SHF* classes can be instantiated and composed to simulate the hardware configuration of an embedded system. During the evolution, if the embedded system is adapted to new hardware technology, the system developer can readjust the *SHF* according to the changes.

Table 1. Major *SHF* object-oriented classes.

SHF Class	*SHF* Implement.	Example Embedded Component
Event Simulator	Thread	Scheduler Timer
IC/IP Simulator	Software API	Programmable Logic Device
IO/Device Bridge	Thread	RS232 connection
IO/Device Simulator	Thread	LCM Display
Timing Guard	Thread	LCM Display

The main advantage of this approach is to provide a simulated hardware environment associated with the stages of hardware architecture evolution in embedded system development. Some of the modular device control and/or inter-connections can be tested in the *TESDP* first, instead of building a total design to test the target system. In our experience, this saves a lot of time, because the *TESDP* is more stable and convenient in access. With confirmed protocol between controller and the device interactions, most of the code can be execute directly after re-compilation. Some of the devices are controlled via general interface, such as: RS232, the mismatch between the two platforms has been harmonized by the baud rate setting for the RS232 interface.

We are also aware of some mismatches between the embedded platform and the *TESDP* approach. These mismatches are timing mismatch, compiler mismatch and abstraction mismatch as follows. (1) *Timing mismatch*: Real-time system scheduling is done based on the clock of a platform. Running *TESDP* on a general purpose Personal Computer has the positive side of higher clock rate than on the target embedded system. Yet it also introduces additional system overhead posed by the underlying operating system. (2) *Compiler mismatch*: Specific compiler and linker might have specific flaws inside. However, *TESDP* could have more matured compiler than target micro-controller. Earlier *ESS* software development on *TESDP* turns out to be an efficient approach. (3) *Abstraction mismatch*: It depends on the availability of the hardware and software capability on the *TESDP* with respect to the embedded system. *TEDSP* user adjusts the configuration of *SHF* for target embedded system configuration, this also introduces deviation between them. However, the standard interface mechanism, like RS232, eases the abstraction deviation between *SHF* and the real target.

4. Application Example

Modular Mobile Data Terminal (*MMDT*), an embedded system to be installed in a commercial or private car, provides various remote management/support functionalities via the mobile data communication capability and global positioning system (*GPS*) built into the *MMDT*. *MMDT* also provides to the car driver a user interface that includes a large size LCM display module, a set of control buttons, and optional PS2 keyboard. *MMDT* handles two types of data communication protocols, MAP27 protocol stack software for trunking radio[8] and AT command interface software for Global System for Mobile communications (*GSM*) module[2]. Figure 2 shows the hardware architecture of *MMDT*.

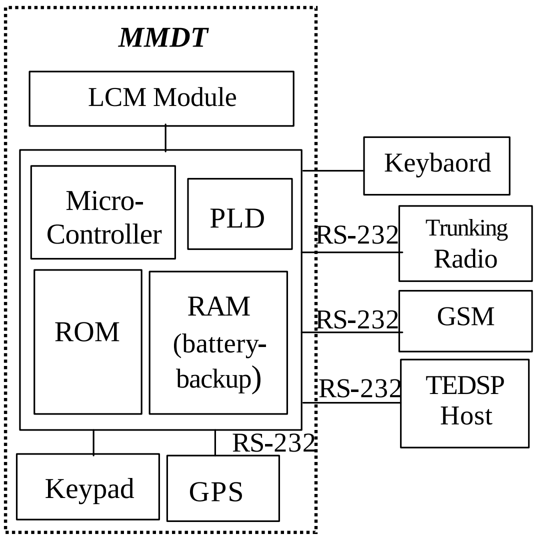

Figure 2. MMDT hardware architecture.

In Figure 2, PLD is a filed programmable logic device that has been programmed with *VHDL* to handle the PS2 keyboard bit-level interrupts, pack the data bits into bytes of data, and send it to Micro-controller with another byte-level interrupts and associated memory-mapped data bytes for further processing. The keypad inputs are also encoded by this PLD, which interrupts the Micro-controller to notify the occurrences of the keypad depression with associated keypad code. The LCM module is a display panel with back light control that is capable to display large traditional Chinese font for easy reading by the car driver. However, the LCM module has relatively low duty cycle as compared to the speed of Micro-controller, and it needs polling based access to confirm the availability for updating further display data. The LCM module display duty-cycle will also be a parameter that varies when the replacement of LCM in different design becomes necessary.

