Universidad Nacional de la Patagonia San Juan Bosco
Facultad de Ingeniería
Arquitectura de Computadoras

Introducción al hardware para Deep Learning

CPU, GPU, TPU & FPGA

Alarcón Karina Soledad

Febrero 2024

RESUMEN

El crecimiento acelerado de la inteligencia artificial ha generado un impulso significativo, haciendo crucial mantenerse al día con las innovaciones en este campo. La aplicación de estas tecnologías se extiende cada vez más a diversos sectores, donde las computadoras adquieren la capacidad de aprender de forma autónoma para posteriormente operar de manera independiente.

A pesar de los notables avances logrados en inteligencia artificial, aún existen diversas áreas por explorar, especialmente en consideración a las demandas computacionales de estos modelos. Este informe se propone como una introducción al hardware especializado en Deep Learning, una subdisciplina del Machine Learning.

En primer lugar, se proporcionará una visión detallada del Deep Learning, incluyendo su funcionamiento, requerimientos y aplicaciones. Seguidamente, se explorarán las unidades de procesamiento a nivel de hardware utilizadas en este ámbito, abordando el CPU, GPU, TPU y FPGA mediante sus definiciones, una introducción a sus arquitecturas y a sus respectivos ámbitos de aplicación.

Finalmente, se realiza la clasificación de hardware de las unidades de procesamiento según la Taxonomia de Flynn, se realiza una introducción a CUDA y OPENCL, y por último se presentarán tres publicaciones que comparan las unidades de procesamiento mencionadas, proporcionando una visión integral de las opciones disponibles en el panorama actual de hardware especializado en Deep Learning.

DESARROLLO

Deep Learning

El término Deep Learning (DL) “hace referencia a un subconjunto del área de Machine Learning” [24]. Realizando un análisis léxico del término, “el concepto “deep” simplemente significa que consta de varias capas sucesivas de aprendizaje. [Y] al número de capas que forman parte del modelo se les denomina “depth” (profundidad)” [6].

Mediante el uso del Deep Learning se llevan a cabo estimaciones de modelos mediante computadoras, empleando redes neuronales artificiales con múltiples capas ocultas y utilizando cientos o miles de variables [24].

“[La] estructura jerárquica de redes neuronales artificiales, [...] se construyen de una forma similar a la estructura neuronal del cerebro humano, con los nodos de neuronas conectadas como una tela de araña” [1].

En este sentido, “se podría decir que consiste en un proceso de depurar información en múltiples etapas, en los que la información pasa por sucesivos filtros para salir cada vez más depurada” [6].

Por otro lado, “el aprendizaje profundo es una técnica que, al igual que otros algoritmos de aprendizaje, enseña a los ordenadores a hacer lo que es natural para los humanos: aprender con el ejemplo” [1].

Diversos autores concuerdan en el hecho de que los comienzos del Deep Learning se remontan a 1980, estableciendo este como el año de su primera teorización. En la década posterior el desarrollo de modelos de DL sufrió una ralentización “debido a lo computacionalmente costosos que eran estos modelos para los ordenadores de aquella época” [24].

Posteriormente, su resurgir en el año 2006 [24], se debe a tres cuestiones. En primer lugar, el Deep Learning requiere millones de datos [1], por lo que uno de los motivos es el aumento de la disponibilidad de datos.
Por otro lado, el poder computacional que se fue desarrollando con el correr de los años, especialmente con el impulso de tarjetas gráficas o GPU. Y en tercer lugar, el interés que

despierta la inteligencia artificial a partir del siglo XX, lo que hace que se destinen más inversiones en estas áreas [24].

A pesar de que el Deep Learning se encuentra dentro del Machine Learning (ML), existen dos diferencias claves entre estas áreas [6].
En primer lugar, el autor plantea la discrepancia en la forma de resolver un problema concreto. En el caso de ML, la máquina se entrena a partir de unos datos iniciales (datos de entrenamiento) y especificaciones sobre las características de los datos. Por otro lado, en el caso del DL no es necesario pasarle especificaciones, con los datos iniciales puede realizar el aprendizaje, ya que en este caso lo que hace es deconstruir los datos en múltiples niveles de detalle, buscando patrones que en caso de aparecer en reiteradas ocasiones procederá a etiquetarlos como una característica importante.
Por otro lado, las limitaciones del Deep Learning sobre el Machine Learning, teniendo en cuenta que el primero se entrena a partir mayor número de datos iniciales que el ML, lo que consecuentemente implica también la necesidad de una potencia de cómputo mayor [6].

Con respecto a la utilización del Deep Learning, algunas de sus áreas de aplicación son: “Análisis Financiero; Procesado de Imágenes en el ámbito de la Medicina, Industria y Defensa; Diagnóstico Médico y Comercial; Robótica y Control; Reconocimiento y Síntesis de Voz; Clasificación de Datos provenientes de sensores; Compresión y Codificación de Información” [5].

Por otra parte, existen modelos de DL que han llegado a comportamientos casi humanos en las áreas de clasificación de imágenes, reconocimiento de voz, transcripción de textos escritos a mano y conducción autónoma. Además, el autor considera otras aplicaciones exitosas del DL como el Text to Speech, la búsqueda de respuestas en lenguaje natural, la asistencia dirigida (por ejemplo Google Now y Alexa), el perfeccionamiento de búsquedas en la web, la publicidad dirigida, y las estrategias de juegos como Ajedrez y GO [6].

Como fue mencionado anteriormente, los modelos de Deep Learning requieren una gran cantidad de datos por lo que necesitan arquitecturas computacionales que puedan cumplir con la tarea. Actualmente, al procesar datos de gran escala, el número de parámetros crece de manera exponencial, lo que resulta en una limitación en recursos, tiempo y energía de cómputo [25].

En un principio en el campo del Machine Learning

> el CPU era la única herramienta de procesamiento del ordenador para realizar entrenamiento e inferencia sobre modelos, sin embargo el principal motivo por el avance del machine learning fue el uso de nuevos elementos hardware para incrementar la velocidad de cálculo tanto en entrenamiento como en inferencia, entre ellos el uso de la unidad de procesamiento gráfica (GPU) y la unidad de procesamiento de tensores (TPU). [9]

INTRODUCCIÓN A LAS UNIDADES DE PROCESAMIENTO

A continuación, se detallarán algunos de los elementos de hardware que se utilizan actualmente en el desarrollo de modelos de Deep Learning.

CPU (Unidad Central de Proceso)

La Unidad Central de Proceso (UCP o CPU) “controla el funcionamiento del computador y lleva a cabo sus funciones de procesamiento de datos” [21].

Está compuesta por “una unidad de control y otra de procesamiento (ALU más registros). Normalmente se construye sobre una pastilla de silicio que llamaremos procesador o microprocesador (μP)“ [3].

Las máquinas tradicionales utilizan la arquitectura Von Neumann, la cual cuenta con un único elemento procesador, el cual

> realiza todos los cálculos ejecutando todas las instrucciones de la secuencia programada en el algoritmo. Cualquier CPU realiza más de cien comandos básicos, incluyendo sumas, restas, y desplazamientos entre otros. Los comandos o instrucciones se ejecutan secuencialmente y sincronizados con el reloj del sistema. [5]

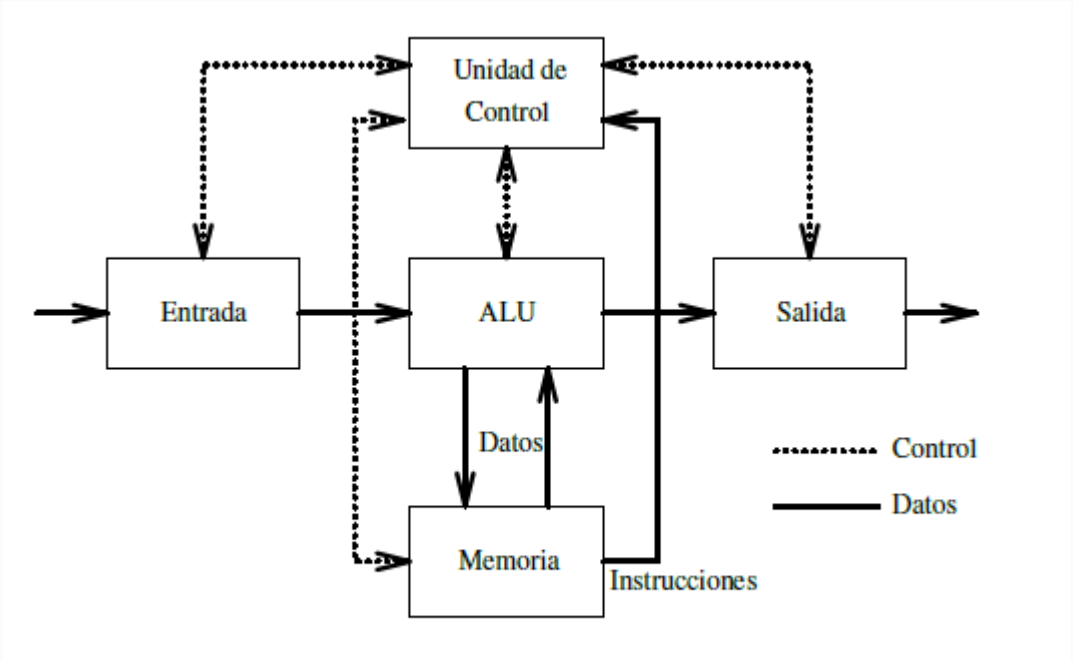

Figura 1.1: Arquitectura Von Neumann

Tal como se muestra en la Figura 1.1, la Arquitectura Von Neumann cuenta con los siguientes componentes [3]:

a) Entrada.- Unidad que transmite instrucciones y datos del exterior a la memoria (pasando por la ALU).

b) Memoria.- Unidad que almacena instrucciones y datos, así como los resultados parciales y finales de los programas.

c) A.L.U.- Unidad que realiza las operaciones aritmético-lógicas (suma, multiplicación, OR, con datos de memoria). Tiene registros para almacenar operandos: RALU.

d) Unidad de Control.- Interpreta las instrucciones y coordina el resto del sistema.

e) Salida.- Transmite los resultados al exterior.

