Embedded Systems: Hardware Components (part I)

Todor Stefanov

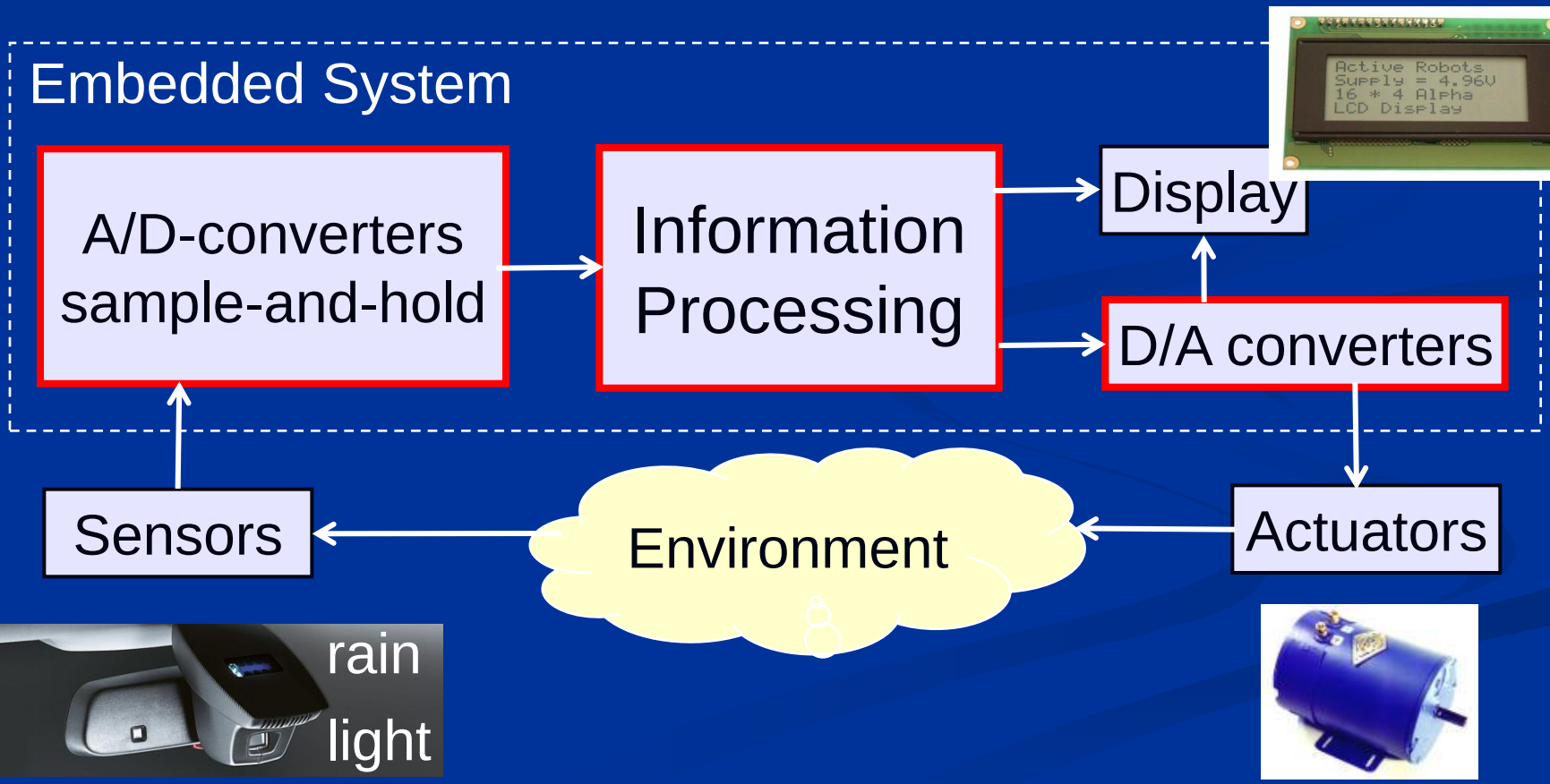
Leiden Embedded Research Center
Leiden Institute of Advanced Computer Science
Leiden University, The Netherlands

Outline

- Generic Embedded System component structure
- Sensors
- Analog-to-Digital (A/D-) converters
- Computation Components
  - General Purpose Processors (GPPs)
  - Application Specific Instruction Set Processors (ASIPs)
  - Reconfigurable Processing Units (RPUs)
  - Application Specific Integrated Circuits (ASICs)
- Memory
- Input/Output Devices
- Communication Infrastructure
- Digital-to-Analog (D/A) converters
- Actuators

Embedded Systems Hardware

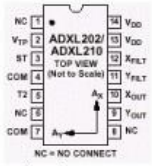
Embedded Systems hardware is frequently used in a loop (***"hardware in a loop"***):



Sensors

- Capture physical data from the environment
- Can be designed for every physical and chemical quantity
  - weight, velocity, acceleration, electrical current, voltage, temperatures, etc…
  - chemical compounds
- Many physical effects used for constructing sensors, e.g.,
  - laws of induction (generation of current in a magnetic field)
  - light-electric effects
- Huge amount of sensors designed in recent years

Examples of Sensors



Accelerometer


Gyro


Pendulum Resistive Tilt Sensors


Piezo Bend Sensor

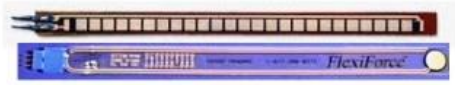

Metal Detector


Gas Sensor


Gieger-Muller Radiation Sensor


Digital Infrared Ranging

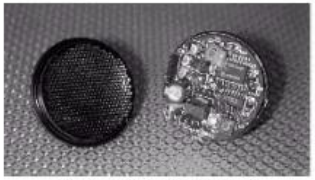

CDS Cell
Resistive Light Sensor



Resistive Bend Sensors


UV Detector

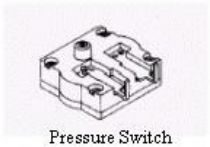

Pyroelectric Detector


Miniature Polaroid Sensor

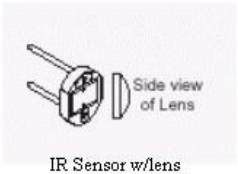

Limit Switch


Mechanical Tilt Sensors

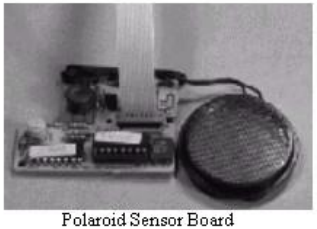

Touch Switch


Pressure Switch


IR Pin Diode



IR Sensor w/lens


Polaroid Sensor Board


Thyristor


Magnetic Sensor

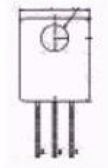

Hall Effect Magnetic Field Sensors

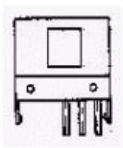

IR Reflection Sensor


IR Amplifier Sensor


IRDA Transceiver


Magnetic Reed Switch

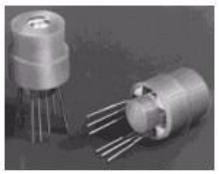

Lite-On IR Remote Receiver

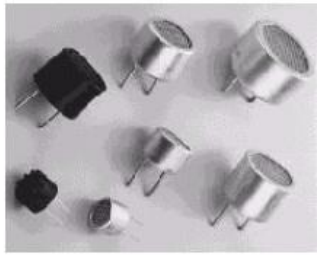

Radio Shack Remote Receiver


IR Modulator Receiver


Solar Cell


Compass


Compass


Piezo Ultrasonic Transducers

Signals

- Sensors generate *signals*
  Definition: ***signal*** *s* is a mapping from time domain D_T to value domain D_V:
  - $s: D_T \rightarrow D_V$
  - D_T : continuous or discrete time domain
  - D_V : continuous or discrete value domain

Digital computers require **discrete sequences of physical values,** i.e., $\boldsymbol{D_T}$ should be **discrete time domain.**

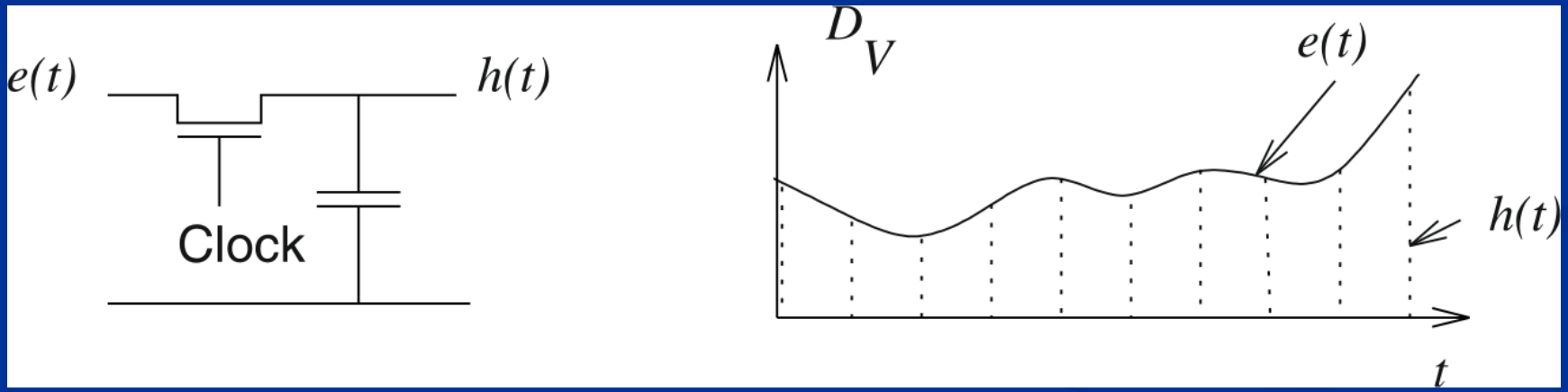
Digital computers require **digital/binary form of physical values,** i.e., $\boldsymbol{D_v}$ should be **discrete value domain.**

- How to do this for continuous physical signals?
  - Sample-and-hold circuits (discretization in time or *sampling*)
  - A/D-converters (discretization of values or *quantization*)

Discretization in Time: Sample-and-hold circuit

Clocked transistor + capacitor;

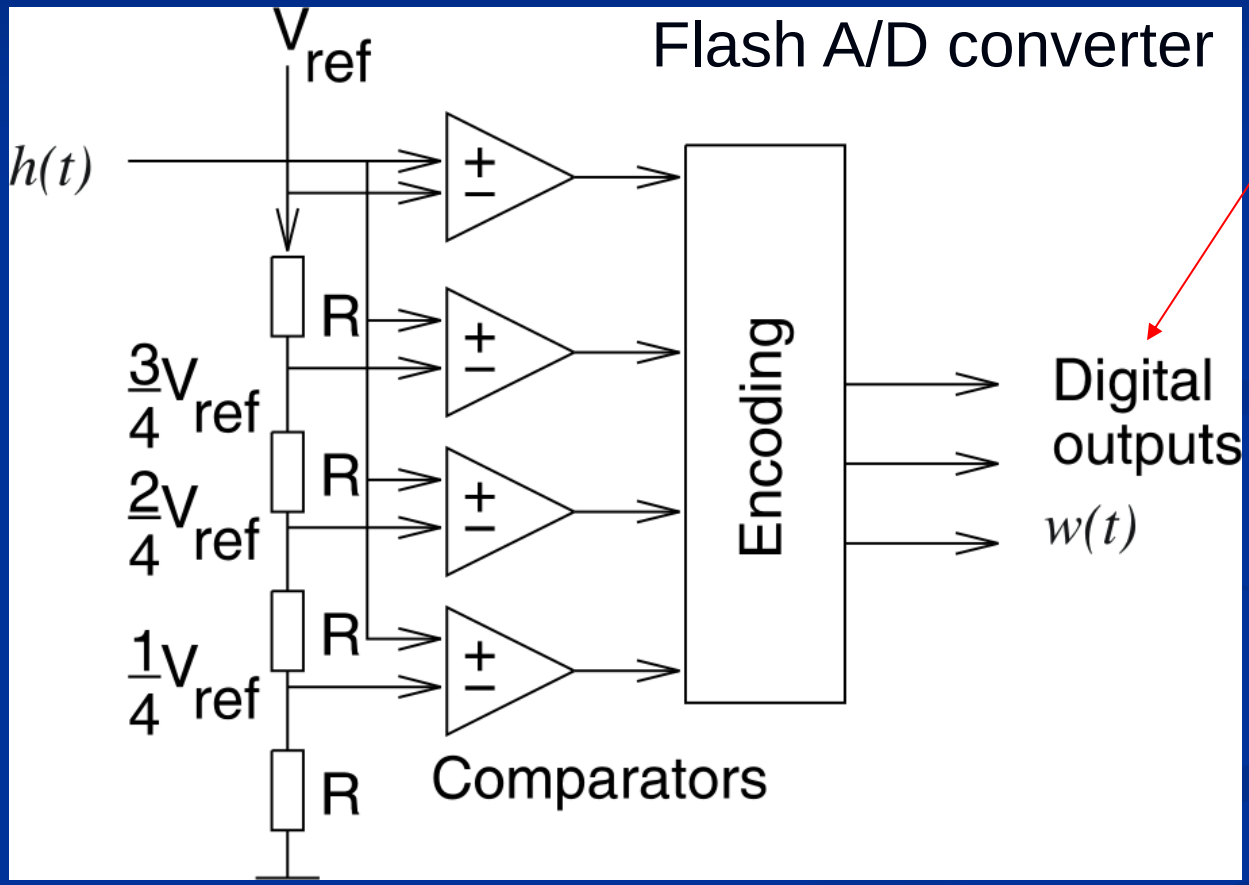
Capacitor stores sequence values



$e(t)$ is a mapping $\mathbb{R} \rightarrow \mathbb{R}$

$h(t)$ is a ***sequence*** of values or a mapping $\mathbb{Z} \rightarrow \mathbb{R}$

