

Lecture #2: Embedded Systems Architecture

CSE594
Jennifer Wong
9/9/08

New Definition: Embedded Systems

It's a world where *embedded systems* operate using an *optimized architecture* designed to operate in a *specific environment* to solve a particular application running a *specific and optimized software algorithm*.

-Dario Gonano

Design Metrics

- NRE Cost (Non-recurring Engineering Cost)
 - 1 time \$ cost of designing system
- Unit Cost
- Size, Performance, Power
- Flexibility
 - Change to the hardware with low NRE cost
- Time-to-prototype, Time-to-market
- Maintainability
- Correctness & Safety

Always in competition Trade-offs!

NRE and Unit Cost Metrics

- Costs:
 - Unit cost: the monetary cost of manufacturing each copy of the system, excluding NRE cost
 - NRE cost (Non-Recurring Engineering cost): the one-time monetary cost of designing the system
 - $total\ cost = NRE\ cost + unit\ cost * \#\ of\ units$
 - $per-product\ cost = total\ cost / \#\ of\ units = (NRE\ cost / \#\ of\ units) + unit\ cost$
 - Example
 - NRE=\$2000, unit=\$100
 - For 10 units
 - $total\ cost = \$2000 + 10 * \$100 = \$3000$
 - $per-product\ cost = \$2000/10 + \$100 = \$300$
- Amortizing NRE cost over the units results in an additional \$200 per unit*

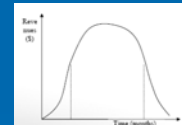
NRE and Unit Cost Metrics

- Compare technologies by costs -- best depends on quantity
 - Technology A: NRE=\$2,000, unit=\$100
 - Technology B: NRE=\$30,000, unit=\$30
 - Technology C: NRE=\$100,000, unit=\$2
- For 100 copies: a=\$120, b=\$330, c=\$1002
- For 1,000,000 copies: a=\$100, b=\$30.03, c=\$2.1

But, must also consider time-to-market

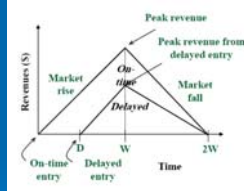
Time-to-Market

- Time required to develop a product to the point it can be delivered to a customer
- Market window
- Period during which the product would reach highest sales
- Average time-to-market constraint is about 8 months
- Project delays can be costly

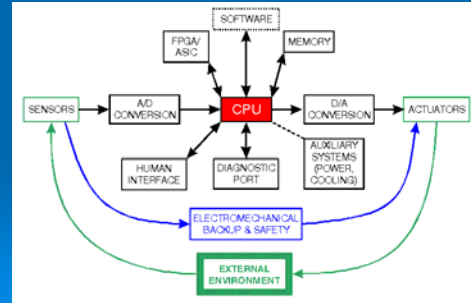


Delayed Market Entry

- Simplified revenue model
 - Product life = $2W$, peak at W
 - Time of market entry defines a triangle, representing market penetration
 - Triangle area equals revenue
- Loss
 - The difference between on-time and delayed triangle areas



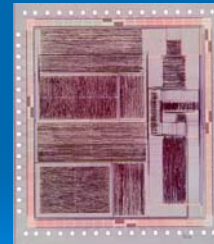
“Traditional” Software Embedded Systems = CPU + RTOS



Generic Architecture



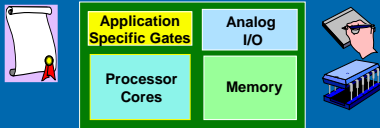
“Traditional” Hardware Embedded Systems = ASIC



ASIC Features

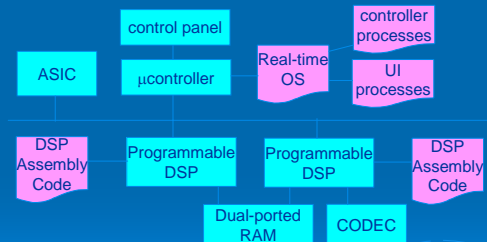
- Area: 4.6 mm x 5.1 mm
- Speed: 20 MHz @ 10 Mcps
- Technology: HP 0.5 μm
- Power: 16 mW - 120 mW (mode dependent) @ 20 MHz, 3.3 V
- Avg. Acquisition Time: 10 μs to 300 μs

Modern Embedded Systems?



