Embedded Systems

Embedded Systems

End-of-term exam, Monday February 14, 2011, 14-17

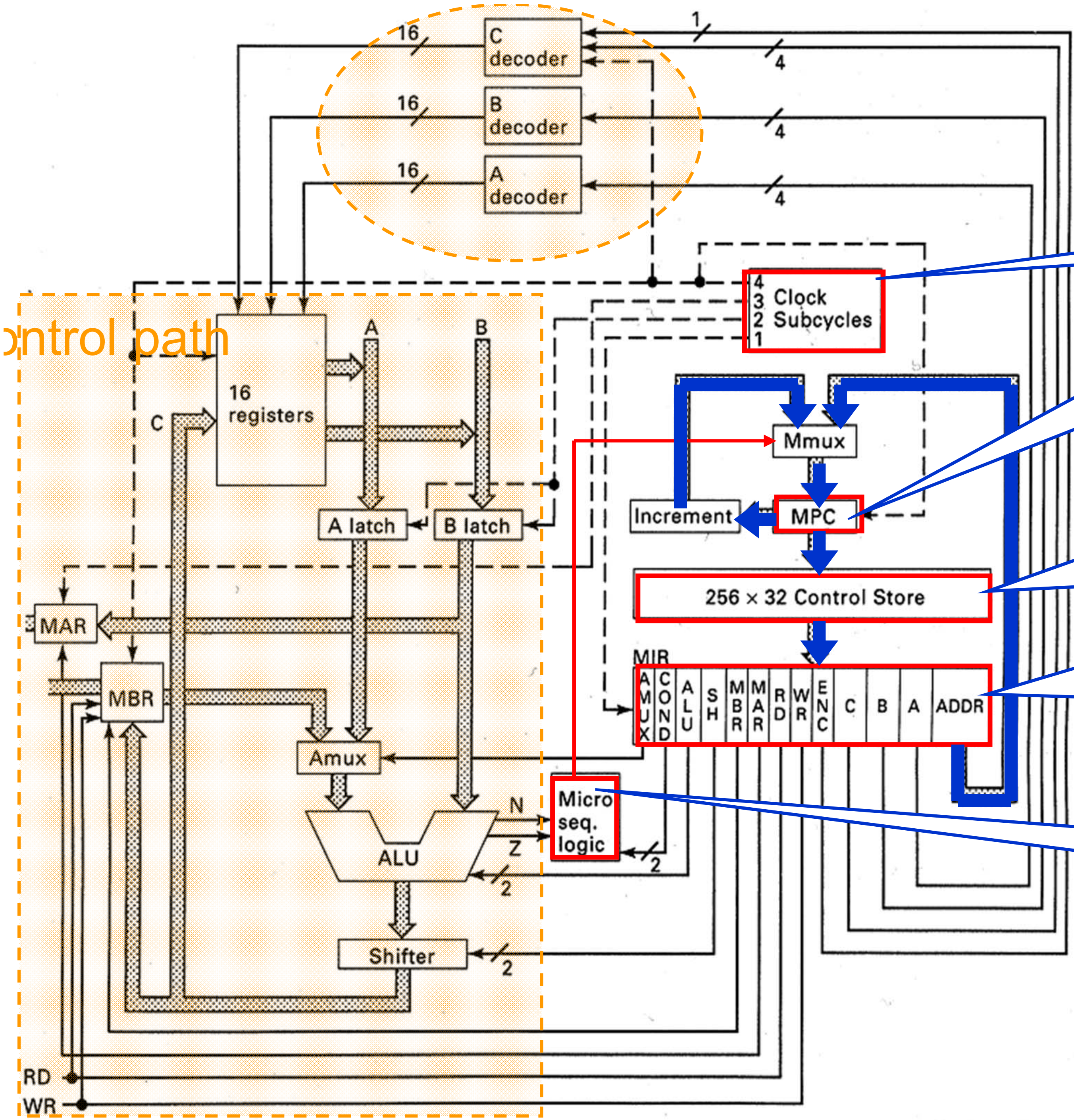
End-of-semester exam : Tuesday March 22, 2011, 14-17

Final grade:

best grade in end-of-term or end-of-semester exam.

REVIEW

Microprogram control unit



Clock generator (4-phases)

(MPC)

Microprogram memory
(256 words x 32 Bits):
Stores the micro program

Microinstruction register (MIR)

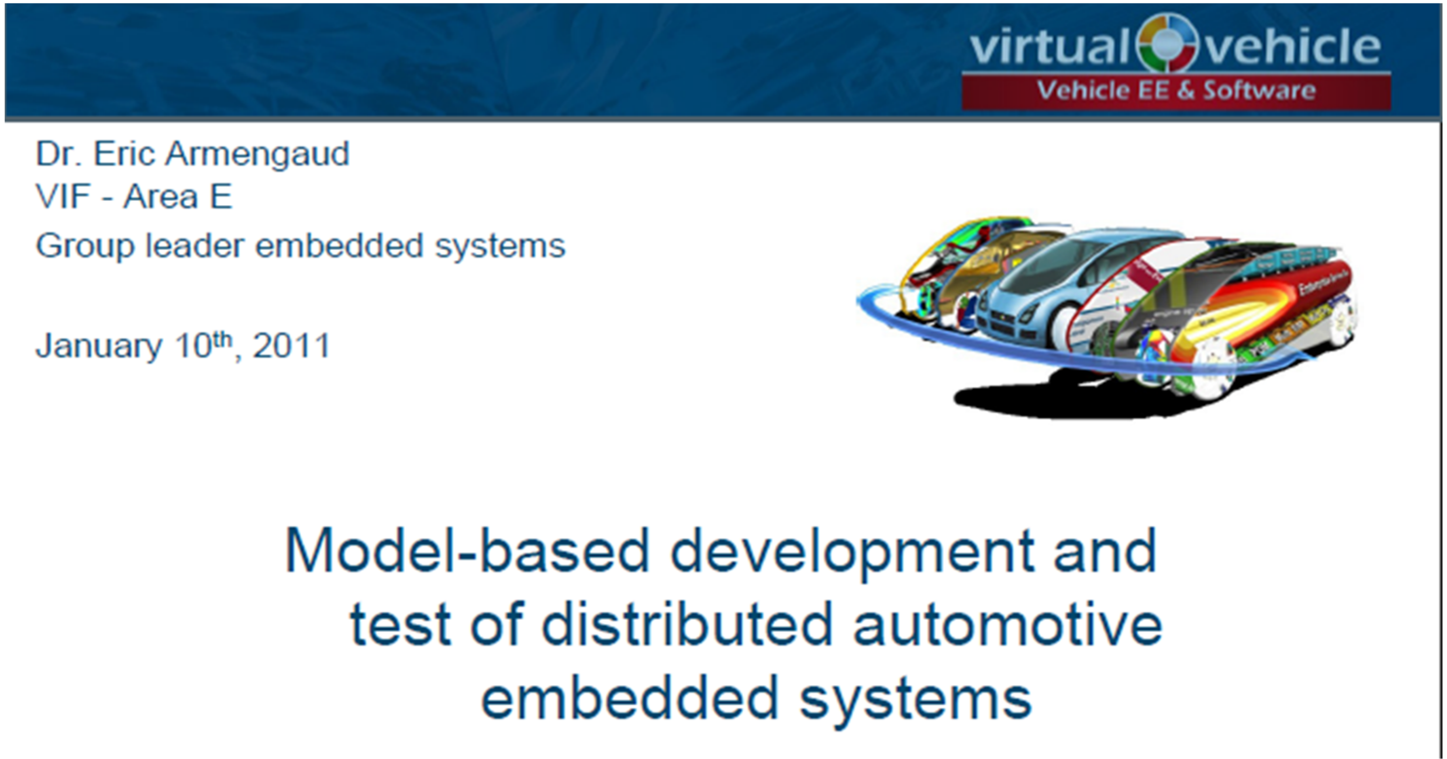
Microsequencer
(„next-Adrress-Logic)

Hazards

- It would be happy if we split the datapath into stages and the CPU works just fine
  - But, things are not that simple as you may expect
  - There are **hazards**!

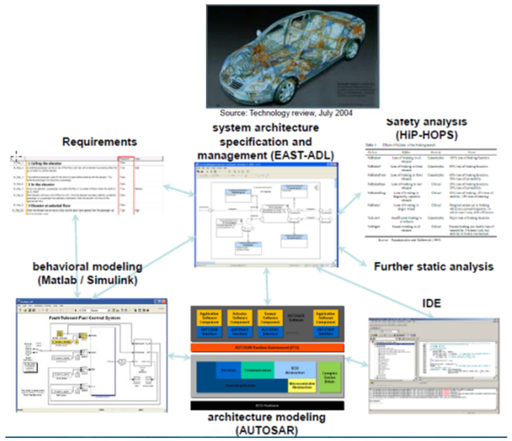
- Situations that prevent starting the next instruction in the next cycle
  - **Structure hazards**
    - Conflict over the use of a resource at the same time
  - **Data hazard**
    - Data is not ready for the subsequent dependent instruction
  - **Control hazard**
    - Fetching the next instruction depends on the previous branch outcome

REVIEW



- Requirements
- Model-based design
- Safety
- FlexRay

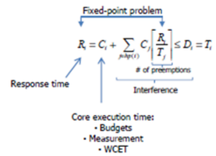
From Architecture to Application

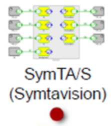


- **Timing behavior has to be predictable**
  Features that cause problems:
  - Unpredictable access to shared resources: caches, pipeline (bubbles), communication times (e.g.; multiprocessors)
  - Branch prediction, Interrupts, Instructions that with data-dependent execution times

  ☞ Trying to avoid as many of these as possible.

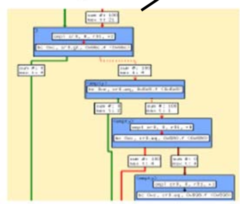
System level
- Multiple functions or tasks
- Focus on
  - Integration and scheduling
  - End-to-end timing
  - Worst-Case Response Time (WCRT)





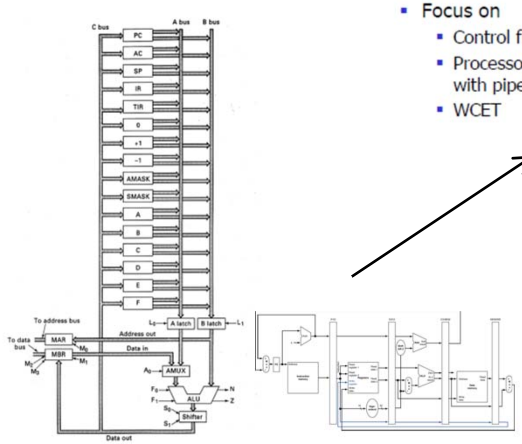
SymTA/S
(Symtavision)

- Code level
  - Single process, task, ISR
  - Focus on
    - Control flow
    - Processor architecture with pipelines and caches
    - WCET



aiT
(AbsInt)

Timing analysis → lecture next week



Key requirements for processors

- Code size efficiency

  - **Compression techniques (instruction, e.g.;** ARM Thumb instruction set**)**
  - **Cache**-based decompression

  CISC machines: RISC machines designed for run-time-, not for code-size-efficiency

- Energy efficiency of processors (motivation lecture 1)
  – Mobiles devices
  – general purpose processors (temperature hot-spots)

  →Power Aware Computing (lectures February)

- Run-time efficiency
  – Domain-oriented architectures (e.g.; DSPs)

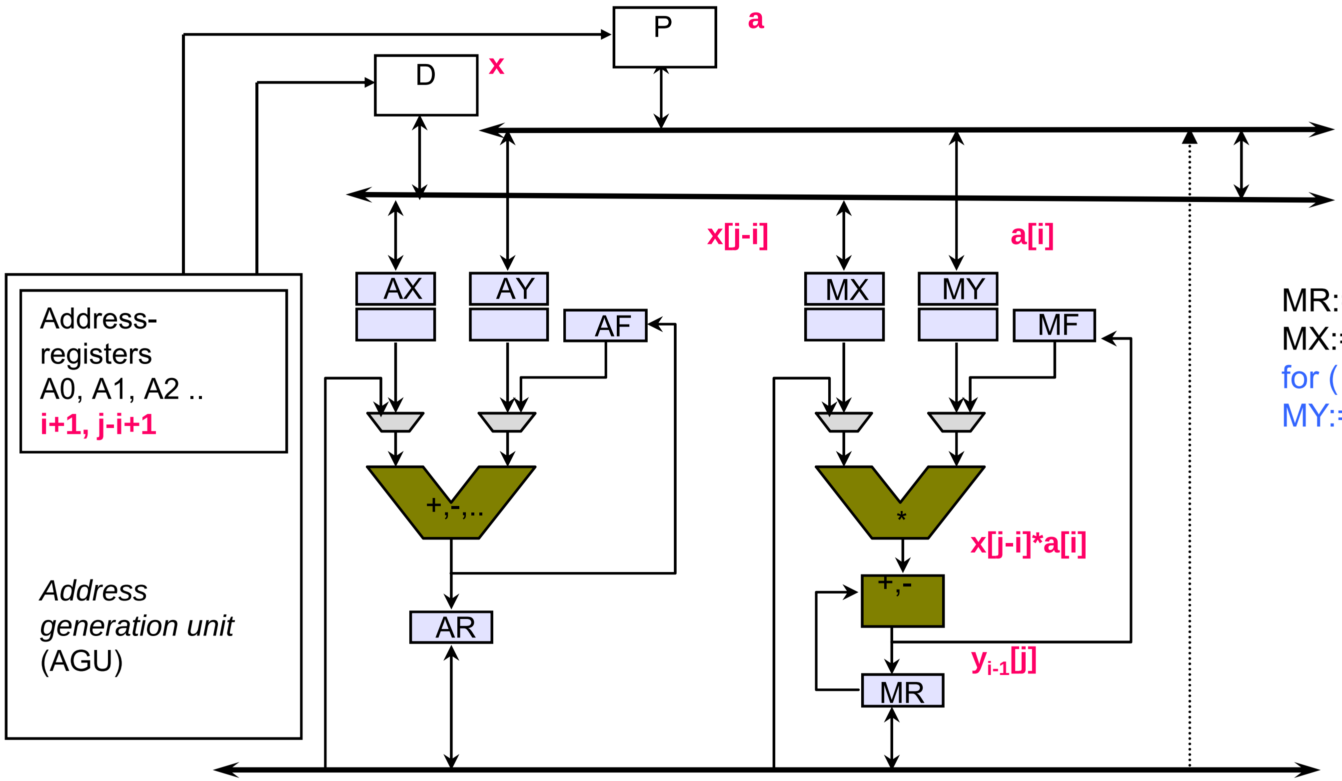
Key requirement : Run-time efficiency
Domain-oriented architectures

Application: $y[j] = \sum_{i=0}^{n-1} x[j-i]*a[i]$

$\forall i: 0 \le i \le n-1: y_i[j] = y_{i-1}[j] + x[j-i]*a[i]$

Architecture: **Example: Data path ADSP210x**

Application maps nicely onto architecture



MR:=0;
MX:=x[n-1]; MY:=a[0]; A1:=1; A2:=n-2;
for (j:=1 to n) {MR:=MR+MX*MY;
MY:=a[A1]; MX:=x[A2]; A1++; A2--}

DSP-Processors: multiply/accumulate (MAC) and zero-overhead loop (ZOL) instructions

```
MR:=0; A1:=1; A2:=n-2; MX:=x[n-1]; MY:=a[0];
for ( j:=1 to n)
  {MR:=MR+MX*MY; MY:=a[A1]; MX:=x[A2]; A1++; A2--}
```

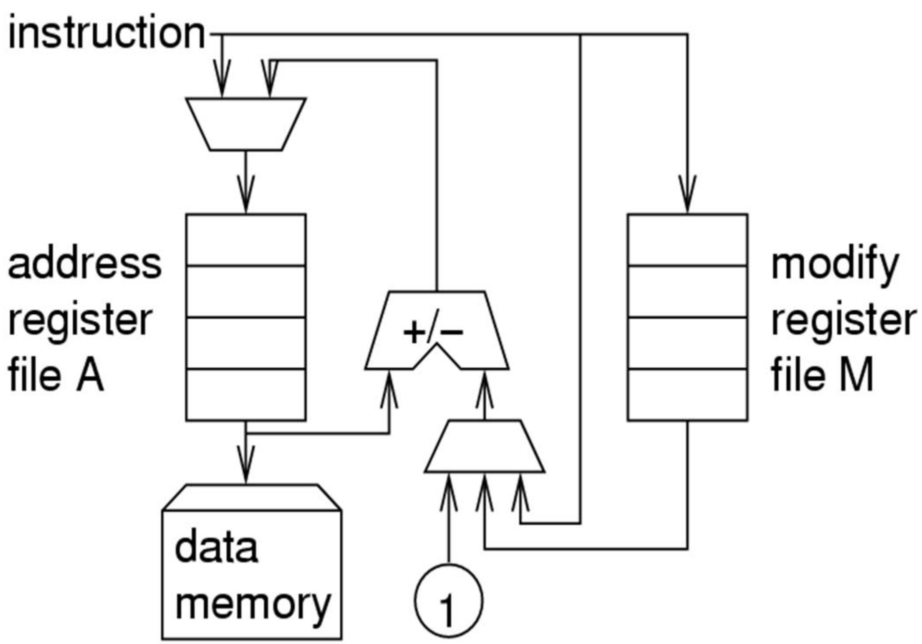
Multiply/accumulate (MAC) instruction

Zero-overhead loop (ZOL) instruction preceding MAC instruction.
Loop testing done in parallel to MAC operations.

