CSE 237A

Introduction to Embedded Systems

Tajana Simunic Rosing
Department of Computer Science and Engineering
UCSD

Welcome to CSE 237A!

- Instructor:
  - Tajana Simunic Rosing
    - Email: tajana-at-ucsd.edu; put CSE237a in subject line
  - Office Hours:
    - TTh 11:30-12:30pm, CSE 2118
- Admin:
  - **Sheila Manalo**
    - Email: shmanalo@ucsd.edu
    - Phone: (858) 534-8873
    - Office: CSE 2272
- TAs: Baris Aksanli, Kunal Korgaonkar
  - Email: baksanli@ucsd.edu, korgaon@ucsd.edu
  - Office hrs: TBD
- Class Website:
  - http://www.cse.ucsd.edu/classes/wi13/cse237A-a/
- Grades: http://ted.ucsd.edu
- Discussion board: piazza.com/#winter2013/cse237a

About This Course

- Part of a three course group
  - CSE 237A: Introduction to Embedded Systems
  - CSE 237B: Software for Embedded Systems
  - CSE 237C: Validation and Prototyping of ES
- Depth sequence:
  - Embedded Systems and Software

Course Objectives

- Develop an understanding of the technologies behind the embedded computing systems
  - technology capabilities and limitations of the hardware, software components
  - methods to evaluate design tradeoffs between different technology choices.
  - design methodologies
- Overview of a few hot research topics in ES
- For more details, see the schedule on the webpage

Course Requirements

- No official graduate course as prerequisite, but, many assumptions!
- Knowledge
  - Digital hardware, basic electrical stuff, computer architecture (ISA, organization), programming & systems programming, algorithms
- Skills
  - Advanced ability to program
  - Ability to look up references and track down pubs (Xplore etc)
  - Ability to communicate your ideas (demos, reports)
- Initiative
  - Open-ended problems with no single answer requiring thinking and research
- Interest
  - Have strong interest in research in this or related fields

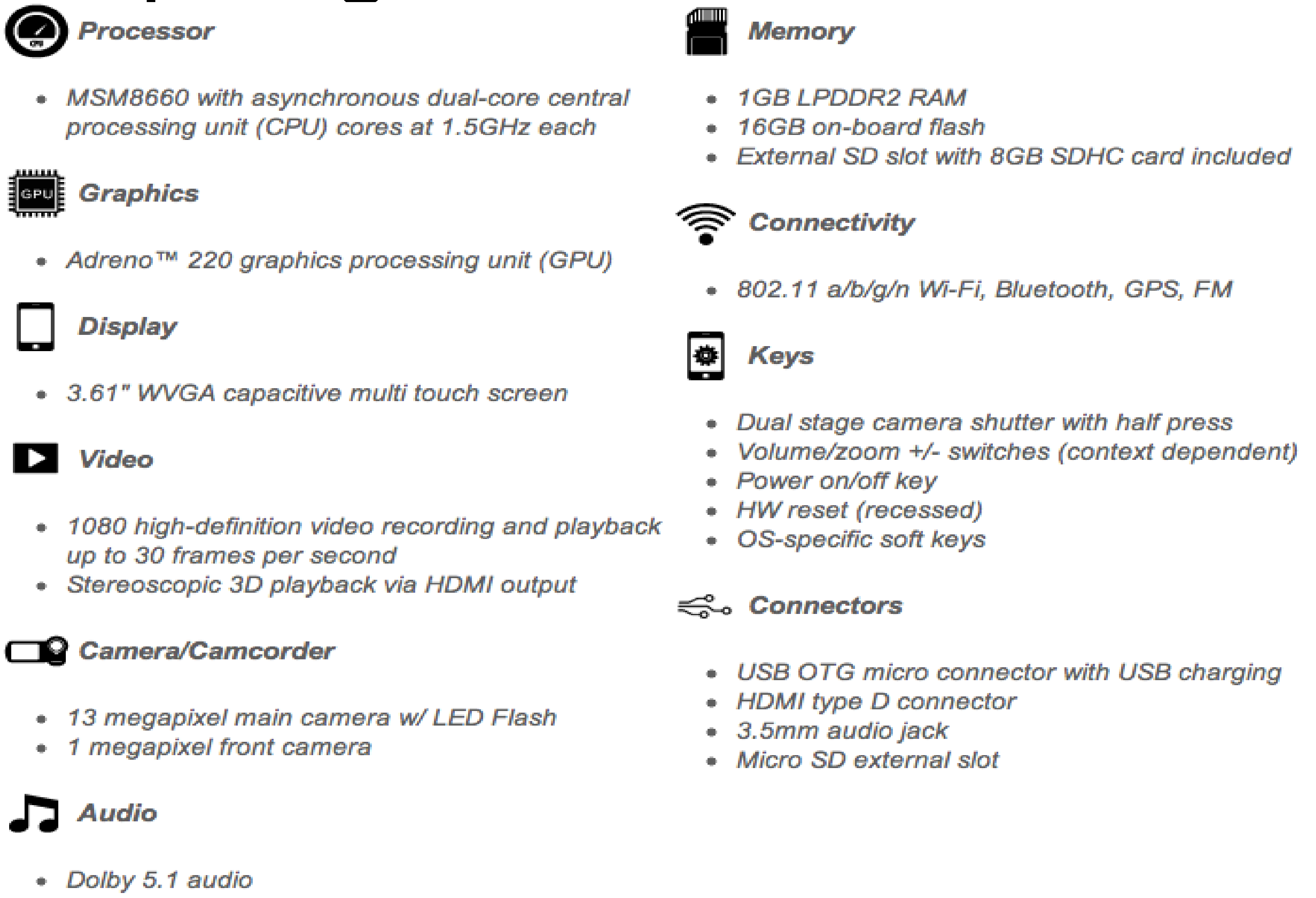
Course Grading

- Homework (3-4): 15%
- Embedded systems project 40%
  - Install OS onto an embedded platform, cross-compile, run and analyize performance and energy consumption of apps, make kernel more energy efficient
- Final exam: 40%
- Class participation, attendance, engagement: 5%
  - Come prepared to discuss the assigned paper(s)

Project Overview

- Compile and flash Android onto a bare mobile phone
  - Cyanogen Android 2.3
  - Snapdragon MDP MSM8660
- Implement your own intelligent DVFS policy as a kernel module
  - Utilize performance counters on the CPU
  - Minimize power consumption for a variety of workloads
- One phone provided for each group of two students

Snapdragon MDP MSM8660



Processor

- MSM8660 with asynchronous dual-core central processing unit (CPU) cores at 1.5GHz each

Graphics

- Adreno™ 220 graphics processing unit (GPU)

Display

- 3.61" WVGA capacitive multi touch screen

Video

- 1080 high-definition video recording and playback up to 30 frames per second
- Stereoscopic 3D playback via HDMI output

