Power Management

Battery operated portable appliances impose tight constraints on the power dissipation of their components.

Such constraints are becoming tighter as complexity and performance requirements are pushed forward by user demand.

Reducing power dissipation is a design objective also for stationary equipment, because excessive power dissipation implies increased cost and noise for complex cooling systems.

IT IS EASIER TO SAVE MONEY THAN TO MAKE IT, SO CUT YOUR ENERGY CONSUMPTION & BE REWARDED WITH SMALLER BILLS TO PAY!

Power Management

Almost every portable electronic appliance is far more complex than a single chip.

Portable devices such as cellular telephones and laptop computers contain tens or even hundreds of components.

In most electronic products, digital components are responsible for only a fraction of the total power consumed. Analog, electro-mechanical, and optical components are often responsible for the largest contributions to the power budget.

Example (Laptop computer): On average, 36% of the total power is consumed by the display, 18% by the hard drive, 18% by the wireless LAN interface, 7% by non critical components (keyboard, mouse, etc.), and only 21% by digital VLSI circuitry (mainly memory and CPU).

Power Management Techniques

Draw a state diagram for the predictive shutdown mechanism of a pager. The pager wakes itself up once every 5 minutes for .01 second to listen for its address. It goes back to sleep if it does not hear its address or after it has received its message.

Replace incandescent globes with compact fluorescents, which use up to 80% less energy

Most Aggressive Policy (Eager Policy)

Turns off every system component as soon as it becomes idle.

Whenever the functionality of a component is required to carry out a system task, the component must be turned on and restored to its fully functional state.

Ensure that unused rooms are not lit.

Drawbacks of Eager Policy

The transition between the inactive and the functional state requires time and power.

As a result, the eager policy may be often unacceptable because it degrades performance and may not decrease power dissipation.

Example (highly simplified model of a hard-disk drive) : Consider a device that dissipates 2 W in fully operational state and no power when set into inactive state.

The opposite transition takes 2 s. During the transition, the power consumption is 4 W.

If the eager policy is chosen, the user will experience a 2-s delay *every time* a request for the device is issued after an idle interval.

Dynamic Power Management

Create a monster and try to control it!

"One of the most successful techniques employed by designers at the system level is *dynamic power management*.

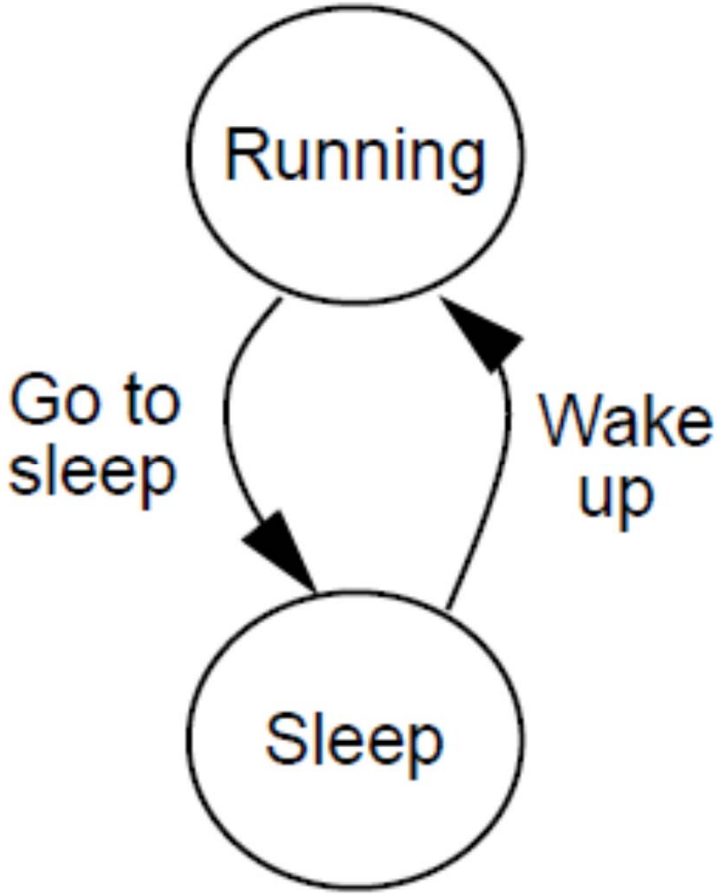
This technique reduces power dissipation by selectively turning off (or reducing the performance of) system components when they are idle (or partially unexploited).

Building a complex system that supports dynamic power management is a difficult and error-prone process.

Long trial-and-error iterations cannot be tolerated when fast time to market is the main factor deciding the success of a product."

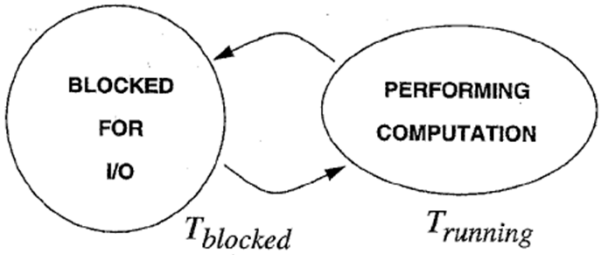
Use motion detector to turn off office computer when worker falls asleep on the job.

