

Lecture 11: Embedded Processors

Seyed-Hosein Attarzadeh-Niaki

Based on the slides by P. Marwedel

Review

- Synchronous reactive MoC
- Timed MoCs
 - Time-triggered model
 - Discrete-event model
 - Ptides

Outline

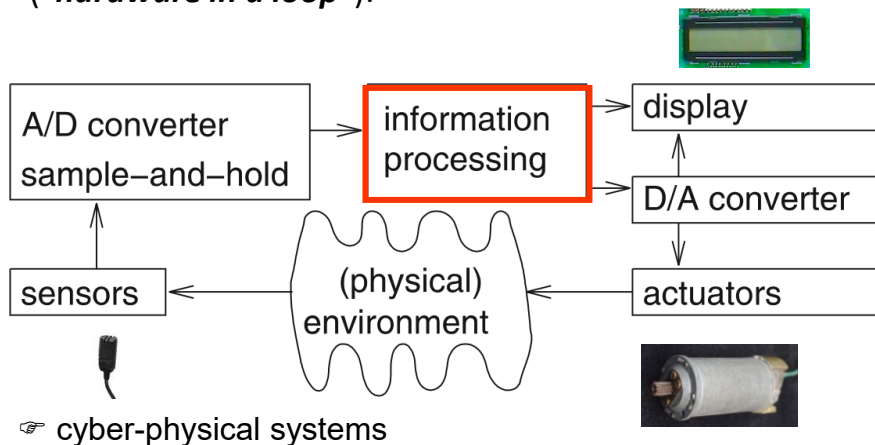
- Types of processing units
- Efficiency of embedded processors
 - Power/energy efficiency
 - Code size efficiency
 - Runtime efficiency
- Realtime capability

Embedded Real-Time Systems

3

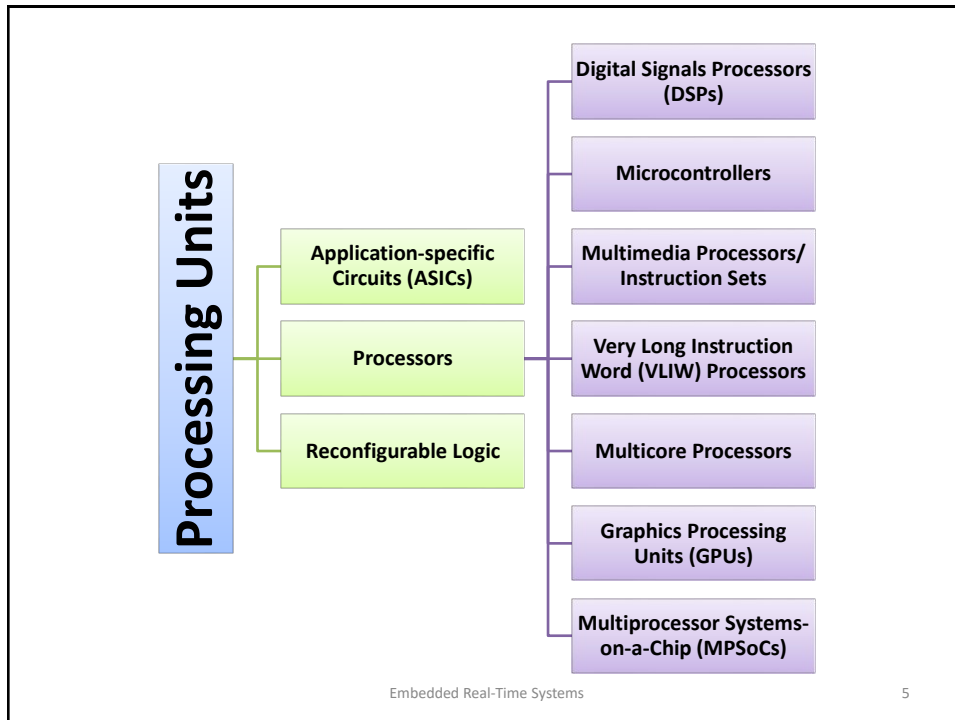
Embedded & CPS System Hardware

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



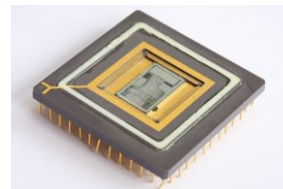
Embedded Real-Time Systems

4



Application Specific Circuits (ASICs) or Full Custom Circuits

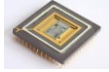




- Approach suffers from
 - long design times,
 - lack of flexibility (changing standards) and
 - high costs (e.g. mill. \$ mask costs).
- Custom-designed circuits necessary
 - if ultimate speed or
 - energy efficiency is the goal and
 - large numbers can be sold.



☞ HW synthesis not covered in this course, let's look at processors

Efficiency: Applied to Processing

– CPS & ES must be **efficient**

- ➡ • Code-size efficient (especially for systems on a chip) 
- ➡ • Run-time efficient 
- Weight efficient 
- Cost efficient 
- ➡ • Energy efficient 

Embedded Real-Time Systems

7

Why care about energy efficiency ?