Camera/Camcorder

- 13 megapixel main camera w/ LED Flash
- 1 megapixel front camera

Audio

- Dolby 5.1 audio

Memory

- 1GB LPDDR2 RAM
- 16GB on-board flash
- External SD slot with 8GB SDHC card included

Connectivity

- 802.11 a/b/g/n Wi-Fi, Bluetooth, GPS, FM

Keys

- Dual stage camera shutter with half press
- Volume/zoom +/- switches (context dependent)
- Power on/off key
- HW reset (recessed)
- OS-specific soft keys

Connectors

- USB OTG micro connector with USB charging
- HDMI type D connector
- 3.5mm audio jack
- Micro SD external slot

https://developer.qualcomm.com/develop/development-devices/snapdragon-mdp-msm86

Project Components

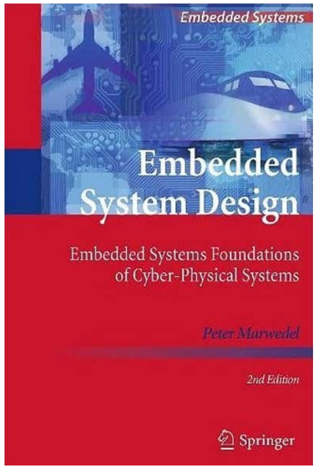
1. Set up the development environment
2. Install required tools and get source code
3. Compile and flash Android onto the device
4. Implement a simple DVFS policy based on CPU utilization
5. Implement an advanced DVFS policy that makes use of additional information (cache misses, stalls, etc…) provided by the processor’s PMU (performance measurement unit)

<u>Deliverables</u>

- In class demonstration of the DVFS policy running on a variety of workloads.
- A 5-page report that includes the details of the implemented DVFS policy and a table of results for the provided test cases.

Textbook & Assigned Reading

- Required text:
  - By Peter Marwedel
  - 2nd edition, Springer 2011


Embedded Systems
Embedded System Design
Embedded Systems Foundations of Cyber-Physical Systems
Peter Marwedel
2nd Edition
Springer

- A set of papers will be required reading
  - will relate to the core topic of that class
  - you are expected to read it BEFORE the class
- In addition I will give pointers to papers and web resources

Reference books

- Peter Marwedel, “Embedded Systems Design,” Kluwer, 2004.
- “Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers,” National Research Council. http://www.nap.edu/books/0309075688/html/
- John A. Stankovic and Kirthi Ramamritham, "Hard Real-Time Systems," IEEE Computer Society Press.
- G.D. Micheli, W. Wolf, R. Ernst, “Readings in Hardware/Software Co-Design,” Morgan Kaufman.
- S.A. Edwards, “Languages for Digital Embedded Systems,” Kluwer, 2000.
- R. Melhem and R. Graybill, “Power Aware Computing,” Plenum, 2002.
- M. Pedram and J. Rabaey, “Power Aware Design Methodologies,” Kluwer, 2002.
- Bruce Douglass, "Real-Time UML - Developing Efficient Objects for Embedded Systems," Addison-Wesley, 1998.
- Hermann Kopetz, "Real-Time Systems : Design Principles for Distributed Embedded Applications," Kluwer, 1997.
- Hassan Gomaa, "Software Design Methods for Concurrent and Real-Time Systems," Addison-Wesley, 1993.
- P. Lapsley, J. Bier, A. Shoham, and E.A. Lee, “DSP Processor Fundamentals: Architectures and Features,” Berkeley Design technology Inc,, 2001.
- R. Gupta, "Co-synthesis of Hardware & Software for Embedded Systems," Kluwer, 1995.
- Felice Balarin, Massimiliano Chiodo, and Paolo Giusto, "Hardware-Software Co-Design of Embedded Systems : The Polis Approach," Kluwer, 1997.
- Jean J. Labrosse, "Embedded Systems Building Blocks : Complete And Ready To Use Modules In C ," R&D Publishing, 1995.
- Jean J. Labrosse, "uC / OS : The Real Time Kernel," R&D Publishing, 1992.

Embedded Systems on the Web

- Berkeley Design technology, Inc.: http://www.bdti.com
- EE Times Magazine: http://www.eet.com/
- Linux Devices: http://www.linuxdevices.com
- Embedded Linux Journal: http://embedded.linuxjournal.com
- Embedded.com: http://www.embedded.com/
    - *Embedded Systems Programming* magazine
- Circuit Cellar: http://www.circuitcellar.com/
- Electronic Design Magazine: http://www.planetee.com/ed/
- Electronic Engineering Magazine: http://www2.computeroemonline.com/magazine.html
- Integrated System Design Magazine: http://www.isdmag.com/
- Sensors Magazine: http://www.sensorsmag.com
- Embedded Systems Tutorial: http://www.learn-c.com/
- Collections of embedded systems resources
    - http://www.ece.utexas.edu/~bevans/courses/ee382c/resources/
    - http://www.ece.utexas.edu/~bevans/courses/realtime/resources.html
- Newsgroups
    - comp.arch.embedded, comp.cad.cadence, comp.cad.synthesis, comp.dsp, comp.realtime, comp.software-eng, comp.speech, and sci.electronics.cad

Embedded Systems Courses

- Alberto Sangiovanni-Vincentelli @ Berkeley
  - EE 249: Design of Embedded Systems: Models, Validation, and Synthesis
    - http://www-cad.eecs.berkeley.edu/~polis/class/index.html
- Brian Evans @ U.T. Austin
  - EE382C-9 Embedded Software Systems
    - http://www.ece.utexas.edu/~bevans/courses/ee382c/index.html
- Edward Lee @ Berkeley
  - EE290N: Specification and Modeling of Reactive Real-Time Systems
    - http://ptolemy.eecs.berkeley.edu/~eal/ee290n/index.html
- Mani Srivastava @ UCLA
  - EE202A: Embedded and Real Time Systems
    - http://nesl.ee.ucla.edu/courses/ee202a/2003f/
- Bruce R. Land @ CMU
  - EE476: Designing with Microcontrollers
    - http://instruct1.cit.cornell.edu/courses/ee476

Conferences and Journals

- Conferences & Workshops
  - ACM/IEE DAC
  - IEEE ICCAD
  - IEEE RTSS
  - ACM ISLPED
  - IEEE CODES+ISSS
  - CASES
  - Many others…

- Journals & Magazines
  - ACM Transactions on Design Automation of Electronic Systems
  - ACM Transactions on Embedded Computing Systems
  - IEEE Transactions on Computer-Aided Design
  - IEEE Transactions on VLSI Design
  - IEEE Design and Test of Computers
  - IEEE Transactions on Computers
  - Journal of Computer and Software Engineering
  - Journal on Embedded Systems

