Diseño de un sistema basado en SoC

Arquitectura de Sistemas Integrados

Alejandro Solá
asola01@ucm.es

Pablo Suárez
pasuar03@ucm.es

Abril 2023

Contents

1 Introducción

"*Turing propuso que la pregunta «¿puede pensar una máquina?» era demasiado filosófica para tener valor y, para hacerlo más concreto, propuso un «juego de imitación», la prueba de Turing, en la que intervienen dos personas y una computadora. Una persona, el interrogador, se sienta en una sala y teclea preguntas en la terminal de una computadora. Cuando aparecen las respuestas en la terminal, el interrogador intenta determinar si fueron hechas por otra persona o por una computadora. Si actúa de manera inteligente, según Turing es inteligente. Turing, señaló que una máquina podría fracasar y aún ser inteligente. Aun así creía que las máquinas podrían superar la prueba a finales del siglo xx.*" Wikipedia et al [Wik23].

Con el tiempo, el enfoque de Turing sobre la inteligencia artificial como una imitación del comportamiento humano no fue práctico, y el enfoque dominante ha sido el comportamiento racional.

El diseño del SoC que se expone en este documento, está basado en una FPGA, y tiene como objetivo principal el entrenamiento de redes neuronales convolucionales. Las redes neuronales convolucionales *(CNN)* es uno de los modelos de *deep learning* más utilizados para la detección y clasificación de imágenes, debido a su alta precisión en comparación con otros algoritmos de aprendizaje automático. Las *CNN* consiguen mejores resultados a costa de mayores requisitos de computación y memoria. Véstias et al [Vé19].

2 Propósito

El propósito de este diseño, como bien se ha mencionado en la introducción (1), es el entrenamiento de redes neuronales convolucionales. Investigando acerca de este modelo de *deep learning*, lo que más destaca es su potencial en el análisis de imágenes, algo de especial importancia cuando se quieren automatizar determinadas tareas, e.g., saber si un tumor es benigno, detectar defectos en radiografías de piezas industriales (alerta spoiler) o simplemente saber si hay gatos en una imágen. Todo este potencial requiere de una gran capacidad de cómputo, lo que incrementa el consumo del SoC. En este diseño se ha intentado ser lo más riguroso posible con el consumo, intentando que este no supere ciertos limites, pero también se han tenido en cuenta las limitaciones presentes y los *trade-offs* realistas que pueden surgir en este tipo de diseños. Para el diseño del sistema, hemos consultado fuentes como [ONN].
A continuación se muestra el orden que se suele llevar a cabo a la hora de diseñar un SoC:

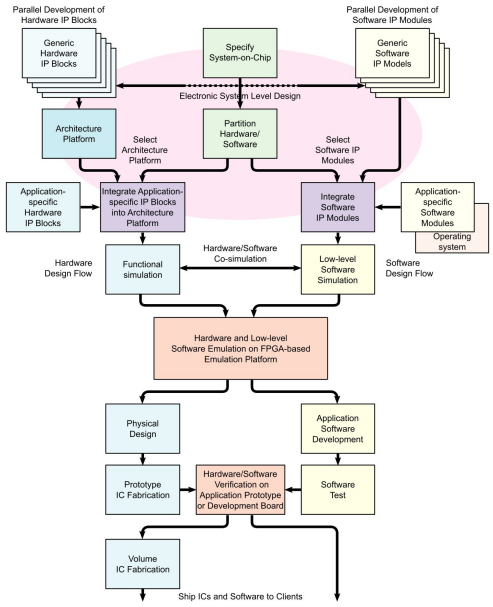

Figure 1: Fuente: https://en.wikipedia.org/wiki/System_on_a_chip

3 Requisitos y limitaciones

Como bien se ha comentado en la sección anterior, entrenar redes convolucionales requiere una gran capacidad de cómputo, que en términos prácticos se traduce en el uso de CPUs y/o unidades de procesamiento más potentes y probablemente grandes, e.g., GPUs, TPUs, etc. A esto hay que añadirle el coste adicional del uso de módulos y componentes adicionales que permitan su conectividad con otros dispositivos, o que permitan el uso de interfaces de usuario.

3.1 Entradas & salidas

La solución basada en SoC debe contar con una interfaz de usuario intuitiva y fácil de usar, diseñada para simplificar el proceso de configuración, entrenamiento y monitoreo de las redes neuronales. Esto implica directamente el uso de un sistema operativo.

