Timing problems in electronic lockout chains

R. P. Lorétant[†]

Department of Electrical Engineering Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ, Essex

Lockout circuits similar to the arrangements used in electro-mechanical telephone exchanges to allow access to common resources are needed in multiprocessor computer configurations. The propagation time of the control signals, negligible in low speed relay circuits, can now lead to ambiguous situations resulting in temporary double connections. The problem is analysed and an improved circuit presented.

(Received January 1975)

1. Background

Lockout systems have been used in automatic telephone switching for a long time, especially since the advent of crossbar techniques (Korn, 1939; Joel, 1948). Their purpose is to solve the problem of competition which occurs when, for example several markers wish to gain simultaneous access to the same piece of equipment, e.g. a switch frame serving several hundred subscribers. Various levels of sophistication have been achieved in this form of circuitry, one of the simplest being shown in **Fig. 1.** As can be seen an essential feature of the arrangement is the existence of two chains propagating information and control in opposing directions. The lockout function is achieved by two different mechanisms depending on whether the requesting marker is up or downstream of the already connected marker. Upstream markers are prevented from operating their lockout relays whilst downstream markers may operate this relay but find that they cannot operate their connection relay.

In every lockout system a priority structure is established in case several sources request service at the same time. **Fig. 2** shows a chain with three markers with the conventions used throughout the paper. If two is presently connected and one and three request service, three will be served first. When it releases, the earth condition is restored on the request line, so one will be served next. A round-robin mechanism is therefore in existence for the allocation of the bus to the markers in case of conflict.

2. An electronic equivalent for the relay lockout chain

An analogous situation occurs in a multiprocessor computer configuration in which each of the processors competes for access to a shared pool of peripheral equipment. Although the data transfer mechanism will be very different in this case, the lockout problem remains essentially the same. However electronic rather than electro-mechanical methods will have to be employed. An arrangement like that is generally called a bus system.

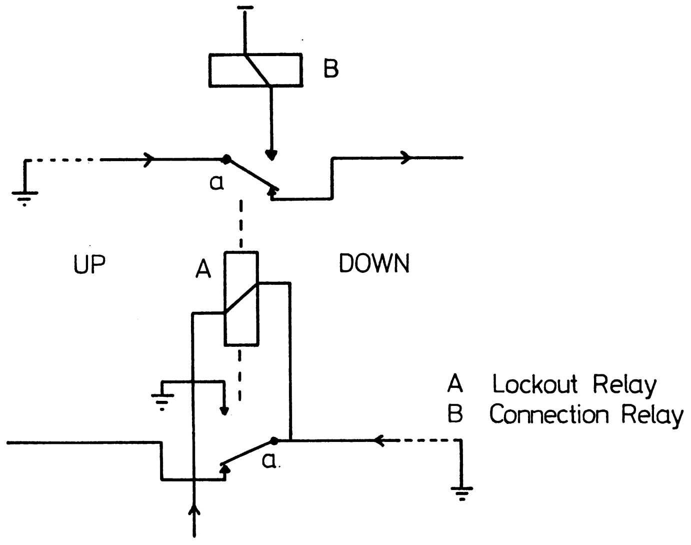


Fig. 1 Relay lockout mechanism

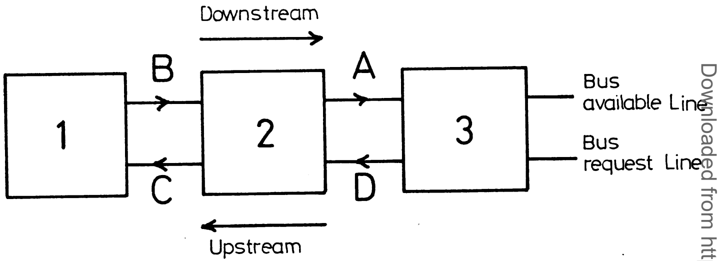


Fig. 2 Conventions used for the description of lockout mechanism

A transposition of the relay circuit is shown in **Fig. 3** with the lockout relay replaced by a bistable. Although the same logic still applies in a static sense, difficulties occur now: as the different ports will usually be in separate locations, the delay of the signals can give an unwanted transient behaviour. Downstream devices may seize the bus if they receive a free signal from both sides. Upstream devices can also do this as long as the busy condition does not intrude and force the first device to release after a short delay. This means that two devices are connected for a period depending on the propagation delay.

