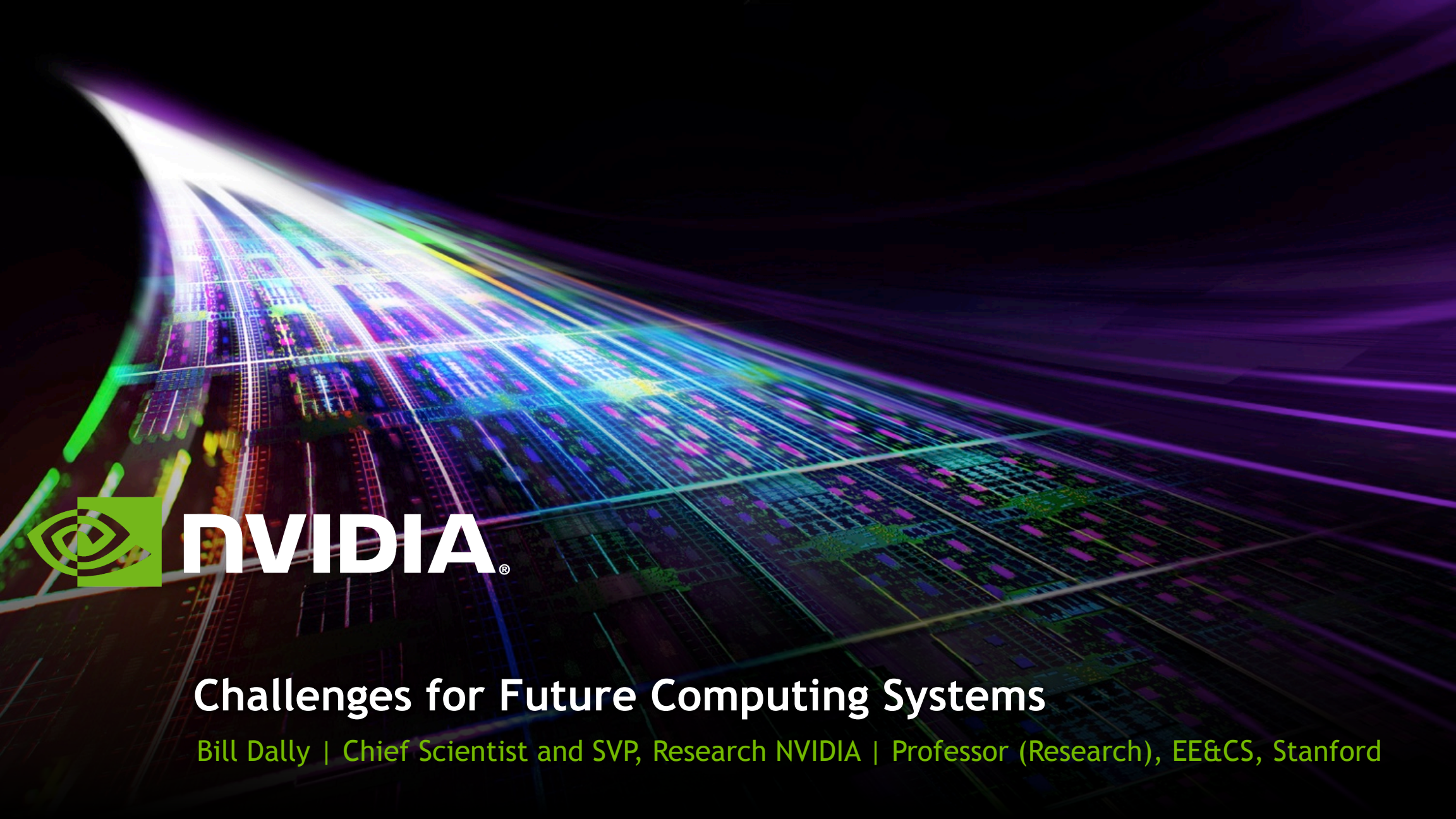


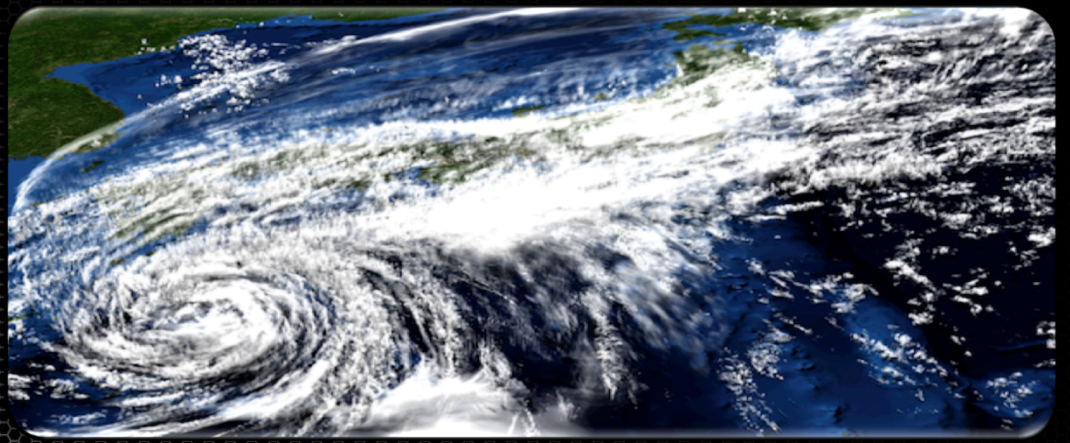
Challenges for Future Computing Systems

Bill Dally | Chief Scientist and SVP, Research NVIDIA | Professor (Research), EE&CS, Stanford

Exascale Computing Will Enable Transformational Science

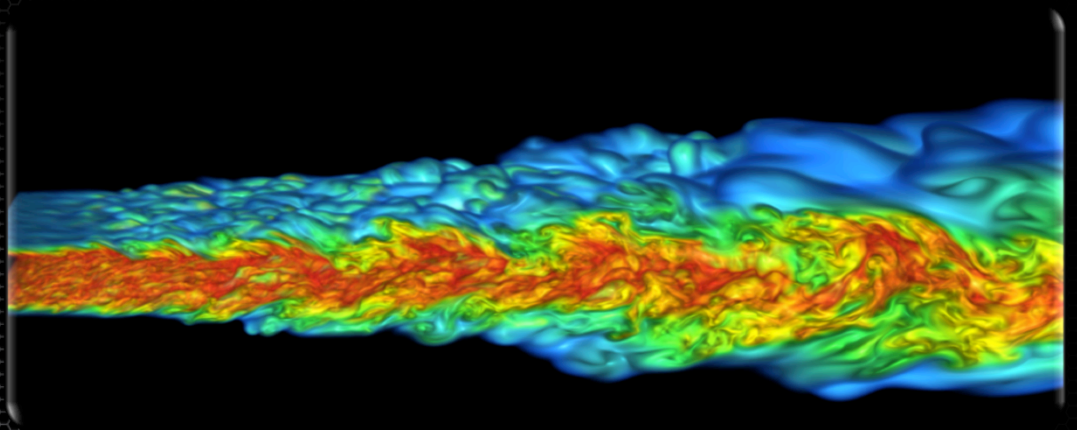
Climate

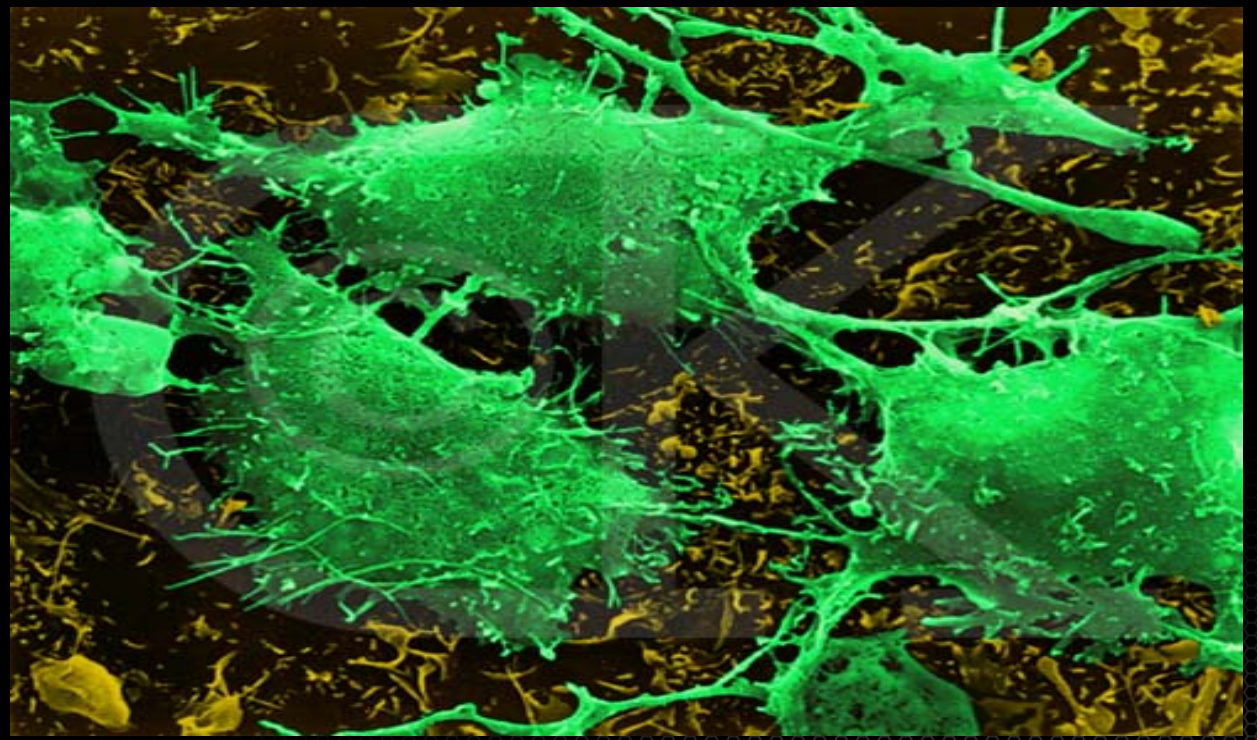
Comprehensive Earth System Model at 1KM scale, enabling modeling of cloud convection and ocean eddies.



Combustion

First-principles simulation of combustion for new high-efficiency, low-emision engines.





Coupled simulation of entire cells at molecular, genetic, chemical and biological levels.

Biology

Predictive calculations for thermonuclear and core-collapse supernovae, allowing confirmation of theoretical models.

Astrophysics

Exascale Computing Will Enable Transformational Science

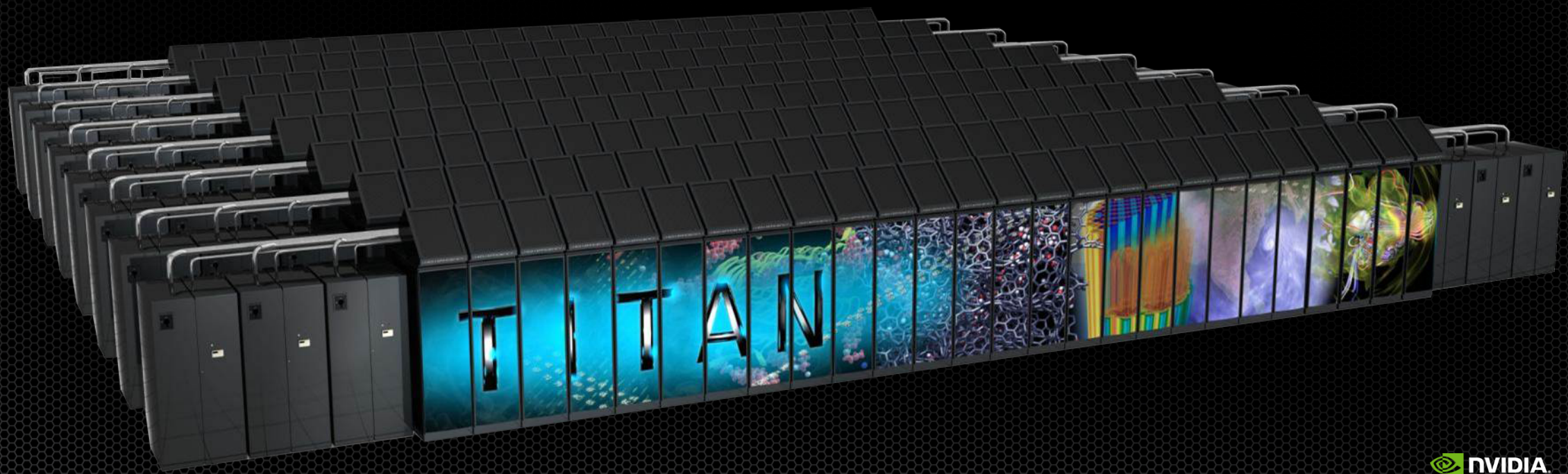
High-Performance Computers are Scientific Instruments

TITAN

18,688 NVIDIA Tesla K20X GPUs

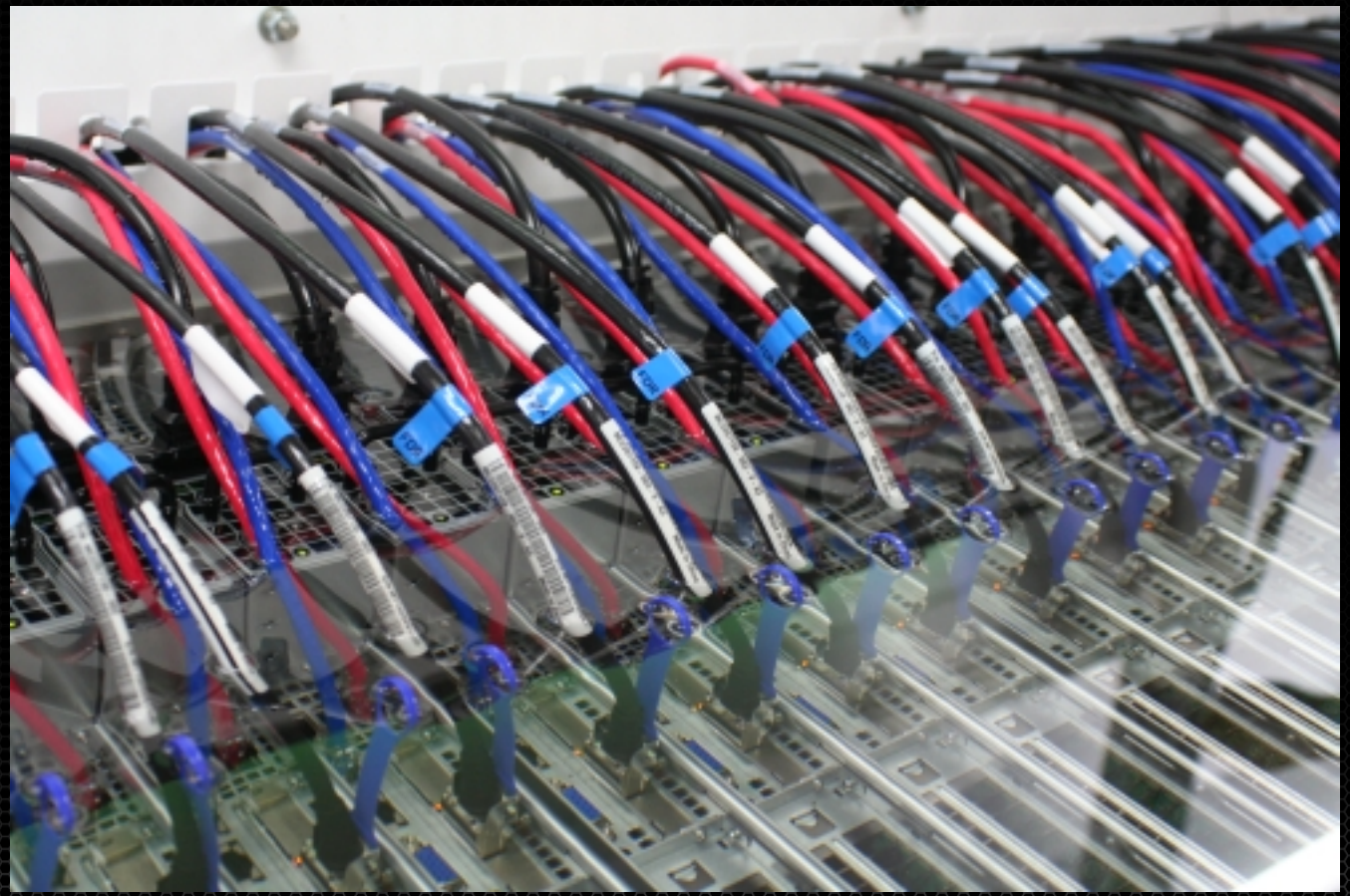
27 Petaflops Peak: 90% of Performance from GPUs

17.59 Petaflops Sustained Performance on Linpack



NVIDIA

Tsubame KFC 4.5GFLOPS/W
#1 on Green500 List



US to Build Two Flagship Supercomputers

OAK RIDGE National Laboratory

Lawrence Livermore National Laboratory

SUMMIT

SIERRA

150-300 PFLOPS Peak Performance

IBM POWER9 CPU + NVIDIA Volta GPU

NVLink High Speed Interconnect

40 TFLOPS per Node, >3,400 Nodes

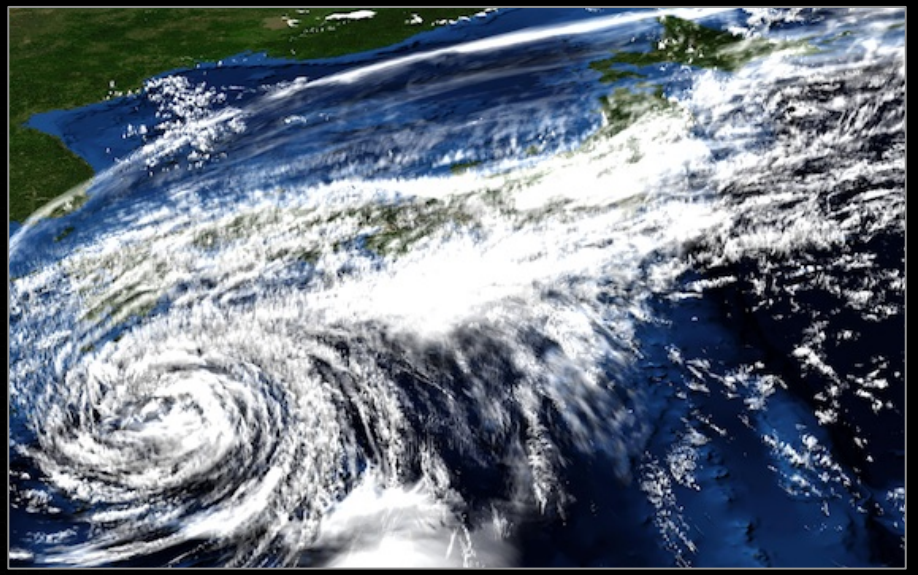
2017

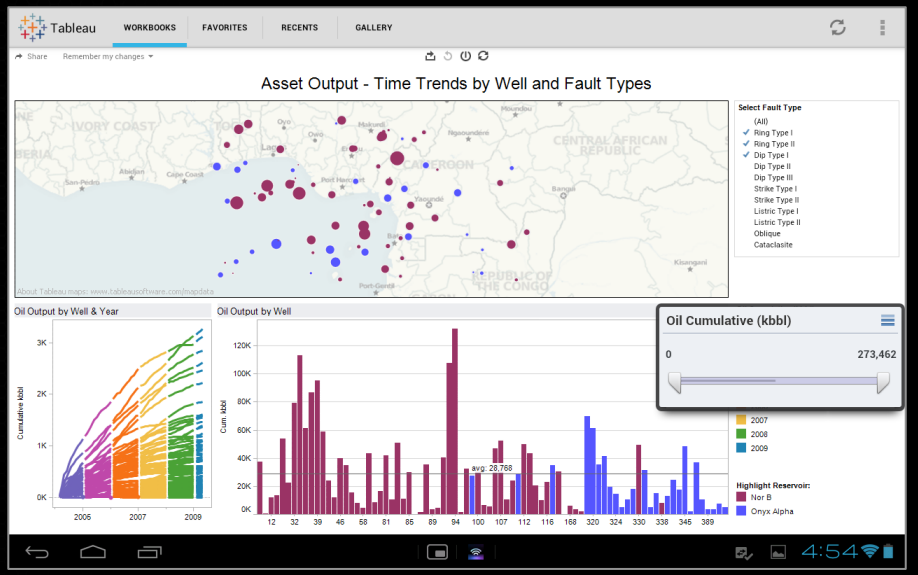
Major Step Forward on the Path to Exascale

NVIDIA

Scientific Discovery and Business Analytics

Driving an Insatiable Demand for More Computing Performance





NVIDIA


htc
10:08 AM
AM 10 08
Sun, Feb 26
Barcelona
Mostly Sunny
12°C
16°C / 8°C
Play Store
Favorites

NVIDIA.

