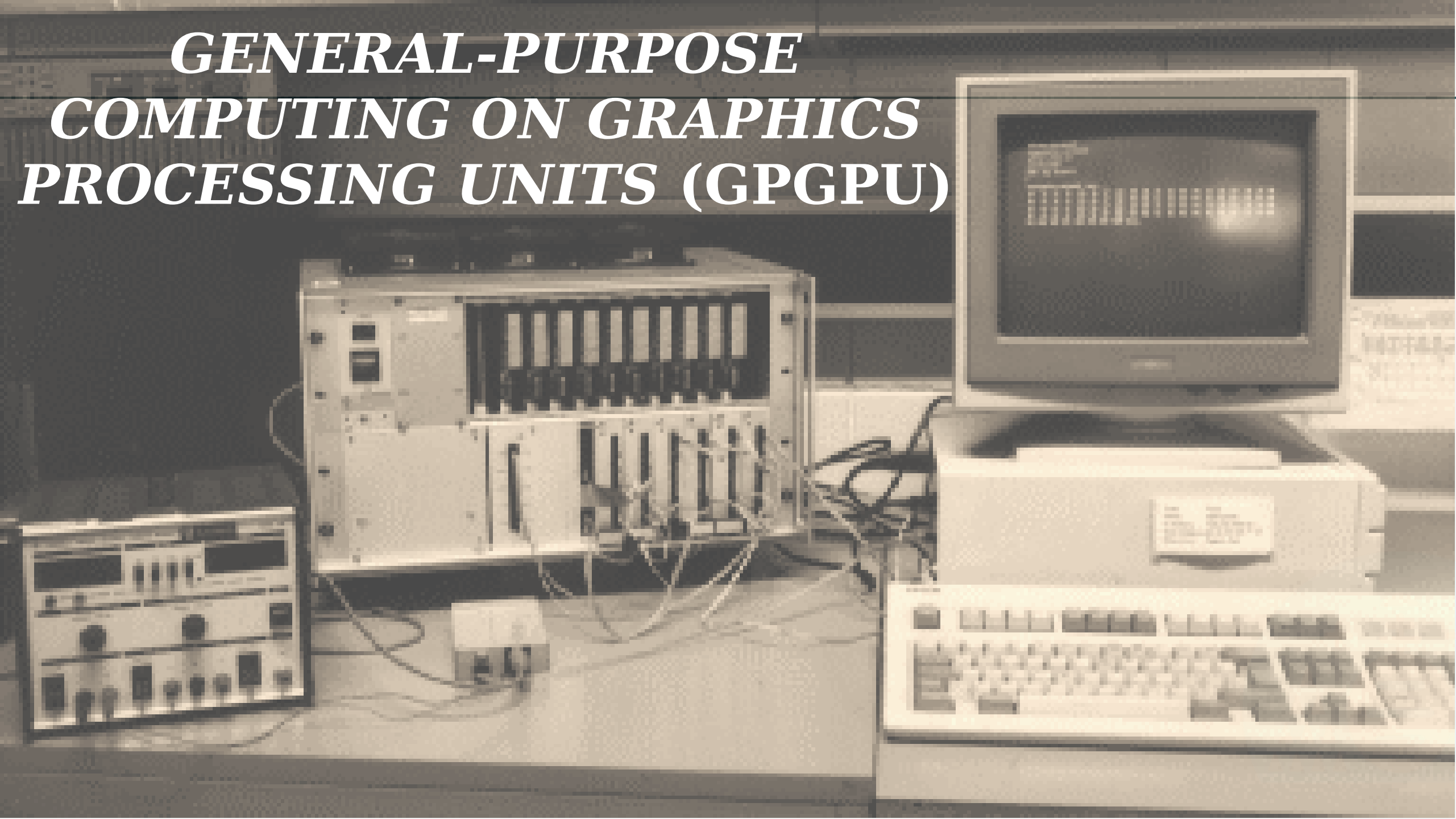
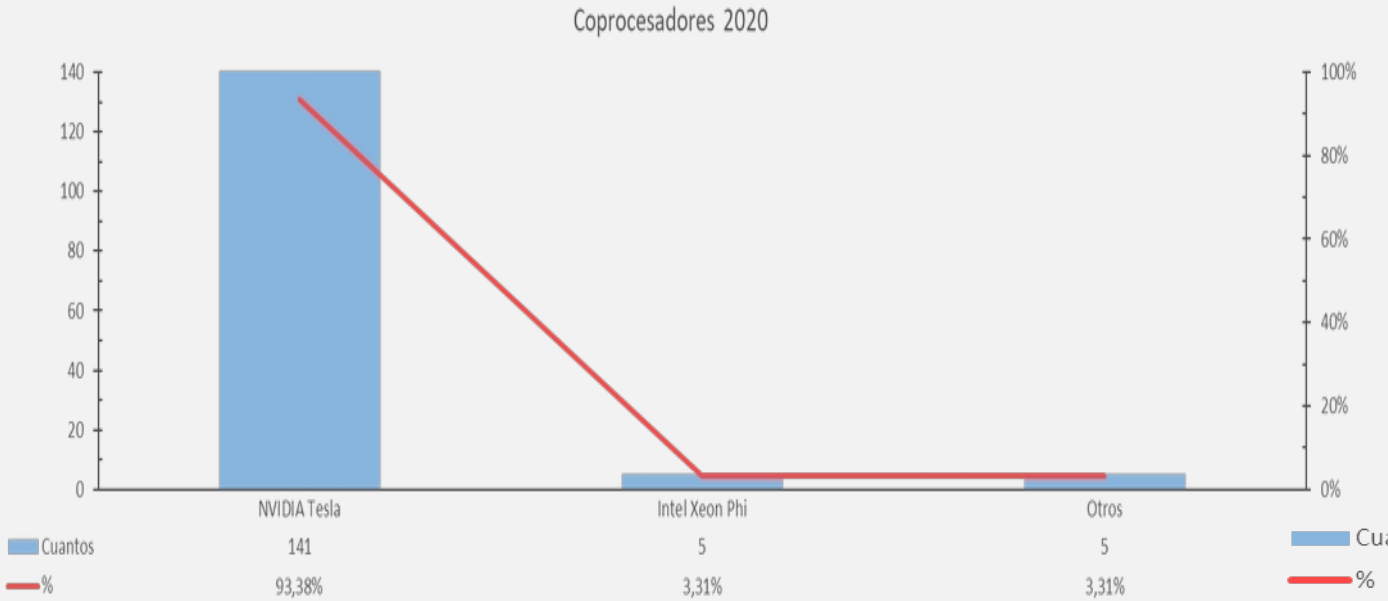
GENERAL-PURPOSE COMPUTING ON GRAPHICS PROCESSING UNITS (GPGPU)