Execution platform	Relevant during use?		
	Plugged	Uncharged periods	Unplugged
E.g.	Factory	Car	Sensor
Global warming	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Cost of energy	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Increasing performance	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Problems with cooling, avoiding hot spots	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Avoiding high currents & metal migration	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Reliability	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Energy a very scarce resource	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>



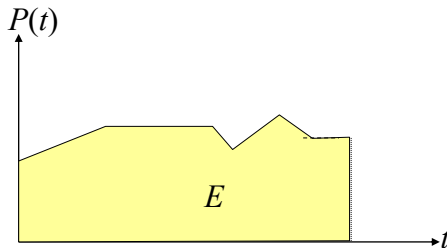
Power ☐ ☐

Embedded Real-Time Systems

8

Should we care about **energy** consumption or **power** consumption?

$$E = \int P(t) dt$$



Both are closely related,
but still different

- Minimizing **power consumption** important for

- design of the power supply & regulators
- dimensioning of interconnect, short term cooling



- Minimizing **energy consumption** important due to

- restricted availability of energy (mobile systems)
- cooling: high costs, limited space
- thermal effects
- dependability, long lifetimes

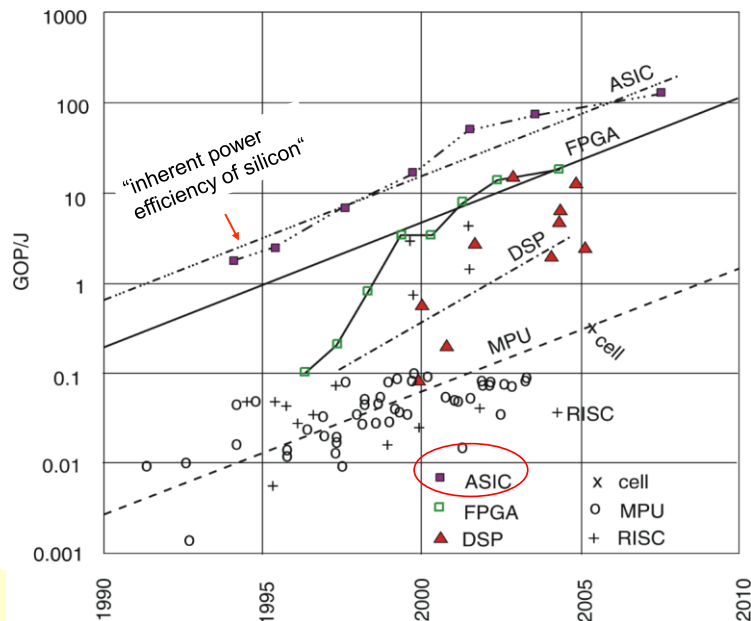


👉 In general, we need to care about both

Embedded Real-Time Systems

9

Energy Efficiency of different target platforms



© Hugo De Man,
IMEC, Philips, 2007

Embedded Real-Time Systems

10

PCs: Surpassed hot (kitchen) plate ...? Why not use it?



Strictly speaking, energy is not “consumed”, but converted from electrical energy into heat energy

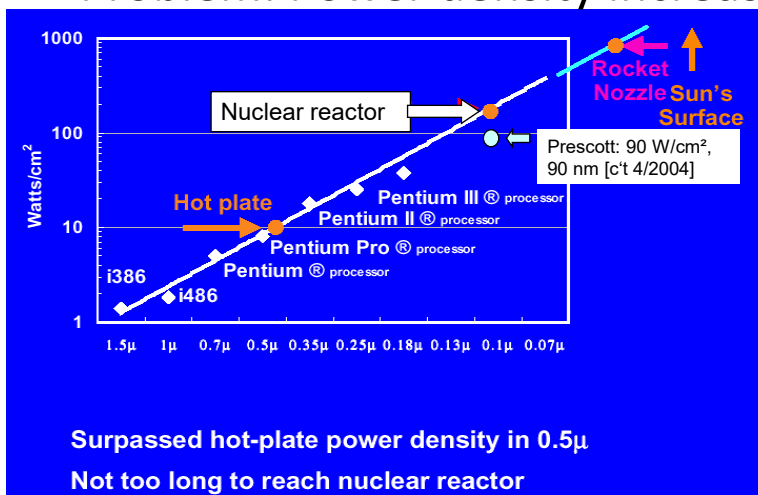
http://www.phys.ncku.edu.tw/~htsu/humor/fry_egg.html

Embedded Real-Time Systems

11

PCs

Problem: Power density increasing

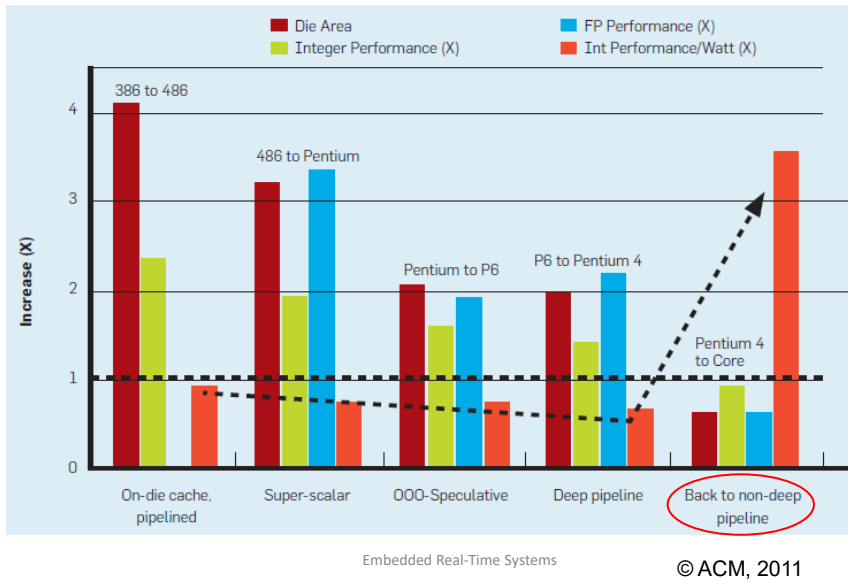


© Intel
M. Pollack,
Micro-32

Embedded Real-Time Systems

12

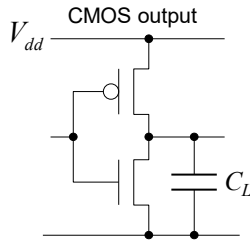
Keep it Simple, Stupid (KISS)



