The CMOS End-Point and Related Topics in Computing

Maurice V. Wilkes
Olivetti Research
Cambridge

In 1954, the IBM 704 computer represented the state of the art with an add time of 36 micro seconds. Now we have add times of a few nanoseconds, representing a speeding by a factor of approximately 10,000. One may get an idea of what this means by noting that an algorithm which once took ten minutes can now be done 15 times per second. As a result, Industrial applications can now be based on algorithms that were once of academic interest only. I need not stress the importance of this in subjects such as pattern recognition.

In 1974, INTEL reached the point at which they were able to put 4500 transistors on a single chip. The result was a primitive processor known as the INTEL 8080; the MITS Altair 8800 which was based on it is regarded by many people as being the first low cost personal computer.

Important as the INTEL 8080 and its immediate successors are now seen to be, when they were first developed they were ignored by companies in the main stream computer industry, who saw them as having nothing to do with the business they were in. They seemed to be good for home computers and for nothing else. At the time, this was a sound enough assessment of the situation. It was ten years before a few—very few—far sighted people came to appreciate the true potential of CMOS for fast processors.

The insight came from an appreciation of the effect of the scaling laws for microelectronics; in their simplest terms, these state that as the feature size decreases the current that gates can supply falls off, but the capacitances that have to be charged fall off more rapidly. The result is that circuits become faster simply by being made smaller.

In 1986 MIPS Computer Systems demonstrated the first R2000 and showed that it was a match for even the large minicomputers of the period. The R2000 had, on the same chip, a RISC integer processor, a memory management unit, and control circuits for off-chip caches. It had an 8 MHz clock and was about five times as fast as a DEC VAX 780, which was one of the most popular minicomputers of the period, and 1.5 times as fast as the VAX 8600. Both these minicomputers were based on bipolar technology.

This rapid transformation of the situation was the result of the entirely unexpected speed at which CMOS process development took place; the CMOS process engineers picked up the ball and ran with it—and they are still running.

In its race to overtake ECL, CMOS was helped by the fact that it was a simpler process, and by the large market that soon developed for personal computers. ECL also progressed towards higher densities, but was always behind. If CMOS had stabilized at about 1 micron, high density ECL might have caught up. This was shown by work at DEC Western Research Laboratory (WRL) where a slightly simplified version of the R3000—a successor chip to the R2000—was fabricated in a 1 micron ECL process. Enough (partly) working parts were obtained to show that operation at 300 MHz was possible.

However, CMOS did not stabilize and it is now clear that the superiority of ECL as a high speed process tends to disappear as feature sizes are reduced. It is also true that the superiority of CMOS as a low power process tends to disappear. This is because CMOS dissipates power only when switching and increasing the switching speed has the effect of putting up the average power dissipation. Thus CMOS and ECL tend to converge in power consumption as well as in speed as the feature size decreases. For example, the DEC ALPHA 21164, announced about a year ago, consumes 50 watts. This may be compared with the 150 watts consumed by the WRL experimental processor referred to above.

Decrease in size of CMOS cannot go on forever and we are now approaching a clearly defined physical limit. When the number of electrons used to represent a 1 is reduced from thousands to hundreds, quantum statistical effects become troublesome. It is now generally agreed that this point will be reached at a feature size of 0.1 micron or perhaps 0.05. This is the CMOS end-point referred to in the title of this paper. Beyond that point, the circuits may fail to work or, if they do work they will not be any faster—they may even be slower. If so, while they may be useful for memory chips, where capacity rather than speed is important, they will not be of interest for high-speed logic.

It is not fabrication problems, in particular problems with lithography, that will determine the CMOS-end-point. Masks are drawn on a large scale and reduced photographically. The image is transferred to a photo-resist on the chip . It has long been anticipated that problems with optical lithography would be encountered when the feature size became comparable with the wavelength of the light used and a great deal of work has been done on X-ray lithography. Surprisingly, however, we are now at 0.35 micron and the use of optical lithography is still possible in spite of the fact that the feature size is about the same as the wavelength.

