THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 4, April 1981
Printed in U.S.A.

Blocking States in Connecting Networks Made of Square Switches Arranged in Stages

By V. E. BENEŠ

(Manuscript received October 16, 1980)

Since the probability of blocking is a principal measure of the performance of a network, this paper examines the blocking states of a network. For two-sided connecting networks made of square switches arranged in stages, a parallel pair (PP) is a pair of paths through the network that meet at no switch. The blocking states of the network are closely related to those that have a busy PP. In particular, a routing algorithm can avoid all the blocking states if and only if it avoids all the states with busy PPs.

I. INTRODUCTION

Connecting networks made of rectangular switches have long been an important feature of telephone central offices, and their combinatorial properties still command substantial practical as well as theoretical interest. The probability of blocking, calculated or measured in a suitable setting, is a principal measure of the performance of a network, and it is of particular interest to identify and study those combinatorial features which give rise to blocked calls.

To be sure, blocking is known to be an unfortunate mismatch of available resources. But precisely, what structural features lead to it? What presages blocking? What are some tell-tale signs and necessary concomitants of blocking? When and how can blocking be avoided? To try to answer these questions, we focus on the blocking states of the network, those in which some call is blocked.

Although in a century of telephony much work has gone into calculations and simulations for blocking probability, there are (except for a few examples) virtually no results on the basic causes and nature of blocking. For networks made of square switches arranged in stages, this paper will partly remedy this lack by introducing some simple concepts suggested by examples; theorems based on these bear on the position and structure of the blocking states within the set (semilattice)

of states, and also on the possibility of routing rules that avoid blocking entirely.

II. PRELIMINARIES

In a network made of square switches each rectangular switch has the same number of inlets as outlets, so there is neither expansion nor concentration. The network is said to be arranged in stages if the switches are partitioned into sets called stages, two of which carry the inlets and outlets, while the others are internal stages, such that every path from an inlet to an outlet passes through each stage only once, always in the same order. The network is full access if each switch in a given stage has a link to every switch in the adjacent stages.

It has been pointed out repeatedly[1,2,6] that the set S of permitted states of a connecting network forms a partially ordered system, in fact a semilattice, in which the order relation $x \leq y$ between states x, y means that x can be reached from y by zero or more hangups: In this case we say that x is below y. The usual Hasse diagram for the partial ordering is then a convenient way of visualizing the set S of states, marred only by the prodigious number of states that arise. This "curse of dimensionality" can be alleviated by considering only equivalence classes of states under a suitable group of symmetries,[2] and we do so when convenient without further discussion.

A state $x \in S$ is maximal if there are no states above x. It can be seen that in a network made of square switches all maximal states carry the same number of calls. A state x is a blocking state (or simply, is blocking) if an inlet u and an outlet v exist, both idle in x, such that there is no available allowed path from u to v, whose use would connect u to v and give rise to a state y covering (immediately above) x in the partial ordering; the call (u, v) is then said to be blocked in x. We use the convention that, in speaking of a call (u, v), u is always an inlet, and v always an outlet. Figure 1 illustrates all the preceding concepts.

A routing algorithm or rule is a method for choosing routes for unblocked calls during the operation of the network to carry traffic. It specifies, for each state x and each call (u, v) not blocked in x, what route should be used to put up (u, v) if these terminals desired connection when the system was in state x. Needless to say, it is of interest to find and describe rules which make for good network performance as evaluated by the probability that an arriving call is blocked, i.e., the fraction of requests for service which cannot be completed because of blocking. Since such denials of service occur only when the network is in a blocking state, it is natural that a good routing strategy will seek to avoid the blocking states. Examples are known in which the blocking states can be avoided entirely; such networks are called nonblocking in the wide sense.[1,3]