S. Borkar, A. Chien: The future of microprocessors, Communications of the ACM, May 2011

Static & Dynamic Power Consumption

- **Dynamic** power consumption: Power consumption caused by charging capacitors when logic levels are switched.



$$P = \alpha C_L V_{dd}^2 f \text{ with}$$

α : switching activity

C_L : load capacitance

V_{dd} : supply voltage

f : clock frequency

Decreasing V_{dd} reduces P quadratically

- **Static** power consumption (caused by leakage current): power consumed in the absence of clock signals
- Leakage becoming more important due to smaller devices

Static & Dynamic Power Consumption

Power consumption of CMOS circuits (ignoring leakage):

$$P = \alpha C_L V_{dd}^2 f \text{ with}$$

α : switching activity

C_L : load capacitance

V_{dd} : supply voltage

f : clock frequency

Delay for CMOS circuits:

$$\tau = k C_L \frac{V_{dd}}{(V_{dd} - V_t)^2} \text{ with}$$

V_t : threshold voltage

($V_t < V_{dd}$)

☞ Decreasing V_{dd} reduces P quadratically, while the run-time of algorithms is only linearly increased

Making processors Energy-Efficient

- Three techniques
 - Parallel execution
 - Dynamic power management (DPM)
 - Dynamic voltage and frequency scaling (DVFS)

Low voltage, parallel operation more efficient than high voltage, sequential

Basic equations

Power: $P \sim V_{DD}^2$,
 Maximum clock frequency: $f \sim V_{DD}$,
 Energy to run a program: $E = P \times t$, with: $t = \text{runtime}$ (fixed)
 Time to run a program: $t \sim 1/f$

Changes due to parallel processing, with β operations per clock:

Clock frequency reduced to: $f' = f / \beta$,
 Voltage can be reduced to: $V_{DD}' = V_{DD} / \beta$,
 Power for parallel processing: $P^o = P / \beta^2$ per operation,
 Power for β operations per clock: $P' = \beta \times P^o = P / \beta$,
 Time to run a program is still: $t' = t$,
 Energy required to run program: $E' = P' \times t = E / \beta$

☞ Argument in favour of voltage scaling, and parallel processing

Rough approximations!

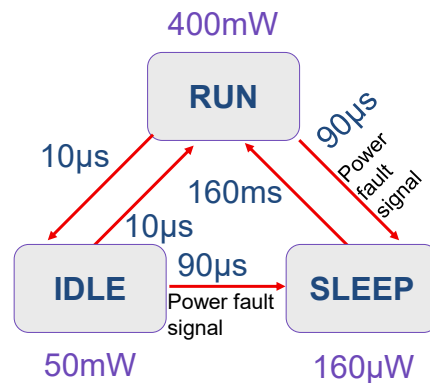
Embedded Real-Time Systems

17

Dynamic Power Management (DPM)

Example: STRONGARM SA1100

- **RUN**: operational
- **IDLE**: a SW routine may stop the CPU when not in use, while monitoring interrupts
- **SLEEP**: Shutdown of on-chip activity

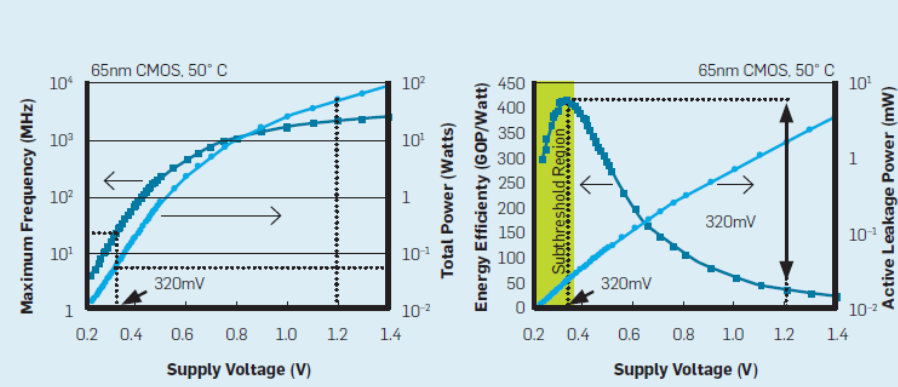


Embedded Real-Time Systems

18

Voltage scaling: Example

Figure 13. Improving energy efficiency through voltage scaling.