Discretization of Values: Analog-to-Digital (A/D)-converters (1)



- Encodes input number of most significant ‘1’ as an unsigned number, e.g. “1111” -> “100”, “0111” -> “011”, “0011” -> “010”, “0001” -> “001”, “0000” -> “000” (priority encoder)

Resolution and Speed of Flash A/D converter

- Resolution (in bits): number of bits produced
- Resolution Q (in volts): difference between two input voltages causing the output to be incremented by 1

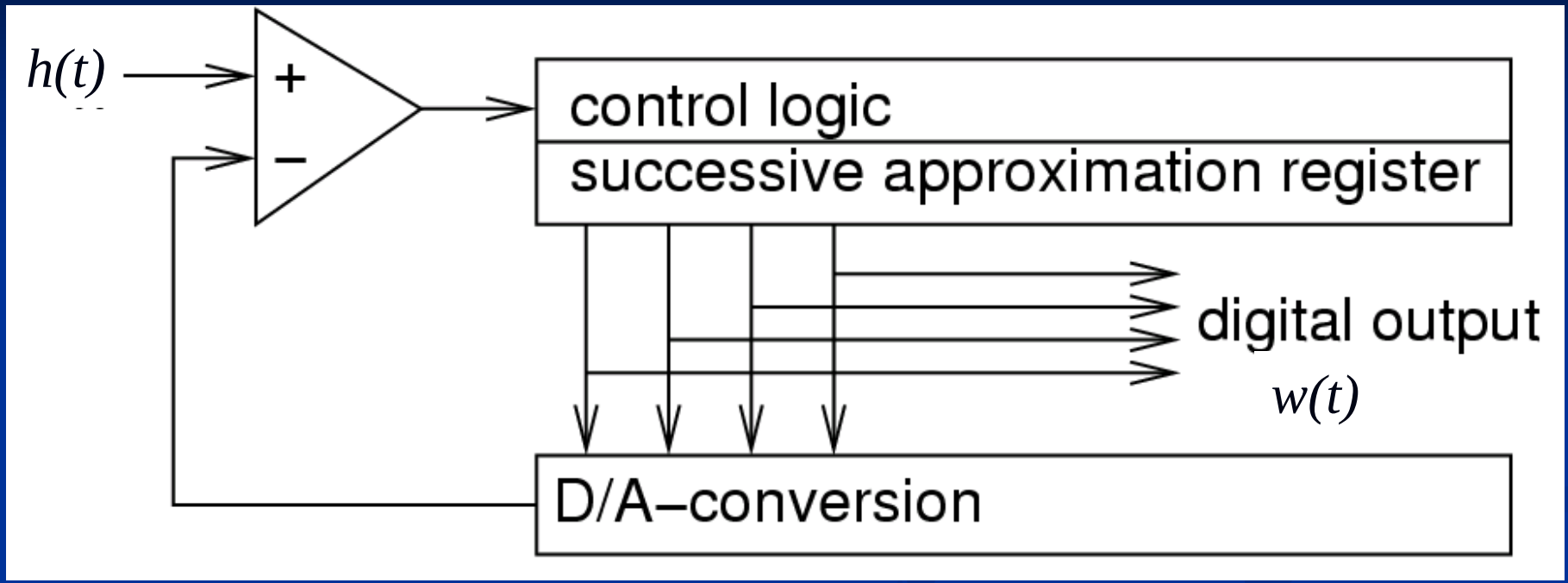
$$Q = \frac{V_{FSR}}{n}$$

with V_{FSR}: difference between largest and smallest voltage

n: number of voltage intervals

- **Parallel comparison with reference voltage**
  - Speed: $O(1)$
  - Hardware complexity: $O(n)$
- **Applications**: e.g. in video processing

Discretization of Values: A/D-converters (2)



Key idea: binary search:

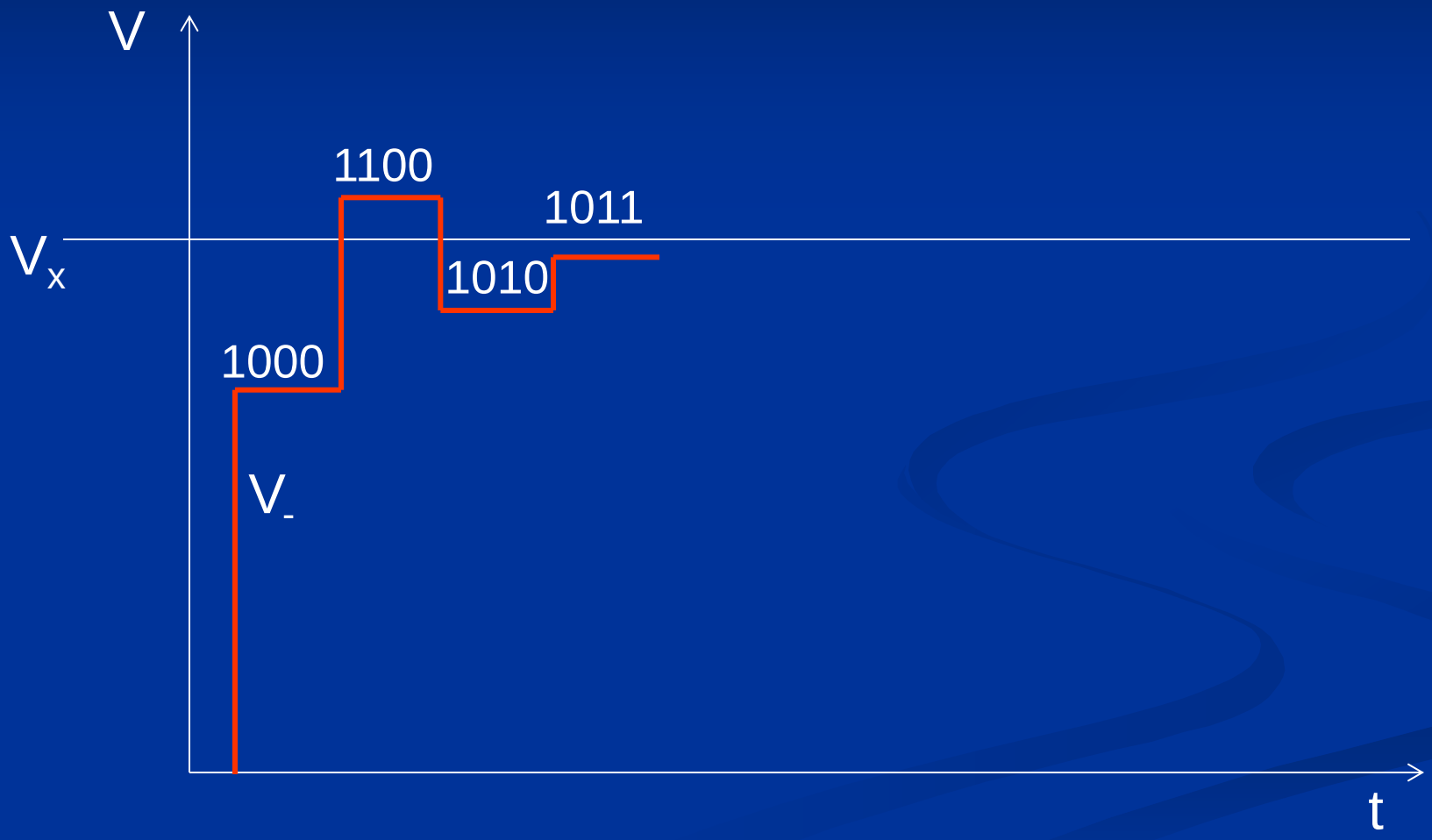
Set MSB = '1'

if too large: reset MSB

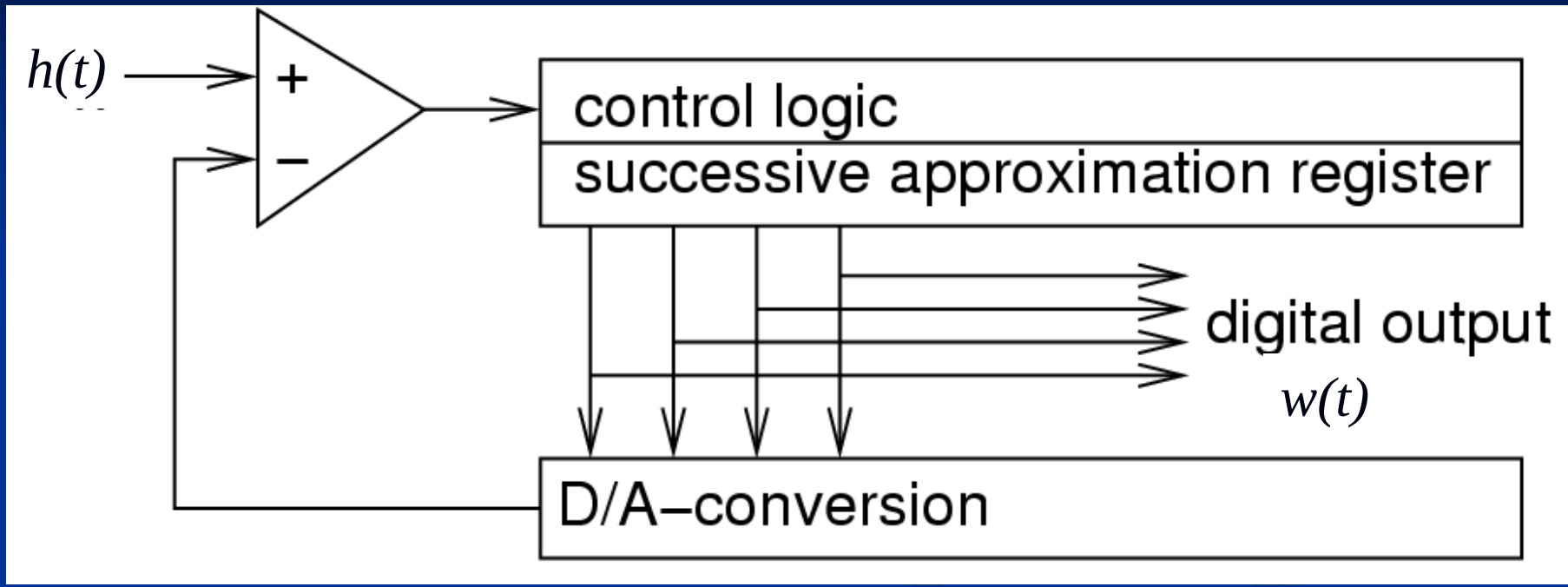
Set MSB – 1 = '1'

if too large: reset MSB-1

Successive Approximation



Discretization of Values: A/D-converters (2)

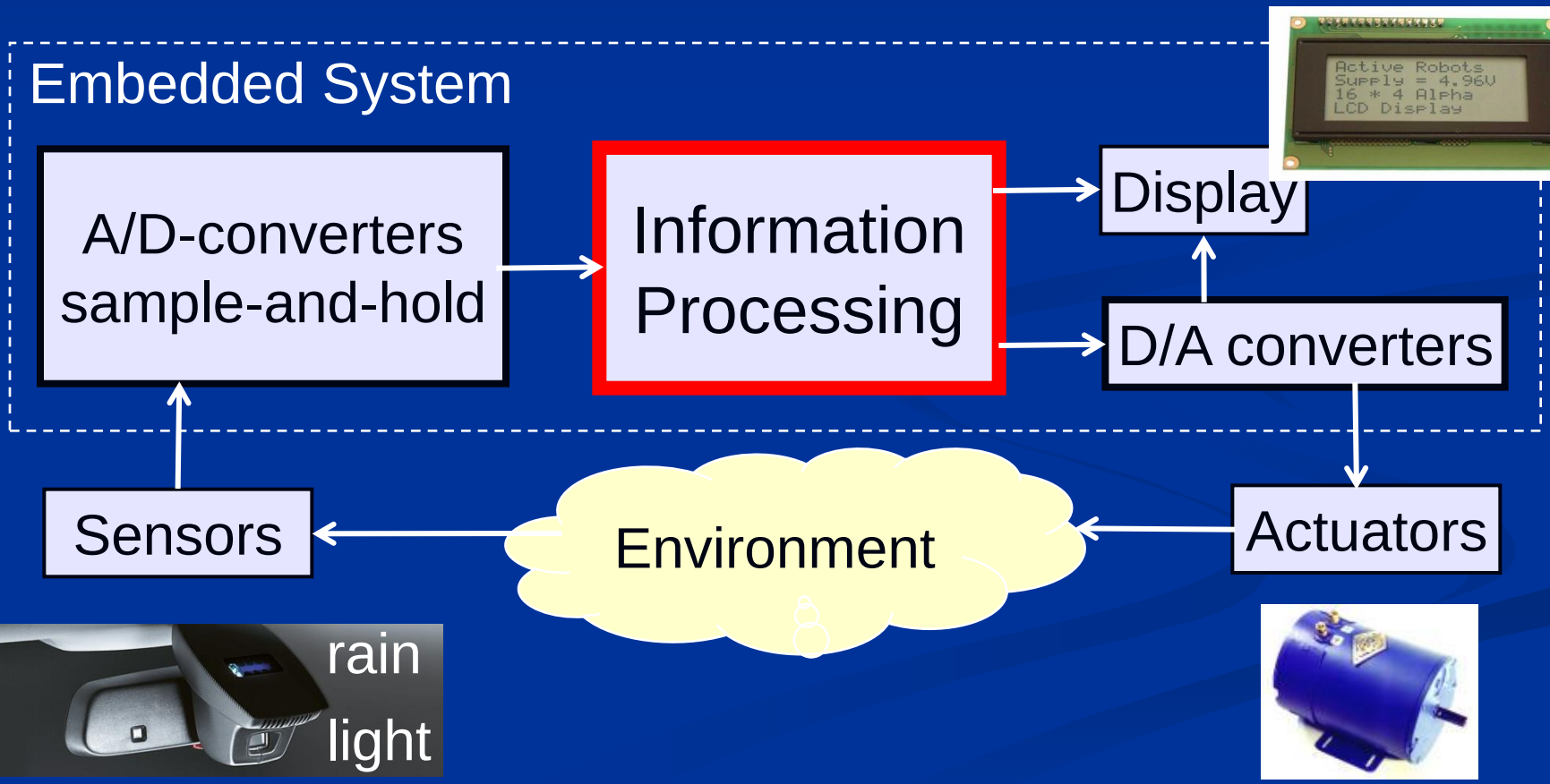


Speed: *O(log2(n))*
Hardware complexity: *O(log2(n))*
n= # of distinguished voltage levels;

