Introduction to Domain Specific Accelerators

David Andrews
Rm 527 JBHT
dandrews@uark.edu
CSCE University of Arkansas

We have crossed the Rubicon

or why do we need domain specific accelerators ?

Introduction - Framing the shift and key statistics

Dennard Scaling - Why CPUs had a free lunch for 30 years, and why it ended

The three walls - power, memory bandwidth, and instruction-level parallelism

Dark silicon - why we can build transistors we can't afford to run

The GPU case study - from pixels to petaflops, and why it maps to AI

Hyperscaler silicon - Google TPUs, Apple Neural Engine, Amazon Trainium

The economics - why specialization is a business imperative, not just physics

Summary - key takeaways and where the field is heading

Dennard Scaling and the Free Lunch

Why CPUs reigned for 30 years ?

From the early 1970s through the mid-2000s, the semiconductor industry lived by two compounding laws that made every new processor generation better than the last — ***for free !***.

Moore's Law (1965)

— Transistor count doubles roughly every two years
— Driven by photolithography shrinks
— Empirical observation, not a law of physics
— Still loosely holds, but slower and at higher cost

Dennard Scaling (1974)

— Shrinking transistors also reduced voltage and power
— Power density stayed constant as transistors shrank
— Meant faster + smaller + cooler, all simultaneously
— Collapsed around 2005–2007

While Dennard Scaling held, software developers got a "free lunch" — just wait for next year's chip and your existing code would run faster, without any changes.

Dennard Scaling and the Free Lunch

A Timeline

Herb Sutter's 2005 article "The Free Lunch Is Over" announced the end of this era and signaled that software developers could no longer rely on hardware improvements alone.
http://www.gotw.ca/publications/concurrency-ddj.htm

1974 -> **Dennard Scaling formalized**
Robert Dennard's paper shows voltage, current, and power all scale with transistor geometry.

2004 -> **Intel cancels Tejas / Prescott burns**
The Pentium 4 hits the power wall. Intel pivots to multicore instead of higher clock speeds.

2005 -> **Dennard Scaling effectively ends**
Leakage current at small nodes means shrinking transistors no longer reduces power density.

Three Physical Walls Halted the CPU

Power, memory bandwidth, and instruction-level parallelism

The end of Dennard Scaling exposed three interrelated barriers that no amount of clever CPU microarchitecture could fully overcome.

The Power Wall

Clock frequency scales with voltage. Reducing voltage causes leakage. At ~90nm nodes, static power consumption explodes. More transistors = more heat, with no free escape.

The Memory Wall

CPU speed improved at ~55%/year; DRAM latency improved at only ~7%/year. By 2005, the processor was spending most of its time waiting for memory, not computing.

The ILP Wall

Out-of-order execution, branch prediction, speculative execution — all squeeze more instruction-level parallelism. But the returns are diminishing; serial bottlenecks in code set a hard ceiling (Amdahl's Law).

These walls explain why CPU single-thread performance has barely improved since ~2012, despite continued transistor scaling. The transistors are there — but they cannot be converted into raw single-thread speed anymore.

> *The response was multicore CPUs — but this pushed complexity onto software. Parallel programs are hard to write. And generic parallelism is still inefficient for many workloads.*

~5GHz

CPU clock speed ceiling since 2004, despite 20+ years of transistor scaling

100ns

DRAM latency today — roughly the same as in 2000, when CPUs were 10× slower

~2%

Annual single-thread CPU perf improvement since 2015, down from ~52%/year in the 1990s

Dark Silicon and Specialization

Why we can build transistors we cannot afford to power

Transistors kept shrinking. Billions, then tens of billions, then hundreds of billions per chip. But power limits mean that at any given moment, only a fraction of those transistors can be active.

> *This is the **dark silicon** problem: like dark matter, billions of transistors sit on a chip doing nothing because powering them all simultaneously would melt the package.*

A 2011 landmark paper by Esmaeilzadeh et al. quantified it starkly: at the 8nm node, up to 50–80% of a chip's transistors must remain powered off at any given moment under realistic thermal constraints.

> *"If you have to turn off most of your chip, you might as well build a chip that does one thing very well."*

This is the economic and physical argument for specialization. Instead of a big flexible CPU core that does many things adequately, build many small specialized engines. Each is optimized for its domain and can run at its efficiency sweet spot.

Fixed-function units
Hardwired logic for one operation (e.g., H.264 decode). Extremely efficient, but inflexible.

SIMD / vector units
Process many data elements with one instruction. Already inside CPUs; GPUs take this to extremes.

Programmable accelerators
Like GPUs and TPUs: flexible enough for a domain, specialized enough to blow away a general CPU.

Full custom ASICs
Designed for one exact workload (e.g., Bitcoin mining, radar signal processing). Maximum efficiency, zero flexibility.

The GPU: From Pixels to Petaflops

How a graphics chip became the backbone of AI

The GPU's path from niche graphics card to the most valuable chip architecture in history is the clearest case study in domain specialization winning over general-purpose design.

1999
NVIDIA coins "GPU"
The GeForce 256 is marketed as a "geometry + rasterization" processor — thousands of small cores running the same shading pipeline in parallel.

2006
CUDA released
NVIDIA exposes GPU hardware to general programmers. Researchers immediately notice the GPU's massive parallelism maps well to scientific simulations and linear algebra.

2012
AlexNet and the deep learning explosion
Alex Krizhevsky trains a deep neural network on two GTX 580 GPUs, winning ImageNet by a massive margin. Deep learning = large matrix multiplications. GPUs = matrix multiplication machines.

2016
Tensor cores introduced (Volta)
NVIDIA adds specialized matrix-multiply-accumulate units to the GPU die — a DSA within a DSA. A single Tensor Core performs a 4×4 matrix multiply in one clock cycle.

2022–present
H100, B200 — trillion-dollar market cap
The GPU becomes the scarce resource powering the AI revolution. NVIDIA's valuation exceeds that of every traditional semiconductor company combined.

> *The GPU's secret: deep learning is overwhelmingly matrix-matrix and matrix-vector multiplications. A GPU's thousands of small cores, designed for pixel shaders, happen to be near-perfect for this workload.*

Why can a GPU do 1000× more matrix multiplications per second than a CPU with the same number of transistors? Because a CPU spends ~85% of its die area on control logic, caches, and out-of-order execution hardware. A GPU spends that area on compute cores instead.

From Claude !!

Google TPUs, Amazon Trainium, Apple Neural Engine

Hyperscalars build their own silicon

By early 2010's hyperscalars realized that even the GPU – itself a domain accelerator – still too general for their specific workloads. The response: build custom silicon optimized for one company's exact models and infrastructure.

