Panel Session:
Will software save Moore’s Law?

A software engineer’s perspective

21 Sep 2005

W. Bail

The MITRE Corporation &

PEO IWS

MITRE

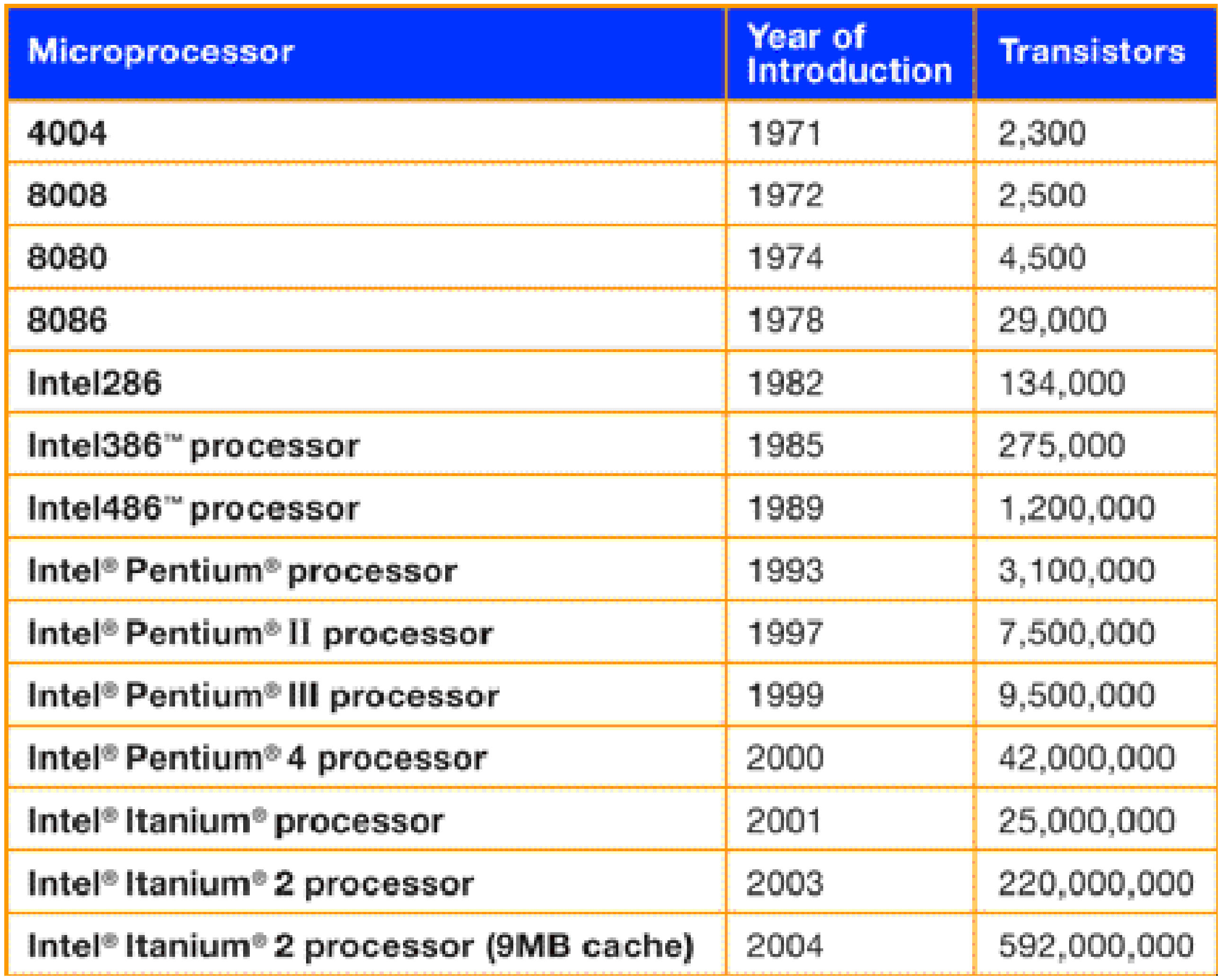
Moore’s Law???

