Modeling Computation for HW/SW Codesign
LCPC 2012 Keynote

David J. Kuck
Software & Services Group

Waseda Univ.
Sept. 12, 2012

Copyright © 2012, Intel Corporation. All rights reserved.
# Prologue for my current codesign work

<table>
<thead>
<tr>
<th>Topic</th>
<th>1960s</th>
<th>70s</th>
<th>80s</th>
<th>90s</th>
<th>2000s</th>
<th>2010s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture</td>
<td>Illiac 4 → {Multics}</td>
<td>Burroughs BSP</td>
<td>Alliant Cedar</td>
<td>Intel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Tranquil, Glypnir</td>
<td></td>
<td>ANSI X3H5 → OpenMP</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use tools &amp;</td>
<td>Parafrase → KAP</td>
<td>Perfect Club</td>
<td>SPEC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model &amp; Measure</td>
<td>Intel</td>
<td></td>
<td>Tools Hard to optimize for all</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[no guessing]</td>
<td>System Capacity →</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Local design intuition weak v. global
- Lang. focus/extension → acceptance
- Good methods have lifetimes
- Extend useful ideas when understanding & need arise

**Codesign Goal:** Methodology for Designing Systems for a Given Workload that we KNOW will Behave as we Specify.
Outline

Prologue & Outline
A. System Codesign Example
B. Modeling
   1. Intro
   2. Examples
   3. Power, Energy
   4. Summary Equations
C. Measurement and virtual nodes
   1. Model synthesis tools
D. Example Cape tool results
   Cost/perf and Energy/perf designs
E. Conclusions

Extensive, coherent data: fast global analysis
A1. System Behavior – Quality Objectives

- Stability(unicore, \textit{perf range}, Data range)
- Scalability(\textit{freq}, D) [ISV: what $D$-range matters?]
- Speedup(\textit{proc}, D) [how many cores/chip?]
- Energy, Energy efficiency, Power, ... [many questions]
- Cost – initial, operating [how do we define?]
- Combinations of the above [how do we define?]
  - Depending on system type
  - Server, laptop, handheld, ...

Goal: HW/SW Codesign Methodology that handles all of this
Example: Consider 2 NR Codelets

Hqr13 Source:
SUBROUTINE codelet (n, m, i, a, res) ! m = 10, i = 1

integer n, j
real*8 a(m, n), s, res (1)

s = real (0)
do j = 1, n
   s = s + abs (a (j, i))
end do
res (1) = s

END SUBROUTINE codelet

Svdcmp11 Source:
SUBROUTINE codelet (n, m, i, a, f) ! m = 20, i = 1

integer n, m, j, i
real*8 a(m, n), f

do j = 1, n
   a (i, j) = a (i, j) * f
end do

END SUBROUTINE codelet

• Which has good or bad: Stab\[D\]<2, Scal[f,D] Sp[D]?
Codelet 1: Sp(4), Sf(2x) vs. D

1 core/low freq

1c/hi f

t=18

Saturation crossover

Perf range

Data size

L3

ram

3 for/hi f

Data size

40K

400K

4c/lo f

4core/high freq

Hqr_13_dp_sse
Codelet 2: Svdcmp_11_dp_sse

1c/lf

All load/store saturated

Floating point time

1c/hf

4c/lf

Perf range

4c/hf

Cycles per iteration

Data Size

Cycles per iteration

Data Size

Cycles per iteration

Data Size

Cycles per iteration

Data Size
## Quality Results

<table>
<thead>
<tr>
<th>Codelet</th>
<th>Hqr13</th>
<th>Svdcmp11</th>
<th>40K&lt;D&lt;400K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stability</td>
<td>OK</td>
<td>2.5 X</td>
<td>range dependent</td>
</tr>
<tr>
<td>Scal f</td>
<td>OK</td>
<td>No @ &gt;60K</td>
<td></td>
</tr>
<tr>
<td>Speedup 4</td>
<td>Bad at 400K</td>
<td>All bad</td>
<td></td>
</tr>
</tbody>
</table>

**Increase L3?**

- **Study sensitivity to D range**
- **Rewrite or recompile? Redesign stride HW?**

**Codesign problems arise everywhere**
B.1. Modeling: 4D Codesign Space

(w.o. Oper. Cost E)

1. Performance

- Best Performance

2. Initial Cost

- Orig. Perf/Cost plane

3. SW Load

- Enhanced Perf/Cost plane

D. Market-focused Systems

A. Original System

B. Performance-enhanced System

C. Application-enhanced System

Electrical dimension: energy via (clock f, V) → Operating Cost
2 Models:

a. HW System Arch.