Accelerator	Maker	Primary workload	Key design choice
TPU v1 (2015)	Google	Neural network inference	Systolic array for matrix multiply; 8-bit int instead of float
TPU v4 (2022)	Google	LLM training at pod scale	High-bandwidth interconnect between chips; bf16 precision
Trainium / Inferentia	Amazon / AWS	Training + inference for AWS customers	NeuronCore architecture; tight integration with EC2
Neural Engine (2017)	Apple	On-device ML inference	Ultra-low power; co-designed with CPU/ GPU for memory sharing
Gaudi 3 (2024)	Intel	LLM training + inference	High-bandwidth ethernet fabric; open ecosystem

Google's TPU story is instructive: in 2013, Google estimated that if users used voice search for 3 minutes a day with deep learning models, Google would need to double its global data center capacity — at GPU prices. The TPU was the answer.

Each hyperscaler has a dominant workload (transformer inference, image recognition, signal processing) with known data types, known model sizes, and known batch shapes. Designing a chip around those specifics yields enormous efficiency gains.

The Economic and Ecosystem of Specialization

Why it's not just physics – it's business!

Physical necessity explains why specialization became possible. Economics explains why it became inevitable. Three forces converged in the 2010s to make DSA investment rational at scale.

Workload concentration
Cloud computing concentrated massive, uniform workloads in a few hyperscalers. Serving billions of identical inference requests makes custom silicon economically viable.

Chip NRE amortization
A 5nm ASIC costs $500M–$1B to design. At 10M units or millions of server-days, that cost per computation becomes negligible versus operational savings.

Energy cost dominance
At hyperscale, electricity is the dominant operational cost. A 10× improvement in TOPS/Watt translates directly to 10× reduction in electricity costs per inference.

Software maturity
CUDA, PyTorch, JAX, XLA — mature ML frameworks now abstract hardware. Programmers no longer write assembly. DSAs can be deployed without silicon expertise.

> *"The best way to go fast is to do less — and to do the right less."*

This has also created new market dynamics: vertical integration (Apple designing its own CPU + GPU + NPU + ISP all on one die), hyperscaler competition (Google vs. Amazon vs. Microsoft each building their own AI chips), and new entrants (Cerebras, Groq, Graphcore, SambaNova) challenging NVIDIA.

> *The risk of specialization: a DSA that's perfectly tuned for today's transformer architecture may be obsolete if the dominant model architecture changes. This is why software-programmable DSAs (like GPUs and FPGAs) still dominate over pure fixed-function ASICs.*

The Architecture of the AI Age

What we've learned and where it leads

We've traced a clear arc: from the free-lunch era of Dennard Scaling through the physical walls that stopped it, to the rise of accelerators purpose-built for the computations that matter most today.

General purpose CPU	Domain-specific accelerator
+ Runs any code without modification	+ 10–1000× better throughput for target workload
+ Optimized for low-latency serial tasks	+ Far superior TOPS/Watt
+ Large software ecosystem	+ Smaller die area per operation
— Poor energy efficiency for parallel math	— Limited or no flexibility outside domain
— Most transistors are control, not compute	— High NRE design cost
— Single-thread perf improvement near stagnant	— Requires domain-specific toolchain / compilers

The key forces that drove the transition:

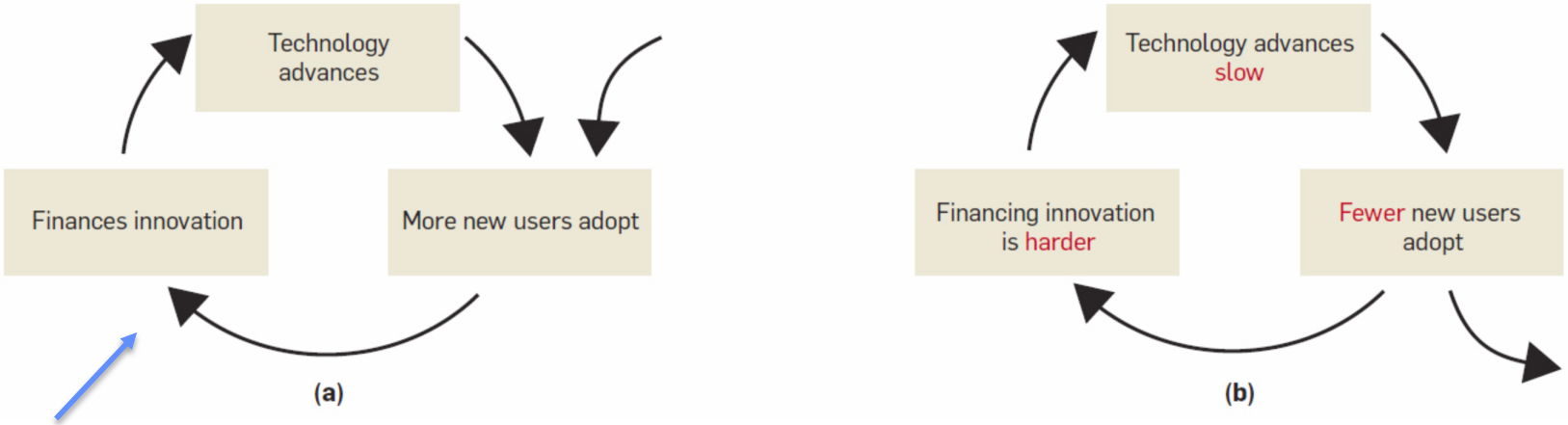
Dennard Scaling ended (~2005) · Dark silicon problem · Deep learning workload explosion · Energy cost at hyperscale · Workload concentration in cloud · GPU → Tensor Core evolution · CUDA and ML frameworks maturing · Hyperscaler vertical integration

> *The future is heterogeneous: a combination of CPU (control flow, serial tasks), GPU/NPU (parallel math), fixed-function accelerators (video, crypto), and memory-near compute (PIM / CIM). No single chip wins everything.*

The deeper lesson is architectural: efficiency comes from matching computation to silicon structure. When your dominant workload is known, building hardware shaped to it is not just an optimization — it is the only viable path forward at the energy and cost budgets modern AI demands.

