12
Embedded $ystem Engineering Economics

October 7, 2015

18-649 Distributed Embedded Systems

Philip Koopman

Electrical & Computer
ENGINEERING

Carnegie Mellon

Introduction

◆ **Hardware economics**

- Why aren't embedded systems all 32-bit CPUs?
- Examples of microcontroller cost tradeoffs

◆ **Software economics**

- Cost of code
- Why saving money on hardware can increase total costs

◆ **Performance margin**

- Why using 100% of resources is a bad idea
- Thoughts on assembly language source code to squeeze out system performance

Recurring & Non-Recurring Costs

◆ **Recurring Expenses (RE)**
  – **directly related to each unit produced**
  • Raw materials
  • Manufacturing labor
  • Shipping

◆ **Non-Recurring Expenses (NRE)**
  – **"one-time" costs to produce the first unit**
  • Engineering time
  • Semiconductor masks
  • Capital equipment (assuming equipment bought up-front)
  • Software

◆ **Cost of goods is generally:**
  • Assumes amortization of NRE over number of items produced

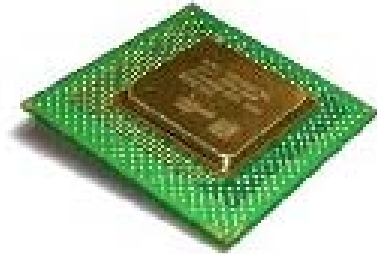$$CostPerItem = RE + \left(\frac{NRE}{\#\,Items} \right)$$

Hardware Costs

◆ **Hardware is more than CPU cost; it includes:**

- CPU
- Memory
- A/D; D/A; other I/O
- Clock
- Power supply
- Circuit board
- Cooling
- …

Figure 1. Pentium 4 Processor 423-Pin OOI Package

◆ **Constraints on embedded hardware beyond cost:**

- Power
- Size, especially for cooling equipment
- Maximum clock speed (due to radio frequency interference issues)
- Availability of integrated features, especially memory & I/O

Figure 9. Install Fan Heatsink and Both Clips

(Photos: Intel, 2004)

4

Why Aren't Embedded CPUs all 32 bits?

- ◆ **The Intel 386 was 32 bits in 1985, with 275,000 transistors**
  - Now we can build billions of transistors on a single chip!

- ◆ **First answer: fast chips are optimized for big programs, not embedded**
  - 4 MB on-chip cache for an Itanium takes 24 million transistors (assume 6T cell)
  - Many, many transistors used for superscalar + out of order execution
  - And, you can't keep it cool in many embedded systems

Moore's Law
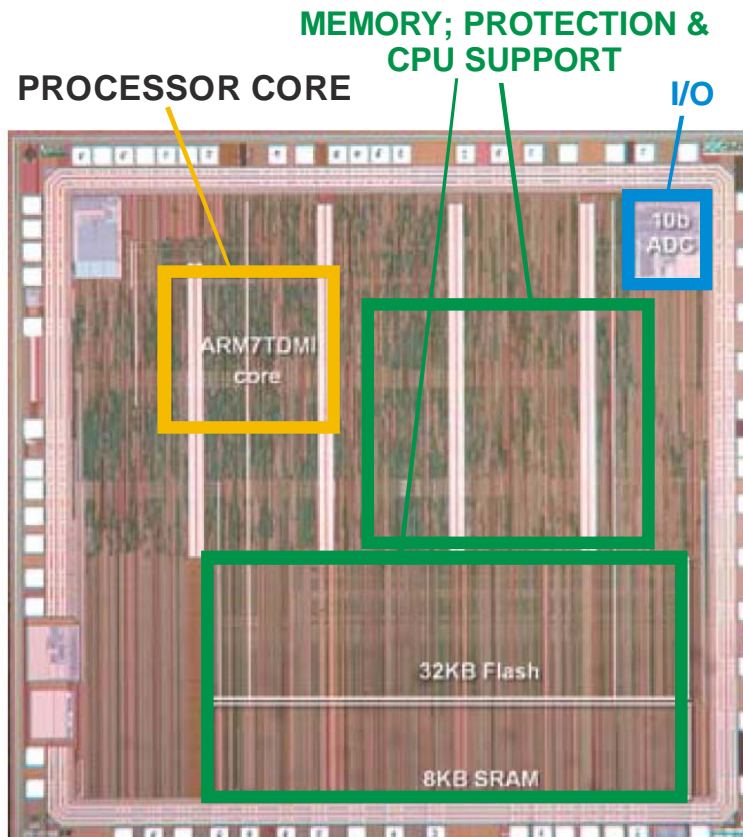### Transistors in High-End Intel CPUs
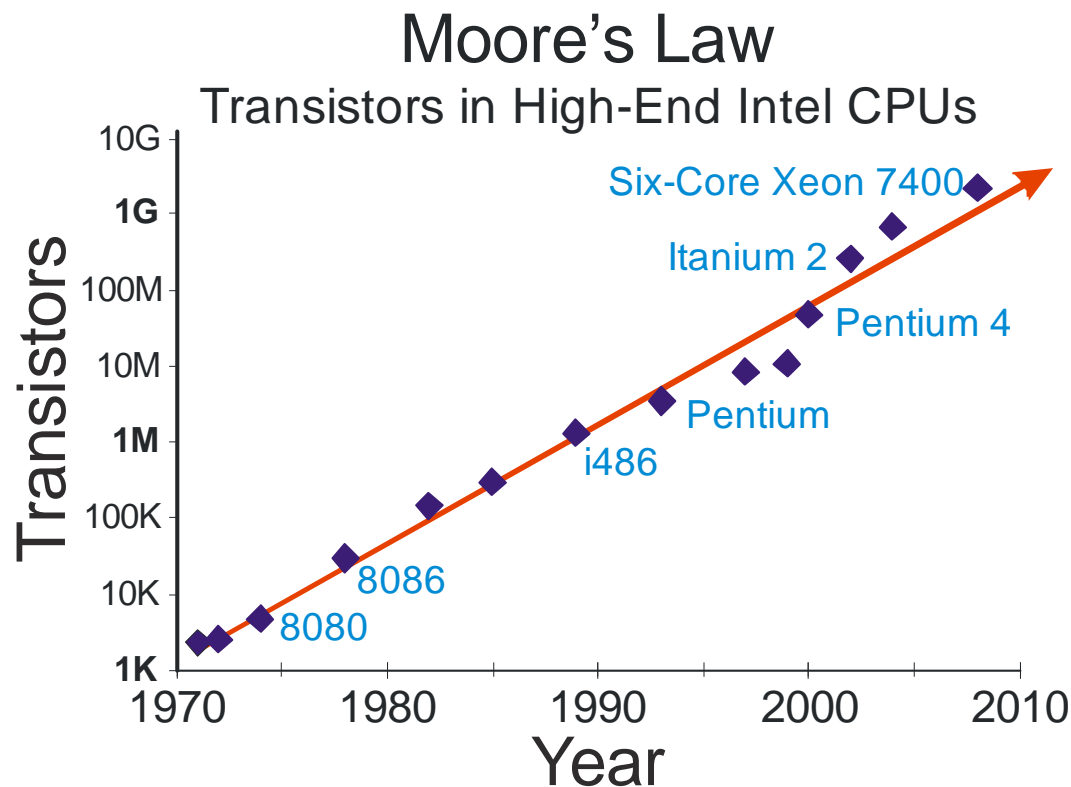
Embedded Chips Have To Be Small(er)

- **Most embedded systems need a $1 to $10 CPU**
  - Can you afford a $500 CPU in a toaster oven?

