University of Pittsburgh

CS 1541 Introduction

Technology Advances

Wonsun Ahn
Department of Computer Science
School of Computing and Information

Technology Advances

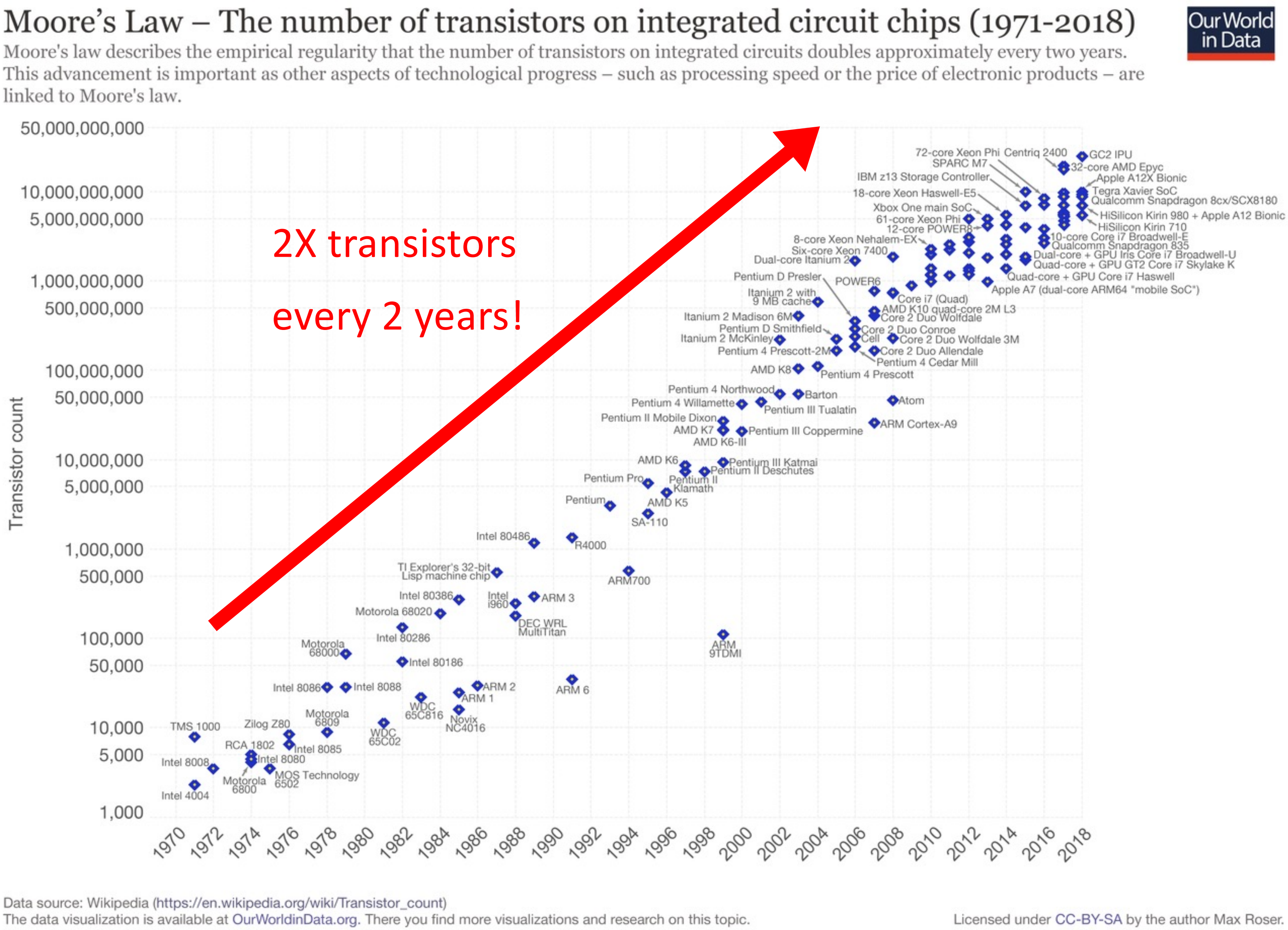
Advances in Technology

- Technology has been advancing at lightning speed
- Architecture and IT as a whole were beneficiaries

- Technology advance is summarized by *Moore's Law*
  - You probably heard of it at some point. Something about …
  - "X doubles every 18-24 months at constant cost"

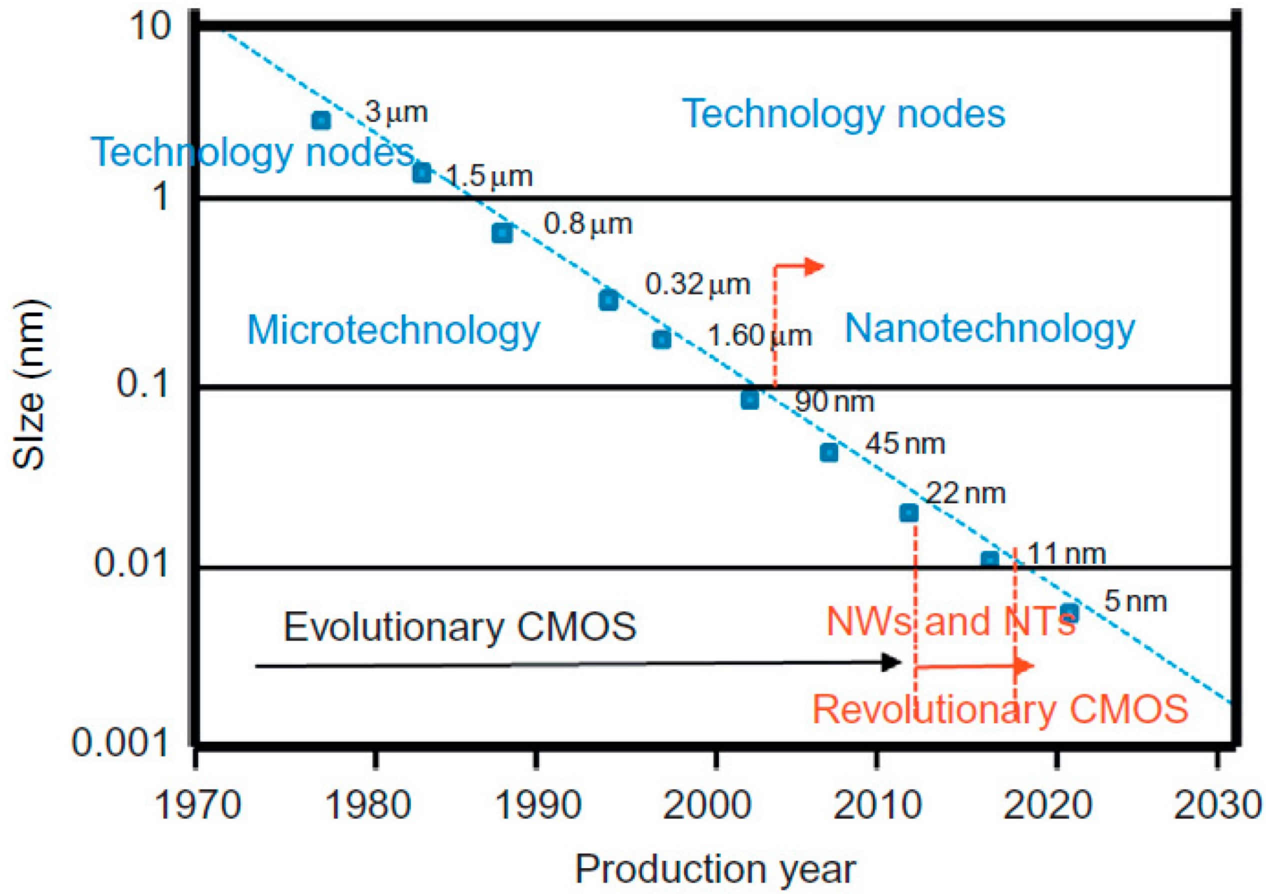
- Is X:
  - CPU performance?
  - CPU clock frequency?
  - Transistors per CPU chip?

