Journal of Circuits, Systems, and Computers
© World Scientific Publishing Company

INTELLIGENT POWER MANAGEMENT FOR EMBEDDED WI-FI DEVICES[*]

WEIYIN HONG[†]

Department of Computer Science and Technology, East China Normal University,
Shanghai, China[‡]
wyhong@ecnu.cn

XIN KUANG

Department of Computer Science and Technology, East China Normal University,
Shanghai, China
xkuang@ecnu.cn

JIANHUA SHEN

Department of Computer Science and Technology, East China Normal University,
Shanghai, China
jhshen@cs.ecnu.edu.cn

TONGQUAN WEI

Department of Computer Science and Technology, East China Normal University,
Shanghai, China
tqwei@cs.ecnu.edu.cn

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

With the increasing deployment of Wi-Fi devices in portable embedded systems, the low power design at system level has attracted considerable research attention in the recent past. In this paper, based on hardware features and software architecture of the embedded Wi-Fi devices, we focus on dynamic power management, dynamic frequency scaling and their influences upon the system power and performance. We propose effective and realizable system power management solution and application modes under various application requirements, such as response, bandwidth and speed. Experimental results show that the proposed solutions can achieve significant energy savings.

Keywords: Embedded Wi-Fi; low-power, dynamic power management, dynamic frequency scaling.

1. Introduction

With the increasing demands for mobility and portability of embedded devices, Wi-Fi that plays an important role in wireless communication field has been widely applied to

[*] This work was supported in part by the Fundamental Research Funds for the Central Universities under the grant 78220021.
[†]
[‡]

embedded systems. However, Wi-Fi modules introduce high power consumption. Heat dissipation due to the high power consumption significantly degrades system reliability. In addition, in most applications deployed in fields, the lifetime of a system mainly depends on the lifetime of the battery of the system. Hence, energy efficiency has become a first-class challenge in the design of these systems.

The hardware of a system has significant impact on the power consumption of a system. Bill Moyer summarizes low-power techniques applied at many levels of the design hierarchy[1]. And more hardware related low-power works were introduced in Ref [2] and Ref [3]. Dynamic power management[4] and dynamic frequency scaling are two system level power management techniques. Dynamic power management algorithms selectively turn off idle system components to reduce power consumption or turn on required ones to recover system performance[5, 6]. Dynamic frequency scaling algorithms dynamically reconfigure the processor operating frequency depending on application demands[7, 8].

Low-power software schemes are used to schedule various hardware and software modules to minimize the power consumption without compromising system performance. Power management schemes are divided into timeout-based policies, predictive policies and stochastic policies[9]. Timeout-based policies turn off or turn on a component after a fixed time interval[10]. Predictive policies make reliable decisions about future events that the system turn off or turn on some components and how long to be in active or inactive state. These decisions depend upon the past history of the system workload[11]. Stochastic polices assume the system can be modeled as Markov chains are more complicated[12, 13]. Besides, because of nondeterministic variations in device parameters, Ref [14] takes these effects account into system-level power analysis and designs two power management schemes to reduce the power. Several researchers have shown how to schedule[15], how to guarantee appropriate levels of fault-tolerance[16], and how to measure the workload[17] in the dynamic voltage scaling system.

However, most works mentioned above are theoretical studies without considering practical constraints in real applications. In other words, most of them[12, 13, 14, and 16] are too complicated to be applicable for embedded devices, especially for tiny and smart devices. Ref [6] found that the request inter-arrival times in the active state for WLAN card and hard disk follow exponential distribution, and idle periods observe Pareto and exponential distribution in experiments. Based on the time-indexed discrete-time Markov decision processes (SMDP) mode, Ref [6] presented time-indexed Markov chain SMDP model. Simulation experiments show better performance when compared to the default windows timeout policy. However, a device has different workload and performance in various operating environments and user requirement is different in various applications. This is needed to set the proper operating mode and strategy to save the energy upon on operating environments and user requirements. Ref [18] designed a power management unit which makes predictions to resume by collecting and analyzing information on the access patterns to I/O devices when the system is running. The experimental results show the high correct prediction ratio on accessing disk, but lower on accessing network.

However, this solution does not consider saving the power when the system is active and is not well suited for energy optimization of wireless communication devices.