- **This means die size is smaller than a huge CPU**
  - Smaller die takes less wafer space, meaning more raw chips per wafer
  - *And* smaller die gets better yield, meaning more good chips per wafer

  - Let's say a big CPU has 100 million transistors for $1000
    - At an arm-waving approximation perhaps you can get 2 million transistors for $10
    - This could fit an Intel 386 and 256 KB of on-chip memory, BUT no I/O

- **Embedded systems have to minimize total size and cost**
  - So real embedded systems combine CPU, memory, and I/O
  - Common to have 8K to 64K of flash memory on-chip
  - (Don't really need more than an 8-bit processor if you only have 64KB of memory and are operating on 8-bit analog inputs!)
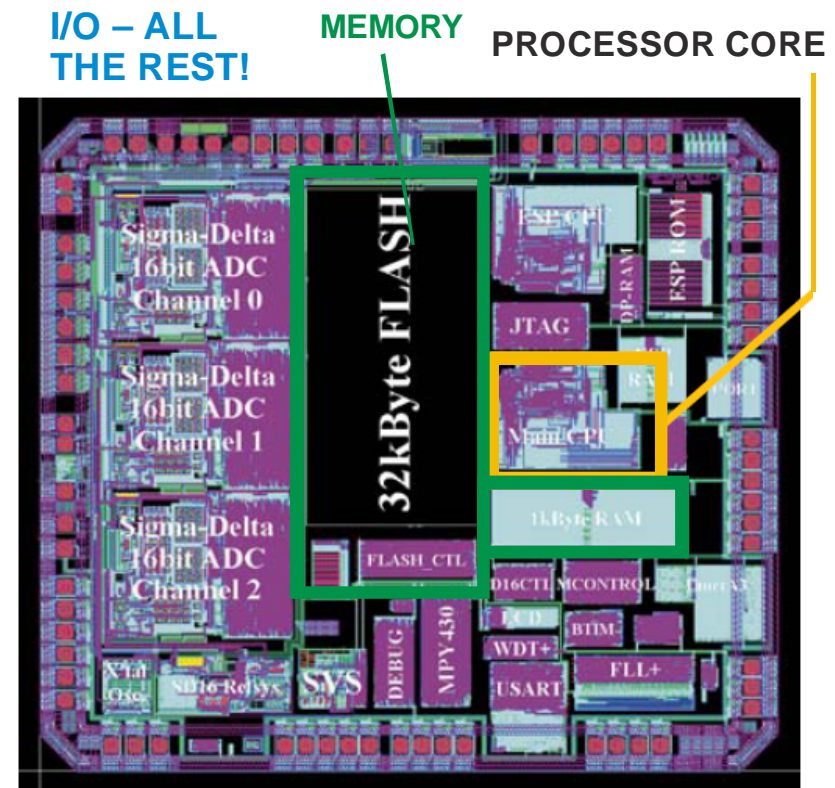
How Embedded Microcontrollers Spend Transistors

◆ **32-bit & 64-bit processors: optimize for speed – often $5 - $100**

◆ **8- & 16-bit processors: optimize for I/O integration**

  • Small memory, no operating system – often $0.50 - $10

◆ **Low-end CPUs spend chip area to lower total system cost**

32-bit ARM CPU   8/16-bit TI CPU

PROCESSOR CORE

MEMORY; PROTECTION &
CPU SUPPORT

I/O

ARM7TDMI core

32KB Flash

8KB SRAM

I/O – ALL
THE REST!