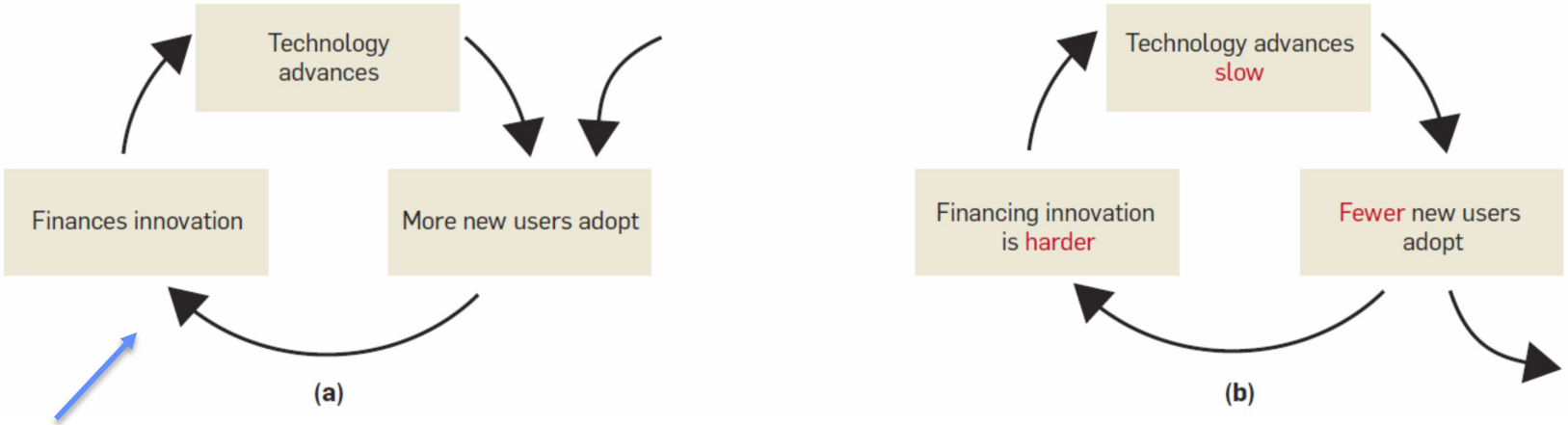
From Claude !!

General Purpose Technology Cycle



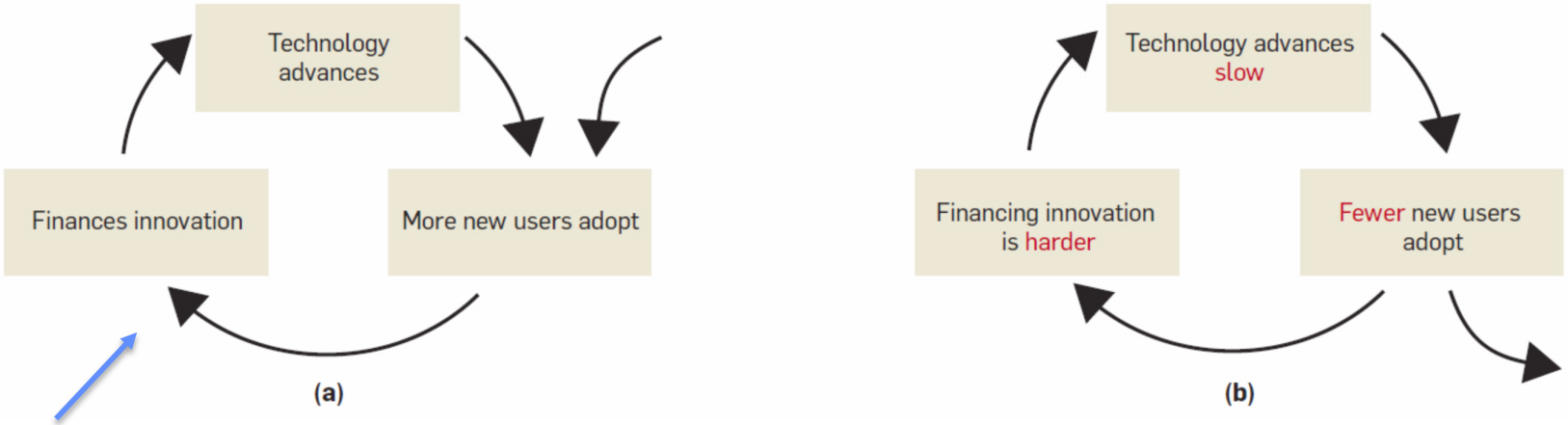
-Moore's Law (transistor scaling) enabled a Unique Virtuous Cycle (VC) which built our Information Technology Economy
-General Purpose Processors: faster, more functionality

General Purpose Technology Cycle



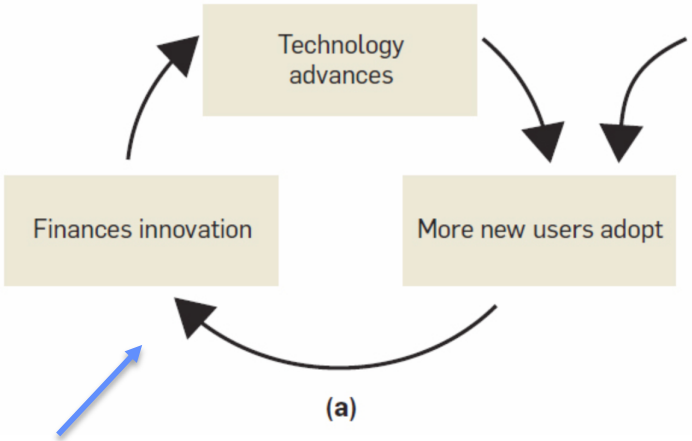
-Moore's Law (transistor scaling) enabled a Unique Virtuous Cycle (VC) which built our Information Technology Economy
-General Purpose Processors: faster, more functionality -> enabled market expansion

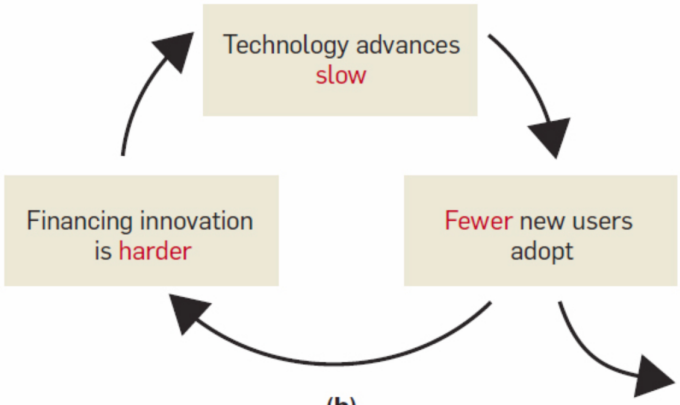
General Purpose Technology Cycle



-Moore's Law (transistor scaling) enabled a Unique Virtuous Cycle (VC) which built our Information Technology Economy
-General Purpose Processors: faster, more functionality -> enabled market expansion -> provided more finances