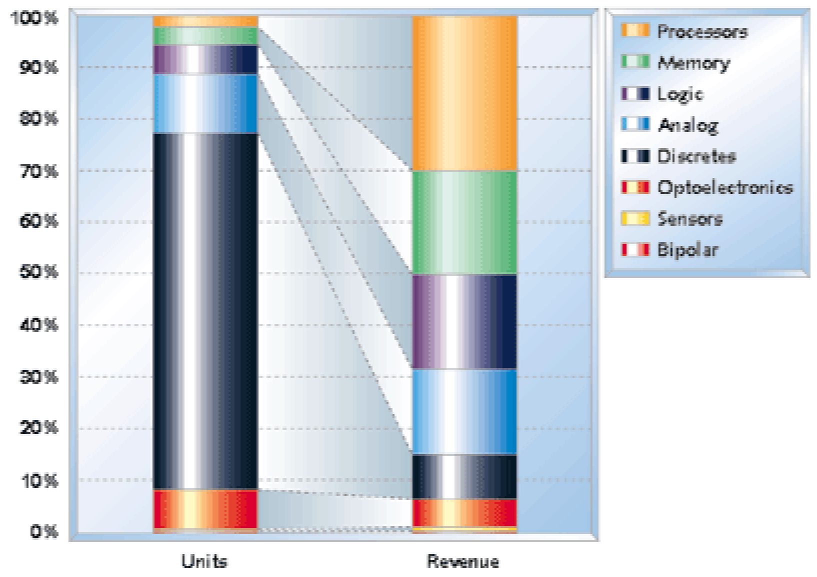
What are embedded systems and why should we care?

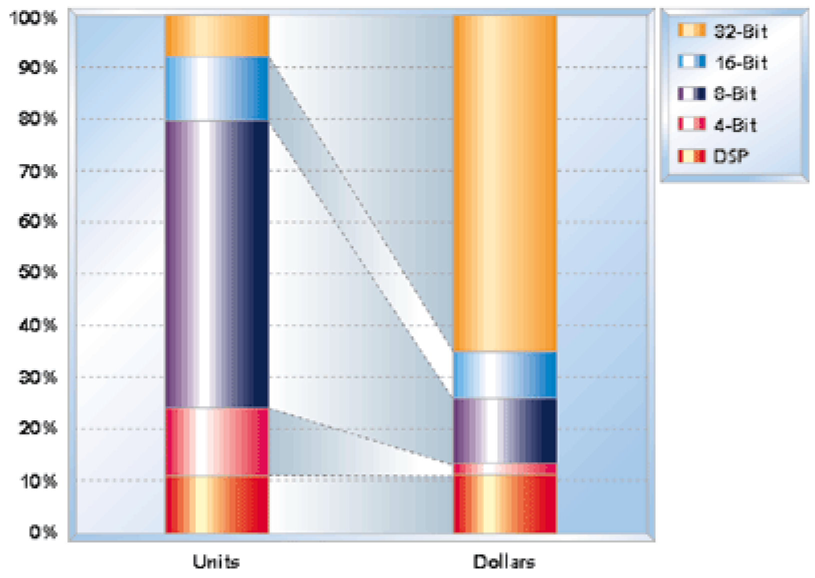
What are embedded systems?

- Systems which use computation to perform a *specific function*
- embedded within a larger device and environment
- Heterogeneous & reactive to environment

Main reason for buying is **not** information processing

Embedded processor market





- Processors strongly affect SW development – keeps their prices high
- Only 2% of processors drive PCs!
- ARM sells 3x more CPUs then Intel sells Pentiums
- 79% of all high-end processors are used in embedded systems

Source: EETimes

Tied to advances in semiconductors

- A typical chip
  - 1-10 GHz, 100-1000 MOP/sq mm, 10-100 MIPS/mW
- Cost is almost independent of functionality
  - 10,000 units/wafer, 20K wafers/month
  - $5 per part
  - Processor, MEMS, Networking, Wireless, Memory
    - But it takes $20M to build one today, going to $50+M
- So there is a strong incentive to port your application, system, box to the “chip”

Source: RG UCSD

Trends in Embedded Systems

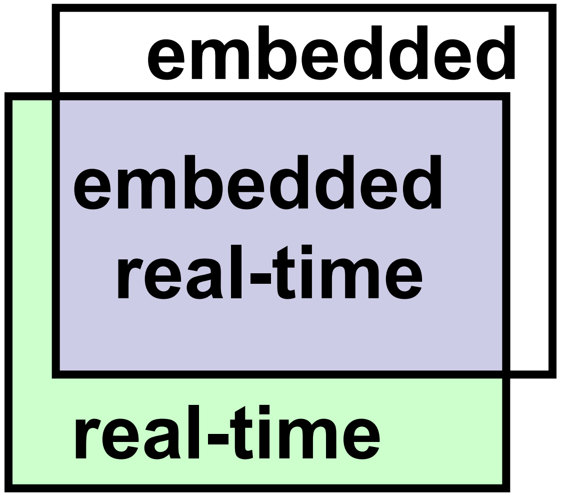
- Increasing code size
  - average code size: 16-64KB in 1992, 64K-512KB in 1996
  - migration from hand (assembly) coding to high-level languages
- Reuse of hardware and software components
  - processors (micro-controllers, DSPs)
  - software components (drivers)
- Increasing integration and system complexity
  - integration of RF, DSP, network interfaces
  - 32-bit processors, IO processors (I2O)

Structured design and composition methods are essential.

Source: RG UCSD

Characteristics of Embedded Systems

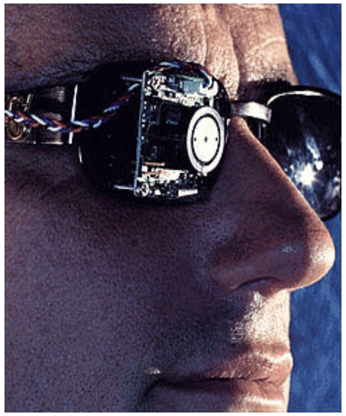
- Application specific
- Efficient
  - energy, code size, run-time, weight, cost
- Dependable
  - Reliability, maintainability, availability, safety, security
- Real-time constraints
  - Soft vs. hard
- Reactive - connected to physical environment
  - sensors & actuators
- Hybrid
  - Analog and digital
- Distributed
  - Composability, scalability, dependability
- Dedicated user interfaces