MEMORY

PROCESSOR CORE

Sigma-Delta 16bit ADC Channel 0

Sigma-Delta 16bit ADC Channel 1

Sigma-Delta 16bit ADC Channel 2

32kByte FLASH

1kByte RAM

JTAG

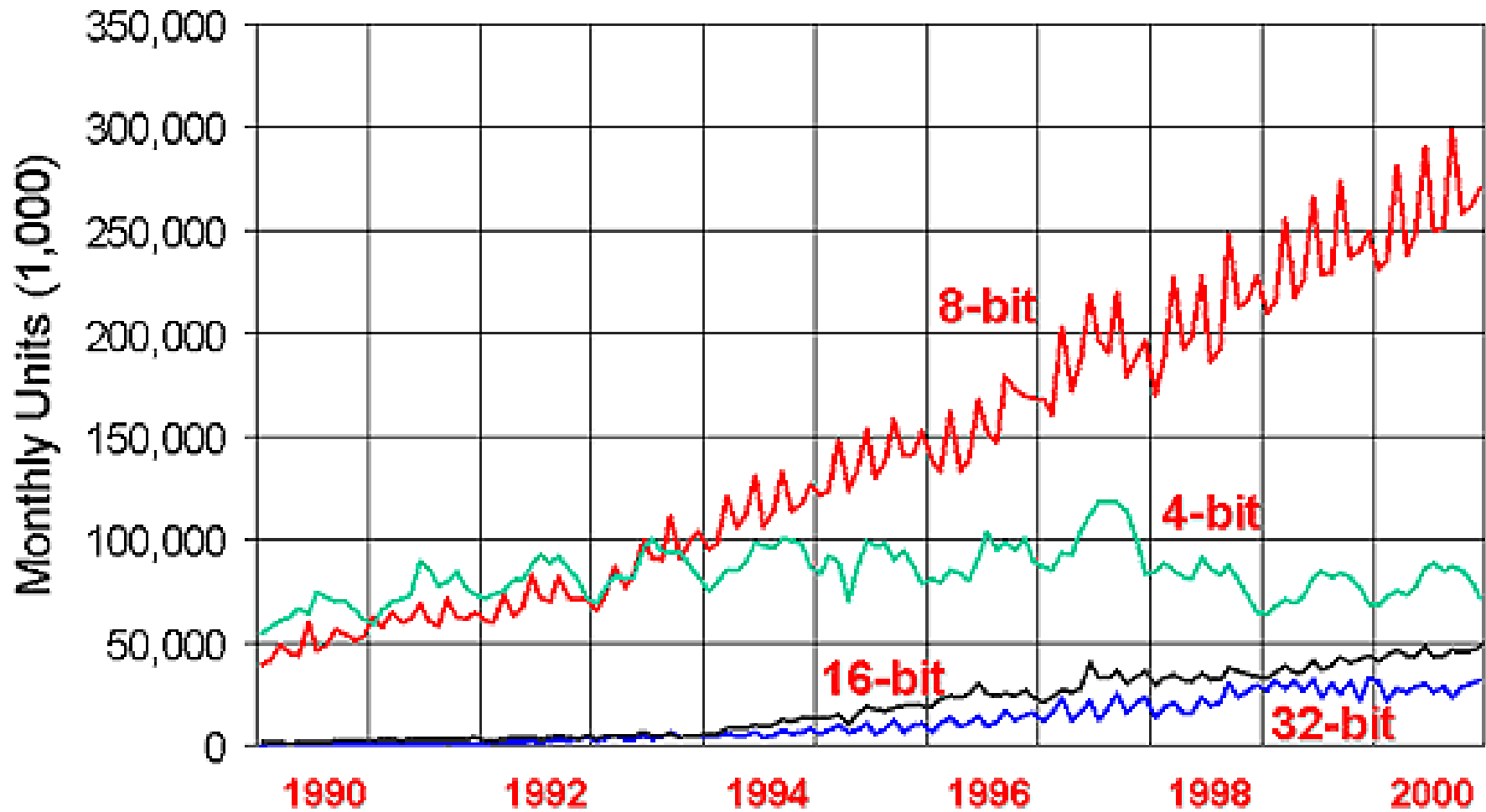Main CPU

[Cravotta05]   7

Microprocessor Unit Sales
All types, all markets worldwide

Source: WSTS

MCU Sales by Category

Legend: 4-/8-bit, 16-bit, 32-bit

Y-axis: Millions of Dollars (2,000 to 12,000)
X-axis: 10, 11, 12, 13, 14F, 15F, 16F, 17F, 18F
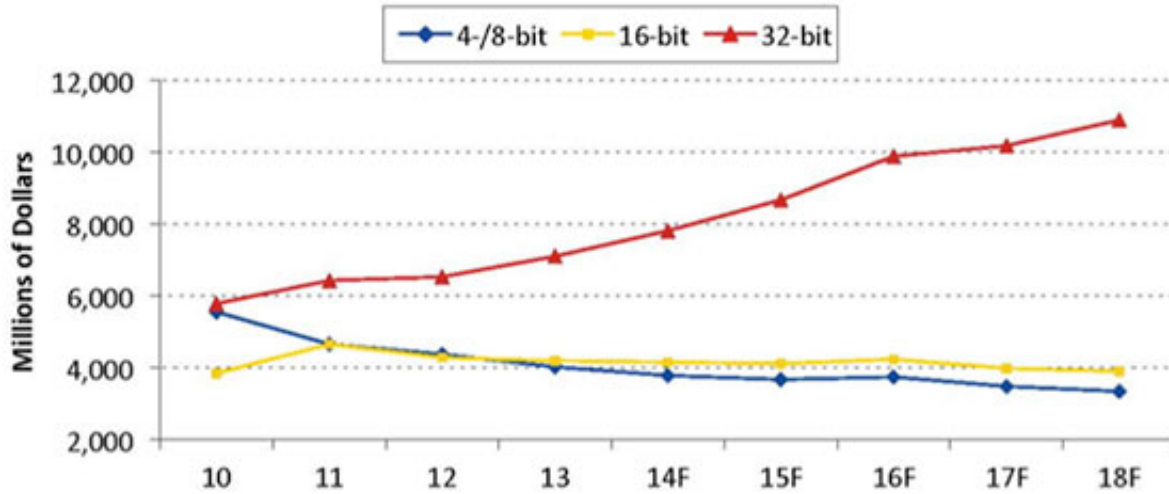
Dollars and units
tell a different story.

32-bit is NOT majority
by number of CPUs.

MCU Unit Shipments by Category

Legend: 4-/8-bit, 16-bit, 32-bit

Y-axis: Millions of Units (0 to 12,000)
X-axis: 10, 11, 12, 13, 14F, 15F, 16F, 17F, 18F
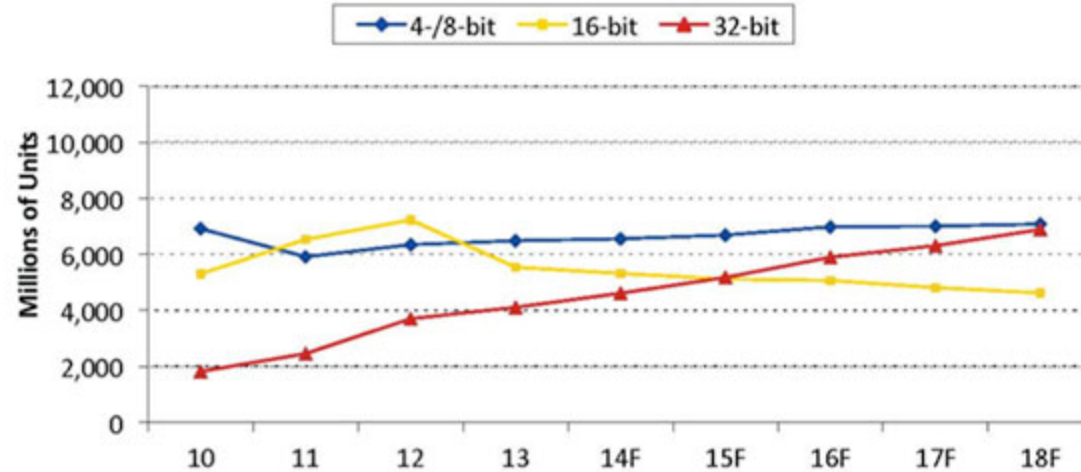
Source: IC Insights' *2014 McClean Report*

http://eecatalog.com/8bit/2014/01/31/microcontrollers-make-
waves-in-automotive-and-internet-of-things/#

Why 8-Bit MCUs Refuse to Go Away

New peripherals are paving the way for the continued success of the 8-bit microcontroller.

TABLE 1: Microcontroller Market, By Revenue		
1.	8-bit	39.7%
2.	32-bit	38.5%
3.	16-bit	21.8%
(Source: Gartner, Inc.)		

The answer is that 8-bit's success has little to do with it being, well, 8 bits. For many engineers, it's more about familiarity. That's why white goods makers employ it in refrigerators, freezers, washing machines, dryers and dishwashers. It's why automakers use it in window lifts, door locks, mirrors, seat motors, and interior lighting. It's why tens of millions of smart cards incorporate it, and why countless low-end motors are controlled by it.