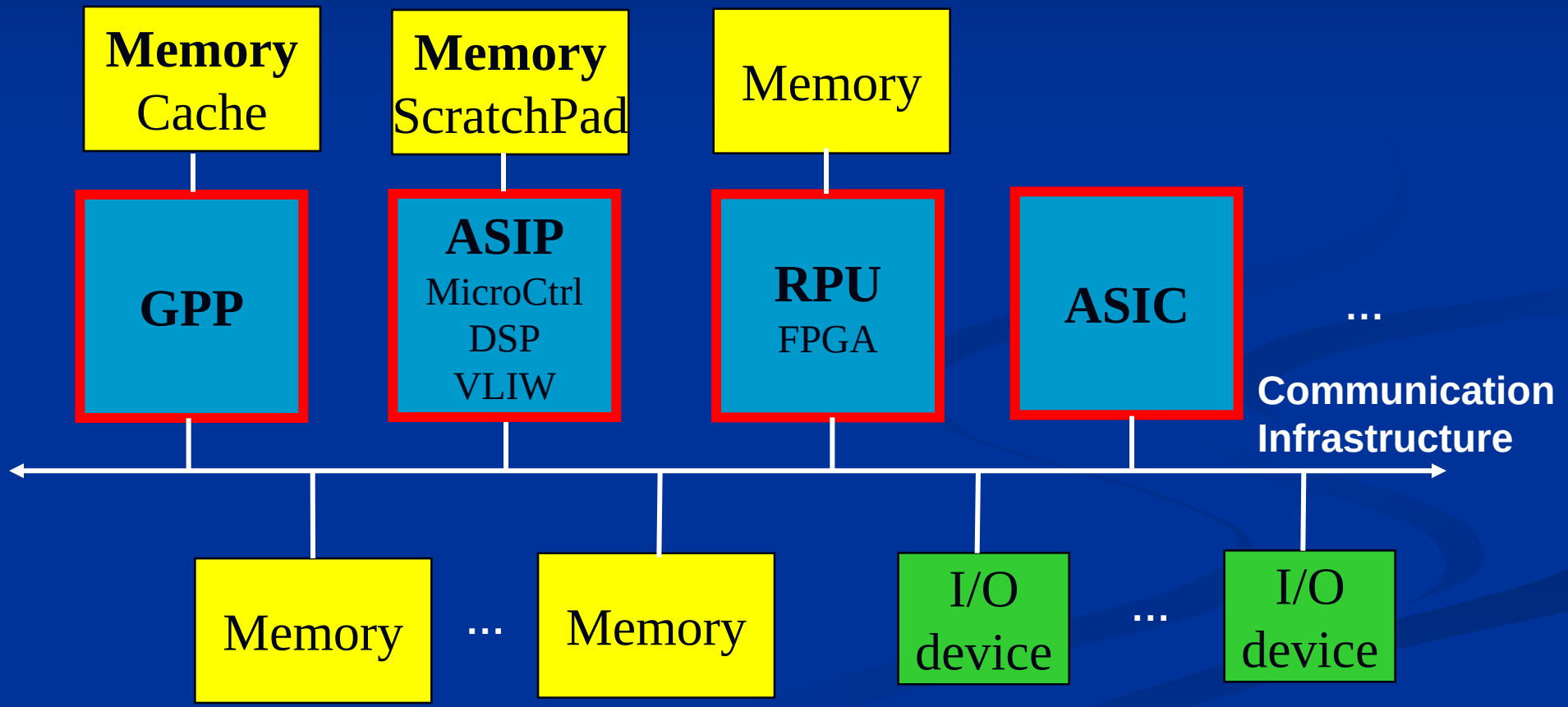
slow, but high precision possible

Embedded Systems Hardware

Embedded Systems hardware is frequently used in a loop (***"hardware in a loop"***):

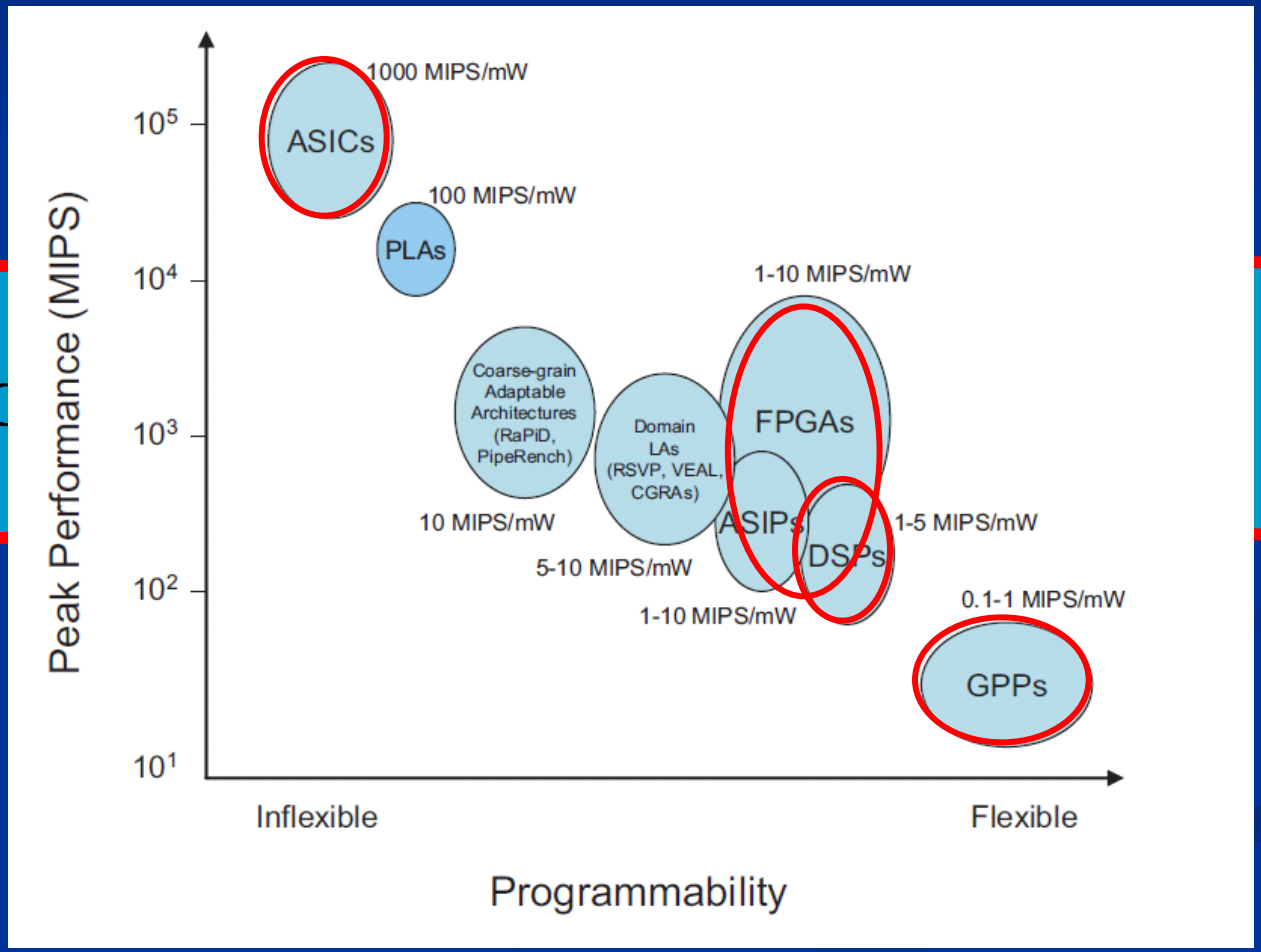


Information Processing System: Computation Components



Why Implementation Alternatives?

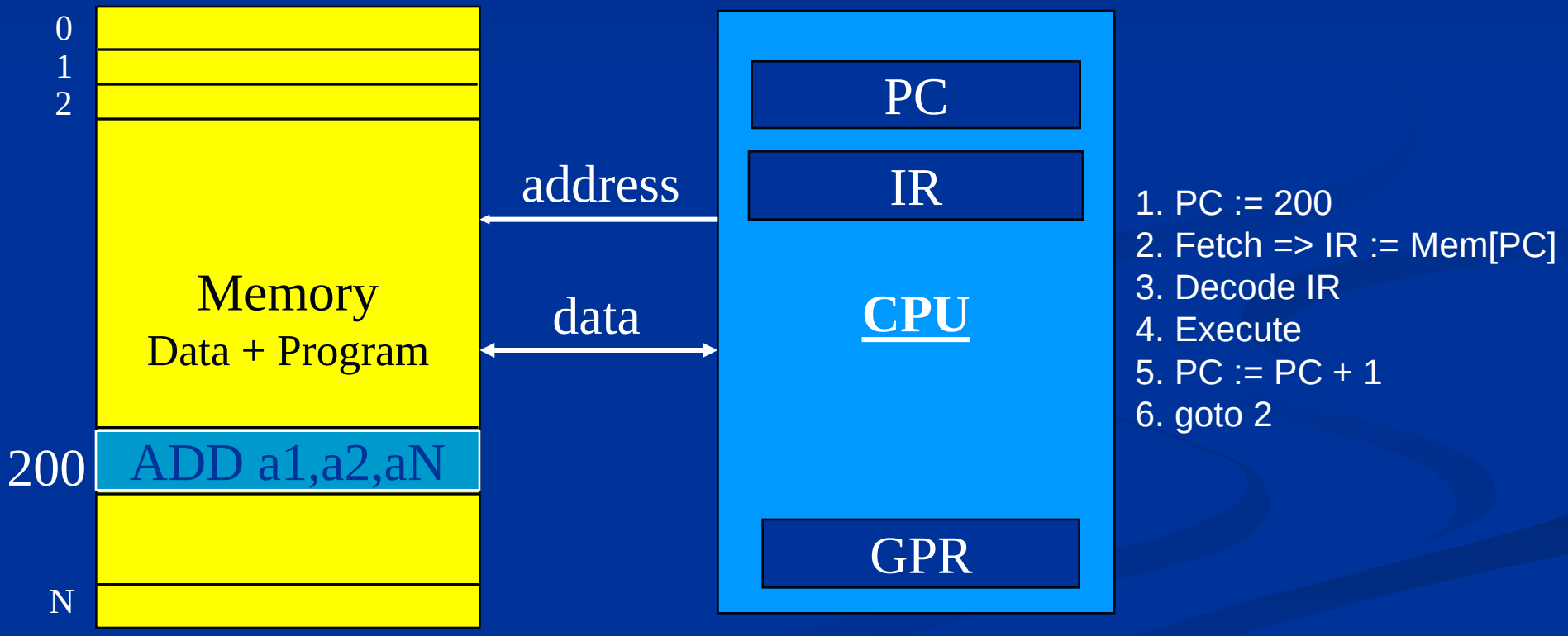
Trade-off between Flexibility and Performance/Power Efficiency



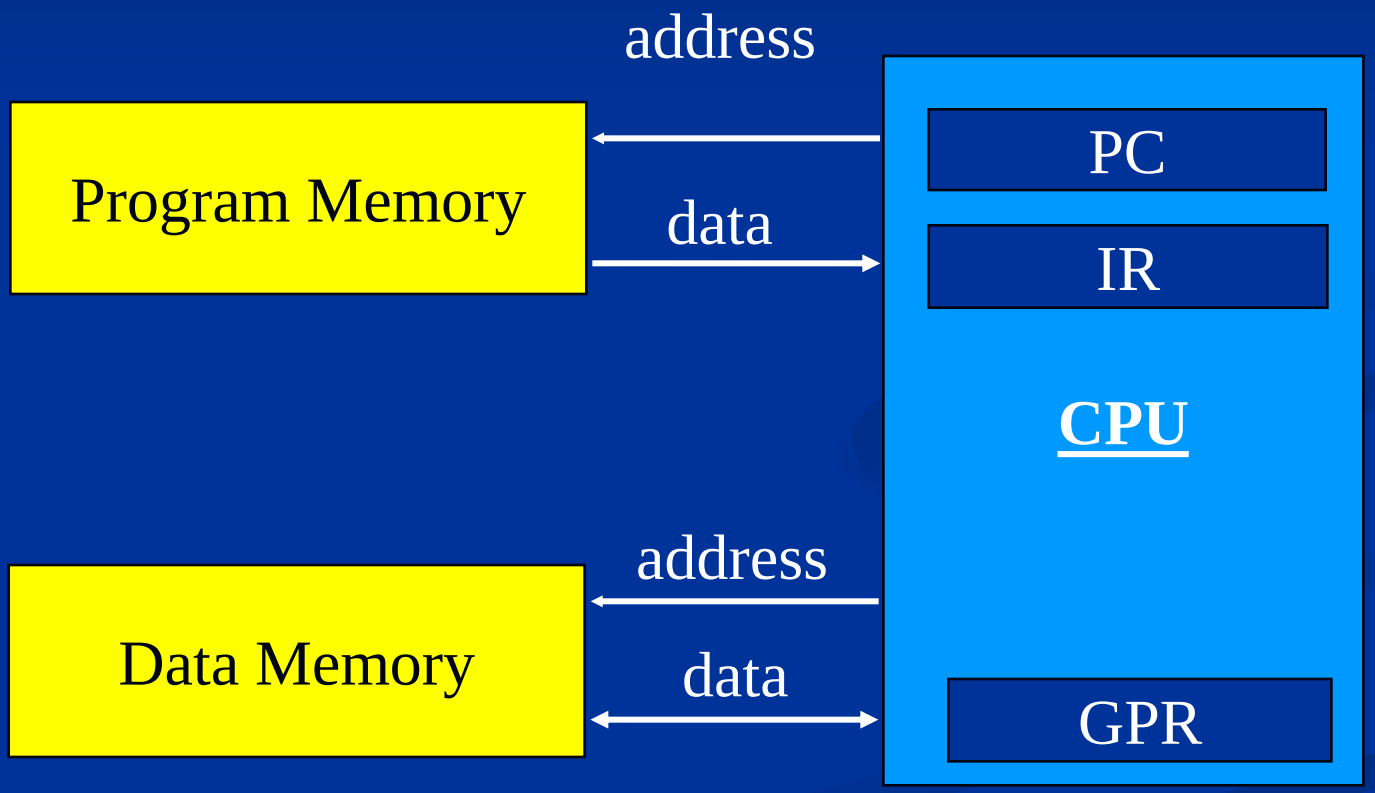
General Purpose Processors (GPPs)