Energy Saving Using a System-Shutdown Technique



To obtain maximum heating or cooling without the cost, turn it off in areas where it is not needed.

Event Driven Applications and Power Management



As shown in Fig. 3, event-driven computations are in one of two states: they are either blocked while waiting for an I/O event, or are performing computation. When running on a dedicated CPU, an event-driven application will alternate between a *blocked* state where it stalls while waiting for external events such as a key press or a mouse click, and a *running* state where it will execute instructions to perform computation. If $T_{blocked}$ and $T_{running}$ are the average time spent in the *blocked* and the *running* states, respectively, then one can improve the energy efficiency by as much as a factor of $1 + T_{blocked}/T_{running}$ provided the system is shutdown whenever it is in the *blocked* state.

Six Paths to Longer Battery Life: These six technologies could save on smart phone power

IEEE Spectrum News, April 2012

- SMALLER TRANSISTORS
- RAZOR-THIN MARGINS
- ALL-DIGITAL PHASE LOCKING
- SMART CONVERTERS
- NEXT-GENERATION DYNAMIC RAM
- NEAR-THRESHOLD COMPUTING

Don't open the oven door too often to check food condition as each opening leads to a temperature drop of 25°C

SMALLER TRANSISTORS

Intel has packed 1.4 billion 3-D transistors onto Ivy Bridge, its next-generation processor.

The switch to less leaky 3-D transistors has given the new chip a big power boost.

The 22-nm Ivy Bridge chips can be run just as fast as the company's previous chips, but with an operating voltage that's 200 mV lower.

Intel has also incorporated designs at the circuit and core level to improve the chip's power management. A separate system-on-a-chip code-named Silvermont, based on the same transistor-making process, will be geared for mobile handsets.

Turn off electric stoves several minutes before the specified cooking time

RAZOR-THIN MARGINS

Engineers typically run chips at a higher voltage than needed in order to prevent clocking errors.

If chips had a way to detect errors and change their operating voltage on the fly, engineers could push chips to operate at the lowest voltage possible, saving power in the process.

The scheme, called Razor, is still largely stuck in academic circles. But researchers from the University of Michigan, in Ann Arbor, and Harvey Mudd College, in Claremont, Calif., showed that the approach works on an ARM Cortex-M3 processor, boosting energy efficiency by 60 percent. The team says it's the first implementation of a Razor-style scheme on a complete
commercial processor.

Do not switch on the power when TV and Audio Systems are not in use i.e. idle operation leads to an energy loss of 10 watts/device

ALL-DIGITAL PHASE LOCKING

Phase-locked loops—which lock in and track an input signal—are vital circuit elements that are used to sync modern processors to their clocks and pick up and transmit radio signals.

In the past, these circuits were built with analog components, but all-digital variants consume a tenth of the power and are easier to fabricate.

Mobile powerhouse Samsung presented a new improvement on the alldigital phase-locked loop, a 0.012-squaremillimeter circuit that consumes just 2.5 mW.

Intel showed off a version of the circuit, built with the company's 22-nm technology, that consumes as little as 0.7 mW.

A computer that runs 24 hours a day uses more power than an energy-efficient refrigerator.

SMART CONVERTERS

Another basic circuit headed for a low-power makeover is the switched capacitor, which is often used to convert analog signals to digital.

A team at Oregon State University, in Corvallis, debuted a low-power component that was inspired by the ring oscillator, a common test circuit made of a loop of inverters.

Another team at the National Chiao Tung University, in Hsinchu, Taiwan,found a way to save power by working on signals in two separate stages—one for crude processing and the other for fine-tuning—that can each be optimized.

Pull the plug and save. Battery chargers, such as those for laptops, cell phones and digital cameras, draw power whenever they are plugged in and are very inefficient.

NEXT-GENERATION DYNAMIC RAM

Memory makers Samsung and Hynix Semiconductor both unveiled details on the next incarnation of synchronous DRAM, the memory that drives today's processors.

The new generation, which goes by the name DDR4, boasts circuit tricks that let Samsung drop the supply voltage to its memory modules from 1.5 V to 1.2 V.

The modules also include better clocking and faster algorithms for encoding data to be sent to and fetched from memory. DDR4 may make its commercial debut as early as 2013.

Leave enough space between your refrigerator and the walls so that air can easily circulate around the refrigerator

NEAR-THRESHOLD COMPUTING

Academics have long toyed with the idea of operating chips at a point very close to the threshold voltage—the amount needed to switch a transistor on.

Now the scheme seems to be getting picked up by industry. Intel researchers discussed a 32-nanometer,Pentium-class chip they've built that can operate from 1.2 volts—de rigueur for today's processors—all the way down to 280 millivolts.

The chip's sweet spot for energy efficiency was 450 mV, just above the threshold voltage. At that level, Intel's chip ran slowly, at less than 100 megahertz, but it also consumed just about a fifth of the energy it did at 1.2 V.

