Modeling Computation for HW/SW Codesign
PACT 2013 Keynote

David J. Kuck
Software & Services Group

Edinburgh
Sept. 9, 2013

Copyright © 2013, Intel Corporation. All rights reserved.
Outline

A. Introduction
B. Basic Equations
C. System Codesign Models
D. Measurement
E. Cape Modeling Results
F. Conclusions
Where does HW/SW design stand

- High-tech, successful, well-advanced field
  - Arguably a major technical/business success
- Unlike similar fields: no system equations
  - Formal models: devices, circuits, algorithms, num. anal., ...
  - Intuition: System design, SW design, cost/performance, ...
- Much work on “getting the right balance”
  - What is “right balance,” how is it defined?
  - How do we get there from here?
  - There is no foundation of global procedures.
- There are several ways to study these gaps
  - How to use computers better to design computer systems?

Goal: HW/SW Codesign Methodology that handles all of this
3 Steps from Basic Codesign Theory

• Start with a basic codesign theory
  - See codesign section; then →

1. Build a *model*+*optimization* tool: solve toy problems
  - *Done and validated* on minimal real data
  - Fast simulation or constrained optimal system design

2. *Measure* real machines
  - SW/HW measurement of many nodes *in process*
    - Vector access and processing
    - Memory hierarchy levels
    - Parallelism, throughput, clock freq, power, energy

3. *Model* new HW/arch ideas
  - Optimize BW of all nodes for perf/energy
  - Vector, parallel tradeoffs
  - BW and size of cache needed? (e.g. L2=0 ?)
Modeling and Measurement Steps

• Choose HW nodes to match design needs
• Choose SW phases with steady-state behavior
• Recognize individually, understand joint behavior

• The above enables measurement and modeling
  – Physical system reconstruction from modeling results
• Captures details, supports optimization – Section C.

• Measurement is a challenge – Section D.
  – HW counters are weak, SW methods have been weaker
B. Basic Equations
B.1. Modeling: 4d Codesign Space  
(w.o. Oper. Cost E)

1. Performance

Best Performance

B. Performance-enhanced System

A. Original System

D. Market-focused Systems

Enhanced Perf/Cost plane

C. Application-enhanced System

2. System Cost

3. SW Load

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/bit** $= \gamma_i \left[ \frac{w * s}{b} \right]$.

Node 1

\[ \text{node 1} \]

\[ \text{node 2} \]

\[ \text{node } i \]

\[ \text{node 3} \]

Multiplexed ports, 0 latency, $\infty$ BW

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

Phase 1

Phase 2

\[ \text{phase 1} \]

\[ \text{phase 2} \]

\[ \text{phase } j \]

\[ \text{prog + data} \]

b. SW Computation

**Physical model characteristics**

Copyright © 2010, Intel Corporation. All rights reserved
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 $i$):
  
  $$
  \begin{cases}
  C_{i,j} = O_{i,j} / t_{i,j} & \text{if } i \text{ is saturated} \\
  C_{i,j} = \overline{B}_i & \text{if } i \text{ is saturated}
  \end{cases}
  $$

  $$0 \leq \sigma^C_i = C_i / B_i \leq 1$$

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

Each phase saturates one or more nodes
Computational codesign variable relations

- **Independent variables of computation**
  - Time \( t \), HW speed \( (f,V) \) abb. \( (f) \), SW ops \( O(D) \) 

- **Basic dependent variables**
  - HW: BW \( B(f) \), power \( W^{\text{max}}(f) \), \( W^{\text{idle}}(f) \), \( W^{\text{dyn}}(f) \) 
  - SW/HW: Computational capacity \( C \leq B \) 

- **Derived dependent variables** (mostly HW/SW dependent)
  - HW: Energy/bit = \( \gamma_i \left[ \frac{w^* s}{b} \right] \)
  - Energy: \( E \)
  - Relative C saturation: \( 0 \leq \sigma^C_i = C_i / B_i \leq 1 \)
  - Intensity: \( C_{i,j} = \mu_{ki,j} C_{k,j} \)
  - Quality: \( E/C \)

Want linear models, but deal with nonlinearity
Balanced nodes, phase balance point

\[ \mu_{x,y,k} = \alpha_{x,y} \]

Nodes balanced, unsaturated

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

Saturated intensity equation

Phase balance point, saturated node

Perfect BW for this computation

Initial BW
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

\( C_i = \text{clock freq.} \)

\( C_{i,j}^{\text{low}} = f_{i,j}^{\text{low}} \)
\( C_{i,j}^{\text{high}} = f_{i,j}^{\text{high}} = B_i^{\text{max}} \)

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

**Power Equation**

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

Model allows arbitrary slopes

Combine with architecture codesign model

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

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

4 Local Eqs.
node \( i \), phase \( j \)

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

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

3 Global Eqs.

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

above + internode equations = complete set
C. System Codesign Models
B.2. Codesign Examples:

1-phase, 3-node system

For node $i$, phase $j$:

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

- **Capacity Equation** (node $i$):
  \[
  C_{i,j} = \begin{cases} 
  O_{i,j} / t_{i,j} & \text{if } i \text{ saturated} \\
  \bar{B}_i & \text{otherwise}
  \end{cases}
  \]

- **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 (program and data)
Ex. 1: Systematically boost syst. perf.?