Loop counter incr., test against end condition, and branching are done by hardware

Separate address generation units (AGUs)

Example (ADSP 210x):



- Data memory can only be fetched with address contained in A,
- but this can be done in parallel with operation in main data path (takes effectively 0 time).
- A := A   1 also takes 0 time,
- same for A := A ± M;
- A := <immediate in instruction> requires extra instruction

Saturating arithmetic

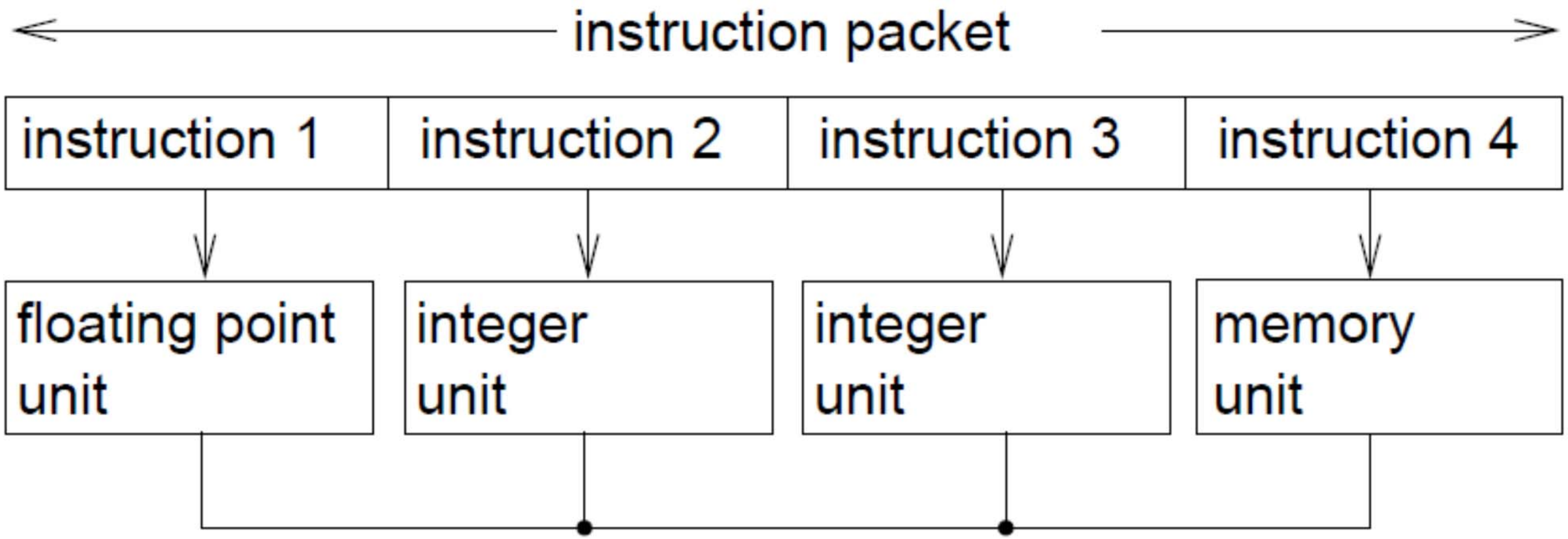
- Returns largest/smallest number in case of over/underflows
- Example:

a			0111
b		+	1001
standard wrap around arithmetic			(1)0000
saturating arithmetic			1111
(a+b)/2:	correct		1000
	wrap around arithmetic		0000
	saturating arithmetic + shifted		0111 „almost correct“

- Appropriate for DSP/multimedia applications:
  - No timeliness of results if interrupts are generated for overflows
  - Precise values less important
  - Wrap around arithmetic would be worse.

Key idea of very long instruction word (VLIW) computers

- Instructions included in long instruction packets.
  Instruction packets are assumed to be executed in parallel.
- Fixed association of packet bits with functional units.



Very long instruction word (VLIW) architectures

- Very long instruction word ("instruction packet") contains several instructions, all of which are assumed to be executed in parallel.
- Compiler is assumed to generate these "parallel" packets
- Complexity of finding parallelism is moved from the hardware (RISC/CISC processors) to the compiler; Ideally, this avoids the overhead (silicon, energy, ..) of identifying parallelism at run-time.

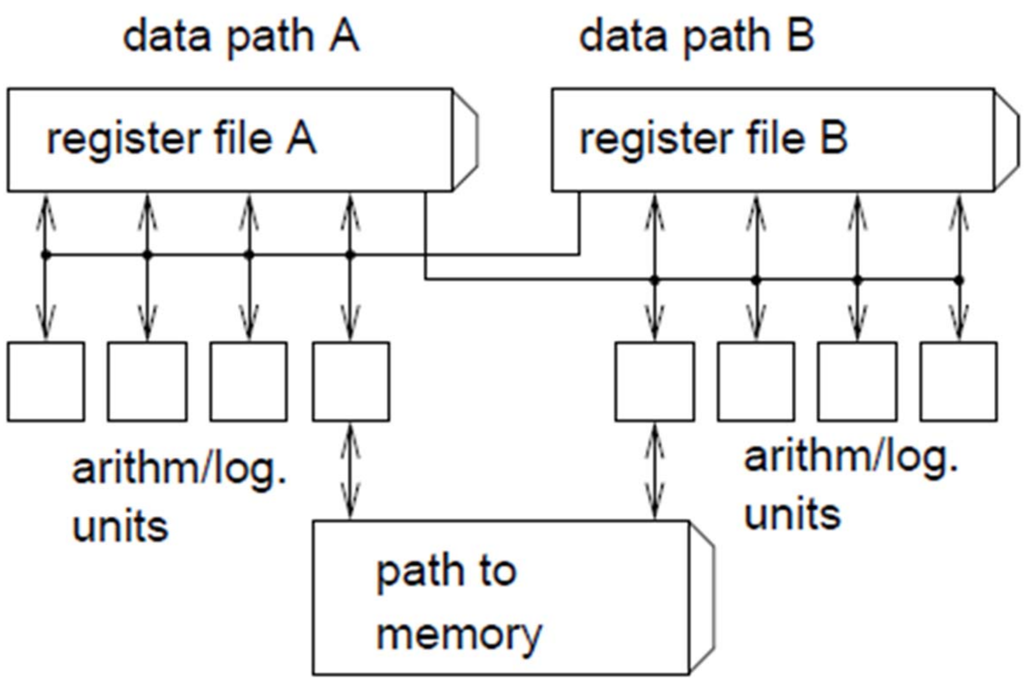
☞ A lot of expectations into VLIW machines

- Explicitly parallel instruction set computers (EPICs) are an extension of VLIW architectures: parallelism detected by compiler, but no need to encode parallelism in 1 word.

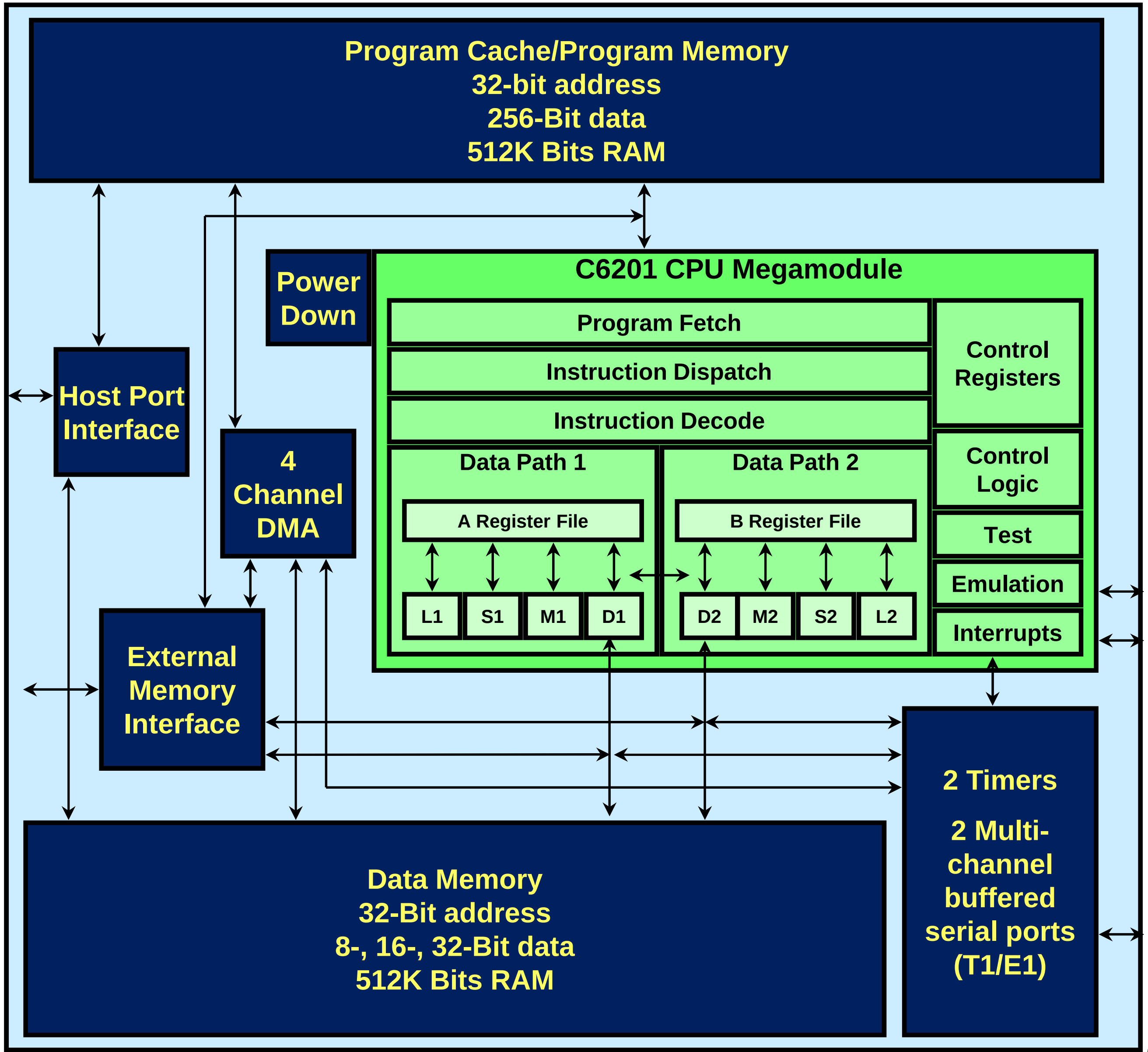
Partitioned register files

- Many memory ports are required to supply enough operands per cycle.
- Memories with many ports are expensive.

☞ Registers are partitioned into (typically 2) sets, e.g. for TI C60x:



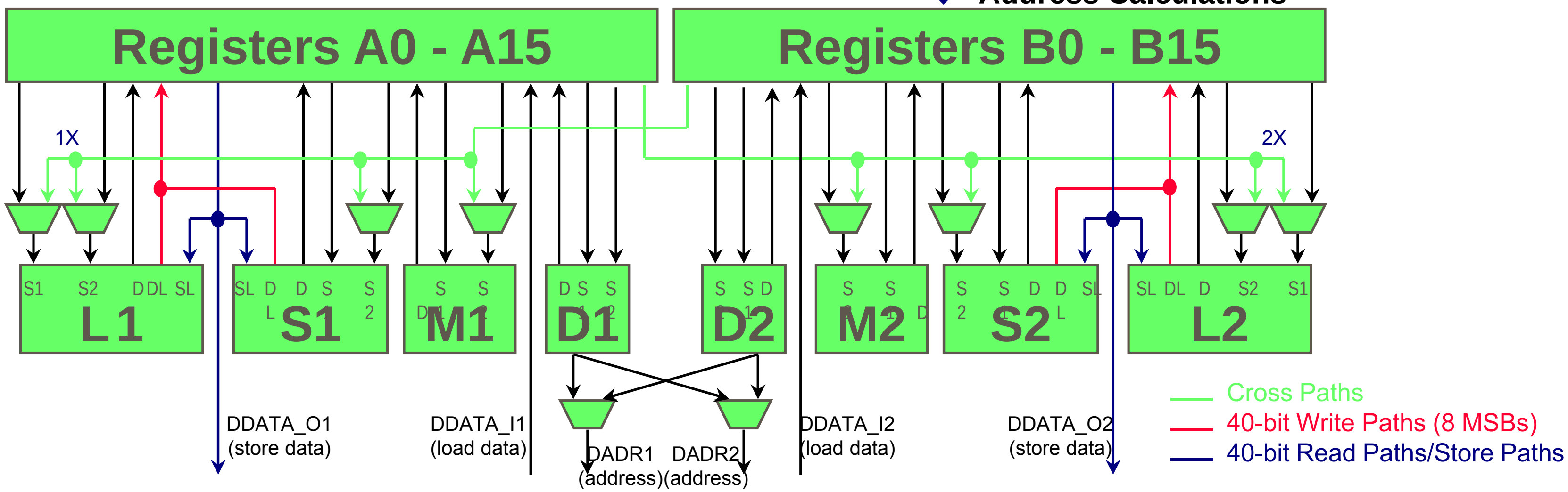
TMS320C6x



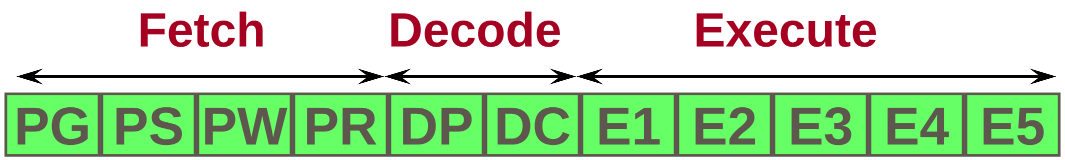
Program Cache/Program Memory
32-bit address
256-Bit data
512K Bits RAM
Power Down
C6201 CPU Megamodule
Program Fetch
Instruction Dispatch
Instruction Decode
Control Registers
Control Logic
Test
Emulation
Interrupts
Data Path 1
Data Path 2
A Register File
B Register File
L1
S1
M1
D1
D2
M2
S2
L2
Host Port Interface
4 Channel DMA
External Memory Interface
2 Timers
2 Multi-channel buffered serial ports (T1/E1)
Data Memory
32-Bit address
8-, 16-, 32-Bit data
512K Bits RAM

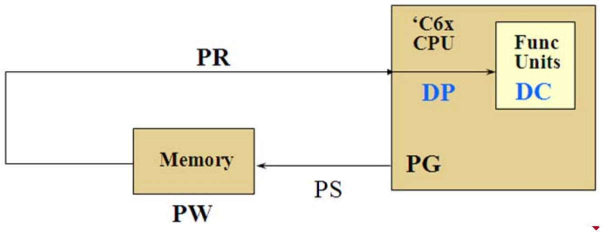
TMS320C6x Datapath

- 2 Data Paths
- 8 Functional Units
  - Orthogonal/Independent
  - 6 Arithmetic Units
  - 2 Multipliers
- Control
  - Independent
  - Up to 8 32-bit Instructions in parallel
- Registers
  - 2 Files
  - 32, 32-bit Registers Total
- Cross paths (1X, 2X)
- L-Unit (L1, L2)
  - 40-bit Integer ALU
  - Comparisons
  - Bit Counting
  - Normalization
- S-Unit (S1, S2)
  - 32-bit ALU
  - 40-bit Shifter
  - Bitfield Operations
  - Branching
- M-Unit (M1, M2)
  - 16 x 16 -> 32
- D-Unit (D1, D2)
  - 32-bit Add/Subtract
  - Address Calculations

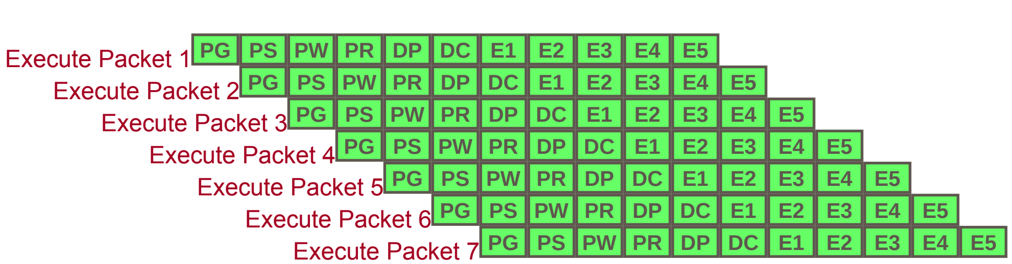


TMS320C6x Pipeline



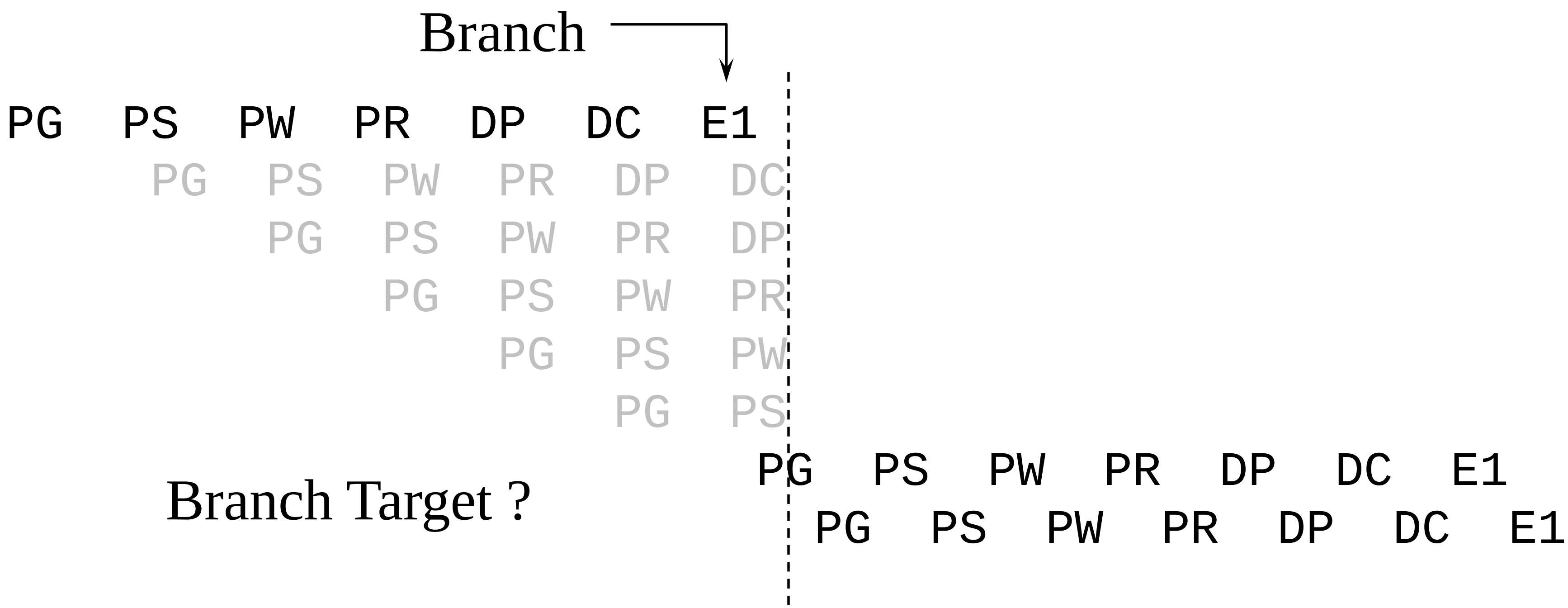


- Single-Cycle Throughput
- Operate in Lock Step
- Fetch
  - PG — Program Address Generate
  - PS — Program Address Send
  - PW — Program Access Ready Wait
  - PR — Program Fetch Packet Receive
- Decode
  - DP — Instruction Dispatch
  - DC — Instruction Decode
- Execute
  - E1 - E5 — Execute 1 through Execute 5



Branch in the Pipeline...