Lo que distingue a esta arquitectura es:

a) Una sola memoria direccionada secuencialmente. La memoria tiene una estructura lineal (vector de palabras direccionables). Es el cuello de botella del sistema.

b) No hay distinción explícita entre instrucciones y datos. Los datos no tienen ningún significado explícito.

c) El procesamiento es totalmente secuencial. Las fases de ejecución de las instrucciones son secuenciales (sólo hay una unidad de control).
d) Hay instrucciones que permiten la ruptura de secuencia (es decir, se permiten los saltos en el programa). [3]

Por otra parte, el CPU cuenta con cuatro componentes estructurales principales [21]:

- Unidad de control: controla el funcionamiento de la CPU y por tanto del computador.
- Unidad aritmético-lógica (ALU, Arithmetic Logic Unit): lleva a cabo las funciones de procesamiento de datos del computador.
- Registros: proporcionan almacenamiento interno a la CPU. Contienen datos en los que la CPU está trabajando en ese momento y facilitan un acceso rápido a los datos. Las CPU tienen varios tipos de registros, como:
    - ❖ Registros de uso general que contienen datos operativos
    - ❖ Registros de instrucciones que contienen la instrucción actual que se está procesando
    - ❖ Un contador de programas que contiene la dirección de memoria de la siguiente instrucción que se va a recuperar

  Los registros proporcionan tiempos de acceso más rápidos que otros niveles de memoria, como la RAM o la memoria caché [4]
- Interconexiones CPU: son mecanismos que proporcionan comunicación entre la unidad de control, la ALU y los registros.

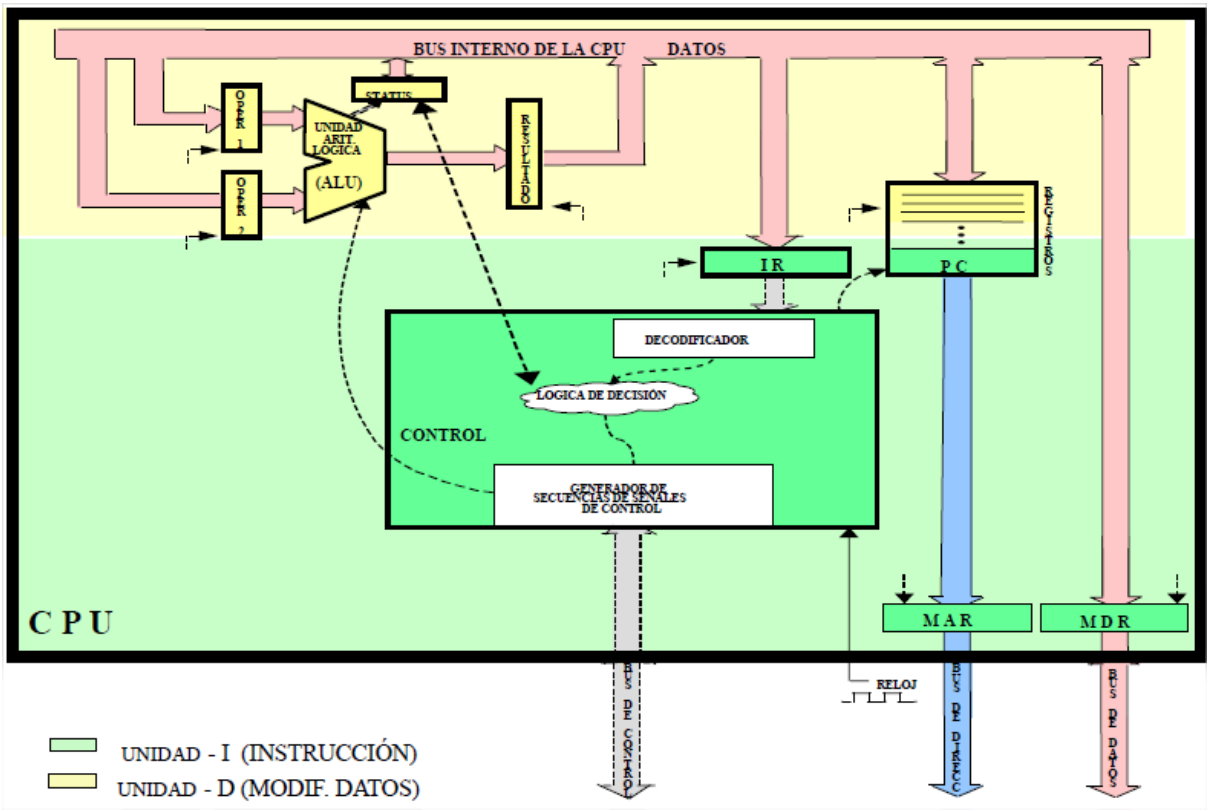

Figura 1.2: Estructura interna del CPU

La forma en que la CPU se conecta con la memoria y los dispositivos de E/S se realiza mediante buses de comunicación. Estos buses, no son más que líneas que transportan información binaria entre los diversos subsistemas. Los podemos clasificar en función del

tipo de información que llevan [3]:

- Bus de direcciones. Mediante el cual la CPU selecciona la posición de memoria o la unidad de E/S de la cual va leer información o en la cual va a escribir información. Evidentemente, si el bus tiene n lıneas, tendremos acceso a 2n posiciones distintas (a repartir entre RAM, ROM y E/S). Este bus es unidireccional.
- Bus de datos. Mediante el cual se transfiere información de (hacia) la memoria o E/S hasta (desde) el procesador. Es un bus bidireccional y normalmente determina la palabra de trabajo del procesador (si el bus tiene líneas decimos que el procesador es de n bits).

- Bus de control. Mediante el cual se transfiere información de control entre el procesador, memoria y E/S. Es un bus bidireccional.

Por otro lado, se pueden distinguir dos grupos principales de procesadores, los de 32 bits y los de 64 bits. Su principal diferencia radica en la forma en que están interconectados los diferentes componentes dentro de la computadora. Son los buses internos los que determinan la velocidad de transmisión de datos. Por un lado, un bus de 32 bits posee 32 canales para transmitir información, mientras que uno de 64 bits tiene 64 canales. Más canales significan una transmisión de datos más rápida. En consecuencia, aumentar la velocidad del bus y su ancho de banda, permite al procesador recibir más datos y realizar más operaciones simultáneamente [22].

En cuanto a la capacidad de cómputo dependiente de la cantidad de núcleos que puede poseer un CPU, cada núcleo posee como máximo 32 canales distintos de bus. Con dos núcleos, se tienen 64 canales, donde los núcleos pueden trabajar independientemente uno del otro, procesando información diferente.

Actualmente existen nuevos procesadores con cuatro núcleos, los cuales al igual que los que tienen dos núcleos, reparten entre ellos las tareas e instrucciones para su óptimo funcionamiento, permitiendo la resolución de operaciones complejas. Si bien se cuentan con más núcleos, estos funcionan con un bus de 64 bits, por lo que no existe gran diferencia con los procesadores de dos núcleos [22].

