18-447 Lecture 27: Hardware Acceleration

James C. Hoe
Department of ECE
Carnegie Mellon University

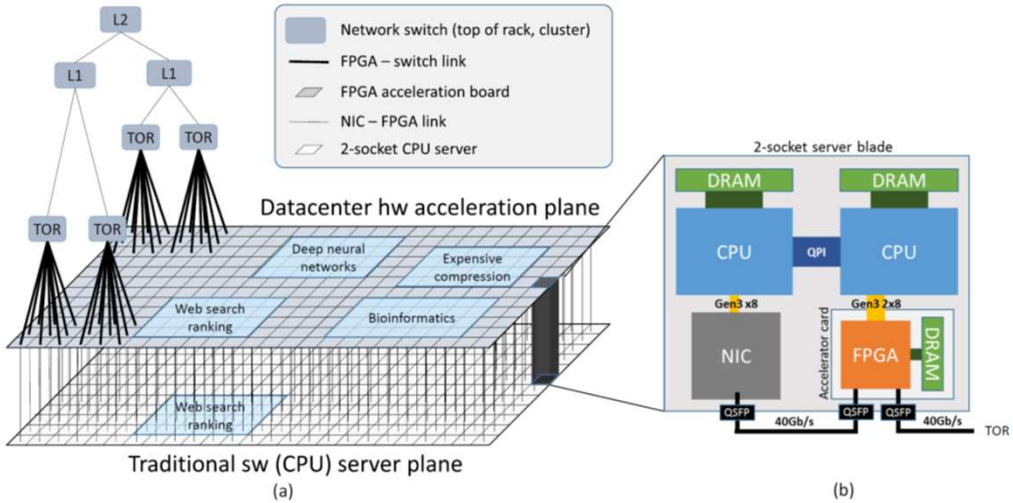
Housekeeping

- Your goal today
  - see why you should care about accelerators
  - know the basics to think about the topic
- Notices
  - Lab4, due this week
  - HW5, past due
- Readings
  - *Amdahl's Law in the Multicore Era*, 2008 (optional)
  - *Single-Chip Heterogeneous Computing: Does the Future Include Custom Logic, FPGAs, and GPGPUs?* 2010 (optional)

"HW Acceleration" is nothing new!

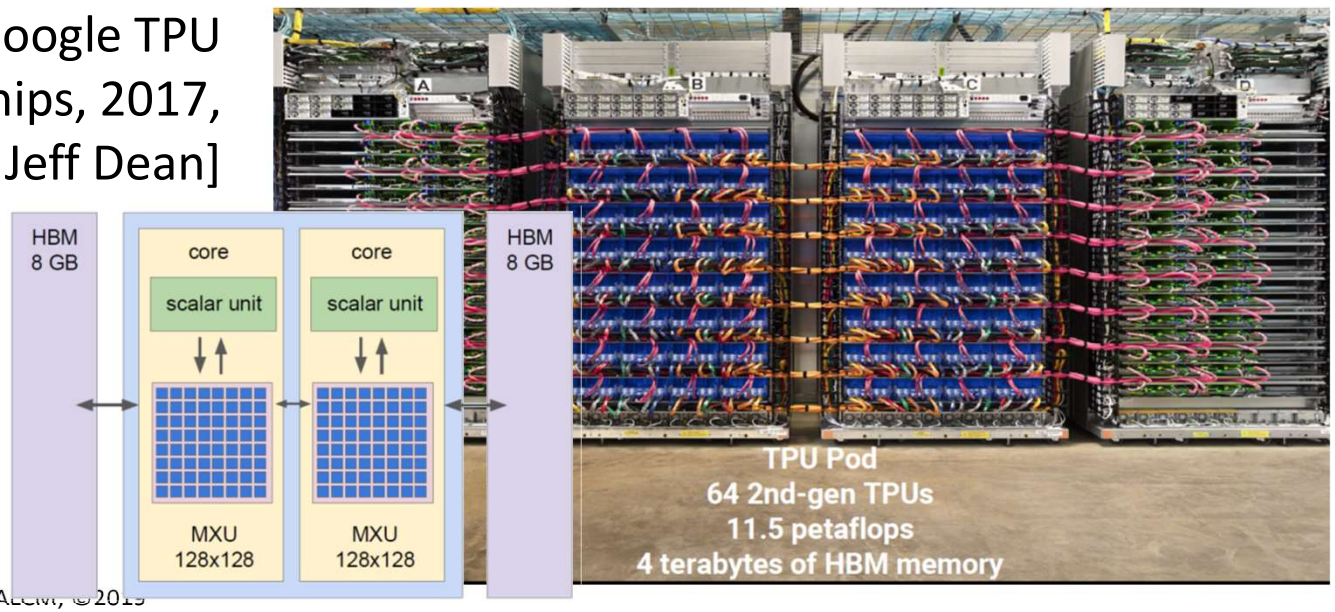
- What needed to be faster/smaller/cheaper/ lower-energy than SW has always been done in HW
  - we go to HW when SW isn't good enough because "good" HW can be more efficient
  - we don't go to HW when SW is good enough because "good" HW takes more work
- When we say "HW acceleration", we always mean efficient and not just correct

Computing's Brave New World

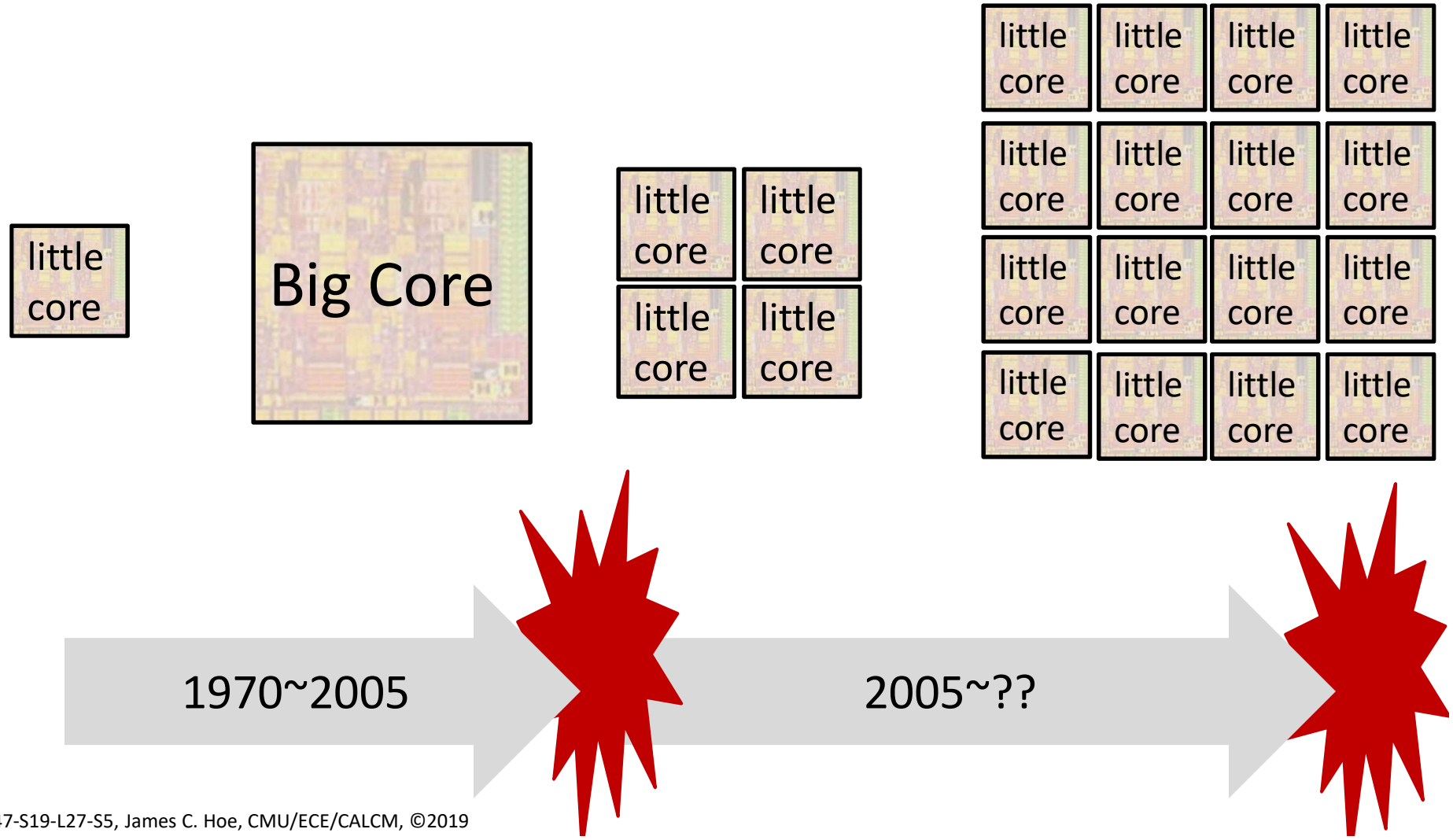


Microsoft Catapult [MICRO 2016, Caulfield, et al.]

