Tema 02: Costo, Performance y Consumo de Energía

Arquitectura de Computadoras

Ing. Nicolás Majorel Padilla (npadilla@herrera.unt.edu.ar)

http://microprocesadores.unt.edu.ar/arqcom/

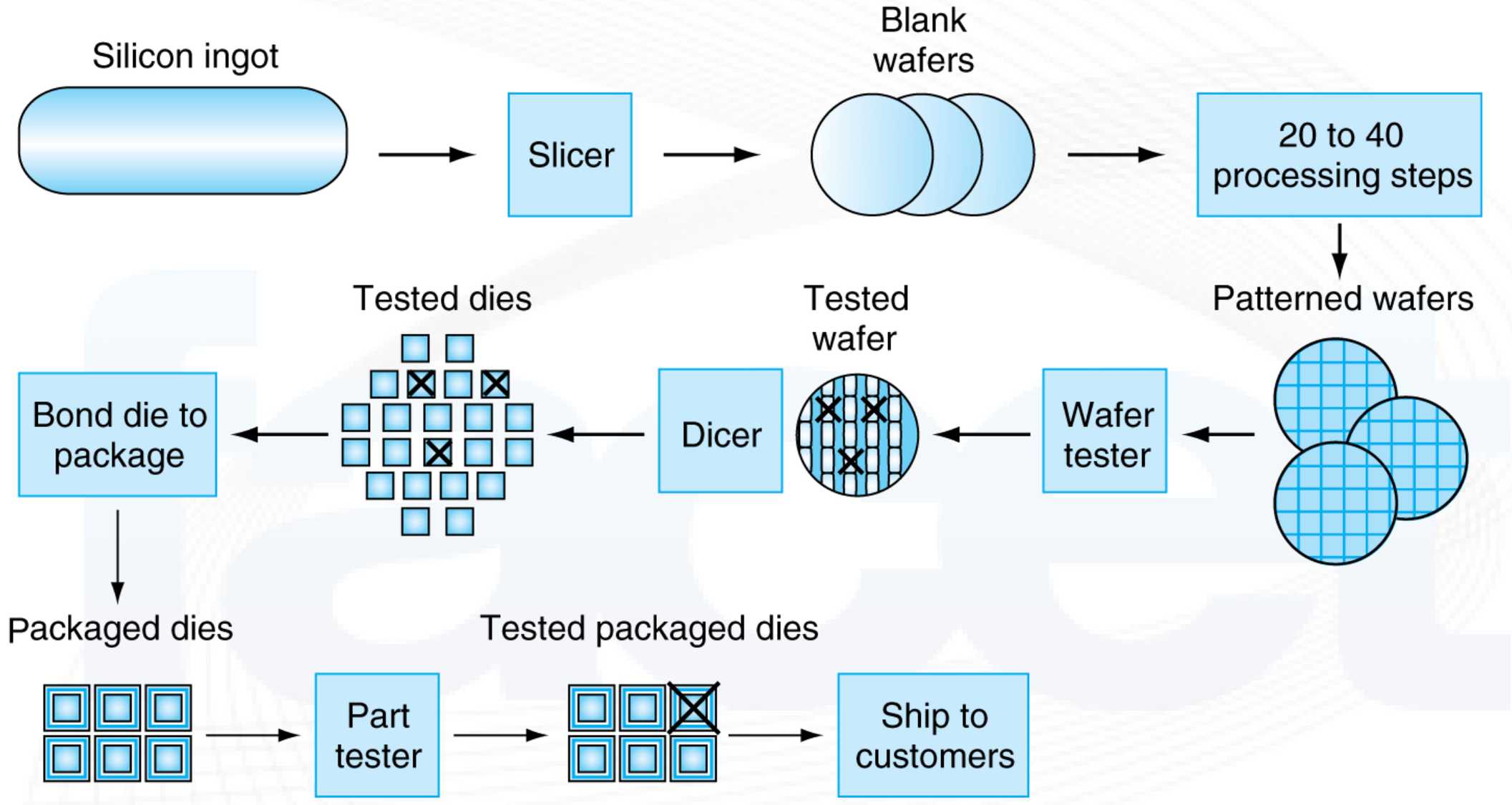
Temas que veremos

- Las “tres P” de Arquitectura de Computadoras
  - ***P**rice:* Costo de Circuitos integrados.
  - ***P**erformance:* Definición de Medidas de Performance.
  - ***P**ower:* Potencia y Energía.
- Tendencias

Lectura recomendada

- Computer Organization and Design, RISC-V Edition (2da ed.)
  - Sección 1.5: *Technologies for Building Processors and Memory*
  - Sección 1.6: *Performance*
  - Sección 1.7: *The Power Wall*
  - Sección 1.9: *Benchmarking the Intel Core i7*
  - Sección 1.10: *Going Faster: Matrix Multiply in Python*
  - Sección 1.11: *Fallacies and Pitfalls*

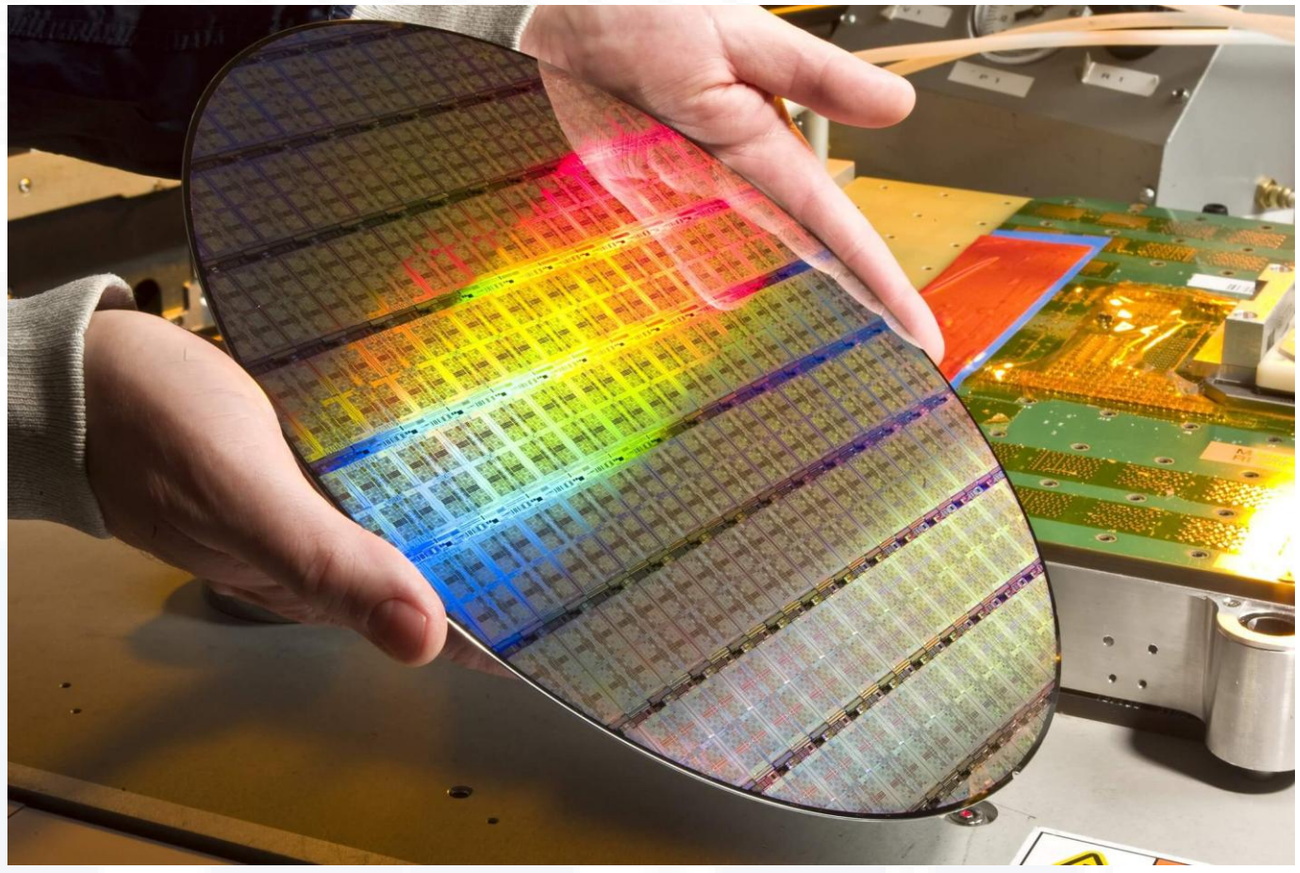
Fabricación de Circuitos Integrados



https://www.youtube.com/watch?v=d9SWNLZvA8g

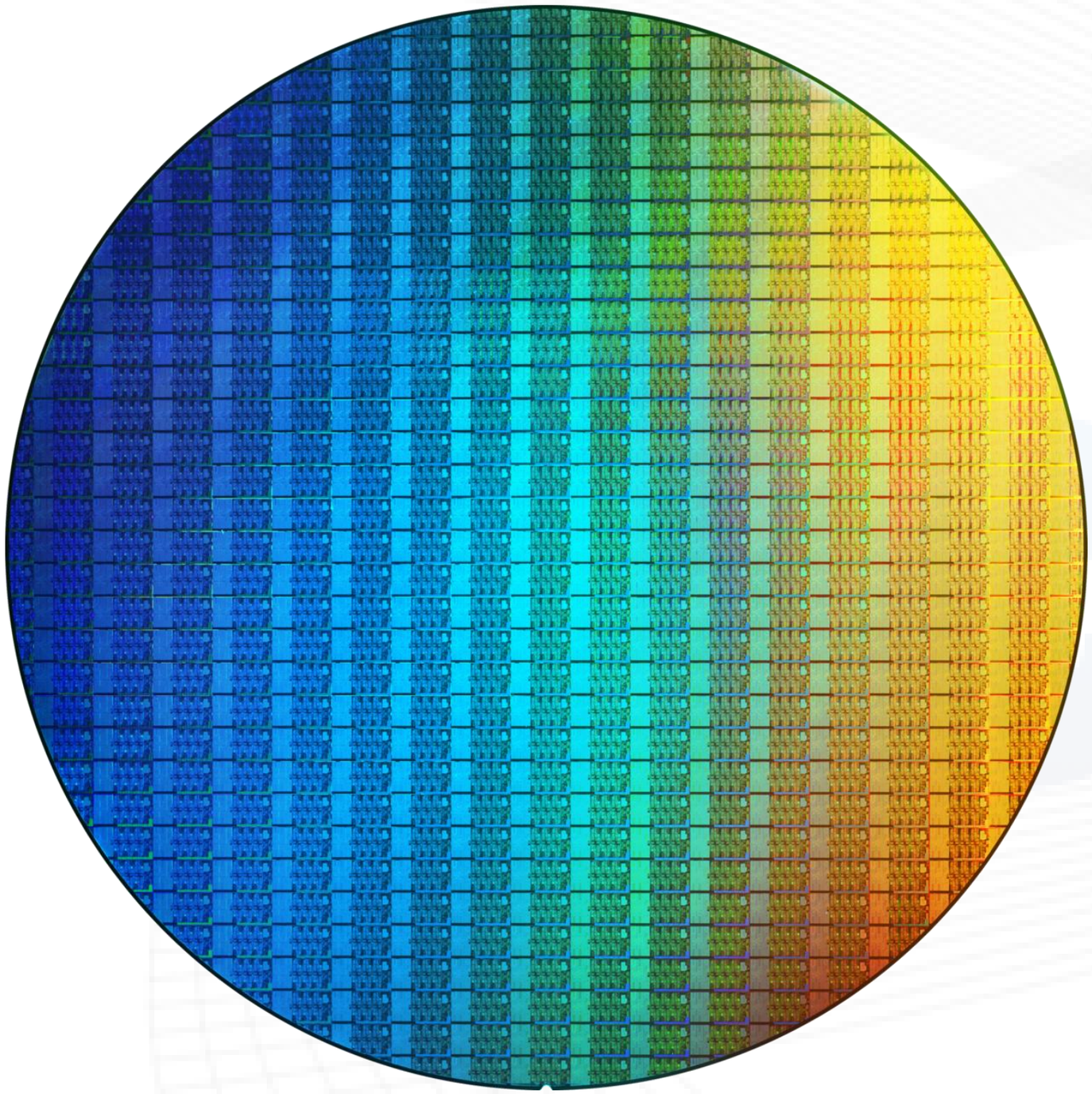
Detalles de la oblea

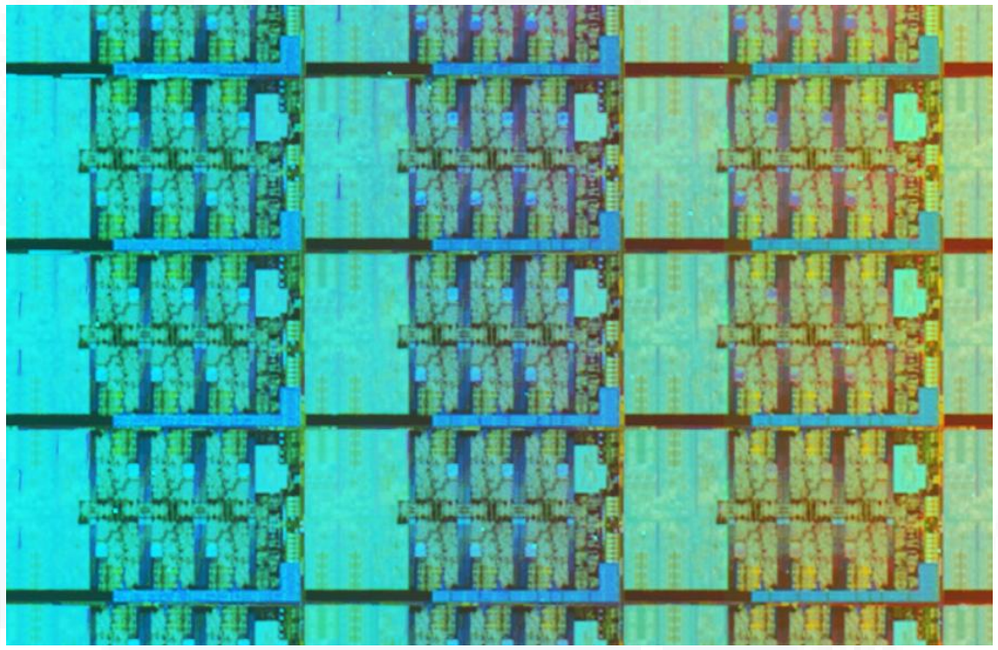
- Diámetro de 300 mm.
  - Estándar desde 2000 aprox.