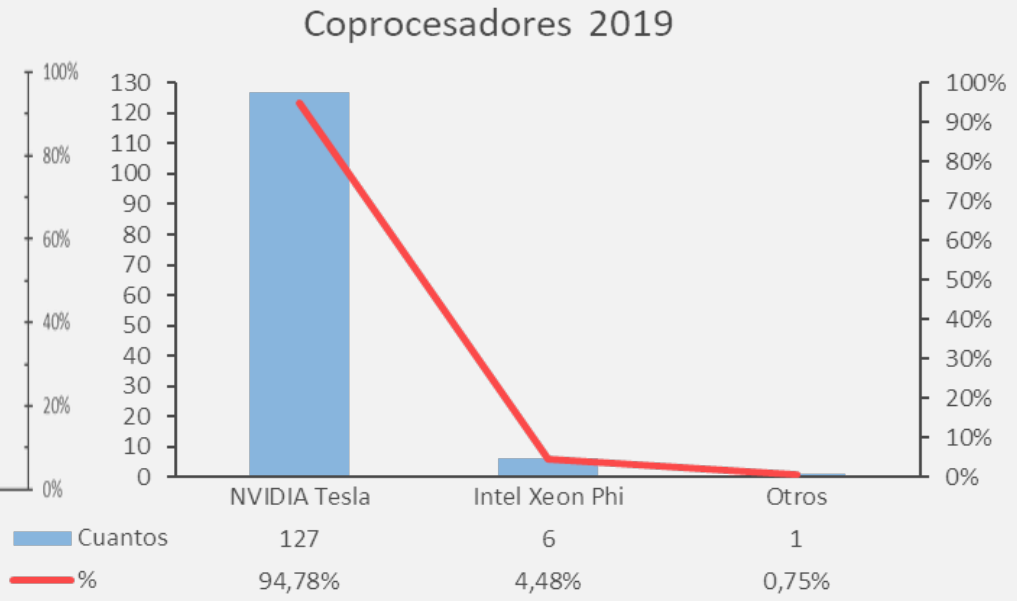


PRESENTACIÓN

Motivación

- Mantener la ley de *Moore* ha implicado el uso de coprocesadores.
- Las siguientes gráficas, del www.top500.org, lo dicen todo.





- No hay indicios de cambio a corto/medio plazo. NVIDIA domina ¿Intel Xᵉ (eXascale for Everyone, 2020) / AMD RDNA 2 (Radeon DNA 2, 2020)?
- La Era de la Computación Elástica (the *Age of Elastic Computing*): adaptación automática de los recursos de la computación a los cambios en la carga de trabajo.

PRESENTACIÓN

Contextualización

- Una introducción
- Memoria Compartida. SIMD – SIMT – Paralelismo de grano fino.

¿Se recuerda la fase 1 de la metodología de diseño?

No son autónomos, necesitan un sistema anfitrión. *PCI Express, NVLink*, etc.

- Cuando el sistema anfitrión y el coprocesador se usan conjuntamente para resolver un problema → Sistema Heterogéneo (o híbrido según la fuente y/o la tendencia imperante).
- Tiene su propia memoria, compleja. El espacio de direcciones físico de memoria no es único (no compartido físicamente con el anfitrión).
- El software actual proporciona un espacio de dirección virtual unificado para acceder a toda la memoria.

PRESENTACIÓN

Análisis del Rendimiento

- El direccionamiento virtual transparente (en CUDA se llama *Unified Memory*) es una característica del modelo de programación: las transferencias de información existen.

- Hay que modelar comunicaciones, sincronización y cálculo.
- Encontrar el referente secuencial puede ser complejo.
- Importante equilibrar la carga en función de la potencia de cada subsistema.

- En resumen:
  - Cada subsistema tiene su constante de cálculo. No se puede simplificar.
  - Cada subsistema tiene su tamaño de problema. Se puede expresar en función del tamaño general.
  - Hay comunicaciones, síncronas o asíncronas, con constantes distintas según sea entre el sistema anfitrión o entre los coprocesadores. Como mínimo considerar las comunicaciones anfitrión – coprocesadores.

NVIDIA: A BUSINESS OVERVIEW

- 1993 Jen-Hsun Huang, Chris Malachowsky, y Curtis Priem fundan la compañía NV (*Next Version*). NVIDIA vino después, de la palabra latina *invidia* (de *inviduos*, hostil, envidia), "*la que más les gusto conteniendo NV*".
- 1995 NV1 primer producto de la compañía.
- 1998 La Fabless Semiconductor Association la elige, por segundo año consecutivo, compañía más respetada de la industria de semiconductores.
- 1999 Oferta pública a $12 acción (4 *splits* "1000 acciones de 1999= 12000 en 2020"; $550 acción hoy).
- 2001 La 1ª compañía en facturar más rápidamente $1000 millones de ingresos. Entra en el S&P 500.
- 2002 Fortune la reconoce como la compañía de mayor crecimiento de EE.UU.
- 2003 La Stanford Business School Alumni Association la nombra compañía emprendedora del año.
- 2006 Más de 500 millones de procesadores vendidos.
- 2007 Forbes la nombra compañía del año. Gana un Emmy (2010, 2011, 2012 presente en los Oscar).
- 2012 Newsweek la designa la sexta compañía más ecológica de América.
- 2013 Compra The Portland Group (antes/después compra más empresas, esta tiene relevancia para PCP).
- 2019 Toyota, Volvo, Audi, Xbox, PlayStation 3, NASA, ORNL-Titán, etc. usan tecnologías NVIDIA.
- 2020 Adquiere MELLANOX y ARM (setiembre, $40.000 millones).