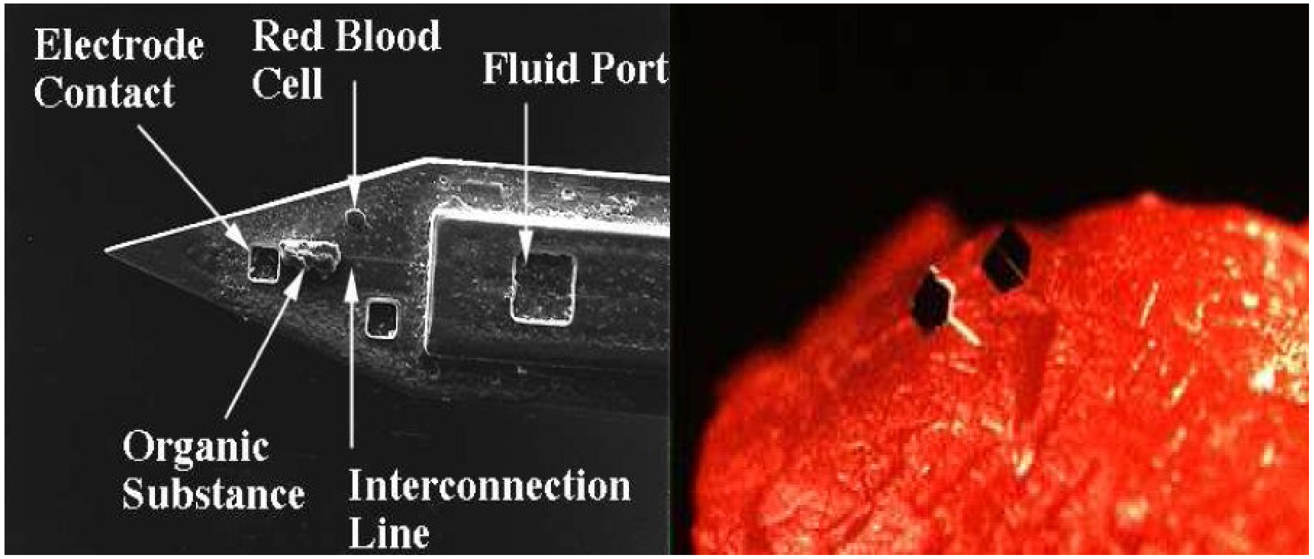
Applications

- Medical systems
  e.g. “artificial eye”

- e.g. “micro-needles”



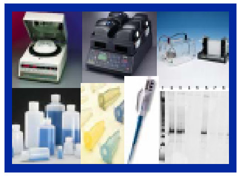
www.dobelle.com

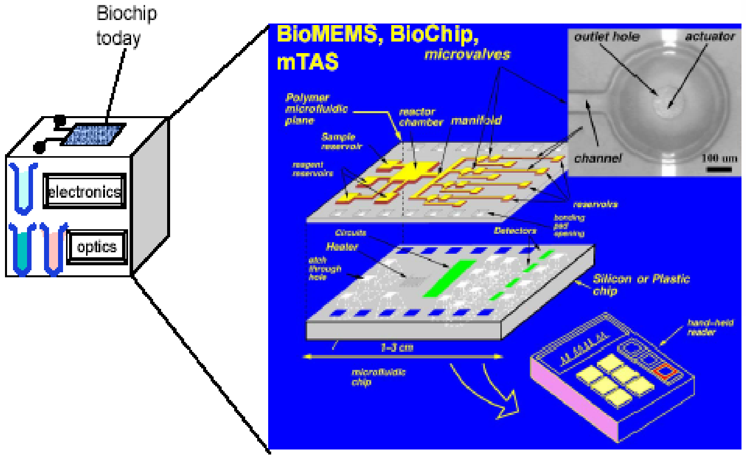


Source: ASV UCB

On-chip Chemistry

Today — In 3 years — In 6 years

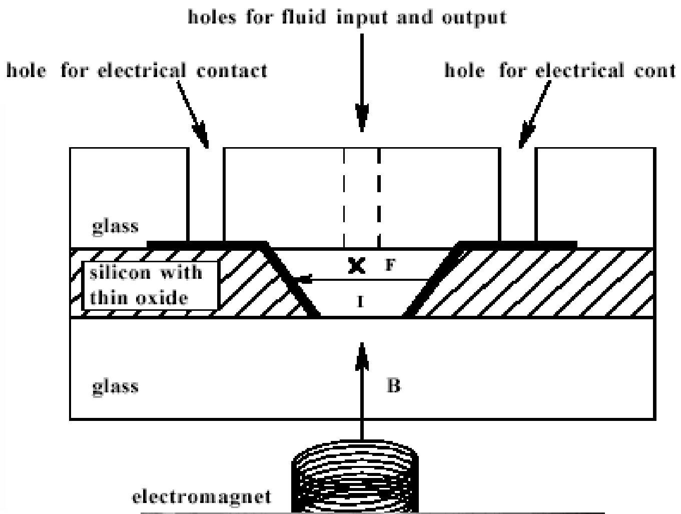


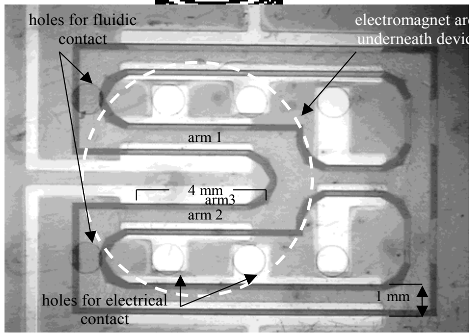


Hours to days — 1 hour — Seconds to minutes

Time from sample to "CORRECT" answer

Abraham P. Lee, Ph.D.





Pedometer

- Obvious computer work:
  - Count steps
  - Keep time
  - Averages
  - etc.
- Hard computer work:
  - Actually identify when a step is taken
  - Sensor feels motion of device, not of user feet

If you want to play

- Lego mindstorms robotics kit
  - Standard controller
    - 8-bit processor
    - 64 kB of memory
  - Electronics to interface to motors and sensors
- Good way to learn embedded systems

Mobile phones

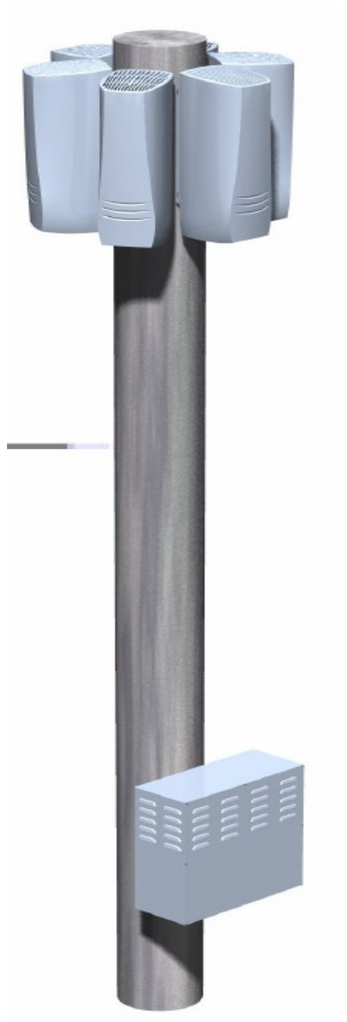
- Multiprocessor
  - 8-bit/32-bit for UI
  - DSP for signals
  - 32-bit in IR port
  - 32-bit in Bluetooth
- 16 GB Flash
- Custom chips
- Power consumption & battery life depends on software

