Competition for memory access in the KDF9

By C. S. Wallace and B. G. Rowswell*

The I/O control of a computer may adopt various strategies for serving competing requests from peripheral channels for access to a single core memory. The strategy adopted places some limit on the peripheral configurations which may be simultaneously active. An "ideal" strategy is exhibited which imposes no constraint other than that implied by the finite speed of the core memory, but it is expensive to implement. Limits on configurations are derived for the ideal, first come-first served, round-robin and priority systems. It is shown that the first come-first served and round-robin systems have little or no advantage over a random choice. The KDF9 at the University of Sydney, which was delivered with a first come-first served system, has been modified to incorporate a priority system. The new system uses 48 fewer circuit boards, but allows the attachment of high-rate channels which otherwise could not be accommodated.

The English Electric–Leo–Marconi KDF9 is unusual among medium-sized computers in having, in most configurations, an autonomous buffered data channel for each peripheral device. All devices work in the block-transfer mode, in which a single peripheral instruction initiates the transfer of a block of data to or from a block of contiguous memory locations. Once initiated, each transfer proceeds in parallel with other transfers and computation, under the control of its own buffered data channel. Each channel competes with other channels and the main control of the computer for access to core memory. Some channels (e.g. magnetic tape channels) require a memory access for each word transferred, others (in general the channels handling slower peripherals) require access for each character transferred.

The method of resolving the ensuing multi-way competition for core memory cycles influences the kind and number of peripheral transfers which may be simultaneously executed. The standard method employed by the KDF9 is far from optimal. The KDF9 at the Basser Computing Department of the University of Sydney has been modified to employ a different method, which requires less hardware but which substantially increases the range of peripheral configurations which can be accommodated.

Statement of the queueing problem

It will be convenient to adopt as the unit of time the memory cycle time of the computer, i.e. 6 μsec for the KDF9. For the present purpose the peripheral configuration can be characterized by

N = the number of peripheral channels;
R_i = the maximum rate at which channel i emits memory access requests ($i = 1, \ldots, N$);
p_i = the patience of channel i, that is, the maximum time which can safely elapse between its emission of a request and the completion of its memory access.

For a channel serving a device such as a magnetic tape and having buffer storage for one word, in addition to any registers used for character assembly and disassembly, the product R_ip_i is usually slightly less than one. That is, the patience of the channel is almost equal to the period between one request and another. For channels serving devices having fewer buffering facilities, R_ip_i may be much less than one. For channels serving peripheral devices which can be stopped in mid-block without error, such as paper tape punches, p_i is effectively infinite.

In what follows, it is assumed except where stated otherwise that all channels have finite patience, and that

$$R_ip_i \leqslant 1 \quad \text{for all } i. \tag{1}$$

The I/O control unit which supervises core memory accesses for peripherals must ensure that even when all channels are running none must wait more than its patience before securing and completing a memory cycle. The following discussion assumes that a single core memory of unit cycle time is used for both peripheral and main control accesses, and that the I/O control prevents any main-control access from being initiated as long as it has any outstanding peripheral request to serve. Thus, after perhaps waiting for the completion of a main control access started when no peripheral request was outstanding, I/O control can serve one peripheral request per unit time. These assumptions are essentially consistent with the design of the KDF9.

The original system (*see* Burr *et al.*, 1964)

Each KDF9 channel issues a request by setting the Ready Pulse flip-flop (RPF) for that channel in I/O control. The standard I/O control contains a device called a Ready Pulse commutator (RPC) which scans the RPFs cyclicly at a steady rate of one per microsecond. Whenever one is found on, it is reset and its corresponding channel number placed in a 16-word by

* *Basser Computing Department, School of Physics, University of Sydney, Sydney, N.S.W., Australia.*

4-bit flip-flop memory called the E-store. The E-store is organized as a first in-first out queue stack. When I/O control is not busy serving a channel, it continually inspects the E-store. If it is not empty, I/O control selects the first loaded entry to serve next. Thus, I/O control serves the channels on a first come-first served basis, save that the ordering is subject to an error of up to 15 μsec introduced by the RPC, which may not find a set RPF until 15 μsec after it was set.

