An Embedded Computing Platform for Robot

Ching-Han Chen
Department of Computer Science and Information Engineering, National Central University
pierre@csie.ncu.edu.tw

Sz-Ting Liou
Department of Computer Science and Information Engineering, National Central University
955202083@csie.ncu.edu.tw

Abstract

As the robotic industry is growing boomingly, the functionalities and system's architecture of robots are more and more complex. The development of robotic application system becomes a time-consuming and difficult task. In this paper, we propose an embedded computing platform for intelligent robot, and then design a reliable real-time operating system (RTOS) on the platform for rapid developing intelligent robotic applications. The proposed embedded computing platform includes a reconfigurable 8-bits processor core and some robot-dedicated hardware intellectual property (IP) which can be generated and reconfigured easily. Based on the embedded processor core, a real-time OS, uC/OS-II, is ported to this platform. The RTOS is adjusted and optimized due to the robot-specific requirements and the hardware resources constrains. Finally, a simple example is applied to demonstrate the software/hardware (SW/HW) co-design flow based on the proposed platform.

Keywords: Intelligent robot, Embedded Computing Platform, RTOS, Reconfigurable, SW/HW co-design

1. Introduction

The research of robotics is originated in 1970's. The purpose of robot's utilization is to replace manpower efficiently, and increase the factory's manufacture ability. Its purpose was using the efficiency of robot to take the place of manpower and increase factory output. With the advancement of science and technology, robots have been moving out from laboratory and existed in our daily life. Furthermore, researchers, biologist, mechanical engineer and scientist of robotics, cooperate together to do the robotic research with the perspective of biomimetic approach. The research involves creating biomimetic and behavior-based robots that bring an upsurge of the study in robot and evolve into some topics of intelligent robot research [1]-[4].

The robotic system is growing extensively in recent years. Many kinds of robot (e.g., Humanoid Robot, Security Guard Robot, Home Robot, Entertainment Robot, etc.) are manufactured rapidly into the market. The development cycle must be very short, and letting the robot into market on time become available; however, the complexity of applications for robotic system is increasing day by day. In order to create a High-Performance and Low-Cost robotic system in fast and flexible way, it is becoming necessary to develop a robotic development platform with hardware and software IP in a hurry. Therefore, in order to coordinate different hardware and software for robot (especially for intelligent robot), an embedded computing platform plays a very important role in the development of robotic system.

Many researches [4]-[7] indicate that a layered approach is gradually becoming a trend in the design of robotic platform. The benefits of this design method include high-level behavior control, task dispatching and flexible design that can make the control structure of robotic platform more clearly and the operation of robot more efficiently. Consequently, we propose a layered platform which is composed of application, operating system, processor and device (from top layer to bottom layer). On the basis of layered approach, we build an embedded computing platform for robot.

Rest of this paper: Section 2 reviews related work and Section 3 presents an overall embedded computing platform for robots. Section 4 demonstrates an experimental example based on the proposed platform. Conclusions and future works are summarized in Section 5.

2. Related work

978-0-7695-3158-8/08 $25.00 © 2008 IEEE
DOI 10.1109/SUTC.2008.69

IEEE computer society

There are many robotic development platforms [8], [9] can aid the development of robotic system so far. The followings intend to describe a variety of platforms which are proposed in industry and academia.

Microsoft Robotics Studio [8] is one of the business software platforms; for instance, it is to supply a software platform that can be used across a wide-variety of hardware and it is also the first robot-dedicated software announced by Microsoft. The Development Environment of Microsoft Robotics Studio includes the following major characteristic: 1. End-to-End Development Platform. The platform enables developers to interact with robots using Windows or Web-based interfaces. 2. Lightweight services-oriented runtime. The platform offers the services which is message-based architecture and make it simple to connect with robot's sensors and actuators by using a Web-browser or Windows-based application. 3. Scalable and extensible platform. The programming model can be applied for a variety of robot hardware platforms. Also, third parties can extend the functionality of Microsoft Robotics Studio by providing additional libraries and services.

iRobot create [9] is one of the business hardware platforms and originated from the invention of MIT Computer Science and Artificial Intelligence Lab; for instance, it mainly offers a basic hardware development platform which facilitates developers to program simple operation of the robot without considering the low-level hardware architecture. Moreover, the additional command module which can be mounted to the platform and is provided as well. This optional module fulfils advanced developers to construe the automatic application of robot and enables users to stretch the application of robotic functionality by means of adding or combining sensors, digital cameras, computers or other electric device.