Una aplicación de escritorio o una interfaz web que permita a los usuarios cargar y configurar datos de entrenamiento y validación, así como personalizar la arquitectura y los hiperparámetros [1] de la red neuronal, haría el proceso de entrenamiento más intuitivo. Se han considerado herramientas de visualización de datos para facilitar la exploración y el análisis de los datos de entrada y salida, ayudando a los usuarios a comprender mejor el funcionamiento y el rendimiento de la red neuronal. Como ejemplo, se ha pensado en un panel de control en tiempo real que muestre métricas clave, como la precisión, la pérdida y el tiempo de entrenamiento, permitiendo a los usuarios monitorear y ajustar el proceso de entrenamiento según sea necesario. Esto requiere una salida capaz de transmitir esta información a una pantalla externa.

El diseño implementado también debe ser capaz de manejar diversos tipos y estructuras de datos, ya que esto permitirá a los usuarios entrenar redes neuronales para una amplia variedad de aplicaciones y dominios. El sistema también tiene que ser compatible con el análisis de datos de texto para campos como el procesamiento de lenguaje natural, la traducción automática y la generación de texto.

Como la finalidad de este diseño es la detección de defectos en imágenes de diferentes formatos y resoluciones, el diseño ha de ser capaz de manejar datos relativos a este campo. Esto está relacionado con aplicaciones de visión por computadora, como la clasificación de imágenes, la detección de defectos y la segmentación semántica.

Dicho esto, las entradas y salidas del sistema podrían resumirse de la siguiente manera:

1. **Entradas**: La solución basada en SoC debe permitir la carga y el preprocesamiento de datos de entrenamiento y validación en diferentes formatos, como `.csv`, `.json`, `.xml` y formatos de imagen estándar (`.jpeg`, `.png`, etc.). Además, debe facilitar la división de los datos en lotes y la aplicación de transformaciones y aumentos de datos según sea necesario[2].

2. **Salidas**: La solución debe proporcionar salidas útiles y procesables del entrenamiento de la red neuronal, como predicciones, clasificaciones, generaciones de texto y detección de objetos. Estas salidas deben poder exportarse en formatos de datos comunes para su posterior análisis y uso en aplicaciones del mundo real.

El SoC incluye diversas interfaces de comunicación para permitir la conexión y comunicación con otros dispositivos y sistemas. Estas interfaces incluyen:

- USB: Para la conexión con dispositivos de almacenamiento externo, periféricos de entrada y salida y otros dispositivos compatibles con USB.
- HDMI: Para la conexión con periféricos para su control.
- Wi-Fi/Bluetooth: Para la conexión inalámbrica con dispositivos y redes, lo que permite una mayor flexibilidad en la ubicación y movilidad del sistema.

El SoC también debe incorporar mecanismos y herramientas de depuración y actualización para facilitar el desarrollo y mantenimiento del software y firmware que ejecuta en el sistema. Estos mecanismos son:

[1] Los hiperparámetros principales y tuneables son; la tasa de aprendizaje, el tamaño del batch, el número de neuronas en la red, el tipo de función de activación y la tasa de aprendizaje.

[2] La generación de nuevos datos a partir de un conjunto de datos de entrenamiento es crucial para entrenar correctamente una red.

- JTAG (Joint Test Action Group): Es una interfaz estándar de prueba y depuración que permite la conexión de herramientas de depuración externas para inspeccionar y controlar el estado interno de la CPU, la memoria y otros componentes del SoC. JTAG es ampliamente utilizado en la industria para depurar y probar circuitos integrados y sistemas en chip.
- SWD (Serial Wire Debug): Es una alternativa más simple y de menor pin-count al JTAG que ofrece funcionalidad similar para la depuración y el acceso a la memoria y registros internos del SoC.
- OTA (Over-The-Air) updates: Es un mecanismo que permite la actualización del firmware y el software del sistema de forma remota a través de una conexión de red, como Wi-Fi o Ethernet. Esto facilita la implementación de actualizaciones de seguridad, corrección de errores y mejoras en el rendimiento sin la necesidad de acceder físicamente al dispositivo.