- Intel Core i7 Sandy Bridge (2nd g., 2011)
  - Tecnología de 32 nm.
  - Cada chip mide 20.7 x 10.5 mm. (~218 mm^2)
  - Salen ~280 chips por oblea.
- Intel Core i7 Coffe Lake (8th gen, 2017)
  - Tecnología de 14 nm.
  - Cada chip mide 9.2 x 16.28 mm. (~151 mm^2)
  - Salen ~400 chips por oblea.

Imagen de una oblea





- Es un i7-8700 de 6 núcleos

Tamaños de algunos chips

Microprocesador	Área [mm²]
ARM Cortex-M4	0.04
ARM Cortex-A55	0.30
ARM Cortex A76	0.90
SiFive U84	2.63
Samsung M4	127
Intel Core i7 10th gen	125
AMD Ryzen (un CPU core, Zen 4)	50-80
AMD Ryzen (un GPU core, Zen 4)	100
Apple M1	119
NVIDIA GPU (H100)	814

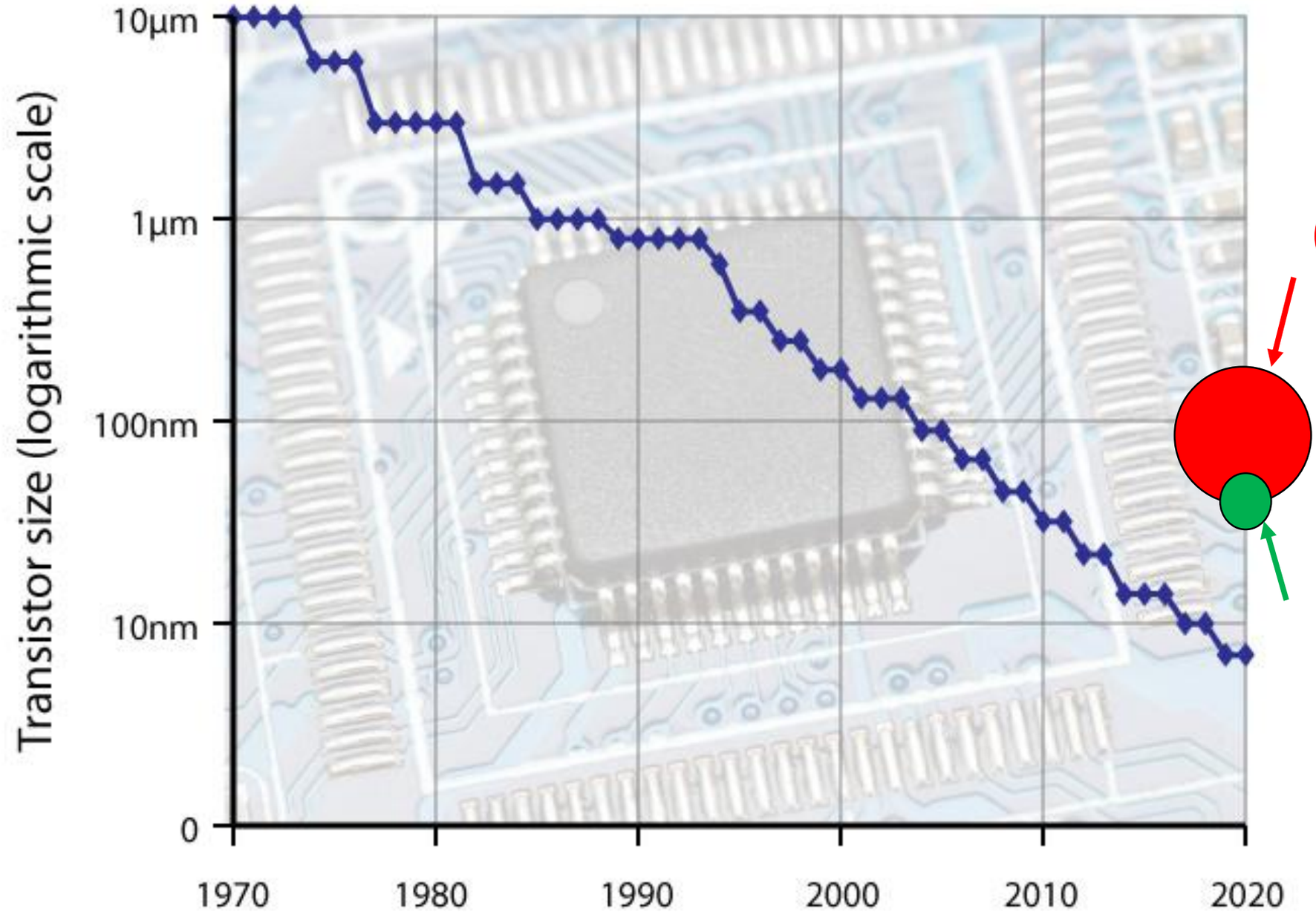
- Algunos son sólo procesador, otros incluyen muchas más cosas (Temas 14 y 16).
- Además no se especifica el proceso de fabricación.

Costo de los circuitos integrados

- Costo del chip = Costo_oblea / #chips_sinDefectos
- #chips_sinDefectos = chips_oblea * **yield**
  - Factor de Producción (**yield**) es una ecuación empírica más complicada.
- chips_oblea = Area_Oblea / Area_chip
  - Y hay que descontar los bordes del círculo.
- Al reducir el área del chip…
  - Aumenta la cantidad de chips por oblea.
  - Aumenta el yield. ***¿Por qué?***
  - Disminuye el costo del chip.
- **El costo del chip es proporcional al cubo de su área.**

Evolución del *feature size*

- **El área del chip se reduce mejorando el proceso de fabricación.**



CoVid19
(50 – 200 nm)

Dengue
(40 – 60 nm)

Aclaración sobre el *feature size*

- Tradicionalmente, este número representaba el ancho mínimo de un transistor (entre S y D).
- Ahora es un indicador de la capacidad de una fábrica (*process node*).
  - Diferentes procesos de diferentes fábricas pueden resultar en transistores de igual ancho.
  - Procesos de nombre similar, de diferentes fábricas, pueden resultar en transistores de distinto ancho.
- Cada empresa de microprocesadores trabaja con su propia fábrica (*foundry*), que posee su propio proceso de fabricación.
  - Intel 2 nm, AMD (TSMC) 3 nm, Samsung 2 nm, etc.
- 0,7x es la escala entre dos *feature size* consecutivos.
  - 90, 65, 45, 32, 22, 15, 10, 7, 5, 3, 2, …
  - **Permite que el área se reduzca aproximadamente a la mitad.**
- Prestar atención a los detalles, y no al marketing.

Evolución del Costo por oblea

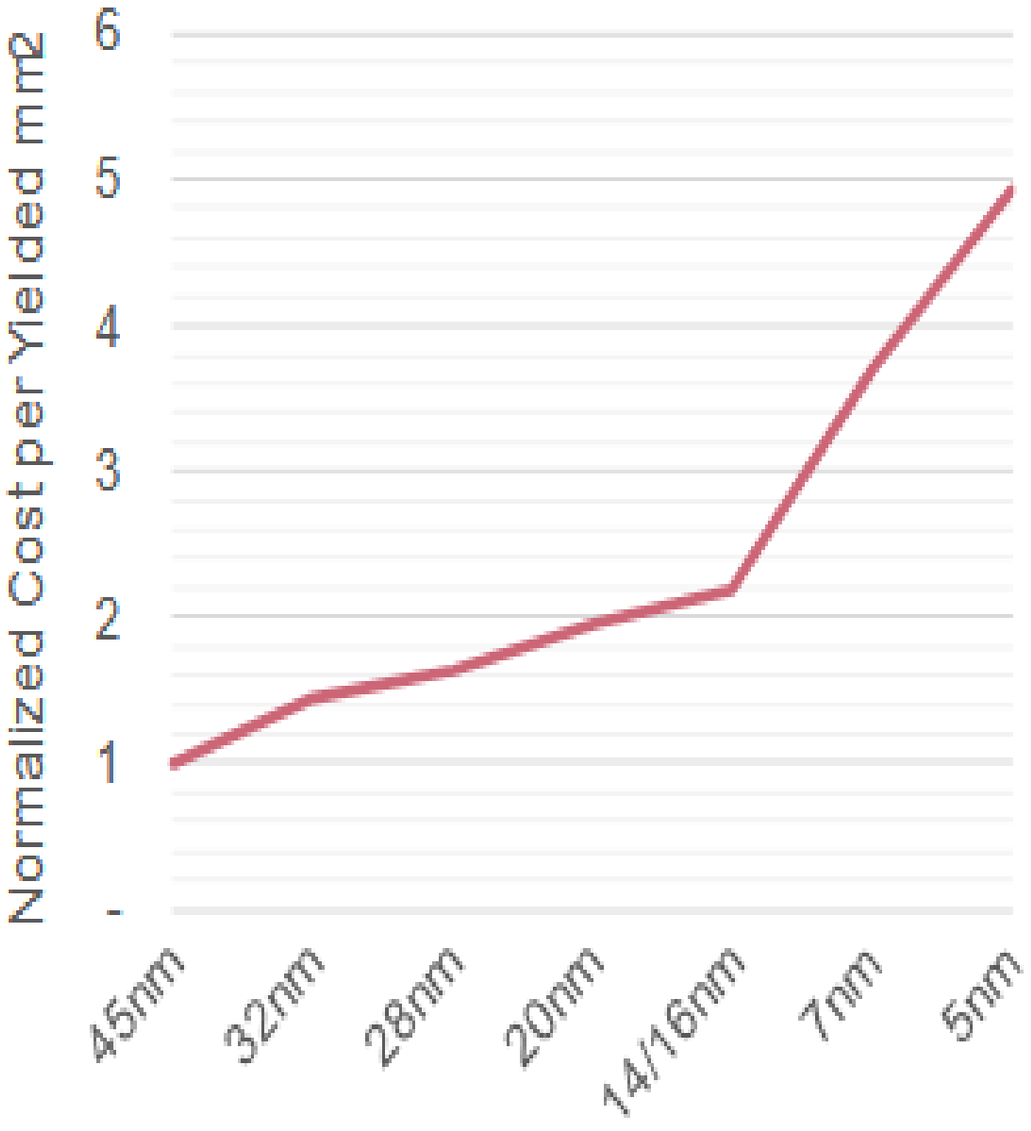
- En las ecuaciones anteriores, el costo por oblea lo considerábamos constante.
- Pero en realidad depende del proceso de fabricación.
- Por ejemplo, considerando valores aproximados de la empresa TSMC:

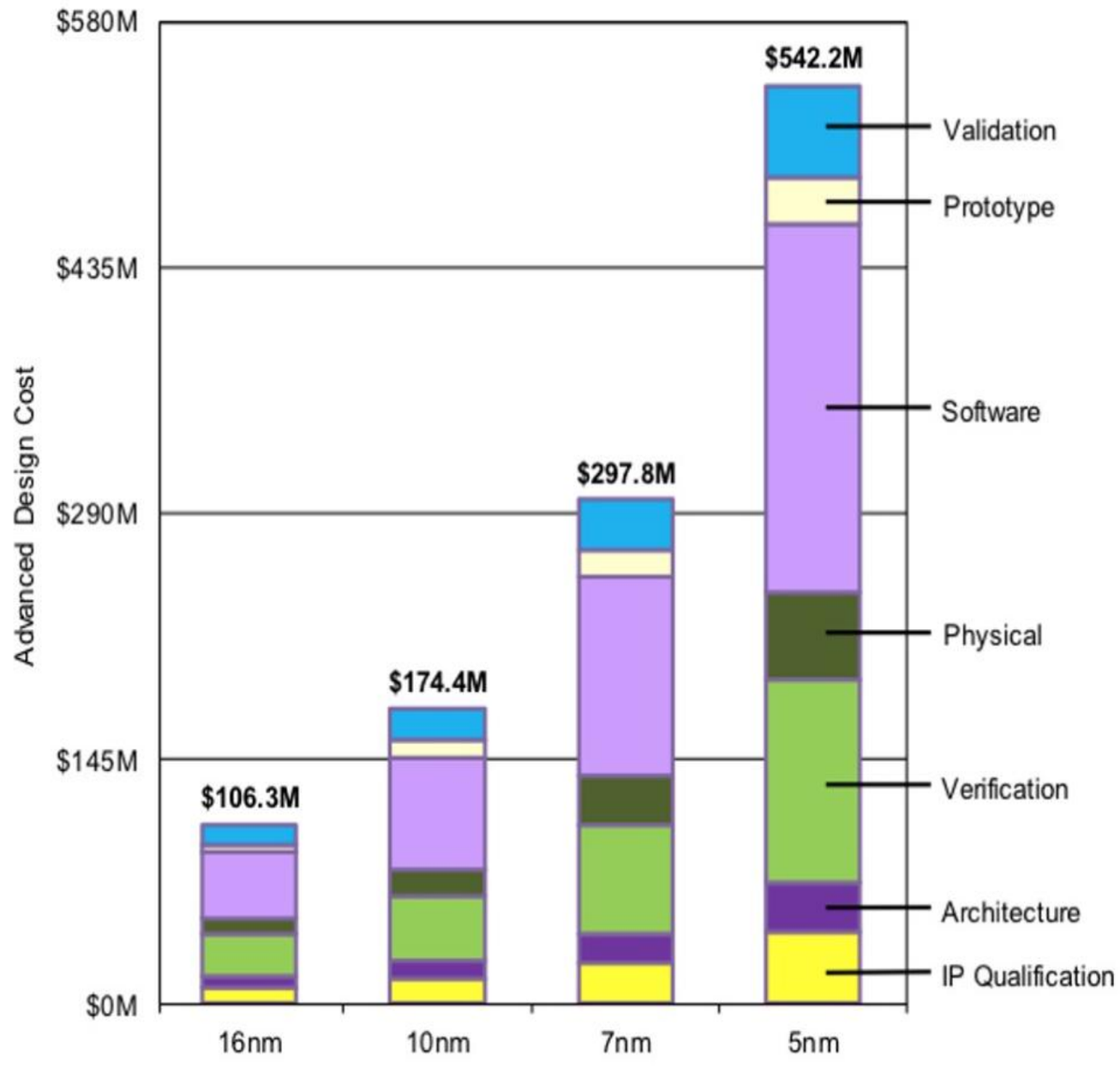
Año	Feature size [nm]	Costo [USD]
2004	90	2.000
2014	28	3.000
2016	10	6.000
2018	7	10.000
2020	5	16.000
2022	3	20.000
2024	2	30.000
2026	1.6	45.000

- Nótese que en los últimos años los incrementos son de aproximadamente el 50%.

