Software Driven Low Power Optimization for ARM Based Mobile Architectures

Frank Schirrmeister, Synopsys
Agenda

- A Mobile Device
  - Where did all the energy go?
  - Components of Energy Consumption

- Software Rules Power
  - Challenges
  - Virtual Platforms
  - SystemC TLM-2.0
  - Enabling Virtual Platforms for Low Power

- Results
  - Demonstration: Freescale i.MX31
  - Demonstration: Texas Instruments OMAP2420

- Conclusions
A Mobile Multimedia Device Example

WHERE DID ALL THE ENERGY GO?
A Mobile Device

CDMA phone puts features in focus

Offering digital zoom and special effects, a camera, of course, takes center stage in a Toshiba phone with a lot going for it
What’s in a Mobile Device?

---

**Why?**

- **A/V and Transport**
  - Toshiba #1C35280
  - Sharp #593M37
  - Qualcomm #RF6100
  - Fujitsu #5B5H5/SK3/5K

- **Baseband**
  - Toshiba #W0100700000G8
  - Sharp #593M37
  - Fujitsu #5B5H5/SK3/5K

- **LNA**
  - Dual LNA
  - Yamaha #MTU7600
  - Analog Devices #AD7118

- **Power**
  - Power Analog ASIC
  - Ricoh #TC5775

- **Memory**
  - Toshiba #W0100700000G8
  - Sharp #593M37
  - Fujitsu #5B5H5/SK3/5K

- **Audio**
  - Power Analog ASIC
  - Ricoh #TC5775

- **Video**
  - Toshiba #W0100700000G8
  - Sharp #593M37
  - Fujitsu #5B5H5/SK3/5K

- **Display**
  - Toshiba #1C35280
  - Sharp #593M37
  - Qualcomm #RF6100

- **RF**
  - Dual LNA
  - RF to Baseband receiver and GPS LNA
  - Qualcomm #RFS6800
  - Epson #D137150

- **Processing**
  - Mobile Station Modem processing
  - Toshiba #W0100700000G8
  - Sharp #593M37
  - Fujitsu #5B5H5/SK3/5K

---

**More info:**

- Toshiba #1C35280: Encoding/decoding of video and speech data and multiplexing/demultiplexing
- Sharp #593M37: Dual dc/dc converter for LCD
- Qualcomm #RF6100: Transmit baseband-to-RF converter
- Qualcomm #RFS6800: Direct conversion RF-to-baseband receiver and GPS LNA
- Epson #D137150: LCD controller
- Analog Devices #AD7118: Video encoder with advanced power management
- Ricoh #TC5775: Power/Analog ASIC
- Yamaha #MTU7600: Power/Analog ASIC
- Fujitsu #5B5H5/SK3/5K: 3-die multichip package
- Toshiba #W0100700000G8: 2-die multichip package
- Sharp #593M37: Dual NAND Flash—16 Mbytes
- Fujitsu #5B5H5/SK3/5K: Dual I—EFCRAM—8 Mbytes

---

**Other technologies:**

- Toshiba: 2-die multichip package
- Sharp: Dual NAND Flash—16 Mbytes
- Fujitsu: Dual I—EFCRAM—8 Mbytes

---

**Source:**

- ARM 2008 Developers' Conference

---

**Confidential**
And where did all the energy go?
Six CPUs and Various Accelerators!

- 3 CPUs
- AMS
- AMS
- AMS
- Control
- AMS
- Analog
- Audio Processor

- Memory
- Memory
- 2 standard CPUs
- 4-6 accelerators

Diagram showing various components such as LCD Driver, A/V and Transport, LCD Control, Memory, Audio Processor, Control, and other labeled parts.
Software Rules!
# Power Optimization Techniques

## Impact on Optimization

<table>
<thead>
<tr>
<th>Design Time</th>
<th>Analog/MS</th>
<th>CTRL HW</th>
<th>DF HW</th>
<th>Memory</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>ES Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RT Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gate Level</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Layout</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Highest Impact Optimizations
A three minute excursion to the basics!