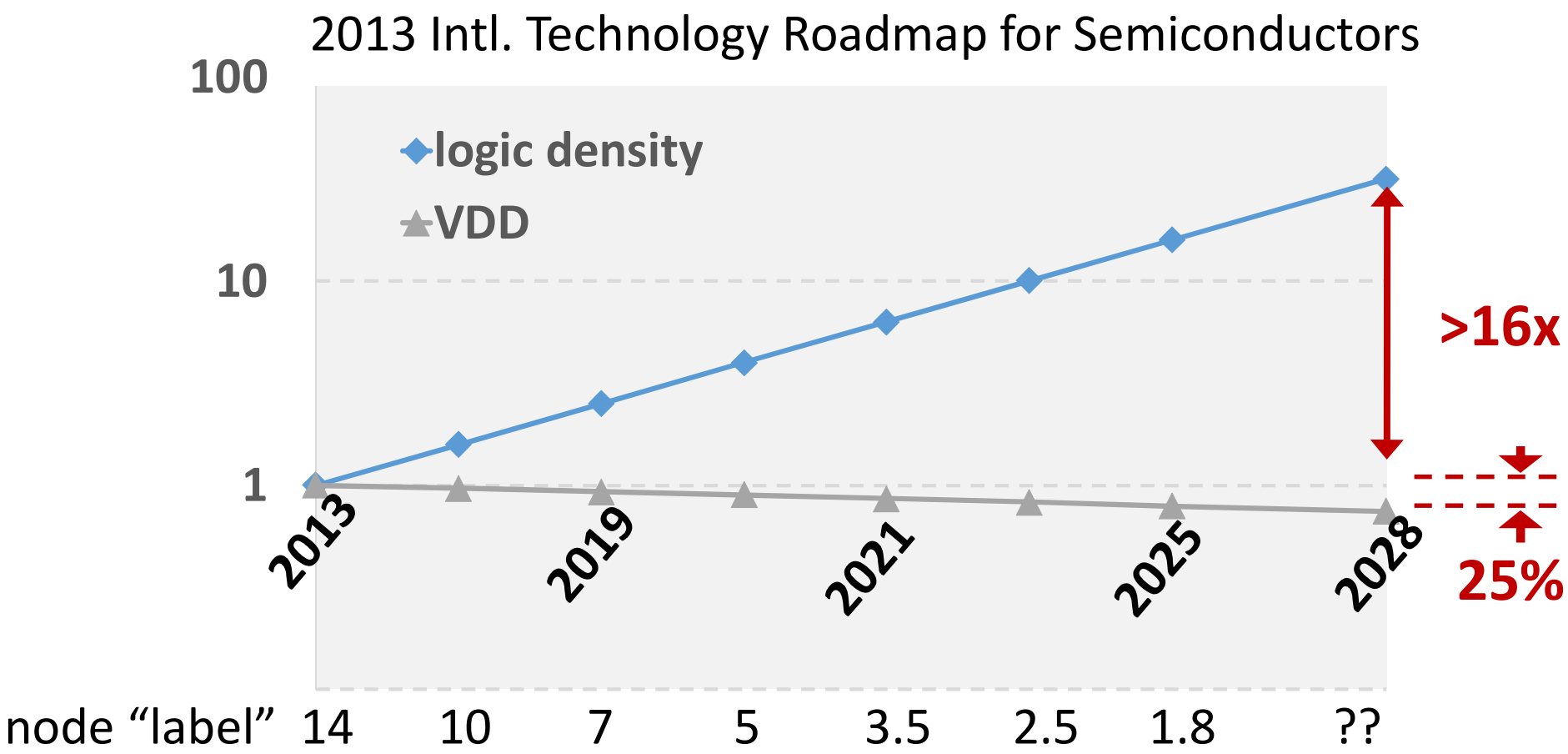
Google TPU [Hotchips, 2017, Jeff Dean]



How we got here . . . .



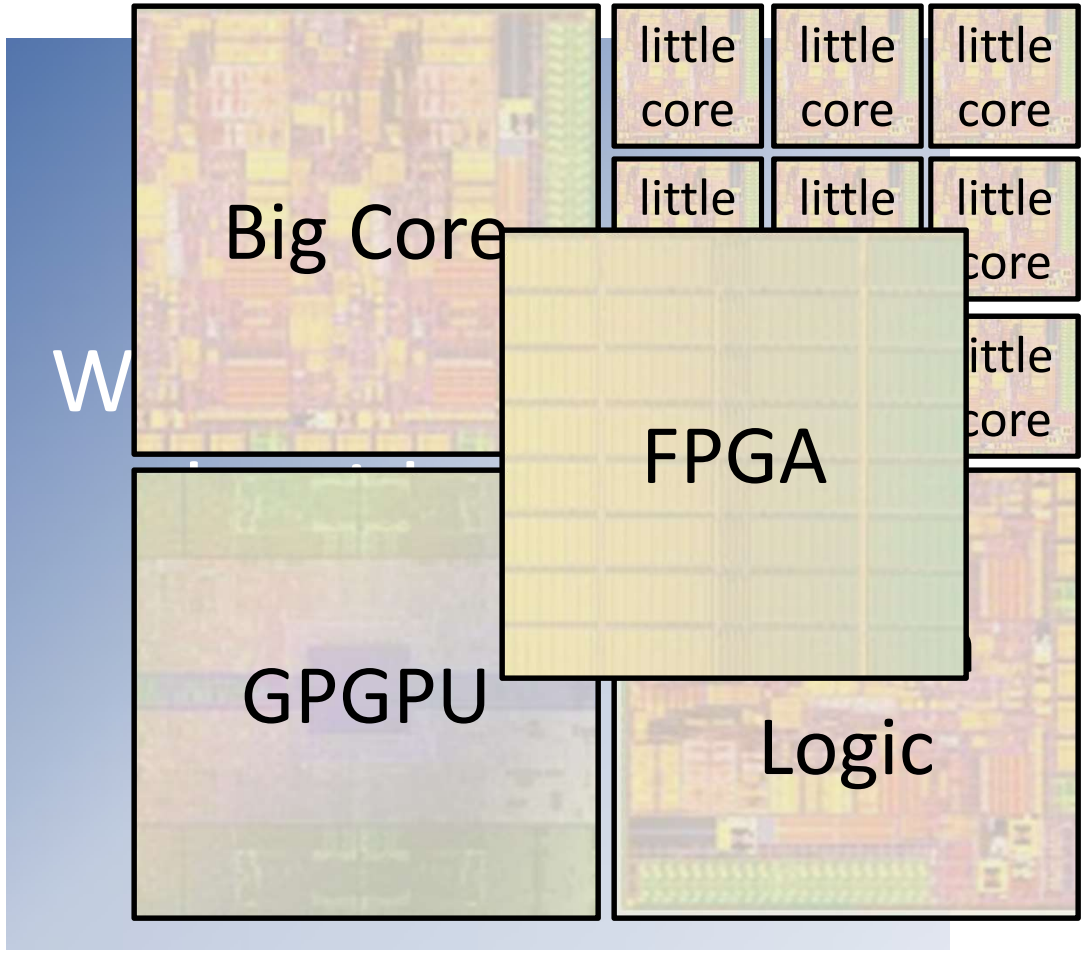
Moore's Law without Dennard Scaling



Under fixed power ceiling, more ops/second only achievable if less Joules/op?

