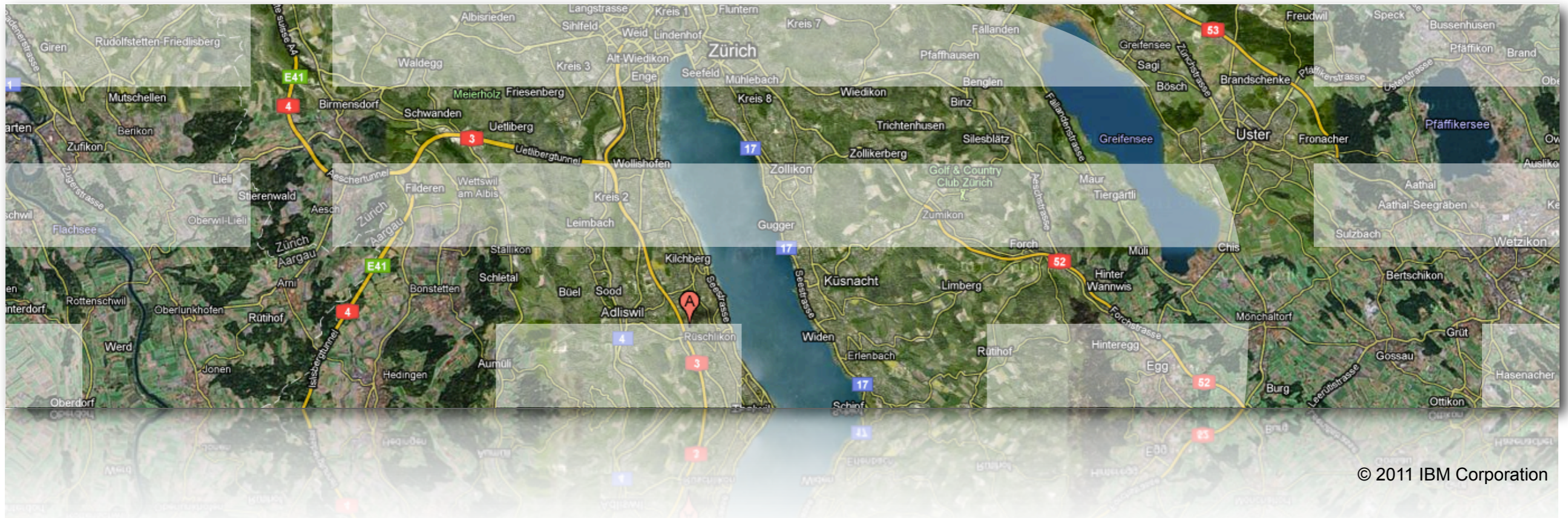


Performance and Energy-Efficiency: Device Technology, Parallelism, Algorithms, and Beyond

Phillip **Stanley-Marbell**

Joint work with Victoria **Caparrós Cabezas**, Rik **Jongorius**, and Ronald **Luijten**

IBM Research—Zurich



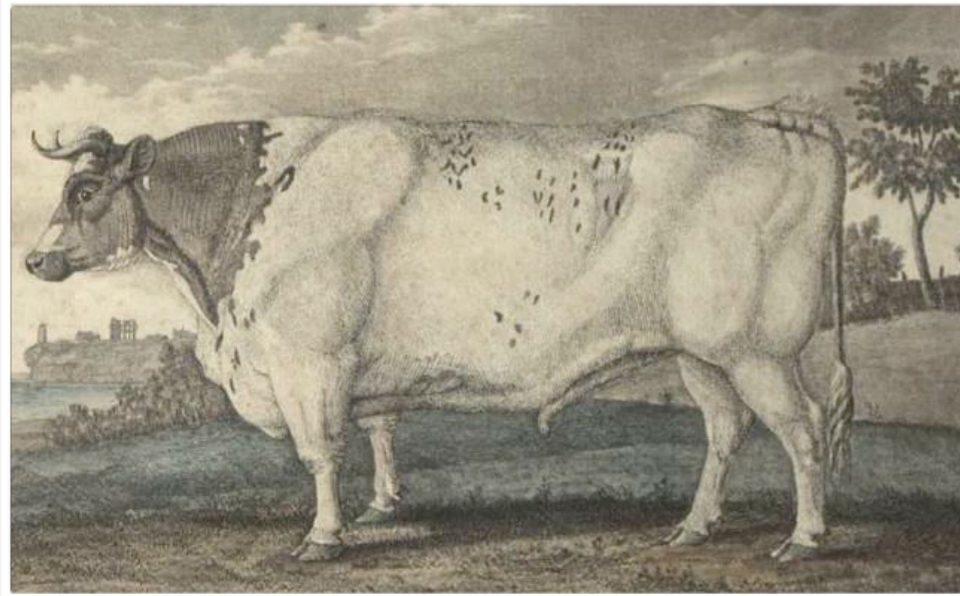
Vehicles of Performance Growth



vehicle |'vēəkəl; 'vē:hikəl|
noun
1 a thing used to express, embody, or fulfill something

- **Enablers of performance growth** over the last 5 decades
 - Complex instructions (CISC): **more co-dependent work per cycle**
 - Simple instructions / hardware (RISC): ride technology scaling curve, **faster cycles**
 - Parallelism (superscalar, multithread, multicore): **more independent work per cycle**
- **Limits to performance vehicles**
 - CISC: uncommon denominators—**complicated compilation**
 - RISC: no longer feasible to run at faster cycles—**power-limited scaling**
 - Parallelism: **available parallelism** at instruction, thread, task, and data levels

Device Technology, Parallelism, and Beyond



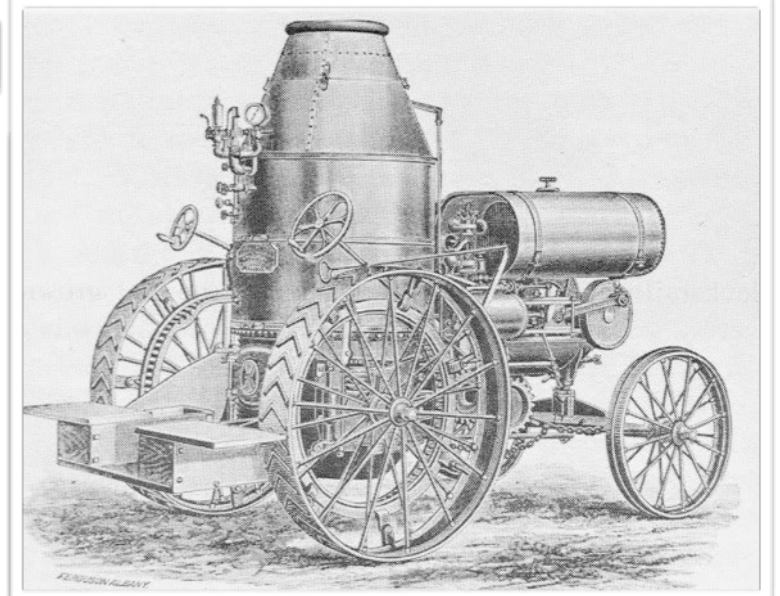
1 ox: single-thread performance



...



1024 chickens: parallelism

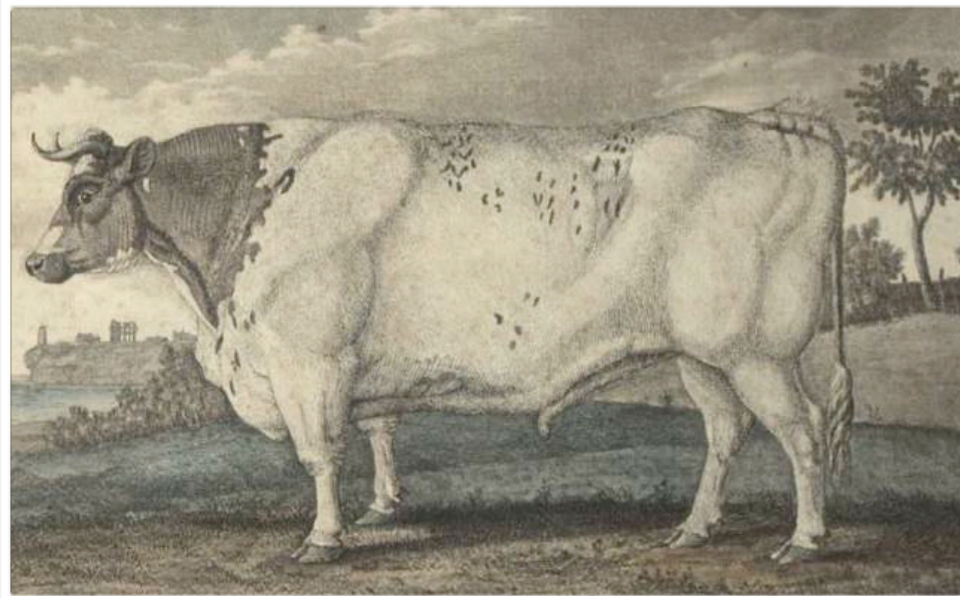


1 tractor: better algorithms

If you were plowing a field, which would you rather use? Two strong oxen or 1024 chickens?

— Seymour Cray

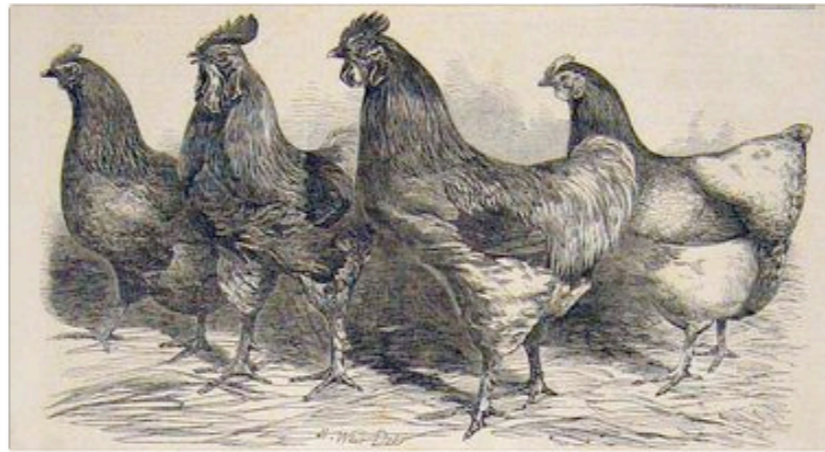
Device Technology, Parallelism, and Beyond



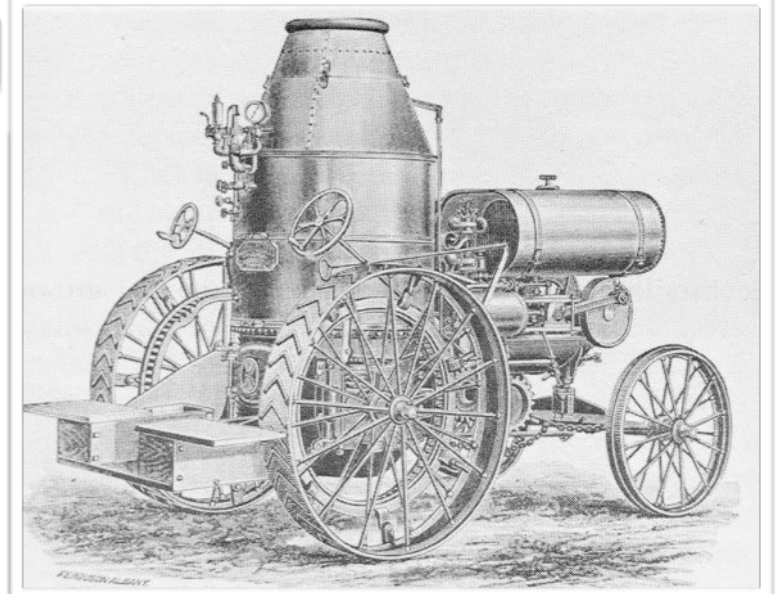
1 ox: single-thread performance



...



1024 chickens: parallelism



1 tractor: better algorithms

● This talk

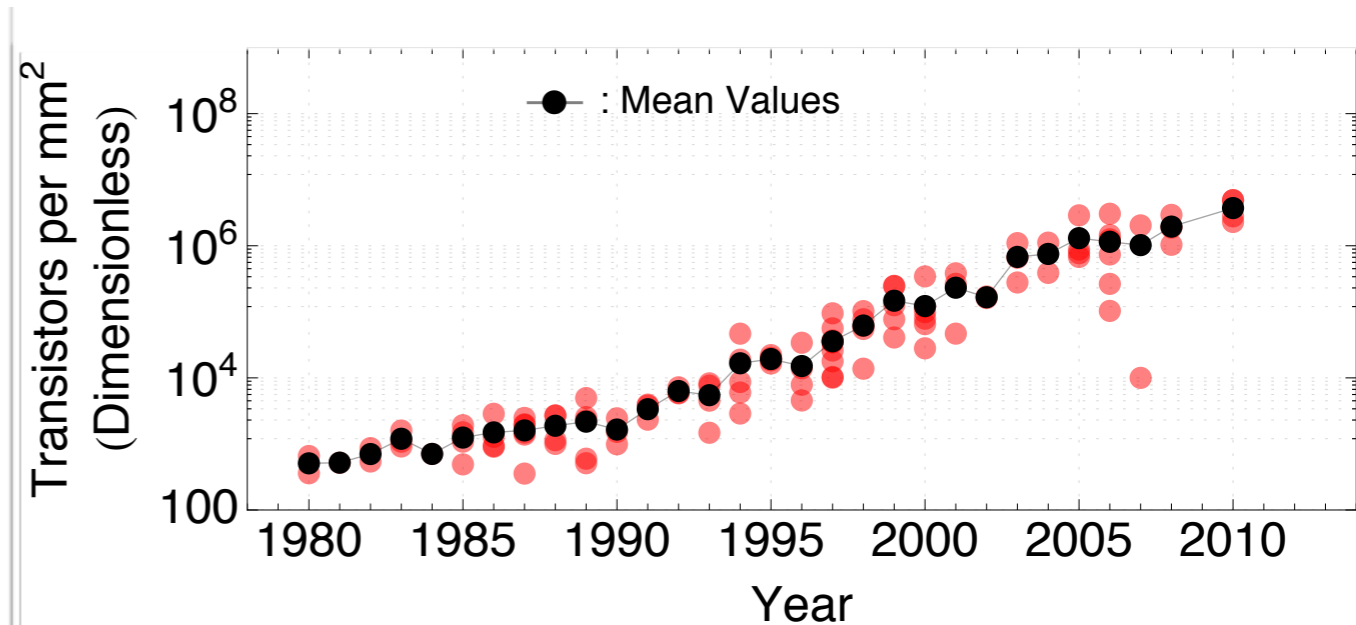
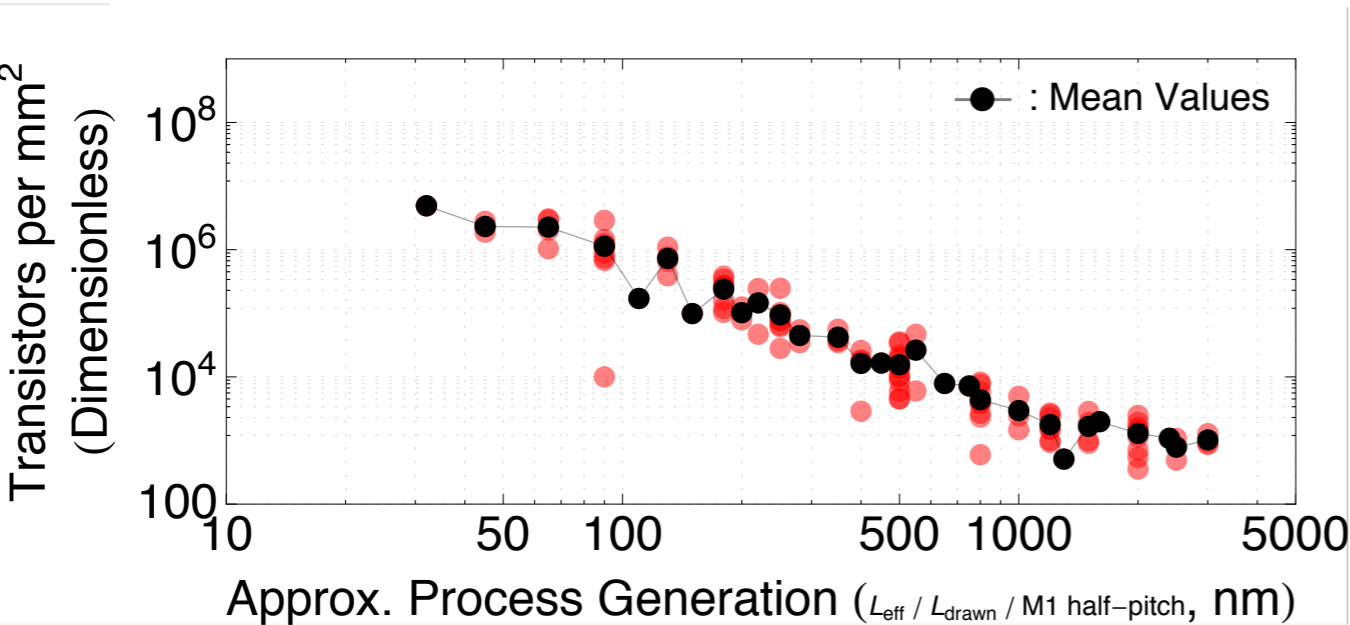
- Device technology trends, spanning the last 30 years, for >150 real designs
- Tools for characterizing parallelism
- Challenges to scaling via parallelism
- Picking the right point in the low-power versus high-performance spectrum
- Opportunities and challenges for the next “performance vehicle”: algorithms