NVIDIA: A COMPUTATIONAL OVERVIEW

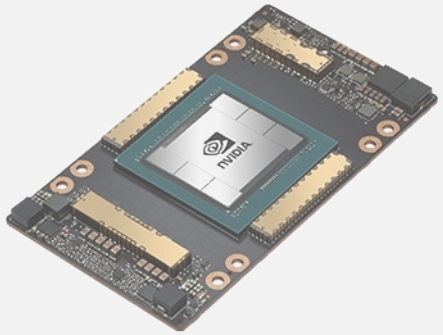
- 1999 NVIDIA “inventa” la GPU (*Graphics Processing Unit*). Presenta las familias *Quadro* y *GeForce* (el nombre GeForce fue por concurso). GeForce 256 la *primera* GPU.
- 2006 Presentación de CUDA (*Compute Unified Device Architecture*).
- 2007 Lanzamiento de la arquitectura **Tesla** (la primera realmente computacional).
- 2008 Lanzamiento del procesador móvil Tegra (consumo 30 veces menor).
- 2009 Lanzamiento de la arquitectura **Fermi**.
- 2012 Lanzamiento de la arquitectura **Kepler**, primera GPU virtualizada y tabletas/Smartphones con Tegra 3.
- 2014 Lanzamiento de la arquitectura **Maxwell**, del procesador Tegra K1 y del sistema Jetson TK1.
- 2015 Lanzamiento del procesador Tegra X1 y del sistema Jetson TX1.
- 2016 Lanzamiento de la arquitectura **Pascal** y del superordenador DGX-1 enfocado a aplicaciones de IA.
- 2017 Lanzamiento de la arquitectura **Volta** y del sistema Jetson TX2 (IA gana protagonismo).
- 2018 Lanzamiento **Turing** (Volta con *Ray-Tracing Cores*), del DGX-2 y del Jetson AGX Xavier. Elimina el *apodo* Tesla (Tesla T4 fue la última con el apodo) para evitar confusiones con *Tesla Automotive*.
- 2019 Lanzamiento de los sistemas Jetson Xavier NX y Nano (para IoT).
- 2020 Lanzamiento de la arquitectura **Ampere** (NVIDIA A100 SXM 4 o A100 PCIe, segundo semestre 2020).

ARQUITECTURA

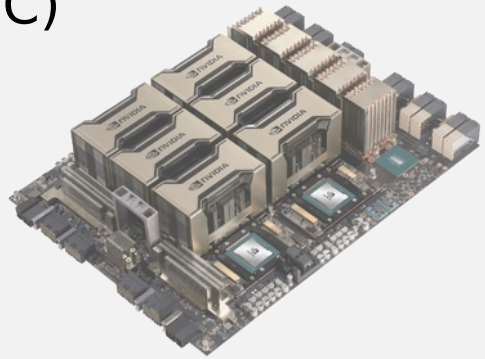
- Cada generación (arquitectura) presenta **características** diferenciadoras y un amplio sustrato común. Cada arquitectura presenta **capacidades** concretas con un núcleo común muy fuerte.
- Hay varias familias (GeForce, Quadro, Tegra, etc.). Las diferencias entre familias de la misma arquitectura son de potencia, memoria, consumo, etc., pero no de características/capacidades. P. ej. NVIDIA A100 PCIe 11000€, NVIDIA V100 5000€, Jetson AGX Xavier 700€, Jetson Nano 100€ (V100, Xavier y Nano son todos arquitectura Volta).