Parallel processing could be used to pick up some of the slack in performance

Always use washing machines only with full loads

Energy Saving Using a System-Shutdown Technique

Let *R* be the running period

I the idle time period,

E the delay overhead of entering the sleep state from the running state,

S the sleeping time period,

W the delay overhead for resuming the running state from the sleep state (the wake-up process).

PR and *PS* are the power consumption values of the system in the running and sleep states, respectively.

PEW is the average power-dissipation overhead of entering the sleep state from the running state and resuming the running state from the sleep state.

Typically, $PR >= PEW >= PS$.

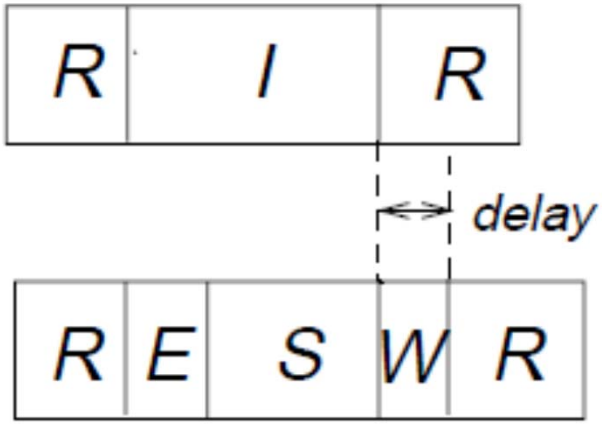
EG denotes the energy gain of the system.

Enter sleep state whenever idle period is detected.

Figure shows the first scenario in which the idle period is longer than the delay overhead of entering the sleep state from the running state (i.e., *I* >= *E*).

Since *I* >= *E*, the system will successfully enter the sleep state and resume the running state when a wake-up signal occurs.

The energy gain *EG* is *PR* x *I* – (*E* + *W*)x PEW - *PS* x *S* and the delay penalty is *W*



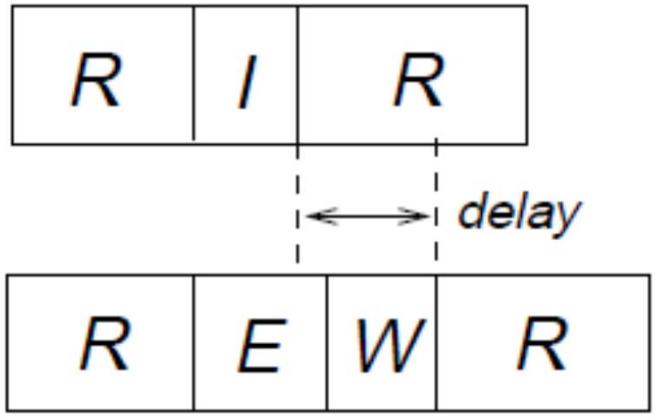
Take advantage of daylight by using light-colored, loose-weave curtains on your windows to allow daylight to penetrate the room. Also, decorate with lighter colors that reflect daylight!

Enter sleep state whenever idle period is detected.

Figure shows the second scenario in which the idle period is shorter than the delay overhead of entering the sleep state from the running state (i.e., *I* <= *E*).

In this case, the system will never enter the sleep state and will suffer a long delay penalty (*delay* = *W* + (*E* - *I*) and a negative energy gain

when *PEW* > *I* x *PR* /(*E* +W)



For energy saving, the idle period must be longer than the delay overhead for entering the sleep state from the running state (i.e., *I* >= *E*).

Minimum required idle period for achieving a positive energy gain,

$$I \geq E, \tag{1}$$

$$EG = I \cdot P_R - (E + W) \cdot P_{EW} - P_S \cdot S; \tag{2}$$

$$= I \cdot P_R - (E + W) \cdot P_{EW} - P_S \cdot (I - E); \tag{3}$$

$$= I \cdot (P_R - P_S) - E \cdot (P_{EW} - P_S) - W \cdot P_{EW} > 0, \tag{4}$$

$$= > I > \frac{E \cdot (P_{EW} - P_S) + W \cdot P_{EW}}{(P_R - P_S)}, \tag{5}$$

$$= > S_{th} = \frac{E \cdot (P_{EW} - P_S) + W \cdot P_{EW}}{(P_R - P_S)}. \tag{6}$$

The *exponential-average* approach

The *exponential-average* approach is one of the commonly used methods for the prediction of the idle period.

Using this method, the next CPU burst is predicted as an accumulative average of the measured lengths of previous CPU bursts.

Similarly, we can use the *exponential-average* approach to predict the next idle period by the accumulative average of the previous idle periods.

$$I_{n+1} = a \cdot i_n + (1 - a) \cdot I_n,$$

I_{n+1} New Predicted value

I_n Last Predicted value

i_n Latest idle value

a : attenuation factor in the range 0 to 1

The *exponential-average* approach :Limitation

$$I_{n+1} = a \cdot i_n + (1 - a) \cdot I_n,$$

I_{n+1} New Predicted value

I_n Last Predicted value

i_n Latest idle value

a : attenuation factor in the range 0 to 1

The *exponential-average* approach :First Problem

The prediction formula can effectively predict idle periods in most cases, except the occurrence of impulse-like idle periods (a sudden, very long idle period *I*3 occurs after continuous, nearly uniform idle periods)

For example, the user works on the system for a while and then goes to answer a telephone, resulting in the system idling for a long period of time.

