Diseño de un **SoC** para el entrenamiento de **CNNs**

Arquitectura de Sistemas Integrados

Pablo Suárez y Alejandro Solá

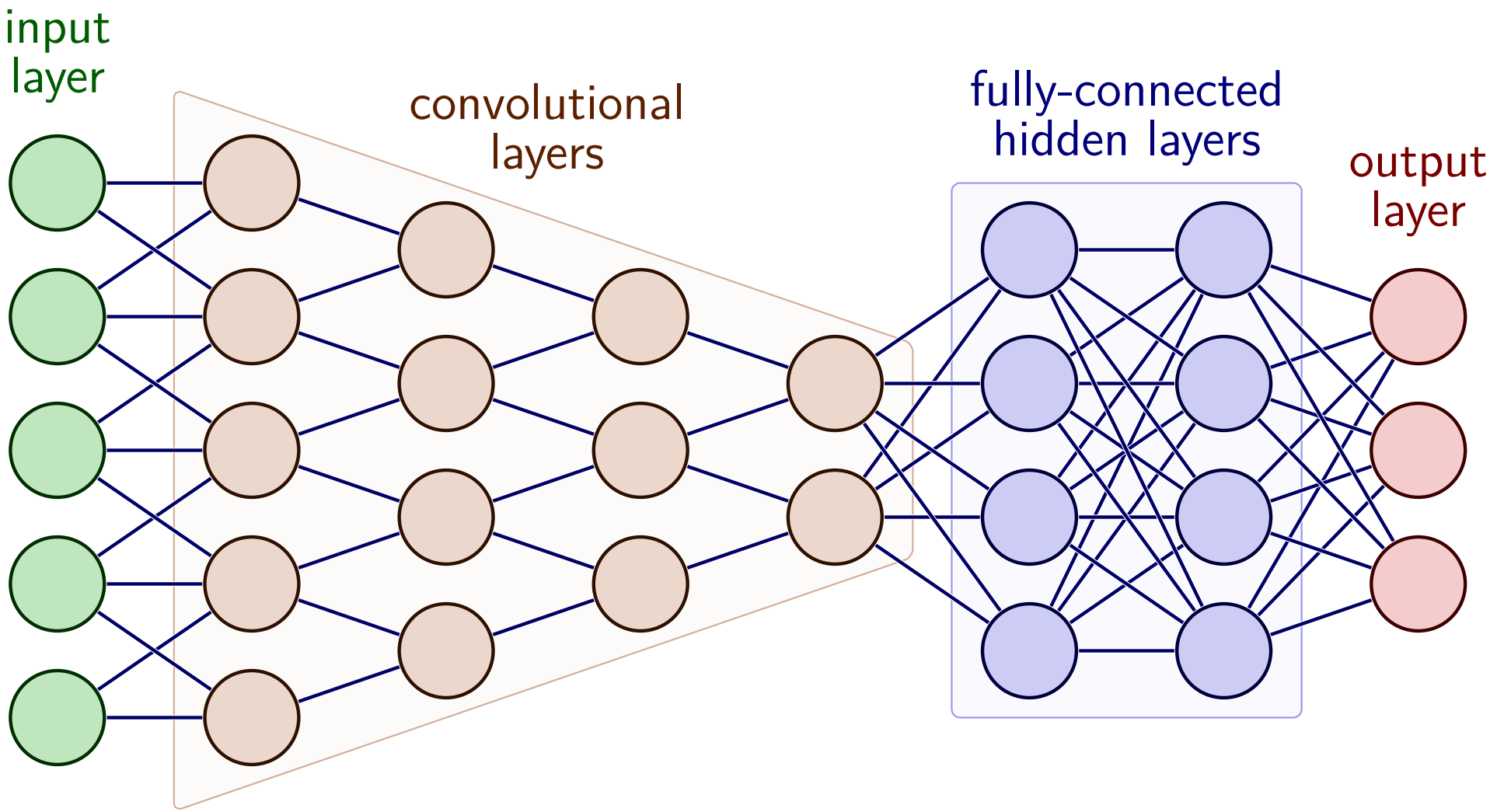
Facultad de Ciencias Físicas, Ingeniería Electrónica de Comunicaciones
Universidad Complutense de Madrid, UCM

28 de abril de 2024

Contenido

1. Introducción
2. Propósito
3. Arquitectura
4. Requisitos y limitaciones
5. Estructura HW
6. Estrategia por capas y optimización