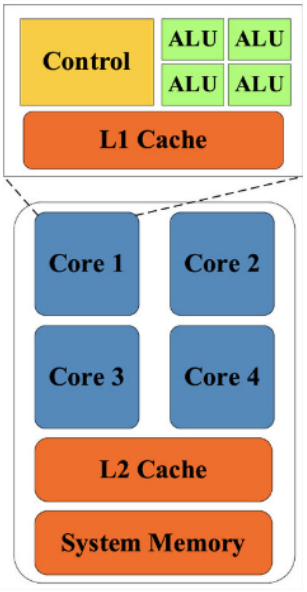

Figura 1.3: CPU con cuatro núcleos

El procesador posee las siguientes características principales [3] :

1. Longitud de palabra: Está definida por el número de bits del bus de datos, bus de direcciones, tamaño de los registros o el número de bits con el que la ALU puede operar.
2. Velocidad de proceso: Dependerá de ciertos parámetros de diseño:
    - Frecuencia de reloj
    - Tiempo de ejecución de cada instrucción (medidas en ciclos de reloj)
    - Tiempo de acceso a memoria y a Entrada/Salida (estas unidades son las que ralentizan al sistema completo)

    Por otra parte, “la velocidad que un procesador invierte en la ejecución de un programa puede obtenerse como el producto de tres factores” [3].
    En primer lugar, el número de instrucciones máquina en las que se transforma el programa, el cual depende de si las instrucciones serán traducidas o interpretadas. Por otro lado está el número medio de ciclos de reloj que se necesitan para ejecutar cada instrucción máquina, relacionado con el conjunto de instrucciones del procesado, y por último el tiempo de ciclo de reloj, relacionado con la tecnología hardware y las técnicas de integración de circuitos.
3. Capacidad de proceso: Se relaciona con
    - Número de instrucciones del microprocesador, distinguiendo entre CISC y RISC, utilizando la métrica MIPS (millones de instrucciones por segundo)
    - Operaciones que puede realizar la ALU
4. Gestión de memoria: Se refiere a las características que posee el procesador en relación con la gestión de memoria. Dos parámetros principales que influyen en las prestaciones de memoria:
    - Tamaño en número de palabras, el cual aumenta la funcionalidad del sistema al permitir ejecutar aplicaciones más complejas
    - Latencia (expresada en nanosegundos), ligada a la rapidez de ejecución de los programas.
5. Manejo de interrupciones y capacidad de interfaz: Se refiere a la capacidad del procesador para relacionarse con otros dispositivos externos.

Funcionamiento interno

Con respecto a su funcionamiento interno, el trabajo de los microprocesadores consiste en procesar instrucciones predefinidas, las cuales sirven de intermediarias entre el software y el hardware o entre los programas y los procesadores [18].
La CPU utiliza una señal de reloj para coordinar sus operaciones internas, generando pulsos constantes a una frecuencia específica. Estos ciclos de reloj sincronizan las acciones de la CPU, cuya velocidad se mide en hercios (Hz) y determina la cantidad de instrucciones que puede ejecutar por segundo. Las CPU modernas ajustan sus velocidades de reloj según la carga de trabajo para equilibrar rendimiento y consumo de energía [4].

Por otra parte está el ciclo de instrucción, que es el tiempo necesario para leer y ejecutar una instrucción, con un tiempo mayor para instrucciones más lentas y menor para instrucciones más rápidas, y que normalmente, se mide en ciclos de reloj [3].

Cada ciclo de instrucción implica el procesado de las instrucciones contenidas en el programa que se quiere ejecutar (donde el Sistema Operativo se encarga de cargar el programa y los datos en memoria, además de hacer que el contador de programa apunte a la primera instrucción del programa) y se lleva a cabo de la siguiente manera [3]:

- Búsqueda:
  El procesador “busca” en la posición de memoria apuntada por el registro PC una instrucción que carga en el registro RI. Estas instrucciones están en binario, y representan datos u operaciones específicas para la CPU.
- Decodificación:
  El Código de Operación de la instrucción se analiza para determinar qué función debe realizar la instrucción. También en esta fase se incrementa el PC para que apunte a la siguiente instrucción a ejecutar.
    - ❖ Cálculo de la dirección efectiva: En caso de que la instrucción necesite operandos almacenados en memoria, se determina la dirección en la

que se encuentran dichos operadores: la dirección efectiva o EA (effective address).

- ❖ Búsqueda de operandos. Acceso a memoria para leer los operandos de instrucción.

- Ejecución:
  Realización de la función especificada por la instrucción. Una vez concluida esta fase volvemos a la fase de búsqueda de la siguiente instrucción.

Por otra parte, la CPU también tiene la capacidad de manejar instrucciones de control de flujo, como saltos y ramificaciones, además de gestionar interrupciones, que son señales generadas por dispositivos externos o eventos que demandan atención inmediata. Cuando ocurre una interrupción, la CPU interrumpe temporalmente la tarea en curso, guarda su estado y se traslada a una rutina de servicio de interrupciones. Una vez que ha procesado la interrupción, la CPU retoma la tarea que estaba realizando anteriormente [4].

Con respecto a la utilización de los CPUs, es posible diferenciar tres tipos de computadoras, las cuales hacen uso de estos para su funcionamiento [7]. En primer lugar están los servidores, los cuales son computadoras de alto rendimiento que brindan servicios a muchos usuarios finales o clientes. En cuanto a su hardware, este se optimiza para obtener un tiempo de respuesta rápida a múltiples solicitudes de red. Para esto, cuentan con múltiples CPUs, grandes cantidades de memoria RAM y varias unidades de discos de alta capacidad que permiten encontrar información de manera muy rápida. Los servicios comunes de los servidores son almacenamiento de archivos, almacenamiento de correo electrónico, páginas Web, uso compartido de impresoras, entre otros.

Por otro lado están las computadoras de escritorio, las cuales se usan por lo general para ejecutar aplicaciones como operadores de texto, hojas de cálculo y aplicaciones de red como correos electrónicos y navegación web.

Por último están las estaciones de trabajo, las cuales son computadoras comerciales más potentes que las anteriores, diseñadas para aplicaciones especializadas como programas de ingeniería, como por ejemplo el diseño asistido

por computadora. Estas computadoras se utilizan para diseños de gráficos 3D, animaciones de video, simulaciones de realidad virtual, administración de telecomunicaciones o equipos médicos. A nivel hardware, suelen tener varias CPUs, memorias RAM y unidades de disco duro de gran capacidad.

GPU (Unidad de Procesamiento Gráfico)

El GPU es un procesador paralelo diseñado con alta capacidad de computación. “GPUs are evolved from fixed pipelining to programmable parallel processor over a period of time. Graphic processing was accelerated by implementing parallel instruction handling by vector architectures, and instruction level parallelism (ILP)” [23].

“Desde el 2003 y gracias en gran parte a la industria de los videojuegos, las tarjetas gráficas han ido evolucionando hasta convertirse en auténticos procesadores” [11]. Por otra parte, el GPU, microprocesador de dichas tarjetas, lidera en rendimiento en lo que se refiere a punto flotante.

Con respecto al objetivo de desarrollo de la Unidad de Procesamiento Grafico, se plantea que “were started with the design goals of: to cover the latency of memory bandwidth so that wide range of data could be fetched in a high frequency, to support parallel graphics shader programming models, to simplify the parallel programming models” [11].

La arquitectura típica de una GPU actual está compuesta por un número escalable de multiprocesadores paralelos(o SMs), los cuales se agrupan de tres y tres y se denomina Cluster de Procesado de Hilos (o TCP) [10].

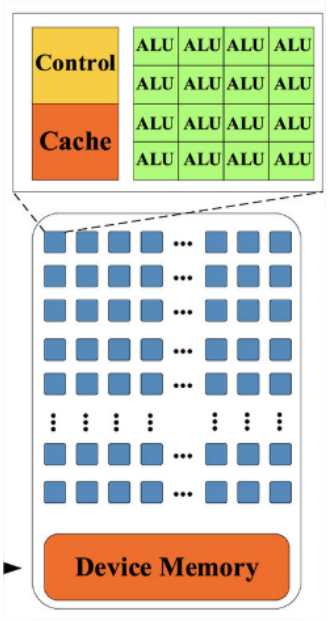

Figura 1.4: Esquema de núcleos del GPU

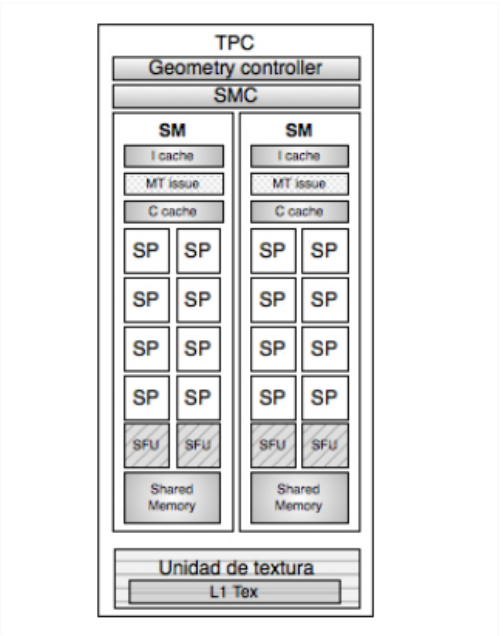

Figura 1.5: Esquema TCP

Cada SM está formado por ocho Streaming or Scalar Processors (SP), los cuales comparten la lógica de control y la caché de instrucciones. Además de los SP, un SM posee 2 SFUs (Special Function Units), una pequeña caché de instrucciones, una unidad MT (MT issues), responsables de enviar instrucciones a todos los SP y SFUs en el grupo, una caché de sólo lectura de datos y una memoria shared o compartida, generalmente de 16KB. Las SFU llevan a cabo las funciones de punto flotante tal como la raíz cuadrada o funciones trascendentales como seno, coseno,