With the present circuit this can only be avoided by introducing a guarding interval covering the worst case. This may not be desirable if a very high speed is essential and individual timing of devices is difficult or impossible.

3. Introduction of a loop

The problem described arises because the devices downstream are not in a position to decide whether requests are made further up at approximately the same time. By looping the two chains this information can be provided and the delay introduced is now minimal. Such a scheme is referred to as a daisy chain (Thurber, Hensen and Jack *et al*, 1972) and an example is given in **Fig. 4.** The line going upstream carries the requests. The other line is interrupted in each device and carries a 'bus available' signal. Once a request is set, this signal is not repeated downstream.

An obvious disadvantage of this solution is, that it is not possible to test if the bus is free without setting a request; which was possible in the previous case. An internal timeout is necessary in each device, but the time allowed for will generally be much longer than the propagation time on the bus and can

[†]Present address: Grossackerstrasser 32, 8041 Zürich, Switzerland.

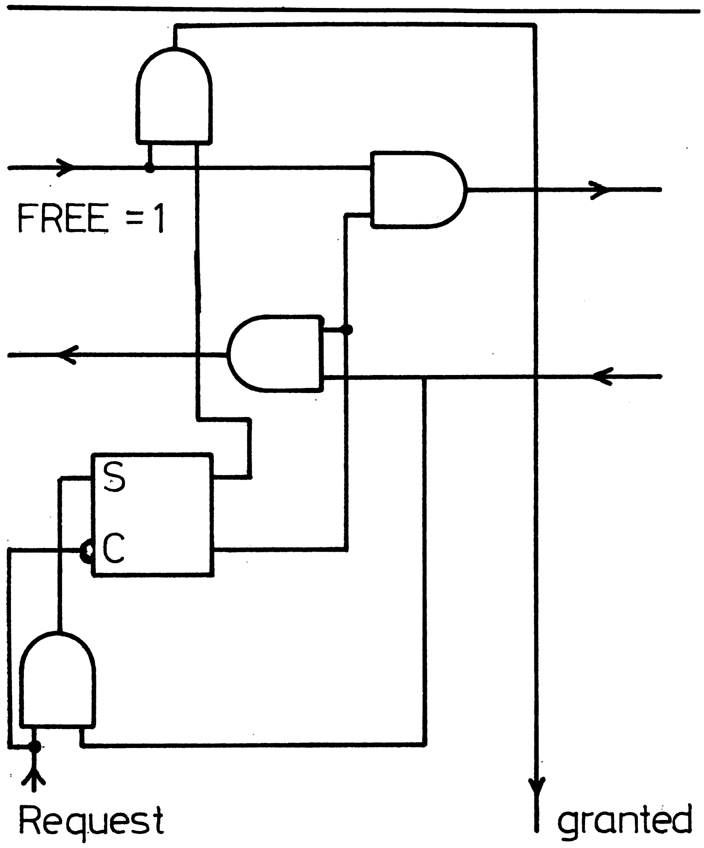


Fig. 3 Electronic transposition of relay lockout chain

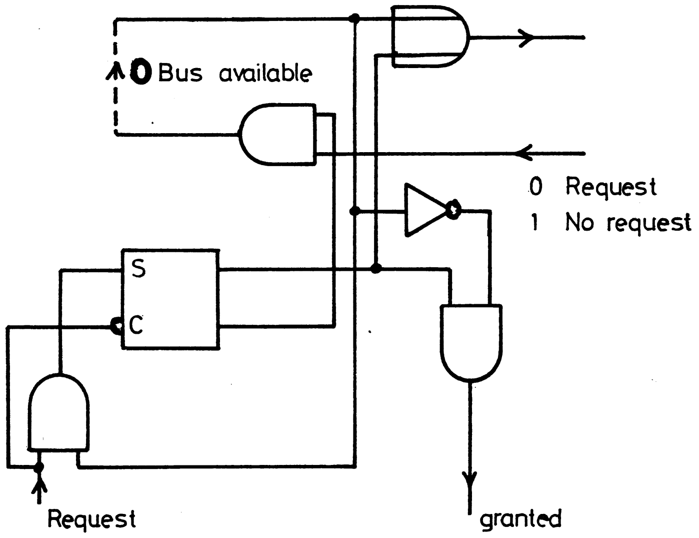


Fig. 4 Implementation of simple daisy chain

include some waiting time for the bus to become free.