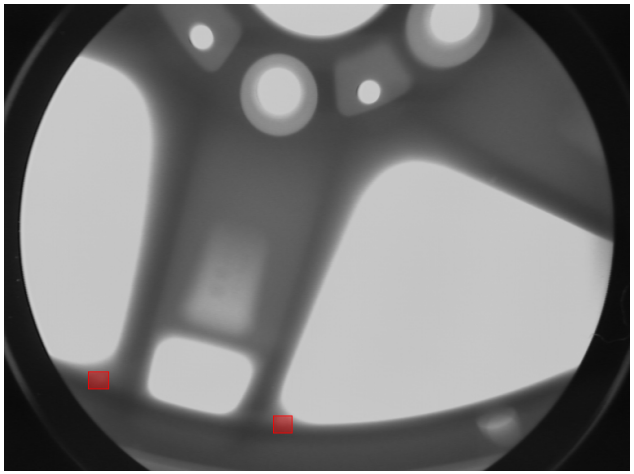
Introducción



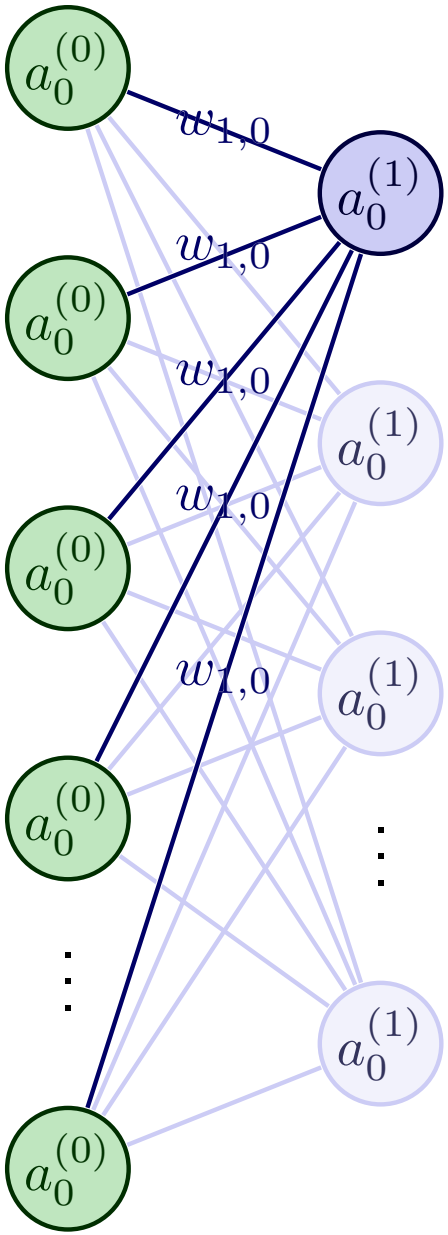
Propósito

- **¿Objetivo?**
  1. Entrenar redes neuronales convolucionales $\Rightarrow$ detectar defectos pequeños.
- **¿Requisitos?**
  1. Aceleración hardware.
  2. Múltiples `CORES`.
  3. Escalabilidad.
  4. Sistema Operativo: `UNIX`
- **Adicionalmente:**
  - Módulos: `WI-FI` y controladores de (imágen y video)
  - Puertos: `HDMI`

$\Rightarrow$ **FPGA** $\Leftarrow$



Tipos de operaciones



$$= \sigma\left(w_{1,0}a_0^{(0)} + w_{1,1}a_1^{(0)} + \ldots + w_{1,n}a_n^{(0)} + b_1^{(0)}\right)$$

$$= \sigma\left(\sum_{i=1}^{n} w_{1,i}a_i^{(0)} + b_1^{(0)}\right)$$

$$\begin{pmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_m^{(1)} \end{pmatrix} = \sigma\left[\begin{pmatrix} w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ w_{2,0} & w_{2,1} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,0} & w_{m,1} & \cdots & w_{m,n} \end{pmatrix}\begin{pmatrix} a_1^{(0)} \\ a_2^{(0)} \\ \vdots \\ a_n^{(0)} \end{pmatrix} + \begin{pmatrix} b_1^{(0)} \\ b_2^{(0)} \\ \vdots \\ b_m^{(0)} \end{pmatrix}\right]$$

$$a^{(1)} = \sigma\left(\mathrm{W}^{(0)}a^{(0)} + \mathrm{b}^{(0)}\right)$$

Descripción de los módulos

1. **GPU**
   - Arquitectura
   - Pipeline
   - Memoria local
2. **CPU**
   - Conjunto de instrucciones
   - Caché
   - Unidad de control
3. **Memoria integrada**
   - SRAM
   - DRAM
4. **Controlador de memoria externa**
   - Ancho de banda
   - Interfaz
   - Latencia

