Reducing the system standby power of a personal computer

Te Huang¹, Ying-Wen Bai², Shih-Kuan Chao²

¹Graduate Institute of Applied Science and Engineering, Fu Jen Catholic University, No. 510, Zhongzheng Road, Xinzhuang District, New Taipei City, 24205, Taiwan
²Department of Electrical Engineering, Fu Jen Catholic University, No. 510, Zhongzheng Road, Xinzhuang District, New Taipei City, 24205, Taiwan
✉ E-mail: 497598030@mail.fju.edu.tw

Abstract: Most previous low-power personal computer designs have been either focused on power efficiency improvement or on software power management in the working states. This research aims to reduce the total standby power amount in the off state. The authors accomplish the goal of reducing the power consumption by a wake-up device which is replaced by a chip with a still lower consumption. The authors redesign the power circuit and cut off the power supply for the unnecessary chips, with the exception of the power needed for the specific chip used to wake up the system. The authors also turn off the power supply of the original power controller chip which is used to control the system’s power status. In addition, the authors use another low-power chip instead of the original one and reduce their system to over 99.3% leakage in an IC design at the component level by inserting leakage reduction of a system’s power status controller is turned off. Finally, as the authors use this low-power chip to manage the standby power source separately by means of the remote wake-up devices, the authors reduce further the standby power consumption to 4.4 mW in the power-off state by use of the various wake-up methods. The total result is an improvement of approximately 99.3%.

1 Introduction

Personal computers (PCs) are becoming visible everywhere in our daily life. They are equipped either with several remote wake-up functions which utilise both a keyboard and mouse that connect via either a PS/2 or a universal serial bus (USB) interface or with a network remote wake-up via a local area network (LAN). Extra standby power is needed when we use these functions. As these PCs remain mostly in the power-off state, operating for just a few hours a day, standby power consumption has become an important source of energy waste [1, 2].

The methods currently used to reduce these systems’ dynamic power consumption are using either hardware both to reduce the clock-swing of the processor and to decrease the leakage current or software to monitor and predict the power consumption [3, 4]. If a system is in the idle state or does not require high performance, dynamic scaling can lower both the operation voltage and the clock rate and thus decrease the leakage current to save dynamic power consumption [5–7]. On the other hand, reduction of a system’s static power consumption not only increases the efficiency of the power supply module but also cuts off the power supply if the system is in the idle state [8, 9]. Some experts have even considered estimating and minimising energy leakage in an IC design at the component level by inserting control points [10, 11]. Some systems can also be controlled remotely to resume the power supply in order to manage the standby power efficiently [12, 13]. The most useful method is by turning on the power supply of an electric home appliance only for the amount of time when we want to use it [14–16]. Based on this technique, in order to include a function which permits the user to turn off the system standby power, we redesign its power circuit and sequence so that the system remains turned off until the user wakes up the system via either a PS/2 interface, a USB, a LAN or a power switch to resume the standby power [17, 18].

We should aim to reduce the environmental impact of products everywhere, and this includes energy consumption. The international environmental protection agencies which define the rules for energy consumption are the US Environmental Protection Agency with its ‘ENERGY STAR Program Requirements for Computers’ and the European Commission with its ‘Eco-design requirements for energy-using products (EuP) directive’ [19]. Table 1 shows that according to the ENERGY STAR Program Requirements for Computers the power consumption of desktop PCs must be < 1 W in the power-off state.

Since 6th January 2010, according to the EuP Directive Lot 6, as implemented by the European Union, for electronic home appliances with standby power supply, which include desktop PCs, the standby power consumption must also be < 1 W in the power-off state [20]. The standby power consumption should be < 2 W if the electronic home appliances support display, remote control or wake-on-LAN (WOL) functions [21]. In addition the standby power consumption had to be reduced to 0.5 W after 6th January 2013 and to 1 W for electronic home appliances that support display, remote control or WOL functions. To reduce the standby power consumption there has been much research focusing on the circuit design, modifying the architecture of the power consumption or increasing the power factor to improve the efficiency of the AC-to-DC power regulator [22–24]. Some other ways that can be used to improve the efficiency of pulse width modulation of the DC-to-DC power regulator are either the zero voltage switching (ZVS) or the zero current switching (ZCS) method; these methods focus on circuit modification to improve both the ZCS and the ZVS to reduce switching loss [25, 26].

