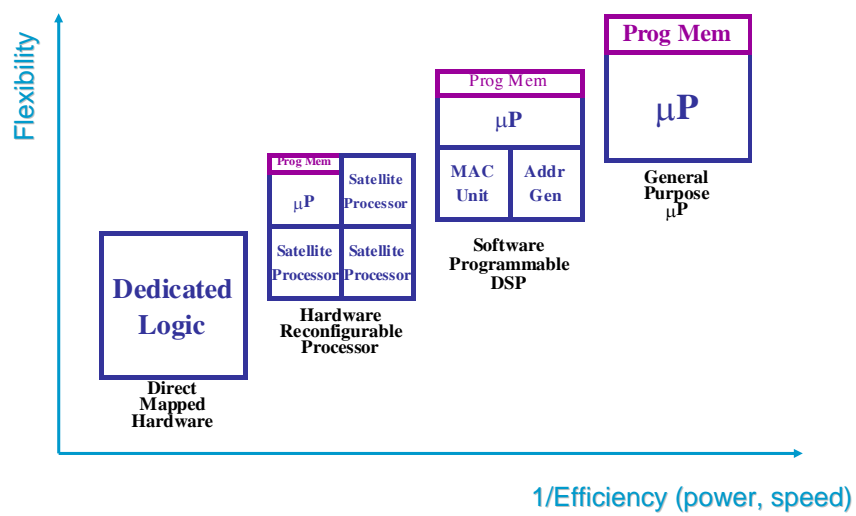


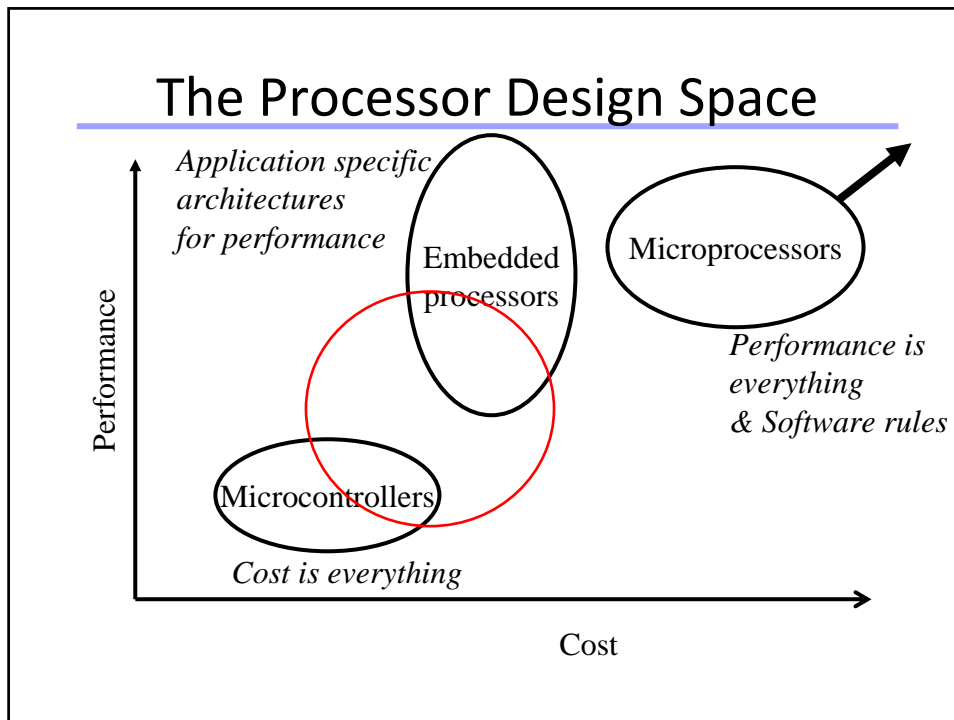
Hardware Platforms for Embedded Computing



Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

Architectural Choices

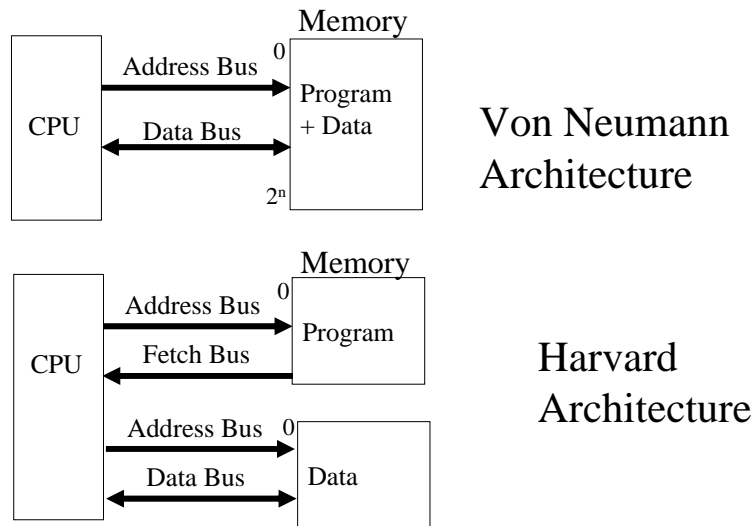




Embedded vs. general-purpose processors

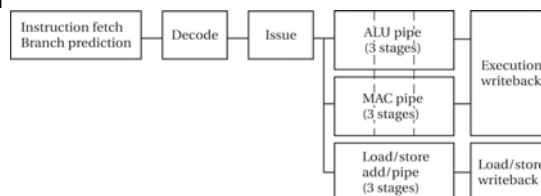
- Embedded processors may be optimized for a category of applications.
 - Customization may be narrow or broad.
- We may judge embedded processors using different metrics:
 - Code size.
 - Memory system performance.
 - Preditability.

Microcontroller Architectures



RISC processors

- RISC generally means highly-pipelined, one instruction per cycle.
- Pipelines of embedded RISC processors have grown over time:
 - ARM7 has 3-stage pipeline.
 - ARM9 has 5-stage pipeline.
 - ARM11 has eight-stage pipeline.