Several facts may help us to understand this surprising result. First, the optical system can be designed for a fixed reduction ratio and a very small depth of focus is all that is required. Secondly, the object is not to make a grey scale image, but to produce sharp edges. Thirdly, the operators of process lines are prepared to pay $1M for the lens.

It may be that a change to X-ray lithography will be necessary to get down to 0.1 micron, although this is not yet clear. Even when X-ray lithography fails, electron beam methods will be available. Thus, it can confidently be asserted that the limit to the reduction in feature size will be set by the physical problems that I have mentioned and not by process difficulties.

If feature sizes had continued to decrease at the rate that we have become accustomed to, we would already be within a few years of the CMOS end-point. However, progress has slowed somewhat, and the National Technology Road Map for Semi-Conductors published in the US by the Semi-Conductor Industry Association shows a feature size of 0.1 micron as being reached in the year 2007 and 0.07 micron in 2010. These figures represent targets rather than predictions. There are many technical problems to be faced and it could be that the final stages of CMOS development will be long drawn out.

Some further progress may be possible when we meet the end of CMOS as we know it. It may be that development of current technology will lead to simpler and faster processes than present day CMOS. Very low operating voltages and small signal swings may be exploitable and there are semi-conductor materials other than silicon to be considered. These approaches may lead to useful advances, but there will certainly be no more routine doubling of speed every two years.

It would be foolish to predict that no new family of semiconductor logic, radically different from CMOS and its current competitors, will emerge. If one does, it might possibility be based on tunneling phenomena, but I do not see the immediate outlook as being good.

Some long-term hopes may be placed on single electron effects. There is nothing unreasonable in this since it may in a sense be said that modern physics started with a single electron experiment, namely the Millikan oil drop experiment. In this experiment, small oil drops were allowed to fall freely in air in an electrostatic field. In the absence of any ionizing radiation, the drops were observed to fall at a rate that depended on their diameter and on the viscosity of the air. When a source of ionising radiation was switched on, a drop occasionally picked up an electron and was seen to rise under the influence of the electric field instead of falling. Sometimes, a drop would pick up two or even three electrons and rise at a correspondingly greater speed. This experiment provided convincing proof of the atomicity of electricity. It must be surely ranked as one of the most elegant experiments in the whole of physics.

The earliest demonstration of single electron effects in semi-conductors were made at very low liquid helium temperatures. Recently however, Haroon Ahmed and his collaborators, working in the Cavendish Laboratory at Cambridge, have demonstrated such effects at liquid nitrogen temperatures. However, it is clear that many decades of work will be necessary before the possibility of practical applications can even be considered. This is an area of physics in which experiments tend to march ahead of theory and there may perhaps be some surprises.

THE SPEED OF LIGHT

It was early said that the speed of light would limit the speed of computers. It is certainly true that in order to get faster, computers must get smaller. However, the speed with which signals pass across a silicon chip is limited by the rate at which the conductors can be charged with the relatively limited current available and is much less than the speed of light. A modern workstation chip is about 1 cm across. If speed of light were the only consideration, the processor could be as much as 10 cm across, this being the distance which, allowing for the presence of dialectric, a light signal can travel in 0.5 nS.

It is possible to send a signal between chips with the velocity of light if matched transmission lines are used. In order to do this the pad must first be charged; this has a capacitance of about 1 pF and to there must be added the output capacitance of the driving transistor, which may amount to several times as much. Since the matched transmission line appears like a resistor, connecting it to the pad does not add to the capacitative load and doing so leaves the waveform sharp. After a time delay determined by the velocity of light, reduced by the presence of dialectric, a close replica of the waveform appears at the far end of the line,

An important qualification must be made to the above argument. A transmission line does not add capacitative loading, but it does add resistive loading and it will therefore cause the signal swing to be reduced. This loss must be made good by providing an amplifier and the amplifier will add delay. However, the argument helps to explain why it was that, in the early 1990s, Hewlett-Packard were able to use off-chip caches in their processors and yet remain competitive.