Comparativa

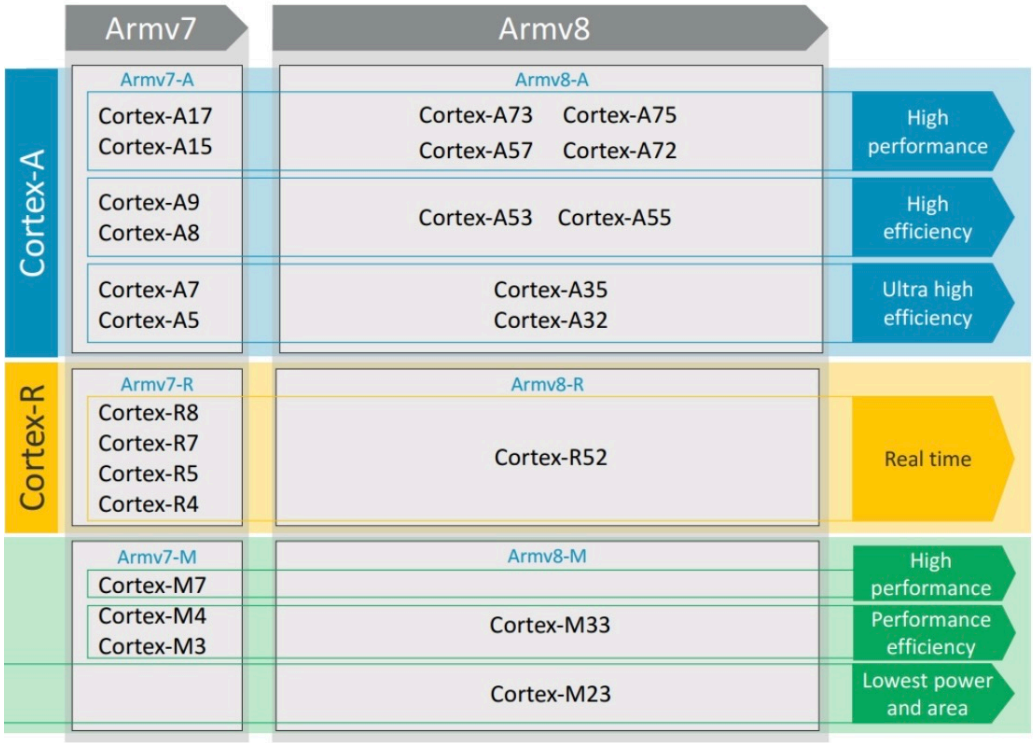


Figura: Comparación

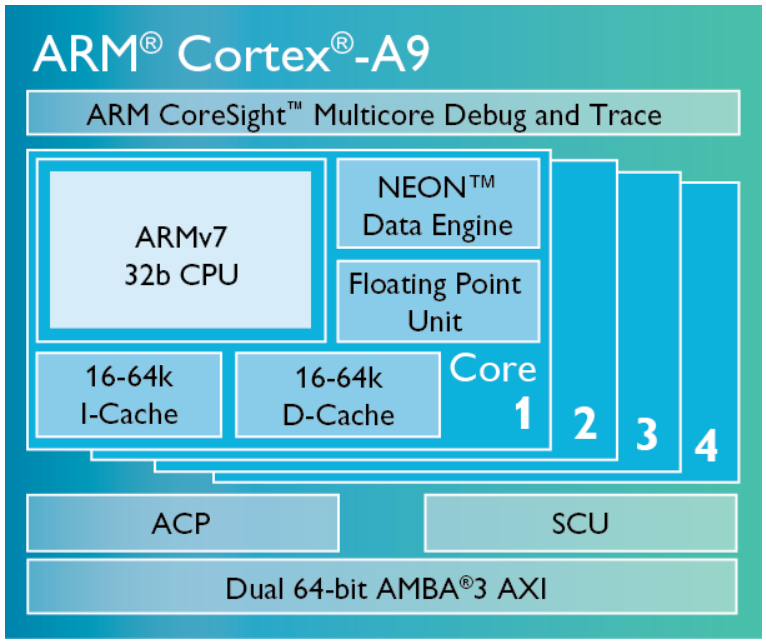


Figura: Esquemático

Procesador escogido

Características **clave** del procesador:

F	#	**f** (GHz)	**L1**	**L2**	Tamaño	(mW)/CORE
ARM	1-4	0.8-2	32KB	128KB-8MB	1.5 mm^2	~ 250 mW

Cuadro: Propiedades del `ARM Cortex-A9`

- `32 bits`.
- Superescalar, fuera de orden y especulativo.
- Repertorio `ARM v7`.
- `¡floating point unit!`

Pipeline

El pipeline de este porocesador tiene la siguiente forma:

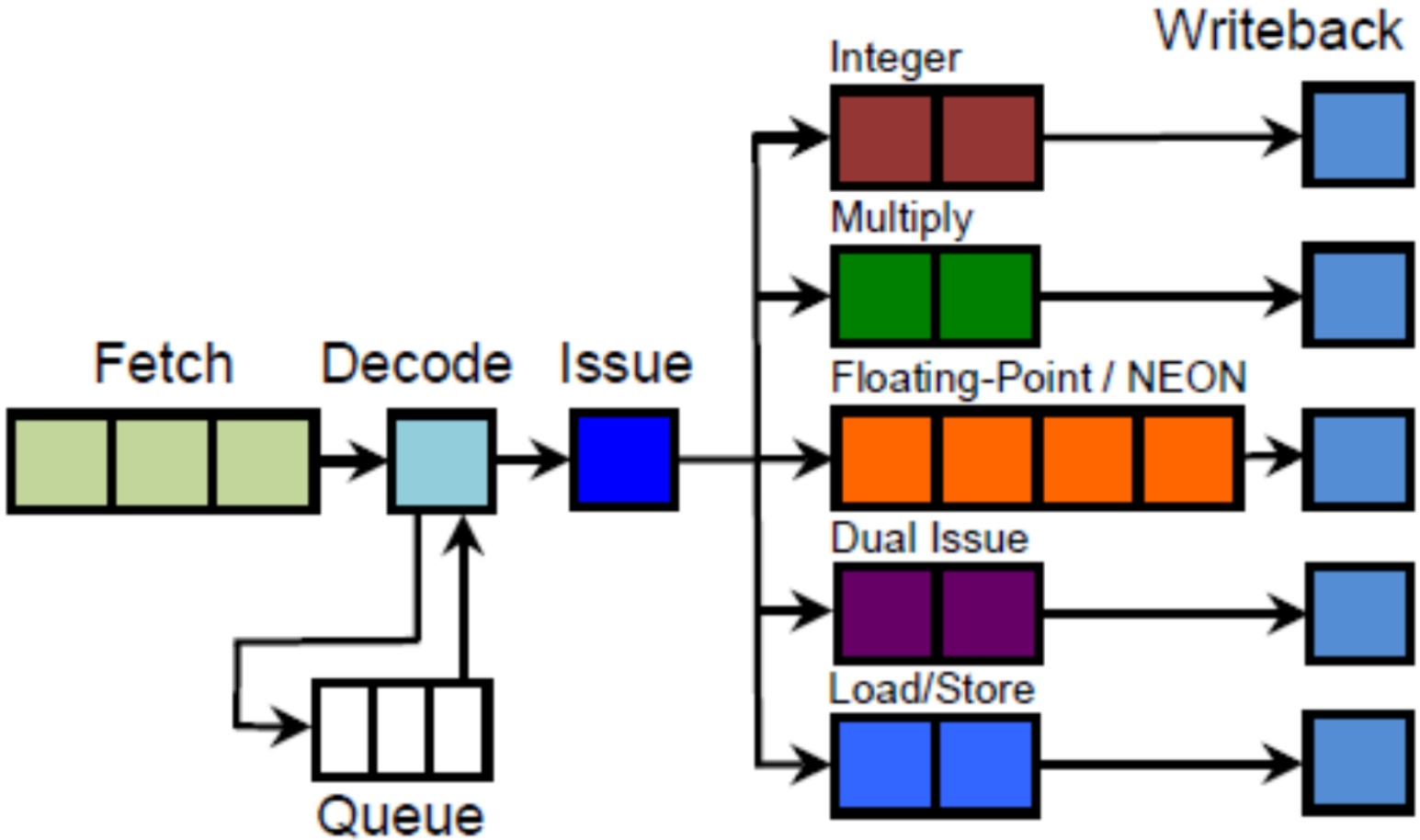


Figura: Pipeline `ARM Cortex A7`

Caché

Caché de **2 niveles** ⇒ **L1** y **L2**.