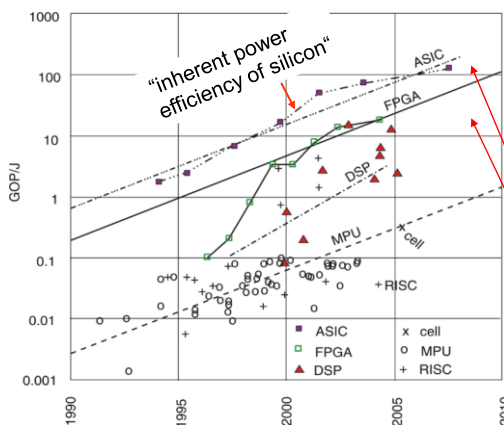
© ACM, 2011

S. Borkar, A. Chien: The future of microprocessors, *Communications of the ACM*, May 2011

Embedded Real-Time Systems

19

More energy-efficient architectures: Domain- and application-specific

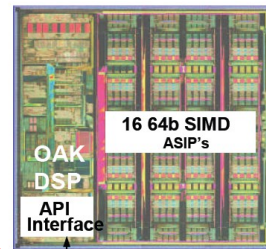


© Hugo De Man: From the Heaven of Software to the Hell of Nanoscale Physics: An Industry in Transition, *Keynote Slides*, ACACES, 2007

Embedded Real-Time Systems

20

VIP for car mirrors
Infineon

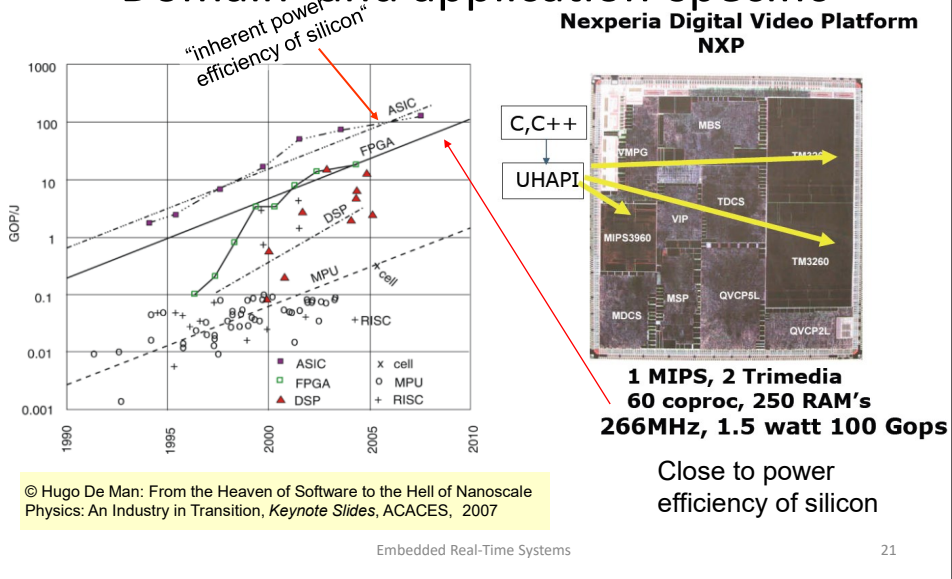


Hd Compiler ← VPL C

200MHz , 0.76 Watt
100Gops @ 8b
25Gops @ 32b

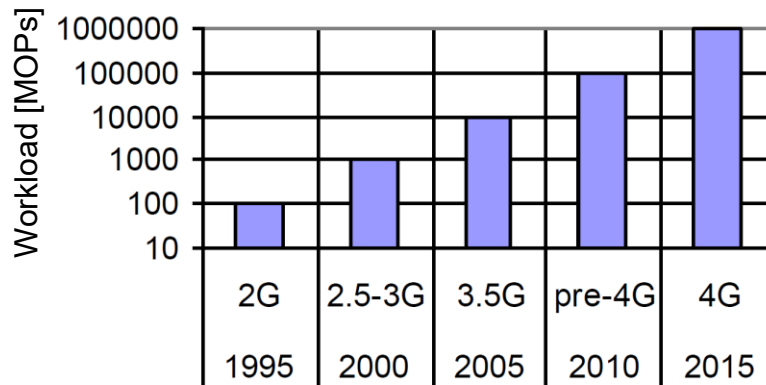
Close to power
efficiency of silicon

Energy-efficient architectures: Domain- and application-specific



Mobile phones: Increasing performance requirements

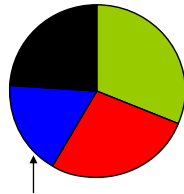
C.H. van Berkel: Multi-Core for Mobile Phones, DATE, 2009;



Many more instances of the power/energy problem

Mobile phones: Where does the power go?

- Mobile phone use, breakdown by type of computation



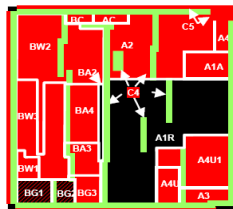
Graphics	(geometry processing, rasterization, pixel shading)
Media	(display & camera processing, video (de)coding)
Radio	(front-end, demodulation, decoding, protocol)
Application	(user interface, browsing, ...)

With special purpose HW!