Future is about

Performance/Watt and Ops/Joule



This is a sign of desperation . . . .

Why is Computing Directly in Hardware Efficient?

Why is HW/FPGA better? no overhead

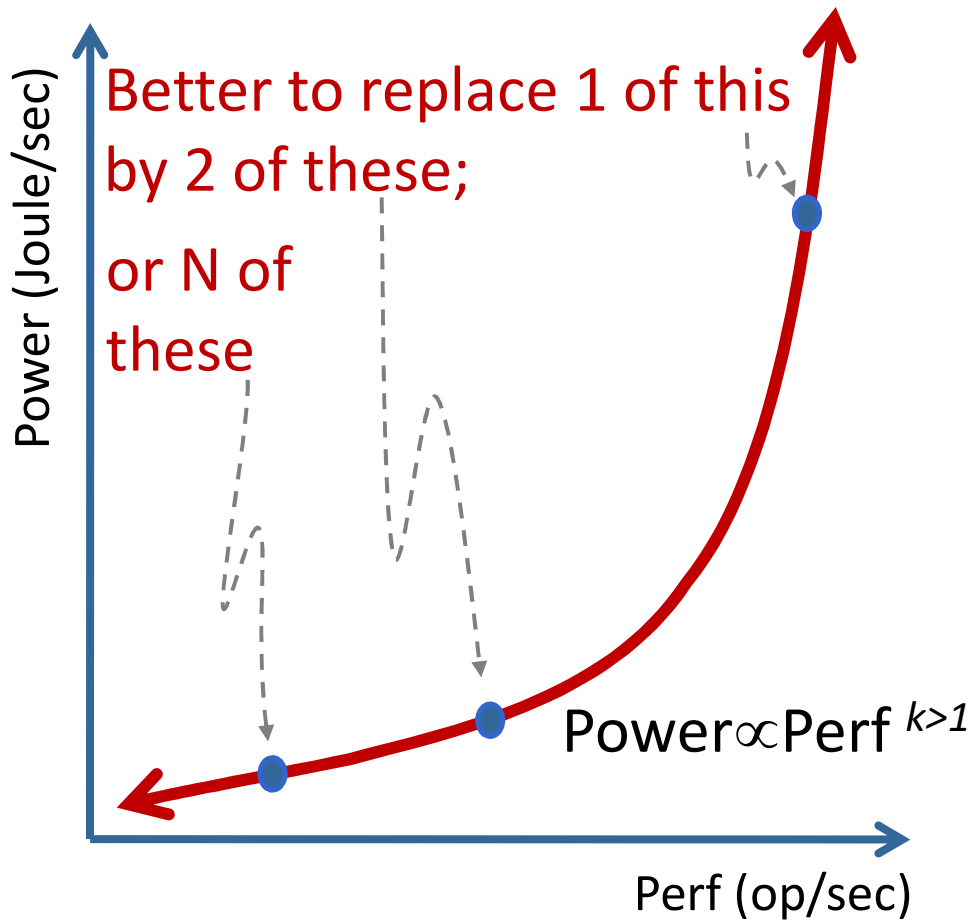
- A processor spends a lot of transistors & energy
  - to present von Neumann ISA abstraction
  - to support a broad application base (e.g., caches, superscalar out-of-order, prefetching, . . .)
- In fact, processor is mostly overhead
  - ~90% energy [Hameed, ISCA 2010, Tensilica core]
  - ~95% energy [Balfour, CAL 2007, embedded RISC]
  - even worse on a high-perf superscalar-OoO proc

Computing directly in application-specific hardware can be 10x to 100x more energy efficient

Why is HW/FPGA better? efficiency of parallelism

- For a given functionality, non-linear tradeoff between power and performance
  - slower design is simpler
  - lower frequency needs lower voltage

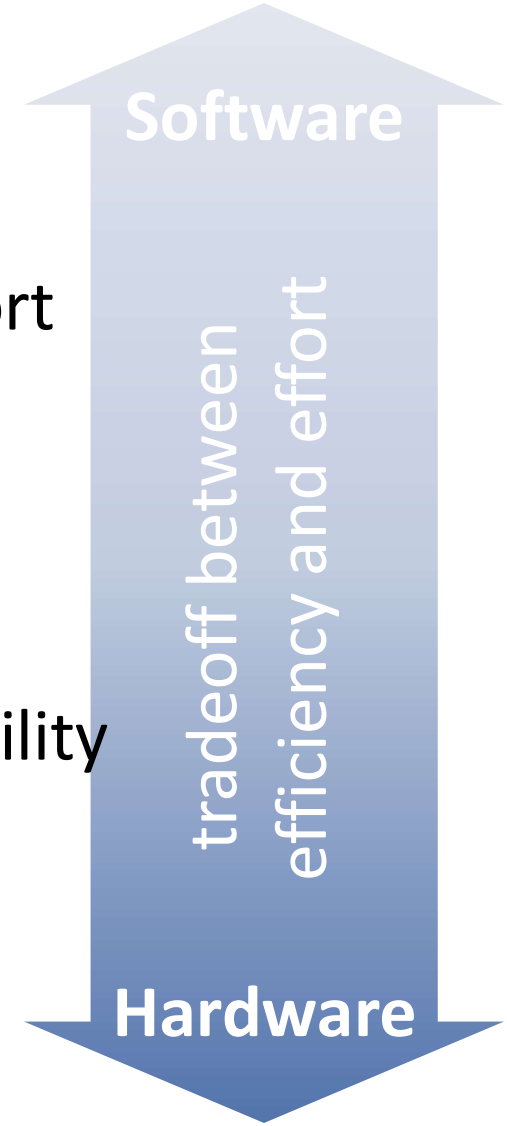
⇒For the same throughput, replacing 1 module by 2 half-as-fast reduces total power and energy



Good hardware designs derive performance from parallelism

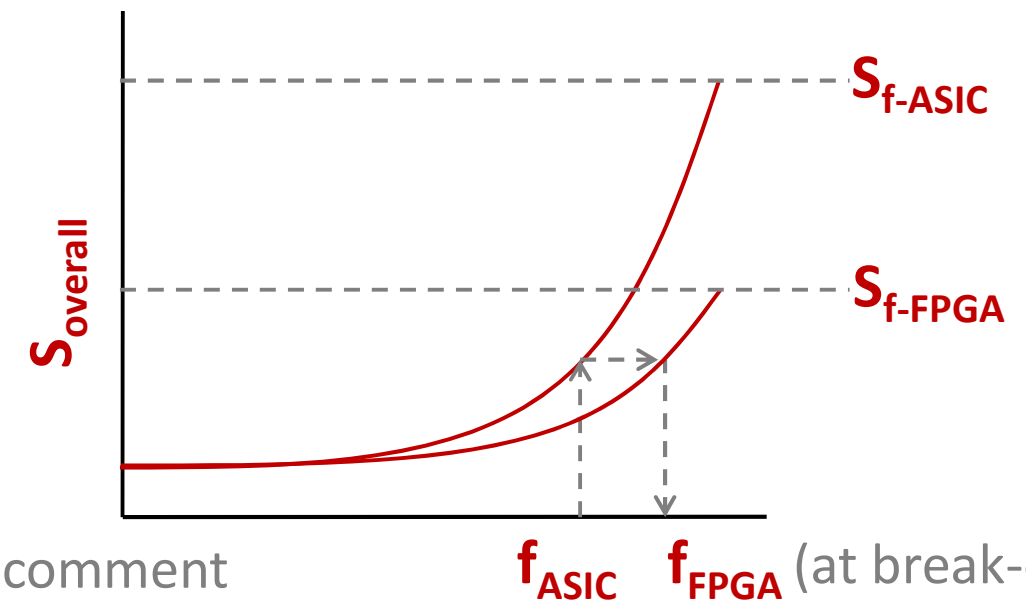
Software to Hardware Spectrum

- CPU: highest-level abstraction /
  most general-purpose support
- GPU: explicitly parallel programs /
  best for SIMD, regular
- FPGA: ASIC-like abstraction /
  overhead for reprogrammability
- ASIC: lowest-level abstraction /
  fixed application and tuning