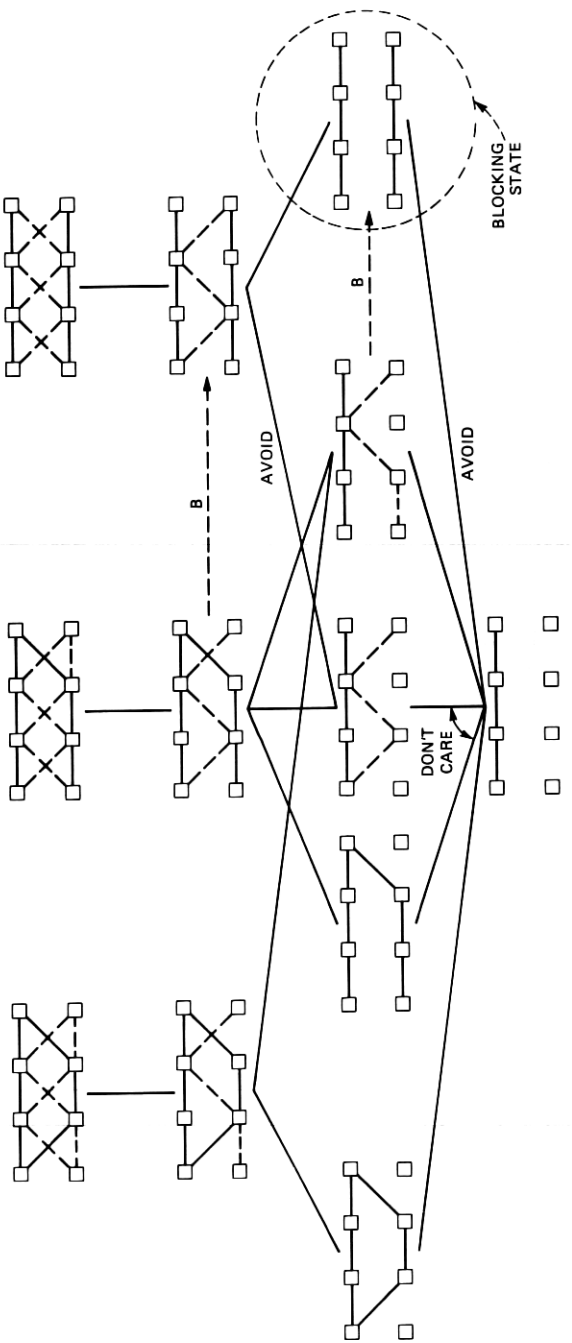

Fig. 1—Reduced state diagram.

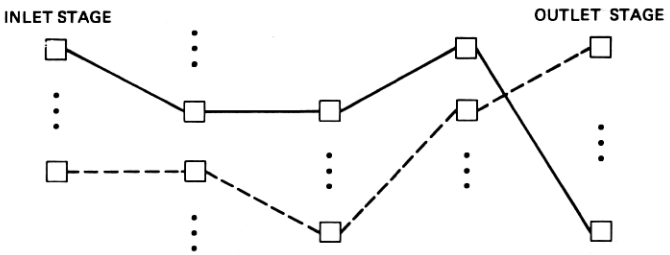

Fig. 2—A parallel pair (PP).

III. SUMMARY OF RESULTS

We assume as given a two-sided connecting network made of square switches arranged in stages, and we introduce the idea of a parallel pair (PP). This is a pair of paths through the network that meet at no switch (Figs. 2, 3). The basic message of this paper is that visiting states with busy PPs is bad for performance, because the property of being a blocking state is closely tied up with that of having a busy PP. Now having a blocked call and having a busy PP are usually not equivalent properties of states; the characterization of blocking is not that simple. There are nonblocking states which have a busy PP, and there are blocking states without a busy PP. Nevertheless, our intuitive conclusion is strongly supported by small examples,[3,4] and here by the following results:

(*i*) Every maximal state that is above a blocking state has a busy PP.
(*ii*) Every blocking state is below a maximal state with a busy PP.
(*iii*) A routing algorithm makes the network nonblocking in the wide sense if and only if it makes inaccessible all states with busy PPs.

It can be seen that in a state without any busy PPs, every path in use must meet every other at a switch. To maintain this property in an operating system, one must keep from adding paths that pass only through empty switches. Thus, the advice to avoid busy PPs almost coincides with (or at least, is very consistent with) the "packing" principle that to avoid blocking one should route calls in the most

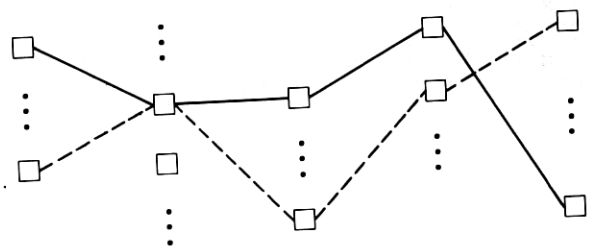

Fig. 3—Paths that meet at the second stage.

heavily loaded part of the network that will accept them: Clearly, in putting up a new call you best avoid busy PPs by having the route you choose meet as many calls in progress as possible. Our result says that completely avoiding blocking and never having any busy PPs are both possible, or both impossible, to achieve by a choice of routing rule.