Moore, Gordon E. “Cramming more components onto integrated circuits”. *Electronics*, Vol 38, No. 8, April 19, 1965.

“Integrated circuits will lead to such wonders as home computers. or at least terminals connected to a central computer automatic controls for automobiles, and personal portable communications equipment. The electronic wristwatch needs only a display to be feasible today.”

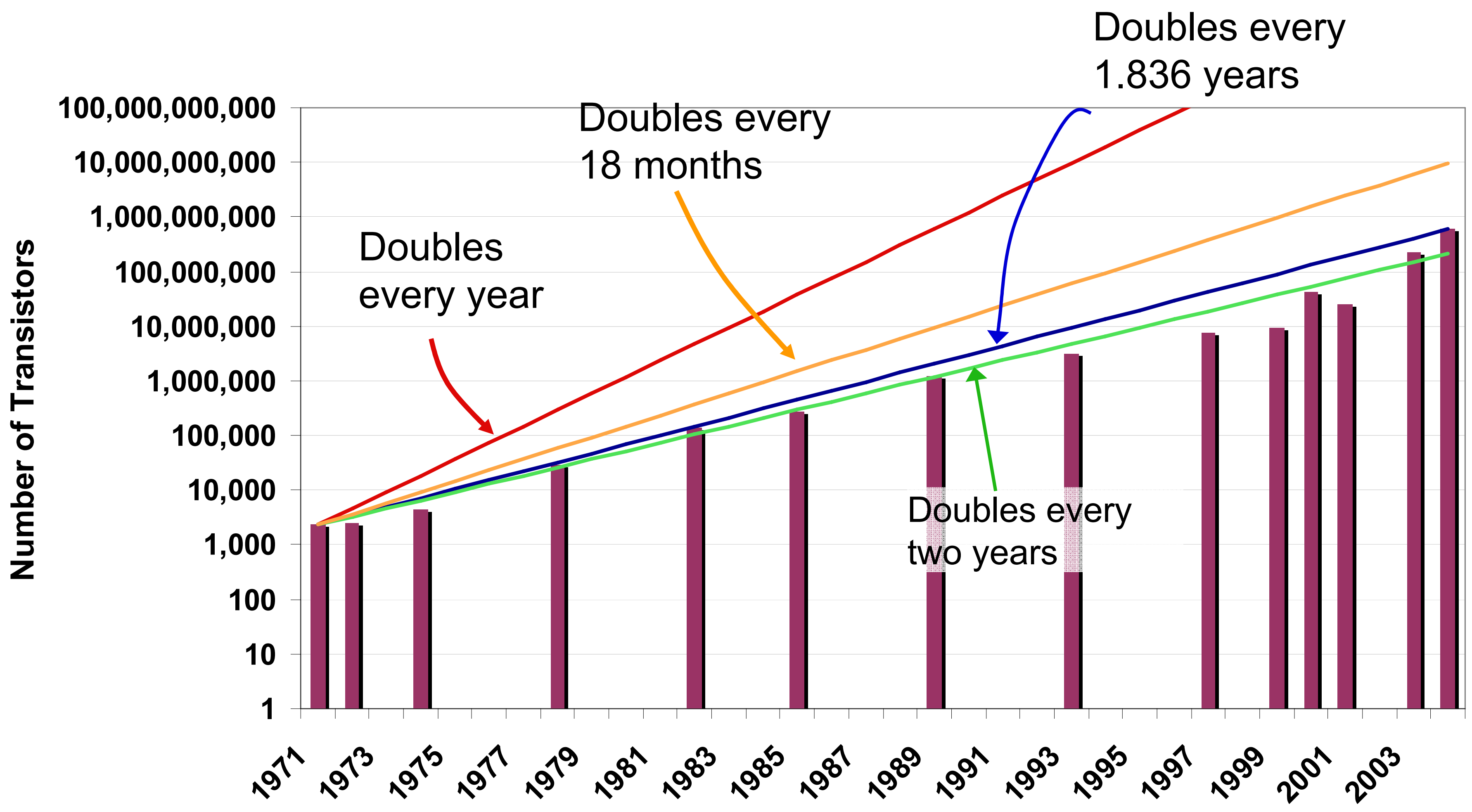
“The complexity for minimum component costs has increased at a rate of roughly a factor of two per year (see graph on next page). Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000.