- Can achieve high performance
  - Highly optimized circuits
  - Use of instruction-level parallelism
    - superscalar: dynamic scheduling of instructions
    - super-pipelining: instruction pipelining, branch prediction, speculation
  - Complex memory hierarchy – caches
- Not suitable for real-time applications
  - Execution times are highly unpredictable
    - Due to caches and dynamic decisions (scheduling of instructions, etc.)
- Properties
  - Good average performance for large application mix
  - High power consumption

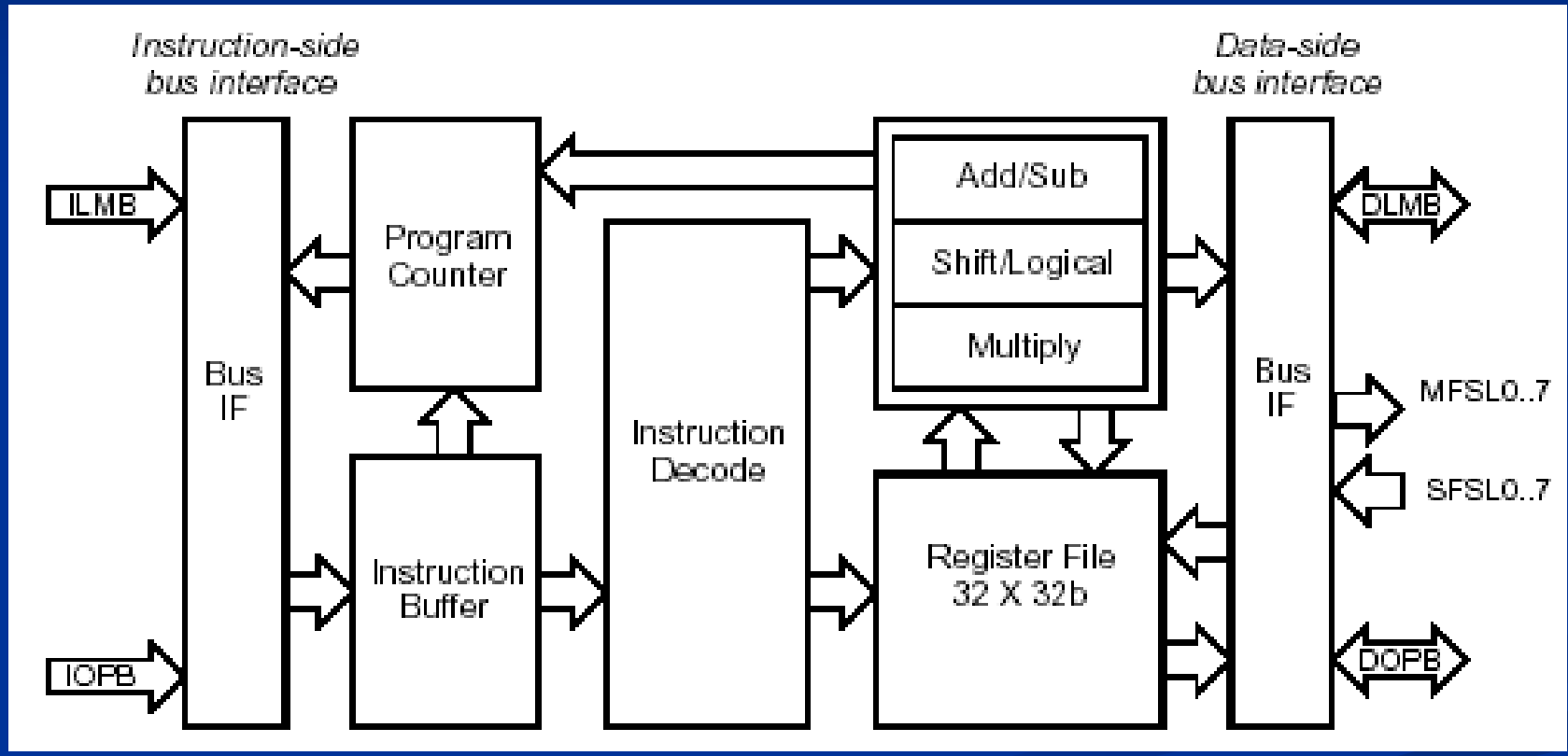
GPP + Memory (von Neumann architecture)



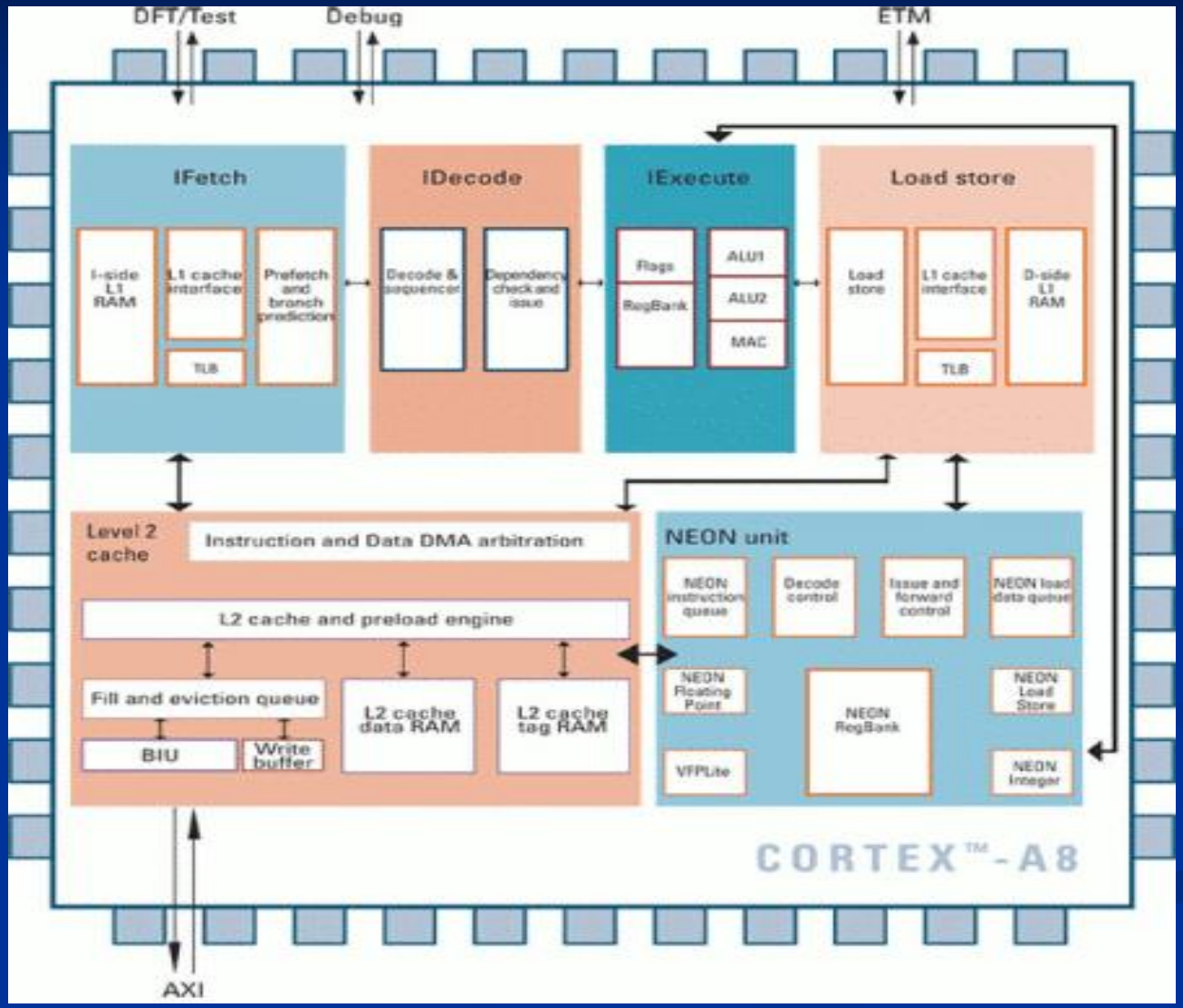
GPP + Memory (Harvard architecture)



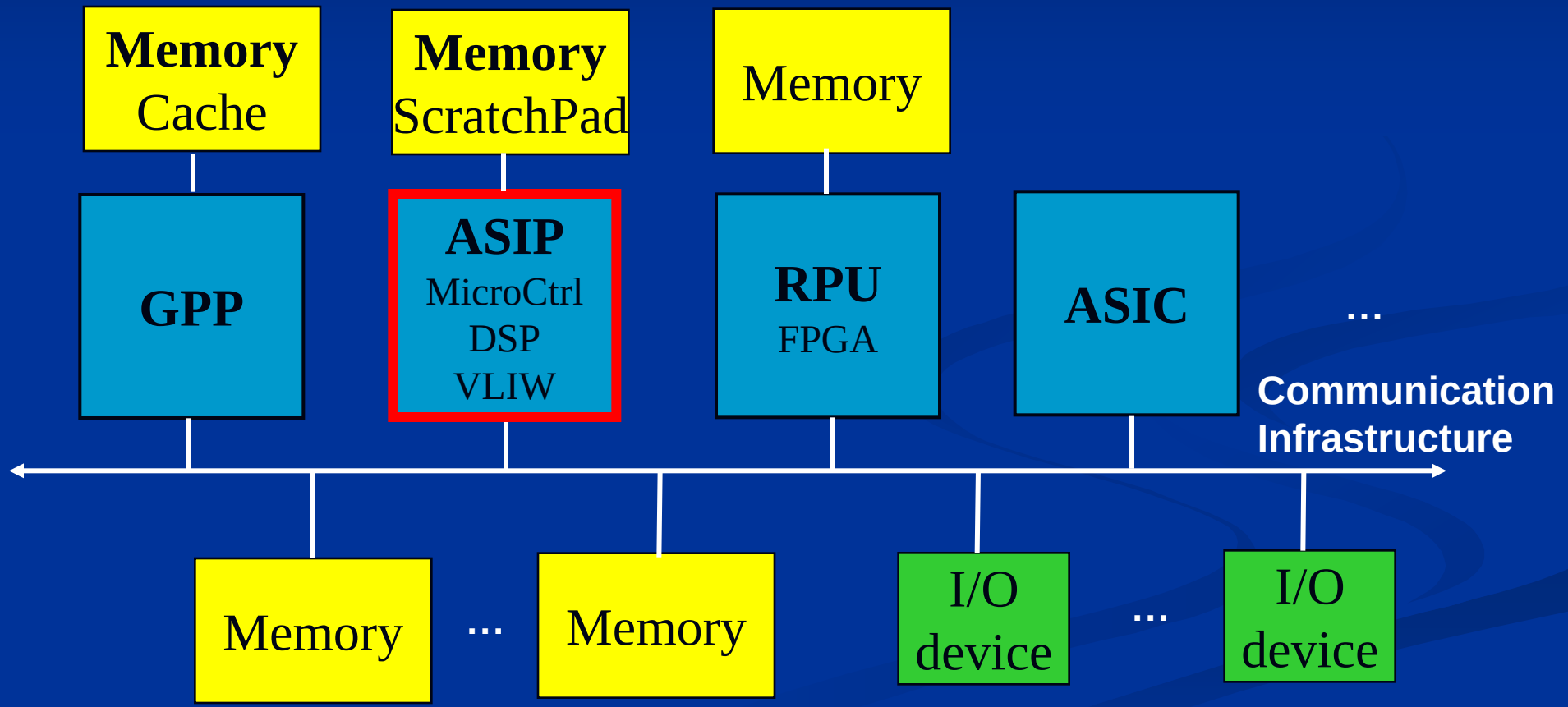
Example of simple embedded GPP: MicroBlaze



Example of complex embedded GPP: ARM 8



Information Processing System: Computation Components

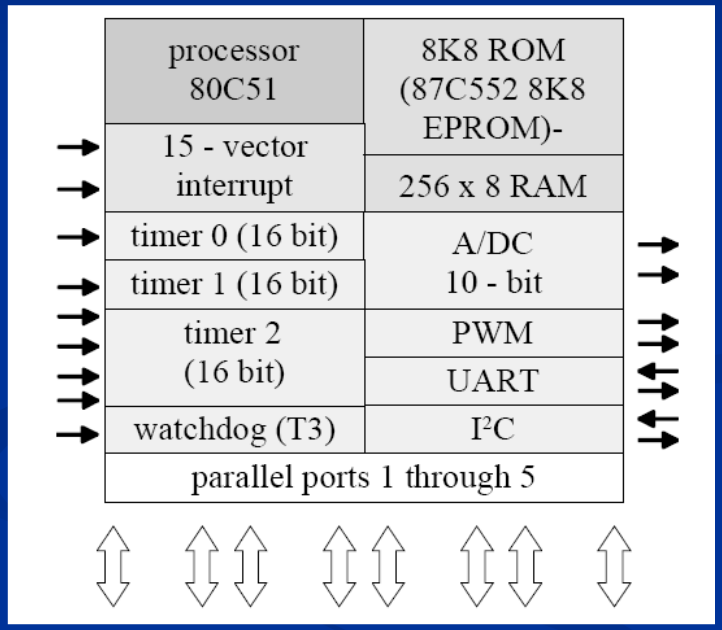


Application Specific Instruction Set Processors (ASIPs)