"It's about legacy," Jim McGregor, founder and principal analyst for Tirias Research, told *Design News.* "The engineers have been using it so long that they just don't want to switch their software base."
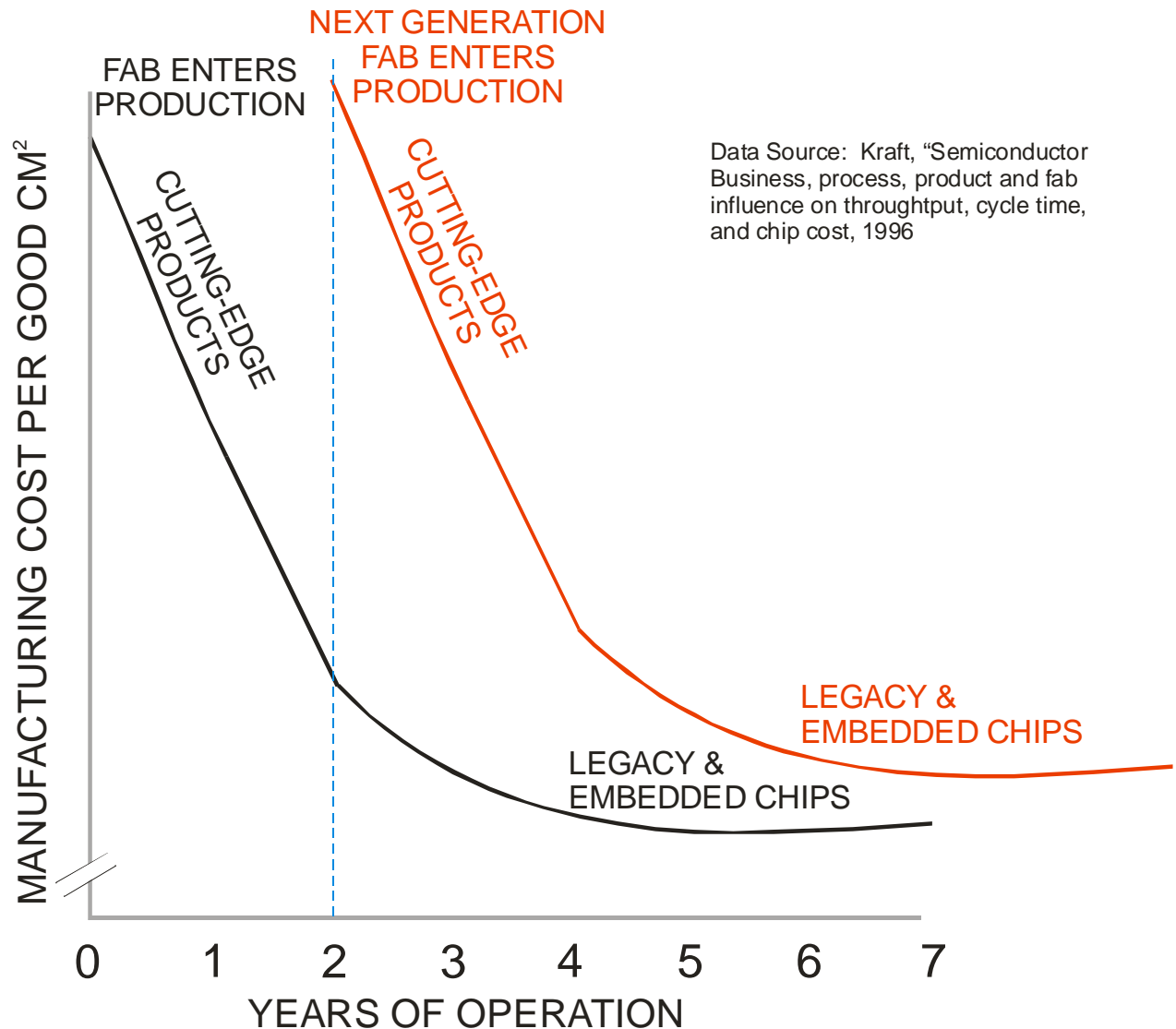
32 bit 2

8 bit 1

16 bit 3

Embedded Chips Tend To Be "Old" Technology

◆ **Semiconductor Fab Economics:**

- Cost per wafer processed drops when equipment is depreciated
- Yield increases with experience

◆ **Result: Chips get inexpensive right about the time they get obsolete**

- Y axis on this curve is about 50% cost reduction over first 2 years

FAB ENTERS PRODUCTION

NEXT GENERATION FAB ENTERS PRODUCTION

Data Source: Kraft, "Semiconductor Business, process, product and fab influence on throughput, cycle time, and chip cost, 1996

CUTTING-EDGE PRODUCTS

CUTTING-EDGE PRODUCTS

MANUFACTURING COST PER GOOD CM^2

LEGACY & EMBEDDED CHIPS

LEGACY & EMBEDDED CHIPS

0   1   2   3   4   5   6   7

YEARS OF OPERATION

Example Embedded Chip Motorola MC9S12A256

◆ **Medium-high end, 16 bit CPU**

- (Partly 16 bits because it has more than 64 KB of memory)

◆ **March 2002 chip introduction for automotive market:**

- - 256K bytes Flash memory
  - 4K bytes EEPROM (byte-erasable)
  - 12K bytes of RAM
  - Three synchronous serial peripheral interfaces (SPI)
  - 25 MHz HCS12 core (40 ns minimum instruction cycle)
  - 16-channel, 10-bit analog-to-digital converter (ADC)
  - 8-channel, 16-bit timer with capture and compare
  - 4-channel, 16-bit or 8-channel 8-bit pulse width modulator (PWM)
  - Two asynchronous serial communication interfaces (SCI)
  - One I^2C communication port (IIC)
  - Low electromagnetic emissions and RF noise susceptibility

- $11.88 in 10,000 quantity

- $9.13 for version with 128 KB flash, 2KB EEPROM, 8KB RAM
  – (Prices indicate memory is almost half the cost of the larger chip)

Hardware Quantity Pricing

- **Hardware costs reduce with high volume**
  - Typical to have price reductions at 10, 100, 1K, 10K, 100K, 1M units/year

- **Factors that drive volume pricing**
  - Lower cost to make the sale and maintain customer relations
  - More efficiency in administration, shipping
  - Better yield when single chip is made on a large run in fab

- **Capital equipment costs dominate semiconductor fab costs**
  - A $2 billion fab depreciates at the rate of $45,630 per hour for five years
  - Large orders reduce risk of expensive idle time at fab

- **So, normally you'd want to buy lots of chips at one time**
  - But, if you have masked ROM this can be a problem if you need to change SW
  - (Recall that in masked ROM a metalization layer is used to program the ROM)

What About Software Costs?

◆ **High-end vehicle costs approaching 40% for electronics and software**

  - "90% of all innovations driven by electronics and software"
  - 70 Electronic Control Units and 5 system buses
  - 50-70% of development costs are related to software

Automotive SW Workshop
San Diego, USA
H.-G. Frischkorn
BMW Group
10 -12. Jan. 2004
Page 3

Automotive Software
An Emerging Domain

Self-Driving Cars

Electronic Injections
Check Control
Speed Control
Central Locking
...

Electronic Gear Control
Electronic Air Condition
ASC Anti Slip Control
ABS Anti Blocking Sys.
Telephone
Seat Heating Control
Autom. Mirror Dimming
...

Navigation System
CD-Changer
ACC Active Cruise
   Control
Airbags
DSC Dynamic
   Stability Control
Adaptive Gear
   Control
Xenon Light
BMW Assist
RDS/TMC
Speech Recognition
Emergency Call
...