If it were possible to use a transmission line with a characteristic impedance of several hundred ohms, instead of the usual 80 ohms or so, the resistive loading would not reduce the signal amplitude significantly and amplifiers would not be necessary. Unfortunately, geometrical considerations make it impracticable to fabricate lines of such high characteristic impedance,

An alternative and practicable way of reducing loss of signal amplitude is to use larger driving transistors with lower output impedances. However, this involves running at very much higher power levels. It was the secret behind the Cray 1 computer. This computer was based on ECL at a very low scale of integration and consumed 100 kW of power, most of which was dissipated in the terminating resistors for the coaxial cables that ran between the boards. In spite of relying on thousands of coaxial cables for interconnect, the Cray 1 was the fastest computer of its day.

The suggestion has often been made that we should use light for communication, retaining electronics for amplification and switching. Gallium arsenide is often spoken of in this connection, since it is possible to make both transistors and light transducers (lasers) in that material. Thus, one can imagine a number of chips, each containing more or less logic and interconnected by fibers.

Unfortunately, the conversion efficiency of gallium arsenide transducers is not high, and it would be necessary to use even larger power amplifies than are required for matched transmission lines. For this reason, and since there is no point in using optical transmission for its own sake, gallium arsenide based optical interconnect has not so far proved attractive.

It has recently become possible to make transducers of much higher efficiency using composite semi-conductors. Moreover, these transducers have the merit of firing at right angles to the plane of the chip, instead of tangentially. This may transform the whole situation. In particular, optical interconnect may find application for connecting high speed memory to the processor.

Memory interconnect is an area of increasing concern, since memory access times have not been keeping pace with the increasing speed of processors. We have a strange situation. That part of the semiconductor industry which deals in processor chips is interested in high performance and architectural innovation, even though the market for the product is at first not large. On the other hand, the part of the industry which makes memory chips appears to be interested in large volumes and nothing else. David May, of SGS-Thompson, recently remarked, in a Keynote Address delivered at the 22nd International Symposium on Computer Architecture, that these two parts might well be two separate industries instead of two parts of the same industry. It is time that designers of processor chips began to concern themselves with the whole system and take a hand in the design of high performance memory chips. There is much scope for architectural innovation beyond what is made possible by the use of the various forms of dynamic RAM now becoming available. Software has a part to play, along with hardware, in improving memory performance.

OPTICAL COMPUTERS

At one time, I would draw comfort from the thought that, when electronics had finally run out of steam, then optical computing might take over. However, it is difficult to avoid the conclusion that the miniaturization of CMOS has already given us speeds beyond anything that we are ever likely to achieve by optical means.

It is necessary to distinguish between opto-electronic systems and purely optical systems. In the former, transistors are used in conjunction with optical devices, in particular, for amplification. In a purely optical computer no use at all would be made made of free electrons.

It is the general consensus that no purely optical computer is in sight. There is a complete lack of any developments that could possibly be seen as leading to a practical logic system in which one light wave controls the passage of another, without any use being made of electronics. On the other hand, there is at least one system in which a light signal can be steered by the application of an electric field. This is based on lithium niobate ($\mathrm{LiNbO_3}$) and is available commercially. Research workers at the University of Colorado have recently demonstrated a simple computer based on lithium niobate switches and incorporating an optical delay line memory. This was of some interest to me personally, since the problems they encountered were very similar to those which my colleagues and I encountered in the late 1940s in building the ultrasonic delay memory for the EDSAC.

A lithium niobate switch has two light inputs, I_1 and I_2, and two outputs, O_1 and O_2. It also has an electrical input. Normally a light input applied to I_1 emerges from O_1 and an input applied to I_2 emerges from O_2. Within the lithium niobate substrate, the two signals traverse two light channels very close together. If a suitable voltage is applied to the electrical input, the two signals change over and the input applied to I_1 emerges from O_2 and vice-versa.

If a photocell followed by an amplifier is connected to the electrical input, the device, although strictly opto-electronic, can be regarded as a surrogate for an—as yet uninvented—purely optical device having three optical inputs and two optical outputs. It can be shown that this constitutes a universal logic device and, by suitable use of the three inputs, can be made to perform all the fundamental logic operations including AND, OR, and INVERT.