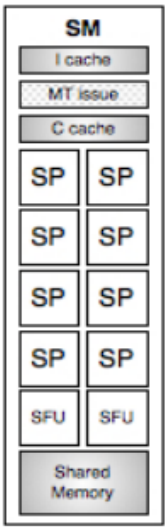

Figura 1.6: Esquema SM

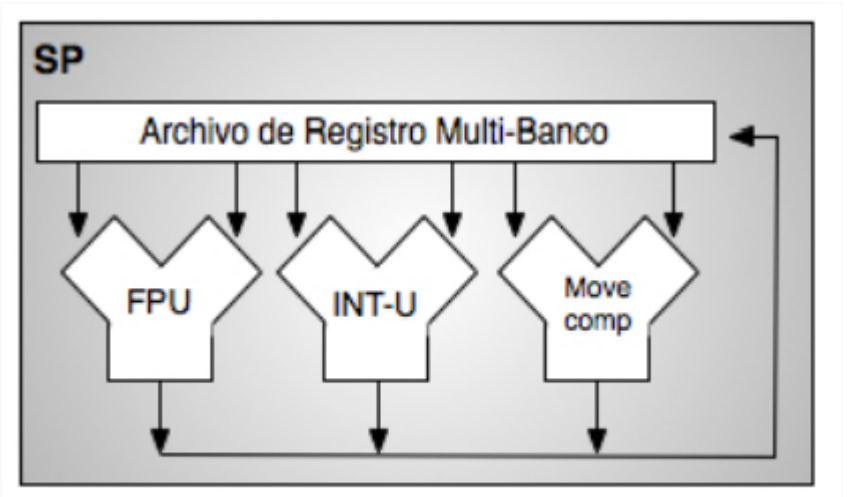

Figura 1.7: Esquema SP

entre otras. Cada SFU ejecuta una instrucción por thread por ciclo de reloj. Si las SFU están ocupadas, la unidad de planificación ejecuta otras sentencias para evitar el tiempo ocioso. [14]
Respecto a los SP, son responsables de las operaciones matemáticas o de direccionamiento de datos en memoria y posterior transferencia de los mismos, trabajan sobre datos escalares. Cada uno cuenta con una unidad MAD (Multiply-Add) y una unidad de multiplicación adicional. En la figura 2.8 se muestra la estructura típica de un SP. [14]

A nivel externo, a nivel global, la tarjeta gráfica cuenta hasta con 4 gigabytes (GB) de memoria DRAM off-chip. En las aplicaciones gráficas, esta memoria almacena imágenes de vídeo, texturas para renderizados 3D, etc. [...] Actualmente la comunicación de la GPU con la CPU se realiza a través de un bus PCI-Express. Dicho bus consta de dos vías (una de envío y otra de recepción). [11]
Con respecto a las instrucciones “los ocho procesadores de un multiprocesador comparten la unidad de búsqueda y lanzamiento de instrucciones de forma que se ejecuta la misma instrucción al mismo tiempo en los ocho procesadores” [11].
Tradicionalmente, el funcionamiento de las GPU es sintetizado como un pipeline de procesamiento formado por etapas con una función especializada cada una. Estas son ejecutadas en paralelo y según un orden preestablecido, cada etapa recibe como entrada la salida de la etapa anterior y su salida es la entrada a la siguiente etapa. [14]
La CPU, mediante aplicaciones, API y drivers, utilizará comandos u objetos a través de una API gráfica y los enviará a la GPU para su procesamiento [14].
La aplicación envía a la GPU una secuencia de vértices, agrupados en lo que se denominan primitivas geométricas: polígonos, líneas y puntos, las cuales son tratadas secuencialmente a través de cuatro etapas bien diferenciadas. [14]

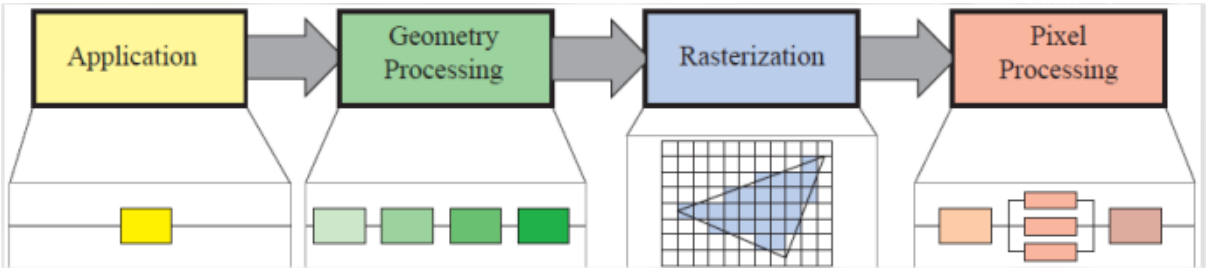

Figura 1.8: Etapas del pipeline gráfico de la GPU

La tarea/s a realizar por cada etapa es/son:
Primera etapa: Transformación de Vértices
Es la fase inicial del pipeline gráfico, donde se realizan operaciones matemáticas en cada vértice suministrado por la aplicación. Esto incluye transformación de posición, generación de coordenadas texturales y asignación de color.

Segunda etapa: Ensamblado de Primitivas
Los vértices transformados en la etapa anterior se agrupan en primitivas geométricas (triángulos, líneas o puntos) basándose en la información recibida. Posteriormente, estos puntos son sometidos a rasterización.

Tercera etapa: Rasterización
La rasterización determina el conjunto de píxeles “cubiertos” por una primitiva, generando localizaciones de píxeles y fragmentos. Un fragmento contiene información de color, así como uno o más conjuntos de coordenadas de textura. El fragmento representa un "píxel en potencia" que en caso de pasar exitosamente las etapas siguientes del pipeline, se actualiza la información del píxel resultante.
Después de la rasterización, los fragmentos se someten a operaciones de interpolación, cálculos matemáticos y de textura para determinar el color final de cada fragmento. En esta etapa, además de establecer el color final, es posible descartar fragmentos para evitar su actualización en memoria. Por lo tanto, esta fase emite uno o ningún fragmento actualizado para cada fragmento de entrada.

Cuarta Etapa: Operaciones Raster,
Cada fragmento se somete a pruebas gráficas en esta etapa para determinar los valores del píxel en memoria. Si alguna prueba falla, el píxel se descarta y no se guarda en memoria. En caso contrario, se realiza la escritura en el framebuffer, representando el resultado final del proceso gráfico. El pipeline gráfico convierte la representación de una escena 3D en una imagen 2D para proyección en pantalla.

La GPU transmite el fotograma generado a través de su salida, conectada a la pantalla, lo que permite visualizar la imagen. La frecuencia de actualización, medida en FPS (Frames Per Second), depende de la capacidad tanto de la GPU como de la

pantalla. La velocidad de actualización de los fotogramas influye en la fluidez y suavidad de la animación o contenido visual en pantalla.

A nivel de hardware, por ejemplo en la arquitectura G80 las operaciones se realizan de la siguiente manera:

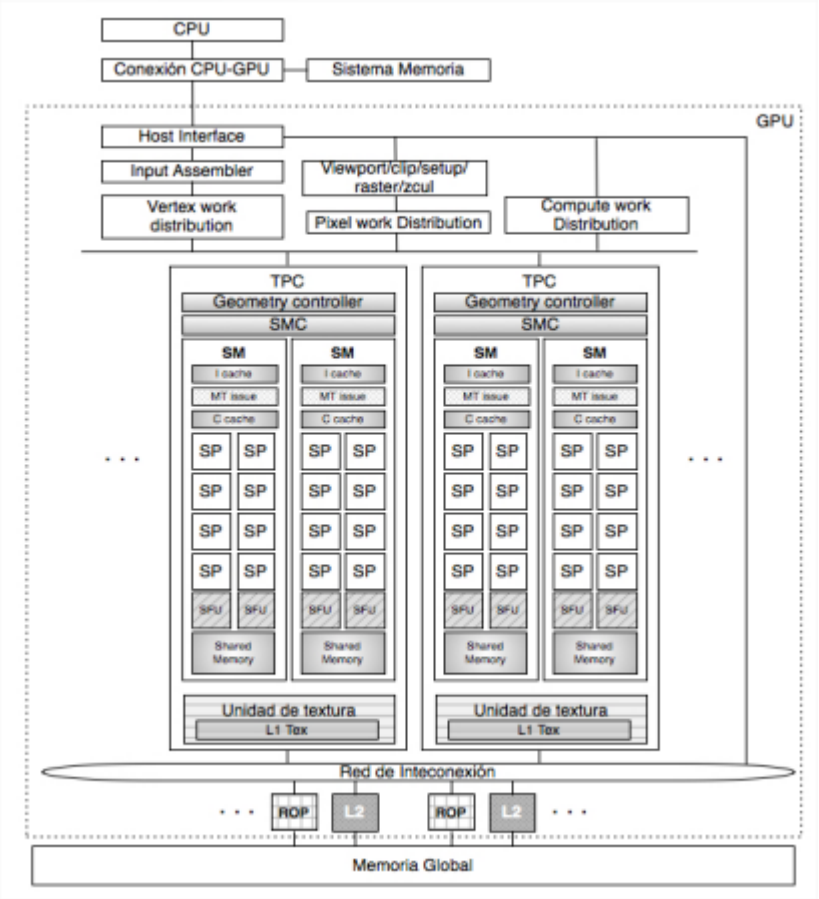

Figura 1.9: arquitectura de modelo G80

- El trabajo comienza desde la CPU, quien le indica el trabajo a la GPU a través del bus PCI-Express.
- En primer lugar los datos recibidos desde el host son preprocesados en hardware específico a fin de organizarlos para aprovechar la máxima capacidad de cálculo del sistema y evitar la existencia de unidades ociosas.
- Una vez hecha esta organización, se transfiere el control de la ejecución a un controlador global de threads, quien decide qué thread se ejecutará en cada instante y dónde.
- Además existe un planificador de threads para determinar su ejecución dentro de las unidades de procesamiento.

