The imagination driving Australia’s ICT future.

Hardware and Power Management

David Snowdon
daves@cse.unsw.edu.au

1. About me
2. How hardware exists
3. Power management
The imagination driving Australia’s ICT future.

ABOUT ME

→ Computer Engineering 1999–2002
→ Honours: Hardware and Software Infrastructure for Sunswift II
→ PhD: Operating system directed power management
→ Solar cars: WSC (99, 01, 03, 05)
The imagination driving Australia’s ICT future.
The imagination driving Australia’s ICT future.

**iBox**

- purpose: intelligent video surveillance;
- 624MHz Xscale (PXA270), 64MB RAM, 4MB Flash;
- analogue video capture, overlays, MPEG 4 encoding, audio;
- USB, CF, Ethernet;
- designed for L4/Iguana (OK inside).
The imagination driving Australia’s ICT future.

HARDWARE

HARDWARE

aka: convincing yourself that it just might work
Overview:

- why talk about hardware design in an OS course?
- hardware designs are difficult to change

Because:

- Operating systems interact closely with the hardware
- Hardware and system software engineers should work together
- OS developers are great embedded systems developers
- OS people think this stuff is cool
The imagination driving Australia’s ICT future.

OVERVIEW

Case Study: the iBox:

- requirements;
- general design;
- schematics;
- layout;
- manufacture;
- board bring-up.
Purpose:

- processing platform for digital video surveillance;
- teaching platform for AOS.
Requirements

Purpose:

- processing platform for digital video surveillance;
- teaching platform for AOS.

Video surveillance:

- motion detection;
- number recognition;
- face recognition.
Functional:

- Performance: five frames per second at $352 \times 288$;
- IO: Ethernet, audio, video, alarm, overlay;
- Video compression: MPEG4 at $768 \times 576$;
- System software update: by an engineer on-site;
- Power supply: up to 24V AC or DC;
- Power consumption: less than 5W.
- Software interfaces: video, audio, network stack, overlay generation, MPEG compression
Non-functional:

- Cost
- Modularity
- Size
- Development time
- Connectors: standard
- Un-trusted software.
Non-functional:

→ Cost
→ Modularity
→ Size
→ Development time
→ Connectors: standard
→ Un-trusted software.

It should cost nothing, use no power, be infinitely small, and take no time to develop. What’s the problem?
The imagination driving Australia’s ICT future.

REQUIREMENTS

Trade-offs:

- Cost vs. Performance
- Size vs. Performance
- Size vs. Utility
- Power vs. Performance
- Time vs. everything
- Cost vs. Quantity
- In-house vs. out-sourcing
Requirements

Summary:

- MPEG 4 requires *serious* processing grunt;
- Motion detection requires *serious* processing grunt;
Strategy:

1. come up with lots of options;
2. pick one that best fits the requirements;
3. lather, rinse, repeat;
4. work out that you’ve made a mistake as early as possible.

We’ll go through a few of the design decisions we made here
FPGA:

- *Field Programmable Gate Array*
- RAM + logic == configurable IC
- “programmed” using VHDL, Verilog, etc.
- can use soft cores
FPGA:

- Field Programmable Gate Array
- RAM + logic == configurable IC
- “programmed” using VHDL, Verilog, etc.
- can use soft cores

For:

- ✓ high performance
- ✓ low power

Against:

- × difficult to implement algorithms
- × some expensive soft-cores
- × no L4
The imagination driving Australia’s ICT future.

**DESIGN**

DSP:

- Digital signal processor
- VLIW processors optimised for signal processing
- e.g. TI TMS320C6416: 8 GMACs/s @ 1GHz
The imagination driving Australia’s ICT future.

