System-level Energy Modelling

Geza Lore
ARM

Jose Nunez-Yanez
University of Bristol

EACO Workshop
13-July-2011
Motivation

- The why is simple:
  - Today’s systems are getting increasingly complex
  - Innovation requires information

- To build a competitive product:
  - you need to consider the big picture

- People are doing work on system level performance exploration

- The power side of the world is also non-trivial
  - Complex power management
  - Varying use cases

- Could we build tools to help us design systems and optimize software with a more power conscious mindset?
Problem statement

- **Goal**
  - Enable power estimation of full systems running complete user applications

- **It needs to be...**
  - Accurate
    - Can’t use simple on/off models based on simple test vectors
  - Fast
    - Can’t use standard backend power analysis tools
  - Full system
    - Need to be able to consider non-SoC parts
Abstraction levels for energy modelling

- **Currents**
- **Voltages**
- **Transistor models**

- **Static probabilities**
- **Toggle rates**
- **Characterised libraries**

- **Macro states**
- **Architectural events**
- **High level power models**

---

**SPICE**

**PrimeTime PX**

**High level modeling**
High level power models - Example

- **Energy estimated based on:**
  - Time spent in defined *Macro States*
  - Frequency of defined *Architectural Events*

\[
E_T = \sum_n P_n t_n + \sum_m E_m N_m
\]

- **Some example CPU states / events:**
  - Core state: Active / Stall / WFI
  - SIMD unit: Clocked / Clock gated
  - Events: D cache hit / D cache miss / Instruction executed
  - Sub module level granularity of trace: CPU0 / CPU1 separate

- **Remember:** This is ONE option. You can build much more complex mathematical models
IP characterisation

- Power models need to be characterized
- For RTL IP, you can follow:
  - Implement design
  - Create an RTL and/or Netlist based test bench
  - Create a set of power benchmarks
  - Capture activity for PTPX (SAIF/VCD) and high level logs
  - Process activity information to get energy and activity vector
  - Create the models using linear regression
CPU average power model

- Relatively simple model
  - Based on 10 inputs
  - Mostly performance counters
- Characterized using random instruction sequences
- EEMBC benchmarks show max 7% error
  - Reference Power from PrimeTime PX
Some applications

System:
- Dual core ARM CPU
- Variable size L2 cache
- AMBA Interconnect
- Dynamic Memory Controller
- LPDDR2 PHY
- LPDDR2 memory chip
- Variable BW traffic generator

Platform & Workload:
- HW emulation
- Running full benchmarks under Linux
Microbenchmarks – P(t)

- Force system to corner cases
- Algorithm choice and dataset can cause 3x diff in CPU Power
- And possibly more in Energy
- You need to worry about the system

![Dynamic power graph]

- Mem Stress
- Calculate
- Factorial
- Recursive Sum large stack
- Recursive Sum small stack
- Loop Sum

Time [s]

Power

Total | CPU | Fabric | PHY | DRAM
Web browser – $P(t)$

Dynamic power

<table>
<thead>
<tr>
<th>Total</th>
<th>CPU</th>
<th>Fabric</th>
<th>PHY</th>
<th>DRAM</th>
</tr>
</thead>
</table>

Power

Time [s]

CPU + L1
L2 Cache
Interconnect
DMC
PHY
LPDDR2

The Architecture for the Digital World®
Web browser –

Dynamic power

Power

Total – CPU – Fabric – PHY – DRAM

Time [s]

The Architecture for the Digital World®
Summary

- Accurate power estimation is possible
  - Accuracy is limited by the accuracy of your reference data and the accuracy of your simulator
- Methodology is flexible
  - Mathematical form of power models
  - Source of reference data
  - Can address non SoC parts and non RTL IP
- Speed is limited by the speed of the system simulator
  - Future work includes power modelling on an execution model
- Can explore system architecture tradeoffs
- Can profile applications for energy consumption