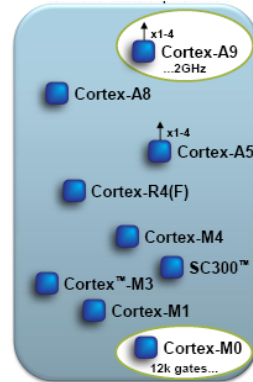


ARM11 pipeline [ARM05].

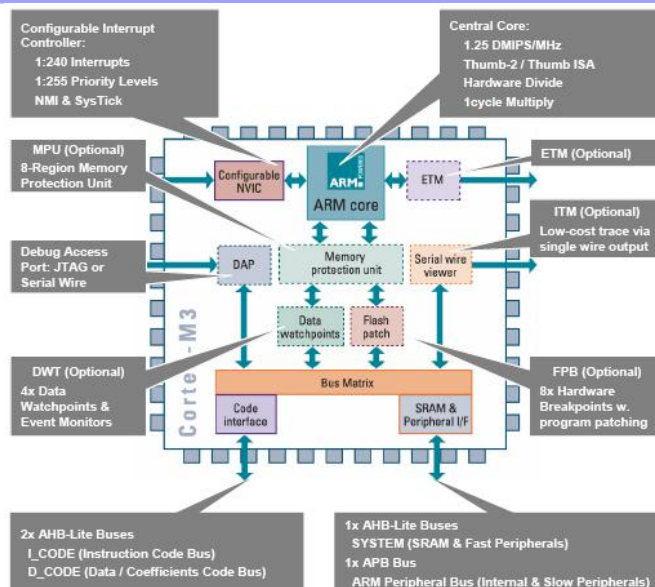
ARM Cortex

Based on ARMv7 Architecture & Thumb[®]-2 ISA

- **ARM Cortex A Series - Applications** CPUs focused on the execution of complex OS and user applications
 - First Product: Cortex-A8
 - Executes ARM, Thumb-2 instructions
- **ARM Cortex R Series** - Deeply embedded processors focused on **Real-time** environments
 - First Product: Cortex-R4(F)
 - Executes ARM, Thumb-2 instructions
- **ARM Cortex M Series - Microcontroller cores** focused on very cost sensitive, deterministic, interrupt driven environments
 - First Product: ARM Cortex-M3 (2uA, 0.5mW/MHz)
 - Executes Thumb-2 instructions



Cortex-M3 “Processor”



Central Core

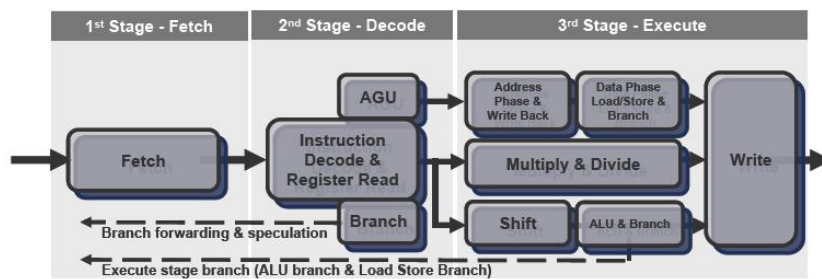
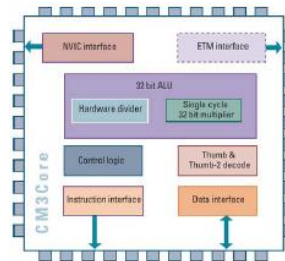
Harvard architecture

Separate Instruction & Data buses
enable parallel fetch & store

Advanced 3-Stage Pipeline

Includes Branch Forwarding &
Speculation

Additional Write-Back via Bus Matrix



DSP Applications

- Audio applications
- MPEG Audio
- Portable audio
- Digital cameras
- Wireless
- Cellular telephones
- Base station
- Networking
- Cable modems
- ADSL
- VDSL

DSP vs. General Purpose MPU

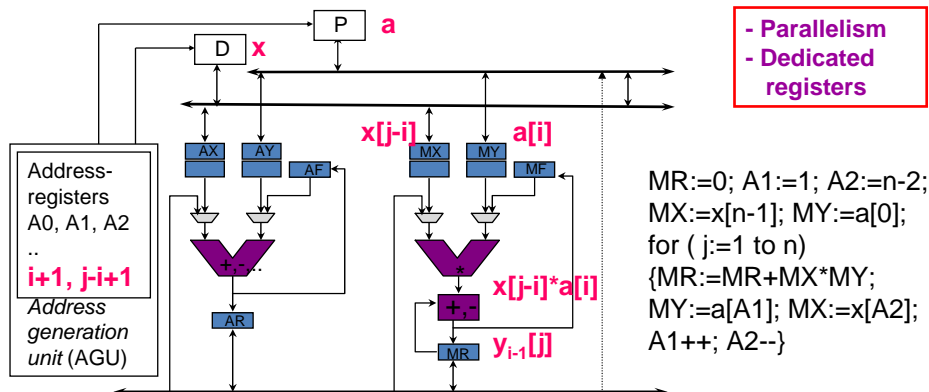
- The “MIPS/MFLOPS” of DSPs is speed of Multiply-Accumulate (MAC).
 - DSP are judged by whether they can keep the multipliers busy 100% of the time.
- The "SPEC" of DSPs is 4 algorithms:
 - Infinite Impulse Response (IIR) filters
 - Finite Impulse Response (FIR) filters
 - FFT, and
 - convolvers
- In DSPs, algorithms are king!
 - Binary compatibility not an issue
- Software is not (yet) king in DSPs.
 - People still write in assembly language for a product to minimize the die area for ROM in the DSP chip.