**DESIGN**

**DSP:**
- Digital signal processor
- VLIW processors optimised for signal processing
- e.g. TI TMS320C6416: 8 GMACs/s @ 1GHz

**For:**
- ✓ very high signal processing performance
- ✓ various implementations with video interfaces

**Against:**
- ✗ no MMU
- ✗ optimisation required to achieve performance
- ✗ expensive compilers
Digital signal processor + Generic processor:

- use a general-purpose processor for control
- use a DSP for signal processing algorithms
- often packaged together (e.g. TI OMAP, DaVinci)
Digital signal processor + Generic processor:

→ use a general-purpose processor for control
→ use a DSP for signal processing algorithms
→ often packaged together (e.g. TI OMAP, DaVinci)

For:

☑ solves lots of problems with a DSP alone
☑ easier to handle real-time issues

Against:

☒ higher software complexity than a single-core solution
☒ need more than one DSP for our application
General-purpose processor:

→ optimised for a wide range of tasks
General-purpose processor:

- optimised for a wide range of tasks

For:

- good support from compilers and operating systems
- MMU
- system-on-a-chip designs make implementation easier/smaller

Against:

- very high required performance == high power, large size
- not optimised for signal processing
- not capable of MPEG 4 encode
The imagination driving Australia’s ICT future.

DESIGN

SMP:

→ multiple generic cores
SMP:

→ multiple generic cores

For:

☑ high performance
☑ retains generic capabilities while providing processing power
☑ OS transparently handles multiprocessing

Against:

☒ large size
☒ high power
☒ higher parts count
The imagination driving Australia’s ICT future.

DESIGN

What next?:

- rough out the designs: identify components
- don’t worry about the easy bits (PSU, connectors)
- obtain development kits and test performance
- look at how well the software fits the design

Calculate:

- cost
- parts count
- part availability
- power consumption
What we considered:

1. PXA270
2. TI DaVinci
3. Sierra MIPS
What we considered:

1. PXA270
2. TI DaVinci
3. Sierra MIPS

We chose the PXA270, combined with an external MPEG 4 codec ASIC.
The imagination driving Australia’s ICT future.

DESIGN

PXA270 based solution:

- 624MHz XScale (ARM) core — fast enough for video algorithms
- quick-capture interface for video input
- tricky circuit for overlay generation
- other useful built-in peripherals (DMA, SRAM, USB, CF, LCD)
- L4 was already ported
- good Linux support
- MMX and DSP instructions
- very low power
The imagination driving Australia’s ICT future.

**DESIGN**

**PX A270 based solution:**

- 624MHz XScale (ARM) core — fast enough for video algorithms
- quick-capture interface for video input
- tricky circuit for overlay generation
- other useful built-in peripherals (DMA, SRAM, USB, CF, LCD)
  - ✓ L4 was already ported
  - ✓ good Linux support
  - ✓ MMX and DSP instructions
  - ✓ very low power
  - ✗ requires some expensive PCB technologies
The imagination driving Australia’s ICT future.

DESIGN

September 15, 2006
The imagination driving Australia’s ICT future.

DESIGN

[Diagram showing hardware components: Video out, Video In, Audio I/O, MPEG Codec, CPU, Video SDRAM, MISC I/O, LCD, Ethernet, Compact Flash, SDRAM, Flash]
The imagination driving Australia’s ICT future.

**Schematics**

Schematics:
- schematics define how components connect logically
- component pins are connected to form *nets*
- hierarchical schematics are possible via *ports*

![Schematic Diagram](image)
Defining components:

1. schematic component
2. footprint
The imagination driving Australia’s ICT future.

**SCHEMATICS**

Things to think about:

- get it right first time
- decoupling – reduce power supply noise
- regulator noise
- pin assignment – shortest paths
- DMA

Hardware should be designed to make software elegant
Title: Top level schematic for iBox

National ICT Australia
Level 4
227 ANZAC Pde
Kensington, Sydney, 2052
AUSTRALIA

File: Z:\project\ibox\design\toplevel.SCHDOC

Page 1 of 1

Date: 12/09/2006
Time: 11:28:57 PM

Title Size: Number: Date: File:
Revision: Sheet of
Time:
A4

PWR_EN
SDATA_IN
SDATA_OUT
SCLK
MCLK
VCC
VDD
GND

nTRST
nRESET
nCS0
nCS1
nCS2
nCS3
nCS4
nPDN
nCD1
nCD2
nPWE
nPIOIR
nPCE2
nPCE1
nPREG
nPPOE
nPIOIS16
nPWE
nPIOIR
nPCE2
nPCE1
nPREG
nPPOE
nPIOIS16