In this paper, we combine dynamic power management and dynamic frequency scaling schemes to optimize power consumption of embedded Wi-Fi devices. We describe the basic strategy of dynamic power management that systematically turns off and turns on system components, discuss how to update system timer, and then propose the intelligent power management scheme which dynamically sets the proper time to be in active or sleep states according to pervious data exchange records. Moreover, based on the general analysis of hardware and software architecture of embedded Wi-Fi devices, we design several low-power application modes which include predictable power management and dynamic frequency. Experimental results show that the system running proposed scheme consumes significantly less energy when compared to the system without the proposed scheme.

The rest of the paper is organized as follows: Section 2 describes the basic strategy of power management, Section 3 proposes the predictive strategy of the power management, Section 4 analyses the hardware and software architectures of embedded Wi-Fi devices, designs low-power application modes of the embedded Wi-Fi devices and describes an application case, Section 5 gives our experimental results, and Section 6 concludes the paper.

2. Basic Strategy of Power Management

For simple power management, the system is divided into a fully functional active state and a low-power sleep state. The time the system is in sleep state is defined as the sleep time, and the time it is in active state is defined as the active time. As is shown in Fig.1, when system is in active state, if there is no task to process, it will go into the sleep state. When the system is in the sleep state, any wake-up event can wake it up and it will return to the active state and execute related operations. If the system becomes idle, then it goes into the sleep state again.

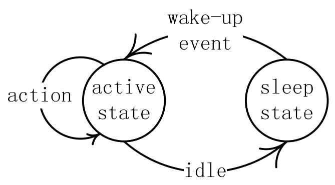


Fig. 1. The state transition diagram of the basic strategy

There are two events that wake up the system. One is periodic wake-up event that periodically wakes up the system to execute related operations. This event depends on the constant sleep time. The other one is non-periodic wake-up event that wakes up the system to process urgent tasks. The system responds more quickly to non-periodic wake-up event when compared with the periodic wake-up event.

2.1. *System Timer Update*

System timer that maintains the system timing plays a vital role in stable operation of operating system and protocol stack. The regular timing keeps system working stably and synchronizing normally with external systems. Because all or most of its components are turned off in sleep state, the system can not update the system timer. Periodic wake-up event and non-periodic wake-up event have different effects on timing, so it is necessary to discuss respectively how to update the timer.

Let SLEEP_TIME denote the constant sleep time. When the system is woken up by periodic wake-up event, clearly, its actual sleep time is SLEEP_TIME and then the timer is updated to incorporate the SLEEP_TIME. When the system is woken up by non-periodically wake-up, the actual sleep time is less than or equal to SLEEP_TIME. This wake-up event occurs randomly and its probability is uniformly distributed, so the average time estimated $\overline{T}$ is given by

$$\overline{T} = \frac{\sum_{i=1}^{N} T_i}{2} \{i, N \in Z \mid 1 \le i \le N, 0 \le T_i \le SLEEP\_TIME\} = \frac{SLEEP\_TIME}{2} \tag{1}$$

In Eq. (1), $\overline{T}$ represents the average value time estimated, T_i represents the actual sleep time, *i* represents the number that system is waked up by non-periodic wake-up event, and *N* represents the sampling number. The timer is incremented by $\overline{T}$ in this case.

Periodic wake-up event does not disturb system timing. The response time of the system that wakes up by this type of event is long. The effect of non-periodic wake-up event on system is opposite with the effect of the former. They are applied to the power management together, which can guarantee the high efficiency of the system operation.

2.2. *The Relationship between Active Time and Sleep Time*

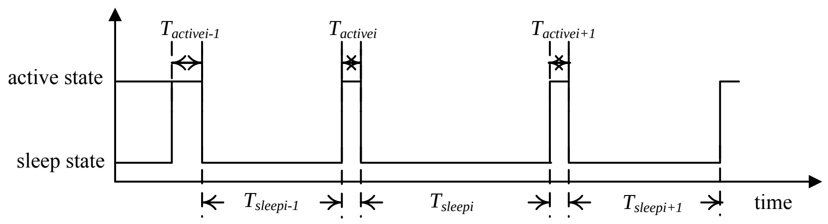


Fig. 2. The state sequence diagram of the basic strategy

The relationship between active time and sleep time impacts on the power consumption the performance of the system. In this subsection, the relationship is discussed as follow. As a result, the sleep time should be as long as possible for reducing power consumption.