2 Trends

The End of Dennard Scaling

Pervasive Parallelism

2 Trends

The End of Dennard Scaling

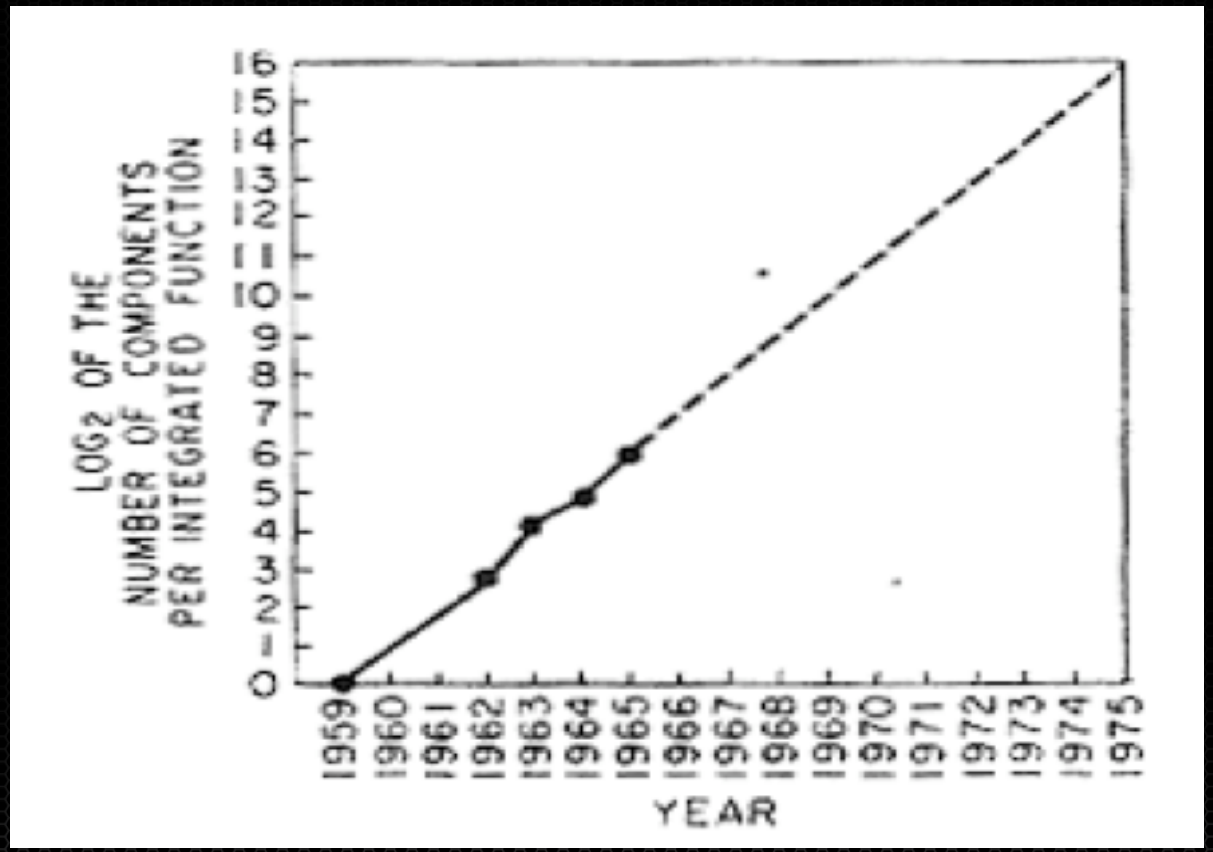
Pervasive Parallelism

Moore’s Law doubles processor performance every few years, right?

Moore’s Law doubles processor performance every few years, right?

Wrong!