Node $i$ has Bandwidth $B_{i} \left[ \frac{b; O}{s} \right]$

$BW \ ratio = \alpha_{i,j} = B_{j} / B_{i}$

Power $= W_{i}[w]$

Energy/\text{bit} $= \gamma_{i} \left[ \frac{w \cdot s}{b} \right]$

Phase $j$ has Operation count $O_{i,j}[ops]$ on node $i$

b. SW Computation

Prog + Data + HWsyst $\rightarrow$
A Computation on a System Produces:

For node $i$, phase $j$:

- **Running time** $= t_{i,j}[s]$
- **Energy** $= E_{i,j}[j \text{ or } w \cdot hr]$

- **Computational Capacity** $= C_{i,j} \left[ \frac{b;O}{s} \right]$

- **Capacity Equation** (node $i$):
  \[
  C_{i,j} = \frac{O_{i,j}}{t_{i,j}} \quad 0 \leq \sigma_{i}^{C} = C_{i} / B_{i} \leq 1
  \]
  
  $C_{i,j}$ s.t. $i$ saturated

- **Capacity Intensity Equation** (nodes $i,k$):
  \[
  C_{i,j} = \mu_{ki,j} C_{k,j}
  \]

Each phase saturates one or more nodes
Balanced nodes, phase balance point

Nodes balanced, unsaturated

Perfect BW for this computation

Phase balance point, saturated node

Saturated intensity equation

Initial BW

$B_{x,k}^{bal} = \mu_{y,x,k} B_y$

$slope = \mu_{xy}$

$\mu_{x,y,k} = \alpha_{x,y}$
B.2. Codesign Examples:
1-phase, 3-node system

For node \( i \), phase \( j \):
- **Computational Capacity**
  \[
  C_{i,j} = C_{i,j} \left[ \frac{b}{s} \right]
  \]

- **Capacity Equation** (node \( i \)):
  \[
  C_{i,j} = \frac{O_{i,j}}{t_{i,j}}\quad (\text{At least one node is saturated})
  \]
- **Intensity Equation** (nodes \( i,k \)):
  \[
  C_{i,j} = \mu_{ki,j} C_{k,j}
  \]

How do we systematically increase system performance?
- Focus on **saturated** node(s)
- Intensity \( \mu_{ki,j} \) is **invariant** if SW is not changed

Simple design questions follow

Copyright © 2010, Intel Corporation. All rights reserved
Ex.1: Systematically boost syst. perf.?

If $C_{disk,j} = \overline{B}_{disk}$, increase disk performance:

- Set saturated node BW to desired performance level
- Adjust other nodes accordingly \( \Rightarrow \) use $C_{cpu,j} = \mu_{disk-cpu,j} C_{disk,j}$

What if performance demand exceeds the fastest cpu or mem available?

Drive toward $B^{waste} = B - C = 0$, subject to discrete node values
Ex.2: Perf. demand > fastest node available?

If \( C_{cpu,j}^{goal} > B_{cpu,j}^{max} \), use parallel cpu model:

- Replicate node – parallel processors or memories add BW
  - \( B_{cpu}^{parallel} = B_i + B_k \)

Parallel capacities lead to multirate nodes, \( C_{cpu,j}^{parallel} \leq 2C_{cpu,j} \).
Ex.3: What if disk has latency?

Latency models: transmission, contention, or rotational delay?

- Physical transmission delay – constant, function of wire length
- Serial nodes add reciprocal BWs, nonlinear capacity

\[
\frac{1}{B_{ik}^{\text{ser non-ov}}} = \frac{1}{B_i} + \frac{1}{B_k} \\
B_{ik}^{\text{ser ov}} = \min\{B_i, B_k\}
\]
Summary Internode Equations

Local Eqs.