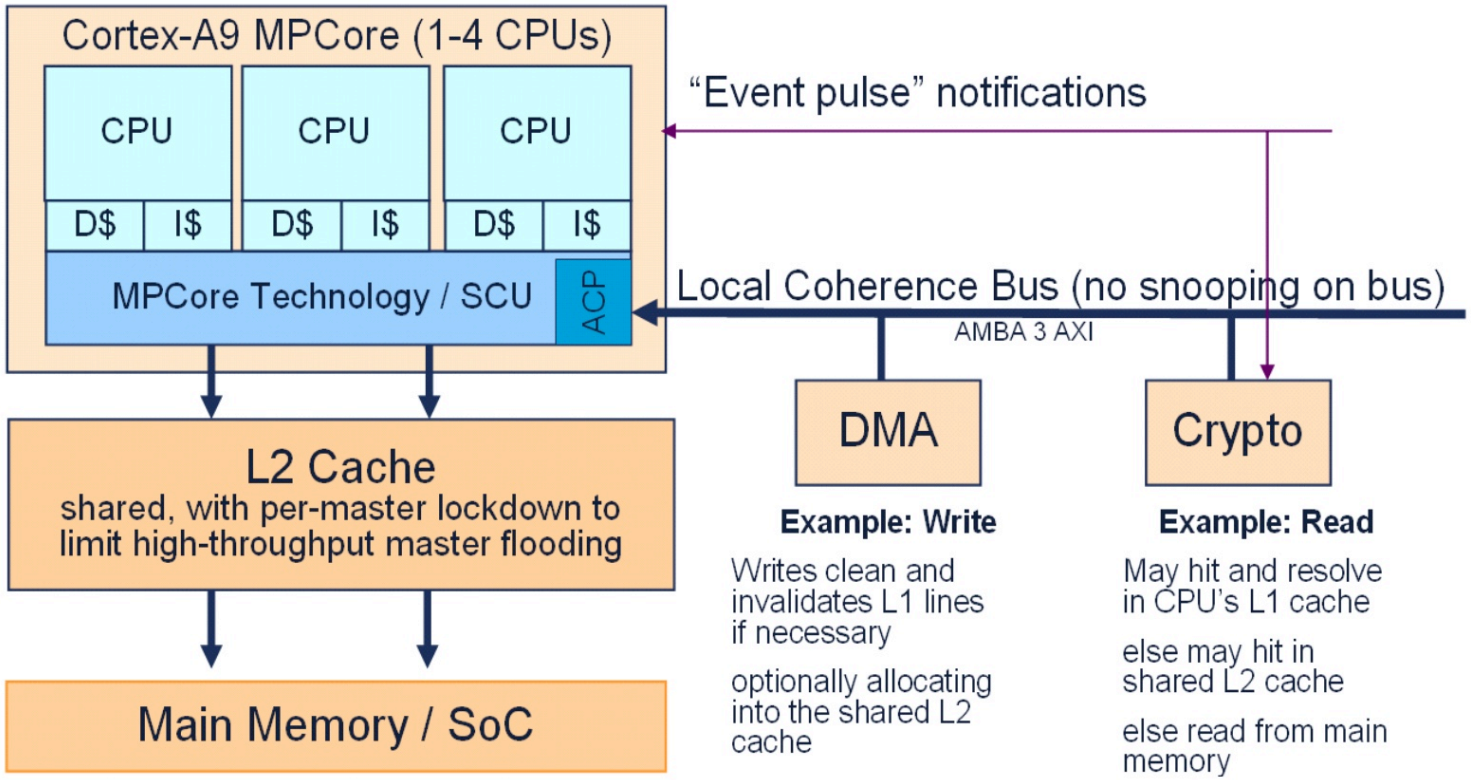


Figura: Niveles de caché

Entradas y salidas

El diseño cuenta con las siguientes vías e interfaces:

- **Entradas:** carga y preprocesamiento de datos en formatos como `.csv`, `.json`, `.xml` `.jpeg`, `.png`, etc ⇒ `print("¡Sistema Operativo!")`
- **Salidas:** predicciones, clasificaciones, generaciones de texto y detección de objetos ⇒ `print("¡Sistema Operativo!")`
- **Interfaces:** `usb`, `HDMI` (visualización de resultados), `wi-fi`, `jtag`, `SWD`,`ota`

Dimensiones

Con el fin de reducir el tamaño:

- **Componentes miniaturizados (SMD).**
- **Técnicas avanzadas de diseño de PCB.**
- **Semiconductores de última generación.**

Dimensiones **finales**:

$$\Rightarrow 70 \times 70 \times 10 \ [mm^3] \Leftarrow.$$

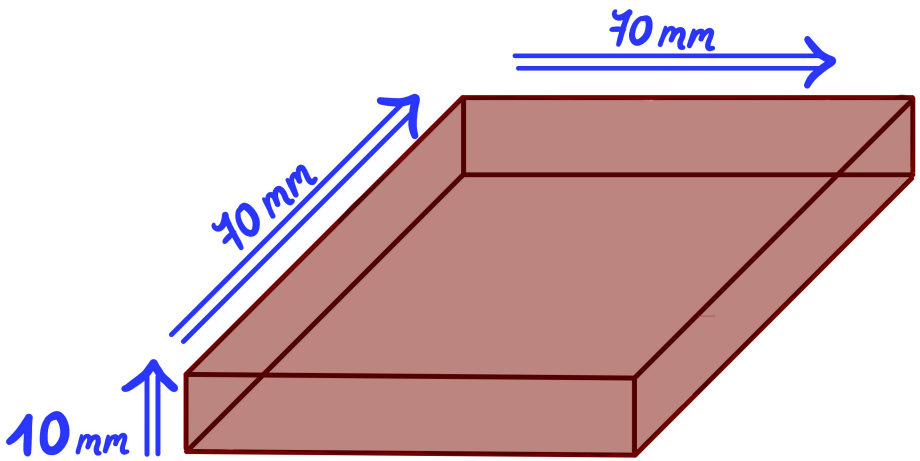


Figura: Para hacerse una idea.

¿Porque una **FPGA**?

Según Huwawei:

- **ALTO** grado de personalización.
- **ALTO** rendimiento computacional.
- **ALTA** η **(%)**.
- **BAJA** latencia.
- Coste **MEDIO**.

Tipo	**CPU**	**GPU**	**FPGA**	**ASIC**
Flexibilidad	gen	gen	semi-custom	custom
Rendimiento	bajo	medio	alto	alto
P_{cons}	medio	alto	bajo	bajo
LATENCIA	alta	media	baja	baja
€	alta	alta	media	baja
¿Aplicaciones?	Muchas	Muchas	¡Deep Learning!	+ o -

Cuadro: Tabla comparativa entre distintos tipos de implementaciones.

Consumo

- ¿Objetivo? $\Rightarrow \eta \uparrow$ y $P_{consumo} \downarrow$ por lo tanto:
  - Circuitos de baja potencia.
  - Componentes de hardware optimizado.
  - Optimización a nivel de software.
  - Sistemas de enfriamiento.
- **¿Consumo final?**
  - $P_{FPGA} \in \{6, 24 - 34, 7\}$ (mW)
  - $P_{CORE} \sim 250$ (mW)
- **Entonces** $\Rightarrow \boldsymbol{P_{total}} = 34, 7 + 4 \cdot 250 =$**1,035 W**
- **Pero...** y ¿los otros **módulos**?

$$\boldsymbol{P_{total,final}} = 1, 035 + (\sim 4) = \textbf{5 W}$$

¿€?