3.2 Dimensiones

El SoC debe ser compacto, con dimensiones máximas de $70 \times 70 \times 10$ [mm], para facilitar su integración en dispositivos portátiles. Para lograr un tamaño reducido, se han considerado los siguientes aspectos:

- Componentes miniaturizados (SMD): Se utilizarán componentes electrónicos de montaje en superficie, que ocupan menos espacio en la PCB y permiten un diseño más compacto.
- Técnicas avanzadas de diseño de PCB: Se aplicarán técnicas de diseño de PCB de múltiples capas, enrutamiento de alta densidad y optimización de la colocación de componentes para minimizar el tamaño total de la placa.
- Tecnologías de fabricación de semiconductores de última generación: Se emplearán procesos de fabricación avanzados, como FinFET y procesos de litografía ultravioleta extrema (EUV), para reducir el tamaño de los transistores y aumentar la densidad de integración en el chip.

Por otra parte, el diseño debe ser confiable y duradero en diversas condiciones de funcionamiento. Para garantizar su robustez y fiabilidad, se deben emplear componentes y materiales de alta calidad que cumplan con estándares internacionales, sistemas de protección (contra sobretensiones, cortocircuitos, descargas electrostáticas), y se deben realizar pruebas exhaustivas en diferentes condiciones ambientales.

3.3 Consumo

El consumo de energía del SoC debe ser optimizado. Teniendo en cuenta que el rango de potencia activa de una FPGA es $P_{FPGA} \in \{6,24 - 34,7\}$ (mW) [RLG+19] y que la CPU escogida tiene 4 cores y cada CORE consume ~ 250 mW, entonces la potencia total consumida para el peor caso, i.e: $P_{FPGA} = 34,7$ mW, por la FPGA y la CPU es: $P_{total} = 34,7 + 4 \cdot 250 = 1,035$ W. A esta potencia hay que sumarle el consumo de los módulos incluidos, e.g: WI-FI, BLuetooth etc, buscando en Mouser se ha estimado un consumo total máximo de 5 W. Para lograr un menor consumo de energía, se han considerado los siguientes aspectos:

- Circuitos de baja potencia: Se diseñarán circuitos electrónicos de baja potencia para minimizar el consumo de energía durante el procesamiento y el entrenamiento de las redes neuronales. Esto incluye el uso de transistores de baja potencia y técnicas de optimización de energía a nivel de circuito.
- Componentes de hardware optimizados: Se incorporará una unidad de aceleración basada en una FPGA para el entrenamiento de redes neuronales, puesto que las FPGAs ofrecen un alto rendimiento computacional, una gran η, baja latencia y un coste medio. Huawei et al [Hua].
- Técnicas de optimización de energía a nivel de software: Se implementarán técnicas de administración de energía dinámica, como la reducción de la precisión de cálculo, el ajuste dinámico de la frecuencia del reloj y la desactivación de módulos no utilizados, para minimizar el consumo de energía en tiempo de ejecución.
- Sistemas de enfriamiento eficientes: Se utilizarán sistemas de enfriamiento pasivos y activos, como disipadores de calor, heat pipes y ventiladores de bajo consumo, para mantener la temperatura de funcionamiento del SoC dentro de los límites aceptables y evitar el consumo excesivo de energía debido al sobrecalentamiento. Un sobrecalentamiento del sistema hará de forma indirecta que nuestro sistema funcione más lento.

3.4 Coste de producción

Para estimar el costo de producción de la solución basada en SoC, se deben considerar los siguientes aspectos:

- Costo de los componentes electrónicos: Adquirir componentes en grandes cantidades y seleccionar proveedores confiables y rentables para mantener el equilibrio entre calidad y precio.
- Costo de fabricación de la PCB: Diseñar la PCB de manera eficiente, reduciendo la cantidad de capas y área total, y seleccionar un fabricante de PCB que ofrezca precios competitivos.
- Costo de ensamblaje y montaje: Utilizar la automatización del proceso de ensamblaje y proveedores de servicios de ensamblaje con precios competitivos y alta calidad.
- Costo de pruebas y validación: Diseñar procedimientos de prueba eficientes y automatizados, y utilizar herramientas de prueba y equipos de medición rentables.

Teniendo en cuenta estos factores, y atendiendo al costo de producción de diversos SoC que hemos encontrado en web distribuidoras como Mouser, la solución basada en SoC puede oscilar entre 100 y 200 euros por unidad. Este costo se vería reducido dependiendo del volumen de producción, componentes seleccionados y acuerdos con proveedores y fabricantes.