Moore's Law gives us transistors Which we used to turn into scalar performance



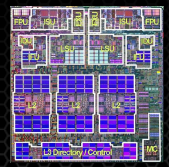
Moore, Electronics 38(8) April 19, 1965

NVIDIA

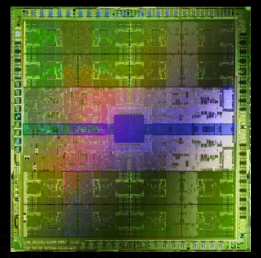
1997: 7.5M Tx

2001: 42M Tx

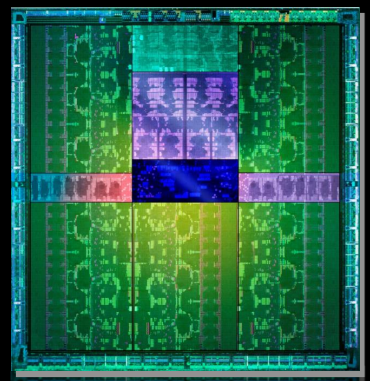
2004: 275M Tx



2007: 580M Tx



2010: 3B Tx



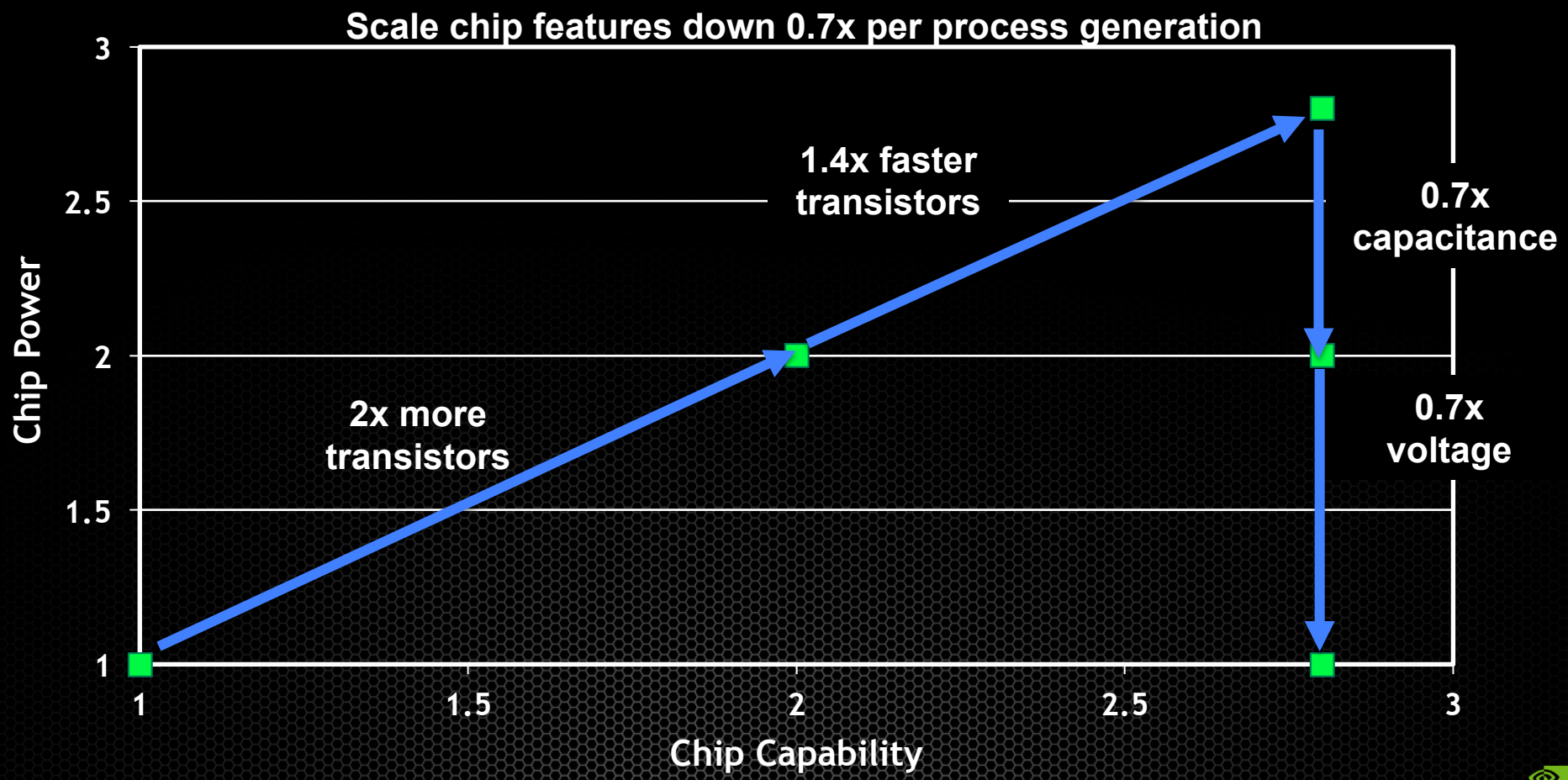
2012: 7B Tx

1993: 3M Tx

Moore's Law is Only Part of the Story

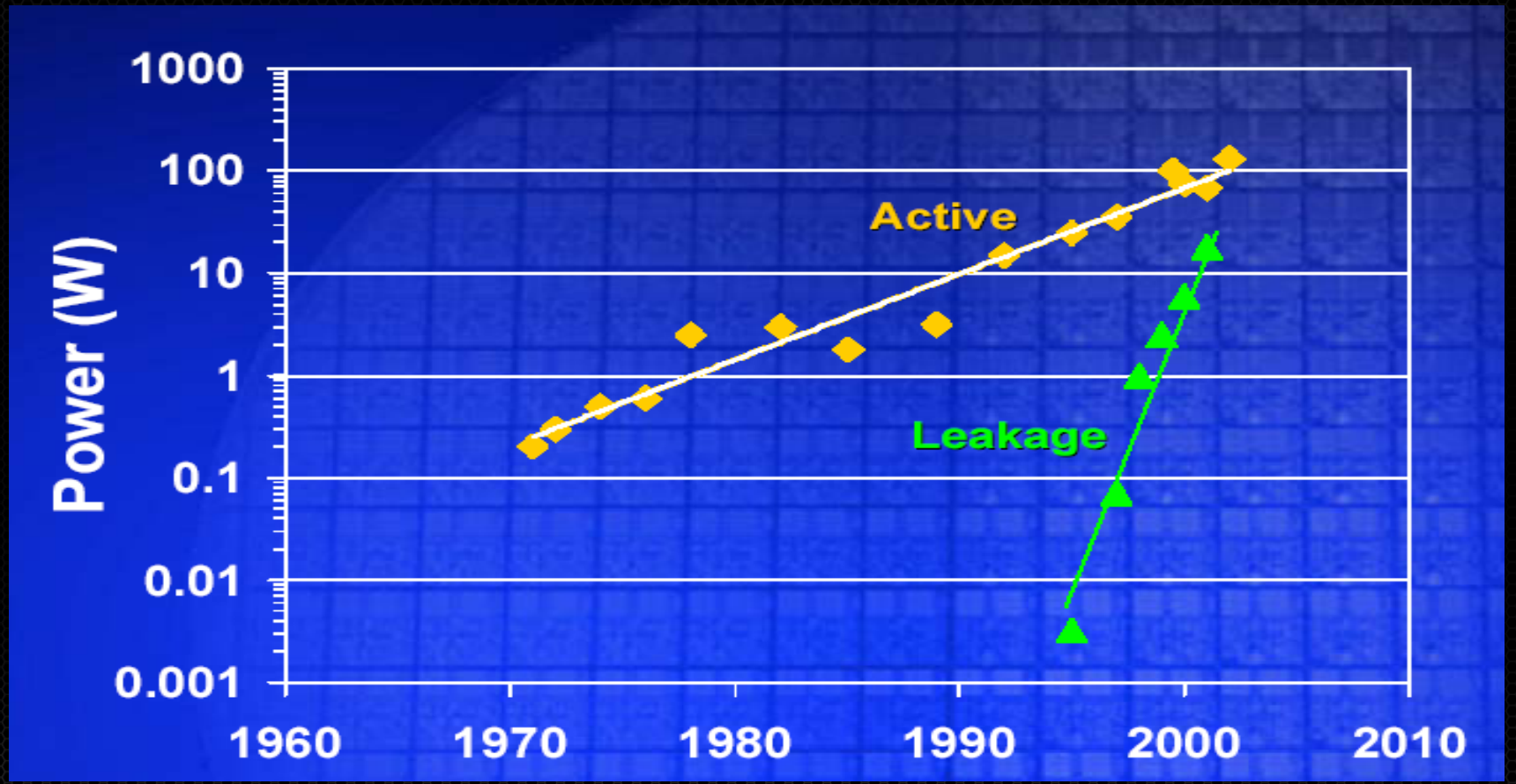
Classic Dennard Scaling

2.8x chip capability in same power



NVIDIA

But L^3 energy scaling ended in 2005

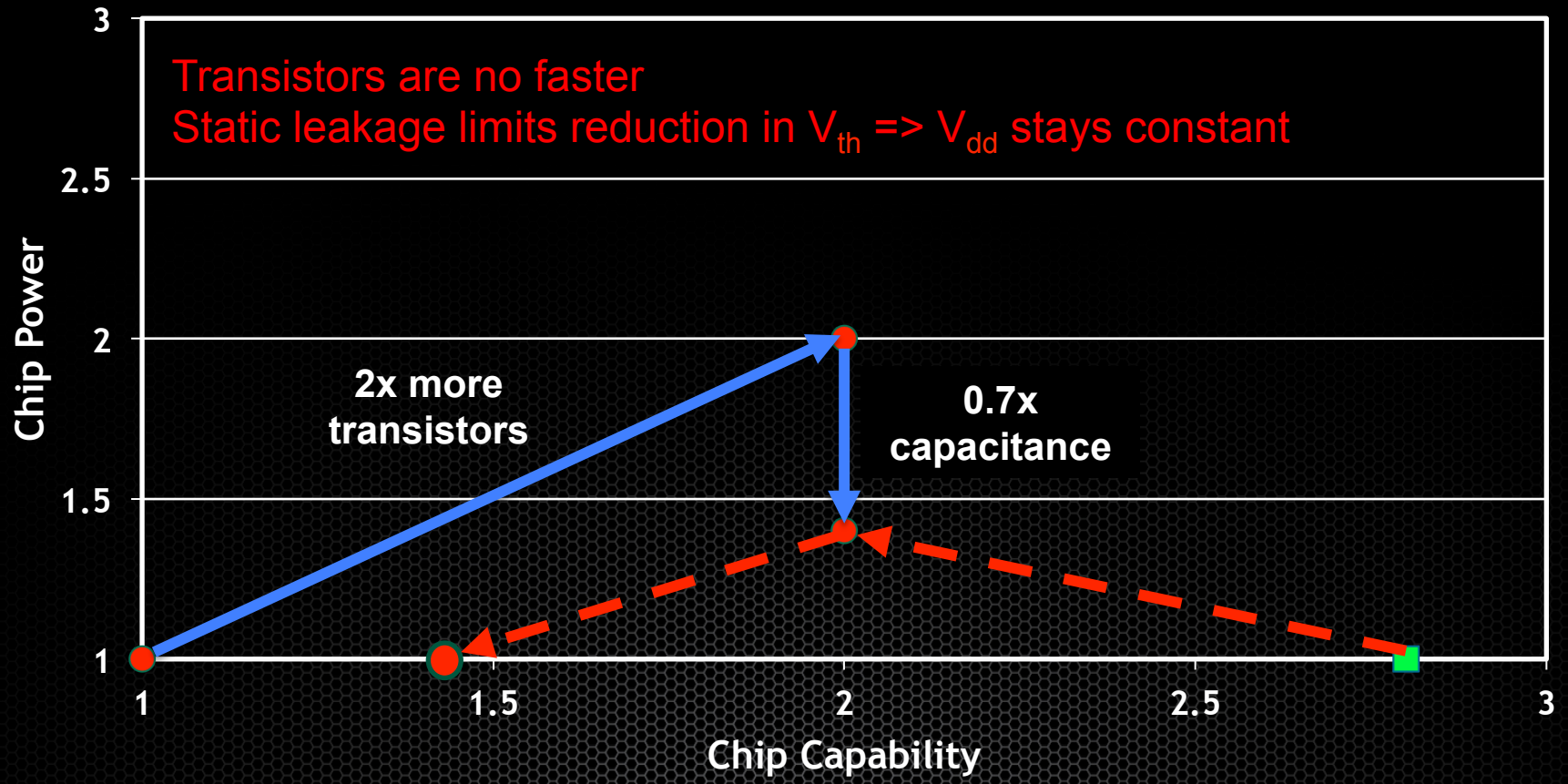


Moore, ISSCC Keynote, 2003

NVIDIA

Post Dennard Scaling

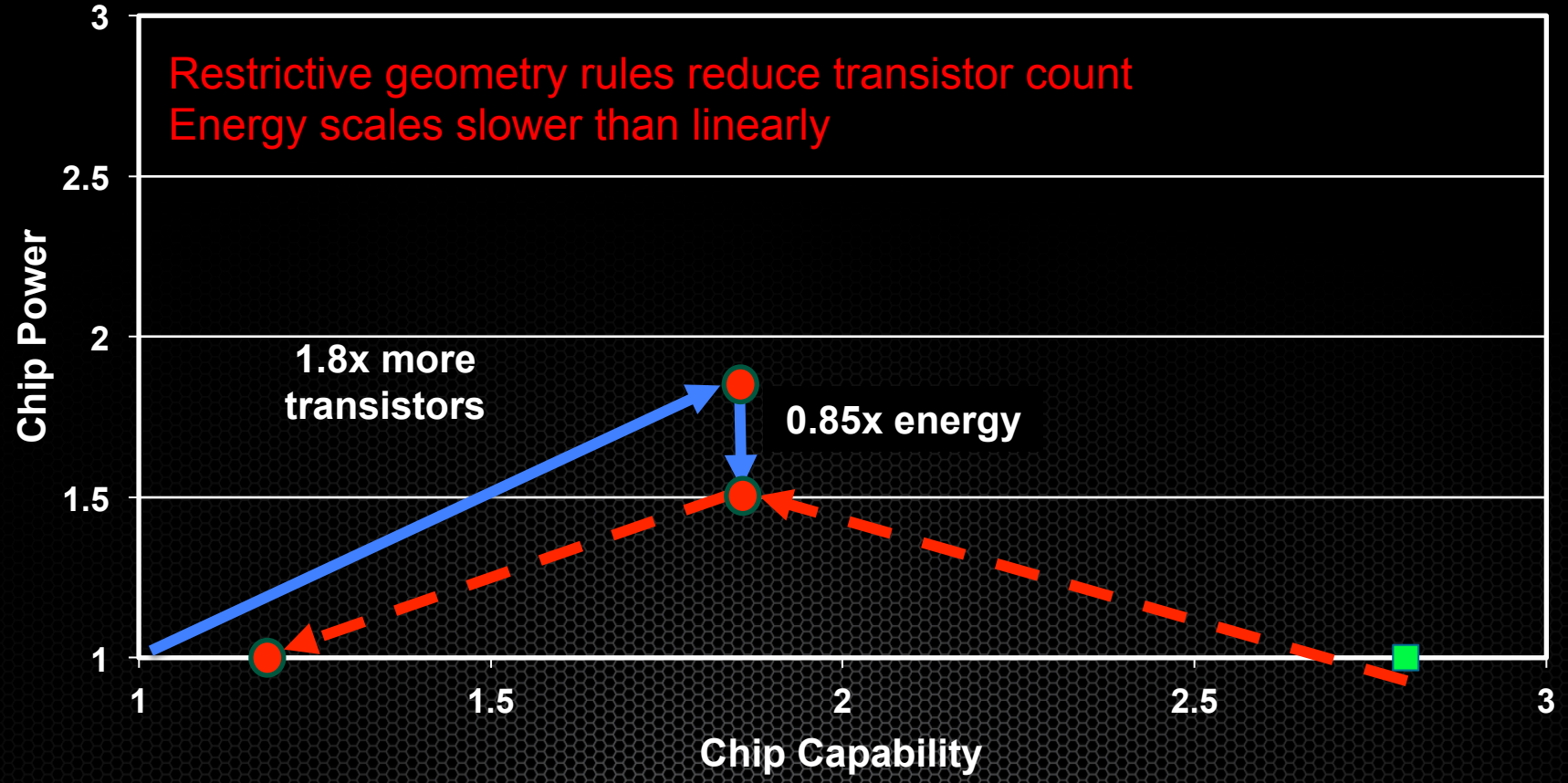
2x chip capability at 1.4x power
1.4x chip capability at same power



NVIDIA

Reality isn't even this good

1.8x chip capability at 1.5x power
1.2x chip capability at same power

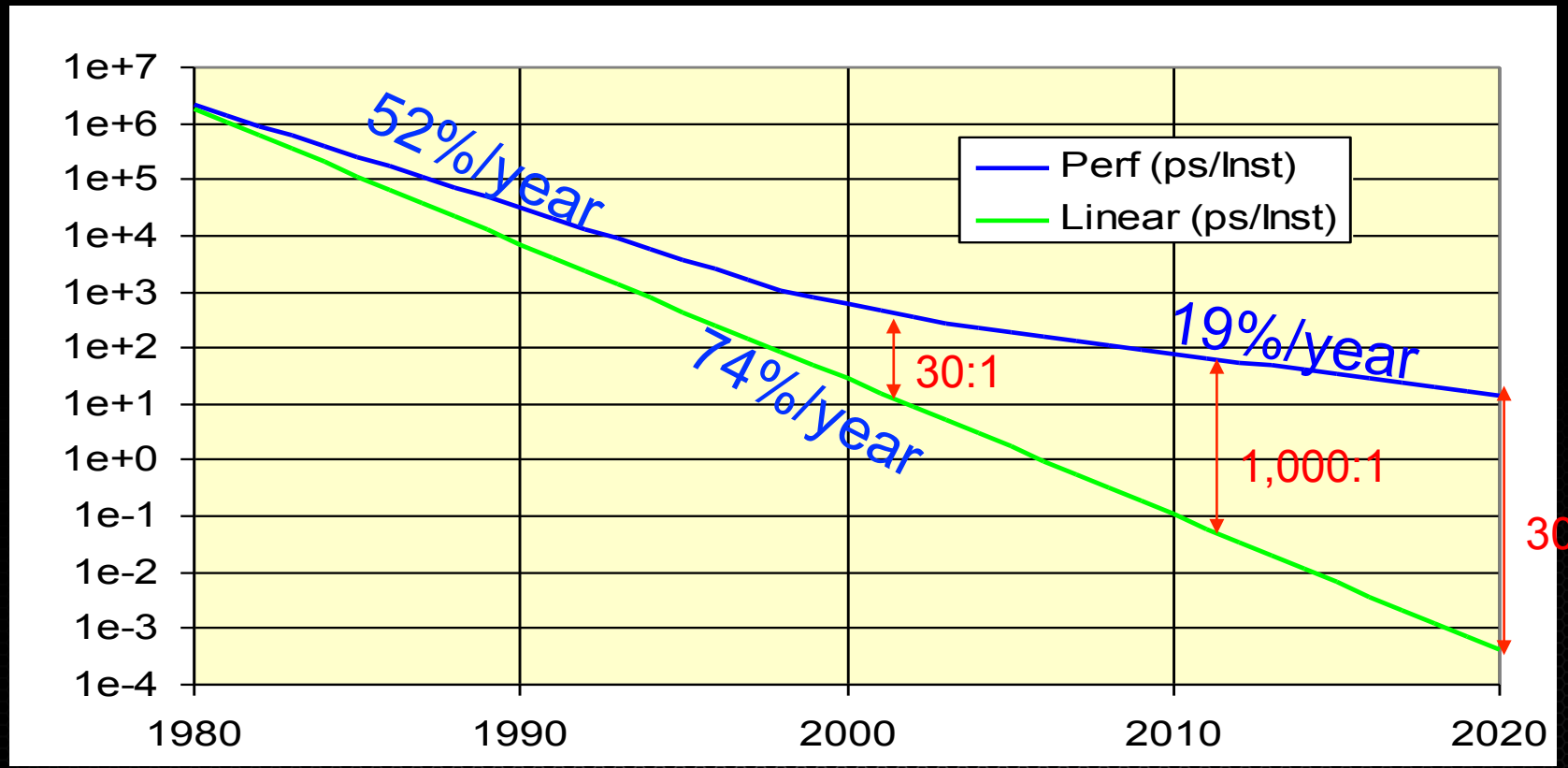


NVIDIA

“Moore’s Law gives us more transistors…
Dennard scaling made them useful.”

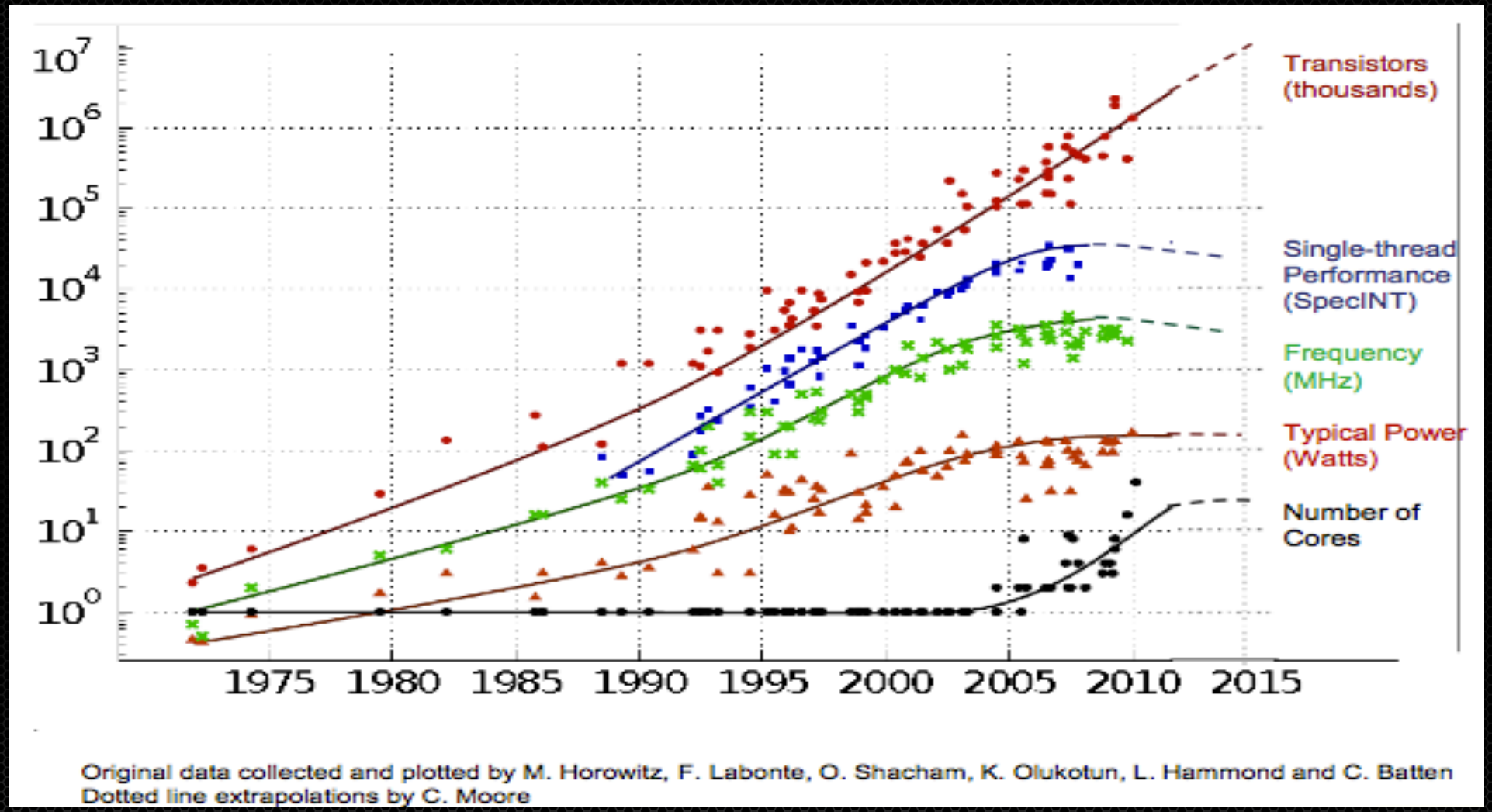
Bob Colwell, DAC 2013, June 4, 2013

Also, ILP was 'mined out' in 2000



Dally et al. "The Last Classical Computer", ISAT Study, 2001

Result: The End of Historic Scaling



C Moore, *Data Processing in ExaScale-ClassComputer Systems,* Salishan, April 2011

The End of Dennard Scaling

- Processors aren't getting faster, just wider
  - Future gains in performance are from parallelism
- Future systems are energy limited
  - Efficiency ***IS*** Performance
- Process matters less
  - One generation is 1.2x, not 2.8x

Its not about the FLOPs

- DFMA 0.01mm^2 10pJ/OP – 2GFLOPs

16nm chip, 10mm on a side, 200W

A chip with 10^4 FPUs:

100mm^2

200W

20TFLOPS

Pack 50,000 of these in racks

1EFLOPS

10MW

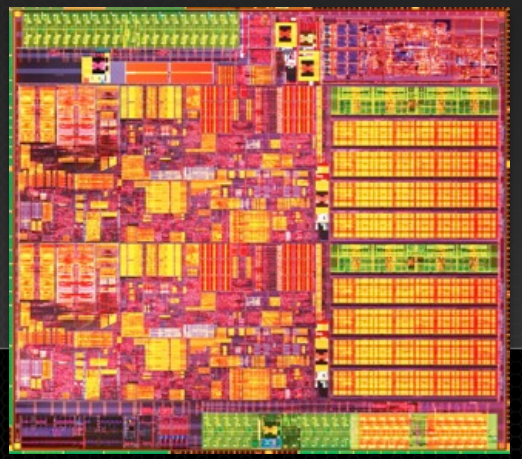
Overhead

Locality

CPU
1690 pJ/flop

Optimized for Latency

Caches

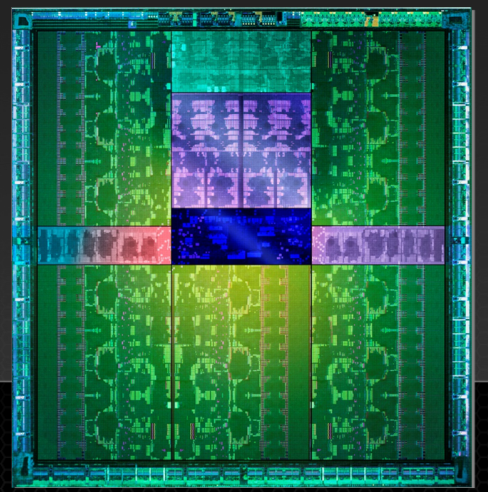


Westmere
32 nm

GPU
140 pJ/flop

Optimized for Throughput

Explicit Management
of On-chip Memory



Kepler
28 nm

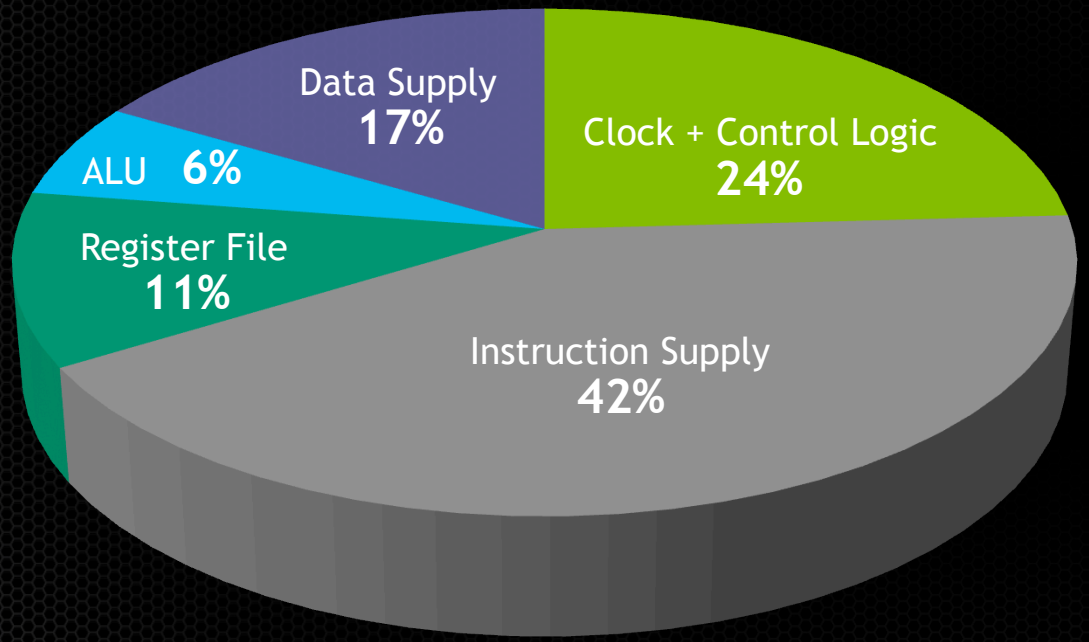
NVIDIA

Fixed-Function Logic is Even More Efficient

	Energy/Op
CPU	1.7nJ
GPU	140pJ
Fixed-Function	10pJ

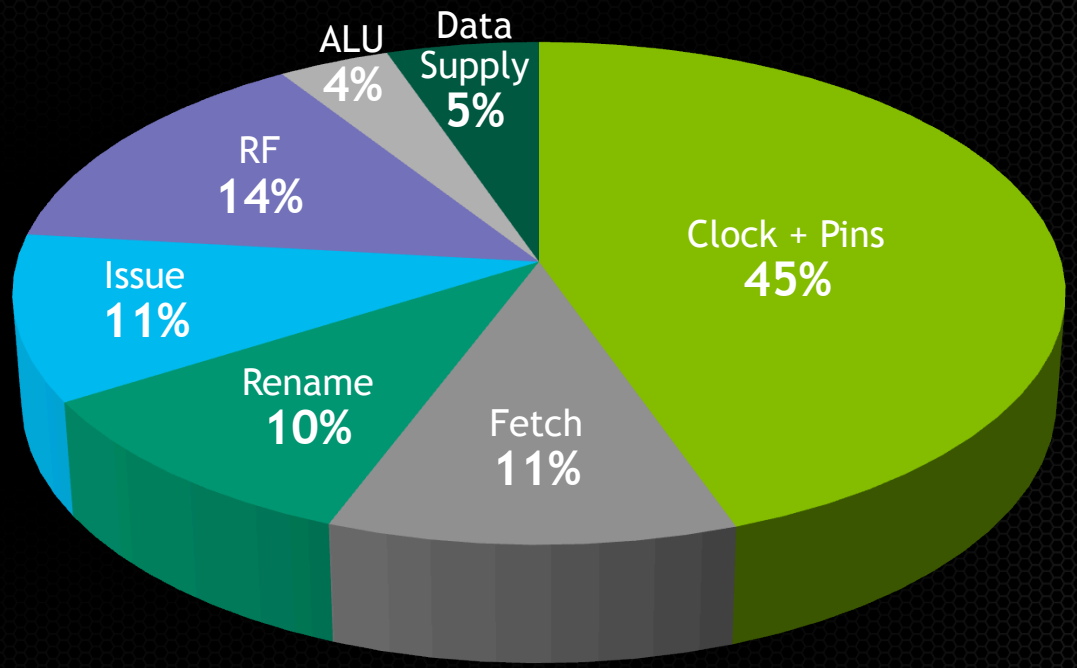
How is Power Spent in a CPU?