Although previous research has focused on the improvement of either the component or the power sub-system, our design aims to improve the system architecture. We know that the most effective way to save energy is either by removing the plug from the AC outlet or by turning off the mechanical switch when the electronic home appliance is not in use. Thus we propose to let the system go into a low-power state similar to when the mechanical switch is turned off. For this, we propose to modify the system power control method from the legacy design. Then we use another low-power controller to maintain the power sub-system. Thus every power sub-system can be maintained at a high standard in order to avoid power leakage by the low-power controller.
There are many methods used to wake up the system, but not every method is actually being used. Most users use only an IR remote controller to wake up their TV and a power switch to wake up their PC. Even other wake-up methods are enabled. Therefore we use the low-power controller to control each power sub-system individually. When the wake-up method is disabled, we turn off the related power sub-system to reduce the power consumption.

Our measurements show that standby power consumption is related to the number of wake-up methods. Our design is equipped with optional methods to wake up the system, but also allows the power switch, which has the lowest standby power consumption at 4.4 mW, now reduced from the previous 616.6 mW, to wake up the system, thus meeting most authorised rules. In addition we measure and compare our design with the deep sleep well (DSW) power saving method [27], which utilises a newly proposed platform that became available in the market in 2012; this new platform consumes 14.4 mW.

In Section 2 we analyse the chips that consume standby power and measure the power consumption of each. Then we describe both how the Deep S5 works and how to shut off their unnecessary power supply to reduce the standby power consumption. We then summarise how we redesign the power sequence to make sure that the system can boot normally. In Section 3 we compare the results of measurements between the DSW technique and our design. In the final Section we draw our conclusions and provide a look at future possibilities.

2 Design and implementation

There are three major chips in the modern PC design: the central processing unit (CPU), the North Bridge chip and the South Bridge chip. The CPU is the central control chip for the PC system. The North Bridge chip is connected from the CPU to the memory, the VGA, the high-speed I/O and the South Bridge chip. The South Bridge chip is a multi-function chip which integrates a memory, the VGA, the high-speed I/O and the South Bridge chip. The CPU is the central control chip for the PC system. The North Bridge chip is connected from the CPU to the memory, the VGA, the high-speed I/O and the South Bridge chip. The South Bridge chip integrates a lot of low speed I/O: USB, LAN, SATA, audio, advanced configuration and power interface (ACPI) power manager, PCIE and PCI bridges. The South Bridge chip must work correctly in the first G3 state, and then the power sequence can be maintained in the following power state. The South Bridge chip maintains the ACPI power state by asserting the S3# and S5# signals; it also maintains the power sequence of the CPU, the memory controller and other low speed I/Os whenever the system is in the S0 to S5 power state.

We measure the power consumption of every chip in the sleep states as shown in Table 2. The total 5VSB is 123.32 mA, which is approximately 616.6 mW.

Table 2 Comparison of power consumption in the sleep state

<table>
<thead>
<tr>
<th>Device</th>
<th>Current, mA</th>
<th>Power consumption, mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 LAN</td>
<td>50.82</td>
<td>254.1</td>
</tr>
<tr>
<td>2 south bridge</td>
<td>48.6</td>
<td>243</td>
</tr>
<tr>
<td>3 PS/2</td>
<td>7.8</td>
<td>39</td>
</tr>
<tr>
<td>4 USB</td>
<td>7.1</td>
<td>35.5</td>
</tr>
<tr>
<td>5 standby power control unit</td>
<td>2.5</td>
<td>12.5</td>
</tr>
<tr>
<td>6 miscellaneous</td>
<td>6.5</td>
<td>32.5</td>
</tr>
<tr>
<td>7 total</td>
<td>123.32</td>
<td>616.6</td>
</tr>
</tbody>
</table>

2.1 ACPI state machine and modified deep S5 state

According to the ACPI specifications there are five states in the standby mode, that is, S0/S1/S3/S4/S5, which may be described as follows [28]:

S0: ‘System on’ is the normal working state. The power consumption is at its highest in this state.

S1: ‘Power on Suspend’ in this state, the system turns off the monitor or the hard disk to save power.

S3: ‘Suspend to RAM’ in this state, the system data are stored in the main memory which consumes power to maintain these stored data.

S4: ‘Suspend to Disk’ in this state, all contents of the main memory is saved to the hard drive. The power consumption is equal to shutdown as there is no extra power needed to maintain the machine state.

S5: ‘System off’ is the system shutdown state, when the power consumption is at its lowest. Only the devices required to power-on the system consume power.

The system sleep state depends on the OS and the users. When the user sets the standby mode in the OS and the basic input/output system (BIOS), the OS sends a standby command request to the ACPI power manager. Then, depending on the different power stages, the ACPI power manager generates the S3#, the S5# and the power OK (POK) signals to control the system sleep state.

Fig. 1a reveals the sleeps states transition in the ACPI and indicates when the system initially starts up in the global system power state 3 (G3). With G3 the mechanical portion is off, the AC power source has been removed, and the system maintains the real time clock (RTC) by using only battery power. The battery is a button cell for the RTC and maintains the system configuration which is stored in the static random-access memory (SRAM). Therefore the users do not need to setup both the system configuration and the computer clock after the AC power has been lost.

