Computer performance/cost has followed an exponential path since the 1940s, doubling about every 18 months. This has enabled increasing function/performace at constant cost ... or decreasing cost at constant function/performace.

Long term trends
Moore's Law

For three decades, Moore's Law applied to microcomputers has made life easier - more transistors, faster clocks... But almost every known technique was applied to use up the transistors!

Over the 1990s there was little architectural innovation, but almost every known technique was applied to keep on the long term trend, we will need to innovate... especially in architecture, design and software...

This is changing - to keep on the long term trend, we will need to innovate!
Increasing performance has involved increasing clock speeds and using more transistors (Moore’s Law) but increasing the processor clock speed and using more transistors does not always provide a corresponding increase in performance.

We are well behind Moore’s law on achieved general purpose performance because the speed of the wires can’t keep up with the transistors.

Increasing performance...
Decreasing cost has had a dramatic impact allowing computers to be embedded in an increasing variety of products (cars, telephones, cameras, games, toys...)

New applications will continue to emerge as a result of improvements in performance/cost - reduce the cost by factor 10, increase the volume by factor 100.

Commercially - and socially - decreasing cost has more effect than increasing performance.

Note: Disposable Computing started around 2000!
There's little evidence to support architectural decisions together!

... computers, compilers, languages - and applications evolve

... and to make matters worse

... we are often fooled by averages

... we don't seem to have a good analysis of what programs do

Computer Architecture
In general purpose architectures, we have been more scope for incremental enhancement - trying to increase the performance of an existing software base. What about ideas that can't be represented easily in existing programming languages? Concurrency? Input and Output? What about new applications? In embedded architectures, there has been more scope for innovation.

Computer Architecture
General purpose microprocessors

- Faster and faster clocks
- Deeper and deeper pipelines
- More and more execution units
- More and more strange instructions
- Bigger and bigger branch prediction/recovery mechanisms
- Longer and longer context switches
- More and more stuff offloaded to hardware gadgets
- Faster and faster clocks
General purpose performance relies on uniform access to a random access memory.

A complex memory hierarchy is an approximation to fast sequential processing.

In the same way, Instruction Level Parallelism provides an approximation to fast sequential processing. Some programs work well; some don’t.

Maybe we’ve made these processors too fast!
A lot of effort is going into big multi-core chips

**General Purpose Computing**

It's not clear what these are for

Predictably these focus on symmetric multiprocessors with complex memory hierarchies

- we will need big servers - but these can be built from larger using thin clients and portables
- we won't need - or want - big processors in PCs - we'll be numbers of slower, more power-efficient processors

David May 10 Cambridge December 2005
Expectations vs. Reality

Symmetric multiprocessors are easy to program... provided you're not bothered about performance!

Multiprocessors need concurrent programs and/or parallel algorithms - but then the architecture can be much simpler!

As a single general purpose processor places a heavy demand on its memory system, you can't expect it to support several processors...

... but then the architecture can be much simpler!

Symmetric multiprocessors are easy to program... provided you're not bothered about performance!
The Exabyte Effect

The Internet in 2010:

- Exabytes everywhere!

- Petabytes servers, petatflop supercomputers, a billion hosts, a.

- Wearables

- Services to ubiquitous and nomadic clients including wearables

- Many different forms of content - visuals, soundscape...
The Exabyte effect

We will see changes in Internet infrastructure aiming to maximise energy efficiency.

The trend towards Ethernet as the standard interconnect will be opened up by the trend towards more, lower-speed processors...
We're going to need architecture for exabytes. Efficient data representation... arithmetic and logic on big numbers... accurate numerics on big numbers... big addresses...
Long arithmetic

To multiply two n-word integers we have to execute n² multiply instructions - if we have the right instructions! And doubling the wordlength reduces the multiply operation count by factor 4. So (eg) a 64 bit CPU with the right instructions will be 10-100 times faster than a 32 bit CPU with the wrong one!

Tomultiplytwon-wordintegerswehavetoexecute \(n^2\) multiply instructions - if we have the right instructions!
Long wordlengths

Let's think about a machine with 128-bit registers:...
Computing without power
We have the potential for a new generation of sealed devices—without the need for wires or replaceable batteries.

A disruptive technology: sentient, communicating devices will be everywhere—in environmental control, industrial monitoring, tracking, implants, packaging... RFID is just the beginning.

They depend on low-energy computing and communications

- low-cost batteries or scavenged power
- embedded sensors and actuators
- low-energy computing and communications

Computing without power
Wearables

We can embed technology in clothes - or wear it like jewelry!

Technologies include audio, cameras, accelerometers and... 

Applications include sports, healthcare, lifestyle, leisure...