4 Arquitectura

Como la finalidad del diseño es la de entrenar redes neuronales, lo primero que se ha pensado ha sido en el uso de una GPU. Las soluciones basadas en GPUs resultan programables y baratas, al tiempo que proporcionan la elevada potencia de cálculo que suele requerirse durante la fase de entrenamiento. Sin embargo, la implementación de redes en GPUs es muy lenta y consume mucha energía.

Por otro lado, las soluciones basadas en `ASIC` [3], por su parte, ofrecen un menor consumo de energía con un rendimiento moderado. En comparación con las soluciones basadas en GPU, los `ASIC` disponen de recursos informáticos relativamente limitados, por lo que resulta difícil desarrollar arquitecturas de red complejas y masivas. Además, no proporcionan flexibilidad y requieren un ciclo de desarrollo más largo. Faisal Shehzad et al [SRS+21].

Por último, se tuvo en cuenta una solución basada en `FPGA`, este tipo de diseño ofrece un alto grado de programabilidad, adaptándose así, a las especificaciones mencionadas anteriormente (módulos WIFI, Bluetooth, etc). Al incluir y permitir el uso de `RAM` integrada, y elementos de entrada salida, permite diseñar sistemas que cumplen con especificaciones mencionadas antes. En comparación con las `GPUs`, las `FPGAs` alcanzan frecuencias menores, aunque hay menos ciclos de reloj, mejorando asi el consumo de potencia. Faisal Shehzad et al [SRS+21].

La arquitectura del SoC para entrenar redes neuronales debe ser eficiente y escalable, optimizando el rendimiento y el consumo de energía (dentro de unas expectativas realistas). Por ello, se ha escogido una solución basada en una `FPGA` y un procesador ARM encargado de controlar la ejecución general del sistema que delega tareas de alta carga computacional en la `FPGA`. A continuación, se presenta una descripción general de los elementos clave que conforman la arquitectura del SoC:

1. Núcleo de procesamiento central (CPU): El SoC debe incluir al menos un núcleo de procesamiento de alto rendimiento, la CPU escogida es la ARM Cortex-9. Este es un procesador de 32 bits fuera de orden, superescalar y especulativo, con múltiples núcleos, que proporciona hasta 4 núcleos coherentes con la caché, cada uno de los cuales implementa el conjunto de instrucciones de la arquitectura ARM v7. Algunas de sus características más relevantes son las siguientes:

Fabricante	$\#_{CORES}$	**f** (GHz)	**L1** CACHE	**L2** CACHE	Tamaño	Consumo/CORE (mW)
ARM	1-4	0.8-2	32KB	128KB-8MB (configurable)	1.5 mm^2	$\sim$ 250 mW

Table 1: Propiedades del `ARM Cortex-A9`

[3] Como se estudió en teoría, `ASIC` hace referencia a sistemas integrados personalizados para un uso concreto, en lugar de destinados a un uso general

La elección de este procesador se debe a varias razones, la primera es por una cuestión realista, pues en todo momento se ha querido seleccionar un procesador confiable y conocido. La segunda razón se debe a que es un procesador superescalar, fuera de orden y especulativo. En clase se han estudiado estos conceptos y se han visto sus ventajas y desventajas, y generalmente este tipo de procesadores escalan bastante bien y por ello se ha considerado esta arquitectura en específico. La última razón que ha condicionado nuestra elección ha sido el tipo de datos que se van a manejar. Como las operaciones llevadas a cabo por las redes neuronales involucran datos de tipo `float` (mayoritariamente), es importante hacer uso de un procesador superescalar con una unidad de punto flotante (*Floating Point Unit*), y el `ARM Cortex A9` posee una de estas unidades.

Es importante destacar que la mayor parte de la energía en un sistema, es gastada en mover datos de la DRAM externa a SRAM interna, de la SRAM a registro, o de registro a registro (flip-flops). Cuanto mayor sea el tamaño de la palabra de punto flotante, más energía se gastará. Por lo tanto es importante ser capaz de reducir el tamaño palabra.