Photons, unlike electrons, interact with each other and with matter with very great difficulty. In the current devices, the length of the double channel in which the two light signals interact is several centimetres. There is the possibility of some reduction, but, inevitably, in devices of this kind, a length equivalent to many wave lengths will always be required. For this reason, optical switches will, as far as can be foreseen, continue to be characterized by long latency compared with electronic switches; the same is true of optical amplifiers based on laser action in a fiber. These devices are, therefore, likely to find application in telecommunications, where latency is not a great problem, rather than in computers where it is fatal.

PARALLELISM

It has often been asserted that, when we reach the point at which computer hardware no longer gets faster, we have merely to switch over to parallelism and progress will continue as usual. It is now becoming generally recognized that this is an oversimplification and it is important to have a clear understanding of why this is so.

Suppose we have a mainframe running a workload—I know that we use workstations now, but never mind—and we wheel in a new advanced mainframe, then all jobs will be speeded up by approximately the same amount. Some recompiling may be necessary, but not reprogramming. If, on the other hand, we wheel in a many-processor parallel computer, then a good deal of replanning and rewriting of the programs will be necessary in order to get any speed increase at all. While some jobs will run much faster without the expenditure of much effort, in most cases, to get even a factor or two or three will require great effort. Some jobs will be capable of very little speeding up, if any at all.

Thus, there is no straight answer to the question of how much gain in speed we can obtain by the use of a given parallel configuration. All depends on the application. It is like asking the question 'how long is a piece of string?'. Anyone responsible for running a computing service on a serial computer, who hopes to get a major speeding up across the whole of his current workload by going over to a parallel computer, is likely to be disappointed.

Until recently, there were few constraints on the parallel architectures that could be proposed. Now there is an overwhelming premium on architectures which use commodity processor chips that were originally developed for use in workstations. Otherwise, the parallel computer must carry the whole overhead of fast chip development. We will hear little more of single instruction multiple data set (SIMD) architectures, and even less of the traditional forms of data flow. This is, in my view, a welcome simplification. It still leaves much scope for variety, especially in the way private and shared memory is arranged.

It has long been the case that the advantages of parallelism are realised only at the top end of the computer range. Lower down it is better to buy a faster machine. In other words, it has never turned out that parallelism is economically attractive if the alternative of using faster circuits exists.

On the software side, it is not enough, as is sometimes assumed, to identify sections of a program that can run as parallel threads and then program those threads to run on separate processors. If it happens, as it is very likely to be the case, that one thread runs much longer than any of the others, all you will get is a lot of idle processors—you will not get great speed. The essential problem in parallel computing is to balance the load of the processors so that they are all in use for the greater part of the time. This principle is sometimes referred to as Amdahl's Law.

In order to balance the load, one needs to know to a good approximation how long the individual threads will take before they run to completion. It is very hard to figure this out by staring at static code or by analyzing it with a computer program. In the general case it is impossible. A human being has an advantage over a computer program in that he can bring to bear a lot of background knowledge—about programs in general and about the particular program that he is working with. Given that knowledge, a human being can make useful guesses, but it is hard to codify the procedures he uses. The problem is essentially one in artificial intelligence. John McCarthy, in some interesting after-thoughts on his Turing lecture on 'Generality in Artificial Intelligence', identifies the problem of coding commonsense or familiar knowledge for use by a computer as one of the main obstacles standing in the way of progress in artificial intelligence.

People often point to the success of optimizing compilers and suggest that that success might be repeated with parallelizing compilers. It seems to me that this represents a complete misunderstanding of what optimizing compilers do and what a successful general purpose parallelizing compiler would have to do. The reason that we use high level languages is not in order to be able to write more efficient code than we could do by hand, but in order to make programming very much simpler. With parallelizing compilers, it is the other way round. The object is to achieve high running speed by efficient optimization even at the cost of making programming more difficult. Often the speeding up to be obtained by parallelizing is small. A parallelizing compiler which gave a gain in speed in easy cases only would be of limited value.

Optimizing compilers perform such tasks as eliminating redundant evaluation of subexpressions, removing statements from loops, replacing tail recursion by iteration, and optimizing the use of registers. These are rather mechanical tasks. There is nothing mechanical about optimizing a parallel calculation. One must begin by designing an effective algorithm, or rearranging an existing one, and then one must implement it so that the individual processors are as fully loaded as possible. This needs much human effort. It is a pipe dream to imagine that it will ever be possible to write a program in C, for example, and optimize it automatically for an arbitrary parallel system. It is certainly not a matter of repeating a success.

