Adaptive Computing

Gus Uht
(w/ Rick Vaccaro)

Dept. of Electrical and Computer Engineering

Copyright © 2003-5, A. K. Uht, R. J. Vaccaro
Patent applied for.

BARC: January 21, 2005
Background

• Three typical goals (alone or combined):
  1. ~Maximize performance.
  2. ~Minimize power consumption.
  3. Adapt to current (or prior) conditions:
     a. Environmental, e.g., temperature. (current)
     b. Operational, e.g., power supply voltage. (current)
     c. Manufacturing, e.g., slower or faster chips. (prior)

• ~All adaptive methods applicable to all.
• “Better-than-worst-case design”
Outline

1. Adaptive Methods
2. TEAtime – Prototype – Picture
3. Adaptive Control System
4. TEAPC & TEA42 – Prototypes and Data
5. Demo – TEA42
6. Summary
Adaptive System Methods

- **Timing Error *Toleration*:**
  1. Change something (freq., volt.), then:
  2. *Let error occur, detect it, then recover.*
     - Complex, hard to design; only ~optimal within cycle.

- **Timing Error *Avoidance*:**
  1. Change something (freq., volt.), then:
  2. *Stop just before error would occur.*
     - Simple, easy to design & build; constant cycles.
TEAtime Prototype
TEAPC, TEA42 Goals

• Motivating Goals:
  – Realize TEAtime characteristics in a real computer.
  – *Adaptive computing*, in particular.

• Additional Goals:
  1. *Workload adaptation.*
  2. *Reduced power consumption.*
  3. *Improved reliability.*
  4. *Disaster tolerance (always enabled).*
  5. *...and all in a production machine.*

• **BUT**: can’t redesign or build Pentium 4’s.
• **SO**: use real IBM/Intel-standard PC.
Current Control System

- Only input is CPU temperature (feedback line).
- Primary output is CPU frequency (N).

[sometimes: Vcore = f'(N)]
TEAxx Prototypes’ Features

• Hardware:
  – No modifications → all COTS parts.
  – System: ~3.0 GHz Pentium 4;
    Intel chipset; 1 GB RAM.

• Software (teapcwin program):
  – Realizes feedback control system.
  – Standard MS Windows application.
  – (Standard OS: W2K SP4; no modifications [of course].)
  – Small: 1.1 megabytes.
  – Fast: low CPU utilization.
freq., Vcore unlinked < Load Adaptation > freq., Vcore linked
...and now it’s time for the: DEMO
Overall Summary

• TEAtime, TEAPC & TEA42 realize:
  2. *Adaptive* operation to both environment and/or loading.
  3. Low-power, high-reliability operation.
  4. Disaster tolerance.

• Feedback-control great for a system, too.

• *Adaptive systems* are the way to go.

• *They Work!!*
Pointers…

- Appendix.
- My website: [www.ele.uri.edu/~uht](http://www.ele.uri.edu/~uht)
- Or μRI website: [www.ele.uri.edu/muri](http://www.ele.uri.edu/muri)
Adaptive Computing

Gus Uht
(w/ Rick Vaccaro)

Dept. of Electrical and Computer Engineering

Copyright © 2003-5, A. K. Uht, R. J. Vaccaro
Patent applied for.

BARC: January 21, 2005
Appendix

15. Timing Error Tolerance.
18. TEAPC Block Diagram.
19. TEAPC Components.
20. [TEAPC] Experiment Setup.
21. TEAtime Block Diagram.
22. [TEAtime Data.]
23. TEAtime Summary.
24. TEA42 Disaster Tolerance.
Timing Error Tolerance

- **Operation:**
  1. *Speed up clock* (resp. *reduce voltage*) until error occurs.
  2. *Slow down clock* (resp. *increase voltage*) for no error.
  4. Repeat: GOTO 1.

- **Examples:**
  - Uht 2000: *TIMERRTOL*: performance ~maximized; not pursued.

- **Plusses:**
  - True (or better than) perf. max., resp. power min. [within cycle]

- **Minuses:**
  - Can be costly (TIMERRTOL: 2x cost; Razor: little extra cost)
  - Is complex and hard to design.
  - Can lead to *increase* in cycles → perf. may NOT be true max.
Timing Error Avoidance

- **Operation:**
  1. *Speed up clock* (resp. *reduce voltage*) ‘til *just before* error occurs.
  2. *Slow down clock* (resp. *increase voltage*) for no error.
  3. *Repeat: GOTO 1.* [No backtracking or repair needed.]

- **Examples:**
  - Uht 2003: *TEAtime*: ∼performance maximization; prototype built.
    - One-bit wide slowest-path test logic always errs before real logic does.