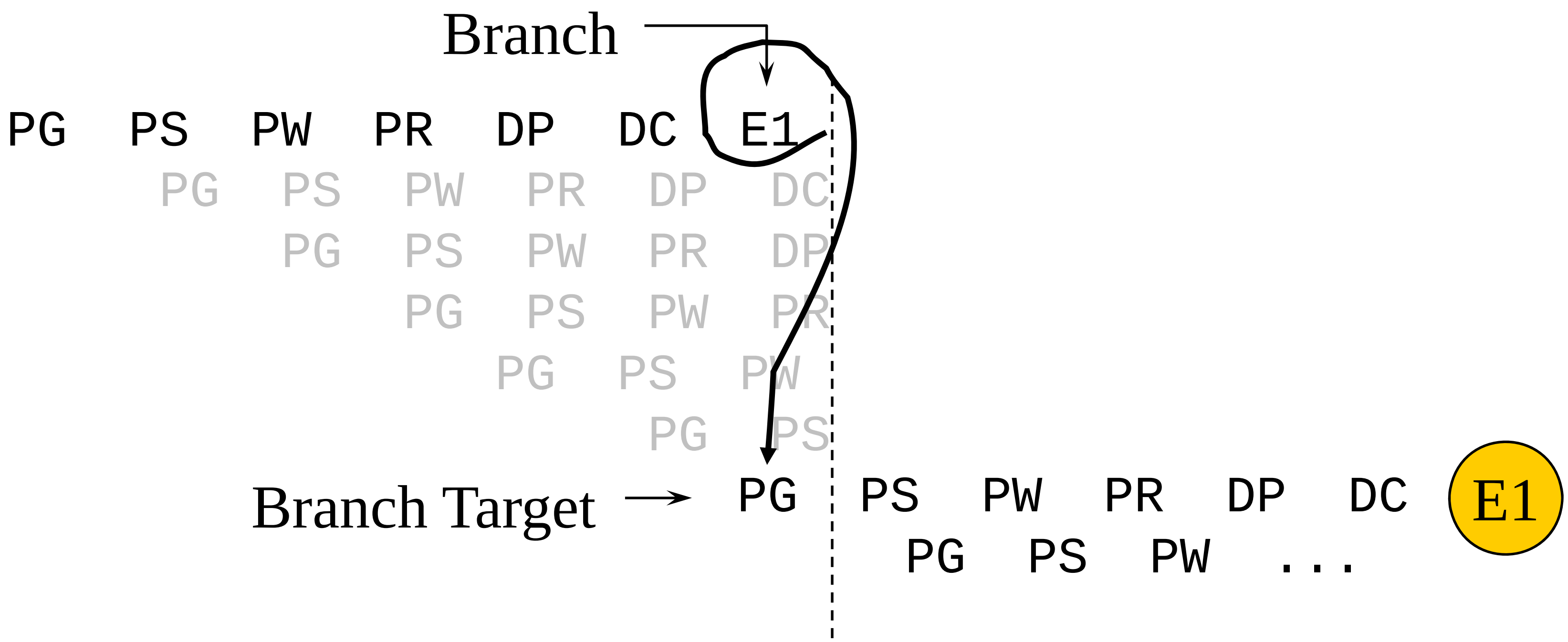
Branch is a 1-cycle instruction?



The execution of 5 instructions has been started before it is realized that a branch was required.

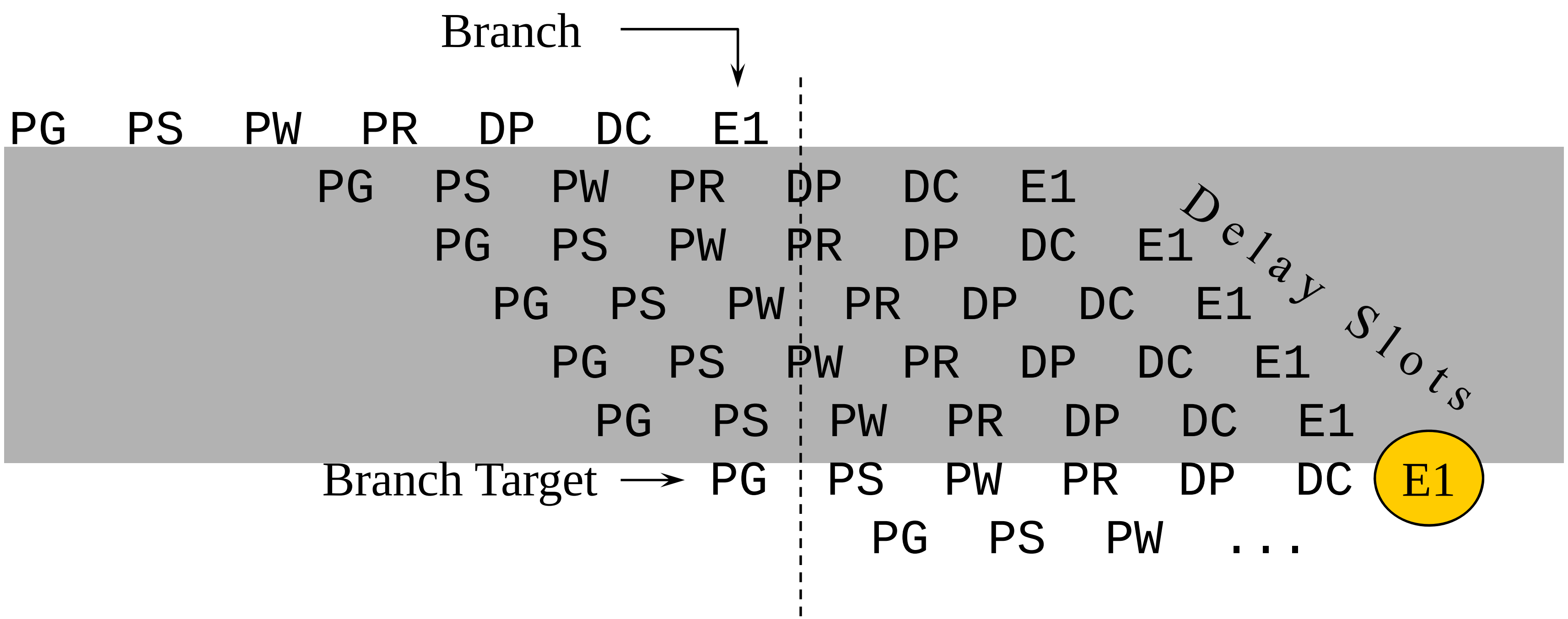
Branch in the Pipeline...

Branch is a 1-cycle instruction?



Branch in the Pipeline...

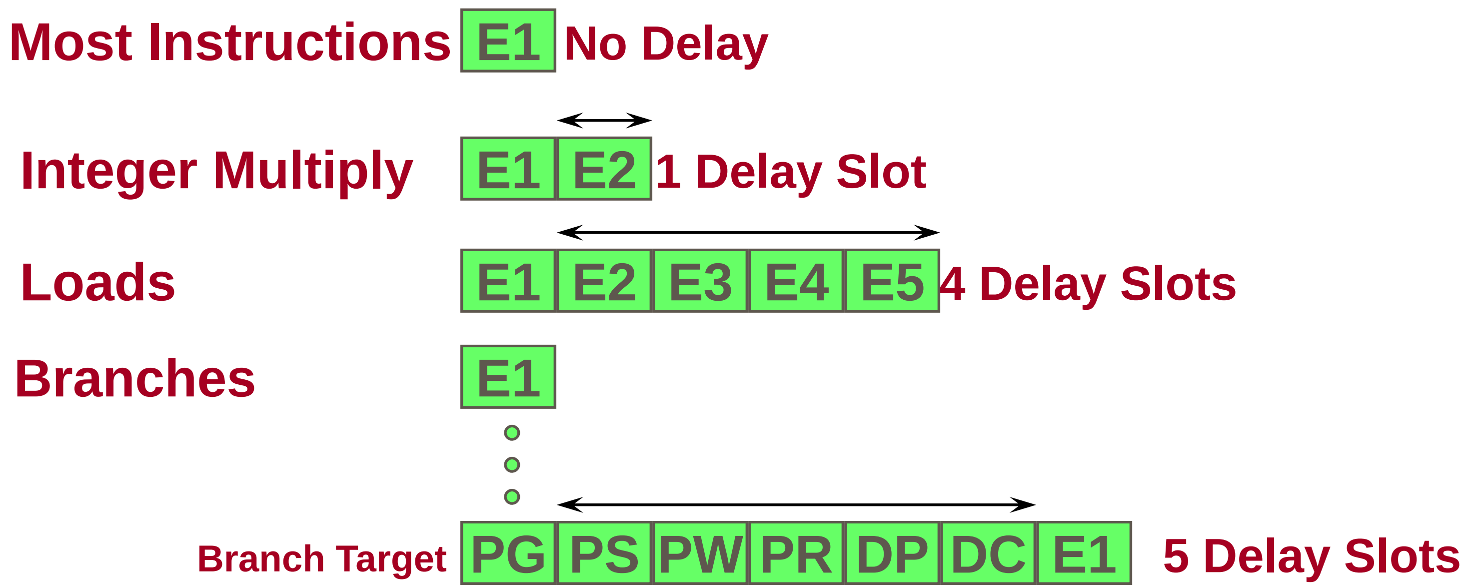
Branch is a 1-cycle instruction?



TMS320C6x Pipeline (2)

Delay Slots: number of extra cycles until result is:

- written to register file
- available for use by a subsequent instructions
- Multi-cycle NOP instruction can fill delay slots while minimizing codesize impact



TMS320C6x instruction set

Signifies a parallel operation

A-side M-unit using an operand from B-side*

B-side M-unit*

```
loop:
        ADD.L1      A0,A3,A0   ; A0=A0+A3
||      ADD.L2      B1,B7,B1   ; B1=B1+B7
||      MPYHL.M1X   A2,B2,A3   ; A3=A2(hi)*B2(lo)
||      MPYLH.M2X   A2,B2,B7   ; B7=A2(lo)*B2(hi)
||      LDW.D2      *B4++,B2   ; load into B2
||      LDW.D1      *A7--,A2   ; load into A2
||      ADD.S2      -1,B0,B0   ; decrement counter
||      [B0]        B.S1 loop  ; branch if B0 nonzero
```

B-Side L-unit using an operand from A-side*

- 8 instructions in parallel (one cylce)
- scheduling at compile time

Embedded System Hardware
- Reconfigurable Hardware -

Reconfigurable Logic

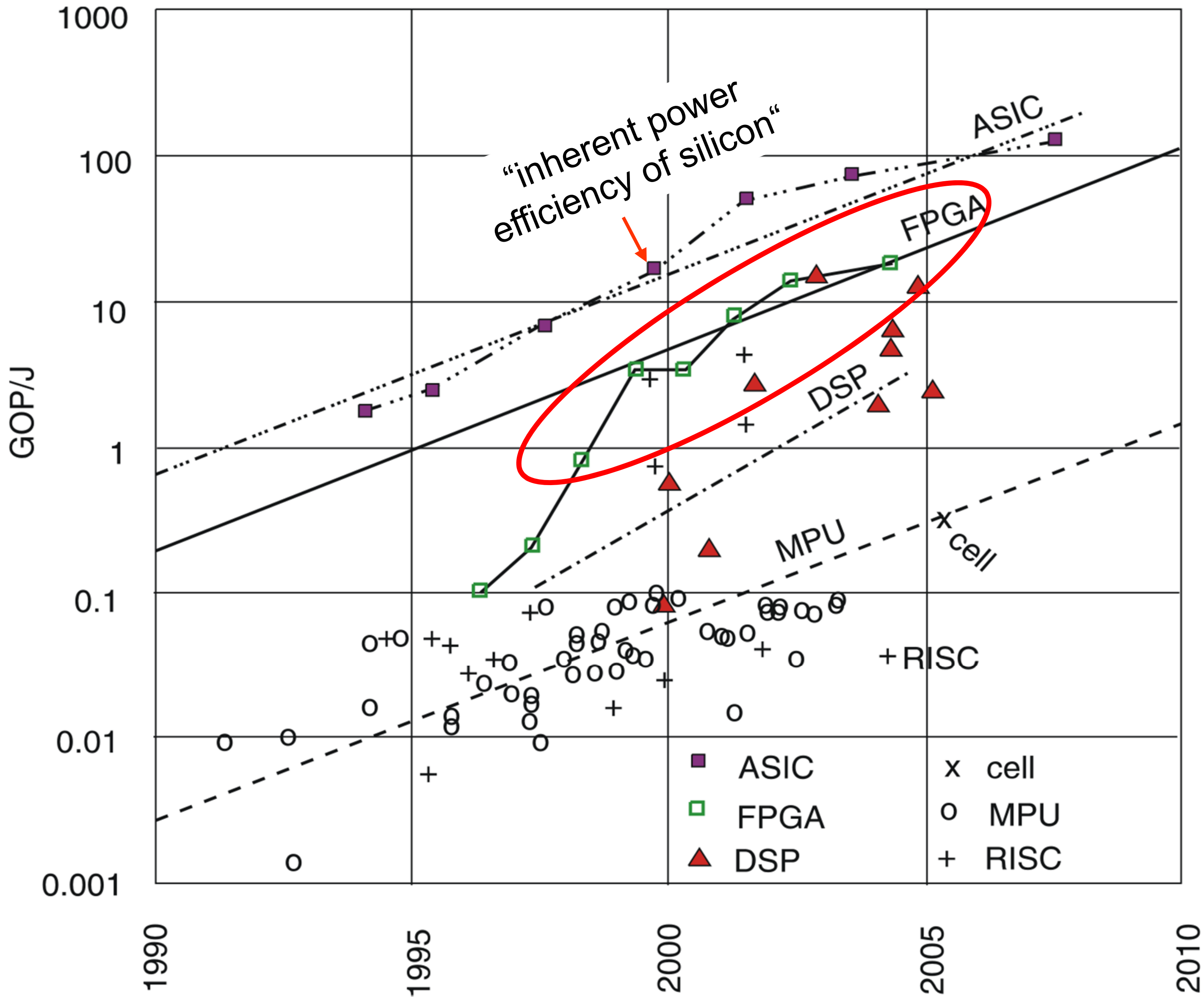
- Full custom chips may be too expensive - high NRE costs (Non-Recurring Engineering), software too slow.
- Combine the speed of HW with the flexibility of SW
  - ☞ HW with programmable functions and interconnect.
  - ☞ Use of configurable hardware;
    common form: field programmable gate arrays (FPGAs)

Applications: bit-oriented algorithms like

- encryption,
- fast "object recognition" (medical and military)
- Adapting mobile phones to different standards – Software defined radios (SDR)

- devices from XILINX Actel, Altera, …

Energy Efficiency of FPGAs



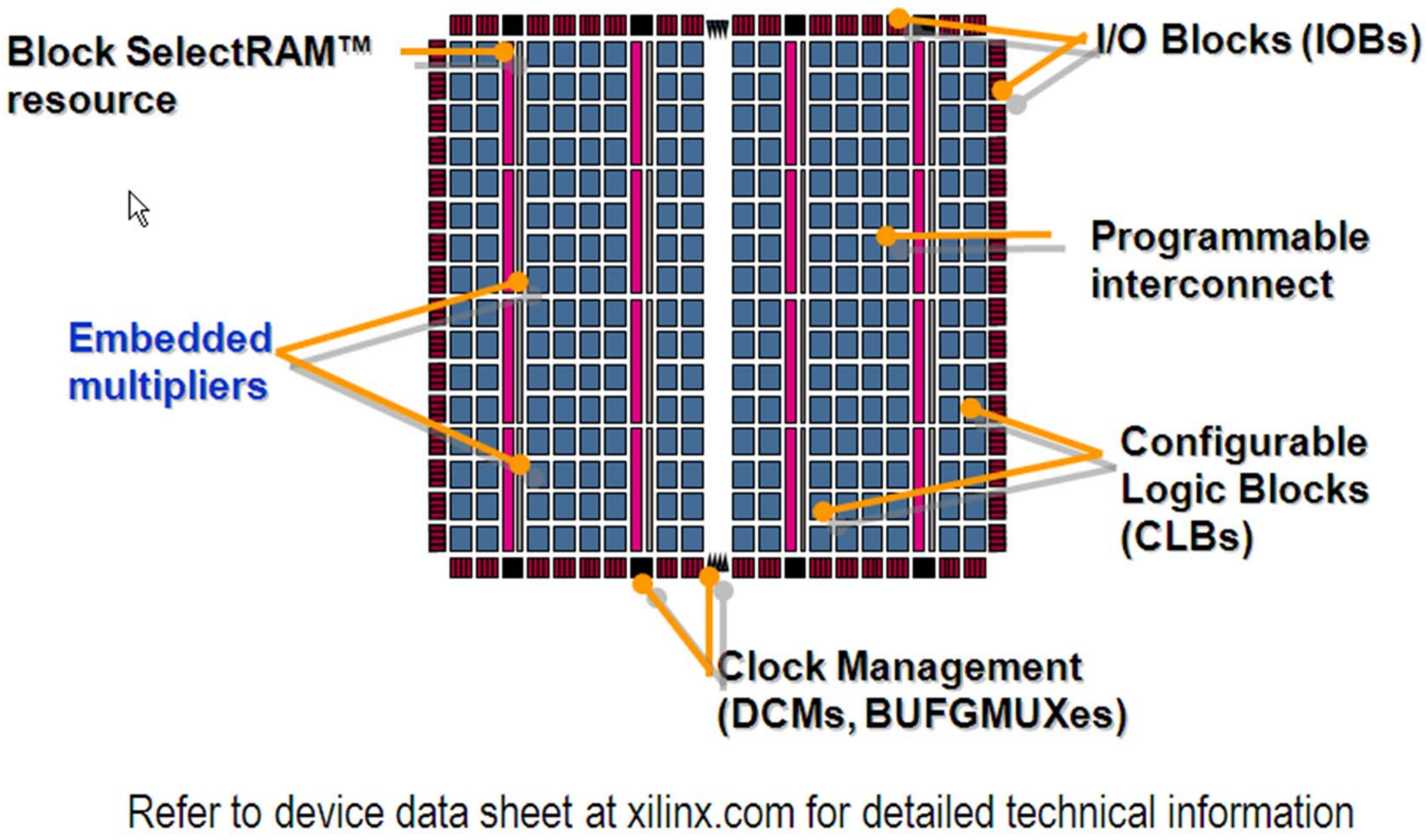
© Hugo De Man, IMEC, Philips, 2007

Overview XILINX FPGA

- All Xilinx FPGAs contain the same basic resources
  - Slices grouped into Configurable Logic Blocks (CLBs)
    - Contain combinatorial logic and register resources
  - IOBs
    - Interface between the FPGA and the outside world
  - Programmable interconnect
  - Other resources
    - Memory
    - Multipliers
    - Global clock buffers
    - Boundary scan logic