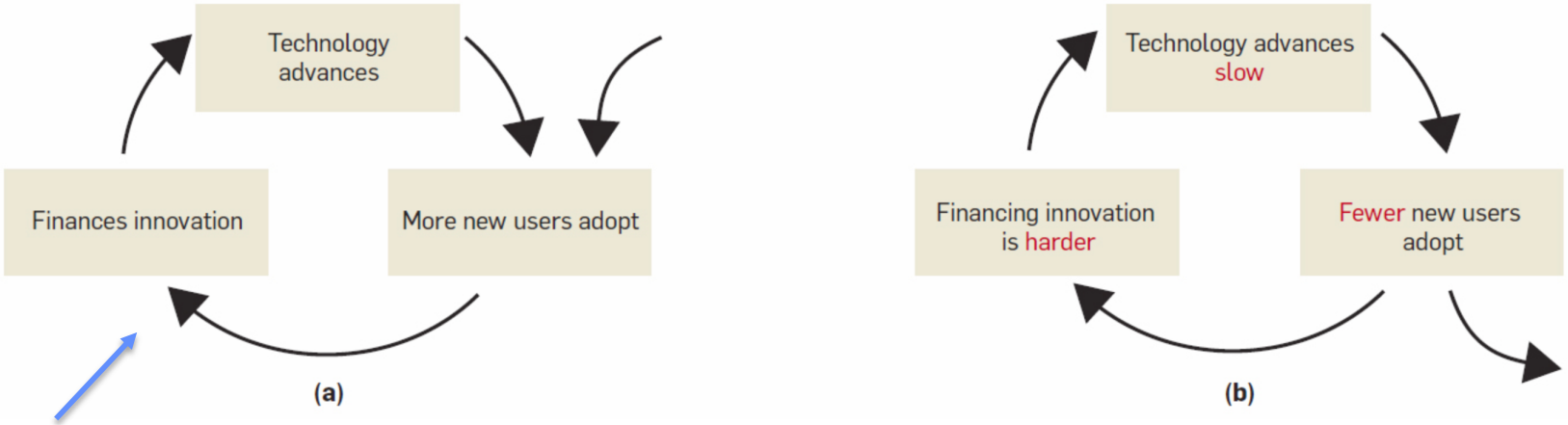
General Purpose Technology Cycle





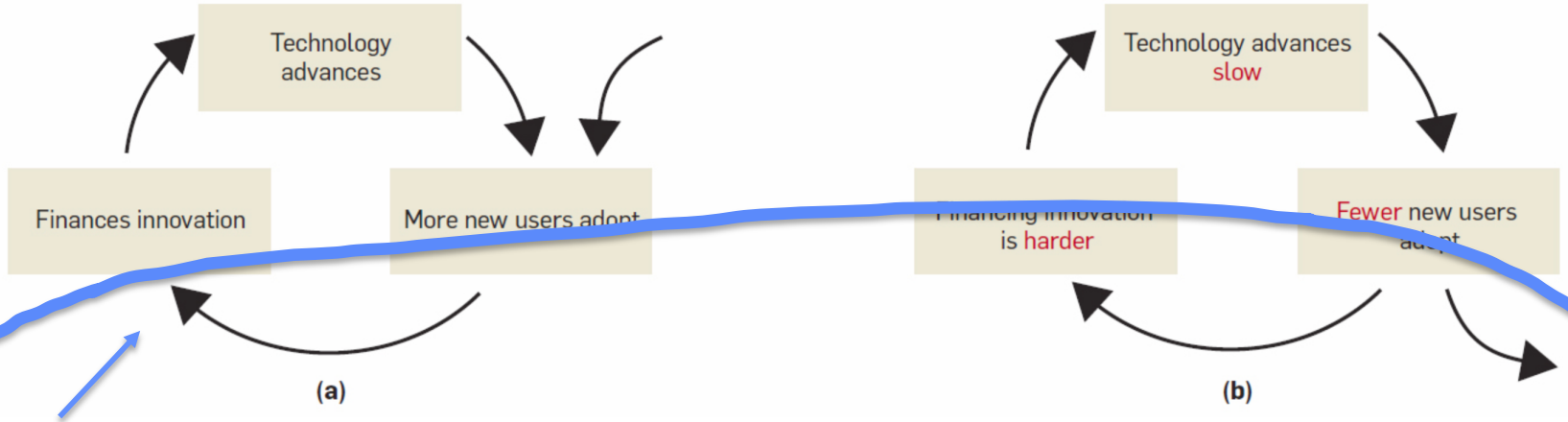
-Moore's Law (transistor scaling) enabled a Unique Virtuous Cycle (VC) which built our Information Technology Economy
-General Purpose Processors: faster, more functionality -> enabled market expansion -> provided more finances -> more technology advances

General Purpose Technology Cycle



-Moore's Law (transistor scaling) enabled a Unique Virtuous Cycle (VC) which built our Information Technology Economy
-General Purpose Processors: faster, more functionality -> enabled market expansion -> provided more finances -> more technology advances -> faster, more functionality…….

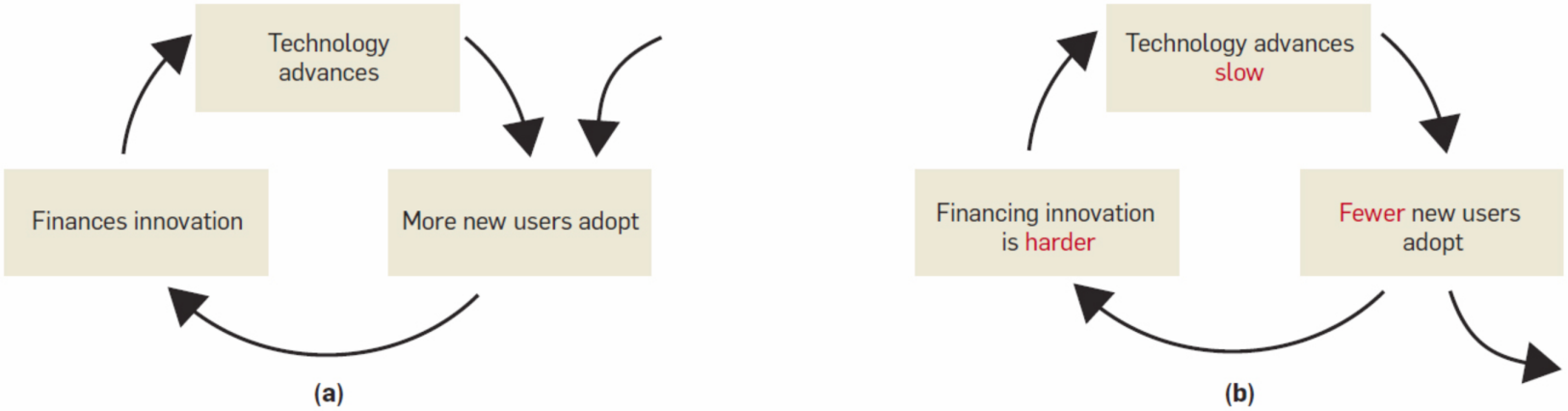
General Purpose Technology Cycle



-Moore's Law (transistor scaling) enabled a Unique Virtuous Cycle (VC) which built our Information Technology Economy