ACC Stop&Go
BFD
ALC
KSG
42-Voltage
Internet Portal
GPRS, UMTS
Telematics
Online Services
Blue-Tooth
Car Office
Local Hazard Warning
Integrated Safety
System
Steer/Brake-By-Wire
I-Drive
Lane Keeping Assist.
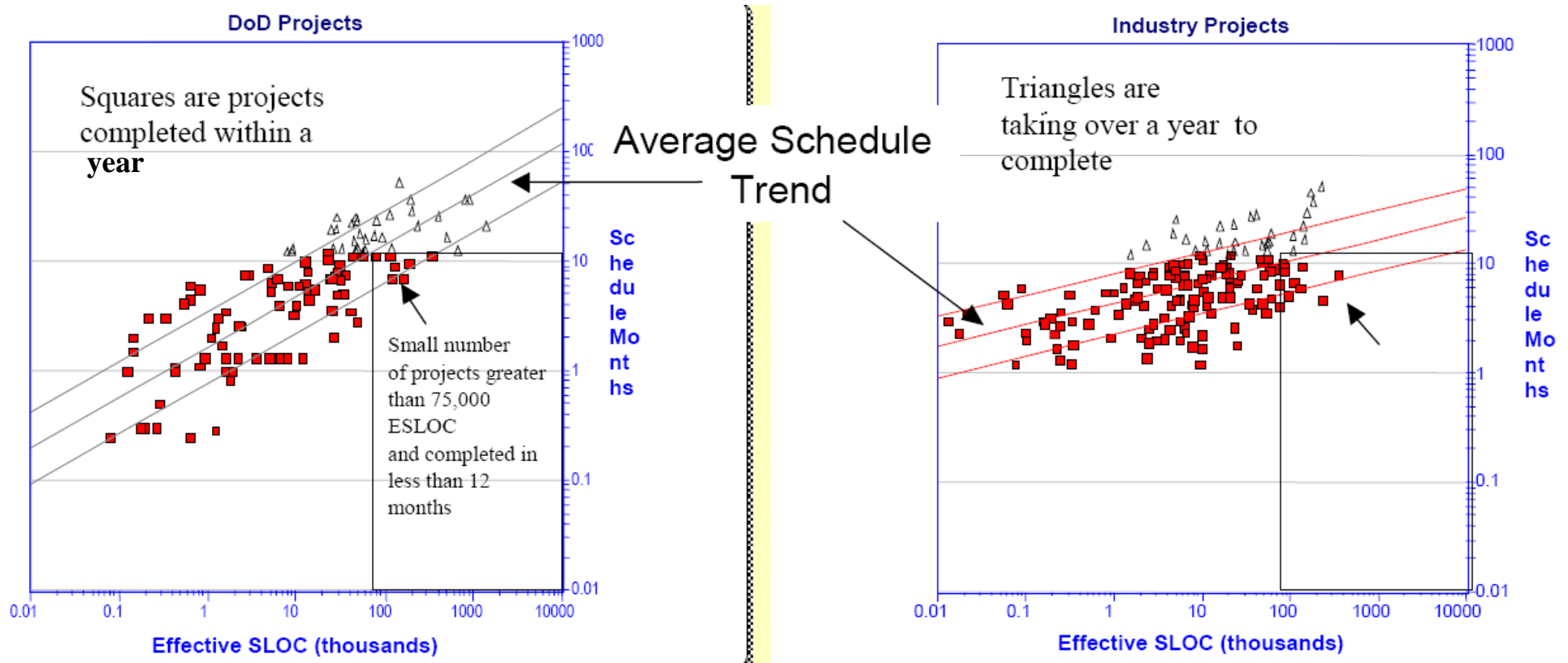Personalization
Software Update
Force Feedback Pedal

1970   1980   1990   2000

Degree of Interdependence

14

Productivity: 8-16 SLOC Per Day (1-2 per hour)

DoD Projects

Squares are projects completed within a **year**

Average Schedule Trend

Small number of projects greater than 75,000 ESLOC and completed in less than 12 months

Schedule Months

Effective SLOC (thousands)

Industry Projects

Triangles are taking over a year to complete

Schedule Months

Effective SLOC (thousands)

http://www.qsm.com/Develop_12%20months.pdf

◆ **Bigger projects take longer no matter how many people work on it**

◆ **For real-world embedded products, my experience is:**

- 8 to 16 Source Lines Of Code per day (1-2 SLOC/hr)
- I've seen 23 SLOC/day of good code from agile methods   (3 SLOC/hr)
  - But with minimal paper design package

15

Software Costs

◆ **"Firmware is the most expensive thing in the universe"**
– Jack Ganssle

◆ **Typical embedded software costs $15 - $40 per line of code**

- Do the math: $100,000/yr; 2000 hrs/yr; 2 SLOC/hr ➔ $25 / SLOC
  - (Note: a $65K engineer might cost $100K with benefits and *no* overhead)
- Defense work with documentation is $100/line
- Space shuttle code perhaps $1000/line
- Safety critical X-by-wire code probably is more expensive than defense code

◆ **Possible costs for software below (SLOC = Source Lines Of Code)**

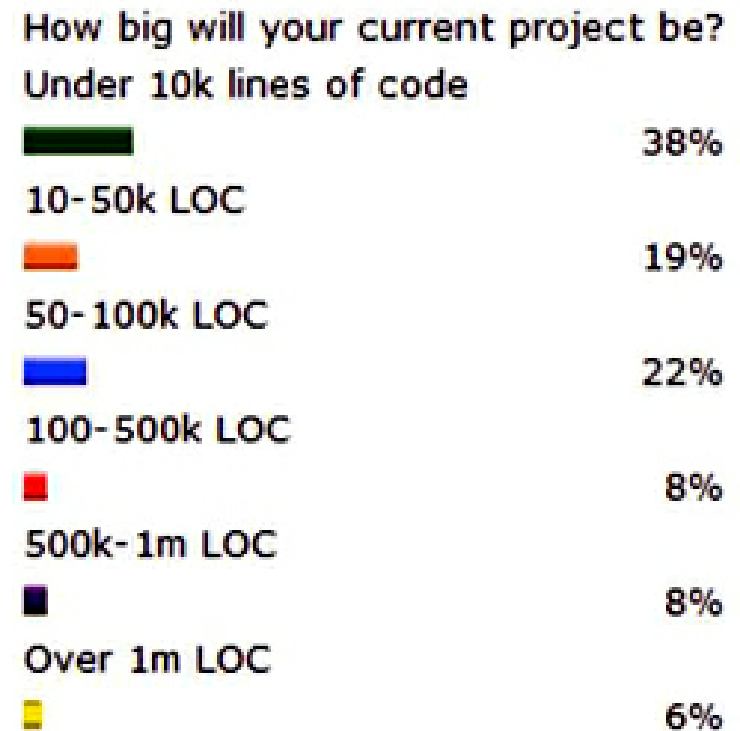| Program Size	$30 / SLOC	$100 / SLOC	$1000 / SLOC
1,000 SLOC	$30K	$100K	$1M
10,000 SLOC	$300K	$1M	$10M
100,000 SLOC	$3M	$10M	$100M
1,000,000 SLOC	$30M	$100M	$1B