A continuación se adjunta un diagrama de bloques relativo al módulo del procesador:

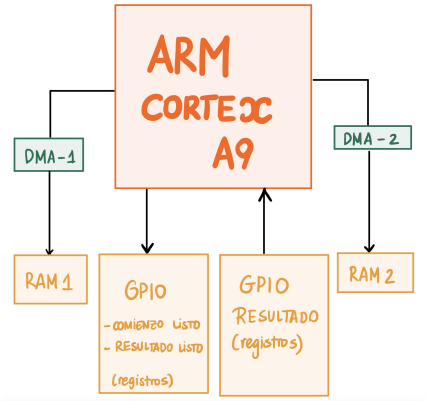

Figure 2: Módulo del procesador

En la figura de arriba se representa el componente procesador del **SoC**, este está formado por el procesador mencionado antes y por una serie de módulos colindantes encargados de llevar hasta él, el resultado del entrenamiento de la red neuronal almacenado en registros de entrada y salida de propósito general (`módulo GPIO`) y el desplazamiento de datos hacia el acelerador hardware para un rápido procesado y cómputo de datos, mediante el uso de `DMAs`[4]

2. Acelerador Hardware: Para acelerar el entrenamiento de redes neuronales, el SoC debe incluir un módulo de alto rendimiento, que cuente con unidades que efectúen las operaciones matriciales y tensoriales necesarias. Para estas operaciones, se ha hecho uso de 32 multiplicadores con un par de entradas de 16 bits cada uno. Pero claro, antes de realizar las multiplicaciones de matrices necesarias, se necesitan activar las neuronas. La función de activación escogida para este caso ha sido la **ReLU**, pues es la más conocida y la que mejores resultados da entre las funciones de activación más conocidas como pueden ser; $\sigma(x)$, $tanh(x)$, *"Leaky ReLU"*. Si $\exists x \mid x \in \mathbb{R}^{n \times m}$, o si $\exists x \mid x \in \mathbb{R}^{n}$, entonces la función **ReLU** se define de la siguiente manera: $f(x) = max(0, x)$. Para ello se ha implementado un módulo que simula el comportamiento de esta función. Por último, para poder calcular la salida de la red neuronal, se ha de aplicar una función que traduzca las operaciones matriciales y tensoriales que se dan dentro de la red en probabilidades, e.g: la probabilidad de que haya un defecto es $\mathbb{P}$, la probabilidad de que haya un gato es $\mathbb{X}$. Para ello se ha implementado un bloque que simula el comportamiento seguido por la función sigmoidea: $\sigma(x) = \frac{1}{1+e^{-x}}$. El módulo acelerador del sistema, puede verse en la figura 3.

3. Memoria integrada: El SoC debe contar con memoria integrada, como SRAM para almacenamiento temporal y DRAM para almacenamiento a largo plazo, que garantice el acceso rápido a los datos y modelos de redes neuronales durante el entrenamiento y la inferencia.

4. Interconexión de alta velocidad: Para permitir la comunicación eficiente entre los componentes del SoC, se debe emplear una interconexión de alta velocidad, como el bus AXI o NoC (Network-on-Chip), que permita la transferencia rápida y concurrente de datos.

[4]Los `DMAs`, en castellano: *acceso directo a memoria*, son funciones ofrecidas por algunas arquitecturas de buses que permiten enviar datos directamente desde un dispositivo conectado (como una unidad de disco) a memoria.

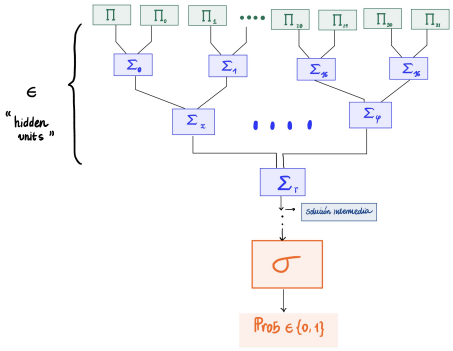

Figure 3: Módulos del acelerador

5. Controlador de memoria externa: El SoC debe incluir un controlador de memoria externa compatible con estándares de memoria populares, como DDR4 o LPDDR5, para ampliar la capacidad de almacenamiento de datos y modelos de redes neuronales según sea necesario.

6. Interfaces de comunicación: Para facilitar la conexión con otros dispositivos y sistemas, el SoC debe incluir una variedad de interfaces de comunicación, como USB, Ethernet, Wi-Fi, Bluetooth, y UART, así como interfaces para sensores y actuadores, como GPIO, SPI, I2C y ADC/DAC.

7. Unidad de gestión de energía (PMU): El SoC debe incorporar una PMU que gestione el suministro de energía a los distintos componentes y permita la implementación de estrategias de ahorro de energía, como el ajuste dinámico de la frecuencia y el voltaje y la desactivación de componentes no utilizados.