...gyros, GPS, RFID

But the big markets will probably be in fashion!

Will the design houses play a major role in the next wave of electronic devices?

Will the design houses play a major role in the next wave of...
What does low energy involve?

Low voltage circuits

Low power logic design

Dynamically switching off stuff when it isn’t in use

Nominal operations including data transfers

Event driven systems including software

Minimal operations including data transfers

Dynamically switching off stuff when it isn’t in use

Low voltage circuits

Low voltage circuits
Software

May’s Law: Software efficiency halves every 18 months, compensating Moore’s Law

- A mixture of
  - Reliance on Moore’s law to solve inefficiency problems
  - Massive overuse of windows and mouse-clicks
  - Copy-paste programming
  - Adding too many features
  - Shortage of skills

David May 20 Cambridge December 2005
high-performance systems and supercomputers •
wearables •
ubiquitous systems •

This is a big opportunity - efficient software can add a lot of value by increasing performance and power-efficiency in ubiquitous systems, wearables, and high-performance systems. When it’s full of bugs, it’s too big and complex to understand, together with an extreme reluctance to re-write software. 

Software
Software and Algorithms

In ubiquitous systems, halving the instructions executed can double the battery life! For \( N = 30 \text{ billion} \), this change is as good as 50 years of technology improvements. A dramatic effect when \( N \) is large:

Reducing the number of operations from \((N)^{8.01} \times N\) to \(N \times N\) has a significant impact.

And big data sets bring big opportunities for better software and algorithms:

Double the battery life in ubiquitous systems, halving the instructions executed can

Software and Algorithms
Generic products

State-of-the-art, design, verification and manufacturing set-up. Costs are escalating...?

This suggests a move to:

- generic programmable and/or reconfigurable chips
- generic platforms, customizable at low cost

Potential opportunity for new industry...

State-of-the-art design, verification and manufacturing set-up...?
Manufacturing companies will not be able to design a fraction of the ASICs made possible by generic platforms. So there's space for new companies which do ASICs purely by programming and configuring platforms - and by selling the programs and configurations, not by selling the platforms themselves. These companies will have specialised, high-value applications and market expertise but will not require huge investment and will not need as much infrastructure as a traditional manufacturing company.

David May 24 Cambridge December 2005
Expose it and exploit it - don't hide it

- in systems on a chip
- in collections of processors
- inside processors

Concurrency, concurrency, and concurrency

Innovations in Architecture
Concurrency

Large collections of processors were successfully employed in a few computers in the 1980s but have only just come into widespread use: few computers in the 1980s, and we are beginning to see chips full of processors.

... supercomputers - a few thousand processors.
... digital animation (typically 1,000 processors).
... Google search engine - tens of thousands of processors.
... digital animation (typically 1,000 processors).

... supercomputers - a few thousand processors.
They are potentially an important generic platform technology... require innovative programming tools... could deliver tera-op performance within 5 years... can exploit local clocks for high speed... are relatively easy to design, verify and test...
General purpose processor arrays

Two key issues:

1. Interconnect is not an 'overhead' - communication is the essence of many algorithms and applications.

2. Scalable interconnect that grows as $n \times \log(n)$, just like random access memory addressing.

- Subroutines - efficient serial re-use of parallel resources.
- Scalable interconnect - grows as $n \times \log(n)$.
Innovations in Design

We have been making hardware design tools look like programming languages for years, but we have failed to produce a single language which can target software, re-configurable hardware, and hardware from the same source.

So this is still a potential opportunity. It would have a big impact on design efficiency and the efficiency of designs.

But we have failed to produce a single language which can program languages for years.

Concurrent languages are the key!

David May 29 Cambridge December 2005
... a market for sensors, actuators and embedded intelligence

There is a market - vacuum cleaners, lawn-mowers, pets...

we have lightweight materials

we know how to do the sensors - even vision

we know how to build the control systems

... has finally come of age!

Robotics
and every one will need a microcontroller!

"and thousands of unnamed ones"

"... there are around 650 named actuators in a humanoid robot"

... or perhaps:

"thousands of unnamed ones"

"... there are about 650 named muscles in the human body and"

Scalable real-time control - computers in the loop:

Robotics
New technologies...

...and even battery technology is moving on - to flexible, printable devices...

...and with silicon, we can continue to reduce costs...

...plastic devices which are flexible - and printable...

...exotic technologies based on molecular structures...

...the 1940s and there are likely to be more...

There have been several jumps in computer technology since...
The long-term cost-performance improvement will continue. There is plenty of scope for new ventures. There is a big opportunity for innovation.

- Technology
- Applications
- Business models
- Architecture and software

Summary