Journal of Physics: Conference Series **490** (2014) 012115 doi:10.1088/1742-6596/490/1/012115

A new paradigm on battery powered embedded system design based on User-Experience-Oriented method

Zhuoran Wang[1] and Yue Wu[2,*]

[1]Solution Dept of iStarChip Semiconductor Technology (Shanghai) Co., Ltd., Shanghai, P.R.C

[2] Department of Chemical and Life Science Engineering, Virginia Commonwealth University, Richmond, VA 23284 USA

Abstract. The battery sustainable time has been an active research topic recently for the development of battery powered embedded products such as tablets and smart phones, which are determined by the battery capacity and power consumption. Despite numerous efforts on the improvement of battery capacity in the field of material engineering, the power consumption also plays an important role and easier to ameliorate in delivering a desirable user-experience, especially considering the moderate advancement on batteries for decades. In this study, a new Top-Down modelling method, User-Experience-Oriented Battery Powered Embedded System Design Paradigm, is proposed to estimate the target average power consumption, to guide the hardware and software design, and eventually to approach the theoretical lowest power consumption that the application is still able to provide the full functionality. Starting from the 10-hour sustainable time standard, average working current is defined with battery design capacity and set as a target. Then an implementation is illustrated from both hardware perspective, which is summarized as Auto-Gating power management, and from software perspective, which introduces a new algorithm, SleepVote, to guide the system task design and scheduling.

1. Introduction

The concept, user experience, has been attracting increasing attention since the revolutionary commercial portable device introduced by Apple Ltd in 2007. As nobody wants to charge their devices every several hours, the 10-hour sustainable time user experience has been brought up and become the facto standard (or reference) for smart phones and tablets since the successful introduction of the iPad in 2010.

Generally, there are two ways to increase battery sustainable time. One is to use a big battery with high capacity, which is widely used by android device vendors. The idea is easy but proved to be a big challenge for production design and is against the desire for thin and tiny commercial portable device with a reasonably long sustainable time. Numerous efforts have been made to increase the energy density since the commercialization of the first Li-ion battery by Sony in 1991[1]. However, the improvement is not satisfied enough up to now [2].The second method of increasing battery sustainable time is to reduce the total system power-consumption, which involves hardware and software integration. This method takes battery as a constant factor in the system and focuses on theories and

* Corresponding author: wuy@vcu.edu

approaches for a) evaluating the lowest system power-consumption with which the device can still provide strictly defined functionality and b) the practical implementation. Compare with the first approach, reducing the total system power-consumption can increase the battery sustainable time significantly without changing the device volume, extend the battery life time, and provide a more accurate state of charge indication in full temperature range (0~45°C).

This paper will apply the second method and reason a practical embedded system design paradigm based Top-Down User-Experience-Oriented modeling, which will be illustrated from both software and hardware perspectives.

2. 10-hour User Experience

The 10-hour User Experience describes a standard from human feelings, which could be ambiguous and arbitrary. To make it a real standard, a precise, practical and reasonable engineering definition is required.

C-rate is the charge or discharge current, which equals in Amperes to the rated capacity in Ah numerically. For example, the C-rate is 1100 mA in the case of an 1100 mAh battery, while the 0.2 C and 0.1 C are 220mA and 110mA respectively[1]. With C-rate, the 10hour User Experience can be interpreted as the remaining run-times (t_r) of a battery under 0.1 C discharge rate in average. The real value of 0.1 C current is determined by the design capacity of the battery. The bigger the battery is, the higher the 0.1 C current value will be. Typically, a discharge current which is less than or equals 0.2 C is regarded as low discharge rate. By using low discharge C-rate, 5-10hours battery sustainable time is expected, meanwhile the battery life-time is extended and the state of charge indication accuracy is also improved[1][2]. By using the non-replaceable Li-ion Polymer batteries, devices can be designed even thinner and lighter than usual, and keep available for at least one year.