LED0
LED1
LED2
LED3
USB_P
USB_N
USB_OC
nRELAY
nPSKTSEL
VCC_IO
GND

VIN
SCLK
SDAT
3V3_ENABLE
2V5_ENABLE
VCORE_ENABLE
LDO_EN
nATTENABLE
nVDDFAULT
nRESET

CA_PCK1
CA_FIELD
VID
CA_DIN[0..7]
VGA_VIDEO_OUT
MPEG_VIDEO_OUT
MPEG_VIDEO_OUT
VPCLK
VID
CA_PCK1
CA_FIELD
CA_DIN[0..7]

R46 3k3
C57 100n
GND
R47 10k
GND
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_OUT
nRESET_Out
B17 is connected to SFRM to ease layout.

Design guide recommends 20-ohm series termination is required on the SDCAS and SDCLK lines. Not required on other lines. Requires a "Balanced T".
Remember that RDY needs a pullup elsewhere in the system.
Selects master mode
Video decoder and overlay generation

- Low pass filter here?
- Many circuits have a 220μF cap here.
- Power supply filtering
- Video decode
- Overlay generation

Overlay generation

- Analogue switch between blank and input
- Buffering and amplification
- Overlay generation

Video decode

- Analogue switch between blank and input
- Buffering and amplification
- Overlay generation

Power supply filtering

- Many circuits have a 220μF cap here.
- Power supply filtering
- Video decode
- Overlay generation

Title: Video decoder and overlay generation
National ICT Australia
Level 4
227 ANZAC Pde
Kensington, Sydney, 2052
AUSTRALIA

Page: 1 of 17

File: Z:\projects\ boxer\ designs\ tvp5150\ tvp5150.pdf

Date: 12/02/2006
Time: 12:32:41 PM
Sheet 11 of 17

Revision: *
Power input and conditioning

- Bridge Rectifier: POWER IN #1
- Protection: D3 Bridge1
- Bulk Capacitance: C48, C49, C50
- Back Regulator: U10

- 3V ripple worst case, populate or not depending on requirements

Components:
- D4: 33V Transient Suppressor, SMAJ33A
- U10: LT1956-5
- C53: 2.2uF, 50V
- R45: 4k7
- C54: 22uF Ceramic
- C55: 220pF
- C56: 4700pF
- L4: 15uH

Title: Power input and conditioning

National ICT Australia
Level 4
22/7 ANZAC Pde
Kensington
NSW, 2032

File: DesignPower_in.SCHDOC
Layout technology and terminology:

- design a **printed circuit board** (PCB)
- layers of fibreglass and copper
- the iBox uses an 8 layer board (8 copper layers)
- **vias** are holes used to connect between layers
Component footprints:

- through-hole
- surface-mount
The imagination driving Australia’s ICT future.

LAYOUT

September 15, 2006

Hardware
The imagination driving Australia’s ICT future.
The imagination driving Australia’s ICT future.
Trade-offs:

- number of layers vs. size
- component size vs. manufacturability
- component size vs. signal integrity
The imagination driving Australia’s ICT future.

**Layout**

**Routing:**
- The physical pins are connected with traces of copper.
- The design software only allows connections defined by the schematic.
- Design rules are defined to ensure manufacturability.
- The *artwork* is used by a manufacturer to build a PCB.
The imagination driving Australia’s ICT future.

LAYOUT

Things to think about:

- Auto-routing vs. manual routing
- boxes and enclosures
- connector locations
- pin assignment
- size of the traces
- isolating noise
- signal integrity
- simulation
Signal integrity: Digital systems are still analogue!