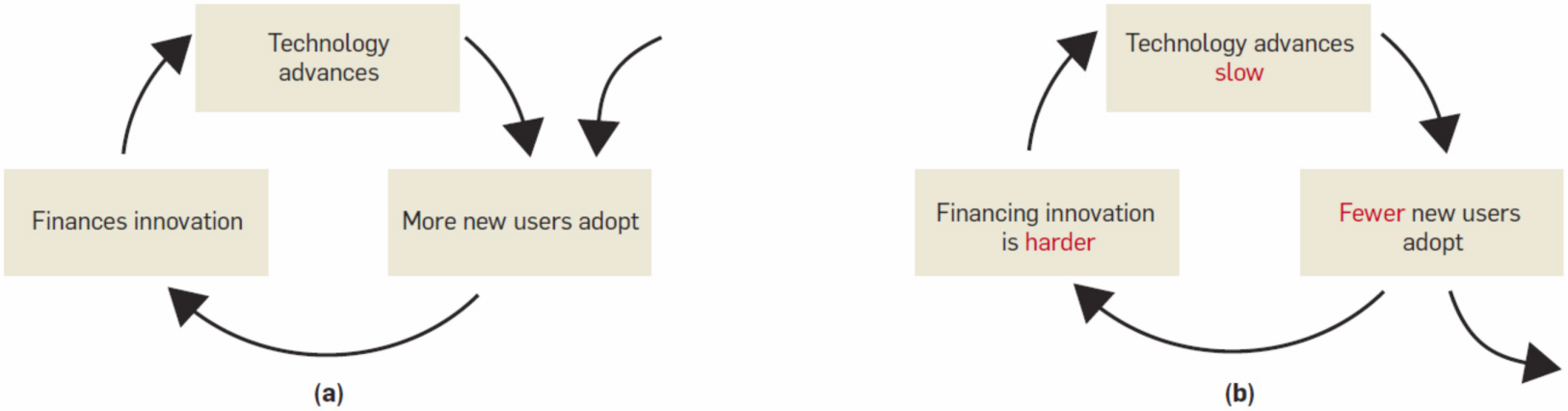
-General Purpose Processors: faster, more functionality -> enabled market expansion -> provided more finances -> more technology advances -> faster, more functionality…….

General Purpose Technology Cycle



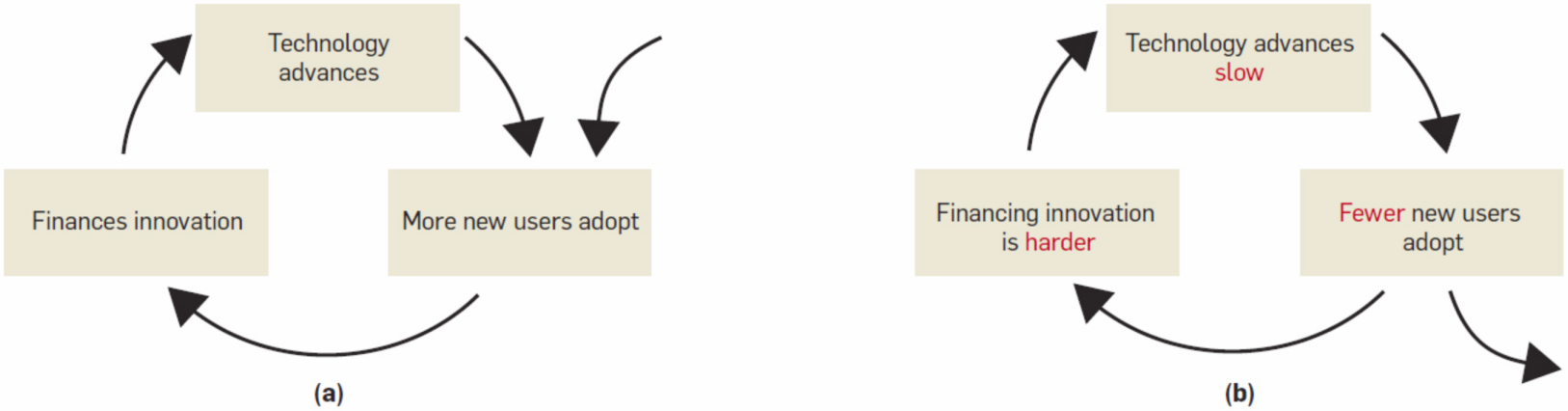
Moore's Law Slows: Technology advances slow

General Purpose Technology Cycle



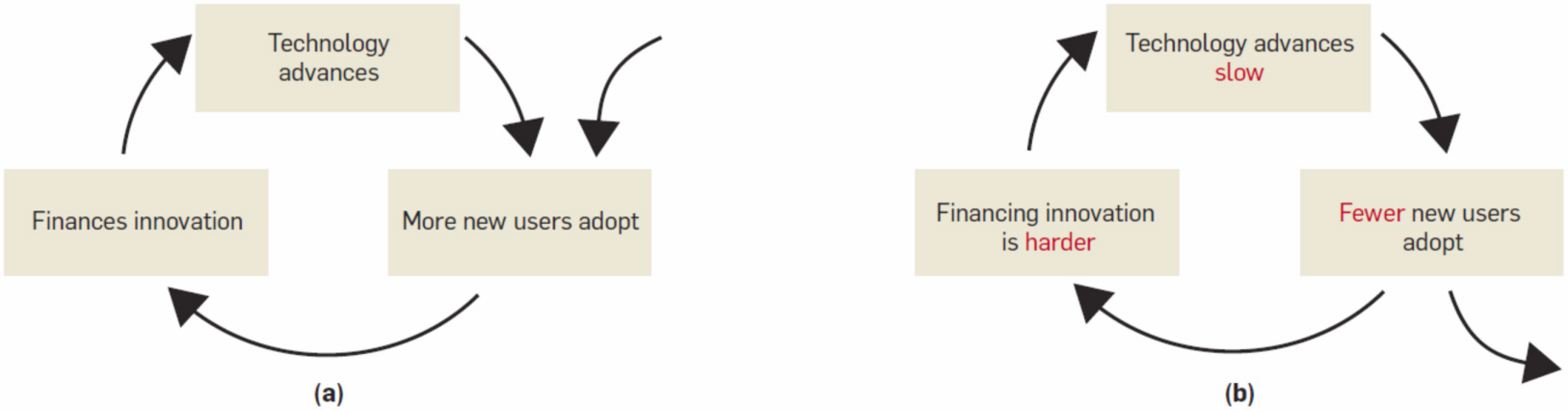
Moore's Law Slows: Technology advances slow -> fewer new users adopt