3. General Model and Working Pattern

The 10-Hour User-Experience set the target power-consumption. To reach the goal, the easy-to-get idea Sleep As Soon As Possible is modelled and interoperated into guidelines first. Building on system modularization, new paradigm creates several models for different system levels and environments. As a general modeling guideline, the client of service provider applies Sleep As Soon As Possible strategy to the target component based on the idle signal(s) provided by services.

3.1. Component

In this paper, the component is a black box providing strictly defined services for client(s). The component always provides idle signal and the method for power-control at the same time. From the power-consumption perspective, the component is passive, providing idle information to the outside and waiting for power-management control.

Idle signal is provided by the component and used to indicate the life-time of self-determined service(s) in the component. Self-determined services is a kind of services that are requested, controlled or triggered by the client(s) and the life time is determined once started, for example, the UART (Universal Asynchronous Receiver / Transmitter) transmission service. By contrast, the life time of the none-self-determined services is not guaranteed. It provides information for client(s) but typically triggered or controlled by others, for example, the UART receiving service. The component should provide idle signal as the busy flag of self-determined services and provide clue(s) to client(s) for power-management if necessary.

The power-management is performed at client side. The target object is not only energy but also including power-related supplies, such as clock source in hardware, CPU time in software and etc. To make it short, power-supply will include all power-related supplies throughout this paper unless special statement is made. For those components consisting of self-determined services, once the idle signal is asserted, the power supply will be cut off immediately, which is the most straightforward strategy. This strategy works for some cases but not all, since repowering may take time, power and/or the worst, the functionality. The power can be switched frequently if the cost is small or negligible,

while sophisticated algorithm should be carefully applied to prioritize the functionality delivering if the cost is large. For those components containing none-self-determined services, the general strategy is the same, but clues are taken into consideration, a) if the life-time of none-self-determined service is determined, the clue will enable the power-supply and hence the idle signal will be used to indicate the service lifetime; b) if the life-time is undetermined, short-term power supply will be applied for each clue or the power of the target component will be manually controlled by application.

3.2. Power-Consumption Component Categories

Table 1 lists several different power-consumption component categories based on the information complexity of idle signal(s), style of power-management and the power-supply types.

Table 1. Power-Consumption Components Categories

	Idle	Style	Supply	Example
Basic	1	On/Off	Single	Simple Software Tasks, divider resistance in A/D conversion circuit, and etc.
Composite	n	Multi-Levels	Single	Microcontrollers Clock Management System
Multi-Layer	n	Multi-Levels	Multiple	Microcontrollers Sleep Modes, A/D Convertor, and etc.

Basic power-consumption component is the smallest unit for power-management. It provides one idle signal and on/off switch for specified power-supply, i.e. most of the software tasks are basic power-consumption components, and the divider resistance in A/D conversion circuit can also be considered as a basic power-consumption component as it is passive and always idle. Composite power-consumption component is a set of components with the same power-supply type, and, hence the set of internal idle signals is able to provide either a simple wired-and idle signal or detailed working status. It's possible to line out all power-switches of internal components but the flexibility introduces significant complexities at the same time, so typically, control templates are designed in certain order, which are referred as power-management levels. The clock management system in microcontrollers is the typical application for composite power-consumption component. A set of components with the same type of power-supply is defined as a power-layer. If a component supports more than one type of power-supply-managements, it is a multi-layer power-consumption component. Analog IPs in IC design always involve clock management and power-management at the same time. User can gate the clock to reduce the dynamic power-consumption when idle and shut-down the energy-supply when it's disabled or not used for certain time to reduce the leakage current.

