Advancing Computer Systems without Technology Progress

ISAT Outbrief, April 17-18, of
DARPA/ISAT Workshop, March 26-27, 2012

Organized by: Mark Hill & Christos Kozyrakis
w/ Serena Chan & Melanie Sineath

Approved for Public Release, Distribution Unlimited

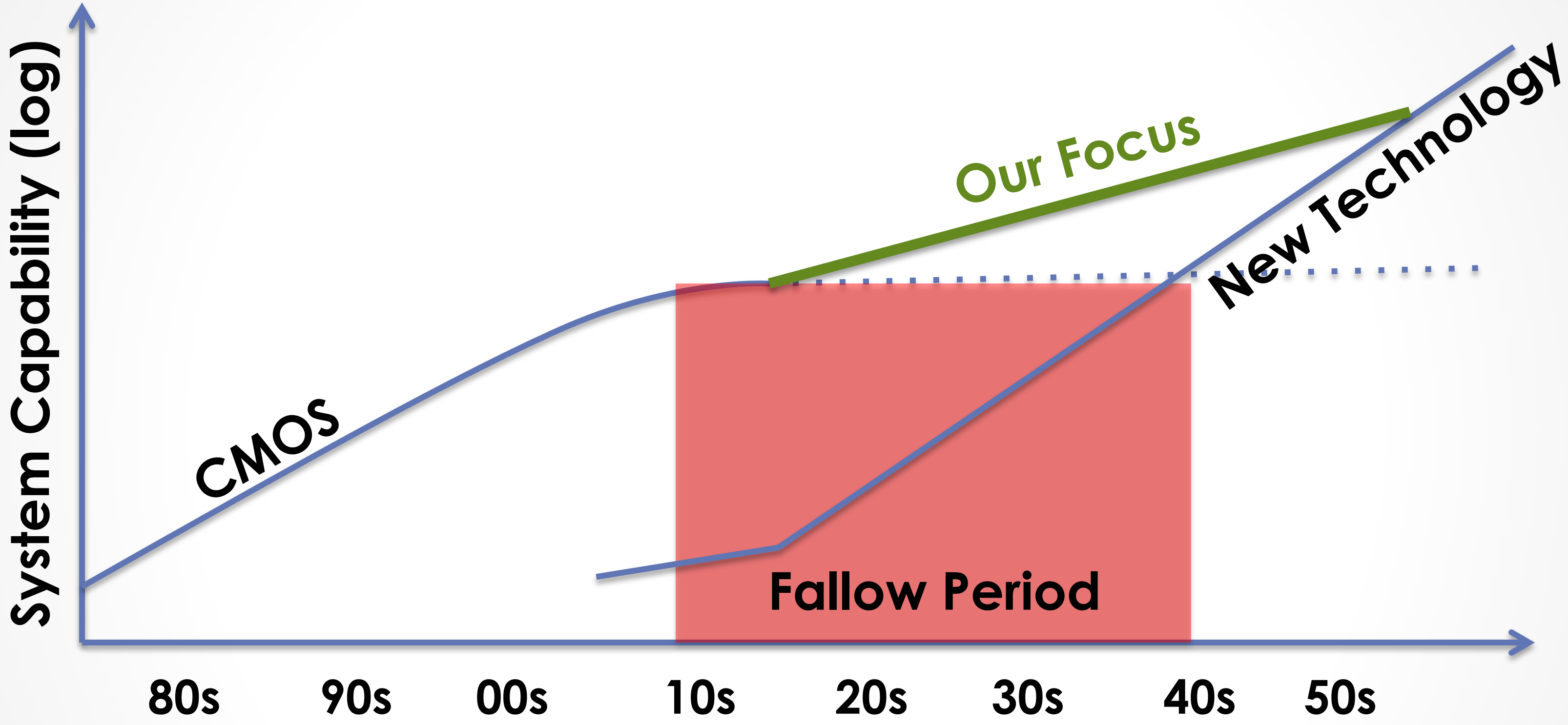
The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

Workshop Premises & Challenge

- CMOS transistors will soon stop getting "better“
- Post-CMOS technologies not ready
- Computer system superiority central to US security, government, education, commerce, etc.
- **Key question: How to advance computer systems without (significant) technology progress?**

Approved for Public Release, Distribution Unlimited

The Graph



Approved for Public Release, Distribution Unlimited

Surprise 1 of 2

- **Can Harvest in the "Fallow" Period!**
- 2 decades of Moore's Law-like perf./energy gains
- Wring out inefficiencies used to harvest Moore's Law

HW/SW Specialization/Co-design (3-100x)

Reduce SW Bloat (2-1000x)

Approximate Computing (2-500x)

~1000x = 2 decades of Moore's Law!

Approved for Public Release, Distribution Unlimited

“Surprise” 2 of 2

- **Systems must exploit LOCALITY-AWARE parallelism**

- Parallelism Necessary, but not Sufficient
- As communication’s energy costs dominate

- Shouldn’t be a surprise, but many are in denial

- **Both surprises hard, requiring “vertical cut” thru SW/HW**

Approved for Public Release, Distribution Unlimited

Maybe Our Work Done? ☺

Approved for Public Release, Distribution Unlimited

Outline

- Workshop Background & Organization
  - Participants
  - Organization
  - Output

- Workshop Insights & Recommendations

48 Great Participants

- Participant distinctions
  - 6 members of the National Academy of Engineering
  - 7 fellows and senior fellows from industry
  - 12 ACM or IEEE fellows
  - 2 Eckert-Mauchly award recipients
  - 8 Assistant/Associate professors

- Diverse institutions (some in two categories):
  - 52% (25) universities
  - 31% (15) industry
    - AMD, ARM , Google, HP, Intel, Microsoft, Oracle, Nvidia, Xilinx
  - 12% (6) IDA, Lincoln Labs, SRI
  - 8% (4) DARPA

Approved for Public Release, Distribution Unlimited

Workshop Organization

- Pre-workshop prep
  - 1-page position statement & bios distributed beforehand
- Day 1
  - Two keynotes
    - Dr. Robert Colwell (DARPA)
    - Dr. James Larus (Microsoft)
  - Five break-out sessions (3.5 hours)
  - Break-out summaries/discussion (1.5)
- Day 2
  - Speed dates (3*15 minutes one-on-ones)
  - Break-out sessions w/ 2 new groups (3)
  - Better break-out summaries/discussion (1.5)

Approved for Public Release, Distribution Unlimited

The Workshop Output

Interaction!

