
# Energy Efficient Software Techniques

Intelligent Systems Group
Intel Corporation

# Legal Disclaimer

- INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

  UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

  Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

  The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

  Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

  Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm
- All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.
- Intel product plans in this presentation do not constitute Intel plan of record product roadmaps. Please contact your Intel representative to obtain Intel's current plan of record product roadmaps.
- Intel® vPro™ Technology is sophisticated and requires setup and activation. Availabiity of features and results will depend upon the setup and configuration of your hardware, software and IT environment. To learn more visit: http://www.intel.com/technology/vpro.
- Requires activation and a system with a corporate network connection, an Intel® AMT-enabled chipset, network hardware and software. For notebooks, Intel AMT may be unavailable or limited over a host OS-based VPN, when connecting wirelessly, on battery power, sleeping, hibernating or powered off. Results dependent upon hardware, setup and configuration. For more information, visit http://www.intel.com/technology/platform-technology/intel-amt.
- No computer system can provide absolute security under all conditions. Intel® Trusted Execution Technology (Intel® TXT) requires a computer with Intel® Virtualization Technology, an Intel TXT-enabled processor, chipset, BIOS, Authenticated Code Modules and an Intel TXT-compatible measured launched environment (MLE). Intel TXT also requires the system to contain a TPM v1.s. For more information, visit http://www.intel.com/technology/security
- Intel Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, and virtual machine monitor (VMM). Functionality, performance or other benefits will vary depending on hardware and software configurations. Software applications may not be compatible with all operating systems. Consult your PC manufacturer. For more information, visit http://www.intel.com/go/virtualization
- 
- Intel, the Intel logo and Inter Atom are trademarks of Intel Corporation in the United States and/or other countries.

# Motivation and Introduction

- Power is the most critical factor in MIDs
  - Longer battery life
    - Battery life dictates user experience
  - Increased reliability
    - Prolonged system life when operated at lower temperatures
    - Reduced heat dissipation
- Power is also important for other products
  - Cost of energy

**Power is Key for All Products**

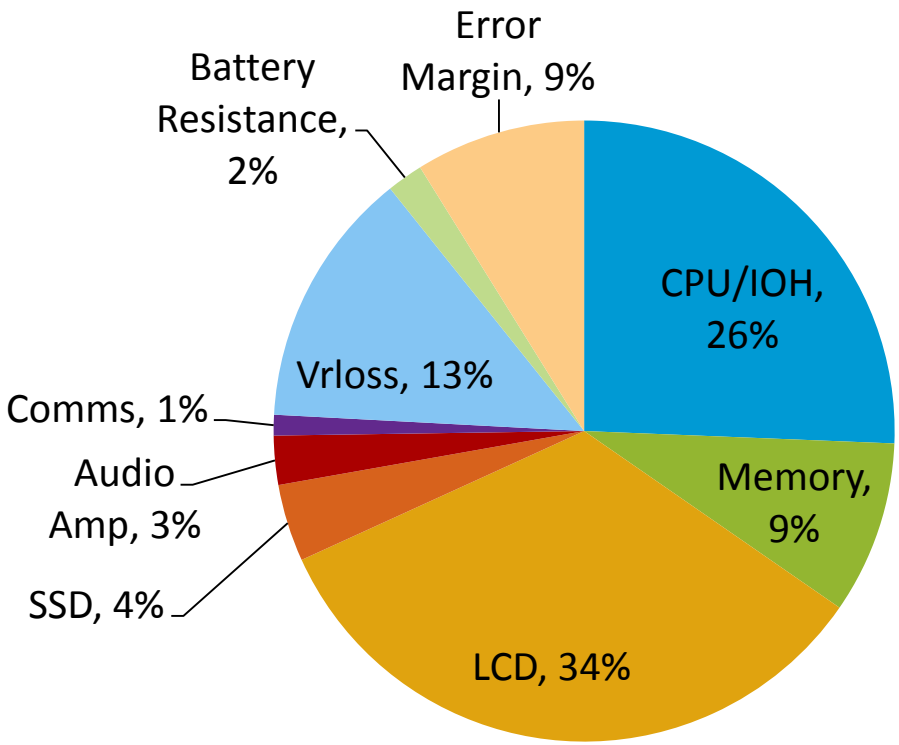
# Software Recommendations for Power Optimization

| Power Guidelines | |
|---|---|
| Hurry up and get idle | Reduce interrupts |
| Improve cache locality | Use timers effectively |
| Manage memory efficiently | Be power-aware and power-smart |
| Avoid polling and busy waits | Use multithreading |