IV. SOME EASY RESULTS

To fix ideas and record a few facts, we first list various results with straightforward proofs. The reader interested in warming up to the concepts in use here might go through these in some detail using Figs. 1 to 5.

(1) *Remark:* No state immediately below a maximal state is blocking.

(2) *Remark:* A blocking state two levels down from a maximal state (i.e., obtainable from a maximal state by two hangups) consists of an idle PP with all other terminals and links busy.

(3) *Lemma:* If x is a blocking state, then some PP is idle in x.

Proof: Let (u, v) be blocked in x. Since the switches are square, there is an idle path from u to the other side, and one from v to u's side of the network. These paths must avoid each other if (u, v) is blocked. Thus, they constitute a PP idle in x.

(4) *Remark:* If x has a busy PP, then states y and z exist such that y is maximal, z is blocking, $y \geq x$, and $y \geq z$.

Proof: Starting in state x, put calls in until a maximal y is reached, then hang up the PP to form z, and reverse its assignment, i.e., if the busy PP provided the two calls (u_1, v_1) and (u_2, v_2), then the calls (u_1, v_2) and (u_2, v_1) are both blocked in z. Compare Remark (2).

(5) *Remark:* If ν is full access and x is blocking, then all inner switches are nonempty in x.

Proof: An empty switch is accessible from any switch in the adjacent stages, hence from any idle inlet, and also from any idle outlet. Thus, no call is blocked. See Fig. 4.

(6) *Remark:* x nonblocking $\not\Rightarrow$ x has no busy PPs.

Proof: Figure 4 provides a counterexample.

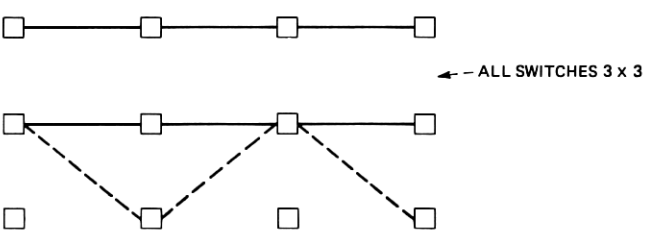

Fig. 4—State with two busy PPs but no blocking.

(7) *Remark:* x blocking $\not\Rightarrow$ x has a busy PP.

Proof: Figure 5 gives a counterexample.

V. BLOCKING STATES ARE BELOW STATES WITH BUSY PPS

(8) *Theorem: A maximal state with no busy PPs has no blocking states under it. Every maximal state that is above a blocking state has a busy PP.*

Proof: We embroider the argument in (3). Suppose that x is a blocking state, and $y \geq x$ is maximal. Let (u, v) be an inlet-outlet pair blocked in x; u and v are on disjoint paths of a PP idle in x. Moreover, since $y \geq x$, it is possible to start with x and put in all the additional calls of y, by the same routes as they use in y, except those involving the terminal pair (u, v). (This is *not* to say that a given routing algorithm would prescribe those routes in operation!) Since (u, v) is blocked in x, u and v are not talking to each other in y. Indeed, in y, u is connected to some switch other than v's, and v to some switch other than u's. If the paths carrying these calls in y met at some switch, (u, v) would not have been blocked in x. Hence, in the state formed from x by putting in all the calls of y except those involving u and v, these paths again form an idle PP with u on one path of the PP, and v on the other. Thus, y necessarily has a busy PP.

(9) *Remark:* We have already seen in (4) that a maximal state with a busy PP is above some blocking state. Thus, for maximal states y,

y has a busy PP iff y lies above a blocking state.

(10) *Corollary: Any blocking state lies below a maximal state which has a busy PP.*

Proof: Every state x lies below some maximal y. If y had no busy PPs, no blocking state could be below it, by (8).

(11) *Corollary: If x is below some maximal state which has no busy PPs, then x is nonblocking.*

Proof: (8) or (9); the property is sufficient, but not necessary.