In-order Embedded

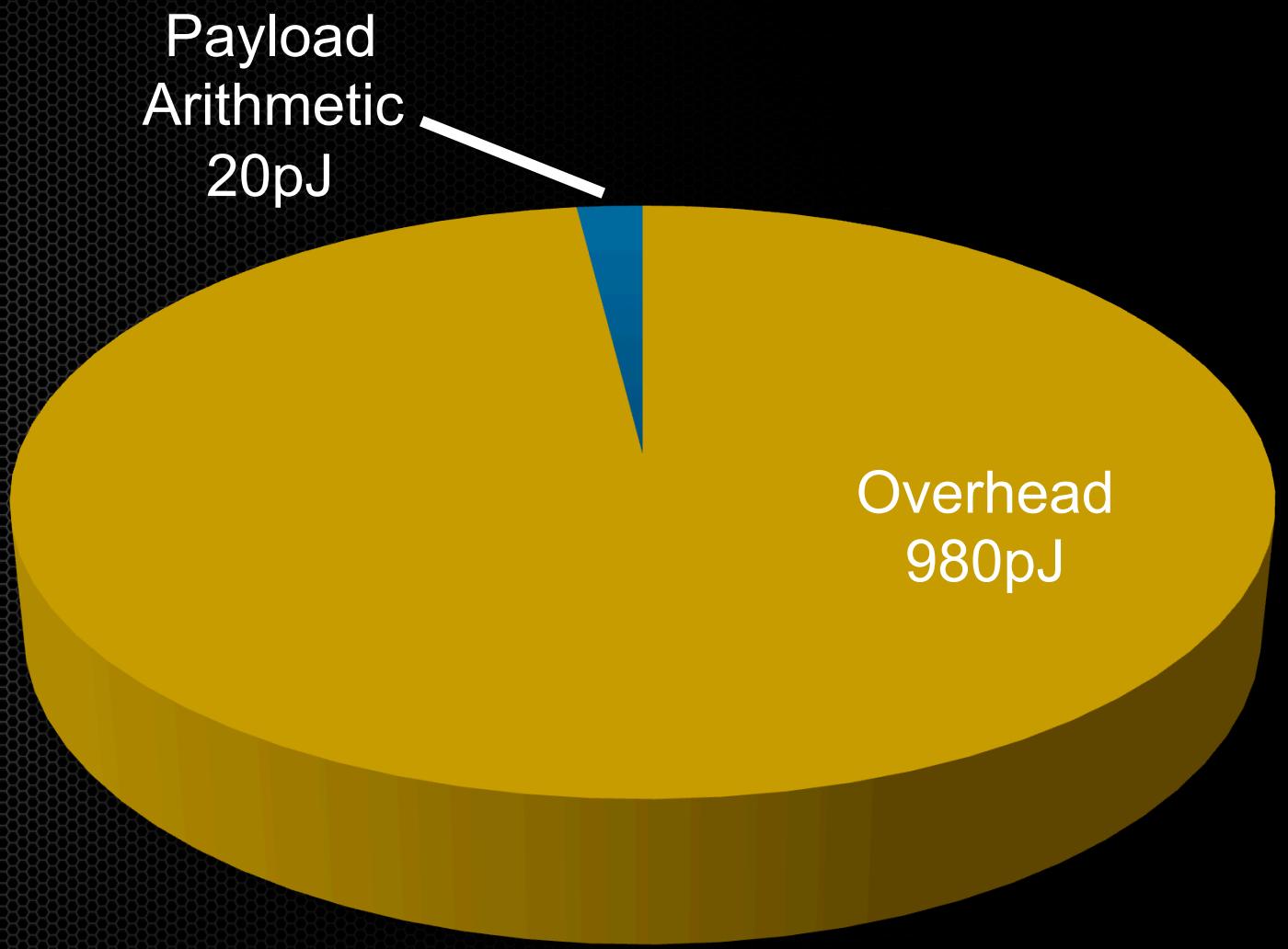


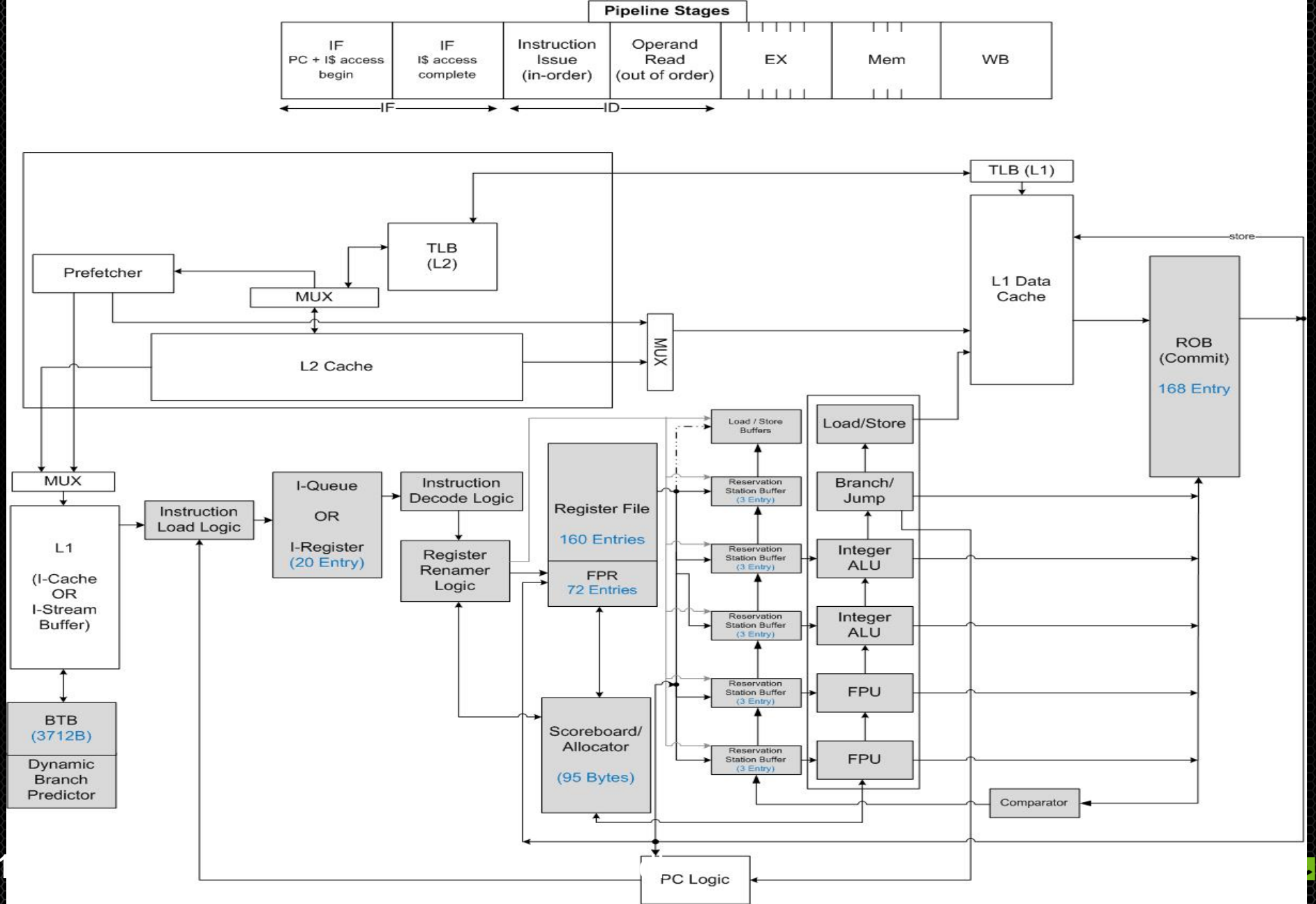
Dally [2008] (Embedded in-order CPU)

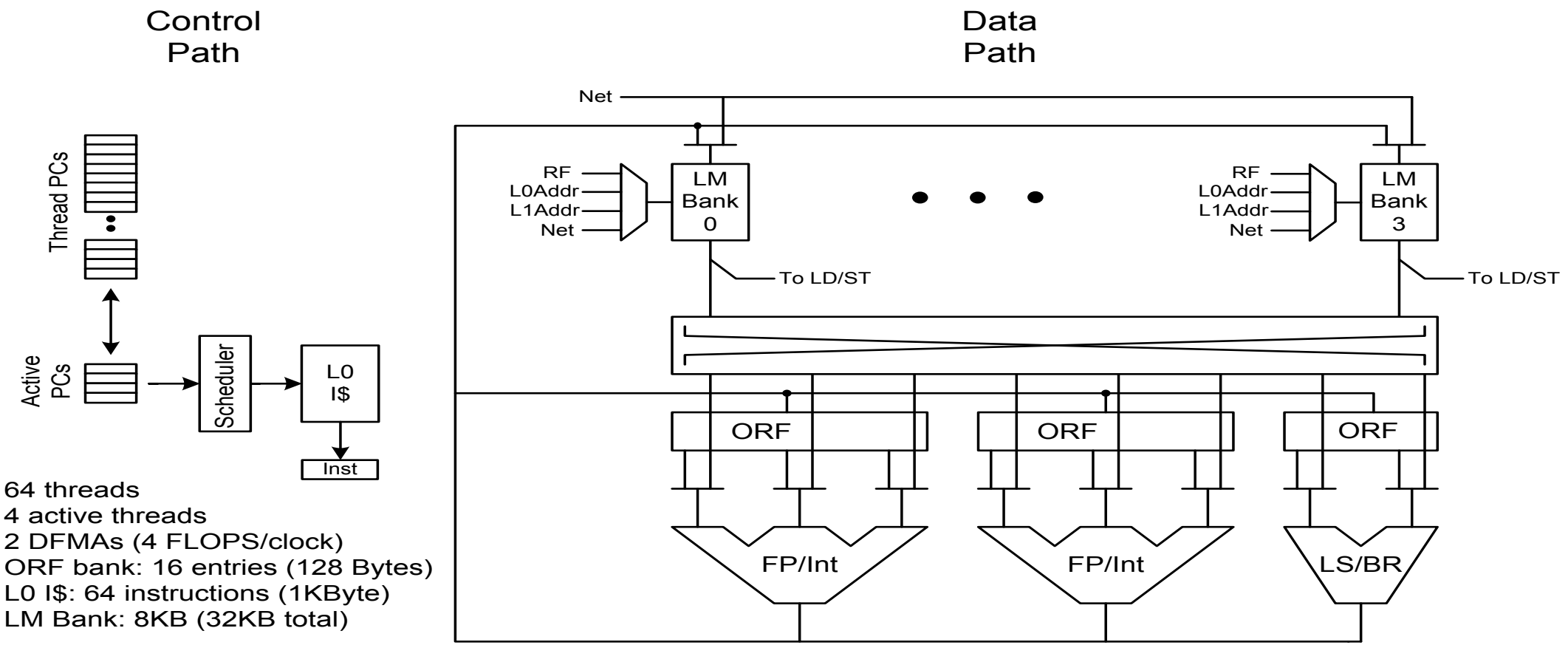
OOO Hi-perf



Natarajan [2003] (Alpha 21264)

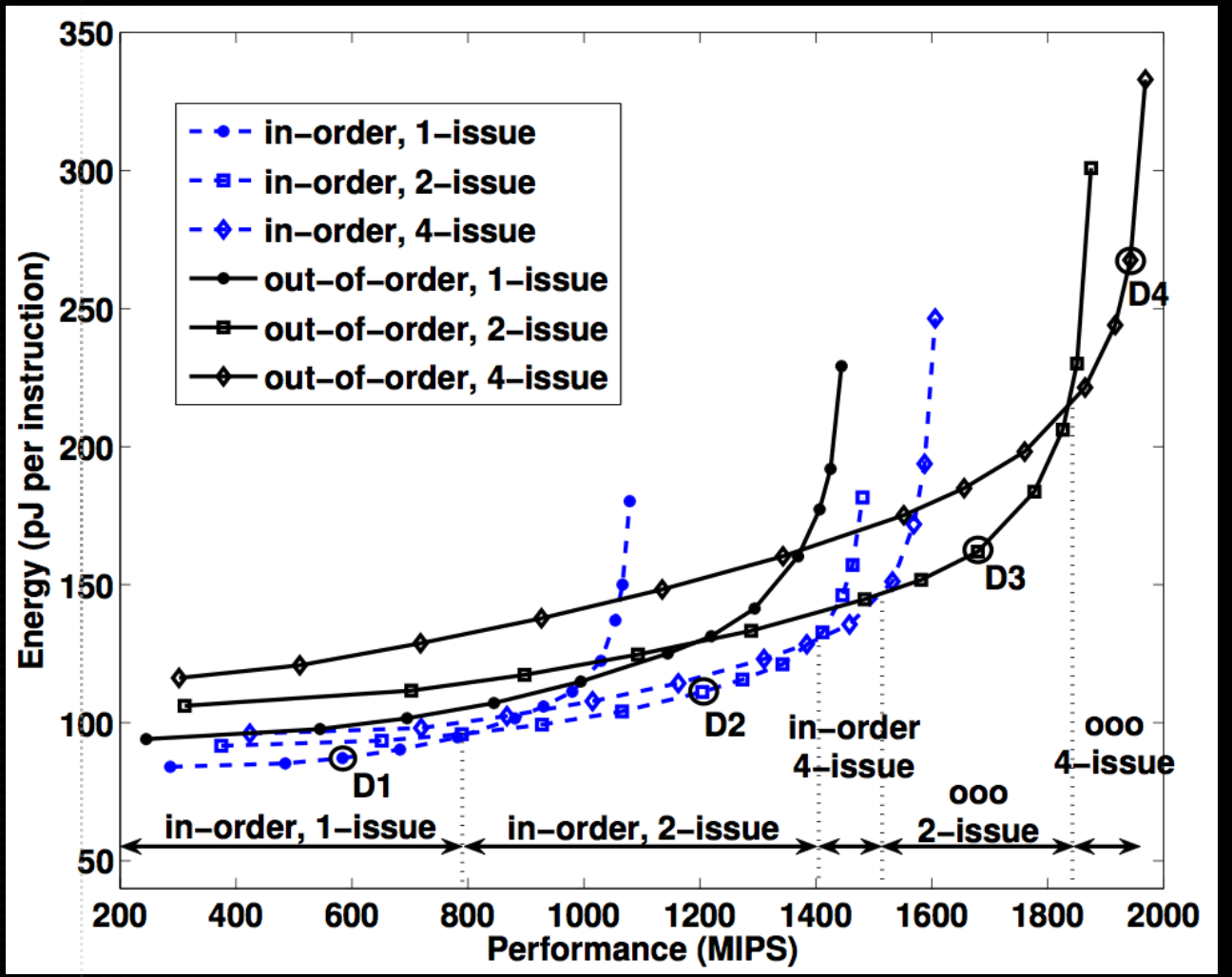



Pipeline Stages
IF
PC + I$ access begin
IF
I$ access complete
Instruction Issue (in-order)
Operand Read (out of order)
EX
Mem
WB
IF
ID
TLB (L1)
TLB (L2)
Prefetcher
MUX
L2 Cache
MUX
L1 Data Cache
store
ROB (Commit)
168 Entry
MUX
L1 (I-Cache OR I-Stream Buffer)
Instruction Load Logic
I-Queue OR I-Register (20 Entry)
Instruction Decode Logic
Register Renamer Logic
Register File
160 Entries
FPR
72 Entries
Load / Store Buffers
Reservation Station Buffer (3 Entry)
Load/Store
Branch/ Jump
Integer ALU
Integer ALU
FPU
FPU
BTB (3712B)
Dynamic Branch Predictor
Scoreboard/ Allocator
(95 Bytes)
Comparator
PC Logic

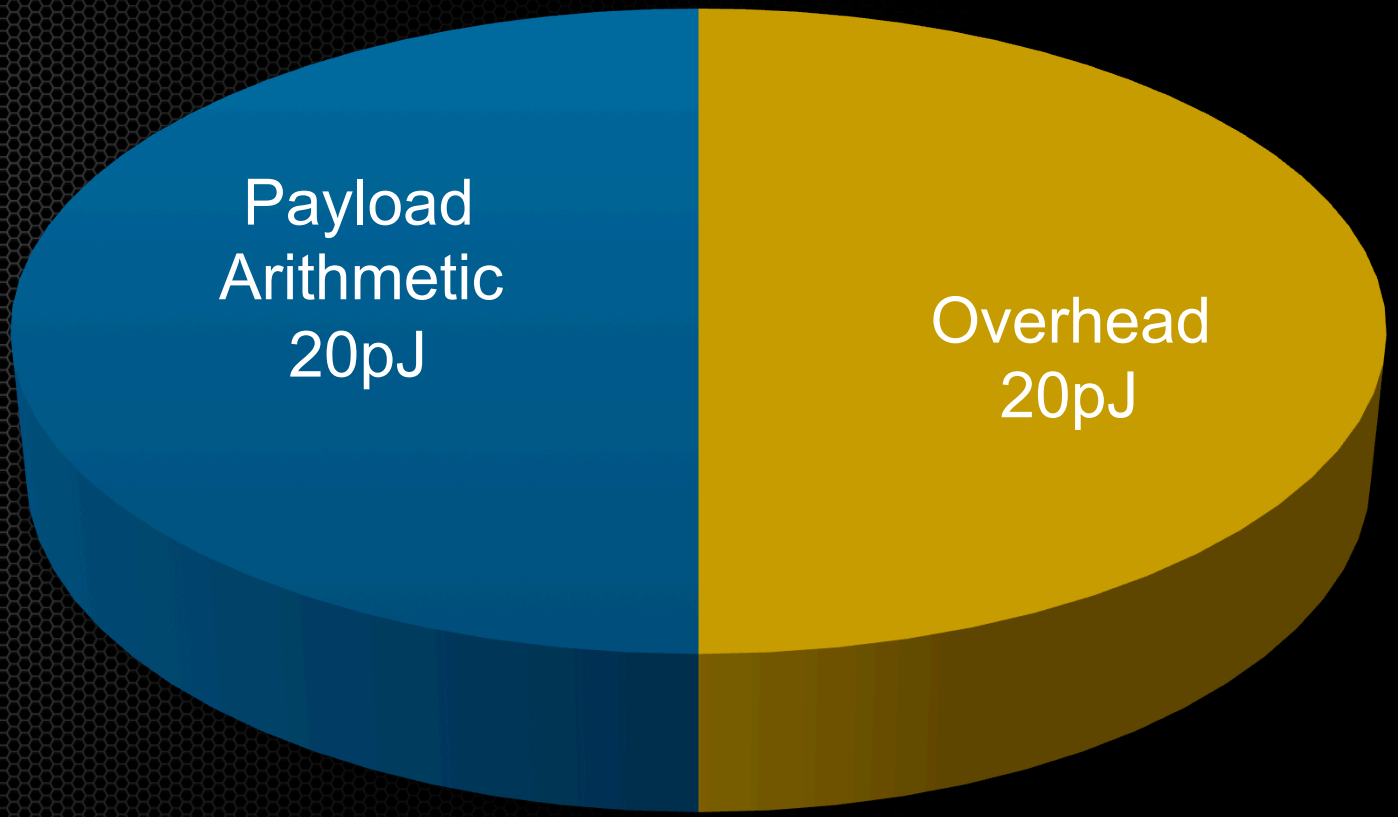


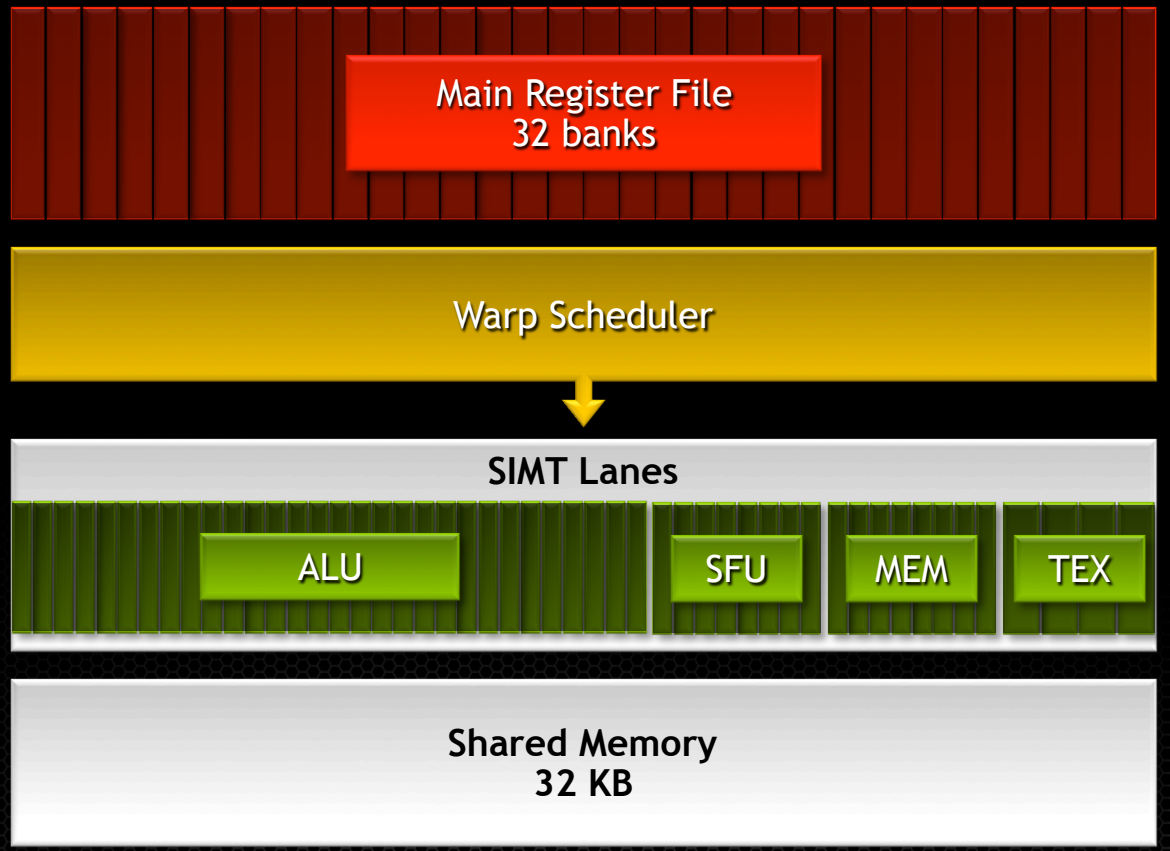
64 threads
4 active threads
2 DFMAs (4 FLOPS/clock)
ORF bank: 16 entries (128 Bytes)
L0 I$: 64 instructions (1KByte)
LM Bank: 8KB (32KB total)