Moore’s Law



Moore’s Law – The number of transistors on integrated circuit chips (1971-2018)
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.
Our World in Data
Transistor count
2X transistors
every 2 years!
50,000,000,000
10,000,000,000
5,000,000,000
1,000,000,000
500,000,000
100,000,000
50,000,000
10,000,000
5,000,000
1,000,000
500,000
100,000
50,000
10,000
5,000
1,000
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
Intel 4004
Intel 8008
TMS 1000
RCA 1802
Intel 8080
Motorola 6800
Zilog Z80
Intel 8085
MOS Technology 6502
Motorola 6809
Intel 8086
Intel 8088
Motorola 68000
WDC 65C02
Intel 80186
Intel 80286
WDC 65C816
Motorola 68020
Intel 80386
ARM 1
ARM 2
Novix NC4016
TI Explorer's 32-bit Lisp machine chip
Intel i960
DEC WRL MultiTitan
ARM 3
Intel 80486
R4000
ARM 6
ARM700
Pentium
Pentium Pro
SA-110
AMD K5
AMD K6
Pentium II Klamath
Pentium II Deschutes
Pentium III Katmai
ARM 9TDMI
AMD K6-III
AMD K7
Pentium II Mobile Dixon
Pentium III Coppermine
Pentium 4 Willamette
Pentium III Tualatin
Pentium 4 Northwood
Barton
AMD K8
Pentium 4 Prescott
Atom
ARM Cortex-A9
Pentium 4 Prescott-2M
Itanium 2 McKinley
Pentium D Smithfield
Itanium 2 Madison 6M
Itanium 2 with 9 MB cache
Pentium D Presler
POWER6
Dual-core Itanium 2
Six-core Xeon 7400
8-core Xeon Nehalem-EX
12-core POWER8
61-core Xeon Phi
Xbox One main SoC
18-core Xeon Haswell-E5
IBM z13 Storage Controller
SPARC M7
72-core Xeon Phi
Centriq 2400
GC2 IPU
32-core AMD Epyc
Apple A12X Bionic
Tegra Xavier SoC
Qualcomm Snapdragon 8cx/SCX8180
HiSilicon Kirin 980 + Apple A12 Bionic
HiSilicon Kirin 710
10-core Core i7 Broadwell-E
Qualcomm Snapdragon 835
Dual-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Quad-core + GPU Core i7 Haswell
Apple A7 (dual-core ARM64 "mobile SoC")
Core i7 (Quad)
AMD K10 quad-core 2M L3
Core 2 Duo Wolfdale
Core 2 Duo Conroe
Cell
Core 2 Duo Wolfdale 3M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.
Licensed under CC-BY-SA by the author Max Roser.

Miniaturization of Transistors



Data source: Radamson, H.H.; He, X.; Zhang, Q.; Liu, J.; Cui, H.; Xiang, J.; Kong, Z.; Xiong, W.; Li, J.; Gao, J.; Yang, H.; Gu, S.; Zhao, X.; Du, Y.; Yu, J.; Wang, G. Miniaturization of CMOS. *Micromachines* **2019**, *10*, 293.

- Moore's Law has been driven by transistor miniaturization
  - CPU chip area hasn't changed much

Future of Moore's Law

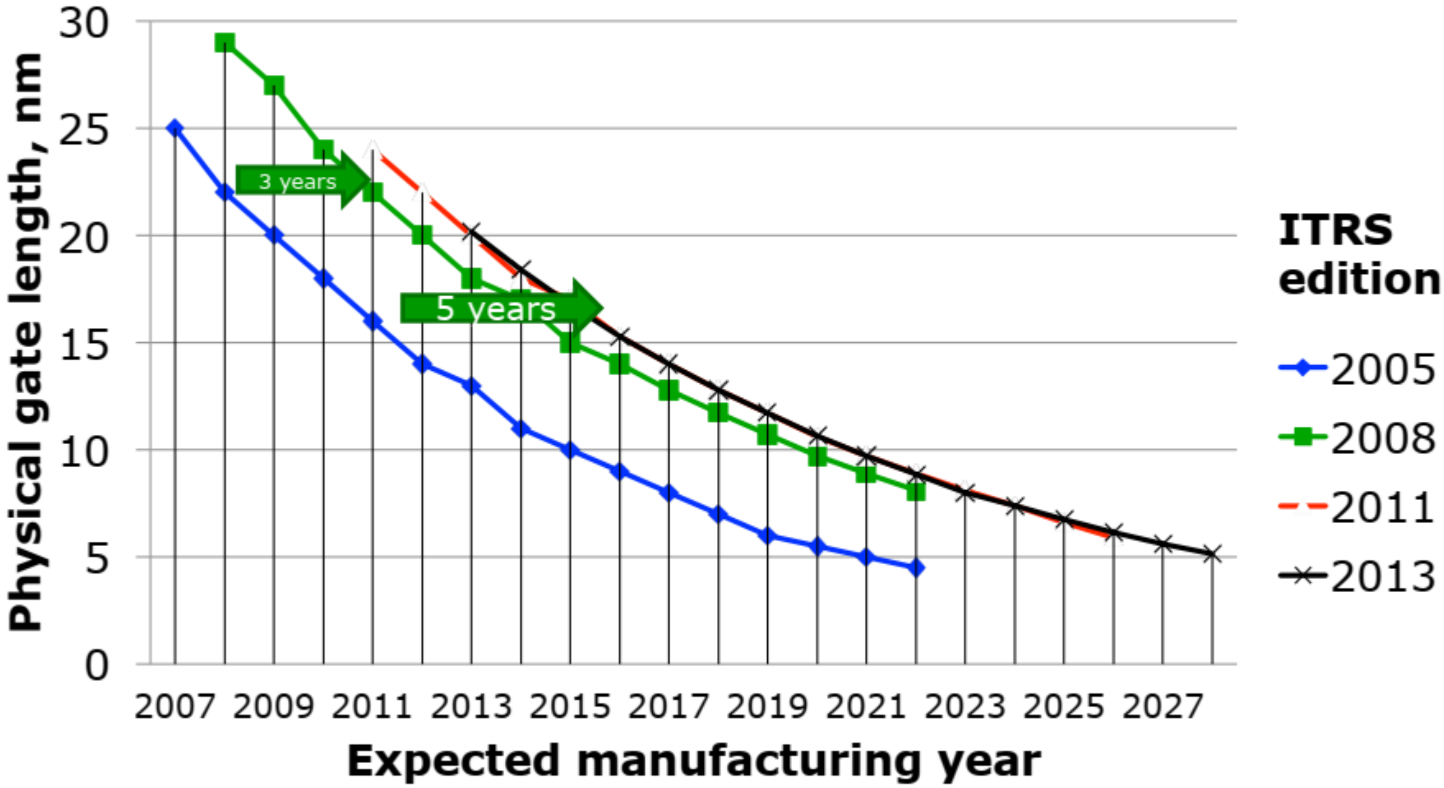
- The semiconductor industry has produced roadmaps
  - Semiconductor Industry Association (SIA): 1977~1997
  - International Technology Roadmap for Semiconductors (ITRS): 1998~2016
  - International Roadmap for Devices and Systems (IRDS): 2017~Present
- IRDS Lithography Projection (2020)