Fig.2 shows an alternating sequence of active state and sleep state. Let T_{sleep} denote the actual sleep time and T_{active} denote the actual active time.

It is assumed that the power consumption of power management is ignored. So the average power consumption $\overline{P}$ of the whole system is discussed as follows.

$$\overline{P}=\frac{\sum_{i=1}^{N}(P_{active}\times T_{activei}+P_{sleep}\times T_{sleepi})}{\sum_{i}^{N}(T_{activei}+T_{sleepi})}\{i,N\in Z\}=\frac{P_{active}\times\overline{T_{active}}+P_{sleep}\times\overline{T_{sleep}}}{\overline{T_{active}}+\overline{T_{sleep}}} \tag{2}$$

In Eq.2, P_{active} represents the power consumption in the active state, P_{sleep} represents the power consumption in sleep state, i represents how many times the system have been in the sleep state or the active state, and N represents the number of sleep state cycles in a power management system. The power optimization rate R of the power management is given by

$$R=1-\frac{\overline{P}}{P_{active}}=1-\frac{\overline{T_{active}}}{\overline{T_{active}}+\overline{T_{sleep}}}-\frac{P_{sleep}\times\overline{T_{sleep}}}{P_{active}\times\left(\overline{T_{active}}+\overline{T_{sleep}}\right)} \tag{3}$$

As shown in Eq. 3, when the sleep time T_{sleep} is much larger than the active time T_{active}, the total power consumption of the system is close to the power consumption in the sleep mode.

3. The proposed Predictive Strategy of Power Management

It is assumed that if there is an action in a time slot, the probability of another action is 1 in the next slot. It is also assumed that if there is no action in a time slot, the probability of another action is 0 in the next slot. Based the above assumptions, predictive strategy is described as follow.

When a task arrives, the active time of the system is increased exponentially or linearly in the current state and the sleep time is halved. If there is no task in the active time, the sleep time is extended exponentially. The active time and sleep time of the system are described in the following subsections.

3.1. *Dynamic Update of Active Time*

When a task arrives, it is needed to extend the active time of the system to ensure that the task can be finished on time in the current active state. And the active time of the system is just lower bounded.

The system is in active state at the system startup. The active time T_{active} is equal to the original active time marked T_{active_orin}, which is defined as the low-bound of T_{active}. The increment component of the active time is $T_{expend\,a}$. At the time when the first task arrives, the T_{active} is updated to $2^0\times T_{expend\,a}$. At the time when the N task arrives, T_{active} is updated to $2^N\times T_{expend\,a}$. When more than N tasks arrive, instead of increasing exponentially, the T_{active} increases in linear with the number of tasks.

Fig.3 shows the curve of T_{active} changes in this algorithm. There is no maximum about T_{active}. It is necessary to keep increasing T_{active} to finish tasks. When system finishes the task and enters sleep state, the system resets T_{active} to T_{active_orin} and prepares for increasing it in next active state.

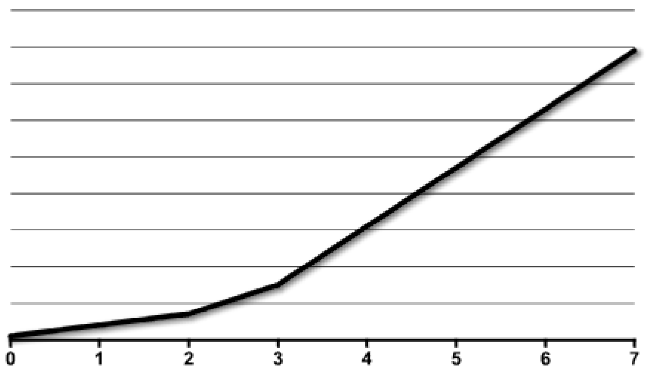


Fig. 3. The curve of T_{Active} change

3.2. *Dynamic Update of Sleep Time*

The system is woken up by non-periodic wake-up event, which means that there is an urgent task to complete, so the sleep time will be not updated at this time. When the system is woken up by periodic wake-up event to be in the sleep state and there is no task to process during this active time, which means that it may be idle during a period of time in the future, so it can sleep longer to reduce power consumption. The sleep time of the system is typically lower and upper-bounded. The upper-bounded time keeps the system from always being in the sleep state.