Unfortunately this solution suffers fiom a similar uncertainty as the previous one in the case when a device in the middle releases when no other devices are requesting, which means that a free condition travels upstream and an available condition downstream. A downstream device requesting at this critical moment will get a confirmation very quickly, but later on when the signal has gone round the loop the bus can be allocated to one of the upstream devices, which forces again a release for the first request. In Thurber, Hensen and Jack *et al*, (1972) it is suggested to wait until such races have settled out. It will be shown that such a guarding interval is again dependent on the length of the transmission path. The daisy chain in this form has therefore no advantages over the simple lockout chain.

4. Analysis of the timing problem

Before a request is granted the lockout system has to make sure that no conflicting requests are being granted. It is not sufficient just to inform the other devices: an active confirmation is necessary.

The ordinary lockout chain cannot provide this, whereas the daisy chain mechanism works on such a principle. Its behaviour will now be analysed.

The following symbols are introduced:

D_{jk} = Propagation delay from device j to device k
T_j = Time bus granted at device j
F_j = Time device j forced to release
R_j = Time device j releases
DQ = Queuing time
DL = Delay of loop at end of bus
n = number of devices.

The times are measured from the instant when device j sets a request. It is assumed that the propagation delays are the same in both directions on the bus, e.g. $D_{jk} = D_{kj}$. Device k lies further up than device j.

Several cases can now be distinguished.

1. *Bus totally free*

Device j may set a request at time 0

This signal will pass the loop and arive at device k as 'Bus available signal' at the time

$$T_k = D_{jn} + DL + D_{nk} \qquad (1)$$

For device j this works out as

$$T_j = DL + 2.D_{jn} \ . \qquad (2)$$

This is the time required for the signal to travel round the loop to make sure that no other requests are present. It has to be considered minimal as a further reduction would again lead to the difficulties mentioned in section 2 and no time is wasted by guaı ding intervals. Full advantage of the position on the bus can be taken.

2. *Bus busy further upstream at device k*

In this case device j has to queue up for the use of the bus. The minimal time is now

$$T_j = DQ + 2.D_{jk} \ . \qquad (3)$$

The section of the bus from device k to the end is now constantly busied and it is not necessary to send a signal round to check this. If device k releases exactly at the instant when the signal arrives from device j, the confirmation can be sent without delay and T_j is now equal to $2.D_{jk}$.

3. *Bus in state of release*

This is the aforementioned critical case. The condition here is that device k releases before the request from device j arrives and device j requests before the busy condition existing on the upper section of the bus is received.