# Basics of Power Management

- Minimize time in active state
- Maximize time in inactive state
- Display is the largest consumer of power.
- Next is CPU: CPU power can be controlled by Software
- If hardware peripherals work autonomously then CPU can sleep

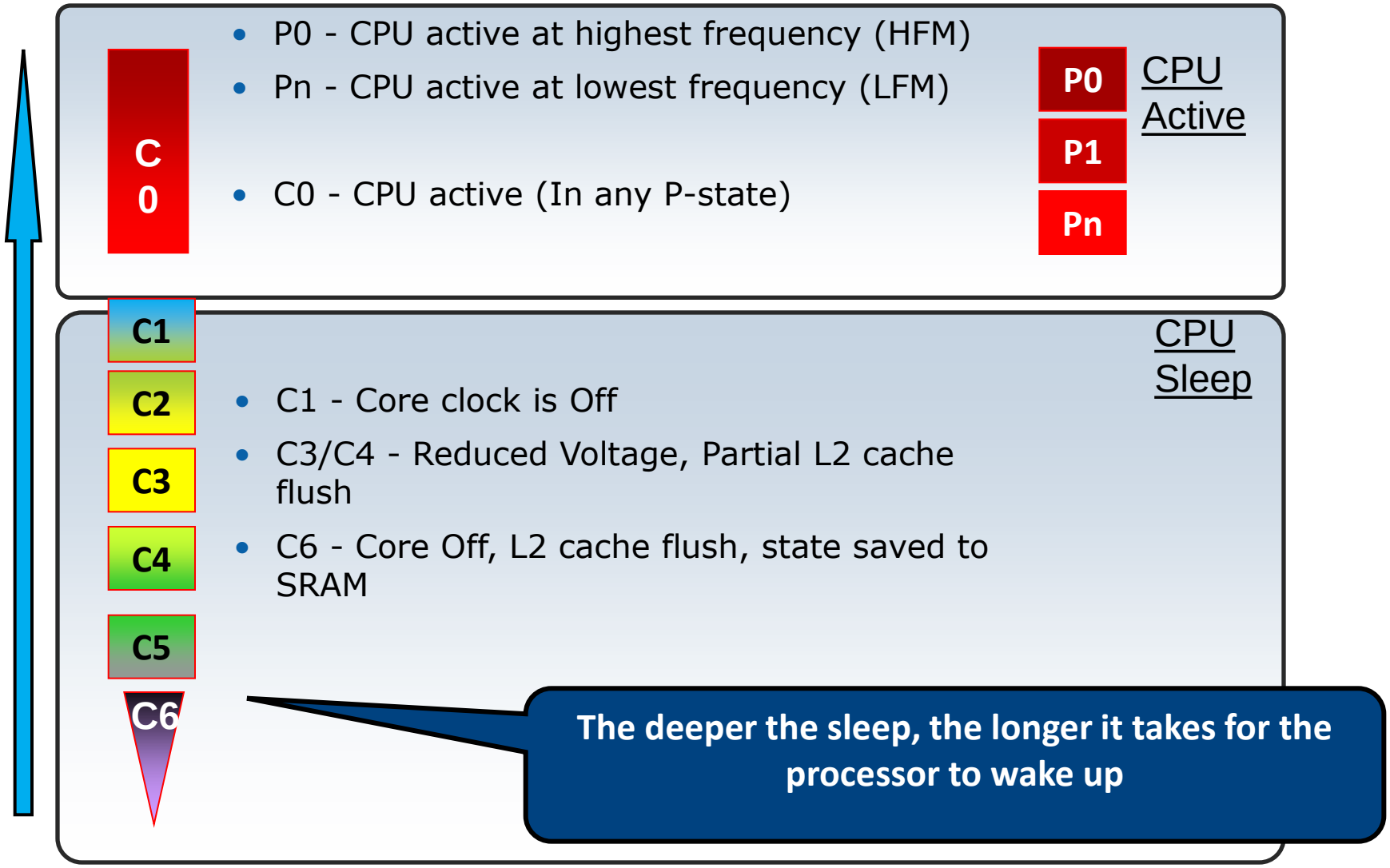
**CIF Video Playback Power Consumption**