C.H. van Berkel: Multi-Core for Mobile Phones, DATE, 2009; (no explicit percentages in original paper)

☞ During use, all components & computations relevant

Energy-efficient architectures: Heterogeneous processors (2)Telephony (W-CDMA)



■ Power on
■ Power off

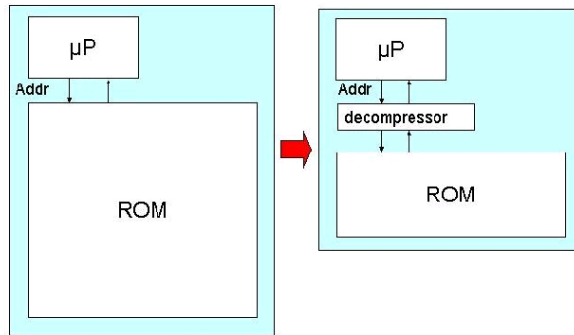
Baseband part	Control	ON
	W-CDMA	ON
	GSM	ON / OFF
Application part	System-domain	ON
	Realtime-domain	OFF
Measured Leakage Current (@ Room Temp, 1.2V)		407 μ A

<http://www.mpsoc-forum.org/2007/slides/Hattori.pdf>

☞ **“Dark silicon”** (not all silicon can be powered at the same time, due to current, power or temperature constraints)

Key requirement #2: Code-size efficiency

- CISC machines
- **Compression techniques:** key idea
 - Overview: <http://www-perso.iro.umontreal.ca/~latendre/codeCompression/codeCompression/node1.html>



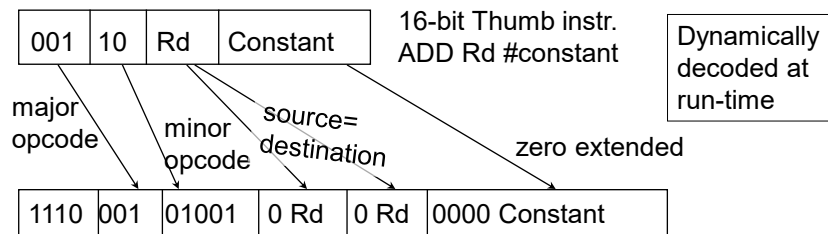
Embedded Real-Time Systems

25

Code-size efficiency

– Compression techniques (continued):

- 2nd instruction set, e.g. ARM Thumb instruction set:



- Reduction to 65-70 % of original code size
- 130% of ARM performance with 8/16 bit memory
- 85% of ARM performance with 32-bit memory

Same approach for LSI TinyRisc, ...
Requires support by compiler, assembler etc.

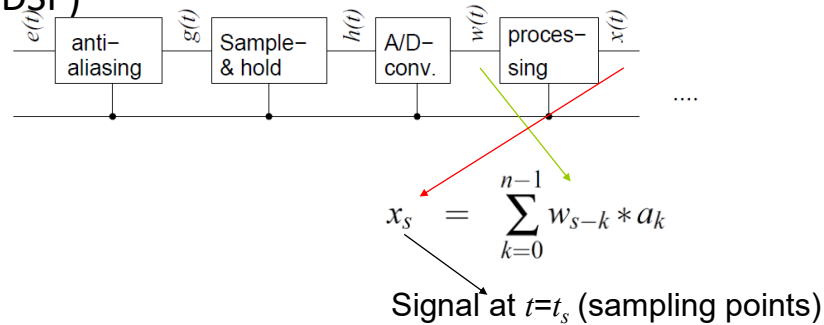
Embedded Real-Time Systems

[ARM, R. Gupta]

26

Key requirement #3: Run-time efficiency

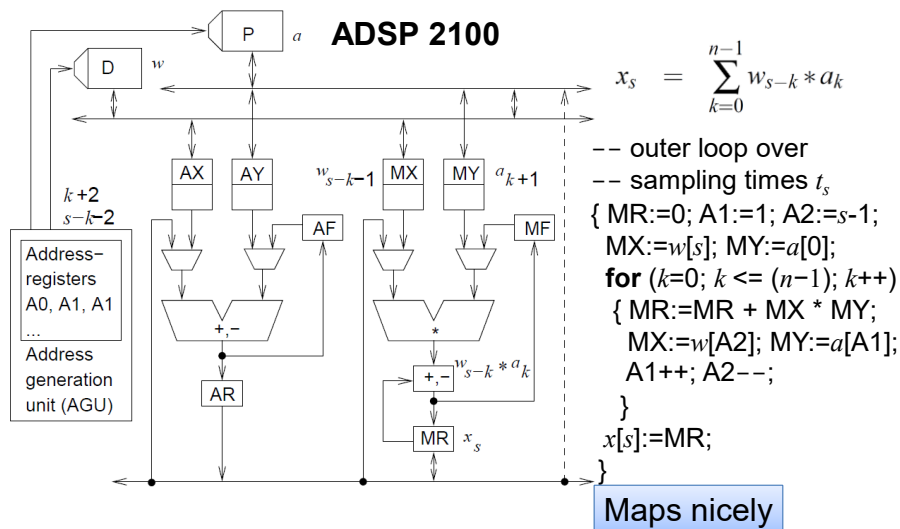
- Domain-oriented architectures
- Example: Filtering in Digital signal processing (DSP)



Embedded Real-Time Systems

27

Filtering in digital signal processing



Embedded Real-Time Systems

28

DSP-Processors

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

MR:=0; A1:=1; A2:=s-1; MX:=w[s]; MY:=a[0];

for (k:=0 <= n-1)

{MR:=MR+MX*MY; MY:=a[A1]; MX:=w[A2]; A1++; A2--}

Multiply/accumulate (MAC) instruction

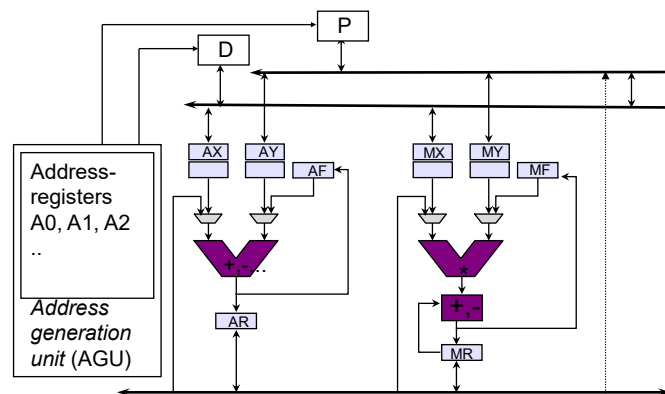
Zero-overhead loop (ZOL) instruction preceding MAC instruction.
Loop testing done in parallel to MAC operations.

Embedded Real-Time Systems

29

Heterogeneous registers

Example (ADSP 210x):



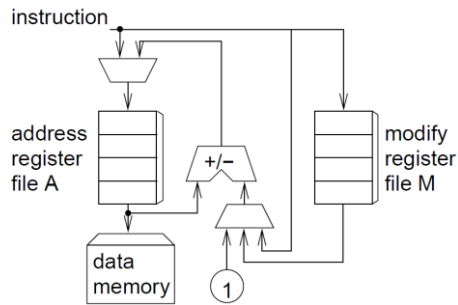
Different functionality of registers An, AX, AY, AF, MX, MY, MF, MR

Embedded Real-Time Systems

30

Separate address generation units (AGUs)

Example (ADSP 210x):



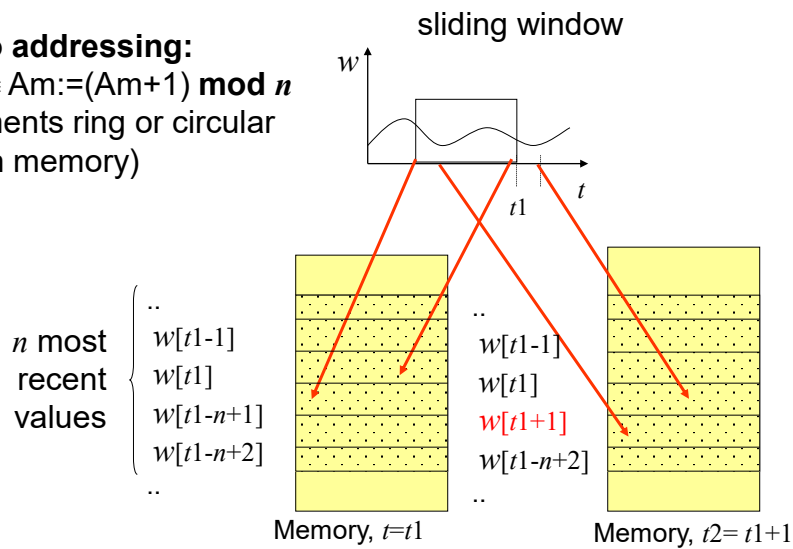
- Data memory can only be fetched with address contained in A,
 - but this can be done in parallel with operation in main data path (takes effectively 0 time).
 - $A := A \pm 1$ also takes 0 time,
 - same for $A := A \pm M$;
 - $A := \langle \text{immediate in instruction} \rangle$ requires extra instruction
- ☞ Minimize load immediates
- ☞ Optimization comes later

Embedded Real-Time Systems

31

Modulo addressing

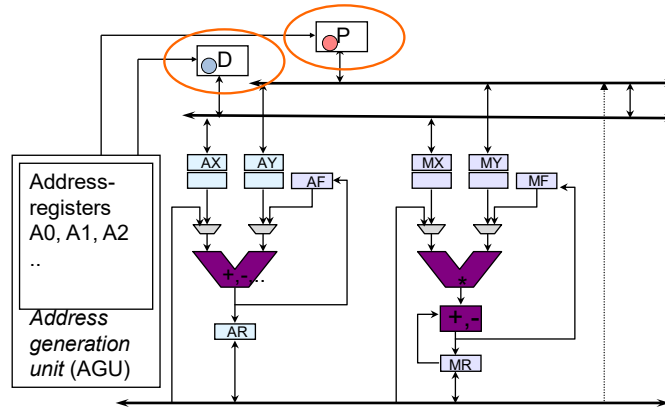
Modulo addressing:
 $A_{m++} \equiv A_m := (A_m + 1) \bmod n$
 (implements ring or circular buffer in memory)



Embedded Real-Time Systems

32

Multiple memory banks or memories



Simplifies parallel fetches

Embedded Real-Time Systems

33

Saturating arithmetic

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

- Example:

a	0111
b	+ 1001
<hr/>	
standard wrap around arithmetic	(1)0000
saturating arithmetic	1111
<hr/>	
(a+b)/2: correct	1000
wrap around arithmetic	0000
saturating arithmetic + shifted	0111

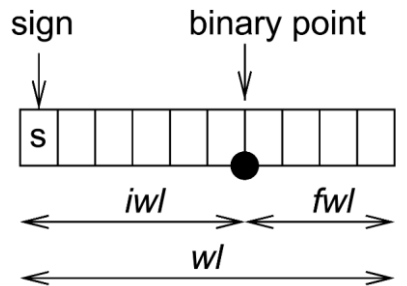
"almost correct"

- 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.

