

Moore's Law – Delaying forever?

'The important thing is that Moore's Law is exponential, and no exponential is forever... But we can delay forever' – Gordon Moore

Gordon Moore's highly influential 'law' concerning the rate of development in integrated circuit engineering has now remained true for nearly four decades.

In its original statement, Moore observed in the 1965 that the number of transistors that could be got on a chip doubled every 18 months. Since the density of features on an integrated circuit determines all the main parameters, including power consumption and maximum clock speed this is seen as a general measure of progress in semiconductor technology.

Moore's Law has held for 38 years, but recently its demise has been increasingly predicted. At several points over the last 20 years commentators have warned that we are on the threshold of the 'last few doublings' and the view has been expressed that before long those who write software will now have to take performance more seriously

Moore's Law has become more than a historical observation, and more than a mere prediction. Today it takes on the role not only of observation and prediction but also of a target for the industry and of goad. It has become such a cause celebre that works as a feedback mechanism and acts to fulfil its own prophecy.

Moore's Law was first proposed at the beginning of the integrated semiconductor era but something like it seems to be true even over the whole history of electronic computing. Influenced by a certain family connection I have taken the example of the Cambridge EDSAC1 as the first real practical computing machine – that is, a machine with real users. EDSAC ran at half a megacycle. If that speed had doubled nearly every two years since the EDSAC was first operational in 1949 we would have reached about the speed of a modern PC today.

Although the factors which I will argue help determine the speed of computer development have been there in some form from the early days, Moore's Law is normally taken as applying to the era of semiconductors, and most recently and consistently to CMOS (Complementary Metal-Oxide Semiconductor). This technology has been with us for 25 years and has not yet run its course, though many expect it to do so within a decade or two.

Industry factors

Compared with other technology industries, the semiconductor industry lacks competing goals. In most engineering fields a number of factors govern the nature of the product. These factors are mixed in differing proportions to yield an end products suitable for a variety of applications. A car may be small or large, high powered or economical, luxurious or utilitarian. Each of these factors: size, power, comfort, entail a cost, and one is traded-off against another to suit a particular purpose. Although semiconductor engineering is not entirely without such tradeoffs, they play a relatively minor part¹. In a processor chip all the important parameters: speed, power, size and cost per transistor go together. One is not had at the expense of another.

¹ A good example of a tradeoff is in the special low power chips which are used in the manufacture of notebooks. Here there is a genuine though temporary tradeoff between power consumption and speed.

One cannot have a slower processor at a lower cost, except at the edges where a slower technology has not yet given way to the next faster, cheaper one. These defining parameters boil down to a single factor, namely the size of the individual components and the density with which they can be deposited onto a chip. From the beginning of the integrated circuit in the 1960s this single technological problem has dominated all others.

The cost of developing and tooling each new reduced size technology runs into billions of dollars. The viability of a fabrication process is governed by yield, the proportion of chips that are manufactured without flaws², and this means that each new process must be tested in near production conditions before its viability can be known for certain.

The constant reduction of feature size drives a market that is largely an upgrade market. The commercial model depends on the manufacturers improving their products so radically every few years that customers will completely replace a proportion of their equipment. This cycle forms a stable, continuing process, so long as technological progress is sufficient to generate the upgrade revenue that will fund the next round of development. The nature of the industry therefore forces a certain minimum rate on development. If the process went too slowly activity would come in bursts and the necessary level of capitalisation could not be maintained.

At the same time a conservative pressure acts to limit this rate to that which will just fuel the upgrade cycle. For corporations like Intel, a balance must be struck between competitive edge and the minimisation of risk. They must remain a little ahead of their competitors at least in some areas, but not so much as to risk expensive failures due to over ambitious engineering targets.

In doing so the planners must judge nicely how much technical advancement can safely be made within the cycle, even though each new round brings different challenges, often involving quite fundamental research. In making these judgements they will naturally be influenced by experience, and Moore's Law will inevitably be used as the yardstick. No one will be fired for setting the hurdles in Moore's Law sized steps and to an extent Moore's Law will tend to maintain its own truth.

