AN APPROACH OF UBIQUITOUS DEVICES USING T-ENGINE IN VIETNAM

Nguyen Hoa Hung, Nguyen Quang Huy, Dinh Duc Anh Vu
University of Technology, VNU-HCM

(Manuscript Received on April 27th 2011, Manuscript Revised November 25th 2011)

ABSTRACT: *The 21st century is the era of Ubiquitous Computing where computing devices are present everywhere in our lives. To satisfy the development of this tendency, many hardware platforms have been proposed for developing Ubiquitous devices. Among them, T-Engine, an open standardized development platform for embedded systems, is one of the most popular platforrms. It is nowadays compatible with embedded equipments for a wide range of fields. In Vietnam, T-Engine has just been introduced for 4 years. However, most of the ubiquitous applications using T-Engine are developed restrictively based on the standard hardware of T-Engine. One issue that arises is the necessity of a solution to expand T-Engine hardware and use it to control automatic systems to satisfy different types of Ubiquitous devices. This research is to propose an approach to use T-Engine in the Ubiquitous Devices that require the attachment of the additional hardware as well as the complicated control mechanism with real time constraints. In this research, we proposed an expanding solution T-Engine through the extension bus. Besides that, we consider the timing problems in bus transaction and problems in real-time programming. A simple robot demonstration has also been designed and implemented to prove the feasibility of our model. This approach will open up a new tendency of developing complicated Ubiquitous devices using T-Engine in Vietnam.*

1. INTRODUCTION

Ubiquitous computing is a post-desktop model of human-computer interaction. The typical characteristic to distinguish ubiquitous computing with other model is that computing takes place everywhere for everyone. The computing devices will be embedded in the environment. In this model, computer was kept in the background presence.

There are three essential elements of ubiquitous computing those are embedded processor and embedded platform, wireless communication, sensor. These three elements also give rise to appropriate research trends besides several application developments. For the network communication, they are security wireless networking, network media, etc; for the sensor, we have developing different kinds of sensor, wireless sensor networking; and finally, they are developing low-power and high-performance processor, developing and utilizing standard platforms for ubiquitous computing.

Each ubiquitous computing system has to possess the following characteristics. Those are the ability of remember events, ability to aware the surrounding environment through various kind of sensors. Especially, this system should be responsive to other ubiquitous computing systems. Therefore, it is required to have the ability to handle a number of complex tasks.

In our research, we concentrate on the last of the mentioned above research trends. That is utilizing standard platforms, particularly, it is a famous platform called T-Engine. We propose a model for using this platform in ubiquitous devices to fulfill the characteristic of ability to handle a number of complex tasks.

T-Engine is an embedded-device standard development platform specified by T-Engine Forum, an industry organization consisting of 500 corporations. It supports various kinds of CPUs. It is equipped with T-Kernel which is a high real-time performance and resource saving operating system. T-Engine is compatible with ubiquitous devices for a wide range of fields.

The research contains the following issues: proposing a model for expanding T-Engine hardware, carrying out timing problems in the bus transaction, the real-time operating system T-Kernel and building a demonstration that proves the correctness of the research. The research is carried out on the T-Engine SH7760 which is equipped with the CPU SH7760. The proposed model is implemented using the CPU local bus as the communication method between T-Engine and expansion hardware. This paper contains four parts. The first part describes the hardware and software model using and explains the reason why the model is selected. The timing problems of bus transaction will be considered in the second part. The third part discusses the main features of T-Kernel, a real-time operating system of T-Engine, and some problems when programming on T-Kernel. The last part describes the robot demonstration using the proposed model.

2. SOLUTION OF EXPANDING T-ENGINE HARDWARE

Three problems must be solved when expanding T-Engine hardware: accessing a separated address location, accepting interrupt requests and direct memory access to the expansion part.

In the T-Engine SH7760 hardware specification in figure 1, there is an extension bus interface that is connected directly to SH7760 local bus. This bus interface provides the fast and direct connection to the CPU system bus for the complicated controlling application. Configuration parameters for the bus transaction of CPU SH7760 are managed by Bus State Controller. This block allows changes by setting up the bus parameter such as: address area, memory type, output control signals, bus width, timing waveform, etc.

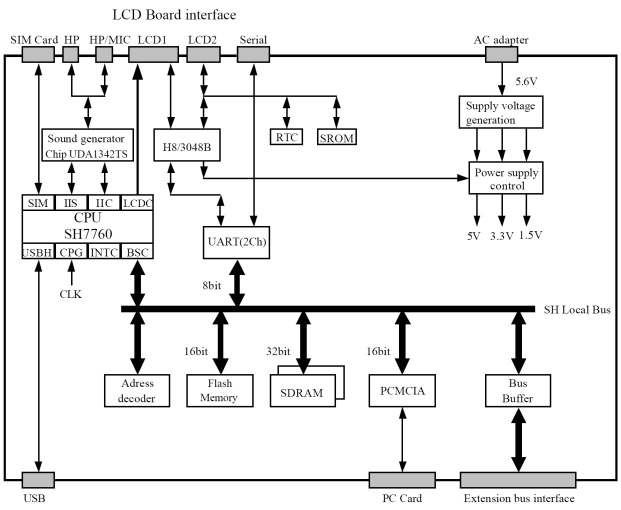

Figure 1. Block diagram of T-Engine SH7760.

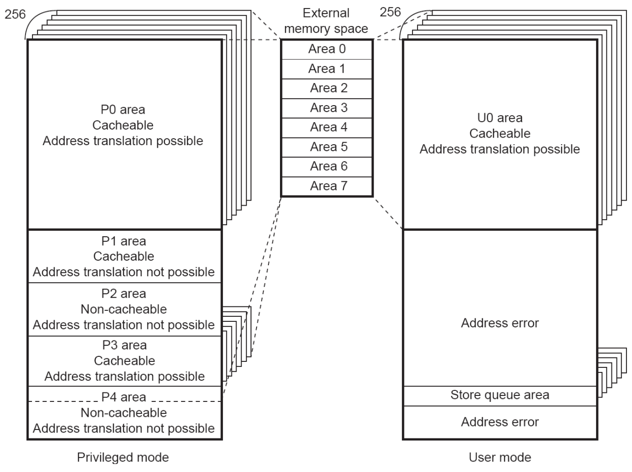
256
External memory space
256
Area 0
Area 1
Area 2
Area 3
Area 4
Area 5
Area 6
Area 7
P0 area Cacheable Address translation possible
P1 area Cacheable Address translation not possible
P2 area Non-cacheable Address translation not possible
P3 area Cacheable Address translation possible
P4 area Non-cacheable Address translation not possible
U0 area Cacheable Address translation possible
Address error
Store queue area
Address error
Privileged mode
User mode

Figure 2. Virtual address space of CPU SH7760

In programming aspect, to access the devices that are attached to the extension bus interface, the application has to have the way to access directly the external address space. CPU SH7760 is equipped with the Memory Management Unit and Cache that is shown in figure 2. The 32-bit virtual address space enhances with the ability of accessing the external memory by different methods. This ability is implemented by dividing the 32-bit virtual space into five areas. Each area owns a specific way of mapping between virtual address space and external address space.

P2 area is the one that allows the accessing without Memory Management Unit and Cache. It means that the virtual address space and external address space are mapped directly. However, P2 area accessing requires the privileged authority with two ways to access. The first one is to set the type of the task that contains the accessing code to system level. By this way, the task will have right to use all other kinds of resource of the system besides the right of accessing the external address space. As the result, system resources may be unintentionally damaged by this task. The second way is using device driver. All of the program portions that contain the external address space will be place in the device driver. The user task will access the external address space through some device driver interface function.

The second problem is accepting interrupt requests from the expansion part. CPU SH7760 supports an interrupt controller with three types of interrupt request: non-maskable interrupt request, IRQ interrupt request, IRL interrupt request. Four lines of IRL interrupt request are encoded by built-in FPGA to become sixteen external IRQ interrupt requests. Four of them are available for external using. As the result, T-Engine SH7760 provides four IRQ type interrupt requests with fixed configuration. To prevent the use this interrupt requests, the external interrupt signal have to be processed before being inputted to T-Engine. The processing includes restricting the activating period of an input signal as well as encoding the input signals if there are more than four interrupt requests.

The third problem is direct memory access to the expansion part. By the assist of direct memory access controller, direct memory access can be done with changing of different parameters such as: channel, data length, transfer mode, address mode, transfer request, bus mode, etc.

3. TIMING PROBLEMS

Timing problems affect the correctness and the speed of the transaction. Figure 3 describes the timing waveform of a standard bus cycle and a bus cycle with wait cycles.

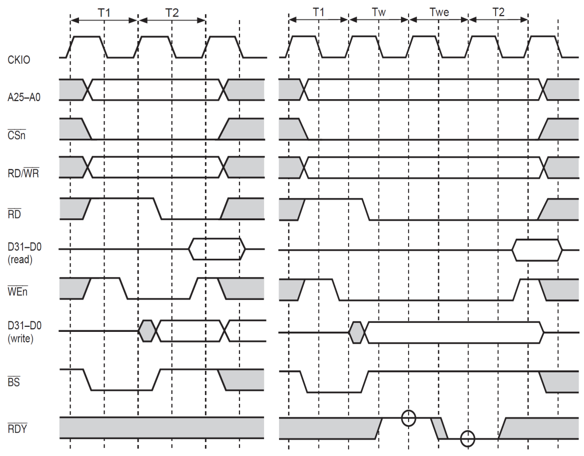

Figure 3. Standard bus cycle (left), bus cycle with wait cycles (right)

A standard bus cycle is the shortest bus cycle. It contains two clock cycles. The first clock cycle is used to activate the address signals and the control signal. The second clock cycle is the time that data signal is activated. Because there are various types of devices that can be connected to the bus interface, the standard timing waveform is not always applicable. The bus state controller provides four 32-bit registers to adjust the timing waveform of bus transaction. This allows various types of wait cycle that can be inserted into the standard bus cycle. The first type is the wait cycle that is inserted between two bus cycles. The second type is the one that is inserted into a bus cycle after all the control signals are activated and before the data signals is activated. The third type is external wait cycle which is inserted when the bus state controller receives a not ready signal of the external device. The fourth type is inserted at the time immediately before read-signal or write-signal is activated. The fifth type is inserted at the time after the address signals and control signal are deactivated but the data signals still present on the bus.

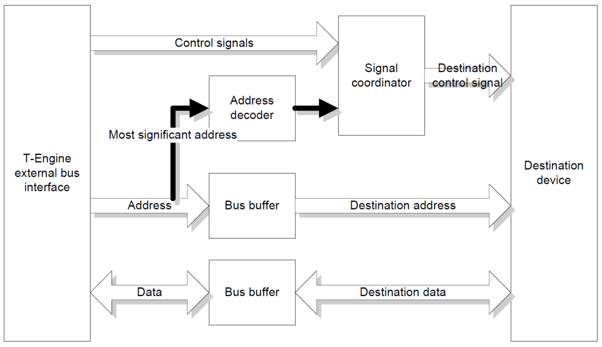

Figure 4. Connection model to the extension bus interface.

There are two main problems that arise when we carry out this research. The first problem involves the incompatibility of timing waveform between T-Engine and devices. This is overcome by stretching the waveform of T-Engine until it is compatible with the one of device by adding wait cycles into the bus cycle. The second problem is the synchronization of bus signal after crossing intermediate devices. Figure 4 describes connection model using the extension bus interface.

In this model, T-Engine and device are connected by address, data and control signals. The most significant address signals are input into address decoder to divide the address area. After that, they are coordinated with control signals to produce the compatible signal for the destination device. The other address signals and data signal cross the buffer to the destination device. The intermediate devices in this model are the signal coordinator, address decoder and the bus buffer.

We assume that the timing waveform of T-Engine and destination device have been made compatible with each other. Normally, bus buffers are faster than address decoder and signal coordinator. As the result, control signals arrive at the destination device later than address and data signals. The bus transaction will be fail or will be wrong.

The solution is either adding more buffer devices to the address and data signals to lengthen the delay time or replace the faster address decoder and signal coordinator.

4. REAL-TIME OPERATING SYSTEM T-KERNEL

T-Engine has a pre-specified real-time operating system T-Kernel. This is the next-generation real-time operating system of TRON project. T-Kernel is the combination of three parts: a scheduler, objects and services with the following kinds of functions: task control functions, task communication functions, memory management functions, exception control functions, time management functions, subsystem management functions.

T-Kernel scheduler is implemented based on the preemptive priority-based scheduling algorithm. The independent thread of execution is defined as a task. Besides task, T-Kernel also provides other types of object to manage the synchronization and the communication between tasks. Those are: semaphore, event-flag, mailbox, mutex, message buffer, rendezvous port.

There are some problems that need considering when programming with T-Kernel. The first problem is resource sharing, the common problem of multitasking operating system. Resource sharing is a function of task priority. The higher priority task has precedence over other tasks when accessing shared resources. However, if higher priority tasks always take resources, lower priority tasks will be in starvation state.

The second problem is deadlock. Deadlock happens when the following conditions are present: mutual exclusion, no preemption, hold and wait, or circular wait.

The third problem is priority inversion. Priority inversion is a situation in which a low-priority task executes while a higher priority task waits on it due to resource contentions. T-Kernel provides two type of mutex object: priority inheritance mutex and priority ceiling mutex.

The solutions for these problems can be found in [1], in which the authors give out several models to overcome specific problems when programming with real-time operating system.

5. DEMONSTRATION

A robot is a typical automatic system application so we design a simple robot to demonstrate our approach. It is implemented using two T-Engines. The first T-Engine controls the action of the robot while the second T-Engine is in the remote control device and controls the interaction with user. Robot can be controlled manually using remote control device. Besides that, robot can run automatically and solve the block-world problem. The block-world problem is a typical artificial intelligence problem. In this problem, the robot has to recognize the order structure of some blocks. In this demonstration, we implement with four blocks. After recognizing the structure, the robot will carry out the planning process and find out the solution to reorder the block to the expected structure.

Robot includes several components such as: run-motor card, lift-motor card, hold-motor card, sensor modules and communication modules. Those are connected to T-Engine through the external bus interface.

Figure 5 describes how a motor card can communicate with T-Engine external bus interface. The motor card acts like a peripheral module of T-Engine. T-Engine controls its operation by setting the value of three registers. When it has finished doing a command, it sends an interrupt signal to T-Engine.

Robot is controlled by six concurrent tasks. Four tasks are used to control motor cards. One task controls the interaction with user. The other task is the main processing task. These tasks communicate with each other's by a message buffer.

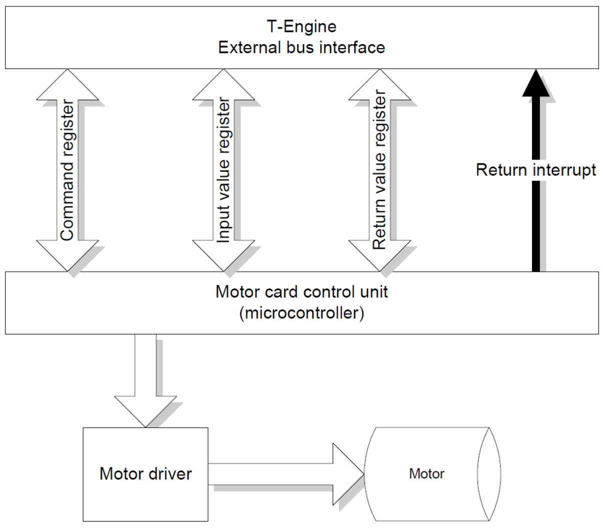

Figure 5. Motor card block diagram.