Year of Production	2018	2020	2022	2025	2028	2031	2034
Technology Node (nm)	7	5	3	2.1	1.5	1.0	0.7

  - Moore's Law will continue into foreseeable future
  - IRDS does not project significant increase in CPU chip size
  - Increases in transistors will come from *transistor density*

IRDS isn't Perfect

- ITRS (predecessor of IRDS) has made corrections before



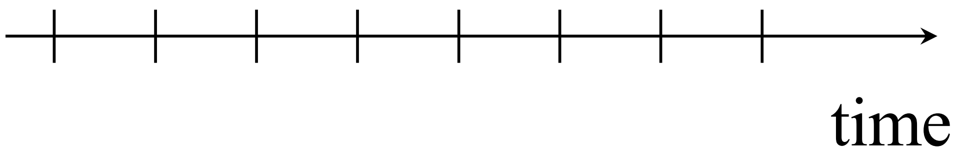
  - After all, you are trying to predict the future
  - But architects rely on the roadmap to design future processors

Moore’s Law and Performance

- Did Moore’s Law result in higher performance CPUs?
- When you decide on a CPU for your desktop, what number(s) do you look at to see how fast it is?
- Best way: try running your favorite apps on it.
- Everyone has different favorite apps so instead publish
  - Results from running a suite of benchmark apps (e.g. SPECCPU)
  - Several important components of performance

Components of Execution Time

- Processor activity happens on clock “ticks” or cycles



  - On each tick, bits flow through logic gates and are latched

- Execution time = $\frac{\text{seconds}}{\text{program}}$

$$\frac{\text{seconds}}{\text{program}} = \frac{\text{cycles}}{\text{program}} \times \frac{\text{seconds}}{\text{cycle}}$$

$$= \frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

How to improve Execution Time

$$\frac{\text{instructions}}{\text{program}} \times \frac{\text{cycles}}{\text{instruction}} \times \frac{\text{seconds}}{\text{cycle}}$$

- Reduce $\frac{\text{seconds}}{\text{cycle}}$:
  - Clock frequency = $\frac{\text{cycles}}{\text{second}}$ = reverse of $\frac{\text{seconds}}{\text{cycle}}$
  - Higher clock frequency (GHz) leads to shorter exec time
- Reduce $\frac{\text{cycles}}{\text{instruction}}$:
  - Also known as CPI (Cycles Per Instruction)
  - IPC (Instructions Per Cycle) = $\frac{\text{instructions}}{\text{cycles}}$ = reverse of $\frac{\text{cycles}}{\text{instructions}}$
  - Higher IPC leads to shorter execution time
- Reduce $\frac{\text{instructions}}{\text{program}}$:
  - Less instructions leads to shorter execution time
  - ISAs that do a lot of work with one instruction shortens time

Moore's Law impacts two layers

- Did Moore's Law result in higher performance CPUs?

- Law impacts both architecture and physical layers

Instruction Set Architecture	Computer Architecture
Processor Organization	
Transistor Implementation	Physical Layer

  - Processor Organization: many more transistors to use in design
  - Transistor Implementation: smaller, more efficient transistors

Moore's Law Impact on Architecture

- So where did architects use all those transistors?

- Well, we will learn this throughout the semester ☺
  - Pipelining ← Improves frequency
  - Parallel execution
  - Branch prediction
  - Speculative execution
  - Memory caching

  Improves IPC

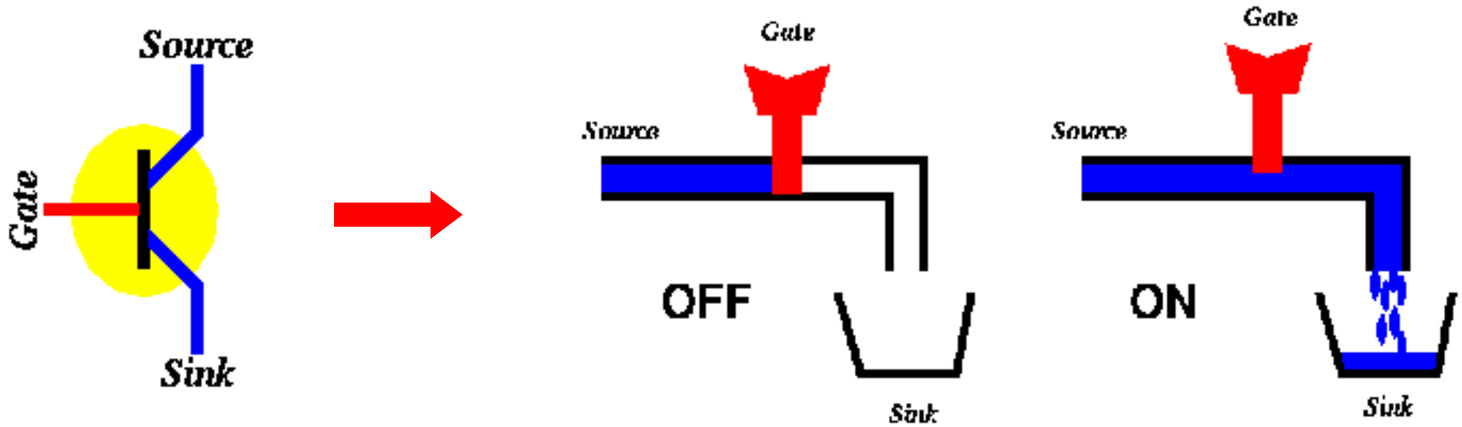
- Let's go on to impact on the physical layer for now

Moore's Law Impact on Physical Layer

- Frequency = transistor speed × number of transistors between clock ticks
  - Transistor speed is decided by the physical layer
  - Number of transistors between ticks is CPU design
- Given a CPU design, faster transistors → faster CPUs
  - Intel's tick-tock model staggers transistor / design improvements
  - Tick: new generation due to new technology node
  - Tock: new generation due to new CPU design
- So did Moore's Law result in faster transistors?
  - In other words, are smaller transistors faster?

Speed of Transistors

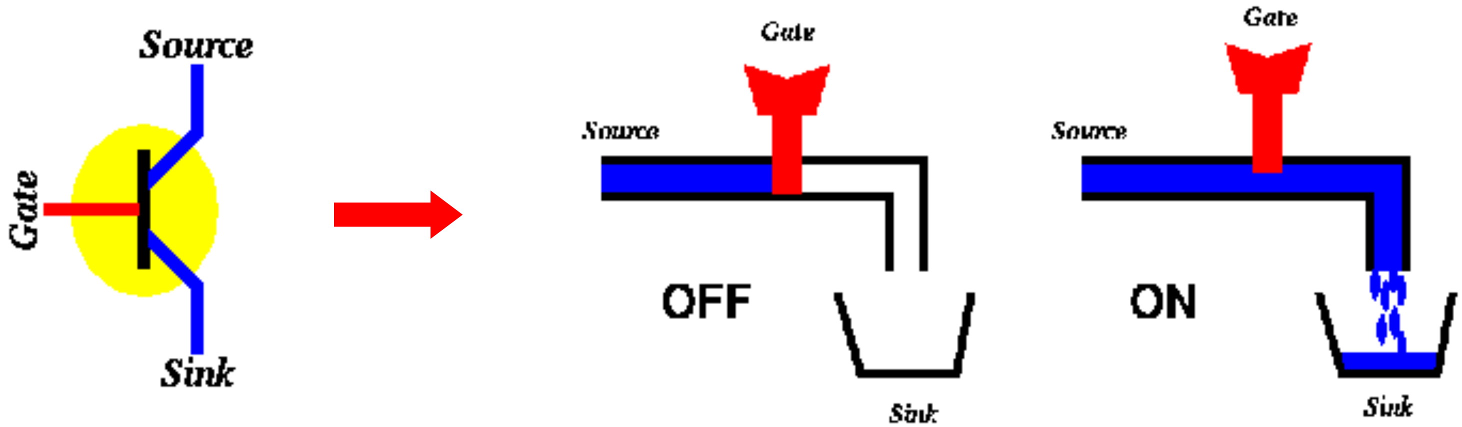
- Transistors are like faucets:



- To make a transistor go fast, do one of the following:
  - Increase water pressure (supply voltage)
  - Change faucet (transistor) design
- Transistor design can be changed in two ways:
  - Increase pipe thickness (reduce channel resistance R)
  - Reduce bucket size (reduce capacitance C)
  - $T_{switch} \propto RC$ (transistor switch delay is proportional to RC)

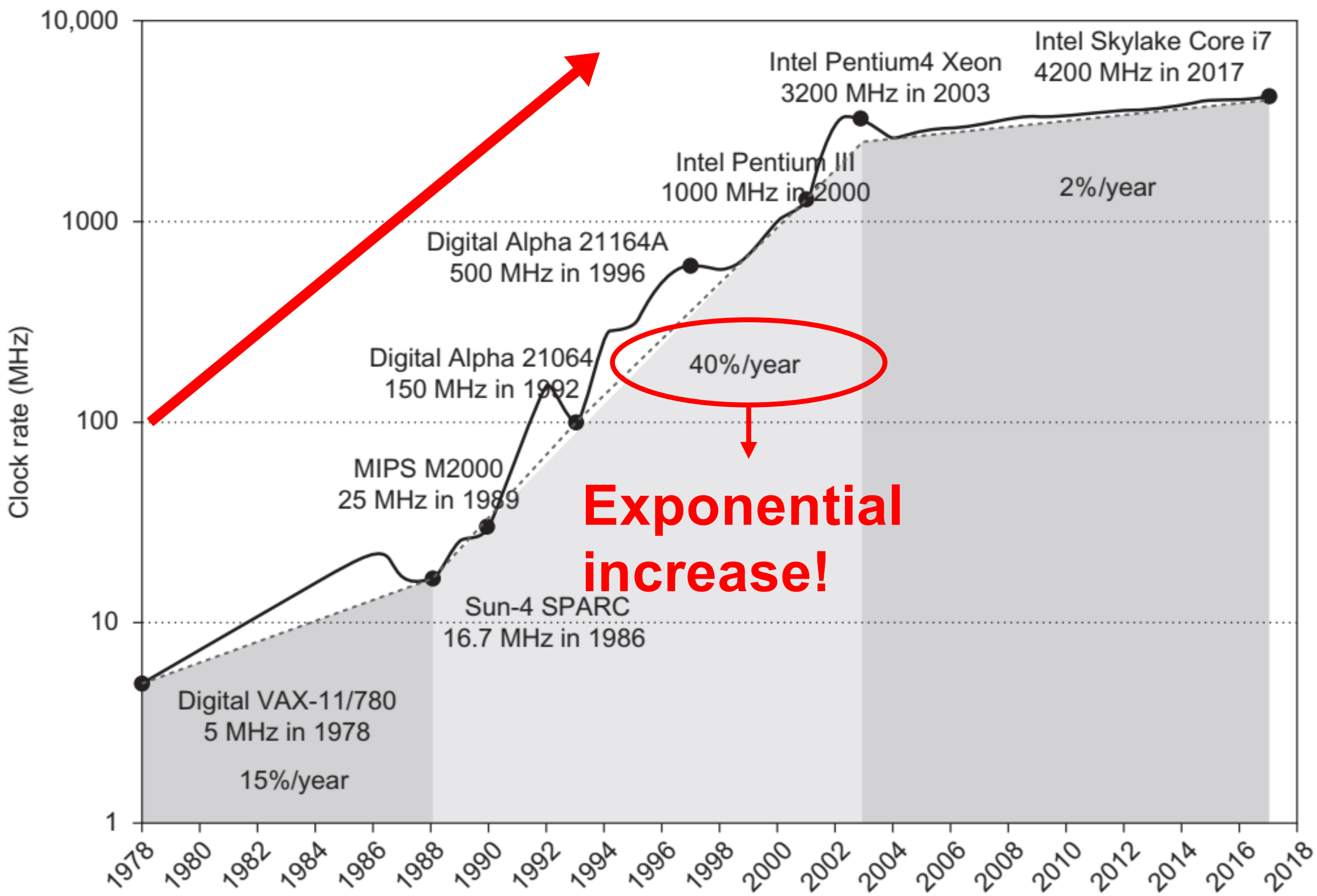
Smaller Transistors are Faster!

- Transistors are like faucets!



- For transistor dimension d (node size)
  - Capacitance $C \propto d$
  - Channel resistance $R \propto$ Channel length $L \propto d$
  - $T_{switch} \propto RC \propto d^2$
- Given the same *supply voltage*, smaller is faster!
- Did Moore's Law enjoy faster and faster frequencies?