ASIC isn't always ultimate in performance

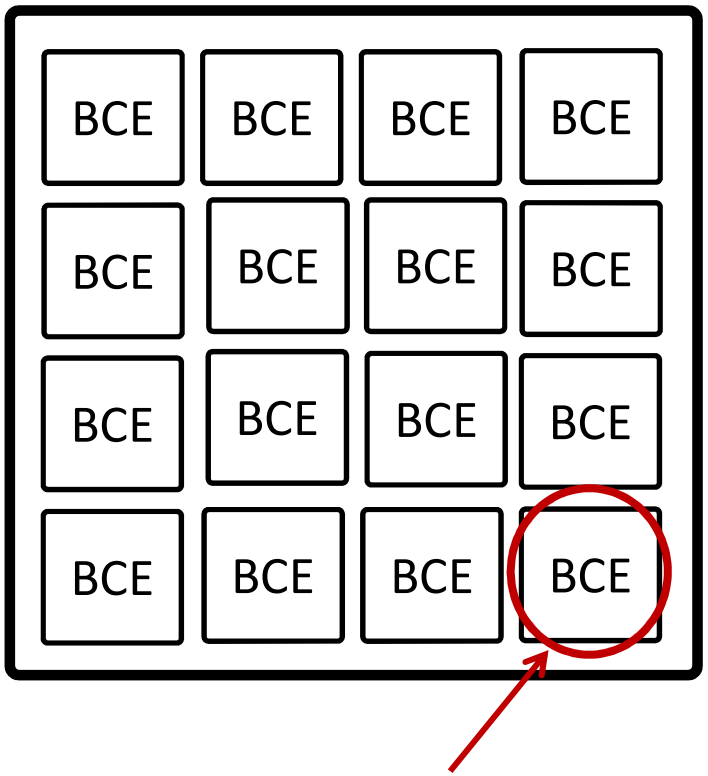
- Amdahl's Law: $S_{overall} = 1 / ((1-f) + f/S_f)$
- $S_{f\text{-}ASIC} > S_{f\text{-}FPGA}$ but $f_{ASIC} \neq f_{FPGA}$
- $f_{FPGA} > f_{ASIC}$ (when not perfectly app-specific)
  - more flexible design to cover a greater fraction
  - reprogram FPGA to cover different applications



[based on Joel Emer's original comment about programmable accelerators in general]

Tradeoff in Heterogeneity?

Amdahl's Law on Multicore



Base Core Equivalent (BCE) in [Hill and Marty, 2008]

- A program is rarely completely parallelizable; let's say a fraction f is perfectly parallelizable
- Speedup of n cores over "sequential"

$$Speedup = \frac{1}{(1-f) + \frac{f}{n}}$$

- But, "sequential" above determined by how many cores to dice an area into

http://research.cs.wisc.edu/multifacet/amdahl/

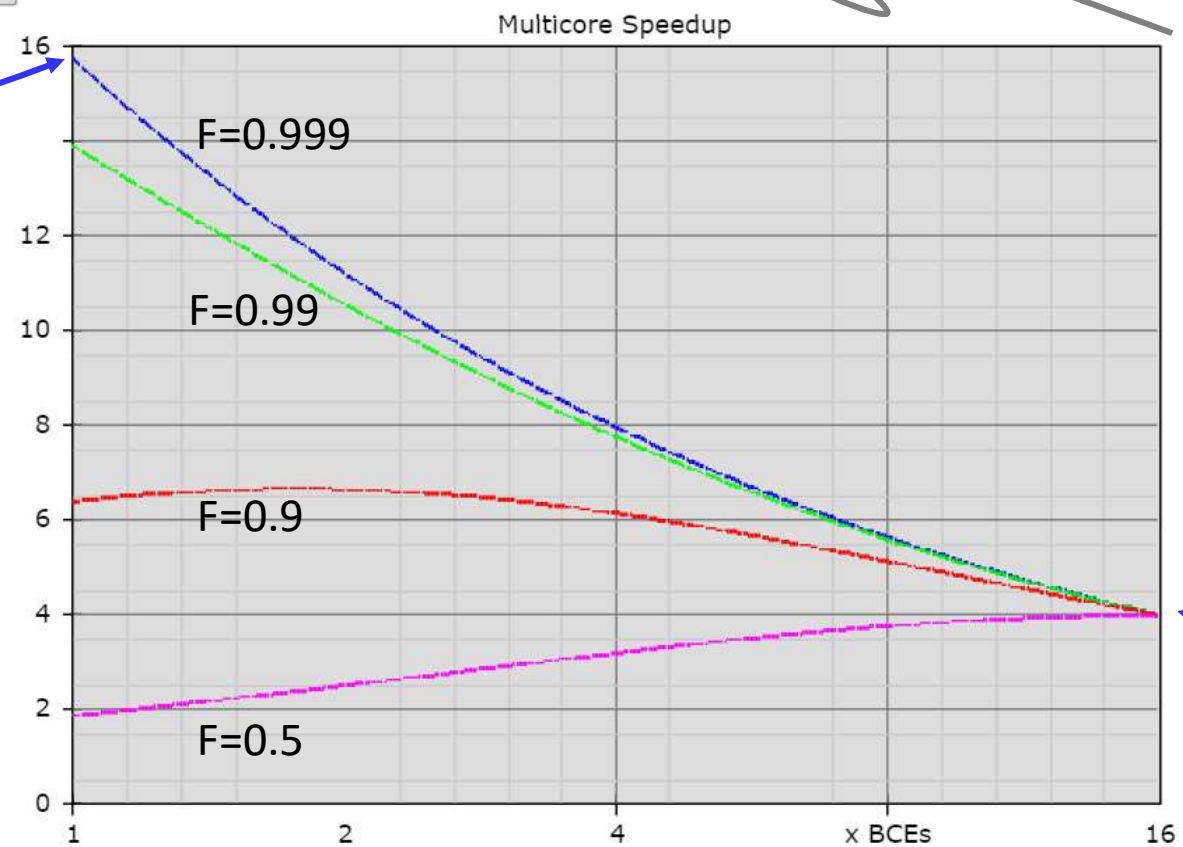
Line 1: ☑ fraction parallel (f): 0.999 perf(x) = sqrt(x) Symmetric ◉ Asymmetric ○ Dynamic ○
Line 2: ☑ fraction parallel (f): 0.99 perf(x) = sqrt(x) Symmetric ◉ Asymmetric ○ Dynamic ○
Line 3: ☑ fraction parallel (f): 0.9 perf(x) = sqrt(x) Symmetric ◉ Asymmetric ○ Dynamic ○
Line 4: ☑ fraction parallel (f): 0.5 perf(x) = sqrt(x) Symmetric ◉ Asymmetric ○ Dynamic ○

Note: x is the number of BCEs harnessed for faster core(s) (was r in the paper)
Note: Pressing ReDraw when re-enabling disabled lines can be REALLY slow and temporarily tie up the browser

ReDraw

assume perf grows with sqrt of area

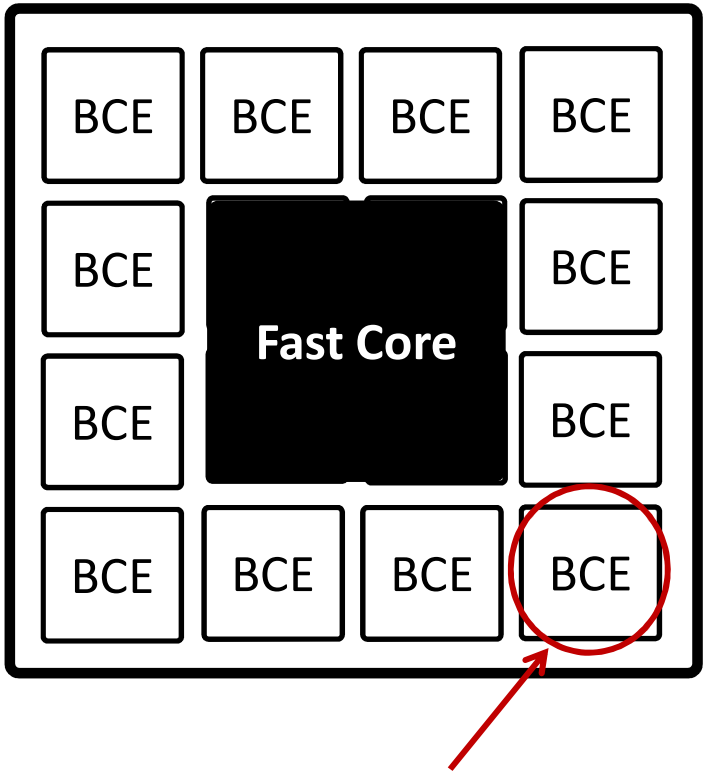
16x 1-BCE cores



1x 16-BCE core

more smaller cores ⟵ size of cores in BCE ⟶ fewer larger cores

Asymmetric Multicores



Base Core Equivalent (BCE) in [Hill and Marty, 2008]