Simpler Cores = Energy Efficiency



Source: Azizi [PhD 2010]

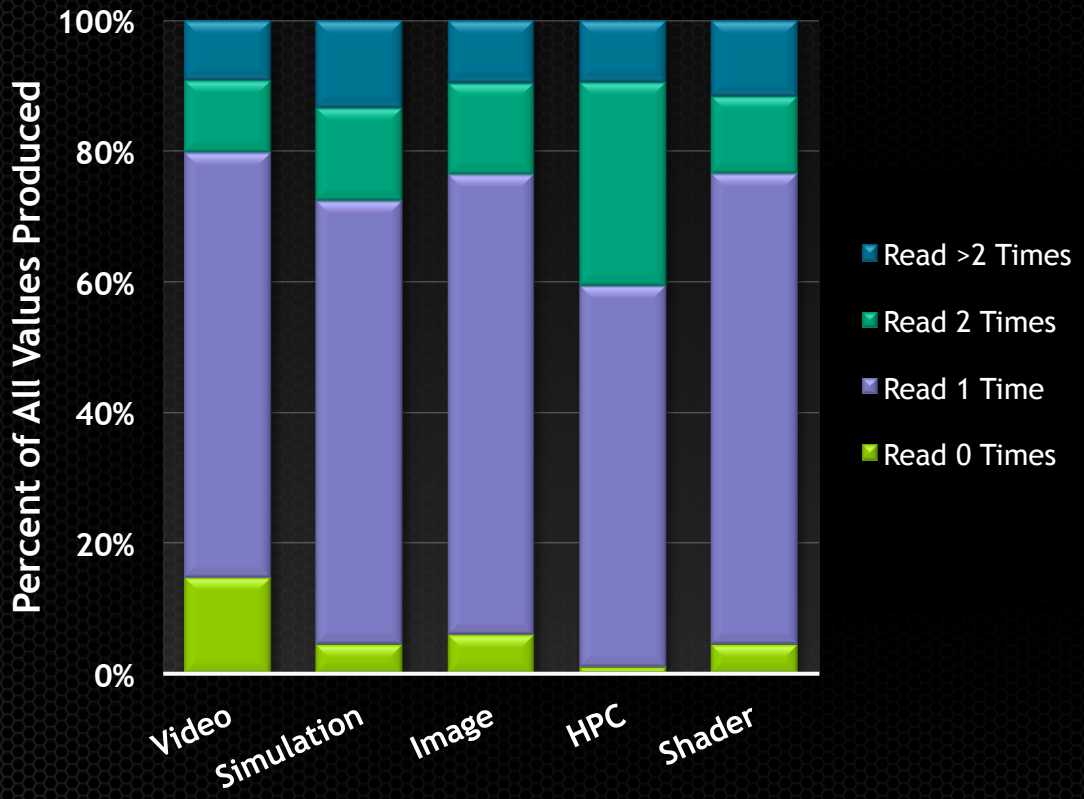


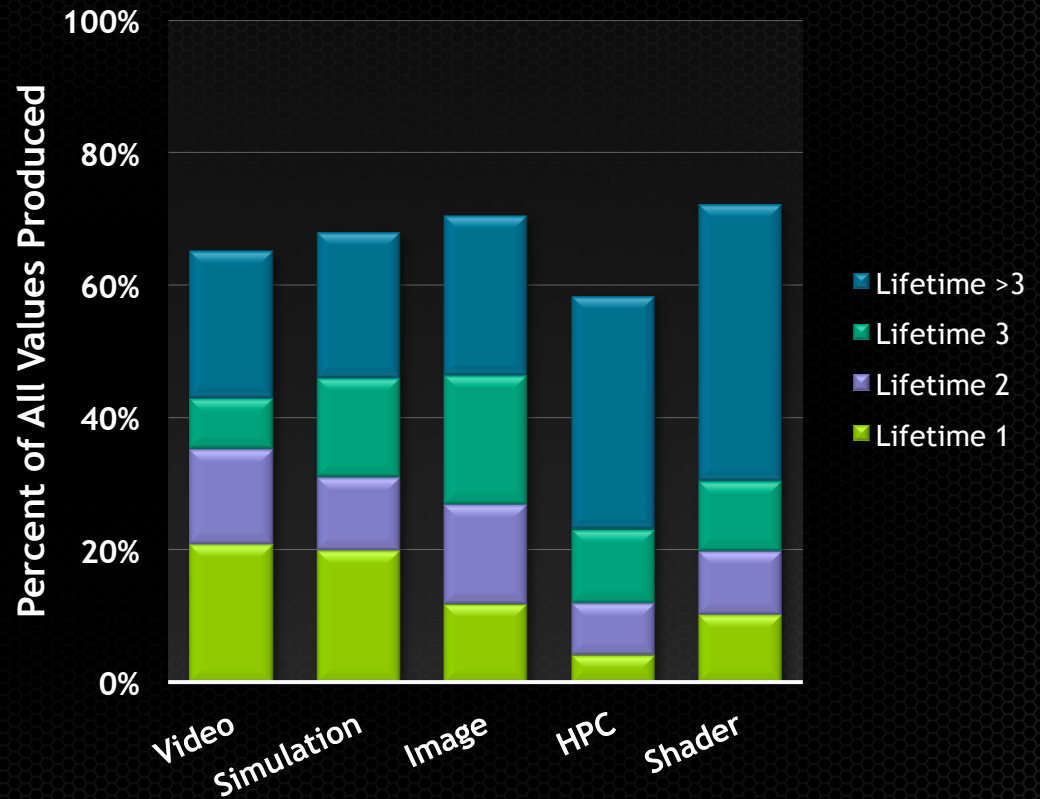


15% of SM Energy

Streaming Multiprocessor (SM)

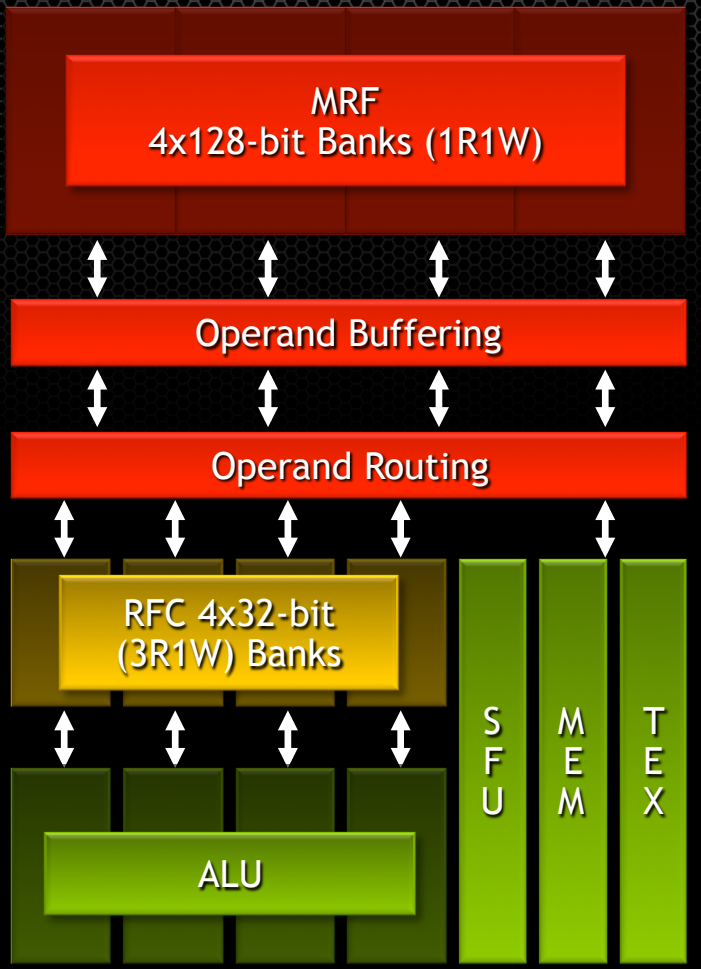
Hierarchical Register File





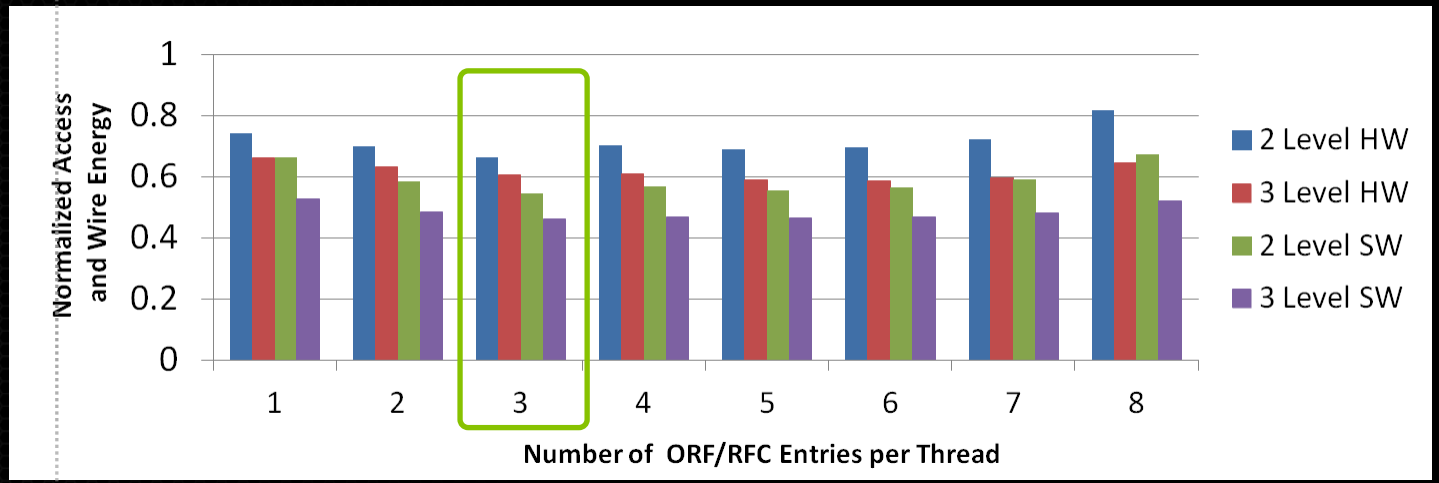
NVIDIA.

Register File Caching (RFC)



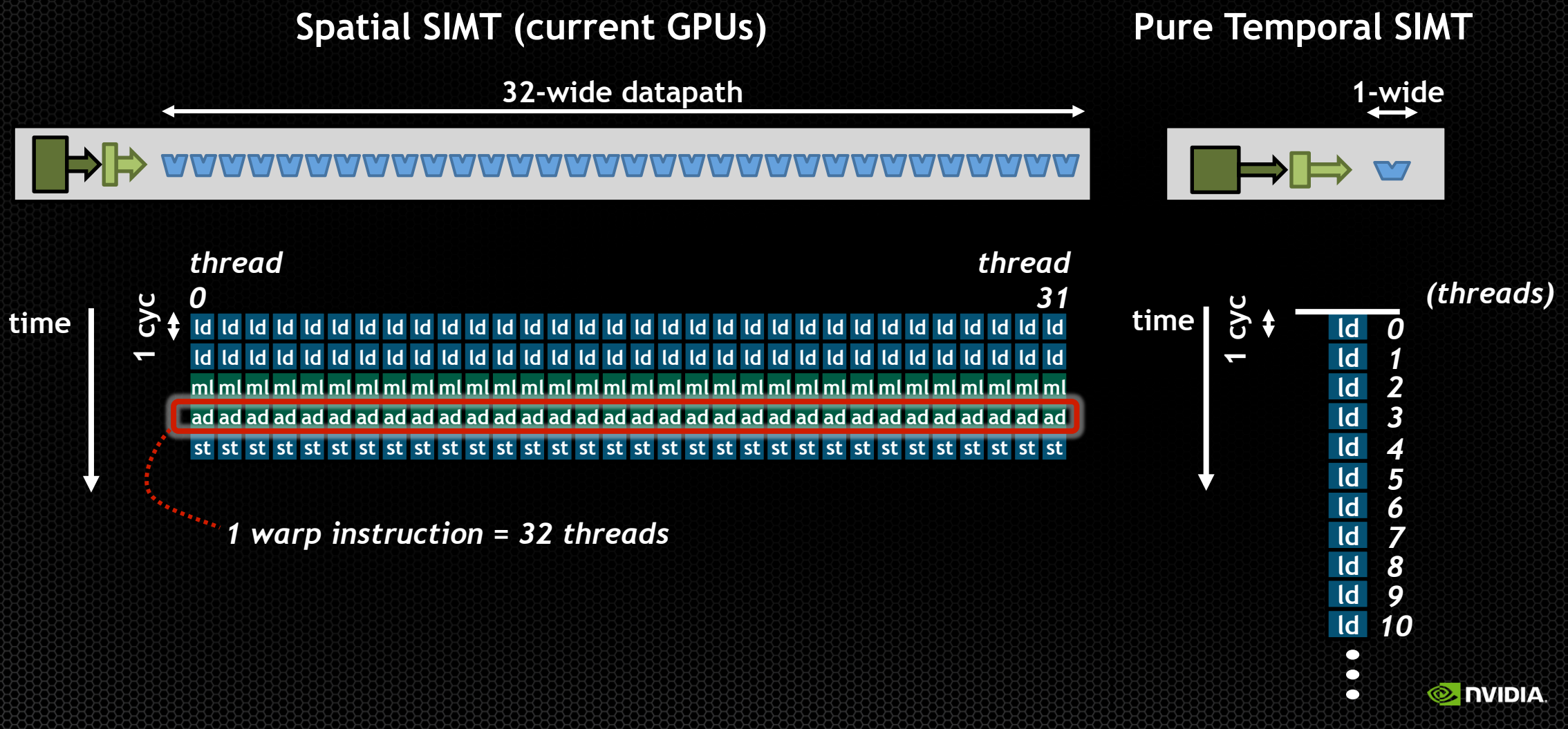
Energy Savings from RF Hierarchy

54% Energy Reduction



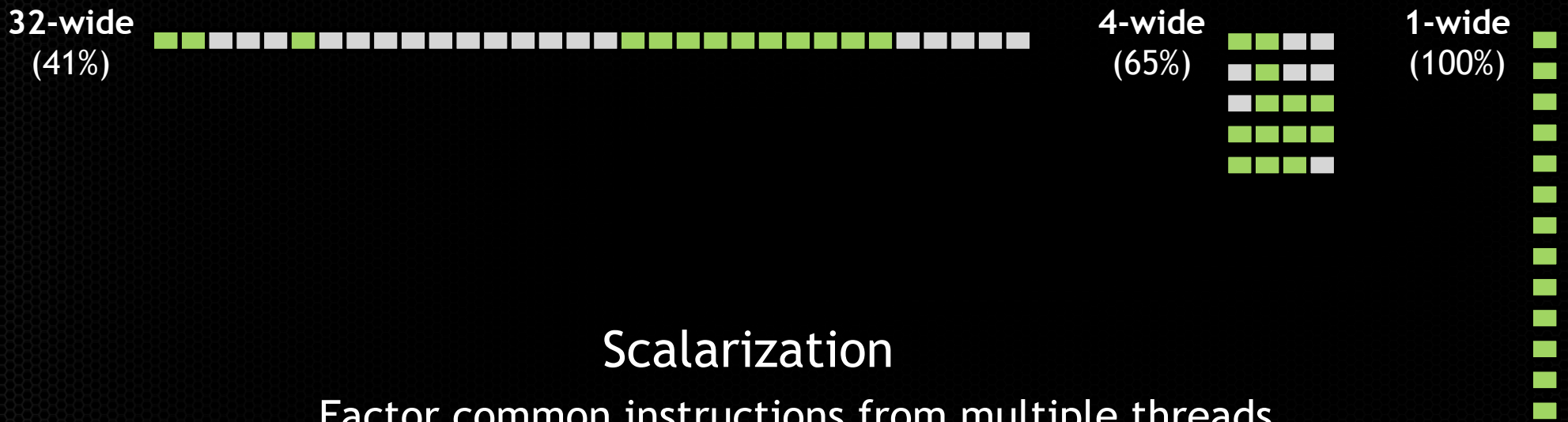
Source: Gebhart, et. al (Micro 2011)