Costo de los circuitos integrados

- Además, el costo de la oblea no es el único costo:



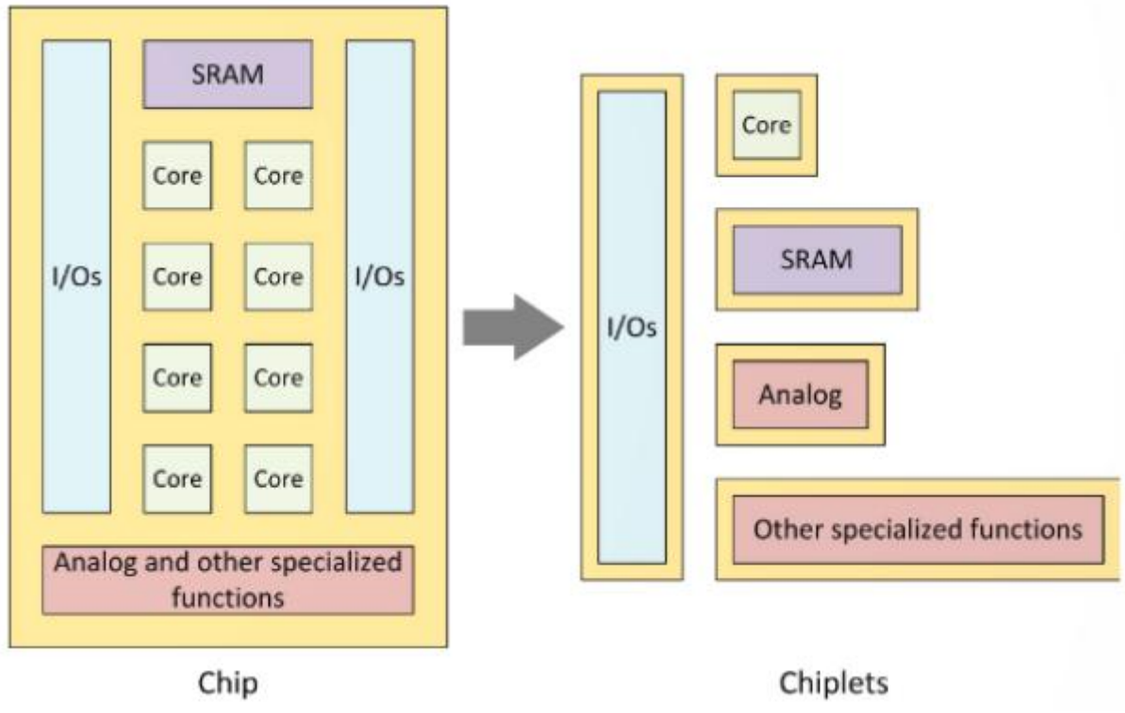


Limitaciones del proceso

- Todo lo expresado anteriormente es un proceso de fabricación **monolítico**.
  - Cada chip es un único pedazo de silicio.
  - Y cada *package* contiene un único chip.
- Posee dos limitaciones inherentes:
  - El área de la oblea es fija y estándar, sin intenciones de cambiarlo a corto plazo.
  - Todos los procesos de fabricación estándar vistos, tienen un límite de tamaño máximo de chip que pueden fabricar.
    - Aproximadamente 850 mm^2.

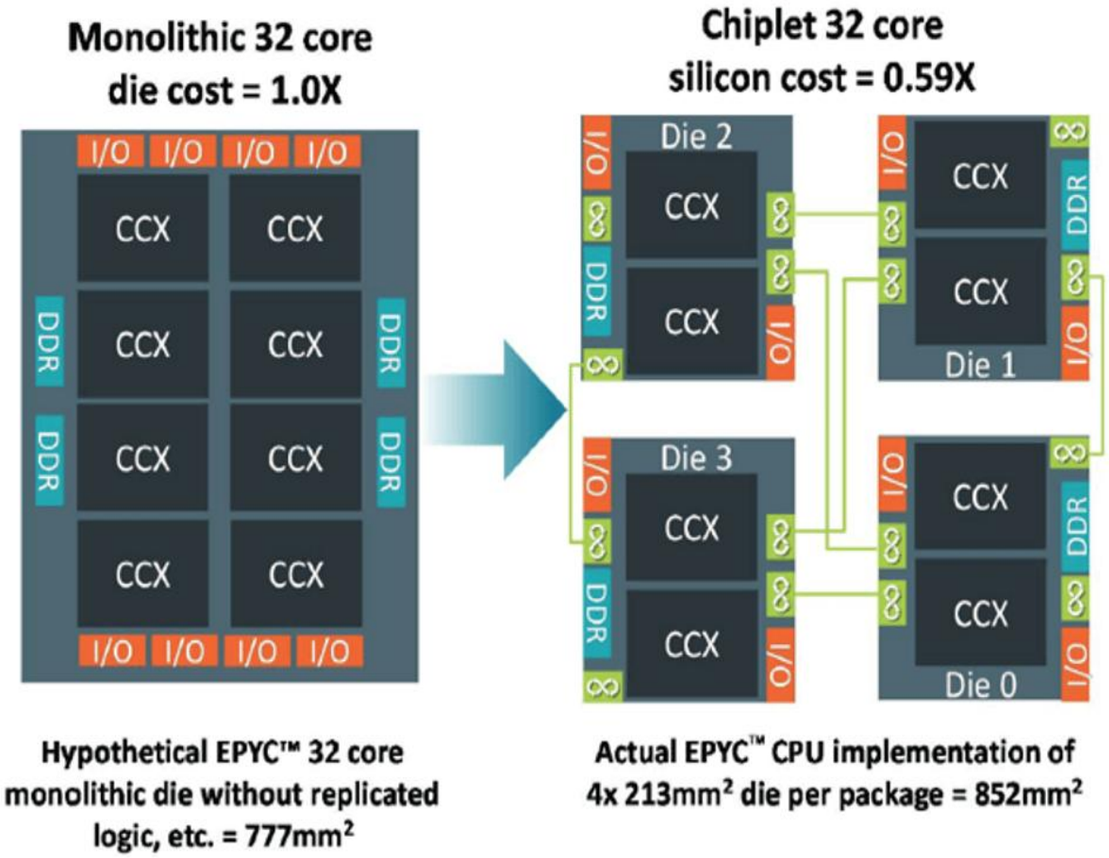
Una solución – *Chiplets*

- ***Chiplets*:** varios chips conectados entre sí, dentro de un mismo empaquetado, al estilo Lego.
  - También conocidos como *Multichip Module* (MCM).
  - Normalmente la conexión es a través de algún sustrato base.
  - Los distintos *chiplets* pueden tener distintas funciones.
- Se pueden combinar *chiplets* fabricados con diferentes procesos.
  - 5 nm para algunas tareas y 28 nm para otras.
  - *¿Por qué sería bueno esto?*
  - *¿Cuáles tareas (o partes) ubicarían en cada chip?*



Chiplets

- Ventajas:
  - **Mejoran el yield**, y la cantidad de chips por oblea, lo cual hace que **baje el costo**.
  - Permiten reutilización.
  - **Mayor escalabilidad** (Tema 08).
  - Mantienen la performance.
  - **Mejoran el consumo de energía** (Tema 15).
  - **Modularidad** (Tema 16).



- Desventajas:
  - Las interconexiones son propietarias (Tema 14).
  - **Aumentan el área total**.
  - Añaden más complejidad.
  - Aún no poseen un ecosistema estándar desarrollado.

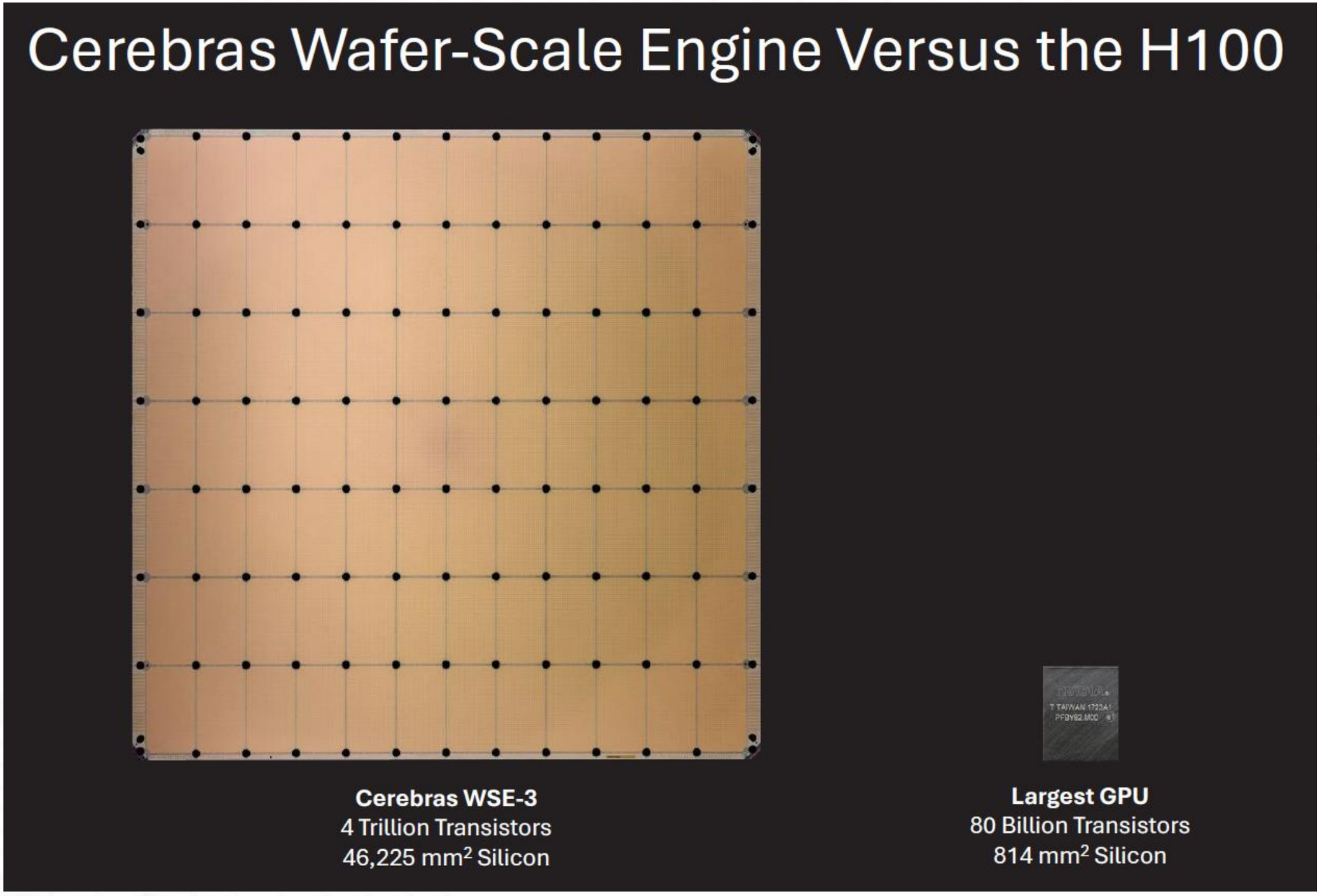
Chiplets

- AMD fueron los primeros “grandes” en hacerlo comercialmente, con la 3ra generación de los EPYC (2017) y los Ryzen (2019).
  - Con excelentes resultados desde entonces.
- Intel adoptó este enfoque a partir de su 14ta generación, denominada *Meteor Lake* (2023).
- Nvidia también lo usa, por ejemplo con su GPU B200 (*Blackwell*) de 2024.
- La misma idea es usada en las memorias HBM (*High Bandwith Memory*), que reemplazan a las DDR.