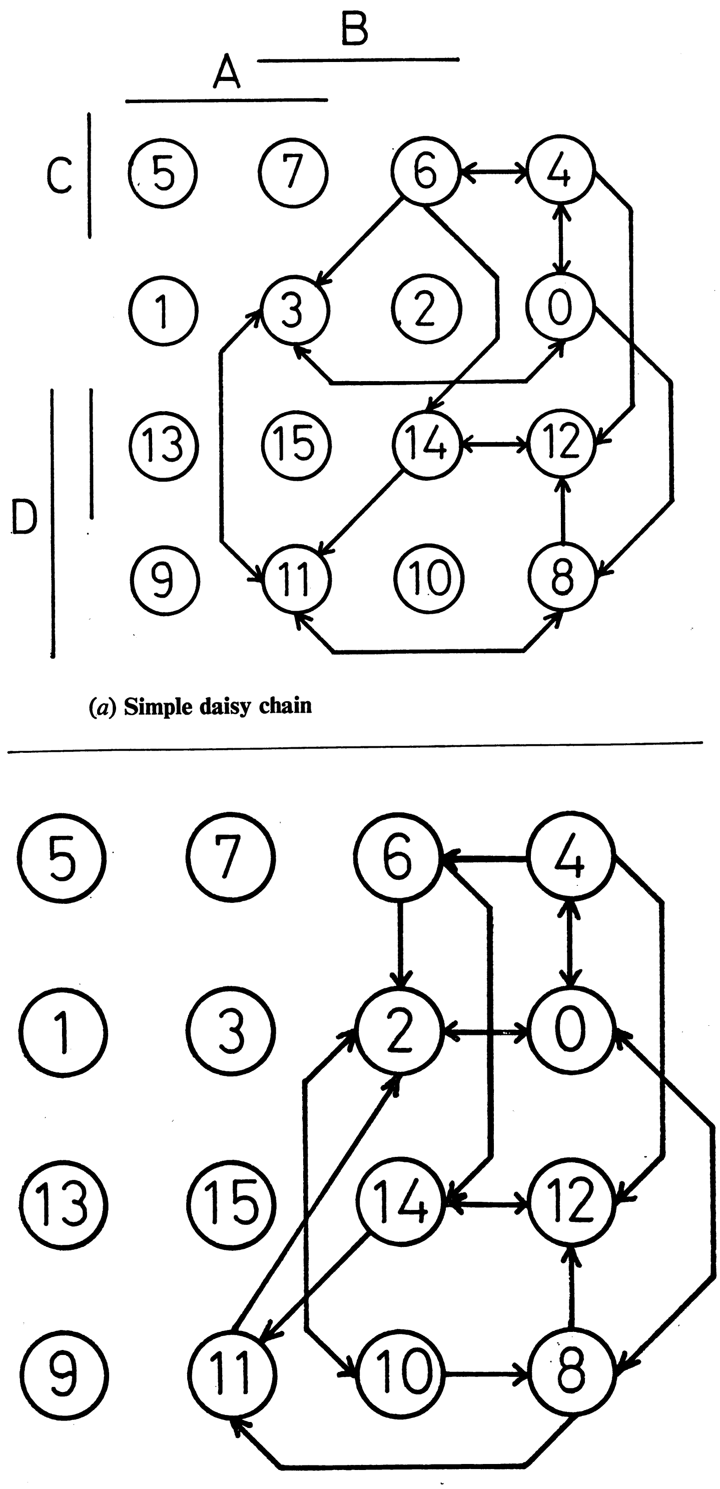
$$-D_{jk} \leqslant R_k \leqslant D_{jk} \ . \qquad (4)$$

The negative minimum time $-D_{jk}$ means that the release of device k can take place before the request of device j up to the period required to send a signal between the two devices. If R_k becomes more negative, no conflict arises as the bus is marked busy. If R_k becomes larger than D_{jk}, case 2 applies. The critical interval is therefore larger for devices which are further apart on the bus.

A pulse of 'no request' is generated at time R_k with the duration $D_{jk} - R_k$ and travels upstream. At the same time the bus available signal travels towards device j where it is received at

$$T_j = R_k + D_{jk} \geqslant 0 \ . \qquad (5)$$

As R_k may be negative T_j can become 0 in the limit which means that the bus is granted immediately. However, another



(*a*) **Simple daisy chain**

(*b*) **Improved daisy chain**

Signals: *A* **Available signal sent downstream**
B **Available signal received**
C **Bus request set**
D **Bus request set downstream**
Common States in both circuits:
0 Idle
4 Bus request
6 Bus granted
8 Bus request set downstream
11 Bus granted downstream
12 Multiple request
14 Bus granted, request set downstream

Fig. 5 Logical states and transitions in a device on the bus

device ι, which may be waiting, can seize the bus at the time

$$T_\iota = R_k + D_{kn} + DL + D_{n\iota} \qquad (6)$$

when the 'no request' signal has travelled round the loop. In this case device *j* is forced to release at time

$$F_j = R_k + D_{kn} + DL + D_{n\iota} + D_{\iota j} \qquad (7)$$
$$= R_k + DL + 2.D_{jn} - D_{jk}$$

as the signal which originally granted the bus (Equation 5) disappears now.

This uncertainty interval can be as long as the time to seize in equation (2) if we assume that $R_k = D_{jk}$ in the worst case. Without the interfering action of device ι the signal reappears again after $D_{jk} - R_k$ in accordance with Equation (2).

The conclusion foı this case is therefore that

1. The bus is granted at $R_k + D_{jk}$
2. The bus is forced to release at $R_k - D_{jk} + DL + 2D_{jn}$
3. The bus may be granted again at $DL + 2D_{jn}$

unless it has been allocated to a device ι in the meantime. The difficulty arises because the signal on the 'bus available' line is misinterpıeted.