Temporal SIMT



Temporal SIMT Optimizations

Control divergence — hybrid MIMD/SIMT

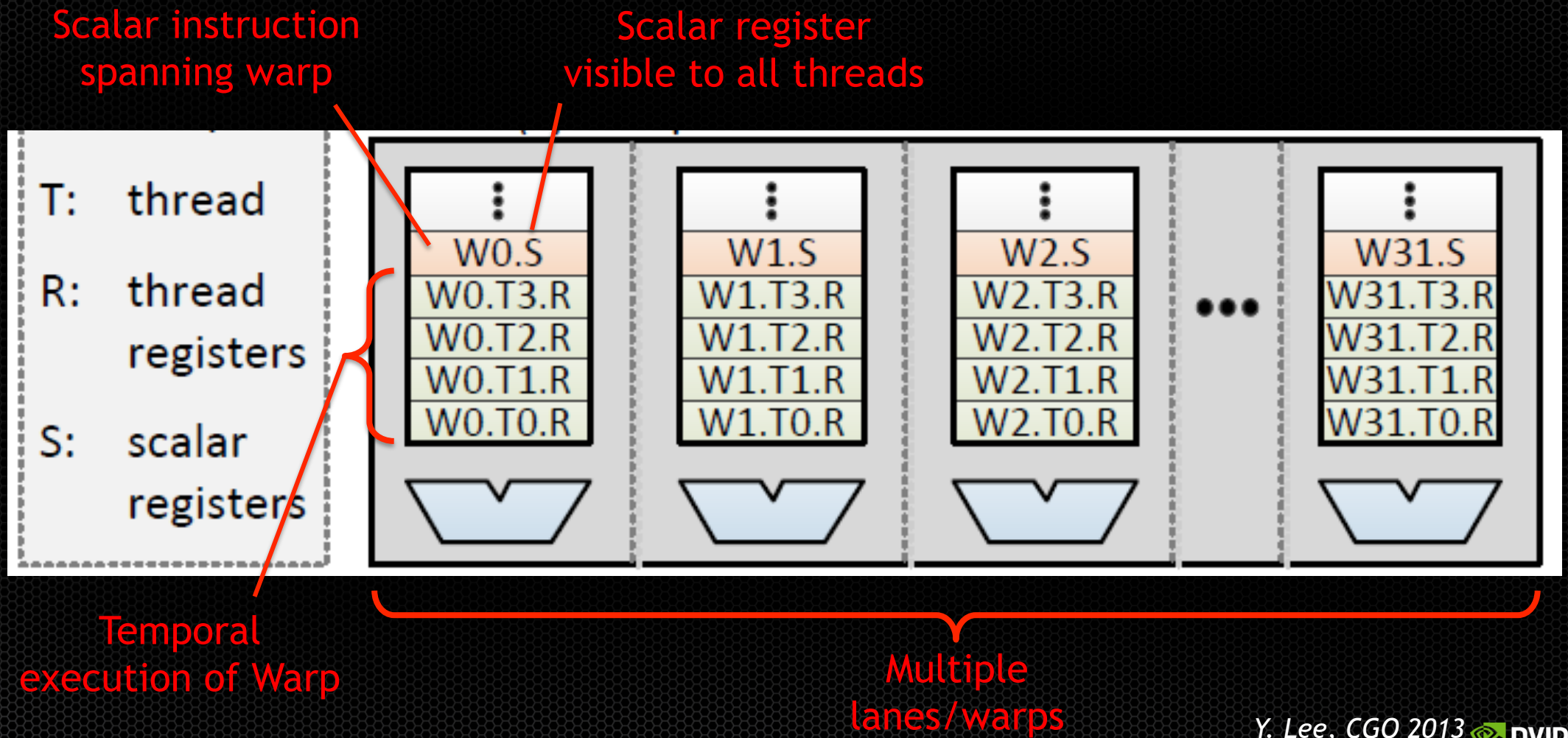


Scalarization

Factor common instructions from multiple threads

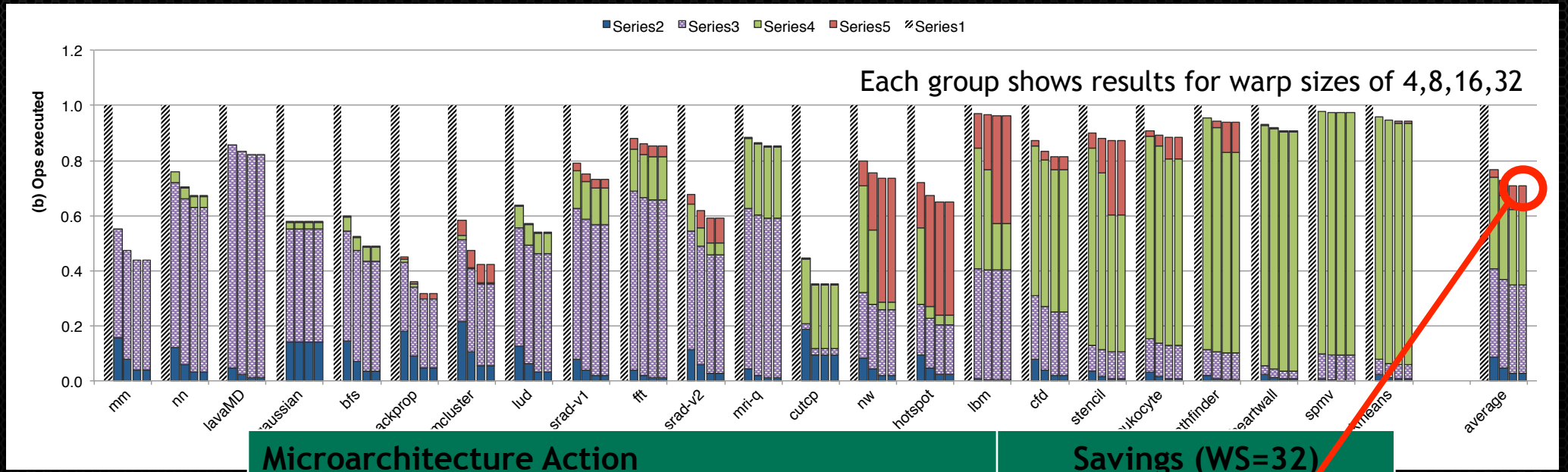
Execute once - place results in common registers

Scalar Instructions in SIMT Lanes



Y. Lee, CGO 2013

Scalarization Eliminates Work

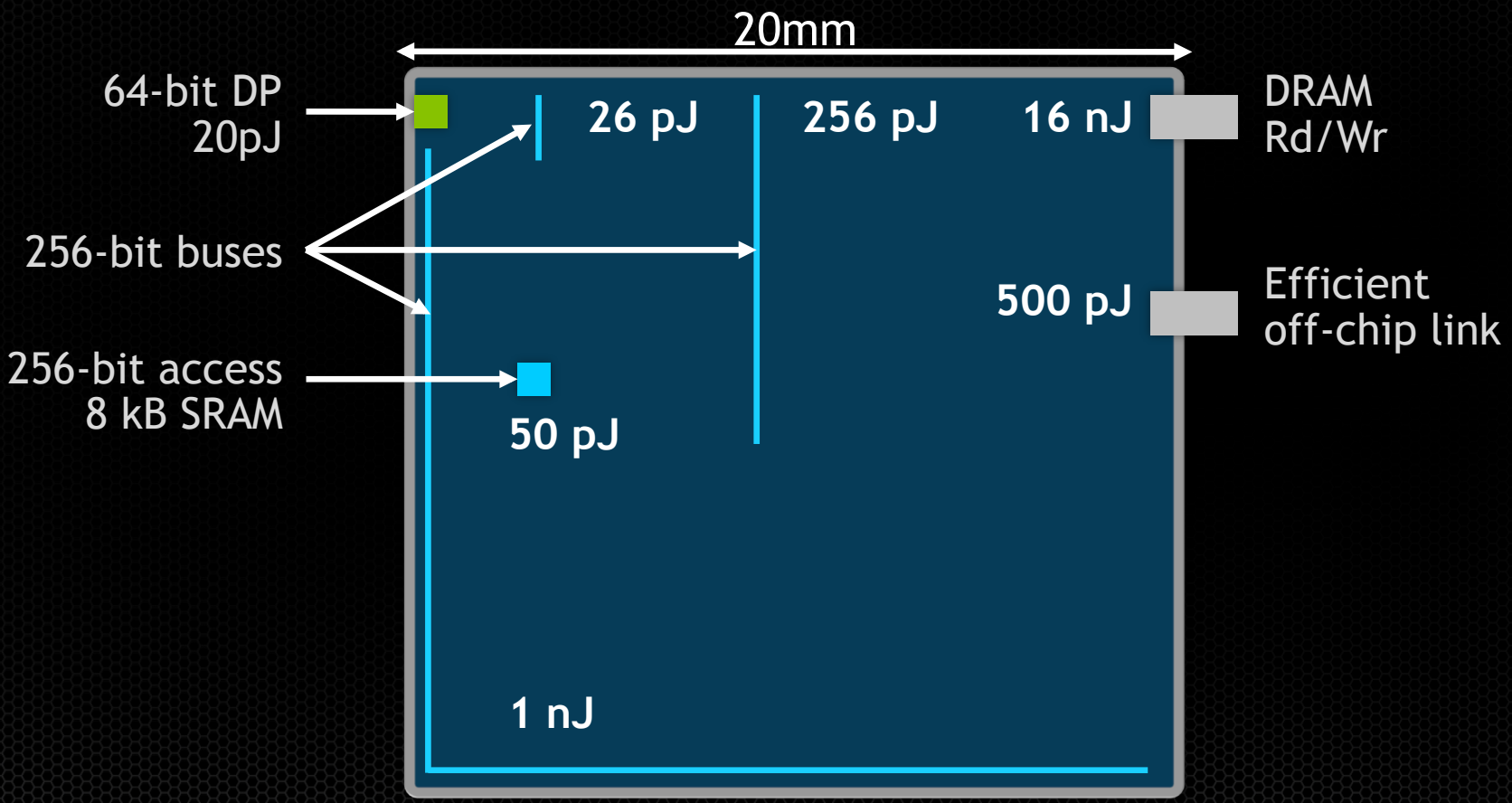


Microarchitecture Action	Savings (WS=32)
Instructions issued	41%
Operations executed	29%
Registers read	31%
Registers written	30%
Memory addresses and cache tag checks	47%
Memory data elements accessed	38%

Y. Lee, CGO 2013

NVIDIA

Communication Dominates Arithmetic



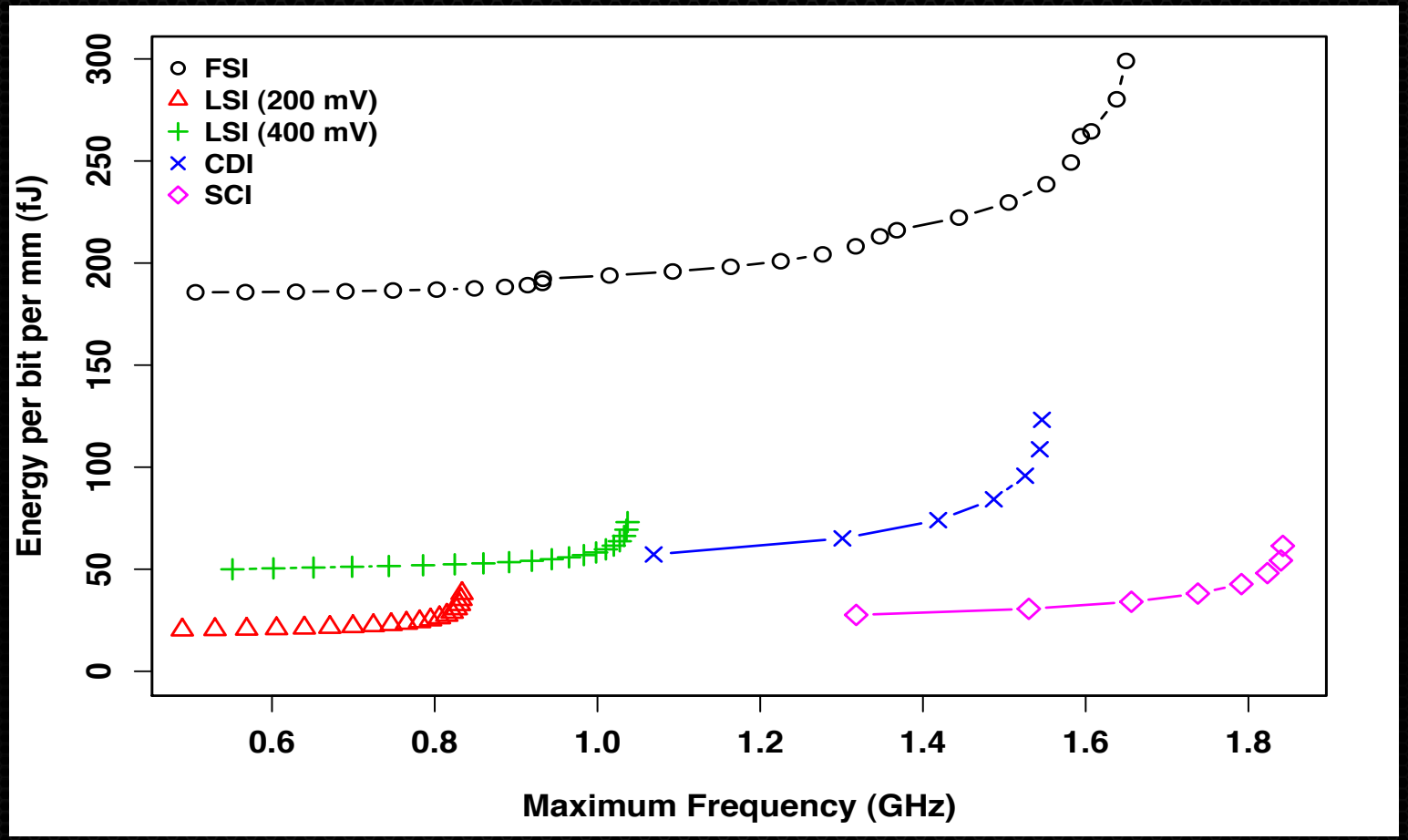
Energy Shopping List

Processor Technology	40 nm	10nm
Vdd (nominal)	0.9 V	0.7 V
DFMA energy	50 pJ	7.6 pJ
64b 8 KB SRAM Rd	14 pJ	2.1 pJ
Wire energy (256 bits, 10mm)	310 pJ	174 pJ

Memory Technology	45 nm	16nm
DRAM interface pin bandwidth	4 Gbps	50 Gbps
DRAM interface energy	20-30 pJ/bit	2 pJ/bit
DRAM access energy	8-15 pJ/bit	2.5 pJ/bit

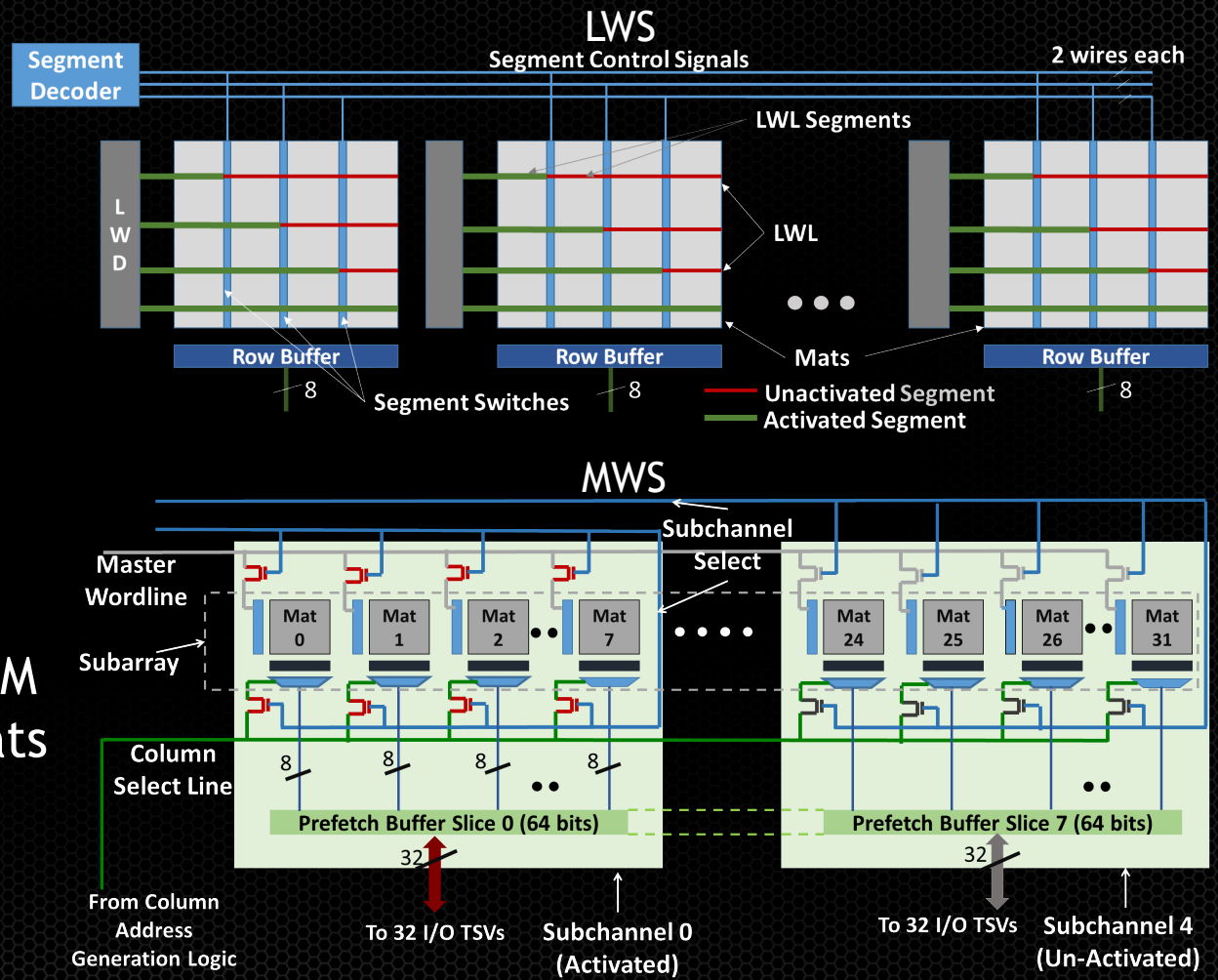
FP Op lower bound
=
4 pJ

Keckler [Micro 2011], Vogelsang [Micro 2010]


FSI
LSI (200 mV)
LSI (400 mV)
CDI
SCI
Energy per bit per mm (fJ)
0
50
100
150
200
250
300
0.6
0.8
1.0
1.2
1.4
1.6
1.8
Maximum Frequency (GHz)

