# The Return of Innovation

David May

## Long term trends

Computer performance/cost has followed an exponential path since the 1940s, doubling about every 18 months

This has enabled

Increasing function/performance at constant cost

... or

Decreasing cost at constant function/performance

## Moore’s Law

For three decades, Moore’s Law applied to microcomputers has made life easy - more transistors, faster clocks ...

Over the 1990s there was little architectural innovation, but almost every known technique was applied to use up the transistors!

This is changing - to keep on the long term trend, we will need to innovate

... especially in architecture, design and software ...

## 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.

... because the speed of the wires can’t keep up with the transistors

We are well behind Moore’s law on achieved *general purpose* 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!

## Computer Architecture

There’s little evidence to support architectural decisions

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

... we are often fooled by *averages*

and to make matters worse

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

## Computer Architecture

In general purpose architectures, we have resorted to *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

## 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

## General purpose performance

... relies on uniform access to a *random access* memory

A complex memory hierarchy is an *approximation* to this

... some programs work well; some don’t

In the same way, Instruction Level Parallelism provides an *approximation* to fast sequential processing

Maybe we’ve made these processors too fast!

## General purpose computing

A lot of effort is going into big multi-core chips

Predictably these focus on symmetric multiprocessors with complex memory hierarchies

Its not clear what these are for

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

## Expectations vs. Reality

Symmetric multiprocessors are easy to program

... provided you’re not bothered about performance!

A single general purpose processor places a heavy demand on its memory system

... you can’t expect it to support several processors

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

## The Exabyte effect

The Internet in 2010:

- Exabytes everywhere!
- Petabyte servers, petaflop supercomputers, a billion hosts, a zetabyte online
- Services to ubiquitous and nomadic clients including wearables
- Many different forms of content - visuals, soundscapes ...

## The Exabyte effect

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

These will include

- more, lower-speed processors
- programmable accelerators for specialised tasks such as transcoding, cryptography, searching ...

The *accelerator-in-server* market will be opened up by the trend towards Ethernet as the standard interconnect

## Architecture for exabytes

We’re going to need

... big addresses

... arithmetic and logic on big numbers

... accurate numerics on big numbers

... efficient data representation

## Long arithmetic

To multiply two n-word integers we have to execute $n^2$ multiply instructions - if we have the right instructions!

```
   carryout, result := MUL op1, op2, carryin
```

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 ones!

## Long wordlengths

Let’s think about a machine with 128-bit registers:

... big integers: $2^{128}$ is around $10^{39}$

... $64.64$ format gives a range of around $10^{20}$ at $1$ in $10^{20}$ precision

... enough addresses for every byte - or bit - on the planet

... accurate double precision floating point multiply-accumulate

... powerful SIMD data representation

## 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
- embedded sensors and actuators
- low-cost batteries or scavenged power

## Wearables

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

Technologies include audio, cameras, accelerometers and gyros, GPS, RFID ...

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

But the big markets will probably be in fashion!

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

## What does low energy involve?

Low voltage circuits

Low power logic design

Dynamically switching off stuff when it isn’t in use

Minimal operations *including* data transfers

Event driven systems *including* software

## Software

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

A mixture of

- shortage of skills
- adding too many features
- copy-paste programming
- massive overuse of windows and mouse-clicks
- reliance on Moore’s law to solve inefficiency problems

## Software

... together with an extreme reluctance to re-write software

- when it’s full of bugs, is too big and complex to understand, and the authors have left (died?), it’s time to re-write it!
- or at least, it should be moved from the company assets to the liabilities!

This is a big opportunity - efficient software can add a lot of value by increasing performance and power-efficiency in

- ubiquitous systems
- wearables
- high-performance systems and supercomputers

## Software and Algorithms

In ubiquitous systems, halving the instructions executed can double the battery life

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

Reducing the number of operations from $N \times N$ to $N \times log(N)$ has a dramatic effect when $N$ is large

... for $N = 30\, billion$ this change is as good as 50 years of technology improvements!

## 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’ customisable at low cost
- ... ??

Potential opportunity for new industry ...

## The Fabless, IP-less chip company

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 result - *not* by selling the programs and configurations

These companies will have specialised high-value applications and market expertise but will not require huge investment

## Innovations in Architecture

Concurrency, concurrency and concurrency

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

Expose it and exploit it - don’t hide it

## Concurrency

Large collections of processors were successfully employed in a few computers in the 1980s but have only just come into widespread use:

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

and we are beginning to see

... chips full of processors

## Processor arrays

... are relatively easy to design, verify and test

... can exploit local clocks for high speed

... could deliver tera-op performance within 5 years

... require innovative programming tools

They are potentially an important generic platform technology

## General purpose processor arrays

Two key issues:

- scalable interconnect - grows as $n \times log(n)$ - just like random access memory addressing!
- subroutines - efficient serial re-use of parallel resources

Interconnect is *not* an ‘overhead’ - communication is the essence of many algorithms and applications

## 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 on the efficiency of designs!

Concurrent languages are the key!

## Robotics

... has finally come of age!

we know how to build the control systems

we know how to do the sensors - even vision

we have lightweight materials

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

... a market for sensors, actuators and embedded intelligence

## Robotics

Scalable real-time control - computers in the loop:

“... there are about 650 named muscles in the human body and thousands of unnamed ones ...”

or perhaps:

“... there are around 650 named actuators in a humanoid robot and thousands of unnamed ones ...”

and every one will need a microcontroller!

## New technologies

There have been several jumps in computer technology since the 1940s and there are likely to be more ...

... exotic technologies based on molecular structures

... plastic devices which are flexible - and printable

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

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

## Summary

The long-term cost-performance improvement will continue

There is plenty of scope for new ventures

There is a big opportunity for innovation

- architecture and software
- business models
- applications
- technology