Current Automotive SW Cost Models

◆ **Current cost model in many automotive situations**

- Costs are based on hardware costs + assembly + overhead
- For example, OEM might pay $6 for something that has $5 of HW costs

◆ **Unrealistic assumptions:**

- Engineering costs are free (a small, fixed fraction of hardware costs)
  - Software is free (a small, fixed fraction of hardware costs)
- Engineering changes after initial design will be minimal
- Electronics & mechanical engineers can handle software with no special training

◆ **Example simplistic calculation:**

- 10K SLOC on $100 ECU at $100 per SLOC
- Assume 500K units produced with annual model changes

$$CostPerItem = RE + \left(\frac{NRE}{\# Items} \right) = \$100 + \left(\frac{\$1M}{500K} \right) = \$102$$

Software Size Trends

◆ **Memory chip sizes increase by factor of 4 about every 3 years**

- Rule of thumb: memory is always "full"
- Conclusion: software increases in size by a factor of 4 every 3 years
- (Perhaps not strictly true in terms of "SLOC", but certainly software gets bigger every year!)

◆ **This is a fundamental driver of embedded system economics**

- Hardware prices drop slightly (with slightly better I/O integration)
- Available program memory grows exponentially

◆ **Chevy Volt has 10M lines of code**

- Includes RTOS and libraries
- Model-based synthesis for some

How big will your current project be?

Under 10k lines of code	38%
10-50k LOC	19%
50-100k LOC	22%
100-500k LOC	8%
500k-1m LOC	8%
Over 1m LOC	6%

Total Responses: 157 [Ganssle08]

But, What If You Mis-Estimate Software Cost?

◆ **Bigger piece of software – original estimate: 100K SLOC at $100**

$$CostPerItem = RE + \left(\frac{NRE}{\#\,Items} \right) = \$100 + \left(\frac{\$10M}{500K} \right) = \$120$$

◆ **What if it's 150K SLOC?**

$$CostPerItem = RE + \left(\frac{NRE}{\#\,Items} \right) = \$100 + \left(\frac{\$15M}{500K} \right) = \$130$$

◆ **What if that 150K SLOC costs $200 per SLOC?**

$$CostPerItem = RE + \left(\frac{NRE}{\#\,Items} \right) = \$100 + \left(\frac{\$30M}{500K} \right) = \$160$$

◆ **As NRE becomes a non-trivial portion of total cost, risk increases**
- Factor of two over-runs on software cost happen all the time
- For 500K units, $160 cost is a $20,000,000 over-run if bid at $120
- It's important to get engineering estimates right!

What About Design Changes?

- **Chips cost less in large volume**
  - Let's say a microcontroller is $5 in lots of 500,000 units with masked ROM
  - So, you buy 500,000 units to get the best price….
  - And after 3 months you need to change the software (bugs; requirements; …)

- **What happens if you have to throw away 9 months of 500K chips?**
  - 500K * $5 = $2,500,000
  - Toss 500K * 9 / 12 = 375,000 chips => $1,875,000
  - Also, the new 375,000 chips will probably be a little more expensive
  - AND, there will be a delay getting new chips due to time to make a new mask

- **This is why many automotive chips use flash memory**
  - Changes sometimes occur every *week*
  - So companies use flash memory, but NOT to change software in the field
    - Rather, to avoid wasting inventory and delays getting replacement chips
    - This is true even though flash memory is more expensive per unit
  - *RARE to see custom VLSI in long-lived complex networked applications!*

What About Limited Resources?

◆ **Embedded systems are often resource constrained**

- How much does it really cost to stuff things into a small system?

◆ **The numbers we'll discuss are examples**

- Based on best available data
- But, there isn't that much data, so consider this discussion an illustration rather than an exact answer


http://www.fiftiesweb.com/fashion/cramming.htm
(22 people at St. Mary's College, CA, 1959

Should You Optimize Hardware Cost?

◆ **If HW costs are seen to dominate, it's tempting to minimize them**
- If memory is half of microcontroller cost, then minimizing memory is important

◆ **Example: say you are using 115KB of Flash Memory:**
- Assume on MC9S12A256 chip; $11.88
  - 256KB flash memory
  - 4 KB EPROM
  - 12KB RAM

- Attempt to save money by substituting the smaller version at $9.13
  - 128KB flash memory
  - 2 KB EPROM
  - 8KB RAM
  - *Still leaves 13KB of flash free*

◆ **Potential HW savings:**
- 500,000 * ($11.88 - $9.13) = $1,375,000 savings per year
- But, this might be a bad idea…

The True Cost Of Full Resource Utilization
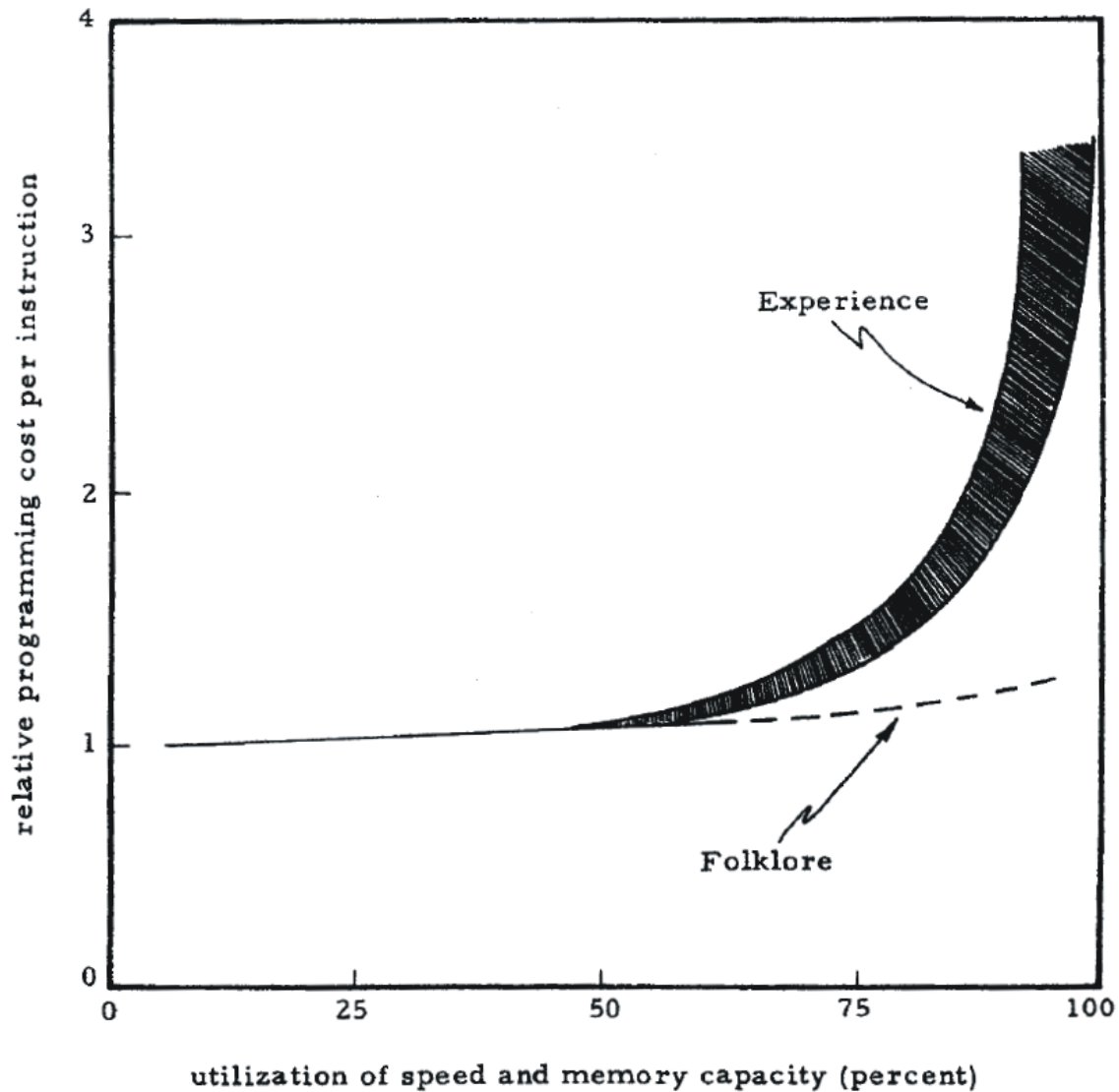
Boehm, "The High Cost of Software", 1975