Otro enfoque… muy diferente

- Una empresa denominada Cerebras hizo en 2019 el chip más grande del mundo, **y de la historia**.
- *Waferscale integration*
  - Usa toda la oblea, sin cortes ni empaquetamientos.
  - Fabricado con un proceso de 16 nm.
  - 50 veces más grande que el chip más grande de ese entonces.
  - 400.000 núcleos, 18 GiB de memoria SRAM integrada.
- Idea no es nueva (tiene más de 40 años), pero nunca había funcionado hasta ahora.
  - *¿Recuerdan lo de las fuerzas en constante movimiento?*

Comparativa *WaferScale*



Otro enfoque: *Waferscale*

- Una única función: es específico para tareas de IA.
  - Brinda una enorme mejora de performance (~300x).
  - Industria farmacéutica, biotecnología, militar, investigaciones de cáncer, procesamiento de datos sísmicos, etc.
- Desafíos
  - Fabricación, *yield.*
    - Resuelto con **redundancia**.
  - Disipación de energía (entre 15-50 kW).
    - Los transistores trabajan con una alimentación de 0,8 V. ¡Pero son tantos que necesitan 20.000 A!
  - Precio: un sistema con 32 de estos chips cuesta u$s 100 M.
  - Programación: requiere reescribir los programas y todo el software de base.
- Volveremos sobre esto en el Tema 16. ☺

Otro enfoque: *Waferscale*

- Luego de su primera versión de 2019, Cerebras hizo:
  - Una 2da versión (2021) fue fabricada en 7 nm, tiene 850k núcleos y 40 GiB de memoria SRAM integrada.
  - Una 3ra versión (2024) fue fabricada en 5 nm, tiene 900k núcleos y 44 GiB de SRAM, con 2x performance que el anterior, con el mismo consumo de energía.
- Tesla quiso fabricar un un procesador similar (*Dojo Training Tile*) en may-24, y lo dio de baja en Nov-25.
  - Se podría concluir que no es tan sencillo.
- El fabricante de estos chips (TSMC) anunció que en 2027 tendrá una mejora en el proceso de fabricación de estos chips que podría resultar en una mejora adicional de 40x.
- En ene-26, Cerebras firmó un convenio de 10.000 M USD con Open AI.
- Conclusión: en 2026 son una alternativa real y se espera que su uso vaya en aumento.

Costos y tiempos de diseño

- El tiempo medio de diseño de un microprocesador, desde escritorio hasta que sale al mercado es entre 3-5 años.
  - Hay un desfasaje entre diseño-fabricación-venta.
  - Empezar un diseño hoy, pensando en el proceso de fabricación que habrá dentro de dos años.
  - Esa futurología funciona gracias a…

Ley de Moore

- En 1965, Gordon Moore de Intel predijo que el **número de transistores** que pueden integrarse en un chip se duplicaría cada 18-24 meses.

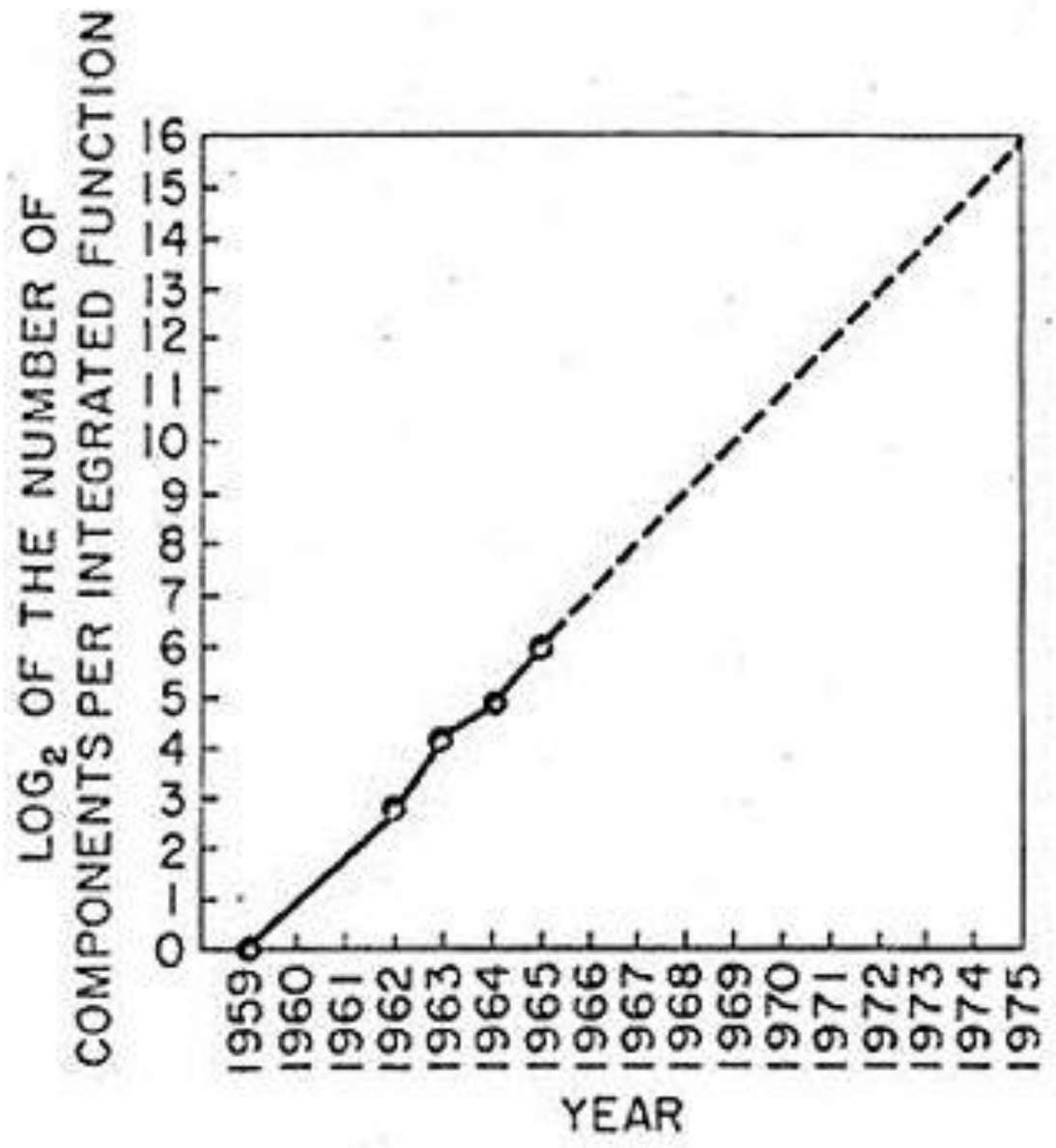
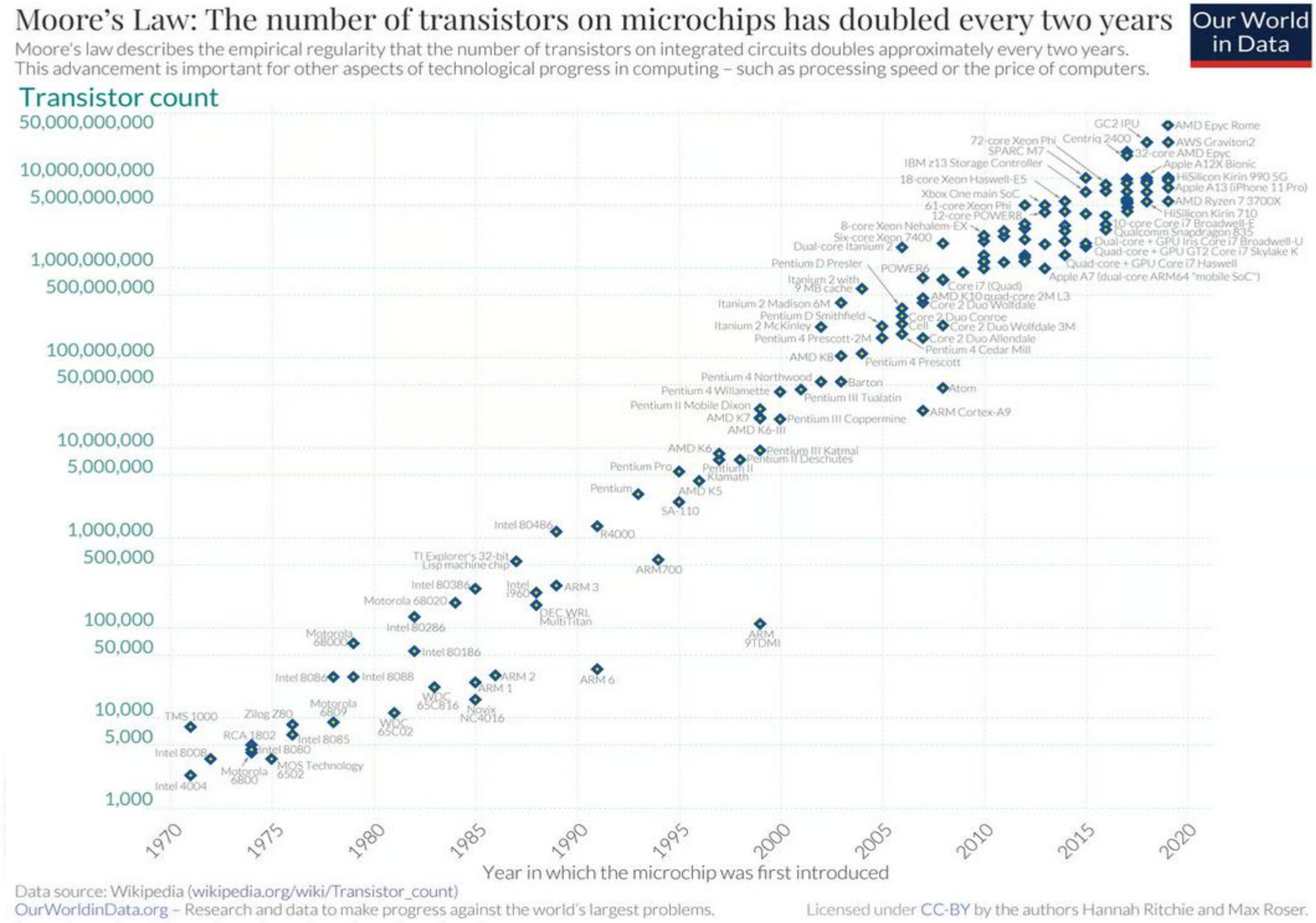


Fig. 2 Number of components per integrated function for minimum cost per component extrapolated vs time.

Ley de Moore


Moore's Law: The number of transistors on microchips has doubled every two years
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.
Our World in Data
Transistor count
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
GC2 IPU
AMD Epyc Rome
72-core Xeon Phi
Centriq 2400
AWS Graviton2
SPARC M7
32-core AMD Epyc
IBM z13 Storage Controller
Apple A12X Bionic
18-core Xeon Haswell-E5
HiSilicon Kirin 990 5G
Xbox One main SoC
Apple A13 (iPhone 11 Pro)
61-core Xeon Phi
AMD Ryzen 7 3700X
12-core POWER8
HiSilicon Kirin 710
8-core Xeon Nehalem-EX
10-core Core i7 Broadwell-E
Six-core Xeon 7400
Qualcomm Snapdragon 835
Dual-core Itanium 2
Dual-core + GPU Iris Core i7 Broadwell-U
Quad-core + GPU GT2 Core i7 Skylake K
Pentium D Presler
POWER6
Quad-core + GPU Core i7 Haswell
Itanium 2 with 9 MB cache
Apple A7 (dual-core ARM64 "mobile SoC")
Core i7 (Quad)
AMD K10 quad-core 2M L3
Itanium 2 Madison 6M
Core 2 Duo Wolfdale
Pentium D Smithfield
Core 2 Duo Conroe
Itanium 2 McKinley
Cell
Core 2 Duo Wolfdale 3M
Pentium 4 Prescott-2M
Core 2 Duo Allendale
Pentium 4 Cedar Mill
AMD K8
Pentium 4 Prescott
Pentium 4 Northwood
Barton
Pentium 4 Willamette
Atom
Pentium III Tualatin
Pentium II Mobile Dixon
ARM Cortex-A9
AMD K7
Pentium III Coppermine
AMD K6-III
AMD K6
Pentium III Katmai
Pentium II Deschutes
Pentium Pro
Pentium II Klamath
Pentium
AMD K5
SA-110
Intel 80486
R4000
TI Explorer's 32-bit Lisp machine chip
ARM700
Intel 80386
Intel i960
ARM 3
Motorola 68020
DEC WRL MultiTitan
Intel 80286
ARM 9TDMI
Motorola 68000
Intel 80186
Intel 8086
Intel 8088
ARM 2
ARM 6
ARM 1
WDC 65C816
Motorola 6809
Novix NC4016
TMS 1000
Zilog Z80
WDC 65C02
RCA 1802
Intel 8085
Intel 8080
Intel 8008
Motorola 6800
MOS Technology 6502
Intel 4004
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
Year in which the microchip was first introduced
Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.
Licensed under CC-BY by the authors Hannah Ritchie and Max Roser.

Ley de Moore

- ¡Se cumplió durante 50 años!
  - ¡Siendo exponencial!
  - *"The greatest shortcoming of the human race is our inability to understand the exponential function"*, Albert Bartlett.
- Fue el motor principal de la gran revolución tecnológica en la que vivimos.
- Posibilitó una mejora de performance de un factor de **10 millones**.
- Y un factor de reducción de los costos de **10 mil**.
- Pero recuerden que la Ley de Moore **no se refiere ni a performance ni a costos**.

¿Fin de la Ley de Moore?