- If you're smart, what you do is make connections. To make connections, you have to have inputs. Thus, try to avoid having the same exact inputs as everyone else. Gain new experiences and thus bring together things no one has brought together before. **–Steve Jobs**

- This outbrief
- 36 position statements
- Break-out session notes & presentations

Approved for Public Release, Distribution Unlimited

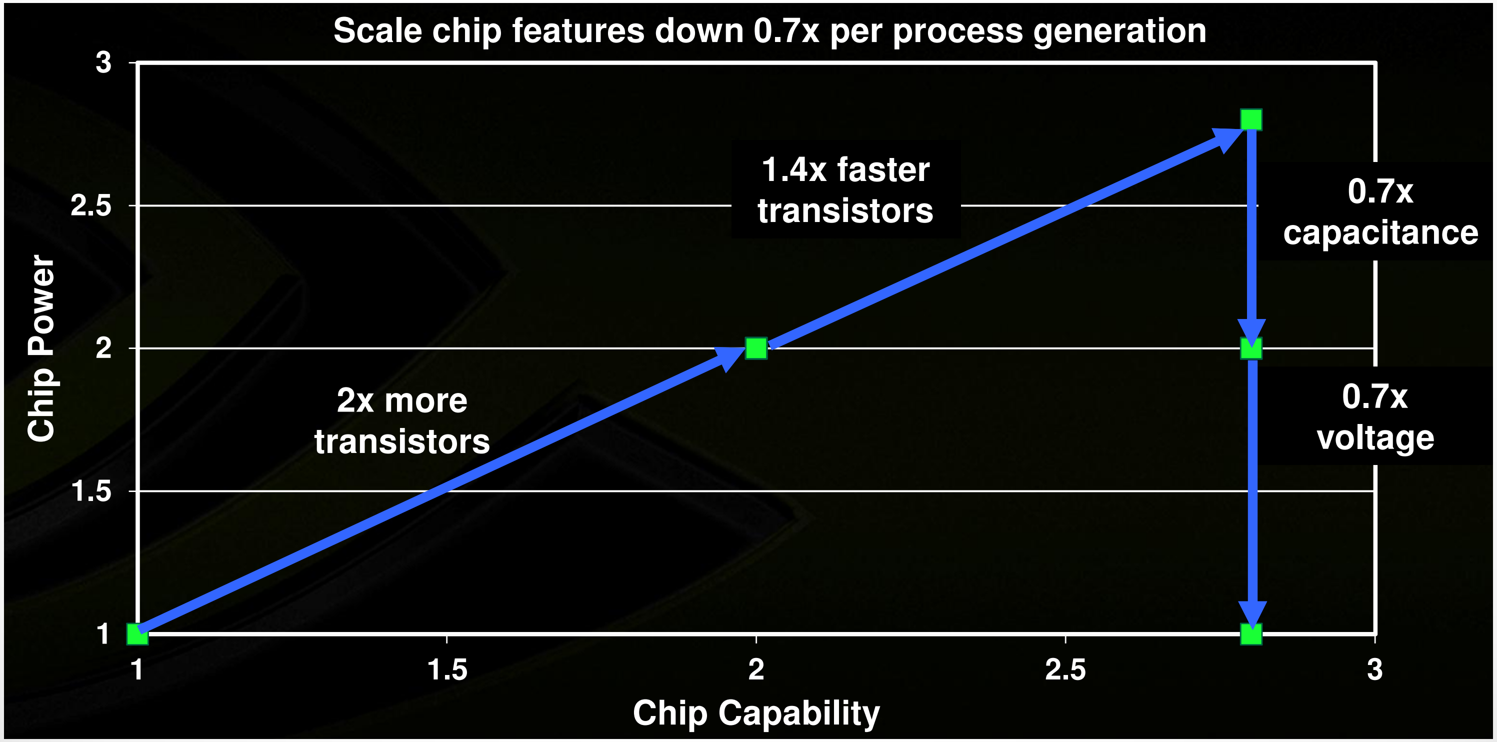
Outline

- Workshop background & Organization
- Workshop Insights & Recommendations
  - Hook & Graph
  - Research
    1. HW and SW specialization and co-design
    2. Reduce SW bloat
    3. Approximate computing
    4. Locality-aware parallelism
  - Delta & Impact
  - Backup (including participant survey data)