XILINX FPGA Virtex-II Architecture

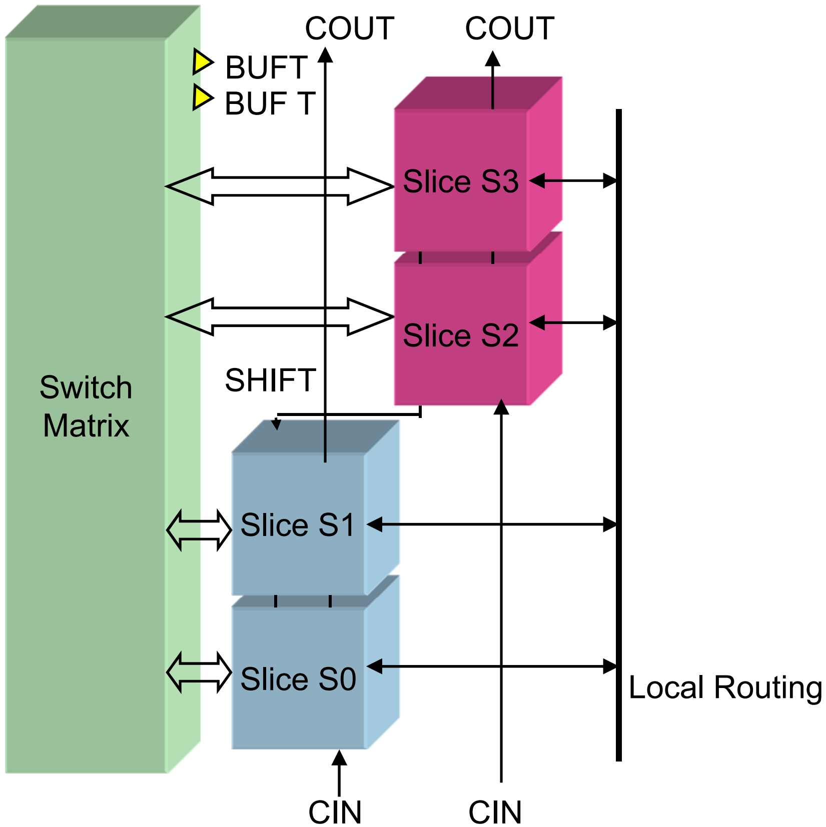
First family with Embedded Multipliers to enable high-performance DSP



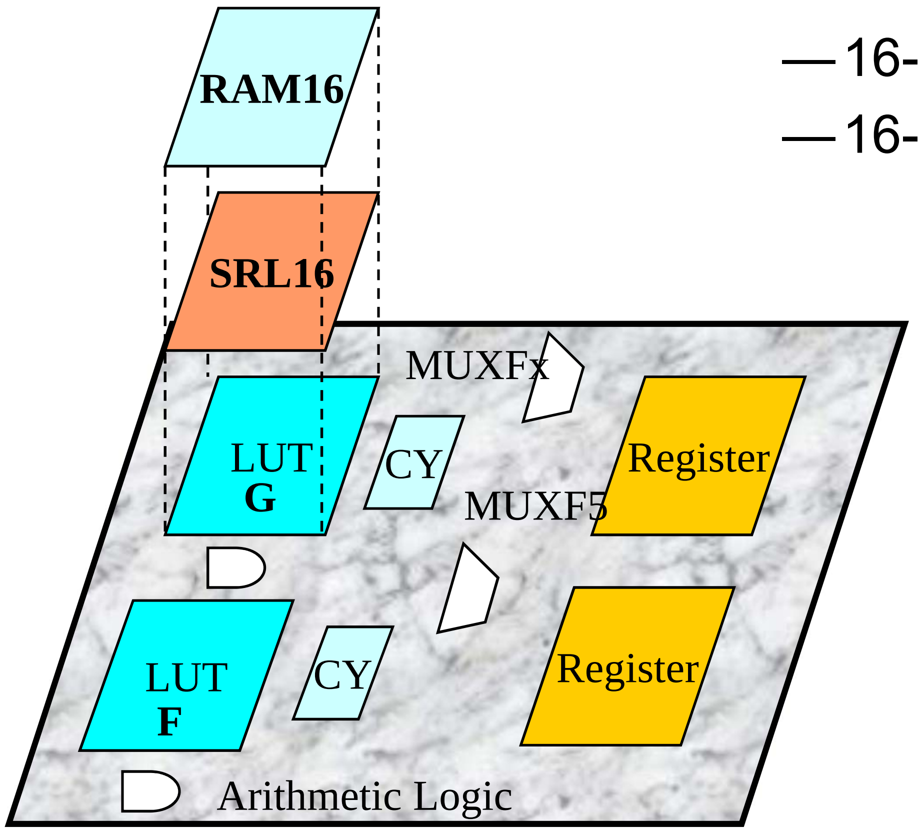
CLBs and Slices

Combinatorial and sequential logic implemented here

- Each Virtex™-II CLB contains four slices
  - Local routing provides feedback between slices in the same CLB, and it provides routing to neighboring CLBs
  - A switch matrix provides access to general routing resources



Slice Resources

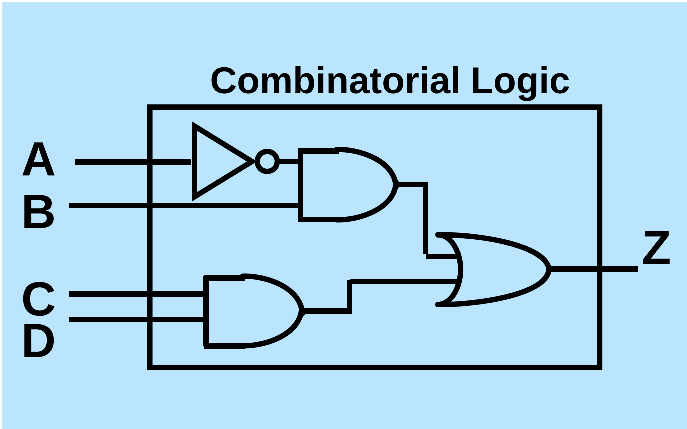


- **Each slice contains two:**
  - — Four inputs lookup tables
  - — 16-bit distributed SelectRAM
  - — 16-bit shift register

- **Each register:**
  - — D flip-flop
  - — Latch
- **Dedicated logic:**
  - — Muxes
  - — Arithmetic logic
    - — MULT_AND
    - — Carry Chain

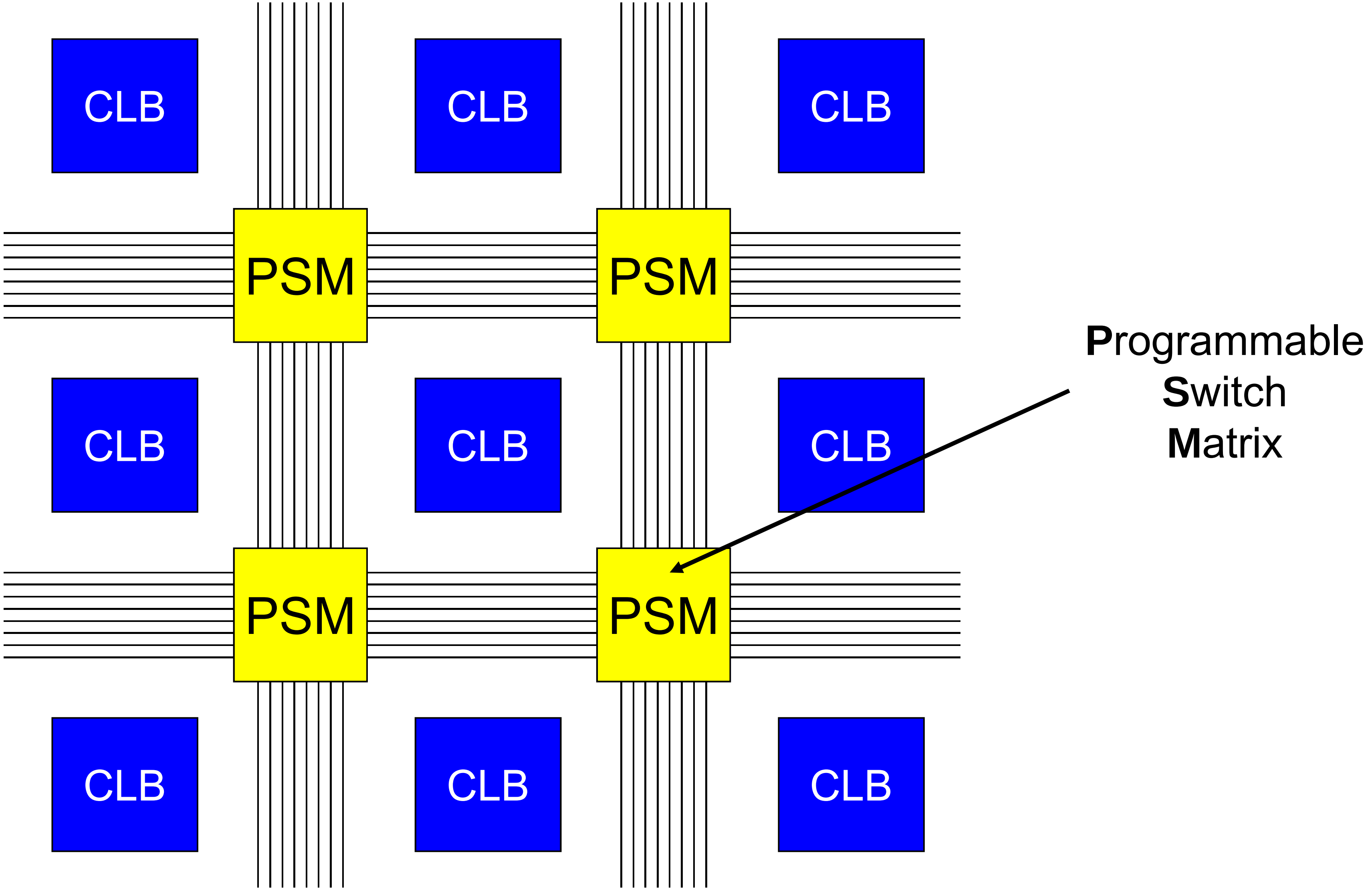
Look-Up Tables

- Combinatorial logic is stored in Look-Up Tables (LUTs)
  – Also called Function Generators (FGs)
  – Capacity is limited by the number of inputs, not by the complexity
- Delay through the LUT is constant



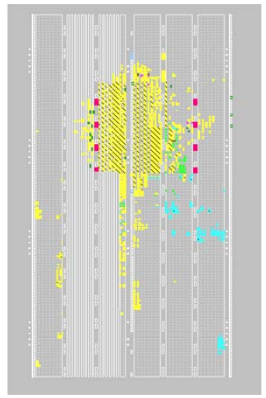
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Routing Resources



Embedded Processors in FPGAs

- Hard Core
  - EP is a dedicated physical component of the chip separate from the programmable logic
  - E.g. Xilinx Virtex families (PowerPC 405)
- Soft Core
  - Embedded processor is also a synthesized to the FPGA to th programmable logic on the chip
  - E.g. Altera (NIOS), Xilinx (MicroBlaze)



Embedded Design Flow

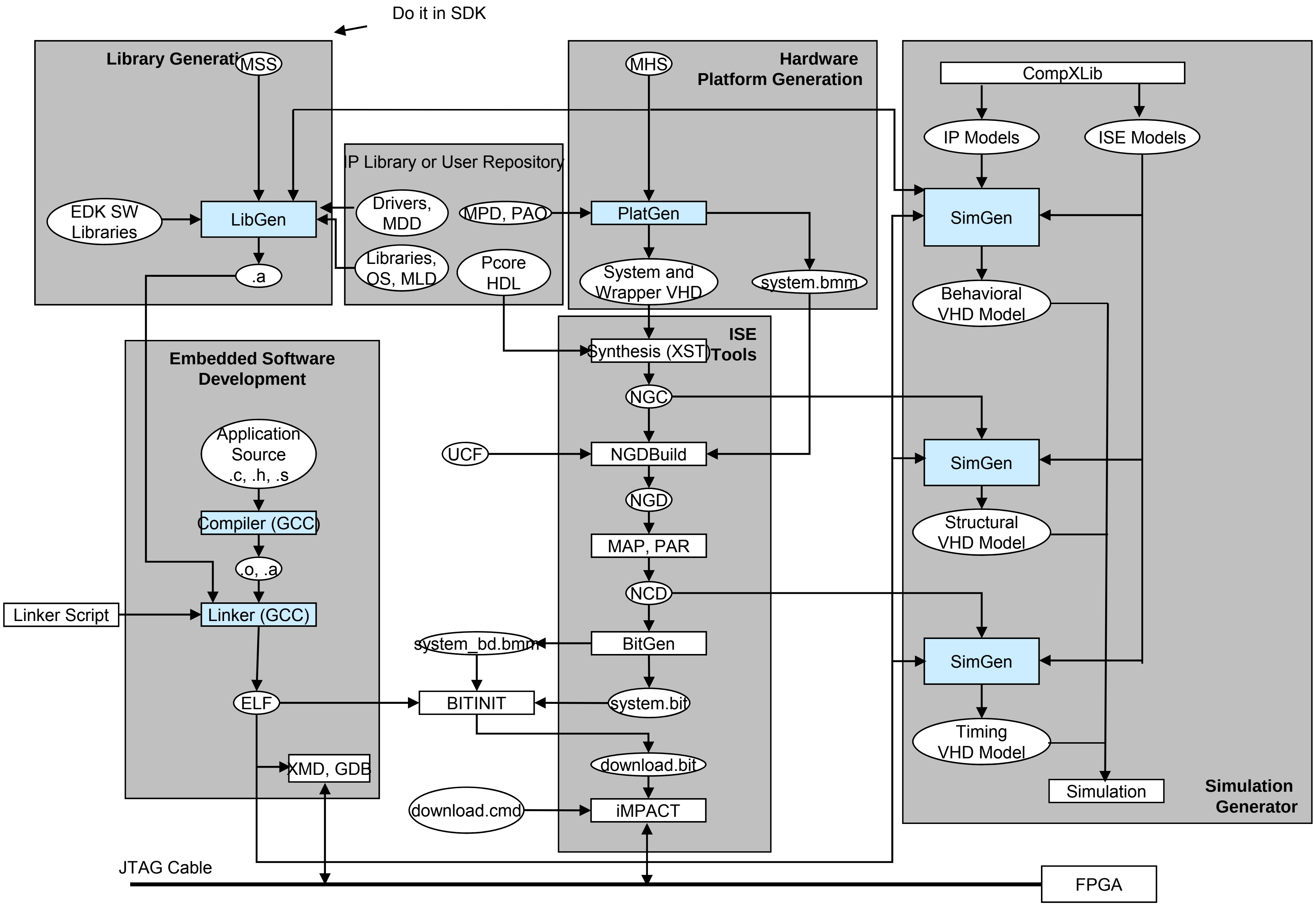
A. Develop the **embedded hardware**
   - Quickly create a system targeting a board using **Base System Builder Wizard**
   - Extend the hardware system, if necessary, by adding peripherals from the **IP Catalog**
   - Generate HDL netlists using **PlatGen**

B. Develop the **embedded software**
   - Generate libraries and drivers with **LibGen**
   - Create and debug the software application using **Software Development Kit (SDK)**
   - Optionally, debug the application using **Xilinx Microprocessor Debug (XMD) and** the **GNU debugger (gdb)**