3.3. Burst Working Pattern

If certain power-management strategy based on idle signal(s) applied to a target component, this part of system behaves in a burst working pattern in some level. With a given component, power-consumption is the function of time under the specified strategy, represented as $f(t)$. The burst working pattern is the shape of $f(t)$ in a selected time-window. So the average power consumption is represented as

$$g(t) = \frac{\int f(t)dt}{t} = \frac{E(x_0)\cdot t_0 + E(x_2)\cdot t_1 + \cdots + E(x_n)\cdot t_n \ (n \epsilon N^+)}{t} \quad (1)$$

where $E(x_n)$ is the power-consumption at level x_n in a component and t_n is the duration time of the specified level. t_n is fixed once the system design is settled. With a given component, the $g(t)$ will be a

bounded function for both self-determined and none self-determined services if the service working time is limited. So, we can further get

$$g(t) = \frac{\int f(t)dt}{t} = \frac{E(x_0)\cdot t_0 + E(x_2)\cdot t_1 + \cdots + E(x_n)\cdot t_n \ (n\epsilon N^+)}{t} = E(x_0)\cdot r_0 + E(x_1)\cdot r_1 + \cdots + E(x_n)\cdot r_n \ (n\epsilon N^+) \quad (2)$$

where r_n is the time ratio for the corresponding power-consumption level. Equation 2 elaborates the essential method for evaluating system minimal power-consumption while delivering strictly defined functionalities and guiding the system design to approach the evaluation result. The system power-consumption can be reduced by dividing big components into necessary sub-components with different power consumptions, increasing the power-management levels and increasing the ratio for low-power consumption levels. Equation 2 also indicates that it is possible to balance the power-consumption by increasing system performance to reduce the time ratio.

With a given component, if the service working time is undetermined, the service should be evaluated together with the object, which triggers and/or controls the none-self-determined service(s), and, hence new integrated-component is generated. If the life-time of the combination is determined, Equation 2 can be applied. If the combination is still undermined, the iteration should be repeated again and again until self-determined service is generated. For some cases, users become the final objects who control the undetermined components such as your smart phones. In those cases the general system power-consumption is controlled by human behaviours.

4. Practical Implementation

In real application, hardware modules provide idle signals to implement auto-clock gating power-management to reduce the dynamic part of power-consumption. Meanwhile, different depths of sleep models are designed to gate unused power-domain with sophisticated strategy and avoid the repowering overhead to reduce the static part of power-consumption caused by leakage current.

In order to enter sleep mode as soon as possible, event-driven techniques is applied to all tasks in software design, so that the task life-time is determined, and self-determined services are always triggered by events. If all tasks complete their services and no event is triggered, idle state will be identified, in which the sleep mode should be entered. To make sure the deepest sleep mode is entered, and system can still wake up by specified events, all sleep sensitive tasks are required to vote just before any sleep mode entered and as the vote result, the shallowest sleep mode will be used. This algorithm is called SleepVote.

5. Conclusion

The new paradigm proposed in this paper provides identical modelling method and guidelines for software and hardware implementation at the same time. By employing it, the theoretical minimal working power-consumption can be evaluated soon after the system is modelled and/or designed with the guideline of burst working pattern. The result can be compared with the target average power-consumption defined by 10-hour user experience battery sustainable time standard to help the decision making for product design with convincible data.

References

[1] H.J Bergveld, W.S. Kruijt, P.H.L Notten, Battery Management Systems, Design by Modelling, Philips Research Book Series, Kluwer Academic Publishers, Boston(2002)

[2] Valer Pop, Henk Jan Bergveld, Dmitry Danilov, Paul P.L Regtien and Peter H.L Notten, Battery Management System, Accurate State-of-Charge Indication for Battery-Powered Application, Philips Research Book Series Volume 9, Springer Science Business Media B.V(2008).

[3] Elaine Rich, Automata, Computability and Complexity Theory and Applications, PEARSON Prentice Hall, ISBN 0-13-228806-0

[4] Ken Pugh, Prefactoring, 1005 Gravenstein Highway North, Sebastopol, CA95472, O'Reilly Media, Inc. ISBN 0-596-00874-0