El coste aproximado entre 100 € y 200 € según la calidad de los componentes.
Fuente: www.mouser.es

Capas

- **Capa de aplicación:** software específico del dominio y la lógica de negocio.
- **Capa de sistema operativo:** entorno de ejecución, gestión de recursos, interfaz gráfica. `librerías`.
- **Arquitectura del Sistema:** `SRAM, DRAM, GPIO, módulos de cada interfaz` , así como su disposición.
- **Capa HW:**
  - Circuito figital: puertas lógicas, `MUXs`, etc.
  - Circuito físico: transistores, resistencias, etc.

Estructura HW

A continuación se adjuntan los **módulos** de la **CPU** y del **ACELERADOR**:

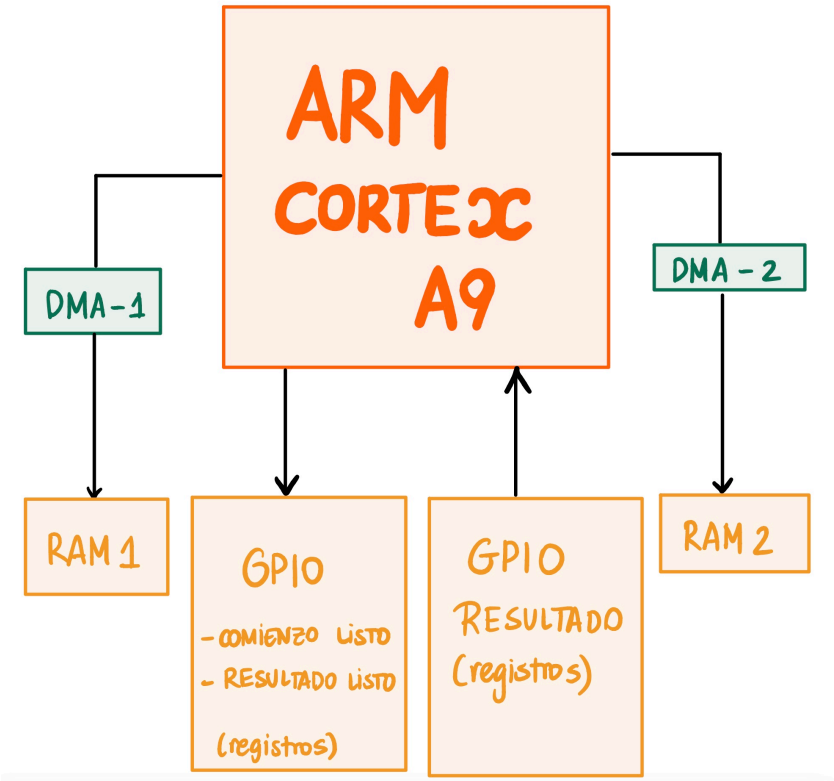


Figura: **CPU**

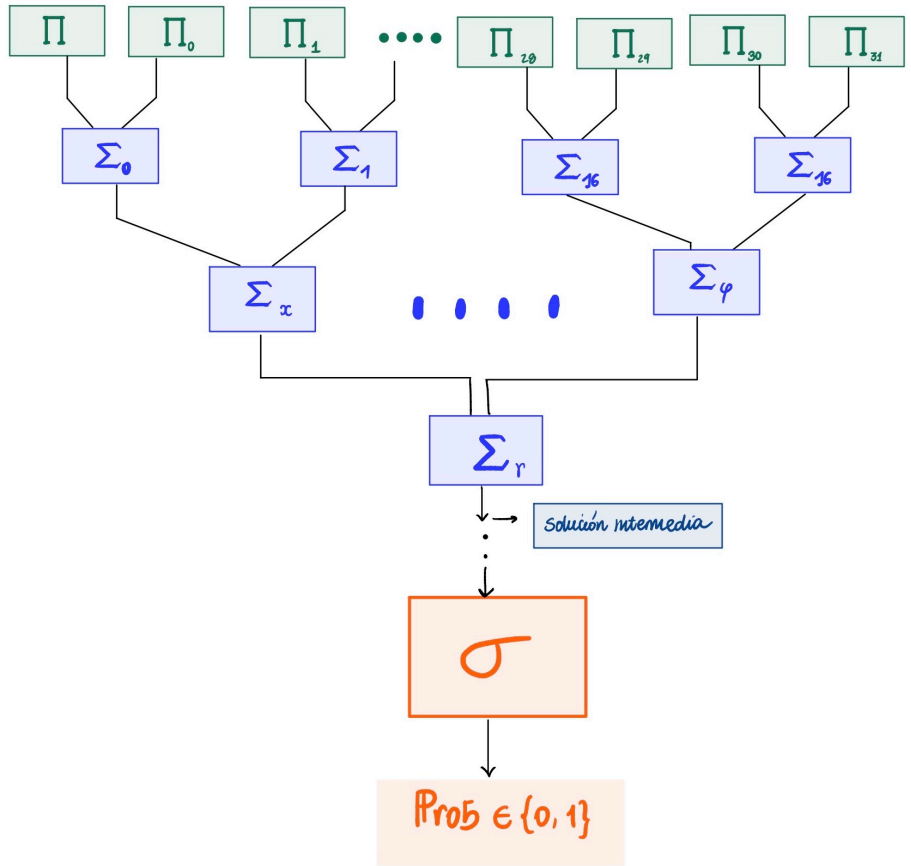


Figura: **ACELERADOR**

Diagrama de bloques

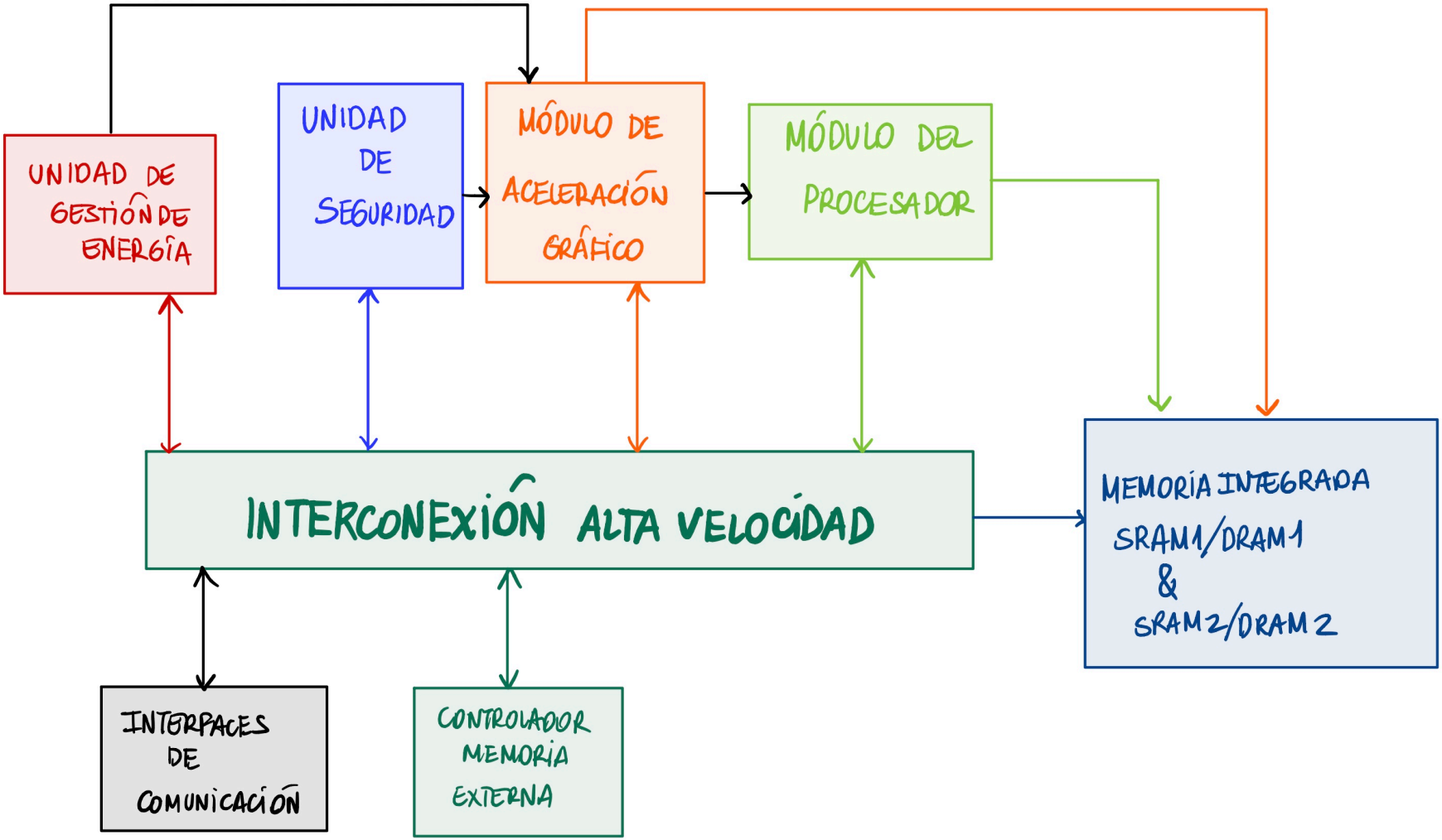


Figura: Diagrama de bloques.