8. Unidad de seguridad: Para garantizar la seguridad y protección de los datos y modelos de redes neuronales, el SoC debe incluir una unidad de seguridad que gestione el cifrado y descifrado de datos, la autenticación de dispositivos y usuarios, y la integridad del firmware y el software del sistema.

Esta arquitectura puede adaptarse y modificarse según los requisitos específicos de la aplicación y el rendimiento deseado, utilizando diferentes configuraciones de núcleos de procesamiento, unidades de aceleración de IA, y capacidades de memoria y comunicación.

4.1 Descripción de los módulos

A continuación, se describen con más detalle los componentes clave de la arquitectura del SoC: CPU, GPU, memoria y controlador de memoria.

1. CPU (Núcleo de procesamiento central) La CPU es el componente principal que gestiona las tareas generales del sistema, ejecuta el software del sistema y controla el funcionamiento de los demás componentes del SoC. En nuestro caso se ha escogido un multicore de 4 núcleos.

   - Conjunto de instrucciones: La CPU utiliza un conjunto de instrucciones específico (ISA) para ejecutar operaciones y procesar datos. Por ejemplo, ARM utiliza la arquitectura ARMv7.
   - Caché: La CPU contiene una memoria caché de dos niveles, donde L1 es de 32 KB y L2 llega hasta los 8MB. Esta memoria sirve para almacenar temporalmente los datos e instrucciones que se utilizan con frecuencia, lo que ayuda a reducir la latencia de acceso a la memoria y mejora el rendimiento.
   - Unidad de control: La unidad de control de la CPU se encarga de decodificar las instrucciones y emitir señales de control para coordinar las operaciones de los demás componentes del SoC.

   Se ha hecho un pequeño estudio de mercado y en Mouser se han encontrando componentes de CPU a unos 10 € la unidad. Este precio se reduce a unos 6 € si se compran más de 600.

2. GPU (Unidad de procesamiento gráfico) / TPU (Unidad de procesamiento tensorial). Aunque no se ha decidido implementar una solución basada en GPU, se ha querido mantener la siguiente explicación. La GPU es una unidad especializada en el procesamiento paralelo de datos, lo que la hace adecuada para tareas gráficas y de aprendizaje profundo. Ejemplos de GPUs populares

incluyen la serie Mali de ARM y Adreno de Qualcomm. Por otro lado, la TPU es una unidad de aceleración específicamente diseñada para acelerar cálculos de tensores y matrices en aplicaciones de aprendizaje profundo.

Características clave:

- Arquitectura: Las GPU y TPU tienen una arquitectura de múltiples núcleos, lo que les permite procesar datos en paralelo y, por lo tanto, acelerar el entrenamiento e inferencia de redes neuronales.
- Pipelining: Tanto las GPU como las TPU utilizan técnicas de pipelining y paralelismo para optimizar el rendimiento y maximizar la eficiencia del procesamiento.
- Memoria local: Estas unidades cuentan con memoria local de alta velocidad para almacenar temporalmente datos y resultados intermedios durante el procesamiento.

Para este componente, se ha encontrado un precio/unidad de unos 30 € en Mouser.

3. Memoria integrada La memoria integrada en el SoC proporciona almacenamiento temporal y a largo plazo para datos y modelos de redes neuronales. La memoria integrada puede incluir SRAM (Static Random-Access Memory) y DRAM (Dynamic Random-Access Memory).

   Características clave:

   - SRAM: La SRAM es una memoria rápida y de bajo consumo que se utiliza para almacenar datos temporales, como registros de CPU y caché. No necesita ser refrescada periódicamente.
   - DRAM: La DRAM es una memoria más lenta y de mayor consumo que la SRAM, pero ofrece una mayor densidad de almacenamiento. Se utiliza para almacenar datos a largo plazo y necesita ser refrescada periódicamente

   La memoria SRAM puede comprarse por 60 €/unidad en Mouser.

4. Controlador de memoria externa El controlador de memoria externa permite la conexión y gestión de memoria externa adicional, como DDR4 o LPDDR5, para expandir la capacidad de almacenamiento según sea necesario. El controlador de memoria externa es fundamental para mantener un rendimiento óptimo al acceder y transferir datos entre la memoria externa y los demás componentes del SoC.