- **Plusses:**
  - Close to true performance maximization, resp. power minimization.
  - Simple, easy to design and build. (TEAtime)
  - No increase in cycles.

- **Minuses:**
  - Shorter papers. *(no, wait, that’s a plus…)*
Adaptive Control Systems

- **TEAtime:**
  - Simple up/down control system – 0 delay.
- **Skadron, Bahar:**
  - Classic feedback control system.
  - In hardware.
- **TEAPC, TEA42:**
  - Bigger delays.
  - State-space feedback control system.
  - In software; little overhead.
TEAPC Block Diagram

- CPU: Intel P4
- Northbridge: Intel 875P
- Main Memory: 1 GB Dual Channel 400 MHz Ultra
- Southbridge I/O Controller: Intel ICH5R
- Super I/O: ITE 8712F
- Clock Synthesizer: ICS 952635
- CPU Vcore Regulator: Control
- CPU Vcore Power Supply
- Main Memory
- FSB (Front Side Bus)
- LPC Bus (Low Pin Count)
- SMBUS - IIC Bus
- CPU Clock
- CPU Vcore Volt.
- CPU Fan Speed
- CPU core Temp.
- Only directly relevant components and connections are shown.

(FSB - Front Side Bus) (LPC - Low Pin Count)
## TEAPC Components

<table>
<thead>
<tr>
<th>PC Component</th>
<th>Manufacturer</th>
<th>Part Number/Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motherboard</td>
<td>Gigabyte</td>
<td>GA-8KNXP (Rev. 2); w/DPS regulator</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel</td>
<td>P4 3.0 GHz, 800 MHz bus</td>
</tr>
<tr>
<td>Chipset</td>
<td>Intel</td>
<td>875P, ICH5R</td>
</tr>
<tr>
<td>Clock Synthesizer</td>
<td>ICS</td>
<td>ICS952635</td>
</tr>
<tr>
<td>Super I/O (Environment Mon.)</td>
<td>ITE</td>
<td>IT8712F V0.6</td>
</tr>
<tr>
<td>CPU Volt. Regulator Control</td>
<td>ITE</td>
<td>IT8206R V0.1</td>
</tr>
<tr>
<td>Main Memory</td>
<td>Ultra</td>
<td>U10-5903R; 2 x 512 MB; 400 MHz DDR, Dual Channel (Operated at 320 MHz.)</td>
</tr>
<tr>
<td>Operating System</td>
<td>Microsoft</td>
<td>Windows 2000 SP4, HT disabled</td>
</tr>
<tr>
<td>Disk System – RAID 0+1</td>
<td>ITE</td>
<td>GigaRAID IT8212F</td>
</tr>
<tr>
<td>Disks</td>
<td>Maxtor</td>
<td>4 x 6E040L0, 40 GB, 133MHz IDE</td>
</tr>
<tr>
<td>Equipment for experiments only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fan Controller &amp; Temp. Mon.</td>
<td>Thermaltake</td>
<td>Hardcano 12; for 4 fans, 4 thermocouples</td>
</tr>
<tr>
<td>Power Meter</td>
<td>Electronic</td>
<td>watts up? PRO</td>
</tr>
<tr>
<td></td>
<td>Educational</td>
<td>(Note: this is the unit’s model name.)</td>
</tr>
<tr>
<td></td>
<td>Devices</td>
<td></td>
</tr>
<tr>
<td>CPU Fan Controller</td>
<td>custom</td>
<td>On/Off, control sel. (MOBO or Hardcano)</td>
</tr>
</tbody>
</table>
Experiment Setup
TEAtime Block Diagram

- **Timing Error Avoidance system**
  - **Blue**: TEAtime hardware
  - **Green**: on FPGA

NOTES: - w, x >> 1 ; - DAC: Digital-to-Analog Convertor ; - VCO: Voltage-Controlled Oscillator.
Frequency & DAC setting vs. time (Temp. = 22° C.)
TEAtime Summary

- Performance of prototype improves by $\geq 34\%$
- TEAtime provides adaptable frequency control of any synchronous digital system
- Always $\sim$maximizes performance
- Very cheap
- Very easy to add to existing or future designs
- $\sim$Can be adapted to physically existing systems
- *It Works!!*
TEA42 Disaster Tolerance

• Example: CPU Fan dies….

• Changes (automatic, via feedback system):
  – Freq: 3.15 GHz → 1.5 GHz
  – Vcore: ~1.5 V. → ~1.1 V.
  → Power: ~160 W. → ~100 W. (~37% savings)

• CPU temperature stabilizes at safe value (with this CPU).

• System still works.