It should be particularly noted that work on the algorithm typically forms a large part of the work that must he done to create a successful parallel program. This greatly limits what can, in the foreseeable future, be expected from a parallelizing compiler, since once a program has been written in a procedural language, the algorithm is fixed in all its essentials. No one whose primary interest is in obtaining speed would suggest the use of functional languages.

Having said what I have just said, I should emphazise the importance of not falling into the trap of assuming that because something is impossible in the general case, it is not possible to do something useful in special cases. In fact, there is a whole class of jobs in which the above problems do not obtrude themselves to any great extent. These are jobs involving independent or nearly independent tasks, each of which may run on a separate processor. In jobs of this class a spectacular speeding up may be achieved. An example of a problem in which the jobs are entirely independent is the simulation of computer hardware by making many runs of the same program for different starting conditions. If a conventional single processor machine is used, these runs have to proceed serially; in a parallel machine with many processors they can proceed independently, each on its own processor. There are problems in physics and other subjects in which each step of the program involves making heavy calculations for each point of a grid. It is not to be expected that these calculations will be completely independent of one another, but they may be sufficiently so for a significant increase of speed to be obtained by running them on separate processors.

It may be that jobs which can be broken down into independent or nearly independent tasks will emerge as the main justification for the use of many-processor machines. I now think that there may be more computing of this kind to be done than I used to believe. Experience shows that a computer with enhanced capability in some particular direction attracts to itself jobs that suit it. It may be that many suitable jobs will come out of the woodwork to feed the parallel machines that are now being installed. Only time can show.

In general, there is little that the computer scientist can do except to provide tools. I do not underestimate the importance of this. However, the real action lies in application groups, working in physics, fluids mechanics, and other subjects. in which the programmers are also specialists in the applications concerned. I always advise students of computer science, who feel attracted by the possibilities latent in parallel programming, to seek out such a group, rather than remain in computer science. They will then have many opportunities for solving technical problems in parallel programming in a realistic environment. Some of the more interesting solutions may not generalise to other scientific areas, but may nevertheless be of great importance in the area in which they arise.

An alternative to using a specially built parallel computer is to make use of a bank of workstations. This is at present being done successfully for the simulation of computer hardware, but in general the workstations do not have sufficient I/O bandwidth nor, if they did, would an ethernet provide a fast enough interconnect. The problem of I/O bandwidth is being addressed by workstation designers and local area networks with wider bandwidths are coming into use. It is possible that for some applications banks of workstations may prove a strong competitor to parallel architectures. Even if they are slower, they will have the advantage of being available, whereas it is not clear who is going to pay for expensive highly parallel machines. It has been remarked that there is little likelihood of a strong commercial market developing for computers that cost several million dollars and are hard to program.

LOW POWER COMPUTERS AND ASYNCHRONOUS DESIGN

The fact that modern CMOS chips get hot has led to an interest in ways of reducing power consumption. This would be desirable even in the case of workstations. There is a particularly pressing necessity in the case of hand-held computers, since batteries can only just provide what is needed and an extension of battery life by a factor of two or three could make all the difference to the acceptability of a product.

Since the clock circuits in a CMOS processor switch all the time, whether computation is being done or not, power can be saved by reducing the clock rate, or stopping it altogether, when there is no work to be done. In due course, an interrupt will cause the clock to be returned to normal. Since this way of saving power does not impact performance appreciably, it is likely to come into general use for workstations as well as for hand-held computers.

The suggestion has been made that a fruitful approach to power economy might be made by way of asynchronous design. Here there is no clock, but each action leads to the next and the circuits run at their natural speed. Asynchronous circuits have a long and, it must be admitted, unrewarding history. Interest in them was revived by Ivan Sutherland's Turing lecture on 'Micropipelines' in 1989.