5. Improvement of the daisy chain

It shall now be investigated how the critical case can be avoided without the introduction of guarding intervals. **Fig. 5**(*a*) shows the different states occurring in a device on the daisy chain. Only eight of the 16 possible logical states are used. The states numbered 1, 5, 7, 9, 13 and 15 are not compatible with the working of the chain and have to be discarded. It is not allowed to propagate an 'available' condition without receiving it (1, 5, 9, 13) and a request can never be placed in the presence of an 'available' signal (7, 15). However, 2 and 10 may be favourably applied for an improved solution.

The analysis showed that state 3 is potentially dangerous because it grants a signal down while no request is placed on the bus, which results in a spurious signal being sent. Fig. 5(*b*) shows how this can be avoided and **Fig. 6** gives the equivalent circuit. From the 'granted' state (6) the circuit will now go into state 2, from where it reaches either the idle state or alternatively state 10 if a request arrives. The bus is now allowed to release completely before an available signal is repeated downstream again. Under the assumptions of Equation (4), the transitions at device *k* occur as follows:

Transition	6–2 :	R_k	Device *k* releases
,,	2–10:	D_{jk}	,, *j* requests
,,	10–8 :	$R_k + DL + 2D_{kn}$	Available signal disappears
,,	8–11:	$D_{jn} + DL + D_{nk}$	Available signal now repeated.

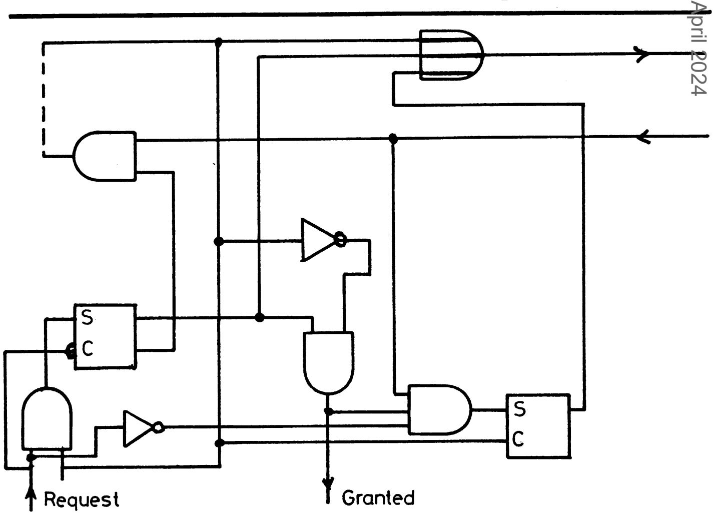


Fig. 6 Implementation of improved daisy chain

Therefore Equation (2) applies again for device j as the exact timing of R becomes irrelevant. The priority structure remains unaffected in case 2, as queuing is still allowed.

The effect of the modifications in the state diagram is the suppression of the pulse of 'bus available' signal which was generated in the original daisy chain at time R_k by forcing the circuit through the sequence 6–2–10–8–11 rather than 6–3–11, introducing the minimal delay required to achieve complete unambiguity.

6. Conclusions

Lockout mechanisms used in relay technology cannot be used in a bus system working at a high speed because the transmission time of the signals providing the lockout can no longer be neglected. The simple daisy chain caters for this problem in most cases, but it is still possible that a request is granted and later release is enforced. Detailed analysis showed that the delay is again dependent on the bus length. By modifying the procedure of release, making use of unused states of the circuit, it is possible to guarantee a complete lockout even under adverse timing conditions. The delay involved is always automatically adjusted by the length of the loop itself.

Acknowledgement

The author thanks Mr. F. Coakley from Essex University for stimulating discussions.

References

KORN, F. A. (1939). Lockout Circuits, *Bell Lab Record 18*, pp. 21-27.

JOEL, AMOS E. (1948). Relay Preference Lockout Circuits in Telephone Switching, *AIEE Transactions 1948*, Vol. 67, pp. 1720-1725.

THURBER, K. J., HENSEN, E. D., and JACK, L. A. *et al.* (1972). A systematic approach to the design of digital bussing structures, *Fall Joint Computer Conference 1972*, pp. 719-746.