Approved for Public Release, Distribution Unlimited

The Hook: For Decades

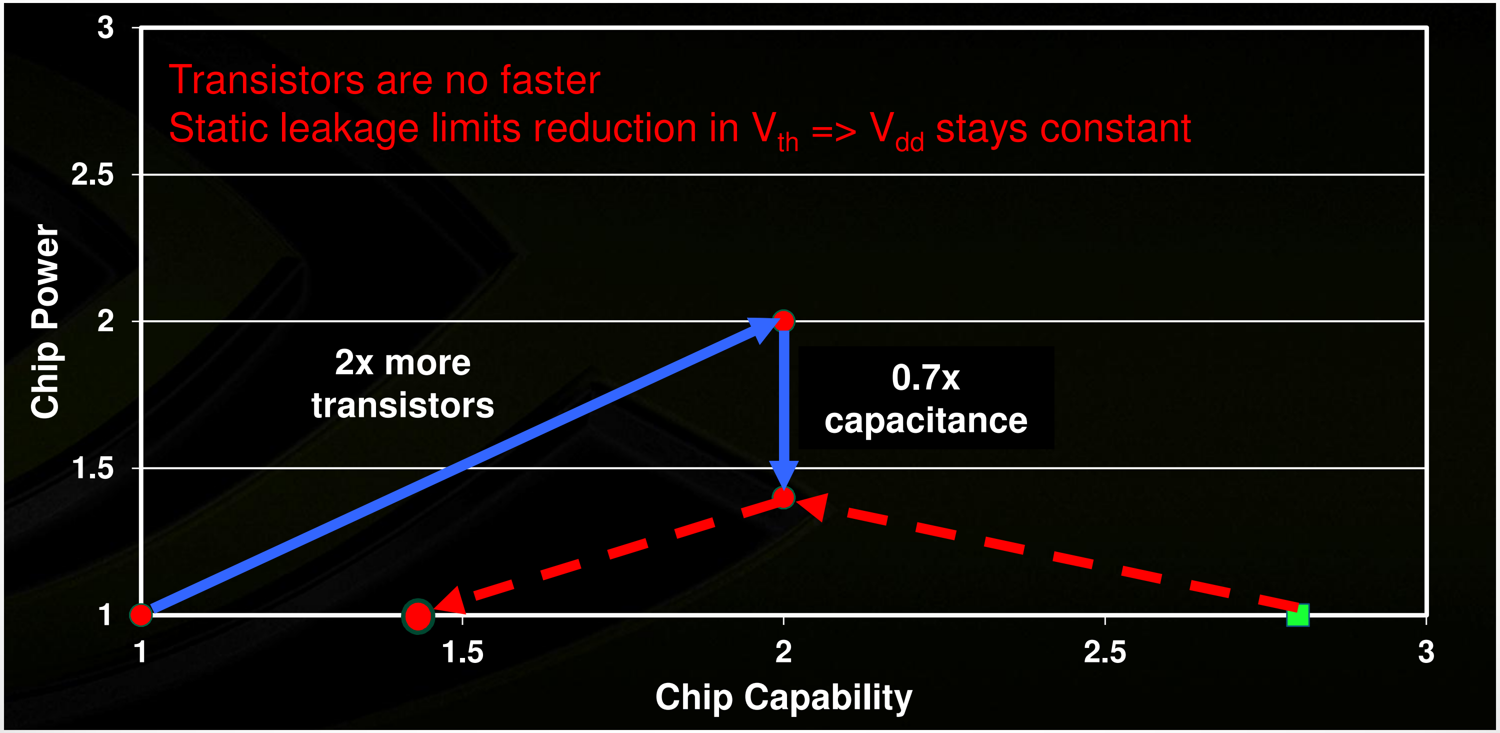
- CMOS Scaling: Moore's law + Dennard scaling
  - 2.8x in chip capability per generation at constant power
- ~5,000x performance improvement in 20 years
  - A driving force behind computing advance



Approved for Public Release, Distribution Unlimited

The Hook: Future

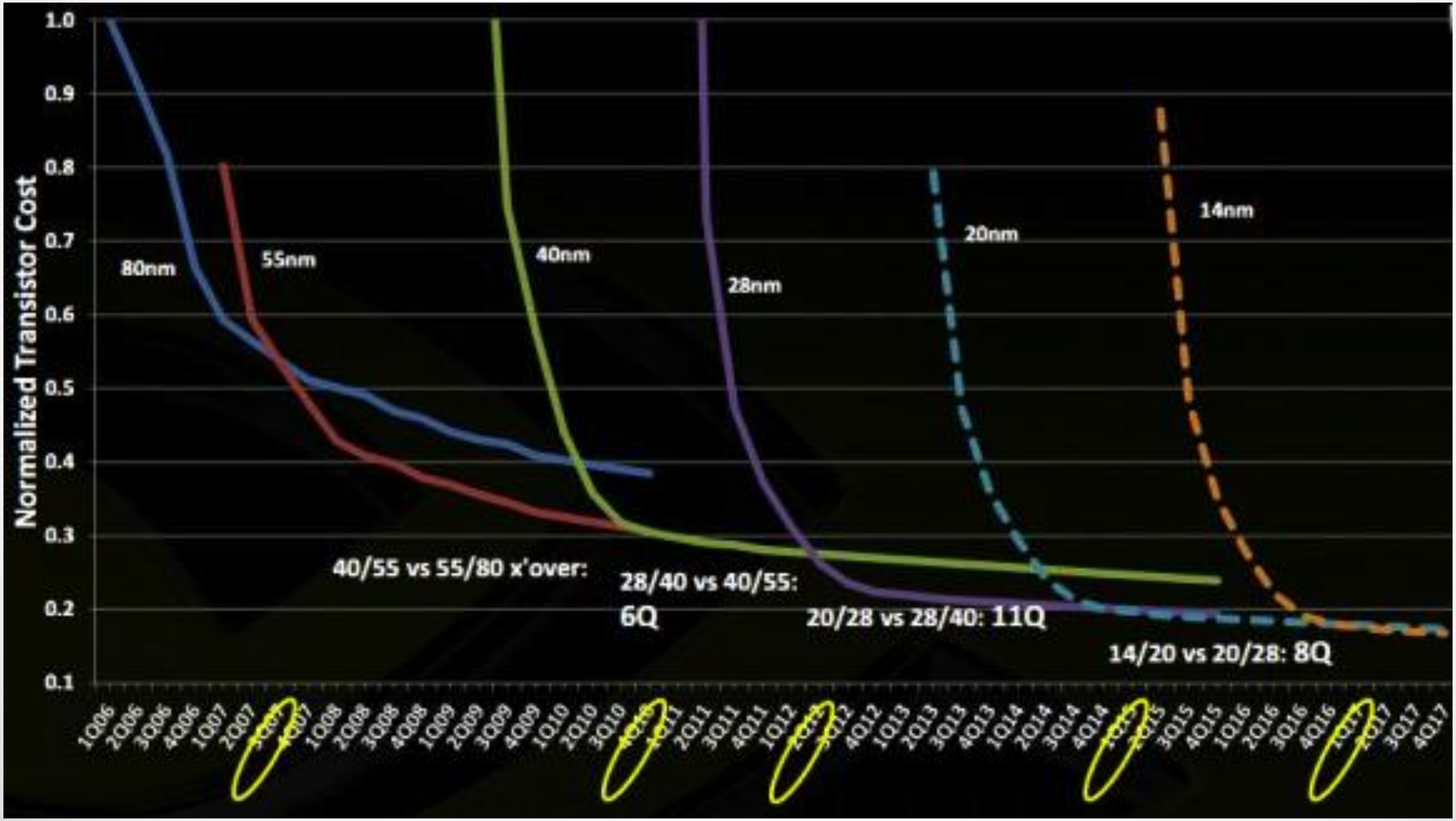
- Maybe Moore's law + NO Dennard scaling
  - Can't scale down voltages; scale transistor cost?
- ~32x gap per decade compared to before