Ecosistema 2020 con GA100 (arquitectura Ampere con litografía de 7 nm fabricada por TSMC)

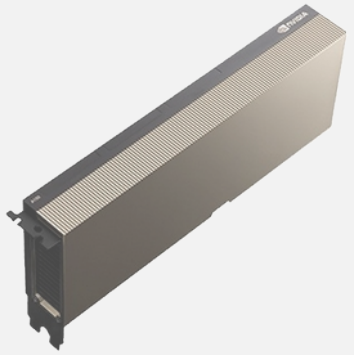
DGX



A100 SXM4



HGX



A100 PCIe

ARQUITECTURA GA100: NOVEDADES

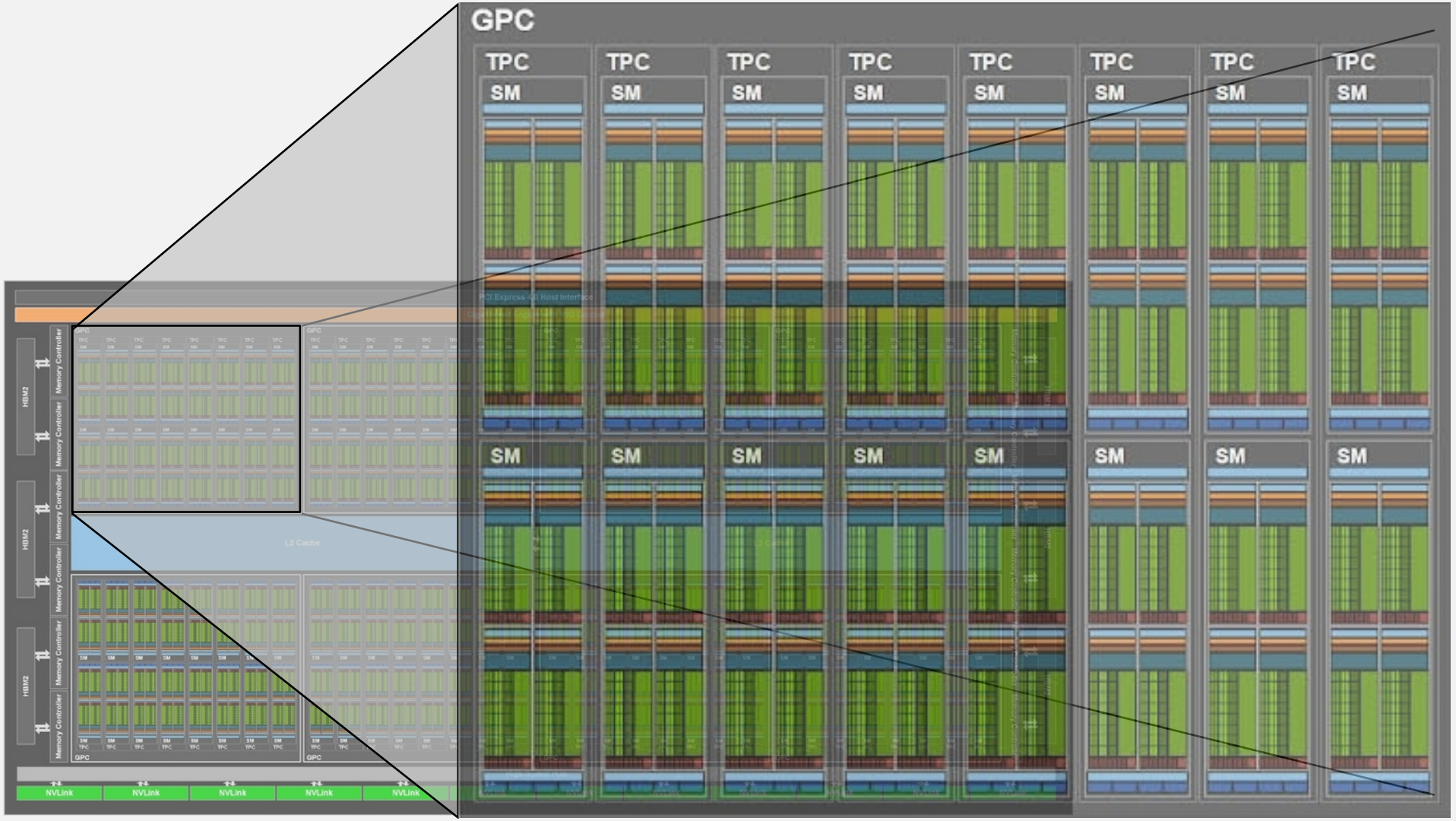
- *Multi-Instance GPU* (MIG). Permite virtualizar y particionar la GPU. Se puede compartir la GPU usando cada usuario una fracción estanca de ella o usar simultáneamente varias GPUs (muy útil para proveer servicios en la nube, QoS, etc. *Elastic Computing*). CUDA 11 lo soporta.
- La cache L1 combina en un solo bloque de memoria las funcionalidades de *data cache* y *shared memory*, mejorando el rendimiento. L1 de 192 KB por SM (se verá muy pronto qué es un *SM*).
- 40 GB de HBM2 DRAM de memoria principal (*global/device memory*) y 40 MB de caché L2.
- Nueva instrucción de copia asíncrona. Carga directamente los datos de la memoria global a la memoria compartida de los SM. Permite copiar a la vez que el SM está haciendo otro tipo de cálculos.
- Incluye *cores* FP32 y INT32 separados (como las anteriores). FP32 y INT32 pueden operar simultáneamente. Útil para aplicaciones con *inner loops* con operaciones de cálculo de direcciones (INT32) y de cálculos (FP32): cada *pipelined loop* direcciona y carga nuevos datos mientras procesa los actuales.
- Soporte optimizado para la nueva característica *Task Graph Acceleration* de CUDA 11.