C. Operate in hardware
   - **Generate the bitstream and configure the FPGA** using iMPACT

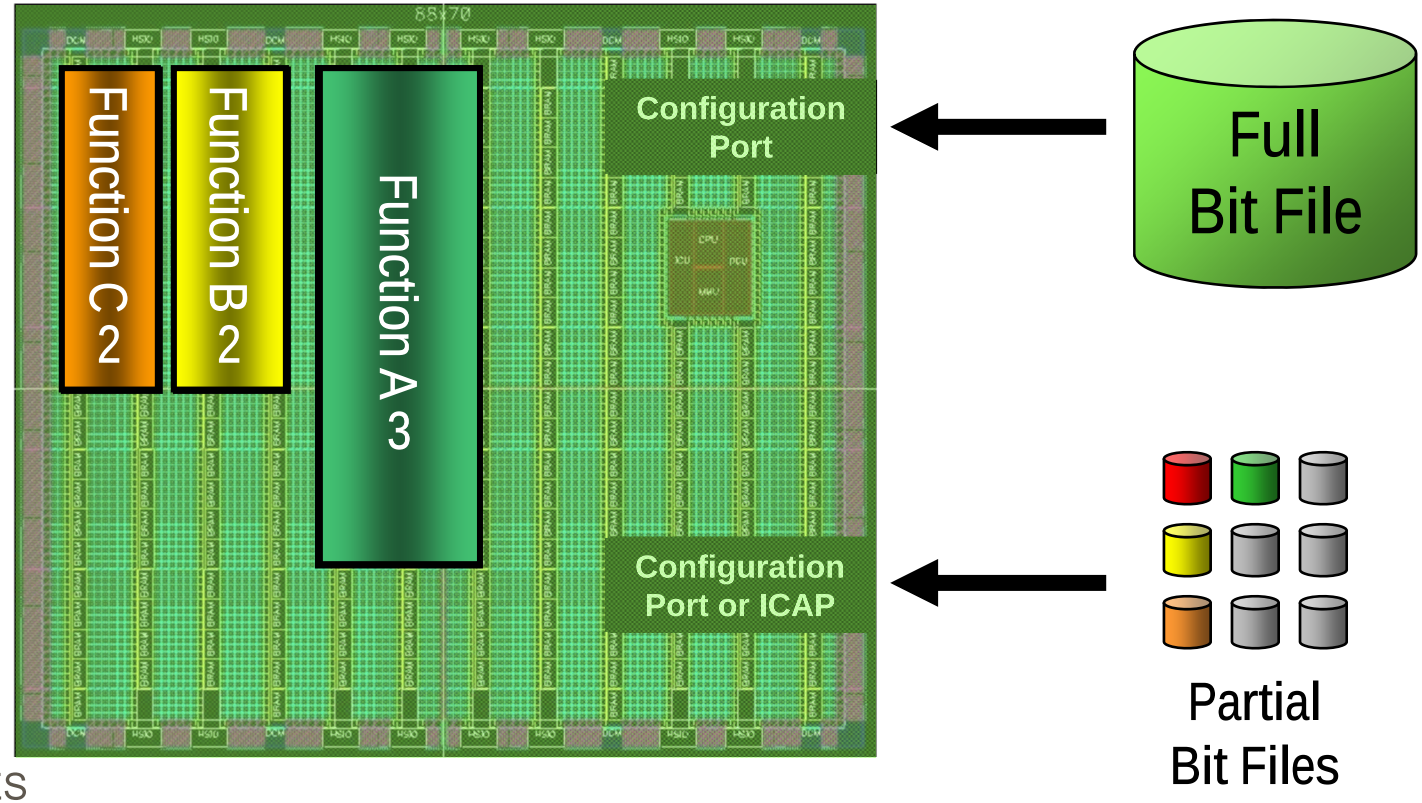
D. Deploy
   - **Initialize external flash memory** using the **Flash Writer utility or boot from** an external compact flash configuration file generated using the **System ACE File generator (GenACE)** script

EDK Tool Flow



Partial Reconfiguration

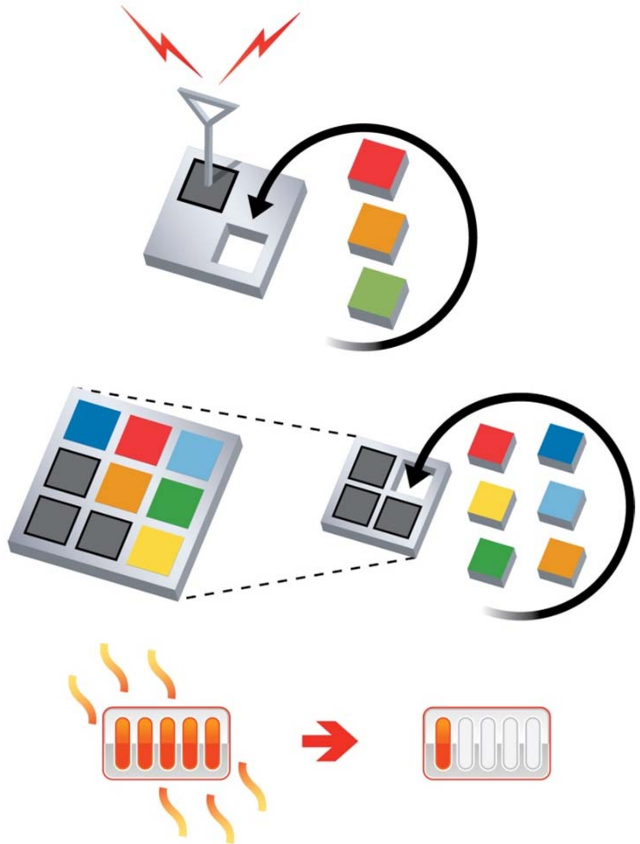
- **Partial Reconfiguration is the ability to dynamically modify blocks of logic by downloading partial bit files while the remaining logic continues to operate without interruption.**



Partial Reconfiguration

Technology and Benefits

- Partial Reconfiguration enables:
  - System Flexibility
    - Perform more functions while maintaining communication links
  - Size and Cost Reduction
    - Time-multiplex the hardware to require a smaller FPGA
  - Power Reduction
    - Shut down power-hungry tasks when not needed

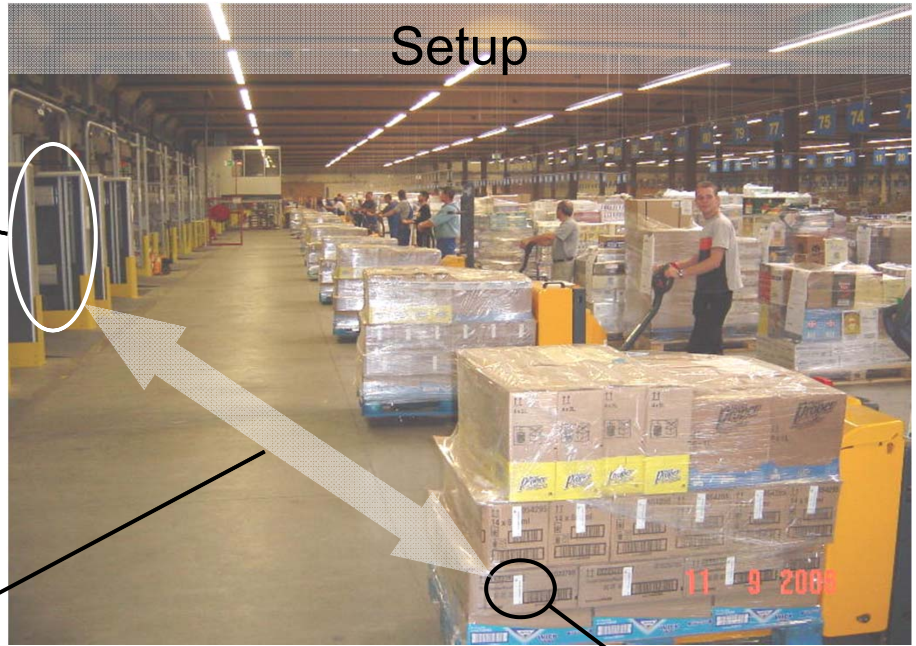


Use Case - Simulation Platform for UHF RFID

Rapid Prototyping with FPGAs

Ultra High Frequency – Radio Frequency IDentification systems

- Reader designs
- Antenna designs



- Communication
- Energy
- UHF field distortions

- Tag design
- ASIC design

Motivation

- Evaluate and optimize application setups
  - Reduced installation time
  - Reduced on site evaluation time
  - Proof of user requirements
  - Worst case scenarios evaluation

- Next generation protocol and product development

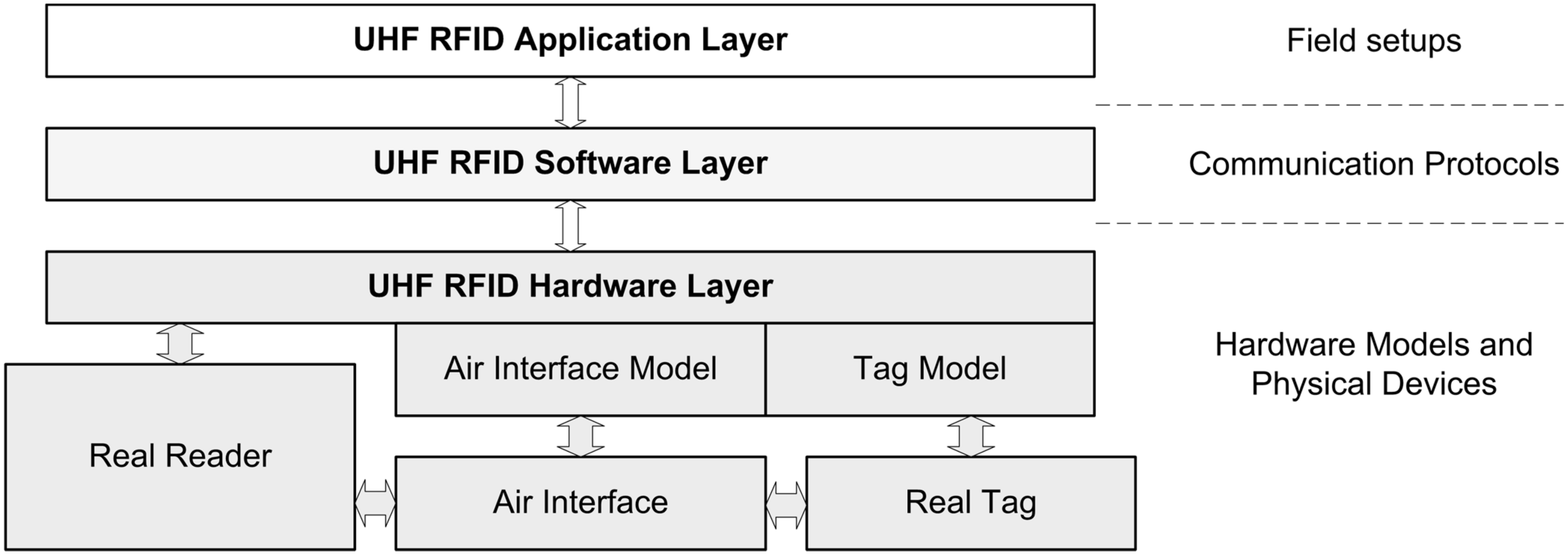
⇓

Evaluation of options by simulating models

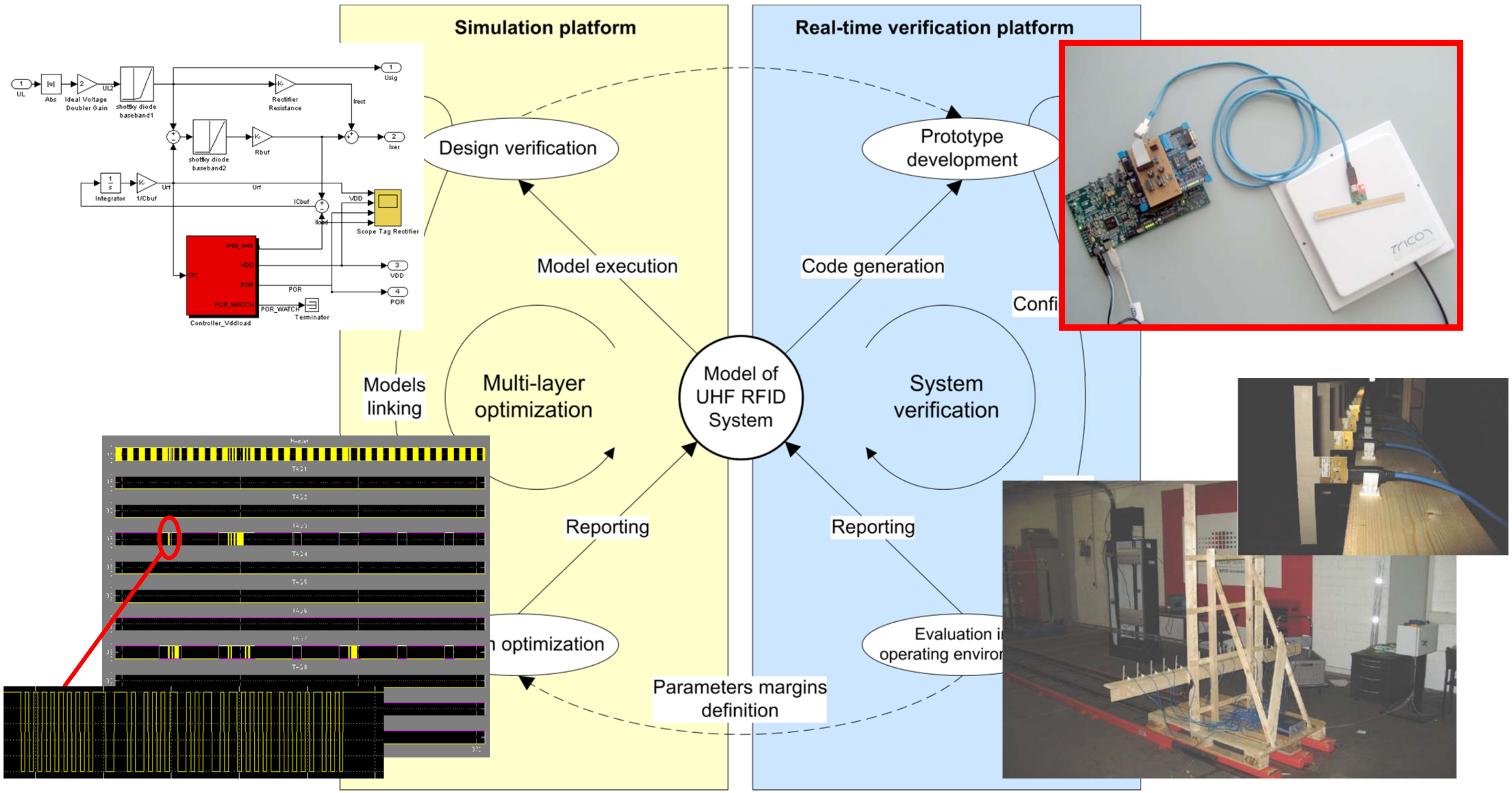
Offline simulations of large tag populations

Verification of designs by real-time simulation of models in target setups

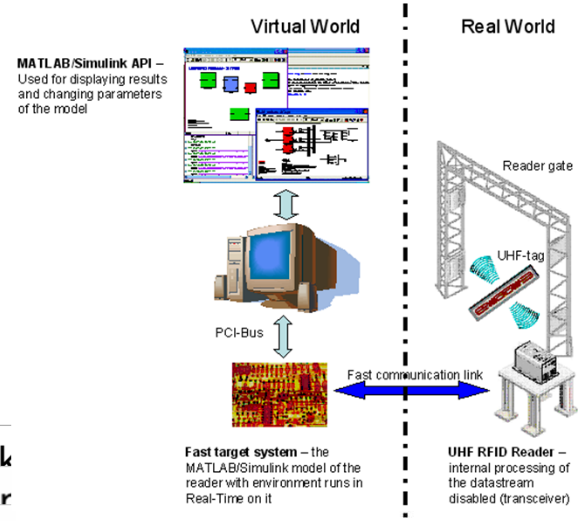
A New Framework for Real-time Verification and Optimization of UHF RFID Systems



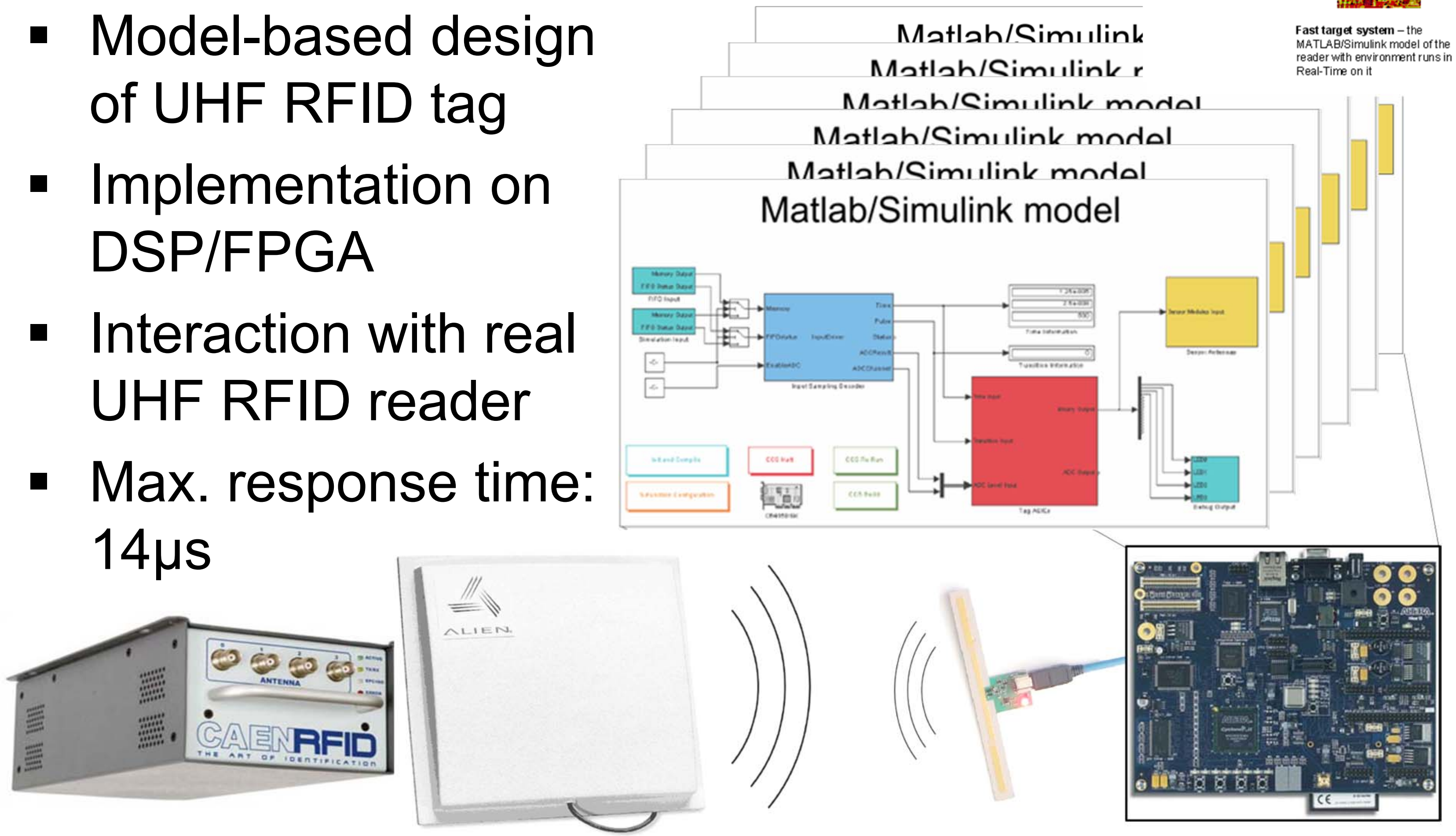
Platforms for Verification and Optimization of UHF RFID Systems