When the AC power source is turned on, the 5 V standby power (5VSB) is ready, the resume reset (RSMRST#) goes high and the system then switches into the S5 state. When the S3# and the S5# are logic high, the system turns on the silver box power supply.
unit (PSU) and waits for the POK to indicate that the power is ready; the system then switches into the S0 state. When the system goes into the S1 state, the S3# and the S5# and the POK do not change. When users shutdown the system, the OS sends the standard ACPI S5 command to the ACPI power manager in the South Bridge chip. The South Bridge chip then goes into the shutdown sequence. When the sequence ends and the South Bridge chip is ready to shutdown, the S3# and the S5# signals will be sent out to the external power control circuit. When the S3# becomes low but the S5# and the POK remains high in the S0 state, the system switches and stays in either the S4 or the S5 state until both the S3# and the S5# change to high. The S0 and S1 states are controlled by the OS; the sleep states are controlled by the ACPI power manager, whereas the S4 and the S5 states of the sleep states are controlled by the OS.

Since we propose to turn off the whole system standby power, the power state switches into a new state which is similar to the G3 state. However, this new state is not exactly the G3 state; because of the ACPI controller, this standby state still requires standby power to wake up the system. We define the new power state as the Deep S5, which is not defined in the ACPI standard. Fig. 1b shows the transition of the sleep states with the Deep S5.

The ACPI specification Rev. 1.0, which was defined in 1996, defined the power state of PCs. This revision covered hardware design and software programming, including the specific application, BIOS, OS, driver and the firmware on the add-on cards. It also standardises the interface between the hardware and software or application. The open source or commercial GUI OS are all in compliance with the ACPI specification during the recent 10 years; when the user shuts down the system, the system generates the System Control Interrupt (SCI) to the OS. Then the OS sends out the standard ACPI command to the ACPI driver. Next the ACPI driver sets up the ACPI BIOS, the ACPI registers and the ACPI tables. The ACPI configurations affect the ACPI power manager contained in the South Bridge chip. The South Bridge chip then asserts the S3# and the S5# signals to notify the external control circuits to change the power state. Therefore the system power state can be managed by this ACPI scheme, as shown in the Fig. 2.

We use the standard ACPI S5 command and extend it as the Deep S5 power state. The system shutdown after the OS sends out the S5 command. When the system goes into the S5 state, the system power is shutdown; both the CPUs and the OS stop working. We can obtain this S5 state from the ACPI manager, and then we allow the system to enter the Deep S5 sequentially by our design. The OS recognises the system is in the Deep S5 state. When the system resumes from the Deep S5, it goes into the S5 first; then it wakes the system up, hence the OS will recognises it has resumed from the S5 state, not the Deep S5. Thus this approach can be integrated with a typical ACPI compliant open source or commercial GUI OS system. In contrast, if the OS does not support the ACPI command, the S5 command cannot be sent out. Thus as our hardware circuit will not receive the S5 state, the Deep S5 state will also not be successful.

### 2.2 Reducing the standby power consumption of the ACPI controller replacement

The system power status controller, standby power control unit and remote wake-up controllers consume standby power to wake up the PC from the sleep state. The power status of the system is managed by the South Bridge chip, and it provides both the S3# and S5# signals to the standby power control unit to control the power state. Fig. 3a shows the power consumption map in the sleep state of a previous typical design. The ‘Switch’ has the MOSFET which is dominated by the standby power control unit, which consists of some types of transistor-transistor logic that are used to switch the power source between VCC5 and 5VSB. The ‘Regulator’ has a voltage regulator which is used to change the 5 V standby power (5VSB) to 3.3 V (3VSB) for standby devices.

---

**Fig. 2 ACPI control flow in the OS**

**Fig. 3 Power consumption map in the sleep state**

*a* Previous typical design  
*b* Modified design for standby power saving

---


© The Institution of Engineering and Technology 2015
The power source of the PS/2 devices is 5V_PS2, which is controlled by the PS2_SEL signal to the MOSFET switch, and the 5V_USB for USB devices is controlled by the USB_SEL signal. The LAN chip and the South Bridge chip are powered by the 5 to 3 V regulators individually. The 3VSB and 3VSB_LAN are always on and not controlled by any power source.

As the South Bridge chip, which is the power state controller, is not allowed to cut off the power supply once it becomes a power manager, we need to use a different chip to manage the power state. We use another low-power application-specific integrated circuit (ASIC) as the ACPI controller to regulate the system power instead of the standby power control unit, and we also use a MOSFET as a switch to turn off the system standby power (SYS5VSB) of the South Bridge chip, the LAN, the PS/2 keyboard and the mouse as well, with the exception of the ACPI controller (ASIC). The standby power system modification is shown by the grey boxes in Fig. 3b.