COMPONENTS OF ENERGY CONSUMPTION
Sources of Power Consumption

\[ P_{\text{average}} = P_{\text{short}} + P_{\text{leakage}} + P_{\text{dynamic}} \]

Energy consumption during the switching of CMOS gates when the complementary parts are open simultaneously

Energy consumption caused by currents during the non-conducting state of gates

Dynamic energy consumption happens during the data dependent switching of capacities in transistors and the connections between them
Short Circuit Energy Consumption

$P_{\text{short}}$

- Energy consumption during the switching of CMOS gates when the complementary parts are open simultaneously
- Shorter transition times can help optimize this type of energy consumption
- Main influencing parameters
  - Voltage
  - Operating Speed

$$\frac{\beta}{12} (V_{dd} - 2V_{th})^3 \frac{\tau}{T}$$
Leakage Energy Consumption

$P_{\text{leakage}}$

- Energy consumption caused by currents during the non-conducting state of gates
- Main influencing parameters:
  - Supply Voltage $V_{dd}$
  - Threshold Voltage $V_t$
  - Transistor Size $W/L$
  - Transistor State
  - Temperature
Dynamic Energy Consumption

\[ P_{\text{dynamic}} \]

- Dynamic energy consumption during data dependent switching of capacities (transistors & connections)
- Often largest contributor
- **Main influencing parameters**
  - \( V_{dd} \) (Squared)
  - Capacity
  - Activity

\[ \frac{1}{2} C_{\text{load}} \cdot \alpha \cdot V_{dd}^2 \cdot f \]

Source: ChipVision
Energy Consumption Trends

**Observations:**
- Smaller geometries require improved handling of leakage
- Dynamic energy due to capacity switching remains significant in mainstream design well into the 65nm technology node

**Conclusion**
- Optimize data efficiency
- Optimize activity, resources
- Optimize Software
- Reduce voltage
  - Lower clock speed for slower cells at lower voltage
Mobile Multimedia Drives Energy

- Multimedia is performance hungry
  - Dominates power consumption
  - Drives processing speed
- Software!
  - Impacts memory (caches)
  - Drives processor speeds
- Memory accesses!

Conclusion
- Multimedia/Consumer
  - Drives higher clock speed
  - Drives lots of data transfers

Source: Linux World 06'06
Impact of Software on Power Consumption

SOFTWARE RULES
“Phone differentiation used to be about radios and antennas and things like that. We think, going forward, the phone of the future will be differentiated by software.”

Steve Jobs
CEO, Apple
August 11, 2008
http://online.wsj.com/article/SB121842341491928977.html
There really are only two issues …

Software starts late

Bugs are expensive

Software has to wait for prototype => Semi cannot sell silicon …
Start Software as Early as Possible

Software has to wait for prototype => Semi cannot sell silicon …
End-To-End Prototyping

**Virtual Platform: Pre-RTL**
- Fully functional software model of SoC, board, I/O, user interface
- Executes **unmodified** production code
- Runs close to real-time with external interfaces as Virtual I/O
- High system visibility and control incl. multi-core debug

**FPGA Prototype: Pre-Silicon**
- Fully functional hardware representation of SoC, board, I/O
- Implements **unmodified ASIC RTL code**
- Runs at almost real-time with all external interfaces and stimulus connected
- High system visibility and control

**Silicon Prototype: Post-Silicon**
- Using the real chip
- Running at real time
- Little visibility and control
Synopsys’ Virtual Platform Portfolio

DesignWare® System-Level Library

High-performance models to build virtual platforms

Innovator

Environment for developing, running & debugging virtual platforms

Services

Expert services for model creation, virtual platform assembly & customization

Virtual Platforms

- Start SW development early and shrink time-to-market using high-performance Virtual Platforms
- Enhance design quality through SystemC executable specification
- Increase design confidence through complete HW/SW system verification
Virtual Platforms in the Design Flow

Functional Specification – Defining the overall hardware / software system

Model Creation
- DesignWare® System-Level Library
- System Studio
- Synopsys Creator (XML)
- SystemC
- Your-lib-1

Platform Creation & Analysis
- Synopsys Innovator

Platform Deployment
- Synopsys Innovator-RT
- 3rd Party Software Debuggers

RTL to GDSII Implementation Flow (Discovery VCS to Galaxy & Synplicity)
The Impact of SystemC TLM-2.0

Previously proprietary (backdoor) APIs & new additions have now been standardized:

- **(DMI) Direct Memory Interface**
  - Direct backdoor access into memory
  - Allows un-inhibited ISS execution
- **LT (Loosely Timed) modeling**
  - Timing declaration
  - Allows speed/accuracy trade-offs
- **Temporal Decoupling**
  - Only synchronize when necessary
  - Allows multicore speedup

![Diagram showing performance and accuracy trade-offs.](Diagram.png)

AV: Application View
LT: SystemC Loosely Timed
AT: SystemC Approximately Timed
CA: Cycle Accurate

 disputed rights
The Impact of SystemC TLM-2.0

- Innovator
  - PV (LT) modeling
  - PV+T (AT) “timing annotation”
- DesignWare® IP
  - Implementation, Verification and System-Level IP
- Modeling Services
- VCS
  - For hard CA requirements running in software
- Synplicity HAPS