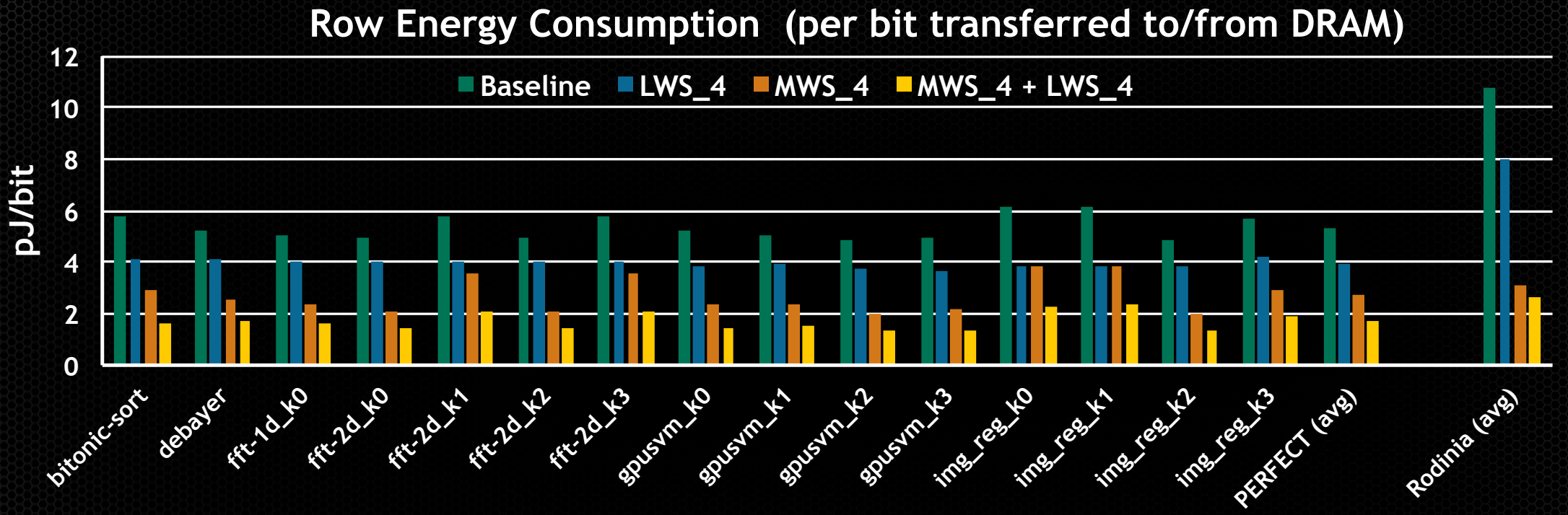
DRAM Wordline Segmentation

- Local Wordline Segmentation (LWS)
  - Activate a portion of each mat
  - Activate the same segment in all mats
- Master Wordline Segmentation (MWS)
  - Activate subset of mats
  - New data layout such that a DRAM atom resides in the activated mats
  - Multiple narrow subchannels in each channel



NVIDIA

Activation Energy Reduction

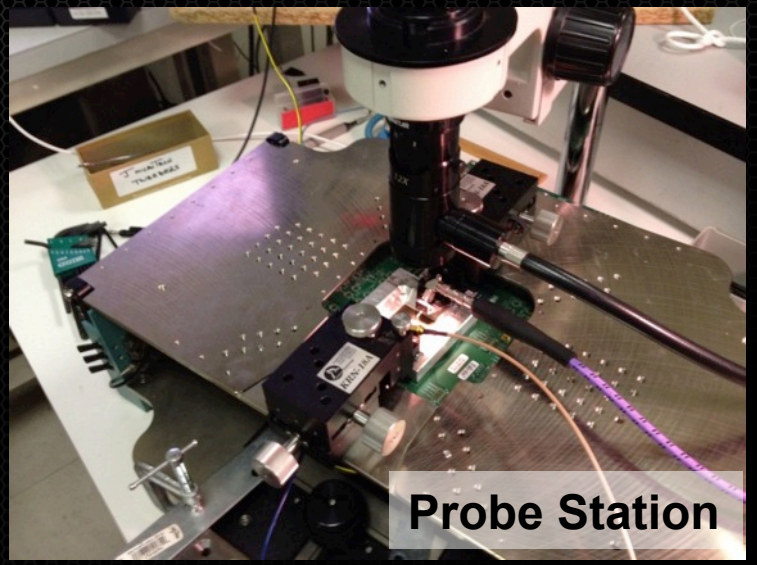


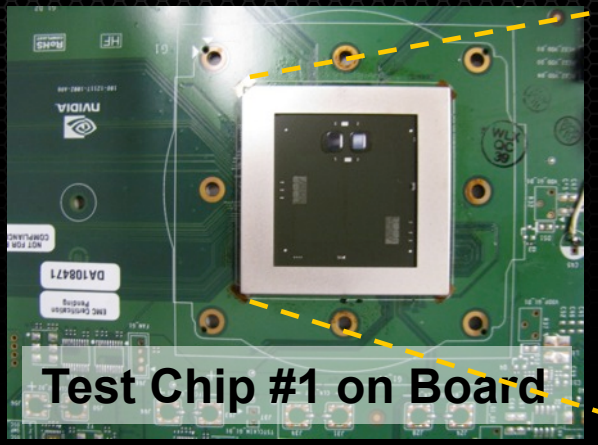
Reductions in DRAM Row Energy

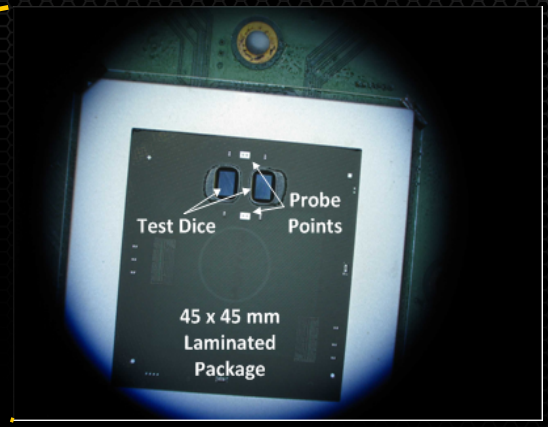
- LWS: 26%
- MWS: 49%
- LWS+MWS: up to 73%, average 68%

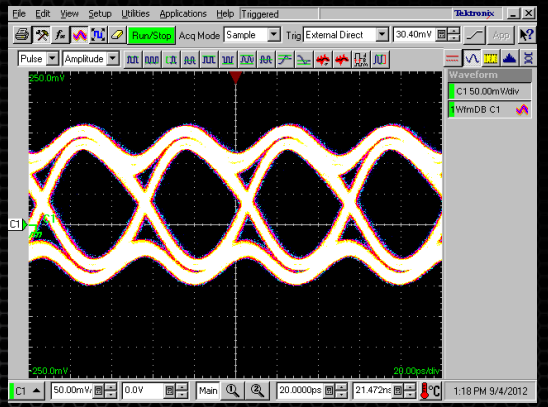
NVIDIA

GRS Test Chips

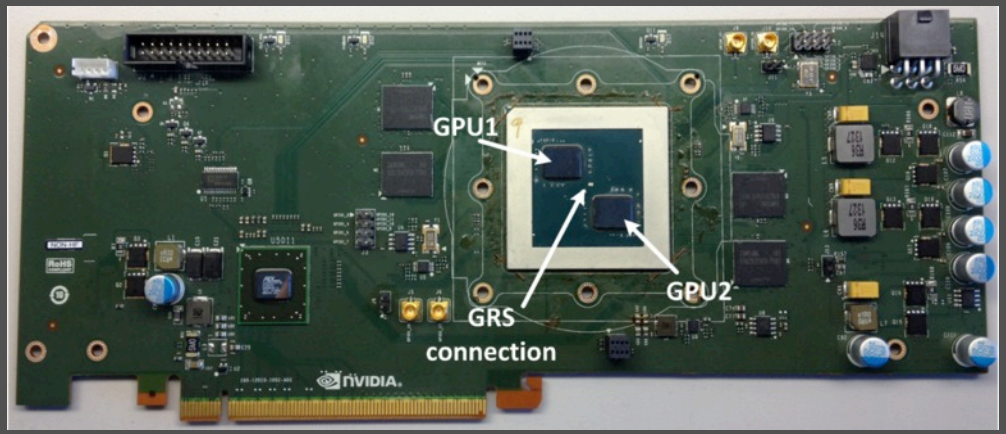








Eye Diagram from Probe



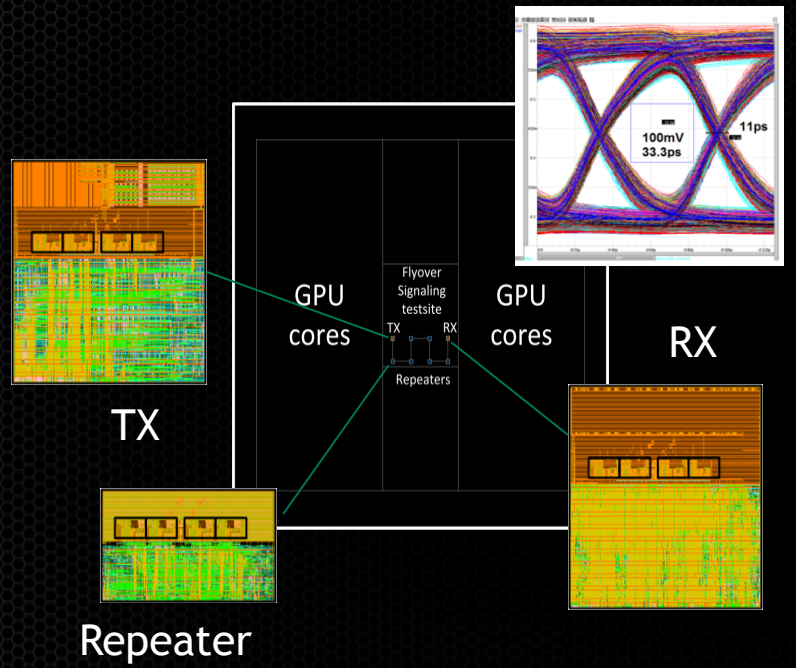
Test Chip #2 fabricated on production GPU

Poulton et al. ISSCC 2013, JSSCC Dec 2013

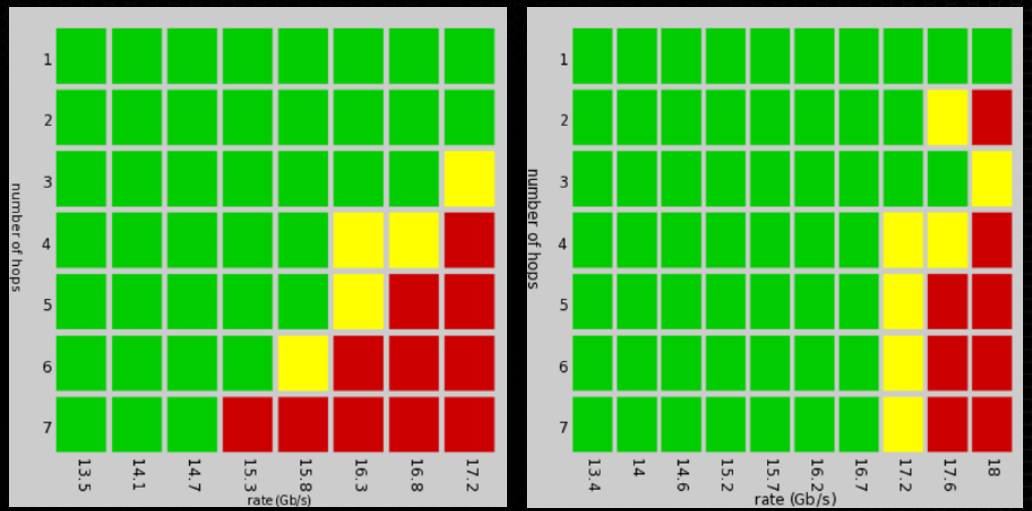
NVIDIA

Circuits: Ground Referenced Signaling

Goal: reduce Energy/bit
200fJ/bit-mm → 20fJ/bit-mm



Measurement Results



With GPU noise

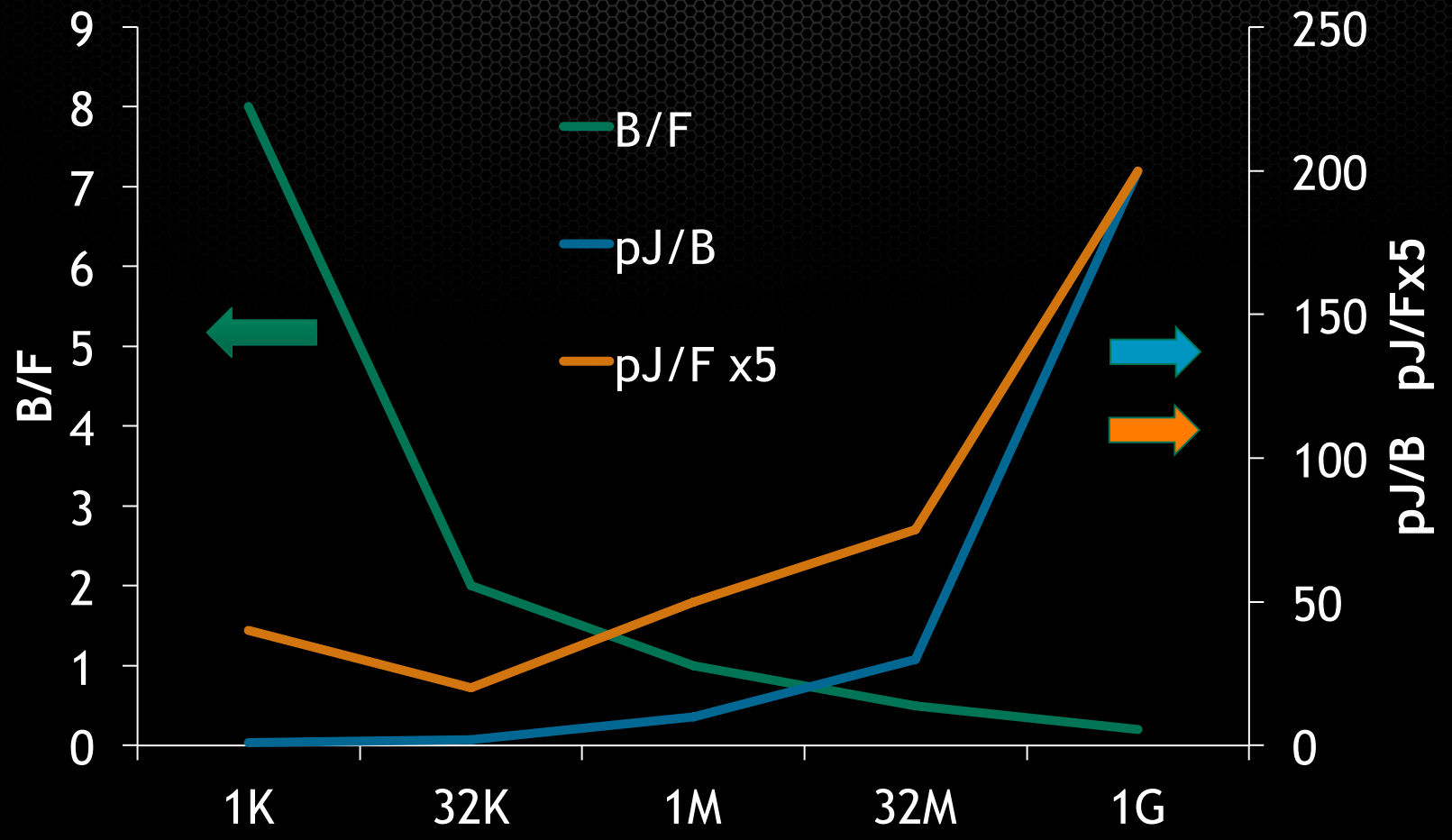
Without GPU noise

On-chip Signaling

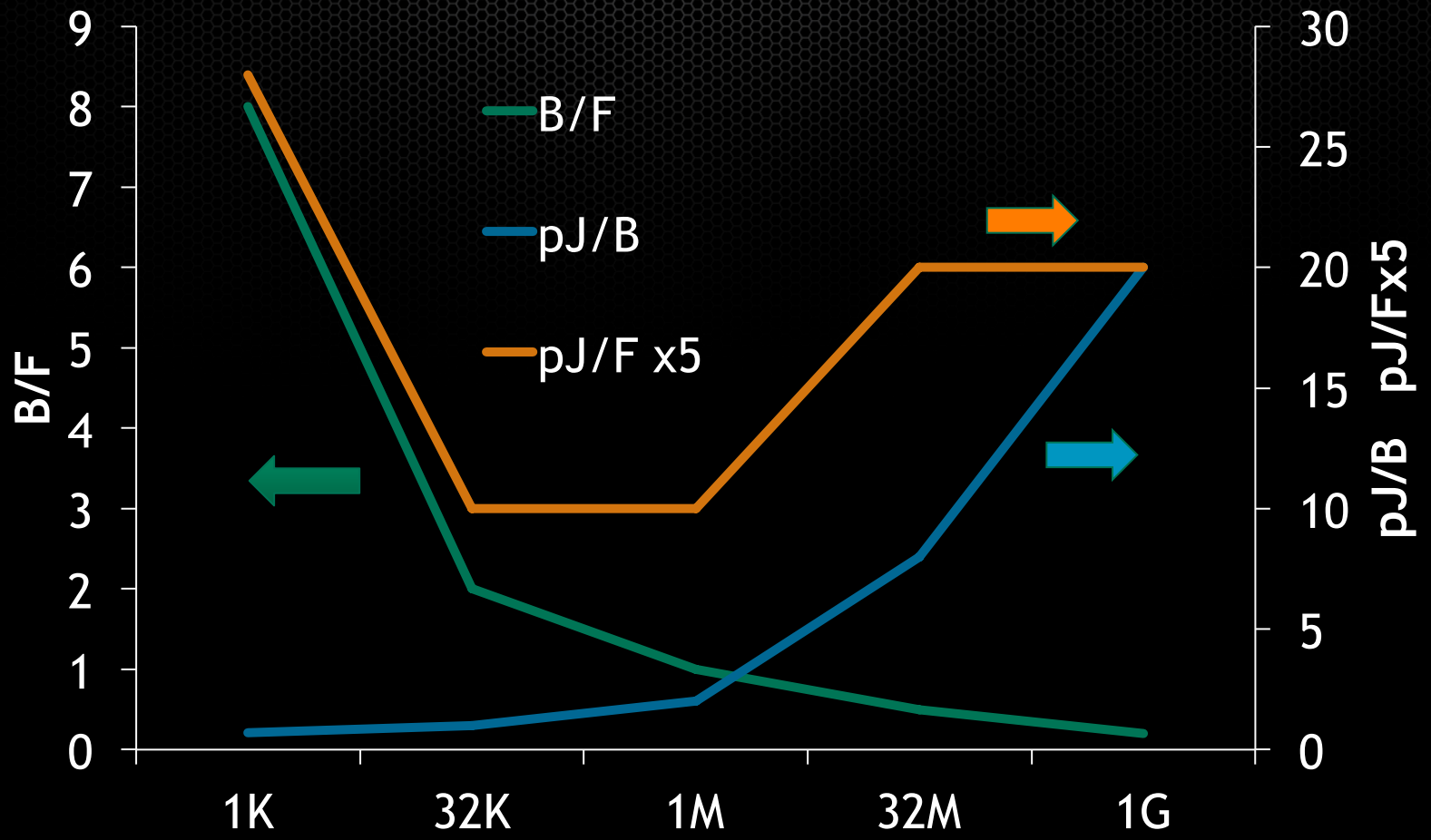
- Testsite on GM2xx GPU
- <45fJ/bit-mm

NVIDIA

The Locality Challenge
Data Movement Energy 77pJ/F



Optimized Architecture & Circuits
77pJ/F -> 18pJ/F

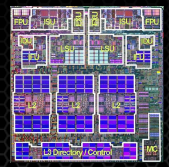


NVIDIA

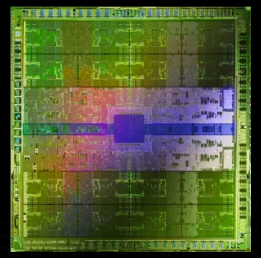
1997: 7.5M Tx

2001: 42M Tx

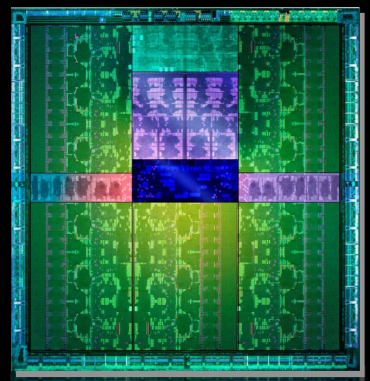
2004: 275M Tx



2007: 580M Tx



2010: 3B Tx



2012: 7B Tx

1993: 3M Tx

Moore's Law is Only Part of the Story

2 Trends

The End of Dennard Scaling

Pervasive Parallelism

Processors aren’t getting faster just wider

Thread Count (CPU+GPU)