Domain-oriented architectures

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

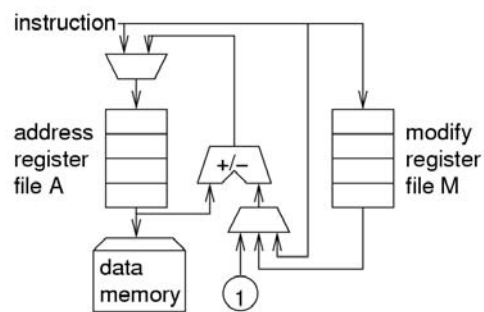
$\forall i: 0 \leq i \leq n-1: y_i[j] = y_{i-1}[j] + x[j-i]*a[i]$

Architecture: Example: Data path ADSP210x



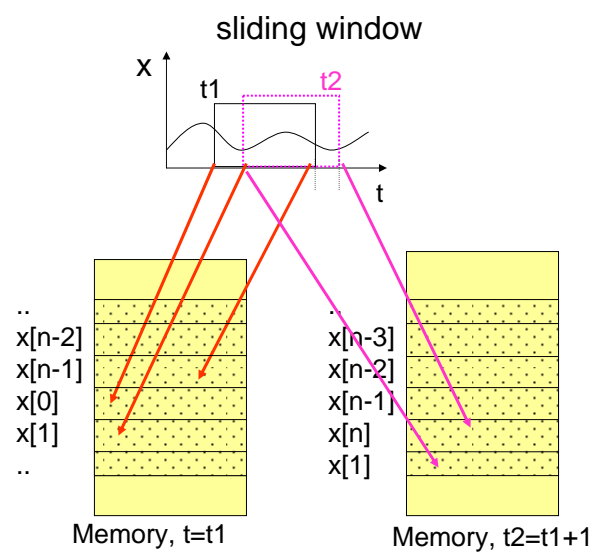
DSP - Features (1)

- **Multiply/accumulate (MAC) and zero-overhead loop (ZOL) instructions** (as shown)
- **Heterogeneous registers** (as shown)
- **Separate address generation units (AGUs)** (as in ADSP 210x)

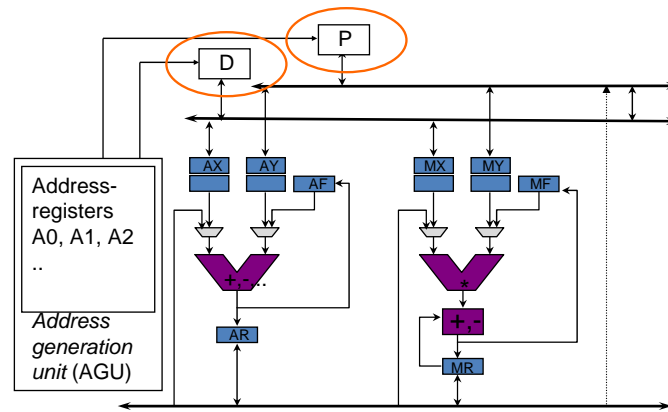


DSP - Features (2)

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



Multiple memory banks or memories

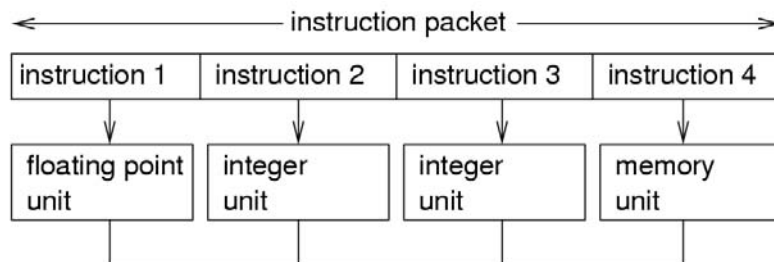


Simplifies parallel fetches

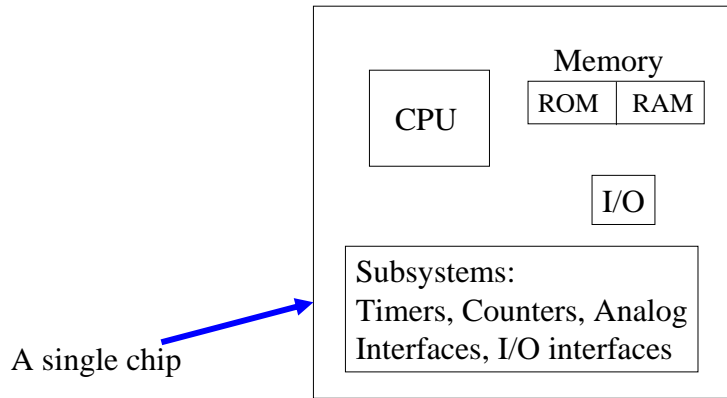
Very long instruction word (VLIW) processors

Key idea: detection of possible parallelism to be done by compiler, not by hardware at run-time (inefficient).

VLIW: parallel operations (instructions) encoded in one long word (instruction packet), each instruction controlling one functional unit. E.g.:

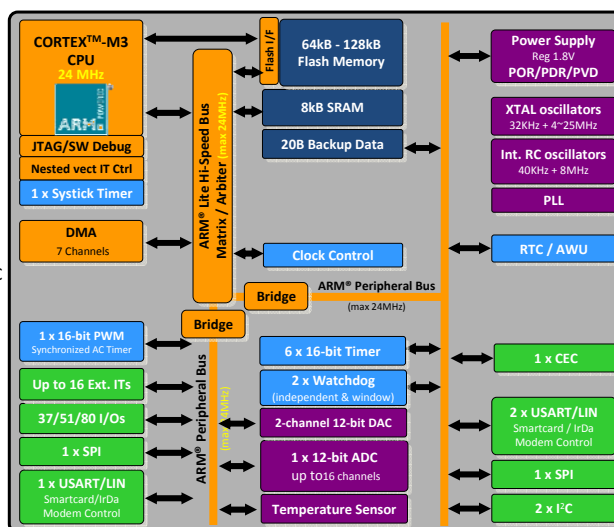


Microcontrollers



A Microcontroller SOC example: STM32 Value line 64K-128KBytes System Diagram

- Core and operating conditions**
 - ARM® Cortex™-M3
 - 1.25 DMIPS/MHz up to 24 MHz
 - 2.0 V to 3.6 V range
 - -40 to +105 °C
- Rich connectivity**
 - 8 communications peripherals
- Advanced analog**
 - 12-bit 1.2 μs conversion time ADC
 - Dual channel 12-bit DAC
- Enhanced control**
 - 16-bit motor control timer
 - 6x 16-bit PWM timers
- LQFP48, LQFP/BGA64, LQFP100**