Inside the PC

- Custom processors
  - Graphics, sound
- 32-bit processors
  - IR, Bluetooth
  - Network, WLAN
  - Hard disk
  - RAID controllers
- 8-bit processors
  - USB
  - Keyboard, mouse

Mobile base station

- Massive signal processing
  - Several processing tasks per connected mobile phone
- Based on DSPs
  - Standard or custom
  - 100s of processors



Telecom Switch

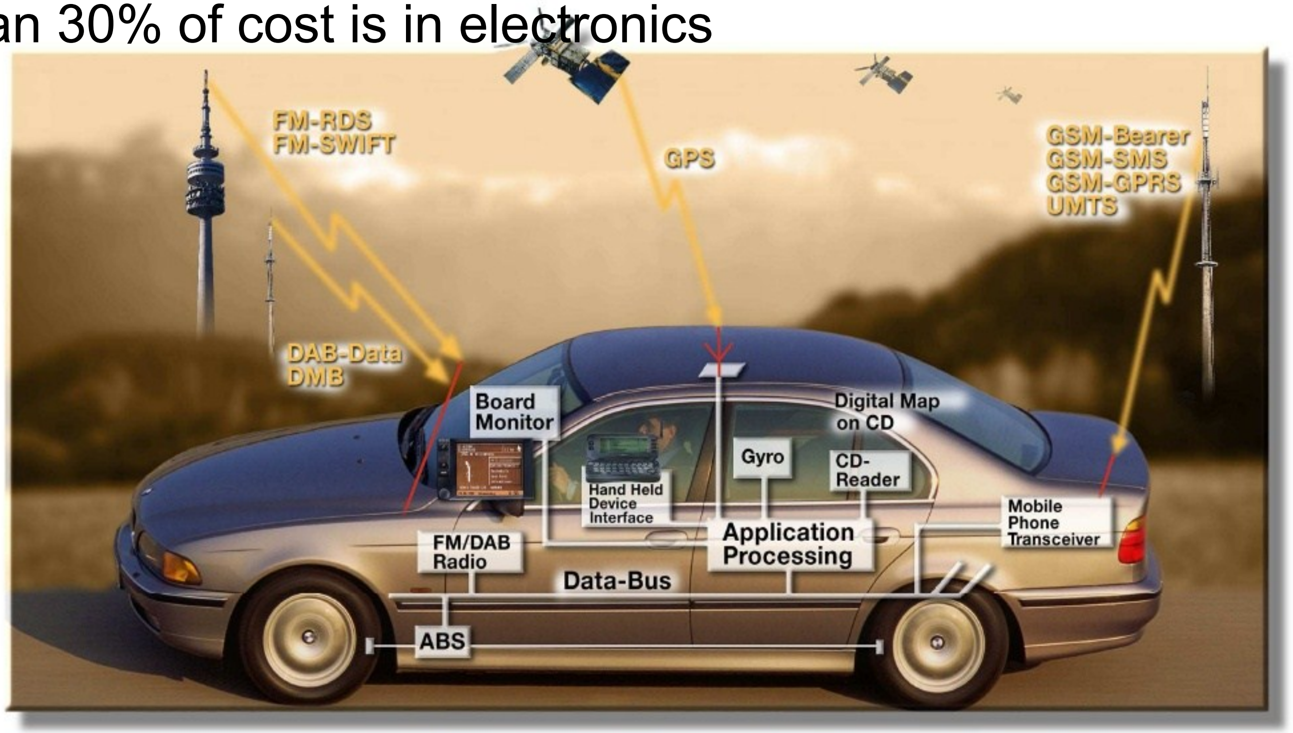
- Rack-based
  - Control cards
  - IO cards
  - DSP cards
  - ...
- Optical & copper connections
- Digital & analog signals

Smart Welding Machine

- Electronics control voltage & speed of wire feed
- Adjusts to operator
  - kHz sample rate
  - 1000s of decisions/second
- Perfect weld even for quite clumsy operators
- Easier-to-use product, but no obvious computer

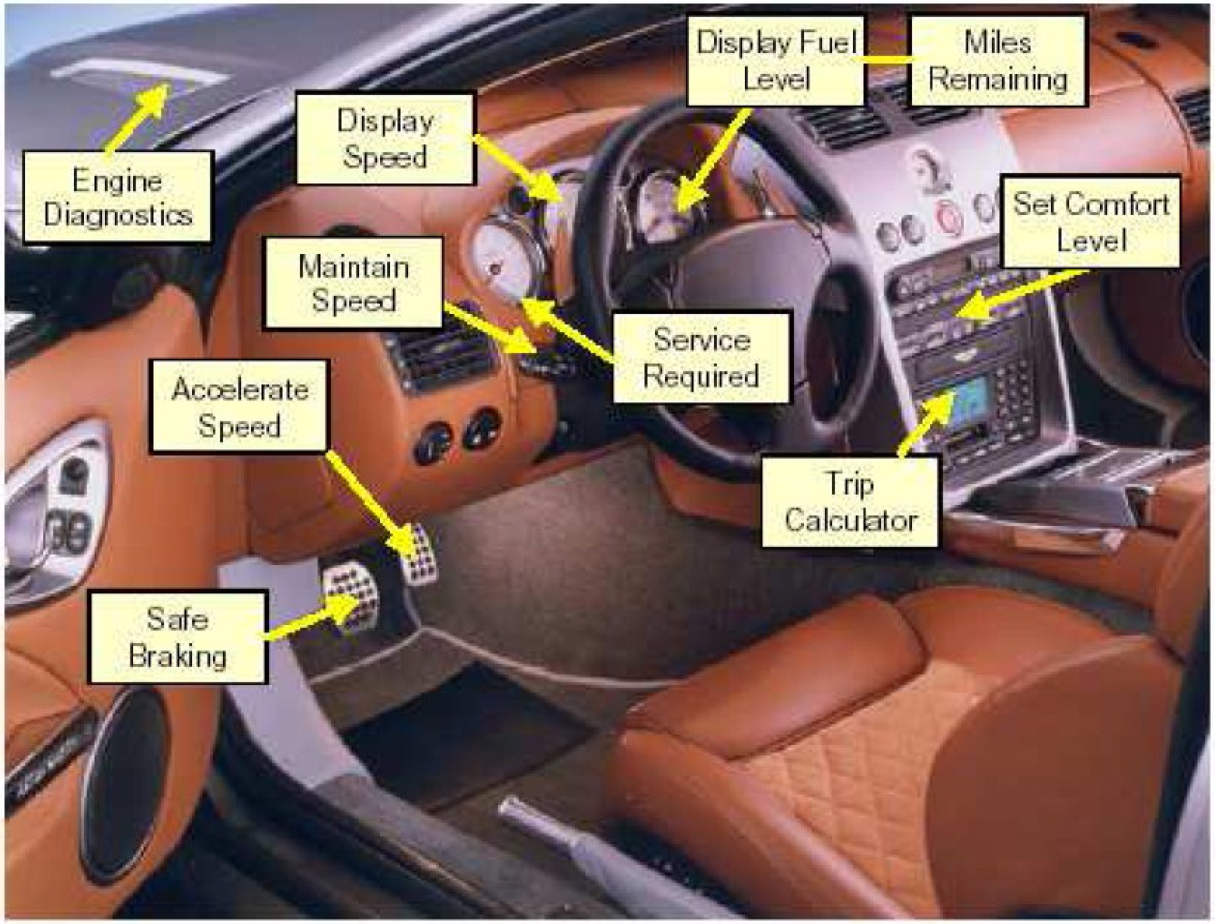
Cars

- Multiple processors networked together (~100), wide variety of CPUs:
  - 8-bit – door locks, lights, etc; 16-bit – most functions; 32-bit – engine control, airbags
- Multiple networks
  - Body, engine, telematics, media, safety
- 90% of all innovations based on electronic systems
- More than 30% of cost is in electronics