- El Input Assembler recolecta el trabajo de vertex que ingresa y lo pasa al distribuidor de trabajo de vertex (Vertex work distribution) quien distribuye los paquetes de trabajo a los TCP en el SPA.
- Los TCP ejecutan programas shader de vertex y programas shader de geometría. Los datos de salida son escritos en buffer on-chip, estos buffer luego pasan sus resultados a la unidad responsable de la rasterización (viewport/ clip/ setup/ raster/ zcull block).
- La unidad de distribución de trabajo de pixel distribuye fragmentos de pixeles a los TPC apropiados para el procesamiento.
- Los fragmentos de pixeles sombreados son enviados a través de la red de interconexión para el procesamiento del color y la profundidad a las ROP.
- La unidad de distribución de trabajo de cómputo (Compute work distribution) envía los threads de cómputo general a los distintos TPC del SPA para su ejecución.
- Todas las unidades que integran la arquitectura de la G80, incluido los múltiples relojes que las rigen, proveen independencia y optimización de la performance.

Por otra parte se encuentra el uso de GPU para tareas de propósito general, donde los programadores enfrentaron desafíos significativos al adaptar aplicaciones no relacionadas con gráficos para su programación a través de APIs gráficas. Debían seguir el pipeline gráfico, sin acceso directo a estados internos sin pasar por las etapas anteriores. Para abordar este problema, se propusieron varios entornos de programación que simplifican la tarea de programar aplicaciones no gráficas en GPU, entre ellos, el CUDA de NVIDIA

Las aplicaciones de propósito general para GPU se estructuran de la siguiente manera:

1) El programador define el dominio de la computación como una estructura grid de threads.

2) Un programa de propósito general es ejecutado por cada thread en paralelo sobre distintos datos según el modelo de programación es SPMD.

3) Cada thread realiza cálculos mediante una combinación de operaciones matemáticas y de accesos de escritura y lectura en la memoria global. La diferencia con las dos formas de programación anteriores está en que el mismo buffer puede

ser usado tanto para leer como para escribir, por ejemplo se pueden programar algoritmos in-place.
4) Los resultados de la operación que se guardan en la memoria global pueden ser usados por otras computaciones de la misma aplicación en la GPU [14].
El modelo de programación propuesto se destaca por varias razones. Permite al hardware aprovechar al máximo el paralelismo de datos de la aplicación al especificar explícitamente dicho paralelismo en el programa. Ofrece un equilibrio entre generalidad en la programación y restricciones, como el modelo SPMD y la gestión de la comunicación de datos entre unidades de computación y kernels. Elimina la complejidad que enfrentan los programadores de las primeras GPU, quienes debían programar utilizando la interfaz gráfica. Los programas actuales se expresan en lenguajes familiares, generalmente extensiones de lenguajes secuenciales como C, y son más simples y fáciles de construir y depurar, con herramientas que facilitan estas tareas.

Con respecto a las áreas en las que se utilizan GPUs, son utilizadas para “data science, analytics, bioinformatics, life sciences, machine learning, imaging and computer visions and in media and entertainment and more” [23].

FPGA (arreglo de compuertas programables en el campo)

“Los dispositivos Field Programmable Gate Arrays, en español Arreglos de Compuertas Programable en el Campo, tal como su nombre lo indica son un arreglo (arrays) matricial de bloques lógicos (gates) programables (programmable) en cualquier espacio físico (field)” [19].

Las FPGA son circuitos integrados reconfigurables compuestos de interconexiones programables que combinan bloques lógicos programables, de memoria embebida y de procesamiento de señales digitales, entre otros. Dicho en palabras más simples, una FPGA es ‘hardware programable’” [18]

Por otra parte:

> Es un dispositivo semiconductor que contiene componentes lógicos programables e interconexiones programables entre ellos. Los componentes lógicos programables pueden ser programados para duplicar la funcionalidad

> de puertas lógicas básicas tales como AND, OR, XOR, NOT o funciones combinacionales más complejas tales como decodificadores o simples funciones matemáticas. [13]

Con respecto al surgimiento:

> Fue en 1985 que Xilinx Incorporated (www.xilinx.com) introdujo la idea de combinar el control del diseñador y el tiempo de desarrollo (time to Market) de los PLDs con la densidad y bajo costo de arreglos de compuertas. Rápidamente la idea tomó vuelo, los FPGAs se empezaron a fabricar [19].

“Las FPGA son generalmente más lentas que sus contrapartes, los circuitos integrados de aplicaciones específicas (ASIC por sus siglas en inglés), no pueden soportar diseños muy complejos, y consumen más energía” [13].
Por otra parte, comenta que a pesar de sus desventajas también cuenta con ventajas como:

> La reducción del tiempo para la salida al mercado de productos, la habilidad para ser reprogramadas después de haber salido al mercado a fin de corregir posibles errores, y reduce los costos de ingeniería tales como investigación, diseño y prueba de un nuevo producto. (p.22)

Actualmente, no existe una arquitecta estándar de FPGA, sino que estas cambian de un fabricante a otro. Sin embargo,

> a pesar de tratar de diferenciarse uno del otro, en realidad los FPGAs de los diferentes fabricantes tienen muchos componentes en común, tales como bloques lógicos programables, bloques de memoria de doble puerto (dual port), bloques para ejecución de MACs (Multiplicador– Acumulador o bloques DSPs), bloques de control de reloj (generación de frecuencias a partir de una frecuencia base, corrimiento de fase), bloques de entrada/salida, etc.” [19].

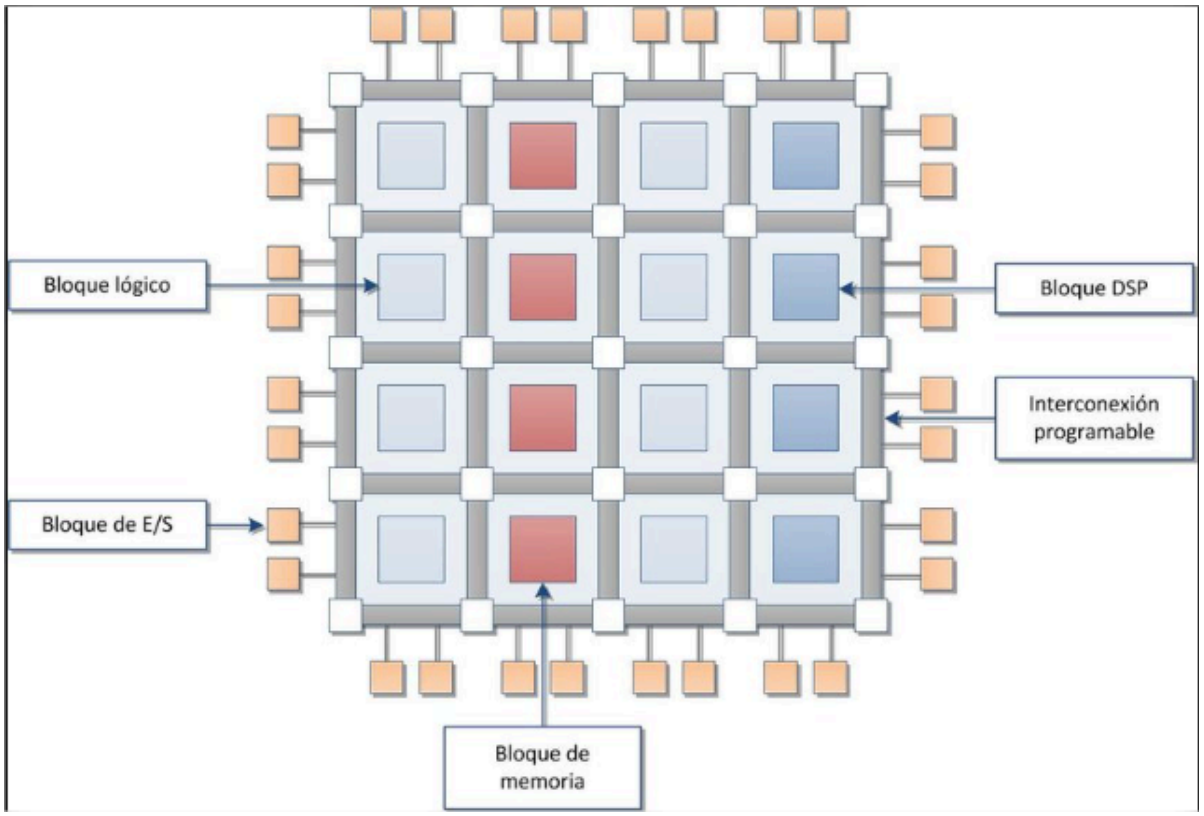

Figura 1.10: Esquema de bloques de FPGA

El desarrollo integral de los FPGA [19], se puede dividir en cuatro pasos para mayor comprensión:

1) Desarrollo del Sistema Digital:
    - “El diseñador desarrolla su sistema digital usando herramientas tipo EDA (Electronics Design Automation), sean dibujos esquemáticos o lenguaje de descripción de hardware (como VHDL), para poder plasmar el sistema en lógica digital”
    - Posteriormente, se simula el sistema digital para verificar su funcionalidad de manera satisfactoria.
2) Configuración del FPGA:
    - Utilización de “herramientas específicas del vendedor del FPGA para crear un archivo de configuración del FPGA”
    - Este archivo, “describe todas las conexiones, interconexiones y lógica que necesita ser programada dentro del FPGA para poder implementar el sistema digital desarrollado”.
3) Descarga y Verificación:

- “A través de un cable USB se conecta el FPGA o el circuito impreso en cual está soldado el FPGA, a una PC”
- “usando el software de configuración del FPGA se descarga el archivo de configuración”.

4) Auto-configuración y Almacenamiento:
   - “Una vez comprobado el correcto funcionamiento del sistema en el FPGA se graba el archivo de configuración en una memoria no volátil”
   - Posteriormente, “el FPGA [la] leerá y usará para auto-configurarse cada vez que se aplica la tensión de alimentación al FPGA o cada vez que se desee re-configurar el FPGA”.

Una vez que el FPGA está programado, ejecuta las funciones definidas por el diseño.

Las áreas de aplicación de las FPGA

> incluyen a los DSP (Digital Signal Processor), radio definido por software, sistemas aeroespaciales y de defensa, prototipos de los ASIC, sistemas de imágenes para medicina, sistemas de visión para computadoras, reconocimiento de voz, bioinformática, emulación de hardware de computadora, y tienen un crecimiento de aplicaciones en otras áreas. [13]

Un ejemplo de uso de FPGA en arquitectura de radio definida por software, se da debido a que tradicionalmente, una radio constaba de una antena y hardware especializado para procesar señales. Actualmente, gran parte de esa funcionalidad se traslada a dispositivos electrónicos, como FPGA, simplificando el hardware a una antena y convertidores ADC y DAC. La ventaja clave radica en que la funcionalidad se define mediante software, facilitando modificaciones o actualizaciones sin necesidad de cambiar elementos de hardware.

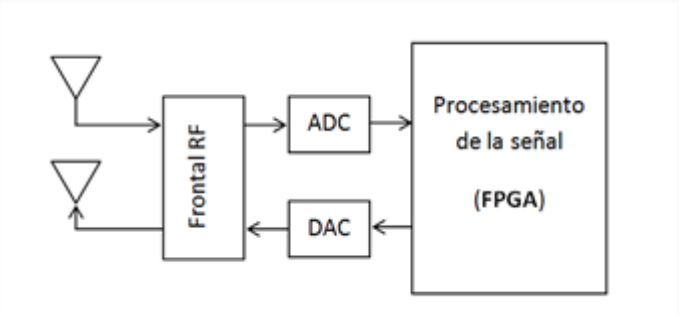

Figura 1.11: Implementación de FPGA para la arquitectura de radio definida por software

Algunos ejemplos concretos de implementación son:
En primer lugar, comenta sobre el aumento de productividad en el buscador Bing de Microsoft al integrar FPGAs en sus centros de datos. Además, destaca que el buscador chino Baidu también está utilizando FPGAs para acelerar aplicaciones de aprendizaje automático, como el reconocimiento de imágenes y de voz. Asimismo, menciona que Baidu tiene planes de emplear estos dispositivos en el desarrollo de vehículos autónomos.
Por otra parte, menciona que la Organización Europea para la Investigación Nuclear (CERN) está considerando la utilización de FPGAs para realizar cálculos de ecuaciones complejas y otras aplicaciones que actualmente se llevan a cabo utilizando CPUs y GPUs. La evaluación implica explorar la viabilidad y eficiencia de los FPGAs en comparación con las tecnologías de procesamiento convencionales.
Por último, hace referencia a empresas prestadores de servicios cloud, como Amazon y Alibaba, las cuales actualmente ofrecen la posibilidad de usar FPGAs en sus plataformas [17].

TPU (Unidad de Procesamiento Tensorial)

“Las unidades de procesamiento tensorial (TPU) son circuitos integrados específicos de la aplicación (ASIC) diseñados por Google para acelerar las cargas de trabajo de aprendizaje automático” [10].

En cuanto al objetivo de las TPU, en la página comentan que ”se diseñaron para realizar operaciones matriciales con rapidez, lo que las hace ideales para cargas de trabajo de aprendizaje automático”. Por otra parte, plantea que se pueden ejecutar cargas de trabajo de aprendizaje automático en TPU mediante frameworks como TensorFlow, Pytorch y JAX [10].

Las Cloud TPU no son apropiadas para las siguientes cargas de trabajo [10]:

- Programas de álgebra lineal que requieren ramificaciones frecuentes o contienen muchas operaciones de álgebra en elementos
- Cargas de trabajo que acceden a la memoria de manera dispersa
- Cargas de trabajo que requieren aritmética de alta precisión
- Cargas de trabajo de redes neuronales que contienen operaciones personalizadas en el bucle de entrenamiento principal

En cuanto a la arquitectura del TPU, establecen que un chip TPU contiene TensorCores, donde la cantidad de los mismos depende de la versión del chip TPU. Por otro lado, “cada TensorCore consta de una o más unidades de multiplicar matrices (MXU), una unidad vectorial y una unidad escalar” [10].

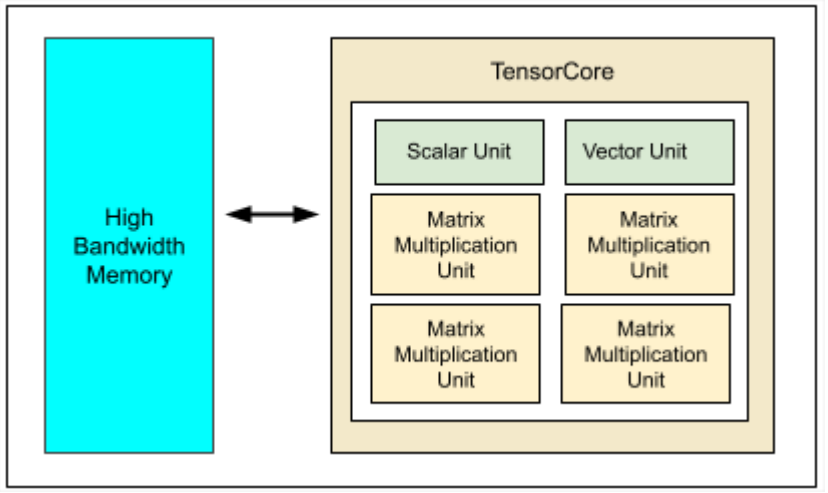

Figura 1.12: chip TPU v5e.

Con respecto a las unidades de multiplicas matriciales (MXU):

> Está compuesta por 128 x 128 acumuladores de multiplicación en un arreglo sistólico. Las MXU proporcionan la mayor parte del poder de procesamiento en un TensorCore. Cada MXU es capaz de realizar 16,000 operaciones de multiplicación y acumulación por ciclo. Todas las multiplicaciones toman entradas bfloat16, pero todas las acumulaciones se realizan en formato de número FP32 [10].

Por otro lado, “la unidad vectorial se usa para cálculos generales, como activaciones y softmax. La unidad escalar se usa para el flujo de control, calcular direcciones de memoria y otras operaciones de mantenimiento” [10].

En la TPU diseñada por Google

> Los operandos se leen una sola vez y se reutilizan en múltiples operaciones consecutivas antes de generar un resultado de salida. La circuitería está

> diseñada de forma que las ALU adyacentes se transfieren las entradas/salidas entre sí en cada pulso (ciclo de reloj). [15]

En específico, este diseño es:

> Ideal para la multiplicación de matrices y operaciones similares, en las que un mismo operando de entrada se usa en múltiples operaciones escalares simples hasta componer el resultado final. Además de los núcleos de cálculo una TPU también incorpora una importante cantidad de memoria RAM, como en el caso de las GPU, lo cual les permite almacenar localmente la información que van a procesar. [15]

La forma en que trabaja el TPU internamente es la siguiente:
El host de TPU transmite datos a una cola de entrada. La TPU carga datos de la cola de entrada y los almacena en la memoria HBM. Cuando se completa el procesamiento, la TPU carga los resultados en la cola de salida. Luego, el host de TPU lee los resultados de la cola de salida y los almacena en la memoria del host.