Approved for Public Release, Distribution Unlimited

The Hook: Cost

- Future scaling failing to reduce transistor cost!

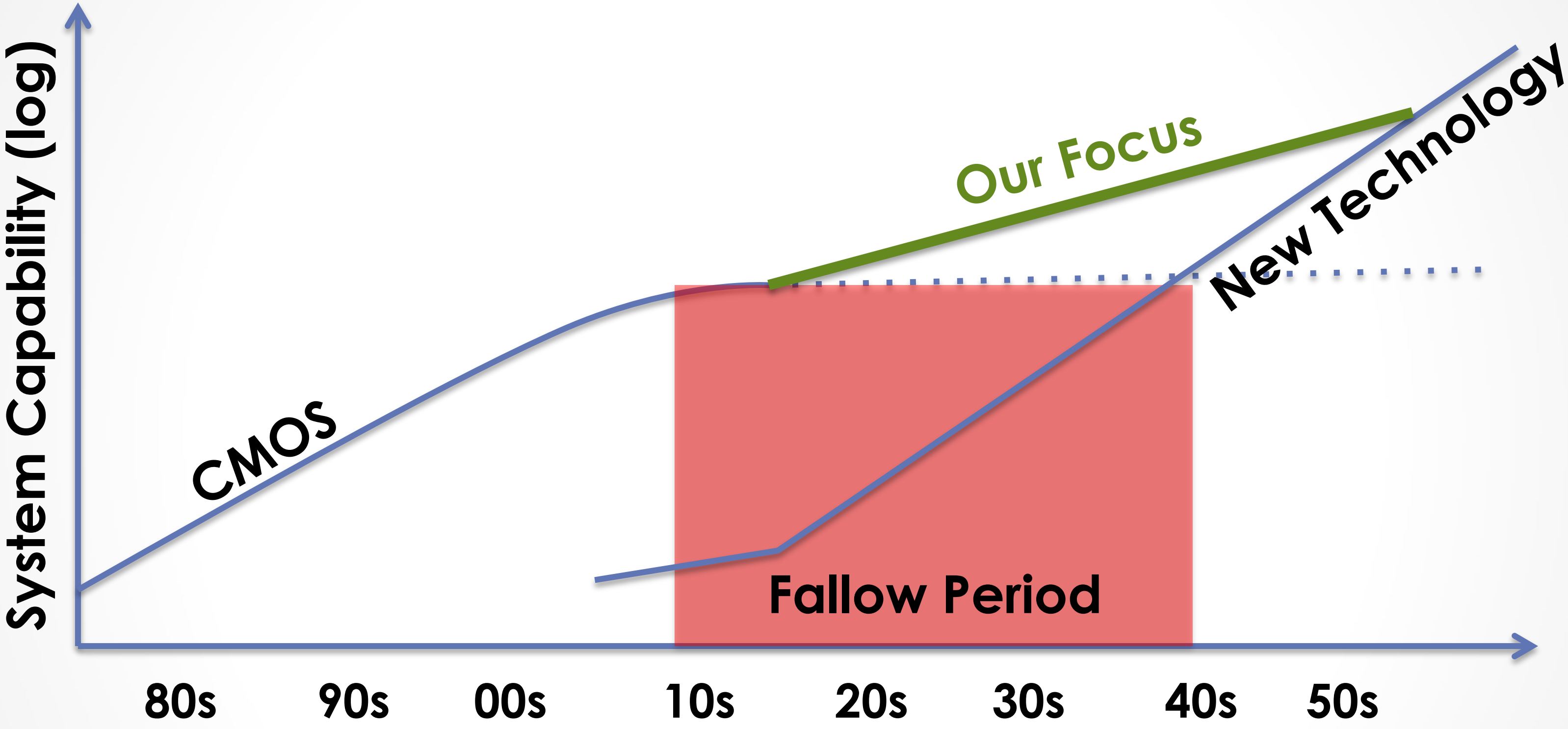


Approved for Public Release, Distribution Unlimited

The Need

- Computer system superiority is central to
  - US security
  - Government,
  - Education,
  - Commerce, etc.
- Maintain system superiority w/o CMOS scaling?
- Extend development time for CMOS replacement?

The Graph



- Fallow period (until CMOS replacement)
- Can we improve systems during this period?