# What does C0 residency mean?

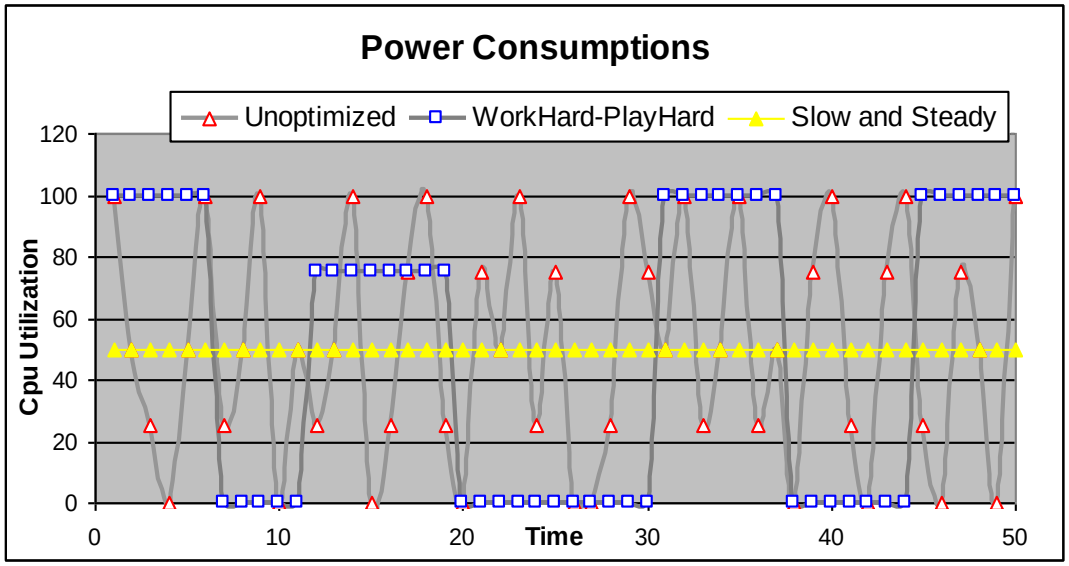
- CPU States
  - Active
  - Idle
- Active states are referred to as p-States
  - The CPU runs at different frequencies
- Idle states are referred to as c-states
  - Different degrees of idleness

# Processor Power States



# CPU P-States

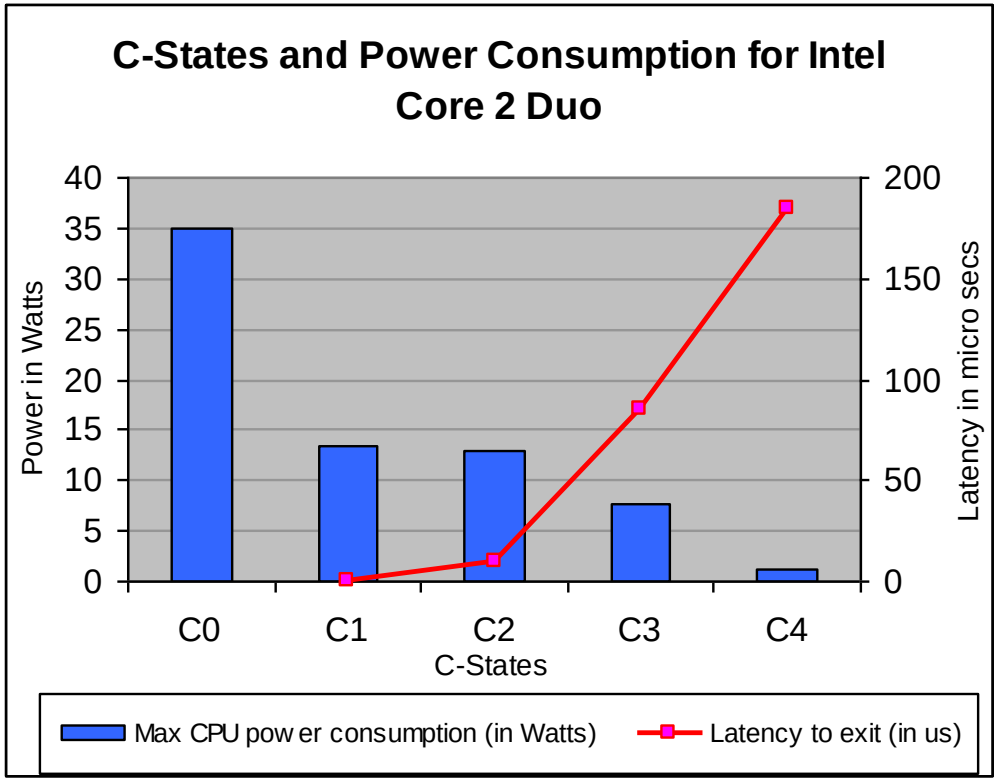
- Offers multiple P-states
  - Active but at different speeds: P-states
  - Example: 34mW for a unit of work at full speed vs. 24 mW/unit of work at half speed