Outline

- Context
- Technology Trends
- Algorithms and Parallelism
- Low-Power versus High-Performance Tradeoff
- Concurrency Limits and Performance Beyond Parallelism
- Summary

Technology, Supply Voltage, and Power Delivery Limits^{1 of 2}

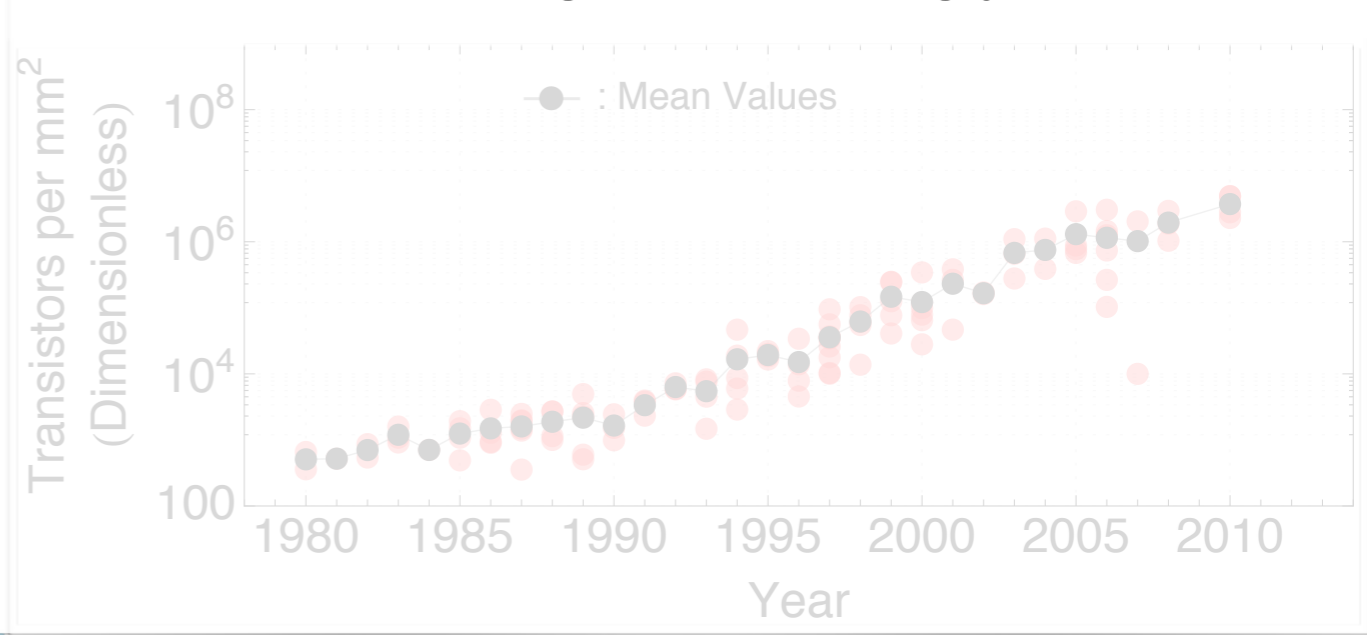
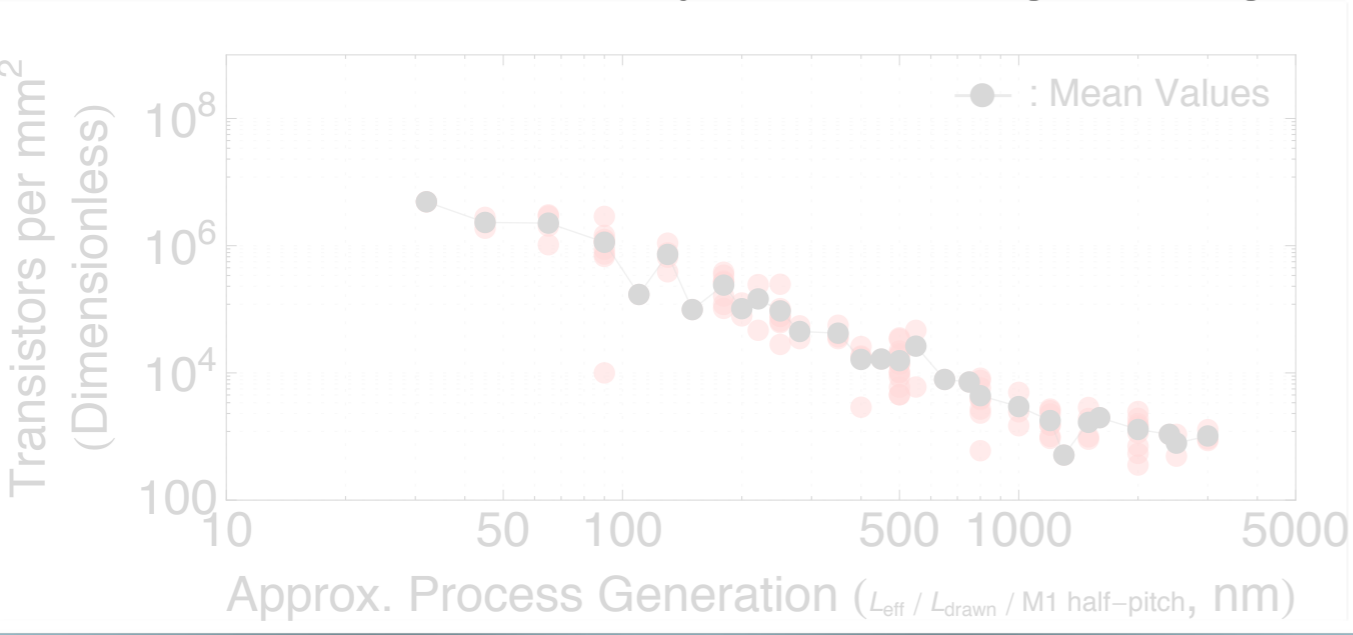
- Density scaling continues (transistors per unit area)



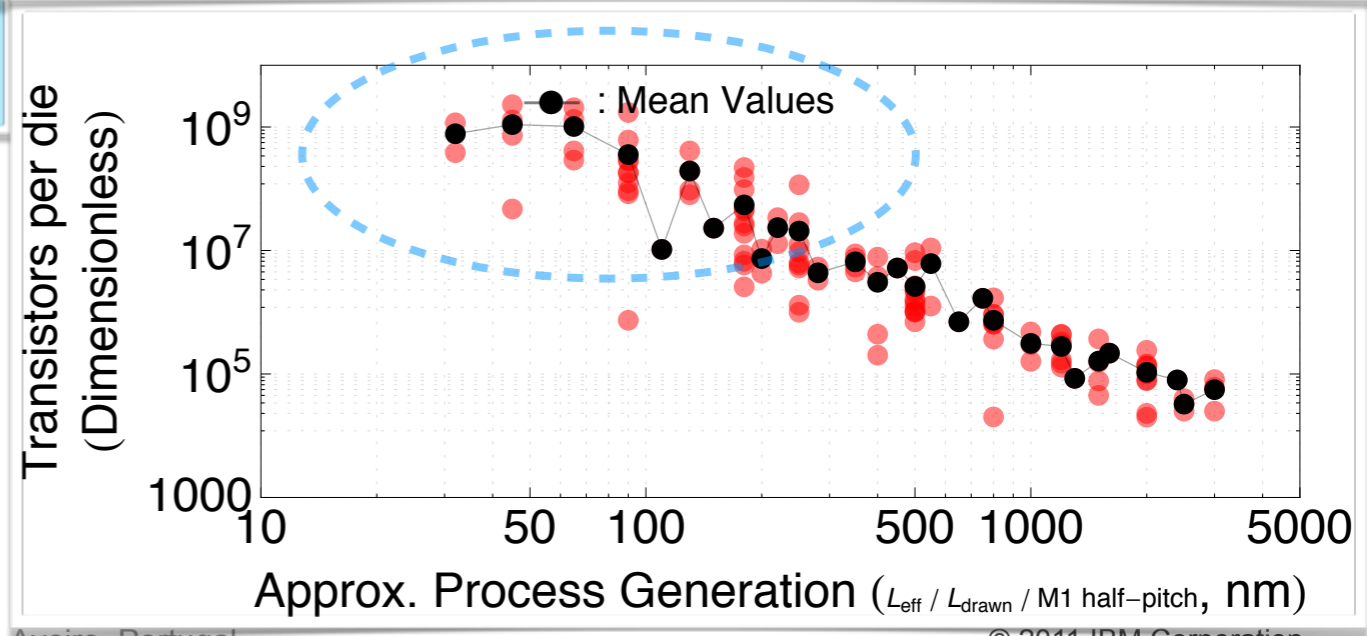
Data: from an analysis of ~150 HW publications in IEEE ISSCC and IEEE JSSC Journal, from 1980–2010

Technology, Supply Voltage, and Power Delivery Limits^{1 of 2}

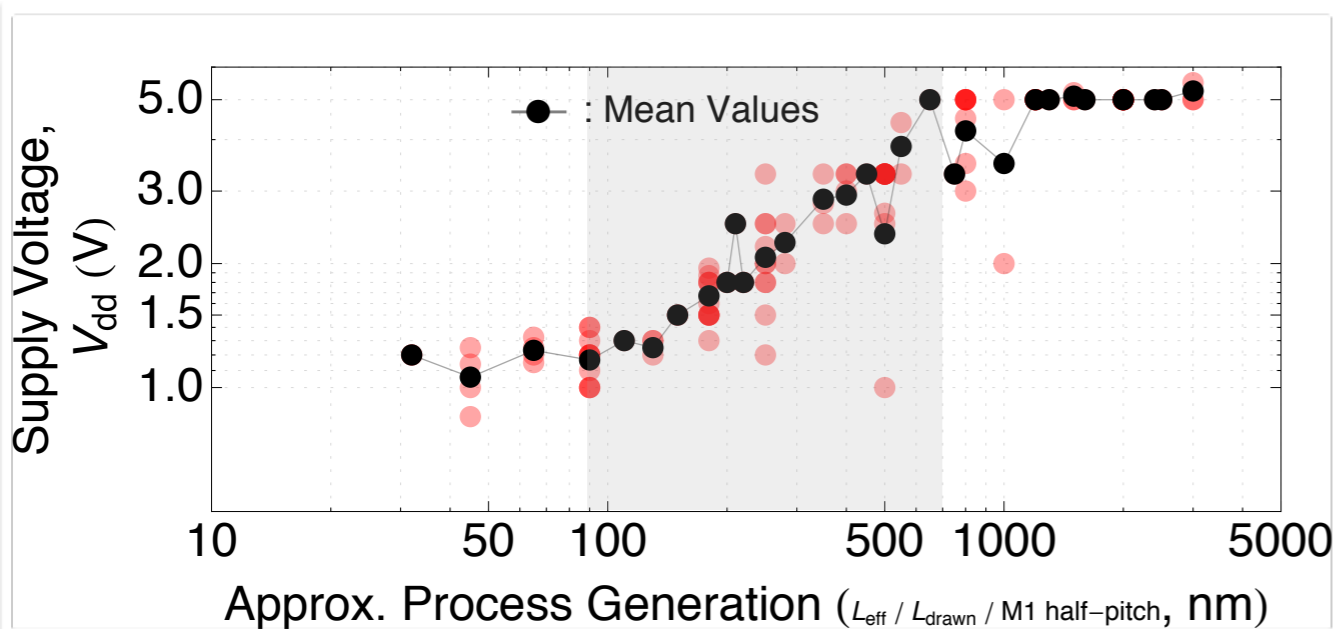
- **Density scaling continues** (transistors per unit area)
 - However, **power per unit area is not scaling**
 - Power delivery and cooling for large-transistor-count designs increasingly difficult



Data: from an analysis of ~150 HW publications in IEEE ISSCC and IEEE JSSC Journal, from 1980–2010