The ACPI state communication between the South Bridge chip and the ACPI controller are the S3# and the S5# signals. The ACPI controller maintains the ACPI state after the power supply of South Bridge chip is turned off.

To manage the system power state we use an independent voltage regulator (3VA) for the ACPI controller. From this point on the system power state of the controller is replaced by the South Bridge chip as the ACPI controller. In the modified structure the standby power consumption of the ACPI controller and miscellaneous circuits is decreased to 18.9 mW from 616.6 mW, which is a reduction of the standby power consumption by approximately 96.9%.

2.3 Reducing the standby power consumption of the USB controller replacement

The original USB wake-up method depends on the South Bridge chip, the USB keyboard and the mouse. As the USB controller is integrated into the South Bridge chip, we have to supply the standby power for the whole of the South Bridge chip when we begin to use the USB wake-up function.

The power consumption of the original USB wake-up structure is approximately 278.5 mW. To decrease this power we propose to use a different, simplified USB controller to replace the complex South Bridge chip [29], as shown in Fig. 4.

The power consumption can be reduced to 171.5 mW when we use another low-power PCIe-to-USB controller. Moreover, we need to redesign the power state transition sequence of the system standby power structure.

2.4 ACPI power sequence and modified power sequence of deep S5 state

Under the current Advanced Technology eXpanding (ATX) structure the power supply of a PC can be divided into two categories: general power and standby power [30]. Except for 5VSB, which supports the standby power under both ‘system on’ and ‘system off’, all other power supplies are general power and are only supplied while the system is on and not in the shutdown state. 5VSB is supplied to the resume system: the keyboard, the mouse, the network, the modem and so on. Apart from devices used for resuming, the ACPI controller also consumes standby power to manage and regulate the power in the power-on, shutdown and standby modes of all states. The legacy power-on sequence is shown in Fig. 5. When the battery of the RTC (VBAT) has been installed, the RTC reset (RTCRST#) switches to high after any 200 ms delay to wait for the power to stabilise, and the system then goes into the initial G3 state. When the standby power (5VSB) is ready, the resume reset (RSMRST#) becomes high, and then the system goes into the S5 state.

When the user presses the power switch, the power switch input signal (PSIN#) switches to low and after its de-bounce the standby power control unit sends the power switch output signal (PSOUT#) to wake up the system power status controller which is within the South Bridge chip. The South Bridge chip sends out the S3# and S5# signals when the power state machine transitions from S5 to S0. The standby power control unit drives the Power Supply ON signal (PSON#) at a level sufficiently low enough to turn on the PSU. Then, as all the main power supplies including 3.3, 5 and 12 V are ready to serve, the system state transitions into the S0 state.

When the user presses the power switch again, the South Bridge chip sends the S3# and S5# signals to the standby power control unit, both to turn off the power and to transition the power state machine from S0 to S5 when it receives the PSOUT# signal. The PSU then turns off the main power. When the PSON# signal goes high, the system goes into the S5 state again.

As the Deep S5 power state which we defined is not subject to the ACPI standard, we have to redesign the system power sequence to make the Deep S5 work as a normal system power sequence. We modify the power sequence as shown in Fig. 6a.

The sequence is the same as the normal power-on sequence when the system goes to S5 from G3 until the system goes to S5 from S0, as shown in Fig. 5. When the user presses the power switch to shutdown the system, the system goes into the S5 state normally. Then the timer in the ACPI controller starts to count down for 5 s. When the time is up, the ACPI controller de-asserts the RSMRST# and turns off the SYS5VSB to let the South Bridge go into a G3-like state. The power remains on only in the ACPI controller, and the system turns into the Deep S5 which we have previously defined.

We also modify the wake-up sequence, as shown in Fig. 6b. When the ACPI controller receives the wake-up events (PSIN# or WAKE#) in the Deep S5 state, the controller first activates the SYS5VSB and as a result the RSMRST# is ready. Meanwhile the system goes into

---

**Fig. 4** Comparison of USB wake-up structure

**Fig. 5** Power sequence
the S5 state. The ACPI controller then sends the power switch output signal (PSOUT#) to wake up the South Bridge chip, and the system resumes with the S0 after a delay of 200 ms by PSON#. The multiple remote wake-up functions have been disabled in the Deep S5 state, because the modified standby power scheme supports only the power switch for power-on whilst the PS/2 keyboard, the mouse, the LAN and the South Bridge chip are not able to wake up the system in the Deep S5 state. We redesign the standby power scheme and control the PS/2, LAN and South Bridge chip power supplies, respectively. Thus we can turn off the power of the South Bridge chip which consumes the most standby power.

