A challenge to memory designers?

In the good old sequential days memory accesses took place one after the other, and a memory technique (such as ferrite core technology) that could accommodate one access at a time was adequate. When independent I/O-channels were introduced, the memory access mechanism became a resource shared between the central processor and the channel(s), and a switch was introduced which --usually on a priority basis-- granted mutually exclusive accesses to the various competitors. ("Cycle stealing" was the term then introduced to describe that arrangement.) It seems worthwhile to point out that originally the need for mutual exclusion was a purely technical one, dictated by ferrite core technology: as program areas, working space and buffers were at any moment in time disjoint, there was no logical reason.

Or: hardly so, to say the honest truth. This disjointness of these areas of concurrent activity could only be guaranteed thanks to a macroscopic synchronization of computing and I/O-activities, and for the implementation of such synchronization constraints, it was sometimes exploited that the mutually exclusive storage cycle consisted of a "read", followed by a "write". Examples are the test-and-set instruction, the swap (between a memory location and a register) or the indivisible "add-to-memory". The mutual exclusion, i.e. the indivisibility of these operations on memory words was suddenly a logical necessity: nothing prevented different processors to attempt to access the same location simultaneously.

The difference is pointed out in connection with the solution described in EWD496 "On-the-fly garbage collection: an exercise in cooperation.", in which two processors are supposed to operate in the same memory: as long as they access different words in memory, there is no logical objection at all to have these different accesses overlap in time. The possibility of simultaneous attempts to access the same storage location --all the time and all through memory-- is the rule, the actual coincidence of such attempts will be the exception.

For that reason one dreams of a memory arrangement in which simultaneous accesses can take place unencumbered as long as different locations are involved, and mutual interference (i.e. delays) will only occur at those exceptional moments when it is logically necessary, i.e. when the same word is involved in more than one attempted access. Compared with that dream, the current solution with the switch which always imposes mutual exclusion independent of such coincidence, is a safe but crude solution. This crudeness is pointed out because it might be acceptable with the number of processors $N = 2$, but it won't remain so, if N grows: for sufficiently large N, the possibility of many simultaneous accesses seems to become some sort of must.

I have no idea how such a memory would lock like --an electronically rotating memory with many sensing stations?--, hence the word "challenge" in my title. Let me finally explain the presence of the question mark: with EWD496 we now have one such algorithm with $N = 2$, say. The question whether we shall be able to design more of those, with values of N that are considerably higher, is still an open question.

9th June 1975
NUENEN - 4565
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow