An Adaptive Power Management System for Wearable Devices based on CPU Mode Switching and DVS technique

ZIXIAN LU[†1] LEI JING[†2] ZIXUE CHENG[†2]

Abstract: With the development of mobile computing, we have created a wearable device named WonderRing, which consists of an acceleration sensor and wireless communicator, and can be used to detect user's hand gestures to control appliances, or communicate with others, etc. However, one problem is the power consumption of hardware components. In this paper we consider to use dynamic techniques to reduce energy consumption not only at CPU level, but also try to save energy of the overall system.

Keywords: Wearable computing, embedded system, wearable devices, power saving, CPU mode switching, DVS.

1. Instruction

With ubiquitous and wearable computing, wearable technology shares the vision of interweaving technology into the daily life. Wearable devices will likely replace appliance controllers, communication terminals, portable computers, and become the main portable electronic devices of the next generation. It has features such as compact and simplicity of operator, which are different from the traditional equipment, improve efficiency, and save more time for us. In the near future, the range and frequency of using wearable devices may expand indefinitely.

Since battery-operated wearable devices have been widely used in ubiquitous computing and wireless communication applications, charging time and frequency is directly related to the evaluation of wearable devices. Maximizing battery lifetime is an important design factor for wearable systems, and it is also a big challenge for greatly improving user experiences.

We have constructed and produced a wearable device named WonderRing, while an important problem is the power consumption of hardware components. To prolong battery life, we have to save the power consumption on each level of the system.

In our vision of wearable device, it consists of SoC, Communicator, and Sensors, which are the main energy consumers. First of all, we construct the power consumption model based on the hardware architecture of the wearable device; then we propose a mode switching method according to running status of SoC based on DVS algorithm; moreover, we use the method to control the CPU in the normal working state of the power consumption at software level; furthermore, we improve the power-saving solution by considering the issues of tasks execution in low power, device sleep, and wake-up management. Given the status of the components, we propose synthetic power-saving system by the means of mode switching and adjustment to other components.

We also consider novel power control technology for user behavior-driven power control. Specifically, our system detects user behavior by WonderRing (WR for short). Our method can change the mode of SoC base on the patterns of the users' behaviors to get the balance between convenience and response time, and optimize energy saving.

†1 Graduate School of Computer Science and Engineers, University of Aizu, Aizuwakamatsu, Fukushima, 965--8580, Japan, m5171121@u-aizu.ac.jp
†2 School of Computer Science and Engineers, University of Aizu, Aizuwakamatsu, Fukushima, 965--8580, Japan

2. Related research

The wearable device WR which we created, could be used to control appliances by finger gestures, or support simple communications, and record life log data, etc. As the introduction section above, WR is mainly constituted by the components as SoC CC1110, sensors, communicator, and circuits which integrate the components.

Power Mode	High speed oscillator	Low-speed oscillator	Voltage regulator(digital)
Configuration	A None	A None	A Off
	B 26 MHz XOSC	B Low power RCOSC	B On
	C HS RCOSC	C 32.768 kHz XOSC	
	D Both		
PM0	B, C, D	B, C	B
PM1	A	B, C	B
PM2	A	B, C	A
PM3	A	A	A

Figure1: Basic Mode of CC1110[1]

CC1110 has 4 basic modes as shown in **Figure 1**. The sensor detects movement of the finger and sends data to SoC, and the algorithm running on SoC will be in charge of gesture identification by processing the sensor data, and sends corresponding pre-defined control command to the communication unit. Currently, the battery life is only up to 4 to 5 hours, since there is no specialized power saving functions in WR except turning the power off manually.

To minimize the number of charge cycles and improve energy efficiency, we consider to constitute the power saving system of entire device in both hardware and software aspects. On the hardware side, we can make the SoC switch the model between the full power output, when it is used intentionally, and power saving mode when stopping using the WR, thus extending the overall time of using. On the software side, we consider using DVS algorithm to save the power consumption when the WR is on its working mode.

Conventional DVS (Dynamic Voltage Scaling) algorithm (Wiser [2] and Govil [3]) multi-cell model is based on

optimization, but no attempt on the battery accurate modeling of physical and chemical phenomena. In fact the behavior of different cells corresponding to different current load on the battery cell behavior under different circumstances for accurate modeling can improve battery efficiency and prolong its life, so further study of the characteristics associated with the battery DVS strategy is very meaningful.

DVS strategy is mainly based on the fact that: deterministic processor energy consumption and operating voltage was proportional to the square of the relationship. Therefore, under the premise of guaranteed performance, it is important to dynamically adjust the voltage to achieve the purpose of reducing energy consumption according to the system working state. DVS algorithm considers the battery characteristics not only on the state of the system in order to dynamically adjust the working voltage, but also the characteristics of the battery based scheduling policy to the rational use of the battery, so that the battery life is further extended. Luo et al. [4] studied the proposed DVS strategy by reducing the peak power of the battery to optimize energy consumption; Rakhmatov et al. [5] proposed the use of a battery self-healing effect of increasing the capacitance in order to save energy, these findings are based on the battery characteristics and obtain better energy efficiency.

Sukwon Choi et al. [6] modeled the unit behavior of the buttery corresponding cyclical current load, and divided the process of battery discharge into three distinct phases based on the strength of various factors, and provided different strategies in every phase to reduce the power consumption. But this method need some improvements.

(1) Using the model of unit behavior of the buttery corresponding non-cyclical current load will apply to a more general case.

(2) For the non-cyclical and random tasks, computational model based on the previous utilization of the processor can be changed by new strategy to accord with the characteristics of random tasks.

(3) By combining with DPM (Dynamic Power Management) strategy and using reasonable sleep/wake algorithm on the system units, it's able to reduce unnecessary consumption of energy.

3. System design

3.1 Basic idea

Human beings have many activities in daily life. We classify these activities into 2 kinds. One kind is slow, regular, or unconscious likes moving and sleeping that does not need to use WR. Therefore, we can save the power consumption of system for this kind. And the other kind is to control the appliances with WR, or use it to detect/record the data of jogging, eating, etc. Therefore, we have to use full working mode for detecting gestures or actions.

As we mentioned in the previous section, we can adjust the CPU mode according to situation of the device, and thereby reducing unnecessary power consumption. At first, we define 3 state modes of the systems, i.e. *work*, *sleep*, and *power-off*, as shown in **Fig. 2**. Each of them corresponds to the power consumption of situation of the device. On the work mode, the system provides full power under normal operating conditions. If the device has not detected intentional gestures for a certain time, the mode will be transited into *sleep* mode. If the device cannot detected any movement (intentional or unconscious ones for a long time, the mode will be changed into *power-off*.

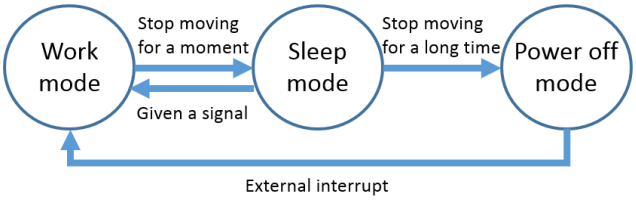

Figure2: State Transition Diagram of Power Control

3.2 Algorithm Design: changing power mode of SoC

We use PM_W, PM_S, and PM_O to denote the Power Mode of CPU in WORK state, SLEEP state and POWER OFF state respectively. The T_{I1} and T_{I2} represent the limited time (time out) of the mode switching as $PM_W \rightarrow PM_S$ and $PM_S \rightarrow PM_O$, respectively.

Variable:

Mode ∈ { PM_W, PM_S, PM_O };
Mode = PM_W;
PMTimer = 0; /* starting timer */

Input
Detection of Signal /* detect a signal of gesture */
Action
PMTimer := 0; /* reset the timer */
If Mode = PM_W **then**
Begin
End /* keep the mode being PM_W */
If Mode = PM_S **then**
Begin
Mode := PM_W; /* sleep to working */
End

Input
PMTimer = T_{I1}. /* no signal during period (0, T_{I1}) */
Action
If Mode = PM_W **then**
Begin
Mode := PM_S;
End

Input
PMTimer = T_{I2}. /* no signal during period (0, T_{I2}) */
Action
If Mode = PM_S **then**
Begin
Mode := PM_O;
End

Figure3: Algorithm for changing CPU's power mode

Just as shown in **Fig. 3,** our algorithm works as follows:

When CPU is in a power on mode (*work mode* or *sleep mode*), any signal received from sensors caused the sleep-timer be reset. If CPU is at *work* mode, and sleep-timer is reset, so that the value of the timer will not be equal to the time T_{I1}, and then CPU doesn't switch its mode. Otherwise CPU will be switched to *sleep* mode, when sleep-timer is reaching the setting time T_{I1}.

When CPU is in the *sleep* mode and sleep-timer is reaching the setting time T_{I2}, (i.e. the timer has not be reset by any detection of device signal,), CPU will be switched to *power-off* mode. In order to switch CPU from *power-off* mode to *work* mode, an external interrupt, such as switching on is necessary.

Let's consider an actual example:

We could control the appliances such as changing channel of the TV and adjusting volume by using WR in its full power working mode with short response time. The trigger from work to sleep is the case when no acceleration signal is received or the signal duration is more than the still time T_{I1} (e.g. 2 minutes).

Oppositely, the sleep mode could save most of the power consumption. We can imagine such a state that a user is looking at TV quietly without body/posture changing or moving. If any activity happened, the system will turn to the work mode immediately as soon as any signal of acceleration is detected. When the system does not change to the work mode for a longer stable time T_{I2} (e.g. 15 minutes), since it became *sleep* mode, the system will switch to the *power off* mode. Depending on the rate of usage on each mode, it is possible to continuous use more than 10 times by reducing power consumption.

When the system has changed to *power-off* mode, we can only press a switch to give an external interrupt to turn it on again.

3.3 DVS for saving power when CPU is working

According power curve characteristics of the dynamic characteristics of the battery, which is the key to design appropriate strategies: The DVS algorithm makes strategic switch based on residual power. The traditional approach divided the remaining battery power based on the two regions of rate capacity effect and the recovery effect. However, when we consider the charge of battery while using the device, the power curve characteristics becomes more complex. Therefore, we will also consider the transient characteristics of the curve, to determine its current status instantly, rather than dividing them into fixed area as mentioned in traditional strategy.

We tried to analyze characteristic of each time point, by observing and recording the voltage values at this time point and the previous point.

3.4 The total management system

Fig. 3 shows the total management system including the management of hardware part, management of software part, and charging part. To achieve this goal, we have to consider not only the coordination between the hardware and software layers, but also methods for collecting and charging power, temporary storage of excess electricity, reasonable control of the system clock and external interrupts, etc. Effective way to reduce energy consumption in embedded systems in real-time is currently a hot research. The power control issues give us a lot of challenges. We can consider user behavior, habits, and preferences, by using some methods of intelligence analysis and self-learning to make more improvement on power management.

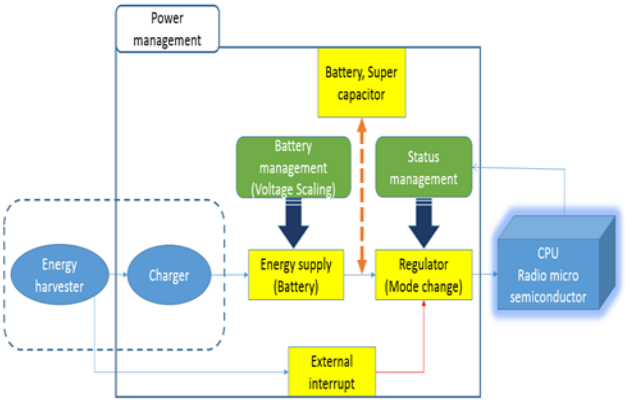

Figure4: Power management system of entire system

4. Toward Implementation

We have not completed the management system. Now we are focusing on the hardware layer management, and did some preparation for the implementation.

As we mentioned in the previous sections, we installed and tested the power saving system on the Foundation Board which has the same structures and components with WR and equipped with a few of external access ports. In the computer side, we choose windows8 OS which is currently widely used, 「IAR Embedded Workbench」— the development environment that includes a C/C++ compiler and debugger, and 「FlashPro-CC Elprotronic」− the USB Flash Programmer to install program in Foundation Board through hardware adapter (USB-FPA).

We use four AA batteries provide power to the entire board, we measured at the input voltage is stable 4.87V, and the voltage which continuous supply SoC is 3.30V. When the equipment is operated in *work* mode (PM0), we detected the current on SoC is 8.16mA (MCU (Micro Controller Unit) running at full speed (26mHz), XOSC running. No Peripherals). When switching the mode to *sleep* mode (PM1), it is changing to 0.58mA (Digital regulator on, High Speed RCOSC and crystal oscillator off. 32.768kHz XOSC, POR and ST active. RAM retention). When SoC is switched to the Power-off mode (PM3), the detected current has fallen below 1μA (No clocks. RAM retention. Power On Reset (POR) active).

According to the measured data, we can see that the current in *sleep mode* becomes about one-sixteenth of the *work mode*, and the power consumption difference value is the exponential relationship with the difference of current. While in the actual using case, we still need to calculate the degree of power saving

according to the user's usage, but the extension of use time and buttery life is predictable.

5. Conclusion

Right now, the power management system for wearable devices we have achieved is only the first stage of our expected. As mentioned before, now we have done some experiment for the design of hardware layer for power saving. We need fully implement the algorithm for control the mode of hardware, and develop the software control based on DVS to control and coordinate the hardware systems in order to achieve saving energy as much as possible in every specific situation of battery.

Reference

[1] True System-on-Chip with Low-Power RF Transceiver and 8051 MCU (Rev. H), Texas Instruments Incorporated, 2014, http://www.ti.com/lit/ds/symlink/cc1110f32.pdf

[2] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced CPU energy, Proceedings of USEN/X Symposium on Operating Systems Design, Monterey, California, United States, 1994. pp. 13–23.

[3] K. Govil, E. Chan, H. Wasserman, Comparing algorithms for dynamic speed-setting of a low-power CPU, International Conference on Mobile Computing and Networking, Berkeley, California, United States, 1995. pp. 13–25.

[4] J. Luo, J.K. Niraj, Battery-aware static scheduling for distributed realtime embedded systems design, Proceedings of ACM/IEEE Conference on Design Automation, Las Vegas, Nevada, United States, 2001 pp. 444–449.

[5] D.Rakhmatov, S. Vrudhula, Energy management for battery-powered embedded systems, ACM Transactions on Embedded Computing System 2 (3) (2003) pp. 277–324.

[6] Sukwon Choi, Hojung Cha, Rhan Ha, A selective DVS technique based on battery residual, Microprocessors and Microsystems, Yonsei University, 2005 pp. 1–10