ARQUITECTURA GA100

Similar a GV100 (Volta) y GP100 (Pascal)

ARQUITECTURA GA100

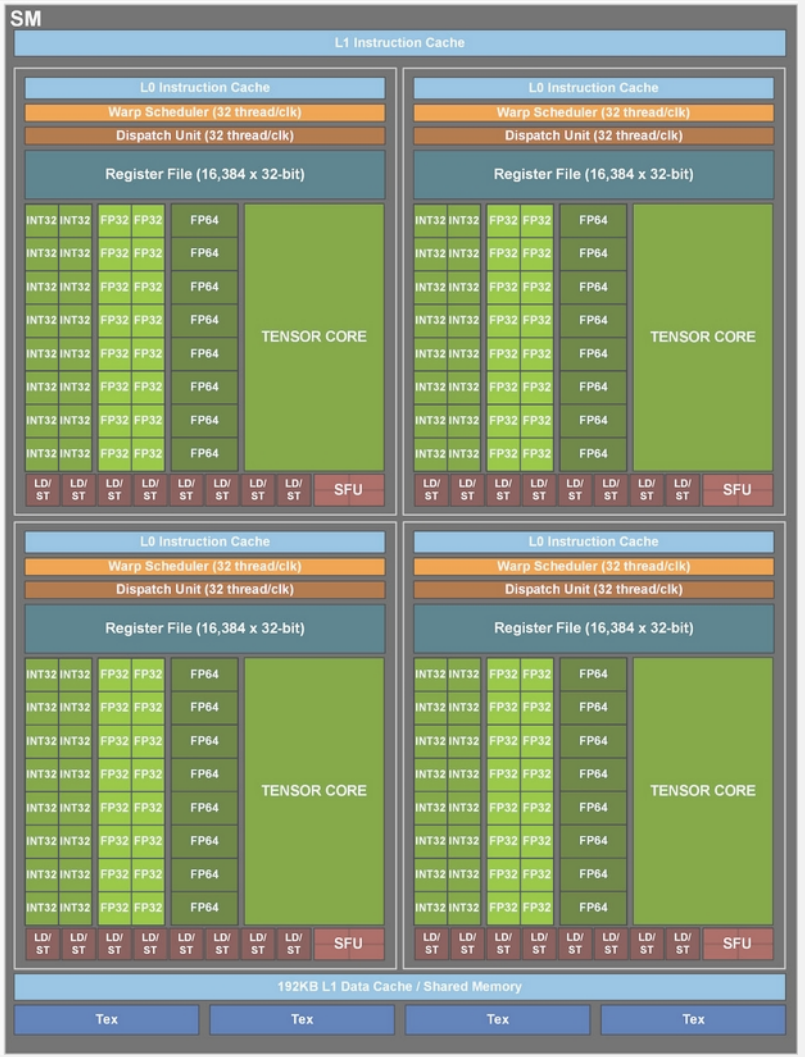
Similar a GV100 (Volta) y GP100 (Pascal)



