

Sistemas Empotrados Distribuidos

Tema 3: Arquitectura HW de un Sistema
Empotrado Distribuido

Prof. Alberto A. Del Barrio
Prof. Guillermo Botella

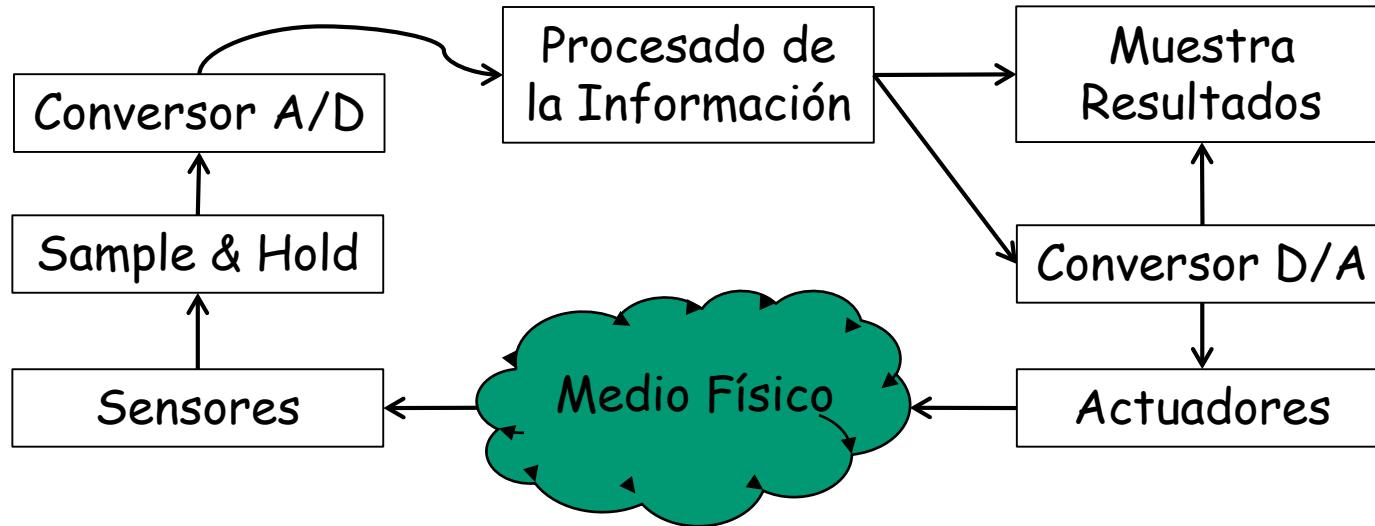


DEPARTAMENTO DE
ARQUITECTURA DE COMPUTADORES
Y AUTOMÁTICA

Curso 2022-2023

- Introducción
- Periféricos: E/S
- Procesando los datos
 - Procesadores
 - Microcontroladores
 - FPGAs
 - ASICs
 - MPSoCs
- Comunicación
 - Buses
 - Wireless
- Bibliografía
 - P. Marwedel, "Embedded System Design" [Ch. 3]

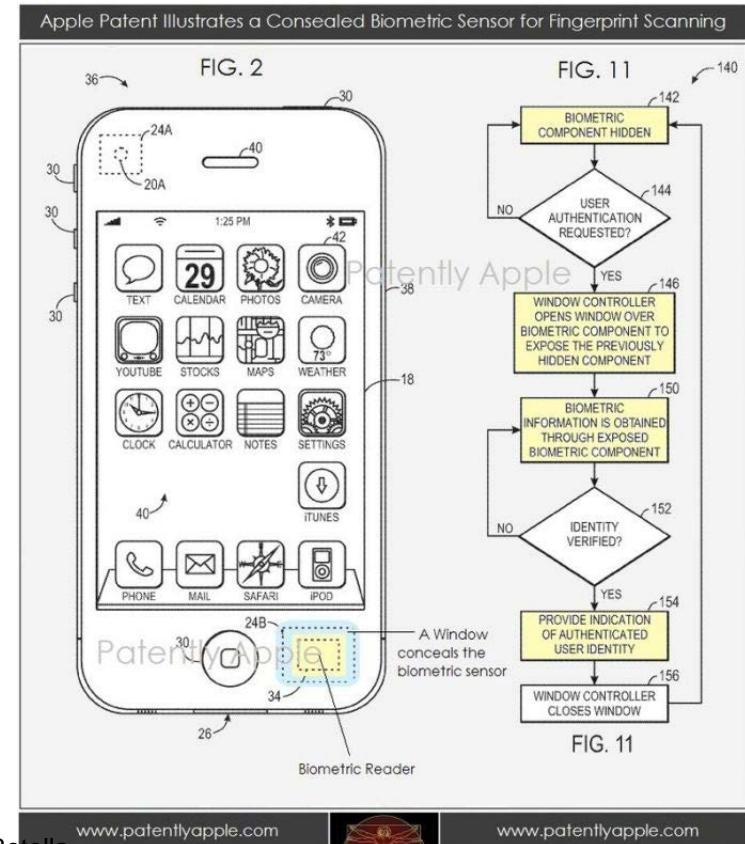
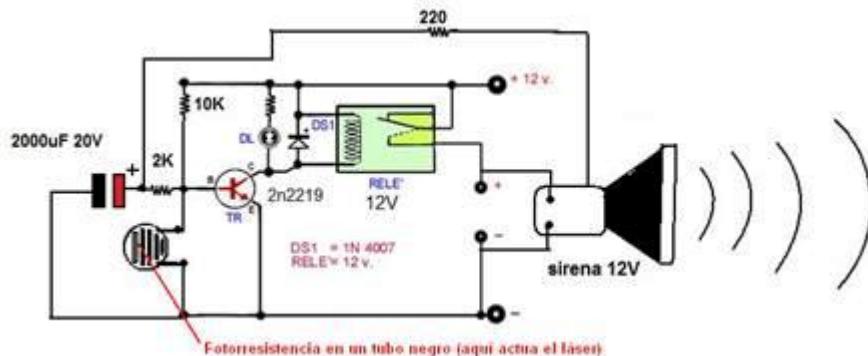
- $SED = HW + SW$
- La base es la reutilización de componentes
- Arquitectura básica de un Sistema Empotrado genérico



Periféricos: Entrada

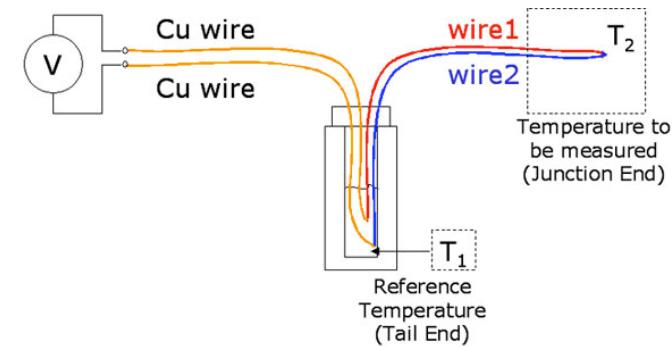
□ Sensor

- Dispositivo que detecta una determinada acción externa, temperatura, presión, etc., y la transmite adecuadamente
- Sensores de temperatura, movimiento, lluvia, RFID, biométricos, de visión



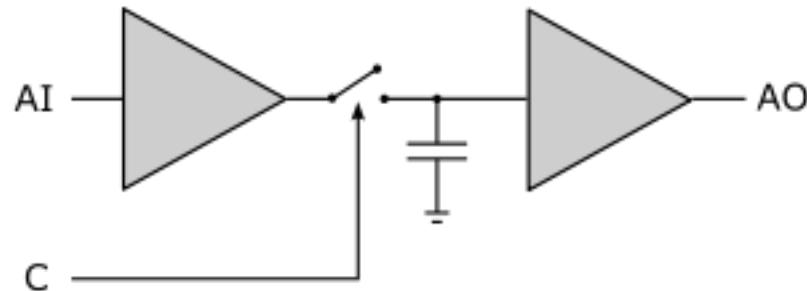
□ Transductor

- Dispositivo que transforma el efecto de una causa física, como la presión, la temperatura, la dilatación, la humedad, etc., en otro tipo de señal, normalmente eléctrica.
- Analógico o digital
 - Ej. Termistor, termopar, dinamo



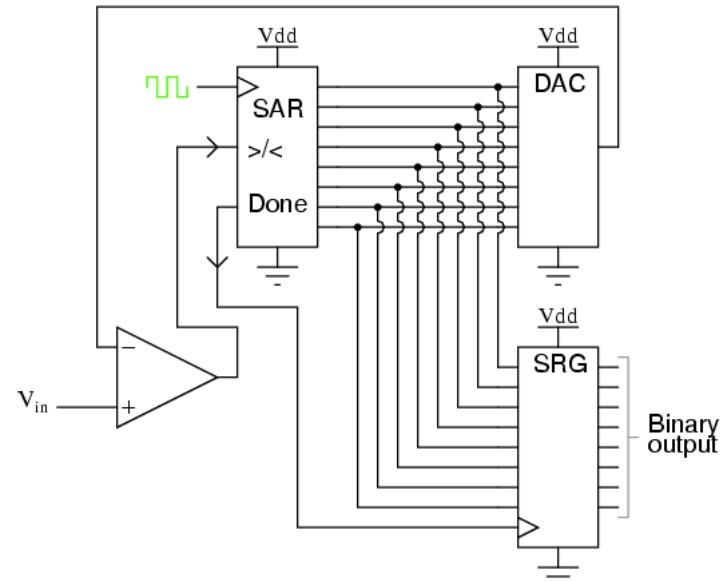
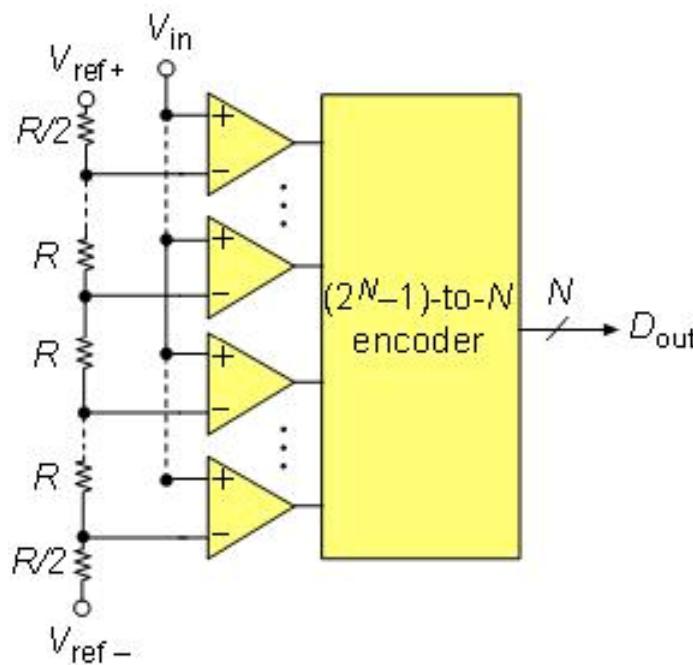
□ Sample & Hold

- Discretiza la información analógica
- No digitaliza
- Transformada de Fourier
- Muestrea de acuerdo a la frecuencia de Nyquist
 - La frecuencia de muestreo tiene que ser al menos 2 veces mayor que la frecuencia más rápida



Periféricos: Entrada

- Conversor Analógico/Digital (ADC)
 - Transforma la información analógica a bits ('0's ó '1's)
 - Múltiples diseños
 - ADC Flash. Varios niveles de referencia. Rápido y sin reloj, pero costoso en HW
 - ADC por Aproximaciones Sucesivas. Circuito secuencial de búsqueda binaria ($\log(n)$). Más barata en HW, no tan rápida como la Flash

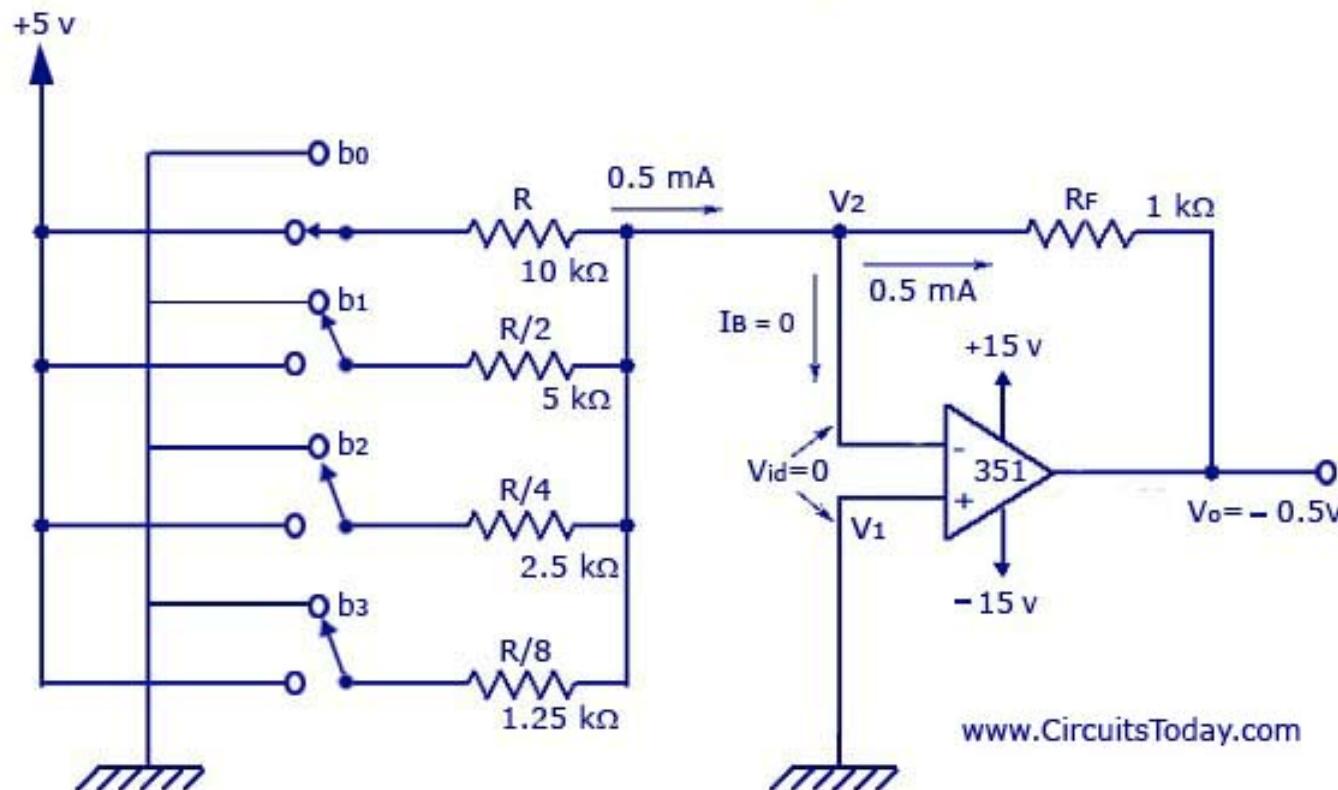


Periféricos: Salida

□ Conversor Digital/Analógico (DAC)

- Transforma la información digital (bits) en una señal analógica (continua)

D/A Converter With Binary Weighted Resistors

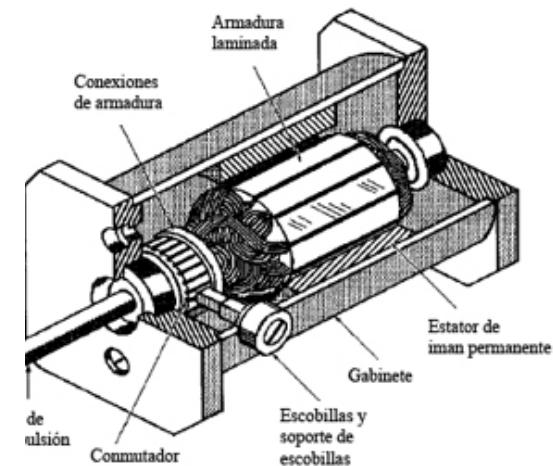
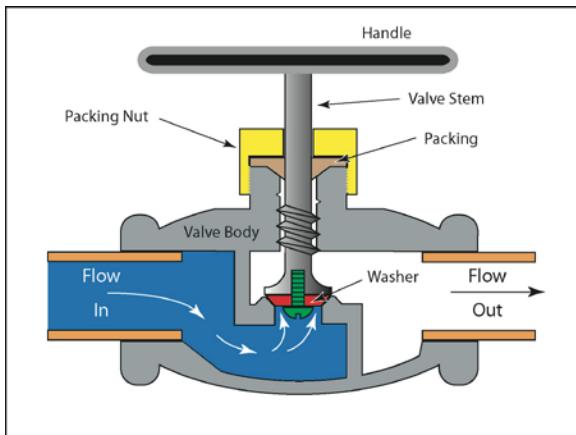


www.CircuitsToday.com

Periféricos: Salida

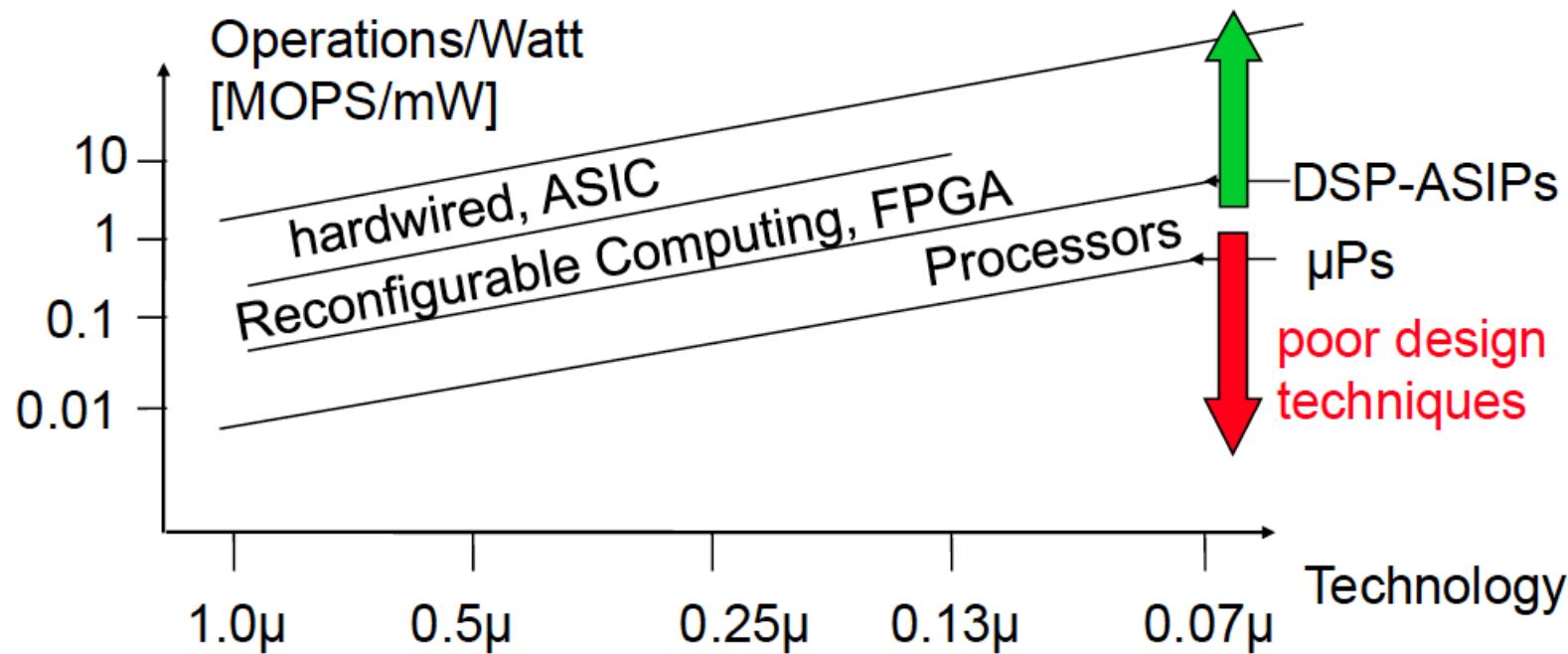
□ Actuador

- Elementos finales que permiten modificar las variables a controlar en una instalación automatizada
 - Ej. Válvulas, relés, servomotores