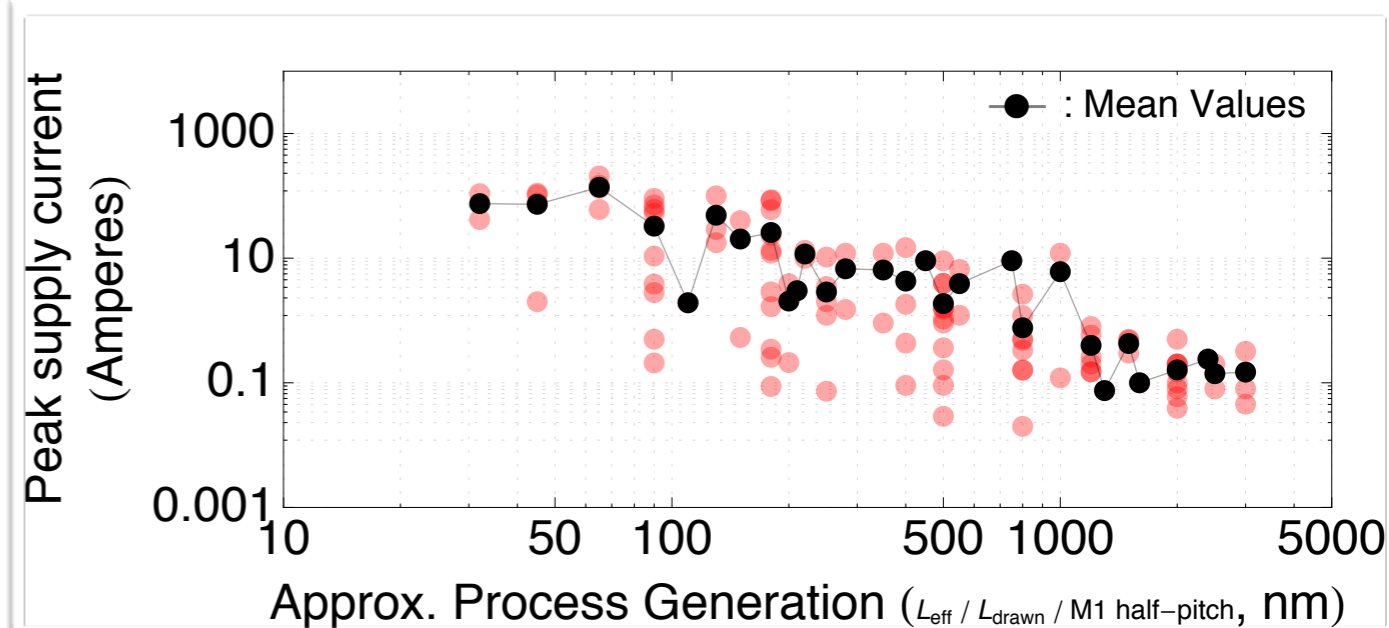
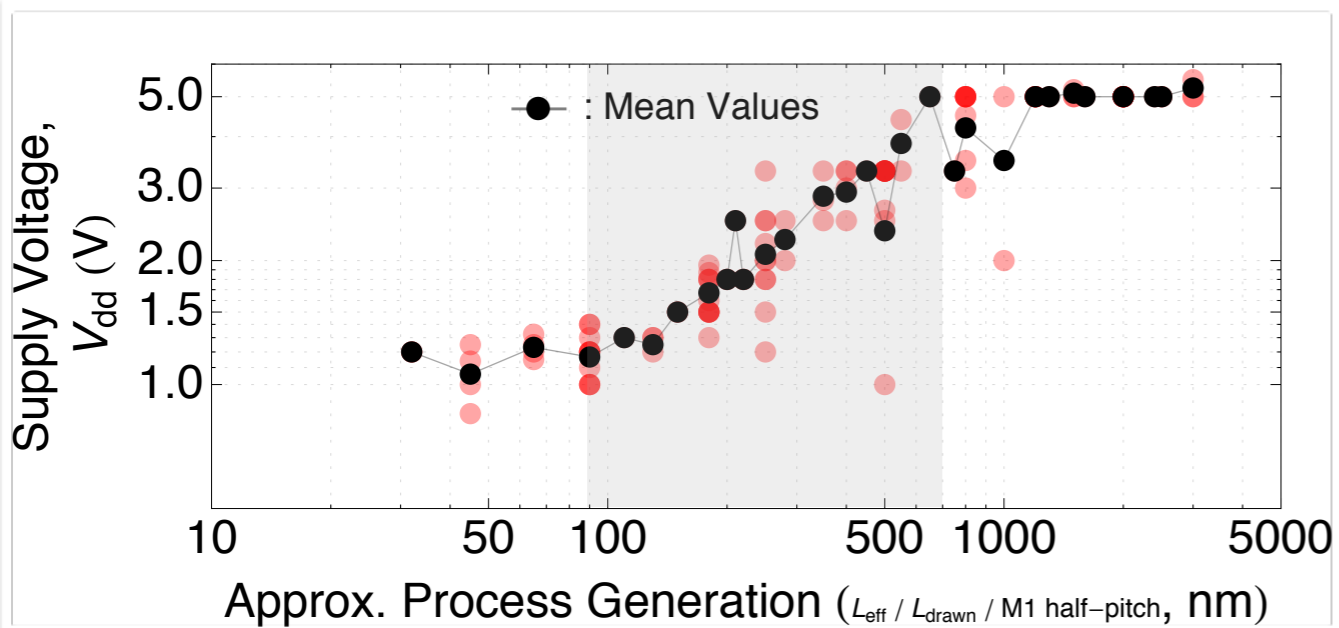
Technology, Supply Voltage, and Power Delivery Limits^{2 of 2}



Data: from an analysis of ~150 HW publications in IEEE ISSCC and IEEE JSSC Journal, from 1980–2010

- Power per unit area not scaling because **supply voltages stagnated**
 - Due to concerns with **sub-threshold leakage current**

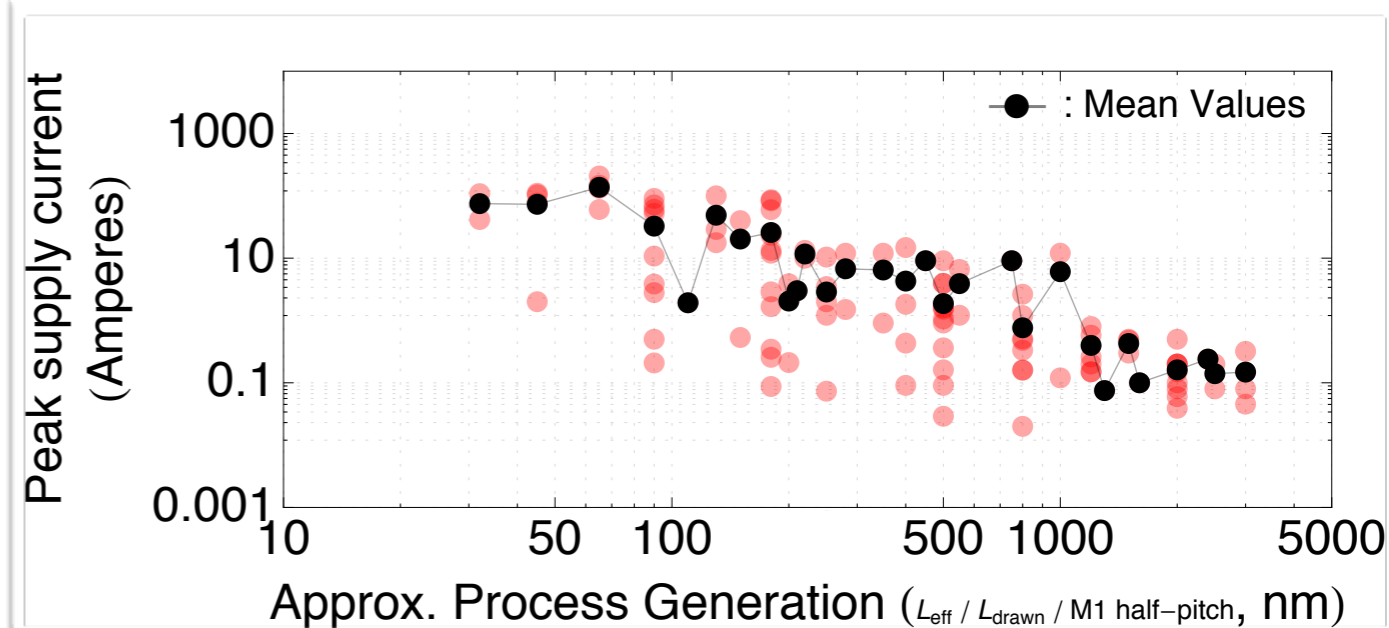
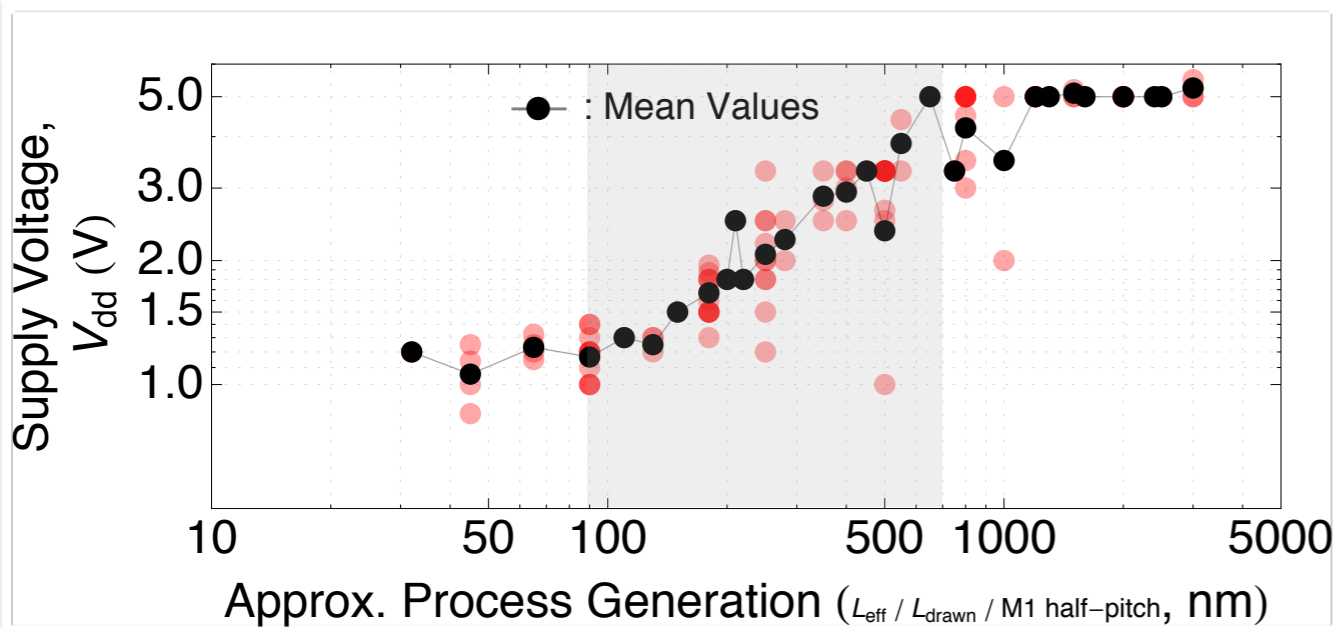
Technology, Supply Voltage, and Power Delivery Limits^{2 of 2}



Data: from an analysis of ~150 HW publications in IEEE ISSCC and IEEE JSSC Journal, from 1980–2010

- Power per unit area not scaling because **supply voltages stagnated**
 - Due to concerns with **sub-threshold leakage current**
 - Lower supply voltages also lead to **higher supply currents at constant power**

Technology, Supply Voltage, and Power Delivery Limits^{2 of 2}



Data: from an analysis of ~150 HW publications in IEEE ISSCC and IEEE JSSC Journal, from 1980–2010

- Power per unit area not scaling because **supply voltages stagnated**
 - Due to concerns with **sub-threshold leakage current**
 - Lower supply voltages also lead to **higher supply currents at constant power**
- Power and cooling challenges **limit clock speeds, # active transistors**
 - Industry-wide shift to increased **performance via parallelism and heterogeneity**
 - Parallelism can be harnessed **at different granularities...**

Outline

- Context
- Technology Trends
- **Algorithms and Parallelism**
- Low-Power versus High-Performance Tradeoff
- Concurrency Limits and Performance Beyond Parallelism
- Summary

Challenge: Systems and Architectures Tailored to Workloads

- Mapping application parallelism to hardware concurrency
 - Appropriate form and amount of hardware concurrency depends on applications
- Hardware properties, e.g., microarchitectures
 - Power7 : 8 cores × 4-way SMT × 12-wide superscalar (8-issue, but 12 FUs)
 - ARM Cortex A8 : 1 core × 1 thread × 2-wide superscalar
 - Power A2 cores : 1 core × 4-way SMT × 2-wide superscalar
- Granularities of parallelism in software
 - Available data-, task-, thread- (TLP) and instruction-level parallelism (ILP)

Application Parallelism Characterization Framework

Stage 1

Stage 2

Input program binary

```

...
addu  r3,r2,r6
sw    r0,0(xx)
addu  r2,r2,r5
addiu r4,r4,1
sw    r0,0(r3)
...

```

+

Input dataset

```

01101000011001
01011011000110
110001101111

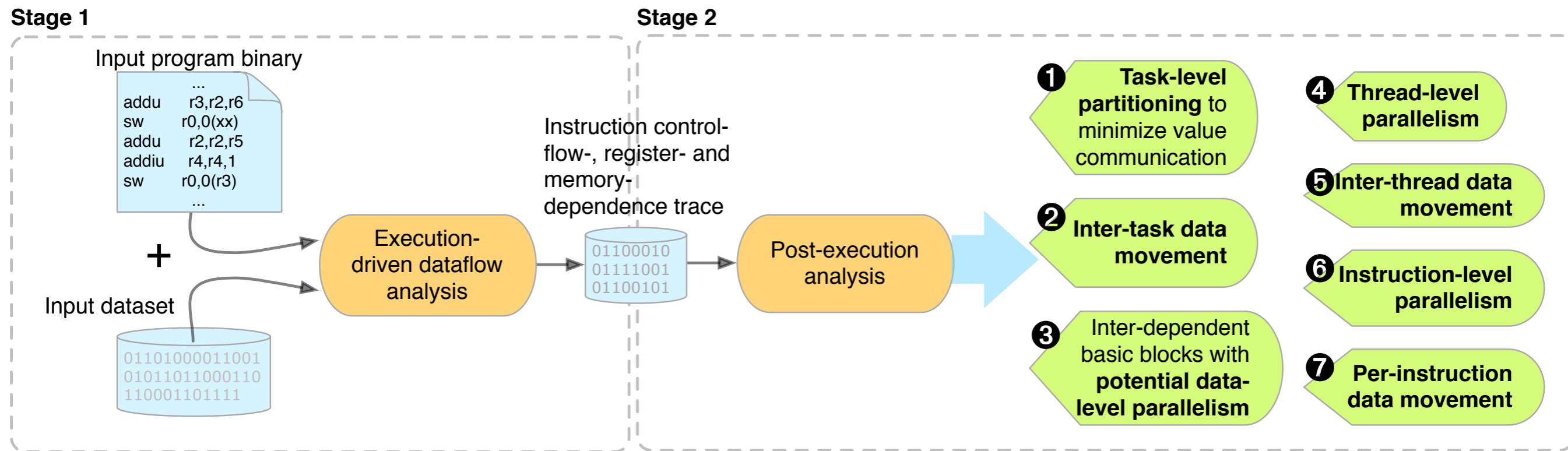
```