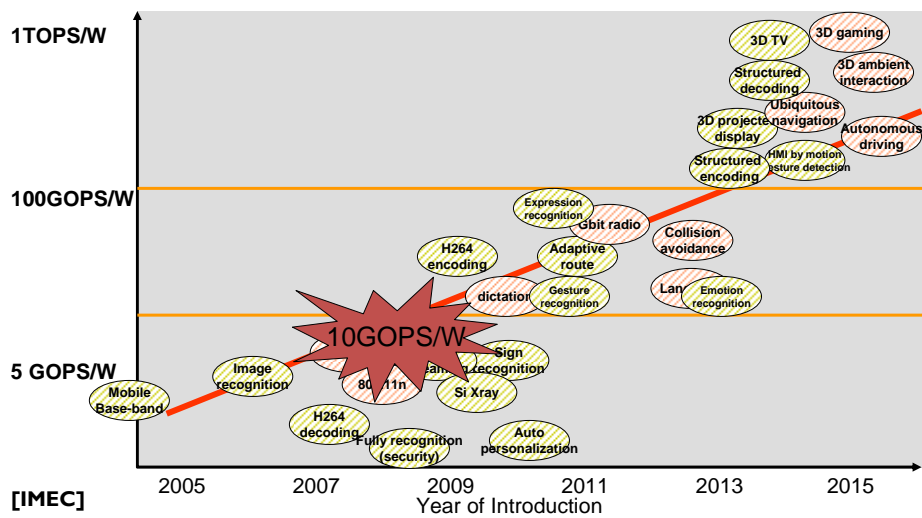


Multi-processors SoCs for Embedded Computing



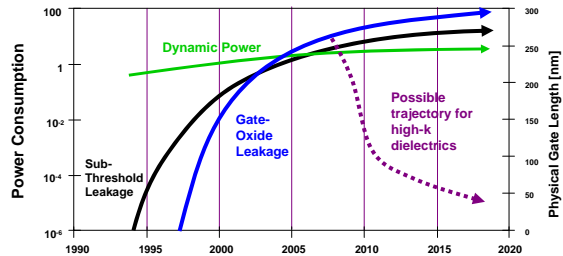
Graphics: © Alexandra Nolte, Gesine Manwedel, 2003

Application pull

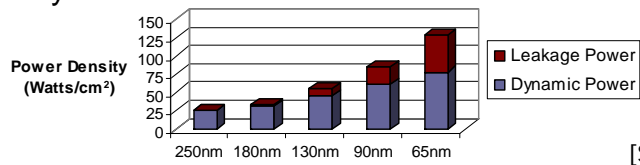


Power Bottleneck

- Power trend

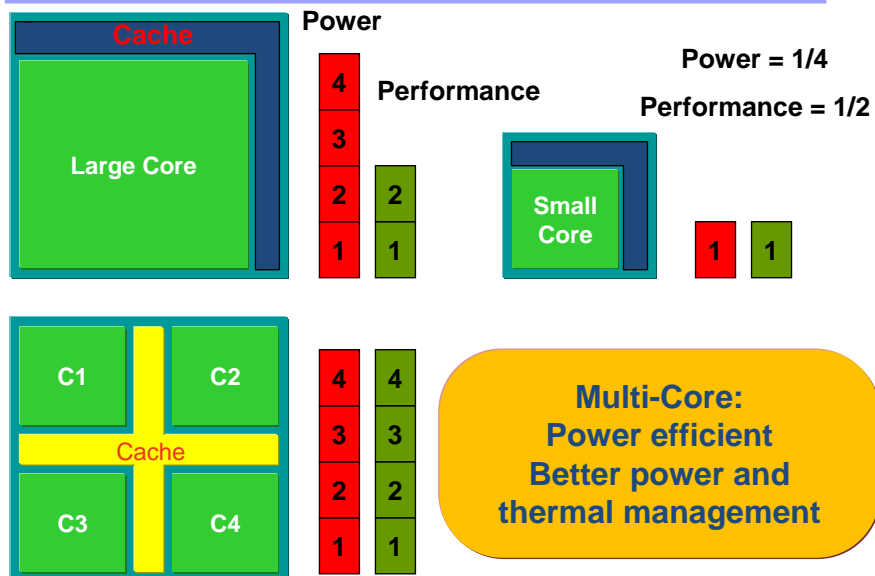


Power density trend



[STM ASIC]