Besides, the academic community proposed many layered platform architectures, too. In [4], a layered behavior planning is established for optimizing robot's behavior that helps to modify the behavior model of intelligent robot in accordance with environmental characteristics. In [5]-[7], some group numerous controllers into master/slave control mode and some divide the system into three layers in roughly which include application layer, OS layer and physical layer. These methods are mainly aimed at robotic system and can not only speed up the communication capability of internal system, but also hold the property of reconfiguration and elastic expansion.

Along with progress of times, the application fields of robot are increasing extensively and the applications are going to be designed not only for specific purpose anymore. At the meanwhile, the robot's behavior will also tend towards complexity to let the control flow of the device more inextricable. Therefore, a development platform which integrates hardware and software will become an indispensable consideration in the future. The platform must be highly scalable to make the development of robot's hardware and software can have the flexible design advantages which include cost, performance and time-to-market.

3. Embedded computing platform for robots

Fig. 1 is the architecture of embedded computing platform which we propose. Refer to the layered perspective of embedded computing platform in [4]-[7], we describe our robotic embedded computing platform with platform view, system view and robot view respectively.

The platform can separate into four layers in platform view which composed of application, operating system, processor and device (from top layer to bottom layer). The system view includes application layer, management layer, computing layer and physical layer separately.

The robot view divides the platform into three layers because the robot's behavior mode and the driving of device may be altered due to environment situation. The layers comprise robot's intelligent and behavior decision in the highest level, the motion control in the lowest level and transition zone in the middle which can configure the control flow of hardware and software since environment situation changes.

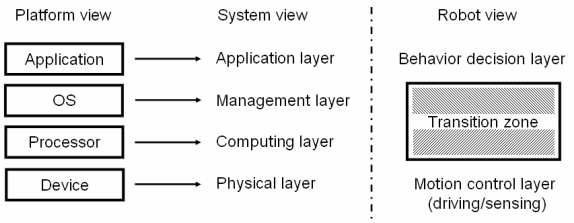

Fig. 1. Layered architecture of embedded computing platform for robots

3.1. Application layer

On the embedded computing platform, the developers can use C/C++, high-level languages, to develop the applications for robot. Also, we now use the off-the-shelf Keil development tools [10] to do the programming task of compilation and simulation. In addition, OS in next layer will provides Application-programming interface (API) for developing applications and drivers for propelling devices that can

help developers to create applications easily and speedy without considering the hardware construction in low-level and the driving methods.

In the utilization of embedded system, there is usually a great amount of input/output (I/O) demand for communicating with external component to carry out the application's intention. Accordingly, the main purpose of API and driver is to encapsulate I/O flow of the system that helpfully let the developers can concentrate on application's algorithm developing and high-level management and decision program's designing without worrying about the I/O control flow. Section 3.2 will discusses API and driver in more detail.

3.2. OS

By the foundation of robot's attribute, a satisfied OS for robotic purpose needs a well management mechanism to deal with tasks and devices that can coordinate various tasks inside the robot to work fine. As well as offering real-time kernel to let the robot react quickly and operate smoothly, the OS kernel ought to have IPC methods which are dedicated to the convenient of robot's applications developing.

Fig. 2 presents the structure of OS kernel. API provides the interface between OS and top-level applications. Driver provides the interface between OS and low-level devices (e.g. actuator, sensor, etc.).

In this paper, we adopt uC/OS-II [11] as an implementation example of robotic OS. uC/OS-II is a RTOS which is open source and widely used especially for control system, and it has the advantages of high performance, small footprint, excellent real-time and scalable.

In our design, inside OS layer which requires API, system call, kernel and driver. To meet the requirement of intelligent robot's behavior control, we rewrite API and driver. API and driver mainly encapsulate the I/O control flow as mentioned in section 3.1. We implement the more top-level part of the I/O control flow into API and the more low-level part of the I/O control flow into driver since API is the top-level interface, driver is the low-level interface and the coverage of the I/O control flow includes top layer and low layer.