FUNCTION OF CONTROLS

Typical minivan application



Configure

Sense

Actuate

Regulate

Display

Trend

Diagnose

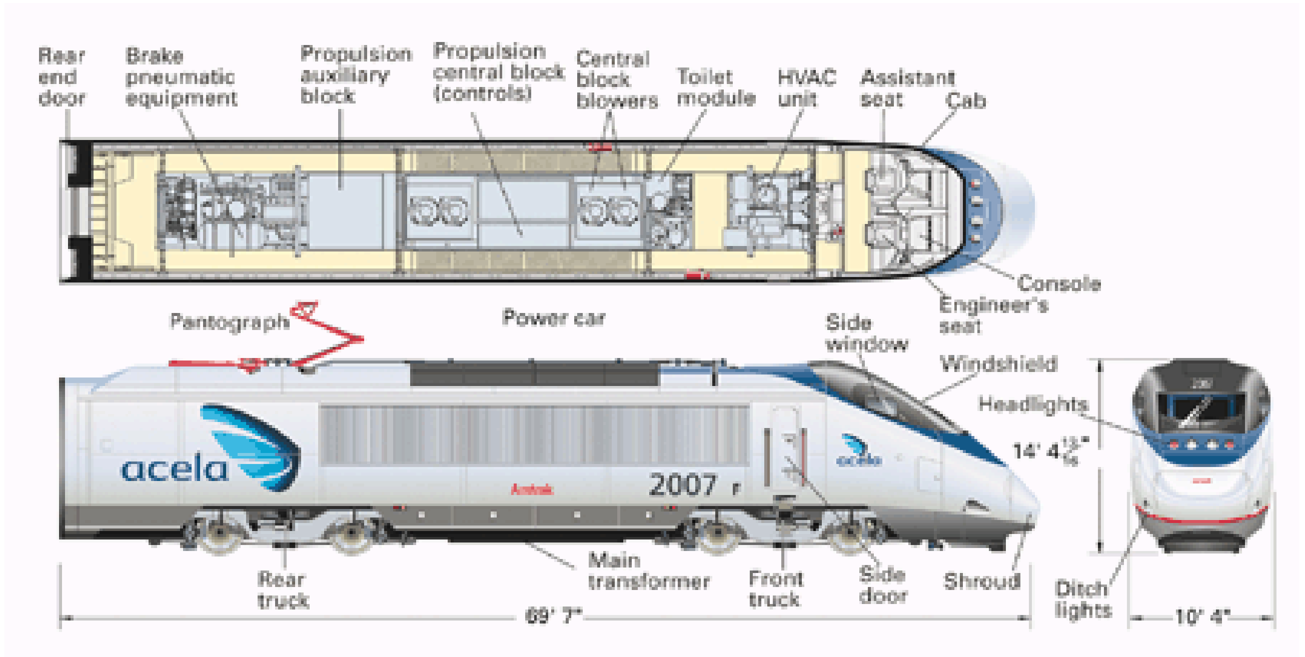
Predict

Archive

Source: ASV UCB

Transportation

Amtrak Acela High Speed Train



- High speed tilting train service between Boston, New York, and Washington, D.C.
- Built by Bombardier, uses FT-10 free topology twisted pair channel to monitor and control propulsion, power inverters, braking, fire protection systems, ride stability, safety, and comfort.

Source: Echelon

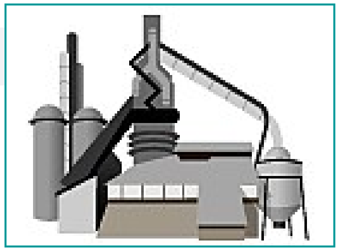
Building Automation

Coeur Défense, Paris

- Location and access
  - The biggest office property complex in Europe located at the heart of the central esplanade of the Paris-La Défense business district
- The building
  - Property complex with a total floor area of 182,000 m² in two towers 180 metres high (39 floors) and 3 small (8-floors) buildings linked to each other by a "glass cathedral".
- Building Automation System
  - 15000 embedded control devices
  - One (1) *i*.LON™ 100 per floor (150 floors) for routing data

Source: Echelon

Process Control



Bellagio Hotel, Las Vegas NV

- Water fountain show
- Fountain and sprinkler systems controls
- Pump controls
- Valve controls
- Choreographed lights and music
- Leak detection

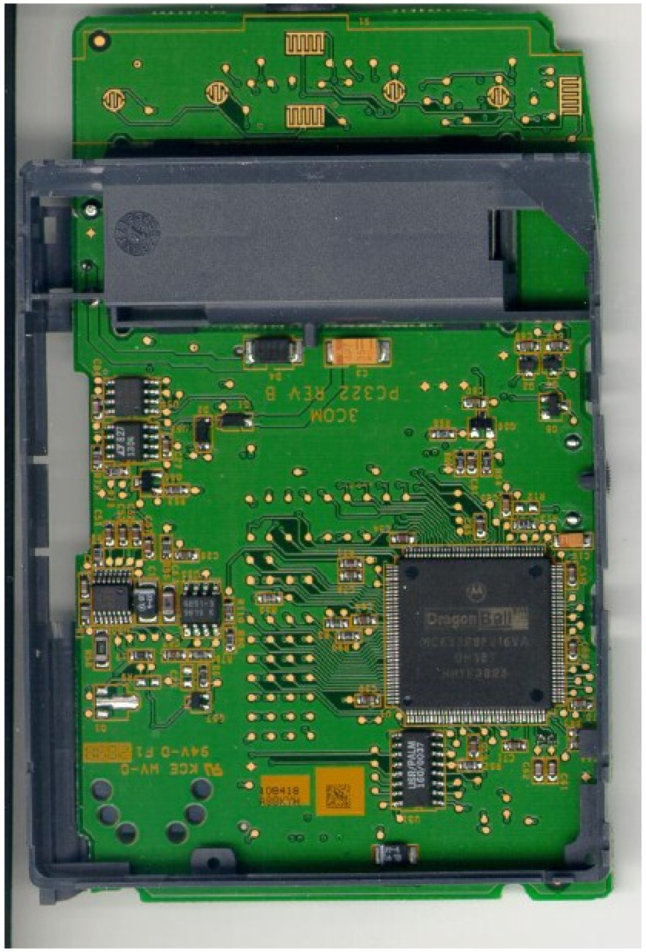
Source: Echelon

Embedded system metrics

- Some metrics:
  - *performance*: MIPS, reads/sec etc.
  - *power*: Watts
  - *cost*: Dollars
    - Nonrecurring engineering cost, manufacturing cost
  - *size*: bytes, # components, physical space occupied
  - Flexibility, Time-to-prototype, time-to-market
  - Maintainability, correctness, safety
- MIPS, Watts and cost are related
  - technology driven
  - to get more MIPS for fewer Watts
    - look at the sources of power consumption
    - use power management and voltage scaling

Source: MS HPL

Example: PDA design



Why did they design it this way?