Ignoring this uncertainty, which has no beneficial effect on the system, this serving method imposes on the peripheral configuration the restriction

$$p_{min} > N \qquad (2)$$

where p_{min} is the largest integer not exceeding the minimum $\{p_i\}$.

(In this and following discussions, the time required for I/O control to secure control of the memory is ignored except where specifically mentioned. Its neglect does not materially affect the argument.)

The above restriction (2) arises from considering the case when all channels save that of minimum patience request access simultaneously, and the one of minimum patience requests immediately thereafter. Since $R_i p_i \leqslant 1$, the minimum period between two requests from any one channel exceeds p_{min}. Thus, no further requests can be issued before expiry of p_{min}. In this case, I/O control will make $N - 1$ services before granting access to the least patient channel, so a time N elapses before completion of its service.

If, rather than being served in order of requesting, the requesting channels are served in a random order, satisfaction of (2) is still sufficient to guarantee error-free operation. Restrictions (1) and (2) together imply that the sum of the channel rates is less than one. Hence I/O control is occasionally idle. In a period of length p_{min} starting from the end of an idle period, (1) implies that no channel can issue two requests, and so the maximum number of requests issued in this period is N. However, I/O control can serve p_{min} channels in this period, which, by (2), exceeds N. Thus I/O control cannot be continuously busy for a time exceeding p_{min}. Thus all channels are served with a delay not exceeding p_{min}.

It follows that where (1) obtains, as it does in a standard KDF9, the use of a first come-first served method is no better than a random method, and is therefore no better than any other method. Thus, the E-store in the KDF9, which exists solely to implement this method, serves no useful purpose.

The round-robin system

The round-robin system has been used on several computers, for instance the CDC 1700 (see Thomas, *et al.*, 1966), for allocating memory accesses. In KDF9 terms, the system may be described as follows. A scanner, analogous to the RPC, rapidly inspects each RPF in turn. When one is found on, the scanner is halted, the RPF reset, and its channel served. When service is complete, the scanner is restarted from the position in which it was halted.

This system is simpler to construct than the first come-first served system, but has a similar performance if the scanning rate is so rapid that the time to find the next requesting channel can be ignored.

After a channel i has requested service, other channels can be served at most once before channel i is attended to, since channels are inspected in rotation. Thus service of channel i is completed after at most N cycles. Hence (2) is a sufficient condition for the operation of this system. That it is also necessary can be seen by considering the case in which all channels request simultaneously. The last to be served, which may be the one of least patience, must wait N cycles for completion of its service.

An ideal system

An ideal system for deciding the competitions for memory accesses would guarantee error-free operation provided only that it was possible, i.e. that some ordering of outstanding requests existed which would answer each within the limit of its patience. One method which is ideal in this sense is to cause I/O control, whenever it is free to start serving a channel, to choose that channel which then has least time to run before expiry of its patience.

Suppose that, when using this system, a channel i is not served before expiry of its patience. Clearly, I/O control has never been idle since channel i issued its request. Thus, to prevent failure of channel i, a memory cycle which I/O control had, under the above method, given to some other channel, must be given to channel i, and service of one of the channels which were served during the patience of i deferred until after the expiry of the patience of i. However, any channel which was so served by I/O control under the above method must have had its patience expire before that of channel i, and so its request cannot be deferred without failure. Thus, if the above method fails, no method can succeed.

Unfortunately, implementation of this method requires a counter or similar circuit for each channel to record its unexpired patience, and is hence quite expensive.

The condition for successful operation of a peripheral configuration under the ideal method can be written

$$\sum_{i=1}^{N} [(t - p_i)R_i] \leqslant t - 1 \quad \text{for all integral } t > 0 \qquad (3)$$

where by $[x]$, both here and below, is meant the least integer greater than x.

(3) implies $\Sigma R_i < 1$, an obviously necessary condition for satisfactory operation. Given this condition, there must be periods when I/O control has no requests outstanding from channels of finite patience. In an integral time t starting from the end of such a period, the right-hand side of (3) represents the number of services which can be completed by I/O control for channels of finite

patience, as the first cycle of the time t may have been pre-empted by the computer main control or by a channel of infinite patience. That is, $t-1$ "deadlines" can be met in the time t. For a channel i, at most $[R_i t]$ requests may be issued in a time t. However, the deadline for each request occurs a time p_i after the request, so that only requests issued by time $(t-p_i)$ have deadlines before time t. Thus the left-hand side of (3) represents the maximum number of deadlines to be met for all channels by time t.