The course of the next two decades will be determined chiefly by the technological problems of making smaller chips. Fundamental theoretical limits on the speed of computing will some day impose a limit. But not on today's technology. The exact nature of these limits on computing are not well understood. The possibilities are bounded at least by the speed of signal propagation, and perhaps ultimately by considerations of space and time granularity. The speed of light is closest limitation as well as being the most tangible. Light travels about a foot in a nanosecond. For an Itanium running at 4GHz the path that a chain of signals can take within the chip and between successive clock cycles cannot exceed a few inches. This probably puts us within an order of magnitude or two of what can be achieved with a traditional semiconductor architecture, though still a long way off if we allow the possibility of gates of atomic size. To speculate wildly, switches composed of features closer to the plank length would provide even greater headroom. Another factor which is sometimes cited as limiting the possible speed of classical computation is the minimum dissipation of energy. Application of the 2nd Law of Thermodynamics suggests that in the limit, the gating action itself, generates heat occasioned by the loss of information³. However the energy needed to

² More correctly the term yield refers to the proportion of chips at or above a certain quality. The newly manufactured wafers are tested to ascertain the greatest speed at which they will run reliably. They are then sorted into a range of product grades.

³ It has been suggested that this difficulty could be circumvented by the use of fully reversible and hence information conserving gates.

drive any imaginable switch would be likely to be more significant. Present technology is a long way from such considerations.

CMOS

CMOS is now the single fabrication technology that has come down to us. A couple of decades ago there were several alternatives. At that time you could pay a little more and have a faster ECL logic chip, and bipolar technology was seen by many as the way forward. But in the 80s and 90s all the other contenders dropped out as CMOS steamed ahead.

The main advantage of CMOS over NMOS and bipolar technology is the much smaller power dissipation. Unlike NMOS or bipolar circuits, a CMOS gate has almost no static power dissipation. Power is only dissipated while the circuit is actually switching.

The fact that CMOS is the only viable technology at today's clock speeds puts the industry in a precarious position. If CMOS were to hit the buffers there would be nothing that could step in and take over. The most talked about alternatives, optical and quantum computing are not serious contenders. Quantum processors which remain highly speculative effectively address a different problem, and optical computers are not looking promising.

There are several reasons why this is so. Firstly the wavelength of the electron is very much shorter than that of light. This means that interference effects with the switching frequency will occur sooner in an optical computer than an electronic one. A more important factor is that while electrons interact readily with matter, photons do not and it is difficult to obtain any effect that might be useful for gating in distances comparable with those routinely used in electronic circuits – distances that are getting smaller all the time. It is generally accepted that any imaginable optical computer would be hopelessly slow.

The fact that there is no new technology that is anyway near ready to step in should CMOS run out of steam means that continued doublings are dependent on a single, and increasingly fragile line of engineering research, and one which gets more difficult every year.

The packing density which determines the important performance parameters is expressed in terms of feature size, a the measurement relating to the size of silicon transistors and other components. In the early 80s this was around 5 microns. Today it is approaching .09 (90nm). As features become smaller a number of issues become more pressing: Each transistor generates heat as it switches. With transistors packed more and more densely the heat becomes harder to dissipate. Reducing the voltage will help, and this has been done several times in recent years. But the fastest processors will soon consume 1000 Watts and this can't go on forever. Insulation presents another problem. As the insulating layers become thinner they begin to leak electrons. Ultimately, and somewhat further downstream, there will be a lower limit on the number of electrons that can constitute a reliable switching transition.

In production terms, at the limits of CMOS miniaturisation two effects are possible. Firstly the cost of the fabrication plant needed to make the semiconductors will become so great as to exceed available capitalisation, making further progress unviable. Secondly, as progress becomes more difficult new generations of chips may begin to appear less frequently or in smaller increments of performance.

Of course not all increases of computing capacity come from faster clocks. Chip designers have always looked for ways of raising processor performance without increasing the clock rate. The large on-chip

instruction and data caches which now dominate chip real estate increase throughput by up to an order of magnitude for many workloads, and the use of pipelines has improved performance still further.

In the latest generation of chips pipelining with branch prediction, predication and code and data speculation take this process further. Increasingly sophisticated caching means that processors now run largely out of on-chip memory. Hyperthreading, where multiple logical processors within the chip execute separate threads in a manner analogous to symmetric multiprocessing, allows the processor to exploit more and more of a program's inherent (or at any rate explicit) parallelism. For performance to continue rising at the present rate, the contribution of such techniques to overall performance must progressively increase, and the hunt is on for further innovations that will make better use of the available cycles as well as those that will deliver more of them.