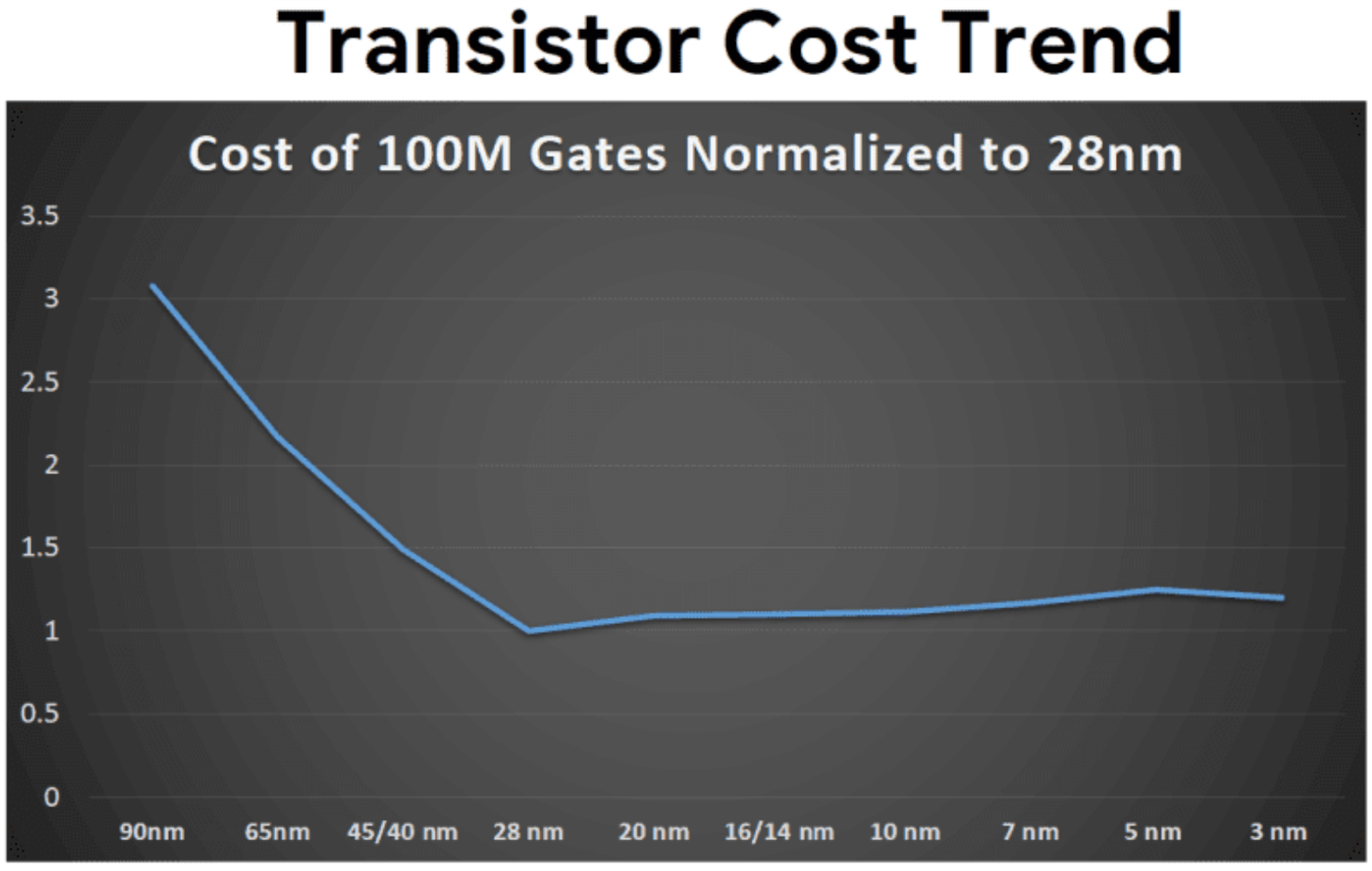
- Pareciera que está llegando a su fin
  - Intel Xeon E5 del 2010, 2.3 B transistores.
  - Intel Xeon E5 del 2016, 7.2 B transistores.
    - *¿Cuántos debería tener según la Ley de Moore?*

- Limitaciones por cuestiones físicas y de costos.
  - Algunas estimaciones pronostican el final entre el 2025 y el 2030.
- Aunque ya se predijo muchas veces este final.

¿Fin de la Ley de Moore?

- La tecnología de fabricación va a seguir mejorando, pero a un ritmo mucho menor.
  - Se habla de una **desaceleración** de la Ley de Moore.
- Deberán surgir nuevas técnicas que hagan mejor uso de esos transistores.
  - *"The end of Moore's Law could be the best thing that has happened in computing since the beginning of Moore's Law."*

¿Fin de la Ley de Moore?



- Transistor cost scaling (0.7X) stalled at 28 nm and remains flat gen over gen

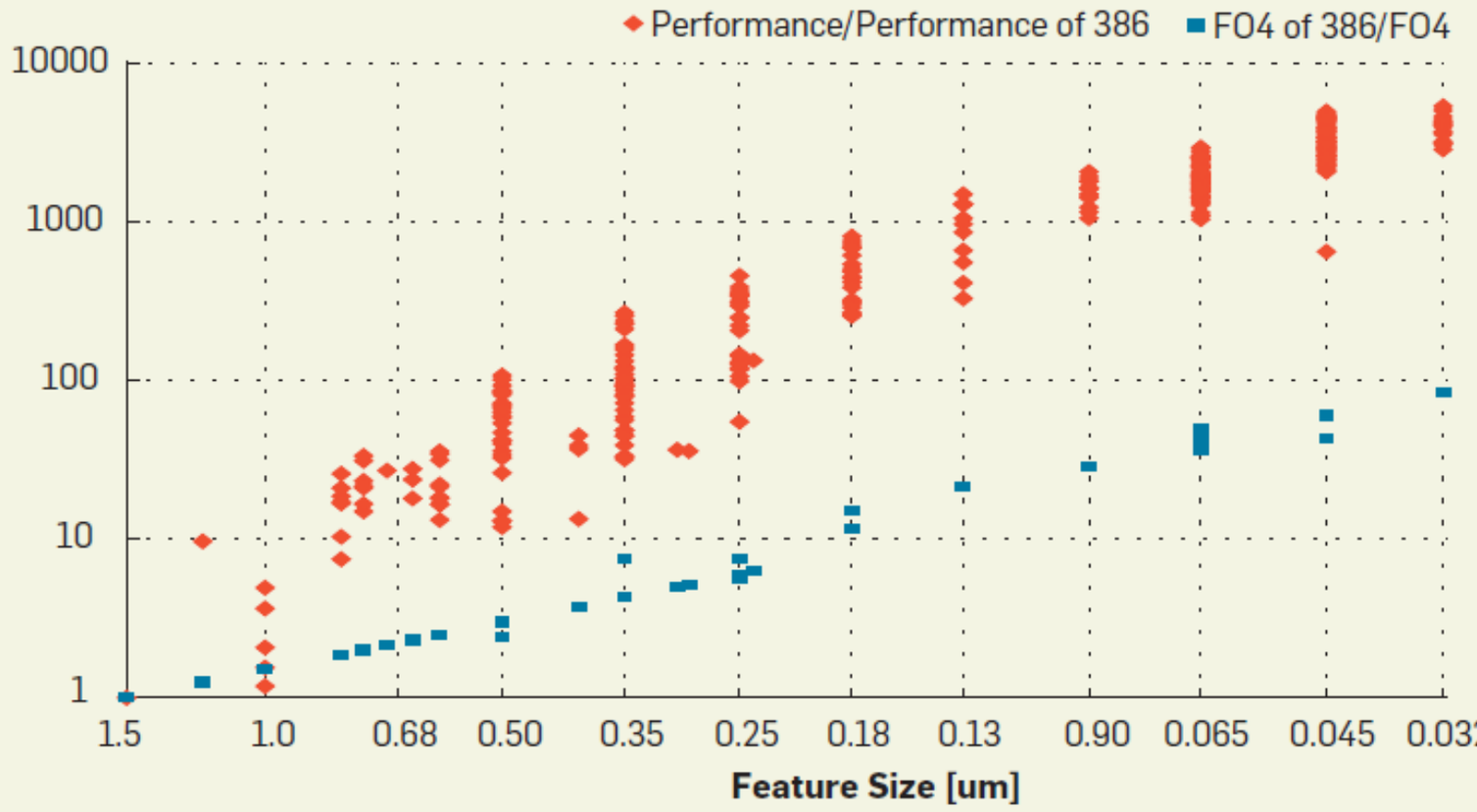
4

- Hay un notorio punto de inflexión en un *feature size* de 28 nm.
- Si un *feature size* más pequeño no resulta en transistores más baratos, *¿entonces por qué seguir achicándolo?*

Performance vs. *feature size*

- Disminuir el *feature size* también hace que los transistores sean más rápidos.

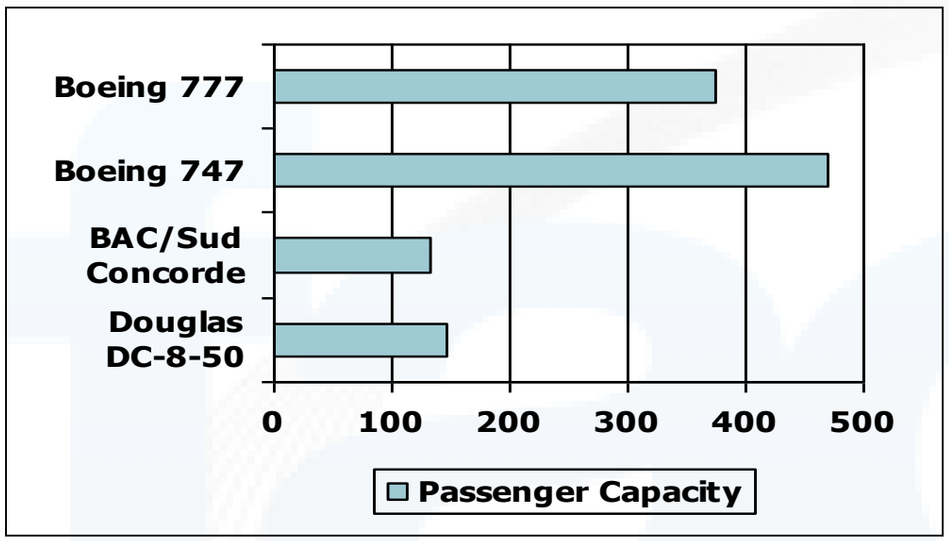
Figure 1. The diamonds indicate how processor performance actually scaled with time, while the squares denote how much speedup came from improving the manufacturing process.

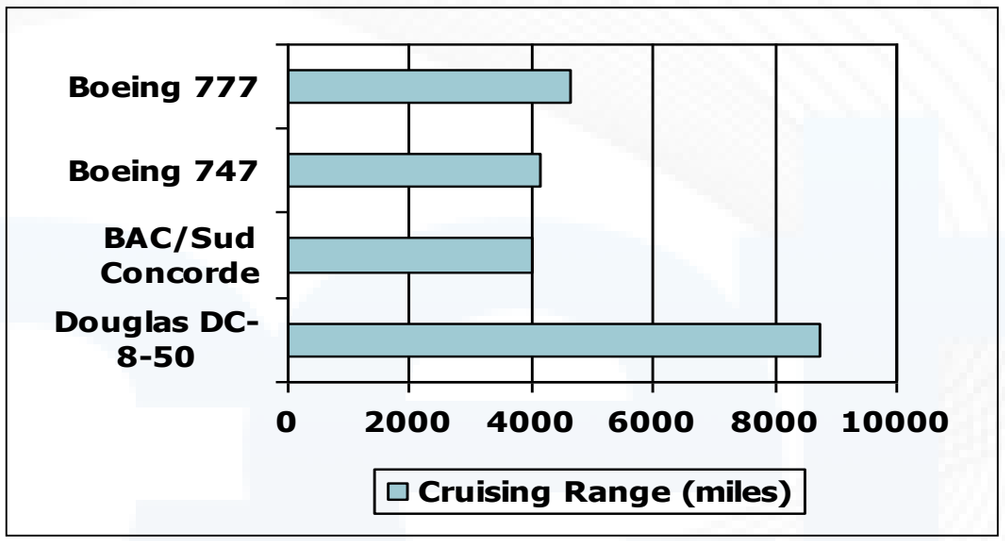


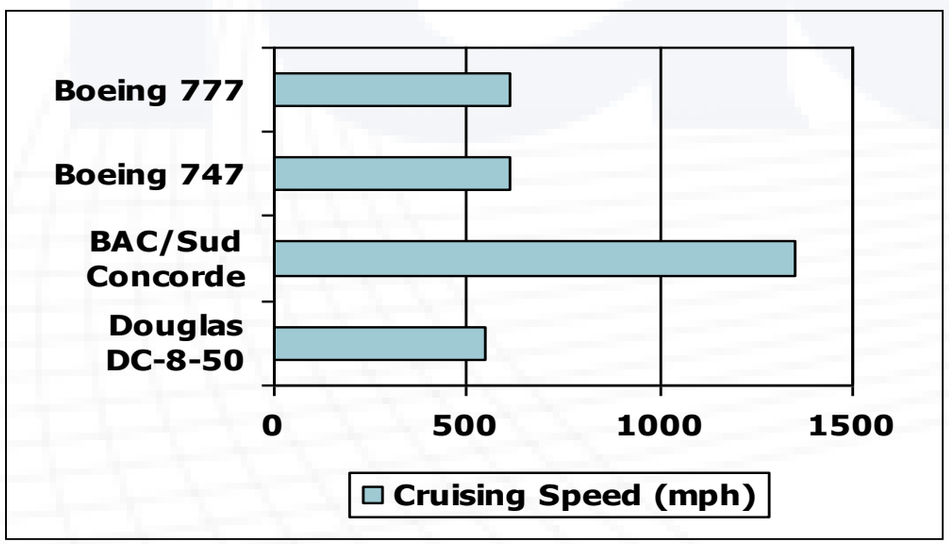
- Se observa que las mejoras tecnológicas en el proceso de fabricación contribuyen de igual manera que otros tipos de mejoras.