Approved for Public Release, Distribution Unlimited

The Research

Four main directions identified

1. HW/SW specialization and co-design

2. Reduce SW bloat

3. Approximate computing

4. Locality-aware parallelism

Approved for Public Release, Distribution Unlimited

HW/SW Specialization & Codesign

- Now: General purpose preferred; specialization rare
- Want: Broad use of specialization at lower NRE cost
  - Languages & interfaces for specialization & co-design
  - HW/SW technology/tools for specialization
  - Power & energy management as co-design
- Apps: Big data, security, mobile systems, ML/AI on UAV systems, …
- Areas: I/O, storage, image/video, statistical, fault-tolerance, security, natural UI, key-value loopups, …

Approved for Public Release, Distribution Unlimited

Spectrum of Hardware Specialization

Metric	**Ops/mm²**	**Ops/Watt**	**Time to Soln**	**NRE**
Normalized to General-Purpose	1	1	1 (programming GPP)	1
Specialized ISA (domain specific)	1.5	3-5	2-3 (designing & programming)	1.5
Progr. Accelerator (domain specific)	3	5-10	2-3 (designing & programming)	2-3
Fixed Accelerator (app specific)	5-10	10	10 (SoC design)	3-5
Specialized Mem & Interconnect (monolithic die)	10	10	10 (SoC design)	10
Package level integration (multi die: logic,mem,analog)	10+	10+	5 (silicon interposer)	5

Reduce SW Bloat

- Now: Focused programming productivity
  - Launch complex, online services within days
  - But bloated SW stacks w/ efficiency obscured
    - Next slide: 50,000x from PHP to BLAS Parallel
- Want: Improve efficiency w/o sacrificing productivity
  - Abstractions for SW efficiency (SW "weight")
  - Performance-aware programming languages
  - Tools for performance optimization (esp. w/ composition)

SW Bloat Example: Matrix Multiply

PHP	9,298,440 ms	51,090x
Python	6,145,070 ms	33,764x
Java	348,749 ms	1816x
C	19,564 ms	107x
Tiled C	12,887 ms	71x
Vectorized	6,607 ms	36x
BLAS Parallel	182 ms	1

- Can we achieve PHP productivity at BLAS efficiency?

Approximate Computing

- Now: High-precision outputs from deterministic HW
  - Requires energy/margins & not always needed

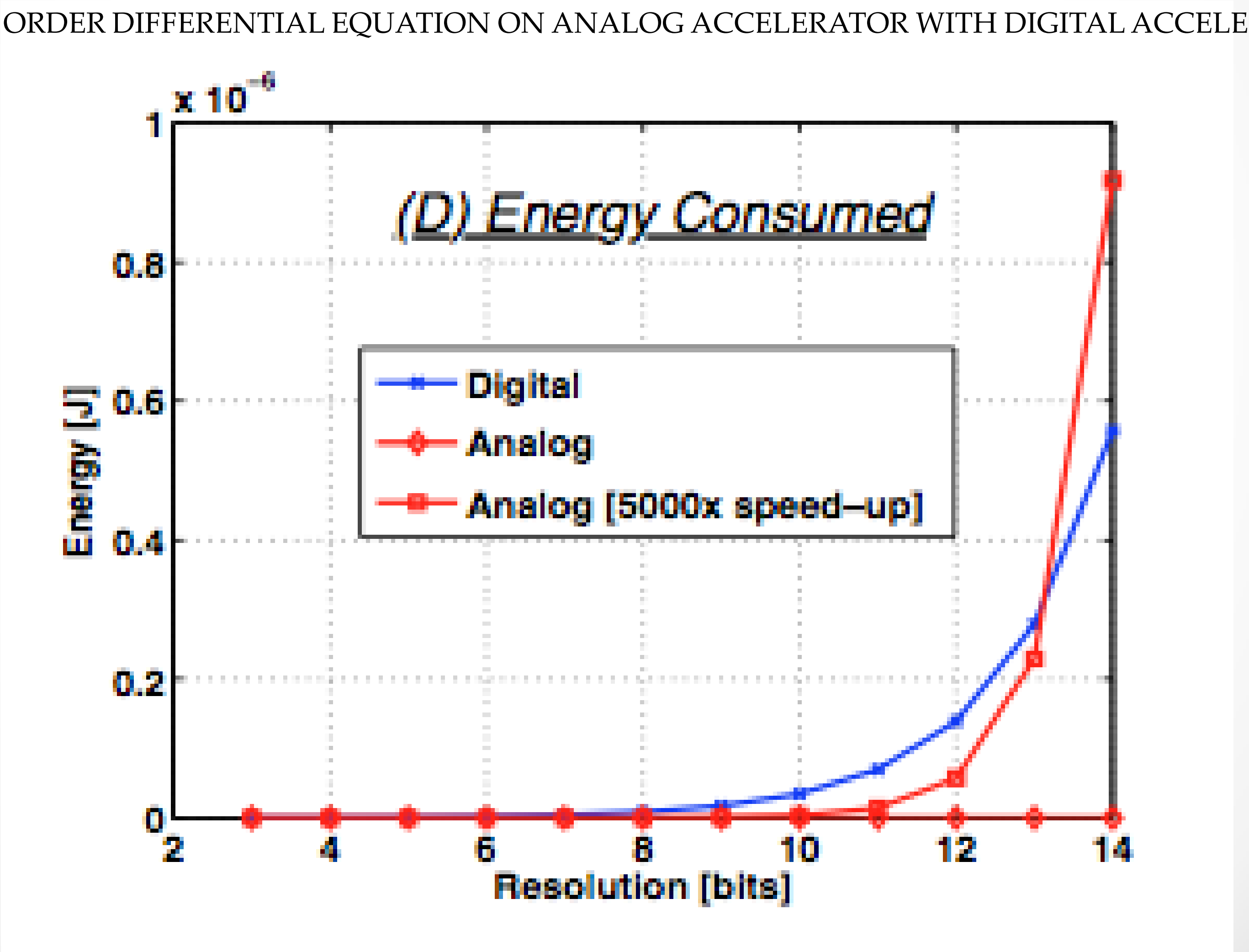
- Want: Make approximate computing practical
  1. Exact output w/ approximate HW (overclock but check)
  2. Approximate output w/ deterministic HW (unsound SW transformations)
  3. Approximate output w/ approximate HW (even analog)

  - Programming languages & tools for all the above

- Apps: machine learning, image/vision, graph proc., big data, security/privacy, estimation, continuous problems