By replacing $[x]$ by $(1+x)$ in (3) it may be shown that:

(*a*) (3) is satisfied by all t greater than

$$\frac{1+N-\Sigma(R_i p_i)}{1-\Sigma R_i}$$

and hence need be checked only for lesser values of t.

(*b*) (3) is satisfied if (but not only if)

$$\sum_{i=1}^{N} 1/(p_i-1) \leqslant 1.$$

The priority system

Suppose the channels are numbered from 1 to N in order of increasing patience. The priority method of deciding I/O competitions is to make I/O control, whenever it is free to start serving a channel, choose the lowest-numbered channel among those then requesting. This system, which has been implemented on the KDF9 at the University of Sydney, is not ideal in the above sense. It will function without error provided that:

$$\text{for } i = 1 \qquad p_i \geqslant 2 \tag{4}$$

$$\text{for } 2 \leqslant i \leqslant N \qquad p_i \geqslant q_i + 1 \tag{5}$$

where q_i is the least integral solution of

$$q_i = \sum_{j=1}^{i-1} [q_i R_j] + 1. \tag{6}$$

In (6), q_i represents the longest possible continuous period during which I/O control is serving or has outstanding requests from channels of number less than i. Just before such a period, there are no such outstanding requests. The Σ term represents the maximum number of requests issued by channels of number below i in the period q_i, all of which are served by the end of the period. In addition, the first memory cycle of the period may be occupied by service of a channel of number above i which requested immediately before the period began. Thus, the right-hand side of (6) gives the maximum number of memory cycles which can elapse before I/O control is free to begin service of channel i. A further cycle is needed to complete the service for channel i, hence the patience of channel i must be at least $q_i + 1$.

Relations (4) to (6) apply whether or not the channels are numbered (i.e. assigned priority) in order of patience, but since q_i as given by (6) increases monotonically with i, it is clearly advantageous to use the assumed priority ordering. We now obtain an explicit upper bound for the patience required of the ith channel.

If q_i is the least integer satisfying (6), then for all integral m, $1 \leqslant m < q_i$,

$$(q_i - m) < \sum_{j=1}^{i-1} [(q_i - m)R_j] + 1.$$

Hence, substituting the right-hand side of (6) for q_i in the left-hand side of the above,

$$\sum_{j=1}^{i-1} [(q_i - m)R_j] + 1 > \sum_{j=1}^{i-1} [q_i R_j] + 1 - m.$$

Since all terms are integral,

$$\sum_{j=1}^{i-1} [q_i R_j] - \sum_{j=1}^{i-1} [(q_i - m)R_j] \leqslant m - 1$$

$$\sum_{j=1}^{i-1} \{[q_i R_j] - [(q_i - m)R_j]\} \leqslant m - 1. \tag{7}$$

Each of the terms $\{[q_i R_j] - [(q_i - m)R_j]\}$ is either zero or a positive integer. Hence (7) implies that at most $(m-1)$ of them are non-zero. Thus for $k = 1, 2, \ldots, (i-1)$, and $j(k)$ some permutation of the integers 1 to $(i-1)$,

$$[(q_i - k)R_{j(k)}] = [q_i R_{j(k)}].$$

Hence

$$q_i = \sum_{k=1}^{i-1} [(q_i - k)R_{j(k)}] + 1. \tag{8}$$

(Note that q_i is not necessarily the least integer satisfying (8).) An upper bound to q_i can now be found by replacing $[x]$ by $(x+1)$ in (8) and solving for q_i.

$$q_i \leqslant \frac{i - \sum_{k=1}^{i-1} k R_{j(k)}}{1 - \sum_{j=1}^{i-1} R_j}. \tag{9}$$

(9) is maximized by that $j(k)$ such that $k < l$ implies $R_k > R_l$. Commonly, and in the KDF9, $p_i < p_j$ implies $R_i > R_j$, in which case, if channels are numbered in order of increasing patience, the maximum value of (9) becomes

$$q_i \leqslant \frac{i - \sum_{j=1}^{i-1} j R_j}{1 - \sum_{j=1}^{i-1} R_j}. \tag{10}$$

Relation (9) or (10) gives an upper bound to the patience required of channel i in terms of the rates of channels of higher priority. Provided $\sum_{j=1}^{i-1} R_j < 1$, i.e. provided the channels of higher priority do not exhaust the capacity of I/O control, then q_i is finite. Thus, channels of finite patience may be added if I/O control has some unused capacity, in contrast with the situation under restriction (2).