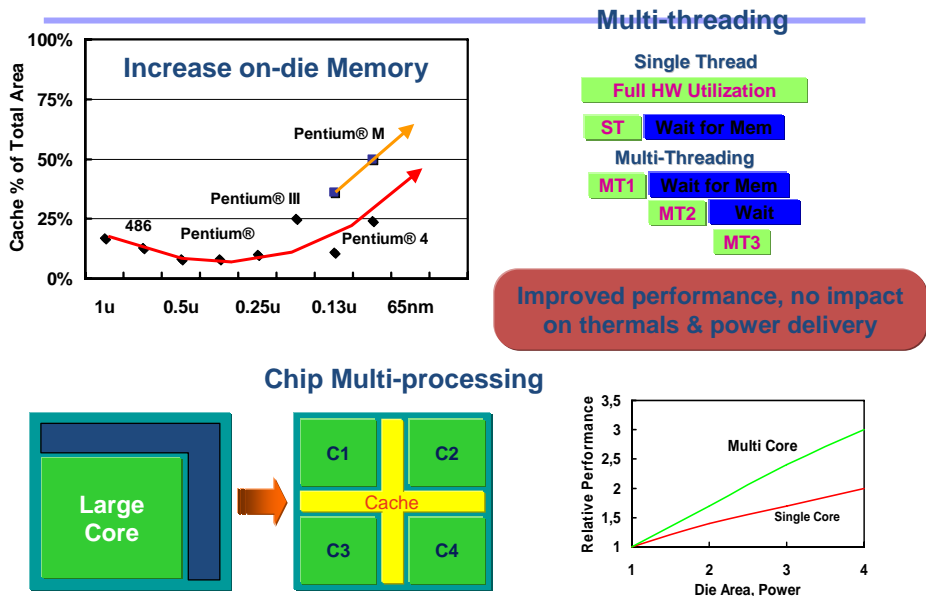
Multi-Core & Power



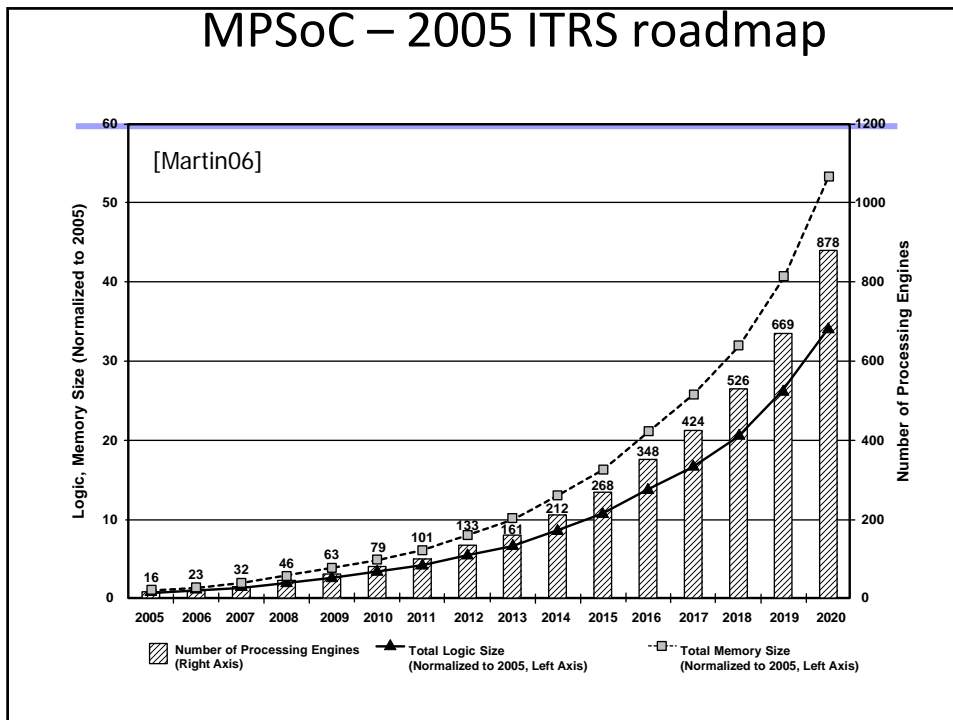
Near Term Solutions

- Move away from Frequency alone to deliver performance
- More on-die memory
- Multi-everywhere
 - Multi-threading
 - Chip level multi-processing
- Throughput oriented designs
- Performance by higher level of integration

μArchitecture Techniques



MPSoC – 2005 ITRS roadmap



Embedded vs. General Purpose

Embedded Applications

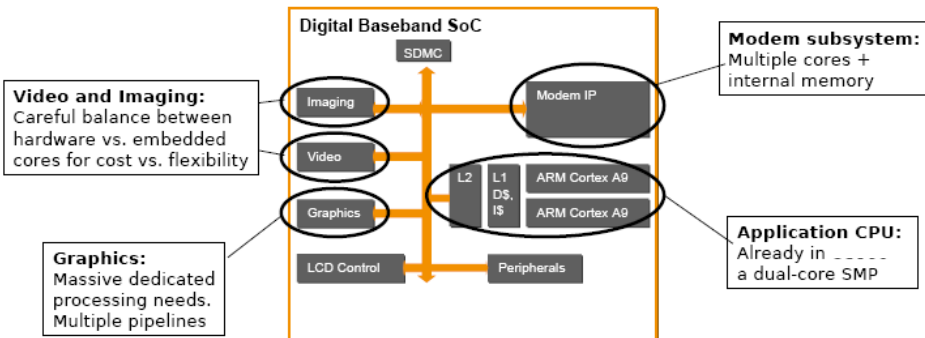
- *Asymmetric Multi-Processing*
 - Differentiated Processors
- Specific tasks known early
 - Mapped to dedicated processors
- Configurable and extensible processors: performance, power efficiency
- Communication
 - Coherent memory
 - Shared local memories
 - HW FIFOs, other direct connections
- Dataflow programming models
- Classical example – Smart mobile – RISC + DSP + Media processors

Server Applications

- *Symmetric Multi-Processing*
 - Homogeneous cores
- General tasks known late
 - Tasks run on any core
- High-performance, high-speed microprocessors
- Communication
 - large coherent memory space on multi-core die or bus
- SMT programming models (Simultaneous Multi-Threading)
- Examples: large server chips (eg Sun Niagara 8x4 threads), scientific multi-processors

Integrated SoC Mobile

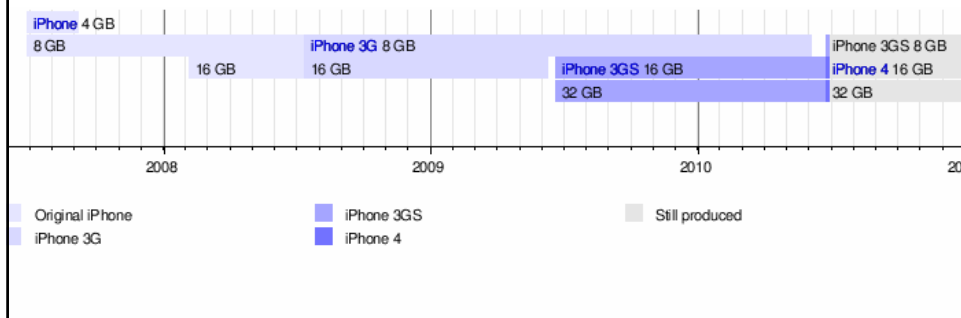
- High-speed SMP for “almost sequential” GP
- “Processor arrays” for domain-specific throughput computing (100x GOPS/W)