VI. THE CLOSURE TOPOLOGIES INDUCED BY $\leq$

A partial ordering $\leq$ induces a topology in a familiar way. If $X \subseteq S$ is a subset of states, its lower closure, denoted X_{lc}, is the set of states

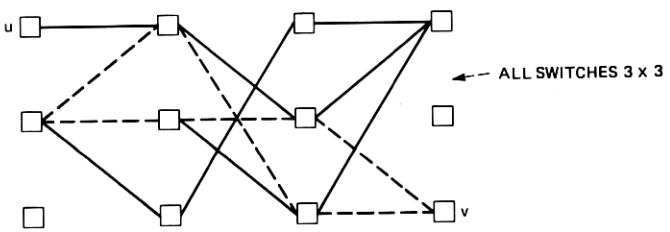

Fig. 5—Blocking state with no busy PPs: (u, v) is blocked.

that are below some member of X in the sense of $\leq$, including, of course, the elements of X as well. Lower closure satisfies the Kuratowski axioms for a topological closure operation. There is a dual topology of upper closure, wherein the dual $\geq$ replaces $\leq$, denoted by X^{uc}.

By introducing notation for some of the important subsets of states we have encountered so far, we can use lower closure to express various preceding results in a concise way. Let us define

$$\begin{aligned} \text{Max} &= \text{set of states that are maximal in } \leq, \\ B &= \text{set of blocking states}, \\ S - B &= \text{set of nonblocking states}, \\ bPP &= \text{set of states that have a busy parallel pair.} \end{aligned}$$

(12) *Remark*: $(bPP)_{lc} = (\text{Max} \cap bPP)_{lc}$.

Proof: If $x \in (bPP)_{lc}$, then there exists y with a busy PP such that $x \leq y$. Pick a maximal $z \geq y$. Then $z \in bPP$, since bPP is closed above. Thus, we have $z \geq x$ and $z \in \text{Max} \cap bPP$, and so also $x \in (\text{Max} \cap bPP)_{lc}$. The converse is trivial.

In these notations, Theorem (8) may be cast as saying that

(13) $B \cap (\text{Max} \cap (S - bPP)_{lc} = \emptyset = \text{empty set}$,

while Corollary (10) claims that

$$B \subseteq (\text{Max} \cap bPP)_{lc},$$

which by (12) simplifies to $B \subseteq (bPP)_{lc}$. Similarly Corollary (11) can be rendered as

$$[\text{Max} \cap (S - bPP)]_{lc} \subseteq S - B.$$

VII. WIDE-SENSE NONBLOCKING NETWORKS

A telephone network is called nonblocking in the wide sense (NBWS) if a rule exists for assigning routes to incoming calls that precludes all blocking, that is, under the rule no blocking state is accessible. Run according to the rule, the system can satisfy all demands for connection as they arise without ever rejecting any, or rerouting existing calls. The point is to try to reduce network complexity (from the extreme of strict nonblocking) by using a network that has blocking states, but to avoid them by clever routing.

Two necessary and sufficient conditions are known for a general connecting network to be NBWS: One is phrased in terms of denseness in a simple metric on the states,[5] the other in terms of a semilattice homomorphism of S onto the set of assignments realized by the states in S.[6] For networks made of square switches arranged in stages, we give a new necessary and sufficient condition, based, perhaps not surprisingly now, on avoiding states with busy PPs.

If α is a routing algorithm, we use $Ac(\alpha)$ to denote the set of states that are accessible under α, i.e., that can be reached by the operating

network if it is run using α to make routing choices. A good routing algorithm derives its worth from making bad states unlikely or inaccessible, so it is important to know what states can be reached. Further, it is clear than an NBWS network is one for which an α can be found such that $Ac(\alpha) \cap B$ is empty, i.e., no blocking state can be reached by operating under α. We note these properties of $Ac(\alpha)$:

(14) $Ac(\alpha)$ is closed below: $[Ac(\alpha)]_{lc} = Ac(\alpha)$,

(15) $x \in Ac(\alpha) \Rightarrow Ac(\alpha) \cap \text{Max} \cap \{x\}^{uc} \neq \emptyset$.

Property (15) says that if x is accessible, then some maximal state above x is accessible. For networks made of square switches, however, a specialized sharp form of (15) holds, in which both hypothesis and conclusion are stronger, and which relates blocking to having a busy PP. To be specific, we shall use a precise version of the following heuristic idea: An algorithm α must complete an unblocked call by *some* route; so starting in a state x that has a blocked call (u, v), it should be possible to keep picking successive unblocked new calls not involving u or v, and routing them by α, until as many terminals are busy as possible, and the system has arrived at a state z accessible from x under α by new calls, with (u, v) idle and blocked in z; we claim in particular that the calls can be picked so that z consists of an idle PP which has u and v on switches that terminate its paths, and all other terminals are busy. Compare Remark (2).