→ Above 100MHz, edges on digital signals tend to have high-frequency components
→ transmission line effects
→ inductance and capacitance in the traces/packages
→ noise, cross-talk, EMI problems

So:

→ keep stub lengths short
→ use termination
→ think about return paths
→ use simulation
The imagination driving Australia’s ICT future.
The imagination driving Australia’s ICT future.

**Layout**

iBox layout:

- 624MHz PXA270 is a uBGA — 0.5mm between balls
- We need very small vias — laser drilling
- We need buried and blind vias
The imagination driving Australia’s ICT future.
The imagination driving Australia’s ICT future.

Manufacture

Surface-mount:
- Solder stencil
- Pick and place
- Reflow

Through-hole:
- Wave soldering
JTAG: Joint Test Action Group – a standard for board testing
- boundary scan
- debugging
- custom commands

JTAG can be daisy-chained – multiple chips on one port
PCB Testing:

- PCBs are tested using a “bed of nails”
- this doesn’t work with BGA and PGA packages
- JTAG defines boundary scan: the device pins can be controlled
- can be used in combination with “bed of nails”
The imagination driving Australia’s ICT future.
iBox JTAG:

- Boundary scan lets us control all of the pins on the PXA270
- so we can manually control the bus
- so we can program the flash memory
Debug:

- some processors have extra, debugging JTAG instructions
- Xscale cores can execute instructions from a mini-instruction cache
- breakpoints
- single step
- watch variables

Trace:

- a trace port exports which instructions are run
- analysis tools can step backward
Bootloader:

To load software, you need software?

- prepares the system for boot
- handles device-specific tasks
- loads and jumps to the operating system
- commonly saved in read-only memory as firmware
- can be very simple or complex
- some systems use multi-stage loaders

The iBox uses U-Boot, the NSLU2 uses Redboot
Start up on the PXA270:

1. start executing at 0x00000000 (Flash, on iBox)
2. configure the GPIO pins
3. initialise the static memory and SDRAM
4. initialise the IRQs
5. set the core clock frequency
6. enable the caches
7. copy into RAM
8. set up a stack
9. read configuration
10. set up devices and other services
Porting an OS to the new platform:

- configure for PXA270
- configure specific devices (network, CF, USB, etc)
BRING-UP

Bringing up the iBox:

→ Power supplies
→ Soldering the PXA270
→ Testing using JTAG
→ Flash
→ SDRAM
→ Bootloader
→ Ethernet
→ The rest
The imagination driving Australia’s ICT future.
The imagination driving Australia’s ICT future.

POWER MANAGEMENT

aka: doing more with less
Overview:

① Why?
② Concepts
③ OS directed power management
④ Evaluation techniques
⑤ Energy accounting
Why do we care about power?

- Thermal management
- Electricity consumption
- Energy limited devices

Power-conscious systems can use energy more effectively.
Why do we care about power?

- Thermal management
  1. Enclosure size and noise
  2. Performance
- Electricity consumption

- Energy limited devices

Power-conscious systems can use energy more effectively.
Why do we care about power?

- Thermal management
  ① Enclosure size and noise
  ② Performance

- Electricity consumption
  ① Environment
  ② Cost

- Energy limited devices

Power-conscious systems can use energy more effectively.
Power Management

Why do we care about power?

➢ Thermal management
  ① Enclosure size and noise
  ② Performance
➢ Electricity consumption
  ① Environment
  ② Cost
➢ Energy limited devices
  ① Battery powered
  ② Solar powered

Power-conscious systems can use energy more effectively.
Where does the energy go?:

- switching circuits
- peripherals
- voltage conversion
- batteries
Causes of Energy Use

Switching circuits:

- short circuit power
- gate capacitance
- leakage
The imagination driving Australia’s ICT future.

CAUSES OF ENERGY USE

NOT

A \rightarrow \overline{A}

MOSFET-P

MOSFET-N

GND

NAND

A

\overline{B}

MOSFET-P

MOSFET-N

A \cdot B

VCC

GND
The imagination driving Australia’s ICT future.