27

Apple platforms

- iPhone “2G”/3G/3GS
- iPhone 4
- iPad



iPhone "2G"/3G/3GS

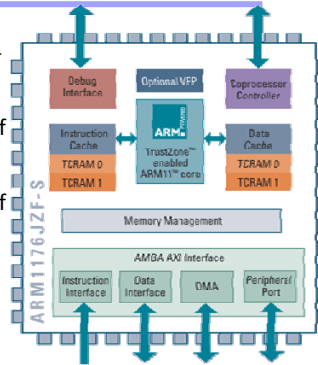
	iPhone 3GS	iPhone 3G	iPhone 2G	iPod Touch
App Store, multitouch, accelerometer	Yes	Yes	Yes	Yes
cellular data, camera	Yes	Yes	Yes	
GPS, 3G data	Yes	Yes		
video, autofocus, compass, more speed & RAM	Yes			

ARM based CPU

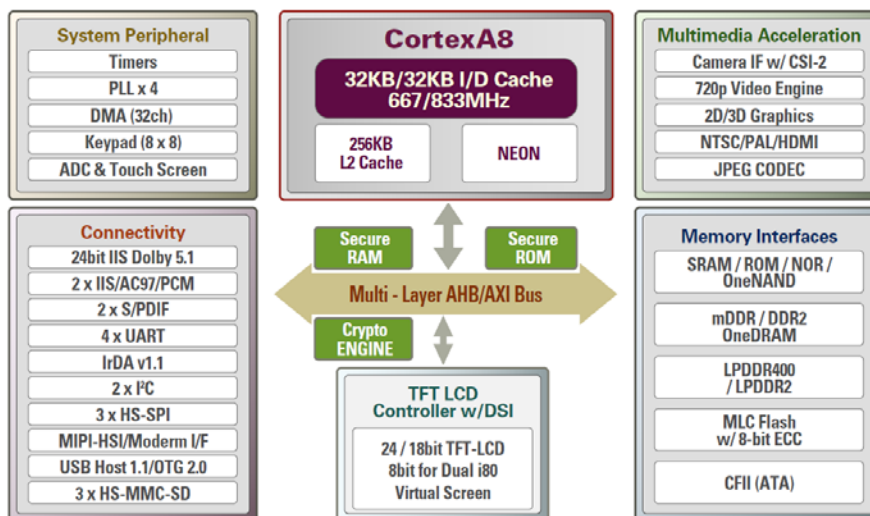
- The iPhone uses the ARM CPU architecture (Samsung)
- The iPhone "2G"/3G
 - ARM 11 (1176JZ(F)-S), (Jazelle)
 - 412MHz (Underclocked from 620MHz)
 - 16KB L1 cache
 - SIMD, 8 Stage Pipeline
 - 90nm manufacturing process
- iPhone 3GS
 - ARM Cortex-A8
 - 600MHz (Underclocked from 833MHz)
 - 32KB L1 cache
 - 13 Stage Pipeline
 - 65nm manufacturing process

The iPhone "2G"/3G processor

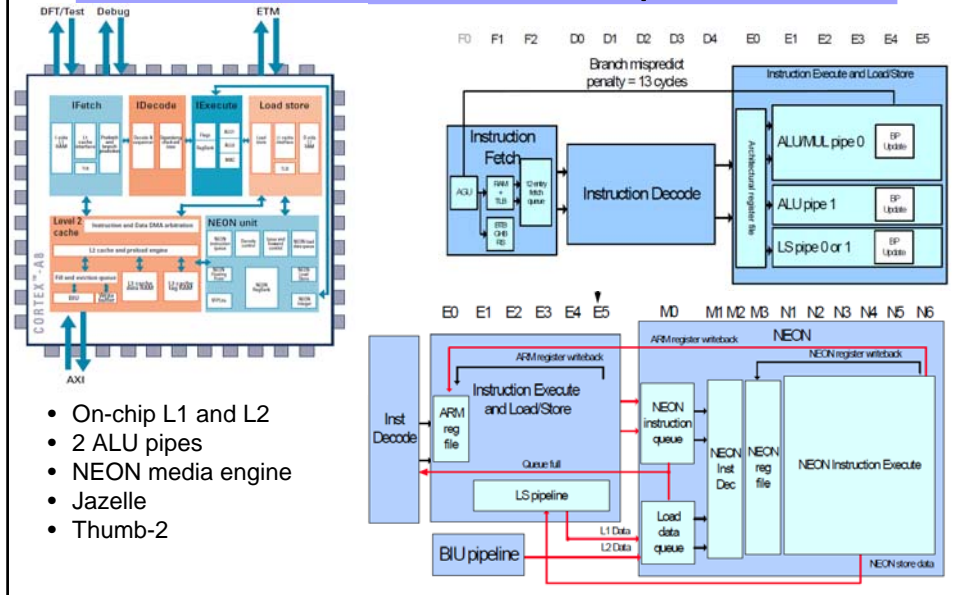
- The Samsung chipset at the heart of the iPhone utilizes a 32-bit RISC ARM processing core, the ARM1176JZ(F)-S v1.0.
- The ARM device is capable of running at 620MHz, but Apple has downclocked it to 412MHz, presumably in the interest of extending battery life.
- The ARM device is capable of running at 620MHz, but Apple has downclocked it to 412MHz, presumably in the interest of extending battery life.
- Some notable features of the ARM processor in the iPhone:
 - High performance integer processor
 - 8-stage integer pipeline delivers high clock frequency
 - Separate load-store and arithmetic pipelines
 - Branch Prediction and Return Stack
 - Up to 675 Dhrystone 2.1 MIPS in 0.13u process
 - High performance memory system
 - Supports 4-64k cache sizes
 - Optional tightly coupled memories with DMA for multi-media applications
 - Quad-ported AMBA 3 AXI bus interface speeds instruction and data access
 - ARMv6 memory system architecture accelerates OS context-switch
 - Vector Floating Point coprocessor for powerful acceleration of embedded 3D-graphics