   - Tipos de memoria soportados: El controlador de memoria externa puede ser compatible con distintos tipos de memoria externa, como DDR3, DDR4, LPDDR4, LPDDR5, etc. Se ha optado por un término medio y se ha escogido una memoria de tipo LPDDR4X. Esta memoria consume menos potencia que la DDR4 y aunque no alacanza la velocidad de 6400 MBps de la LPDDR5, es capaz de alcanzar velocidades de 4267 MBps.
   - Ancho de banda: El controlador de memoria externa debe ser capaz de manejar un ancho de banda suficiente para garantizar la transferencia rápida de datos entre la memoria externa y los demás componentes del SoC. El ancho de banda dependerá del tipo de memoria externa utilizada y de las necesidades de la aplicación.
   - Interfaz: El controlador de memoria externa se comunica con la memoria externa a través de una interfaz especializada, como un bus de direcciones y datos. La interfaz debe ser compatible con los estándares de la memoria externa seleccionada.
   - Control de latencia y temporización: El controlador de memoria externa es responsable de gestionar la latencia y la temporización de las operaciones de lectura y escritura en la memoria externa, garantizando que se cumplan los tiempos de acceso y refresco especificados por el fabricante de la memoria.

   El precio de este módulo en diferentes distribuidores está entorno a los 70$.

4.2 Estrategia por bloques

A continuación, se presenta la estrategia por capas para nuestro SoC:

- Capa de hardware (bajo nivel): Esta capa incluye todos los componentes de hardware del SoC, como la CPU, la GPU, la memoria, las interfaces de comunicación y los módulos de entrada/salida. La capa de hardware es responsable de ejecutar las instrucciones y operaciones a nivel de bits y registros.

- Capa de sistema operativo: Un sistema operativo (SO) proporciona un entorno de ejecución y gestión de recursos para el software de aplicación. El SO es responsable de la asignación de recursos de hardware, la planificación de tareas, la gestión de la memoria y la implementación de servicios esenciales, como la comunicación de red y la seguridad. Algunos ejemplos de sistemas operativos utilizados en sistemas embebidos y SoC incluyen Linux, FreeRTOS y Zephyr.
- Capa de aplicación: La capa de aplicación incluye el software específico del dominio y la lógica de negocio que utiliza las bibliotecas y frameworks de aprendizaje profundo para llevar a cabo tareas específicas, como el reconocimiento de imágenes, la traducción automática o la detección de anomalías. Esta capa es donde los desarrolladores implementan sus algoritmos y soluciones específicas del dominio y se conectan con otros sistemas y servicios a través de interfaces de usuario y API.

Esta estrategia por capas simplifica el proceso de desarrollo, implementación y mantenimiento del SoC, al permitir a los desarrolladores y equipos de ingeniería centrarse en componentes y aspectos específicos del sistema sin tener que lidiar con la complejidad total del SoC en cada nivel. Además, esta estrategia promueve la reutilización de componentes y la compatibilidad entre diferentes plataformas y dispositivos.