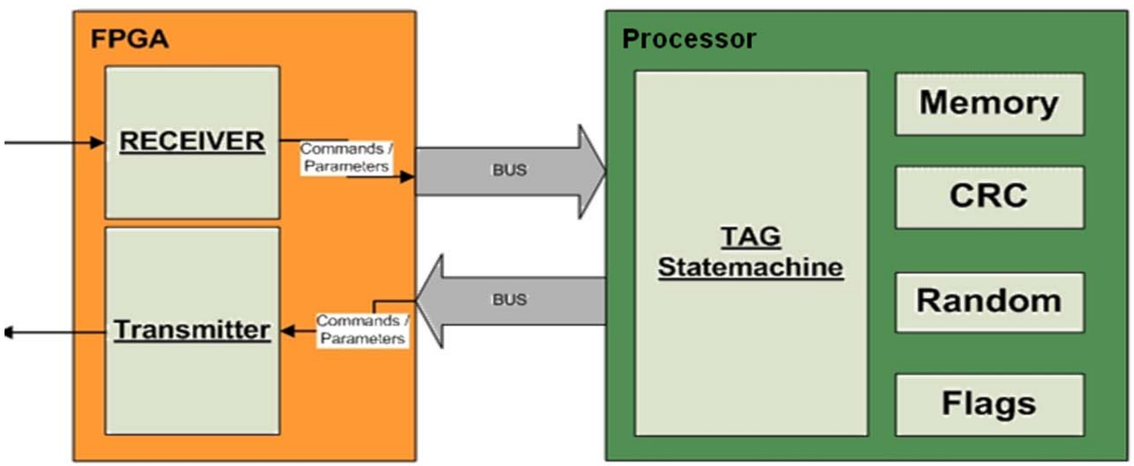
Hardware-In-the-Loop Simulation



- Model-based design of UHF RFID tag
- Implementation on DSP/FPGA
- Interaction with real UHF RFID reader
- Max. response time: 14µs



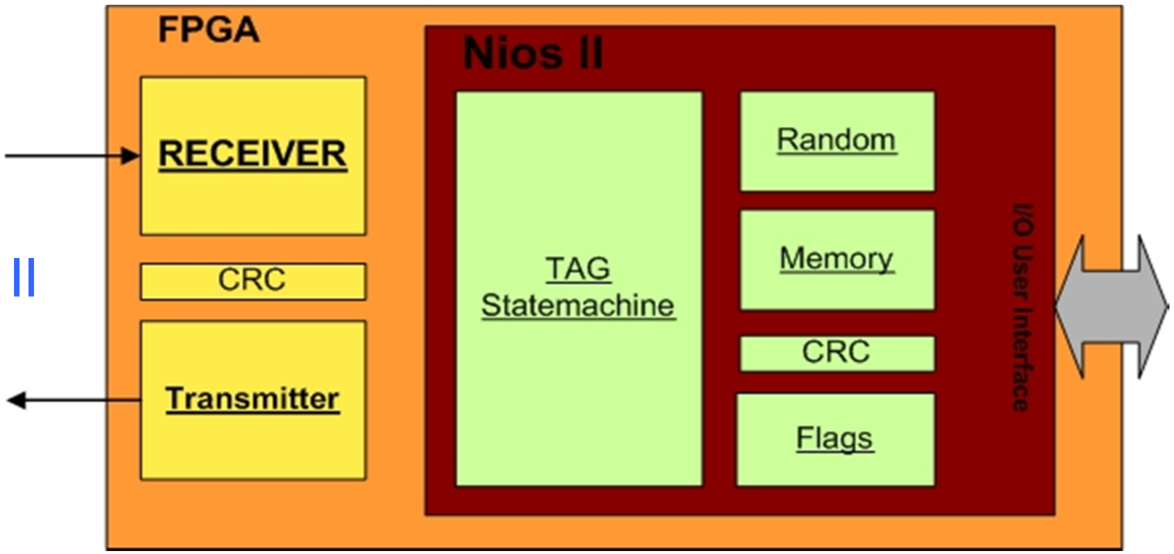
FPGA-based HIL Simulation



- Time critical parts implemented in hardware (synthesized on FPGA)
- Non time critical parts implemented in software (NIOS II)

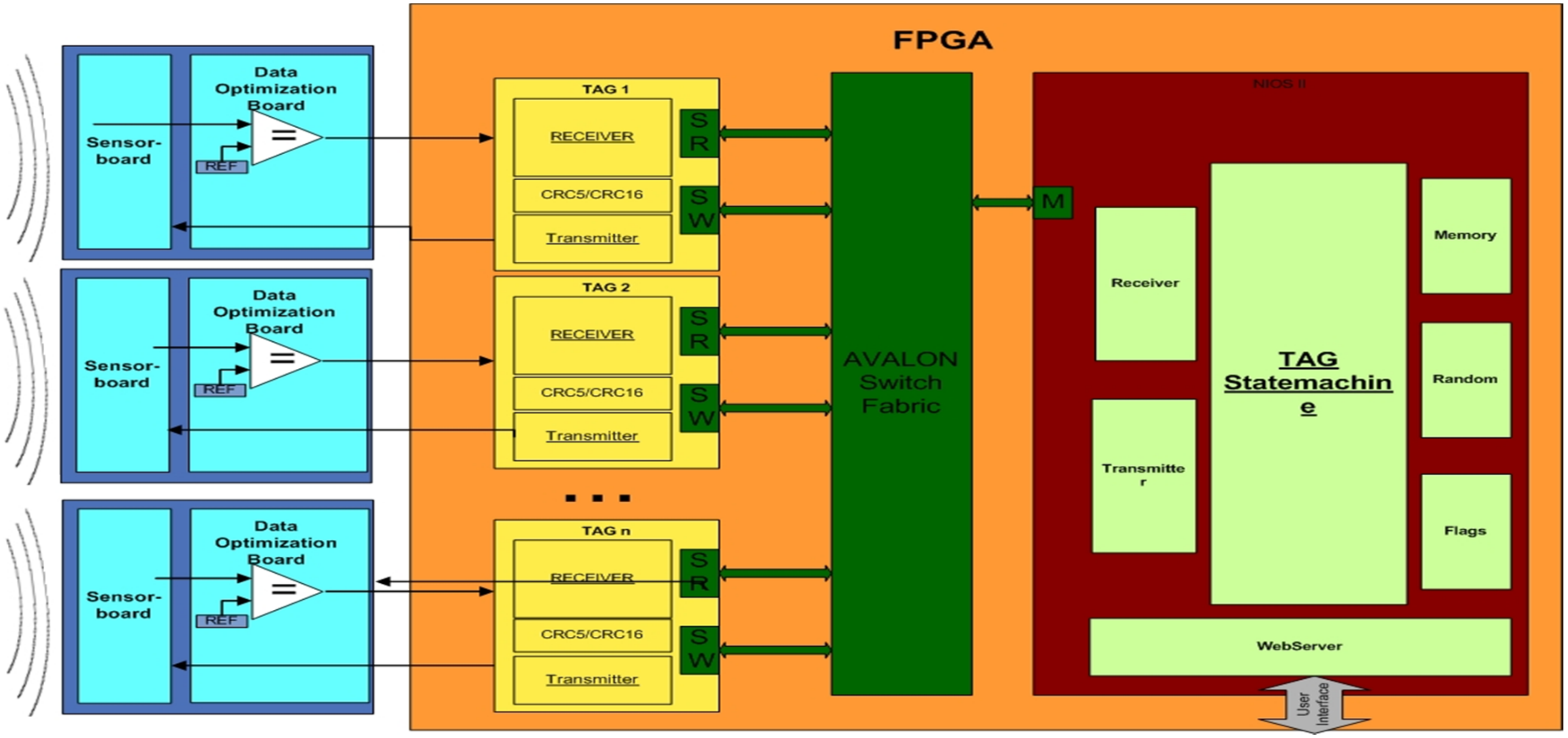
FPGA - SOPC

- HW and SW on the FPGA
- Software on the NIOS II soft-core processor
- Communication over common bus

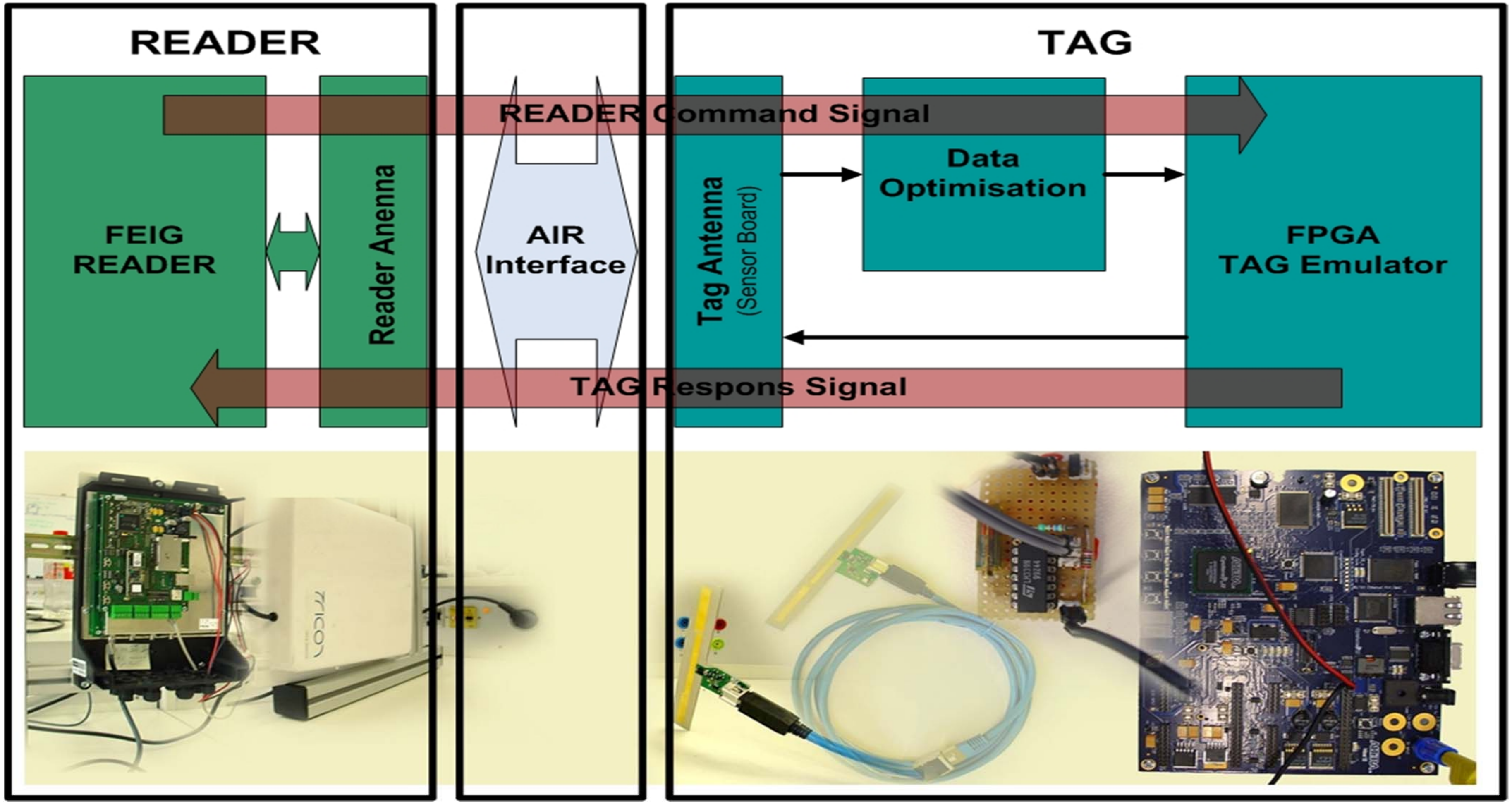


Multiple Tag Design

- Time critical parts implemented in hardware for every simulated UHF RFID tag = Parallel execution
- Non time critical parts implemented in software just once = Sequential execution



Implemented Prototype



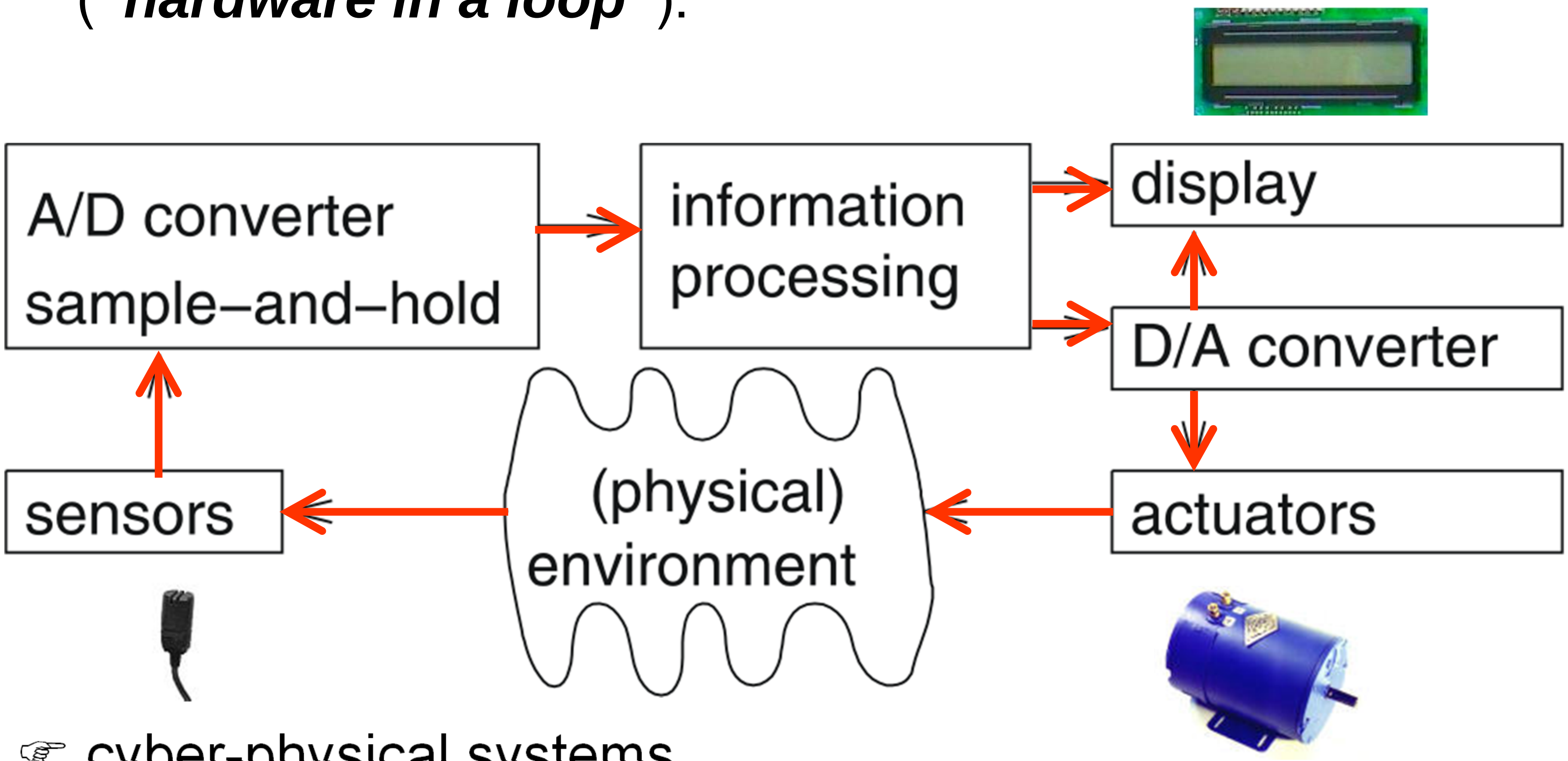
Conclusion

Two implementations:

- DSP TMS320C6416 simulates a model of one tag in real-time
  - No parallel execution achieved without manual code optimization
- FPGA architecture with soft-core processor achieves to simulate 4 tags on one HW
  - 20% FPGA Chip area utilized
  - HW max delay of ~10ns
  - SW is not optimized for performance (C++) → improvements possible

Embedded System Hardware

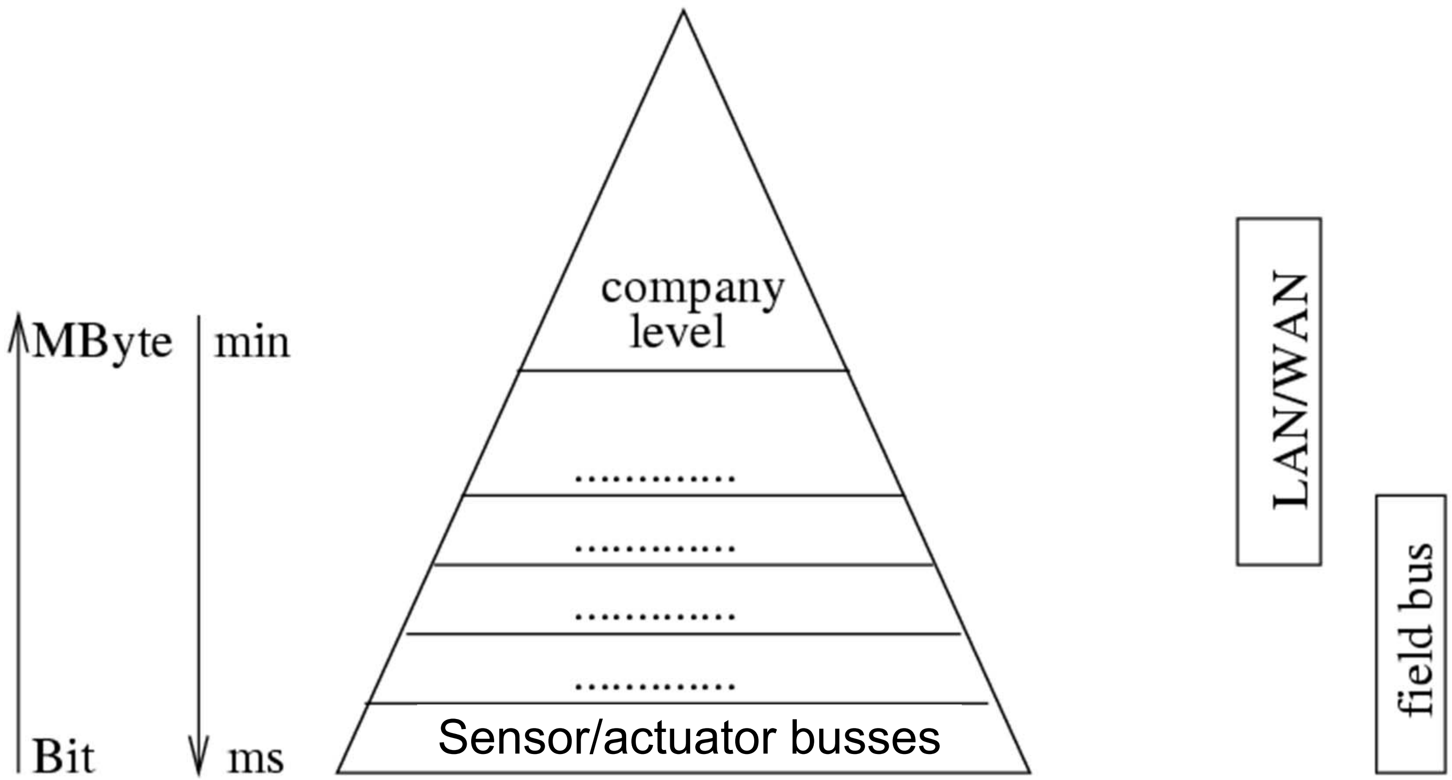
- Embedded system hardware is frequently used in a loop (***“hardware in a loop“***):