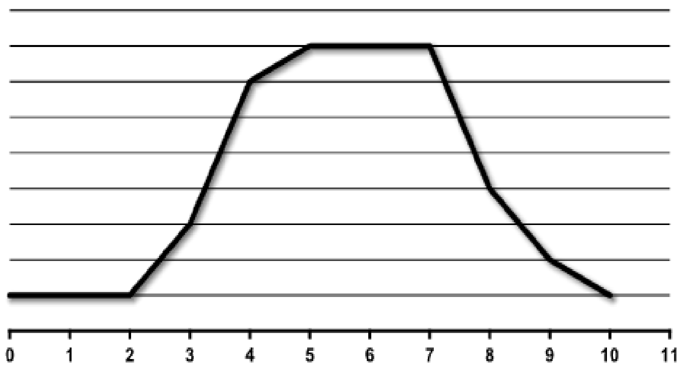


Fig. 4. The curve of T_{Sleep} change

At the beginning of system start-up, the sleep time T_{sleep} is equal to the original sleep time T_{sleep_ori}, which is defined as the lower-bound of T_{sleep}. The add/drop component of the sleep time is $T_{expend\ s}$. At the first time when there are no tasks during the active time, T_{sleep} does not change. At the second time, T_{sleep} is updated to $2^0 \times T_{expend\_s}$. It will increase exponentially until it exceeds the upper-bound of T_{sleep}.

Fig 4 shows a legend of increment and reduction of the sleep time. When the active time is increasing, the sleep time should be reduced to increase frequency of response to events. Whenever the active time arrives, the sleep time will not be halved until it less than T_{sleep_ori}.

4. Implementation of the Intelligent Power Management Scheme on Embedded Wi-Fi Devices

Any embedded device is composed of the hardware and the software. In order to implement the proposed intelligent power management scheme on an embedded Wi-Fi device, the hardware and software architecture of embedded Wi-Fi devices are analyzed and power characteristics are summarized in this section. According to these characteristics and the actual use of embedded Wi-Fi devices, the whole system of the embedded Wi-Fi device is divided into several low-power application modes. The system selects the mode that is suitable for a particular application, and minimizes the power consumption in the selected mode.

4.1. *Hardware and Software Architecture of Embedded Wi-Fi Devices*

An embedded Wi-Fi device is a communication device. It also can be a data processing module in a system. It consumes significant power of a system and the energy saving is achieved by using power efficient modes.

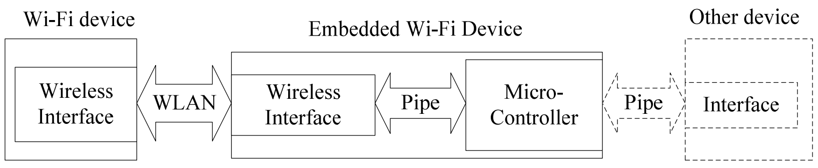


Fig. 5. The application model of embedded Wi-Fi Device

Fig.5 shows the model of the embedded Wi-Fi device. The dotted part is not enabled in some applications. There are more than two main hardware components. One is a micro-controller where TCP/IP protocol stack runs. The other one is the wireless signal transceiver chip. Without loss of generality, low-power modes of some widely used micro-controllers and RF chip are summarized.

Table 1. STM32 low-power mode characteristics[19, 20]

Mode	Clock	Current	Wake-up Condition	Wake-up Time
Sleep	CPU clock off	26mA(72MHz)	Any interrupt	1.8μs
Stop	All 1.8V domain clock off	11μA	Any EXTI interrupt	5.4μs
Standby	All 1.8V domain clocks off	2μA	Special wake-up method	50μs

Table 1 shows three low-power modes, power consumptions, and wake-up conditions and time of STM32F103 MCU from STMicroelectronics. When the system is in standby mode, it can save significant power, but wake-up condition is strict and wake-up time is too long. Based on these comparisons, micro-controller low-power mode characteristics such as response speed, interrupt method, power consumption and state recover, are summarized in Table 2.

