SYSTEM DESIGN FOR NON-ENGINEERING STUDENTS AND THE SYNCHRONISATION PROBLEM

D. J. Wheeler

Rapporteurs: Mr. J. S. Clowes
Dr. P. Henderson

Abstract: Dr. Wheeler began by describing, briefly, the course on hardware design which he teaches at the University of Cambridge. As he remarked, the organisation of Computer Science teaching at Cambridge is rather different from the arrangements described by other speakers.

Two courses in computing science are offered at Cambridge; a third year undergraduate course and a post-graduate diploma course. The two courses have some lectures in common and Dr. Wheeler teaches the parts dealing with hardware design.

The main purpose of the instruction in logical design is simply to give the students some idea of what goes on inside a machine; it is not intended to train as computer designers. A secondary purpose is to enable students, when they become users, to attach instrumentation and other peripheral devices to computers. In fact, it is apparent that only a very small percentage of the total logical design effort required for a large system is associated with the CPU. A much larger proportion goes into the design of peripherals, many of these being designed locally for some special purpose.

The method adopted for teaching about hardware is the historical approach. Starting with a description of a simple computer possessing a store and a processor, new devices are introduced in turn and the reasons for their introduction explained. Thus the students are introduced to index registers, sub routines, adder enhancements of various types and store overlapping. Next input and output buffering

and the use of the common store as a buffer which introduces the idea of interrupts. This leads to the consideration of simple channels for autonomous transfers, and immediately brings in present day problems. As soon as interrupts are allowed, one has to consider modes of computer working, program protection, and system software with its ramifications.

At this point they leave the general development of computers and pursue the great drive of designers, the quest for speed! Overlapped arithmetic units, overlapped stores and methods for speeding-up other parts of the computer are discussed. Returning to the general theme, dynamic storage protection is considered. Finally, it is shown that, since the simple machine has become so complicated, it is now too expensive to use it for any ordinary purpose and so peripheral processors are necessary, each like the original simple computer!

Other logical design topics treated more briefly are communications, microprogramming and coding. This hardware course is supported by other courses on system design and by courses on Boolean algebra and sequential logic.

One of the essential functions of a university course seems to be to produce examination questions! In practice, the hardware course is not very good for this and it is very difficult to invent good questions and even harder to mark them.

The course will obviously develop in the future and the next innovation is likely to be the introduction of hardware practice. Teaching about hardware is now given in schools and this will soon begin to affect university courses. Thus, it is not possible at present to forecase what the course will be like after, say, a period of 2 or 3 years.

The problem of synchronisation

Dr. Wheeler devoted the rest of his talk to a discussion of a particular problem in logical design. He chose to do this, rather than give a more general talk, because he considers that discussion of this point should form part of every course on hardware or logical design. His reasons for isolating and emphasizing this point, which he calls the problem of synchronisation, are as follows:

(1) Many existing computers have faults because of neglect of this point. (Dr. Wheeler has found that at least 50% of the computers whose logical design he has studied in detail have faults of this kind).
(2) The point is rarely taught well and only occasionally appears in the text books.
(3) It is apparently difficult to appreciate. Furthermore, people trained in switching theory or logical design find it especially difficult.
(4) The problem is general. It is common to all forms of logic and may also be present in systems programs. It touches many disciplines, for example, circuit theory, logical design, system programming and information theory.
(5) The occasional malfunctioning of all practical computers and peripherals is to be expected if this point is neglected.

The problem of synchronisation occurs whenever we have to connect together two devices one of which is synchronous and cannot be stopped. The standard kind of design for a computer with peripheral units is an example. In order to be specific, we shall pursue this particular example, which leads us,to consider how to deal with interrupts.

When an interrupt occurs, the computer receives a signal from the outside world saying that some device is ready and the machine has to use this signal to make a decision about what to do next. We may assume that the signal has passed through suitable circuitry so that it is a perfectly shaped signal exactly like all the other signals in the computer, except that it is not synchronised to them in any way.

One naive way of handling interrupts would be to use the interrupt signal to switch between two possible paths of control at the end of each instruction cycle. In practice this involves examining the wave form on the interrupt channel and since this is not synchronised with the CPU ambiguity is possible. Since this violates the rules of logical design it should never be adopted in practice and we need consider it no further.

An alternative design is to gate the signal into some form of flip-flop and use the state of this flip-flop to select between the alternative paths of control at the appropriate time. This solution is logically correct but, in practice, it will not work.

In order to see why this circuit is unsatisfactory one has to consider, in detail, how flip-flops work. The results of such an examination can be looked at from a number of points of view. Mathematically, the output voltage of the flip-flop is the solution of some differential equation and cannot change value instantaneously but must vary continuously with time and also with the initial conditions. Thus if the output is used at a particular time, the output will depend continuously on the initial conditions when the flip-flop was set. Thus there exists an input condition to make the output or not settled at any future time!