The bound (10) can be achieved. For instance, if $R_1 = 1/3$, $R_2 = 1/4$ and $R_3 = 1/7$, then (10) yields $q_i \leqslant 10$, and 10 is in fact the least integer satisfying (6). Thus $p_4 \geqslant 11$ and $R_4 \leqslant 0{\cdot}09$.

Note that the I/O capacity unused by channels 1 to 3, $(1 - R_1 - R_2 - R_3)$, is $0{\cdot}274$. The "ideal" method described above would allow $R_4 = 0{\cdot}25$, $p_4 = 4$, but the priority system does not.

In some cases, (6) is satisfied by an integer less than the bound given by (10). For instance, if $R_j = 0{\cdot}1$ for $j = 1, 2, \ldots, 8$, (6) is satisfied by $q_9 = 9$, but (10) yields $q_9 \leqslant 27$. In this case, channel 9 may have $R_9 = 0{\cdot}1$ and $p_9 = 10$.

The priority system has been used in several computers, for instance the CDC 3300 (CDC, 1965), and was implemented by one of us (C.S.W.) on the Illiac II (Brearley, 1965). However, its properties do not appear to have been described in the literature.

Channels of small and infinite patience

The presence in a peripheral configuration of channels whose patience p_i is substantially less than $1/R_i$ does not materially affect any of the above arguments. For a first come-first served method, (2) applies as before. In a priority system, such channels must be assigned higher priorities than they would be given on the basis of their rates. It can be shown that there is never any advantage to be gained by departing from a priority assignment in order of decreasing patience. No KDF9 channel has a small patience in this sense.

Some KDF9 channels have effectively infinite patience, i.e. they automatically stop and wait until their requests have been served. Such channels must be counted among the N in restriction (2). Hence, on a first come-first served or round-robin system, the operation of a number of such channels requires increased patience of all other channels. However, such systems are then superior to a random system, for which (2) must be replaced by $p_{min} > \sum_{i=1}^{N} [R_i p_{min}]$. On the priority system such channels are assigned lower priority than any channel of finite patience, and hence can in no way reduce the capacity of the system to handle channels of finite patience. The highest priority infinite patience channel will absorb all I/O control cycles not used by channels of finite patience, up to a limit set by its maximum rate. Only when its maximum rate is achieved will other infinite patience channels receive service.

Implementation on the KDF9

A priority system such as described above has been implemented on the KDF9 at the University of Sydney. The Ready Pulse commutator and E-store were removed, and in their place was connected a circuit called the Server, which, on demand by the I/O control, places in the register previously used as the read-out buffer of the E-store the channel number of the highest numbered Ready Pulse flip-flop then on. Thus the priority of a channel is determined by its hardware channel number, the order of numbering being the reverse of that assumed in preceding sections of this paper.

This correspondence between channel numbers and priority assignment is not an essential part of priority systems, but in this case it led to a useful simplification of the design of the server, and happened, with the standard KDF9 channel assignments, to lead to a sensible priority order.

The Server is constructed from standard KDF9 printed circuit boards. By use of a logical design analogous to the carry skipping logic of a fast parallel adder, the effective operation time of the Server was made $0{\cdot}5$ μsec, or one clock phase of the KDF9. In addition to the Server, a small number of boards were required to reset the RPF of the chosen channel, which was previously done by the RPC.

All the new logic was contained in one frame of 24 circuit boards. It was wired on a spare frame in advance of the change-over. The computer was withdrawn from service for two days while the conversion was accomplished and checked out. No trouble has been experienced with the Server in approximately one year of operation. In total, 64 circuit boards were removed from the machine and replaced with 16 circuit boards.