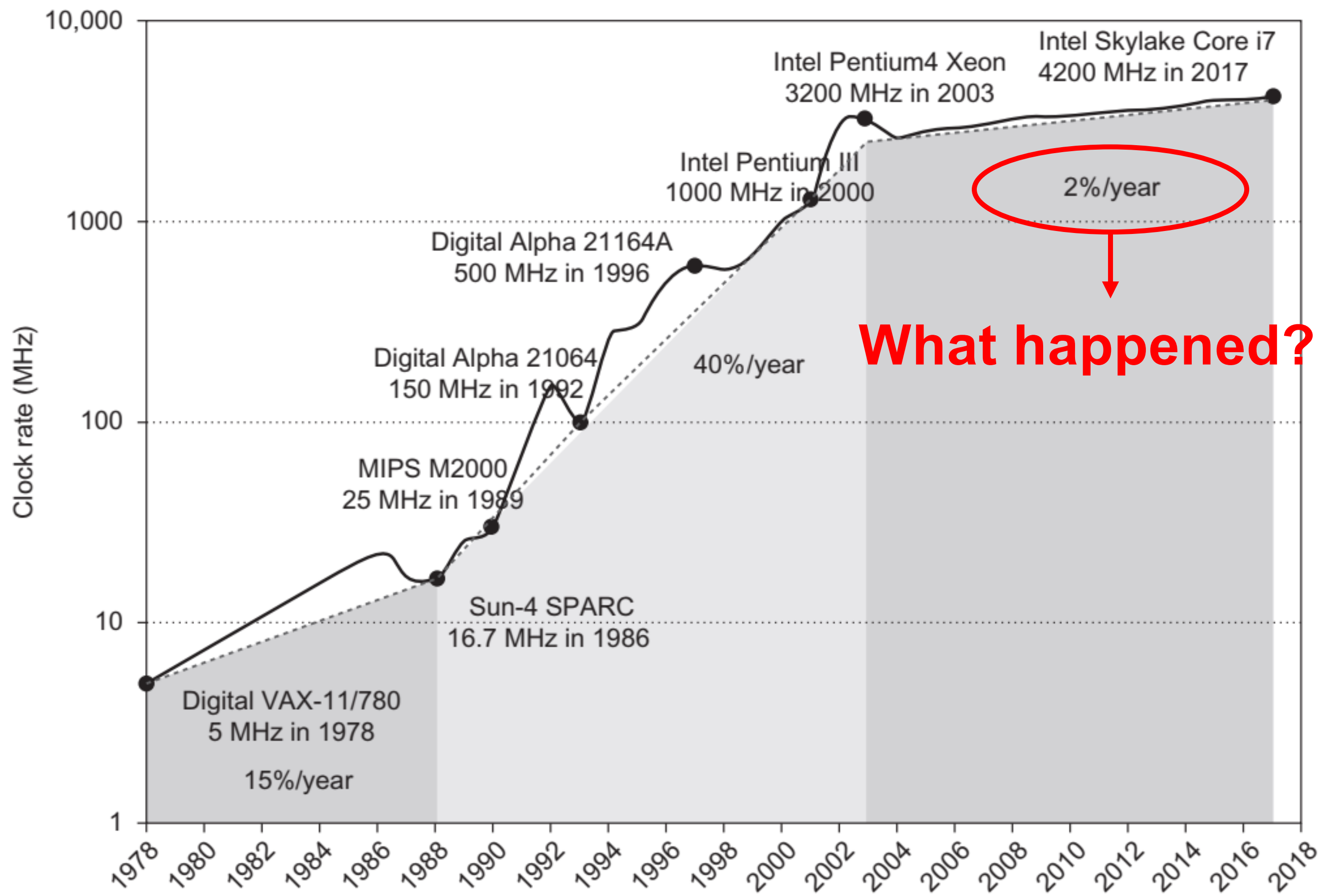
Yes, for a while …



Source: Computer Architecture, A Quantitative Approach (6th ed.) by John Hennessy and David Patterson, 2017

- Due in large part to improvement in transistor speed
  - CPU design (pipelining) contributed but we'll discuss later

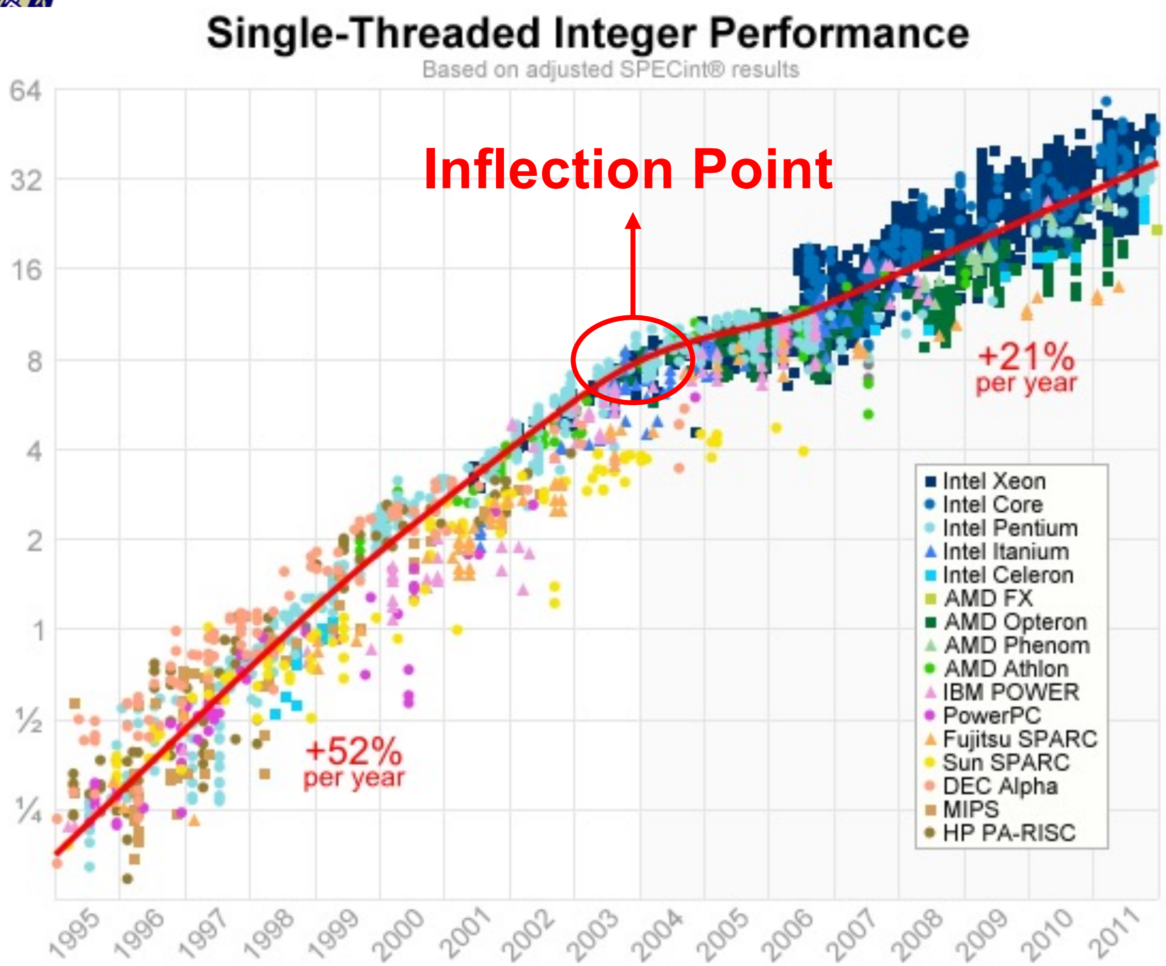
But not so much lately



Source: Computer Architecture, A Quantitative Approach (6th ed.) by John Hennessy and David Patterson, 2017

- Suddenly around 2003, frequency scaling stops. Why?
  - Improvements in transistor speed stopped.
  - CPU design (pipelining) has met its limits.

Dent in CPU Performance

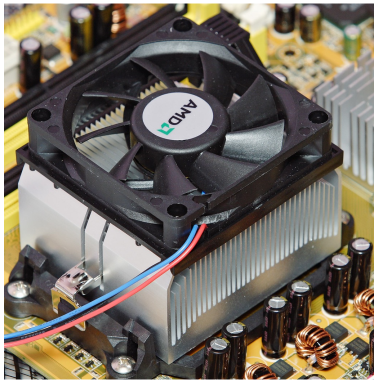


Source: https://preshing.com/20120208/a-look-back-at-single-threaded-cpu-performance/

- This caused a big dent in CPU performance circa 2003
- Improvements henceforth mostly came from IPC

Why did frequency scaling stop? TDP.

- *TDP* (*Thermal Design Power*):
  - Maximum power (heat) that CPU is designed to generate
  - Capped by the amount of heat cooling system can handle
  - Cooling system hasn't improved much over generations
- CPU Power = A * N * CFV^2 must be < TDP
  - A = Activity factor (% of transistors with activity)
  - N = Number of transistors
  - C = Capacitance
  - F = Frequency
  - V = Supply Voltage



- What happens to each factor with Moore's Law?

CPU Power (with Fixed Voltage)

- Change in CPU Power $\propto$ A * N * CFV^2, with fixed V:
  - A = Activity factor
  - N = Number of transistors $\propto 1/d^2$ ⇧ ⇧
  - C = Capacitance (size of bucket) $\propto$ d ⇩
  - F $\propto 1/T_{switch} \propto 1/d^2$ ⇧ ⇧ (if supply voltage is kept constant)
  - V = Supply Voltage (water pressure)
  - → CPU Power $\propto 1/d^3$ ⇧ ⇧ ⇧ !

- Recipe used until 1990's until power started to matter
  - That meant F could not keep increasing while meeting TDP

Q) So how did CPU frequency keep increasing up to 2003?