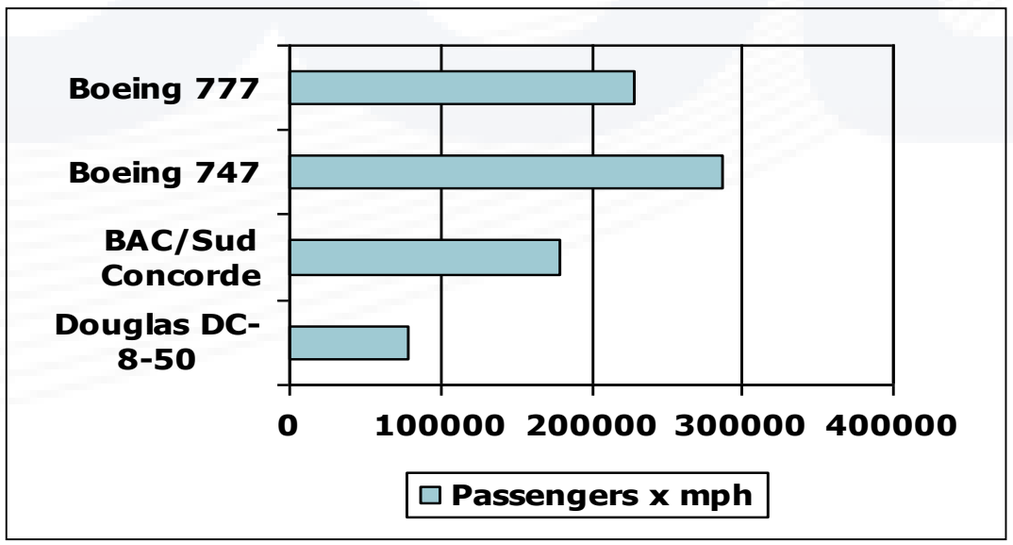
Performance

- Nos permite elegir inteligentemente, por sobre las modas del marketing.
- Clave para comprender las razones de diferentes estructuras de CPU.
- ***¿Cuál de estos aviones tiene mejor performance?***









Performance en computadoras

- Tres métricas fundamentales:
- **TIEMPO, TIEMPO y TIEMPO.**

- **Latencia** (Tiempo de respuesta):
  - ¿En cuánto tiempo se ejecuta mi trabajo?
  - ¿Cuánto esperar por una consulta a una base de datos?
- **Productividad**
  - ¿Cuántos trabajos pueden ejecutarse simultáneamente?
  - ¿Cuál es el tiempo de ejecución promedio?
  - ¿Cuánto trabajo se efectúa por unidad de tiempo?

Latencia vs. Productividad

- *¿Qué se mejora si cambiamos el CPU de una máquina por otro más rápido?*
- *Si agregamos una nueva máquina a un laboratorio para estudiantes, ¿qué mejoramos?*

¿Cuál tiempo medir?

- Tiempo total (*elapsed time*)
  - Cuenta todo (accesos a disco y memoria, I/O, etc.).
  - En general no es bueno para comparar. ***¿Por qué?***

- Tiempo de CPU (*cpu time*)
  - No cuenta I/O ni el tiempo que se usa para correr otros programa.
  - Puede separarse en tiempo del usuario y tiempo del sistema.

- **Nuestro interés: tiempo de CPU del usuario**
  - Tiempo para ejecutar las instrucciones de “mi programa”.

Definición de Performance

- Problema
  - La máquina A corre un programa en 20 segundos
  - La máquina B necesita 25 segundos.
  - **¿Cuánto más rápido es A que B?**

- Definimos la performance para un dado programa P que corre en la máquina X, como:
  Performance(P,X) = 1 / tiempo de ejecución(P,X)
  - La performance es inversamente proporcional al tiempo.

- Decimos que "X es n veces más rápida que Y“.
  PerformanceX/PerformanceY = tiempo(y)/tiempo(x)

Ciclos de reloj

- Los circuitos de hardware suelen estar gobernados por un reloj de pulsos constantes.
- El ciclo de reloj es el tiempo entre pulsos. T [seg/ciclo]
- La frecuencia de reloj es la cantidad de ciclos por segundos. f [Hz] [ciclos/seg] = 1 / T
- Ej: un reloj de 2 GHz tiene un tiempo de ciclo de 0.5 ns.
- Para iguales frecuencias de reloj, en lugar de hablar de tiempos de ejecución en segundos, se pueden usar los ciclos de reloj.
  - **[seg/programa] = [ciclos/programa] * [segundos/ciclo]**

Ejemplo

- Un programa demora 10 s en ejecutarse en una computadora A, cuya frecuencia es de 2 GHz.
- Se quiere diseñar una computadora B en la que el programa demore 6 s en ejecutarse.
  - Pero lamentablemente, la cantidad de ciclos necesarios se ve incrementada un 20%.
- *¿Qué frecuencia deberá tener la computadora B?*
  - Recordamos la última fórmula:
    - **[seg/programa] = [ciclos/programa] * [segundos/ciclo]**

Ecuación de Performance del CPU

- Dijimos que tiempo de ejecución es:
  - [seg/programa] = [ciclos/programa] * [segundos/ciclo]
- Podemos descomponer el primer término en:
  - [ciclos/programa] = [instrucciones/programa] * [ciclos prom/instruccion]
    - La cantidad de instrucciones es dinámica, no estática.
- Luego, podemos escribir la “**Ley de Hierro de la Performance**”:
  - [seg/programa] = [instrucciones/programa] * [ciclos prom/instruccion] * [segundos/ciclo]
  - **t = CI * CPI * T**
  - El CPI nos da información sobre el set de instrucciones, la implementación del ISA y sobre el programa.

Factores que influyen en la ecuación

$$\text{CPU Time} = \frac{\text{Instrucciones}}{\text{Programa}} \times \frac{\text{Ciclos de reloj}}{\text{Instruccion}} \times \frac{\text{Segundos}}{\text{Ciclo de reloj}}$$

- La utilidad de esta ecuación es que permite identificar cuáles son los factores que influyen directamente en la performance.
- En la cantidad de instrucciones (CI) influyen:
  - El lenguaje de programación, el algoritmo utilizado, el compilador utilizado y el **ISA de la máquina**.
- En el CPI influyen:
  - La **implementación del ISA** y la **organización del hardware**.
  - En menor medida, el compilador.
- En la duración del ciclo (T) o en la frecuencia (f) influyen:
  - La **organización del hardware** y la tecnología.

Ejemplo CPI

- Supongamos que tenemos dos implementaciones distintas de un mismo ISA, como RV32I: máquinas A y B, dos implementaciones multiciclo diferentes.
  - *¿Cuál de las siguientes variables serán siempre idénticas para un programa?*
    - *Frecuencia de reloj*
    - *CPI*
    - *Tiempo de ejecución*
    - *Cantidad de instrucciones.*

- Para cierto programa:
  - La máquina A tiene un ciclo de reloj de 10 ns y un CPI de 2,0.
  - La máquina B tiene un ciclo de reloj de 20 ns y un CPI de 1,2.
  - Las dos máquinas funcionan con el mismo compilador.
  - *¿Cuál máquina es más rápida para este programa y por cuánto?*

CPI

- Una característica muy importante del CPI es que es una métrica aditiva.
  - Puede descomponerse en varios factores que se suman.
- De las ecuaciones anteriores, podemos despejar CPI:
  - CPI = #total de ciclos / CI
- Considerando que el tiempo de ejecución es la suma de los tiempos de todas las instrucciones, se puede llegar a:
  - **CPI = sumatoria(CPI_i * F_i)**
  - CPI_i es la cant de ciclos de la instrucción *i*.
  - F_i = $\#instr_i$ / CI (frecuencia relativa de la instrucción *i)*
  - Nos permite conocer la **mezcla de instrucciones** de un programa.
- Los términos de la sumatoria anterior nos permiten analizar dónde se consume el tiempo de CPU.

Ejemplo CPI 2

- Para un programa en un determinado CPU tenemos la siguiente mezcla de instrucciones, con sus duraciones:

Operación	F(i)	CPI(i)	CPI(i) * F(i)	% tiempo
ALU	50%	1	0,5	33%
Load	20%	2	0,4	27%
Store	10%	2	0,2	13%
Branch	20%	2	0,4	27%
Totales	**100%**		**1,5**	

- *¿Cuál es el CPI promedio de esa mezcla de instrucciones?*
- *¿En cuál tipo de operación se utiliza mayor porcentaje del tiempo de ejecución?*
  - El % tiempo es CPI(i)*F(i) / CPI promedio.
- *¿Si pudiese reducir a la mitad el CPI de un solo tipo de operación, cuál elegiría?*

Quiz rápido

- *¿Cuál de las siguientes variables mide mejor la performance?*
  - # de ciclos para ejecutar un programa.
  - # de instrucciones de un programa.
  - # de ciclos por segundo.
  - # promedio de ciclos por instrucción.
  - # promedio de instrucciones por segundo [1/(cpi.T)]

MIPS: el triunfo del marketing

- **M**illones de **I**nstrucciones **P**or **S**egundo.
  - = 1 / (CPI * T * 10^6) = F / (CPI * 10^6)
- ¿Máquinas con diferentes conjuntos de instrucciones?
  - Para un mismo programa distinta cantidad de instrucciones.
  - Una con menos CPI puede necesitar más instrucciones.
- ¿Programas con diferentes mezclas de instrucciones ?
  - Cantidad dinámica de instrucciones es lo que cuenta.
  - Programas distintos tienen diferentes CPI y también MIPS.
- Por tanto MIPS nada que ver con performance.
  - Pasa lo mismo con los **GFLOPS** (operaciones de punto flotante)
  - Generalmente no miden en dónde se gasta el tiempo.
  - *Los cálculos no mienten, pero hay mentirosos que calculan.*

Ejemplo Falacia MIPS

- Dos compiladores se testean para una máquina de 1 GHz con 3 clases diferentes de instrucciones: A, B, y C, de 1, 2, y 3 ciclos respectivamente. Ambos compiladores se usan para producir código de una gran pieza de Sw.
  - El primer compilador genera 5 millones de instrucciones clase A, 1 millón clase B, y 1 millón clase C.
  - El segundo genera 10 millones de instrucciones A, 1 millón Clase B, y 1 millón Clase C.

- *¿Cuál compilador será más rápido de acuerdo a los MIPS?*

- *¿Y de acuerdo al tiempo de ejecución?*

Benchmarks: necesidad

- *“Si no se puede medir, no se puede mejorar”*, Lord Kelvin.
- Es difícil obtener las variables para los cientos de miles de programas en los cientos de miles de escenarios diferentes.
  - Que además van cambiando constantemente con el tiempo.
- Surgen **conjuntos estándar de programas** utilizados para comparar.
  - Ya sea sistemas diferentes.
  - O cambios a un mismo sistema.
  - Idealmente, son representativos de una amplia gama de programas similares.
- Dan forma a nuestro campo de acción
  - Los buenos aceleran el desarrollo, fijando objetivos.
  - Los malos ayudan a… los vendedores tramposos.

Benchmarks: ejemplos

- Juegos ‘difíciles’: Go, Ajedrez, A*.
- Benchmarks sintéticos:
  - Intentan lograr frecuencias medias de cargas de trabajo reales
  - **Dhrystone**: programas científicos, usado para el top500.
- Kernels: segmentos críticos de programas reales.
- Programas reales: gcc, spice, perl, zip.
- Mezcla de programas reales: **SPEC**
- Videojuegos: mejores ejemplos para gráficos.
- Sistemas embebidos: **EEMBC CoreMark**
- Machine Learning: *MLPerf*.
- IA: hay varios, todos recientes. ¡MMLU es un mal ejemplo!
- Pero… *¿cuál es el mejor benchmark?*

Benchmarks: comparativa

	Linpack	Dhrystone	SPEC CPU	CoreMark	MLPerf 0.5
Year	1977	1984	1989	2009	2018
Initial Target	Supercomputer	Systems programming	Unix Server	Embedded	Server (ML training)
Type	Library	Synthetic	Applications	Synthetic	Applications
Quality Reputation	Low	Low	High	Low	High
Free (no cost)	✓	✓	$1000 ($250 academia)	✓	✓
Easy to Port	✓	✓	✓	✓	No
Organization to Evolve Benchmark	✓	No	✓	✓	✓
Revision Frequency	None since 1991	None since 1988	3 years	Never	1 year (planned)
Number of Programs	1	1	10 SPEC89-23 SPEC2017	1	7
Single Summary Score	FLOPS/second	Speed Ratio	Geometric Mean of Speed Ratios	Speed Ratio	Weighted Mean Ratios + Std. Dev.
Developer	Academia	Academia	Academia & Industry	Industry	Academia & Industry

El Benchmark ideal debería ser…

- **Gratuito**.
- **Simple** de ejecutar, y de migrar entre dispositivos.
- Un **conjunto** de programas **reales**.
- Resumido en un único valor.
  - Con una media geométrica y una desviación estándar.
- Que sea **reproducible**:
  - Listar todo lo que se debe hacer para obtener un resultado (versión del sistema operativo, configuraciones del compilador, datos de entradas, configuración específica de la computadora (frecuencia, tamaño cache y tiempo de acceso, tamaño de memoria, etc.))
- Frecuentemente revisado y respaldado por alguna organización.
  - Incluyendo industrias y universidades.

Performance – Un ejemplo real de SW

- Diferentes versiones de un algoritmo de multiplicación de matrices de 4096x4096:

Version	Implementation	Running time (s)	GFLOPS	Absolute speedup	Relative speedup
1	Python	25,552.48	0.005	1	–
2	Java	2,372.68	0.058	11	10.8
3	C	542.67	0.253	47	4.4
4	Parallel loops	69.80	1.969	366	7.8
5	Parallel divide and conquer	3.80	36.180	6,727	18.4
6	plus vectorization	1.10	124.914	23,224	3.5
7	plus AVX intrinsics	0.41	337.812	62,806	2.7

- La versión 1 es la más simple: minimiza tiempo de desarrollo.
- La v3 es más difícil de programar, pero es ~50 veces más rápida.
- La v4 y la v6 aplican paralelismo (Temas 03 y 09).
- La v5 explota el principio de localidad (Temas 11 y 12).
- La v7 aplica conceptos que veremos en los Temas 04 y 08.