Embedded Real-Time Systems

34

Fixed-point arithmetic



Shifting required after multiplications and divisions in order to maintain binary point.

Embedded Real-Time Systems

35

Real-time capability

- **Timing behavior has to be predictable**
Features that cause problems:
 - Unpredictable access to shared resources
 - Caches with difficult to predict replacement strategies
 - Unified caches (conflicts between instructions and data)
 - Pipelines with difficult to predict stall cycles ("bubbles")
 - Unpredictable communication times for multiprocessors
 - Branch prediction, speculative execution
 - Interrupts that are possible any time
 - Memory refreshes that are possible any time
 - Instructions that have data-dependent execution times
- ☞ Trying to avoid as many of these as possible.

[Dagstuhl workshop on predictability, Nov. 17-19, 2003]

Embedded Real-Time Systems

36

Embedded Processors for Safety-Critical Real-Time Applications

- High requirements in terms of timing predictability
 - Lower and upper bounds on task execution times
 - Called BCET and WCET
 - Must be *safe* and *tight*
- Threats to predictability
 - Architectural features
 - Software
 - Task-level
 - Distributed operation
 - Cross-layer

Embedded Real-Time Systems

37

Microcontrollers Example: Intel 8051

- 8-bit CPU, optimized for control applications,
- large set of operations on Boolean data types,
- program address space of 64 k bytes,
- separate data address space of 64 k bytes,
- 4 k bytes of program memory on chip, 128 bytes of data memory on chip,
- 32 I/O lines, each of which can be addressed individually,
- 2 counters on the chip,
- universal asynchronous receiver/transmitter for serial lines available on the chip,
- clock generation on the chip,
- many variations commercially available.

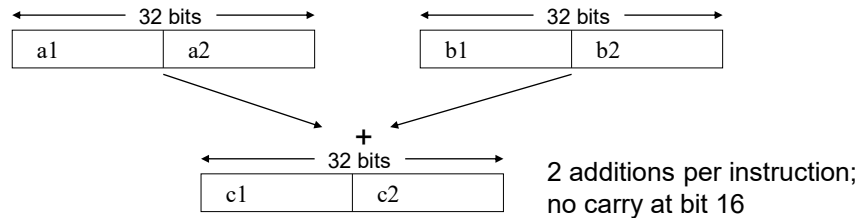
Embedded Real-Time Systems

38

Multimedia-Instructions, Short vector extensions, Streaming extensions, SIMD instructions

- Multimedia instructions exploit that many registers, adders etc. are quite wide (32/64 bit), whereas most multimedia data types are narrow

☞ 2-8 values can be stored per register and added. E.g.:



- Cheap way of using parallelism
- ☞ SSE instruction set extensions, SIMD instructions

Embedded Real-Time Systems

39

Single ISA Heterogeneous Multi-core Processors ARM's big.LITTLE as an example

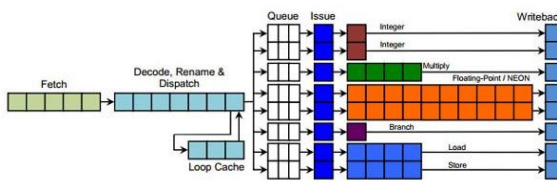


Figure 2 Cortex-A15 Pipeline

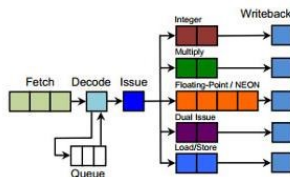
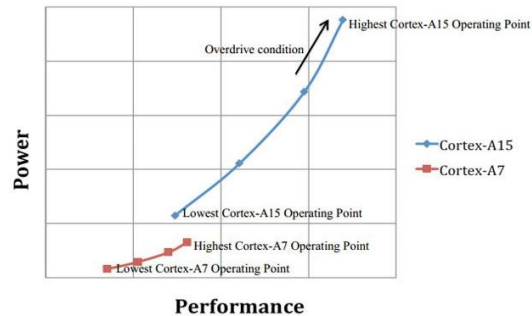


Figure 1 Cortex-A7 Pipeline

Used in
Samsung S4



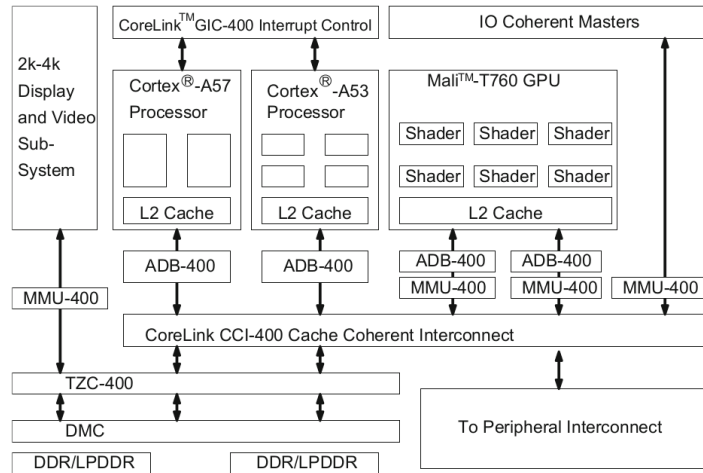
Embedded Real-Time Systems

© ARM, 2013

40

Multiprocessor Systems-on-a-Chip (MPSoCs)

ARM® big.LITTLE System on Chip (SoC)