CPU Power (with Dennard Scaling)

- Goal: to scale frequency while keeping power constant
- By reducing V proportional to d,
  then change in CPU Power $\propto$ A * N * CFV^2 is:
  - A = Activity factor
  - N = Number of transistors $\propto 1/d^2$ ⇧ ⇧
  - C = Capacitance $\propto$ d ⇩
  - F $\propto 1/d$ ⇧ (Can scale by 1/d while keeping power constant!)
  - V = Supply Voltage ($\propto$ d) ⇩ $\rightarrow$ V^2 ⇩ ⇩

- Dennard Scaling: Above recipe for scaling up frequency, while reducing supply voltage to keep power constant

Dennard Scaling and V_{th}

- So, it's that easy? Just reduce V until you meet TDP?

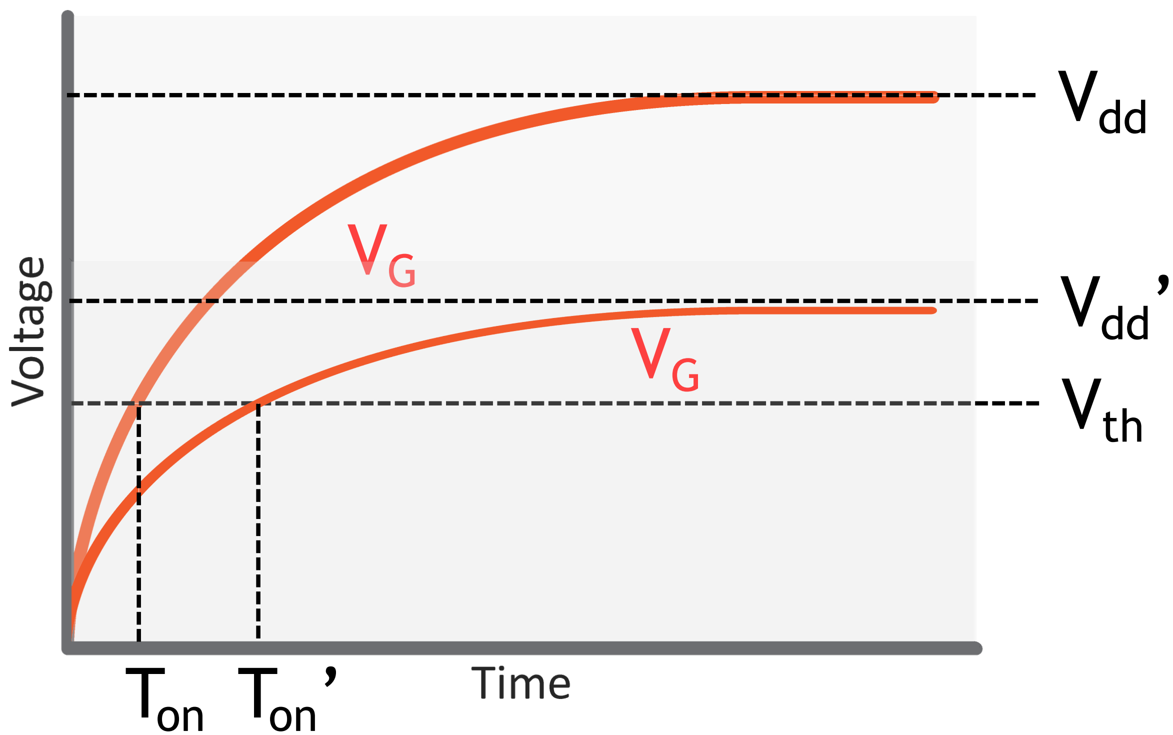
- No, it's not that simple ☹.

- Reducing V_{dd} (supply voltage) affects CPU operation
  - As V_{dd} is reduced, transistor becomes slower (lower pressure)
  - Eventually, CPU stops working altogether

- Transistors need redesigning to work at lower voltage
  - V_{th} is the threshold voltage when switching happens
  - V_{th} needs to be reduced along with V_{dd} to maintain speed

Speed is determined by V_{dd} if V_{th} is fixed

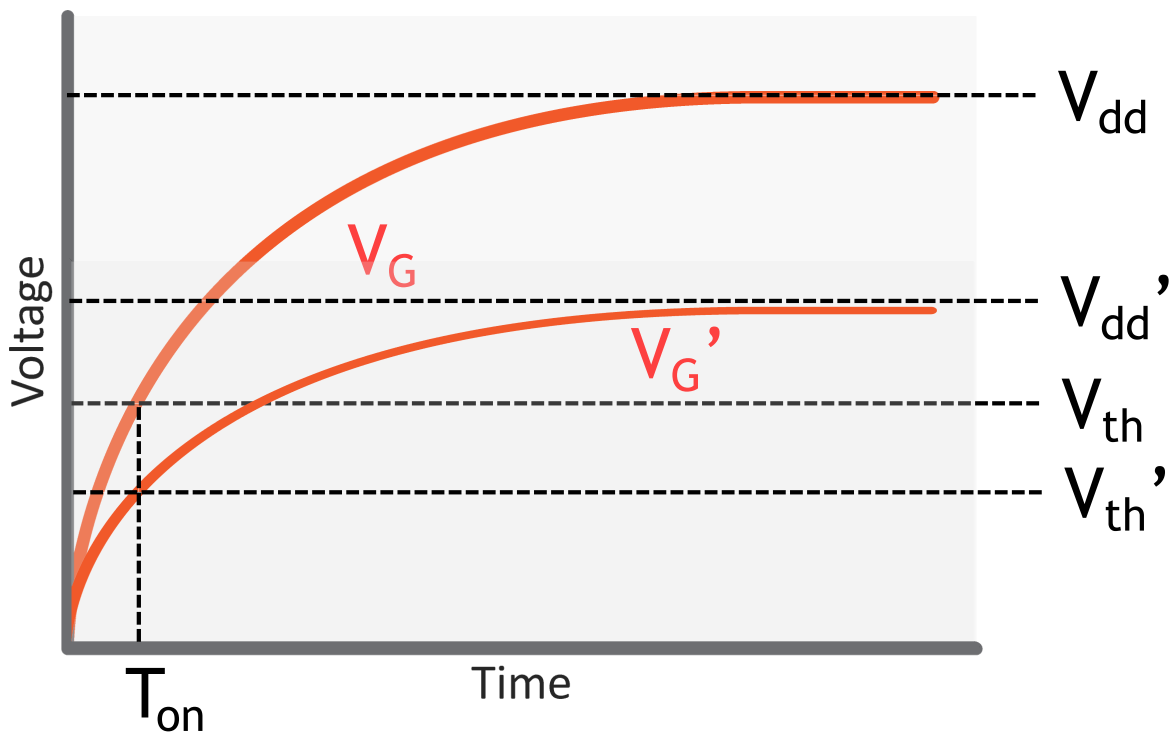
- RC Charging Curve of V_G (Gate Voltage)



- V_{dd} is high → V_G reaches V_{th} slowly at T_{on} (high pressure)
- V_{dd}' is high → V_G reaches V_{th} slowly at T_{on}'(low pressure)