Table 2. Low-power mode summary of micro-controller

Mode	Response speed	Interrupt method	Power consumption	State recover
Sleep	Fast	Many	Low	Needed
Stop	Normal	Single	Normal	Needed
Standby	Slow	Single	High	No

Beside, the low-power mode of RF chip, such as 88W8686 from Marvell, is in the DeepSleep mode. The chip turns off the WLAN subsystem in DeepSleep mode. It cost at least 1 second to recover from DeepSleep.

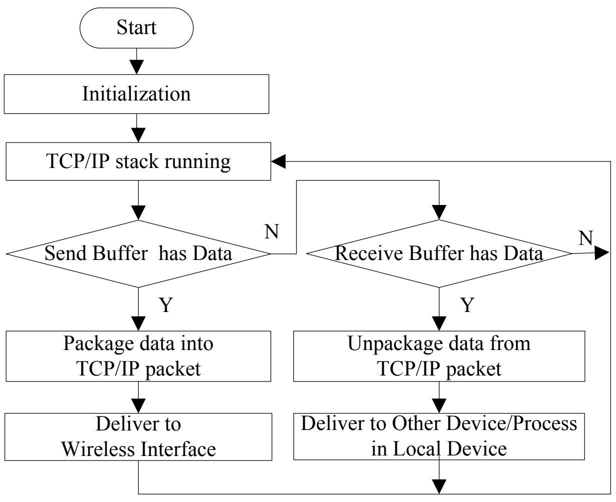

Start
Initialization
TCP/IP stack running
Send Buffer has Data
N
Y
Receive Buffer has Data
N
Y
Package data into TCP/IP packet
Unpackage data from TCP/IP packet
Deliver to Wireless Interface
Deliver to Other Device/Process in Local Device

Fig. 6. Flow chart of embedded Wi-Fi device

As shown in Fig. 6, the system will initialize clock, peripherals and TCP/IP protocol stack of embedded Wi-Fi device. Then the system goes into the main loop after initializations. System will handle a variety of protocol stack operations, and then checks whether there are data in send and receive buffer. If send buffer have received data from external module, the system will package the data and deliver it through TCP/IP protocol stack. If receive buffer gets data from WLAN, the system extracts the data and sends them out through UART or processes directly in local device.

4.2. *Design of Low-power Application Modes of Embedded Wi-Fi Device*

In this paper, low-power application modes of the embedded Wi-Fi device include low-power modes of the micro-controller and wireless RF chip. When a device is running, it switches among modes to satisfy the requirements of an application.

The low-power mode of an embedded Wi-Fi device is divided into three low-power modes, including sleep mode, supersleep mode and deepsleep. In sleep mode of the embedded Wi-Fi device, the micro-controller is in sleep mode and wireless RF chip is in full functional operation state. In supersleep mode of the embedded Wi-Fi deice, the micro-controller is in sleep mode and the wireless RF chip is in its low-power mode, and system works under the basic power management strategy. In deepsleep mode of the embedded Wi-Fi deice, the micro-controller is in standby mode and the wireless RF chip is in its low-power mode. Table 3 shows low-power modes of embedded Wi-Fi device and it reflects the influence of several aspects, such as response speed, power and application scenario.

Table 3. Low-power modes of embedded Wi-Fi device

Low-power mode	Mode of MCU	Mode of RF Chip	Response speed	Power compumption
Sleep mode	Sleep	Normal	Fast	High
Supersleep mode	Sleep	DeepSleep	Slow	Low
Deepsleep mode	Standby	DeepSleep	Very slow	Very low

The low-power mode determines the sleep degree and sleep time of the device in sleep state. Generally, in order to minimize power consumption of the whole device, the micro-controller can change the operating frequency to satisfy demands. Dynamic frequency determines the maximum load of the device in active state.

The operating frequency of the micro-controller is divided into four frequency levels in this case. The operating frequency of this micro-controller represents the operating frequency of the embedded Wi-Fi device. Dynamic frequency -0 means the micro-controller is in highest frequency. Dynamic frequency -1 means the micro-controller is in the second highest frequency which is the highest operating frequency of all peripherals. Dynamic frequency -3 means the micro-controller is in the lowest frequency which can meet the minimum requirements of all peripherals. Dynamic frequency -2 is the middle one between dynamic frequency -1 and dynamic frequency -3.