The CMOS process has now taken us further than anyone would have guessed and at regular intervals, commentators point to this or that looming problem as the block that will prove the end of CMOS. During the 90s many thought that progress would be halted by the difficulty of actually constructing increasingly tiny machines. But this proved to be no problem for companies willing to spend millions of dollars on a lens. Later when a the need for resolution approaching the wavelength of light became a serious problem they increased the frequency, using extreme UV lithography at a wavelength under 200nm⁴.

Today the show stopper is insulation. As the insulating layers between the conductors become thinner electrons begin to leak - a phenomenon known as tunnelling. The size of the silicon atom is rather less than an angstrom. In a few years the insulating layers of silicon dioxide will reach 15 or so angstroms – currently regarded as the limit. Who knows how they will solve this – but they probably will. The quotation at the beginning of this article, taken from Gordon Moore's address to the 2003 IEEE International Solid-State Circuits Conference expresses the predicament of the semiconductor industry perfectly.

The outlook then for chip production in the next decade or two is governed by the economics increasingly difficult and expensive engineering. Moore's Law is indeed an exponential relationship, but for the commercial computer industry it is effectively a steady state. That steady state depends on new and upgrade revenue continuing to pay for new, bigger fabrication plants and for research. The cost rises every generation and so far, so does the revenue. But both sides of the equation are at risk. Eventually market saturation or some other effect could begin to erode revenue growth. Alternatively the capital cost of the new technology could exceed what even a healthy industry can bear.

If CMOS should reach its limit and the industry were to re-align itself successfully around a slower moving technology, the cost of semiconductors would eventually drop. Since the main competitive pressure would then be price and the billions now going into new fabrication plants would be saved, semiconductors could become very cheap indeed. At one time software was given away with the much more expensive hardware. In the future the reverse could happen. It is not impossible to imagine a time when a display and the computer to which it is attached cost little more than a sheet of good quality paper.

Software

What then would be the likely consequences for software? The demand in the marketplace that has fuelled processor development is for increasingly functional applications. Speed is not usually an end in itself. The power is used to deliver new features and so fuel the upgrade market.

⁴ the visible spectrum lies between 400 and 700nm.

It is reasonable to suppose that when hardware stops getting much faster there will be pressures on the performance of software.

- There are always some areas where increasing and changing demand even without any improvement in perceived functionality demands greater speed. A present day example is internet security. With more stress on security the Internet is increasingly mediated by complex security systems operating in real time. Not only do these eat more cycles, they have to serve more users. The same applies to many distributed services. If the hardware can't get faster the software will have to get more efficient.
- The software application upgrade market has always depended on greater speed. Vendors have to pack in more functionality to justify the upgrade, and this costs cycles. Insofar as the industry could continue along present lines these cycles would in future have to be found through more efficient software.
- Constantly improving hardware has left behind an amount of slack in software systems. ISVs tend to concentrate on new features. Inefficiencies are tolerated at time of development and then forgotten about as they are papered over by faster hardware. When hardware growth is flat these inefficiencies will once again become visible.

To what extent is software able to meet this need?

You only have to look at the difference in performance between Windows and Linux to see that for some software at least there are cycles that could be freed by more efficient design. But possible gains through better coding and design are limited and can only happen once. There are certainly ways that software can be made to perform more efficiently, but because of the huge cost of optimising software at the statement level these tend to be in the form of generic improvements applied to the system as a whole. Caching and particularly the use of parallelism are probably not exploited to the extent they could be and flat processor speeds would probably act as a spur. But the area where, if it could be improved, there would be an impact similar (in quality) to the impact of speeding up the hardware, is in the way systems are integrated and the wider architecture dovetailed together. This applies at the fine level of procedure call, the use of libraries, protocol stacks etc, and also at the larger grained level of interfaces between applications, distributed interfaces (such as CORBA and COM) and protocols themselves.

Because of the complexity of systems generally and the cost of writing software, there has been much concentration on the techniques which make software complexity manageable. We have chosen quite reasonably to use increasingly cheap cycles to buy these improvements. Advances such as the formal interface, scoping rules, structured programming, object oriented languages and design, and the various methodologies promote organisation, quality, and co-operation among programmers.

However there is a natural tendency for techniques which enable rapid well structured development to do so at the expense of run-time efficiency. Encapsulation, interfaces, modern languages and late binding have revolutionised software productivity and quality. They have allowed individuals to concentrate on small functional units without the need to be concerned about the rest of the system. But these advances are often made at the expense of efficiency. Techniques that aid encapsulation tend to incur losses due to friction at the interfaces.