![Diagram showing the impact of SystemC TLM-2.0 on application software development, Firmware development, and Hardware development, highlighting the accuracy and performance trade-offs between Application View (AV), SystemC Loosely Timed (LT), SystemC Approximately Timed (AT), and Cycle Accurate (CA).]
LOW POWER ENABLED VIRTUAL PLATFORMS
Power Instrumentation Objectives

- Increase visibility
  - Global system state
  - Individual power/clock domains
- Power trade-offs
  - Performance vs. power
  - Power schemes with “system software” load
- Power-related SW development
  - Relative power consumption used for optimization
  - Power management software
  - System power consumption while running actual software
Power Modeling Support

- The following power support is added to a TLM platform:
  - Power management modeling
    - Clock modeling – gating, scaling freq(t), …
    - Voltage distribution - power domains, scaling V(t) , …
    - Power state control – power state sequencing (power down, retention, …)
  - Power estimation equations – evaluated at run-time
  - Dashboards – clocks, state, voltage, power
- Applies both to LT & AT modeling
  - LT
    - Instantaneous power consumption, at each point in time
  - AT*
    - Supports trade-off of performance vs. power
    - Graphs: Power(t) & Energy(t)
    - Improved accuracy for accounting for (memory) transactions power contribution
  
* Under development
Functional clock modeling

- Models functional operation of the clock controller, including:
  - Clock distribution
  - Includes its control over peripheral clock gating
  - Registers, to model software control over clock frequencies

Signal is used to transmit value of clock frequency from controller to peripheral; does not model actual clock waveform
Voltage & Power State Control Modeling

- Functional models of:
  - On-chip Power Manager (SoC)
    - Control of (internal) voltage distribution & domains
    - State control & sequencing
  - Power management chip (PMIC)
    - Voltage scaling of SoC
    - SoC I2C control interface, power sequencing, LDO regulators control, DC/DC convertors, …
Power Estimation - Concepts

- Parametrizable power model, consisting of:
  - Component power characteristics
  - Component power calc. equations
  - Power accumulator
Parametrizable model:
- Power characteristics
- Power equations
- Power accumulator

Example – PRCM (OMAP2420)
Component Characterization for Power

- Power parameters
  - Components are characterized by a set of representative power parameters ('kernels')
  - Used in power equations to calculate power
  - Flexible to support specific component characteristics
  - Interactively changeable by user

- Example
  - CPU
    - Power active (mA / MHz)
    - Power dormant (mA)
    - Power inactive (mA)
    - Power shutdown (mA)
  - Peripherals:
    - Power clock off (mA)
    - Power idle (mA / MHz)
    - Power typical (mA / MHz)
    - Power maximum (mA / MHz)
  - On-chip Memories (RAM / ROM)
    - Power clock off (mA)
    - Power idle (mA / MHz)
    - Power read (mA / MHz)
    - Power write (mA / MHz)

- Source
  - Power consumption numbers are delivered by semiconductor company
  - Based on (1) budget planning, (2) estimations, (3) measurements
Power Estimation

- System power estimations expressions, contain:
  - Component power characteristics
  - Power state of APLLs & DPLLs \{ off, on \}
  - Power state of domain \{ off, ret, on \}
  - Voltage applied (to domain)

- At run-time power estimation expressions are evaluated on the fly, as triggered by user requests
  - Complements functional component model
  - Leverage power state & frequency modeling of func. component model
  - Expressed in C code (Magic-C or C++)

- Expressions can be linear, or more complex, depending on:
  - Component type
  - Data / characteristics which can be measured / estimated

- Fixed modeling APIs for voltage, frequency & power state updates, and reporting to accumulator
Power Estimation - Example

- ARM1136 (OMAP2420)

- MPU Voltage (Volt)
- MPU Clock (frequency (MHz))
- MPU Power State (on, off, ...)

ARM MPU Power Estimation Component

Graphical C Power Model
Power Estimation – AT Platform

- Fixed power $f(V,f,st)$
- Penalty when Cache miss ($\Delta_{cache 	ext{ iss}}$)
- Penalty for each bus transaction ($\Delta_{\text{transaction (length)}}$)
- Fixed power $f(freq,St)$
- Penalty for each bus transaction ($\Delta_{\text{sys memory (length)}}$)

Platform Analyzer *

* Under development
Dashboards Examples

- On-chip Power Reset Controller
- Power Dashboard
- Clock Dashboard
- Voltage Monitor
- SoC Voltage Dashboard
Demonstrations

RESULTS
Demo 1: Linux on OMAP2420
Demo 2: WinCE on i.MX31
CONCLUSIONS
Software Rules Power Consumption

- Multimedia drives power consumption in mobile applications
  - Software
  - Memory Accesses
- Virtual Platforms allow presilicon embedded software development
  - TLM-2.0 enables interoperability
  - Power instrumentation allows trade offs and development of power related software