2.5 System power state control of the ACPI controller

Based on the ACPI controller and the modified power sequence we can cut off the power of the South Bridge chip, thus reducing the power consumption while the USB wake-up function is enabled.

Fig. 7 shows the block diagram of the ACPI controller. The ACPI controller is powered by a 3VA power source which is a non-controlled voltage regulator from the 5VSB. The ACPI state controller begins to work when the 3VA reaches 2.8 V. It is the main control unit in the ACPI controller and monitors both the 3VSB and 3VA power states to obtain the current power state, and it also monitors the S3#, the S5# and the PSU OK (PSUOK) signals to obtain the current2 system state.

The power supply of the wake-up devices (LAN, keyboard, mouse or IR remote controllers connected through USB and PS/2 ports) controlled by the ACPI state controller through the power enable (PWREN#) signals is latched in the output buffer. The wake-up (WAKE#) signals allow the wake-up devices to assert an interrupt event to switch the ACPI state controller into the S5 state from the Deep S5 state. The wake-up conditions are programmable through the system management bus and are recognised in the internal battery powered SRAM [31].

The RSMRST# asserts when the 3VSB comes and reaches > 2.8 V with a 120 ms delay, and also when the system is controlled by the ACPI state controller. The RSMRST# signal is de-asserted by the ACPI state controller whereas the system is in the Deep S5 state. It re-generates depending on the 3VSB reaching 2.8 V from the Deep S5 to the normal S5 state.

The PSOUT# signal comes through a de-bounce circuit as a result of turning on the PSIN# power switch. The ACPI state controller monitors the PSIN# signal and controls the buffer depending on
the power state. The PSOUT# buffer passes through into the normal S5 state and is disconnected in the Deep S5 state to re-generate the PSOUT# signal to wake up the system.

The PSON# signal, normally a reserved signal from the S3# signal, is activated at a low level to turn on the PSU, and then the PSU supplies the main power for the system and asserts the PSUOK signal when the

---

**Fig. 8** External latch circuits for the ACPI controller

*a* Block diagram of the external latch circuits

*b* 3VA power control circuit for the ACPI controller

*c* X_PWREN# external latch circuit for the ACPI controller
main powers are ready. The PSON# needs to be isolated from the S3# signal of the South Bridge chip in the Deep S5 state, because the power supply of the South Bridge chip is turned off in that state. The PSON# re-connects to the inverted S3# signal to turn on the PSU when the system returns to the S5 state from the Deep S5 state.

2.6 Multiple wake-up support in deep S5 state

We modify the multiple wake-up functions which are marked as thick dotted lines in Fig. 3b. We use the independent PWREN# signals to control the power supply for the wake-up devices. We use the LAN_PWREN# and SB_PWREN# signals to manage the voltage regulators of the LAN and the South Bridge chip. As the PS2_PWREN# and USB_PWREN# control the MOSFET switch of the PS/2 and USB ports, the keyboard, mouse or IR remote controllers can be turned off independently. When we disable all wake-up functions except the power switch, since the SYS_PWREN# can be the main switch of the 5VSB to turn off the SYS5VSB, we can reduce the standby power consumption of the miscellaneous circuits.

Fig. 9 Waveform of Deep S5 Design

- Deep S5 counter timeout
- Deep S5 wake-up
2.7 External latch circuits for ACPI controller

The ACPI controller does not need the active management in the Deep S5 state, needs to latch the X_PWREN# signals and waiting for the WAKE# or PSIN# signals to wake the system up. In order to reduce the standby power consumption of the ACPI controller, we propose to cut off the power supply of the ACPI controller as well. Thus we add external latch circuits to latch the X_PWREN# signals hence the ACPI controller can be turn off. The modification is shown by the grey boxes in Fig. 8.

We use the low voltage CMOS D-type positive-edge-triggered flip-flops (U1) to control the 3VA power supply. The Q1 is the MOSFET switch for the ACPI controller power controller. When the circuit starts from the initial G3 state, the preset pin (PRE) and clear pin (CLR) of U1 was designed for ensure that initial output level of Q is high and overline Q is low. Thus we can use the Q to control Q1 turned on the 3VSB to 3VA at the initial state, and then the ACPI controller is powered. When users shutdown the system, the system goes into the S5 and then goes into the Deep S5 after 4 s. The ACPI controller asserts the SB_PWREN# high signal to clock pin (CLK) of U1, the U1 latch a low signal from D to the Q and the overline Q becomes high and hold this state, the Q1 turns off, then the 3VA power off and latch by the U1 until next power on. The ACPI controller is now shutting down, but the wake-up devices are still workable once the power states have been latched in other D-type flip-flops.