Execution-
driven dataflow
analysis

Input

- Compiled program binaries and their associated input datasets

Application Parallelism Characterization Framework



Input

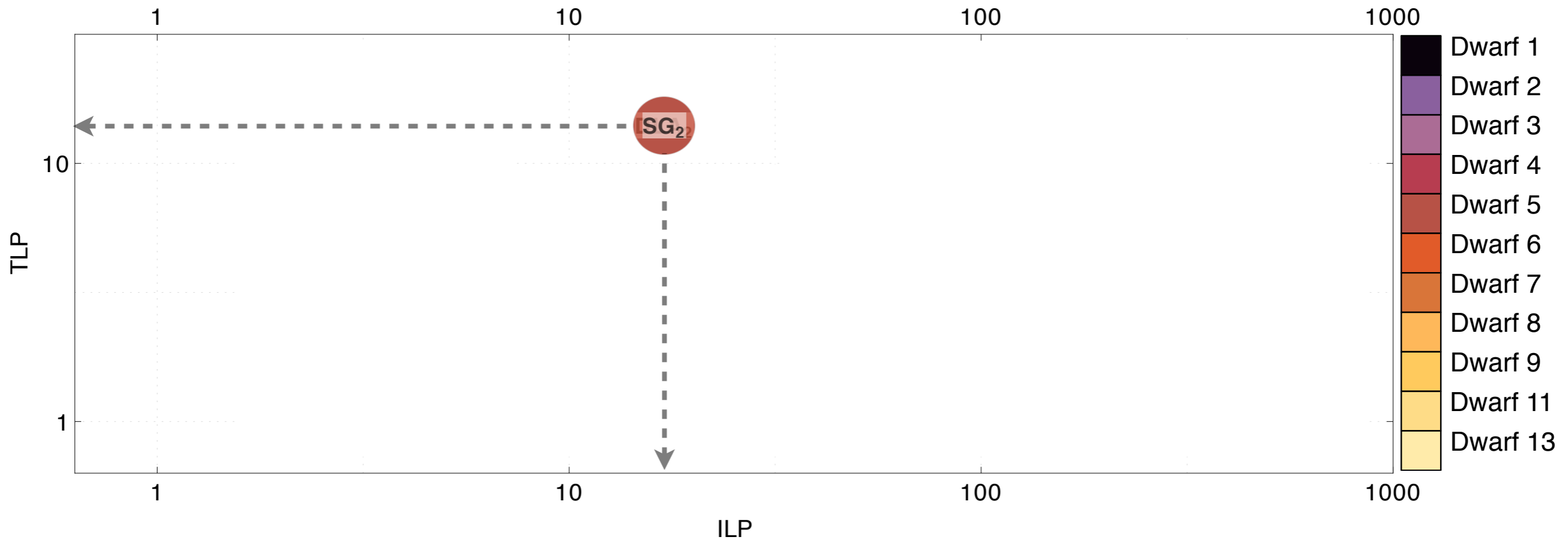
- Compiled program binaries and their associated input datasets

Output

- Best-case **instruction-level parallelism** (ILP) and **thread-level parallelism** (TLP), and **potential data-level parallelism**
- Communication-minimizing **MPMD code partitioning**
- **Data movement** characterization

V. Caparrós Cabezas and P. Stanley-Marbell, “**Parallelism and Data Movement Characterization for Contemporary Application Classes**”, *23rd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, June 2011

Characterization of Applications Spanning Berkeley “Dwarfs”

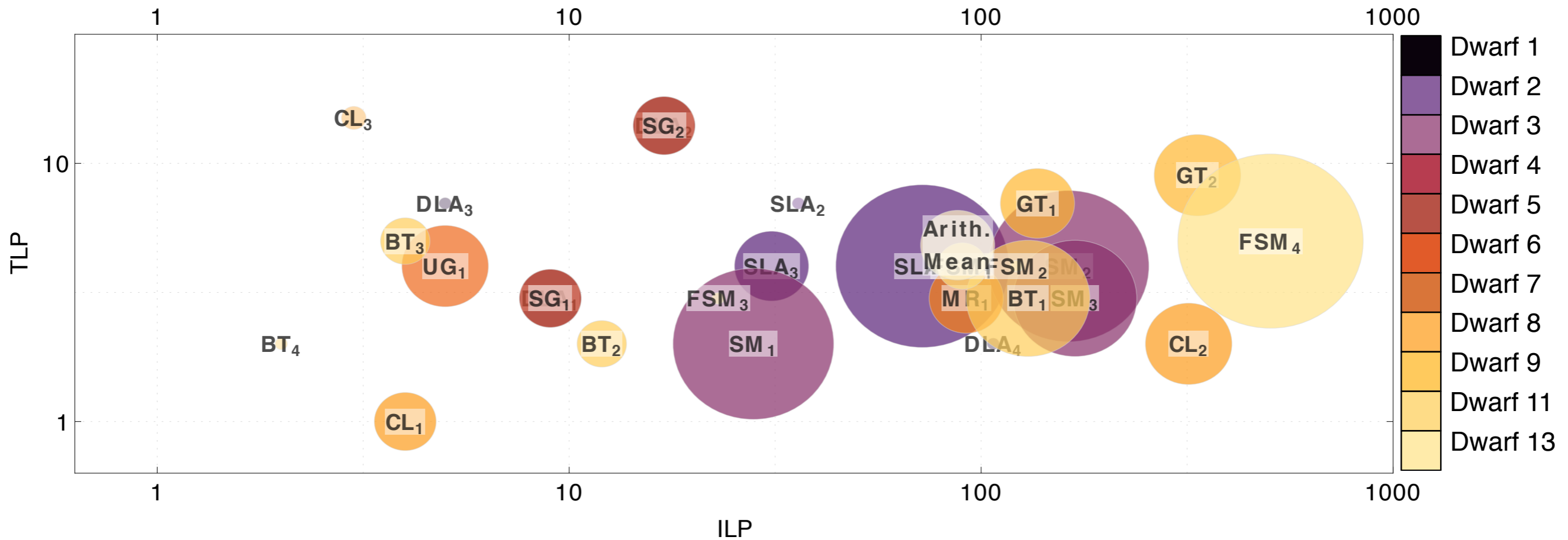


Dwarf 1: Dense Linear Algebra (DLA)	JPEG(encode,decode) ^b , kmeans ^c ,pca ^c
Dwarf 2: Sparse Linear Algebra (SLA)	basicmath ^b , bitcount ^b , matrix_multiply ^c
Dwarf 3: Spectral Methods (SM)	lame ^b , FFT ^b , IFFT ^b
Dwarf 4: N-Body Methods (NB)	ammp ^a
Dwarf 5: Structured Grids (SG)	JPEG ^b (encode, decode)
Dwarf 6: Unstructured Grids (UG)	quake ^a
Dwarf 7: MapReduce (MR)	mesa ^a
Dwarf 8: Combinational Logic (CL)	rijndael ^b (encode, decode), sha ^b
Dwarf 9: Graph Traversal (GT)	qsort ^b , Dijkstra ^b
Dwarf 10: Dynamic Programming (DP)	None
Dwarf 11: Backtrack / Branch+Bound (BT)	vpr ^a , mcf ^a , art ^a , susan ^b
Dwarf 12: Graphical Models (GM)	None
Dwarf 13: Finite State Machine (FSM)	gzip ^a , gcc ^a , parser ^a , stringsearch ^b

26 Applications

^a SPEC 2000
^b MiBench
^c Phoenix MapReduce Suite

Characterization of Applications Spanning Berkeley “Dwarfs”



Dwarf 1: Dense Linear Algebra (DLA)	JPEG(encode,decode) ^b , kmeans ^c , pca ^c
Dwarf 2: Sparse Linear Algebra (SLA)	basicmath ^b , bitcount ^b , matrix_multiply ^c
Dwarf 3: Spectral Methods (SM)	lame ^b , FFT ^b , IFFT ^b
Dwarf 4: N-Body Methods (NB)	ammp ^a
Dwarf 5: Structured Grids (SG)	JPEG ^b (encode, decode)
Dwarf 6: Unstructured Grids (UG)	equake ^a
Dwarf 7: MapReduce (MR)	mesa ^a
Dwarf 8: Combinational Logic (CL)	rijndael ^b (encode, decode), sha ^b
Dwarf 9: Graph Traversal (GT)	qsort ^b , Dijkstra ^b
Dwarf 10: Dynamic Programming (DP)	None
Dwarf 11: Backtrack / Branch+Bound (BT)	vpr ^a , mcf ^a , art ^a , susan ^b
Dwarf 12: Graphical Models (GM)	None
Dwarf 13: Finite State Machine (FSM)	gzip ^a , gcc ^a , parser ^a , stringsearch ^b

26 Applications

^a SPEC 2000

^b MiBench

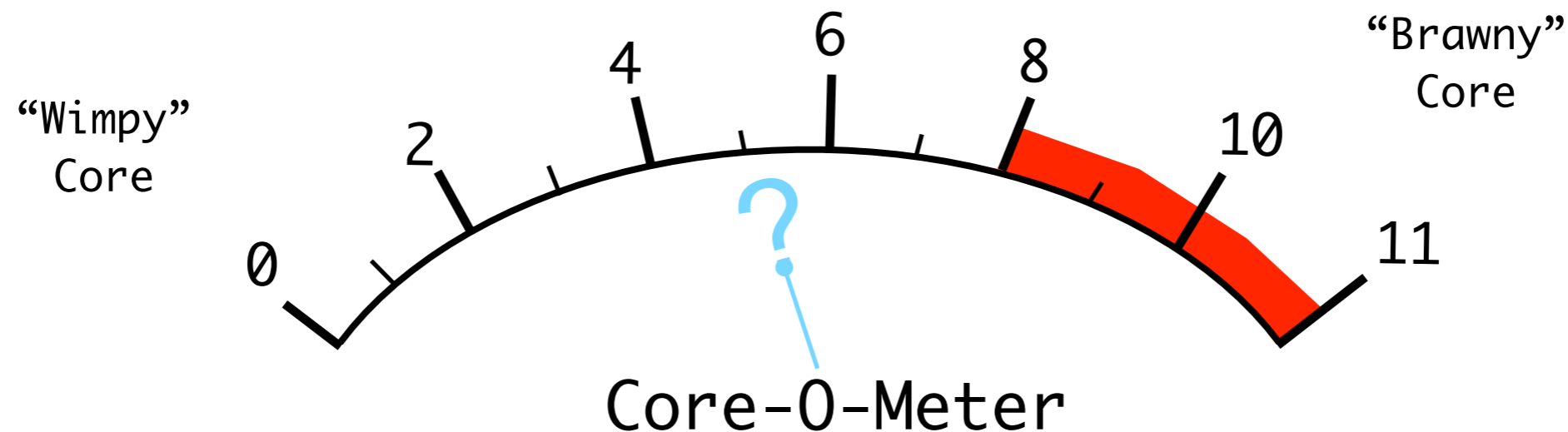
^c Phoenix MapReduce Suite

Outline

- Context
- Technology Trends
- Algorithms and Parallelism
- **Low-Power versus High-Performance Tradeoff**
- Concurrency Limits and Performance Beyond Parallelism
- Summary

How Low-Power? How-Parallel?

- **How low-power and restricted-performance should one go?** [Barroso 2010]
 - Single-thread performance still matters for multi-core [Hill and Marty 2008]
 - When cores are too simple, cost becomes dominated by **packaging costs**

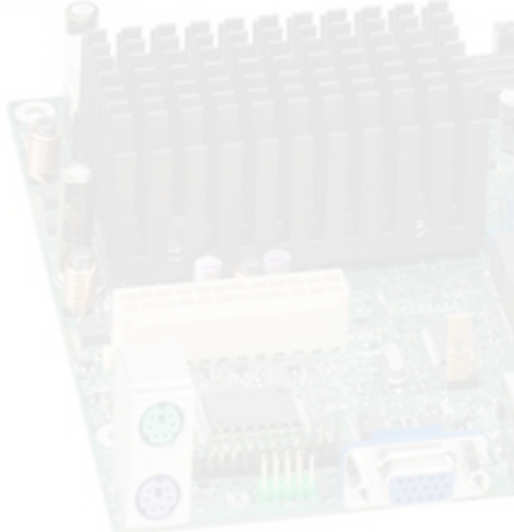


[Barroso 2010] T. Mudge and U. Holzle. Challenges and Opportunities for Extremely Energy-Efficient Processors. *IEEE Micro*, (3) 4, 2010

[Hill and Marty 2008] M. D. Hill and M. R. Marty. Amdahl's Law in the Multicore Era. *IEEE Computer*, (41) 7, 2008

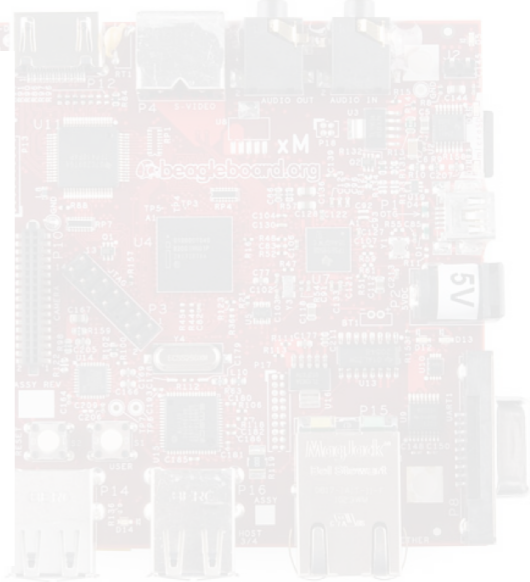
Platforms

- **Candidate processors** for scale-out systems: Atom, PowerPC, ARM
 - Three hardware reference designs; **all processors implemented in 45nm CMOS**
 - All three systems running Linux distributions based on kernel 2.6.32



Common Properties:

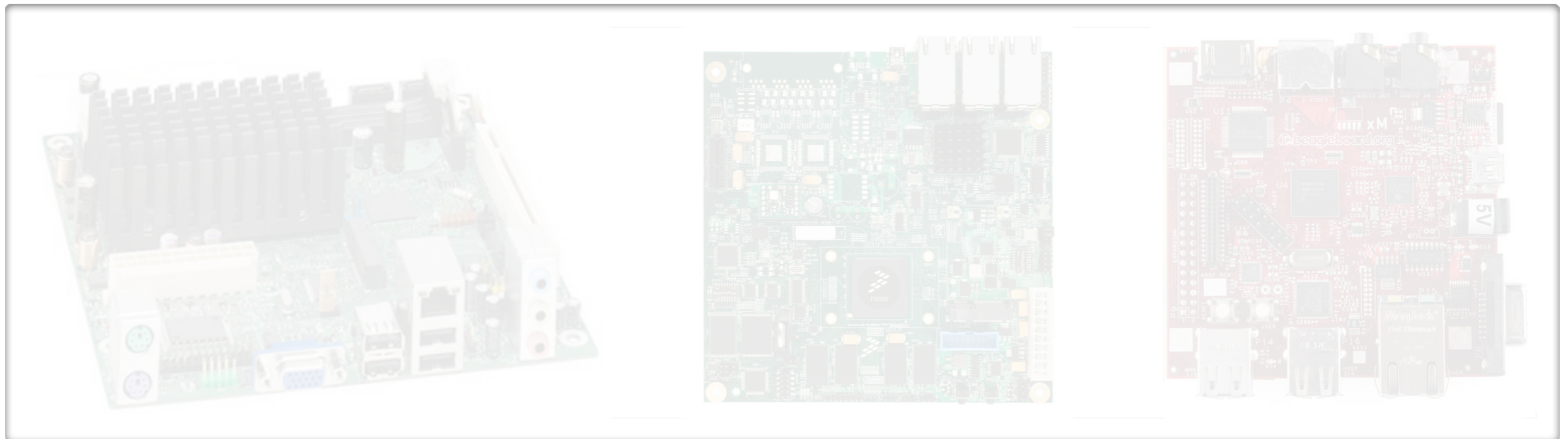
- Identical 4GB flash disk
- Lab-grade power meter, 1 measurement per second, sub-mA resolution
- Power measured for whole platform