Reasons: recall that our proposed prediction formula predicts the upcoming idle period by the accumulative average of the previous idle periods.

This underestimation is undesirable for energy saving, especially when the predicted value is lower than *Sth*. In this case the system will stay in the running state instead of power saving mode which results in a large amount of unnecessary power consumption.

$$I_{n+1} = a \cdot i_n + (1 - a) \cdot I_n,$$

The *exponential-average* approach :2nd problem

When predicting the idle period *I*4, which followed a long idle period, our proposed formula tends to overestimate the duration of the idle period

*I*4 < I4predicted.

It is also undesirable because the system may falsely enter the sleep state and suffer unnecessary power consumption and delays.

I_n Last Predicted value

i_n Latest idle value

a : attenuation factor in the range 0 to 1

$$I_{n+1} = a \cdot i_n + (1 - a) \cdot I_n,$$

A Solution to the First Problem

Use a watchdog scheme to periodically monitor the current idle period.

When an idle period occurs, the system predicts the duration of the idle period. If the predicted value is lower than *Sth*, the system stays in the busy waiting state and starts up a timer to trace the actual idle period.

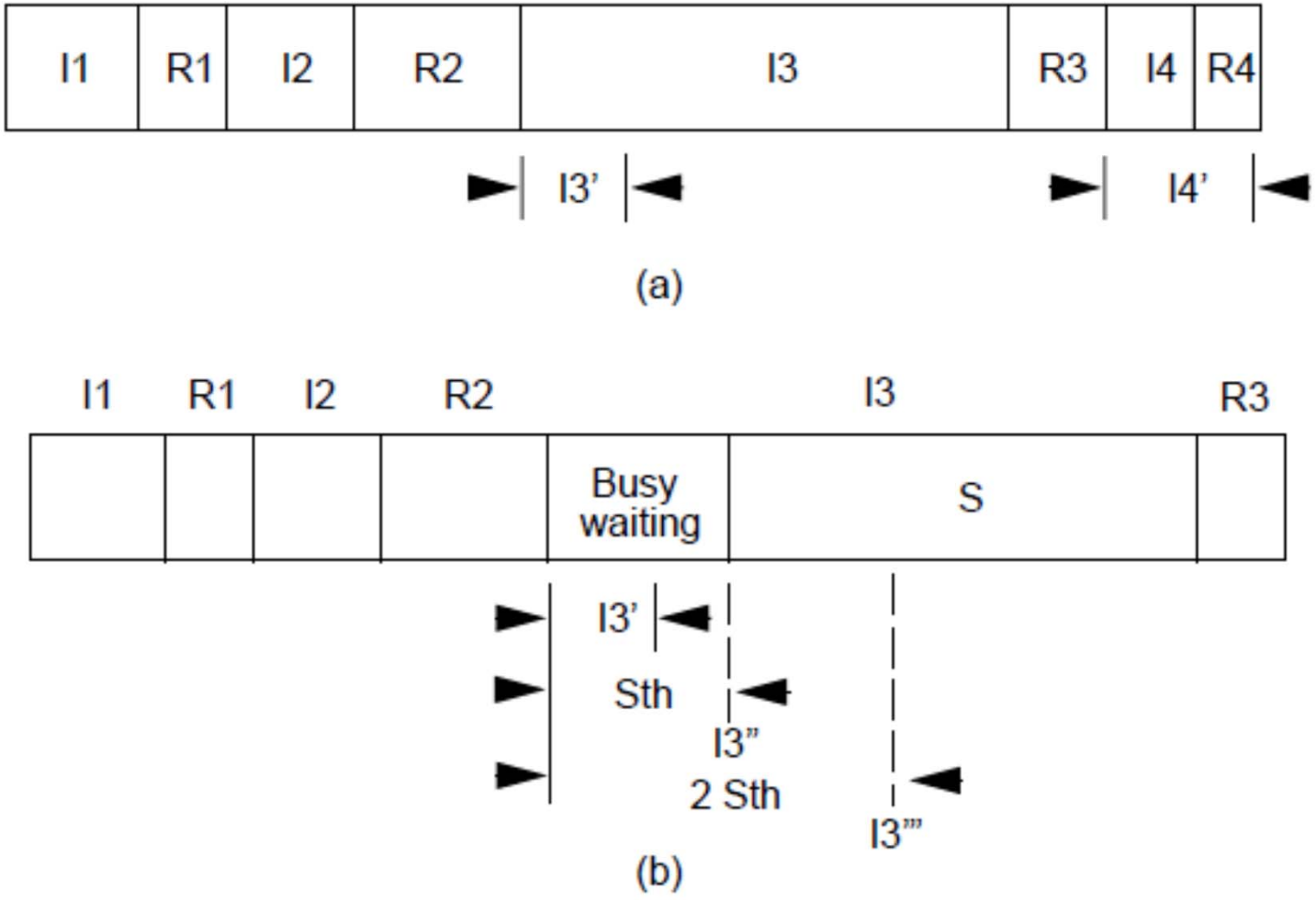
The system then performs a new prediction every *Sth* time to determine whether the system should enter the sleep state.