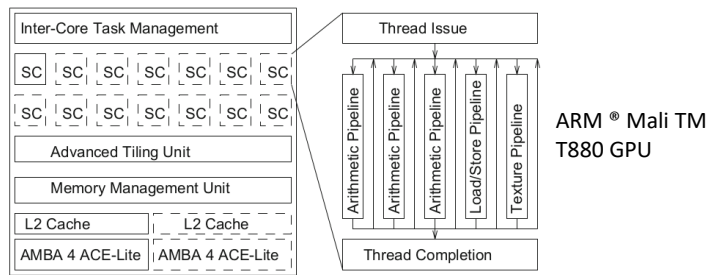


Embedded Real-Time Systems

41

Graphics Processing Units (GPU)

- Programmable GPUs
- Run many fine-grained threads at the same time
- Power efficiency important in embedded systems
- Interface to OpenGL, OpenCL, etc.



Embedded Real-Time Systems

42

ARM's Neural Processing Units

- Object classification
- Object detection
- Face detection/identification
- Human pose detection/hand-gesture recognition
- Image segmentation
- Image beautification
- Super resolution
- Framerate adjustment (super slow-mo)
- Speech recognition
- Sound recognition
- Noise cancellation
- Speech synthesis
- Language translation

Key Features		Ethos-N78	Ethos-N77	Ethos-N57	Ethos-N37
	Performance	10, 5, 2, 1 TOP/s	4 TOP/s	2 TOP/s	1 TOP/s
	MAC/Cycle (8x8)	40x6, 20x8, 10x4, 5x2	20x8	10x4	5x2
	Efficient convolution	Winograd support delivers 2.25x peak performance over baseline			
	Configurability	90+ Design Options	Single Product Offering		
	Network support	CNN and RNN			
Memory System	Data types	Int-8 and Int-16			
	Secure mode	TEE or SEE			
	Multicore capability	8 NPUs in a cluster 64 NPUs in a mesh			
	Embedded SRAM	384 KB – 4 MB	1–4 MB	512 KB	512 KB
	Bandwidth reduction	Enhanced Compression	Extended compression technology, layer/operator fusion, clustering, and workload tiling		
	Main interface	1xAXI4 (128-bit), ACE-5 Lite			
Development Platform	Neural frameworks	TensorFlow, TensorFlow Lite, Caffe2, PyTorch, MXNet, ONNX			
	Inference deployment	Ahead of time compiled with TVM Online interpreted with Arm NN Android Neural Networks API (NNAPI)			
	Software components	Arm NN, Arm NPU software (compiler and support library, driver)			
	Debug and profile	Heterogeneous layer-by-layer visibility in Development Studio 5 Streamline			
	Evaluation and early prototyping	Ethos-N Static Performance Analyzer (SPA), Arm Juno FPGA systems, Cycle Models			

Embedded Real-Time Systems

43

Arm's ML processor architecture key features

Efficient convolutions

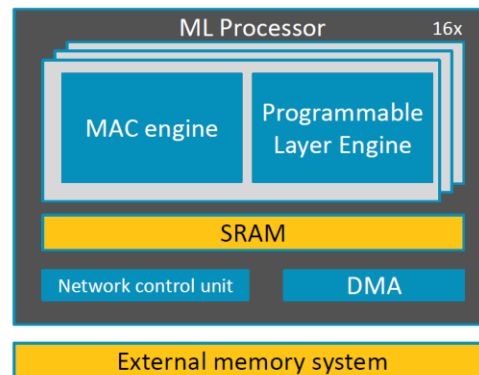
- Convolutions represent the bulk of computation
- We provide dedicated 8-bit hardware for convolutions

Efficient data movement

- More energy is spent moving data than computing
- We amortize activation accesses and compress weights

Sufficient programmability

- New operators are invented and topology is changed frequently
- We provide programmability to future-proofed as new network architectures appear

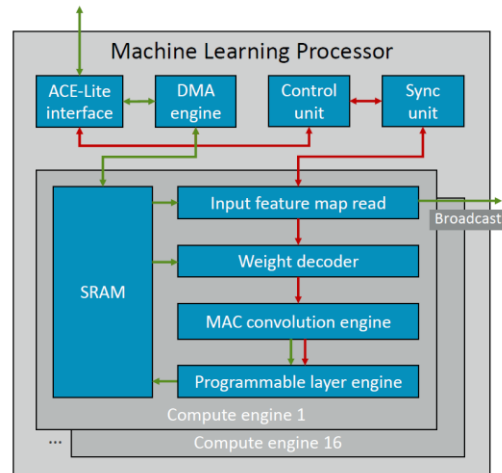


Embedded Real-Time Systems

44

Arm's ML processor

- A microcontroller and DMA engine manage overall network scheduling
- The compute engine processes major sections of the neural network
 - Stores weights
 - Stores and manipulates activation data
 - Handles convolution in 128-wide MAC units
 - Handles other layer operators via PLE
 - Pipelines data to and from SRAM
- Internal broadcast network manages SRAM population and synchronization



Embedded Real-Time Systems

45

Reconfigurable Logic

- Fast prototyping
- Low-volume applications
- Real-time systems
- High level of parallel processing
- Field programmable gate arrays (FPGAs) are the most common (Xilinx, Intel, Lattice, etc.)
 - Configurable logic
 - Memory
 - IO
 - Hard cores

Embedded Real-Time Systems

46