Platform	General-Purpose Cores	CPU Clock	Cache Hierarchy
Atom™ D510MO	2 x86-64	1.66 GHz	32 K/24 K L1 I/D (per core) and 1 M L2
Freescale™ P2020RDB	2 Power Architecture® e500	1.0 GHz	32 K/32 K L1 I/D (per core), 512 K shared L2
TI DM3730 (Beagleboard-xM)	1 ARM® Cortex™ -A8	1.0 GHz	32 K/32 K L1 I/D, 256 K L2

Platforms

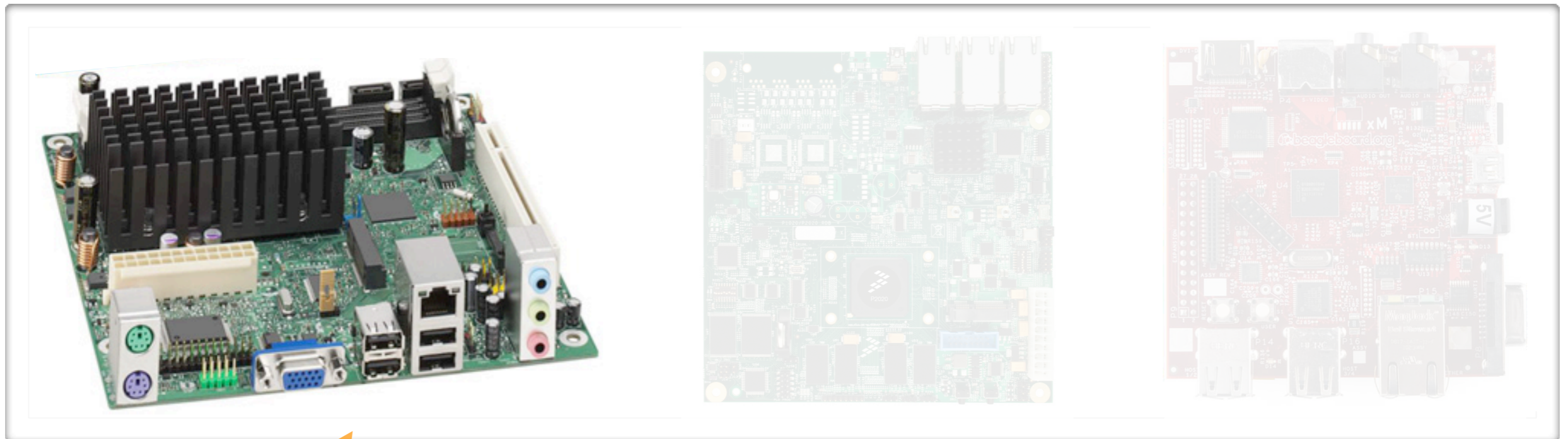
- **Candidate processors** for scale-out systems: Atom, PowerPC, ARM
 - Three hardware reference designs; **all processors implemented in 45nm CMOS**
 - All three systems running Linux distributions based on kernel 2.6.32



Platform	General-Purpose Cores	CPU Clock	Cache Hierarchy
Atom™ D510MO	2 x86-64	1.66 GHz	32 K/24 K L1 I/D (per core) and 1 M L2
Freescale™ P2020RDB	2 Power Architecture® e500	1.0 GHz	32 K/32 K L1 I/D (per core), 512 K shared L2
TI DM3730 (Beagleboard-xM)	1 ARM® Cortex™ -A8	1.0 GHz	32 K/32 K L1 I/D, 256 K L2

Platforms

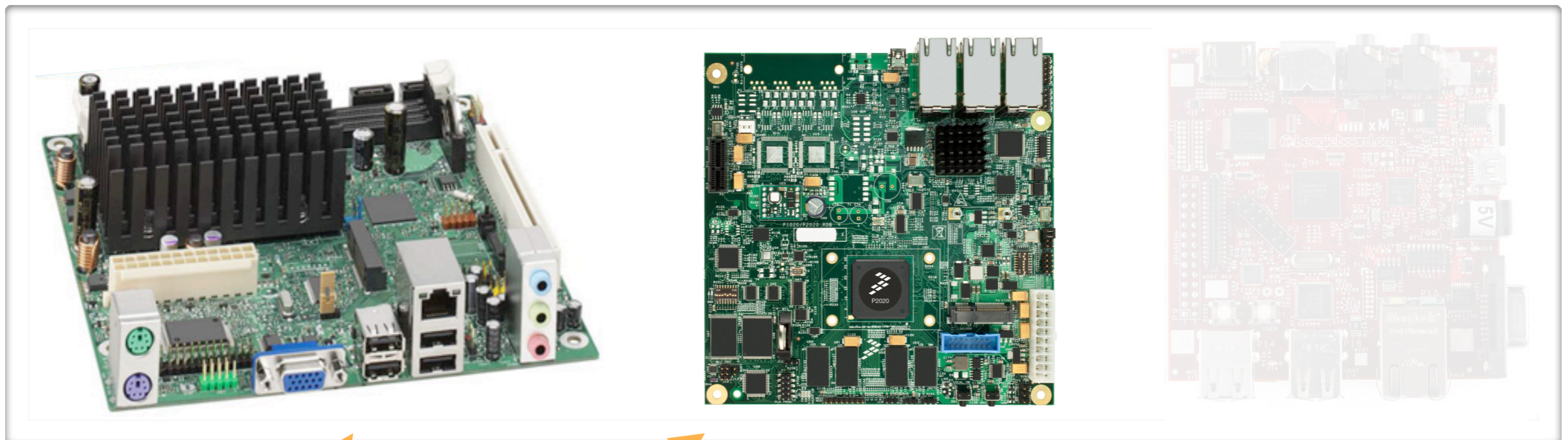
- **Candidate processors** for scale-out systems: Atom, PowerPC, ARM
 - Three hardware reference designs; **all processors implemented in 45nm CMOS**
 - All three systems running Linux distributions based on kernel 2.6.32



Platform	General-Purpose Cores	CPU Clock	Cache Hierarchy
Atom™ D510MO	2 x86-64	1.66 GHz	32 K/24 K L1 I/D (per core) and 1 M L2
Freescale™ P2020RDB	2 Power Architecture® e500	1.0 GHz	32 K/32 K L1 I/D (per core), 512 K shared L2
TI DM3730 (Beagleboard-xM)	1 ARM® Cortex™ -A8	1.0 GHz	32 K/32 K L1 I/D, 256 K L2

Platforms

- **Candidate processors** for scale-out systems: Atom, PowerPC, ARM
 - Three hardware reference designs; **all processors implemented in 45nm CMOS**
 - All three systems running Linux distributions based on kernel 2.6.32



Platform	General-Purpose Cores	CPU Clock	Cache Hierarchy
Atom™ D510MO	2 x86-64	1.66 GHz	32 K/24 K L1 I/D (per core) and 1 M L2
Freescale™ P2020RDB	2 Power Architecture® e500	1.0 GHz	32 K/32 K L1 I/D (per core), 512 K shared L2
TI DM3730 (Beagleboard-xM)	1 ARM® Cortex™ -A8	1.0 GHz	32 K/32 K L1 I/D, 256 K L2

Platforms

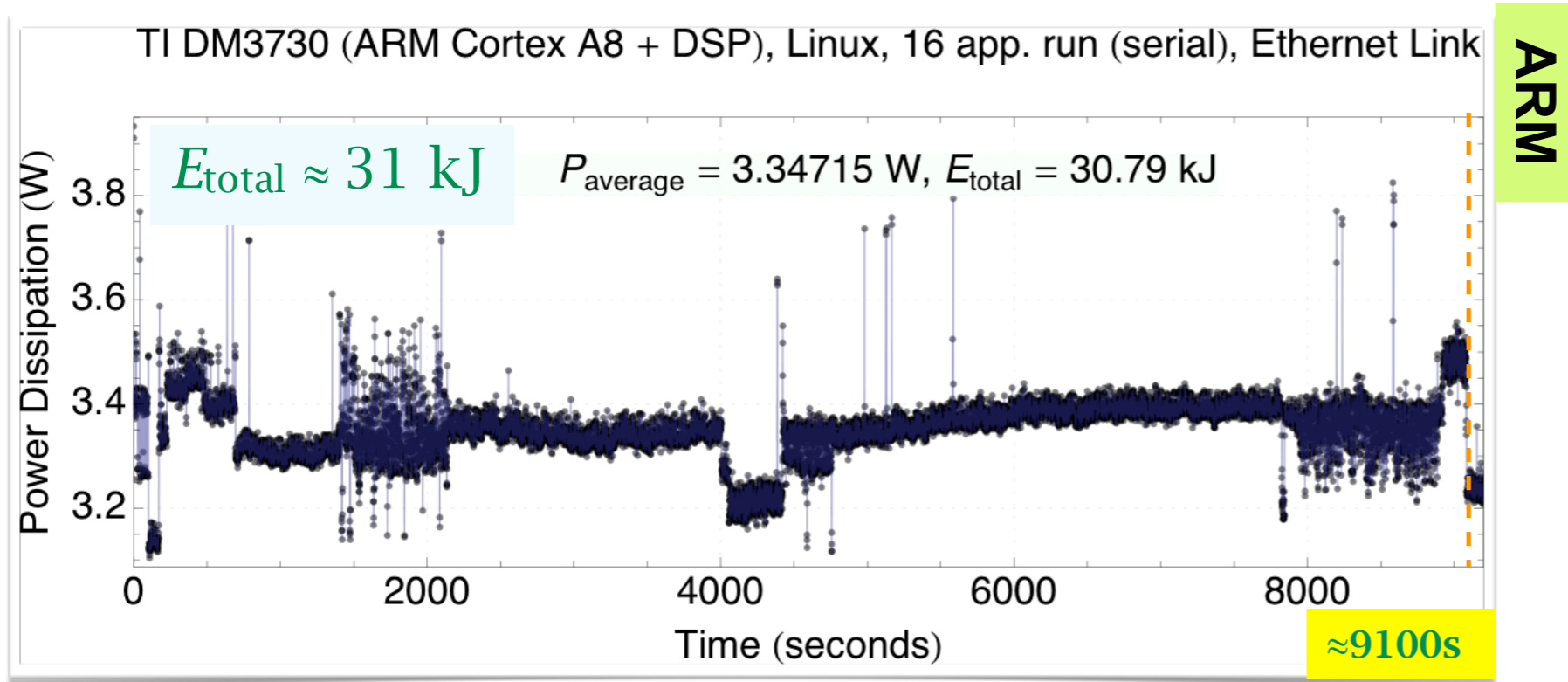
- **Candidate processors** for scale-out systems: Atom, PowerPC, ARM
 - Three hardware reference designs; **all processors implemented in 45nm CMOS**
 - All three systems running Linux distributions based on kernel 2.6.32



Platform	General-Purpose Cores	CPU Clock	Cache Hierarchy
Atom™ D510MO	2 x86-64	1.66 GHz	32 K/24 K L1 I/D (per core) and 1 M L2
Freescale™ P2020RDB	2 Power Architecture® e500	1.0 GHz	32 K/32 K L1 I/D (per core), 512 K shared L2
TI DM3730 (Beagleboard-xM)	1 ARM® Cortex™ -A8	1.0 GHz	32 K/32 K L1 I/D, 256 K L2

Pictures are NOT to scale

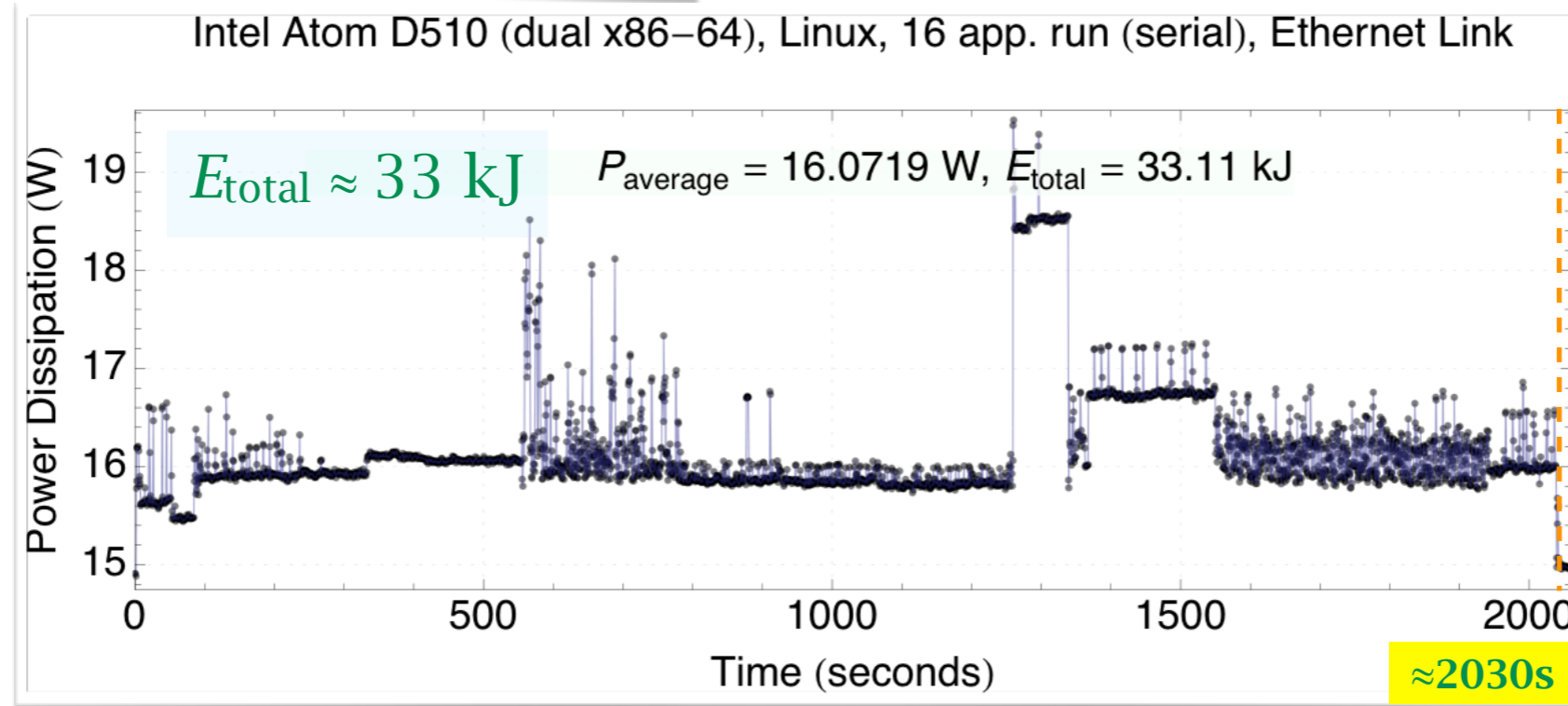
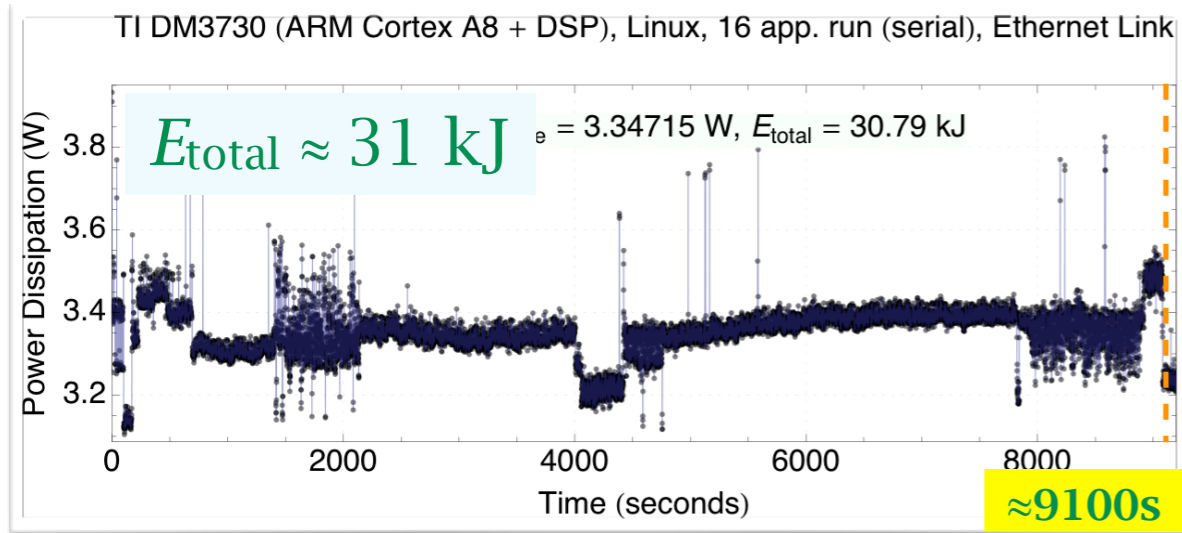
Application-Driven Whole-System Power Analysis