For example, in Figure, a long idle period *I*3 occurs by the end of a running state *R*2.

If the predicted idle period *I*3' is lower than *Sth*, the system stays in the busy waiting state.
After an *Sth* time period, the system performs the prediction again, resulting in *I*3''. If the predicted value is larger than *Sth*, then it enters the sleep state.

Otherwise, it will perform idle period prediction after another *Sth* time.

A Solution to the First Problem: Keep a watch dog



Coin operated switch to power your TV

A Solution to the 2nd Problem

To resolve the second problem, we add a saturation condition as below:

$$if\ (ai_n + (1-a)I_n > cI_n)$$

$$I_{n+1} = cI_n,$$

where *c* is a constant. Under the saturation condition, the growing rate of *I* is limited to *c* times per update.

Operate the ceiling fan in conjunction with your window air conditioner to spread the cooled air more effectively throughout the room and operate the air conditioner at higher temperature

Need for Prewake up

When the system resumes the running state from the sleep state (i.e., system wake-up), the system needs to perform some recovery procedures.

As a result the system suffers a delay penalty *W* by restoring the system status.

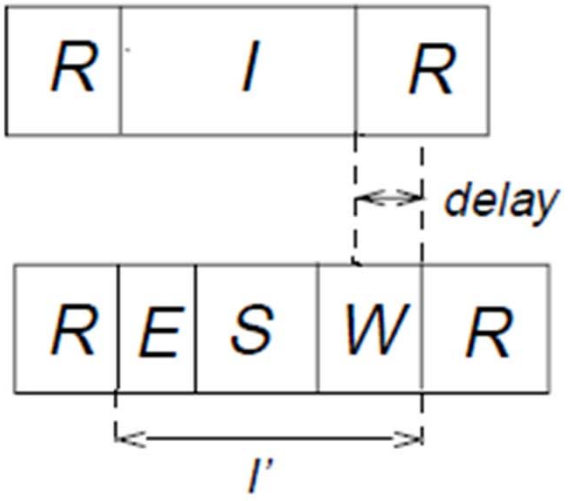
This delay penalty may have a great impact on system responsiveness in some cases, especially when *W* is too long to be neglected.

One way to resolve this problem is to prewake-up the system before the arrival of the next wake-up signal. This can be accomplished by predicting the occurrence of the next wake-up signal.

Operate the ceiling fan in conjunction with your window air conditioner to spread the cooled air more effectively throughout the room and operate the air conditioner at higher temperature

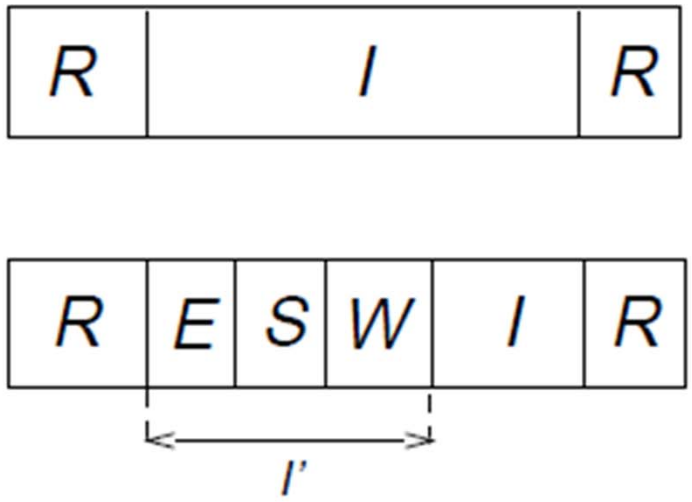
Need for Prewake up

Let *I* be the actual idle period, *I*' the predicted idle period, and *D* =| I'- I | is the error of the prediction. Two situations are shown below



$I' > I \text{ and } D \leq W$

Figure a



$I' < I.$

Figure b

Coin operated switch to power your computer.

Prewake up: First Scenario

I’ > I and D <= W

We overestimate the predicted idle period by *D* and *D* <= *W*.

In this case the system will wake up *W* - *D* time ahead of the next wake-up signal, as shown in Figure (a).

Thus, the delay penalty is *D*, which is shorter than the original delay penalty *W*.

The energy gain is *EG* = *I* x(PR-PS) – (E+W)x(PEW – PS) - DxPS

I’ > I and *D* > *W*,

the system will be woken up by the original wake-up signal. In this case, the prewake-up has no effect on the reduction of the delay penalty.

Prewake up: Second Scenario

Figure (b) shows the second scenario in which $I' < I$.

In this case the delay penalty is zero, but the energy gain is reduced to

$EG = (I - D)x(PR - PS) - (E + W)x(PEW - PS)$

In other words, when the predicted idle period is less than the actual idle period, there will be no responsiveness delay. However, the energy saving will not be as effective as the original one.

Dynamic Power Management : ACPI Standard

The Advanced Configuration and Power Interface (ACPI) specification was developed to establish industry common interfaces enabling robust operating system (OS)-directed motherboard device configuration and power management of both devices and entire systems. ACPI is the key element in Operating System directed configuration and Power Management (OSPM).