☞ cyber-physical systems

Communication: Hierarchy

- Inverse relation between volume and urgency quite common:

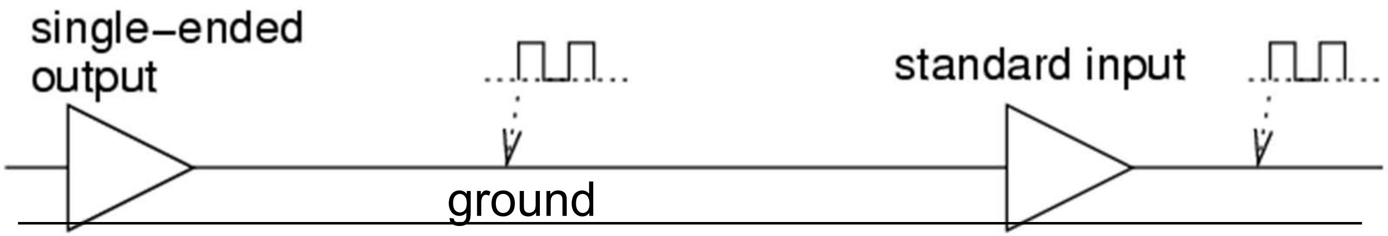


Communication
- Requirements -

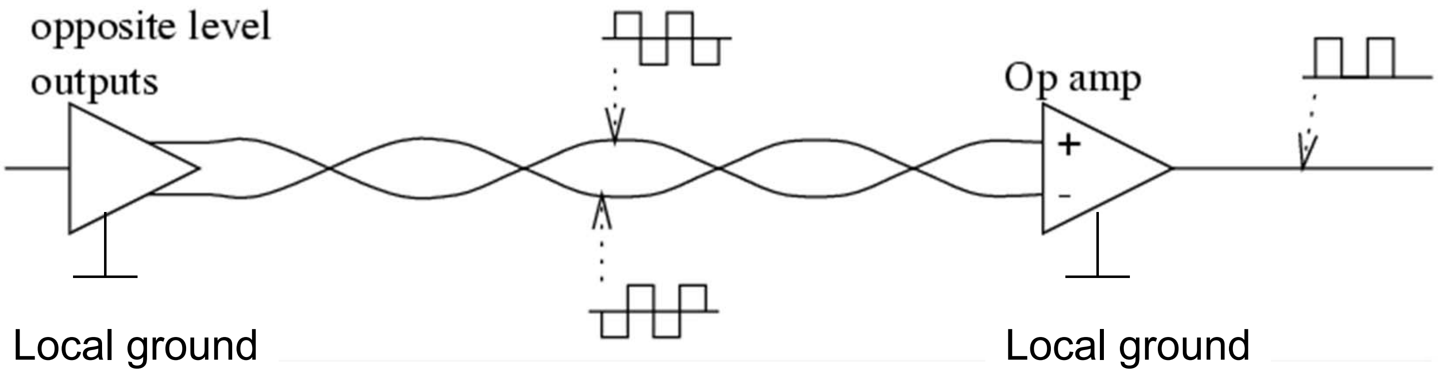
- Real-time behavior
- Efficient, economical
  (e.g. centralized power supply)
- Appropriate bandwidth and communication delay
- Robustness
- Fault tolerance
- Maintainability
- Diagnosability
- Security
- Safety

Basic techniques: Electrical robustness

- Single-ended vs. differential signals



Voltage at input of Op-Amp positive → '1'; otherwise → '0'



Combined with twisted pairs; Most noise added to both wires.

Evaluation

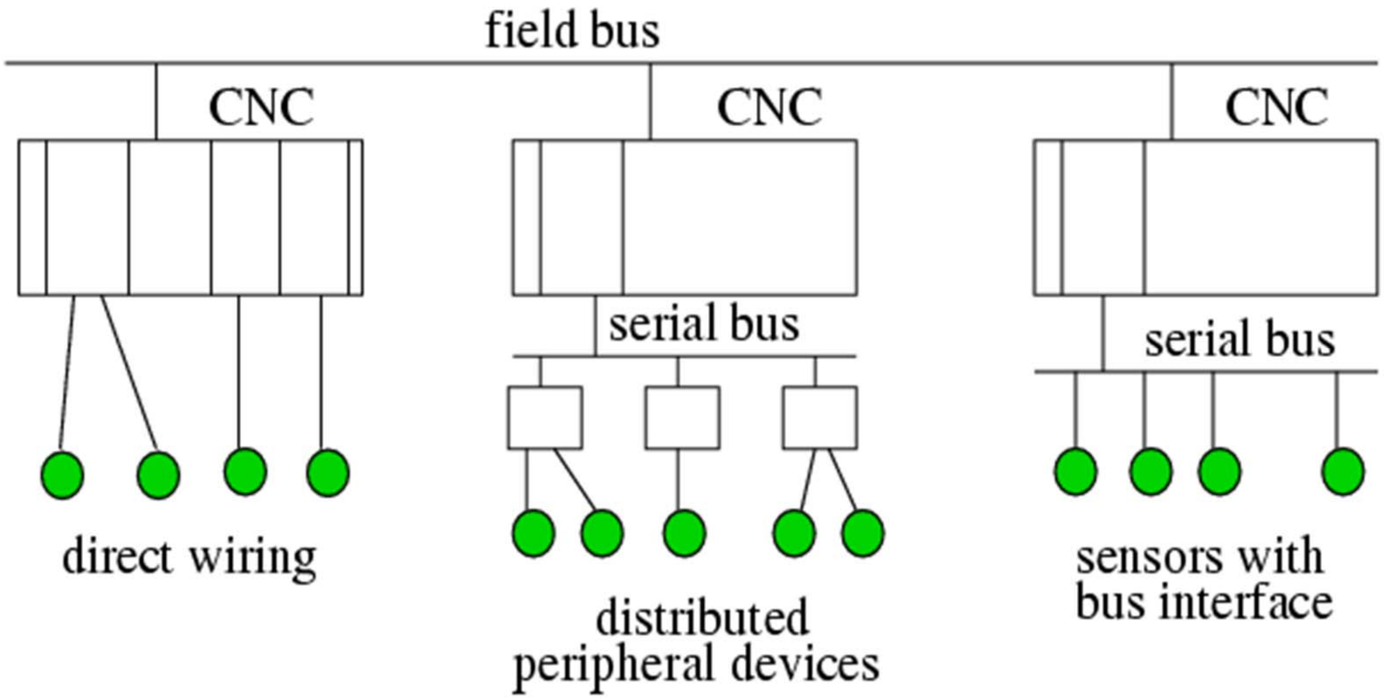
- **Advantages:**
  - Subtraction removes most of the noise
  - Changes of voltage levels have no effect
  - Reduced importance of ground wiring
  - Higher speed
- **Disadvantages:**
  - Requires negative voltages
  - Increased number of wires and connectors
- **Applications:**
  - USB, FireWire, ISDN
  - Ethernet (STP/UTP CAT 5/6 cables)
  - differential SCSI
  - High-quality analog audio signals

Real-time behavior

- Carrier-sense multiple-access/collision-detection (CSMA/CD, Standard Ethernet) no guaranteed response time.
- Alternatives:
  - token rings, token busses
  - Carrier-sense multiple-access/collision-avoidance (CSMA/CA)
    - WLAN techniques with request preceding transmission
    - Each partner gets an ID (priority). After each bus transfer, all partners try setting their ID on the bus; partners detecting higher ID disconnect themselves from the bus. Highest priority partner gets guaranteed response time; others only if they are given a chance.

Sensor/actuator busses

1. **Sensor/actuator busses**: Real-time behavior very important; different techniques:



Many wires　　less wires　　expensive & flexible

Field busses: Profibus

- More powerful/expensive than sensor interfaces; mostly serial. Emphasis on transmission of small number of bytes.
- Examples:

1. **Process Field Bus (Profibus)**
   Designed for factory and process automation.
   Focus on **safety**; comprehensive protocol mechanisms.
   Claiming 20% market share for field busses.
   Token passing.
   $\leqq$93.75 kbit/s (1200 m);1500 kbits/s (200m);
   12 Mbit/s (100m)
   Integration with Ethernet via Profinet.

[http://www.profibus.com/]

Controller area network (CAN)

- **2. Controller area network (CAN)**
  - Designed by Bosch and Intel in 1981;
  - used in cars and other equipment;
  - differential signaling with twisted pairs,
  - arbitration using CSMA/CA,
  - throughput between 10kbit/s and 1 Mbit/s,
  - low and high-priority signals,
  - maximum latency of 134 µs for high priority signals,
  - coding of signals similar to that of serial (RS-232) lines of PCs, with modifications for differential signaling.
  - See //www.can.bosch.com

Time-Triggered-Protocol (TTP)

3. The **Time-Triggered-Protocol (TTP)** [Kopetz et al.]
   for fault-tolerant safety systems like airbags in cars.

FlexRay

4. **FlexRay**: developed by the FlexRay consortium (BMW, Ford, Bosch, DaimlerChrysler, …)
   Combination of a variant of the TTP and the Byteflight [Byteflight Consortium, 2003] protocol.
   Specified in SDL.
   - Improved error tolerance and time-determinism
   - Meets requirements with transfer rates >> CAN std.
     High data rate can be achieved:
     - initially targeted for ~ 10Mbit/sec;
     - design allows much higher data rates
   - TDMA (Time Division Multiple Access) protocol:
     Fixed time slot with exclusive access to the bus
   - Cycle subdivided into a static and a dynamic segment.

- See guest lecture from Jan. 11th. 2011

Other field busses

- **LIN:** low cost bus for interfacing sensors/actuators in the automotive domain
- **MOST:** Multimedia bus for the automotive domain (not really a field bus)
- **MAP:**MAP is a bus designed for car factories.
- **EIB:**The European Installation Bus (EIB) is a bus designed for smart homes. **European Installation Bus (EIB)**
  Designed for smart buildings; CSMA/CA; low data rate.
- **IEEE 488:** Designed for laboratory equipment.

- Attempts to use standard Ethernet.
  However, timing predictability remains a serious issue.

Wireless communication: Examples

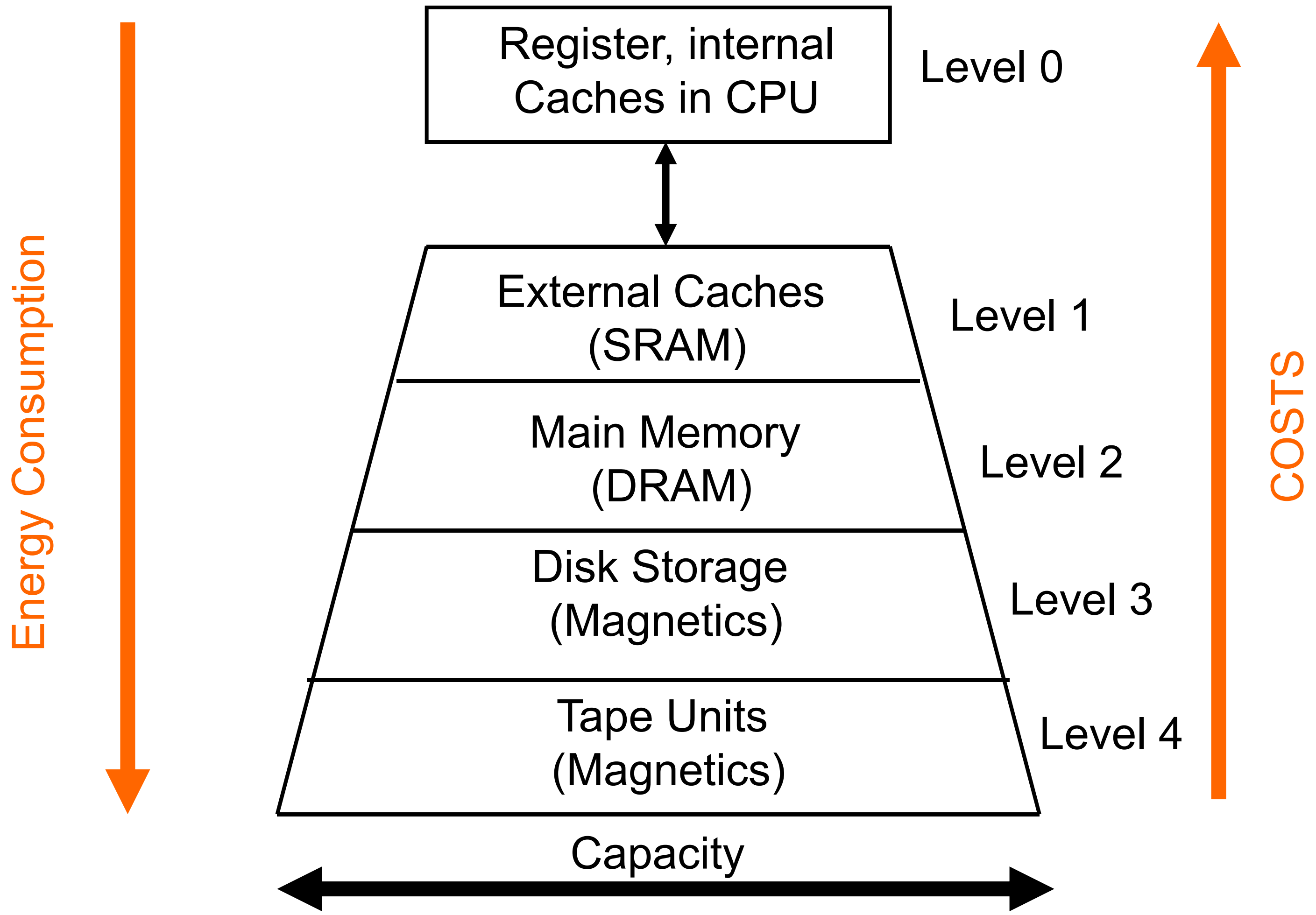
- IEEE 802.11 a/b/g/n
- UMTS; HSPA
- DECT
- Bluetooth
- ZigBee
- NFC

Timing predictability of wireless communication?

Memory

- For the memory, efficiency is again a concern:
  - speed (latency and throughput); predictable timing
  - energy efficiency
  - size
  - cost
  - other attributes (volatile vs. persistent, etc)

Memory hierarchy



"Small is beautiful"
(in terms of energy consumption, access times, size)

The Principle of Locality

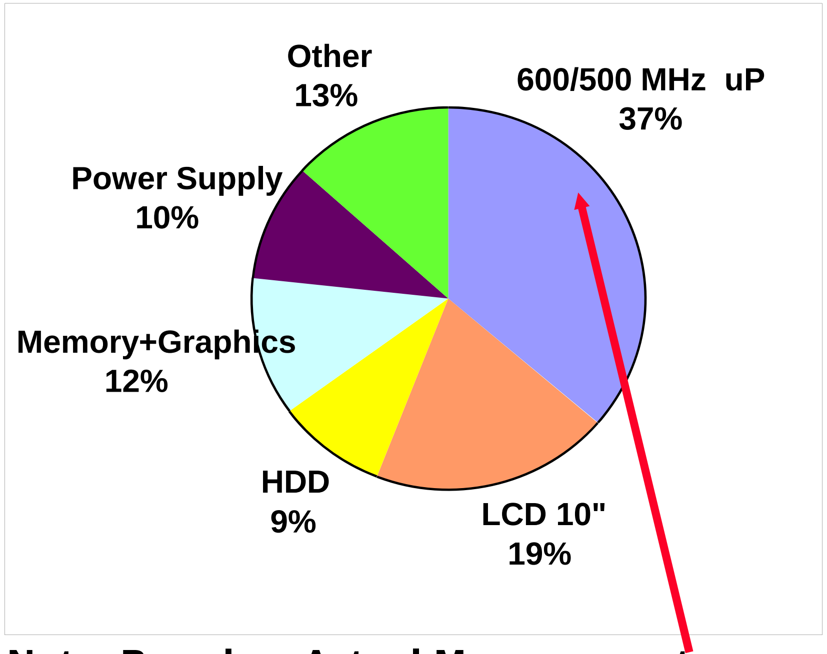
- The Principle of Locality:
  - Program access a relatively small portion of the address space at any instant of time.

- Two Different Types of Locality:
  - **Temporal Locality** (Locality in Time): If an item is referenced, it will tend to be referenced again soon (e.g., loops, reuse)

  - **Spatial Locality** (Locality in Space): If an item is referenced, items whose addresses are close by tend to be referenced soon (e.g., straightline code, array access)

How much of the energy consumption of a system is memory-related?

