THE BASIS OF PRESENT COMPUTER DESIGN

W.A. Clark

Rapporteurs: Mr. J. Givens
Mr. M. Martin
Mr. P. White

Abstract

Professor Clark's talks gave an outline of the development in computer design. Included were some ideas on the reasons for development along this path together with some queries as to the correctness of this approach. There followed a discussion on alternative computing models with the questions of performance and convenience uppermost. Finally he speculated on some possible techniques that may be introduced as a result of new technology either currently being manufactured or on the verge of development.

The first lecture can perhaps best be summed up by a quote Professor Clark used at the close of his lecture.

'Oh speculators on things, boast not of knowing the things that nature ordinarily brings about, but rejoice if you know the end of those things which you yourself devise.' (Leonardo da Vinci.)

The beginnings of understanding of computers can be attributed to Turing. The formulation he derived was based, not on computers, but on the decidability question in Mathematics. To help in this aim he invented an algorithmic device (1936) which is now known as the Turing Machine. From this he developed the notion of the Universal Machine, and it is this device which forms the basis of the modern computer. At the time it was thought that a building the size of the Albert Hall would be required to house such a machine, but of course this is not the case. It is interesting,

however, that the Turing Machine is an excellent vehicle for developing algorithmic ideas.

Of course, no theory exists to predict the smallest number of states for a Universal Machine, and it is not even possible to predict the smallest program for any given problem. From an engineering, finite-state viewpoint, a different picture emerges. If a state space is composed, then after being placed in any particular state, it is always possible to predict which state will be entered next. This implies that to reach a particular answer state it is only necessary to reach the path leading to this answer state, and it is the job of the programmer to find any point on this path, one, hopefully, close to the answer state.

In the early days there was always the question of whether to use a generalised or a specialised machine. Clearly things specially built are more efficient than those immitating a specialised machine. However, economics dictates that these machines be centralised and this causes the following two problems. There is never enough memory, and a machine is never exactly what is required. And so to make what you have into what is required takes both time and memory.

To combat these problems there is more centralisation and this leads to the development of bottlenecks which prevent the use of certain parts of the machine by anyone, other than the one user causing the bottleneck.

The follow on from this was the concept of virtual memory which makes possible larger address spaces and which may raise the question as to whether there is now a 'virtual computer science.' As yet we cannot convert computers into convivial tools, that is to say that we cannot use them exactly as we desire, and then on completion of the task, pack them up and put them away. In fact we seem to like the idea of things gaining in complexity and becoming intellectual playthings.

Time-sharing as an idea is nominally sound enough. A machine has to be kept busy for as much time as possible and the time-sharing notion aimed to help in reaching this goal. It has never quite lived up to the expectations of the enthusiasts; in fact, it has put a huge burden on the peripheral devices and requires large central stores. It seems anomalous to have such a large central memory and then split this into small pieces, one per user. This puts the desirability of time-sharing in doubt whereas multi-access certainly is not. Along with this drive towards centralisation came the notion that everything was cheaper the larger it was, and thus the swing towards time-sharing was encouraged. Now the tendency is back towards the mini-computer and greater efficiency.

The problem of reliability is especially troublesome, for there is no reason that computers cannot be made reliable, and yet not enough emphasis is put on this facet of machine design and manufacture. Indeed, manufacturers sometimes behave very irresponsibly by releasing 'bugged' software, and yet the users, on the other hand, are impatient for anything new and prepared to accept software which is not error free. The ultimate problem here in any case is that we don't know how to make our intentions manifestly algorithmic.

Professor Clark's second lecture was concerned with a review of the present state of several subjects in hardware design.

In the past few years, one matter that has been given much consideration in this area is that of timing and synchronisation. It may be said that the engineering fraternity has had a rather improper view of what constitutes synchronous operation, since as Petri points out, events that strictly alternate are more

synchronised than those that happen 'sensibly and simultaneously'. However, the simplest answer found to sequencing things, and trying to do things nominally simultaneously, was to use 'self-timed' or 'self-sequencing' logic where each operation can take an indeterminate amount of time to complete but having done so, must announce its completion to the next operation in line.

It is surprising that more machines have not been built in this form; in fact most of the computers in the world are regulated by their single clock. It becomes more and more difficult to keep in step with the central clock as its speed increases, and the apparatus it controls becomes more widely distributed. Nearly all these machines meet up somewhere with another clock which they beat against, giving rise to synchronisation problems. The difficulty of synchronising is that there does not exist the perfect bistable flip-flop. We have, instead, devices which can go into a 'metastable' state representing neither 0 nor 1 when marginally triggered by an insufficient signal, with potentially disastrous consequences in such things as synchronisers and asynchronous arbiters which must decide, for example, which of two competing processes should have some resource in the case of conflict. It has been suggested that many of the mysterious system crashes seen today in the world's large computer configurations may be due to this marginal triggering problem. The frequency of occurrence of such a malfunction is related to the frequency of the system clock, and that is why this problem is only being encountered now, as the driving clock speeds increase.