**Causes of energy use**

Gate capacitance:

- to turn on a transistor, we charge the gate beyond its threshold voltage
- to turn it off, we discharge it
- the charge disappears and is the main source of power consumed
- the charge transferred is: \( q = Cv \)
- the energy transferred is: \( E = \frac{1}{2} Cv^2 \)
The imagination driving Australia’s ICT future.

CAUSES OF ENERGY USE

Hardware techniques:

→ clock gating
CAUSES OF ENERGY USE

Software energy consumption:
- more instructions = more energy
- different instructions use different energy
- different data use different energy

Dynamic frequency scaling:

- reduce the system frequency to reduce the power
- each switch we lose $E = \frac{1}{2} C v^2$
- $C$ comes from the gate and interconnect capacitance
- frequency is the rate at which we switch
- conclusion: power scales linearly with frequency
Dynamic frequency scaling:

- reduce the system frequency to reduce the power
- each switch we lose $E = \frac{1}{2}Cv^2$
- $C$ comes from the gate and interconnect capacitance
- frequency is the rate at which we switch
- conclusion: power scales linearly with frequency

The energy required to complete a given amount of work is unchanged.
Dynamic voltage scaling:

→ if we reduce the frequency, the transistors don't have to switch as fast
→ transistor speed is dependent on the driven voltage
→ if we reduce the frequency, we can reduce the voltage
→ given $E = \frac{1}{2} Cv^2$, we get quadratic energy savings
Dynamic Voltage Scaling

Voltage/frequency switching overheads:

- frequency generation is usually via a crystal and PLL
- clock multipliers control the PLL
- clock dividers generate lower frequencies
- the PLL has to settle following a change
- on the PXA255 (XScale) this is \( \approx 500 \mu s \)
- or, always run the PLL at full speed and change dividers
- on the PXA255 this takes 20 cycles.
DYNAMIC VOLTAGE SCALING


➔ assumption: idleness is bad
➔ eliminate idleness using DVS
   • if idle time is large, decrease speed
   • if idle time is small, increase speed
   • adjust the speed periodically

Other algorithms:
➔ lots of ways of predicting idleness
Dynamic Voltage Scaling


Peukert:

\[ Q = \frac{k}{I^\alpha} \]

- battery capacity is related to rate of discharge
- steady discharge may increase the battery life
- can DVS be used to attain this?
QoS issues:

- reducing frequency reduces performance
- missed deadlines
- reduced QoS
- real-time techniques may be employed
QoS issues:

- reducing frequency reduces performance
- missed deadlines
- reduced QoS
- real-time techniques may be employed

*every man and his dog has an RT DVS technique*
Trouble in wonderland:

- ignoring leakage, transistors only use power when they switch
- not every part of the circuit switches on every clock cycle
- different sections run at different frequencies
- off-chip devices aren’t affected
Dynamic Voltage Scaling

Trouble in wonderland:

- ignoring leakage, transistors only use power when they switch
- not every part of the circuit switches on every clock cycle
- different sections run at different frequencies
- off-chip devices aren’t affected

Also:

- We can’t ignore leakage!

The simple models only work in special cases

→ most academic research uses simulation for evaluation
→ the naive models lead to incorrect results.
DVS: Myths and Facts


- most academic research uses simulation for evaluation
- the naive models lead to incorrect results.

So:

- we take real measurements of a real system
- we run an appropriate benchmark suite (MiBench)

→ most academic research uses simulation for evaluation
→ the naive models lead to incorrect results.