If \( C_{\text{disk},j} = \overline{B}_{\text{disk}} \), increase disk performance:

- Set saturated node BW to desired performance level \( \rightarrow \overline{B}_{\text{disk}} \)
- Adjust other nodes accordingly \( \rightarrow \) use \( C_{\text{cpu},j} = \mu_{\text{disk-cpu},j} \cdot C_{\text{disk},j} \)

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

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

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

- Replicate node – parallel processors or memories add BW
  - $B_{ik}^{\text{parallel}} = B_i + B_k$

Parallel capacities lead to multirate nodes, $C_{\text{parallel}, \text{cpu}, j} \leq 2C_{\text{cpu}, j}$
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

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

Variable latencies lead to nonlinear multirate nodes
Summary Internode Equations

Local Eqs.

**Capacity Intensity:** \[ C_{i,j} = \mu_{k_i,j} \overline{C}_{k,j} \rightarrow \mu_{k_i,j} \overline{B}_k \]

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

**Serial supernode** \[ \frac{1}{C_{ij}^{\text{sernon-ov}}} = \frac{1}{C_i} + \frac{1}{C_j}; \quad C_{ij}^{\text{serov}} = \min\{C_i, C_j\} \]

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

System of (nodes X phases) inequalities ➔ Global Codesign

Copyright © 2010, Intel Corporation. All rights reserved
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 (initial cost) – max (performance)

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

\[
\min \frac{E_{\text{system}}}{C_{\text{system}}}
\]

Codesign goals \(\Rightarrow\) Capture Complexity: may be nonlinear
D. 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: \( f(\text{stride}, \text{length}, \ldots) \)

Input assumptions \( \rightarrow \) output interpretation
D. Measurement:
Decan SW + HW counters

Joint work with Intel Exascale Lab, Paris
## Codelet $\rightarrow$ sv-node decomposition

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

Memory Saturated

Saxpy2 (MEDIUM) : 12 cores

Time

Frequency

intel
E. Cape Modeling Results

1. Recommender
   - Select best system from list of designs

2. Simulation
   - Explore design space

3. Optimal codesign
   - Best C/E for SW workload and HW constraints

Cape implemented by David Wong, Intel
E1. System Behavior – Quality Objectives

- **D-Stability** (unicore, *perf range*, *Data range*)
- **f-Scalability** (*freq*, *D*)  
  [D-range vs. *f*]
- **p-Speedup** (*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, to maintain contracts with ISVs, OEMs, & end users**
E.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
Simulation Example: 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

Saturation crossover

L3

time/iteration

Perf range

Data size

$\mu(D)$

4 c/lo f

40K

400K

4 core/high freq

Hqr_13_dp_sse
Codelet 2: Svdcmp_11_dp_sse

1c/lf

All load/store saturated

Floating point time

t=35

t=80

1c/hf

4c/lf

Perf range

4c/hf
## 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</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. Note that HW and/or SW changes are suggested by Q violations.**
5 phase (mixed lsi & fpi sat) – 35 node simulation

CAPE input model

CAPE output

flops vs. BL2 vs. Bv div

Cape design parameters
Example: 2X flops, .7 BW L2, 5X arith & L1

Problem

- 5 codelet workload: 3 proc bound, 2 mem bound
- Attempt to cut BW L2 while doubling perf

<table>
<thead>
<tr>
<th>BW</th>
<th>Sc+</th>
<th>Sc*</th>
<th>Sc/</th>
<th>v/</th>
<th>L s</th>
<th>Lsse</th>
<th>S s</th>
<th>Ssse</th>
<th>L2</th>
<th>L3</th>
<th>Agg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Orig</td>
<td>.06</td>
<td>.06</td>
<td>.04</td>
<td>.04</td>
<td>.11</td>
<td>.44</td>
<td>.06</td>
<td>.44</td>
<td>10.2</td>
<td>6.0</td>
<td>.32</td>
</tr>
<tr>
<td>Design</td>
<td>.26</td>
<td>.19</td>
<td>.16</td>
<td>.16</td>
<td>.34</td>
<td>2.3</td>
<td>.26</td>
<td>2.86</td>
<td>7.2</td>
<td>11.1</td>
<td>.63</td>
</tr>
<tr>
<td>Ratio</td>
<td>4x</td>
<td>1x</td>
<td>1x</td>
<td>1x</td>
<td>3x</td>
<td>5x</td>
<td>4x</td>
<td>6x</td>
<td>7x</td>
<td>2x</td>
<td>2x</td>
</tr>
</tbody>
</table>