iPhone 3GS: Samsung System-on-Chip



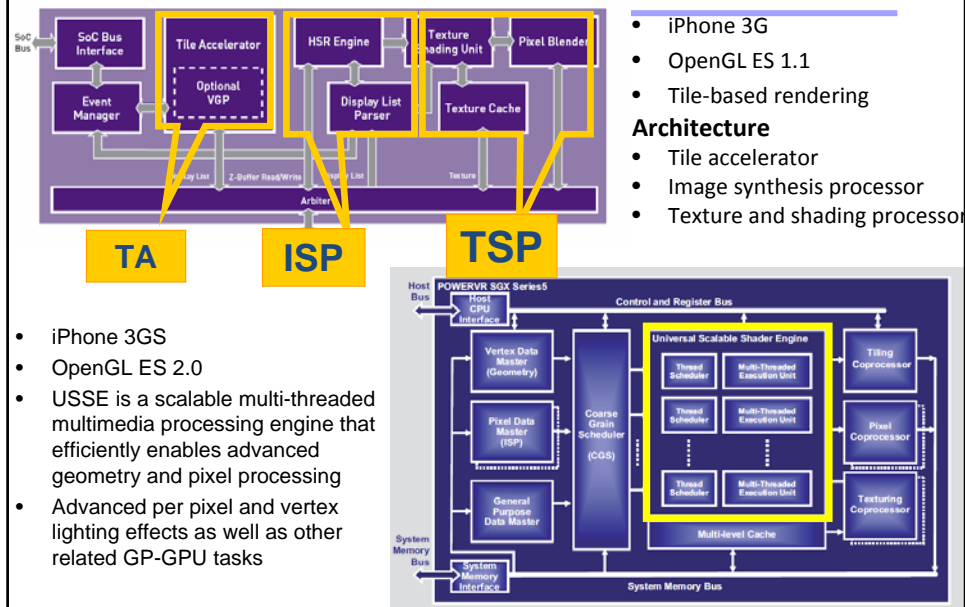
The CORTEX-A8 Microprocessor



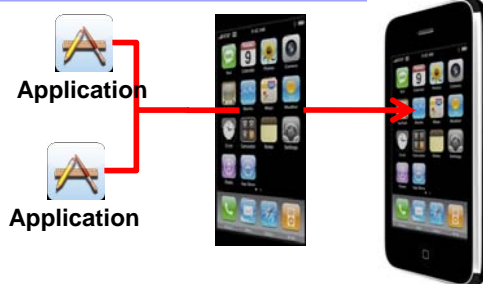
Memory hierarchy

- Main System Memory
 - The iPhone "2G"/3G have 128MB of main working memory (RAM).
 - The iPhone 3GS have 256MB of RAM.
- Storage Flash Memory
 - The iPhones mainly contains from 4GB to 64GB of NAND flash storage memory.
 - Apple decided to leave out a flash memory card slot and force users to choose iPhone models based on fixed amount of non-removeable flash memory.
- Firmware Flash Memory (from 4MB to 16MB)
 - In addition to regular storage flash memory, the iPhone also has additional NOR flash memory to store bootup code (similar to BIOS in PC).
 - This data usually holds the minimum operating system for booting up the device. After booting up it lets other instructions stored in the storage flash memory to take over.

PVR 3D – MBX vs GTX5

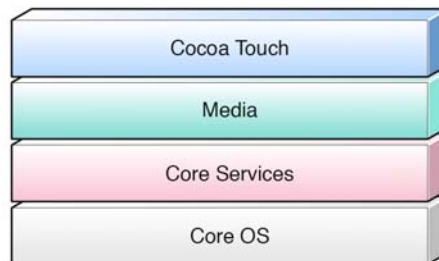


The iPhone OS Architecture



- At the high level, iPhone OS acts as an **intermediary** between the iPhone and iPod touch hardware and the applications that appear on the screen.
- Applications that you create never interact directly with the hardware but instead go through system interfaces, which interact with the appropriate drivers.
- This abstraction protects your application from changes to the underlying hardware.
 - Even though your application is generally protected from changes to the underlying hardware, you still need to account for differences between iPhone and iPod touch devices in your code.

Layers of iPhone OS



- The higher-level frameworks are there to provide object-oriented abstractions for lower-level constructs.
 - They make it much easier to write code
 - They reduce the number of lines of code you have to write and encapsulate potentially complex features, such as sockets and threads.
- Although they abstract out lower-level technologies, they do not mask those technologies from you.
- The lower-level frameworks are still available for developers who prefer using them or who want to use aspects of those frameworks that are not exposed at the higher level.

GP-GPUs



Graphics Processing Units (GPUs)

- Original GPUs: fixed-function devices for generating 3D graphics (mid-late 1990s) including high-performance floating-point units
 - Provide workstation-like graphics for PCs
 - User could configure graphics pipeline, but not really program it
- More programmability added (2001-2005)
 - E.g., New language Cg for writing small programs run on each vertex or each pixel, also Windows DirectX variants
 - Massively parallel (millions of vertices or pixels per frame) but very constrained programming model
- Some users noticed they could do general-purpose computation by mapping input and output data to images, and computation to vertex and pixel shading computations
 - Incredibly difficult programming model as had to use graphics pipeline model for general computation

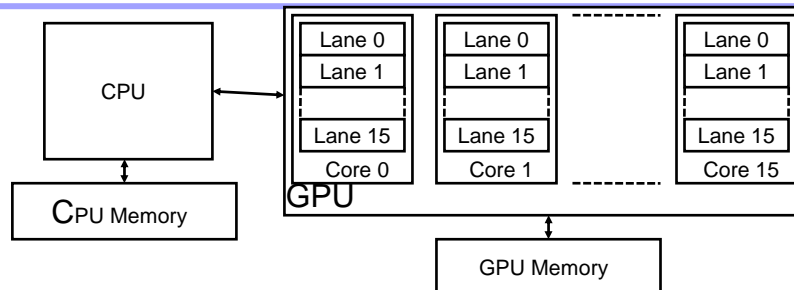
39

General-Purpose GPUs (GP-GPUs)

- In 2006, Nvidia introduced GeForce 8800 GPU supporting a new programming language: CUDA
 - “Compute Unified Device Architecture”
 - Subsequently, broader industry pushing for OpenCL, a vendor-neutral version of same ideas.
- Idea: Take advantage of GPU computational performance and memory bandwidth to accelerate some kernels for general-purpose computing
- Attached processor model: Host CPU issues data-parallel kernels to GP-GPU for execution
- This lecture has a simplified version of Nvidia CUDA-style model and only considers GPU execution for computational kernels, not graphics
 - Would probably need another course to describe graphics processing