**Capacity Intensity:**
\[ C_{i,j} = \mu_{ki,j} C_{k,j} \rightarrow \mu_{ki,j} B_k \]

**Parallel supernode**
\[ C^\text{parallel}_{ij} = C_i + C_j \]

**Serial supernode**
\[ \frac{1}{C^\text{ser non-ov}_{ij}} = \frac{1}{C_i} + \frac{1}{C_j}; \quad C^\text{ser ov}_{ij} = \min\{C_i, C_j\} \]

**Multirate nonlinear supernode**
\[ C_i = \frac{f_1(C)}{1 + f_2(f_3 - 1)} \quad 0 \leq f_2, f_3 \leq 1 \]

Piecewise linearize

fractional use

perf function

System of (nodes X phases) inequalities\(\rightarrow\) Global Codesign

Copyright © 2010, Intel Corporation. All rights reserved
B.3. W and E: Average-power model

\[ W_i = \text{power [watts]} \]

- Node on high: \( W_i^{\text{high}} \)
- Low power/BW state: \( W_i^{\text{low}} \)
- Node idle: \( W_i^{\text{idle}} \)
- Node off

\[ C_i^{\text{low}} (f^{\text{low}}) \]
\[ C_i^{\text{high}} (f^{\text{high}}) = B_i^{\text{max}} \]

Model allows arbitrary slopes

\[ \gamma_{i,j}^{k} \text{ [E/b]} \text{ for node } i, \text{ phase } j, \text{ power state } k, \ 1 \leq k \leq s \]

Power Equation

\[ W_{i,j}^{k} = \gamma_{i,j}^{k} C_{i,j}^{k} + W_{i}^{\text{idle}} \]

Combine with architecture codesign model
Energy efficiency vs. Capacity Rel. Saturation

Ideal operating area, good $\varepsilon_{i,j}^{xE\sigma}$, even at low load; perf/E proportional computing

Network server

Battery laptop

HPC or gamer

Want low idle $W$, high $E/b$

$\varepsilon_{i,j}^{xE\sigma} = \frac{xC_{i,j}}{\tau_{i,j}^x}$

Energy efficiency $\varepsilon_{i,j}^{xE\sigma} = \frac{\sigma_{i,j}^{xC}}{\sigma_{i,j}^{xyE}} = \frac{C_{i,j}^x W_{i,j}^{yE_{max}} t_{i,j}^{yE_{max}}}{B_{i,j}^x W_{i,j}^x t_{i,j}^x} = \frac{E_{i,j}^{max}}{E_{i,j}}$
Multiprogrammed jobs/energy vs. Capacity relative saturation

Ideal Operating Range: Perf/E proportional comp.

Unnormalized $\varepsilon_{i,j}^{x E\sigma}$

Unnormalized $\frac{j \text{ obs/} E_{syst}^x}$

Serious overhead time Interference

Multiprogramming system Overhead

“Server consolidation” problem

**Capacity**
\[ 0 \leq C_{i,j} = O_{i,j} / t_{i,j} \leq B_i^k (f, V) \leq B_i^{\max} = \max_k \{ C_{i,k} \} \rightarrow B_i^{\phy} \]

**Latency**
\[ 0 \leq B_i^{-1} = L_i^{\min} \leq L_i = C_i^{-1} \leq L_i^{\max} \]

**Power**
\[ 0 \leq W_i = \gamma_i C_i + W_i^{\idle} \leq W_i^{\max} \]

**Energy**
\[ E_{i,j} = W_{i,j} t_{i,j} = \gamma_{i,j} O_{i,j} + W_i^{\idle} t_{i,j} \]

- **Initial Cost** = \[ \sum B_j \]
- **Performance** = \[ \sum \sum f(C_{i,j}) \]
- **Operating Cost** = \[ \sum \sum E_{i,j} \]

above + internode equations = complete set
Objective Functions

\[
\min B_{\text{system}}^{\text{waste}} = \min \sum_{i=1}^{n} (B_i - \sum_{j=1}^{m} \phi_j C_{i,j})
\]

\[
= \min \text{ (initial cost)} - \max \text{ (performance)}
\]

\[
\min E_{i,j} = \min \sum_{i} \sum_{j} [\gamma_{i,j} O_{i,j} + W_i^{\text{idle}} t_{i,j}]
\]

\[
\min \frac{E_{\text{system}}}{C_{\text{system}}}
\]
C. Measurement: sources, types of data