When users wake up the system by pressing the power switch or any other wake-up devices, the PSIN# goes low first while the user presses the power switch. The ACPI controller first turns on the 3VA power supply and asserts the RSMRST# after a 120 ms delay while the 3VSB gets ready. It then asserts the PSOUT# to the South Bridge chip to wake up the system. It is just like the power sequence in Fig. 6b.

We also compare the total wake-up delay, which is approximately 280 ms for the modified power sequence. This 280 ms delay is connect the LAN_PWREN# to the clock pin, once the LAN_PWREN# programmed to high by the ACPI controller, the U2 latch the low from D to the Q and the Q which means the LAN_PWREN_L# becomes high and hold the level. The power supply of the LAN chip has been turned off. The LAN_PWREN# state has been latch to the LAN_PWREN_L# to maintain the power supply in the Deep S5 state. When users wake up the system, the U1 turns on the ACPI power supply first and then the RSMRST# goes high after the 3VSB ready and a 120 ms delay. The RSMRST# connects to Q2 and generates a low pulse through C4, and U2 receives the PRE signal. Then the Q changes to high and Q changes low to turn on the power supply of LAN chip. The external latch circuit was shown in Fig. 8c.

3 Results and measurement

3.1 Power sequence measurement of our design

Fig. 9a shows the waveform measurement of the Deep S5 power sequence control. Once the user presses the power switch to turn off the system, the PSIN# goes low first and the PSOUT# goes low after an 80 ms delay for a de-bounce. The ACPI controller starts to count down for 5 s. The RSMRST# goes low until the time is up. Then the ACPI controller turns off the MOSFET to cut off the SYS5VSB.

Fig. 9b shows the wake-up power sequence of the Deep S5. The PSIN# goes low first while the user presses the power switch. The ACPI controller first turns on the SYS5VSB, and asserts the RSMRST# after a 120 ms delay while the SYS5VSB gets ready. It then asserts the PSOUT# to the South Bridge chip to wake up the system. It is just like the power sequence in Fig. 6b.

We also compare the total wake-up delay, which is approximately 280 ms for the modified power sequence. This 280 ms delay is...
designed to avoid problems with unstable power. The 280 ms delays are caused by: PS_IN# to PS_OUT# de-bounces 70 ms, PS_OUT# to SYS5VSB enables 60 ms, SYS5VSB soft-start 30 ms and 3VSB > 2.8 V delay 120 ms. Then our system asserts the RSMRST#. The system goes into the S5 state from the Deep S5 state, as shown in Fig. 10.

The multiple level DC to DC regulators start working when the SYS5VSB are turned on, and the regulators have to become active one by one in sequence to avoid any leakage current inside the chip, because the leakage current may damage the chip during an incorrect power sequence.

3.2 Standby power measurement of our design

Fig. 11a shows the block diagram for the measurement of standby power consumption. The power source of the 5VSB is from the silver box PSU. The 5VSB first connects to a high-precision multi-meter to observe the input current (I\(_{\text{IN}}\)) and then to an oscilloscope to measure the input voltage (V\(_{\text{IN}}\)) and the control signals. It then connects the 5VSB to the system (DUT). The ASIC placed at the right-upper location of the DUT. Fig. 11b shows the actual photo of the measurement.

First, we set up the wake-up condition and boot the system into OS. We wait for 30 s after the system booted into OS to avoid that the system becomes unstable. Then we shutdown the system and wait for 20 s till the capacitors are discharged. We check the input current and calculate the power consumption. Finally, we fill the result into every power consumption table in this paper.

With this new implementation the user can choose any method both to wake up and to determine the signal to control the power for the device. We have measured the standby power consumption in terms of all wake-up functions, and the test results are shown in Table 3 which summarises the power consumption using different wake-up methods.

Item 1 is an original configuration; these wake-up functions consume a total of 616.6 mW. In item 2 we use the ACPI controller to replace the power management function of the South Bridge chip. Then we turn off the power of the South Bridge chip. To keep all wake-up functions enabled, we reduce the power consumption to 479.8 mW. With item 3, as we disable the PS/2, the USB keyboard and the mouse wake-up functions and turn off the power supply for them, we thereby reduce the power consumption to 269.7 mW. With item 4 we disable the LAN, the PS/2 keyboard and the mouse wake-up functions and turn off the power supply. The power consumption is now 186.5 mW. With item 5 we disable the LAN, the USB keyboard and the mouse wake-up functions and turn off the power supply for both the LAN and the USB ports. The power consumption is now 54.1 mW. With item 6 we disable the LAN, the USB keyboard and the mouse wake-up functions and turn off the MOSFET switch of the SYS5VSB shown in Fig. 3b, so that the system allows only the power switch to wake up. Then we use an external latch to maintain power enable signals of the ACPI controller and turn off the ACPI controller, the total power consumption is 4.4 mW.