	2013 (28nm)	2020 (7nm)
Cell Phone	4+1,000	16+4,000
Server Node	12+10,000	32+40,000
Supercomputer	10^5+10^8	10^6+10^9

Parallel programming is not inherently any more difficult than serial programming

However, we can make it a lot more difficult

A simple parallel program

```
forall molecule in set { // launch a thread array
    forall neighbor in molecule.neighbors { // nested
        forall force in forces { // doubly nested
           molecule.force =
             reduce_sum(force(molecule, neighbor))
        }
    }
}
```

Why is this easy?

```
forall molecule in set { // launch a thread array
    forall neighbor in molecule.neighbors { // nested
        forall force in forces { // doubly nested
           molecule.force =
             reduce_sum(force(molecule, neighbor))
        }
    }
}
```

No machine details
All parallelism is expressed
Synchronization is semantic (in reduction)

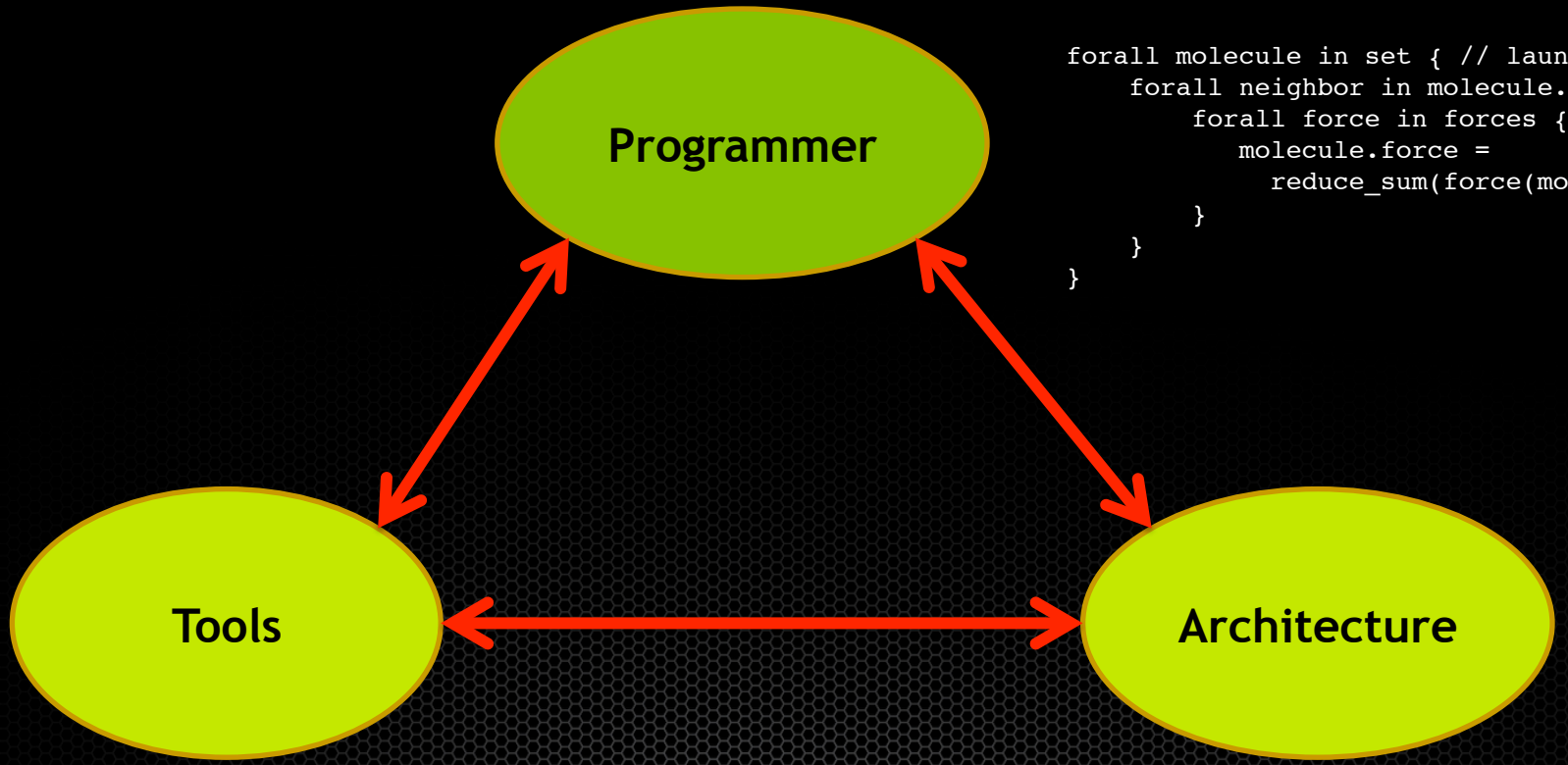
```
pid = fork() ; // explicitly managing threads

lock(struct.lock) ;  // complicated, error-prone synchronization
// manipulate struct
unlock(struct.lock) ;

code = send(pid, tag, &msg) ;  // partition across nodes
```

We could make it hard

Programmers, tools, and architecture Need to play their positions

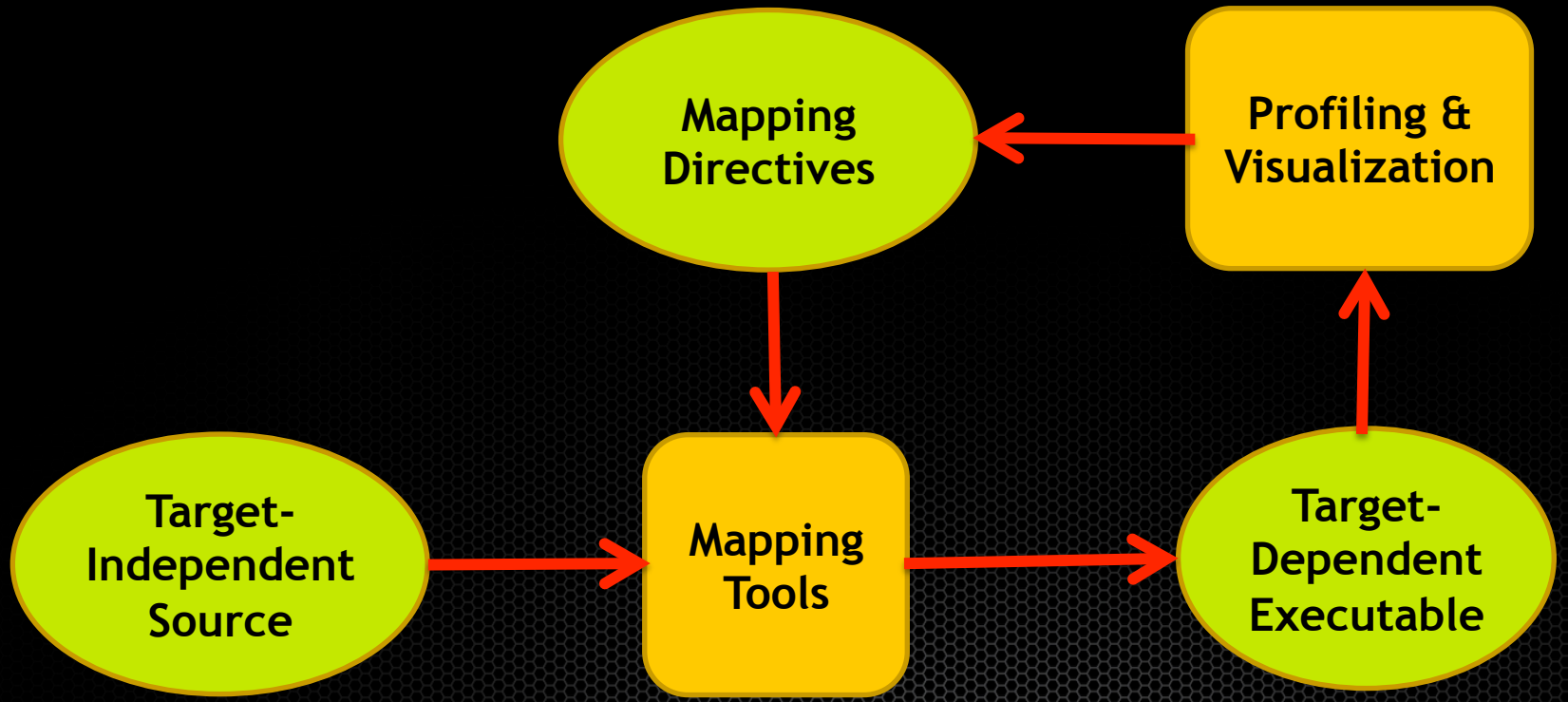



```
forall molecule in set { // launch a thread array
    forall neighbor in molecule.neighbors { //
        forall force in forces { // doubly nested
            molecule.force =
                reduce_sum(force(molecule, neighbor))
        }
    }
}
```

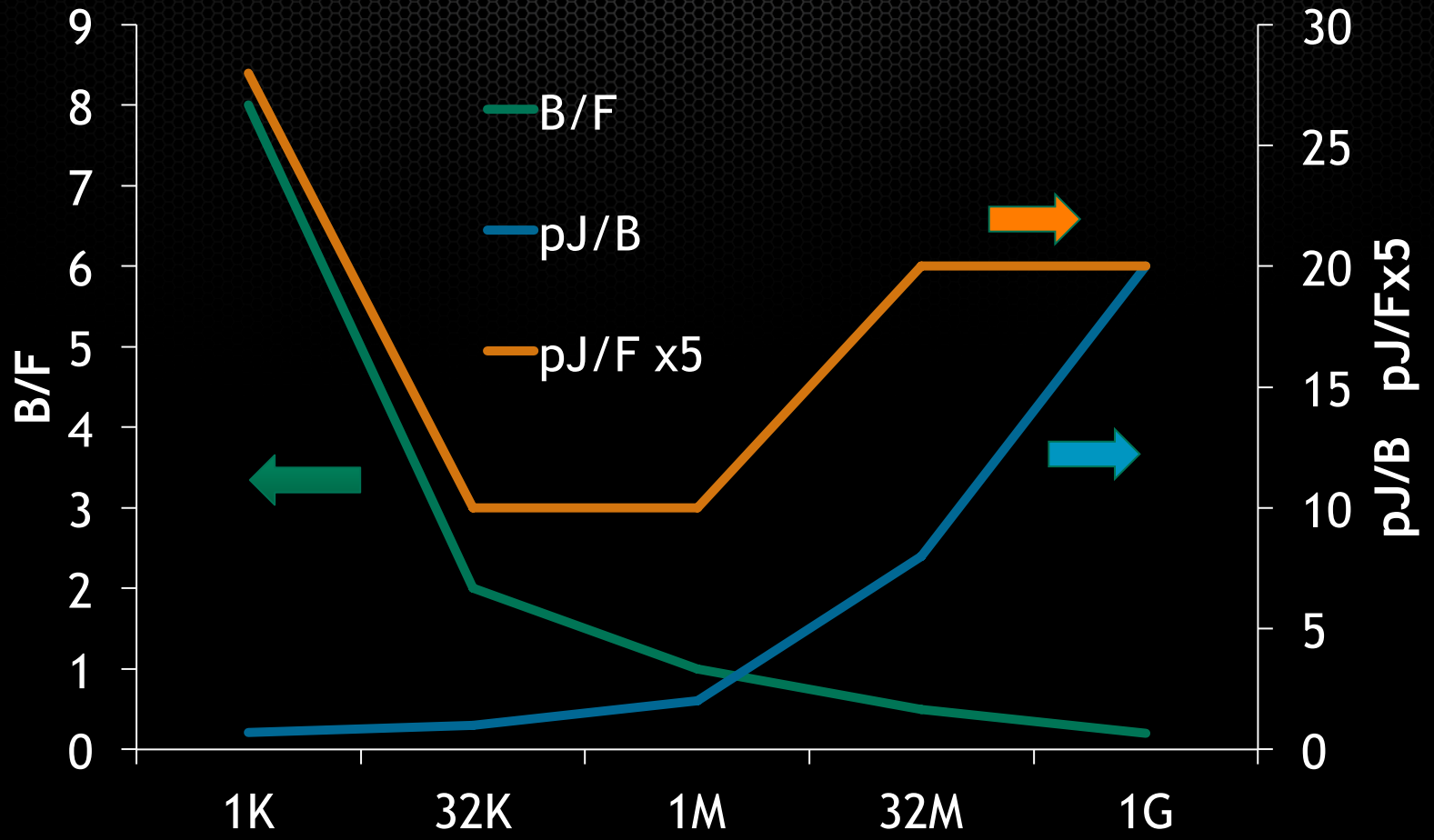
Map foralls in time and space
Map molecules across memories
Stage data up/down hierarchy
Select mechanisms

Exposed storage hierarchy
Fast comm/sync/thread mechanisms

NVIDIA

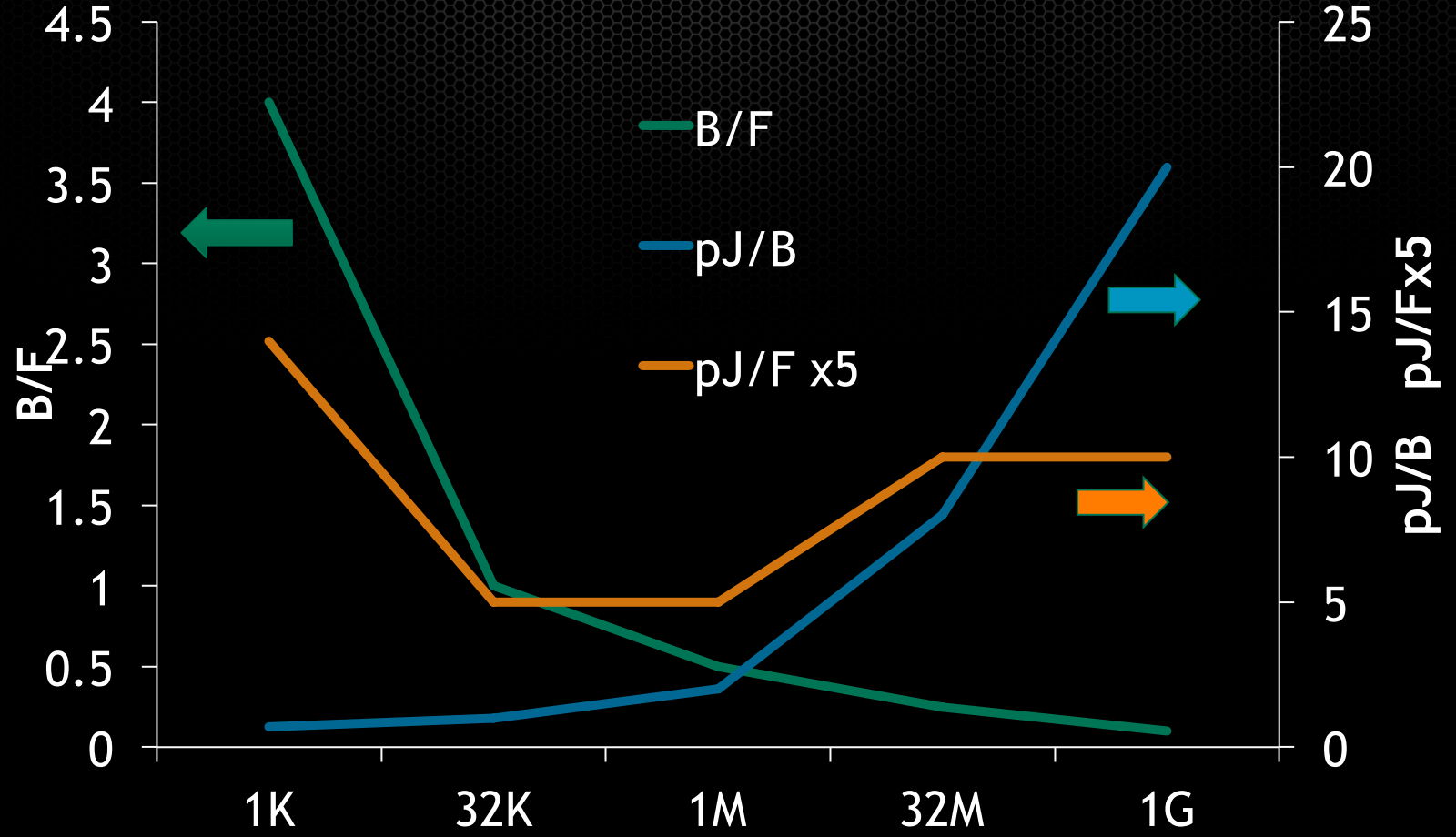

Mapping Directives
Profiling & Visualization
Target-Independent Source
Mapping Tools
Target-Dependent Executable

Optimized Circuits
77pJ/F -> 18pJ/F



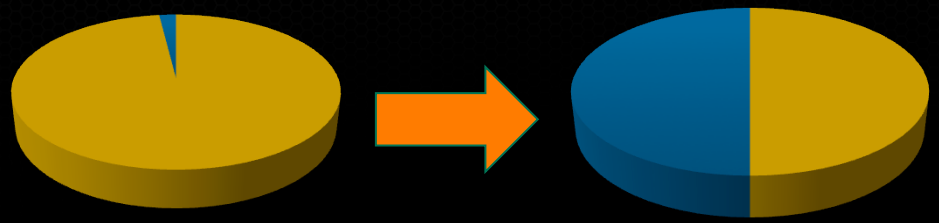
NVIDIA

Autotuned Software
18pJ/F -> 9pJ/F



Conclusion

- Power-limited: from data centers to cell phones
  - Perf/W is Perf
- Throughput cores
  - Reduce overhead



- Data movement
  - Circuits: 200 -> 20
  - Optimized software
- Parallel programming is simple – we can make it hard
  - Target-independent programming – mapping via tools