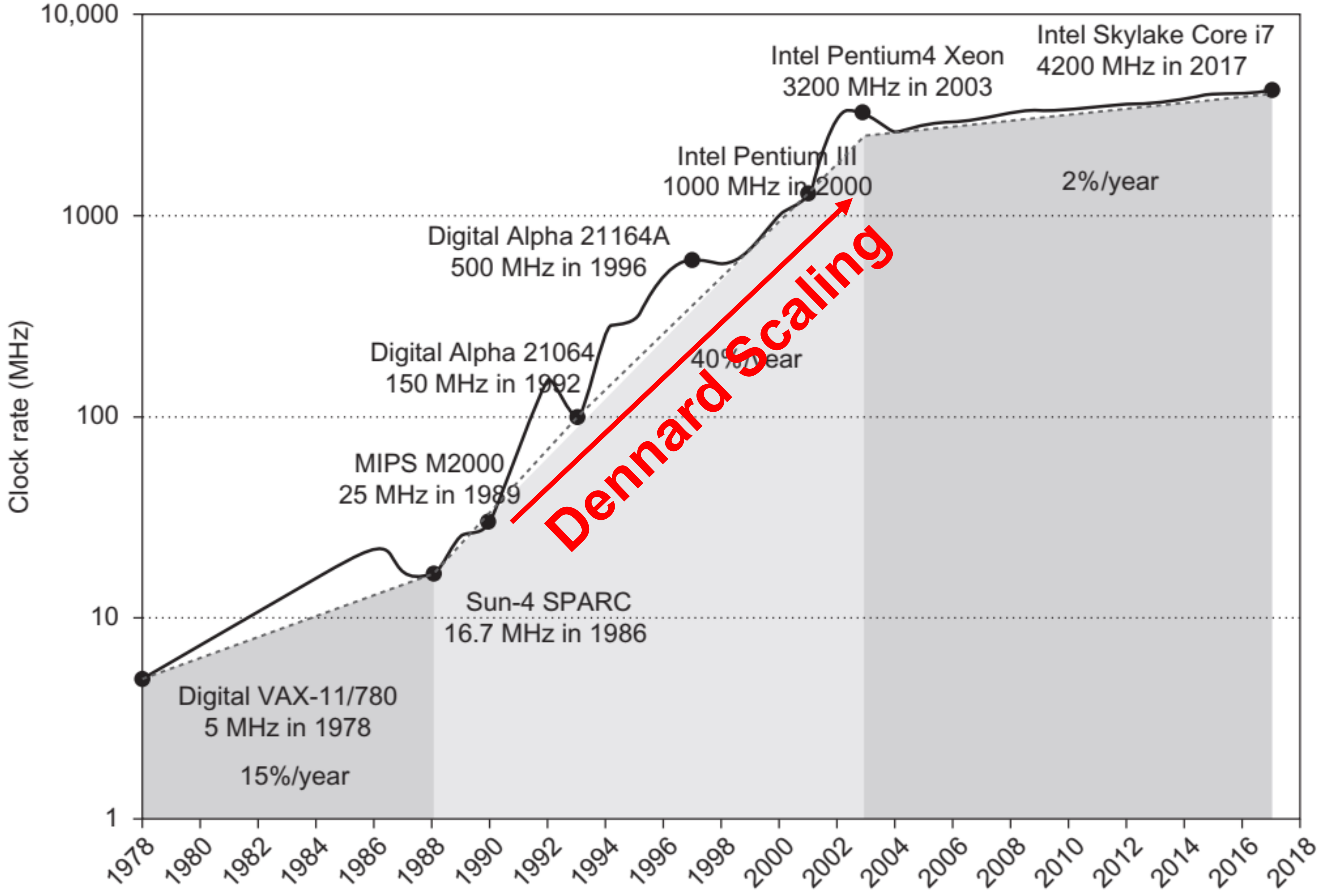
Speed is maintained with lower V_{th}

- RC Charging Curve of V_G



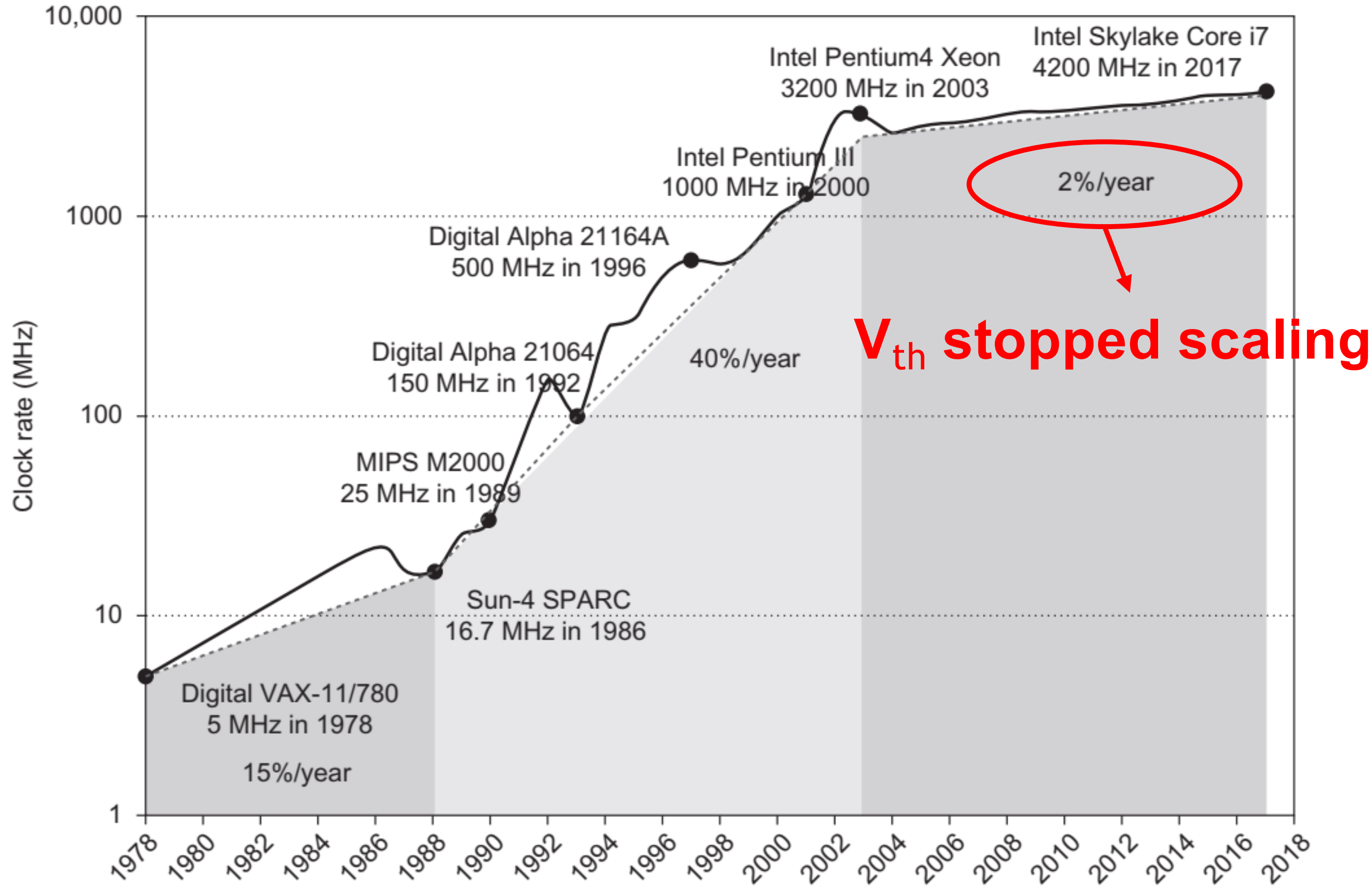
- Speed (T_{on}) is maintained while reducing V_{dd} to V_{dd}', but only if V_{th} is also reduced to V_{th}'