ARQUITECTURA GA100

Full board

- 8 GPCs (*Graphics Processing Cluster*)
- Cada GPC tiene 8 TPCs (*Texture Processing Cluster*)
- Cada TPC tiene 2 SMs (*Streaming Multiprocessors*)
- Cada SM
  - 64 INT CUDA Cores
  - 64 FP32 CUDA Cores y 32 FP64 CUDA Cores
  - 4 Third-generation Tensor Cores



En resumen, una *full board* tiene

- 128 SMs
- 8192 INT, 8192 FP32 y 4096 FP64 CUDA Cores
- 512 Third-generation Tensor Cores
- 6 HBM2 stacks, 12 512-bit Memory Controllers

No todas las tarjetas tienen todos los elementos ni en la misma cantidad.

ARQUITECTURA GA100: *CUDA CORE* (CC)

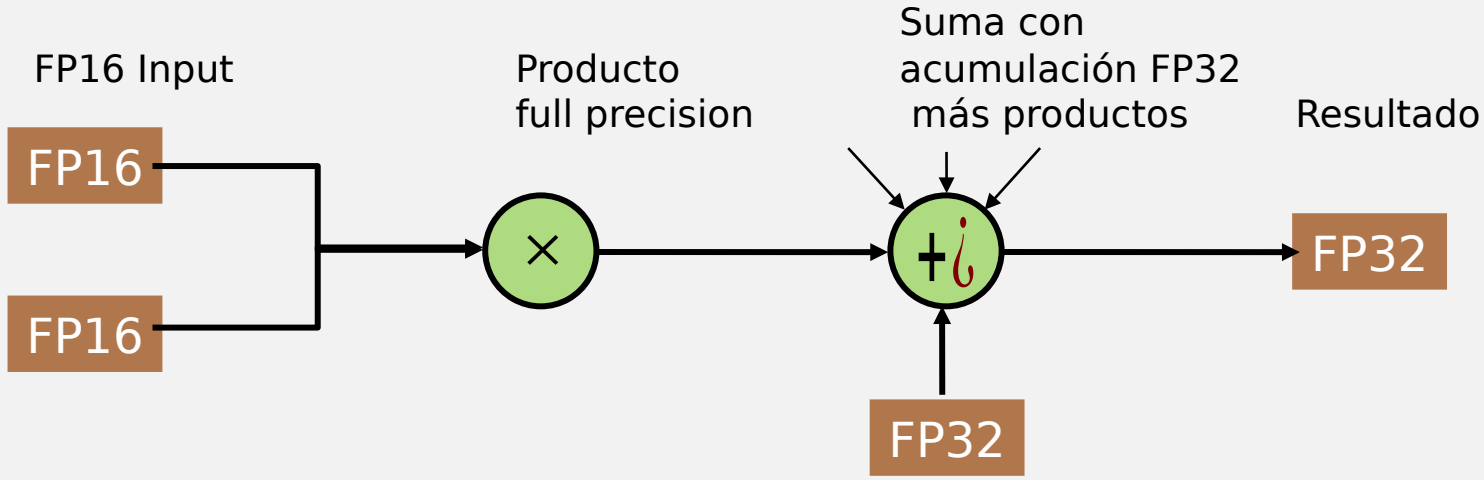
ARQUITECTURA GA100: *THIRD-GENERATION TENSOR CORE* (TTC)