3.3 Measurement results of DSW power saving method

Fig. 12 shows the DSW power saving method. It uses a MOSFET as a switch to cut off the SYS5VSB while in the DSW state. The new South Bridge chip separates the 3VSB power planes into two power supplies (3VSB and 3VA).

The 3VA supplies the standby power control unit and part of the standby power control logic of the South Bridge chip. Before the system turns into the DSW state, the South Bridge chip asserts the

Table 3 Summary of consumption in different multiple wake-up selections of our design

<table>
<thead>
<tr>
<th>PS/2 wake-up</th>
<th>USB wake-up</th>
<th>LAN wake-up</th>
<th>Power switch wake-up</th>
<th>Current, mA</th>
<th>Power consumption, mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>enable</td>
<td>enable</td>
<td>enable</td>
<td>enable</td>
<td>123.32</td>
<td>616.6</td>
</tr>
<tr>
<td>2* enable</td>
<td>enable</td>
<td>enable</td>
<td>enable</td>
<td>95.96</td>
<td>479.8</td>
</tr>
<tr>
<td>3* disable</td>
<td>disable</td>
<td>enable</td>
<td>enable</td>
<td>53.94</td>
<td>269.7</td>
</tr>
<tr>
<td>4* disable</td>
<td>enable</td>
<td>disable</td>
<td>enable</td>
<td>37.3</td>
<td>186.5</td>
</tr>
<tr>
<td>5* disable</td>
<td>disable</td>
<td>disable</td>
<td>enable</td>
<td>10.82</td>
<td>54.1</td>
</tr>
<tr>
<td>6* disable</td>
<td>disable</td>
<td>disable</td>
<td>enable</td>
<td>0.88</td>
<td>4.4</td>
</tr>
</tbody>
</table>

*Implements the USB and ACPI controller replacement solutions
Table 4  Power saving measurement by DSW method

<table>
<thead>
<tr>
<th></th>
<th>PS/2 wake-up</th>
<th>USB wake-up</th>
<th>LAN wake-up</th>
<th>Power switch wake-up</th>
<th>Current, mA</th>
<th>Power consumption, mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>not supported</td>
<td>enable</td>
<td>enable</td>
<td>enable</td>
<td>132.65</td>
<td>663.3</td>
</tr>
<tr>
<td>2</td>
<td>not supported</td>
<td>disable</td>
<td>disable</td>
<td>enable</td>
<td>132.65</td>
<td>663.3</td>
</tr>
<tr>
<td>3</td>
<td>not supported</td>
<td>disable</td>
<td>disable</td>
<td>enable</td>
<td>103.51</td>
<td>517.6</td>
</tr>
<tr>
<td>4</td>
<td>not supported</td>
<td>disable</td>
<td>disable</td>
<td>enable</td>
<td>2.88</td>
<td>14.4</td>
</tr>
</tbody>
</table>

*System in the DSW state

Table 5  Power saving measurement of different platforms

<table>
<thead>
<tr>
<th></th>
<th>Original, mW</th>
<th>Disable PS/2, mW</th>
<th>Disable PS/2, USB wake-up, mW</th>
<th>Disable PS/2, USB and LAN wake-up, mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1960</td>
<td>1432.1</td>
<td>501.2</td>
<td>7.8</td>
</tr>
<tr>
<td>2</td>
<td>1320</td>
<td>1268.6</td>
<td>648.6</td>
<td>10.4</td>
</tr>
<tr>
<td>3</td>
<td>503.9</td>
<td>461.3</td>
<td>285.5</td>
<td>8.1</td>
</tr>
<tr>
<td>4</td>
<td>616.6</td>
<td>440.7</td>
<td>269.1</td>
<td>4.4</td>
</tr>
</tbody>
</table>

*The product we introduce in this paper.

suspend warning (SUSWRN#) signal to give note that the system will go into the DSW state. When the standby power control unit is ready to go into the DSW state, it returns the suspend acknowledge (SUSACK#) signal to the South Bridge chip. Then the South Bridge chip turns off the MOSFET switch of the SYS5VSB. Thus the standby power of the system is turned off, except for the 3VA which is needed to wake up the system in the future. From this point on the system goes into the DSW state.

Besides this redesign we have measured another reference board with both a South Bridge chip like ours and a new DSW power saving solution. The test results are shown in Table 4.

We found that this system will not cut off the standby power even if the wake-up functions are disabled, until the system turns into the DSW state. However, in our design, we can not only disable the wake-up functions individually but also turn off each standby power while the wake-up function is disabled. In the meantime we can enable any of the wake-up functions individually.