- Pwr/area-efficient "slow" BCEs vs pwr/area-hungry "fast" core
  - fast core for sequential code
  - slow cores for parallel sections
- [Hill and Marty, 2008]

$$Speedup = \frac{1}{\frac{1-f}{perf_{seq}} + \frac{f}{(n-r)+perf_{seq}}}$$

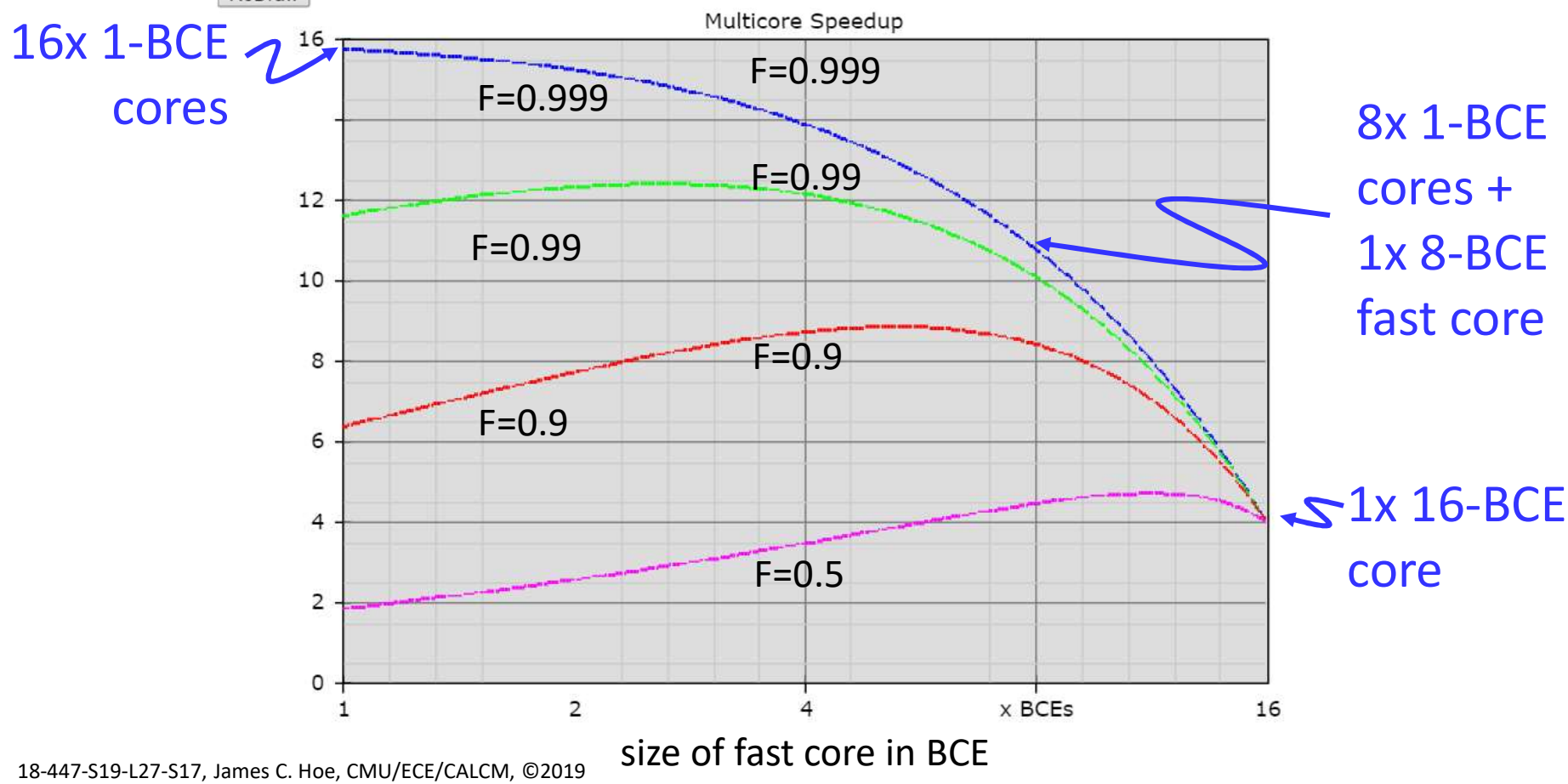
  - r = cost of fast core in BCE
  - $perf_{seq}$ = speedup of fast core over BCE
  - solve for optimal die allocation

http://research.cs.wisc.edu/multifacet/amdahl/

Line 1: ☑ fraction parallel (f): 0.999 perf(x) = sqrt(x) Symmetric ○ Asymmetric ◉ Dynamic ○
Line 2: ☑ fraction parallel (f): 0.975 perf(x) = sqrt(x) Symmetric ○ Asymmetric ◉ Dynamic ○
Line 3: ☑ fraction parallel (f): 0.9 perf(x) = sqrt(x) Symmetric ○ Asymmetric ◉ Dynamic ○
Line 4: ☑ fraction parallel (f): 0.5 perf(x) = sqrt(x) Symmetric ○ Asymmetric ◉ Dynamic ○

Note: x is the number of BCEs harnessed for faster core(s) (was r in the paper)
Note: Pressing ReDraw when re-enabling disabled lines can be REALLY slow and temporarily tie up the browser

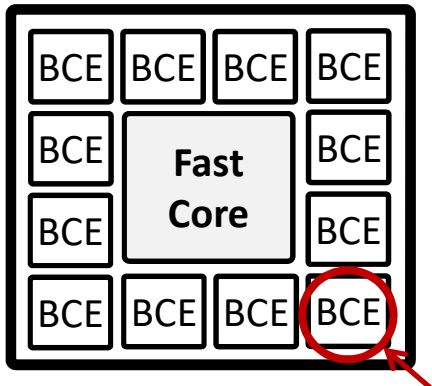
ReDraw



Heterogeneous Multicores

[Chung, et al. MICRO 2010]

Asymmetric



Base Core Equivalent

$$Speedup = \frac{1}{\frac{1-f}{perf_{seq}} + \frac{f}{(n-r)}}$$

[Hill and Marty, 2008] simplified

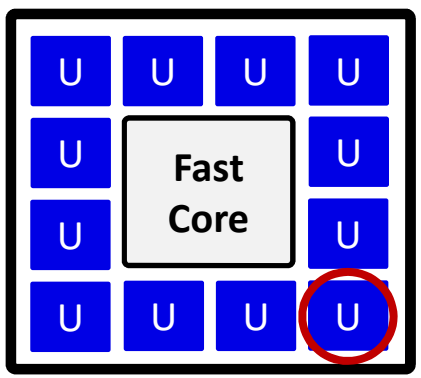
f is fraction parallelizable

n is total die area in BCE units

r is fast core area in BCE units

$perf_{seq}(r)$ is fast core perf. relative to BCE

Heterogeneous

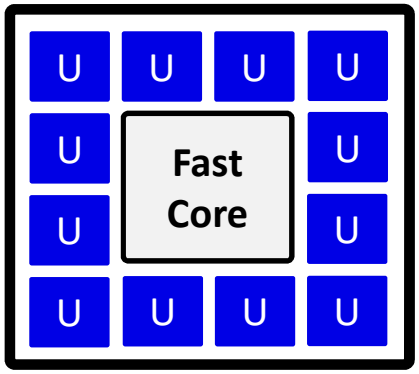


$$Speedup = \frac{1}{\frac{1-f}{perf_{seq}} + \frac{f}{\mu \times (n-r)}}$$

For the sake of analysis, break the area for GPU/FPGA/etc. into units of **U-cores** that are the same size as BCEs. Each U-core type is characterized by a relative performance μ and relative power φ compared to a BCE