Approximate Computing Example

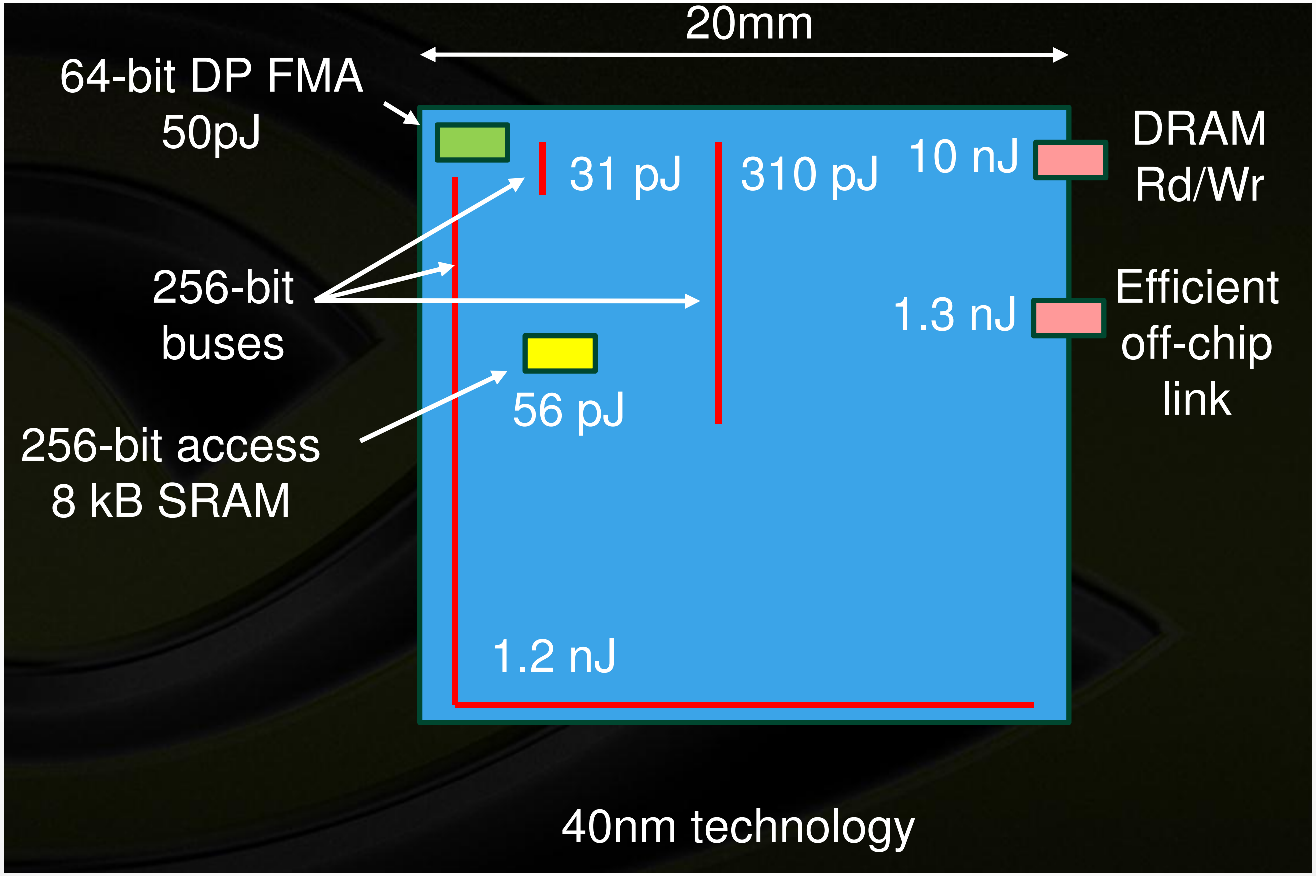
SECOND ORDER DIFFERENTIAL EQUATION ON ANALOG ACCELERATOR WITH DIGITAL ACCELERATOR.



Approved for Public Release, Distribution Unlimited

Locality-aware Parallelism

- Now: Seek (vast) parallelism
  - e.g., simple, energy efficient cores
- But remote communication >100x cost of compute



Approved for Public Release, Distribution Unlimited

Want: Locality-aware Parallelism

- Abstractions & languages for expressing locality
  - E.g., places in X10, locales in Chapel, producer-consumer, …
- Tools for locality optimization
  - Locality-aware mapping/management
  - Data dependent execution
- Tools that balance locality & specialization
- Architectural support for locality

The (Surprise) Delta

- **Can Harvest in the "Fallow" Period!**

HW/SW Specialization/Co-design (3-100x)

Reduce SW Bloat (2-1000x)

Approximate Computing (2-500x)

~1000x = 2 decades of Moore's Law!

- **Systems must exploit LOCALITY-AWARE parallelism**
  - **As communication's energy costs dominate**
  - **10x to 100x over naïve parallelism**

The DoD Impact

Continued computer systems efficiency scaling to:

1. Real time query support [from the cloud] to troops [squads] on the ground
2. Real time social network analysis
3. Real time tracking of targets & activities.
4. Improved cyber defense
5. In-situ sensor data pre-processing before comm.