So:

→ we take real measurements of a real system
→ we run an appropriate benchmark suite (MiBench)

Results:

→ real measurements differ dramatically from models
→ optimum processor setting is dependent on workload
The naive model:

\[ P \propto f v^2 \]

Limitations:

- ignores “static” power (core-frequency independent)
- ignores external components
  - affects computation time
  - energy consumption scales differently to the CPU
Experimental setup:

- single board computer – PLEB 2
- instrumented power supplies

Valid configurations:

<table>
<thead>
<tr>
<th>$V_{core}$ (V)</th>
<th>$f_{cpu}$ (MHz)</th>
<th>$f_{internalbus}$ (MHz)</th>
<th>$f_{mem}$ (MHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>1.0</td>
<td>200</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1.1</td>
<td>300</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1.3</td>
<td>400</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>1.3</td>
<td>400</td>
<td>200</td>
<td>100</td>
</tr>
</tbody>
</table>
Execution time for CPU bound:

![Execution time vs. Frequency for cpbound](chart.png)

"cpbound.dat" index 0:7 using 1:2
"cpbound.dat" index 8 using 1:2
Execution time for Memory bound:

"membound.dat" index 0:7 using 1:2
"membound.dat" index 8 using 1:2
Number of cycles for CPU bound:

![Graph showing normalised cycles vs. Turbo Frequency for cpbound with data from "cpbound.dat" index 0:7 using 1:2 and index 8 using 1:2](image)
Number of cycles for Memory bound:

Normalised cycles vs. Turbo Frequency for membound

"membound.dat" index 0:7 using 1:2
"membound.dat" index 8 using 1:2
The imagination driving Australia’s ICT future.

DVS: MYTHS AND FACTS

CPU energy:

The chart shows the model predicted energy for Cpu-intensive and Memory-intensive configurations across different settings of V_core, f_cpu, and f_int bus.

- **V_core** values: 1.0, 1.0, 1.1, 1.3, 1.3
- **f_cpu** values: 100, 200, 300, 400, 400
- **f_int bus** values: 50, 100, 100, 100, 200

The graph indicates a significant increase in CPU energy with higher values of V_core and f_int bus, while f_cpu has a less pronounced effect.
The imagination driving Australia’s ICT future.

DVS: MYTHS AND FACTS

Memory energy:

![Memory Energy Graph](image)

- **V_core**: 1.0, 1.0, 1.1, 1.3, 1.3
- **f_cpu**: 100, 200, 300, 400, 400
- **f_int_bus**: 50, 100, 100, 100, 200

Cpu-intensive
Memory-intensive

SEPTEMBER 15, 2006

POWER MANAGEMENT
The imagination driving Australia's ICT future.

DVS: Myths and Facts

Total energy:

- Model predicted energy
- Cpu-intensive
- Memory-intensive

![Normalized Total Energy Graph]

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Cpu-intensive</th>
<th>Memory-intensive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vcore</td>
<td>1.0</td>
<td>1.3</td>
</tr>
<tr>
<td>f_cpu</td>
<td>100</td>
<td>400</td>
</tr>
<tr>
<td>f_int_bus</td>
<td>50</td>
<td>100</td>
</tr>
</tbody>
</table>

September 15, 2006

Power Management

75
The imagination driving Australia’s ICT future.

DVS: MYTHS AND FACTS

Padded with low-power mode power:

![Graph showing energy consumption vs configurations]

- Configurations:
  - V_core: 1.0, 1.0, 1.1, 1.3, 1.3
  - f_cpu: 100, 200, 300, 400, 400
  - f_int_bus: 50, 100, 100, 100, 200

Total Energy

Cpu-intensive
Memory-intensive

September 15, 2006
Power Management
Conclusions:

- the memory/cpu frequency ratio affects the number of cycles
- the most effective DVS policy depends on usage
- the lowest-energy setting depends on the workload
- increasing frequency doesn’t improve membound performance
- idling in a low-power mode may be better than running at a low frequency

1. maintain per-thread CPU performance counters
2. use cache miss and instruction counters
3. use a lookup table to select a frequency
4. construct the table for a 10% performance hit

Demonstrated energy savings of up to 37% with a 10% slowdown for gzip

1. maintain per-thread CPU performance counters
2. use cache miss and instruction counters
3. use a lookup table to select a frequency
4. construct the table for a 10% performance hit

Demonstrated energy savings of up to 37% with a 10% slowdown for gzip

How would this affect a μK based system?
Sleep modes:

- processors and devices have low-power modes
- these modes switch off parts of the processor
- some state or functionality will be lost
- lower-power modes tend to lose more functionality
- lower-power modes take longer to enter and exit
Sleep modes:

- processors and devices have low-power modes
- these modes switch off parts of the processor
- some state or functionality will be lost
- lower-power modes tend to lose more functionality
- lower-power modes take longer to enter and exit

How do we choose when to sleep?