The most common example of handling timing and ordering in computers is provided by machines in which state is entirely specified by the contents of the memory, together with the value of a single instruction counter, as in the Turing machine and most simple computers. An extension of this is the pipeline (perhaps better called an assembly line), where a better use is made of the time available while preserving the sequentiality of operations.

However, more interesting systems are multi-sequential processes, where there is more than one instruction counter in use, and all of them are concerned with separate sub-problems of the same single task. It is important to distinguish this from, for example, time-sharing systems where the inter-relationship is really at the operating system level, whereas the multi-sequence systems devote all sequences to simple tasks related within the machine system itself. Another interesting design has been called a broadcast architecture, for all the instructions are kept in a large common store and are broadcast in sequence simultaneously to any number of users. Broadcast systems no longer exhibit the co-ordination difficulties due to feedback processes but there are other problems, such as the need for very high transmission bandwidths since the store simply circulates through the entire collection of programs in memory and the receivers pick off the line that which they find relevent to their purposes. This architecture has some similarity to the packet-broadcasting schemes coming into use in communication networks.

Micro-electronics has seen a great change in the size and speed of devices: there has certainly been a decrease of at least six orders of magnitude in the size of components, but it is interesting to note that change in their reliability is only of three or four orders (measured as the number of failures per week, say). Although reliability has improved as component engineering has improved, the devices have become more and more complex, so that their effective reliability has stayed about the same. Professor Hamming suggested that this was just a question of cost, since we had built reliable computers, for example for the space programme. Professor Clark agreed that almost any level of reliability could be achieved with a great deal of attention and money, but argued that commercial manufacturers would only design components to a minimum acceptable level of reliability. One of the problems at the moment is that new designs are for faster and faster devices, and not for the

increased reliability of components that is needed.

With the ever-increasing scale of integration it is becoming easier to 'churn out' complex devices in quantity, but there is now the question of yield, that is, the proportion of these devices that actually work. It is quite expensive to produce perfect components, but if a few imperfections are acceptable, then the price reduces dramatically. An example of this is the adoption of MOS (metal-oxide semiconductor) memories in the computer industry. These memories are inherently unreliable, and so have to be made with built-in error-correcting circuits. It should be noted however that the real problems in reliability are going to be at the software level rather than in hardware, which is far less complex, and so there will be a need for fault-tolerant software engineering just as there is for built-in error-correcting circuits at the hardware level.

Over the last few years, microprogramming has become more and more important in hardware design. The use of read-only memories to store the control of the machine has greatly simplified the wiring of computers, and given greater flexibility and power at the machine level. However, microprogramming has its limitations: if a structural feature not in the basic 'data structure' of the machine is needed (for example, two arithmetic units, where there is only one) then it must be simulated in microprogram, with a substantial loss of speed and efficiency. Traditionally, general-purpose machines are designed and then specialised with software, but there are demanding problems that are not suited to this approach and require greater speed or complexity. They can be solved by specialising the hardware rather than the software, as in the Washington University project to devise electronic macro-modules that would be building-blocks for use by programmers, rather than engineers, to design and build special-purpose hardware, the modules being plugged together to provide anything from digital filters to three-dimensional display

processors. It is possible that the larger levels of integration now coming into use will enable this sort of apparatus to be produced in manageable pieces of hardware, hence allowing an even greater range for processing in combination with a stored program.

Professor Clark concluded the lecture by mentioning that communication networks seemed to have a bright future with present networks well established and larger bandwidth paths already being envisioned.

In the discussion which followed, Professor Dijkstra wondered if there was any hope for solving the problem of quality control of chips (integrated circuits). As it was so difficult to check whether an integrated circuit was all right, he thought that perhaps manufacturers did not bother to try.

Professor Clark agreed that manufacturers (and users) were not concerned enough about checking components, or about their expected lifetimes. This had not mattered much to date, possibly because a new generation of technology came in before the last one had outlived its natural life span. The introduction of electron-beam MOS-technology memories may alter this situation, since these devices actually wear out with use. Mr. Laver asked what was the scale of component lifetimes, and Professor Clark suggested about two to three years at present though no doubt more exotic materials than MOS would be discovered and change this.

Dr. Holt took the opportunity of the mention of the arbiter problem to say that one of the benchmarks for something that he would accept as a satisfactory theory of systems is one that would deliver a rigorous demonstration (in the manner of Galois) of the impossibility of constructing an arbiter, for instance.

Professor Randell concluded the discussion by referring those present to an interesting talk given by David Wheeler in this seminar series three years ago on the arbiter problem.