Modeling Power and Bandwidth Budgets

Heterogeneous



$$Speedup = \frac{1}{\frac{1-f}{perf_{seq}} + \frac{f}{\mu \times (n-r)}}$$

- The above is based on area alone
- Power or bandwidth budget limits the usable die area
  - if P is total power budget expressed as a multiple of a BCE's power,

    usable U-core area $n - r \le P/\phi$
  - if B is total memory bandwidth expressed as a multiple of BCEs,

    usable U-core area $n - r \le B/\mu$

φ and μ example values

		MMM	Black-Scholes	FFT-2^{10}
Nvidia GTX285	Φ	0.74	0.57	0.63
	μ	3.41	17.0	2.88
Xilinx LX760	Φ	0.31	0.26	0.29
	μ	0.75	[illegible]	[illegible]
Custom Logic	Φ	0.79	4.75	4.96
	μ	27.4	482	489

On equal area basis, 3.41x performance at 0.74x power relative a BCE

Nominal BCE based on an Intel Atom in-order processor, 26mm^2 in a 45nm process

Case Study [Chung, MICRO 2010]

	CPU	GPUs		FPGA	ASIC
	Intel Core i7-960	Nvidia GTX285	ATI R5870	Xilinx V6-LX760	Std. Cell
Year	2009	2008	2009	2009	2007
Node	45nm	55nm	40nm	40nm	65nm
Die area	263mm^2	470mm^2	334mm^2	-	-
Clock rate	3.2GHz	1.5GHz	1.5GHz	0.3GHz	-

Single-prec. floating-point apps				
M-M-Mult	MKL 10.2.3 Multithreaded	CUBLAS 2.3	CAL++	hand-coded
FFT	Spiral.net Multithreaded	CUFFT 2.3 3.0/3.1	-	Spiral.net
Black-Scholes	PARSEC multithreaded	CUDA 2.3	-	hand-coded

"Best-Case" Performance and Energy

	Device	GFLOP/s actual	(GFLOP/s)/mm^2 normalized to 40nm	GFLOP/J normalized to 40nm
MMM	Intel Core i7 (45nm)	96	0.50	1.14
	Nvidia GTX285 (55nm)	425	2.40	6.78
	ATI R5870 (40nm)	1491	5.95	9.87
	Xilinx V6-LX760 (40nm)	204	0.53	3.62
	same RTL std cell (65nm)	---	19.28	50.73

- CPU and GPU benchmarking is compute-bound; FPGA and Std Cell effectively compute-bound (no off-chip I/O)
- Power (switching+leakage) measurements isolated the core from the system
- For detail see [Chung, et al. MICRO 2010]

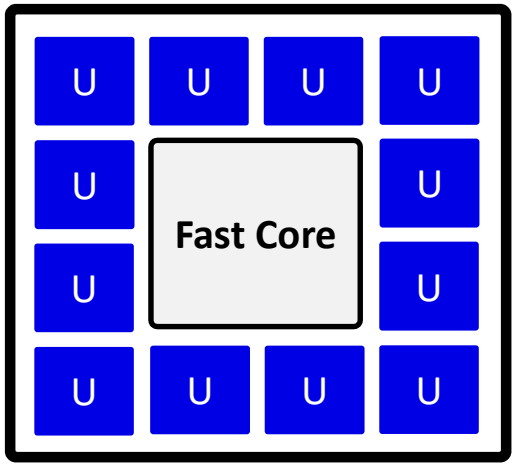
Less Regular Applications

		GFLOP/s	(GFLOP/s)/mm^2	GFLOP/J
FFT-2^{10}	Intel Core i7 (45nm)	67	0.35	0.71
	Nvidia GTX285 (55nm)	250	1.41	4.2
	ATI R5870 (40nm)	-	-	-
	Xilinx V6-LX760 (40nm)	380	0.99	6.5
	same RTL std cell (65nm)	952	239	90

		Mopt/s	(Mopts/s)/mm^2	Mopts/J
Black-Scholes	Intel Core i7 (45nm)	487	2.52	4.88
	Nvidia GTX285 (55nm)	10756	60.72	189
	ATI R5870 (40nm)	-	-	-
	Xilinx V6-LX760 (40nm)	7800	20.26	138
	same RTL std cell (65nm)	25532	1719	642.5

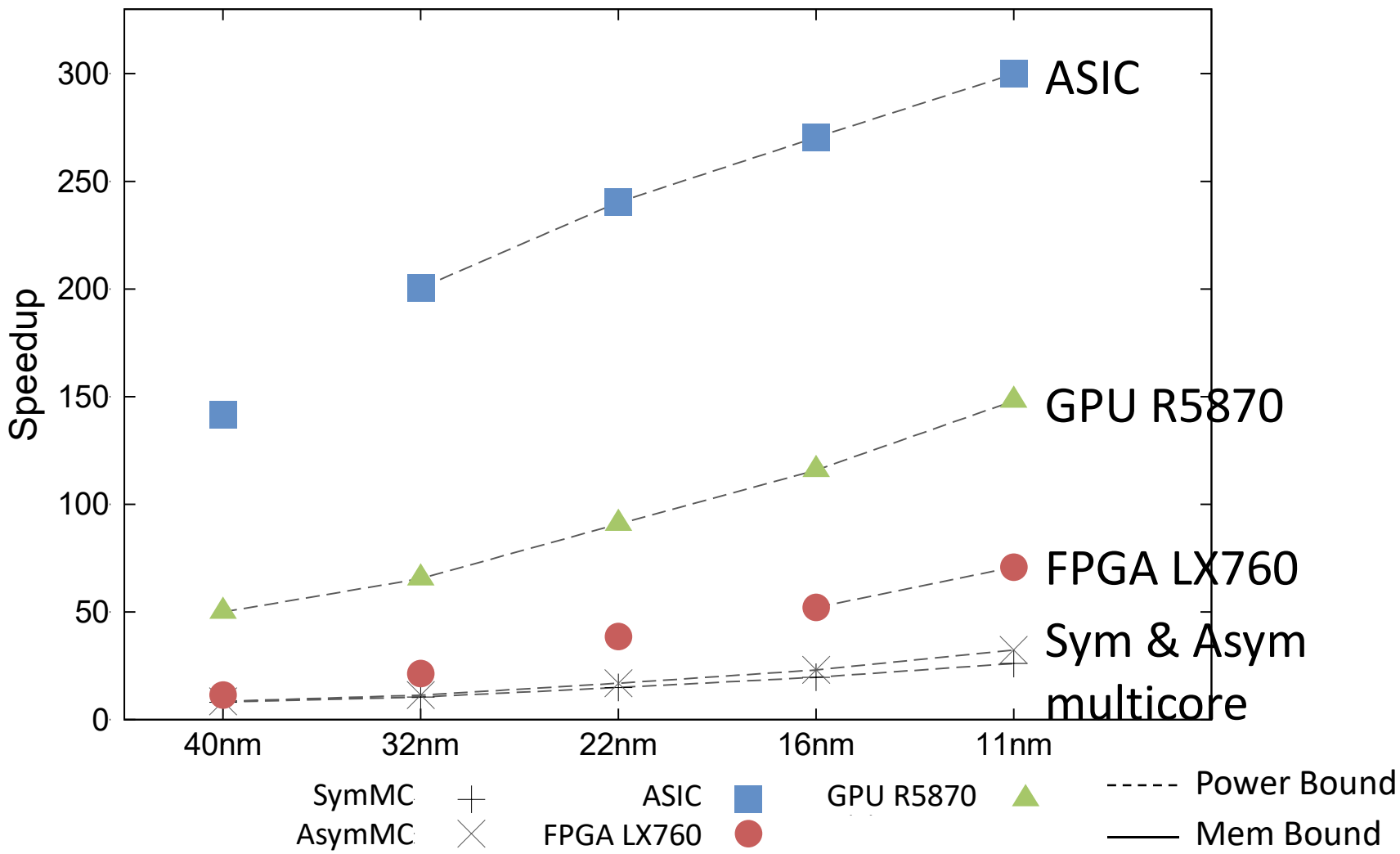
Combine Model with ITRS Trends

Year	2011	2013	2016	2019	2022
Technology	40nm	32nm	22nm	16nm	11nm
Core die budget (mm^2)	432	432	432	432	432
Normalized area (BCE)	19	37	75	149	298 (16x)
Core power (W)	100	100	100	100	100
Bandwidth (GB/s)	180	198	234	234	252 (1.4x)
Rel pwr per device	1X	0.75X	0.5X	0.36X	0.25X