- Micro Controllers (MicroCtrl)
  - Used in Control Systems
  - Reactive systems with event driven behavior
  - Application examples: cars, consumer electronics (washing machines, dishwashers etc.)
- Digital Signal Processors (DSPs)
  - Used in Data Processing Systems
  - Streaming-oriented systems with mostly periodic behavior
  - Application examples: signal processing
- Very Long Instruction Word Processors (VLIWs)
  - Used in Data Processing Systems
  - Application examples: image processing

Micro Controllers

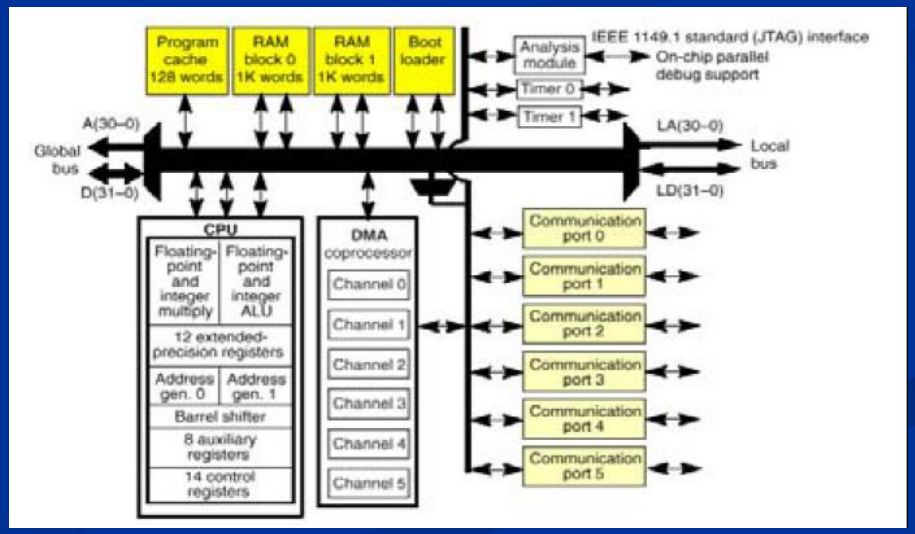
- Control-dominant applications
  - Supports process scheduling and synchronization
  - Preemption (interrupt), context switch
- Low power consumption
  - low frequency (up to 12MHz)
- Peripheral units often integrated
- Suited for real-time applications
  - short latency times (during context switch)
  - time predictable (no caches)



Philips 83 C552: 8 bit-8051 based microcontroller

Digital Signal Processors

- Optimized for data-flow applications
- Parallel hardware units
- Specialized instruction set
- High data throughput
- Zero-overhead loops
- Specialized memory
- Suited for real-time applications



TMS320C40 Block Diagram

MAC (Multiply & Accumulate)

```
// Product of two vectors a[] and b[]
float X = 0.0;
for ( i = 1; i <= N; i++ ) {
     X = X + a[i]*b[i];
}
```

TMS320C3x Assembler (Texas Instruments)

```
   LDF   0, R0
   LDF   0, R1
   RPTS  N    // Zero-overhead loop (repeat next instruction N times)
   MPYF3 *(AR0)++, *(AR1)++, R0
|| ADDF3 R0, R1, R1
```

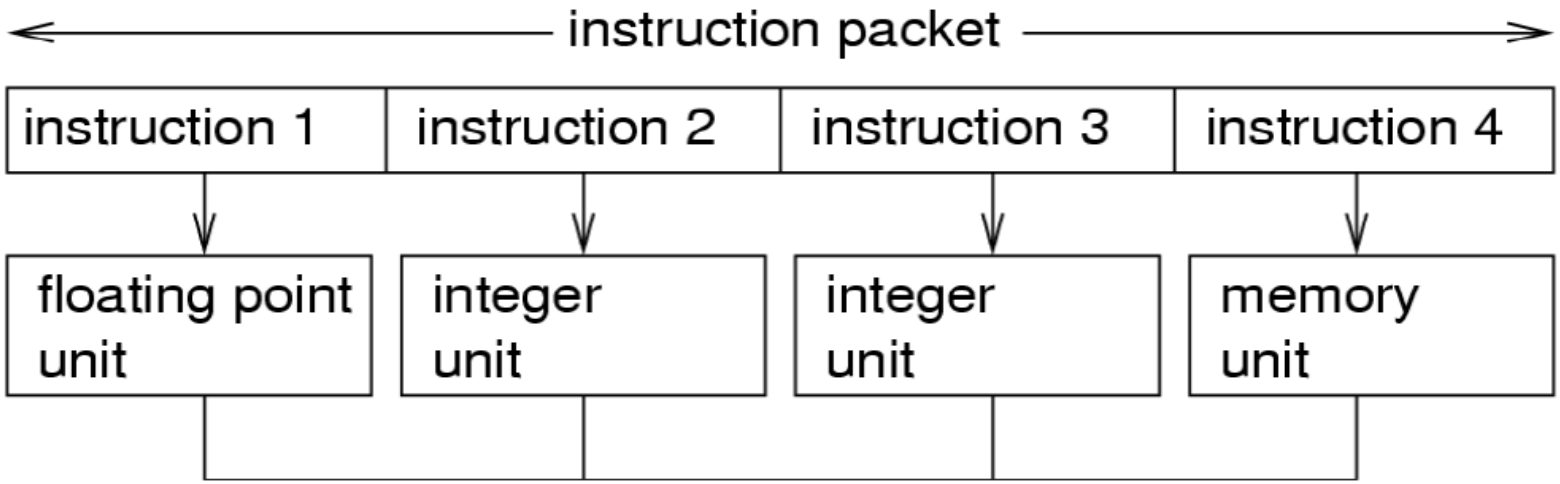
} MAC instruction

Very Long Instruction Word Processors

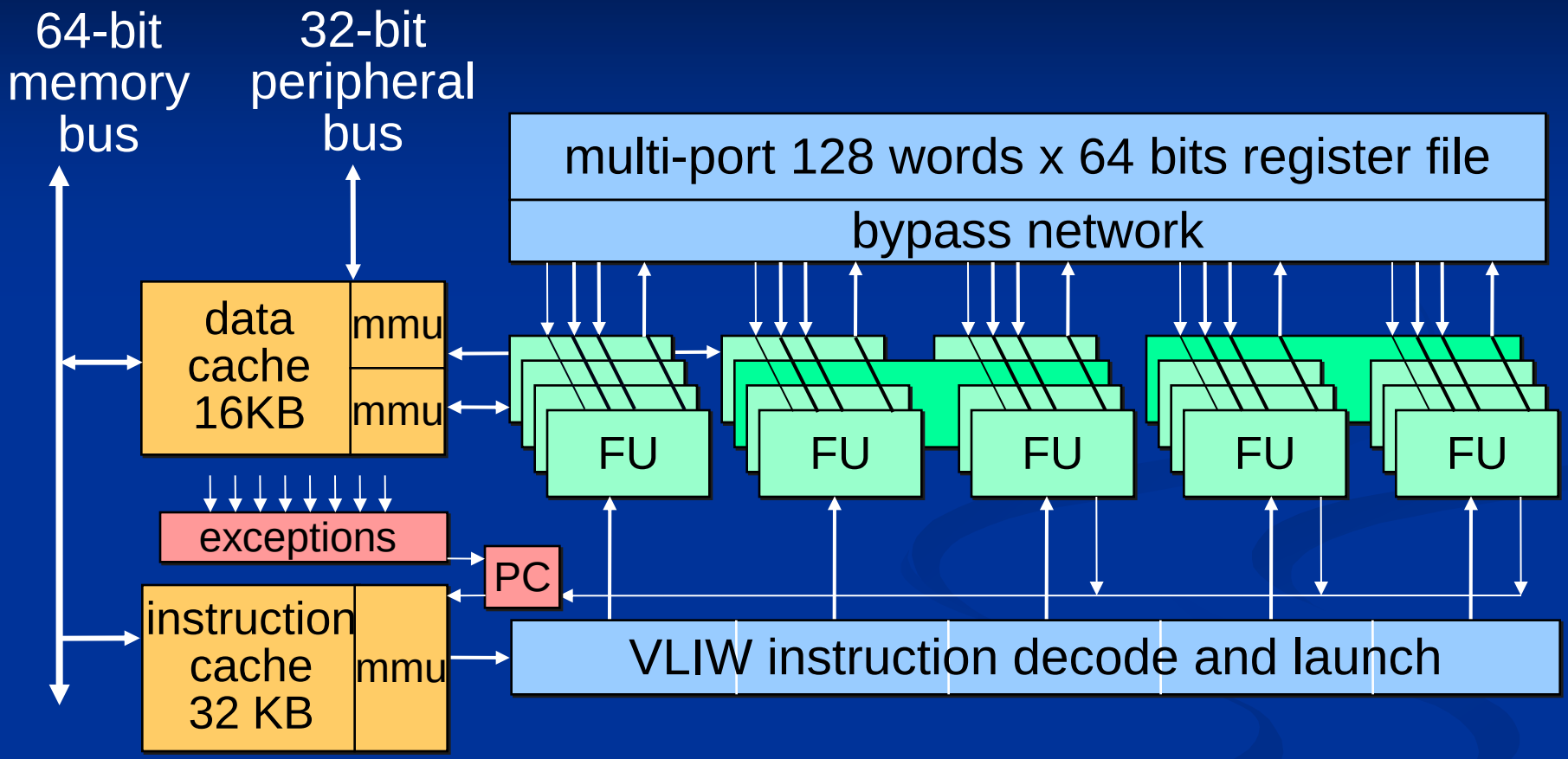
Key idea: Detection of possible instruction parallelism by the compiler, not by hardware at run-time (inefficient)

VLIW: parallel instructions encoded in one long word, each instruction controlling one functional unit

VLIW processors are an example of the so called ***Explicit Parallelism Instruction Computers (EPIC)***

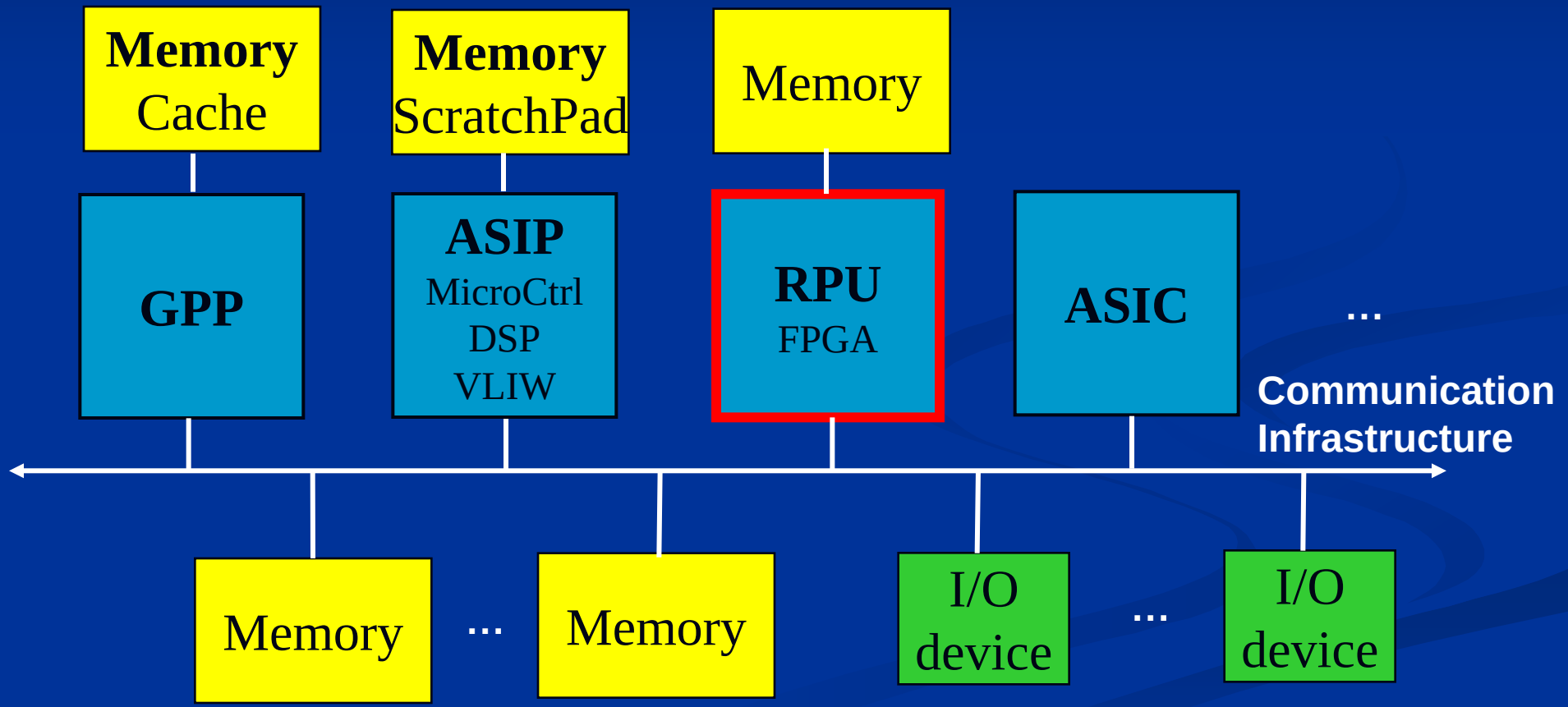


Example: Philips TriMedia VLIW CPU



5 issue slots (functional units FU),
therefore up to 5 instructions can be executed in parallel

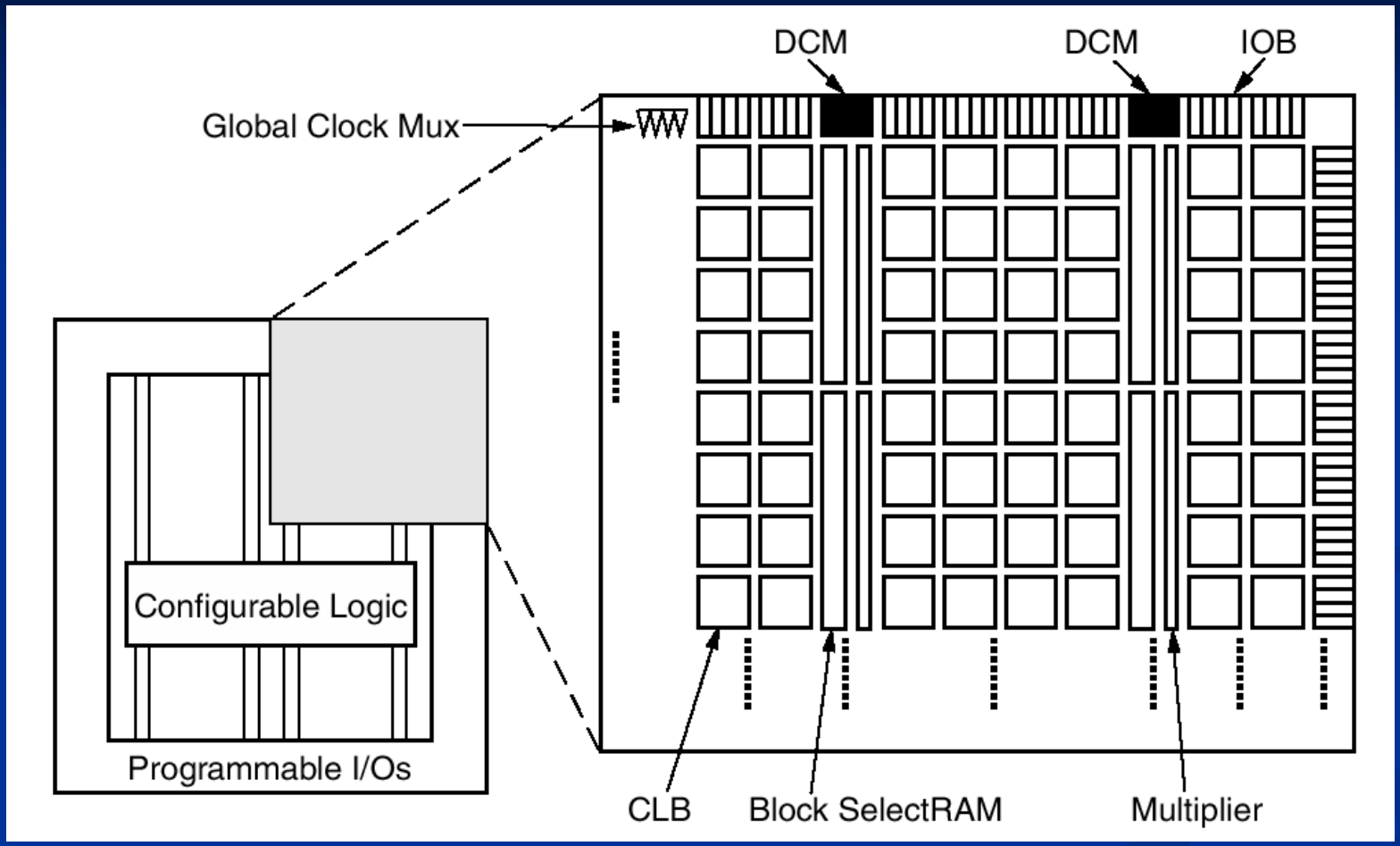
Information Processing System: Computation Components



Reconfigurable Processing Units (RPUs)

- Full custom HW may be too expensive, SW maybe too slow
- Combine speed of HW with flexibility of SW
  - HW with programmable functions and interconnect
  - HW (Re-)Configurable at design-time or at run-time (dynamic reconfiguration)
- Field Programmable Gate Arrays (FPGAs)
  - Currently the most sophisticated and used RPUs
  - Applications
    - Fast and very cheap prototyping of (MP-)SoCs
    - Encryption
    - Fast “object recognition“ (medical and military)
    - Adapting mobile phones to different standards
- Very popular devices from
  - XILINX (Virtex 6, Virtex 7, Virtex UltraScale+)
  - Altera, Actel and others

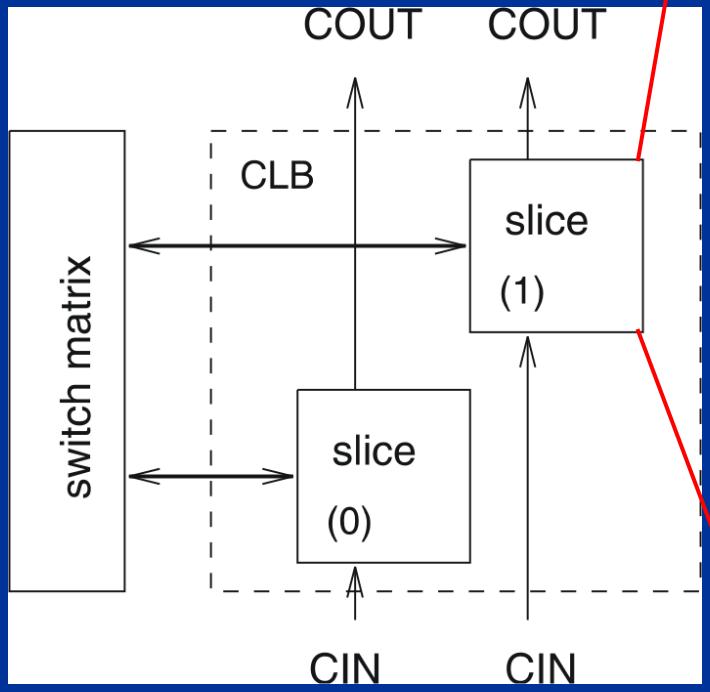
Floor Plan of Virtex FPGAs

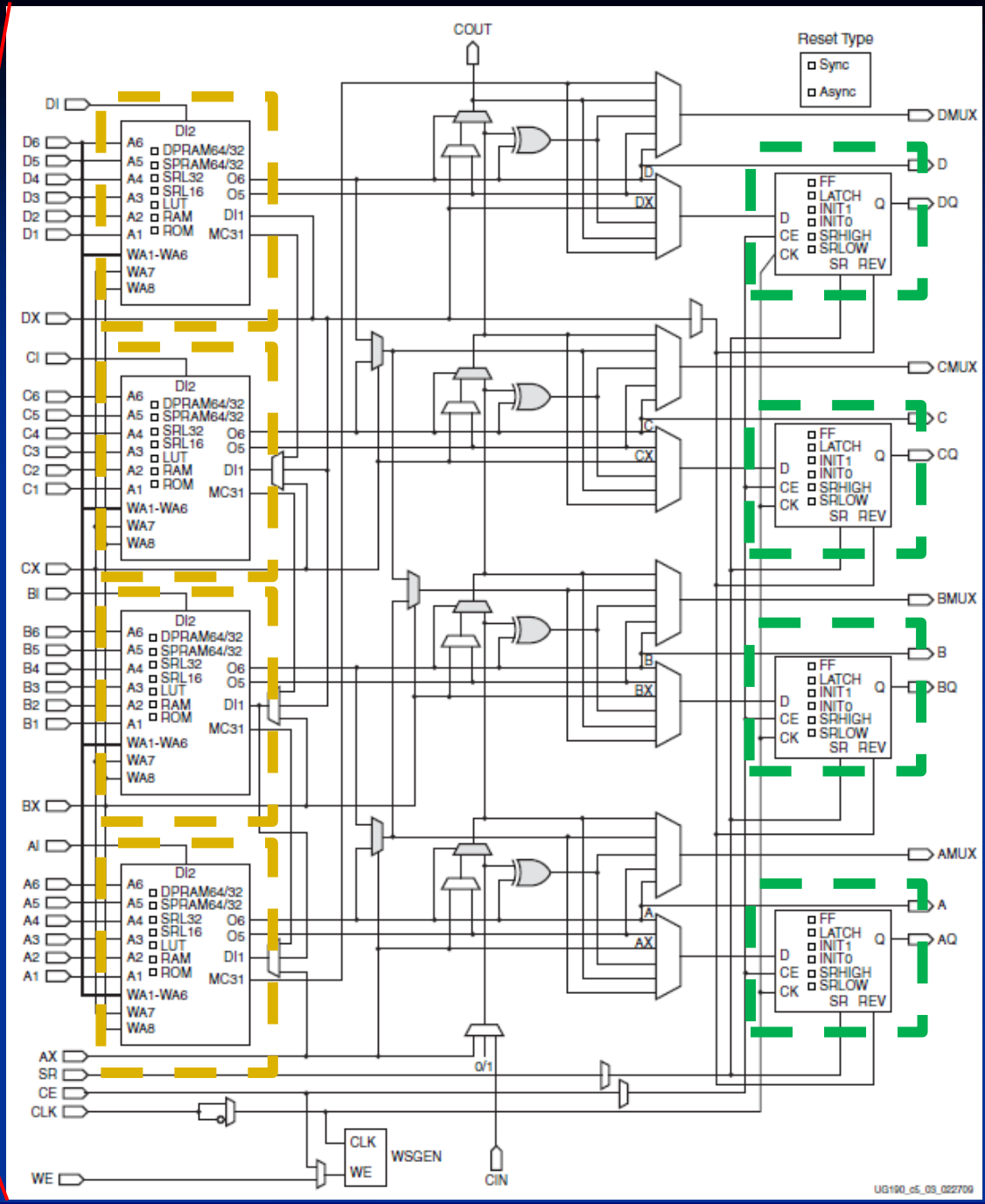


- Configurable Logic Block (CLB)
- Digital Clock Manager (DCM)
- Input/Output Blocks (IOB)

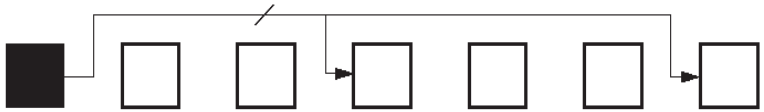
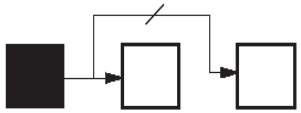
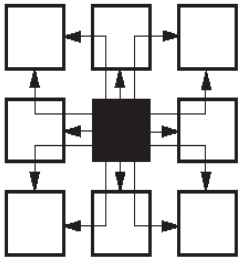
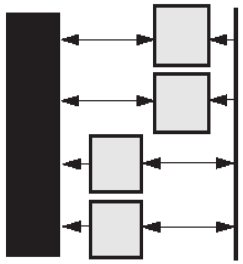
CLB Structure

- 1-bit registers used as
  - Flip-Flops or Latchs
- 64-bit memories used as
  - Look-up tables to implement any Boolean function of up to 6 variables
  - Memories for storing data
  - Shift registers

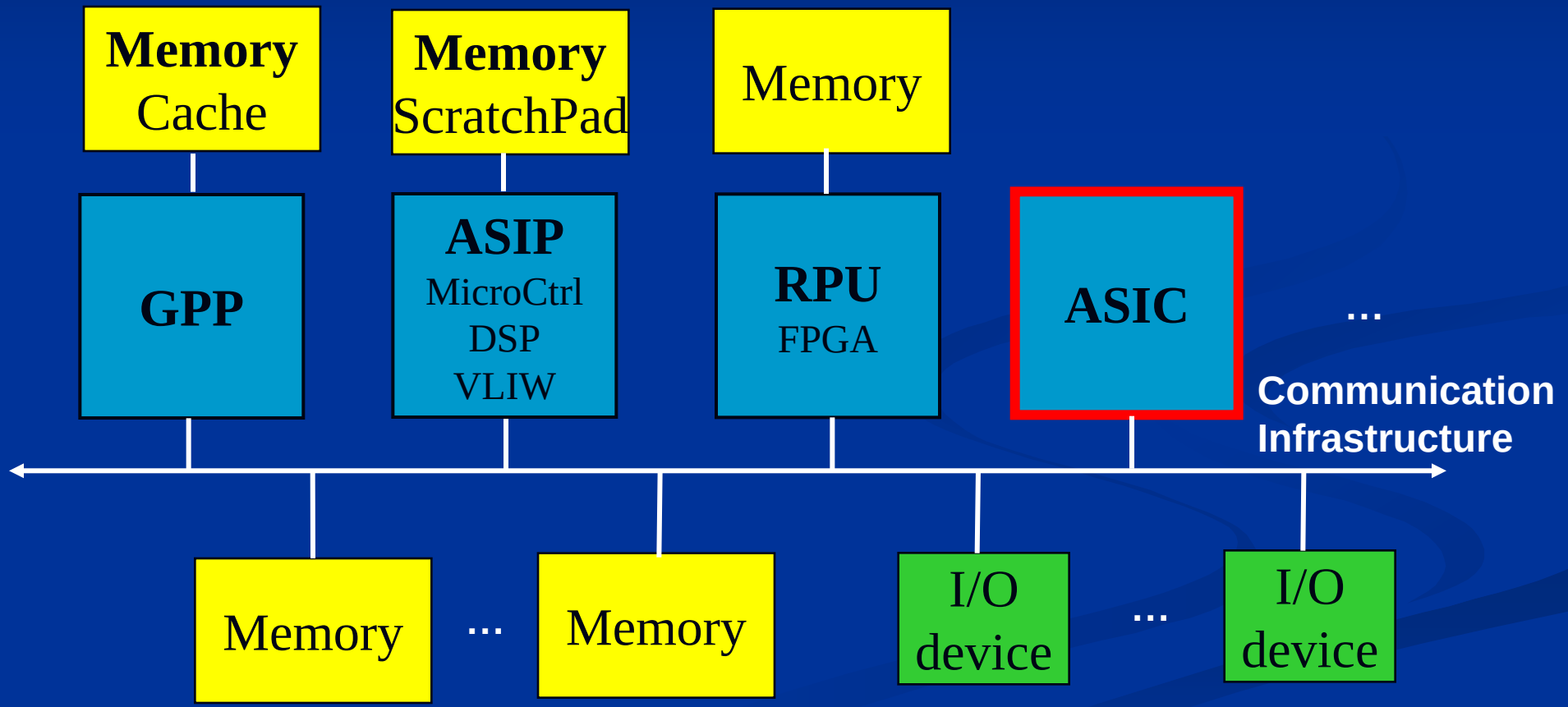




Interconnect Infrastructure

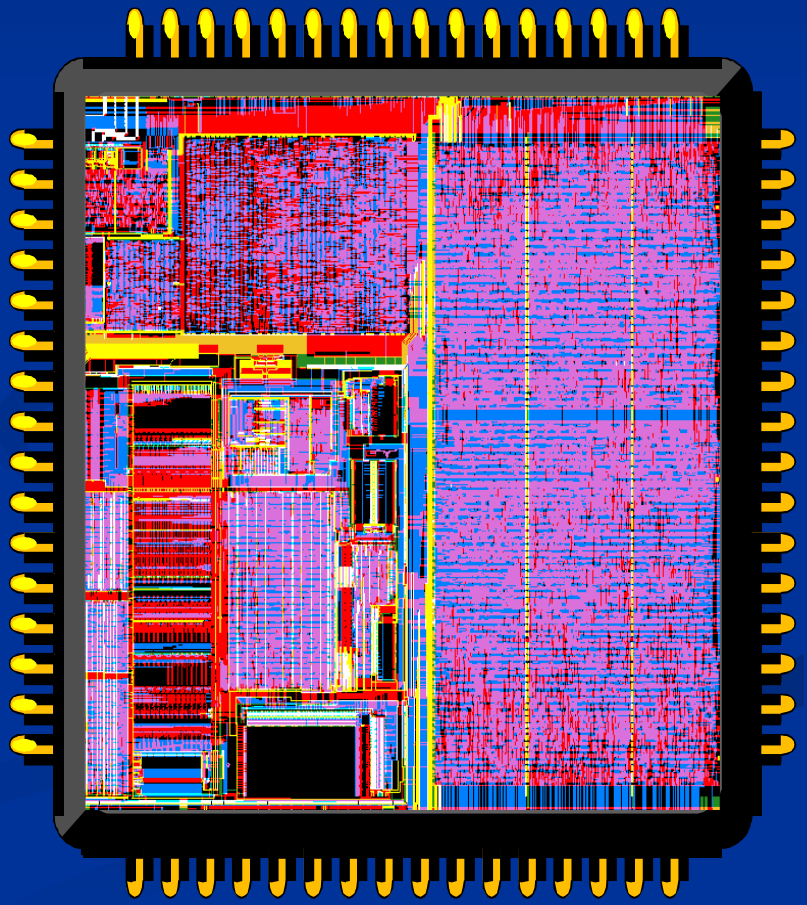
24 Horizontal Long Lines 24 Vertical Long Lines	
120 Horizontal Hex Lines 120 Vertical Hex Lines	
40 Horizontal Double Lines 40 Vertical Double Lines	
16 Direct Connections (total in all four directions)	
8 Fast Connects	

Information Processing System: Computation Components

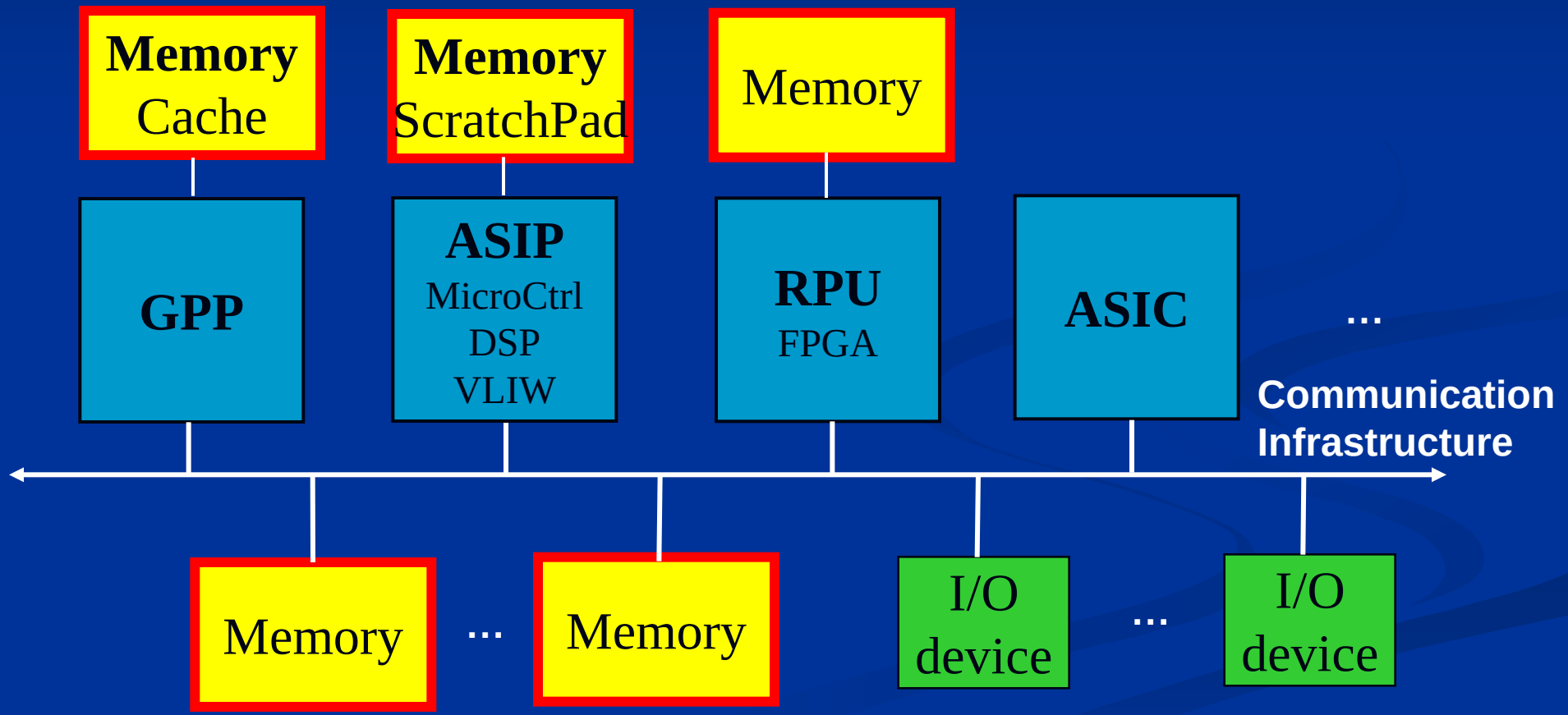


Application Specific Integrated Circuits (ASICs)

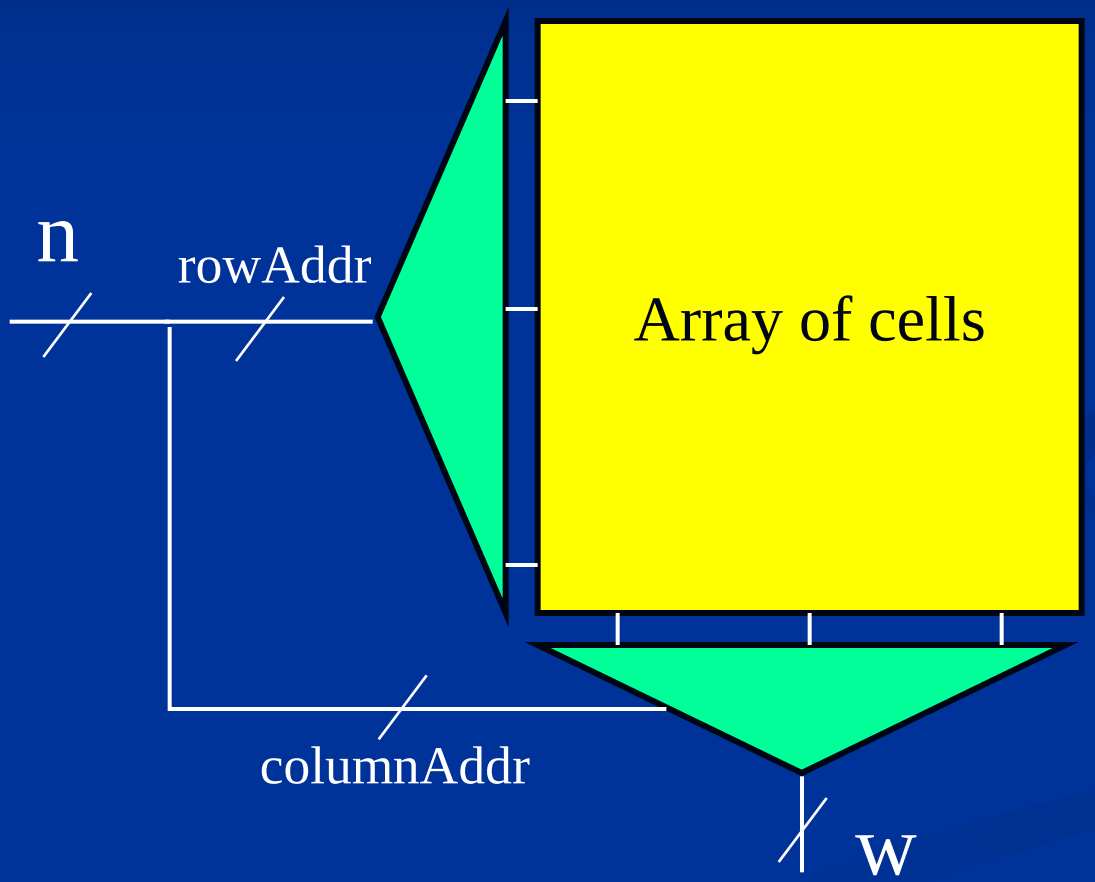
- Custom-designed circuits necessary
  - if ultimate speed or
  - energy efficiency is the goal and
  - large numbers can be sold
- Approach suffers from
  - long design times,
  - lack of flexibility (changing standards)
  - high costs, i.e., Millions of $ mask costs



Information Processing System: Memory



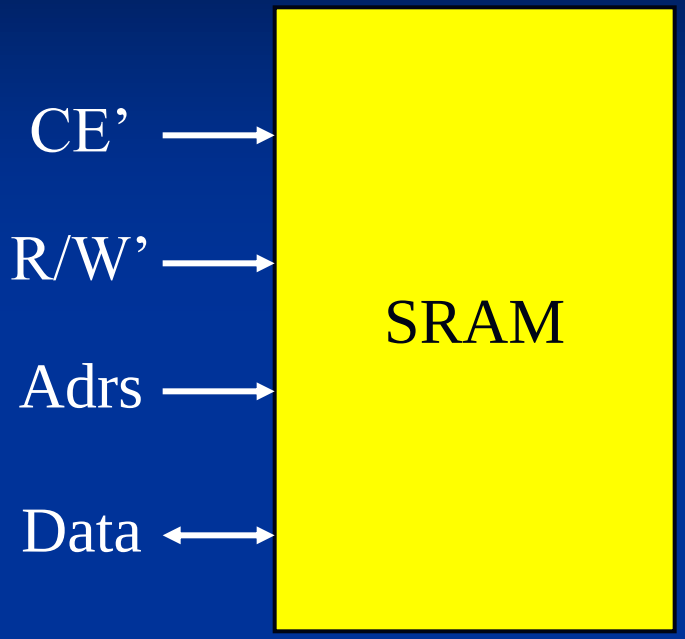
Generic Memory Device Organization



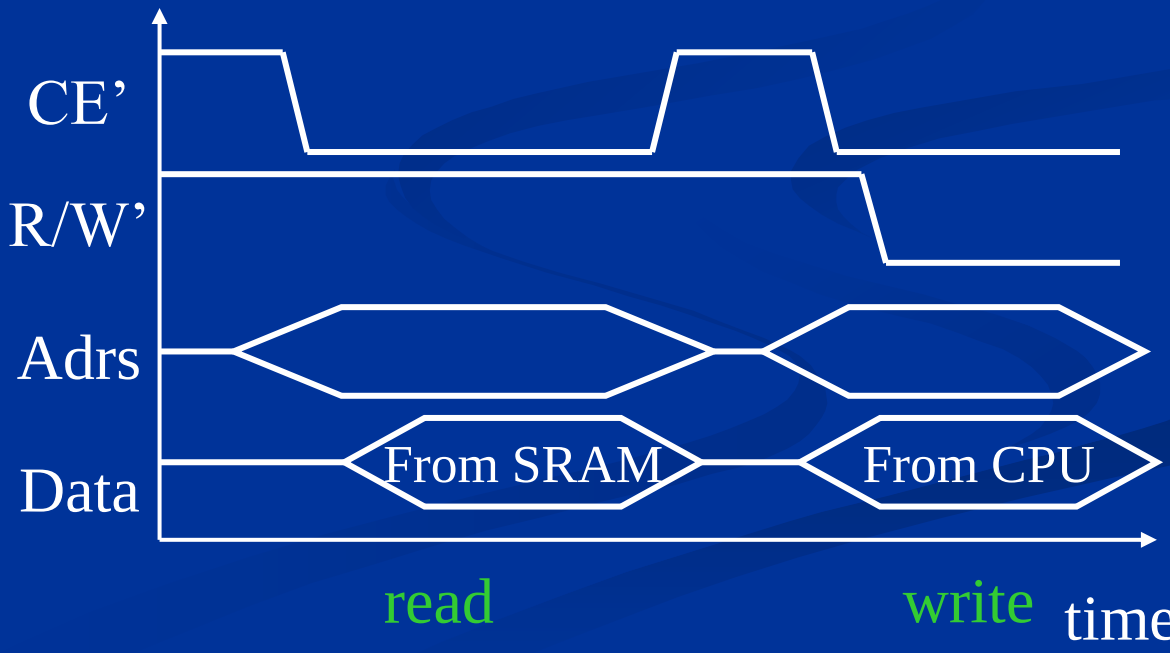
Data Width: w
Addres Width: n
Size: 2^n cells
2^n x w bits