In this system, RS232 interfaces are used to connect external devices that include *GPS*, Trunking Radio, *GSM* module and a remote host computer. *Simulated Hardware Framework (SHF)* on *TESDP* uses actual RS232 interface to control external hardware modules. We use Windows-98 platform for this *TESDP* design and implementation. Table 2 shows the *SHF* classes and objects instantiated for the *MMDT* execution on our *TESDP* environment.

Table 2. *MMDT* components to *SHF* classes mapping.

Embedded components	*SHF* object	*SHF* class	*SHF* implement.
Scheduler Timer	Timer Thread	*Event Simulator*	Timer Thread
Keypad	Keypad Thread	*IP/IC Simulator*	Software API
PS2 Keyboard	Keyb Thread	*IP/IC Simulator*	Software API
GSM connection	GSM Bridge	*IO/Device Bridge*	RS232 Thread
Trunking Radio connection	Trunking Radio Bridge	*IO/Device Bridge*	RS232 Thread
GPS connection	GPS Bridge	*IO/Device Bridge*	RS232 Thread
LCM module	LCM display Simulator	*IO/Device Simulator*	Software API
TEDSP Host Connection	Host Computer Bridge	*IO/Device Bridge*	RS232 Thread

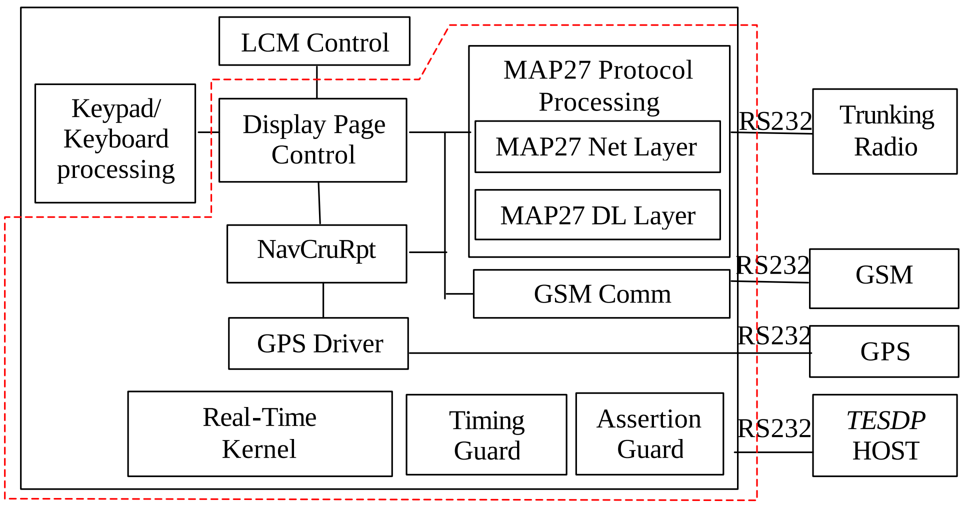

Figure 3. *MMDT* software architecture.

MMDT embedded software consists of a baseline *OORTSF* [13, 14] framework as its *Real-Time Kernel* to control application tasks. Figure 3 shows the embedded software architecture of *MMDT*. The *MAP27 Protocol Processing* consists data link layer processing *MAP27 DL Layer*, and network layer processing *MAP27 Net Layer*. An external radio set is connected to the *MMDT* for trunking radio communication. *NavCruRpt* does system navigation and cruise function based on the *GPS* data from *GPS Driver*, it reports the current vehicle position data to control center through mobile communication. *GPS* is a module built inside the *MMDT* box. *Display Page Control* manages the page displayed on the LCM module through *LCM Control*. *Keypad* selects the display menu page for user controls. *Keyboard* inputs data to the data field on LCM display page. *GSM Comm* controls the external *GSM* communication module using AT commands[2]. In Figure 3, the box with dashed-line delineates the implementation boundary between *ESS* and *SHF*.

During the development of *MMDT*, some hardware modules were added into the system follows the evolution of system requirements. The *GSM* communication module control has been integrated on the *TESDP* environment first and integrated into the real target platform after the control and operation scenario confirmed. Many of the devices are connected via *RS232* interface and the *SHF* uses actual *RS232* to drive the devices on *TESDP*. Therefore, there are little differences in the operational scenarios between actual embedded platform and *TESDP*, this further proves the feasibility of our *TESDP* approach.