Circuit theory indicates that the flip-flop may settle down in a third, unstable, state with output ½. However, a more detailed analysis shows that the probability of the output remaining at this value diminishes with time as e^{-Bt}, where B is the effective bandwidth of the flip-flop, and t is the time allowed for the flip-flop to settle before the output is used.

Looking at the problem in another way, we can say that all we have to do is to decide which of two events happens first, the intercept or sampling pulse. To make such a decision does not require many bits of information, except in the case when the two events are close together. So information theory tells us that we will occassionaly require a large number of bits to make a correct decision and this will require a long time.

The final conclusion of all these theories is the same, that to synchronise two devices requires more time than is needed in simple logic design at every other part of the computer!

At this stage it is interesting to introduce some figures. Suppose that we wish to make a synchronising decision every microsecond, this involves approximately e^{30} samples per annum. If we also wish to design for not more than 1 error per year, then, with current values of B^{-1}, we must allow 30 times as much time for the decision. In

practise the situation can be worse than this because flip-flops are not designed to turn over quickly in the intermediate state. Thus the really critical case requires even more time than average bandwidth considerations will show.

When logical designers are first confronted with this problem they believe that they can get round it by elaborating their circuits. More elaborate design does alleviate the trouble but the fundamental difficulty, that synchronisation requires more time than a simple circuit transition. still remains.

The error rate can be reduced to any desired level by

introducing suitable delays into the circuitry. Since synchronisation problems tend to occur at crucial parts of the machine cycle, e.g. when one is about to access the store, this solution reduces the overall speed of the machine.

In principle one could remove the difficulty altogether by using asynchronous logic. Logical elements currently available from manufacturers are not suitable for this purpose, as one requires circuits which will indicate when they are ready. Theoretically such circuits can be produced but, in practice, one would always be involved with unstoppable drums and magnetic tapes, so asynchronous logic does not really offer a solution to the problem.

A practical technique for alleviating the problem is to take advantage of the idea of parallelism. A long delay is introduced into the interrupt circuit, of the order of an instruction. The practical effect of this is to delay response to the interrupt. This is often acceptable but can cause difficulties for the software writers who require early warning of fault conditions, e.g. overflow, if error tracing is not to become too difficult.

Dr. Wheeler concluded his talk with the following general remarks on the problem of synchronisation.

Other instances of the occurrence of synchronisation problems are:

Telephone networks. Here it is the main cause of cross-connections and the situation is made worse because the bandwidth of a relay is quite small. Supervisors. Dijkstra's technique of semaphores involves sensing and changing something in the same instant it thus assumes an infinite bandwidth which is probably permissible in the case of a program much as this.

The incidence of faults due to synchronisation problems will tend to rise as software activity increases but they will occur randomly. They could cause friction between engineers and software writers.

It is difficult to see how one can devise techniques for tracing these faults when software and logic diagrams appear to be perfect.

Laboratory hardware experiments tend to conceal this problem since they nearly always involve strictly synchronous circuits.

Discussion

Dr. Wheeler asked for the views of members of his audience on the importance of the point he had raised and whether, or not, it should be taught to computer science students.

Professor Ercoli said he thought it was not very important, at least for computing science students. It is really a problem of sampling theory and belongs, properly, to the field of control engineering.

Dr. Wheeler remarked that he had never seen the point discussed in books on sampling theory.

Professor Seitz thought the point was important but, although it was often passed on by word of mouth, it was never written about.

Dr. Wheeler said there had been a paper on the subject round about 1950[1]

Professor Seitz "That is word of mouth".

Next Professor Suchard expressed the view that thiw was a particular aspect of a general problem, namely, the absence of any consideration of time in most teaching on computer design. This

1. In spite of an extensive search this paper has not been identified.

was the fault of mathematicians who reduced logical design to Boolean algebra. Points of this nature were also ignored in courses on circuit design because, there, one was not concerned with how the circuits were used.

Dr. Wheeler agreed that this point lay on the boundary between logical design, circuit design and information theory and this was the cause of its general neglect.

Professor Seitz said that the point was very difficult to demonstrate experimentally.

Professor Michaelson thought that students should be made aware of the point, although there was nothing they could do about it.

Mr. Elphick remarked that the effect of teaching about this point would be to destroy the students belief that certain things could be treated as "black boxes". He asked if Dr. Wheeler thought they should be reassured.

Dr. Wheeler replied "No". The training of computer science students should be such as to make them sufficiently suspicious of everthing.