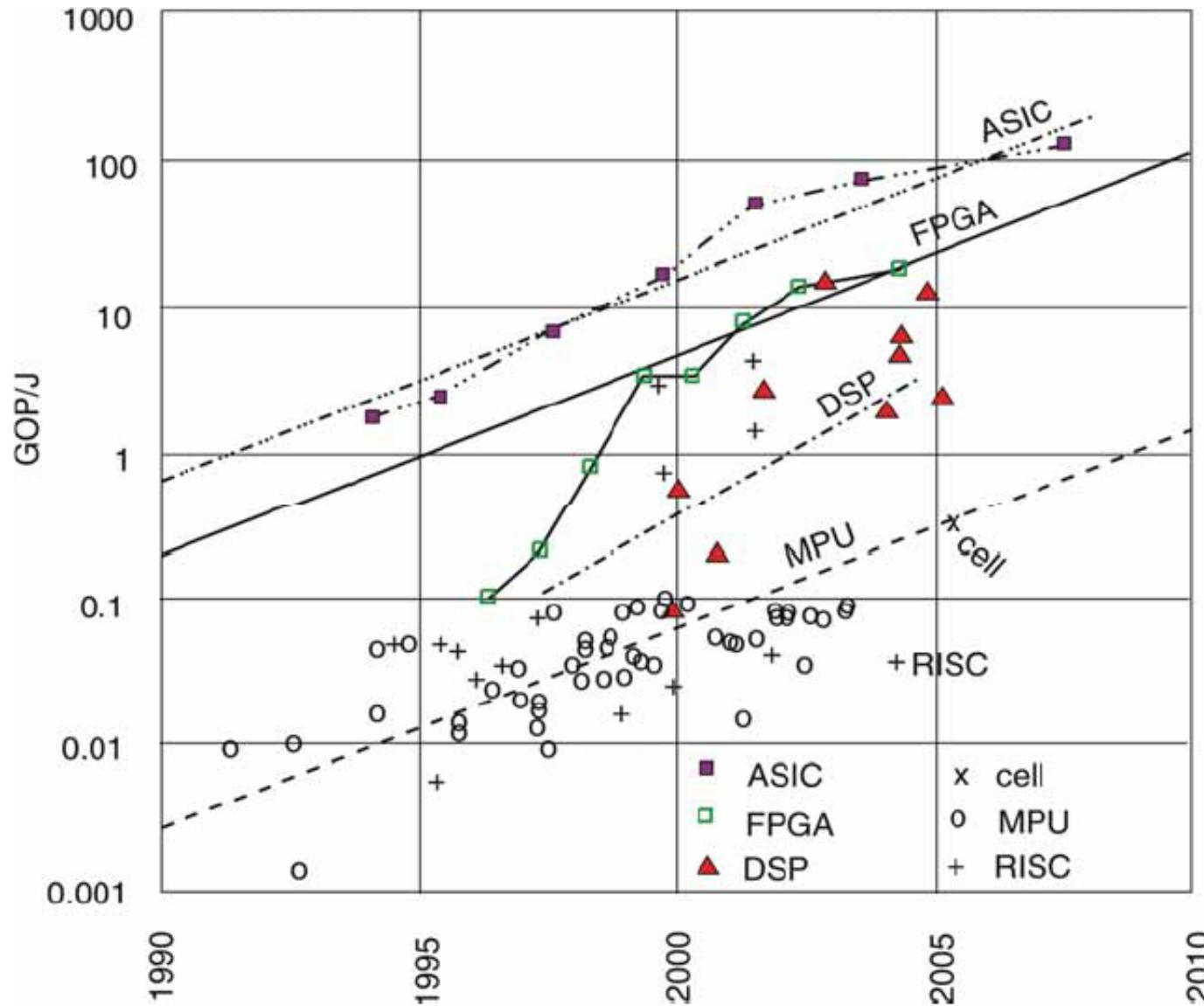
Procesando los datos

- Los SEDs son sistemas heterogéneos
 - Power is considered as the most important constraint in embedded systems [Eggermont, 2002]
- Múltiples elementos de proceso de datos con diferente tradeoff rendimiento/energía y rendimiento/potencia



Procesando los datos

Cuanto más específico es el HW, tiene mayor eficiencia, ¿pero por qué seguimos utilizando procesadores entonces ?

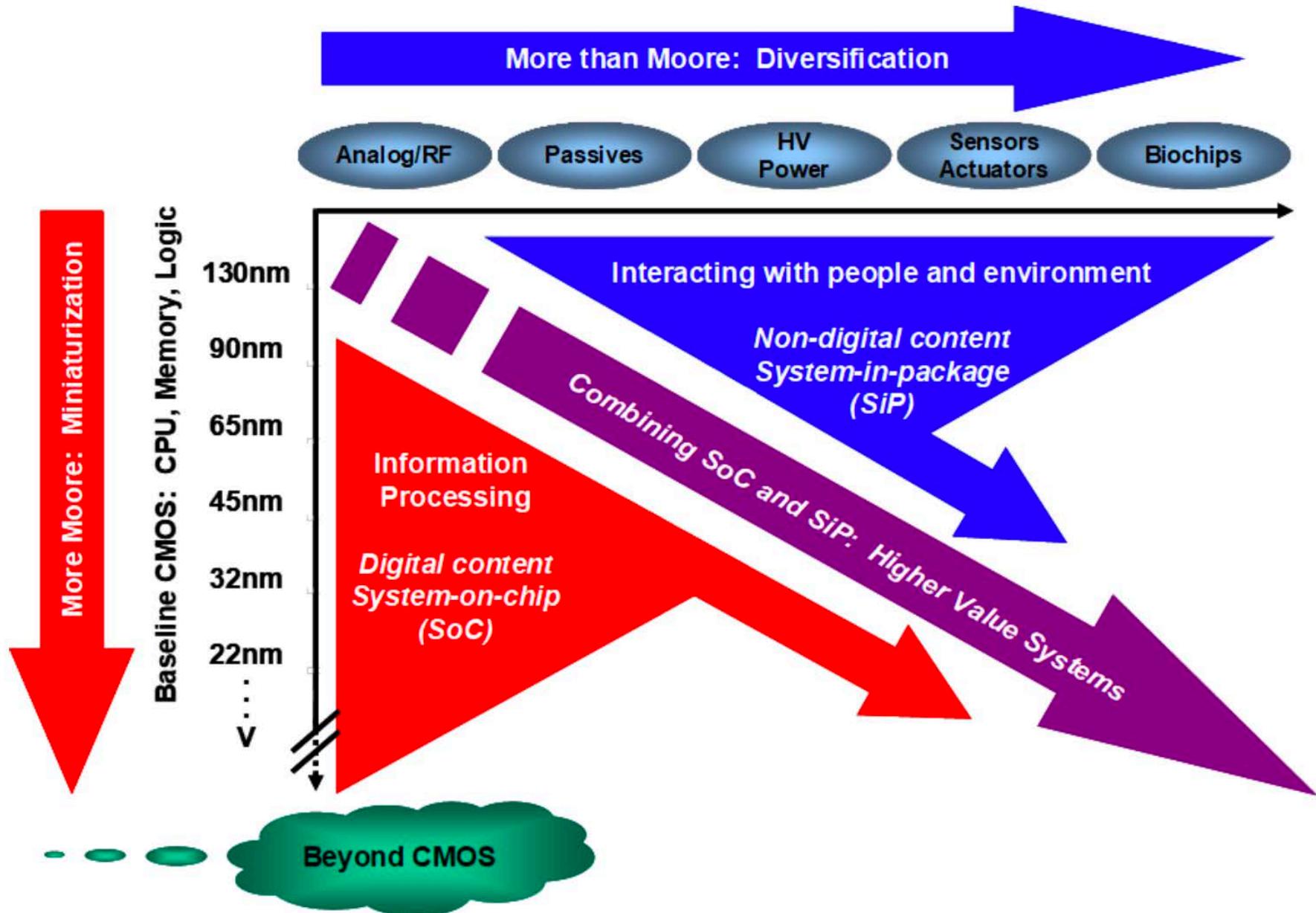


© Hugo De Man,
IMEC, Philips, 2007

- An historical observation by Gordon Moore is that the market demand (and semiconductor industry response) for functionality per chip (bits, transistors) doubles every 1.5 to 2 years.
- He also observed that MPU performance [clock frequency (MHz) * instructions per clock = millions of instructions per second (MIPS)] also doubles every 1.5 to 2 years. Although viewed by some as a "self-fulfilling" prophecy, Moore's Law has been a consistent macro trend and key indicator of successful leading-edge semiconductor products and companies for the past 40 years.

- "More Moore" refers to the continued shrinking of physical feature sizes of the digital functionalities (logic and memory storage) in order to improve density (cost per function reduction) and performance (speed, power).
- "More than Moore" (MtM) refers to the incorporation into devices of functionalities that do not necessarily scale according to Moore's Law, but provide additional value in different ways. The More-than-Moore approach allows for the non-digital functionalities (e.g., RF communication, power control, passive components, sensors, actuators) to migrate from the system board-level into the package (SiP) or onto the chip (SoC).

Moore & More than Moore (IRDS)



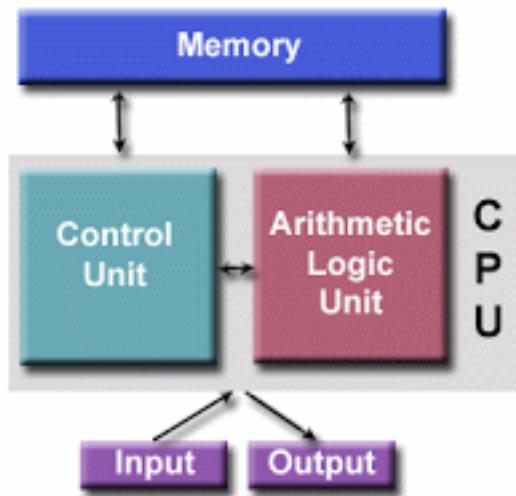
- Arquitecturas
 - Von Neumann
 - Harvard
- Procesadores de bajo consumo
 - VLIW
 - DSP
- Técnicas de bajo consumo
 - RISC vs CISC
 - DVFS
 - Power Gating
 - Power Management
 - SIMD
 - Code compression

Procesando los datos: Procesadores

□ Arquitectura Von Neumann

○ Elementos

- Unidad de Proceso de Datos
- Memoria
- Unidad de Control
- Periféricos

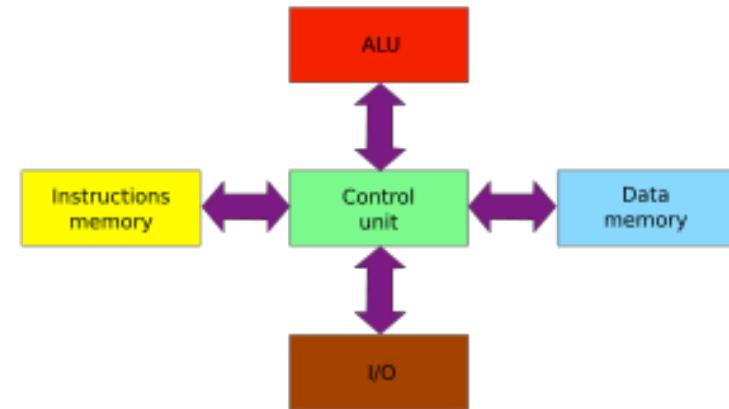


□ Arquitectura Harvard

- Distinto espacio de direcciones de memoria para instrucciones y datos

□ Harvard Modificada

- Cache Harvard
- Niveles altos de la jerarquía de memoria: Von Neumann

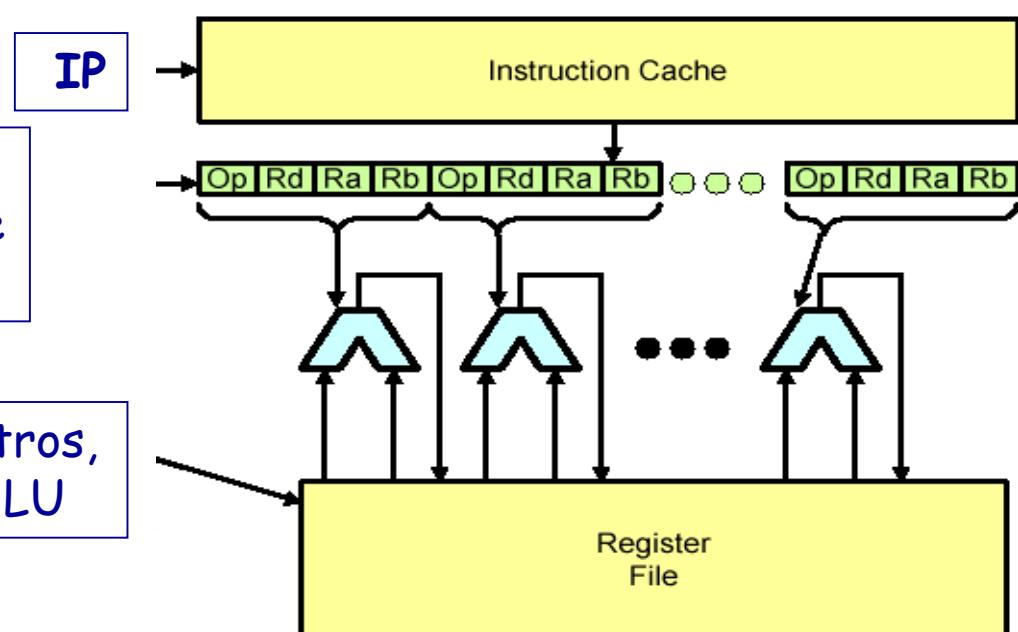


Procesadores: Very Long Instruction Word (VLIW)

- El análisis de dependencias en tiempo de compilación
- Muchas operaciones por instrucción
- Todas las operaciones de una instrucción se ejecutan en paralelo
- Instrucciones con muchos bits

Instrucción: Incluye varias instrucciones convencionales de tres operandos una por ALU

Bloque de registros,
3 puertos por ALU



Procesadores: Very Long Instruction Word (VLIW)

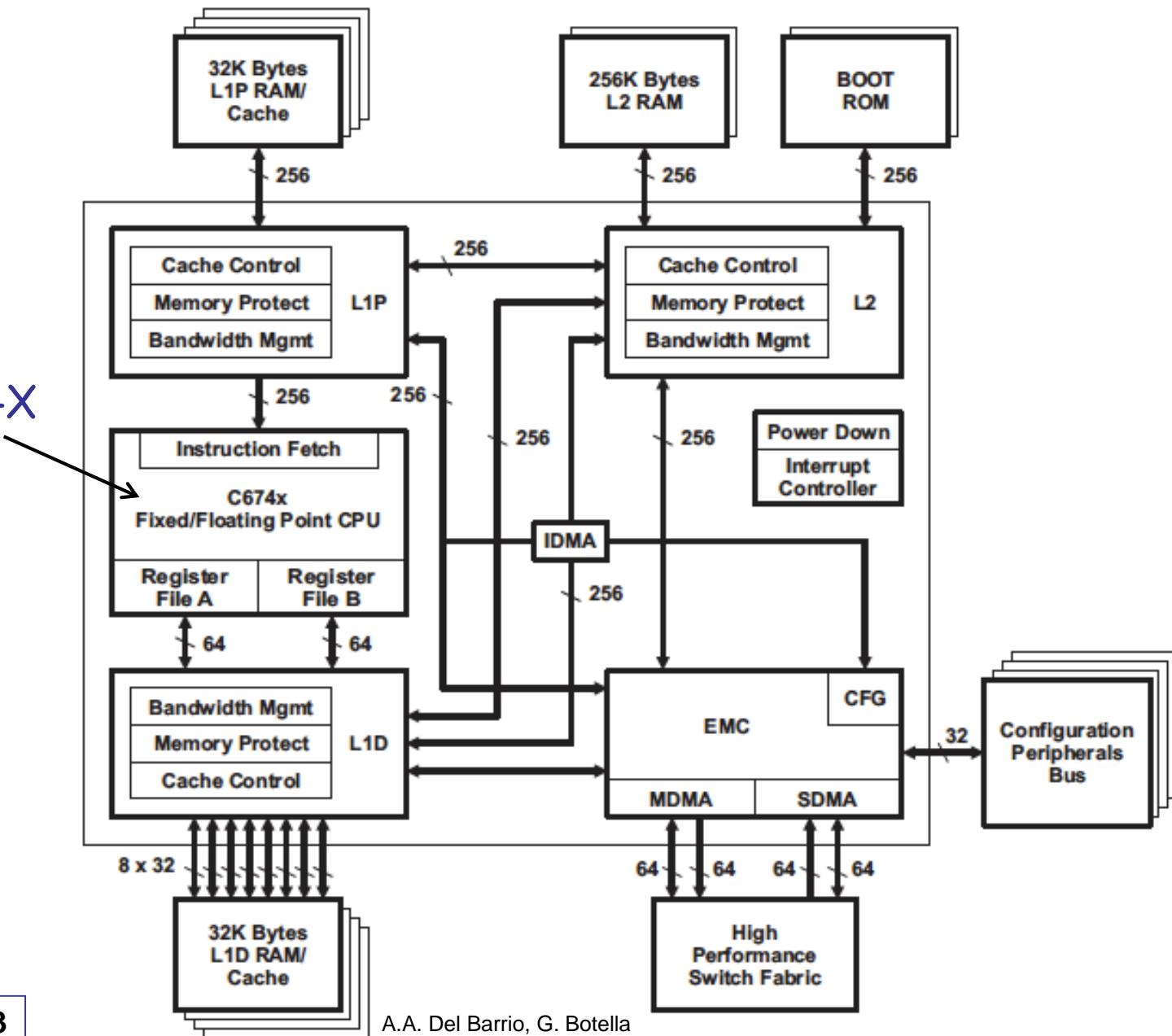
- Ventajas:
 - Hardware de control muy simple
 - No detecta dependencias
 - Lógica de lanzamiento simple
 - Puede explotar paralelismo a todo lo largo del programa
- Desventajas:
 - Planificación estática. Muy sensible a fallos de cache
 - Cargas especulativas
 - Necesita desenrollado muy agresivo
 - Bloque de registros muy complejo en área y tiempo de acceso
 - Muchas NOP
 - Poca densidad de código
 - Capacidad y AB de la cache de instrucciones
 - Compilador muy complejo
 - No binario compatible
 - Operación síncrona para todas las operaciones de una instrucción

Procesadores: Digital Signal Processors (DSPs)

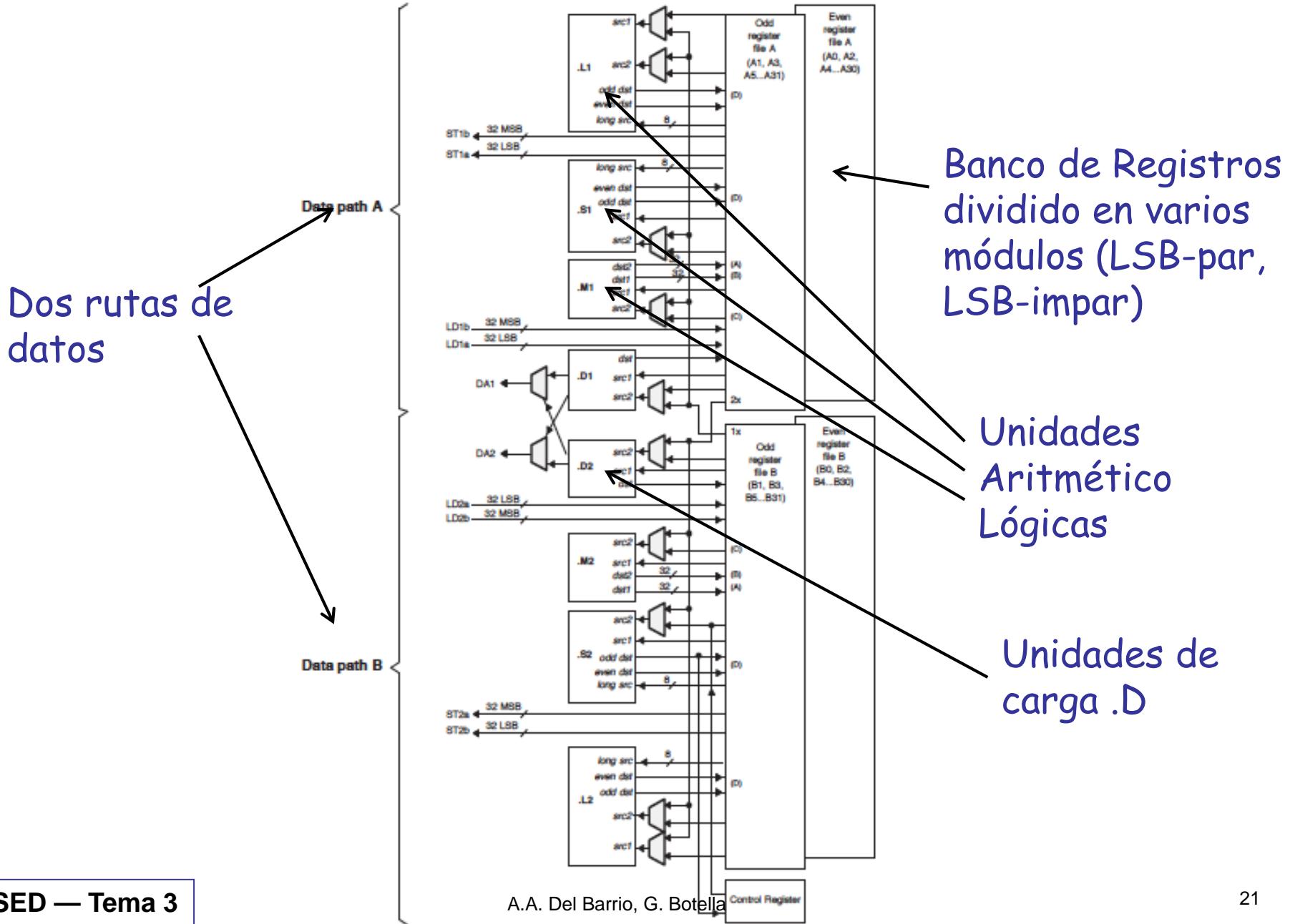
- Procesadores específicamente diseñados para procesar señal
- Orientados a Real-Time
- Optimizaciones de memoria
 - Arquitectura Harvard
 - Memorias de datos y programa separadas
 - Fetch múltiple de datos e instrucciones
 - Unidades de Cálculo de Direcciones dedicadas
 - DMA
- Optimizaciones de datos
 - Soporte SIMD
 - Aritmética saturada
 - Punto fijo
 - MAC
 - HW específico para loops
 - Instrucciones específicas para direccionamientos típicos: modulo, bit-reversed (FFT), etc.

DSPs, un ejemplo: C674X

Core C674X

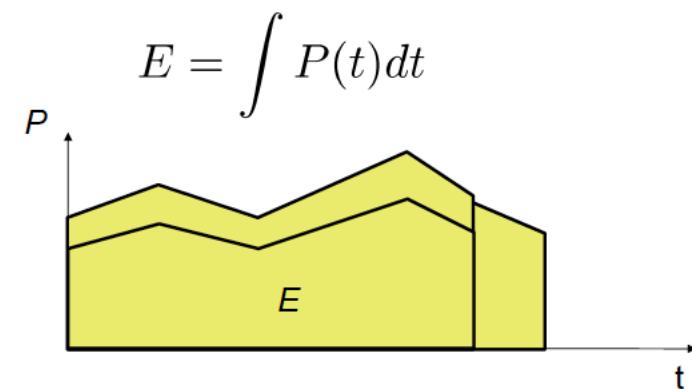


DSPs, un ejemplo: C674X



Procesadores: Técnicas de bajo consumo

- Potencia y Energía no son lo mismo
- Potencia = Energía por unidad de tiempo
- Disminuir potencia es importante para ...
 - Diseño de la fuente de alimentación
 - Dimensión interconexiones
 - Refrigeración (cooling)
 - Hotspots
- Disminuir la energía es importante por ...
 - Capacidad limitada de la batería
 - Precio del kW*h
 - Longevidad del dispositivo

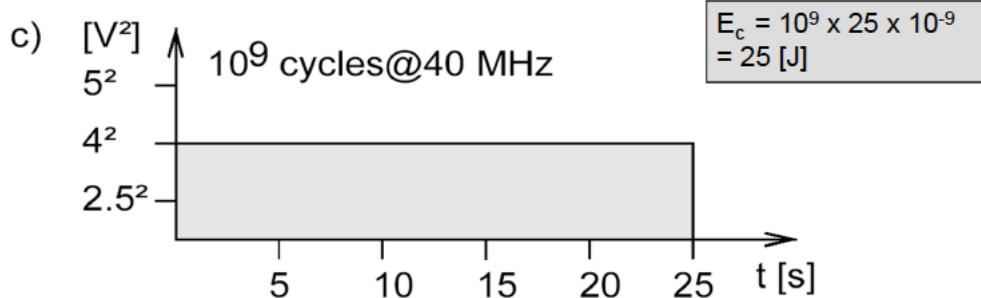
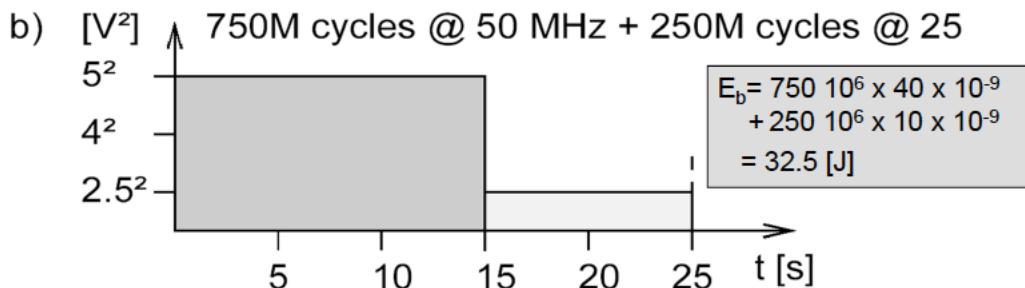
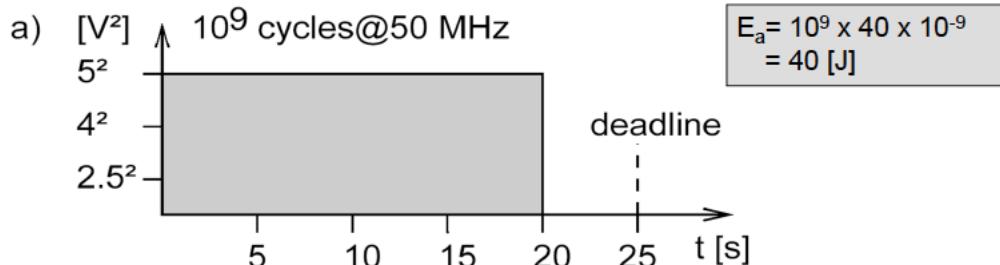


- DVFS
 - En muchas ocasiones el V y la f máximos no son necesarios
- Power Gating
 - Los módulos no utilizados pueden apagarse
- Power Management
 - Hay módulos que tienen distintos niveles de consumo. Ej. Memorias (idle, active, low power, etc)
- SIMD
 - La mayoría de los datos son más pequeños que el ancho máximo del procesador
 - Aplicaciones multimedia
- Code compression
 - Fetches más baratos

Técnicas de bajo consumo, un ejemplo: DVFS

V_{dd} [V]	5.0	4.0	2.5
Energy per cycle [nJ]	40	25	10
f_{max} [MHz]	50	40	25
cycle time [ns]	20	25	40

Task that needs to execute 10^9 cycles within 25 seconds.



La potencia máxima no siempre es necesaria

Si el deadline lo permite, minimizaremos el consumo de energía

□ RISC vs CISC

- Debate abierto: ISA Wars [1,2]
 - x86 (CISC) vs ARM (RISC): decisión comercial
- Microcontroladores sencillos → RISC
- Nodos potentes → Menor P con RISC, en términos de E no hay nada probado
- VLIW son CISC
 - VLIW+DSP es bastante común [3,4]
- ARM (Acorn RISC Machine) son RISC
 - ARM en casi todos los dispositivos empotrados [1,2]

[1] Emily Blem, Jaikrishnan Menon, Thiruvengadam Vijayaraghavan, and Karthikeyan Sankaralingam. 2015. ISA Wars: Understanding the Relevance of ISA being RISC or CISC to Performance, Power, and Energy on Modern Architectures. *ACM Trans. Comput. Syst.* 33, 1, Article 3 (March 2015), 34 pages. DOI: <https://doi.org/10.1145/2699682>

[2] E. Blem, J. Menon and K. Sankaralingam, "Power struggles: Revisiting the RISC vs. CISC debate on contemporary ARM and x86 architectures," 2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA), Shenzhen, 2013, pp. 1-12. doi: 10.1109/HPCA.2013.6522302

[3] M. E. A. Ibrahim, M. Rupp and H. A. H. Fahmy, "Power estimation methodology for VLIW Digital Signal Processors," 2008 42nd Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, 2008, pp. 1840-1844. doi: 10.1109/ACSSC.2008.5074746

[4] Chan-Hao Chang and D. Marculescu, "Design and analysis of a low power VLIW DSP core," IEEE Computer Society Annual Symposium on Emerging VLSI Technologies and Architectures (ISVLSI'06), Karlsruhe, 2006, pp. 6 pp.-. doi: 10.1109/ISVLSI.2006.36

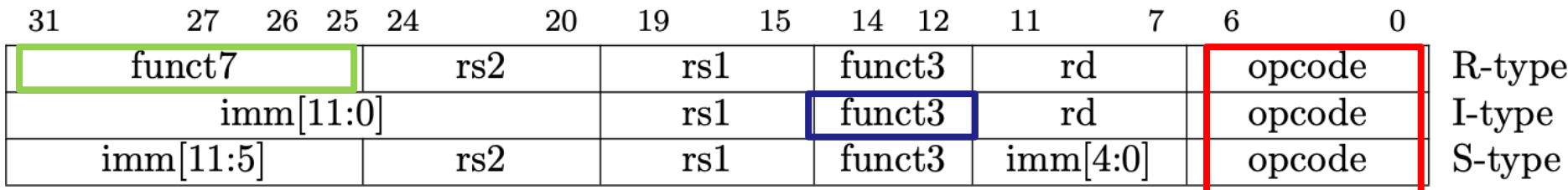
- El proyecto RISC-V nace en 2010 en el Parallel Computing Laboratory (UC Berkeley), dirigido por David Patterson
- El proyecto lo lidera Krste Asanovic
 - Basado sobre todo en las tesis de Andrew Waterman y Yunsup Lee
 - RISC-V es una ISA abierta, de sintaxis sencilla, pensada para ser customizada
- En 2015 nace la RISC-V Foundation
 - AMD, Google, NVIDIA, Microsoft, Qualcomm, Oracle ... y la UCM
- En 2015, Asanovic funda SiFive, primera compañía en fabricar chips basados en RISC-V
- Actualmente hay 89 cores/SoCs basados en RISC-V
 - Rocket, BOOM, CVA6 (Ariane), Shakti son los más utilizados [1]
 - Orientados a IoT: SWERV (WD), Ibex (lowRISC/ETH) ...

[1] Alexander Dörflinger, Mark Albers, Benedikt Kleinbeck, Yejun Guan, Harald Michalik, Raphael Klink, Christopher Blochwitz, Anouar Nechi, and Mladen Berekovic. 2021. A comparative survey of open-source application-class RISC-V processor implementations. In Proceedings of the 18th ACM International Conference on Computing Frontiers (CF '21). Association for Computing Machinery, New York, NY, USA, 12–20.

RISC-V: ejemplo de customización

```
1 # Posit extension
2 # opcode=Custom0  fmt=2 (32 bit (single) precision)
3 # Single-precision posit computational and quire instructions
4 padd.s          rd rs1 rs2 31..27=0 26..25=2      14..12=0 6..2=0x02 1..0=3
5 psub.s          rd rs1 rs2 31..27=1 26..25=2      14..12=0 6..2=0x02 1..0=3
6 pmul.s          rd rs1 rs2 31..27=2 26..25=2      14..12=0 6..2=0x02 1..0=3
7 pdiv.s          rd rs1 rs2 31..27=3 26..25=2      14..12=0 6..2=0x02 1..0=3
8 psqrt.s         rd rs1     31..27=6 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
9 pmin.s          rd rs1 rs2 31..27=4 26..25=2      14..12=0 6..2=0x02 1..0=3
10 pmax.s         rd rs1 rs2 31..27=5 26..25=2      14..12=0 6..2=0x02 1..0=3
11
12 qmadd.s        rs1 rs2   31..27=7 26..25=2      14..12=0 11..7=0 6..2=0x02 1..0=3
13 qmsub.s        rs1 rs2   31..27=8 26..25=2      14..12=0 11..7=0 6..2=0x02 1..0=3
14 qclr.s         31..27=9 26..25=2 24..15=0 14..12=0 11..7=0 6..2=0x02 1..0=3
15 qneg.s         31..27=10 26..25=2 24..15=0 14..12=0 11..7=0 6..2=0x02 1..0=3
16 qround.s       rd       31..27=11 26..25=2 24..15=0 14..12=0      6..2=0x02 1..0=3
17
18 # Single-precision posit/integer conversion and move instructions
19 pcvt.w.s       rd rs1   31..27=12 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
20 pcvt.wu.s      rd rs1   31..27=13 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
21 pcvt.l.s       rd rs1   31..27=14 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
22 pcvt.lu.s      rd rs1   31..27=15 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
23 pcvt.s.w       rd rs1   31..27=16 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
24 pcvt.s.wu      rd rs1   31..27=17 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
25 pcvt.s.l       rd rs1   31..27=18 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
26 pcvt.s.lu      rd rs1   31..27=19 26..25=2 24..20=0 14..12=0 6..2=0x02 1..0=3
27
28 psgnj.s        rd rs1 rs2 31..27=20 26..25=2 14..12=0 6..2=0x2 1..0=3
29 psgnjn.s       rd rs1 rs2 31..27=21 26..25=2 14..12=0 6..2=0x2 1..0=3
30 psgnjx.s       rd rs1 rs2 31..27=22 26..25=2 14..12=0 6..2=0x2 1..0=3
31
32 # Single-precision posit compare instructions
33 peq.s          rd rs1 rs2 31..27=23 26..25=2 14..12=0 6..2=0x2 1..0=3
34 plt.s          rd rs1 rs2 31..27=24 26..25=2 14..12=0 6..2=0x2 1..0=3
35 ple.s          rd rs1 rs2 31..27=25 26..25=2 14..12=0 6..2=0x2 1..0=3
36
37 # Single-precision and quire load and store instructions
38 plw            rd rs1     imm12 14..12=1 6..2=0x02 1..0=3
39 ql             rd rs1     imm12 14..12=2 6..2=0x02 1..0=3
40 psw            imm12hi rs1 rs2  imm12lo 14..12=3 6..2=0x02 1..0=3
41 qs             imm12hi rs1 rs2  imm12lo 14..12=4 6..2=0x02 1..0=3
```

RISC-V: formato de instrucciones



RV64I

Instruction LS bits [6:0] = c c b b b a a

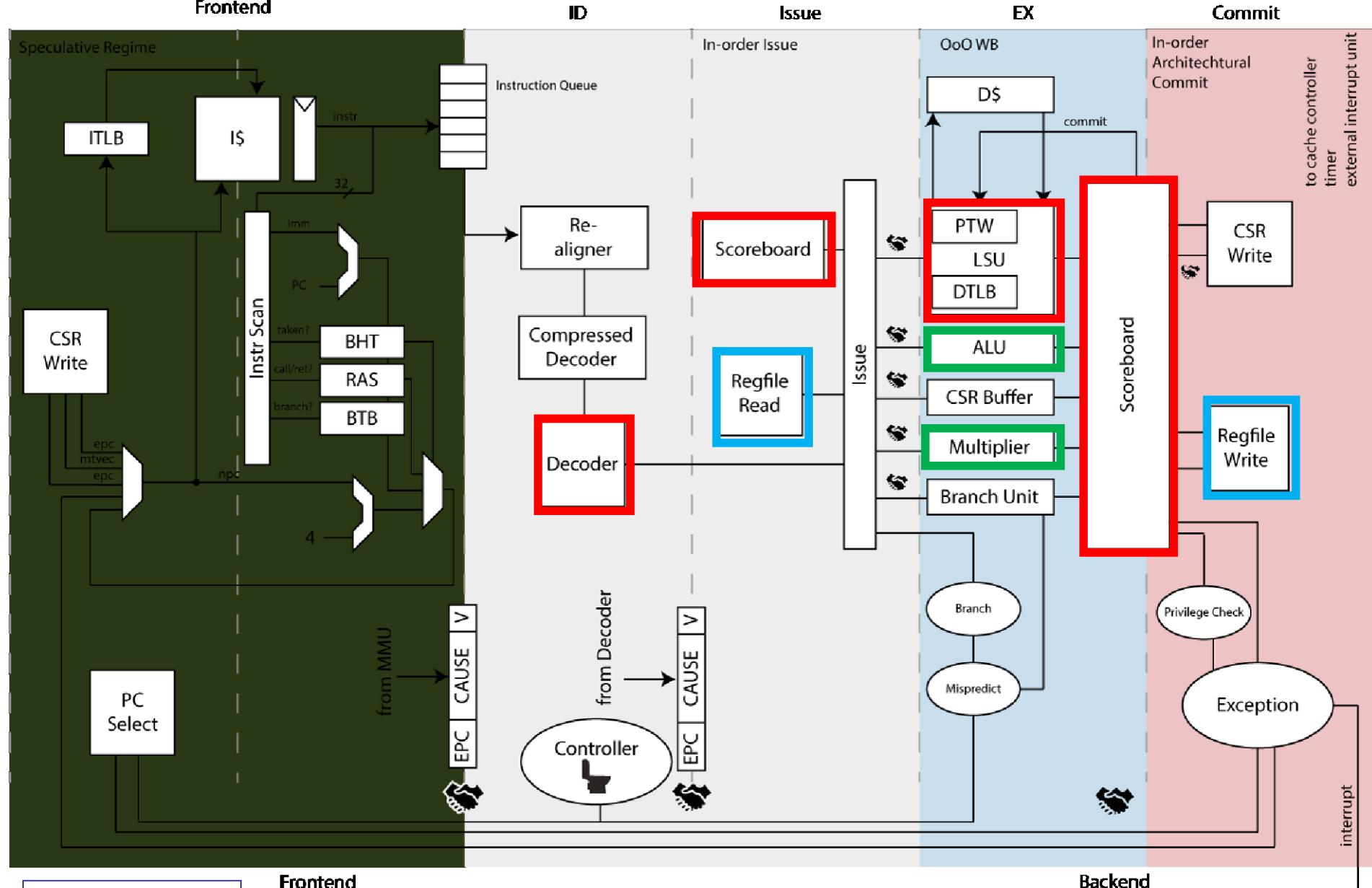
a a = x11 for standard
integer base opcodes

Instruction bits [4:2] (b b b)

	000	001	010	011	100	101	110	111
00	green	green	yellow	green	green	green	grey	
01	green	green	yellow	green	green	green	grey	
10	green	green	green	green	light blue	pink	grey	
11	green	green	light blue	green	light blue	pink	grey	

- Available for custom use
- Reserved for RV128I
- Reserved for future ext.
- Used by RV32G/RV64G
- For instructions >32-bit

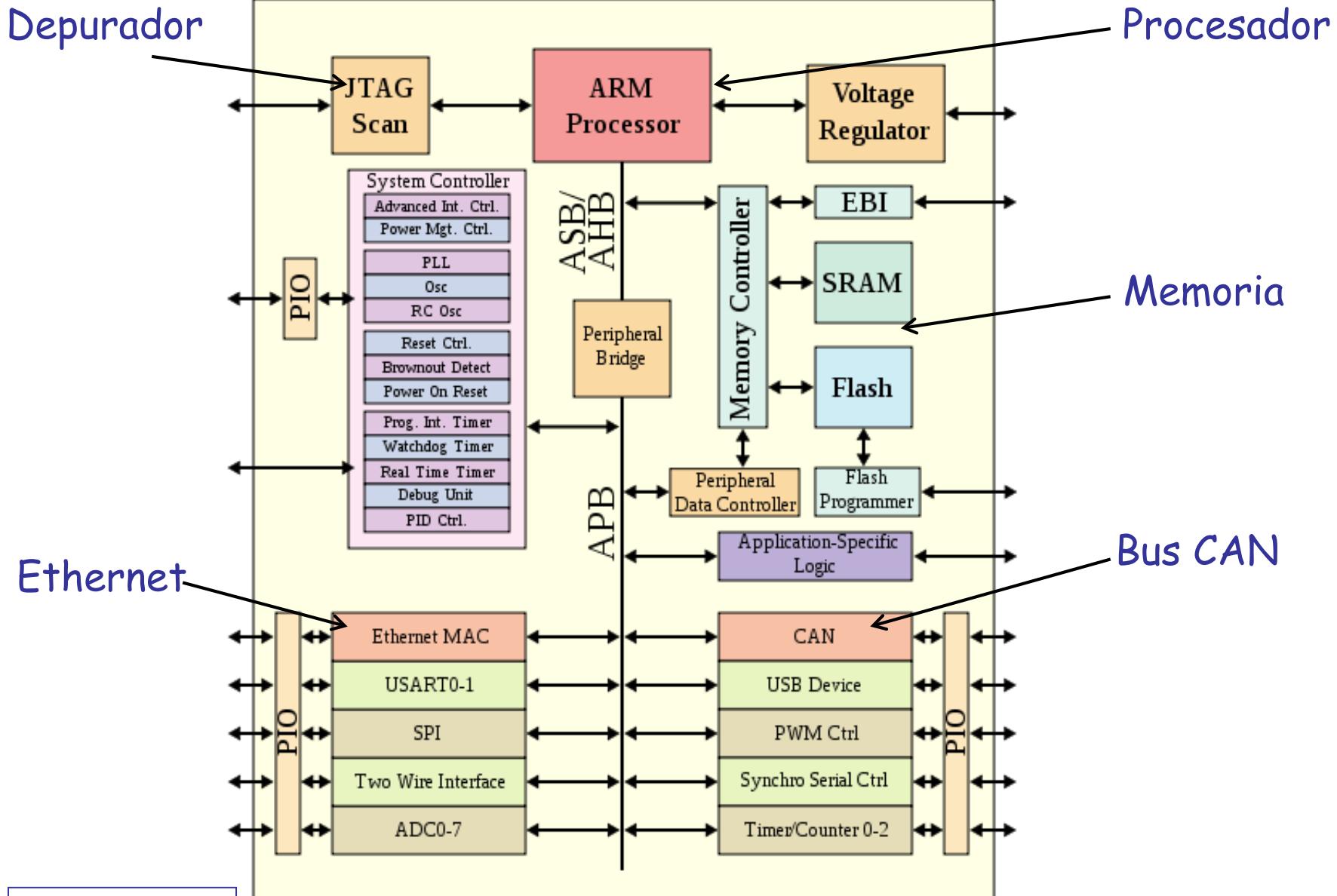
RISC-V: ejemplo de customización



Procesando los datos: Microcontroladores

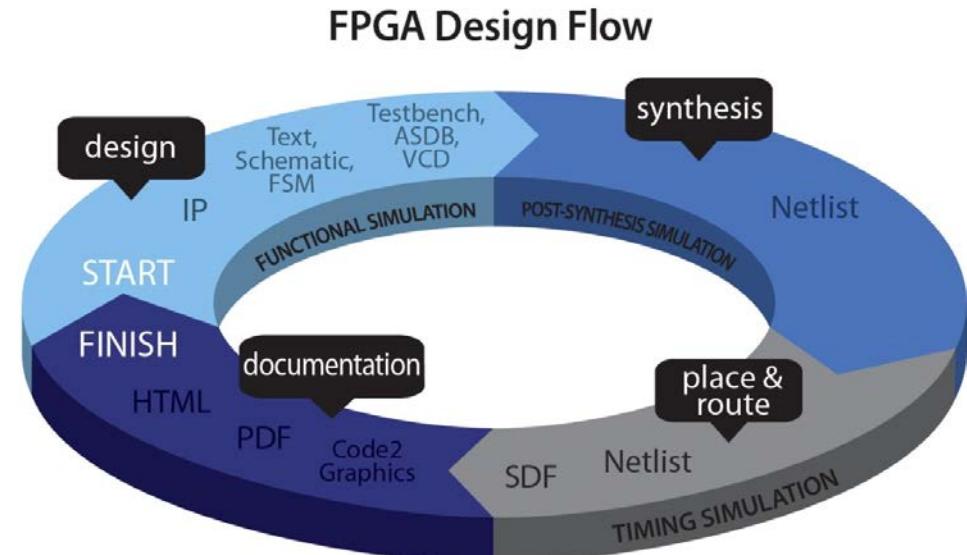
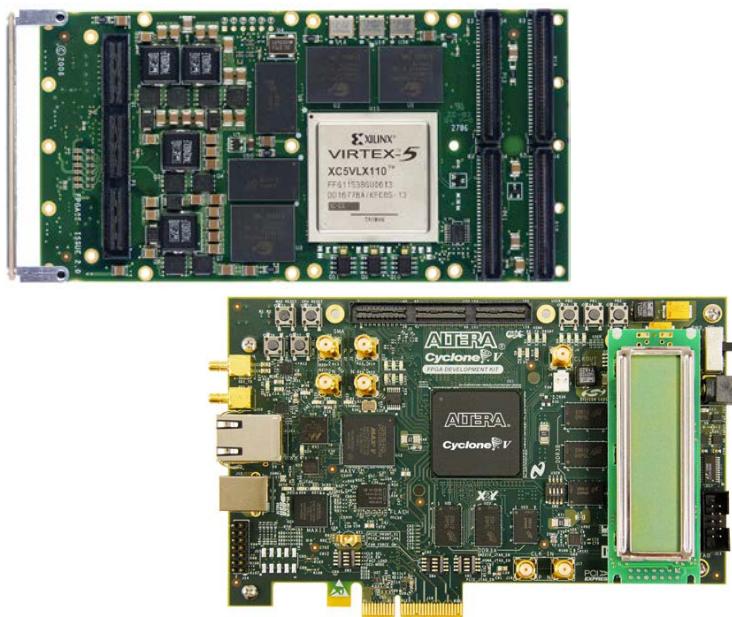
- Microcontrolador (uC)
 - Circuito empotrado que contiene varios elementos
 - CPU de bajo consumo
 - Memoria RAM/ROM/Flash
 - Periféricos
 - System On Chip
- No confundir con microprocesador
 - uC
 - Diseñado con propósito específico
 - Baja frecuencia (en el orden de MHz)
 - Más baratos
 - Microprocesador
 - Propósito general
 - Alta frecuencia (varios GHz)
 - No pueden usarse solos, necesitan otros elementos (RAM, I/O, etc)

Procesando los datos: Microcontroladores



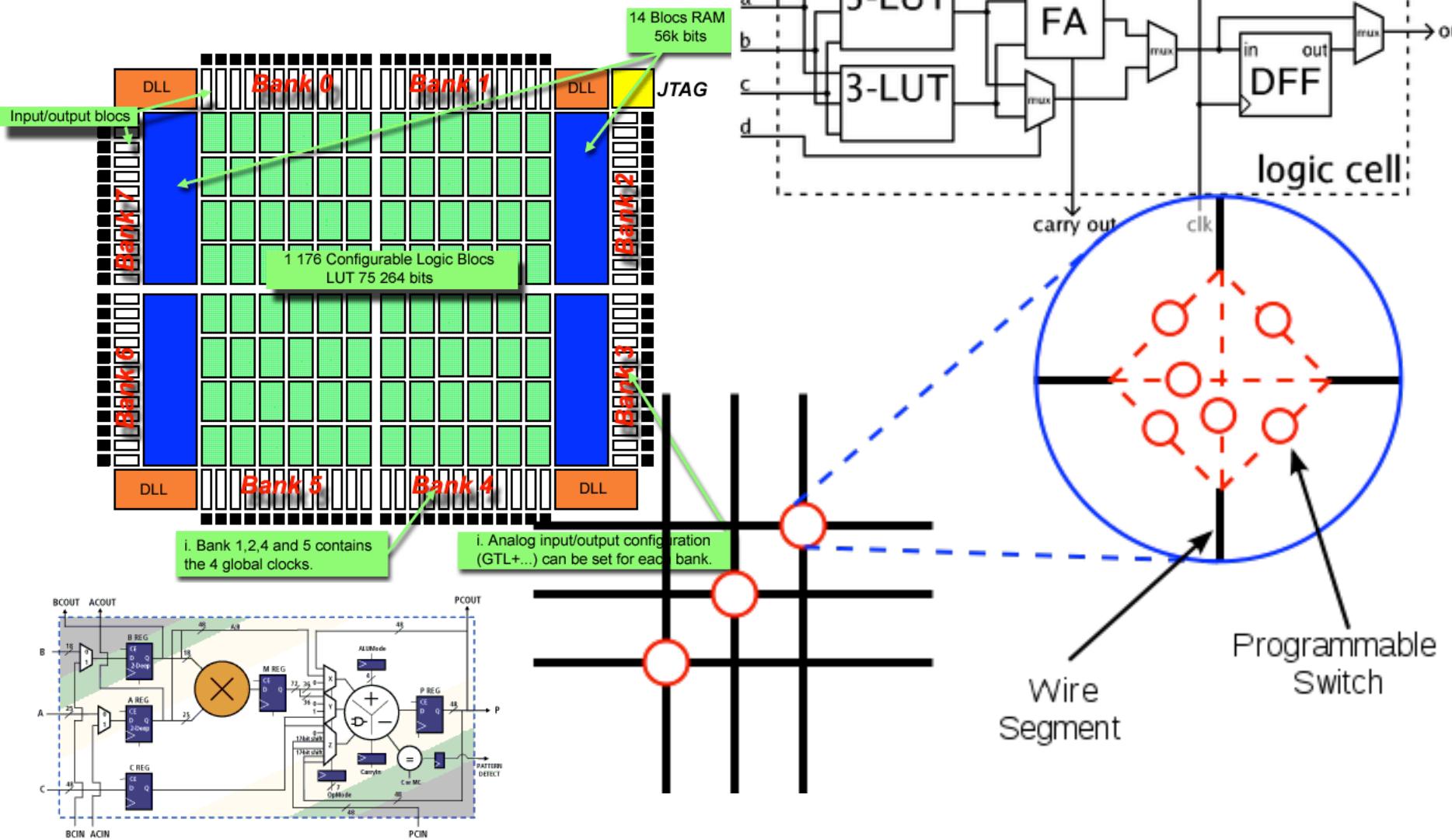
Procesando los datos: FPGAs

- Field Programmable Gate Arrays (FPGAs)
- HW con bloques y conexiones reconfigurables: 10-20X más eficientes (op/watt) que procesadores [Kuon and Rose, 2007]
- Solución híbrida
 - HW a medida (ASICs) demasiado caro y SW demasiado lento
 - Combina rapidez HW con flexibilidad SW



Procesando los datos: FPGAs

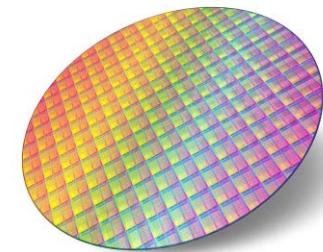
□ Matriz de celdas básicas



- Tendencia a acercar la FPGA al procesador (SoC). El mercado confía mucho en las FPGAs
 - Intel adquiere Altera (\$16.7 billones americanos, 2015)
 - AMD adquiere Xilinx (\$35 billones americanos, 2020)
- Ejemplos
 - Zynq = ARM + FPGA (Artix, Kintex, Ultrascale)
 - <https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html>
 - Pynq = Zynq programable con Python
 - <http://www.pynq.io/>
 - Intel Arria 10, Stratix 10, Agilex
 - <https://www.intel.es/content/www/es/es/products/details/fpga.html>

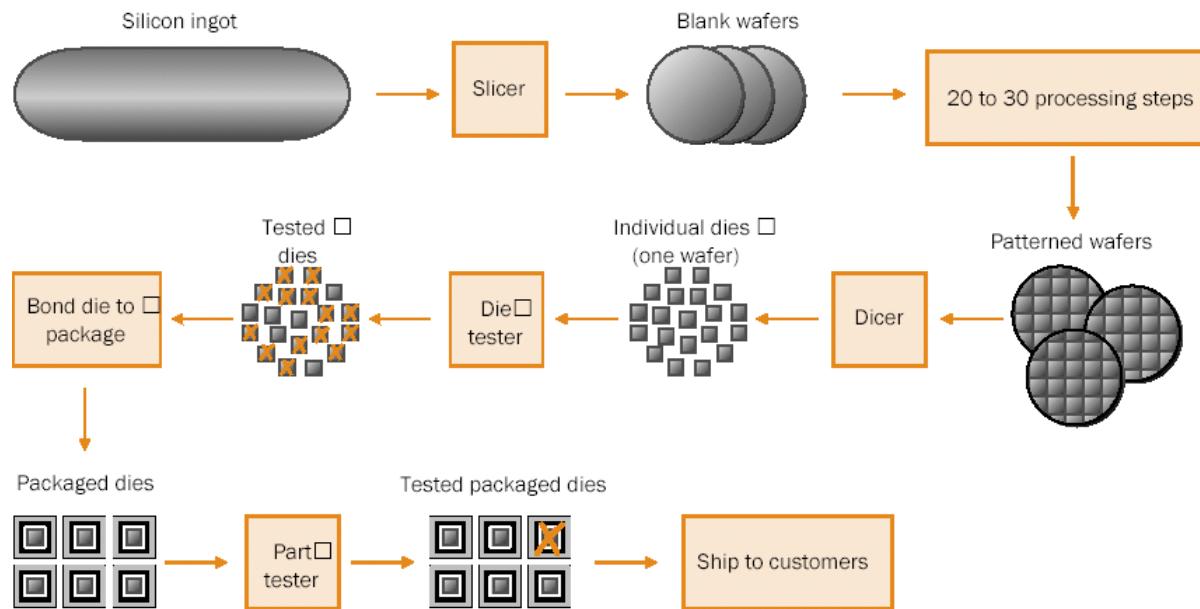
Procesando los datos: ASICs

- Application Specific Integrated Circuits (ASICs)
- Máximo rendimiento/eficiencia energética
 - 10X más eficientes (op/watt) que FPGAs [Kuon and Rose, 2007]
 - 35X más eficientes en área que FPGAs
- Tipo específico de IC (Integrated Circuit)
 - Chips impresos en obleas (wafers) de silicio
 - Las obleas se dividen en dados (dies) y cada dado se empaqueta
 - Coste elevado
 - Diseño, simulación, síntesis, verificación, empaquetado, test
 - Herramientas CAD: reducen time to market



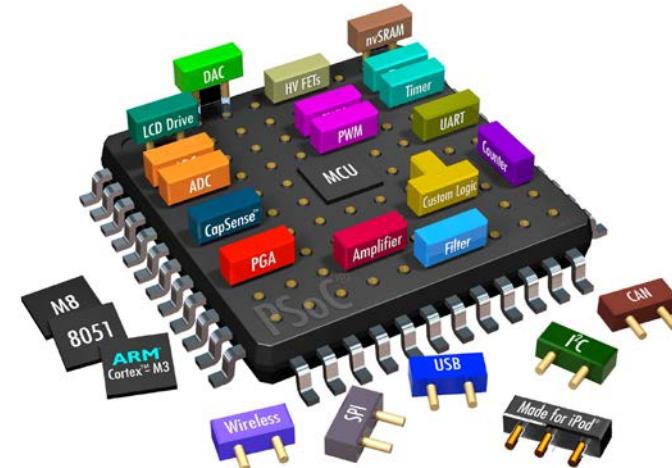
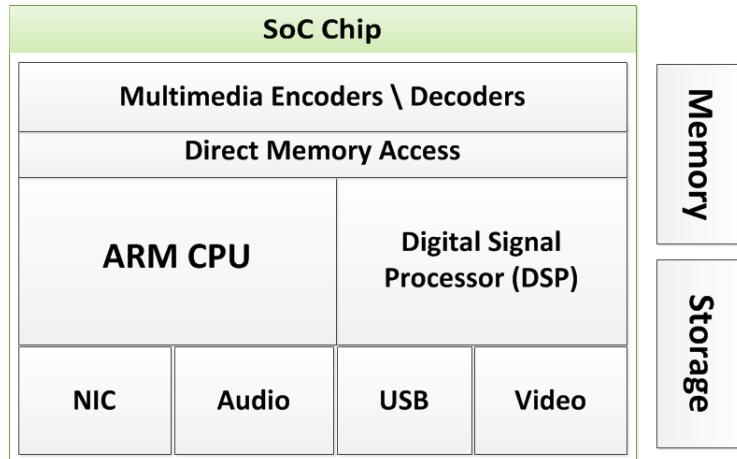
Procesando los datos: ASICs

- Coste por dado = $\text{coste_Oblea}/(\text{dados_por_oblea} * \text{yield})$
 - yield = Fracción de datos que pasan la fase de testing
 - $\text{yield}_{\text{empirical}} = 1/(1 + (\text{tasa_defectos} * \text{area_dado})/2)^2$
- $\text{dados_por_oblea} \approx (\text{area_oblea}) / (\text{area_dado})$
 - Ignora los bordes que no caben en un dado (oblea circular)

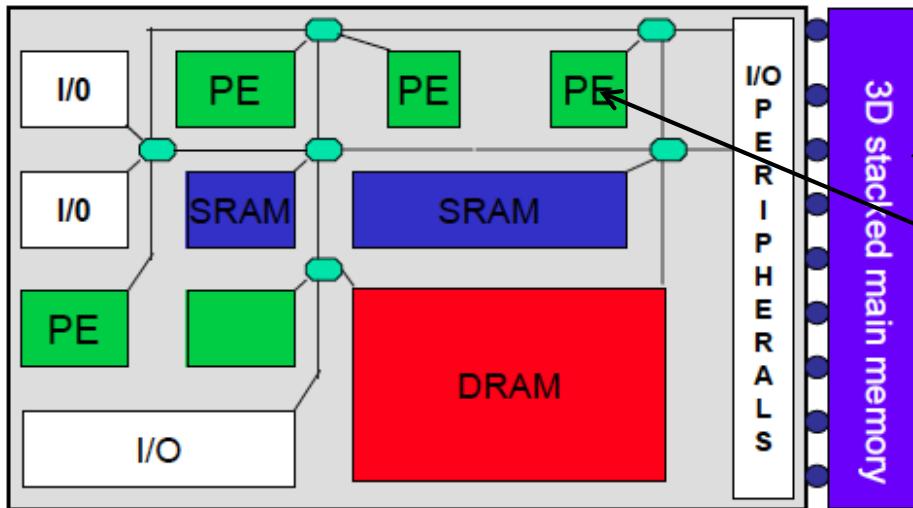


Procesando los datos: MPSoCs

- Concepto de *System on Chip* (SoC)
 - IC que integra todos los elementos funcionales básicos dentro de un chip
 - Sistema completo dentro de un chip
 - Microprocesador + Memoria + Periféricos
 - uC, ADC y DAC, JTAG, FPGA, ASIC, etc.
- MPSoC: Varios procesadores/cores + SoC
 - Ej. Zynq



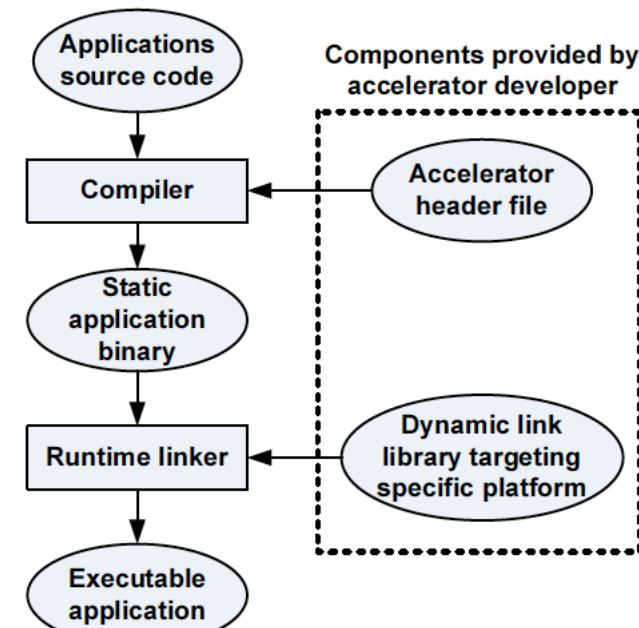
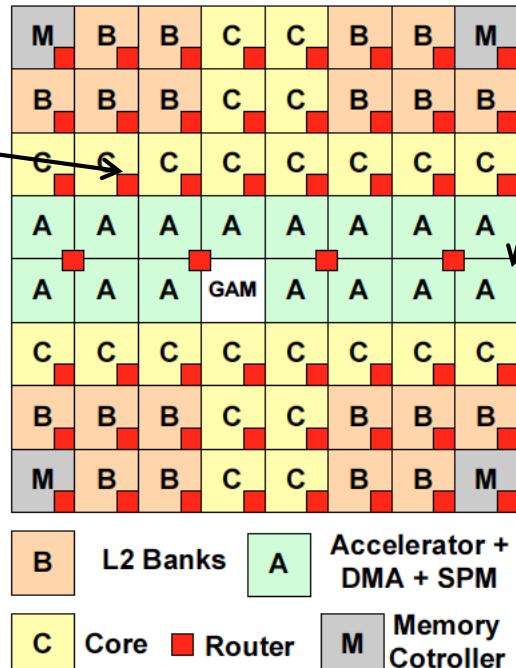
Procesando los datos: MPSoCs



[L. Benini et al.]

Los Processing Elements (PEs) o Accelerators (A) pueden ser FPGAs, GPUs, e incluso ASICs

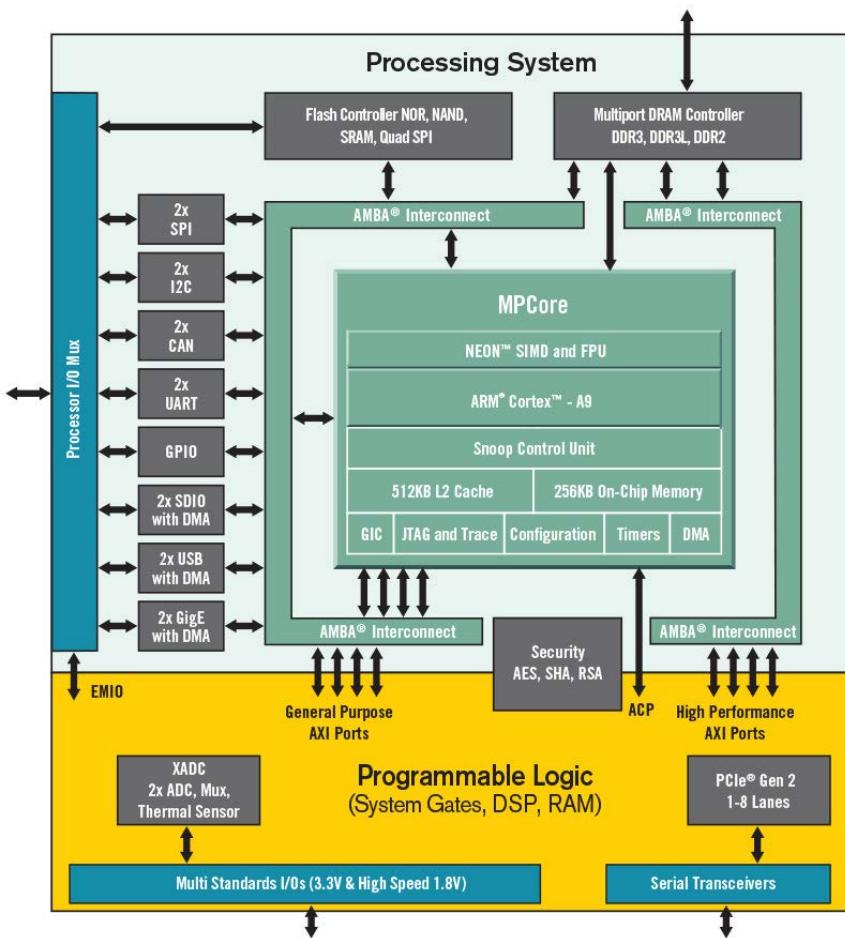
[J. Cong et al.]
Arquitectura con múltiples aceleradores (en lugar de cores)



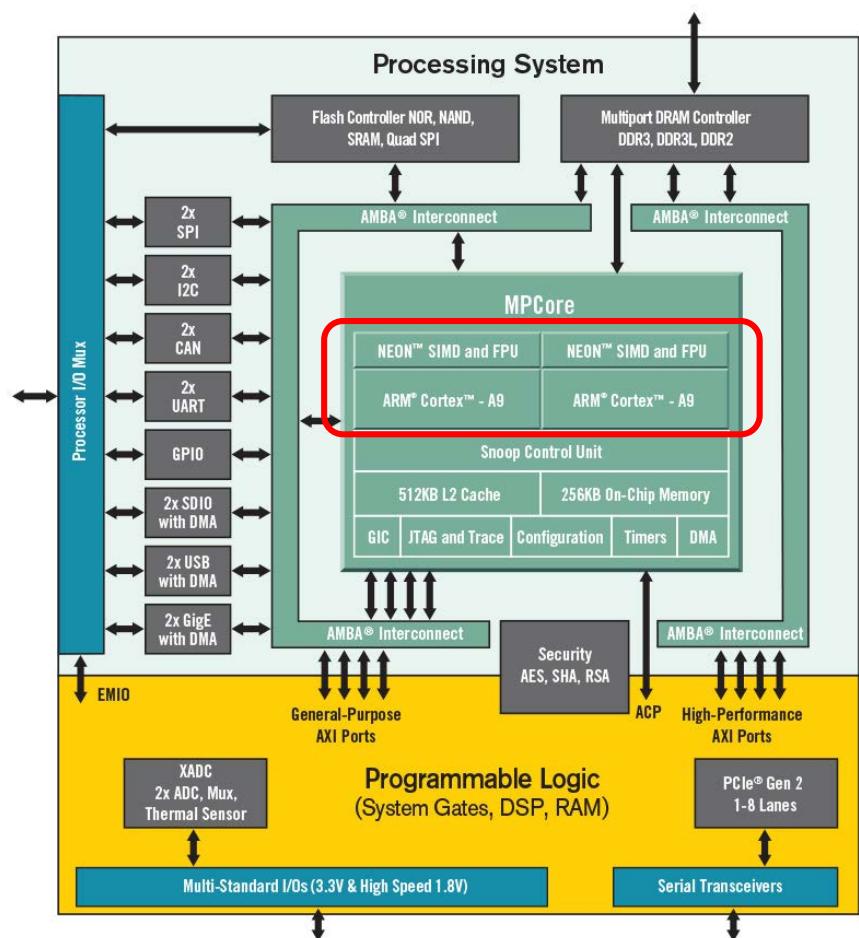
Algunos MPSOCs comerciales: familia Zynq de Xilinx

Familia Zynq 7000S =
ARM Cortex A9 +
Artix-7

Low-power,
low-cost



Familia Zynq 7000 =
ARM Cortex A9 +
Artix-7/Kintex-7



Algunos MPSoCs comerciales: Google Coral

- Google Coral = ARM Cortex M0+ + TPU
 - Montan Tensor Processing Units (TPUs), ASICs para realizar productos de matrices intensivos
 - TPU: arquitectura de array sistólico 2D

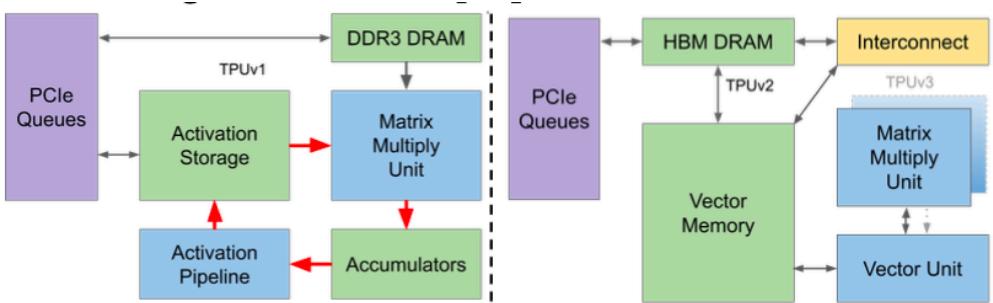


Figure 1. TPUv1 block diagram (left) vs TPUv2/v3.

N. P. Jouppi et al., "Ten Lessons From Three Generations Shaped Google's TPUv4i : Industrial Product," 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), 2021, pp. 1-14, doi: 10.1109/ISCA52012.2021.00010.

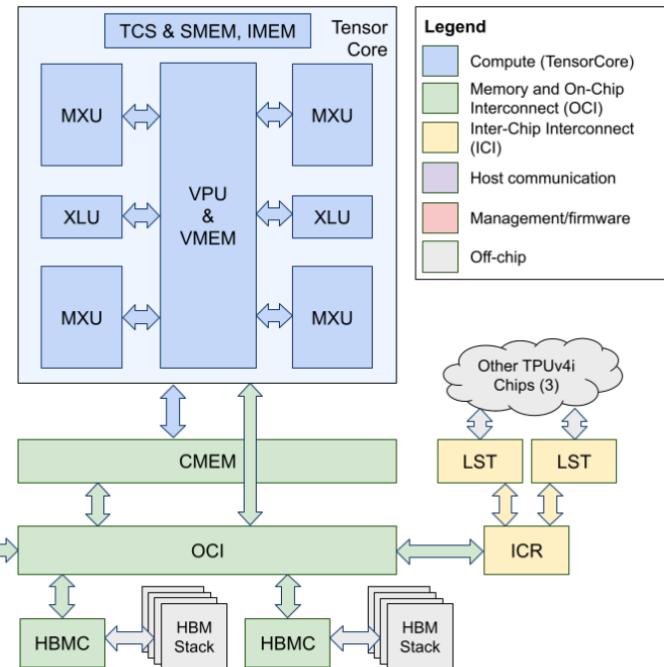
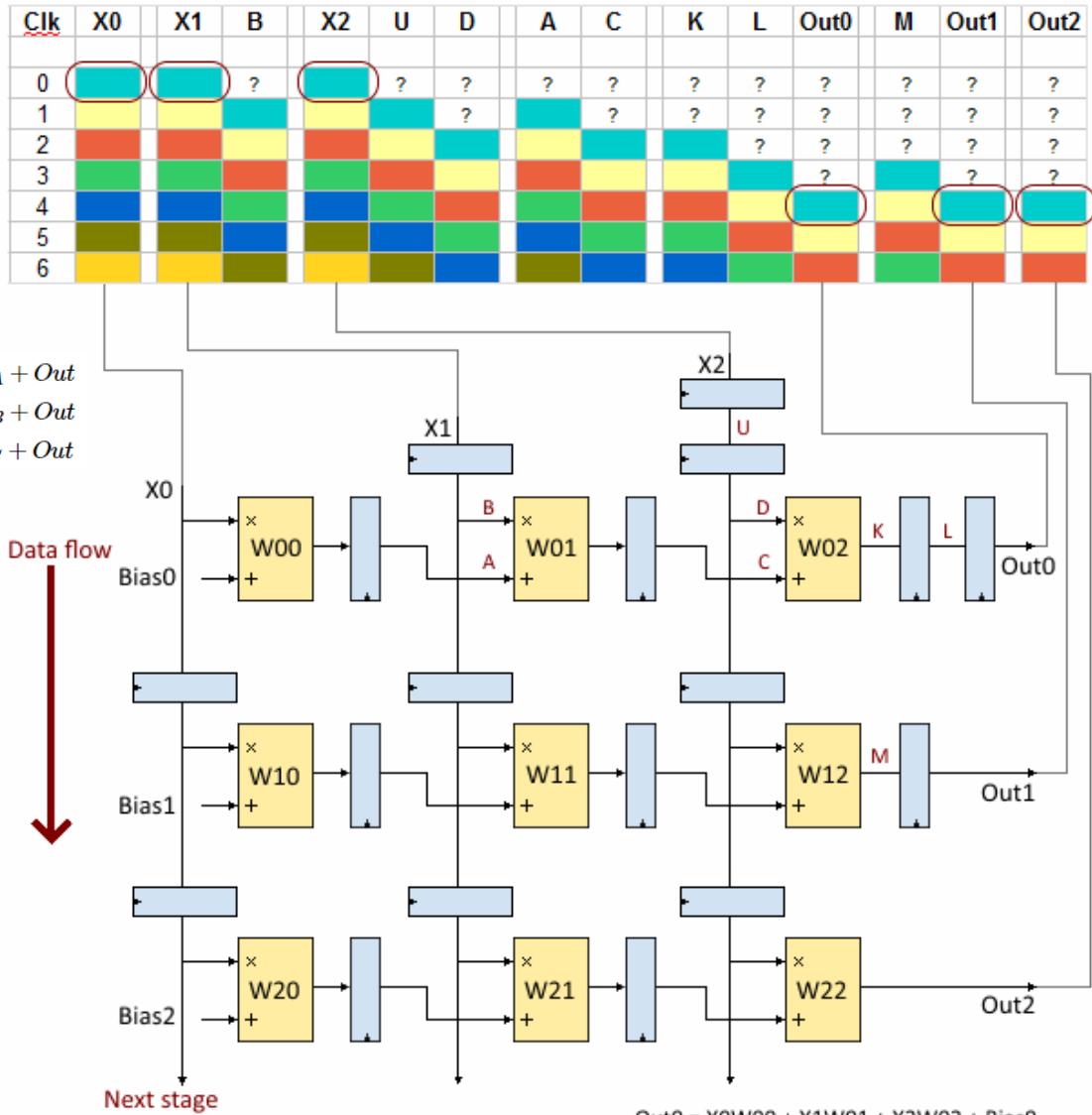
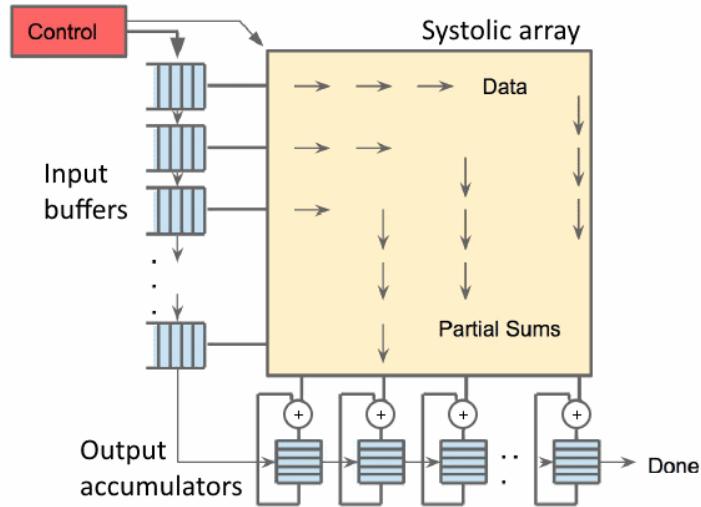


Figure 5. TPUv4i chip block diagram. Architectural memories are HBM, Common Memory (CMEM), Vector Memory (VMEM), Scalar Memory (SMEM), and Instruction Memory (IMEM). The data path is the Matrix Multiply Unit (MXU), Vector Processing Unit (VPU), Cross-Lane Unit (XLU), and TensorCore Sequencer (TCS). The uncore (everything not in blue) includes the On-Chip Interconnect (OCI), ICI Router (ICR), ICI Link Stack (LST), HBM Controller (HBMC), Unified Host Interface (UHI), and Chip Manager (MGR).

Algunos MPSoCs comerciales: Google Coral

□ Arquitectura de array sistólico 2D

$$\begin{aligned}
 Out_A &= X_0 \cdot W_0^0 + X_1 \cdot W_0^1 + X_2 \cdot W_0^2 + X_3 \cdot W_0^3 & \rightarrow & Out = Out_A + Out \\
 Out_B &= X_4 \cdot W_0^4 + X_5 \cdot W_0^5 + X_6 \cdot W_0^6 + X_7 \cdot W_0^7 & \rightarrow & Out = Out_B + Out \\
 Out_C &= X_8 \cdot W_0^8 + X_9 \cdot W_0^9 + X_{10} \cdot W_0^{10} + X_{11} \cdot W_0^{11} & \rightarrow & Out = Out_C + Out
 \end{aligned}$$

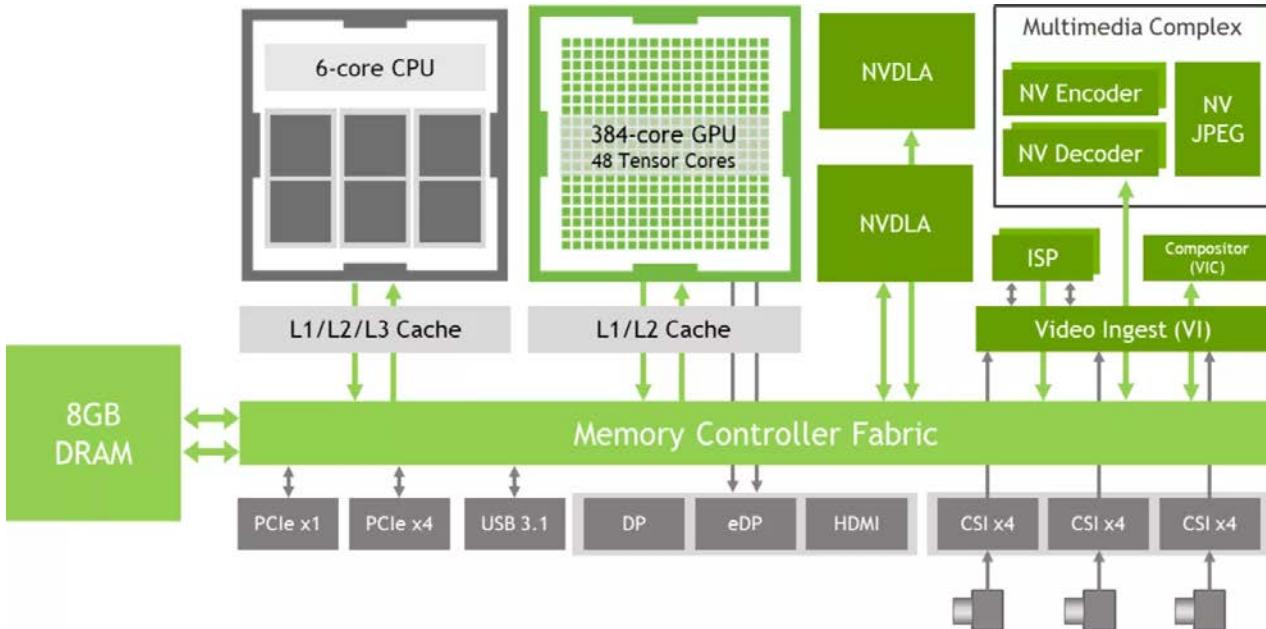
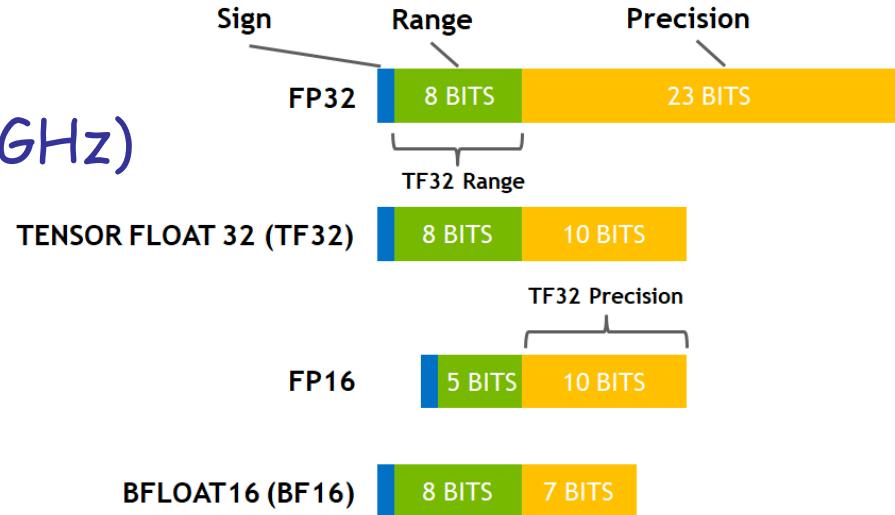


<https://storage.googleapis.com/nexttpu/index.html>

<https://qengineering.eu/google-corals-tpu-explained.html>

❑ Ej. Jetson Nano Xavier

- o 6 ARM (4x1.2GHz, 2x1.5GHz)
- o 21 TOPS (15W),
14 TOPS (10W)
- o 384 CUDA cores
- o 48 tensor cores



<https://developer.nvidia.com/blog/accelerating-ai-training-with-tf32-tensor-cores/>

<https://developer.nvidia.com/blog/accelerating-winml-and-nvidia-tensor-cores/>

Jetson Nano

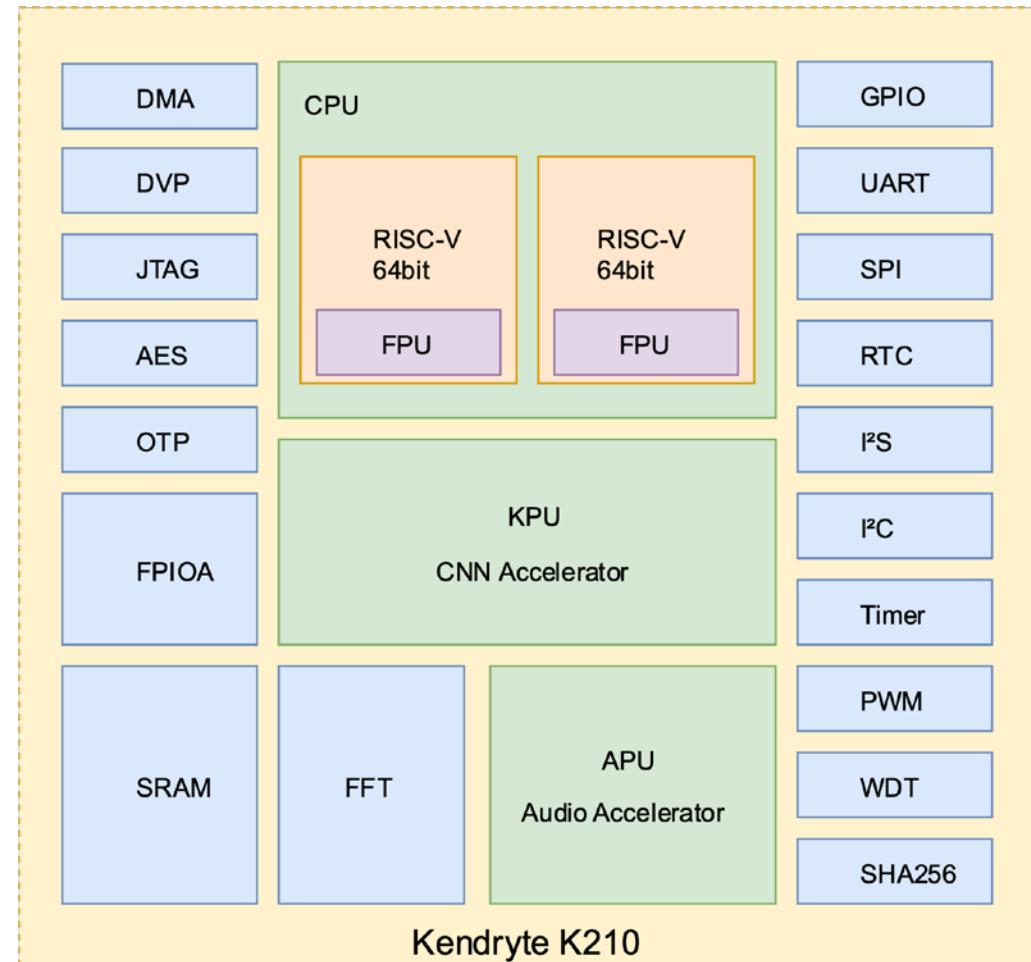
NVIDIA Jetson Family Specifications				
	Xavier NX (15W)	Xavier NX (10W)	AGX Xavier	Jetson Nano
CPU	4x/6x Carmel @ 1.4GHz or 2x Carmel @ 1.9GHz	4x/ Carmel @ 1.2GHz or 2x Carmel @ 1.5GHz	8x Carmel @ 2.26GHz	4x Cortex-A57 @ 1.43GHz
GPU	Volta, 384 Cores @ 1100MHz	Volta, 384 Cores @ 800MHz	Volta, 512 Cores @ 1377MHz	Maxwell, 128 Cores @ 920MHz
Accelerators	2x NVDLA		2x NVDLA	N/A
Memory	8GB LPDDR4X, 128-bit bus (51.2 GB/sec)		16GB LPDDR4X, 256-bit bus (137 GB/sec)	4GB LPDDR4, 64-bit bus (25.6 GB/sec)
Storage	16GB eMMC		32GB eMMC	16GB eMMC
AI Perf.	21 TOPS	14 TOPS	32 TOPS	N/A
Dimensions	45mm x 70mm		100mm x 87mm	45mm x 70mm
TDP	15W	10W	30W	10W
Price	\$399		\$999	\$129

Algunos MPSoCs comerciales

- ❑ MaixCube = Kendryte 210 + Periféricos (cámara, TFT, ...)
- ❑ Kendryte 210 = MPSoC RISC-V dual core 64-bits
 - Acelerador para ML
 - KPU para CNNs (visión). Soporta YOLO
 - APU para sonido
 - Otros aceleradores
 - FFT
 - AES, SHA



- | | | |
|-----------------------|--------------------|----------------|
| ① 1.3 inch TFT Screen | ② Reset Button | ③ Power Button |
| ④ Grove Interface | ⑤ SP-MOD Interface | ⑥ TF Card Slot |
| ⑦ Camera | ⑧ Type-C Interface | ⑨ 3-Way Button |

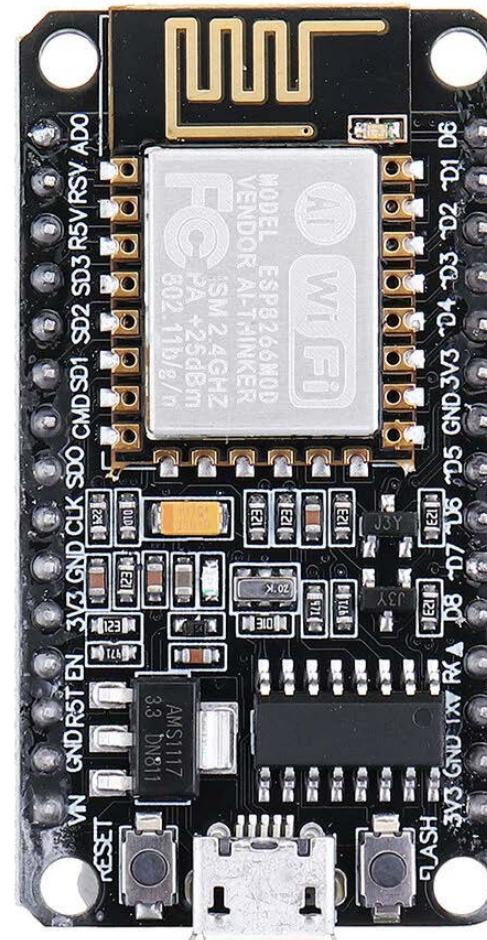


Kendryte K210

Algunos MPSoCs comerciales

□ NodeMCU

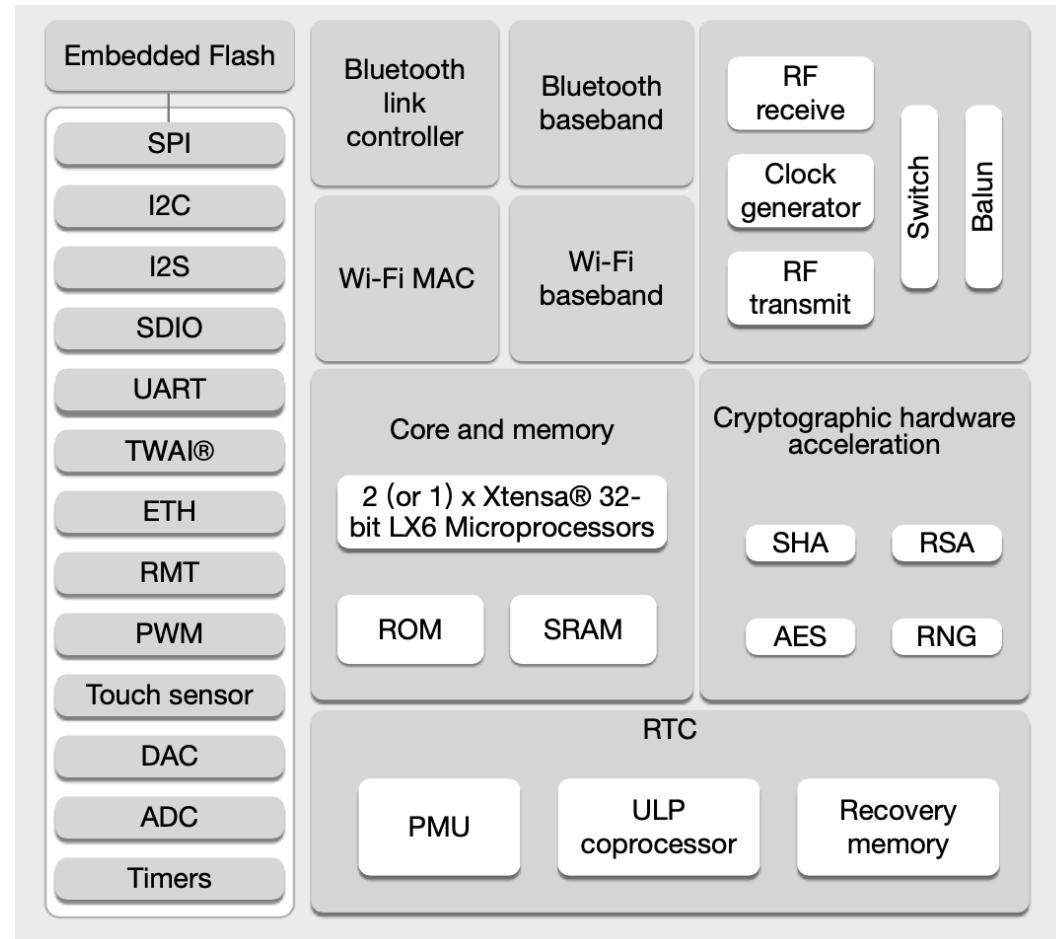
- ESP8266+firmware LUA
- Ultra low-cost, low-power
- Fabricado por Expressif, uc de Xtensa, 80MHz (hasta 160MHz)
- Pila TCP/IP integrada
- Programable con entorno Arduino
- Pequeño tamaño, ideal para wearables
- ESP-Mesh



Algunos MPSoCs comerciales

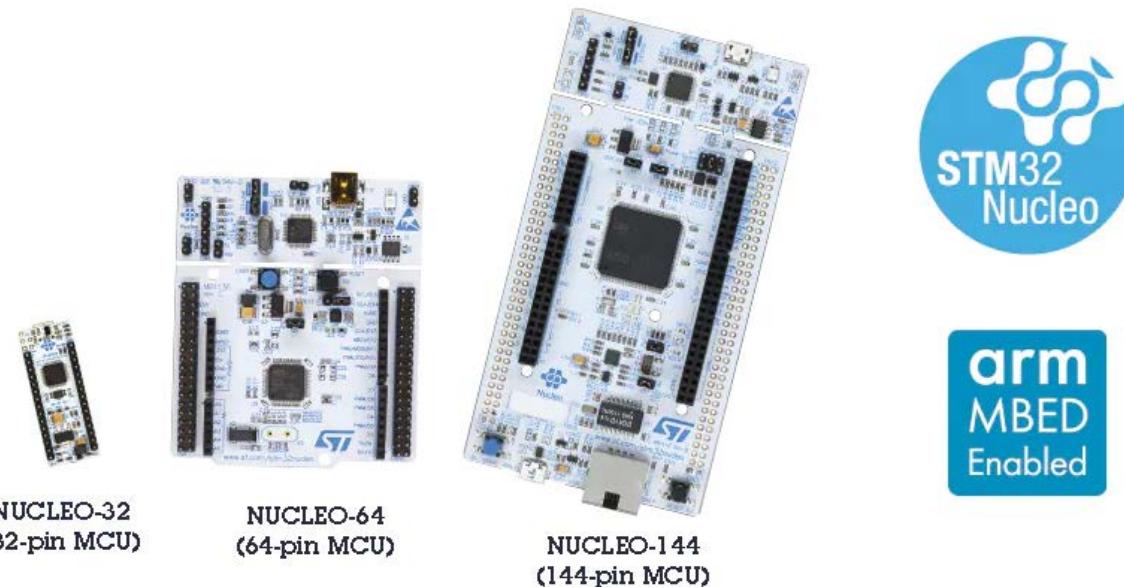
□ ESP32

- Low-cost, low-power
- Procesador dual-core de Xtensa (hasta 240MHz)
- Más GPIOs que ESP8266
- Soporta Bluetooth también
- ESP-Mesh



□ STM32 NUCLEO

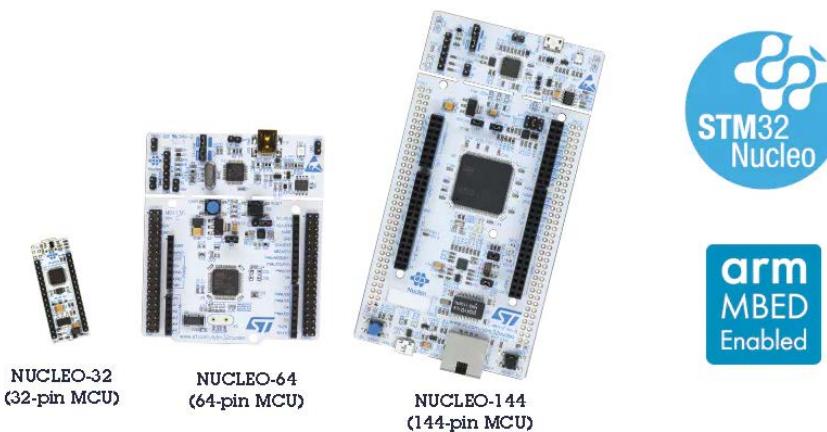
- Low-cost, low-power
- Basados en Procesador cortex, M0+, M3, M4, M7, M133 dual-core de Xtensa (hasta 240MHz)
- Variedad de modelos según especialización
- Soporta Conexión con pines de Arduino



Algunos MPSoCs comerciales

□ STM32 NUCLEO

- <https://www.st.com/en/ecosystems/stm32-nucleo.html#>
- https://www.mouser.es/new/stmicroelectronics/stm32/?gclid=CjwKCAiA85efBhBbEiwAD7oLQNh9KS5MimNw1WuechrmziRwE07I5HfXKaVuajIHjkI8XjgAMKCwdxoCRckQAvD_BwE



Algunos MPSoCs comerciales

□ STM32 MCU 32 bits ARM Cortex -M



- Tan importante como los elementos de procesado
 - Si la conexión es lenta/insegura, ¿para qué un SED?
Mejor un SE
 - Estructura de un mensaje



- Tipos principales de comunicaciones
 - Wired (buses)
 - Wireless
- Topologías
- Codificación
- Protocolos

Comunicaciones: wired vs wireless

□ Wired

o Pros

- Rápidas
- Seguras
- Fácil conexión

o Cons

- Cableado
- Conceptualmente incompatible con los "smart devices"

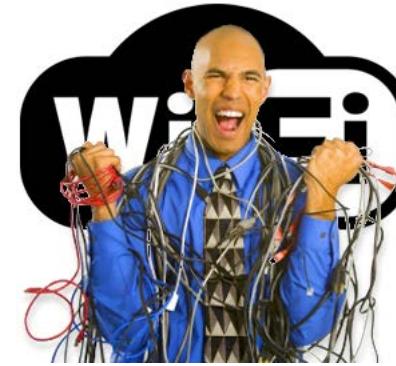
□ Wireless

o Pros

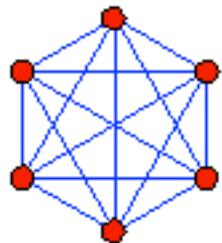
- Flexibilidad. Conectividad con distinto tipo de redes sin conexión física

o Cons

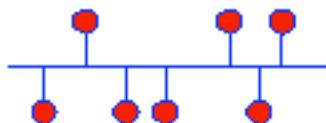
- Ruido
- Consumo de potencia
- Fiabilidad



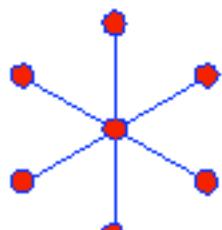
Buses: topologías



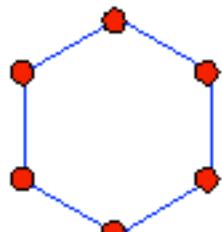
a) Fully Connected Topology



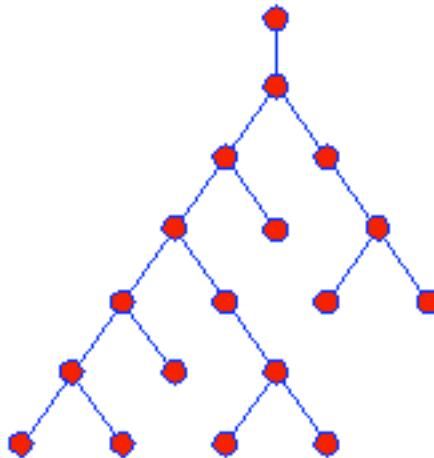
b) Bus Topology



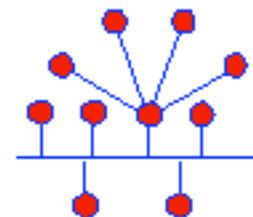
c) Star Topology



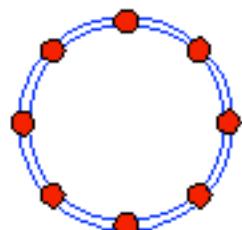
d) Ring Topology



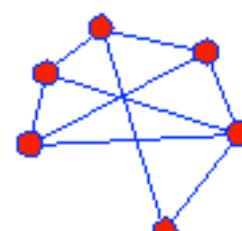
e) Tree Topology



g) Hybrid Topology
(example: combination of
Star topology and Bus topology)



h) Dual Ring Topology



f) Mesh Topology

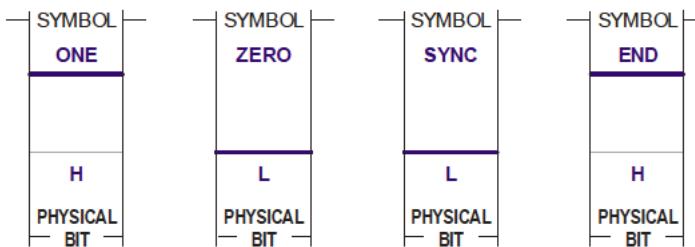


i) Linear Topology

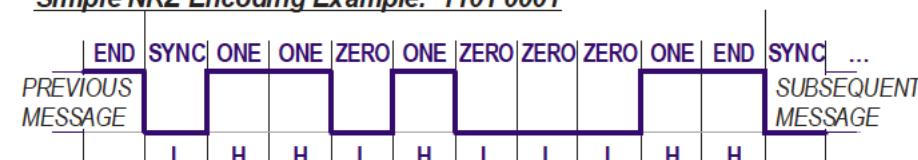
Nodes ● — Branches

Buses: codificación de datos

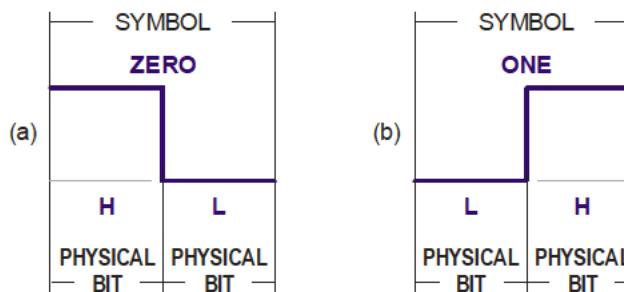
Simple NRZ Bit Encoding



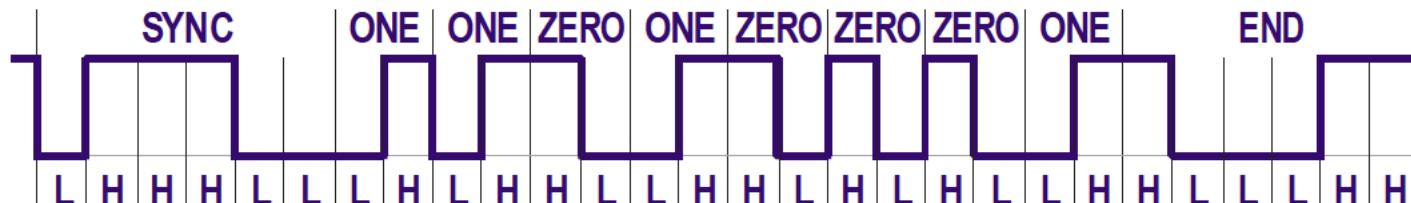
Simple NRZ Encoding Example: 1101 0001



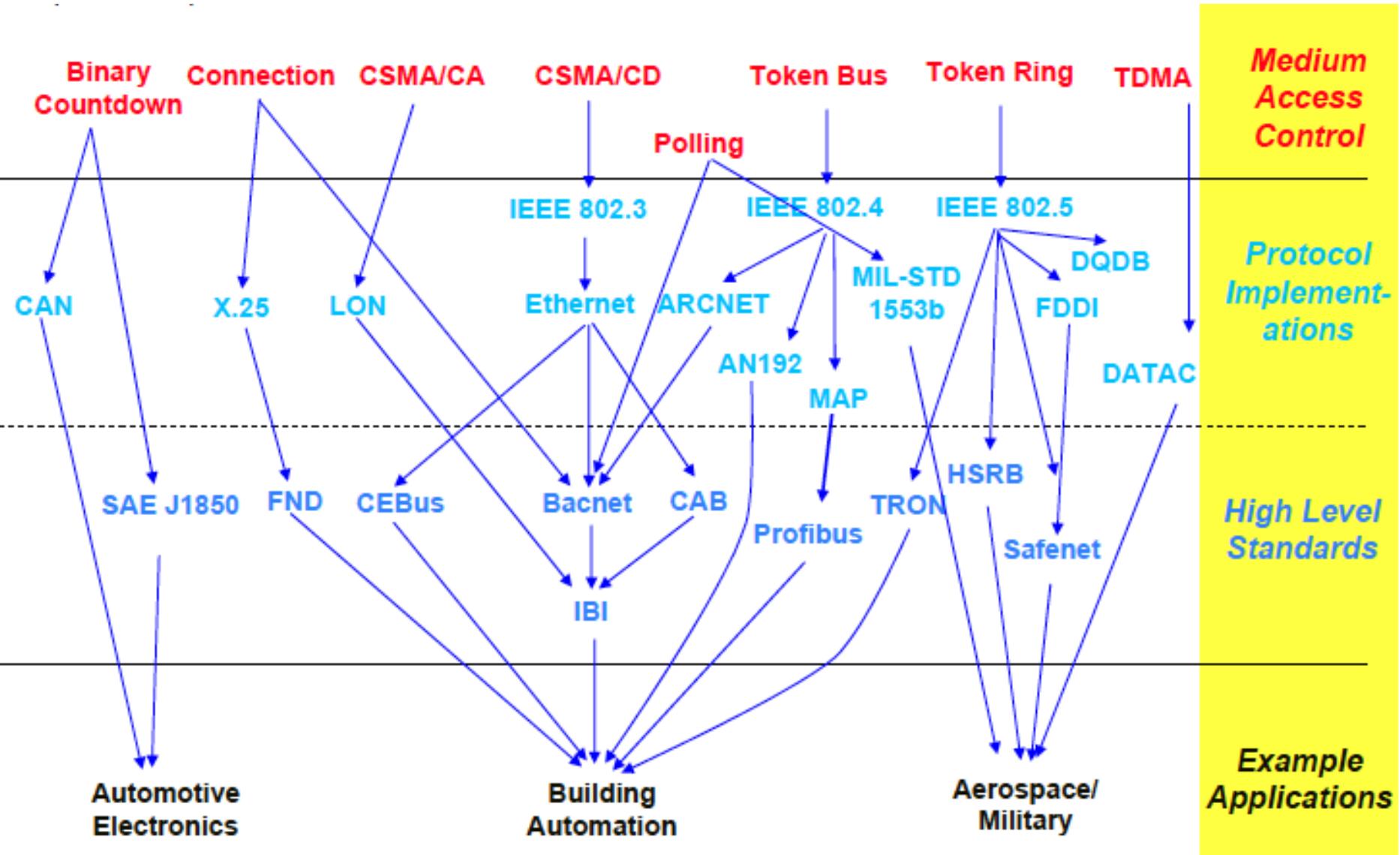
Manchester Bit Encoding



Manchester Encoding Example: 1101 0001



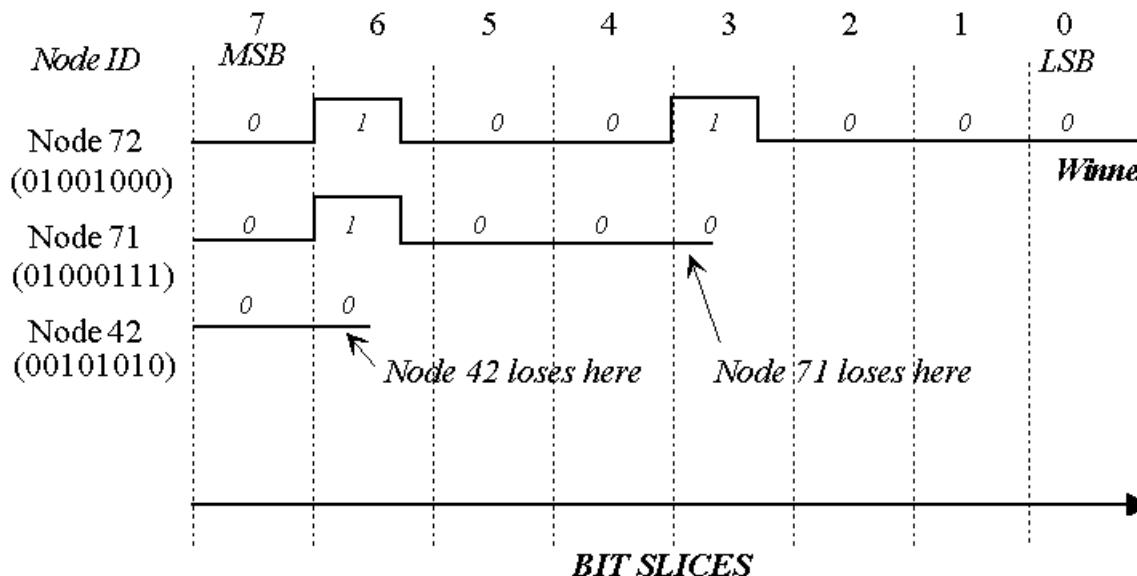
Comunicaciones: Protocolos



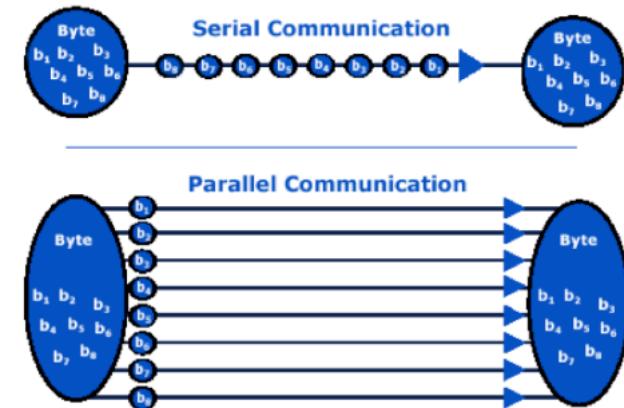
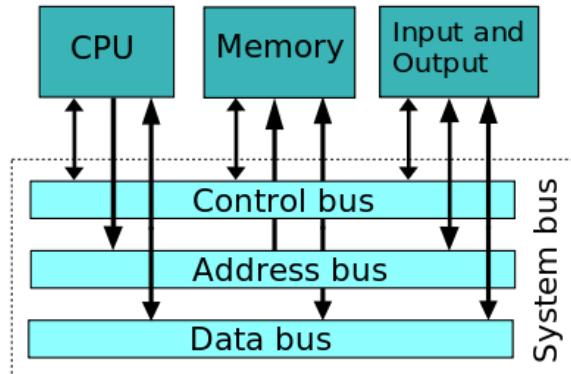
<http://www.ece.cmu.edu/~ece649/>

Protocolos: Binary Countdown

- Binary Countdown (Bit Dominance)
 - A cada nodo se le asigna un *id* numérico
 - Cada nodo que quiere transmitir envía su id al canal (dentro del mensaje)
 - Cuando un nodo detecta otro id más dominante, abandona la lucha por el canal
- Pros: alto rendimiento, arbitraje empotrado en el msg
- Cons: alta latencia para nodos con prioridad baja
- Ej. CAN

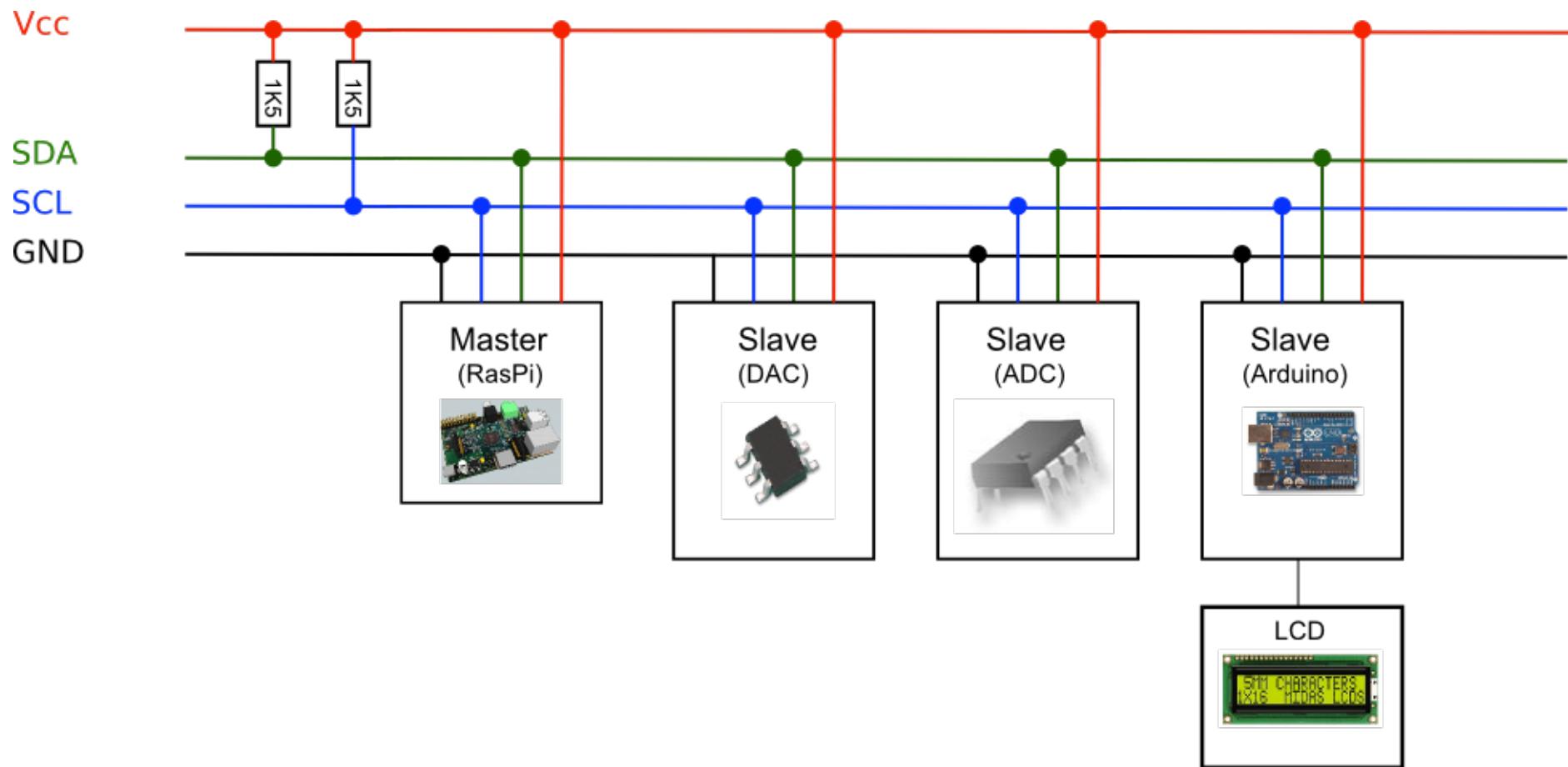


- Bus (omnibus, "for all")
 - Transfiere datos entre componentes de un sistema
 - Componentes
 - Líneas de datos
 - Líneas de dirección. Habitualmente coinciden con las de datos
 - Líneas de control. Implementan el protocolo
 - Serie vs Paralelo
 - Compartido (Árbitro) vs Punto a Punto
 - Síncrono (clk) vs Asíncrono



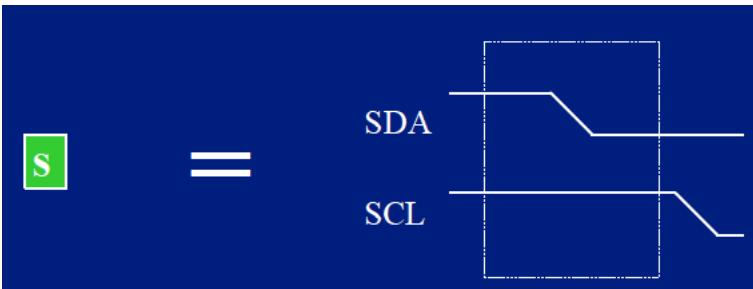
- Inter-Integrated Circuit (IIC, I²C ó I2C)
- Desarrollado por Phillips Semiconductor (NXP Semiconductors)
- Bus serie, síncrono, multimáster y bidireccional (ACK)
 - Cada trasferencia debe ser respondida con ACK o NACK
- 2 líneas
 - Datos (Serial DAta, SDA)
 - Clock (Serial CLock, SDL), generado por el máster
- Cada línea tiene una resistencia de pull-up
 - Dos valores: float-high y float-low
 - Si no hay dispositivos usando el bus, el valor leído será float high

Buses: I2C

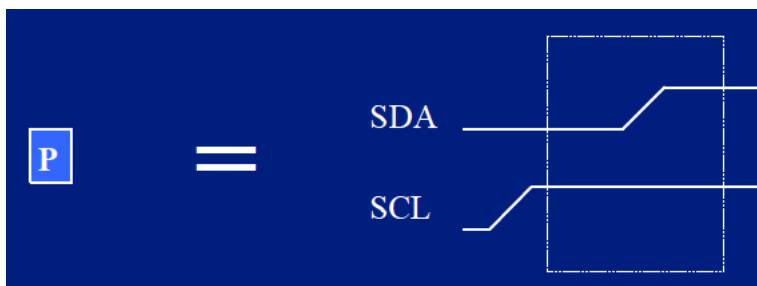


Buses: I2C

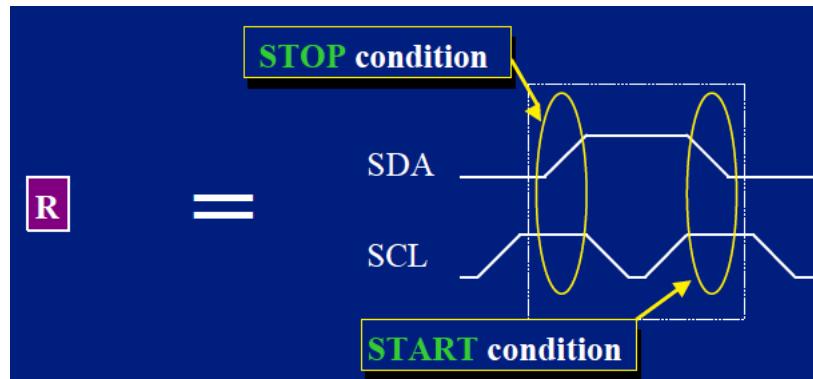
Start



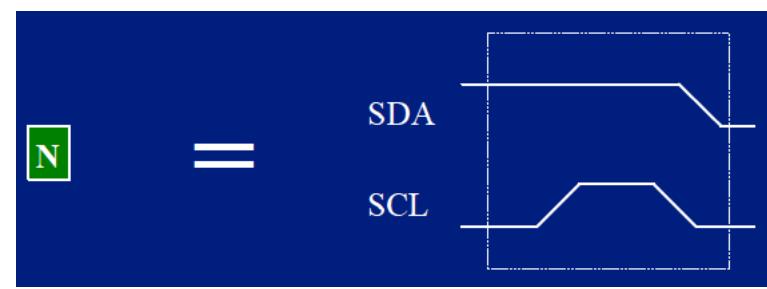
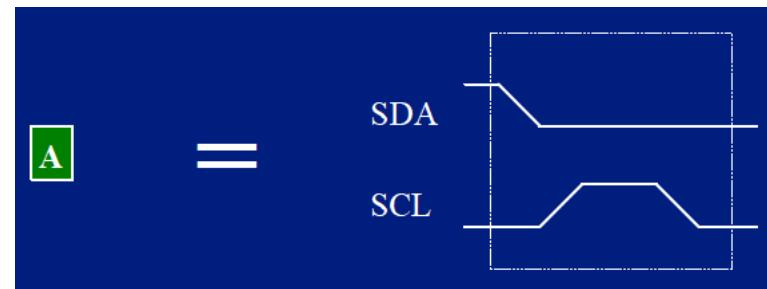
Stop



Restart



ACK



NACK

Buses: I2C

Escribimos en una EEPROM I2C (como la de la S3CEV40) = Write Addr + Write Data

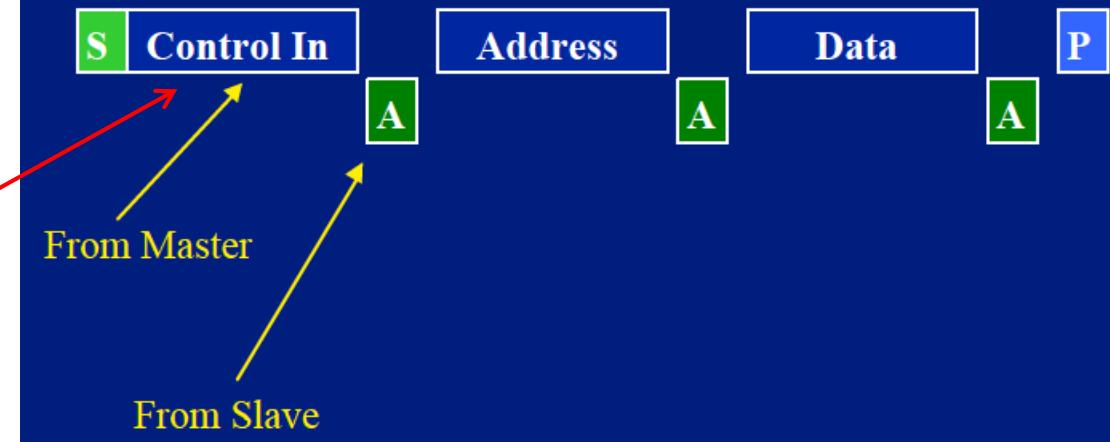
Indicamos Write

Leemos de una EEPROM I2C = Write Addr + Read data

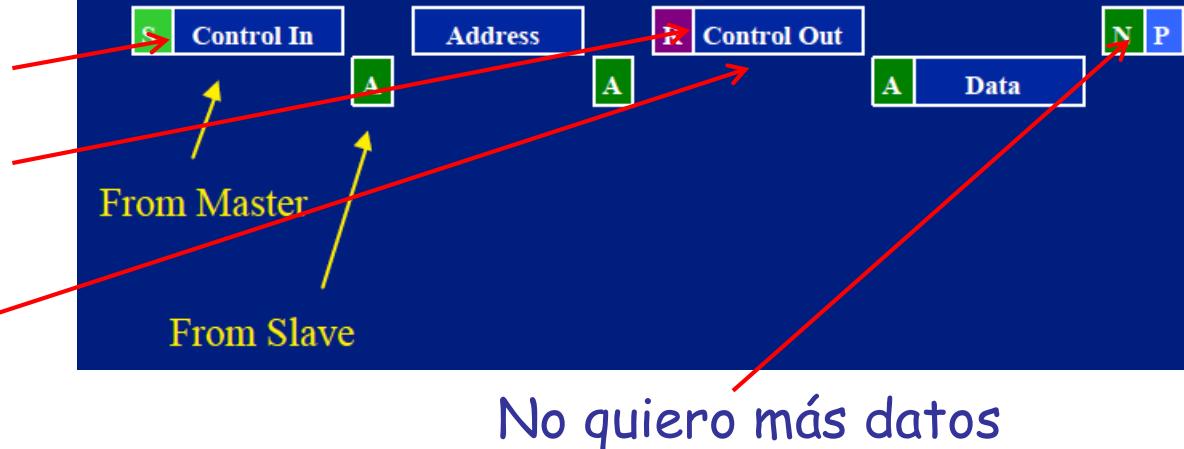
Indicamos Write
Indicamos Read

Dato disponible

Write Example



Read Example



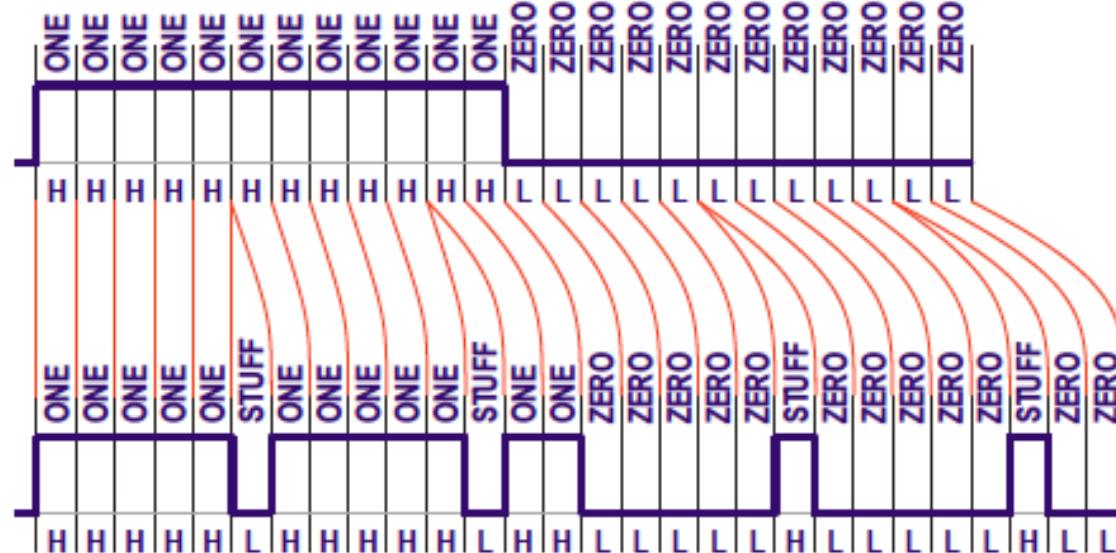
Buses: CAN

Controller Area Network

- o Protocolo muy usado en la industria automovilística
 - o Desarrollado por Bosch
 - o Acceso MAC por Binary Countdown ("0" dominante)
 - o Codificación NRZ con *stuff bits* para forzar flancos de reloj: ayuda a la resincronización con el emisor

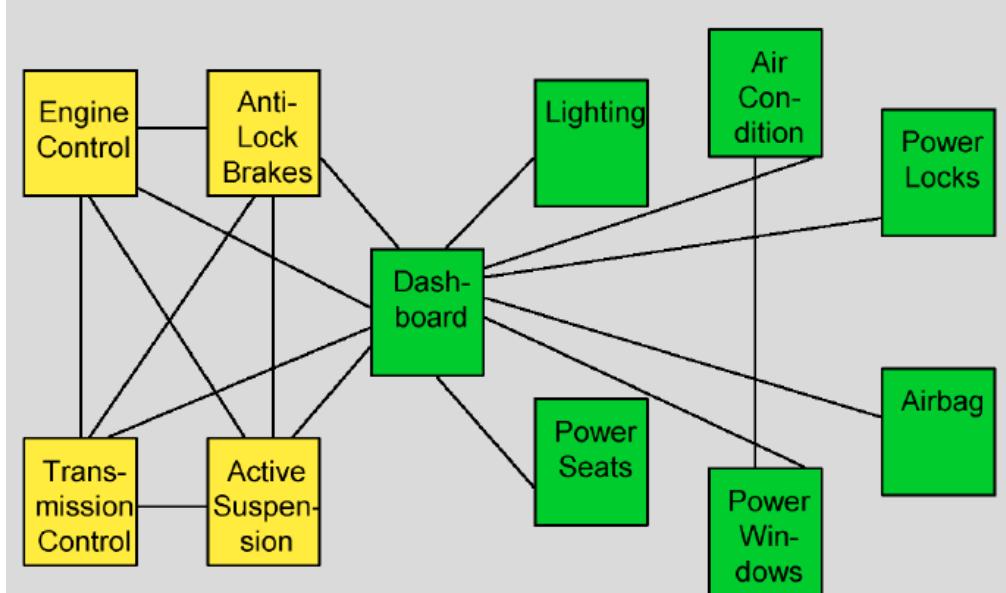
BIT STUFF IDEA:

SIMPLE NRZ ENCODING OF: 1111 1111 1111 0000 0000 0000



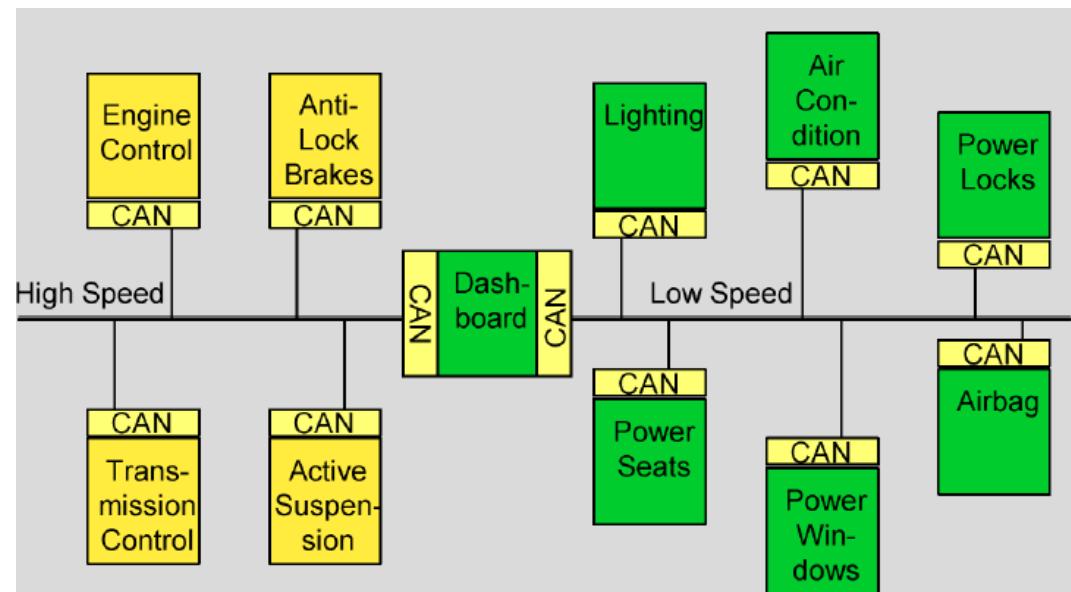
BIT-STUFFED NRZ ENCODING OF: 1111 1111 1111 0000 0000 0000

Buses: CAN



Antes de CAN.
Múltiples
conexiones punto a
punto. Interfaces
heterogéneas

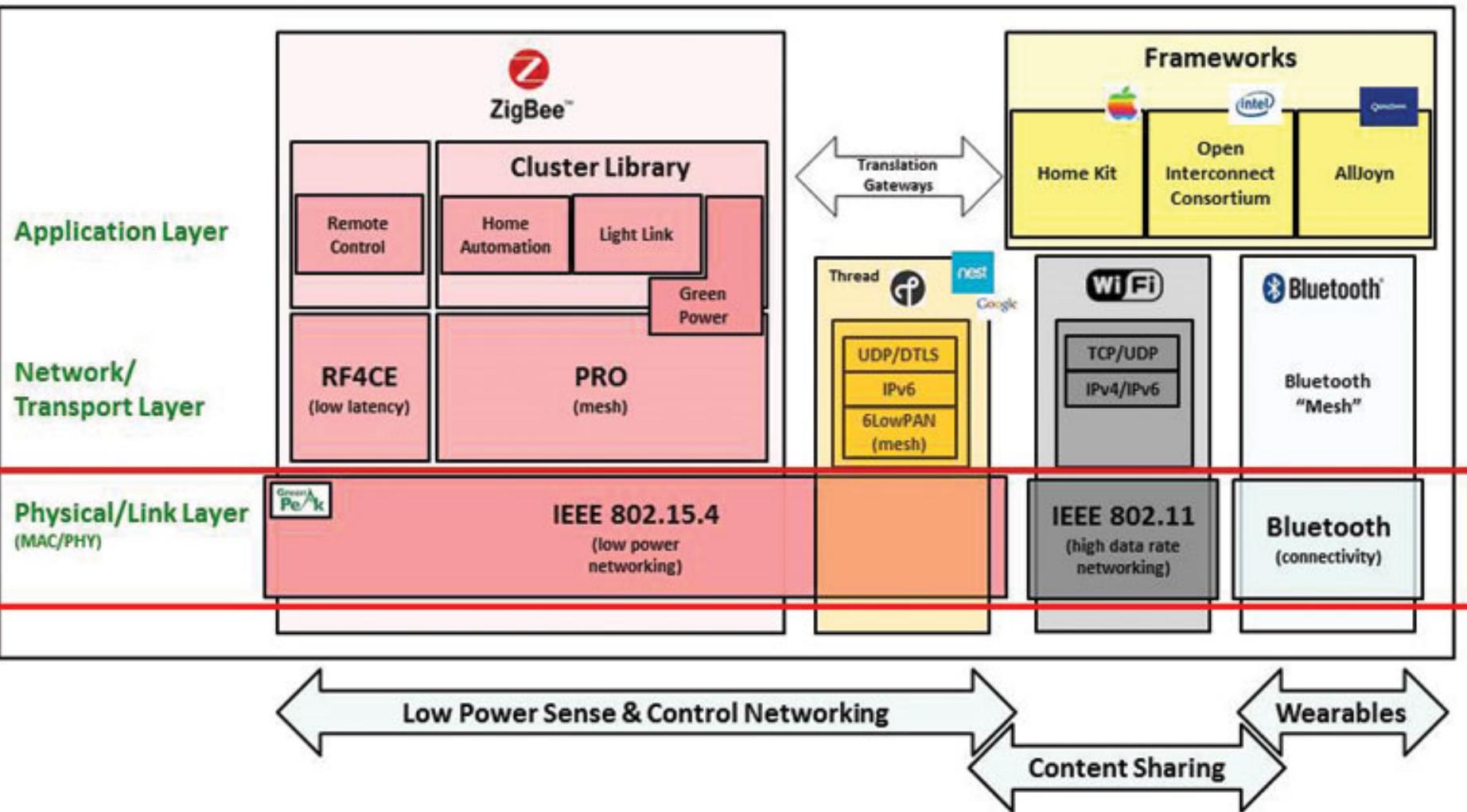
CAN: una única
interfaz para
comunicar todos los
elementos en un
bus compartido

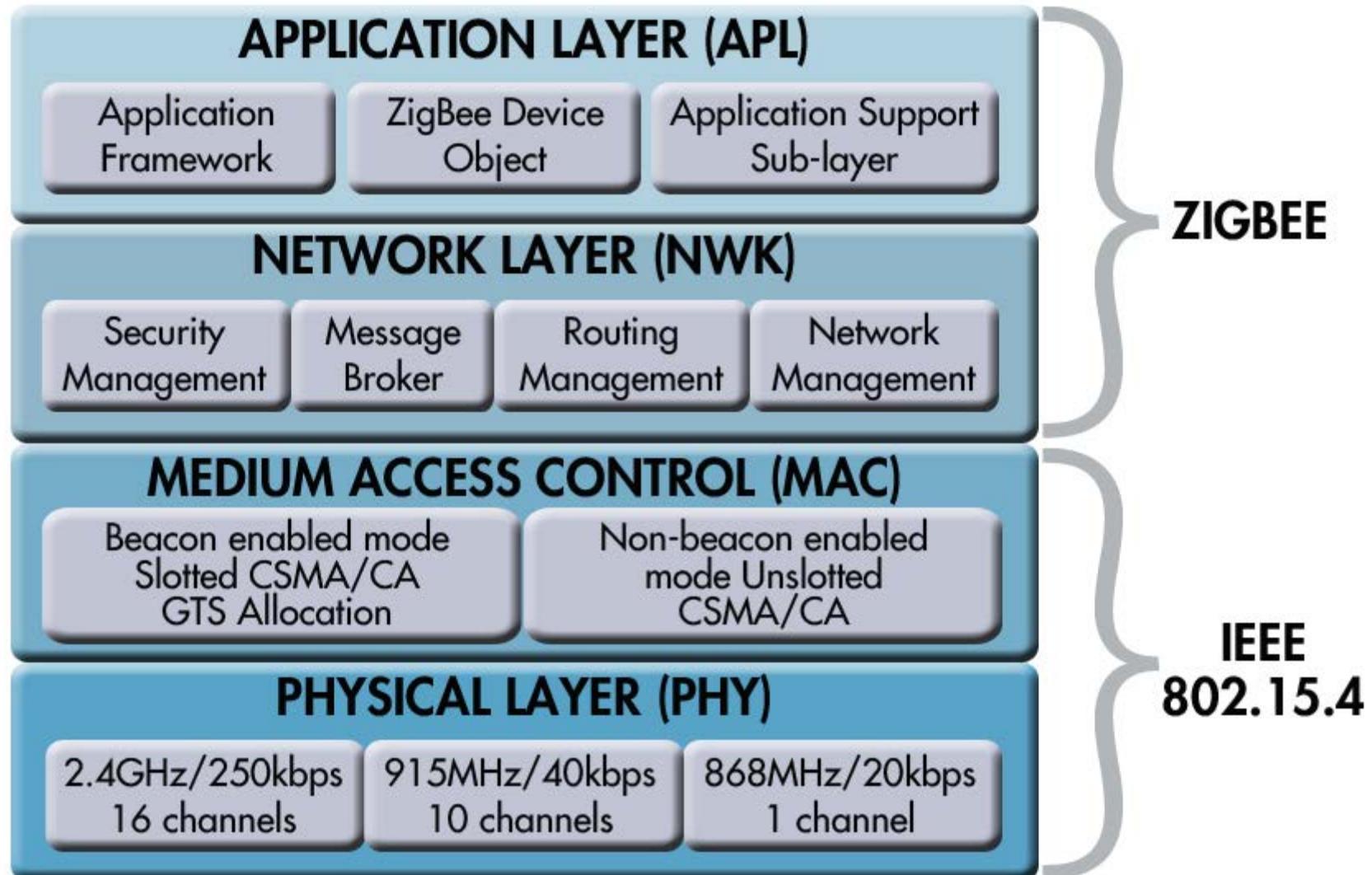


- LAN
 - WiFi
- PAN
 - Bluetooth
 - Bluetooth Alliance
 - 2.4GHz
 - Zigbee
 - Zigbee Alliance
 - 2.4GHz and sub 1GHz
 - 6LoWPAN
 - Low Power Wireless PAN
 - IPv6 on PAN
 - Others: Z-Wave, NFC
- WAN: implementaciones de LPWAN
 - LoRaWAN. LoRa Alliance
 - SigFox
 - Cellular. On GSM/3G/4G

La aparición de IPv6 y IoT son clave en la evolución de nuevos estándares y tecnologías de bajo consumo

Wireless





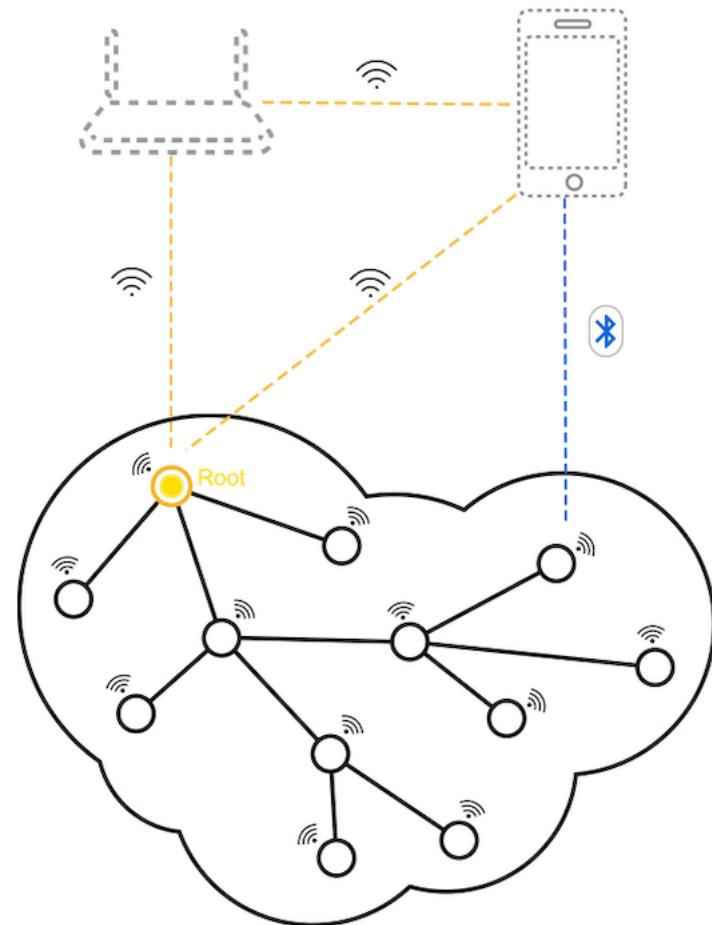
Wireless: Zigbee

	ZigBee RF4CE		ZigBee PRO							ZigBee IP
Application Standard	ZigBee Remote Control	ZigBee Input Device	ZigBee Building Automation	ZigBee Health Care	ZigBee Home Automation	ZigBee Retail Services	ZigBee Smart Energy 1.x	ZigBee Telecom Services	ZigBee Smart Energy 2.0	
Network	ZigBee RF4CE		ZigBee PRO							ZigBee IP
MAC	IEEE 802.15.4 – MAC									
PHY	IEEE 802.15.4 Sub-GHz (specified per region)			IEEE 802.15.4 – 2.4 GHz (worldwide)						

Open standard para topologías de malla basadas en IPv6

Wireless: ESP-Mesh

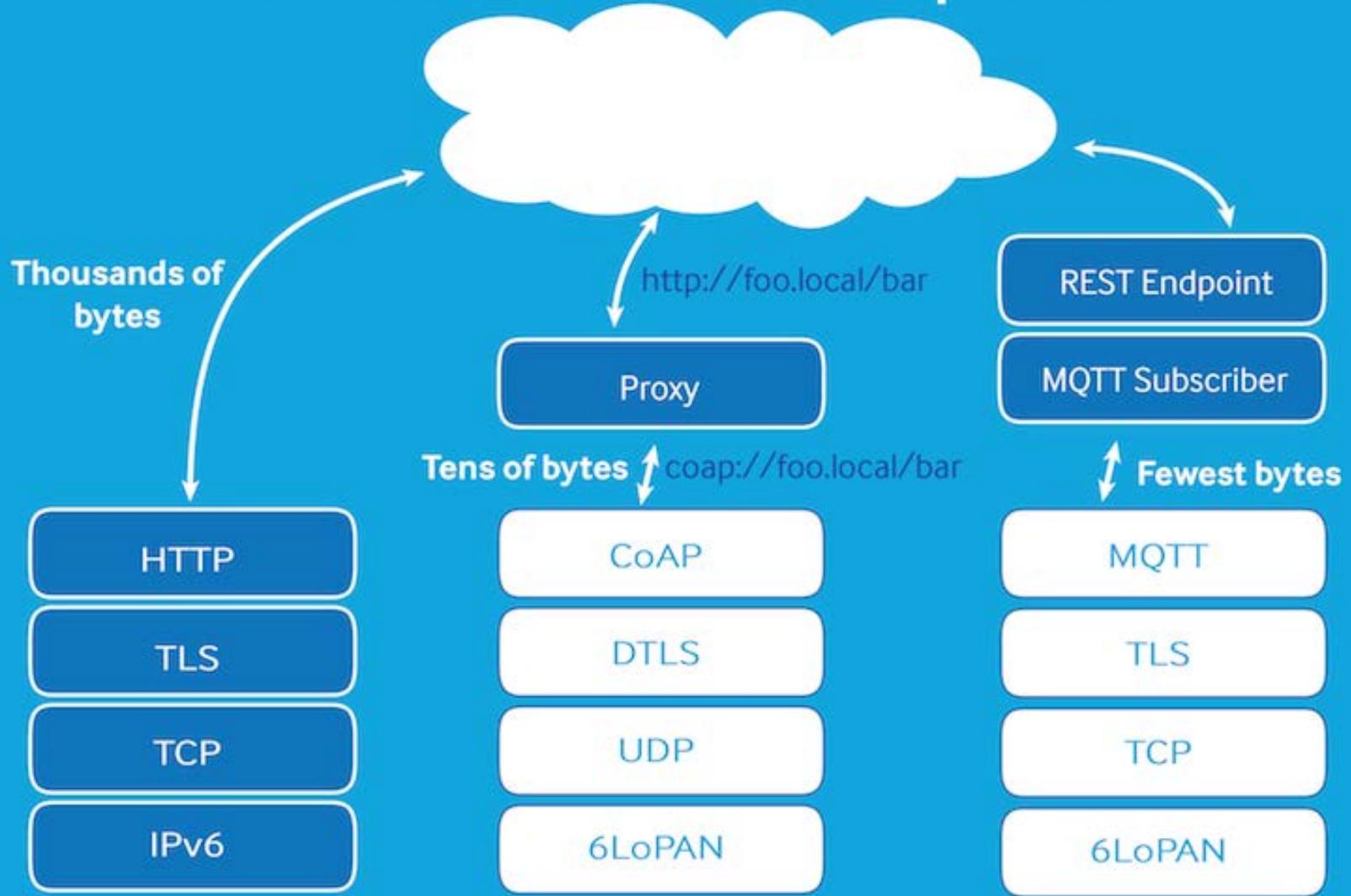
- Topología de malla en conexiones WiFi
- Desarrollado por Expressif para sus nodos ESP
- Bajo coste: no es necesario un gateway
- Bajo consumo: toda la malla puede funcionar en modo low-power
- Fácil configuración y despliegue: smartphone puede conectarse por Bluetooth
- Alta escalabilidad: hasta 1000 nodos
- Gran alcance: hasta 200m de separación entre nodos

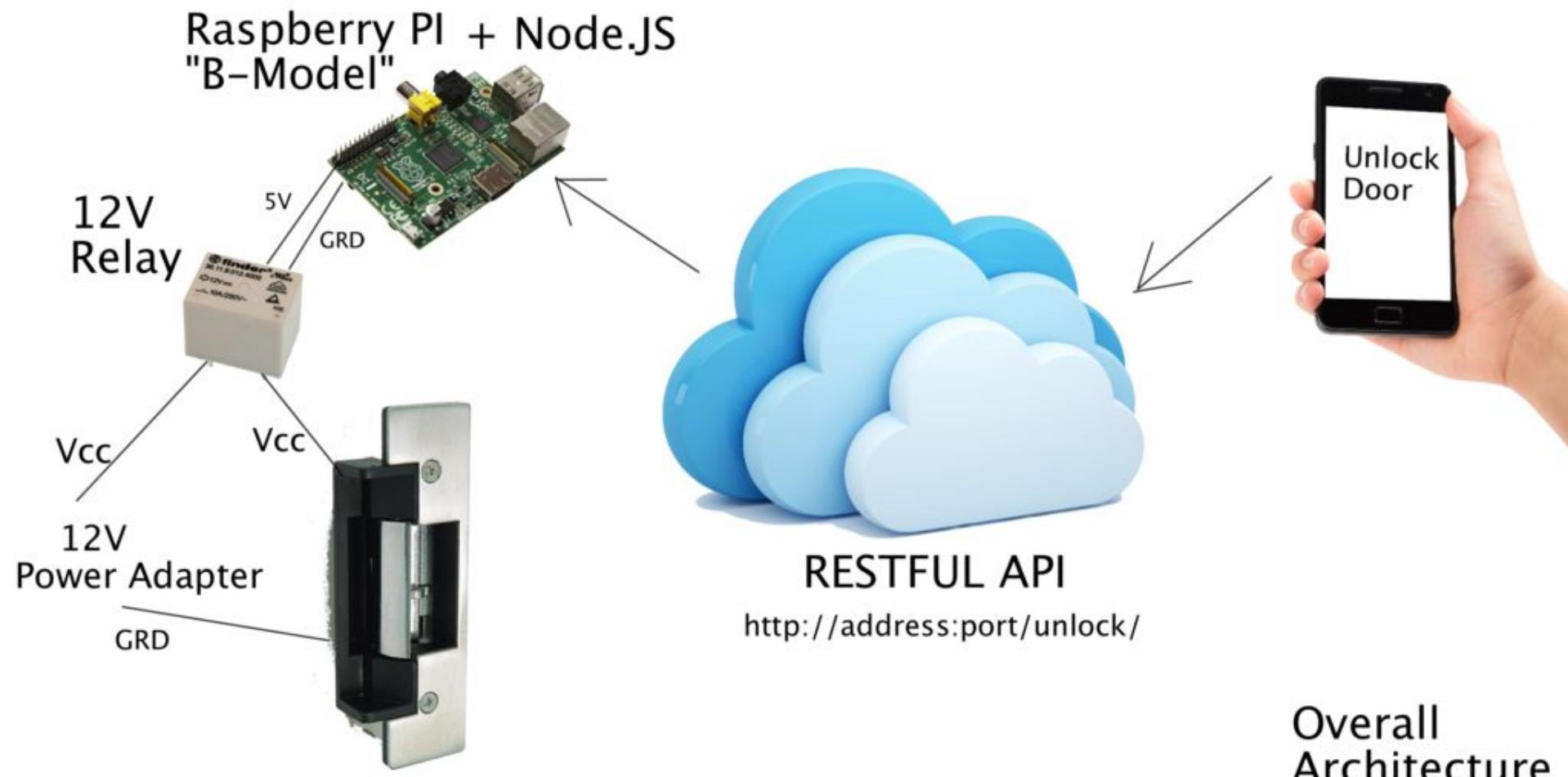


- Con la expansión del paradigma IoT, han aparecido nuevos protocolos mucho más ligeros, especialmente en la capa de aplicación, muy adecuados para comunicaciones wireless
 - MQTT
 - XMPP
 - CoAP



IoT Network Protocol Comparison





Overall
Architecture