5. Conclusion

The *TESDP* approach for embedded software development provides the possibility of parallel development in embedded hardware and software. Using the proposed development platform, the development of embedded system software can be de-coupled from the hardware platform while maintaining very close semantic similarity for the function operates on both platforms. This kind of development platform will be very desirable for electronic industry that is seeking for the grasp of booming embedded system market, such cell phone, digital camera, personal digital assistance (PDA), etc. We have used this approach in an evolutionary development of embedded *Modular Mobile Data Terminal* (*MMDT*) system for intelligent transportation system applications. With comprehensive communication and user interface, this *MMDT* can also be adapted to support other domain of application. With the support of *TESDP* approach, the cost of future adaptation of *MMDT* could be reduced. This also provides evidence for the value of *TESDP* approach in the development of other embedded systems.

Our *TESDP* approach supports the development of software for embedded system that consists of micro-controller and peripheral integrated circuits (ICs). From hardware aspect, some hardware component providers provide their intellectual property (IP) cores of micro-controllers and peripherals in the form of electronic files. System designer integrates various IP cores as required and turn them into a System On a Chip (SOC). This SOC approach further provides embedded system hardware with even more cost competitiveness. However, SOC needs embedded software that executes on the micro-controller and elaborates the peripheral to fulfill the system functionality. We believe that our approach can also be applied to the SOC type of embedded system development.

We are working on the integration of more comprehensive software frameworks and code synthesis capability [1, 3, 4, 6, 15] to support the development of different types of embedded systems. We are also integrating software verification capability into this development platform [7] and make it both tunable and verifiable development environments for embedded system development environment.

6. Reference

[1] K. Altisen, G. Gobler, A. Pneuli, J. Sifakis, S. Tripakis, and S. Yovine, "A Framework for Scheduler Synthesis," In *Proceedings of the Real-Time System Symposium* (*RTSS'99*), IEEE Computer Society Press, 1999.

[2] AT command set for GSM Mobile Equipment (ME) (GSM 07.07 version 4.4.1), Digital cellular telecommunications system (Phase 2), European Telecommunications Standards Institute, France, March 1999.

[3] F. Balarin and M. Chiodo. Software synthesis for complex reactive embedded systems. In *Proceedings of International Conference on Computer Design (ICCD'99)*, pp. 634 – 639. IEEE CS Press, October 1999.

[4] L. A. Cortes, P. Eles, and Z.Peng, *"Formal Co-verification of Embedded Systems using Model Checking*," In *Proceedings of EUROMICRO*, pp. 106-113, 2000.

[5] P. -A. Hsiung, "RTFrame: An object-oriented application framework for real-time applications," In *Proceedings of the 27th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS'98)*, pp. 138-147, IEEE Computer Society Press, September 1998.

[6] P. -A. Hsiung, "Formal synthesis and code generation of embedded real-time software," In *International Symposium on Hard-ware/Software Codesign* (CODES'01, Copenhagen, Denmark), pp. 208 – 213. ACM Press, New York, USA, April 2001.

[7] P. -A. Hsiung, W.-B. See, T.-Y. Lee, J.-M Fu and S.-J. Chen, "Formal verification of Embedded Real-Time Software in Component-Based Application Frameworks," to appear in *The 8th Asia-Pacific Software Engineering Conference* (APSEC 2001).

[8] Introduction to MAP27 protocol, Web Site: "http://www.condor-cci.com/trunking.new/map27.htm".

[9] R. E. Johnson, "Frameworks = (Components + Patterns)," In *Communications of the ACM*, Vol. 40, No. 10, pp. 39-42, October 1997.

[10] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *Journal of ACM*, Vol. 20, No. 1, pp. 46-61,1973.

[11] J. -F. Lin, W. -B. See, and S.-J. Chen, "Performance Bounds on Scheduling Parallel Tasks with Communication Cost," *IEICE Trans. Information & Systems*, Vol. E78-D, No. 3, pp. 263-268, March 1995.

[12] M. Lippert and C. V. Lopes, "A Study on Exception Detection and Handling Using Aspect-Oriented Programming", In *Proceedings of ICSE'2000*, ACM Press.

[13] W. –B. See and S. -J. Chen, "High-level reuse in the design of an object-oriented real-time system framework." In *Proceedings of the International Computer Symposium*, pp. 363-370, December 1996.

[14] W. –B. See and S. –J. Chen, "Object-oriented real-time system framework." *Domain-Specific Application Frameworks*, pages 327-338, Ed. M.E. Fayad and R.E. Johnson, Wiley, 2000.

[15] M. Sgroi and L. Lavagno, "Synthesis of Embedded Software Using Free-Choice Petri Nets," In *Proceedings of IEEE/ACM Design Automation Conference* (*DAC'99*), ACM Press, June 1999.

[16] F. Vahid and T. Givargis, "Platform Tuning for Embedded Systems Design," *IEEE Computer*, No. 34, Vol. 3, pp. 112-114, March 2001.