- processor(s)
- devices
## SLEEP MODES

### Case study: PXA255

<table>
<thead>
<tr>
<th>Mode</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>200MHz Run/Turbo</td>
<td>178mW</td>
</tr>
<tr>
<td>200MHz Idle</td>
<td>63mW</td>
</tr>
<tr>
<td>33MHz Idle</td>
<td>45mW</td>
</tr>
<tr>
<td>Sleep</td>
<td>0.175mW</td>
</tr>
</tbody>
</table>
### Case study: Hard disk

<table>
<thead>
<tr>
<th>Mode</th>
<th>Sleep delay</th>
<th>Wake delay</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idle</td>
<td>0</td>
<td>0</td>
<td>657mW</td>
</tr>
<tr>
<td>Standby</td>
<td>0.85s</td>
<td>1.03s</td>
<td>235mW</td>
</tr>
</tbody>
</table>
SLEEP MODES

Transition management:

- some energy is saved while “asleep”
- some energy is wasted while transitioning
- introduced delay reduces quality-of-service

We need to predict the future
SLEEP MODES


- an “idleness detector” should predict the start time and duration of an idle period

Idle start:

- timeout based: (static, variable, adaptive)
- rate-based: (static, adaptive thresholds)
- periodic: (maintain a DPLL)
→ an “idleness detector” should predict the start time and duration of an idle period

Idle start:
→ timeout based: (static, variable, adaptive)
→ rate-based: (static, adaptive thresholds)
→ periodic: (maintain a DPLL)

Idle duration:
→ static (infinite, none, fixed)
→ moving average (filtered, unfiltered)
→ backoff (geometric increase, arithmetic decrease)
→ periodic
The imagination driving Australia’s ICT future.

SLEEP MODES

OS support:

- Cooperative IO (Weissel, et al)
- Ghosts in the machine (Anand, et al)
- Power-aware page allocation (Lebeck, et al)
- Timer ticks (IBM Watch project)

Other possible power down areas:

- LCD backlight (user context sensing, selective lighting)
- wireless network
- sound card
Conclusions:

- keep it simple - DTT with break-even time does OK.
- adaptive techniques can save power
- operating system co-operation improves
- Dougliis et al present potential reductions of 60% beyond a 5s timeout
Adaptation:

- applications can sometimes adjust their computation best
  - MPEG/JPEG
  - audio bitrate
  - game FPS
  - distributed process migration
  - map viewer detail (Flinn)
- applications can scale their QoS to meet a goal
  - a given battery lifetime
  - a given CPU temperature
- requires operating system feedback
Evaluation techniques:

- Simulation
- State-based accounting
- Indirect measurement
- Real measurement
The imagination driving Australia’s ICT future.

EVALUATION TECHNIQUES

- **Simulation**
  - necessary information may not be available
  - non-trivial to build
  - usually limited to CPU/memory.
  - computing power vs. detail
  - difficult to interface

- **State-based accounting**
  - assumes a constant power in each state

- **Indirect measurements**
  - only really applies to the CPU and memory (at the moment)
  - Insufficient performance counters for energy-related events

- **Real power measurements**
  - may measure background activity incorrectly
  - overhead vs. resolution
  - cumbersome difficult to obtain setup
EVALUATION TECHNIQUES

Powerscope (Energy profiling):

- Flinn, 99
- designed to be similar to performance profile
- A host computer monitors a computer being profiled
- a computer-controlled multimeter samples the actual energy used
- the profiled computer records PIDs and PC for each sample
- post-processing attributes energy to procedures/processes
The imagination driving Australia’s ICT future.