The standard was originally developed by Intel, Microsoft, and Toshiba, then later joined by HP, and Phoenix.

First released in December 1996, .

Revision 3.0
September 2, 2004

The latest version is "Revision 5.0," published on November 23, 2011.

Dynamic Power Management : ACPI Standard

Principal Goals

ACPI is the key element in implementing OSPM. ACPI-defined interfaces are intended for wide adoption to encourage hardware and software vendors to build ACPI-compatible (and, thus, OSPM-compatible) implementations.

• Enable all computer systems to implement motherboard configuration and power management functions, using appropriate cost/function tradeoffs.

• Computer systems include (but are not limited to) desktop, mobile, workstation, and server machines.

• Machine implementers have the freedom to implement a wide range of solutions, from the very simple to the very aggressive, while still maintaining full OS support.

• Wide implementation of power management will make it practical and compelling for applications to support and exploit it. It will make new uses of PCs practical and existing uses of PCs more economical.

ACPI supported Global Power states

G3: the mechanical off state in which the system consumes no power.
G2: the soft off state which requires a full OS reboot to restore the machine to working condition. There are four sub states.

- S1, a low wake up latency state with no loss of system context
- S2, a low wake up latency state with a loss of CPU and system cache state
- S3, a low wake up latency state in which all system state except for the main memory is lost
- S4, the lowest power sleeping state, in which all devices are turned off

G1: the sleeping state, in which the system appears to be off and the time required to return to working condition is inversely proportional to power consumption.

G0, the working state in which the system is fully usable

The legacy state in which the system does not comply with ACPI

Advanced Configuration and Power Interface (ACPI)

ACPI provides some basic power management facilities and abstractions for the hardware layer.

The operating system has its own power management module that determines the policy and the OS uses ACPI framework to send the required controls to the hardware and to observe the hardware state as input to the power manager.

It does not provide insight on how and when to power manage them. It does not specify a *power management policy* (*policy* for brevity) or a procedure that takes decisions upon the state of operation of system components and on the state of the system itself.

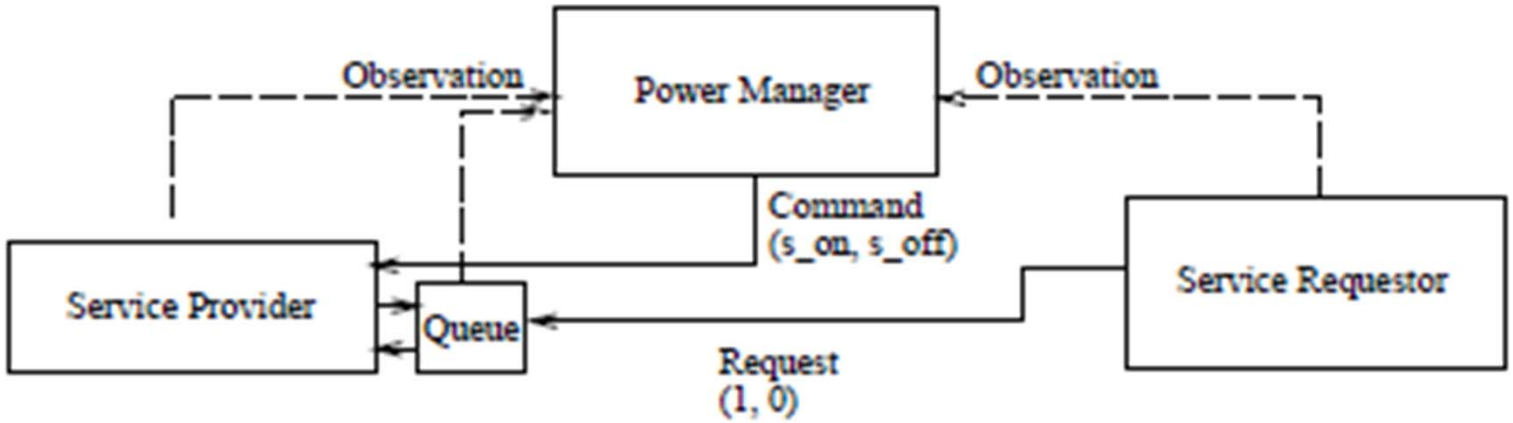
Dynamic Power Management : A Formal Approach

Abstract

In the past, power management policies have been formulated heuristically. The main contribution of this paper is to introduce a finite-state, abstract system model for power-managed systems based on Markov decision processes. Under this model, the problem of finding policies that optimally tradeoff performance for power can be cast as a stochastic optimization problem and solved exactly and efficiently. The applicability and generality of the approach are assessed by formulating Markov model and optimizing power management policies for several systems.