Doing things slow ≠ half the power
Work fast and go to sleep long

# CPU C-States

- Offers multiple C-states
  - Determine how deep the processor can sleep
  - Depth determined by exit latency and power saving tradeoff




**Software needs to make use of the processor C-States for power savings**

# Where does Software fit in?

- Hardware Power Mitigation Techniques exist today
- A single faulty piece of software can blow the power envelope.
- Optimizing software can reduce power in the CPU, memory and other pockets of the platform

# Main categories Nine Mantras for writing Power-efficient Software

- Timers & Interrupts
  - Thou shall not Poll
  - Thou shall not "busy wait"
  - Thou shall reduce Interrupts
  - Thou shall use timers effectively
- Threading Model
  - Thou shall use multithreading efficiently
- Memory Management
  - Thou shall improve cache locality
  - Thou shall manage memory efficiently
- Power Awareness
  - Thou shall be "power-aware"
- Data Efficiency
  - Thou shall program "power-smart"

# Timers & Interrupts: 1 - Thou shall NOT Poll

- Avoid Polling
- Subscribe to events and wait for events to happen
- Bad examples:
  - checking if the mouse moved .. once per second (gnome-screensaver)
  - checking if the sound volume changed .. 10 times per second (mixer applet)
  - checking if it's time to show the next minute in the clock .. once per second (clock applet)
  - checking to see if a smartcard reader got inserted on USB ... 10 times per second (gdm-daemon)
  - waking up 200 times per second (Macromedia Flash plugin)

**Need for an event-driven software architecture**

# Timers & Interrupts: 2 - Thou shall NOT "busy wait"

- Minimize the use of 'tight loops'
- Eliminate spinning loops
- Convert polling loops to be 'event driven'

# Timers & Interrupts: 3 - Thou shall reduce Interrupts

- Avoid waking up the CPU
  - An interrupt will wake the CPU if it was sleeping
- Defer interrupts to the maximum delay tolerable
- Coalesce interrupts
- Synchronize multiple threads to interrupt at the same time

# Timers & Interrupts: 4 - Thou shall use timers effectively

- Timers are needed for house-keeping
- Reduce usage of high-resolution periodic timers
  - Do you really need a periodic timer event every 10 ms?
  - Disable periodic timers when not in use

- Group timers effectively
  - In kernel space : Use timer APIs in application
    - use `round_jiffies()` and `round_jiffies_relative()`

# Threading Model: 5 - Thou shall use multithreading efficiently

- Use multiple threads to speed up execution
  - ➔ Race to Halt
- Balance the threads
  - Synchronize threads to work on different cores simultaneously and idle simultaneously
  - Balance the threads
  - Shut down threads if they are not doing anything
  - Idle threads should be interrupt driven
  - Block a thread on read instead of polling select/poll

# Memory Management: 6 - Thou shall improve cache locality

- Operate on small data at one time so data stays in caches
  - Improves performance
  - Reduces power
    - Less bus activity
    - Less memory traffic
  - If program behavior is “jumpy” , ensure most likely jumps are taken first
  - Reduce working set to fit in cache

# Memory Management: 7 - Thou shall manage memory efficiently

- Get more memory than you need
  - Saves time searching for memory at run time
- Pre-Allocate in larger chunks for creating big buffers
  - Avoids data movement between disk/memory and cache
- IF memory is needed at run time: allocate in smaller chunks: 1 page vs. multiple contiguous pages

# Power - awareness: 8 - Thou shall be “power-aware”

- Don’t’ let application do actions when in background
- Update graphics less when running on battery
- Disable animations when running on battery
- Update window content only when necessary
- When running on battery use “low-power” algorithms
- Reduce read/write to storage in battery mode

# Data Efficiency: 9 - Thou shall program "power-smart"

- Fast execution
  - Handle CPU sleep transitions effectively
    - Eg. Be effective in handling suspend-resume
  - Use high performance algorithms and data structures
    - Use instruction set architectures (SSE) to accelerate computations
    - Use performance libraries
    - Avoid recursion: may be power inefficient
  - Compiler optimized for the most common execution path

# Data Efficiency: 9 - Thou shall program "power-smart"

- Disk Data efficiency
  - Prefetch data from disks if possible: can power-off disks
    - Disks and CDs are moving, analog parts
    - They consume a lot of power when in use
  - Avoid excessive reads/writes to disk
  - Batch requests to disk
  - Cache data close to the processor
  - Opening files that are in cache (atime update)
    - use O_NOATIME flag to the `open()` call if possible
- Network
  - Consider using the network interfaces the least you can
  - Send information in bursts