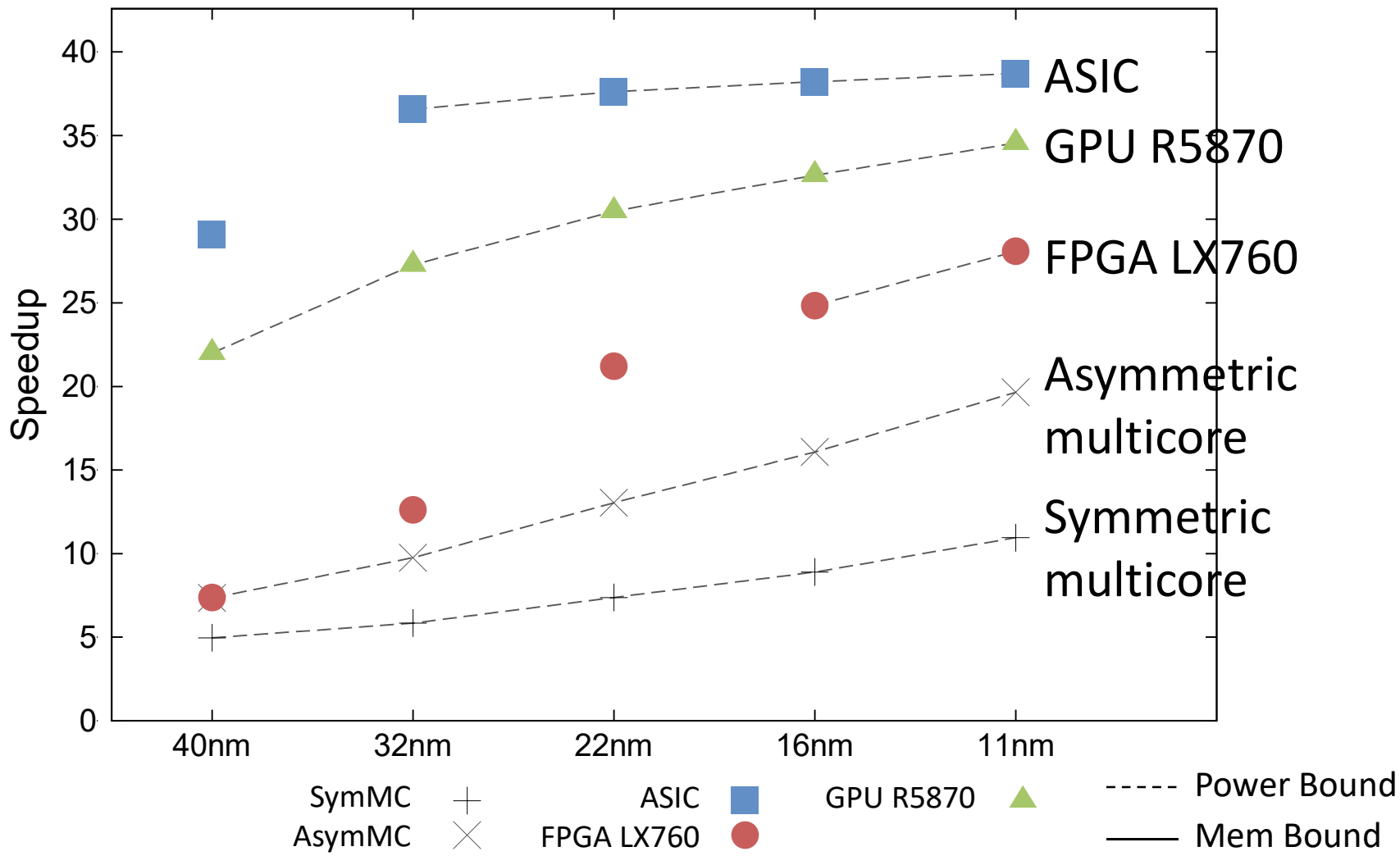


- 2011 parameters reflect high-end systems of the day; future parameters extrapolated from ITRS 2009
- $432mm^2$ populated by an optimally sized Fast Core and U-cores of choice

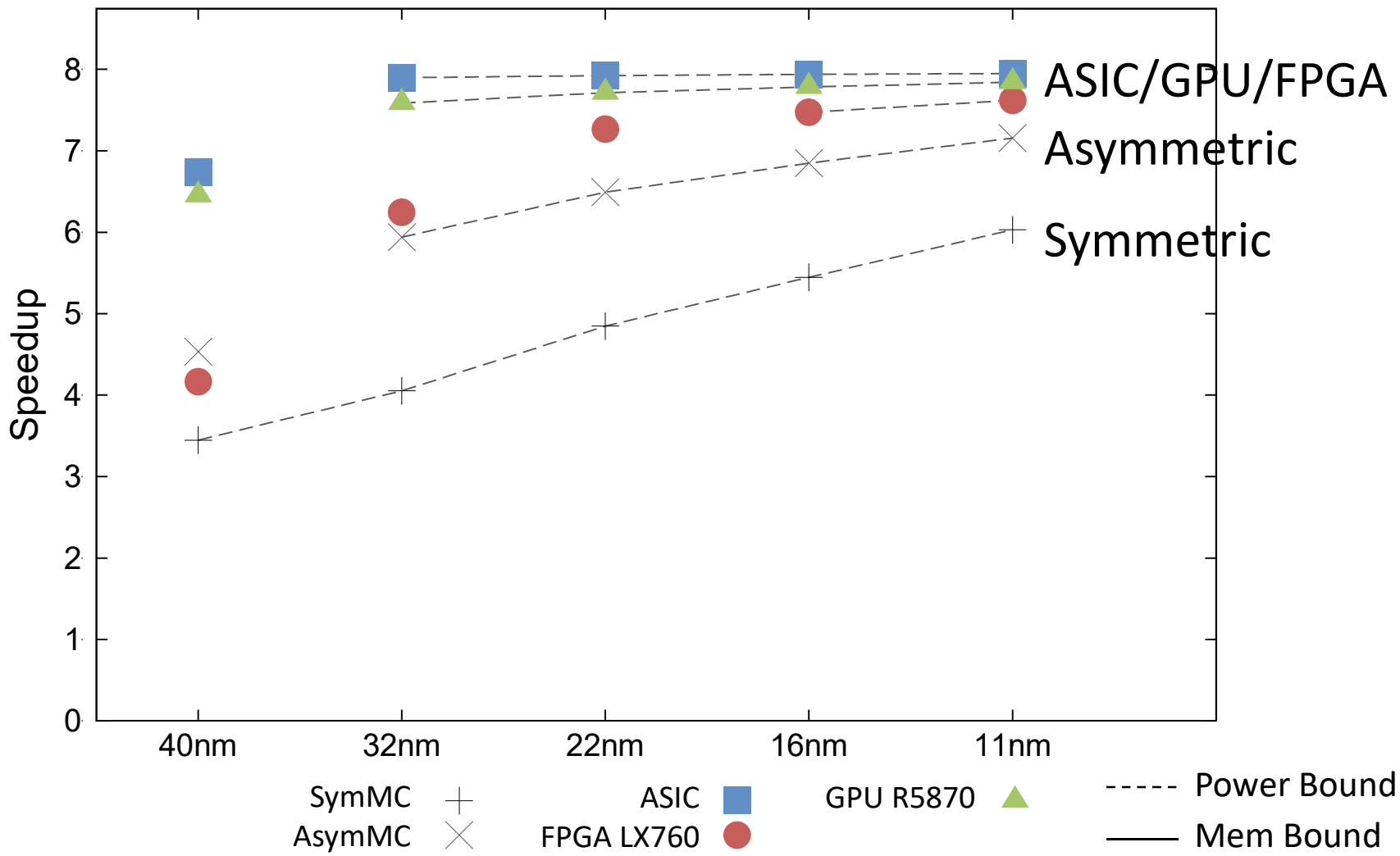
Single-Prec. MMMult (f=99%)