Table 4. Low-power application mode of embedded Wi-Fi device

The amount of data exchange	Bandwidth of data exchange	Response speed of data exchange request	Low-power application mode
Large	High	Fast	Full functional operating mode
Large	High	Slow	Sleep mode
Large	Low	Fast	Dynamic frequency -1
Large	Low	Slow	Sleep mode + Dynamic frequency -1
Small	High	Fast	Dynamic frequency -2
Small	High	Slow	Sleep mode + Dynamic frequency -2
Small	Low	Fast	Dynamic frequency -3
Small	Low	Slow	Sleep mode + Dynamic frequency -3

We consider some factors in low-power application mode of embedded Wi-Fi device, including amount of data exchange, bandwidth of data exchange and response speed of data exchange request. Based on different demands of these three factors, Table 4 shows specific design of low-power applications modes. Because it is hard to wake up the device in suppersleep mode and deepsleep mode, these two modes are not suitable for low-power application mode in this paper.

4.3. *An Application Case*

In this subsection, we describe an application case of intelligent power management for embedded Wi-Fi device. Given a scenario where the amount of data exchange is small, bandwidth of data exchange is low and response speed of data exchange request is slow, the low power mode and frequency selection is Sleep mode + Dynamic frequency -3.

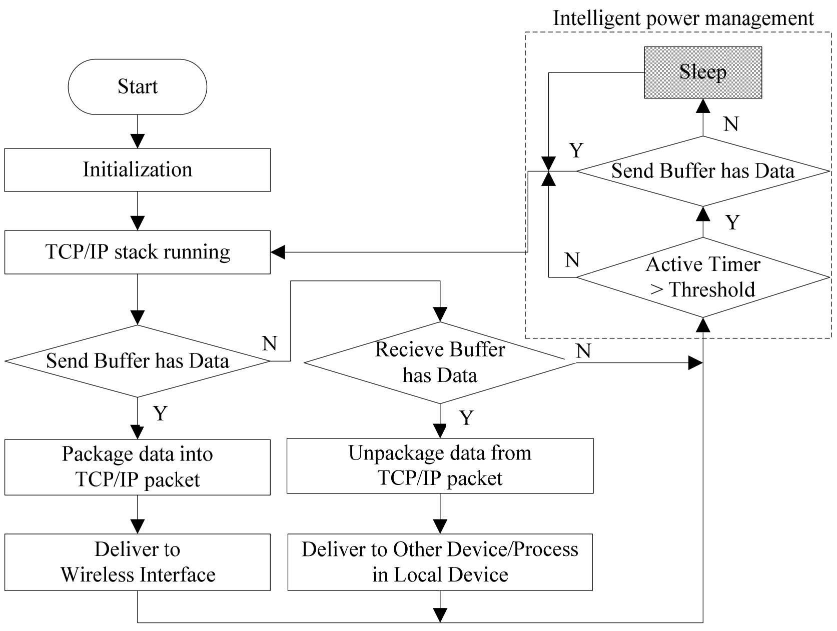


Fig. 7. Flow chart of an application case

Fig 7 shows the flow chart of the application case of intelligent power management for embedded Wi-Fi device. The proposed intelligent power management is described in the dotted box.

In this application case, the system will work at Dynamic frequency -3 in active state. In the main loop after system initiation, the management checks whether the active timer is timeout. If the active timer is timeout, the management will check again whether there are data in the buffer. If there are data in the buffer, the system will extend active time; otherwise, the system will go to sleep mode. When the sleep time is over, the system will wake up to run the TCP/IP protocol stack and check the data buffer again.

5. Experimental Results

In this section, an experimental platform running intelligent power management scheme was constructed and a customized testing platform is introduced to measure power characteristics. The initial values of T_{active} and T_{sleep} of the predictable power management scheme are 0.1s and 1s, respectively.

We measure the voltage waveforms of the load resistance on the testing platform using an oscilloscope, and calculate the current and estimate the power consumption. We estimate the power consumption at each frequency level and in each mode, then derive average power consumptions.

The experimental platform mainly consists of STM32F103C from ST Microelectricity and Marvell ® 88W8686. Fig 8 shows the block diagram of the embedded Wi-Fi device that runs predictive strategy and low-power application modes. It could be viewed as a UART to WLAN Converter[21].