As well as many civilian benefits

Backup

HW/SW Specialization & Codesign

- So far, GPP good enough due to CMOS scaling
  - Specialization only for highly constrained, high volume apps
- Challenge: specialization at low NRE cost
  - Tools and interfaces for specialization and co-design
    - Application modeling tools, domain specific languages
    - Static/dynamic mapping tools, HW customization tools
  - Hardware technology for specialization
    - Rapid design tools, next-gen reconfigurable hardware, memory specialization, …
  - Software technology for specialization
    - We can use effectively what we build?
  - Power and energy management as co-design
  - Key apps for specialization
    - Big data, security, mobile systems, ML/AI on UAV systems, …

Approved for Public Release, Distribution Unlimited

Spectrum of Hardware Specialization

Metric	Ops/mm²	Ops/Watt	Time to Soln	NRE
GPP	1	1	1 (programming GPP)	1
Specialized ISA (domain specific)	1.5	3-5	2-3 (designing & programming)	1.5
Progr. Accelerator (domain specific)	3	5-10	2-3 (designing & programming)	2-3
Fixed Accelerator (app specific)	5-10	10	10 (SoC design)	3-5
Specialized Mem & Interconnect (monolithic die)	10	10	10 (SoC design)	10
Package level integration (multi die: logic,mem,analog)	10+	10+	5 (silicon interposer)	5

Reduce SW Bloat

- So far, we have focused on improving SW productivity
  - The success: can launch complex, online services within days
  - The price: bloated SW stacks, no understanding of efficiency
  - Example: 50,000x gap between PHP and BLAS Parallel
- Challenge: improve efficiency wo/ sacrificing productivity
  - Abstractions for SW efficiency
    - E.g., software weight as a constraint and optimization metric
  - Performance-aware programming languages
    - Capture key info for efficiency optimizations
  - Tools for performance optimization
    - Compilers, runtime systems, debuggers
    - Dynamic optimization and specialization based on usage
    - Techniques for composition, isolation, performance predictability
  - Learn from clean-slate approaches, enhance existing base

SW Bloat Example: MxM

PHP	9,298,440 ms	51,090x
Python	6,145,070 ms	33,764x
Java	348,749 ms	1816x
C	19,564 ms	107x
Tiled C	12,887 ms	71x
Vectorized	6,607 ms	36x
BLAS Parallel	182 ms	1

- Can we achieve PHP productivity at BLAS efficiency?

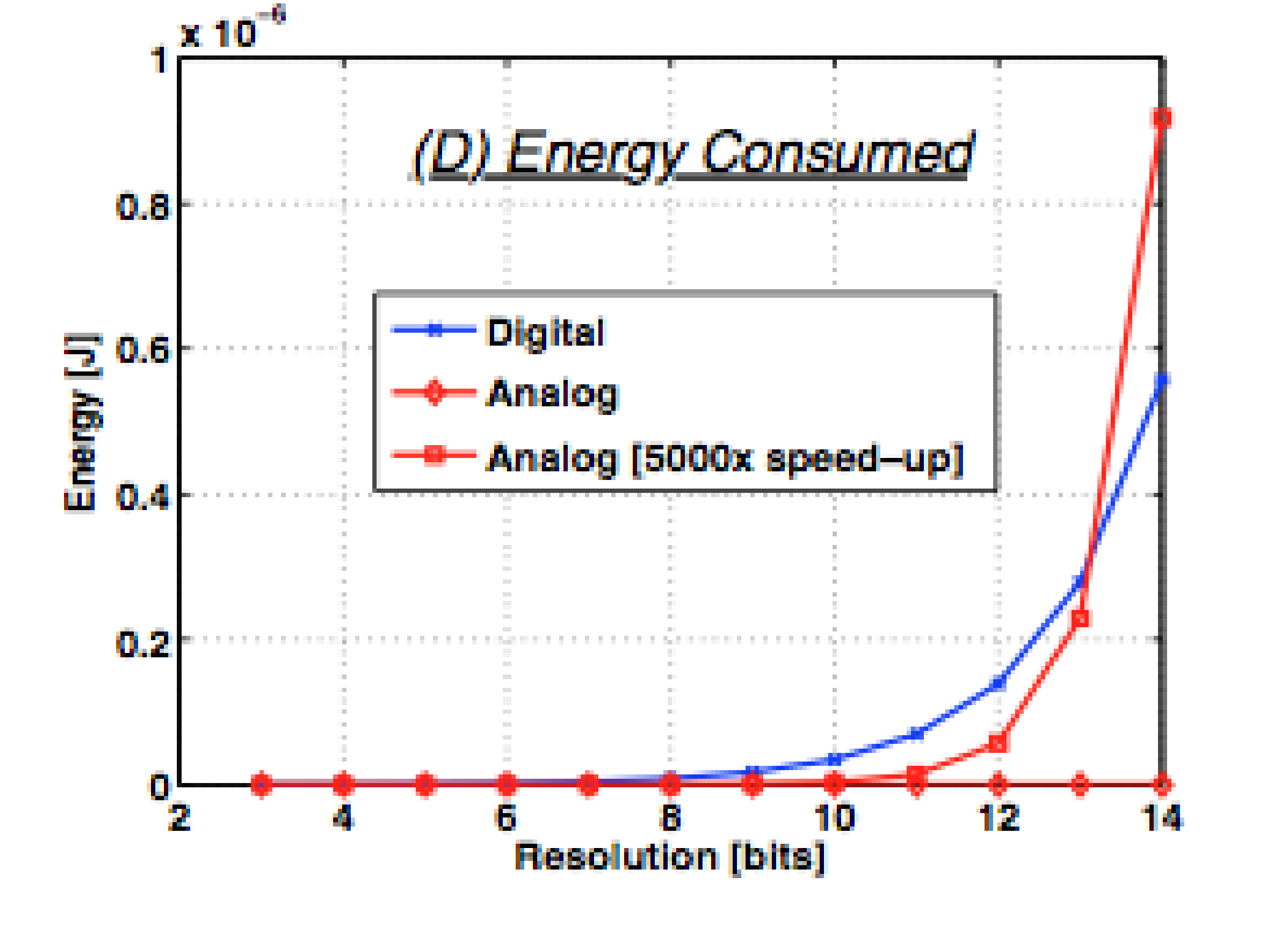
Approximate Computing

- Thus far, expecting exact outputs from deterministic HW
    - But accurate exact outputs not always needed (e.g., AI/ML)
    - Higher HW efficiency if few errors can be tolerated