It is generally true to say that within the compiled unit modern build tools produce efficient code. Widely used building blocks such as databases, where the economies of scale justify large development budgets, also tend to be highly efficient. But because we have concentrated on speed of development rather than speed of execution, software systems now pay a penalty in performance for the methodologies that made them possible.

The balance will not be redressed overnight, but a protracted period of flat hardware performance could provide the motivation for innovation in the way systems of programs involving dynamic interfaces are realised in code. There is considerable scope for improvement, but much of it can only be realised through an analysis of the way individual systems behave dynamically at run time. There must be some way of monitoring systems during use to maintain a detailed view of the running code and to factor out unnecessary complexity. In cases where execution crosses the boundary between independently produced components, interfaces can be simplified or factored out as they might have been if those components had been written as one. This problem is not easily amenable to static analysis because of the widespread need for late (run time) binding.

This approach can be extended into general adaptive self optimisation. In principal the details of program behaviour would be observed on the fly in parallel and optimised on a statistical basis for best performance. The idea of adaptive optimisation not new. Examples already exist in software and hardware:

Adaptive Java compilers exist which use run-time performance statistics to recompile sections of code on-the-fly into native code for faster execution. Programs running under these compilers can visibly speed up as they execute.

Something similar happens in modern processor pipelines which are fed with rearranged instructions in which a particular, dynamically predicted flow of control is favoured.

To give one example of a possible future technique, program branch points might be re-weighted to reflect observed flow. Once a sequence of instructions has been seen to take a particular path again and again, that path can be optimised at the expense of less frequently followed routes. It is not uncommon for one of a number of alternative execution paths to be favoured by one or two orders of magnitude in the presence of a particular workload. Allowance for this cannot be built into every program. But techniques similar to those of the classical compiler optimiser, directed by execution statistics could be applied at run time to adjust the logic adaptively as it executes.

Admittedly such refinements are no substitute for a platform that doubles in speed every couple of years. But that is not the effective position. By no means all performance is dependent on the raw speed of the silicon. Many activities are dominated by factors such as I/O and synchronisation, and timeouts also contribute. The idea of synchronisation goes through the whole of software and hardware architecture, from the idea of the clocked CPU to the communication between threads and processes. Many problems in synchronisation are not amenable to adaptive optimisation but some are, though the difficulty of realising them should not be underestimated.

Today the motivation for such schemes is not strong. In practice conventional hardware doubling is seen as the normal source of increased performance. But the future may look different.

In addition to opportunities for actively improving software performance, more static hardware performance may have some positive natural spin-offs. It is likely that should hardware stop getting faster for a time, there would also be a general taking up of slack. The rate of change in the underlying hardware means that systems are often never truly optimised. This is often seen in the presence of bottlenecks. Today systems change so fast that unevenness in the performance or capacity of individual components may never be addressed. This is true of the large distributed databases, as it is of routing on the internet. In many systems,

Anthony Wilkes 2003

processing or throughput is bound at one or two points, and it is this which dominates their performance. An upgrade to the hardware at some point in the system can completely change the situation and put the bottlenecks somewhere else. Things change more quickly than can be assimilated and the true potential performance of the system is never realised.

A period of consolidation would, over time, tend to put this right. A further positive effect might arise from the fact that in the context of a slower changing industry, code would be used for longer. If upgrades are fewer and applications last longer a higher quality production process can be justified.

Whether innovation overall would suffer is difficult to gage. Many important advances have been triggered by faster hardware. Each jump in processing power makes new things possible and this makes the field exciting. The first computer graphics in the 60s and speech and handwriting recognition today were both made possible by new, faster processors. With static hardware performance such innovations would be fewer, and there is always the danger that a slower moving technology would be less attractive to the innovative.

It is inevitable that the end of CMOS would lead to general commercial changes in the industry, and this could lead to instability. The huge budgets now going into each new generation of fabrication plants would no longer be needed – at least for that purpose. There would be a drop in the cost of semiconductors as the main players compete increasingly on price. Whether software makers would benefit is uncertain. For a time there would be real performance and quality benefits to be gained. After that we might see an increasing commoditisation of computing and this could make it more difficult for growth to pick up again should a new technology become available.

It is impossible to say when or indeed whether processing power will start to level off. Most people think that it will be somewhere in the next 10 or 15 years. The effect on software could be quite dramatic but of course it will not happen suddenly and we will begin to see the signs early on as the large semiconductor manufacturers begin realigning their businesses. We shall have to wait and see.