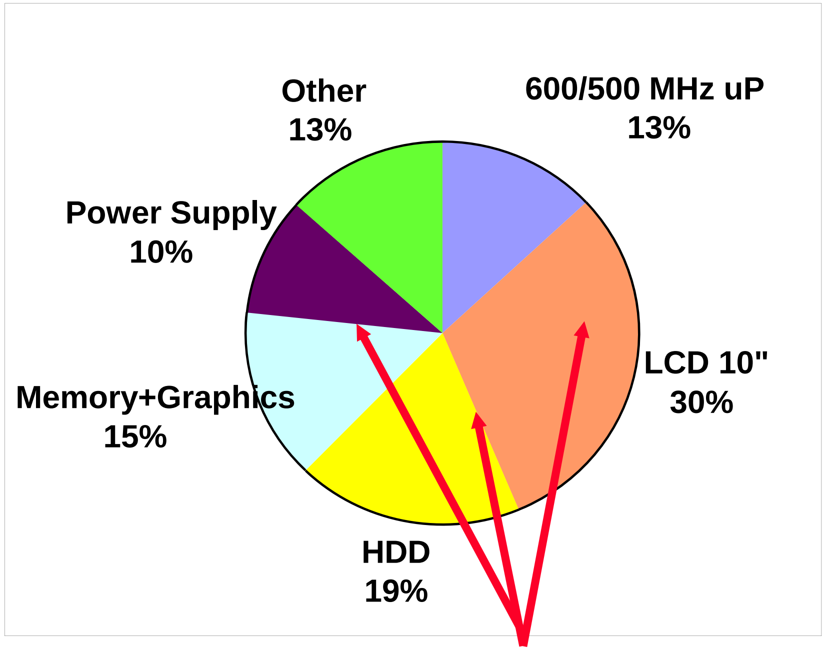
Mobile PC
Thermal Design (TDP) System Power



Note: Based on Actual Measurements

CPU Dominates Thermal Design Power

Mobile PC
Average System Power

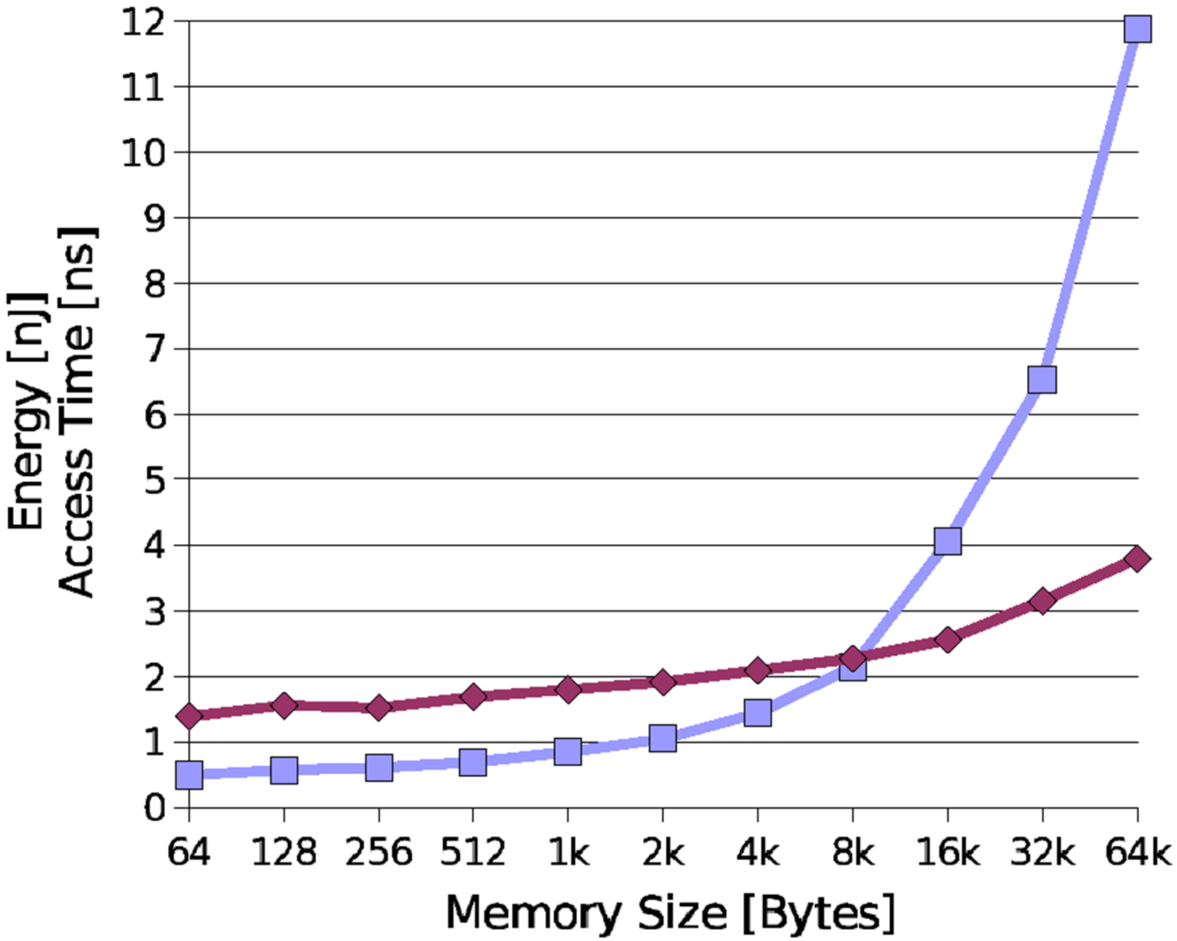


Multiple Platform Components Comprise Average Power

[Courtesy: N. Dutt; Source: V. Tiwari]

Access times and energy consumption increases with the size of the memory

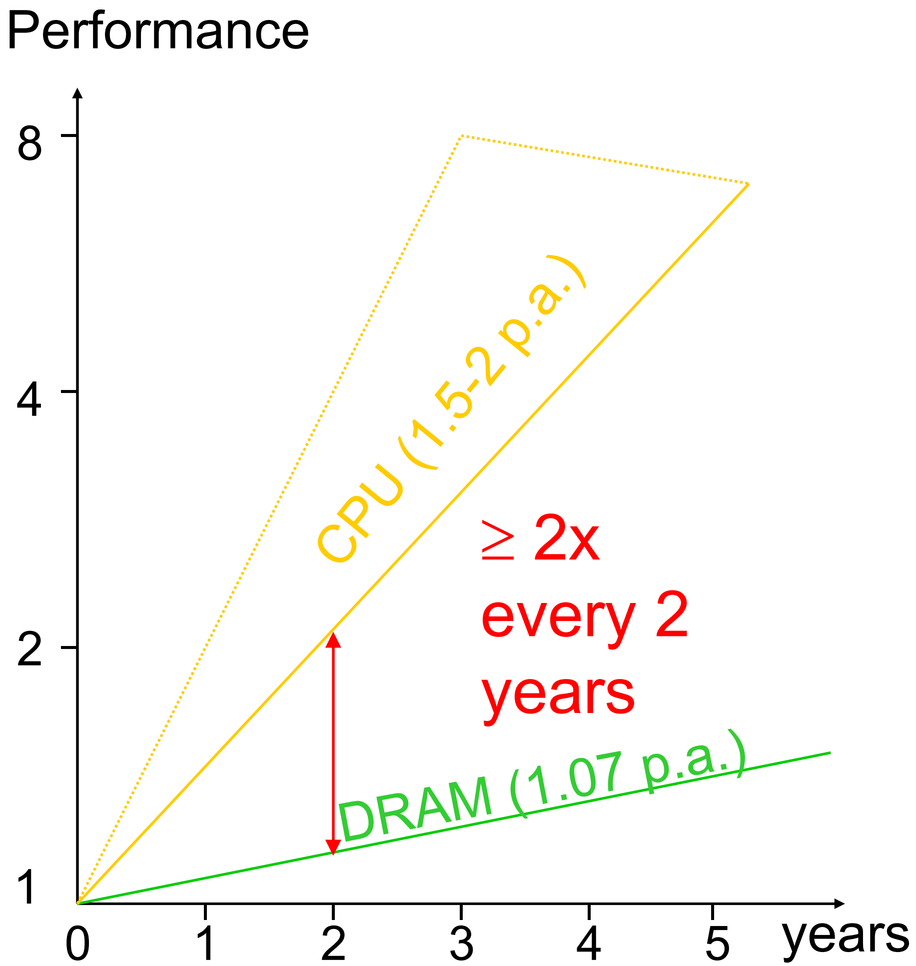
Example (CACTI Model):



"Currently, the size of some applications is doubling every 10 months"
[STMicroelectronics, Medea+ Workshop, Stuttgart, Nov. 2003]

Access-times will be a problem

- Speed gap between processing and main DRAM increases



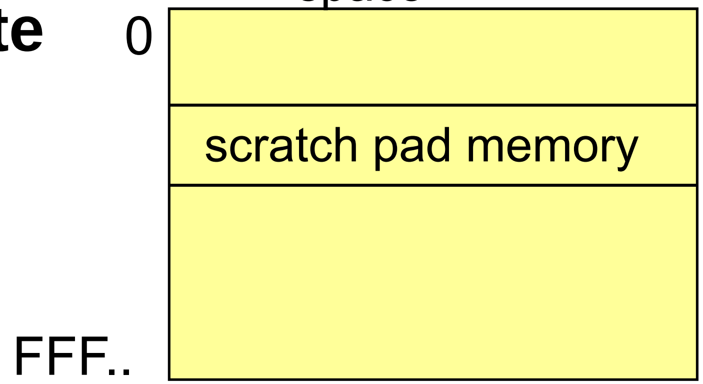
→ Use smaller and faster memories that act as a buffer between the memory

[P. Machanik: Approaches to Addressing the Memory Wall, TR Nov. 2002, U. Brisbane]

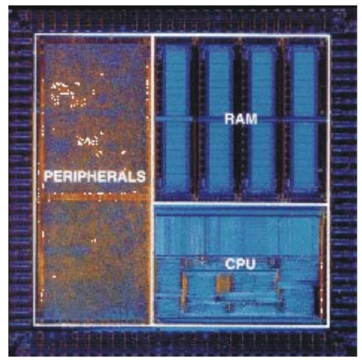
Hierarchical memories using scratch pad memories (SPM)

SPM is a small, physically separate memory mapped into the address space

- Address space

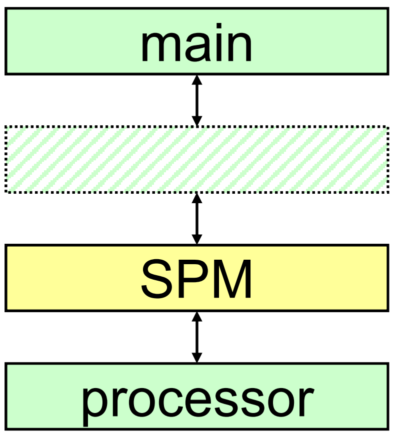


Example

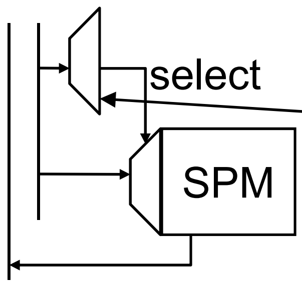


ARM7TDMI cores, well-known for low power consumption

Hierarchy



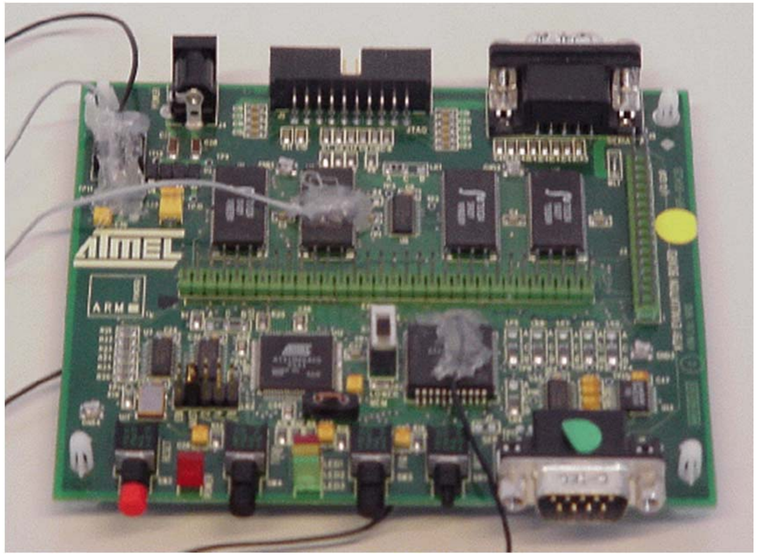
no tag memory

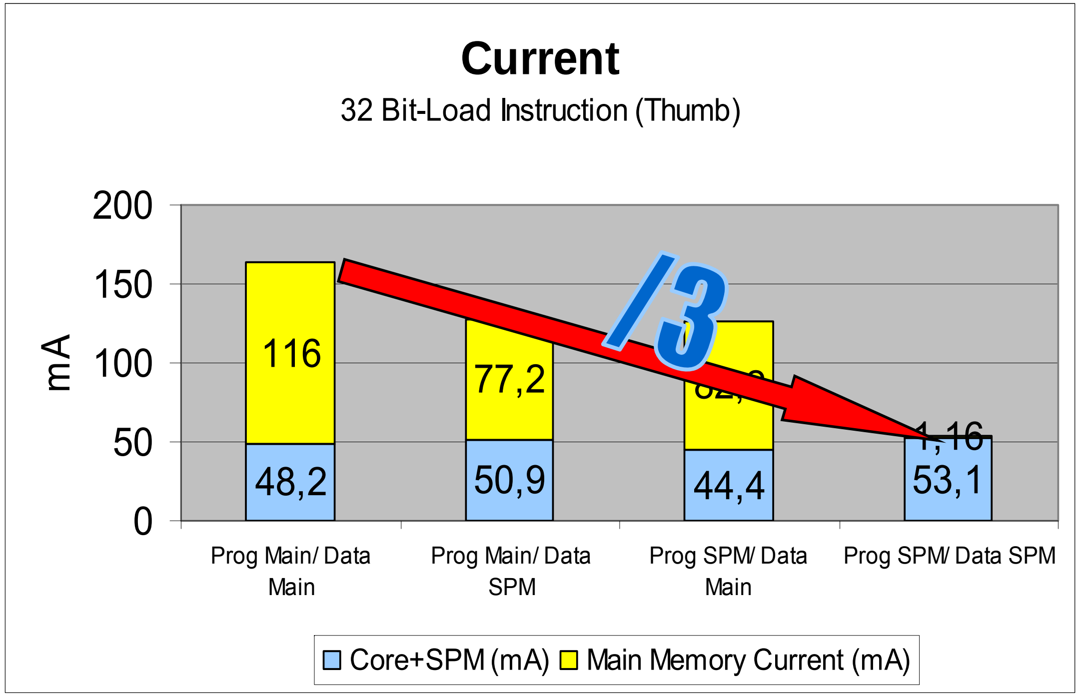


Selection is by an appropriate address decoder (simple!)

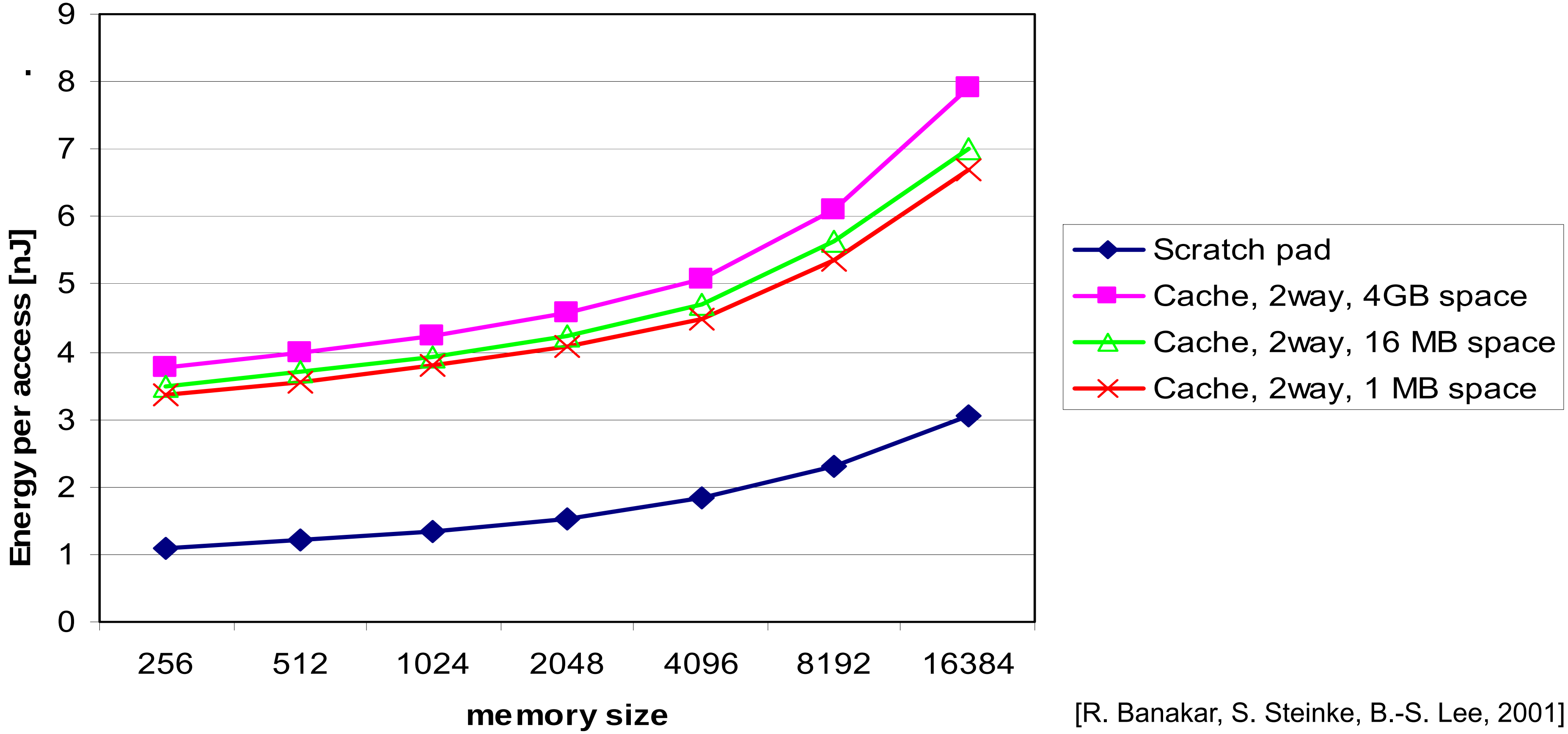
Comparison of currents using measurements

E.g.: ATMEL board with ARM7TDMI and ext. SRAM





Why not just use a cache ?



[R. Banakar, S. Steinke, B.-S. Lee, 2001]

Overview of embedded systems design