- Simulator
  - RTL level: all details, phase=inst or more, very slow
  - Functional: less detail, faster
  - Numbers very arch-specific, constrains codesign variations

- Math model, e.g. queueing
  - Fast, but localized, and may be architecture constrained

- HW counters
  - Very fast, fixed meas. points, quirky (defs tricky, changing)

- Single-valued virtual nodes (details following)
  - Combine HW/SW at intuitively useful level
  - Flexible, allows various architectural mappings
  - Measure via binary instrumentation, nopping, ubenchmarks
  - Example: mem-mem vector ops: f(stride, length, ... )

Input assumptions → output interpretation
Virtual Node Model Synthesis

Joint work with Intel Exascale Lab, Paris
## Codelet → sv-node decomposition

<table>
<thead>
<tr>
<th>Level</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original program → phases</td>
<td>Irregularities removed, e.g. alignment, aliasing, ...</td>
</tr>
<tr>
<td>Codelet</td>
<td>Significant time, automatically isolatable, similar $\mu$ values, ...</td>
</tr>
<tr>
<td>Macro</td>
<td>Mutually exclusive inst. seq., i.e. satisfy time linearity test</td>
</tr>
<tr>
<td>Single rate v-node</td>
<td>Similar phy. node execution and memory access, so constant execution rate</td>
</tr>
</tbody>
</table>

**Tools list: next foil**
SW Tools for sv-node modeling

• Microbenchmarking – node $B = \max_k \{C_{i,k}\}$
  - Generation tool
• Capacity analysis – $<\text{node, phase}> C$
  - Maqao: static analysis of assembly code
  - Decan: dynamic analysis of binary code
    - Replace selected instructions, run modified binaries
    - Nopping – change, kill, or replace op with nop
    - Destroys semantics, but gets accurate Capacity values
• Intel Exascale Lab – W. Jalby, Versailles
Decan: Magma Codelet Behavior (2)

Processor Saturated

Saxpy2 (MEDIUM) : 12 cores

Memory Saturated
D. 3 example problems, Cape solutions

1. Min cost, max perf codesign problems
   a. Analytical models of critical breaks in codesign space
   b. Cape tool for codesign

2. From system set, choose max perf (or min E/C)
   a. Recommender system for OEM vendor website

3. Codesign energy efficient systems
   a. Offline phase analysis predicts future
     → online (f,V) control governor \[ B^{\text{waste}} \to 0 \]
   b. min E or min E/C solutions

   Treat measured \( \mu \) ratios as constants
Cape codesign inputs/outputs

- **Performance**
  - Minimal thresholds or step ahead: codesign process input
  - Bandwidth used units: input/output

- **Costs**
  - Initial cost = BW needs, operating cost = $E$ etc.: input/output
  - Max limits
  - Variable as function of value to buyers of design

- **Load**
  - Defined using node $C_{i,j}$, $\mu$, and saturations
  - Data sets $\rightarrow$ computation program paths: vary phase weights
  - Stability of design
    - *How sensitive are perf and cost to load-usage uncertainty?*
D.1. Solving cost/perf codesign problems  
(3node X 2phase) example

<table>
<thead>
<tr>
<th>Cost</th>
<th>maxPerf</th>
<th>Perf range</th>
<th>m1</th>
<th>m2</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>1.1110</td>
<td></td>
<td>1.7</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>4.2000</td>
<td>1.1110</td>
<td>1.1100</td>
<td>1.7000</td>
<td>1.2000</td>
<td>1.3000</td>
</tr>
<tr>
<td>4.3000</td>
<td>1.1110</td>
<td>1.1100</td>
<td>1.7000</td>
<td>1.2000</td>
<td>1.4000</td>
</tr>
<tr>
<td>4.3000</td>
<td>1.1110</td>
<td>1.1100</td>
<td>1.6000</td>
<td>1.2000</td>
<td>1.3000</td>
</tr>
<tr>
<td>4.3000</td>
<td>1.1110</td>
<td>1.1100</td>
<td>1.9000</td>
<td>1.1000</td>
<td>1.3000</td>
</tr>
<tr>
<td>4.3000</td>
<td>1.1110</td>
<td>1.1100</td>
<td>2</td>
<td>1.3000</td>
<td>1.3000</td>
</tr>
</tbody>
</table>

Given perf = 1.11, cost = 4.3  
Each B varies by 20%  
Next foil examines this Bm2 = .775 cut
D1. cont. Sensitivity of Performance to the System

Processor perf vs. \( B_{m1} \), showing 3 perf regions; \( B_{m2} = .775 \)

Balance breaks computed analytically using \( \mu_{xy} = \alpha_{xy} \), predicts performance instabilities

Vizualize on surface, understand analytically
Mem BW vs. Perf. Stability?