- Serial workload
 - Essentially tied between ARM (30.8kJ) and Atom (33.1kJ): 7.8% difference

P. Stanley-Marbell and V. Caparrós Cabezas “Performance, Power, and Thermal Analysis of Low-Power Processors for Scale-Out Systems”, In *IEEE High-Performance and Power-Aware Computing (HPPAC 2011)*.

Application-Driven Whole-System Power Analysis

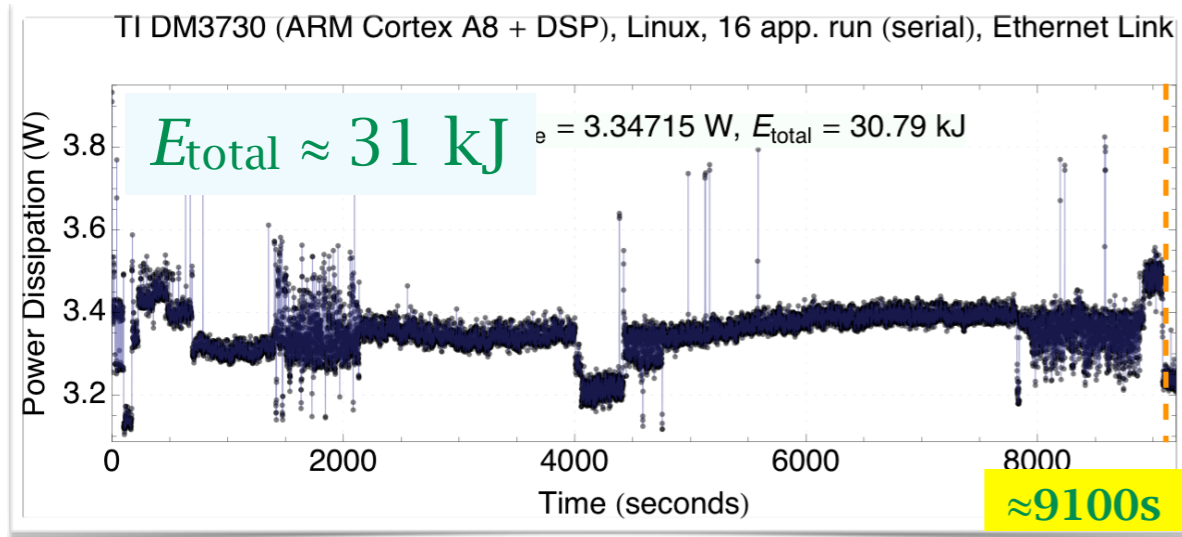


Serial workload

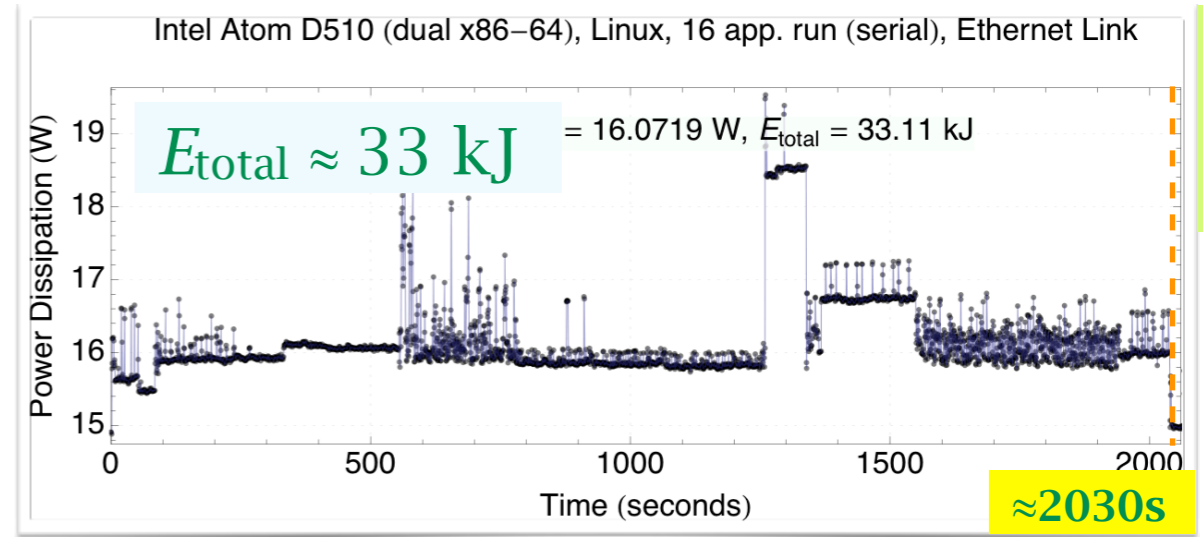
- Essentially tied between ARM (30.8kJ) and Atom (33.1kJ): 7.8% difference

P. Stanley-Marbell and V. Caparrós Cabezas “Performance, Power, and Thermal Analysis of Low-Power Processors for Scale-Out Systems”, In *IEEE High-Performance and Power-Aware Computing (HPPAC 2011)*.

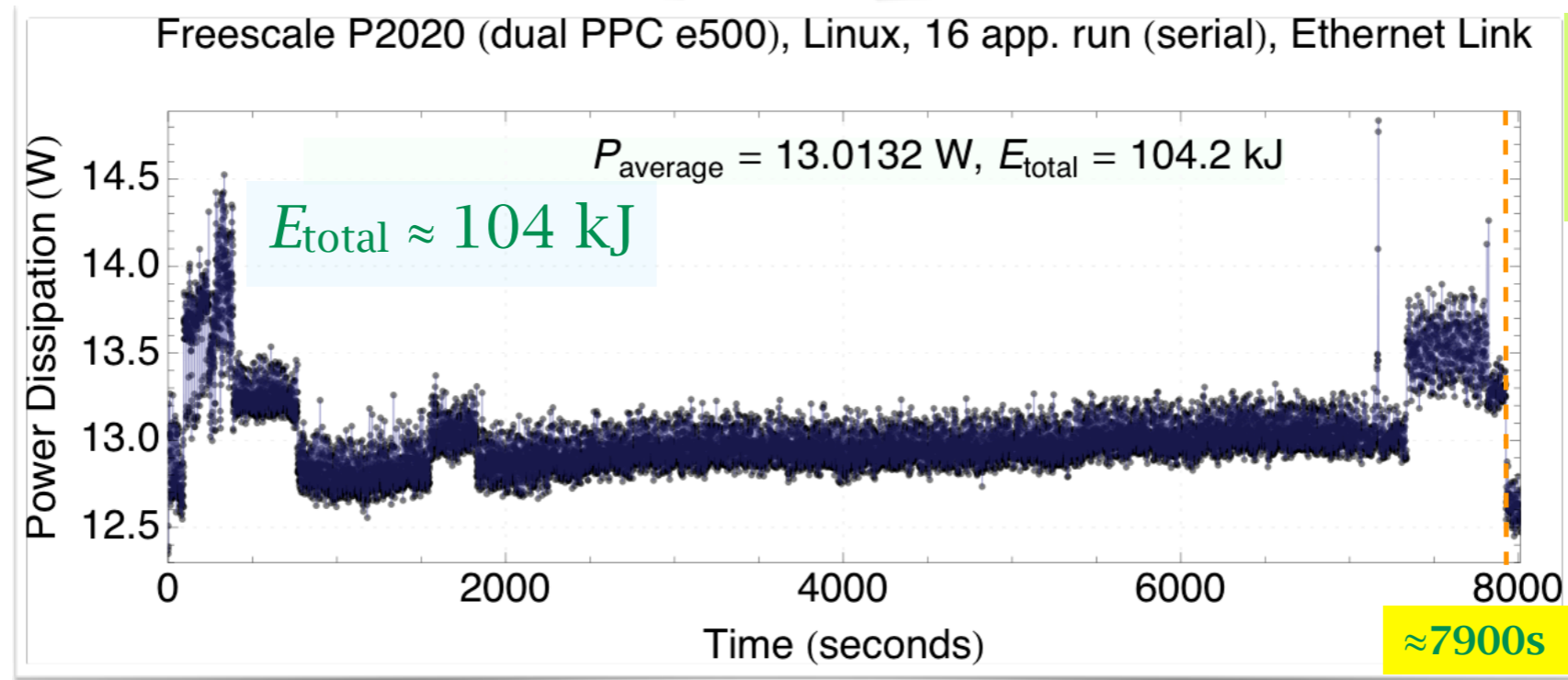
Application-Driven Whole-System Power Analysis