Some data – Moore’s Law Transistor Count Chart



Microprocessor	Year of Introduction	Transistors
4004	1971	2,300
8008	1972	2,500
8080	1974	4,500
8086	1978	29,000
Intel286	1982	134,000
Intel386™ processor	1985	275,000
Intel486™ processor	1989	1,200,000
Intel® Pentium® processor	1993	3,100,000
Intel® Pentium® II processor	1997	7,500,000
Intel® Pentium® III processor	1999	9,500,000
Intel® Pentium® 4 processor	2000	42,000,000
Intel® Itanium® processor	2001	25,000,000
Intel® Itanium® 2 processor	2003	220,000,000
Intel® Itanium® 2 processor (9MB cache)	2004	592,000,000

Did it happen that way?



Intel processor history

Overall question

- ***Back to the question - Will software save Moore's Law?***

- **We can't even save ourselves…what do you want from us?**
- **Our track record of developing large, complex, software-intensive systems has not been exemplary**
- **And the expectations for what SW does keep increasing**
- **We have enough problems getting our own act together**

Current focus of software

- Speed up SW development – greater productivity
- Reduce the cost of developing complex SW systems
- Make better software
  - Avoid introduction of defects into products
  - Facilitate finding and removing defects from products
- Many needed SW features slow the system down
  - Fault tolerance
  - Modular designs
  - Reusable components
  - Portable software
  - COTS OSs and middleware
  - (Very) high-level languages

Hardware support

- We have relied on hardware to help us deal with increased SW complexity
  - Breathing room
- We often discover that our systems do not fit into the box
  - Particularly after we add features the customer wants
- So we try to optimize the software in various ways
- But we still relied on Moore's Law to keep us going
- Our systems have grown in size disproportionately to HW growth
  - ref: Bob Bond

Strategies to make SW faster

- Optimize source code
  - *e.g.*, restructure based on knowledge of run-time models
  - *e.g.*, replace with assembler code
- Optimize algorithms
  - Semantics-preserving transformations
  - Reduction of functionality (less precise,…)
- Transform software from sequential to parallel
  - At coarse and fine-grained levels
- Optimize translation – source code $\Rightarrow$ executable
  - Optimizing compilers
  - Parallelizing compilers

Limits

- But there is a limit to what can be done by SW on existing processing resources
- For any given processor architecture/system, will approach limit asymptotically
- But we can
  - Wait for faster processors
  - Switch to ASICs and FPGAs

What can we do?

- The goal is to process more information in less time
- With processors following Moore's Law, SW could add complexity and stay even or get ahead
  - Using fairly straightforward techniques
- But if/when the slope trails off, SW needs to take a more proactive approach
  - Be ready to exploit radical new architectures
  - Be flexible enough to break out of von Neumann-style computation
  - Do not wait for HW to do all (most of) the work

What is needed?

- Exploitation of model-driven techniques
  - That provide for rapid retargeting to alternate computing architectures
- Next generation of compiler techniques
  - Able to accommodate radical new processing strategies, flexibly and quickly
- New language designs that
  - enhance domain-specific programming
  - reflect common algorithm clichés
  - provide features to programmers for advanced optimizations (*e.g.*, parallelization)
- If SW technologies cooperate with HW technologies, total computing can continue to increase