Max $C_{cpu}$ sensitivity to $B_{mem}$

Mem saturated phases 1, 2

Cpu saturated phases 1, 2

Mem sat phase 2, cpu sat phase 1

3-node, 2-phase balance points

$B_{balance_{cpu,1}}$, $B_{balance_{cpu,2}}$

$B_{balance_{mem,1}}$, $B_{balance_{mem,2}}$

$B_{mem}$

$\Delta \mu$

$\mu_{mem,cpu,1}$, $\mu_{mem,cpu,2}$

$C_{cpu}$

$C_{mem}$

$B_{balance_{mem,i}} = \mu_{cpu,mem,i} B_{cpu} = \frac{B_{cpu}}{\mu_{mem,cpu,i}}$
D.2 Capacity-based Recommender System

- When a user asks about purchasing a new system
  - Current websites give extensive lists, little insight
  - Reco tool recommends top choices
    - Perf ranking among user’s options
  - Explains why chosen, based on current SW apps
    - v-nodes represent user-specific HW/SW combinations

- OEM customer-support program feature
  - Anonymously measuring 8M users continuously
  - Run “capacity model” periodically on user’s system

- System model constrained
  - Apps include all processes running: 1 sec. samples
  - Increasing HW nodes selectable

Apps usage of system?: Users/OEMs don’t understand
D.3. Power and Energy Objectives

- Design-in low power model: $W_{i,\text{idle}}$, $W_{i,\text{max}}$, $\gamma_i$
  - C/E or C/W proportional computing: use W and E efficiency
    - Market needs depend relative loading of systems
- Keep instantaneous power < thermal limit
- Run-time energy control (f,V) scaling, DVFS
  - P and C states: C for various idle W levels, P for (f,V) levels
    - Energy and time consumed making transitions
  - Race-to-Idle: only useful if W model is sufficiently poor
    - Conditions easy to state using the model
    - Multiprogramming complications if all cores scale together
- Preprocess apps for phase-level (f,V) self-scaling
  - OS scheduling interactions, depending on (f,V) resolution
3.a Perf vs. Energy for (2x3) model

3 W-states

max perf $C = 38.9$
$E = .87$

min $E/C$
perf $= 37.5$ (-3.6%)
energy $= .77$ (-11.7)

max perf $C = 38.9$

$E = .87$

$E/C$
perf $= 31.8$ (-18%)
energy $= .66$ (-23%)

Codesign Pareto front $\rightarrow$ Design choices
3.b Network perf results

Packet size $\rightarrow$ system behavior including $E(f, V)$
E. Summary

1. Problem: computer system perf, cost, energy, ...
   - Pre-Si ← post-Si system analysis
   - Architecture, compiler and application codesign
   - Is that benchmark in valid form for this arch. simulation?

2. Approach: Codesign using HW/SW combinations:
   - SW load in HW design
   - HW characteristics in SW design

3. Break problem into convenient pieces
   - HW into nodes (virtual-node flexibility)
   - SW into phases (codelets are optimal forms)

4. Tools to automate the process
   - Aids to human decision makers

Is it really necessary to defer understanding computer behavior to post-Si experimentation – and incur perpetual surprise?
Key benefits of capacity-based codesign

1. Top-down codesign of optimal systems
   - Include system-wide interactions
   - Mixed fidelity saves modeling effort and simulation time

2. Simultaneous use of all “known” load/BW info
   - Overcome human-limiting complexity via automatic process
   - Capture parameter uncertainties via sensitivity analysis

3. Design focused-system families
   - Specialized system-per-market always beats general systems
   - Family codesign softens combinatorial explosion

Fast optimization (coherent data) → Codesign results
References