Relación entre **CPU** y **ACELERADOR**

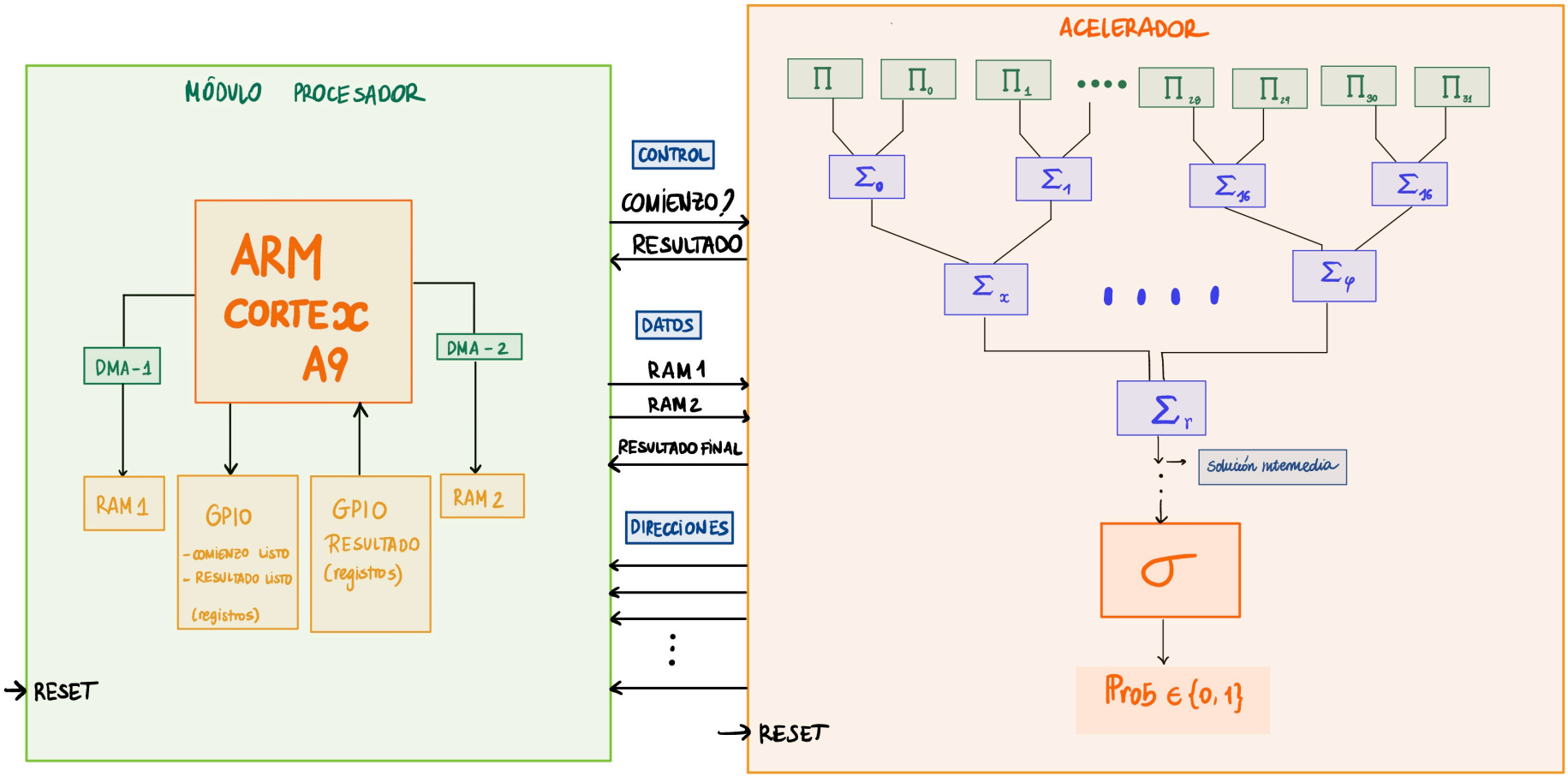


Figura: Interconexionado entre los dos módulos.

Estrategia por capas

- **¿Objetivo inicial?**
  - **OPTIMIZACIÓN** $\in$ **capa SW** i.e., ~~**Python**~~ $\rightarrow$ **C/C++**
  - **PÉRDIDA de funcionalidades**: `keras`, `numpy`, `pandas`, etc.
- **¿Implementación final?**
  - **OPTIMIZACIÓN a bajo nivel** i.e., HW SENCILLO y EFICAZ.
  - ~~**C/C++**~~ $\rightarrow$ **Python**

COHERENCIA

Modos de operación

Los modos de funcionamiento contemplados para este diseño son los siguientes;

Modo	**Acción $\in$ procesador**	**RAM**	**Wake-Up mech**
Run Mode	"encendido" y v_{normal}	"encendido"	-
Standby	"encendido" v_{normal}[1]	"encendido"	depende
Adormecido	$(v \downarrow)$ pero `CACHES ON`	keep states	Acción externa
Apagado	apagado	apagado	Acción externa

Cuadro: Modos de operación del `ARM Cortex-A9`

- El MODO predominante $\Rightarrow$ *Standby*
- ¿Para entrenamiento? $\Rightarrow$ *Run Mode*

[1] se deshabilitan la mayoría de los clocks y v_{normal} solo cuando se despierta

Gracias