1. Policy Optimization for Dynamic Power Management Luca Benini et. al.,, IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 18, NO. 6, JUNE 1999

Dynamic Power Management



The abstract system model: A Simple Model

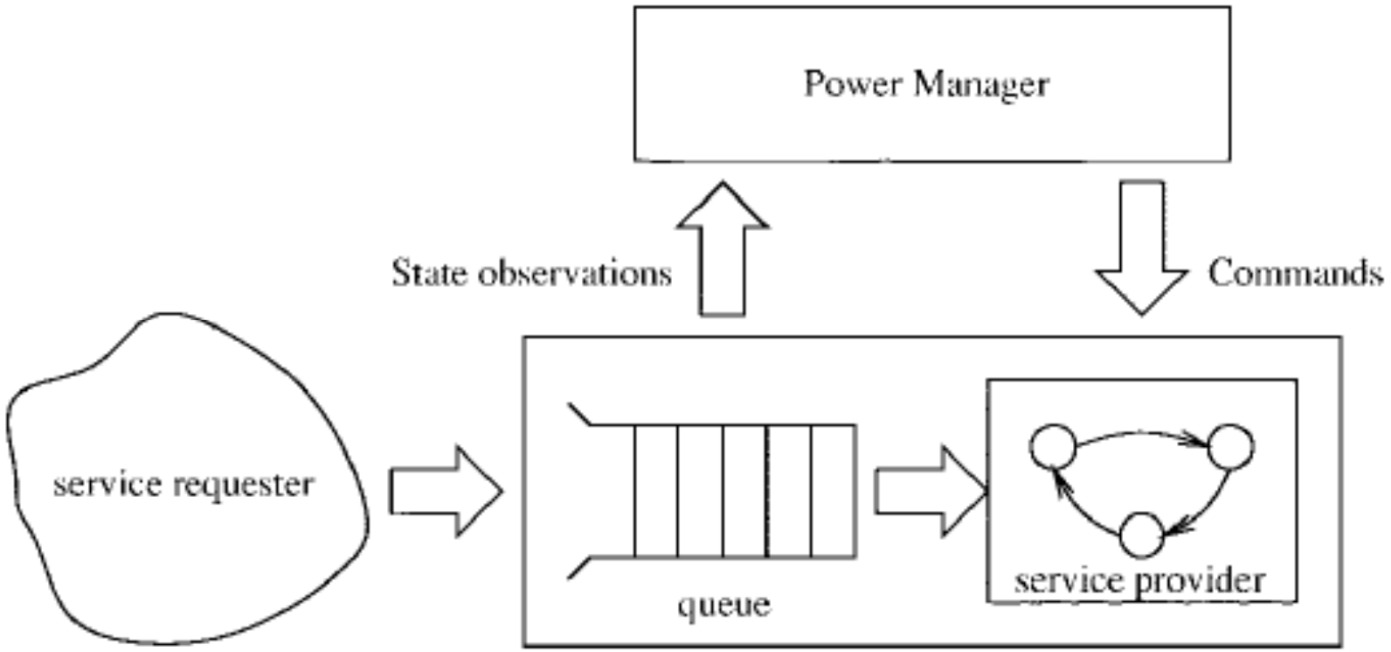
Components are modelled as Markov chains whose state transition graphs are represented for a simple example situation.

The service requestor (SR) has only two states, 0 and 1, representing the number of requests per time period sent to the provider.

The queue of the service provider (SQ) has two states, 0 and 1, representing the number of requests to be serviced. The service provider (SP) has two states, on and off, representing its functional state. When on, it serves up to a request per time period taken from the queue. When off it does not serve any request.

SR evolves independently, while the transition probabilities of SP depend on the command issued by the power manager (PM) and those of SQ depend on the state of both SP and SR.

Components of the system model



The criterion used for choosing what command to issue and when is called *policy*.

Drive sensibly; aggressive driving such as speeding, and rapid acceleration and braking, wastes fuel.

Components of the system model : Power Manager

The power manager is a controller that observes the history of the service provider and of the queue and issues commands.

There is a finite number of commands, and their purpose is to cause the transition of the service provider from one state to another.

The service provider responds to commands in a nondeterministic fashion. In other words, there is no guarantee that the service provider changes state as soon as a command is issued

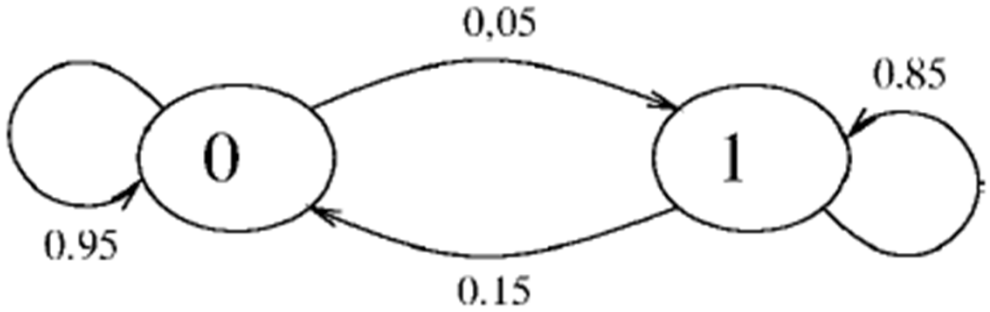
But there is a probability that the transition will be performed in the future.

Policy Optimization Problem : PO Problem

The choice of the policy that minimizes power under performance constraints (or maximizes performance under power constraint) is a constrained optimization problem which is of great relevance for low-power electronic systems.

During spring and summer months, line dry your clothes instead of using a clothes dryer. This can save you as much as $75 a year.

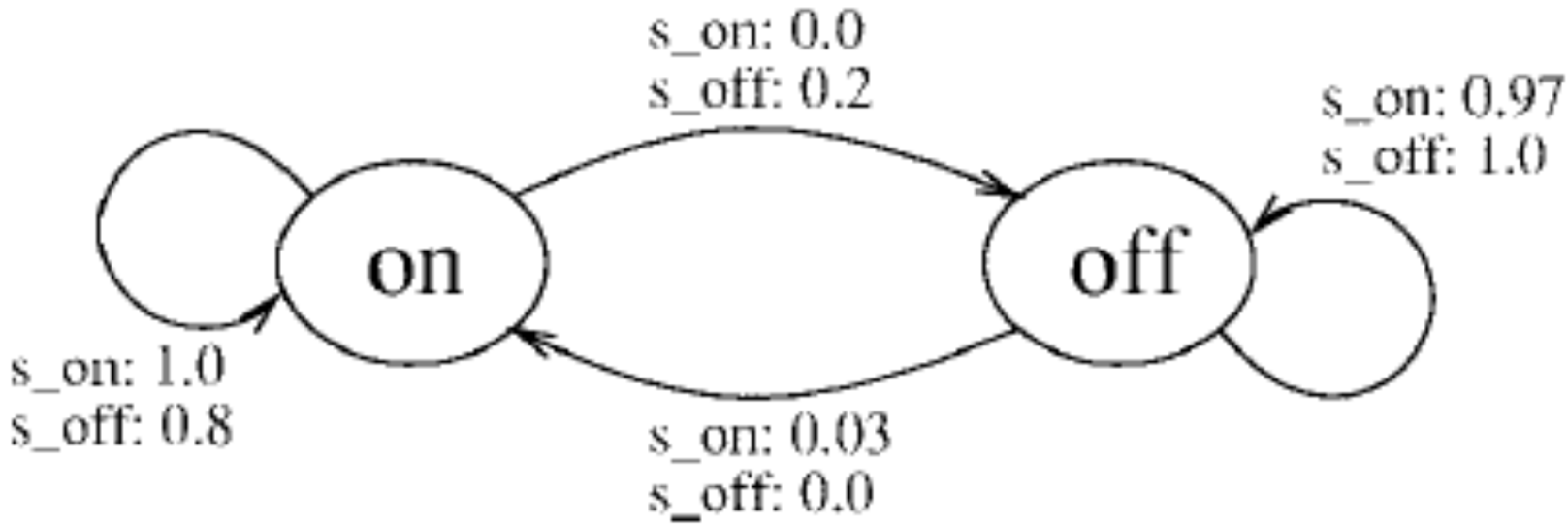
Markov chain model of the service requester.



At any time only two possibilities are given: either a single request or no request is received. An example of a stochastic matrix of SR is

$$\mathbf{P}^{\mathrm{SR}} = \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 0.95 & 0.05 \\ 0.15 & 0.85 \end{pmatrix} \end{array}.$$

Markov chain model of the Service Provider.



$$\mathbf{P}^{\mathrm{SP}}(s\_\mathrm{on}) = \begin{array}{c} \\ \mathrm{on} \\ \mathrm{off} \end{array} \begin{array}{c} \begin{array}{cc} \mathrm{on} & \mathrm{off} \end{array} \\ \begin{pmatrix} 1 & 0 \\ 0.1 & 0.9 \end{pmatrix} \end{array}$$

$$\mathbf{P}^{\mathrm{SP}}(s\_\mathrm{off}) = \begin{array}{c} \\ \mathrm{on} \\ \mathrm{off} \end{array} \begin{array}{c} \begin{array}{cc} \mathrm{on} & \mathrm{off} \end{array} \\ \begin{pmatrix} 0.8 & 0.2 \\ 0 & 1 \end{pmatrix} \end{array}.$$

Consider a SP with two states,on off . Assume that two commands are defined s_on s_off , with the intuitive meaning of "*switch on" and* "*switch off," respectively. When a command is issued, the SP* will move to a new state in the next period with a probability dependent only on the command

References

1. Policy Optimization for Dynamic Power Management Luca Benini, *Member, IEEE*, Alessandro Bogliolo, *Member, IEEE*, Giuseppe A. Paleologo, *Student Member, IEEE,* and Giovanni De Micheli, *Fellow, IEEE,* IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, VOL. 18, NO. 6, JUNE 1999

2. A Predictive System Shutdown Method for Energy Saving of Event-Driven Computation CHI-HONG HWANG and ALLEN C.-H. WU Tsing Hua University, ACM Transactions on Design Automation of Electronic Systems, Vol. 5, No. 2, April 2000, Pages 226–241.

3. M. Srivastava, A. Chandrakasan, and R. Brodersen, “Predictive system shutdown and other architectural techniques for energy efficient programmable computation,” *IEEE Trans. VLSI Syst.,* vol. 4, pp. 42–55, Mar. 1996.