Type of Memory:
- RAM, ROM
- SRAM, DRAM
- …

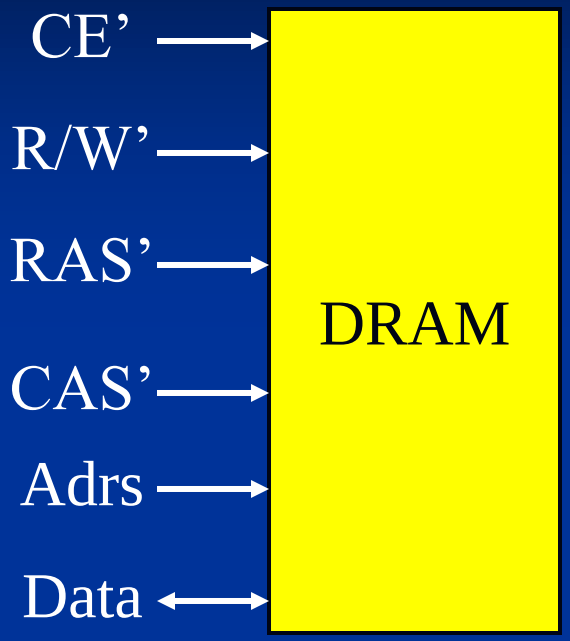
Typical Generic SRAM: Structure and Timing



- Very Fast
- High Area
- Low Capacity
- …

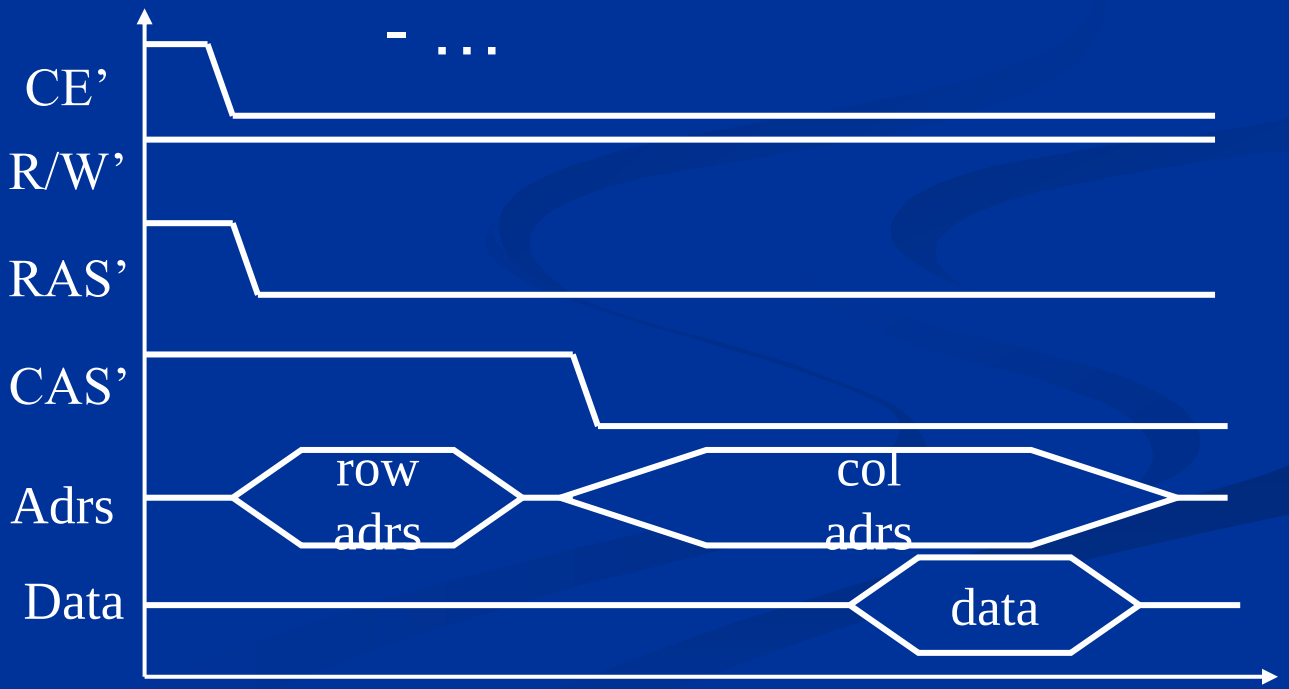


Typical Generic DRAM: Structure and Timing



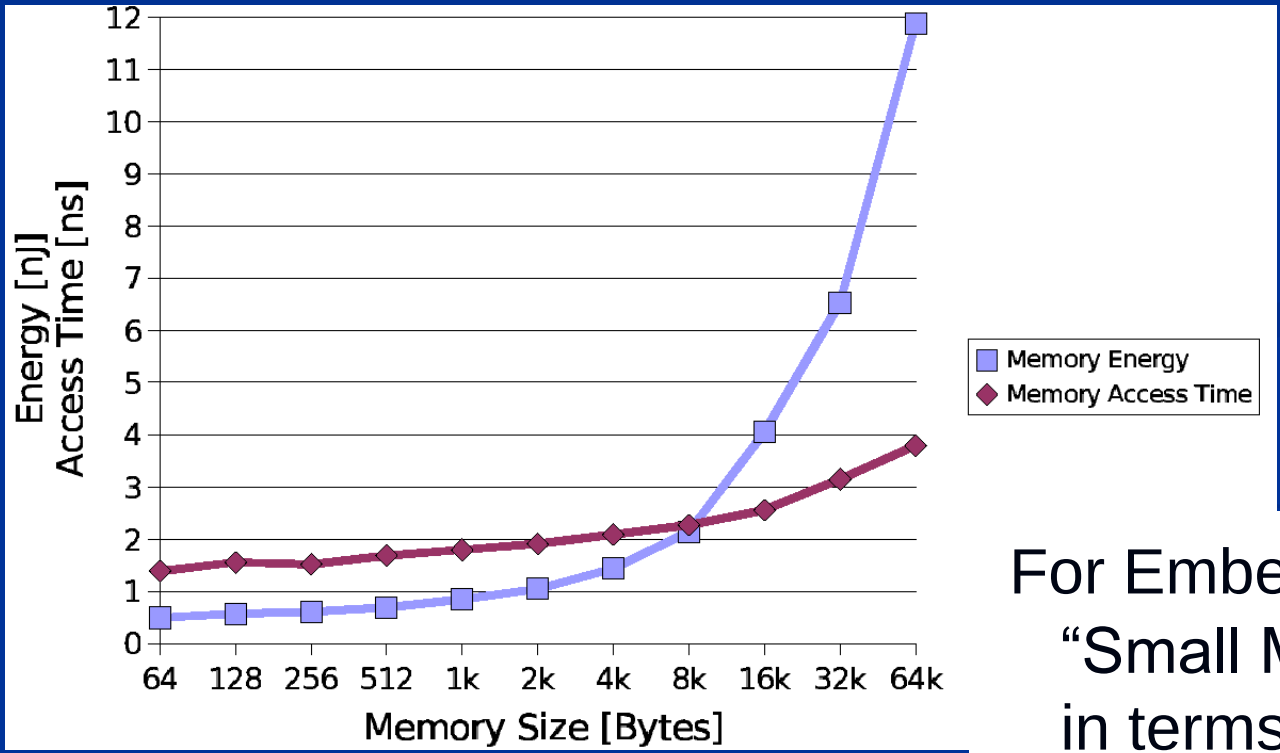
- Slower than SRAM
- Low Area
- High Capacity
- Refresh is needed!
- …

Refresh:
- done every few milliseconds
- by addressing and reading all rows



Some Concerns about Memory in ES: Access Time and Energy Efficiency

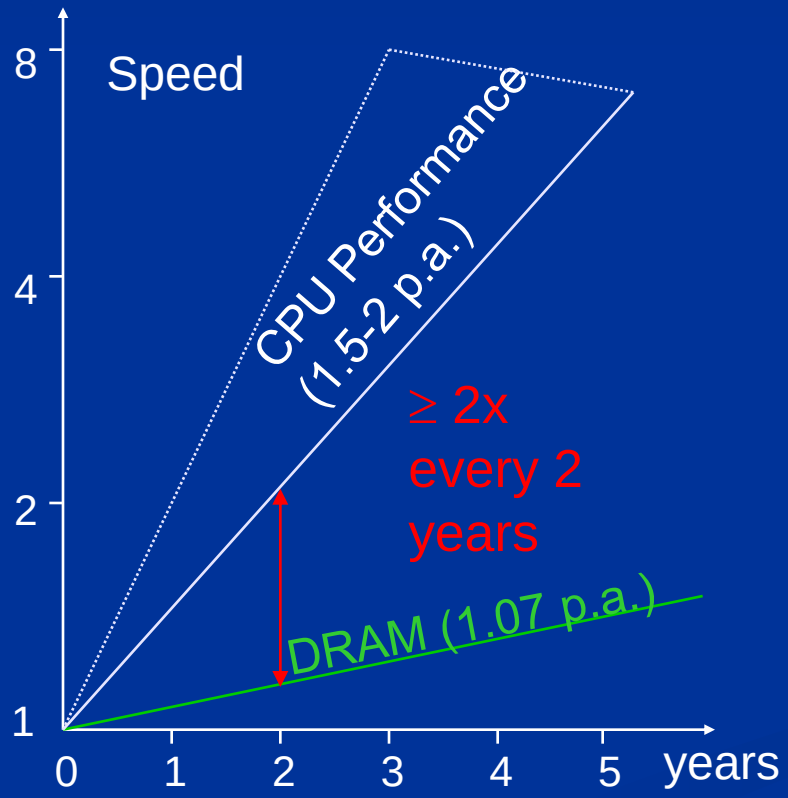
Access times and energy consumption increase with the size of the memory!



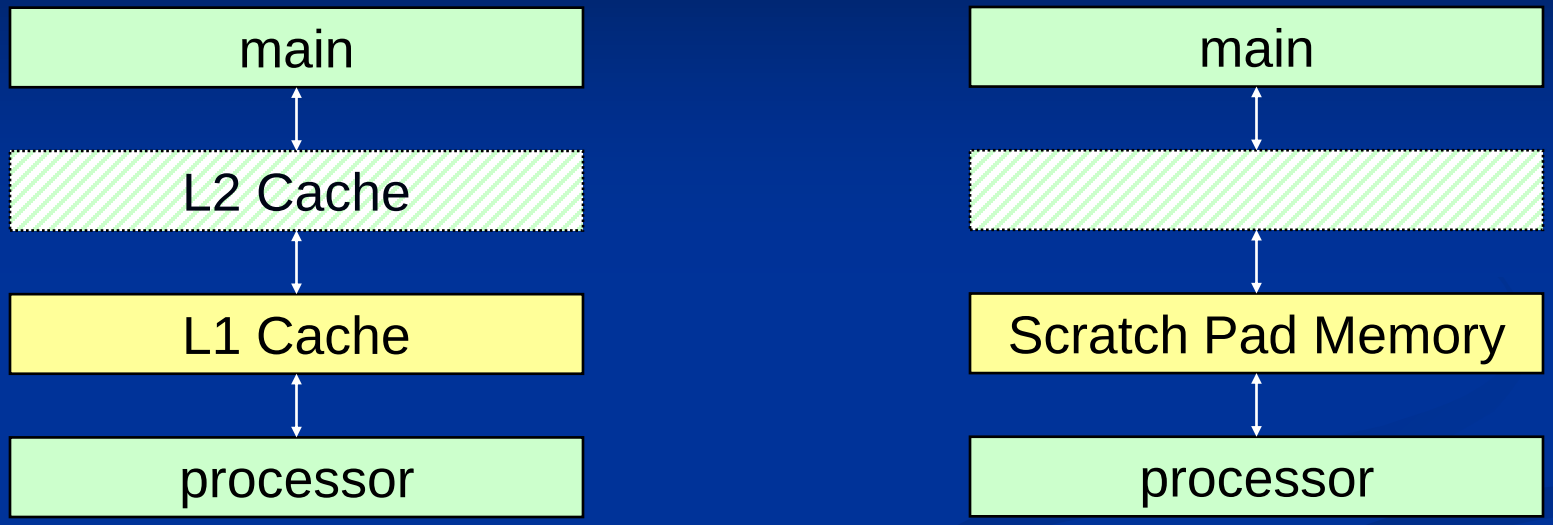
For Embedded Systems
"Small Memory is beautiful"
in terms of access time and energy consumption

Some Concerns about Memory in ES: The old "Memory Wall" Problem

Memory Access Times >> Processor Cycle Times!



Relaxing the "Memory Wall" problem: Hierarchical Memories



- For Embedded Systems, Scratch Pad Memories (SPM) are more suitable than Caches
  - Caches have unpredictable behavior (cache misses)
  - Caches consume more energy (in comparators, multiplexers, etc.)

To be continued …