So far, no practically viable asynchronous CMOS microprocessor has been demonstrated. This is perhaps not surprising since asynchronous circuits are much more complicated than synchronous circuits and harder to design. However, Prof. S. Furber working at Manchester University has designed an experimental asynchronous ARM processor and has some working chips. While this is far from being a basis on which a practically viable system could be constructed, it is much more than a toy. Furber has demonstrated his processor in operation with supply voltages ranging from 2.6 to 6 volts and temperatures from -50 degrees Centigrade to over 100 degrees. He found that raising the voltage or lowering the temperature increased the running speed in the way that theory indicated.

In a thorough-going asynchronous design, two wires are used for each digit; a transition on the first indicates a binary zero and a transition on the second indicates a binary one. When the value has been accepted by the receiving circuit an acknowledgement is sent back, thus establishing a pipeline. This system is *delay insensitive*, in the sense that it is guaranteed to work correctly whatever the delays in the gates and the wires may be, or whatever they may become. However, it requires slightly more than double the silicon area than would be needed for a synchronous design.

A more practical scheme is to have one extra wire per word, instead of one per digit. The digits of the word are encoded conventionally on a set of wires, and a transition on the extra wire indicats that all the digits are valid. This system—known as bundled data—is the one that Sutherland advocated. It has the disadvantage that is not delay insensitive in the above sense, so that its correct functioning depends on the ability of the chip designer to ensure that the delays in the digit wires are controlled and matched in the same way as they would need to be for clocked circuits.

Validity of the word may be indicated by a transition in either direction on the extra wire or, alternativly, by a transition in a specified direction followed by a transition in the opposite direction. The single transition system, which is the one that Furber has implemented in his current design, makes it simpler to design the asynchronous circuits themselves, while the second has advantages when it comes to sending signals from the chip to the outside world.

The original reason for an interest in asynchronous circuits was that their advocates thought that they would run faster than synchronous circuits. This expectation was apparently based on the fact that asynchronous circuits run as fast the circuit parameters allow, whereas with synchronous circuits the clock has to be set at a conservative frequency. In my view, this apparent advantage is always likely to be more than cancelled by the slowing effect of the handshake essential to asynchronous operation, as well as by the greater complexity of the circuits . It is certainly the case that, up to the present, no speed advantage for asynchronous circuits has been demonstrated.

It remains to be seen whether asynchronous circuit design will enable better power economy to be achieved. On one of the Alpha chips 40% of the power is dissipated in clock distribution. This is an upper limit, not likely to be closely approached, to the saving of power under peak computing conditions.

When assessing the possible advantage of asynchronous operation in economizing power when some of the circuits are idling, it should always be remembered that they have a rival in switched clock systems, that is, systems in which the clock is temporarily removed from circuits that have no work to do.

Further potentiality for saving power can be tapped if the possibility of varying the supply voltage is admitted. The speed at which CMOS circuits run is proportional to the supply voltage, while the power consumed is proportional to the voltage squared. It has therefore been suggested that power could be saved by reducing the voltage according to the current backlog of work and the anticipated future demand. In a clocked system, a corresponding adjustment must be made to the clock frequency. In an asynchronous system, the circuit speed will automatically adjust itself.

Acknowledgements

I am indebted to many colleagues with whom I have discussed the topics treated in this note, especially E. D. Maynard, Jane Alexander, and Bob Laheny of ARPA. Jeremy Dion and Norman Jouppi have kindly commented on the manuscript.

REFERENCES

Sutherland, I.E. *Micropipelining* CACM vol 32 no 6 pp 729-738 (1992)

Furber, S. *Computing Without Clocks: Micropipelining the ARM Processor* in 'Asynchronous Digital Circuit Design' ed Birtwistle, G. and Davis A. Springer, pp 212-262 (1995)

McCarthy, John. 1987. Generality in Artificial Intelligence, Comm. ACM, Vol. 30, No. 12, pp. 1030-5. Reprinted, 1987, in ACM Turing Award Lectures: The First Twenty Years, ACM Press, New York.

Jordan, H.F., Heuring, V.P., and Feurstein, R. *Optoelectronic Time-of-Flight Design and the Demonstratin of an all-Optical, Stored Program,Digital Computer.* Proc. IEEE vol 82 no 11 pp 1678-1689 (1994)