ARM



Atom



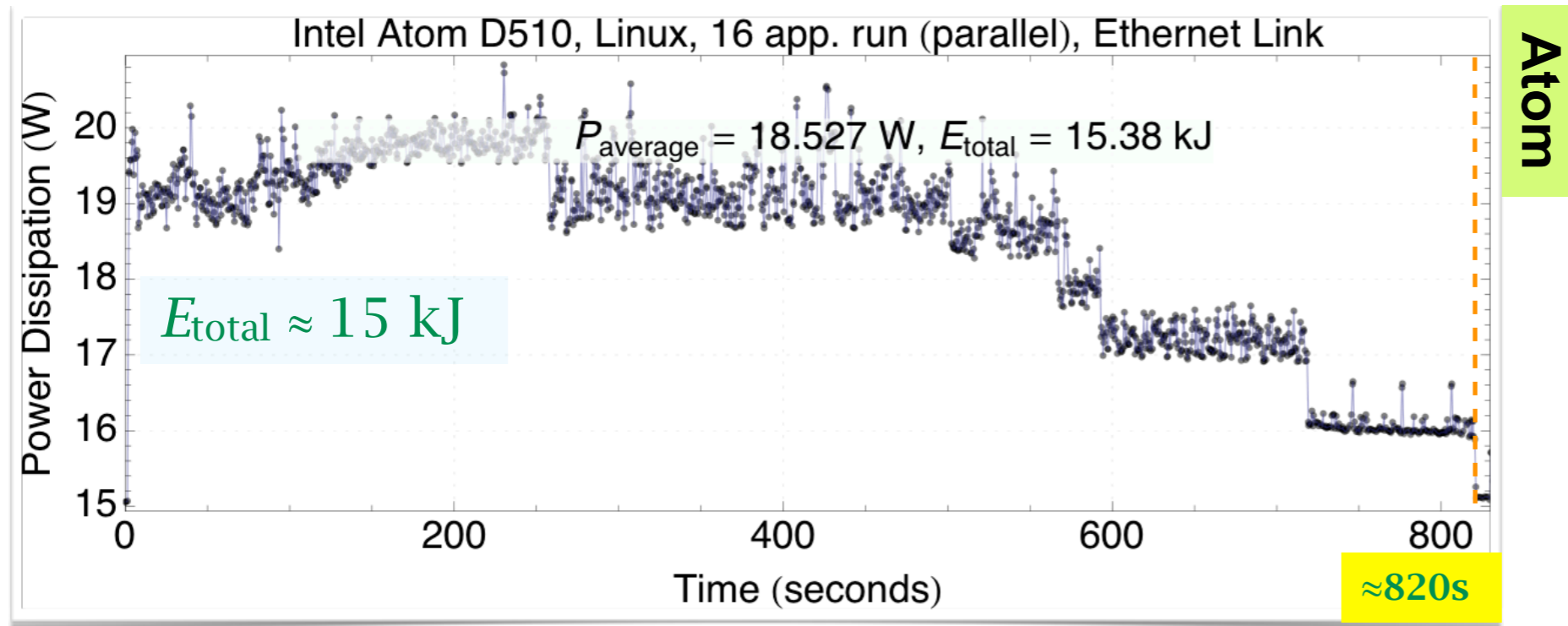
PPC

Serial workload

- Essentially tied between ARM (30.8kJ) and Atom (33.1kJ): 7.8% difference

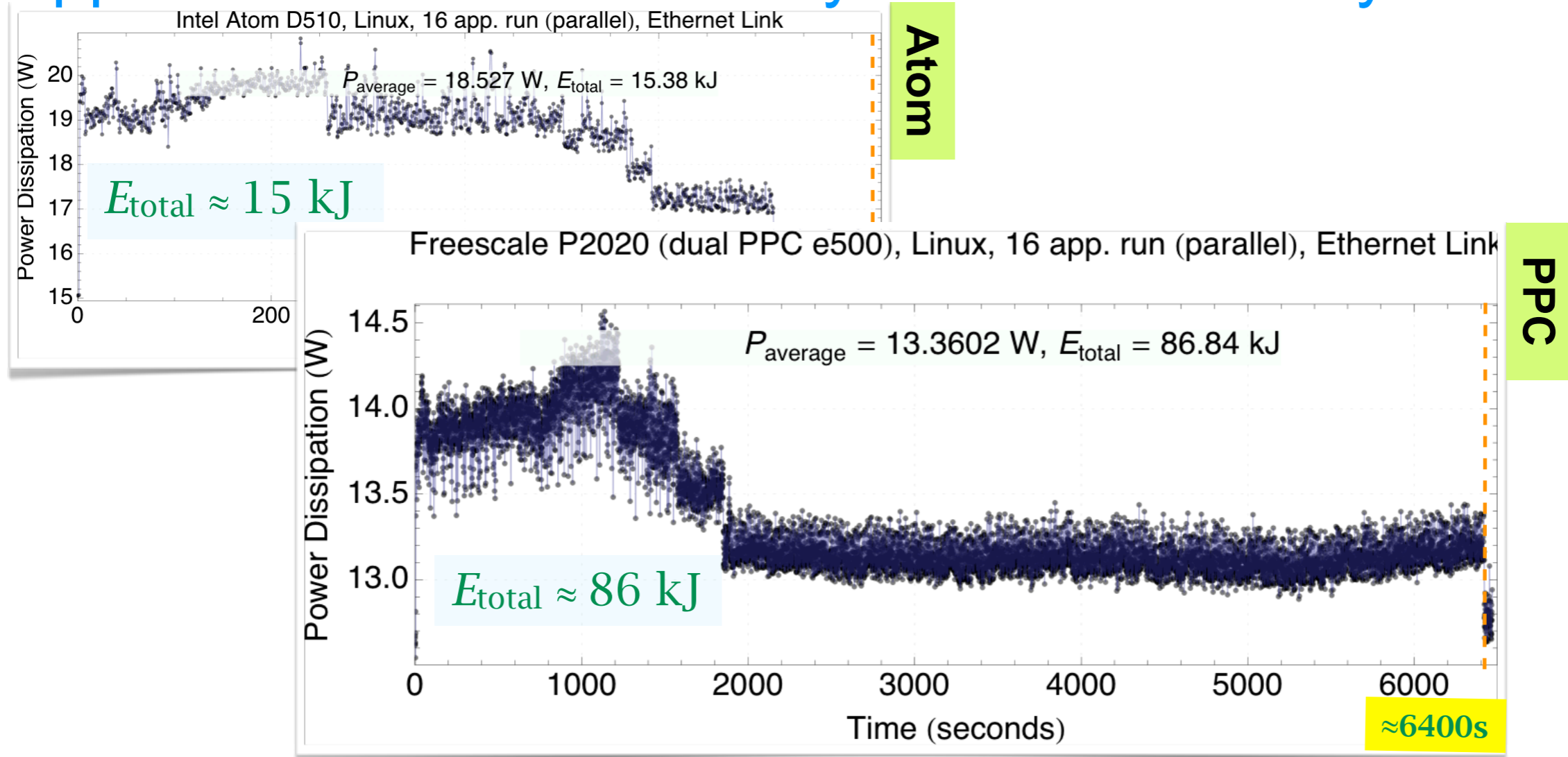
P. Stanley-Marbell and V. Caparrós Cabezas “Performance, Power, and Thermal Analysis of Low-Power Processors for Scale-Out Systems”, In *IEEE High-Performance and Power-Aware Computing (HPPAC 2011)*.

Application-Driven Whole-System Power Analysis



- “Throughput” workload
 - All 16 applications launched simultaneously; on Atom and PPC, utilize 2 cores
 - PPC: has limited FPU; perf. limited by floating-point-intensive **Equake** benchmark
- Highest-average-power system is the most energy-efficient
 - Atom platform uses least energy (15.4kJ), vs. ARM (30.8kJ) and PPC (86.8kJ)

Application-Driven Whole-System Power Analysis

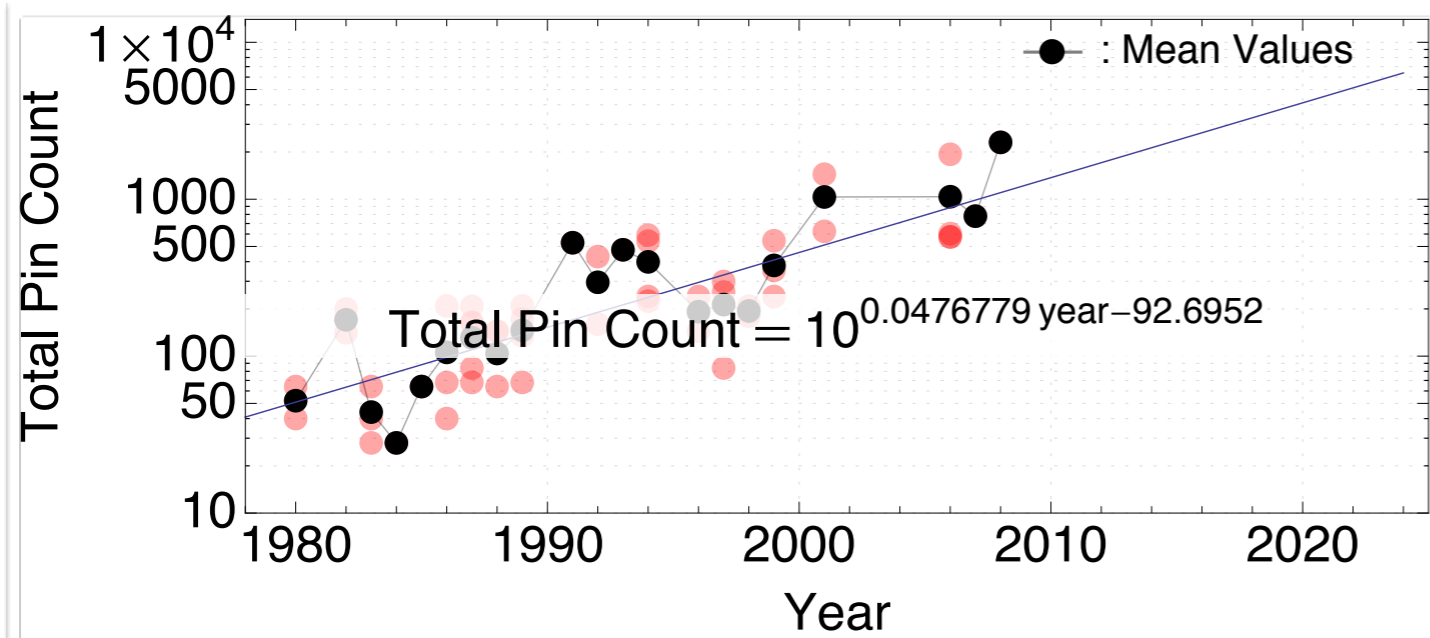


- “Throughput” workload
 - All 16 applications launched simultaneously; on Atom and PPC, utilize 2 cores
 - PPC: has limited FPU; perf. limited by floating-point-intensive **Equake** benchmark
- Highest-average-power system is the most energy-efficient
 - Atom platform uses least energy (15.4kJ), vs. ARM (30.8kJ) and PPC (86.8kJ)

Outline

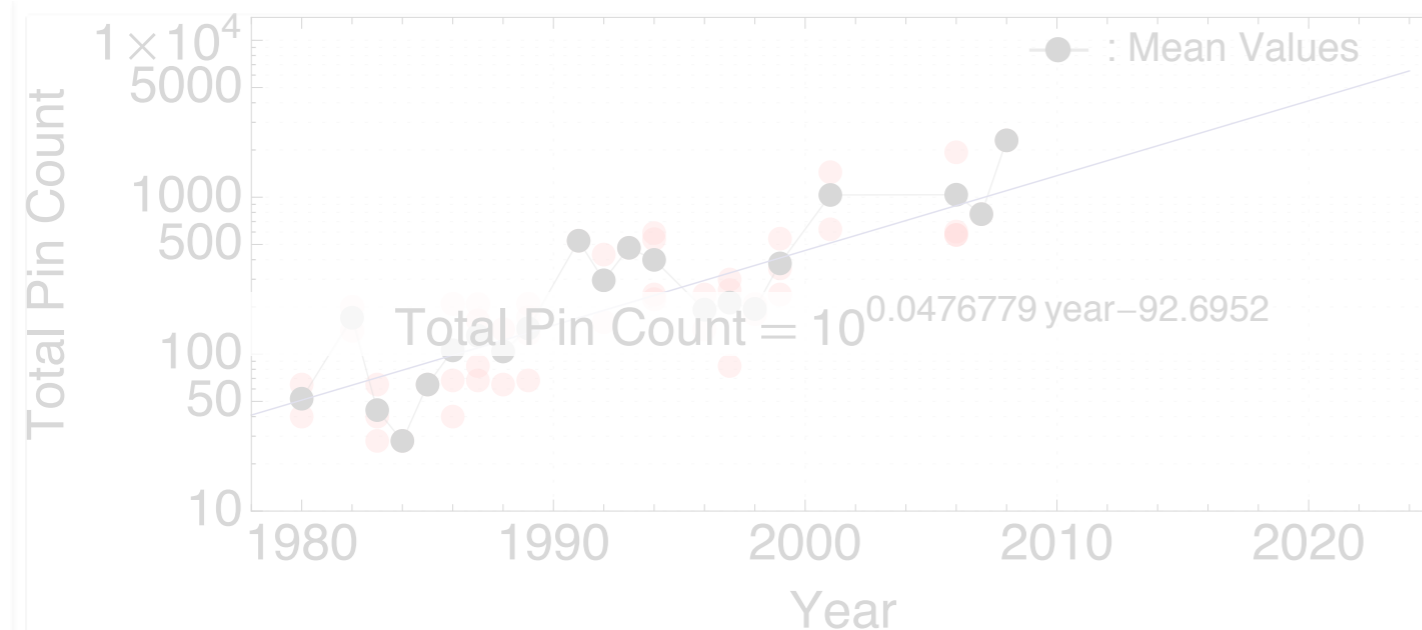
- Context
- Technology Trends
- Algorithms and Parallelism
- Low-Power versus High-Performance Tradeoff
- **Concurrency Limits and Performance Beyond Parallelism**
- Summary

Interplay Between Power and Bandwidth: Pins

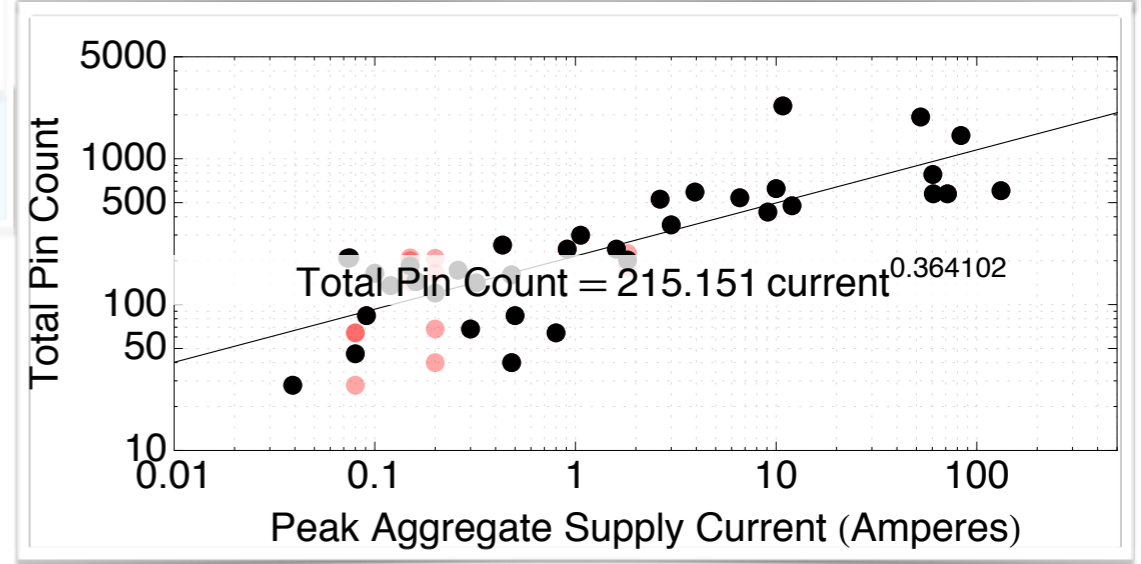
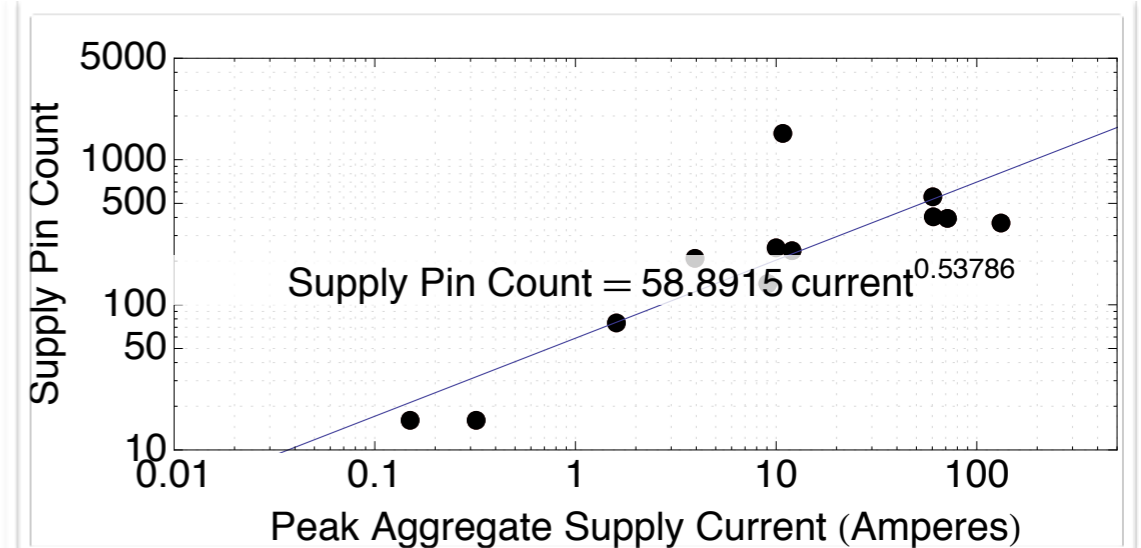


Data: from an analysis of ~150 HW publications in IEEE ISSCC and IEEE JSSC Journal, from 1980–2010