4.3 Relación entre bloques

5 Diagrama de bloques relacionando CPU y "GPU"

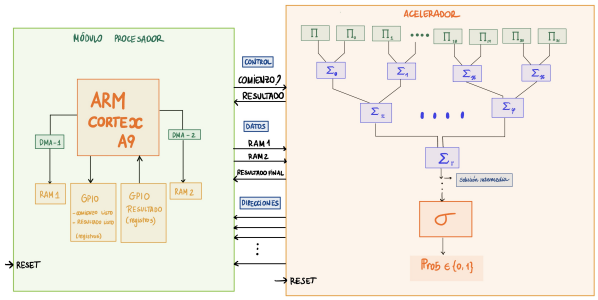

Figure 4: Diagrama de bloques relacionando CPU y GPU

6 Diagrama de bloques del sistema completo

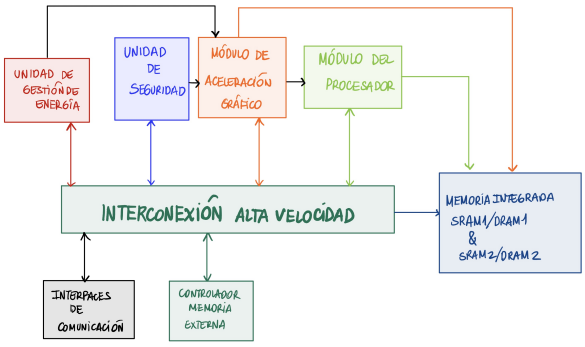

Figure 5: Diagrama de bloques

Se han establecido algunas interconexiones entre módulos para reducir la carga de tráfico en el bus de interconexión. Cabe destacar además, la necesidad de un protocolo de comunicación entre módulos, con el fin de que no haya pérdida de datos. Se ha tenido en cuenta el protocolo AMBA. La Arquitectura de Bus de Microcontrolador Avanzado ARM (AMBA) es una especificación de interconexión en chip de estándar abierto para la conexión y gestión de bloques funcionales en diseños SOC. En el siguiente enlace se explica al detalle el funcionamiento de este protocolo. Protocolo AMBA.

7 Estructura HW y funcionamiento

En el caso del módulo del procesador, el chip escogido (`ARM Cortex A9`) se encargará de ejecutar códigos más livianos encargados de especificar los pesos asignados en la red y las muestras para el entrenamiento y el test de la red neuronal, con el fin de realizar tareas menos computacionales, pues de esto se encarga el acelerador escogido.

8 Estructura SW

El entrenamiento de la red neuronal, se realizará implementando librerías como `TensorFlow` y `Keras`. Estas dos librerías son la norma dentro del mundo de la inteligencia artificial y por ello, conforman el `back-bone` del proceso de entrenamiento. Como estas librerías están escritas en `Python`, la implementación de código será de alto nivel, esto aumentará la latencia, pues no estamos tan cerca del lenguaje máquina como lo está C, pero esto es algo sabido y en cierto modo inevitable, cuando uno quiere implementar algoritmos de inteligencia artificial. El sistema empotrado ha de ser capaz de acceder a esas librerías e implementarlas correctamente. Como consecuencia, el SoC presentará más capas software, lo que incrementará el uso de recursos hardware y software para que funcione correctamente.

Es importante destacar que a parte de las librerías como `TensorFlow` y `Keras`, la capa software ha de cargar librerías para el manejo y lectura de datos e imágenes como: `numpy, pandas, opencv, PIL` y `torchvision.`

9 Conclusiones

En conclusión, diseñar un SoC para entrenar redes neuronales implica planificación y una estrategia por capas, enfocándose en rendimiento, eficiencia energética y conectividad. La estructura de capas facilita el desarrollo, implementación y mantenimiento, mientras que la integración de mecanismos de depuración, actualización y diversas interfaces garantiza escalabilidad y flexibilidad. Un SoC bien diseñado y optimizado permitirá aprovechar el potencial del aprendizaje profundo en múltiples aplicaciones.
[TG22],[SRS+21],[Ali09],[Vé19][Hua]

References

[Ali09] Rafael Colom-Palero Ricardo Monzó J. Lerche Christoph Martínez Aliaga, Ramón Gadea. System-on-chip implementation of neural network training on fpga. *International Journal on Advances in Systems and Measurements*, (2):44–55, 2009.

[Hua] Huawei. Differences between cpu, gpu, fpga, and asic. https://forum.huawei.com/enterprise/en/differences-between-cpu-gpu-fpga-and-asic/thread/966389-895.

[ONN] ONNX. Onnx supported tools. https://onnx.ai/supported-tools.html#deployModel.

[RLG+19] Marwen Roukhami, Mihai Teodor Lazarescu, Francesco Gregoretti, Younes Lahbib, and Abdelkader Mami. Very low power neural network fpga accelerators for tag-less remote person identification using capacitive sensors. *IEEE Access*, 7:102217–102231, 2019.

[SRS+21] Faisal Shehzad, Muhammad Rashid, Mohammed H. Sinky, Saud S. Alotaibi, and Muhammad Yousuf Irfan Zia. A scalable system-on-chip acceleration for deep neural networks. *IEEE Access*, 9:95412–95426, 2021.

[TG22] Morrison Abigail Trensch Guido. A system-on-chip based hybrid neuromorphic compute node architecture for reproducible hyper-real-time simulations of spiking neural networks. *Frontiers in Neuroinformatics*, 16(3):342–351, 2022.

[Vé19] Mário Véstias. A survey of convolutional neural networks on edge with reconfigurable computing. *Algorithms*, 12:154, 2019.

[Wik23] Wikipedia. Historia de la inteligencia artificial — wikipedia, la enciclopedia libre, 2023. [Internet; descargado 7-abril-2023].