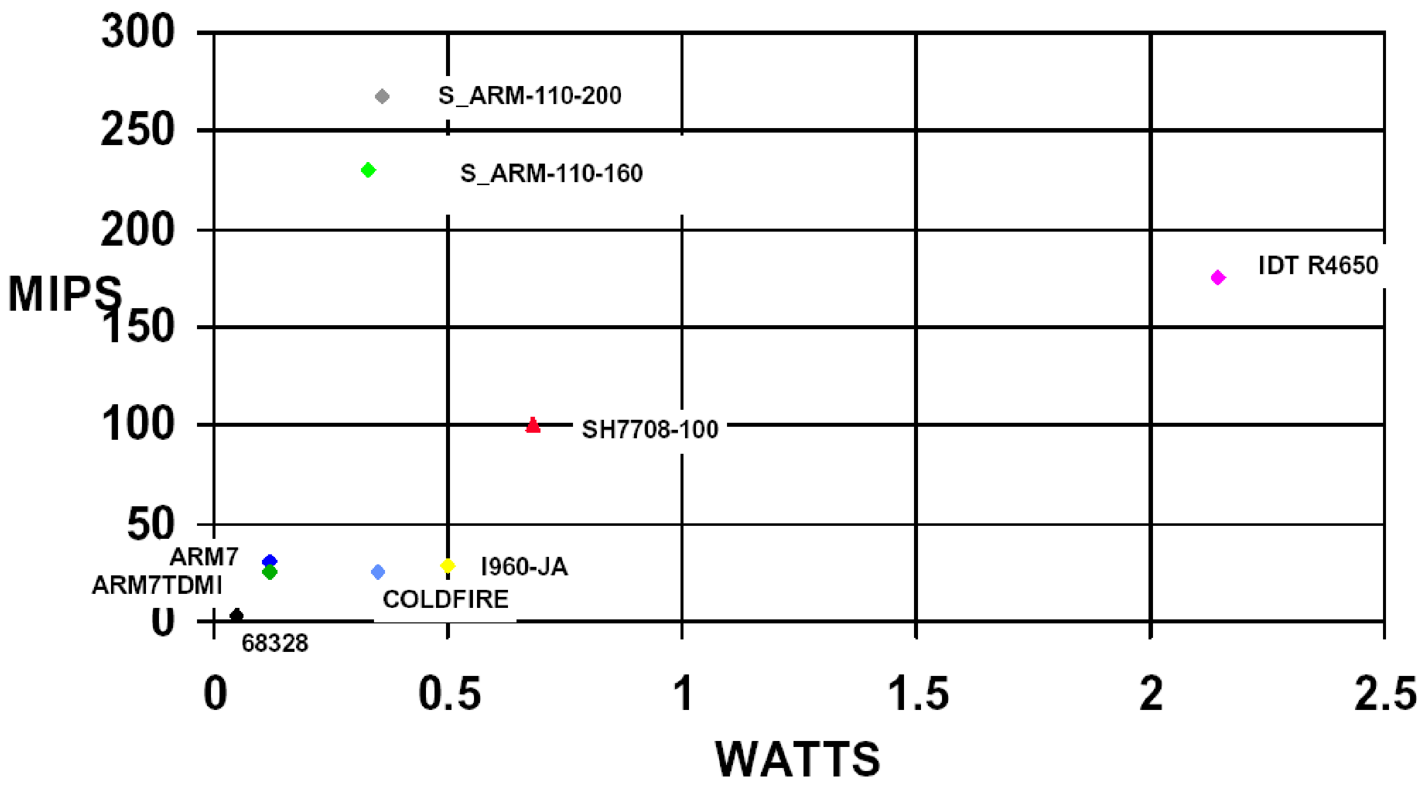
A ‘Dragonball*’ processor?