Performance – Un ejemplo real de SW

- Diferentes versiones de un algoritmo de multiplicación de matrices de 4096x4096:

Version	Implementation	Running time (s)	GFLOPS	Absolute speedup	Relative speedup
1	Python	25,552.48	0.005	1	–
2	Java	2,372.68	0.058	11	10.8
3	C	542.67	0.253	47	4.4
4	Parallel loops	69.80	1.969	366	7.8
5	Parallel divide and conquer	3.80	36.180	6,727	18.4
6	plus vectorization	1.10	124.914	23,224	3.5
7	plus AVX intrinsics	0.41	337.812	62,806	2.7

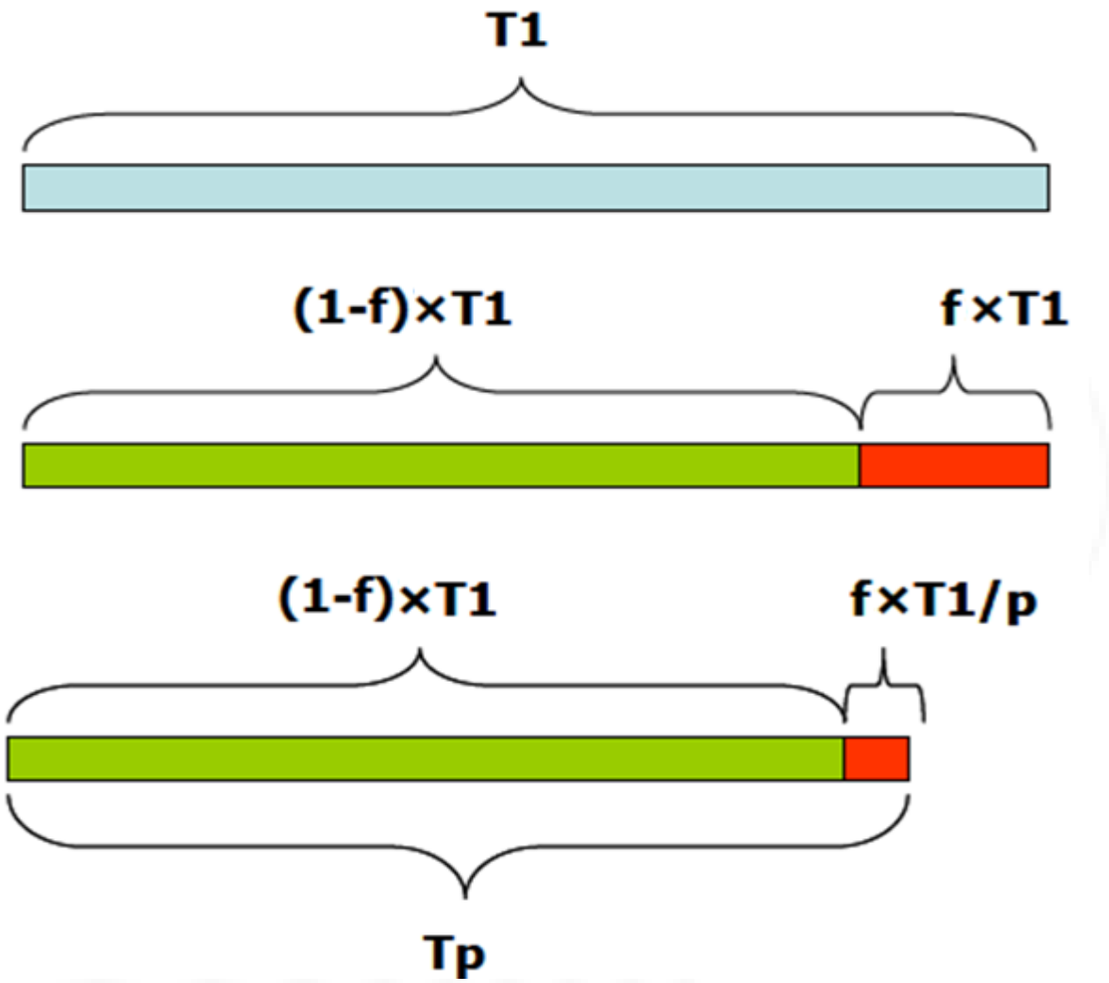
- ¡Con todas las mejoras, la v7 es ~60.000 veces más rápida que v1!
  - Ejecutándose sobre el mismo hardware.
  - Si lo ejecutamos sobre una GPU… ~360.000 veces más rápido.
- ***¿Cuál de todas las versiones es mejor?***
  - Oportunidad para Ingenieros en Computación.

Ley de Amdahl

- Es una limitación a la mejora de performance.
- Evalúa el efecto total de una mejora sobre una parte de la tarea.
  - Es una falacia mejorar un aspecto de la computadora y esperar que la totalidad se mejore de igual forma.
  - **$T_{mejora} = T_{noAfectado} + (T_{afectado} /$Factor de mejora$)$**

Ley de Amdahl

- Supongamos que una mejora acelera una parte *f* de la tarea en un factor *p*, y que no influye en el resto de la tarea, entonces:
  - Tp = T1 * ((1-*f*) + *f*/*p*)
  - **A = 1 / [(1-*f*) + *f*/*p*]**



- **Corolario (Gran Idea en Arq Comp): Hacer más rápido el caso más común.**

Ley de Amdahl – Ejemplos

- Mejoramos una máquina para que todas las instrucciones de punto flotante se ejecuten 5 veces más rápido. Si el tiempo de ejecución de cierto benchmark antes de la mejora es 10 s y la mitad del tiempo del mismo se pasaba ejecutando instrucciones de PF:
  - *¿Cuál será el nuevo tiempo de ejecución? ¿Cuánto la aceleración total?*
  - *¿Cuánto debería ser la mejora en las instrucciones de PF para lograr un tiempo de ejecución de 5 seg?*
- Supongamos que el área de un chip se divide en un 56% en bloques lógicos y el resto en bloques de I/O. Si mediante una mejora del proceso de fabricación puede reducirse a la mitad el área de los bloques lógicos, sin tocar los bloques de I/O.
  - *¿Cuánto será el área del nuevo chip relativa al original?*

Consumo de Energía

- En tecnología CMOS, cada transición lógica (de 0 a 1 o viceversa) disipa energía de conmutación.
  - La energía disipada depende de la capacitancia de carga y del cuadrado de la tensión de alimentación.
- **$P = n * A * C * V^2 * f$**, donde:
  - n es la cantidad de transistores,
  - A es el factor de actividad, la probabilidad promedio de que un transistor transicione.
  - C es la capacitancia de carga promedio,
  - V es la tensión de alimentación,
  - f es la frecuencia de reloj.
- Esto es el consumo **dinámico** de energía.
  - El consumo estático está dado por la Ley de Ohm.

¿Cómo podemos reducir el consumo?

- **$P = n * A * C * V^2 * f$**
- Disminuyendo la frecuencia.
  - Pero se desea velocidad. Además un trabajo más rápido o más lento consume lo mismo, porque demorará más tiempo.
- Disminuyendo la tensión de alimentación.
  - Obliga a bajar la frecuencia, porque aumenta el delay.
  - Menor tolerancia a ruido.
- Disminuyendo la cantidad de Transistores.
  - Pero más transistores permiten más trabajo.
- Disminuyendo C – con mayor miniaturización
  - Se hace constantemente, pero tiene limitaciones y es costoso.
- Conclusión: reducir el consumo de energía va en contra de otros objetivos.

Consumo de energía: el *Dennard Scaling*

- Observó (en 1974) que con la reducción del *feature size* de los transistores (0.7x), ocurría lo siguiente:
  - El área se reducía a la mitad. O se podía poner el doble de transistores en la misma área.
  - Se reducía la tensión de los transistores (para mantener constante el campo eléctrico), la capacitancia y los retardos de propagación.
  - **Implicaba que se podía aumentar la frecuencia, manteniendo el consumo de energía constante.**

Component	Dicription	Past scaling ratio
n	Number of transistors	2
C	Capacitance/Transistor	0.7
V	Voltage	0.7
f	Frequency of operation	1,4
POWER		~1

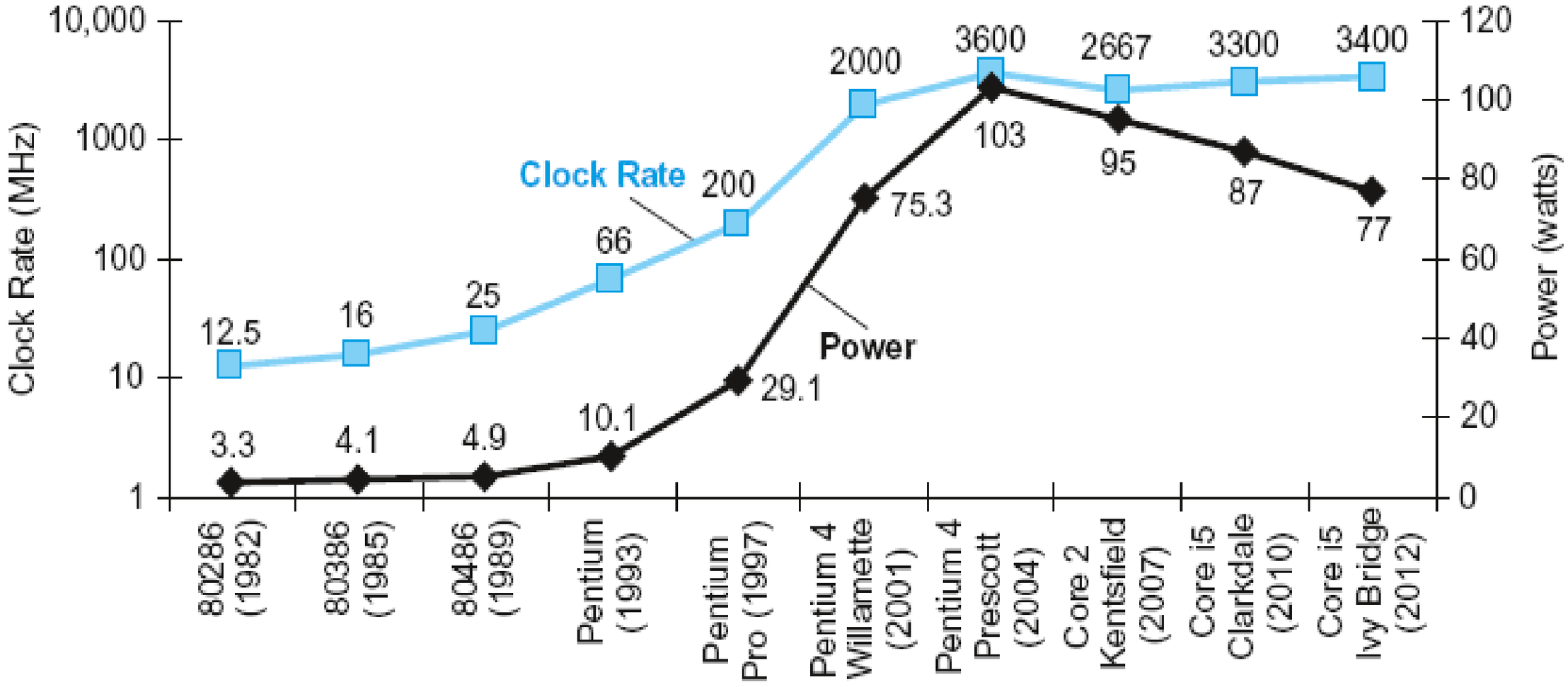
- **¿Por qué la frecuencia aumenta 1,4?**
- **¿Cómo es que el producto da 1?**

Fin de *Dennard Scaling*

- Gran importancia hasta 2006 aprox.
  - Cuando los transistores se hicieron muy pequeños, perdió validez.
    - Corrientes de pérdida, tensiones de umbral.

Component	Dicription	Past scaling ratio	Current scaling ratio
n	Number of transistors	2	2
C	Capacitance/Transistor	0.7	0.7
V	Voltage	0.7	0,95
f	Frequency of operation	1,4	1,1
POWER		~1	~1.4

Consumo de Energía vs frecuencia



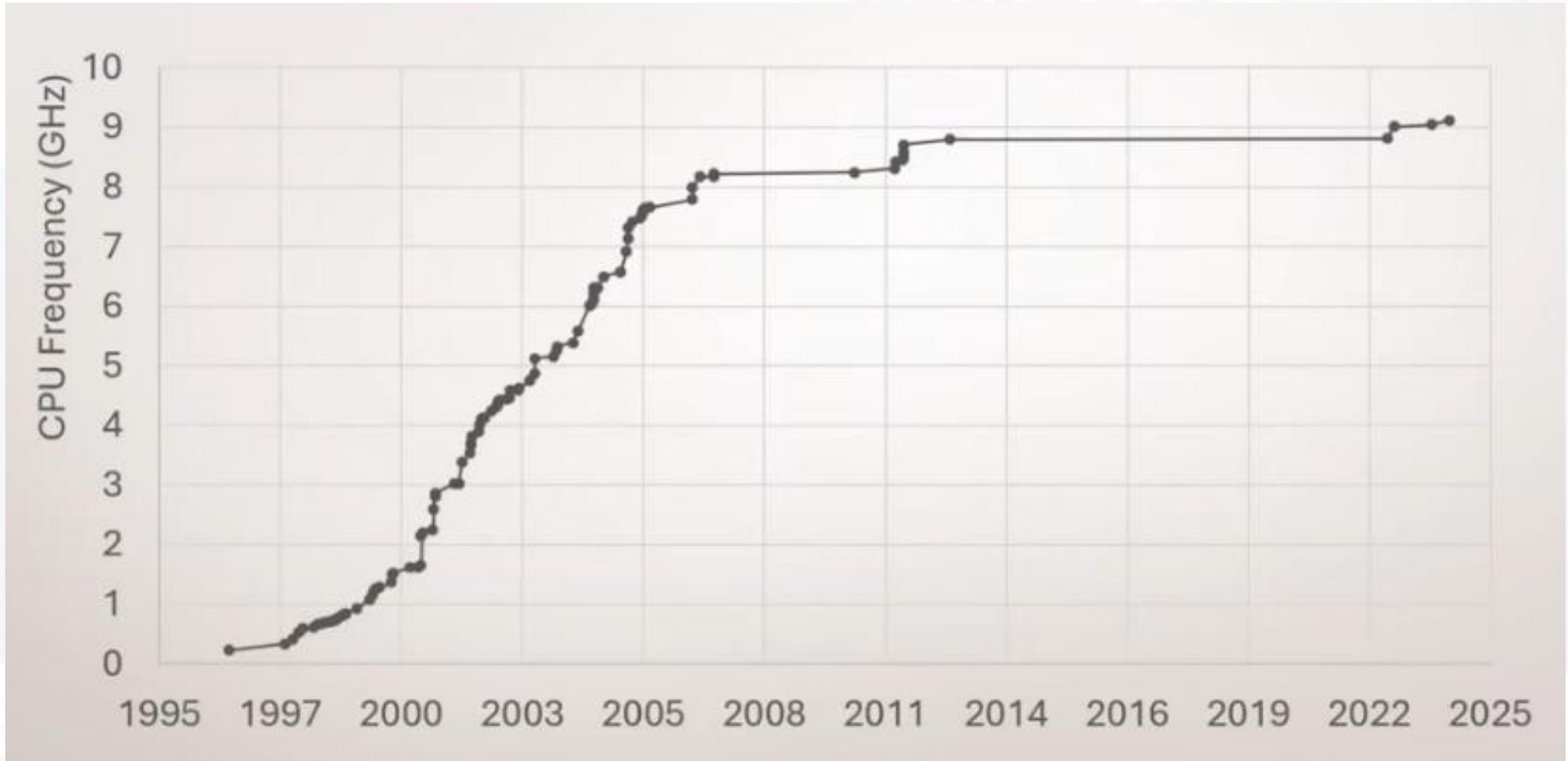
- La performance aumentaba incrementando la frecuencia de reloj.
- Hasta el año 2004, cuando salió el Pentium 4.
  - Tenía una densidad de energía (W/mm^2) igual a la de un reactor nuclear.