40

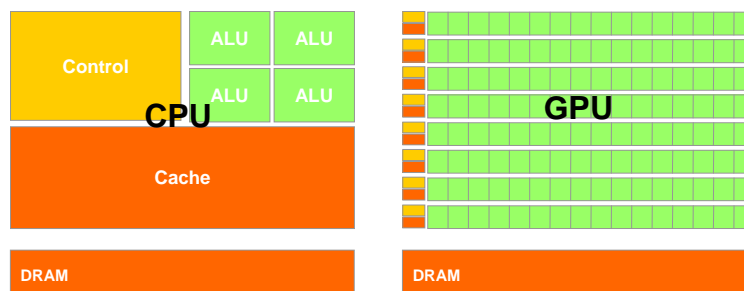
Hardware Execution Model



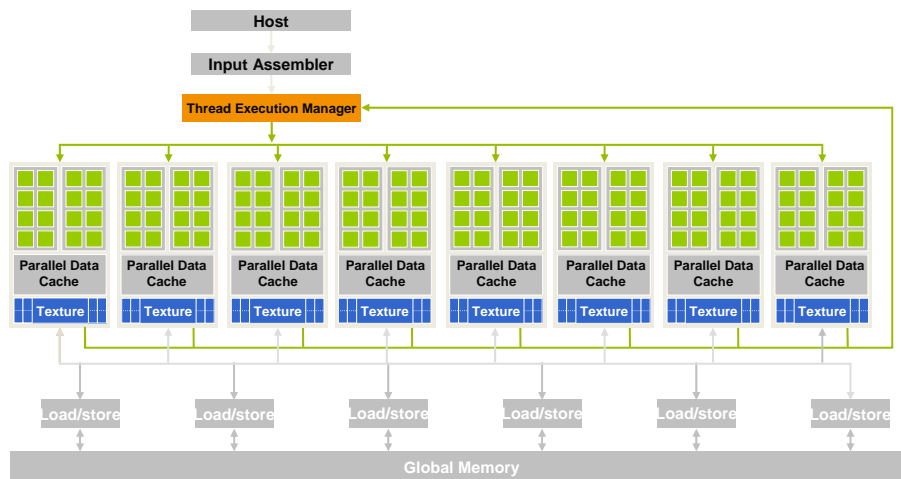
- GPU is built from multiple parallel cores, each core contains a multithreaded SIMD processor with multiple lanes but with no scalar processor
- CPU sends whole “grid” over to GPU, which distributes thread blocks among cores (each thread block executes on one core)
 - Programmer unaware of number of cores

41

CPUs vs GPUs



Architecture of a CUDA-capable GPU



© David Kirk/NVIDIA and
Wen-mei W. Hwu, 2007-
2010
ECE 408, University of

43

Simplified CUDA Programming Model

- Computation performed by a very large number of independent small scalar threads (*CUDA threads* or *microthreads*) grouped into *thread blocks*.

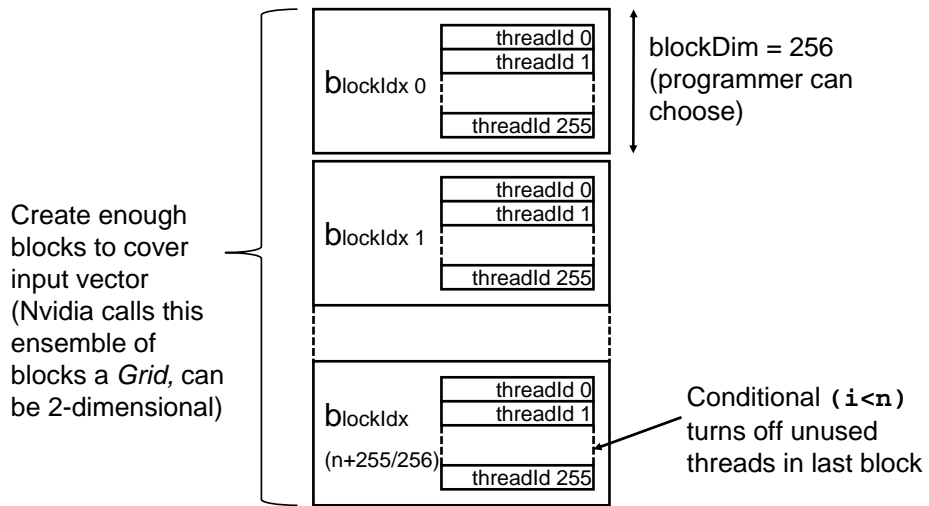
```
// C version of DAXPY loop.  
void daxpy(int n, double a, double*x, double*y)  
{ for (int i=0; i<n; i++)  
  y[i] = a*x[i] + y[i]; }
```

```
// CUDA version.  
__host__ // Piece run on host processor.  
int nblocks = (n+255)/256; // 256 CUDA threads/block  
daxpy<<nblocks,256>>(n,2.0,x,y);
```

```
__device__ // Piece run on GP-GPU.  
void daxpy(int n, double a, double*x, double*y)  
{ int i = blockIdx.x*blockDim.x + threadIdx.x;  
  if (i<n) y[i]=a*x[i]+y[i]; }
```

44

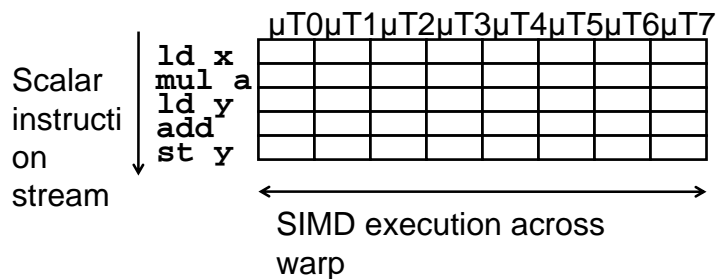
Programmer's View of Execution



45

"Single Instruction, Multiple Thread"

- GPUs use a SIMT model, where individual scalar instruction streams for each CUDA thread are grouped together for SIMD execution on hardware (Nvidia groups 32 CUDA threads into a *warp*)



46

NVIDIA Roadmap