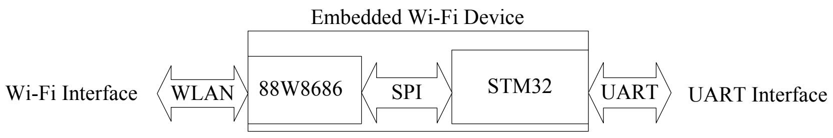


Fig. 8. Hardware block diagram of embedded Wi-Fi device

Fig 9 shows the circuit diagram of the testing platform. INA196 is a current shunt monitor. The voltage is measured using oscilloscope when the experimental platform is running. The measuring pins are Ground and OUT.

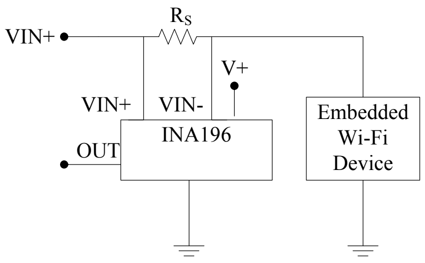


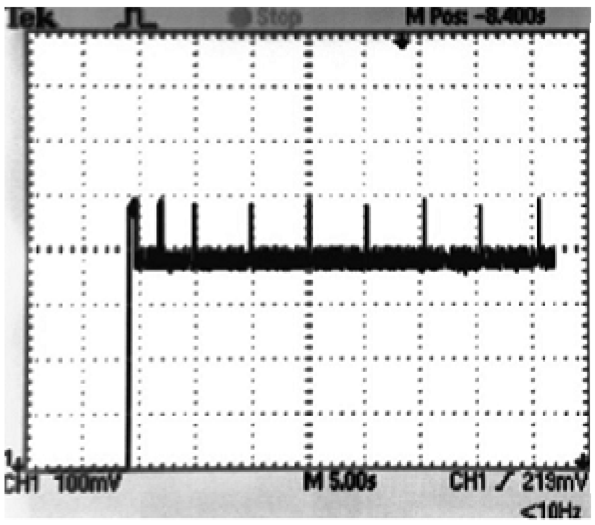
Fig. 9. Testing circuit diagram for embedded Wi-Fi device

The formula to calculate the current is given by

$$I_c = \frac{V_m}{R_m \times M_a}, \tag{5}$$

where V_m is the voltage value; R_m is the load resistance (0.22Ω), M_A is the magnification of the amplifier (20V/V); and I_c is the current. The measured voltage value from the test platform can be inserted into Eq.(5) to calculate the current value.

Fig10 shows the process where the device is becoming idle. T_{active} gradually narrows to original value. Meanwhile, T_{sleep} gradually widens to upper-bound and stops to widen for the performance.



CH1

Coupling
DC
BW Limit
Off
200MHz
Volts/Div
Coarse
Probe
1×
Invert
Off

Fig. 10. Waveform of Intelligent Power Management

Table 5 shows the average current value of each low-power mode.

Table 5. Average current of low-power mode

Mode	The average current value(mA)
Normal	214.8
Sleep	195.7
Suppersleep	2.1
Deepsleep	0.9

Table 6 shows the average current value of each operating frequency level. 72MHz is the highest frequency and 16MHz is the lowest one.

Table 6. Average current value of dynamic frequency

Frequency Level	Frequency(MHz)	The average current value(mA)
Dynamic Frequency -0	72	214.8
Dynamic Frequency -1	40	203.6
Dynamic Frequency -2	32	200.0
Dynamic Frequency -3	16	194.1

Table 7 shows average current values of each low-power application mode.

Table 7. Average current of low-power application mode

Low-power application mode	Current value in active state(mA)	Current value in sleep state(mA)	Average current value(mA)
Normal mode	214.8	None	214.8
Sleep mode	214.3	193.9	195.7
Dynamic Frequency -1	203.6	None	203.6
Sleep mode + Dynamic Frequency -1	203.6	189.3	192.3
Dynamic Frequency -2	200.0	None	200.0
Sleep mode +Dynamic Frequency -2	200.0	187.5	188.9
Dynamic Frequency -3	194.1	None	194.1
Sleep mode + Dynamic Frequency -3	194.1	184.8	185.7

6. Conclusions

In this paper, we propose an intelligent power management scheme for embedded Wi-Fi devices. We design several application modes to reduce the power in different application scenarios. Experimental results show that our management achieves significant power savings. As part of the future work, a better scheduler will be designed to dynamically change the mode of low-power application modes for energy savings. MAC layer and TCP/IP protocol stack also can be further optimized.