## EVALUATION TECHNIQUES

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Elapsed Time (s)</th>
<th>CPU Power (W)</th>
<th>Memory Power (W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>synth_1to1</td>
<td>21.520</td>
<td>0.676</td>
<td>0.458</td>
</tr>
<tr>
<td>dct36</td>
<td>3.741</td>
<td>0.657</td>
<td>0.612</td>
</tr>
<tr>
<td>III_antialias</td>
<td>1.226</td>
<td>0.652</td>
<td>0.488</td>
</tr>
<tr>
<td>dct64</td>
<td>4.594</td>
<td>0.652</td>
<td>0.481</td>
</tr>
<tr>
<td>do_layer3</td>
<td>0.226</td>
<td>0.639</td>
<td>0.567</td>
</tr>
<tr>
<td>main</td>
<td>0.016</td>
<td>0.635</td>
<td>0.893</td>
</tr>
<tr>
<td>dct12</td>
<td>0.069</td>
<td>0.635</td>
<td>0.701</td>
</tr>
<tr>
<td>stream_head_shift</td>
<td>0.001</td>
<td>0.625</td>
<td>0.000</td>
</tr>
<tr>
<td>III_dequantize_sample</td>
<td>3.642</td>
<td>0.622</td>
<td>0.547</td>
</tr>
<tr>
<td>set_pointer</td>
<td>0.017</td>
<td>0.613</td>
<td>0.649</td>
</tr>
<tr>
<td>III_hybrid</td>
<td>0.523</td>
<td>0.608</td>
<td>0.724</td>
</tr>
<tr>
<td>init_layer3</td>
<td>0.032</td>
<td>0.604</td>
<td>0.419</td>
</tr>
<tr>
<td>audio_flush</td>
<td>0.007</td>
<td>0.598</td>
<td>0.427</td>
</tr>
<tr>
<td>III_get_scale_factors_1</td>
<td>0.164</td>
<td>0.575</td>
<td>0.703</td>
</tr>
<tr>
<td>read_frame</td>
<td>0.065</td>
<td>0.571</td>
<td>0.821</td>
</tr>
<tr>
<td>III_get_side_info</td>
<td>0.129</td>
<td>0.569</td>
<td>0.753</td>
</tr>
<tr>
<td>stream_read_frame_body</td>
<td>0.011</td>
<td>0.543</td>
<td>1.160</td>
</tr>
<tr>
<td>decode_header</td>
<td>0.037</td>
<td>0.530</td>
<td>0.742</td>
</tr>
<tr>
<td>play_frame</td>
<td>0.021</td>
<td>0.528</td>
<td>0.574</td>
</tr>
<tr>
<td>wav_write</td>
<td>0.009</td>
<td>0.523</td>
<td>0.475</td>
</tr>
<tr>
<td>stream_head_read</td>
<td>0.004</td>
<td>0.521</td>
<td>0.832</td>
</tr>
<tr>
<td>__divsi3</td>
<td>0.002</td>
<td>0.508</td>
<td>0.000</td>
</tr>
</tbody>
</table>
Energy accounting:

➔ ECOSystem (Zeng, et al, 2001)
  • *currentcy* is used as a primitive for energy allocation
  • *currentcy* is only valid for a certain time
  • resource containers are used for resource charging
  • model-based techniques are used to charge RCs.
  • internal policy is used to select resource containers

➔ LEA
  • charges using real power measurements
Energy accounting:

- ECOSystem (Zeng, et al, 2001)
  - currentcy is used as a primitive for energy allocation
  - currentcy is only valid for a certain time
  - resource containers are used for resource charging
  - model-based techniques are used to charge RCs.
  - internal policy is used to select resource containers

- LEA
  - charges using real power measurements

Energy Budgeting:

- a budget is allocated to each resource container
- resource containers in debt are not scheduled
- the budget is periodically refreshed
Energy Accounting


- use CPU performance counters to estimate CPU temp
- account the energy contribution of each RC
- schedule “hot” processes interspersed with “cold” processes
- can maintain a given temperature limit