Para realizar las operaciones de la matriz, la TPU carga los parámetros de la memoria HBM en la unidad de multiplicación de matriz (MXU).

Luego, la TPU carga datos de la memoria HBM. A medida que se ejecuta cada multiplicación, el resultado se pasa al siguiente acumulador de multiplicación. El resultado es la suma de todos los resultados de multiplicación entre los datos y los parámetros. No se requiere acceso a la memoria durante el proceso de multiplicación de matrices [10].

En cuanto a las áreas de aplicación de los TPU, “son ideales para una variedad de casos de uso, como chatbots, generación de código, generación de contenido multimedia, voz sintética, servicios de visión, motores de recomendaciones y modelos de personalización, entre otros” [10].

CLASIFICACIONES DE HARDWARE DE CADA UNIDAD

Existen diversas clasificaciones para hardware de una computadora. Una de las más conocidas es la taxonomía de Flynn, donde se consideraron dos aspectos: en

primer lugar el flujo de instrucciones y en segundo lugar el flujo de datos. Considerando la multiplicidad de cada uno se clasifican en [14]:

- SISD (Single Instruction, Single Data): Computadoras con un solo flujo de instrucciones y un solo flujo de datos.
- SIMD (Single Instruction, Multiple Data): Computadoras con un solo flujo de instrucciones y múltiples flujos de datos, incluyendo procesadores vectoriales con varias CPUs ejecutando la misma instrucción en diferentes conjuntos de datos.
- MISD (Multiple Instruction, Single Data): Computadoras con múltiples flujos de instrucciones y un solo flujo de datos. No se han implementado arquitecturas de este tipo.
- MIMD (Multiple Instruction, Multiple Data): Computadoras con múltiples flujos de instrucciones y múltiples flujos de datos. Se dividen en Multiprocesadores (con memoria compartida) y Multicomputadoras (con memoria distribuida), y pueden ser fuertemente o débilmente acopladas según la interacción entre procesadores. La mayoría de las computadoras MIMD comerciales se consideran débilmente acopladas, y SIMD se puede ver como un caso especial de MIMD.

Las CPU se clasifican como SISD, teniendo en cuenta que es esta clasificación se encasillan las arquitecturas Von Neumann clásicas, debido a la ejecución secuencial de instrucciones.

Las GPU emplean una arquitectura SIMD, lo que les permite ejecutar la misma instrucción en múltiples puntos de datos de forma paralela.

Debido a la versatilidad de las FPGA, estas no podrían encasillarse en una única clasificación, entonces debido a la configuración que se les aplique podría ser MIMD o SIMD.

Por otra parte, las TPU podrían clasificarse como SIMD, ya que ejecutan operaciones tensoriales idénticas en múltiples datos simultáneamente.

CUDA Y OPENCL

CUDA

NVidia “ha creado un nuevo paradigma de desarrollo sobre GPU: CUDA [...] que es un modelo de programación y una arquitectura de cálculo” [16].

“CUDA son las siglas de (Compute Unified Device Architecture) que hace referencia tanto a un compilador como a un conjunto de herramientas de desarrollo creadas por NVIDIA que permiten a los programadores usar una variante del lenguaje de programación C para codificar algoritmos en GPUs de NVIDIA”.

Los autores exponen que CUDA

> aprovecha el procesamiento paralelo y el poder del hardware de la GPU (que solo se pensaba que era para renderización) para realizar operaciones de propósito general, con mayor eficiencia y pasar del tradicional procesamiento central en el CPU a una nueva ventana informática, como es el coprocesamiento. (p.40)

El coprocesamiento, implica que las operaciones ya no se centran solamente en la CPU, sino que se reparten entre la CPU y la GPU.

El procesamiento CUDA se realiza ejecutando los siguientes pasos:

1. Se copian los datos de la memoria principal a la memoria de la GPU.
2. La CPU encarga el proceso a la GPU.
3. La GPU lo ejecuta en paralelo en cada núcleo.
4. Se copia el resultado de la memoria de la GPU a la memoria principal. (NN, n.d.)

OPENCL

OpenCL es la abreviatura de Open C omputing L angguage. OpenCL es un marco para la programación paralela en plataformas heterogéneas, llamadas dispositivos informáticos , que van desde CPUs a través de GPUs a plataformas más especiales como FPGAs [20].

OpenCL tiene como objetivo principal la creación de código eficiente y portátil de manera simultánea, superando la tradicional dificultad de conciliar estas dos características. En este enfoque, cada dispositivo compatible con OpenCL posee su propia implementación específica. Durante la ejecución, interpreta el lenguaje OpenCL del programa que se le solicita ejecutar. En consecuencia, es responsabilidad de cada fabricante desarrollar controladores (drivers) para su producto, optimizando así su capacidad de procesamiento para una instrucción OpenCL específica. [8]

OpenCL facilita la asignación de tareas de un programa al dispositivo de procesamiento más adecuado en una plataforma, sin requerir cambios en el lenguaje de programación. Diseñado para aprovechar al máximo el paralelismo, permite la ejecución eficiente en múltiples núcleos y dispositivos, como GPU y CPU, al permitir la distribución selectiva de código paralelizable en la GPU y código más serial y específico en la CPU.
Su aplicación abarca desde dispositivos portátiles como teléfonos celulares o dispositivos de mano hasta PCs en entornos domésticos y servidores mainframe. Este estándar proporciona una interfaz unificada que permite a los desarrolladores aprovechar la potencia de procesamiento de diversos dispositivos, facilitando la creación de software eficiente y portátil en un amplio rango de plataformas. [8]

CUDA vs OPENCL

	CUDA	OPENCL
Desarrollador/Fabricante	Nvidia	Grupo Khronos/Apple
Dispositivos	Limitado a Nvidia	Compatible con varios fabricantes (AMD, INTEL, NVIDIA, etc)
Lenguaje de programación	Utiliza CUDA C (se pueden utilizar wrappers	C99 o OpenCL C

	para usar Python, Fortran, Julia y Java)	
Abstracción	Proporciona un alto nivel de abstracción, facilitando la escritura de código específico para GPUs NVIDIA.	Ofrece un nivel de abstracción más bajo, lo que puede proporcionar a los desarrolladores un mayor control sobre el hardware, pero también puede ser más complejo.
Host y Dispositivo	Integra la programación del host y del dispositivo en el mismo código, proporcionando una interfaz más sencilla para los desarrolladores.	Separación más clara entre la programación del host y del dispositivo, lo que puede ofrecer mayor flexibilidad, pero también puede requerir más código.
Otras herramientas	Proporciona herramientas de profiling y depuración específicas, como NVIDIA Nsight, para facilitar el análisis y la optimización del rendimiento.	Puede depender más de herramientas de terceros, y la disponibilidad y calidad de estas herramientas pueden variar entre diferentes proveedores.

INFORMES DE COMPARACIÓN ENTRE CPU, GPU, TPU y FPGA

CPU vs FPGA

Maestre Betolaza (2021) hace una comparación entre CPU y FPGA para la implementación de redes neuronales en el ámbito de la ingeniería eléctrica. Las principales comparaciones que realiza, se detallan en el siguiente cuadro.

	CPU	FPGA
Potencia de cálculo y arquitectura	Procesa secuencialmente una larga secuencia de pequeños pasos aritméticos	Trabaja en paralelo, realizando operaciones en la entrada que pueden desencadenar una cadena de reacciones distribuidas entre muchas puertas, ejecutando múltiples cálculos complejos en un solo paso.
Flexibilidad	Es flexible y puede cambiar su funcionalidad modificando el programa.	Tiene funcionalidad fija debido al cableado pre-programado. La tabla de conexiones se carga durante el arranque y permanece inalterada.
Tamaño de los números binarios	Utiliza números de tamaño fijo (16, 32 o 64 bits)	Los números son flexibles, ya que el cableado del chip funciona a nivel de bits, permitiendo tamaños no estándar (por ejemplo, 21 bits).
Programación	Generalmente es más fácil de programar con herramientas convencionales.	Difícil de programar, requiere conocimientos profundos del hardware utilizado. El proceso de diseño puede ser complejo.

Cambios de diseño y Costos	Cambios en el programa pueden alterar la funcionalidad de manera relativamente sencilla.	Una vez diseñado, no es tan fácil cambiar sus características. Algunas placas de diseño son caras.
Eficiencia Energética	Eficiencia energética estándar.	Mayor eficiencia energética, especialmente beneficioso en sistemas con baterías.
Aplicaciones Específicas	Adecuada para procesamiento secuencial y tareas generales.	Destacada en tareas en paralelo, control fino del tiempo y sincronismo. Es eficiente para implementar redes neuronales en sistemas con baterías.

Si bien el autor no destaca una preferencia por ninguna de estas dos opciones, manifestó que “Las FPGA son una plataforma ideal para la implementación de redes neuronales, pero la complejidad de su programación y el poco soporte disponible en la red, junto con el alto coste de los kits de desarrollo hacen que sea una opción menos interesante”(p. 79).