Preliminary output: node units not normalized

Cape Result (no optimization, 5 min manual search)

- L2 BW cut to .7 original (energy savings) [area vs. size]
- Main cost is 5X arithmetic speed, and register/ L1 access
- Optimization can exploit all such opportunities

This is a typical consequence of fpi vs. lsi interaction; syst perf vs. node BWs
Surface irregularities are hard to predict intuitively.
3. Cape optimal 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?*

Whatever is not Input, tool chooses as Output
E. 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 stable 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
      \[ \rightarrow \text{online} (f,V) \text{ control governor} \text{ [ } B_{\text{waste}} \rightarrow 0 \text{ ]} \]
   b. min E or min E/C solutions

Treat measured \( \mu \) ratios as constants
D.1. Solving cost/perf codesign problems
(3node X 2phase) example

Data Arrangement
Data Group by

<table>
<thead>
<tr>
<th>Cost</th>
<th>maxPerf(p)</th>
<th>Perf range(p)</th>
<th>m1</th>
<th>m2</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>1.1110</td>
<td>0</td>
<td>1.7</td>
<td>1.2</td>
<td>1.2</td>
</tr>
<tr>
<td>4.200</td>
<td>1.1110</td>
<td>0</td>
<td>1.7000</td>
<td>1.2000</td>
<td>1.3000</td>
</tr>
<tr>
<td>4.200</td>
<td>1.1110</td>
<td>0</td>
<td>1.9000</td>
<td>1.1000</td>
<td>1.2000</td>
</tr>
<tr>
<td>4.300</td>
<td>1.1110</td>
<td>0</td>
<td>1.7000</td>
<td>1.2000</td>
<td>1.4000</td>
</tr>
<tr>
<td>4.300</td>
<td>1.1110</td>
<td>0</td>
<td>1.8000</td>
<td>1.2000</td>
<td>1.3000</td>
</tr>
<tr>
<td>4.300</td>
<td>1.1110</td>
<td>0</td>
<td>1.9000</td>
<td>1.1000</td>
<td>1.3000</td>
</tr>
<tr>
<td>4.300</td>
<td>1.1110</td>
<td>0</td>
<td>2</td>
<td>1.0</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

Bm1, phase 2 saturation

Bm2, phase 2 saturation

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

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

$B_{balance_{cpu,1}}$ \(\rightarrow\) $\mu_{mem,cpu,1}$

$B_{balance_{cpu,2}}$ \(\rightarrow\) $\mu_{mem,cpu,2}$

$B_{balance_{mem,1}}$ \(\rightarrow\) $\mu_{mem,cpu,1}$

$B_{balance_{mem,2}}$ \(\rightarrow\) $\mu_{mem,cpu,2}$

$\Delta \mu$

Stable $C_{cpu}$ range if $C_{mem,2}$ rises, i.e. $\mu_{mem,cpu,2}$ drops

cpu saturated phases 1, 2

mem sat phase 2, cpu sat phase 1

$B_{mem}$ \(\rightarrow\) $C_{mem}$

$B_{balance_{mem,i}} = \mu_{cpu,mem,i}$

$B_{cpu} = B_{cpu} / \mu_{mem,cpu,i}$

3-node, 2-phase balance points
D.3. Power and Energy Objectives

• Design-in low power model: $W_i^{idle}, W_i^{max}, \gamma_i$
  – 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)

perf = 31.8 (-18%)
Energy = .66 (-23%)

Codesign Pareto front (black dots) → Design choices
3.b Network perf results

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

512B: Network saturated at all proc $f$

64B: Proc saturated, perf increases with $f$
Energy efficiency vs. Capacity Rel. Saturation

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

- **Battery laptop**

- **Network server**

- **HPC or gamer**

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

**Diagram:**

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 numerator and denominator

Server consolidation problem

"Server consolidation" problem
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
   - Cluster usages partition market by HW needs
   - Specialized system-per-market always beats general systems
   - System-family codesign softens combinatorial explosion

Fast optimization (coherent data) → Codesign results
References