3.4 Measurement results of different platforms

We also measured three other platforms that implemented our solution, as shown in Table 5. The first column of Table 5 is the original standby power consumption. The PS/2, the USB, the LAN and the South Bridge chip are all powered in the S5 state. Column 2 is the power consumption of the PS/2 keyboard and the mouse wake-up function disabled, and the power source of PS/2 is shut down. Column 3 is the power consumption of the PS/2, with the USB wake-up function disabled. The power supply of the South Bridge chip is turned off. The last column is the power consumption of the PS/2, when both the USB and the LAN wake-up functions are disabled. Only the power switch can wake the system up. The standby power only supplies the ACPI controller in order to maintain the Deep S5 power state.

Case 1 is product A with a brand A chipset; the power consumption has been reduced from 1960 to 7.8 mW. Item 2 is product B with a brand N chipset. The power consumption decreased from 1320 to 10.4 mW. Item 3 is product C with brand I chipset, the power consumption dropped from 503.9 to 8.1 mW. Item 4 is product D with a brand I chipset which we introduce in this paper. The power consumption improved from 1960 to 7.8 mW. Item 2 is the power consumption of the Deep S5 state. We have also redesigned the system’s power sequence to ensure that the system allows the power-on function to be switched from the Deep S5 to a normal S5. Moreover we have redesigned the circuit to allow the ACPI controller to dominate the power of the PS/2, the USB and the LAN separately.

Since our design is a modification based on the ACPI specifications, it can be applied to common platforms. Most PC-based digital electronic home appliances are based on ACPI-compliant systems. These appliances are currently using a legacy standby power design. No matter what commercial GUI OS or open source OS they have installed, they can easily use our design to reduce the standby power consumption because in the standby state our design consumes only 4.4 mW, a power consumption reduction of approximately 99.3%.

Moreover, our design complies with the 2013 EuP Lot6 standard which stipulates that home appliances should have a consumption of less than 0.5 Watt in both the standby and the power-off states. In the future, in order to reduce the energy drain in the standby state and to prevent adding to the global warming problem, we have to redesign the system architecture to achieve a zero standby power consumption system.

5  Comparison of previous conference version

The major changes from the previous paper of 24th March 2011 are:

1. With reference to the DSW power saving method, we add a MOSFET as a switch from the 5VB to the SY5VSB, and it is controlled by the SYS_PWREN# of the ACPI controller. Although the wake-up method utilises only the power switch, we can turn off the SYS5VSB. Thus we can obtain a lower power consumption of 18.9 mW instead of 19.6 mW. (Chapter II. B).
2. We use an external USB controller instead of the power wasting South Bridge chip that was integrated with the USB controller while the USB wakeup function is enabled. Because of this modification the South Bridge chip can be turned off for all of the Deep S5 state. We can obtain a lower power consumption of 201.05 mW instead of the previous 319.3 mW for the USB wake-up function when it is enabled. (Chapter II. C).
3. We modify the power sequence of PSOUT# so that it takes place earlier than before. We move the PSOUT# signal from the behind RSMRST# to behind the PSIN#. with a 70 ms de-bounce delay; thus the South Bridge chip can perform a wake-up function earlier. The wake-up delay is reduced from the previous 450 ms to 280 ms. (Chapter II. D).
4. We use an extra latch and an 74LVC74 both to latch the power control of the wake-up devices and to turn off the power supply of the ACPI controller. (Chapter II. G).
5. We re-design the 3VA power control circuit for the ACPI controller, hence the ACPI controller can turn off its power supply by itself and the 3VA can become active by means of either the PSIN# or the WAKE# signals. The 3VA power control circuit which controls the power supply of the ACPI controller can be both turned off and resumed properly. (Chapter II. G).
6. We add the waveform measurement for the modified system to verify the sequence of our design. (Chapter III. A).
7. We explain the detail of the 280 ms delay of our design. (Chapter III. A).
8. We implement our design in different platforms, and measure the standby power consumption. (Chapter III. D).
9. We can obtain a lower standby power consumption of 4.4 mW instead of the previous 19.6 mW. (Chapter III. B).

6 References

3 Leviq, D., Yazid, M., Kawaguchi, H., et al. ‘Half VDD clock-swing flip-flop with reduced contention for up to 60% power saving in clock distribution’. Int. Conf. of 33rd European Solid State Circuits, September 2007, pp. 190–193
6 Kwok, S. ‘Power-saving algorithms in electricity usage comparison between the power saving algorithms and machine learning techniques’. Int. Conf. of Innovative Technologies for an Efficient and Reliable Electricity Supply, September 2010, pp. 246–251
29 Kang, D., Yoo, J., Lee, J. ‘A new remote USB architecture for live migration of virtual machines in SoD (System-on-Demand) service’. Int. Conf. on Control Automation and Systems, October 2010, pp. 1143–1146