- Challenge: make approximate computing practical
    - Exact output with approximate HW
        - Analog compute with digital checks, Vdd overscale with resiliency HW
    - Approximate output with deterministic HW
        - Unsound software transformations, Learning-based approximation
    - Approximate output with approximate HW
        - Analog compute, voltage overscale exposed to application, probabilistic circuits, approximate memory and communication
    - Programming languages & tools for approximate computing
        - Management of error propagation, composition, …
    - HW design technique for approximate computing

Approximate Computing Example

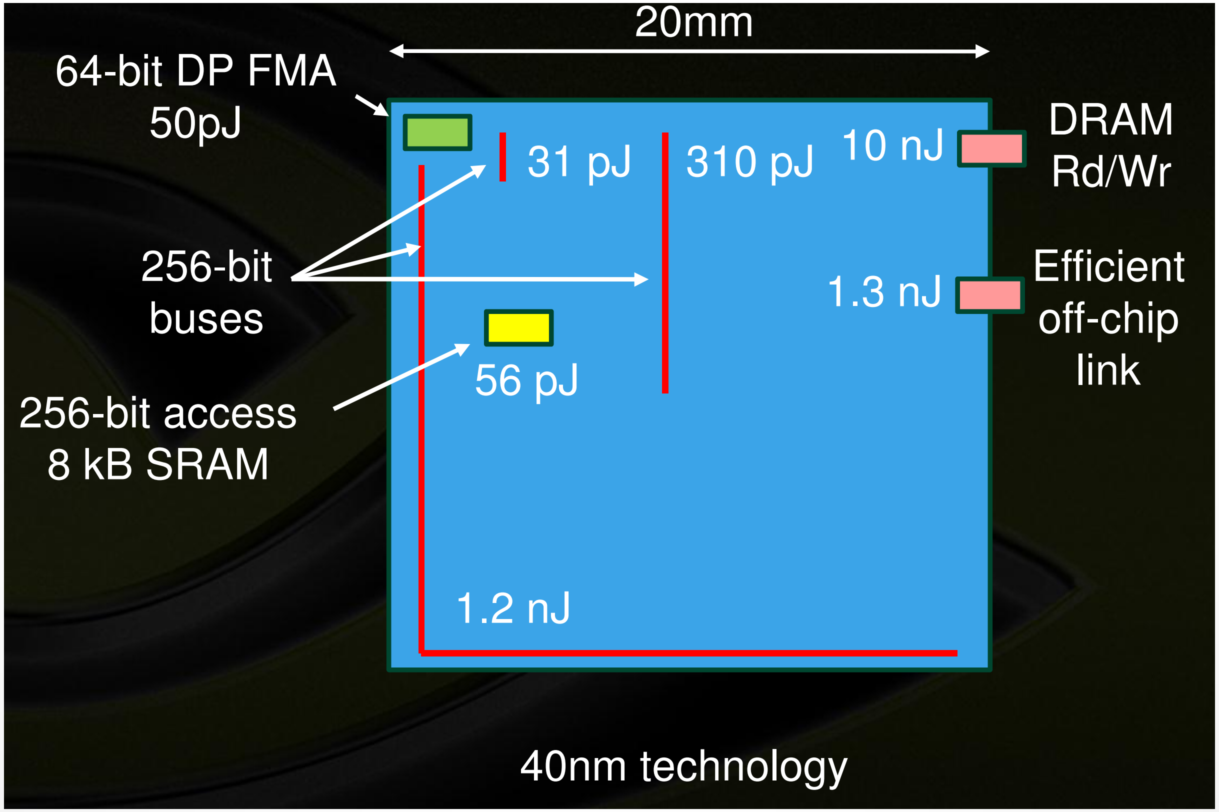
SECOND ORDER DIFFERENTIAL EQUATION ON ANALOG ACCELERATOR WITH DIGITAL ACCELERATOR.



Approved for Public Release, Distribution Unlimited

Locality-aware Parallelism

- Thus far, focus on parallel execution
  - Parallelism enables the use of simple, energy efficient cores
  - But communication latency/energy can cancel parallelism
  - Remote communication >100x cost of compute operations



Approved for Public Release, Distribution Unlimited

Locality-aware Parallelism

- Thus far, focus on parallel execution
    - o Parallelism enables the use of simple, energy efficient cores
    - o But communication latency/energy can cancel parallelism
    - o Remote communication >100x cost of compute operations

- Challenge: parallelism with locality-awareness
    - o Abstractions and languages for expressing locality
        - E.g., places in X10, locales in Chapel, producer-consumer, …
    - o Tools for locality optimization
        - Locality-aware mapping, data dependent execution, locality aware runtime management
    - o Tools that balance locality and specialization
    - o Architectural support for locality

Approved for Public Release, Distribution Unlimited

Participant Feedback 1/2

- 32 Responses
  - % Strongly Like/Like/Neutral/Dislike/Strongly Dislike

- Overall regarding workshop (47%/38%/13%/0/0/0)
- Position statements (34/56/6/0/0)
- Keynotes
  - Colwell (56/34/6/0/0/0) Larus (28/41/19/9/0/0)
- Breakouts
  - Assigned (25/41/28/3/0/0), Self-Organized (19/38/28/6/0/0)
  - Self-Org useful? [Y78 neut19 N0], Ok time? [Y84 neut6 N6]
- Speed-dating
  - Like 2nd day? (59/34/3/0/0/0), Move to 1st day? [Y66 neut19 N13], Do twice? [Y56, neut25 N25]

Participant Feedback 2/2

1. Other aspects of workshop you particularly liked?
   - People! (! = repeated), venue!, format!, discussion time!
2. Other aspects you particularly disliked?
   - Too big!, ignorance of parallel computing, too few SW people
3. Other ideas we should try in the workshop?
   - Need multiple meetings!, wild&crazy, recurring event
4. Any other comments?
   - More time for deep results!, clearer goals, good work!, more DoD people
5. Any suggestions of topics for future ISAT workshops?
   - Productive, portable parallel programming; autonomous agents; post-Moore's Law for big data; robust high-frequency trading algorithms; making computer systems less annoying

➔ Raw responses at end of slide deck