Interplay Between Power and Bandwidth: Pins

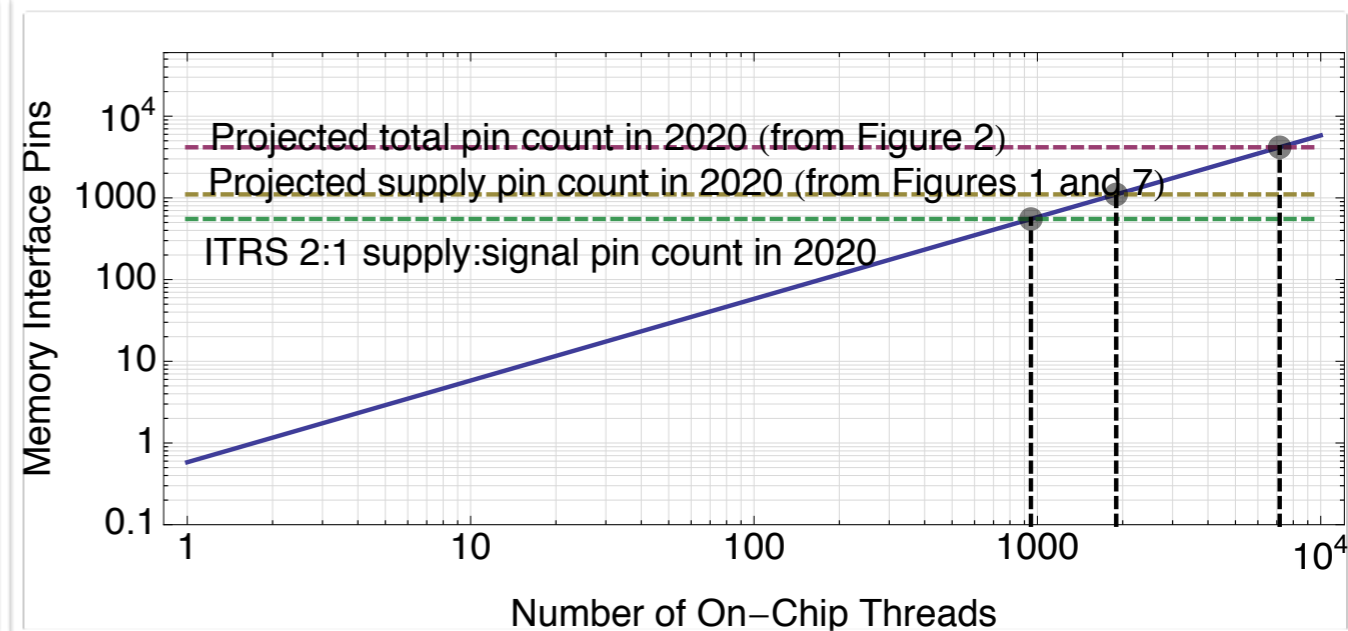
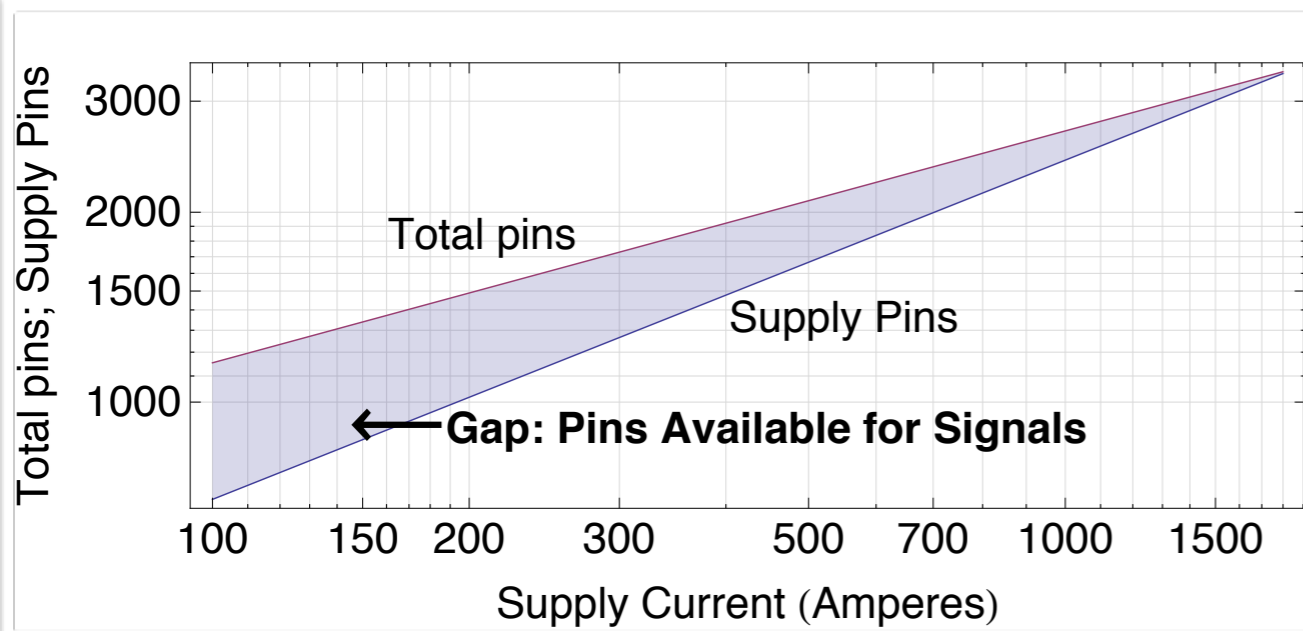


Data: from an analysis of ~150 HW publications in IEEE ISSCC and IEEE JSSC Journal, from 1980–2010



- **Bandwidth and power** are both limiters; **both require more pins**
 - Limited pin count growth (doubling every ~6 years)
 - **Supply pin count growing more rapidly than total pin count**
 - Tradeoff b/n **more supply pins for lower resistive losses**, but restricted I/O bandwidth

Interplay Between Power and Bandwidth: Pins



● Projections

- From hardware trends of previous slide, and application memory reference analyses

● Takeaway message

- In addition to constraints on available parallelism in applications, **existing packaging techniques and their improvement trends may not support >1000 cores circa 2020**

P. Stanley-Marbell, V. Caparrós Cabezas, and R. Luijten “**Pinned to the Walls—Impact of Packaging on the Memory and Power Walls**”, to appear, IEEE/ACM International Symposium on Low-Power Electronics and Design (ISLPED 2011).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture **algorithms**

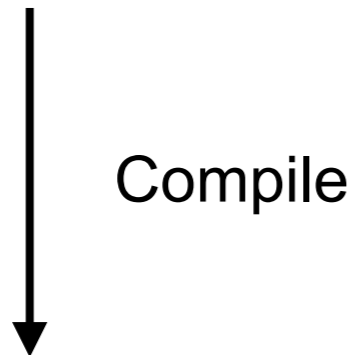
Application specifies an **algorithm for solving problem**, without being specific about what **compute problem** is

P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”, ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture **algorithms**

Application specifies an **algorithm for solving problem**, without being specific about what **compute problem** is

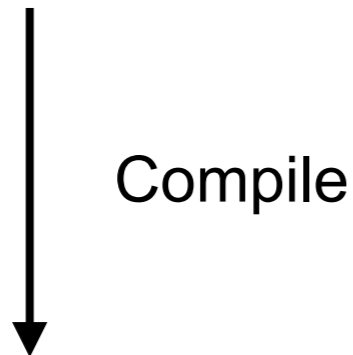


P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”, ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture **algorithms**

Application specifies an **algorithm for solving problem**, without being specific about what **compute problem** is



```
110010000
000000000
011001010
100110001
```

Instruction sequence for
stored-program computer

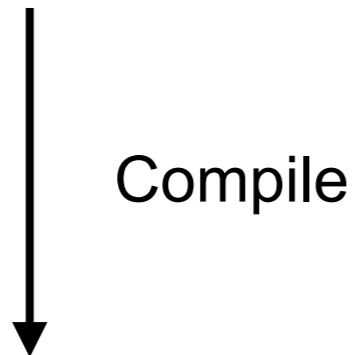
(Essentially an efficient
Turing Machine implementation)

P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”,
ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture **algorithms**

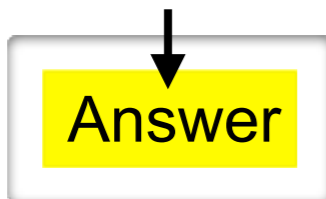
Application specifies an **algorithm for solving problem**, without being specific about what **compute problem** is



```
110010000
000000000
011001010
100110001
```

Instruction sequence for
stored-program computer

(Essentially an efficient
Turing Machine implementation)



P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”,
ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture algorithms

Application specifies an algorithm for solving problem, without being specific about what compute problem is

Compile



Instruction sequence for stored-program computer

(Essentially an efficient Turing Machine implementation)

Answer

P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”, ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture **algorithms**

Application specifies an **algorithm** for **solving problem**, without being specific about what **compute problem** is

Compile



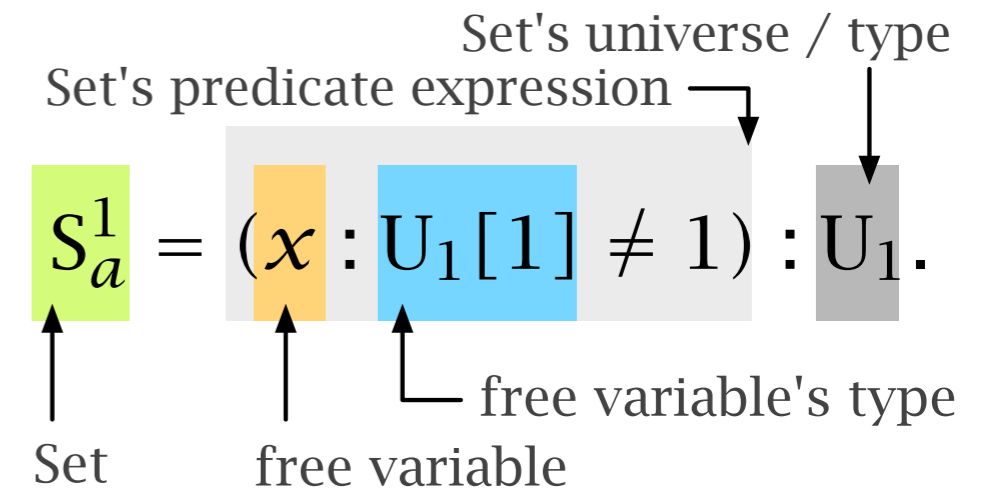
Instruction sequence for stored-program computer

(Essentially an efficient Turing Machine implementation)

Answer

New: “Binaries” capture **compute problems**

Problem Specification (**Sal**)



P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”, ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture **algorithms**

Application specifies an **algorithm** for **solving problem**, without being specific about what **compute problem** is

Compile



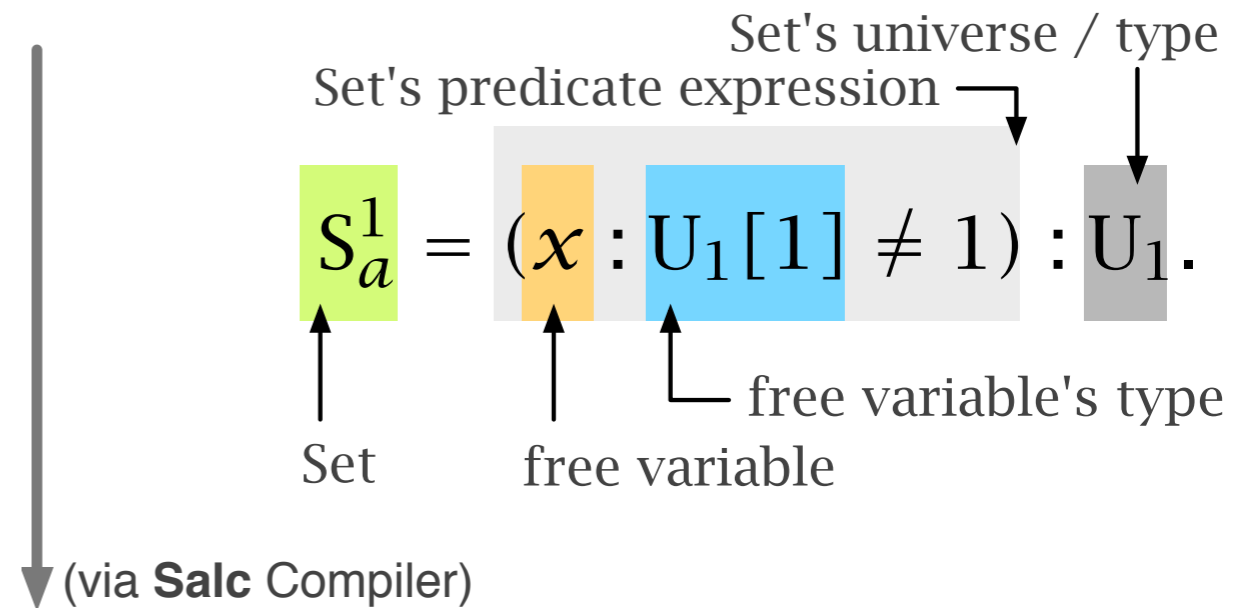
Instruction sequence for stored-program computer

(Essentially an efficient Turing Machine implementation)

Answer

New: “Binaries” capture **compute problems**

Problem Specification (**Sal**)



P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”, ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture **algorithms**

Application specifies an **algorithm** for **solving problem**, without being specific about what **compute problem** is

Compile



Instruction sequence for stored-program computer

(Essentially an efficient Turing Machine implementation)

Answer

New: “Binaries” capture **compute problems**

Problem Specification (**Sal**)

$$S_a^1 = (x : U_1[1] \neq 1) : U_1.$$

Annotations:
 - S_a^1 : Set
 - x : free variable
 - $U_1[1]$: free variable's type
 - U_1 : Set's universe / type

(via **Salc** Compiler)

Svm Machine State Representing Problem

P Registers	S Registers	U Registers	Runtime Symbol Table		
	P1:U3	<1,5, ..., 94>	Variable	Value	Scope
...	(P1 P7) : U1	<"en", ..., "bij">	qx	2	14
...
...	true:U2	{2.1, ..., -1.3}	mquantvar	"hello"	2

P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”, ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Beyond Parallelism: Algorithms versus Compute Problems

Old: “Binaries” capture **algorithms**

Application specifies an **algorithm** for **solving problem**, without being specific about what **compute problem** is

Compile



Instruction sequence for stored-program computer

(Essentially an efficient Turing Machine implementation)

Answer

New: “Binaries” capture **compute problems**

Problem Specification (**Sal**)

$$S_a^1 = (x : U_1[1] \neq 1) : U_1.$$

Annotations:
 - S_a^1 : Set
 - x : free variable
 - $U_1[1]$: free variable's type
 - U_1 : Set's universe / type

(via **Salc** Compiler)

Svm Machine State Representing Problem

P Registers	S Registers	U Registers	Runtime Symbol Table		
	P1:U3	<1,5, ..., 94>	Variable	Value	Scope
...	(P1 P7) : U1	<"en", ..., "bij">	qx	2	14
...
...	true:U2	{2.1, ..., -1.3}	mquantvar	"hello"	2

Answer

P. Stanley-Marbell “**Sal/Svm—An Assembly Language and Virtual Machine for Computing with Non-Enumerated Sets**”, ACM Virtual Machines and Intermediate Languages (VMIL 2010).

Outline

- Context
- Technology Trends
- Algorithms and Parallelism
- Low-Power versus High-Performance Tradeoff
- Concurrency Limits and Performance Beyond Parallelism
- **Summary**

Summary

● Context

- Continued performance increases desirable for, e.g., future SKA backend

● Technology trends

- Performance growth no longer through clock speed increases; power-limited

● Parallelism as a performance vehicle

- Presented framework for characterizing available parallelism (ILP, TLP) in applications

● Parallelism, performance, and low-power tradeoffs

- Characterization of not just average power, but energy-to-solution for whole systems

● Potential limits to scaling via parallelism

- New approaches desirable for improved bandwidth to support parallelism
- Alternatives to single-algorithm scaling: algorithmic choice

Backup Slides