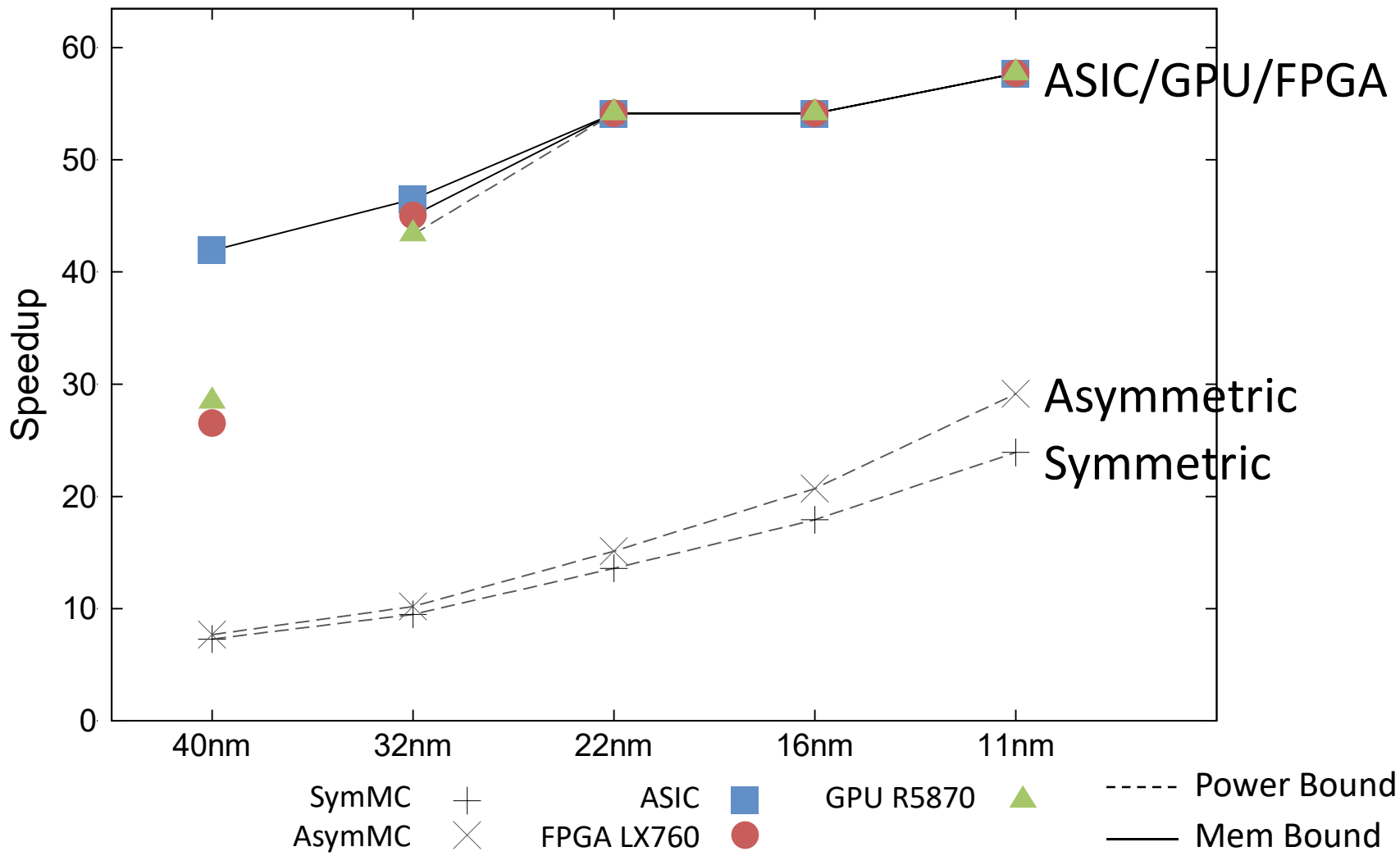
Single-Prec. MMMult (f=90%)



Single-Prec. MMMult (f=50%)

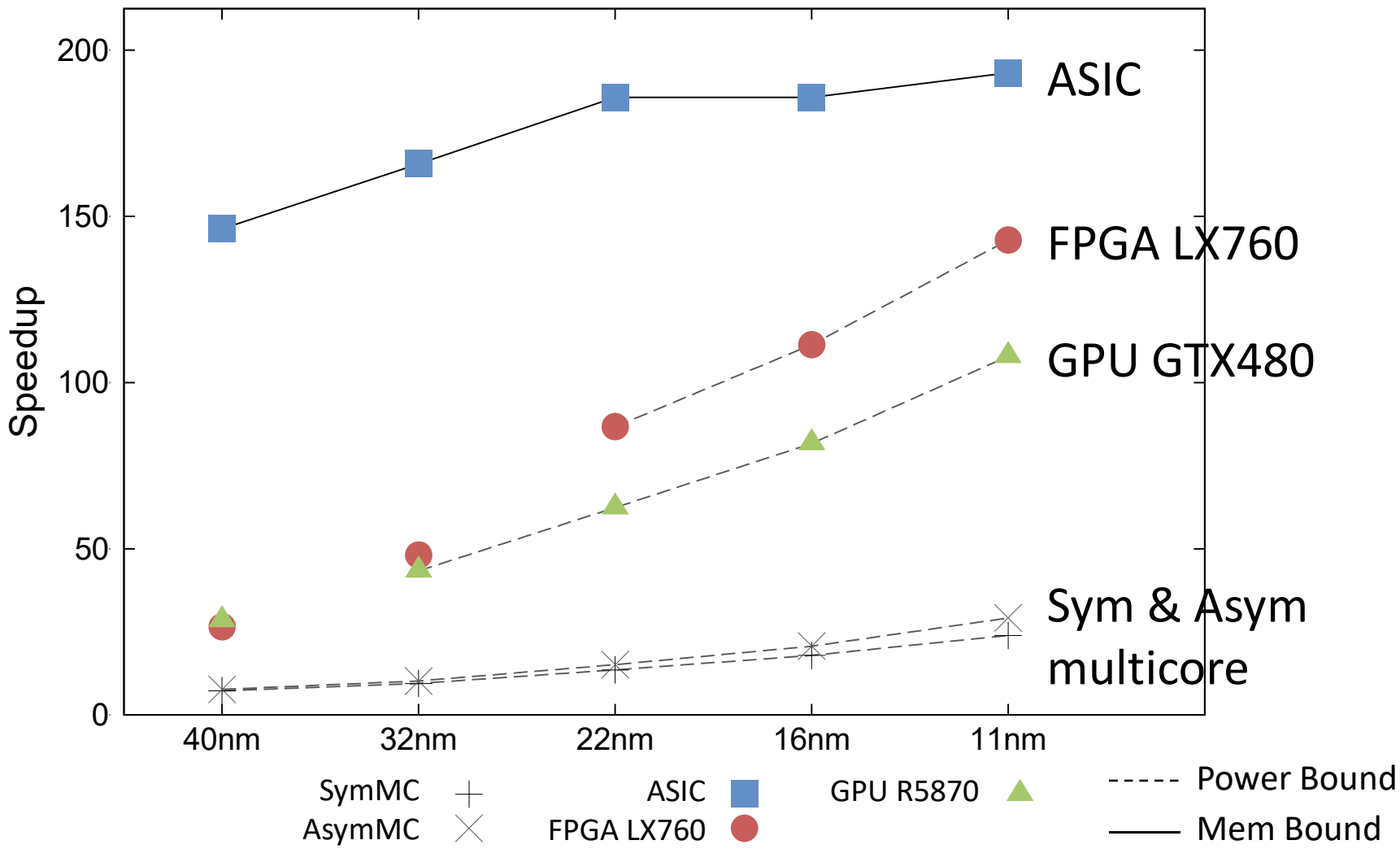


Single-Prec. FFT-1024 (f=99%)



FFT-1024 (f=99%)

if 1TB/sec memory bandwidth



You will be seeing more of this

- Performance scaling requires improved efficiency in Op/Joules and Perf/Watt
- Hardware acceleration is the most direct way to improve energy/power efficiency
- Need better hardware design methodology to enable application developers (without losing hardware's advantages)
- Software is easy; hardware is hard?

Hardware isn't hard; perf and efficiency is!!!