General Purpose Technology Cycle



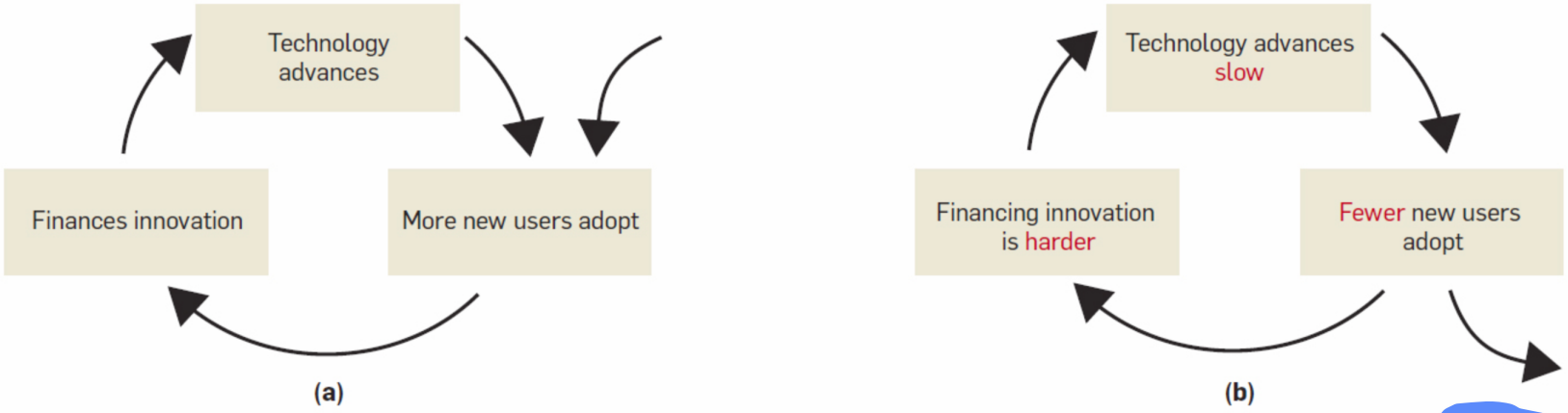
Moore's Law Slows: Technology advances slow -> fewer new users adopt -> less money for innovation

General Purpose Technology Cycle



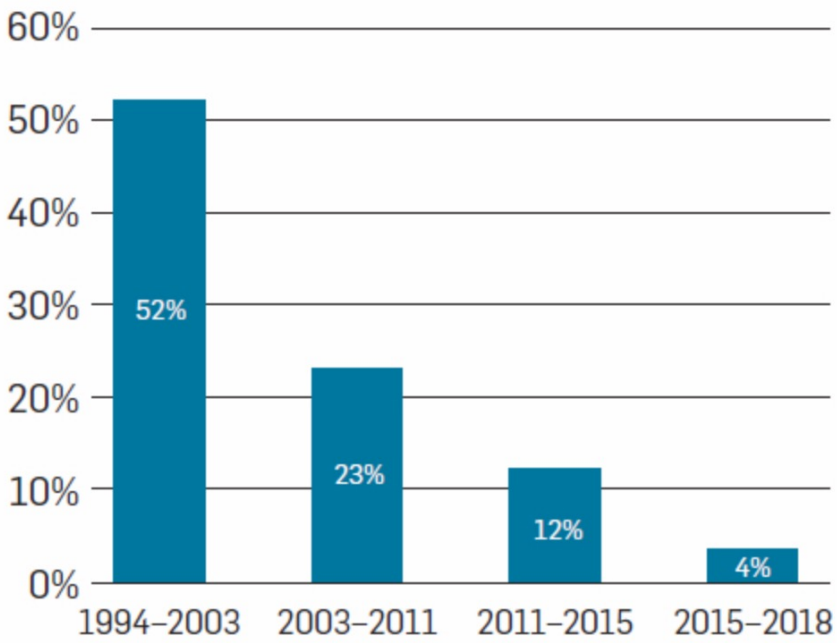
Moore's Law Slows: Technology advances slow -> fewer new users adopt -> less money for innovation -> technology advances slow more

General Purpose Technology Cycle

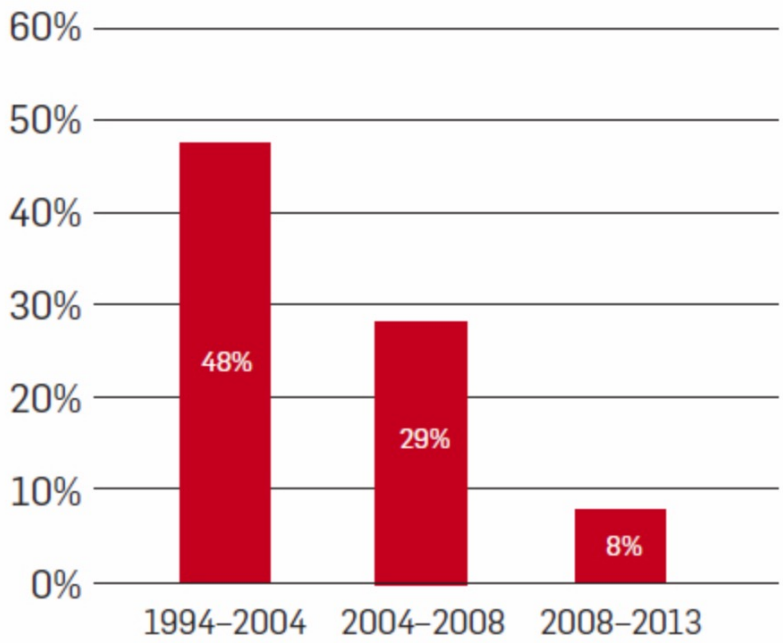


Moore's Law Slows: Technology advances slow -> fewer new users adopt -> less money for innovation -> technology advances slow more -> ……

As Performance Improvements Slow…..



(a)
Microprocessor performance improvement



(b)
Microprocessor performance per dollar improvement

What other factors have affected slowing of improvement?

N. Thompson, S. Spanuth,"*The Decline of Computers as a General Purpose Technology*" CACM March 2021

"That's mathematics son! You can argue with me but you can argue with figures"

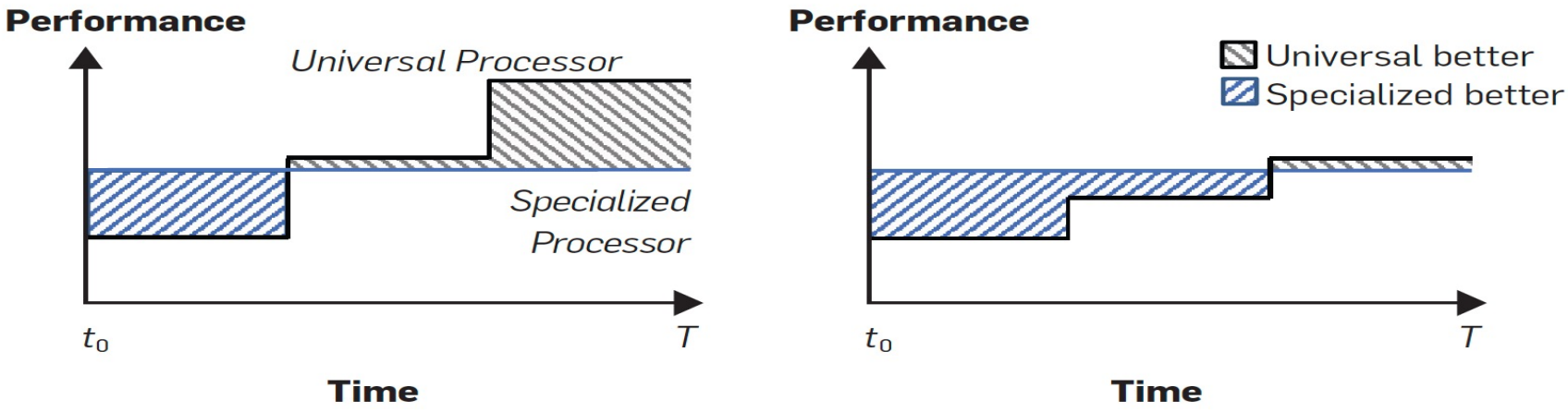


Illustration of processor choice trade-off when universal processor improvement rate is (a) fast or (b) slow