# Case Study: Video Playback Power Issue

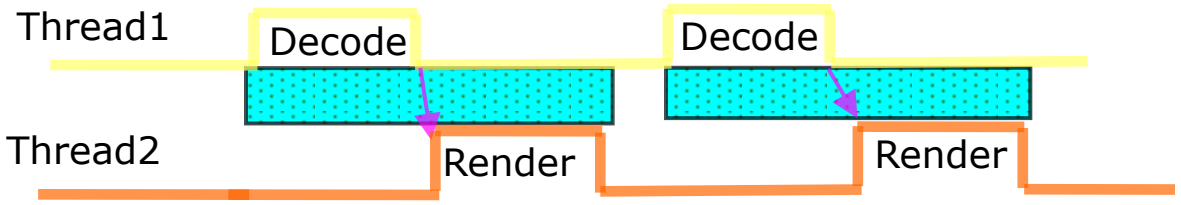
- Goal: Reduce c0% via SW optimization

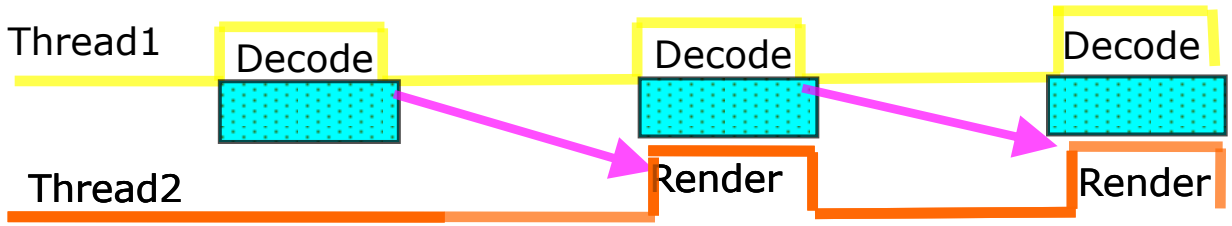
- Observations:
  1. Excessive number of CPU wakeups shown by Powertop
  2. Excessive time spent in memory management as shown by profiling tools

# Case Study:

## Wakeup Reduction…

- Problem
  - Two threads for decode and render not aligned, one active and the other inactive at a time.
- Solution
  - Align the two threads so that on Cpu wakeup both threads were utilized in parallel
  - Benefit: C0% reduced



# Summary

- HW is designed for Low Power:
  - Different states of CPU and platform
  - Different kinds of timers
- SW needs to make use of the features
  - Devices
    - Turn off devices that are not being used.
    - Reduce hard drive activity
    - Reduce LCD brightness
  - Interrupts
    - Reduce CPU wakeups, so the CPU can stay longer in deep power saving c-states
    - Coalesce interrupts by deferring them
    - Reduce polling on devices
  - Memory management
    - Allocate everything early on if possible
    - Pre-Allocate in bug chunks and in small chunks at runtime

**Good software programming practices can reduce power!**

# Intel® Atom™ Processor Specification Docs

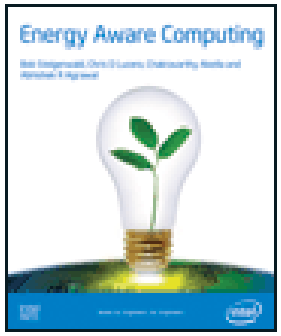
- You can find detailed sleep state info in the Intel® Atom™ processor tech manuals here: http://www.intel.com/products/processor/atom/techdocs.htm
- For instance: Chapter 2 in http://download.intel.com/design/processor/datashts/319535.pdf detailing all the valid Intel Atom processor sleep states.

# References

Creating Energy-Efficient Software:
http://software.intel.com/en-us/articles/creating-energy-efficient-software-part-1/

New Intel Press Book

Energy Aware Computing
http://www.intel.com/intelpress/sum_tmip.htm



# Questions?

# Definitions

- **spinlock** - a lock where the thread simply waits in a loop (spins) repeatedly checking until the lock becomes available.
  - Pro: Spinlocks are efficient if threads are only likely to be blocked for a short period of time, as they avoid overhead from operating system process re-scheduling or context switching.
  - Con: Can become wasteful if held for longer durations, both preventing other threads from running and requiring re-scheduling. Could cause a semi-deadlock.
- **jiffie** – the duration of one tick of the system timer interrupt.
- **CIF** - (*Common Intermediate Format*), also known as **FCIF** (*Full Common Intermediate Format*), is a format used to standardize the horizontal and vertical resolutions in pixels of YCbCr sequences in video signals, commonly used in video teleconferencing systems. It was first proposed in the H.261 standard.