The principal motive for the conversion was to allow the attachment of faster peripherals than were originally envisaged for the KDF9. Our configuration will soon comprise 14 peripheral devices, all of which may operate simultaneously. Under the old system, the minimum tolerable patience would have been approximately 15 memory cycles, or 90 μsec. Thus no peripheral could operate at a rate above about 11,000 accesses per second. Three of our peripherals have rates above this figure, and so could not have been added in their present form. In particular, it would have been impossible to attach a disc file or its equivalent without making exceptional provisions for it.

The peripheral of highest priority under the priority system must have a patience according to (4) of about two memory cycles, or 12 μsec. A further 6 μsec must be allowed for transfer of the information to the channel buffers. Since the I/O control of KDF9 always takes at least 6 μsec between when it decides to serve a channel and when it requires any data from that channel, the patience of a channel may usually be extended by about 6 μsec by having it issue requests for service 6 μsec before it is actually ready to be served. Thus the period $1/R_i$ of the highest priority channel can be as low as about 12 μsec. In this way, a peripheral of rate approaching 80,000 accesses per second can be accommodated by the priority system.

The peripheral configuration

In addition to standard KDF9 peripherals, the Sydney installation includes:

(*a*) A National–Elliott card reader (kindly donated by National Cash Register Co.).

Table 1

The rates and patiences of the devices peripheral to the Sydney KDF9. All devices save the CDC 1700 are installed and working

CHANNEL NUMBER	DEVICE	TIME UNIT = 6 μSEC				p MSEC	R KCS	ACCESS MODE
		p_{min}	p_{max}	p	R			
15	CDC 1700	2	—	3	0·33	0·018	55	word
14	SILLIAC	3	3·5	7	0·12	0·042	20	char.
13	Data Input	5	5·4	12	0·072	0·072	12	char.
10	Mag. Tape	6	7·7	32	0·03	0·192	5	word
9	Mag. Tape	8	10·1	32	0·03	0·192	5	word
8	Mag. Tape	11	12·8	32	0·03	0·192	5	word
7	Mag. Tape	12	15·8	32	0·03	0·192	5	word
6	Card Reader	14	19·3	150	0·006	0·9	1	char.
5	Printer	—	—	∞		∞	5	char.
4	Plotter	—	—	∞		∞	0·25	char.
3	P.T. Punch	—	—	∞		∞	0·1	char.
2	P.T. Reader	—	—	∞		∞	1	char.
1	P.T. Reader	—	—	∞		∞	1	char.
0	Typewriter	—	—	∞		∞	0·01	char.

(*b*) A data link to an older computer of the Princeton class, SILLIAC.

(*c*) A Benson–Lehner Model 341 digital incremental plotter.

(*d*) A channel for the input of unblocked data streams from various sources.

All the above are installed and working. In addition, a data link is being planned between the KDF9 and a CDC 1700 computer to be installed in 1967.

The characteristics of the peripherals relevant to this paper are shown in **Table 1**. The column headed p_{min} shows the minimum patience required of each channel as calculated from (4), (5) and (6). The column headed p_{max} shows the bound estimated by (10), and the column headed p shows the actual patience. It will be seen that the priority system accommodates all peripherals comfortably.

The present channel numbering differs slightly from that shown, and will be changed to accord with the table when the 1700 link is installed.

Acknowledgements

The Basser Computing Department, in which the KDF9 is installed, is partially supported by the Nuclear Research Foundation, whose assistance is gratefully acknowledged. While engaged on the work here described, one of us (B.G.R.) was supported by a Commonwealth Scholarship.

References

BURR, R. P., *et al.* (1964). *KDF9 Programming Manual*, English Electric–Leo–Marconi Computers Ltd., Kidsgrove, Staffordshire.

THOMAS, N. L., *et al.* (1966). *Engineering Specification*, 1700 *Direct Storage Access Bus*, Control Data Corporation, St. Paul, Minnesota.

CDC (1965). 3300 *Computer System Reference Manual*, Control Data Corporation, St. Paul, Minnesota.

BREARLEY, H. C., JR. (1965). *ILLIAC* II—*A Short Description and Annotated Bibliography*, *IEEE Trans. Elec. Comp.*, Vol. EC-14, p. 399.