6. CONCLUSION

The research is the first step in developing the controlling application of T-Engine as well as Ubiquitous Devices. A connection model has been proposed for expanding hardware of a complicated embedded platform. Besides that, many issues have been introduced and partly solved. This approach has opened up a new tendency of developing complicated Ubiquitous devices using T-Engine in Vietnam.

MỘT CÁCH TIẾP CẬN VỚI THIẾT BỊ UBIQUITOUS SỬ DỤNG T-ENGINE TẠI VIỆT NAM

Nguyễn Hoà Hưng, Nguyễn Quang Huy, Đinh Đức Anh Vũ
Trường Đại học Bách khoa, ĐHQG-HCM

TÓM TẮT: *Thế kỷ 21 là kỷ nguyên của Ubiquitous Computing trong đó các thiết bị tính toán xuất hiện ở khắp mọi nơi trong đời sống của chúng ta. Để đáp ứng sự phát triển của xu hướng này, nhiều nền tảng phần cứng đã được đề xuất để phát triển các thiết bị Ubiquitous. Trong số đó, T-Engine, một nền tảng chuẩn hoá mở cho hệ thống nhúng, là một trong những nền tảng phổ biến. Nó thích hợp để phát triển những thiết bị nhúng ở nhiều lĩnh vực khác nhau. Ở Việt Nam, T-Engine được giới thiệu cách đây 4 năm. Tuy nhiên, hầu hết các ứng dụng trên T-Engine chỉ hạn chế ở phần cứng chuẩn. Một vấn đề nảy sinh đó là sự cần thiết phải có một giải pháp để mở rộng T-Engine và sử dụng để điều khiển các hệ thống tự động để đáp ứng các yêu cầu khác nhau của một hệ thiết bị Ubiquitous. Nghiên cứu này đề xuất một cách tiếp cận sử dụng T-Engine cho thiết bị Ubiquitous đòi hỏi có thêm các thiết bị phần cứng và yêu cầu về điều khiển phức tạp với các ràng buộc về thời gian thực. Chúng tôi đề xuất giải pháp mở rộng T-Engine thông qua extension bus. Bên cạnh đó, vấn đề timing trong giao tiếp bus và lập trình thời gian thực cũng được xem xét. Một mô hình robot đơn giản để minh hoạ tính khả thi của nghiên cứu được hiện thực. Cách tiếp cận này sẽ mở ra một hướng mới trong phát triển các thiết bị Ubiquitous dùng T-Engine ở Việt Nam.*

REFERENCES

[1] Qing Li, Carolyn Yao, *Real-Time Concepts for Embedded Systems*, CMP book (2003).

[2] Phillip A. Laplante, *Real-Time Systems Design and Analysis*, An Engineer's Handbook, IEEE Press, Piscataway (1993).

[3] Elaine Rich, Kevin Knight, *Artificial Intelligence*, McGraw-Hill Higher Education (1990).

[4] Nguyen Minh Phong, Vu Tuan Thanh, Pham Tuong Hai, *T-Engine - Kiến trúc phát triển tiêu chuẩn mở cho các hệ thống nhúng thời gian thực*, Hội nghị khoa học công nghệ trường ĐH Bách Khoa lần 9 (2005).

[5] SH7760 T-Engine Development Kit User's manual, *T-Engine forum reference document* (2002).

[6] T-Kernel specification, *T-Engine forum reference document* (2002).

[7] T-Engine/SH7760 Development Kit Device Driver Manual, *T-Engine forum reference document* (2002).

[8] T-Engine Forum website www.t-engine.org