The Power Wall

- Hasta el año 2004, el diseño era motivado por costo y performance.
- A partir de entonces, se empezó a considerar el consumo de energía como criterio principal.
  - *¿Cómo seguir mejorando la performance, pero sin aumentar el consumo de energía?*
- Motivó un cambio de paradigma.
  - Surgieron multiplicidad de dispositivos móviles personales, operados a batería y conectados a Internet.
  - También apareció el concepto de *Cloud Computing* y el de *Software as a Service*.
- La evolución tecnológica requiere: Más portabilidad → Más miniaturización → Baterías más pequeñas y duraderas → Menor consumo de energía.

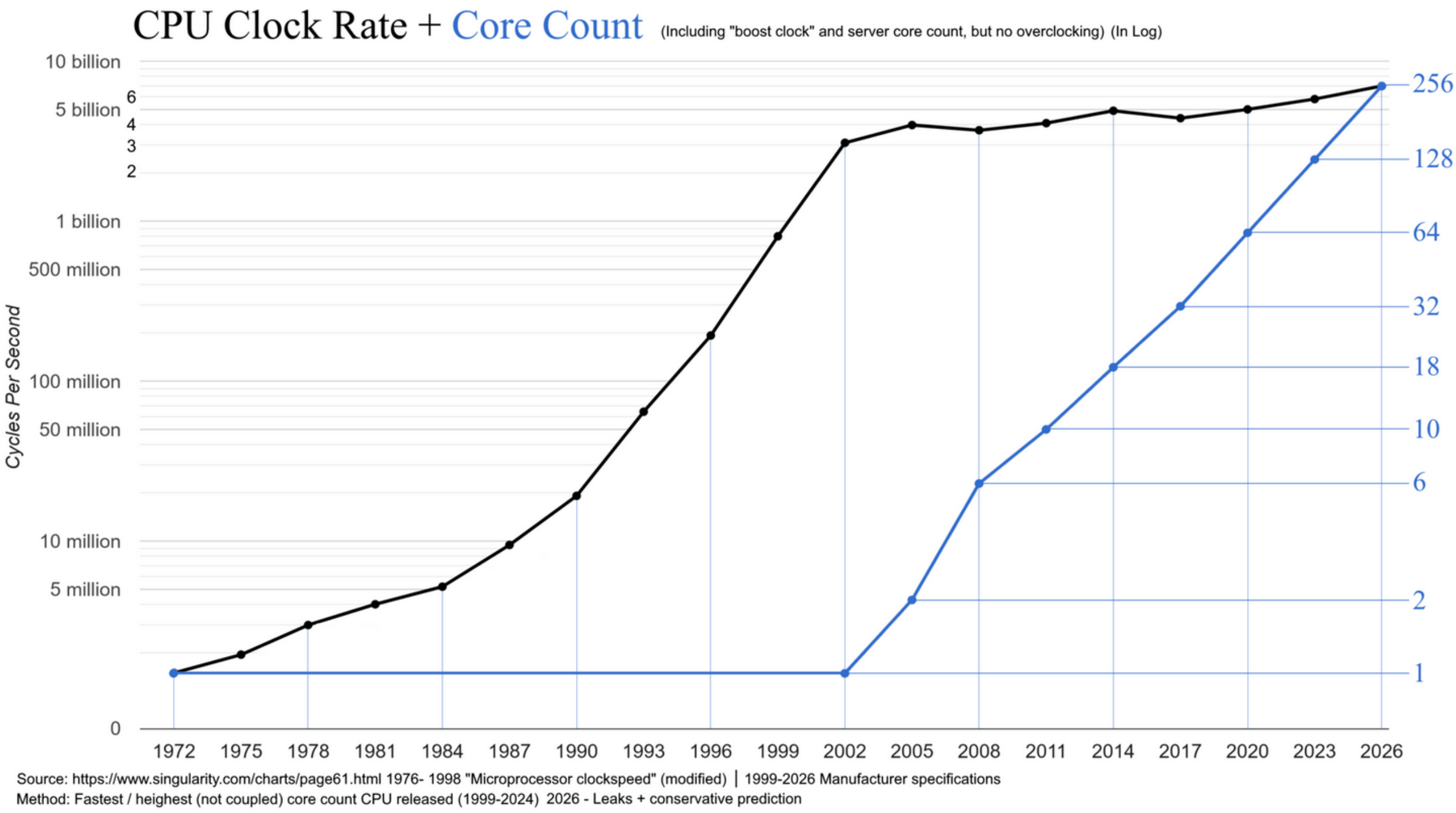
Impacto de la *Power Wall*

- Evolución del récord mundial de overclocking de procesadores
  - https://skatterbencher.com/cpu-overclocking-world-record-history/



- En 1999 se consiguió llegar a 1 GHz, y en 2007 a 8 GHz.
- Pasaron 16 años para llegar a los 9 GHz en 2023.

Impacto de la *Power Wall*



Falacia: Bajo consumo al no hacer nada

- En una notebook trabajando al 100%, el procesador consume aprox. el 50% de la energía.
  - O sea que si el procesador no consumiera nada, la batería duraría apenas el doble (Amdahl).
- En un servidor, un benchmark obtuvo que:
  - Con una carga del 100%, consume 258 W.
  - Con una carga del 50%, consume 170 W (66%).
  - Con una carga del 10%, consume 121 W (47%).
  - **El consumo de energía no es lineal con la carga de trabajo.**
- Estadísticas de un Data Center de Google:
  - En general opera con una carga entre 10% y 50%.
  - Carga del 100%, menos del 1% del tiempo.

Consumo de Energía: Otras implicancias

- En servidores:
  - La disipación de calor limita la cantidad de equipos que pueden colocarse en una sala.
  - La disipación de calor aumenta la cantidad de fallas, disminuyendo la **confiabilidad** (Tema 13).
  - Los equipos deben ser refrigerados con AA.
  - La tarifa eléctrica suele ser muy alta.
- En dispositivos portátiles:
  - La batería suele ser el segundo componente más grande y más pesado.
  - La batería tiene una energía acumulada similar a la de medio cartucho de dinamita.
    - De vez en cuando alguna explota y sale en las noticias.

Consumo de Energía: Otras implicancias

- En Datacenters para IA:
  - Una consulta en ChatGPT 4 consume en promedio 2.9 Wh.
    - Aproximadamente se realizaban 9.000 millones por día.
    - Sin considerar generación de imágenes, videos, etc.
  - Una consulta en ChatGPT 5 consume en promedio 18.9 Wh.
    - ¡Aumento de 6x en dos años!
  - La GPU H100 de NVIDIA, posee un consumo pico de 700 W a pleno trabajo.
    - La B200, 1000 W. La MI-355X de AMD, 1400 W.
  - Con una utilización promedio del 61%, el consumo estimado anual de una sola H100 es de 3740 kWh.
    - Probablemente, más que el de sus casas.
  - Se vendieron aprox 3.5 millones de esas GPU a nivel mundial
    - Consumen 13.100 GWh anual.
      - Eso es más que muchos **países**, como Paraguay, Lituania o Guatemala.
      - Argentina tiene un consumo anual estimado de 127.000 GWh.

Consumo de Energía: Otras implicancias

- Además, ¡esas GPUs no funcionan solas!
  - Hay que sumar también CPUs, placas de red, cables, discos, etc.
  - Un rack con 32 de estas placas consume aprox 80 kW.
  - Las GPU representan sólo el 28% del consumo total del rack.
- Un estudio de 2024 estima que el consumo total en IA en 2026 será 10 veces el de 2023.
  - Otro crecimiento exponencial, pero sus implicancias son negativas.
  - A mediano plazo es insostenible (más sobre esto en Tema 15).
- Estos Datacenters para IA están desplazando varios ejes de diseño:
  - Los consumos de energía rondan los GW, 25x lo que consume la supercomputadora más rápida (en Nov-25).
  - Instalarlos cuesta aproximadamente 50.000 millones de dólares, 100x lo que cuesta la supercomputadora más rápida (en Nov-25).
  - ¡Se está usando “GW de capacidad de cómputo” directamente como métrica de performance!

Ejemplo integrador (real)

- Intel lanzó a fines del año 2022 el procesador i9-13900KS.
  - El primer procesador para PC en alcanzar una frecuencia de 6 GHz (sin overclocking).
  - Líder en varios benchmarks (en enero 2023).
    - Por un 1.5% promedio de ventaja. Un 4% mejor que el mejor de AMD.
  - Primer procesador para PC en alcanzar un consumo pico de 320 W.
  - U$S 699 de precio base.
    - Un 18% más caro que el segundo más rápido en los benchmarks. Un 91% más caro que el mejor de AMD (Ryzen 7 5800X3D).

- Es el más rápido, pero…
  - *¿Conviene si consideramos performance/watt?*
  - *¿Conviene si consideramos performance/$?*
  - *¿Conviene si consideramos performance/watt/$?*
  - *¿Tendremos motherboard, cooler, fuente y gabinete para soportarlo?*

Resumen final sobre *Price*

- Chips rectangulares en obleas circulares de tamaño fijo.
- Los costos de un chip son proporcionales al cubo de su área.
  - Es deseable que los chips sean más pequeños.
- El área del chip se reduce mejorando el proceso de fabricación.
  - 0,7x es la escala entre dos *feature size* consecutivos.
    - Permite que el área se reduzca aproximadamente a la mitad.
- El costo aumenta al mejorar el proceso de fabricación.
- La Industria de la Computación fue y sigue siendo impulsada por la Ley de Moore.
  - Actualmente en desaceleración.
- Nuevos enfoques: *Chiplets* y *Waferscale*.

Resumen final sobre *Performance*

- ¡El tiempo de ejecución es la medida de la performance de una computadora!

$$\text{CPU Time} = \frac{\text{Instrucciones}}{\text{Programa}} \times \frac{\text{Ciclos de reloj}}{\text{Instruccion}} \times \frac{\text{Segundos}}{\text{Ciclo de reloj}}$$

- Performance es la inversa del tiempo de ejecución, y siempre se calcula relativa a algo.
- Para una determinada máquina, se puede mejorar performance si:
  - Mejora el ciclo de reloj (sin que haya efectos adversos en CPI).
  - Mejora en la organización del CPU para que baje el CPI.
  - Mejora compiladores para bajar CPI y/o número de Instrucciones.

Resumen final sobre *Performance*

- En computación, el tiempo tiene dos facetas diferentes pero vinculadas: latencia y productividad.
- Ley de Ahmdal: Incremento en velocidad está limitado por la parte no mejorada de una tarea.
  - Pifia: Esperar que la mejora en un aspecto de la performance de una máquina afecte en igual medida la performance total. (MIPS, MFLOPS).
  - Gran Idea: hacer más rápido el caso más común.
- Se pueden crear buenos productos cuando se parte de buenos Benchmarks.
  - Si no hay buenos benchmarks e indicadores, entonces en la elección entre mejorar productos y mejorar ventas, éstas siempre ganan.

Resumen final sobre *Power*

- Consumo dinámico de energía: $P = n * A * C * V^2 * f$
  - Reducir el consumo va en contra de otros objetivos.
- Fin del *Dennard Scaling* (2006): ya no se puede confiar en mejoras del proceso de fabricación para reducir el consumo de energía.
- *Power Wall* (2004): ya no se puede aumentar la performance aumentando la frecuencia de reloj.
  - Consumo de energía se incorpora como tercer variable de diseño.
- El consumo de energía está rodeado de muchas otras implicancias que no pueden dejarse de lado.
- La evolución tecnológica requiere máquinas que consuman menos energía.
  - Reducir el consumo de energía no es sólo cuestión de tecnología sino también de diseño (Tema 15).
  - Cuidar la energía es cuidar el mundo y mejorar los costos.

Agradecimientos

- Las diapositivas de este tema fueron basadas en las realizadas por el Ing. Daniel Cohen.
- A su vez inspiradas en las clases del curso CS152 de la Universidad de Berkeley, California, USA.
  - Realizadas por los Prof. D. A. Patterson, John Lazzaro, Krste Asanovic.