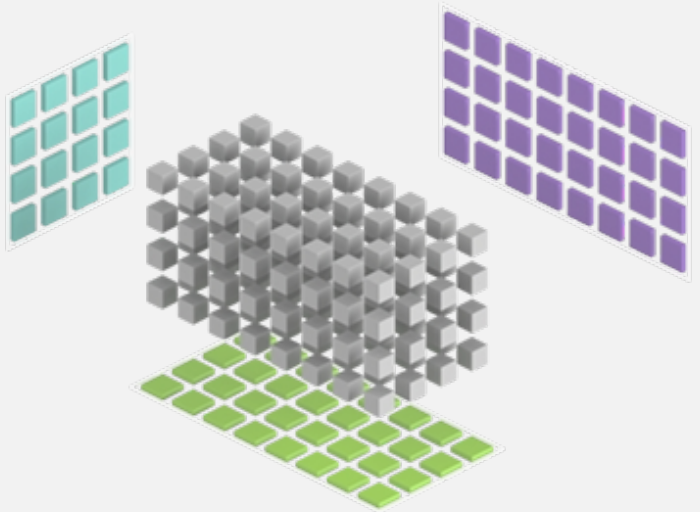
- Un TTC multiplica y acumula matrices (MMA) en un solo ciclo de reloj, muy útil en DL (*Deep Learning*).
- Algunos datos y nuevas características:
  - Cada TTC puede realizar 256 FMAs FP16 (*Fused Multiply-Add operations*) más acumulación FP32 por ciclo.
  - Esto permite calcular D=AB+C de dimensiones m=k=8 y n=4 en un ciclo de reloj.
  - Cada SM tiene 4 TTC. Un SM puede hacer 1024 FMAs por ciclo (o 2048 FP16). Como una *full board* tiene 128 SM podrá computar con sus TTCs 528.288 FMAs por ciclo.
  - Una A100 (no es full board) obtiene 312 TFLOP con TTC FP16 (78 con CC FP16), 156 con TTC FP32 (19.5 con CC FP32) y 19.5 con TTC FP64 (9.7 con CC FP34).
  - También puede usar otros tipos (BF16, TF32, FP64, INT8, INT4, etc.) operando a la misma u otra velocidad. Por ejemplo TTC INT8 llegan a los 624 TOP, 1248 TOP con INT4 o 4992 TOP en binario.
  - *Sparcity*. La nueva instrucción *Sparse MMA* permite explotar la dispersión en las estructuras de las redes neuronales en DL, obviando (no operando) con aquellos valores que son *nulos*.

ARQUITECTURA GA100: *THIRD-GENERATION TENSOR CORE* (TTC)

$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 / FP32 — FP16 — FP16 — FP16 / FP32





ARQUITECTURA: *RAY-TRACING CORE* (RTC)

- Para el trazado de rayos de luz en tiempo real en gráficos. Se basa en los objetos que forman parte del gráfico (de la escena).

RTC es la unidad encargada de recorrer el árbol BVH (*Bounding Volume Hierarchy*) de manera continuada. Recorre todos los posibles caminos para dar respuesta a cada uno de ellos. Todo en tiempo real.

Actualmente solo en la serie GeForce.

GeForce no tiene FP64 (hasta ahora).

Por ahora no hay GeForce GA100.

Demo: https://www.youtube.com/watch?v=NgcYLIvlp_k

ARQUITECTURA: COMÚN A TODAS LAS ACTUALES

- Las instrucciones son ejecutadas en orden (no saltos predictivos, no ejecución especulativa), por ahora.
- Los hilos se agrupan en bloques (*thread block*) de topología 1D, 2D ó 3D.
- Los hilos de un bloque se ejecutan concurrentemente en un SM.
- Múltiples bloques pueden ejecutarse concurrentemente en un SM.
- Cuando un bloque termina se lanza la ejecución de nuevos bloques.

- Cuando a un SM llega un bloque para ejecutar lo divide en ***wraps*** que son planificados (*warp scheduler*) para su ejecución.

- Desde VOLTA cada hilo tiene su propio contador de programa, permitiendo una planificación (*scheduling*) independiente de los hilos. Igual concurrencia entre los hilos, independiente del *warp*.