We all wanted StrongARMs

*The Dragonball used in the early Palm Pilots is a Motorola 68328

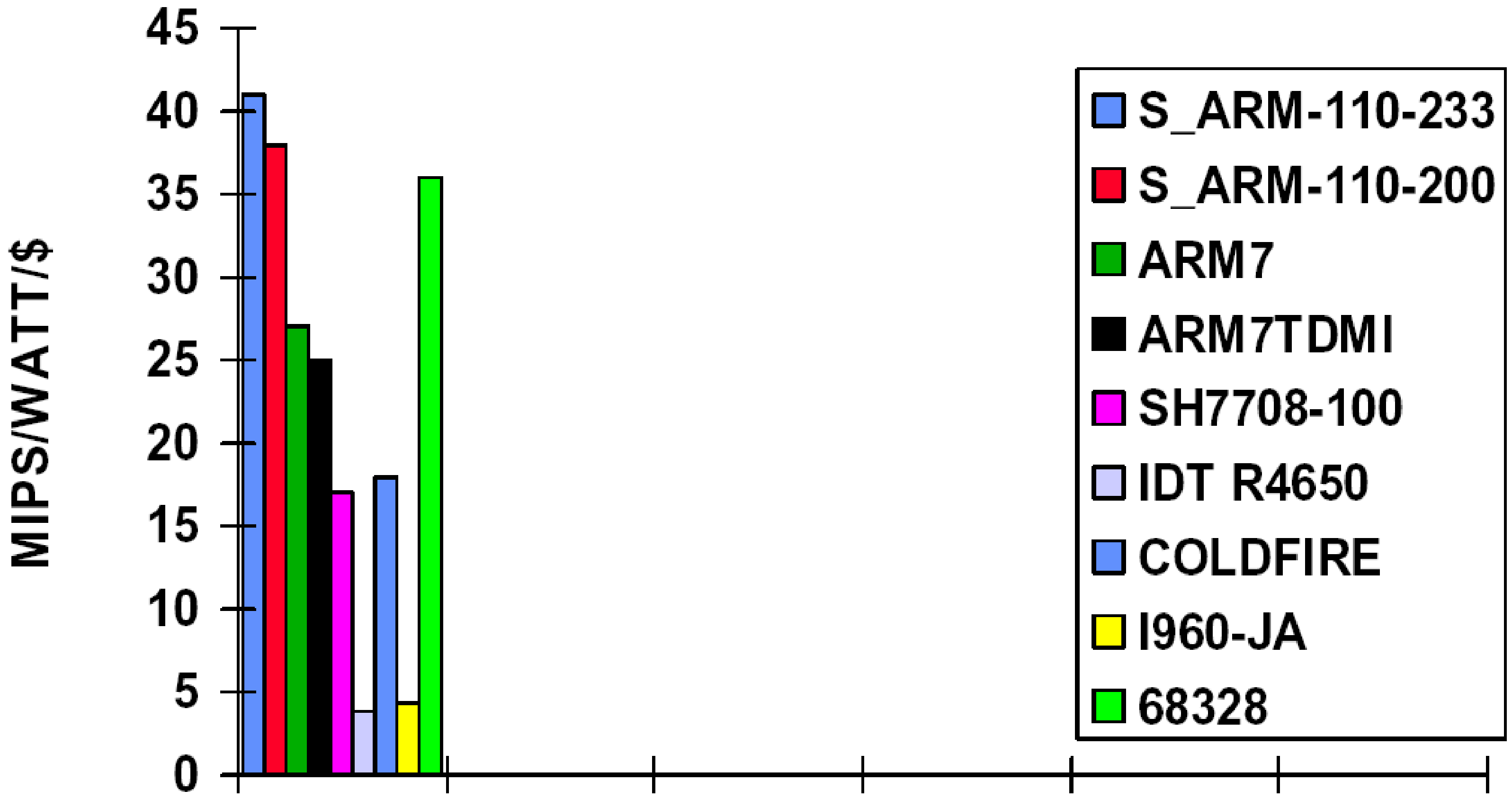
Source: MS HPL

MIPS vs. Watts



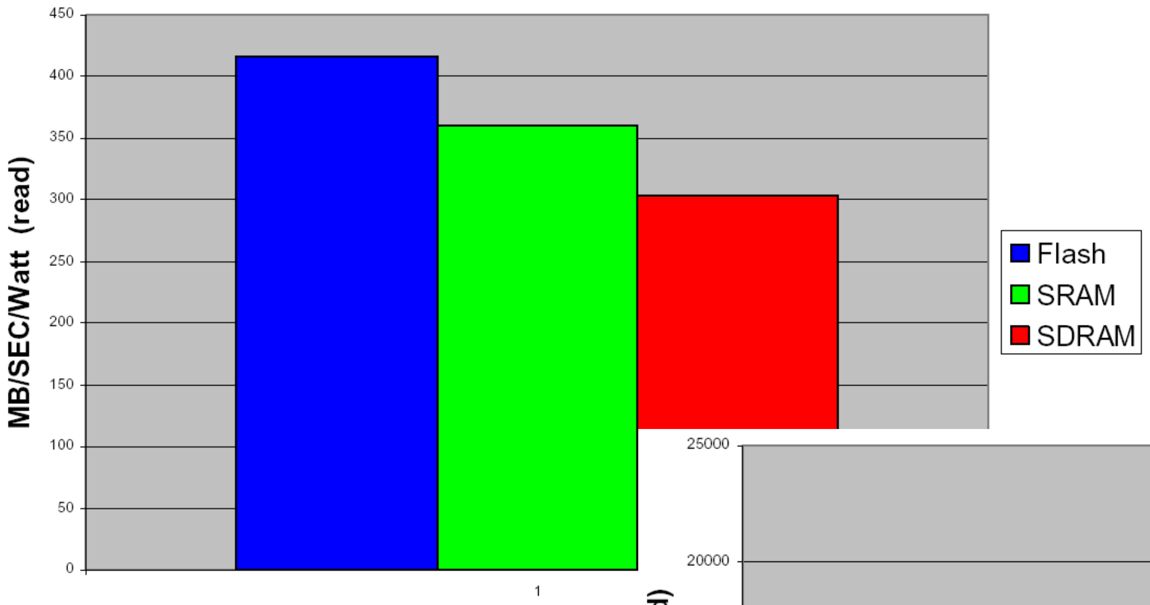
Source: MS HPL

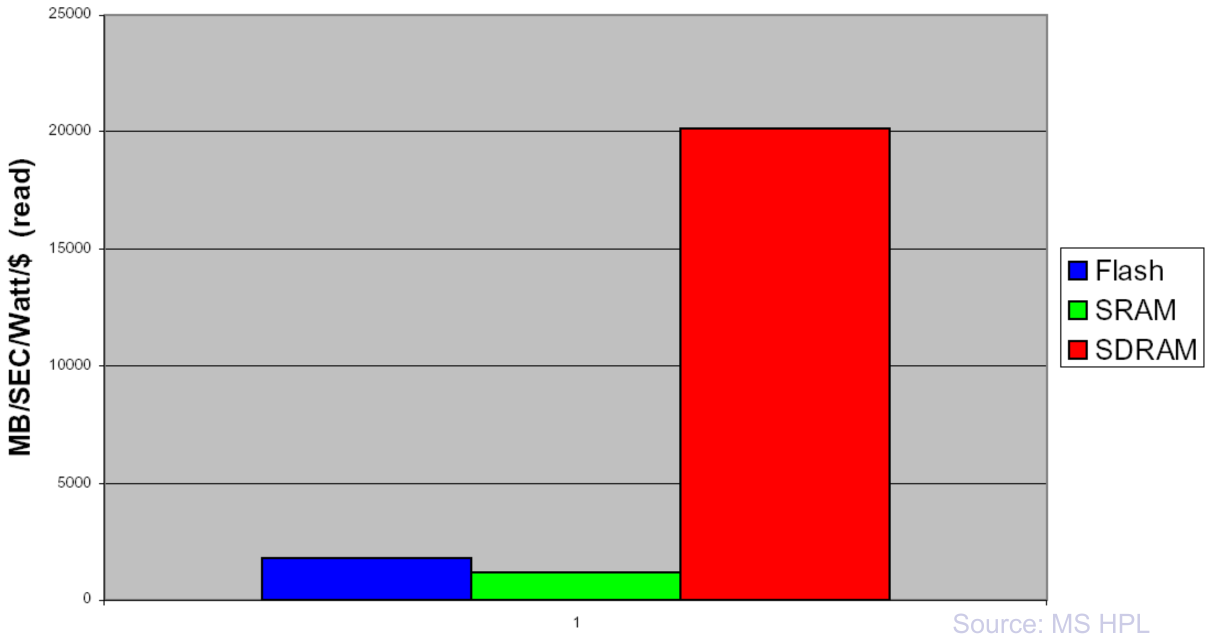
MIPS/W/$



Source: MS HPL

Bandwidth vs. Watt and $





This is why PDAs use SDRAM

Source: MS HPL