Figure 5. Hardware Strains Cause Major Software Impact

True Cost Of Minimal Hardware Resources

◆ **Assume 25,000 SLOC for 115KB at $100/line = $2.5M unadjusted**

◆ **Total cost with 256KB Flash:**
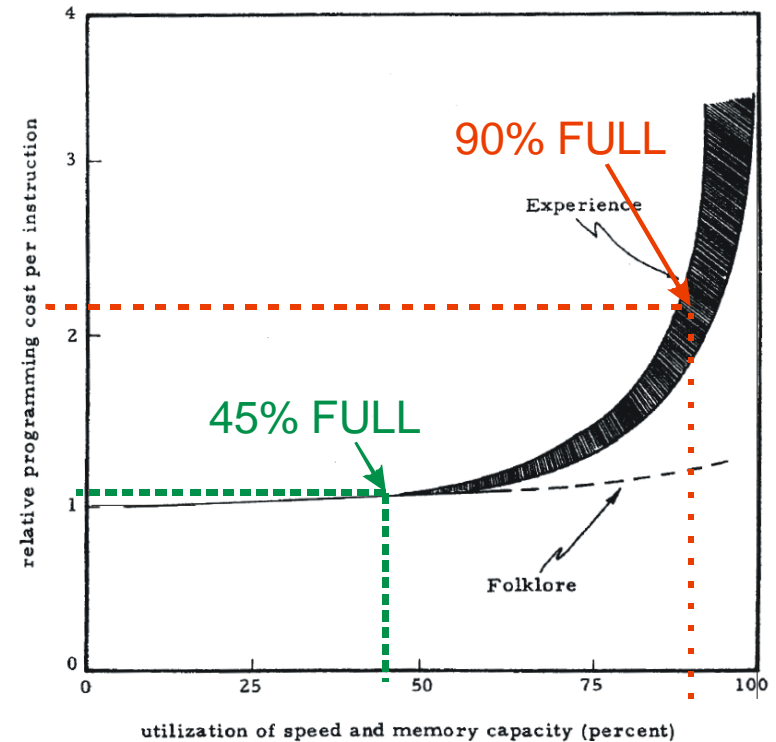 - 115KB / 256 KB = 45% full
 - Cost factor of 1.08

$$ECU\_Cost = \$11.88 + \left(\frac{1.08 \bullet \$2.5M}{500K} \right) = \$17.28$$

◆ **Software cost with 128KB Flash:**
 - 115KB / 128 KB = 90% full
 - Cost factor of 2.16 factor

$$ECU\_Cost = \$9.13 + \left(\frac{2.15 \bullet \$2.5M}{500K} \right) = \$19.88$$

 - Total cost increases by: $1,300,000 in this example using the cheaper chip

90% FULL

45% FULL

Experience

Folklore

relative programming cost per instruction

utilization of speed and memory capacity (percent)

Boehm, "The High Cost of Software", 1975

Figure 5. Hardware Strains Cause Major Software Impact

Why Are Full Resources Expensive?

◆ **Full memory:**

- Makes it difficult to include instrumentation & diagnosis scaffolding
  - Tracking down bugs is difficult if you have no room to move
  - Forces omission of run-time error captures that catch rare bugs
- Makes development difficult (no room for test scaffolding in target system)
- Increases need to compact/compress data structures
  - Example: packing data into bit fields instead of one byte per flag value
- Promotes excessive use of "clever" structure to reduce program size
- Makes it more difficult to deploy maintenance updates (no room to grow)

◆ **Full CPU & network capacity**

- Complicates scheduling meeting real time deadlines
- Encourages use of low-level languages or compiler-specific coding styles
- No spare capacity for new features

◆ **Other things can fill up too…**

Resources To Monitor

◆ **Using too much of any of these resources can cause problems:**

- CPU processing power
- RAM
- Non-volatile memory (Flash, EEPROM, etc.)
- Cycles of reprogramming Flash etc. (special effort to avoid wear-out)
- Disk space
- Battery life (special efforts to reduce power consumption)
- Thermal output (special efforts to avoid overheating)
- Network bandwidth
- Available circuit board real estate
- Number of bits of A/D converter (e.g., 11 bit accuracy on a 12-bit A/D)

◆ **Rule of thumb for US Military-grade systems:**

- Leave 50% unused capacity in all dimensions when system is first delivered
- Your reality will vary – but near-100% is always a bad idea

Why Code In Assembly Language

◆ **Why code in assembly language?**

- Often can get smaller program
- Often can get higher speed
- Sometimes necessary for very specific hardware interactions
- Sometimes necessary to fit in a too-small processor
- Sometimes there are no good compilers available for a tiny microcontroller

- Of course per previous slides you should never get to this situation …
  … but in the real world it happens all the time!

◆ **Why would you need smaller program, higher speed even after you heard this lecture?**

- Pre-selected CPU that is too small for the job
- Maximum clock speed limit because of RFI (to avoid RF shielding expense)
- C compiler advertised to be fantastic by vendor turns out to have problems
- …

Cost of Coding In Assembly Language

◆ **Raw cost of coding in assembly language**

- Generally, productivity in lines of code per day is language independent
- Assume that each line of C produces 5 lines of assembly language
  (this is of course very CPU dependent), but you're clever and only need 4 lines
  programming by hand when you save space.

- 625 lines of C   @ $200/line   = $125,000
- 2,500 lines of assembly   @ $200/line   = $500,000
- AND, there are likely 400% or 500% as many defects at same defect density!


◆ **Switching to assembly language to buy a cheaper processor:**

- Example: Assume $2.75 CPU with 8KB flash; $1.75 CPU with 4KB flash
- Save $1 per CPU;   break-even NRE vs. RE point is 375,000 units
- BUT, there are other hidden costs:
  - Portability
  - More defects!
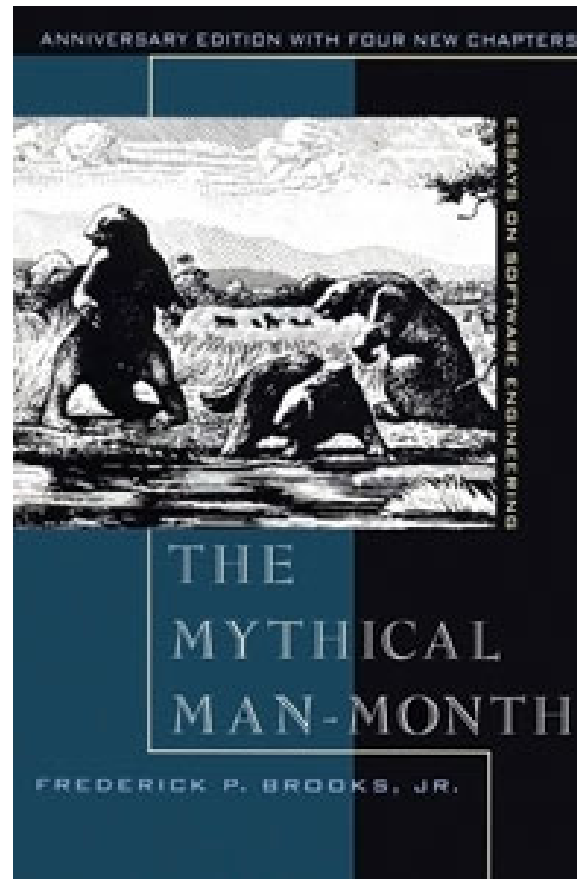 - Harder to trace code to design documents

Nevison's "Rule of Fifty"

◆ **No matter how many hours you work, you only get 50 useful hours per week of results**

- Except for short term projects (perhaps no more than 4 weeks)
- Why waste everyone's time by making them put in longer hours?
- [www.projectsolutions.com]
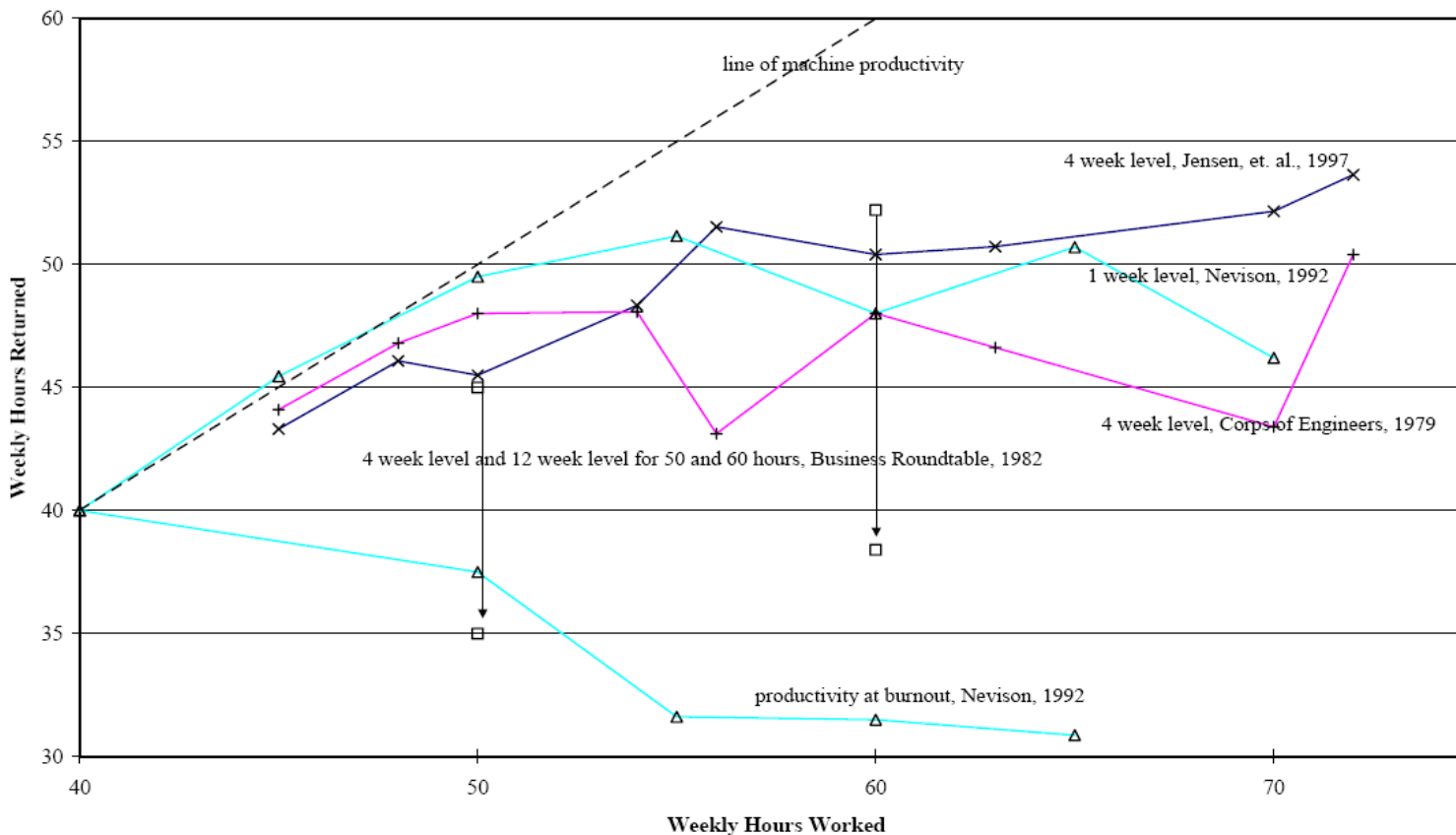
Classic Book:
The Mythical Man-Month
Fred Brooks

Summary

◆ **Hardware economics**

- Embedded systems get low costs by using somewhat old, high volume hardware

◆ **Software economics**

- "Firmware is the most expensive thing in the world"
- **Need to get smart about software, because that's where the costs are**
- Spending hardware $$$ to save software costs is often a smart move

◆ **Performance margin**

- Try to leave 50% performance margin when you can
- At or near 100% usage of any resource, expect to suffer severe pain and costs
- Using assembly language to optimize performance usually a bad idea

◆ **Open, difficult question: charging for software**

- Creativity required to charge for software if customers are used to paying based on hardware component content