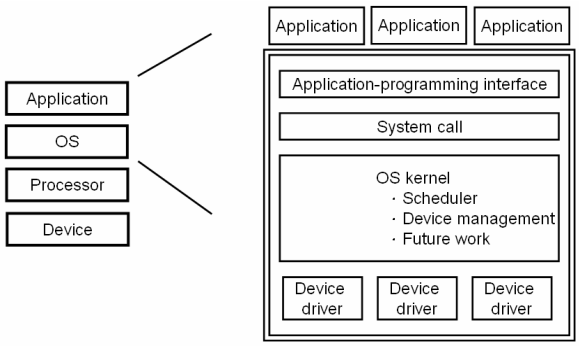

Fig. 2. The software part of embedded computing platform for robots

3.3. Processor

Base on the interactive requirement of intelligent robot and outer physical environment, robot-dedicated processor must possess a number of reconfigurable IPs which can progress in a fast integrated development. These IPs should be reusable and thereby depend on different circumstance can be increased/decreased or substituted. We then can base on the actual need to arrange IPs and optimize the hardware design for the embedded computing platform.

On our platform, we implement a reconfigurable 8051 processor core, MIAT51, which is modified from the open source MC8051 IP-core of Oregano Systems [12]. Because the robotic applications may often communicate with the interface of external component (e.g. I2C and UART) and the controllers for accessing RAM or flash memory, we can do a flexible adjustment for the special function register (SFR) of the processor that lets the processor can easily map to the peripheral interface of new added device. Thus the control and utilization of the device can be more convenient. Fig. 3 shows the architecture of our processor and interface IPs for the platform.

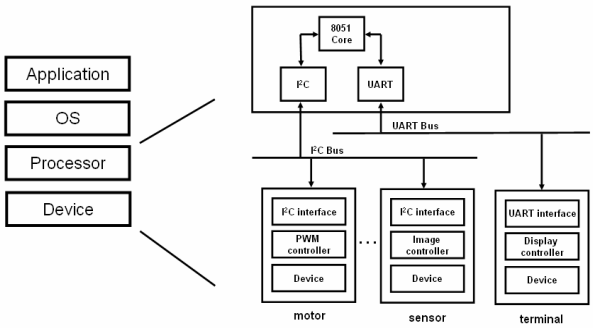

Fig. 3. The hardware part of embedded computing platform for robots

3.4. Device

As the design consideration of API and driver which is mentioned in section 3.2. According to the improvement of the robot's functionality and the increasing of the system's complexity, the driving method of the device becomes more difficult. At the moment, using software to achieve the device's control flow is comparatively more complex. Moreover, the robot may work inefficiently because the waste of CPU resources and Bus bandwidth. For example, the PWM signal generation, which is needed for motors, becomes the most serious problem. Consequently, we make a parametric PWM generator which is also a PWM hardware controller [13], [14] and can be synthesized very fast. This PWM hardware controller, which makes the robotic system can be controlled effectively and eases the load of the top-level application, receives the parameter from applications and generates the high efficiency PWM signal automatically.

Fig. 4 is a basic function block in the PWM hardware IP. To describe the complex behavior mode and control strategy precisely, we will use GRAFCET [15] as a discrete-event behavior modeling tool. We follow a set of automatic synthesis rules which is proposed by Chen et al. [16] to synthesize a customized PWM hardware IP which can be integrated into an automatic system of robot easily.

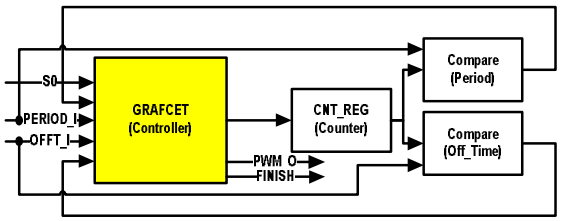

Fig. 4. The basic function block of PWM controller

4. Experimental system implementation in a hexapod robot

This experimental system, which is based on the embedded platform that we propose, implements subsumption system [17] on a hexapod robot. Subsumption system, a behavior-based robot programming method, is proposed by Rodney Brooks in 1986. The suppressor node and the inhibiter node inside the system can facilitate the layered and modular behavior control design.