- Embedded systems employ a combination of
 - application-specific h/w (boards, ASICs, FPGAs etc.)
 - performance, low power
 - s/w on prog. processors: DSPs, μ controllers etc.
 - flexibility, complexity
 - mechanical transducers and actuators

Complexity and Heterogeneity



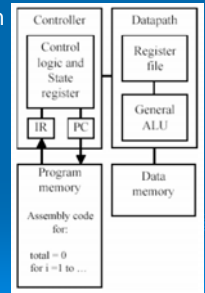
- Heterogeneity within H/W & S/W parts as well
 - S/W: control oriented, DSP oriented
 - H/W: ASICs, COTS ICs

Many Types of Programmable Processors

- Past
 - ◆ Microprocessor
 - ◆ Microcontroller
 - ◆ DSP
 - ◆ Graphics Processor
- Now / Future
 - ◆ Network Processor
 - ◆ Sensor Processor
 - ◆ Cryptoprocessor
 - ◆ Game Processor
 - ◆ Wearable Processor
 - ◆ Mobile Processor

General Purpose Microprocessors

- Programmable device used in a variety of applications
 - Also known as "microprocessor"
- Features
 - Program memory
 - General datapath with large register file and general ALU
- User benefits
 - Low time-to-market and NRE costs
 - High flexibility

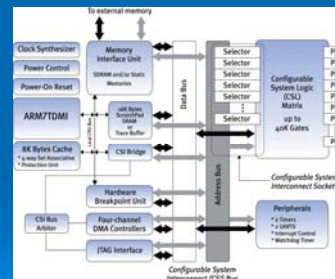


Source: Vahid/ Givargix

Application-Specific Instruction Processors (ASIPs)

- Processors with instruction-sets tailored to specific applications or application domains
 - instruction-set generation as part of synthesis
 - e.g. Tensilica
- Pros:
 - customization yields lower area, power etc.
- Cons:
 - higher h/w & s/w development overhead
 - design, compilers, debuggers
 - higher time to market

Reconfigurable Devices



Other Examples
Atmel's FPSLIC (AVR + FPGA)

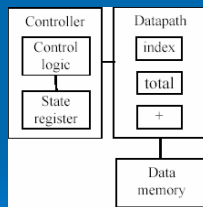
Altera's Nios (configurable RISC on a PLD)

Xilinx's Virtex (multiple RISC cores, distributed DRAM, FPGA)

Triscend's A7 CSoC

ASICs

- Digital circuit designed to execute one program
- Features
 - Contains only the components needed to execute a single program
 - No program memory
- Benefits
 - Fast
 - Low power
 - Small size



ASIC Design

- Extremely complex task
- Must manage close to one billion transistors
 - This continues to exponentially increase
 - Transistors are smaller, but much more complicated
 - Leakage current
 - Interconnect coupling
- Must perform required task(s)
 - Applications expect complicated tasks to be done in hardware
 - Mix of hardware and software components
- Users continually demand better products
 - Lower power, longer battery, smaller, faster, ...

H/W-S/W Architecture

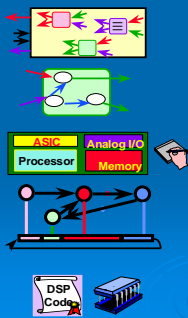
- A significant part of the problem is deciding which parts should be in s/w on programmable processors, and which in specialized h/w
- Today:
 - Ad hoc approaches based on earlier experience with similar products, & on manual design
 - H/W-S/W partitioning decided at the beginning, and then designs proceed separately

Embedded System Design

- CAD tools take care of h/w fairly well
 - Although a productivity gap emerging
- But, S/W is a different story...
 - HLLs such as C help, but can't cope with complexity and performance constraints

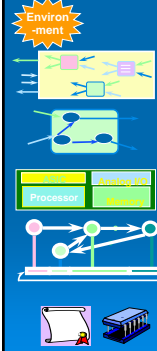
Holy Grail for Tools People: H/W-like synthesis & verification from a behavior description of the whole system at a high level of abstraction using formal computation models

Embedded System Design from a Design Technology Perspective



- Intertwined subtasks
 - Specification/modeling
 - H/W & S/W partitioning
 - Scheduling & resource allocations
 - H/W & S/W implementation
 - Verification & debugging
- Crucial is the co-design and joint optimization of hardware and software

Embedded System Design Flow



- **Modeling**
 - ◆ the system to be designed, and experimenting with algorithms involved;
- **Refining (or "partitioning")**
 - ◆ the function to be implemented into smaller, interacting pieces;
- **HW-SW partitioning: Allocating**
 - ◆ elements in the refined model to either (1) HW units, or (2) SW running on custom hardware or a suitable programmable processor.
- **Scheduling**
 - ◆ the times at which the functions are executed. This is important when several modules in the partition share a single hardware unit.
- **Mapping (Implementing)**
 - ◆ a functional description into (1) software that runs on a processor or (2) a collection of custom, semi-custom, or commodity HW.