Book reviews

APL/360 With Statistical Applications by K. W. Smillie, 1975; 225 pages. (*Addison-Wesley Publishing*, £4·40).

All too slowly APL is becoming at least known to the British computing community, though many potential users in other professions may not yet be aware of its potential application in their respective fields.

Now that time has elapsed for the language and the system to be seen in perspective in relation to other programming languages and to potential useful applications areas, many of the emotional attitudes previously displayed have largely disappeared. Attitudes of love or hatred for the language, of addiction to its use or its total and unevaluated rejection are slowly being replaced, in those communities where APL is available, by a more healthy attitude.

It is recognised that situations which call for the repetitive execution of stabilised programs are often unsuitable for the use of APL. On the other hand in a situation which by its nature calls for strong interaction between the user and the program during execution, in which the logic and content of the program has to be continually developed on the basis of progress of an investigation, or where the user is suitably trained to make use of a mathematical-like symbolism which facilitates the use of arrays of various ranks and sizes—then APL has little competition from existing program languages. This despite the fact that the size and nature of its operator set must be abhorrent to the purist of programming methodology because it does not encourage, though it does permit, the development of well structured, intelligible, non-pornographic programs.

Unfortunately in the UK APL is not widely or economically available to the university user (if this reviewer may grind a personal axe), though the commercial user can and has cost effectively exploited the availability of APL from several commercial vendors of the service. One would hope that statisticians, at least, should see Smillie's book and should have an opportunity to develop their knowledge and understanding of APL and to test it out sitting at a terminal. And learning it by using it they will add their voices to the already widespread clamour, to the armoury of those who wish to effectively integrate suitable computing tools into their day to day activities at universities, research and development laboratories and in other environments which call for the type of interactive usage in which APL has so widely proven itself in the USA.

Smillie is one of the pioneers of the use of APL. He first published programs in the very early days of the language availability about a decade ago. He is *the* pioneer of the use of APL by statisticians and the present paperback book clearly demonstrates his interest in the topic.

Most unfortunately the volume is based on the oldest version of the language (APL/360) which is by now outmoded by the newer developments implemented by all major manufacturers (with the exception of ICL). The newer version of the language includes file and most input/output facilities which might be considered almost essential to the statistician and other commercial users. Equally it contains a number of new and powerful operators which once they have been used make the older version look archaic. Not only are the scan and execution functions omitted by Smillie but even the domino function, the square matrix inversion operator, does not appear. Thus in a way the book hides the true power and value of the language as it now stands. Equally the set of system commands are totally incomplete in relation to any system that the potential user is likely to have access to today.

The concept of a pornographic program was previously mentioned. Professor Smillie's examples are mostly well structured, although one suspects that this is more by chance than planned. Unfortunately, however, he has totally neglected the use of labels, branching instead to actual statement numbers. One can only assume that at the time his examples were formulated the concept of labels in APL had not yet been invented. In fact his chapter on branching does not mention it. The use of actual statement numbers is one of the most certain ways to introduce errors as a program is developed and one can only conclude that the value of this book is primarily historic: a book that can be browsed through and borrowed from the library. The potential purchaser might wish to wait until Professor Smillie brings out an up-to-date second edition, and this reviewer would strongly encourage Professor Smillie to do so. APL deserves to be even more widely known and he is certainly the man to do this for statisticians.

M. M. LEHMAN (London)

A Practical Guide to Algol 68 by F. G. Pagan, 1976; 213 pages. (*John Wiley*, £7·50 cloth, £3·75 paper).

Algol 68 owes no debt to big business and has been shamefully played down by computer journalists, administrators and official bodies alike. It stands only on its merits and those of authors like Frank Pagan. Here at last is a concise, lucid and nicely printed book on the revised and final 1974 version of the language that one can unreservedly recommend to anybody. The author has performed, in a seemingly effortless way, what we know to be an extremely difficult task. His judgment on matters of emphasis and order of presentation is excellent. By stating facts clearly, avoiding fussy repetition and endowing his readers with a little gumption, he communicates the real simplicity and power of the language more clearly than ever before. Scientific programmers will find this book easy to understand, and university students should find it about right for maximum enjoyment and practical use.

P. M. WOODWARD (Malvern)