Fig. 5(a) is a hexapod robot used in this experimental system which includes all kinds of behavior module of subsumption system (refer to Fig. 5(b)).

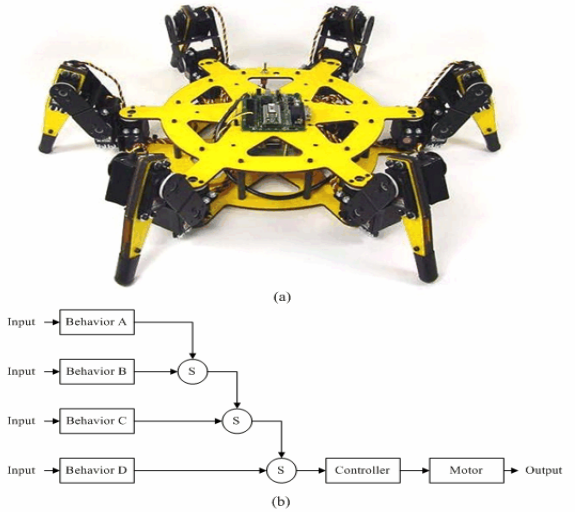

Fig. 5. (a) is the hexapod robot (b) is the subsumption system

In our platform, the high-level intelligent decisive behavior in subsumption system can be built on the application layer of our platform. And we can complete the robot's behavior design by using the mechanisms of scheduler and inter-process communication (IPC) which are the components in OS and can perform as the utility of suppressor node and inhibiter node. According to layer approach design, we may offload most part of the robot's complex and repeated software control flow into hardware portion by using GRAFCET for modeling and constructing the hardware IPs which can ease the load of processor for computing and enhance the whole system's performance.

To show the effectiveness of this designed platform, the PWM hardware IP introduced in section 3.4 can be used to explain the demonstration. We assume that the robot involves an 8-stage procedure to complete a movement, such as a step to go forward/backward, left/right or turn in place etc. However, there are 6 legs of the hexapod robot and each leg has 3 motors. In other words, we need to generate 144 PWM signals to control the robot's single movement by either software or hardware. Instead of using software control to generate PWM signals for every movement every time, we create a motion table on the top of the PWM controller to store the 8-stage procedures' parameter value for each basic movement. Once receiving a command instruction from the application, the PWM controller then can determine the movement with the instruction and acquire the relative parameter values from the motion table to generate corresponding PWM signals.

TABLE I
Impact of different implementation

PWM-signal generation	Software control	Hardware control
Instruction(s) to be sent through bus	144	1
Wasting processor resource	much	less
Extra memory space requirement	no	yes

Even though the PWM controller must requires extra memory space made on it for the motion table, TableTABLE I shows that the resource requirement impact of hardware control is less than software control. And this result also indicates good prospects and development advantages: First, without numerous iterative software control flow, the saving processor resource can devote to some other applications' algorithm computing for increasing the efficacy of the processor; Secondly, a substantial bandwidth saving on the bus may apply to transfer the sensing data from sensors to applications, especially the demanded information which is urgent and critical to the system.

5. Conclusions and future works

In this paper, we propose an embedded computing platform of intelligent robot which is layered architecture. And the platform can be used to solve the fast growing of complex designing problem of the robotic system. This platform considers the functionality requirements and resources constrain of generic robotic system and the tasks of robot's control in different abstraction layers. The platform, then, is programmed and designed by the principle of hierarchical and modular.

The applications of robotic system will tend to have diverse functionality and higher complexity in the future, hence, in the future work we will continue to optimize the RTOS kernel according to the feature of robot's motion and intend to design an optimum on-chip RTOS kernel [18]-[21] which is used to coordinate the operation of hardware and software. Besides, we will also design specific robotic multi-processor [22], [23] which is depend on the nature of robotic system's functionality requirements and attempt to make use of the architectonic property of multi-processor to accelerate the overall robotic system's operating performance.

References

[1] Brooks, R. A., "A Robust Layered Control System for a Mobile Robot", IEEE Journal of Robotics and Automation, March 1986, Vol. 2, No. 1, pp. 14–23.
[2] Arkin, R. C., Behavior-Based Robotics, MIT Press, Cambridge, MA, 1998.
[3] Joseph L. Jones, Anita M. Flynn, and Bruce A. Seiger, Mobile Robots: Inspiration to Implementation, AK Peters, Ltd, 1998.
[4] Rainer Bischoff, Volker Graefe, "Learning from Nature to Build Intelligent Autonomous Robots", Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on Oct. 2006, pp. 3160–3165.
[5] M. Omar Faruque Sarker, ChangHwan Kim, Seungheon Baek, Bum-Jae You, "An IEEE-1394 Based Real-time Robot Control System for Efficient Controlling of Humanoids", Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on Oct. 2006, pp. 1416–1421.
[6] J. Oh, D. Hanson, W. kim, I. Han, J. Kim, and I. Park, "Design of Android type Humanoid Robot Albert HUBO", Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on Oct. 2006, pp. 1428–1433.
[7] Fumio Kanehiro, Yoichi Ishiwata, Hajime Saito, Kazuhiko Akachi, Gou Miyamori, Takakatsu Isozumi, Kenji Kaneko, Hirohisa Hirukawa, "Distributed Control System of Humanoid Robots based on Real-time Ethernet", Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on Oct. 2006, pp. 2471–2477.
[8] http://msdn2.microsoft.com/zh-tw/robotics/default.aspx
[9] http://www.irobot.com/index.cfm
[10] http://www.keil.com/
[11] http://www.micrium.com/
[12] http://www.oregano.at/index2.htm
[13] Stefano Galvan, Debora Botturi, Paolo Fiorini, "FPGA-based Controller for Haptic Devices", Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on Oct. 2006, pp. 971–976.
[14] Narashiman Chakravarthy, Jizhong Xiao, "FPGA-based Control System for Miniature Robots", Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on Oct. 2006, pp. 3399–3404.
[15] R.David, "Grafcet :A powerful tool for specification of logic controllers", IEEE Trans. on Control Systems Technology, 1995, Vol. 3, No. 3, pp. 253-268.
[16] CHEN, Ching-Han; DAI, Jia_Hong; "Design and high-level synthesis of discrete-event controller", National Conference of Automatic Control and Mechtronics System, 2002, vol.1, pp. 610–615.
[17] Brooks, R. A., "How To Build Complete Creatures Rather Than Isolated Cognitive Simulators", Architectures for Intelligence, K. VanLehn (ed), Erlbaum, Hillsdale, NJ, Fall 1989, pp. 225–239.
[18] H. Walder and M. Platzner, "Reconfigurable Hardware Operating Systems: From Design Concepts to Realizations", Proceedings of the 3rd International Conference on Engineering of Reconfigurable Systems

and Architectures (ERSA), CSREA Press, June 2003, pp. 284–287.

[19] C. Steiger, H. Walder, and M. Platzner, "Operating Systems for Reconfigurable Embedded Platforms: Online Scheduling of Real-time Tasks", IEEE Transaction on Computers, November 2004, vol. 53, no. 11, pp. 1392–1407.

[20] M. Ullmann, M. Hubne, B. Grimm, and J. Becker, "On-Demand FPGA Run-Time System for Dynamical Reconfiguration with Adaptive Priorities", Field Programmable Logic and Application: 14th International Conference, FPL, Springer-Verlag Heidelberg, August 2004, pp. 454–463.

[21] Theelen B.D.; Verschueren A.C.; Reyes Suarez V.V.; Stevens M.P.J.; Nunez A., "A scalable single-chip multi-processor architecture with on-chip RTOS kernel", Journal of Systems Architecture, December 2003, Vol. 49, No. 12 , pp. 619–639.

[22] Becker, J., "Configurable systems-on-chip: challenges and perspectives for industry and universities", International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA), 2002, pp.109–115.

[23] Sun, F., Ravi, S., Raghunathan, A., and Jha, N.K., "Custom-instruction synthesis for extensible-processor platforms", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 23 (2), 2004, pp. 216–228.