Professor Clark introduced his third lecture by reminding the audience that the second major advance in hardware is concerned with solid state electronics. CPU power has improved at a somewhat slower rate than that of memory, partly because there has been more pressure to increase the amount of memory - 'there never seems to be enough'. Memory size has increased by about three orders of magnitude in the past 15 years whereas clock rates in CPU's have hardly changed at all except very recently when clock rates have been pushed up to the 100 MHz range. IBM's own published predictions (1975) for the future are of the order of 10 Mega-instructions per second.

Professor Clark then reviewed briefly the current techniques of producing integrated circuits, emphasising that present technology seems limited by the amount of fine detail obtainable through diffraction limits of light, and the advantages of the newer techniques beginning to emerge, notably the electron beam technology, is that they have diffraction limits several order of magnitudes lower than at present. This clearly influences the physical size of units.

The electron beam technology is fairly old (for example, the Williams tube), but has remained idle because of the requirements imposed by analogue stability. These limits have been largely overcome and although it is possible to contemplate fabrication by electron beams on a small scale, initial work is probably limited to the application of memories built to take advantage of the high deflection speeds of electron beams. One can now position the beam with great precision on a target area.

Two slides illustrating the use of electron beam technology with respect to memory were shown. The first illustrating the focusing and deflection arrangements and the second the target details, that is semi-conductor layers etc., and biasing. The problems of flaws in the semi-conductor target must be solved by the use of compensating circuitry 'around' the target.

At present the beams can be made about 2-5 microns wide on the target area, and this makes possible 16×10^6 resolution elements per sq. cm. Because beams spots are not so well defined, enough space must be allowed to resolve positively a bit, and a practical solution would be about four resolution elements per bit giving 4×10^6 bits per sq. cm. of target material. Such devices are currently being made. The access time is of the order of 10 μs and the service time, including any restore if the readout is destructive, is of the order of 1 ms. So we are somewhere between disc technologies and central memory speeds. A practical system might have 20-40 tubes using common deflection electronics. One needs feedback methods with fiducial marks scattered around the target surface to ensure precision and stability, and in this way very stable systems can be achieved. Unfortunately the memory wears out under electron bombardment and to ensure that one part does not wear out more quickly than another the information would have to be moved around physically by some precession method. This precession might take minutes, or hours and the total useful lifetime would be in the order of years. A final slide was shown, illustrating a two-stage deflection system, which allows the target area to be increased from say 1 sq. cm. to perhaps 10 or 20 sq. cms, thereby giving several order of magnitude increase in size of archival memory.

A question posed by the ability to make very large memories is 'How large should a memory be?' As pointed out by Sutherland, if very large memory systems exist, it will take some time to fill them up - the 'swimming pool effect', and this can be a nuisance if they have to be 'off-loaded' for reliability control. One solution, of course, is to make smaller memories and then synthesise larger memories from them.

There is some interest in using electron beam devices under computer control to produce very fine structures in other types of materials, and at some point in the future it can be imagined that there will be micro-factories in which things are scaled down into the microscopic domain. Tools will clearly be a problem.

The lecture then moved into some conjectures concerning multi-computer systems. At present, using mini-computers as intermediaries the ARPA net is a prominent example of a multi-computer system. The 'Pluribus' (1) combines 14 mini-computers each with about 4K words of local store and a large common store, and although at first it was expected to produce something faster, later it was appreciated that the reliability aspect is more important.

Another development in this category is the touch sensitive screen, and this could become a very powerful technique. An example of the use of such a method is that of Dr. L. Weed et al (2) in the University of Vermont where complete medical protocols and all other features necessary to operate the relevant medical record data-base, are controlled entirely from the screen.

The question of giant systems was touched upon and although it would seem possible to build such machines perhaps the fundamental question is 'Do we need them?'

Chemical memories exist as a possibility in the future, based on the observation that small viruses and protein molecules form regular three-dimensional crystals whose surface periodicity might be sensed with an electron beam, and although no work has been done in this area, some has been proposed.

More passively the idea was suggested by the author of a 'computing slurry'; a 'slush of chemicals' with a sensing director. No ideas exist on how to build such a 'slurry'. The suggestion that one instruction on such a computer might possibly be called "flush and add" raised some hilarity within the audience.

A brief discussion of existing 'robots' and their future concluded the lecture series together with quotations from A. Clarke's '2001' and E.M. Forster (The Machine Stops).

References

(1) S.M. Ornstein et al: 'Pluribus - A reliable multiprocessor'. AFIPS Conference Proceedings, pp551-559, vol. 44, 1975.

(2) J.R. Schultz, S.V. Cantrill, K.G. Morgan: 'An initial operational problem-oriented medical record system for storage, manipulation, and retrieval of medical data'. AFIPS Conference Proceedings, pp239-264, vol. 38, 1971.