Peak Moore's Law

48% Improvement/year

$$Sp = \frac{P_s}{P_u} = 100 \Rightarrow 83{,}000$$

$$Sp = \frac{P_s}{P_u} = 2x \Rightarrow 1{,}000{,}000$$

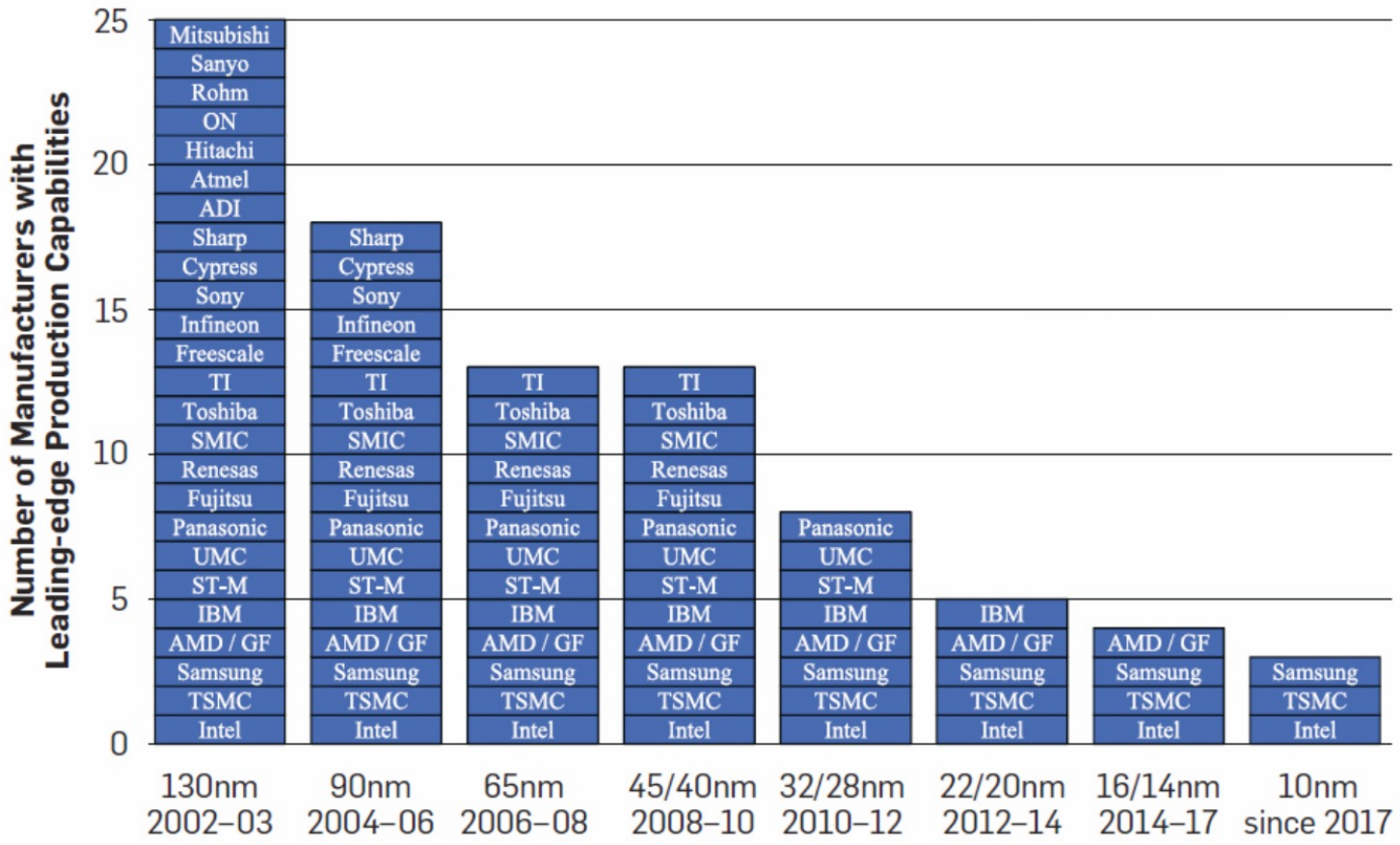
2008-2013

8% Improvement/year

$$Sp = \frac{P_s}{P_u} = 100 \Rightarrow 15{,}000$$

$$Sp = \frac{P_s}{P_u} = 2x \Rightarrow 81{,}000$$

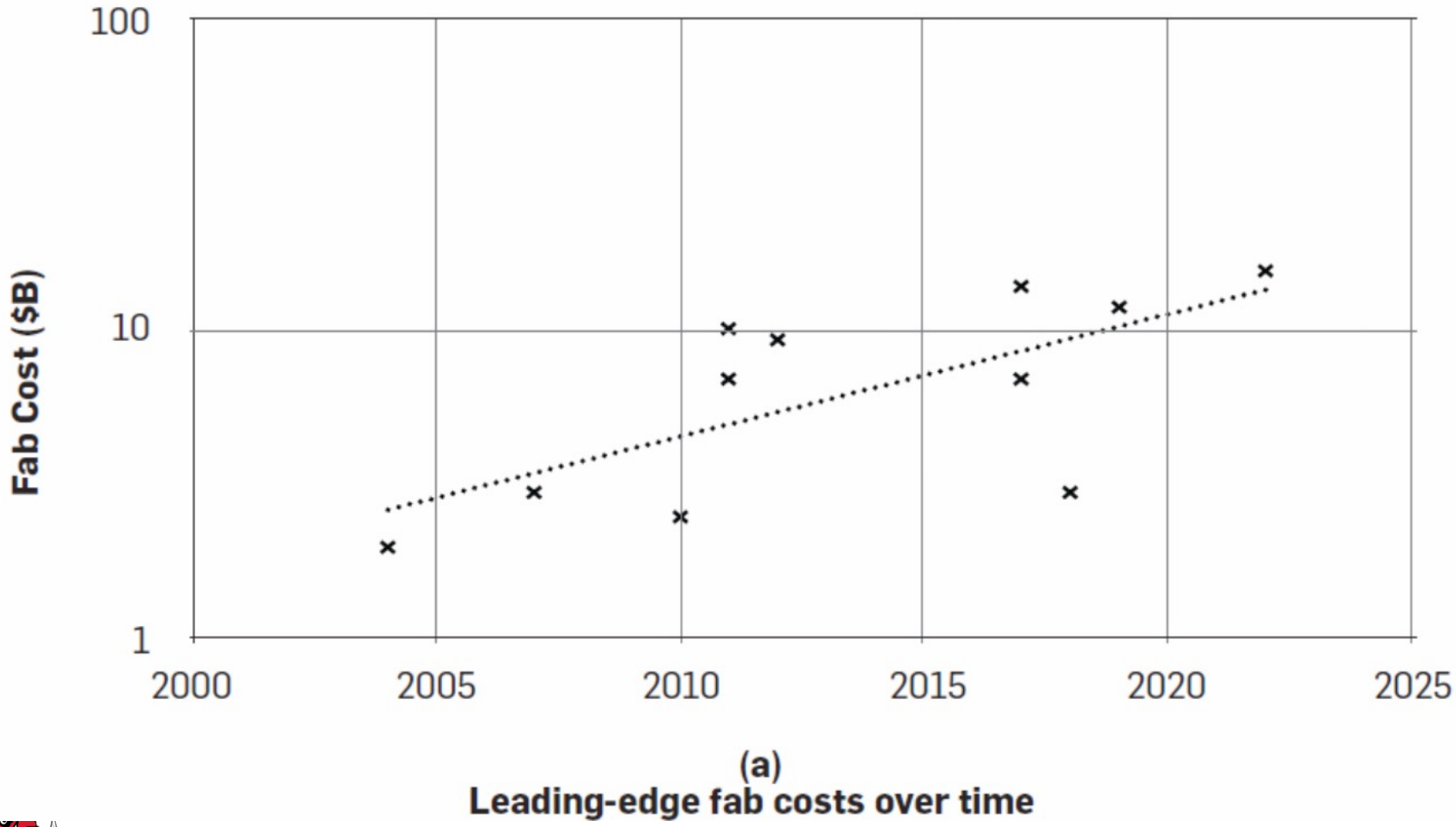
Affects on Manufacturing



(b)

Number of manufacturers with leading-edge production capabilities by node size and year (based on data from Bloom et al.[2appx] and Hemsoth[12appx])

Moore's Law for Fab Costs



N. Thompson, S. Spanuth,"*The Decline of Computers as a General Purpose Technology*" CACM March 2021

What is a Domain Specific Architecture ?

Characteristic	What It Means
Narrowly targeted	Optimized for a specific domain (e.g., ML inference, video encoding, packet processing)
Hardware specialization	Custom datapaths, memory hierarchies, and instructions
High efficiency	Orders-of-magnitude better performance per watt vs CPUs
Limited programmability	Often uses domain-specific APIs, kernels, or compilers
Tightly integrated	Often sits on-die or near-die with CPUs/GPUs

What is a Domain Specific Architecture ?

Custom Datapaths

- Systolic arrays
- Tensor cores
- Cryptographic pipelines
- Bit-manipulation engines
- Fixed-function or semi-programmable units

Specialized Memory Systems

- Multi-banked SRAM
- Scratchpad memories
- Streaming buffers
- Reduced-precision formats (INT8, FP8, BF16)
- Locality-optimized dataflow

What is a Domain?

Machine Learning

- Dense linear algebra
- Tensor operations
- Convolutions
- Activation functions

Video Processing

- Motion estimation
- Transform coding
- Entropy coding

Networking

- Packet Parsing
- Encryption
- Pattern Matching

A domain is a candidate for an accelerator when:

- The workload is high-volume
- Operations are predictable
- Algorithms are stable enough to harden into silicon
- Performance/power gains justify cost

Example Domains

Machine Learning

- Dense linear algebra
- Tensor operations
- Convolutions
- Activation functions

Video Processing

- Motion estimation
- Transform coding
- Entropy coding

Networking

- Packet Parsing
- Encryption
- Pattern Matching

A domain is a candidate for an accelerator when:

- The workload is high-volume
- Operations are predictable
- Algorithms are stable enough to harden into silicon
- Performance/power gains justify cost

Accompanied by a Domain Specific Language:

Python, OpenCL,Pytorch, Tensorflow

Allows expression of the common types of parallelism within the domain

-AI domain dominated by Large Matrix Operations:

i.e., (SIMD) Data level Parallelism

DSL's are good at what they are targeting but are not general purpose

Accompanied by a Domain Specific Language:

Instruction and Programming Models

- Domain-specific instruction sets
- Microcoded kernels
- Graph-level IRs (e.g., MLIR, XLA)
- Limited general programmability
- Often exposed through libraries rather than ISA

What is a Domain Specific Architecture ?

Domain Specific Accelerators exploit four main techniques to get performance and efficiency:

1. Data Specialization: Specialized ops on domain specific data types. Can do in one cycle what may take tens of cycles on GP computer.
2. Exploit Parallelism: Match what is available in the application:
    1. Locality of reference is key
    2. global memory references severely degrade performance

What is a Domain Specific Architecture ?

3. Local and Optimized Memory: Store highly used data structures in small high bandwidth memories close to processing units.

Increase Energy Efficiency

Decrease Processing Latency

4. Reduced Overhead: Specialized hardware and Languages decrease overhead of program interpretation and reduces #instructions.

GP Proc expends ~90% of energy on overhead:

<IF, ID, Data Supply, control>

How Important is Memory Design?

	Unit	Area (mm^2)	(%)	Power (W)	(%)
GACT	Logic	17.6	20.5	1.04	23.6
	Memory	68.0	79.5	3.36	76.4
D-SOFT	Logic	6.2	1.8	0.41	4.4
	Memory	320.3	98.2	8.80	95.6
EIE	Logic	2.8	6.9	0.23	40.3
	Memory	38.0	93.1	0.34	59.7

Area and Power of most accelerators dominated by Memory

-Performance often memory limited

Accelerator Costs

Op	Energy	Area
8-bit Add	10 fJ	4 um^2
Small (8 Kbyte) SRAM Local	50 fJ/bit	.013 um^2 per bit
Larger (100 MB) SRAM Local	.7 pJ/bit	
Global memory	4 pJ/bit	
Local Comm (on Chip)	100fJ/bit-mm	Linear Increase
Global Comm (off Chip)	10 pJ/bit	

$$\text{femto} = 10^{-15}$$

$$\text{pico} = 10^{-12}$$

The Big Three:

- Graphics Processing Units (GPUs)
  - NVIDIA ~88%. (~98% of data center market)
  - AMD ~12%
  - Intel ~0%
- Field Programmable Gate Arrays (FPGAs)
  - Xilinx -> AMD
  - Altera -> Intel -> Altera (split being completed)
- Application-Specific Integrated Circuits (ASICs)
  - Google: Tensor Processing Unit (TPU)
  - Microsoft: Athena this year
  - Amazon Web Services:
  - Some Interesting Startups: Cerebras, Groq

Machine Learning!!