References

1. B. Moyer, Low-power design for embedded processors, *Proc. IEEE*, Vol. 89, no. 11, 2001, pp. 1576–1587.
2. A. P. Chandrakasan and S. Sheng and R. W. Brodersen. Low-Power CMOS Digital Design. *J. Solid-State Circuits*, Vol. 27, April 1992, pp 473–484.
3. Edwin de Angel and Jr. Earl E. Swartslander. Survey of Low Power Techniques for ROMs. *Proc. Int'l Symposium on Low Power Electronics and Design*, 1997, pp. 7-11.
4. L. Benini and G. De Micheli, Dynamic Power Management: design techniques and CAD tools, Norwell, *MA: Kluwer,*1997.
5. J. Lorch andA. Smith, Software strategies for portable computer energy management, *IEEE Personal Commun*., Vol. 5, June 1998, pp. 60–73.
6. T. Simunic et al., Dynamic Power Management for Portable Systems, *Proc. MobiCom '00*, August 2000, pp. 11–19.
7. Y. Zhang, X. Hu, and D. Chen. Task Scheduling and Voltage Selection for Energy Minimization. *Proc. IEEE DAC'02*, June 2002.
8. S. Das, S. Pant, D. Roberts, S. Lee, D. Blaauw, T. Austin, T. Mudge, and K Flautner, A Self-Tuning DVS Processor Using Delay-Error Detection and Correction, *J. Solid-State Circuits*, Vol.41, April 2006.
9. L. Benini, A. Bogliolo and G. De Micheli, A Survey of Design Techniques for System-Level Dynamic Power Management, *IEEE Trans. Very Large Scale Integration(VLSI) Systems (June 2000)*, Vol. 8, pp. 299–316
10. K. Huang, L. Santinelli, J.-J. Chen, L. Thiele, and G. C. Buttazzo. Periodic power management schemes for real-time event streams. *Proc. CDC/CCC, 2009*, pp. 6224.
11. Gaurav Dhiman, Tajana Simunic Rosing, Dynamic Power Management Using Machine Learning, *Proc. ICCAD '06*.

12. L. Benini, A. Bogliolo, G. Paleologo and G. De Micheli, Policy Optimization for Dynamic Power Management, *IEEE Trans. CAD (1999)*, Vol. 18, pp. 813.
13. Peng Rong, Pedram, M., Battery-aware power management based on Markovian decision process, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, July 2006, pp. 1337–1349.
14. Chandra, S., Lahiri, K., Raghunathan, Variation-Tolerant Dynamic Power Management at the System-Level, *IEEE Trans. Very Large Scale Integration (VLSI) Systems (Sept. 2009)*, Vol. 25, pp. 1220–232.
15. Jian-Jia Chen, Tei-Wei Kuo, and Hsueh-I Lu. Power-saving scheduling for weakly dynamic voltage scaling devices. *Algorithms and Data Structures,* 2005, pp. 338–349.
16. P. Pop, K. Poulsen, and V. Izosimov. Scheduling and voltage scaling for energy/reliability trade-offs in fault-tolerant time-triggered embedded systems. *Proc. CODES-ISSS'07*, Salzburg, Austria, October 2007.
17. Canturk Isci, Giberto Contreras, Margaret Martonosi, Live, Runtime Phase Monitoring and Prediction on Real Systems with Application to Dynamic Power Management, *Proc. the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, December 2006.
18. Y. S. Hwang, S. K. Ku, K. S. Chung, A predictive dynamic power management technique for embedded mobile devices, *IEEE Trans. Consumer Electron., vol. 56, no. 2*, July 2010, pp. 713-719.
19. STMicroelectronics, STM32F103 User Manual (2010), http://www.st.com.
20. STMicroelectronics, STM32F103 DataSheet(2010), http://www.st.com.
21. Ji Li, The Implementation and Optimization of Data Conversion between UART and WLAN, *Proc. Int. NSWCTC2010, Wuhan,* April 2009, pp. 365 - 368.