Dennard Scaling paid dividends for a while



- Aided by repeated reductions of V_{dd} and V_{th}

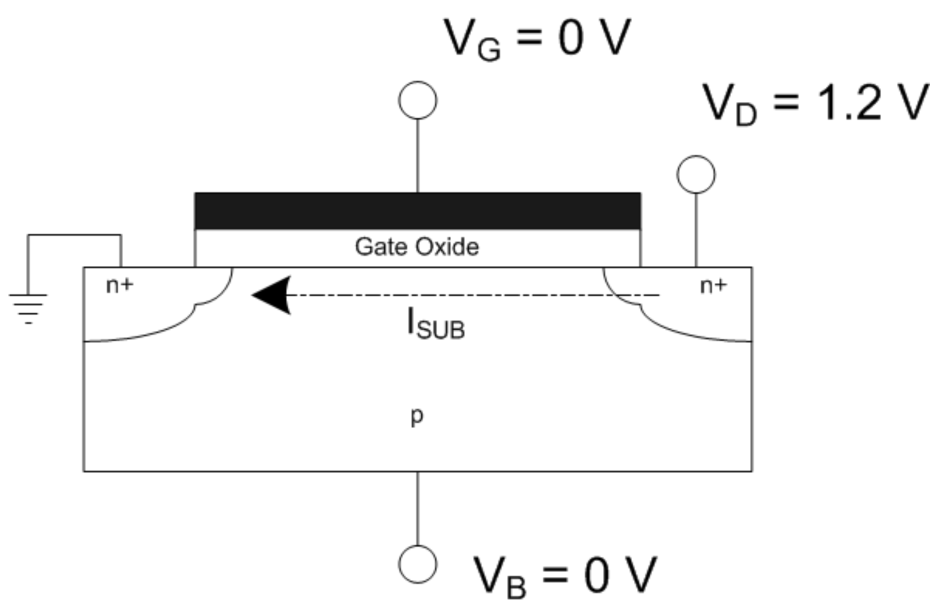
Then Dennard Scaling ended at 2003



- Main reason was that V_{th} could no longer be reduced

Limits to Dropping V_{th}

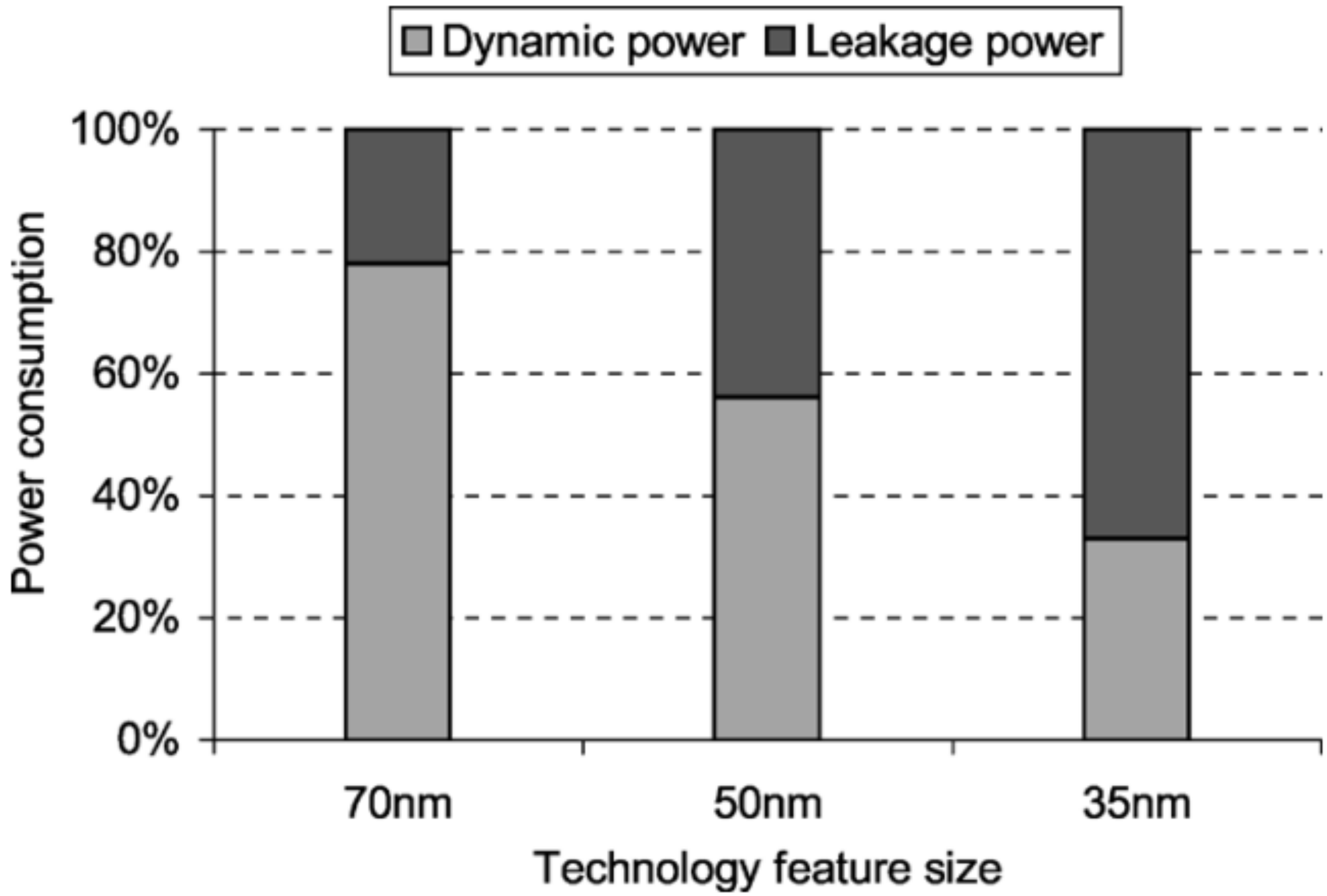
- Subthreshold leakage
  - Transistor leaks current even when gate is off ($V_G = 0$)



  - This leakage current translates to leakage power
  - Leakage worsens when V_{th} is dropped

Leakage Power across Generations

- Leakage power has increased across technology nodes



Source: L. Yan, Jiong Luo and N. K. Jha, "Joint dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded systems," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1030-1041, July 2005

CPU Power must now include leakage

- Previous power calculation was incomplete
  - CPU power is the sum of both dynamic and leakage power
- $Power_{CPU} \propto Power_{dynamic} + Power_{leakage}$
  - $Power_{dynamic} \propto A * N * CFV_{dd}^2$
  - $Power_{leakage} \propto N * V_{dd} * e^{-Vth}$
  - Leakage worsens *exponentially* when V_{th} is dropped
  - Catch-22: when dropping V_{th}, $Power_{dynamic}$ ⇩ but $Power_{leakage}$ ⇧⇧
- V_{th} can't be reduced further, so V_{dd} can't be reduced
- Dennard Scaling relies on reducing V_{dd}, so it's the end

CPU Power (End of Dennard Scaling)

- What happens to frequency without Dennard Scaling?
- $Power_{dynamic}$ ($\propto$ A * N * CFV^2) + $Power_{leakage}$ ($\propto$ N * V * e^{-Vth})
  - A = Activity factor
  - N = Number of transistors ($\propto 1/d^2$) ⇧ ⇧
  - C = Capacitance ($\propto$ d) ⇩
  - V = Supply Voltage ⇔ (Due to fixed V_{th})
  - F = Frequency **???**
- To offset N, you actually have to *decrease* F
- Otherwise, if you want to maintain F, must decrease N
  - That is, you cannot power on all the transistors at any given point
  - **Dark silicon**: situation where chip is only partially powered

Free Ride is Over

- "Free" speed improvements from transistors is over
- Now it's up to architects to improve performance
  - Moore's Law is still alive (although slowing down)
  - Architects are flooded with extra transistors each generation
  - But it's hard to even keep them powered without reducing F!
- Now is a good time to discuss technology constraints
  - Since we already mentioned a big one: TDP