CPU vs GPU vs TPU

GAVIÑA RUEDA, (2022) realiza una investigación sobre la estimación de pose para aplicarlo a la fisioterapia, donde en la descripciones que hace entre las implementaciones de hardware se pueden destacar:

CPU

- “En las CPUs se puede llegar a lograr una alta velocidad en ejecuciones secuenciales, permitiendo cierto nivel de paralelismo de varios cores del

procesador. Por lo que si tenemos un algoritmo secuencial o paralelizable con un bajo grado (de 2 a 8 hilos), la CPU puede ser la mejor opción.” (p. 26)

GPU

- Debido a su gran cantidad de cores, el autor considera que es capaz de “hacer uso de un número mucho más elevado de hilos". Es por ello por lo que la GPU logra maximizar el througput (número de operaciones por segundo), lo que la convierte adecuada para operaciones altamente paralelizables, como las redes neuronales profundas”(p. 27). Sin embargo, considera que en cálculos de ML simples, es mejor la utilización del CPU.

TPU

- El autor comenta que al tener un mayor número de núcleos que la GPU, “especializados en realizar operaciones compuestas sobre tipos de datos simples, esto les permite incrementar el rendimiento respecto a las GPUs y las CPUs”(p.28).

El estudio realizado por el autor se basó en el análisis de videos de personas en cintas de correr, su preferencia de uso estaba en los TPU pero debido a la poca cantidad de videos que tenía para procesar, descartó la implementación de redes neuronales profundas.

FPGA vs CPU vs GPU vs ASIC

Un artículo publicado en la web de Arrow Electronics (arrow.com) plantea el siguiente cuadro, donde se establecen aplicaciones donde se puede utilizar una o la combinación de varias unidades de procesamiento. [2]

Applications	CPU	FPGA	GPU	ASIC	Comments
Vision & image processing		✓	✓	✓	FPGA may give way to ASIC in high-volume applications
AI training			✓		GPU parallelism well-suited for processing terabyte data sets in reasonable time
AI inference	✓	✓	✓	✓	Everyone wants in! FPGAs perhaps leading; high-end CPUs (e.g., Intel's Xeon) and GPUs (e.g., Nvidia's T4) address this market
High-speed Search	✓	✓	✓	✓	Microsoft's Bing uses FPGAs; Google uses TPU ASIC; CPU needed for coordination & control
Industrial motor control	(✓)	✓		✓	Many motor-control MCUs and ASICs available; FPGAs offer a quick-turn ASIC alternative
Supercomputer HPC	✓		✓		Majority of TOP500 supercomputers uses some combination of CPUs and GPUs
General-purpose computing	✓		(✓)		CPU most versatile, flexible option; GPUs beginning to perform some tasks
Embedded control	✓	✓		✓	CPUs (→ MCU) dominant in low-cost, space-constrained, low-power, mobile applications
Prototyping, low-volume		✓			FPGAs best choice for low-volume, high-end applications; also pre-silicon validation, post-silicon validation and firmware development

CONCLUSIONES

El presente informe presentó una introducción resumida de los dispositivos de hardware que actualmente se utilizan en la implementación de modelos de Deep Learning.

Resulta necesario hacer hincapié en el crecimiento exponencial que está teniendo en los últimos años la inteligencia artificial, lo cual desencadena una demanda cada vez mayor de dispositivos de hardware con la capacidad de cómputo suficiente para realizar tareas cada vez más complejas.

El hecho de que ya existan modelos que tengan un comportamiento casi humano, da cuenta de los importante avances que hubo en esta materia. Sin embargo, es necesario recalcar el hecho de que cada unidad de procesamiento tiene sus limitaciones, lo que también puede llegar a limitar el desarrollo o expansión de algunas áreas de investigación.

Si bien no hay una regla específica que indique qué dispositivo es para qué área de desarrollo o tipo de dato, se deben tener en cuenta las ventajas y limitaciones de cada unidad, de acuerdo al modelo de Deep Learning que se desee implementar.

Por otro lado, es importante que se sigan haciendo investigaciones en el área de hardware especializado, con el fin de abaratar costos y ganar capacidad de procesamiento.

REFERENCIAS Y BIBLIOGRAFÍA

[1] Alba Centeno, Franco. 2019. *Deep Learning*. N.p.: Universidad de Sevilla.

[2] ARROW ELECTRONICS. 2018. “FPGA vs CPU vs GPU vs Microcontroller: How Do They Fit into the Processing Jigsaw Puzzle? | Arrow.com.” Arrow Electronics. https://www.arrow.com/en/research-and-events/articles/fpga-vs-cpu-vs-gpu-vs-microcontroller.

[3] Asenjo Plaza, Rafael, Eladio D. Gutiérrez Carrasco, and Julián Ramos Cózar. 2001. *Fundamentos de los computadores*. N.p.: Servicio de Publicaciones e Intercambio Científico de la Universidad de Málaga.

[4] AWS. n.d. “¿Qué es una CPU? Explicación de la unidad central de procesamiento.” AWS. Accessed February 20, 2024. https://aws.amazon.com/es/what-is/cpu/.

[5] BASOGAIN OLABE, Xabier. n.d. *REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES*. N.p.: Escuela Superior de Ingeniería de Bilbao,.

[6] CANO AVALOS, Javier. 2022. *Introducción al Deep Learning. Aplicación en R*. N.p.: Universidad de Sevilla.

[7] CISCO. 2010. *CCNA Discovery Course Booklet: Networking for home and small businesses, version 4.0*. N.p.: Cisco Press.

[8] Fraire, Juan A., Pablo Ferreyra, and Carlos Marques. n.d. *OpenCL Overview, Implementation, and Performance Comparison*.

[9] GAVIÑA RUEDA, Javier. 2022. *Estimación de Pose y sus Aplicaciones*. N.p.: Universidad de Valencia.

[10] Google Cloud. n.d. “Unidades de procesamiento tensorial (TPU).” Google Cloud. Accessed February 19, 2024. https://cloud.google.com/tpu?hl=es-419#accelerate-ai-development-with-google-cloud-tpus.

[11] JUEGA REIMÚNDEZ, Carlos. 2010. *Estudio de rendimiento en GPU*. N.p.: Universidad Complutense de Madrid.

[12] MAESTRE BETOLAZA, XABIER. 2021. *IMPLEMENTACIÓN DE REDES NEURONALES EN PLATAFORMAS HARDWARE PARA SU APLICACIÓN EN INGENIERÍA ELÉCTRICA*. N.p.: ESCUELA DE INGENIERÍA DE BILBAO.

[13] MARÍN DE LA ROSA, José M. n.d. *FIELD PROGRAMMABLE GATE ARRAY (FPGA)*.

NN. n.d. Segunda Parte: TECNOLOGÍA CUDA. Accessed February 21, 2024. https://biblus.us.es/bibing/proyectos/abreproy/11926/fichero/Segunda+parte+TECNOLOGIA+CUDA%252Fi_cuda.pdf.

[14] PICCOLI, María F. 2011. *Computación de alto desempeño en GPU*. N.p.: Editorial de la Universidad Nacional de La Plata (Edulp).

[15] RIVERA, Antonio J., Francisco CHARTE, Macarena ESPINILLA, and María D. PEREZ GODOY. 2018. “Nuevas arquitecturas hardware de procesamiento de alto rendimiento para aprendizaje profundo.” *Enseñanza y Aprendizaje de Ingeniería de Computadores* 8 (6): 67-83.

[16] RIVERA, Iván, and Miguel VARGAS LOMBARDO. 2012. “Principios y Campos de Aplicación en CUDA Programación paralela y sus potencialidades.” *NEXO: Revista Cientifica* 25 (02): 39-46.

[17] RUCCI, Enzo. 2017. “FPGAs: ¿los procesadores del futuro?” *Bit & Byte* 06, no. 3 (Diciembre): 46.

[18] Shen, John P., and Mikko H. Lipasti. 2013. *Modern Processor Design: Fundamentals of Superscalar Processors*. N.p.: Waveland Press.

[19] SISTERNA, Cristian. n.d. “FIELD PROGRAMMABLE GATE ARRAYS (FPGAS).”

[20] Stack Overflow Contributors. n.d. *APRENDIZAJE: opencl*.

[21] Stallings, William. 2006. *Organización y arquitectura de computadores*. N.p.: Pearson Educación.

[22] 200 Respuestas: Hardware. n.d. N.p.: Usershop.

[23] Usha Kiran, K., S. Supritha, BV Vijayalaxmi, P. Yogitha, and R. Jyothi. 2023. *A Review on GPU Architectures and Programming*. N.p.: Global Academy of Technology.

[24] Velo Fuentes, Edward J. 2020. *Introducción a los métodos Deep Learning basados en Redes Neuronales*. N.p.: Universidade da Coruña.

[25] ZHAO, Yangyang, Chao WANG, Lei GONG, Xi LI, Aili WANG, and Xuehai ZHOU. 2017. “Deep Learning Accelerators.” In *High Performance Computing for Big Data: Methodologies and Applications*, edited by Chao Wang. N.p.: CRC Press.