(16) *Lemma: If (u, v) is a call blocked in state x, then an inlet s and an outlet t exist, with $s \neq u$ and $t \neq v$ such that* EITHER

(i) s, t, u, and v are the only terminals idle in x, and there is a PP *idle in x one of whose paths joins s to v, the other u to t,* OR

(ii) an inlet q and an outlet r exist, neither of which coincides with any of s, t, u, or v, such that q and r are both idle in x and (q, r) is not blocked.

Proof: We recall our convention that in a call (u, v), u is always an inlet and v an outlet. Since (u, v) is blocked, there is a PP idle in x one of whose paths connects u's switch to some switch other than v's, which, since the switches are square, has an idle outlet t on it. Similarly, the other path of the PP connects v's switch to some switch other than u's on which there is an idle inlet s. Suppose that alternative (i) fails, so that some inlet q, distinct from each of s and u, is idle in x. Now since switches are square, q can reach an idle outlet r; we shall show that r can be chosen to be distinct from both t and v. There are three cases.

Case (i): u and q are on the same switch. Since u can reach t, q must be able to reach some r other than t. This is because if we put up (u, t) in x, q could still reach some idle outlet, so *a fortiori* it can do so in x. Since (u, v) is blocked, so is (q, v). But (q, r) is not blocked, so $r \neq v$. Thus, r is distinct from both t and v, though it may be on the same switch as t. So alternative (ii) holds in Case (i).

Case (*ii*): s and q are on the same switch. Then q can reach v's switch and possibly t's. If it can reach v's switch *only*, then an idle outlet r must exist on v's switch, $r \neq v$. This is because if an (s, v) call were put up, q could still reach some idle outlet, which must be on v's switch, since adding a call does not make additional outlet switches reachable by q. Thus, $r \neq t$ and $r \neq v$, (q, r) is not blocked, and alternative (*ii*) holds. Suppose then that q can reach t's switch as well as v's. There are two subcases: (*a*) t's switch has another idle outlet r distinct from t, or (*b*) t is the only idle terminal on t's switch. In case (*a*), $r \neq v$, $r \neq t$, and (q, r) is not blocked, so we are done. In case (*b*), since both u and s can reach t, the final arc in the paths from u to t and s to t must be the same. Thus, these paths meet at an inner switch, and indeed on the way through the stages from inlets to outlets they meet for the last time at an inner switch σ. Thus, σ has at least two idle terminals on each side, and there is an idle path from σ to an outlet switch other than t's, which is also reachable by u, because the paths meet. This outlet switch is therefore distinct from v's because (u, v) is blocked; it has an idle outlet r on it distinct from each of t and v, and (q, r) is not blocked. Thus, again alternative (*ii*) holds.

Case (*iii*): q is on neither u's switch nor s's. If q can reach any switch other than t's or v's, we are done; for then there is an idle outlet r, distinct from each of t and v, and (q, r) is not blocked. The same is true if q can reach only t's or v's switch provided the one it can reach has more than one idle outlet. Suppose then that q can reach only t's switch or v's, and that whichever ones of these it can reach have exactly one idle outlet, viz, t or v as the case may be.

If q can reach t but not v, then (q, v) is blocked and moreover there is an inner switch at which q's idle path to t meets one from u to t for the last time (indeed they coincide henceforth), and there is a path, subsequently disjoint from the remaining common path to t, from that inner switch to some outlet switch other than t's or v's, because (q, v) is blocked. On that outlet switch is the r we seek for alternative (*ii*). If q can reach v but not t, the same conclusion follows, with s playing the role of u, *mutatis mutandis*.

If q can reach both t and v, we argue similarly, as follows: Since t's switch has exactly one idle link, an idle path from u to t and one from q to t must have that last idle link in common, there being no other way to reach t's switch. Hence, these paths must meet at an inner switch at least once. As before there is then a last, in the ordering of stages from the inlet side to the outlet side, switch at which these paths meet, and they coincide thereafter. Since the paths are meeting, the switch has at least two idle links incoming from the inlet side, and at least two idle links outgoing on the outlet side. One of the latter is a part of the now common part of the paths on their remaining legs to t's switch. The other idle link on the outlet side is part of an idle path

to the outlets; it does not meet the common part again, and so cannot end at t's switch. If it reached v's, (u, v) would not be blocked. The outlet switch it reaches then has an idle outlet r on it, distinct from each of t and v, and (q, r) is not blocked; hence alternative (*ii*).

(17) *Lemma: If x is a blocking state accessible under a routing algorithm α, then some maximal state $y \geq x$ is accessible under α, and y has a busy* PP. *In symbols,*

$$x \in B \cap Ac(\alpha) \rightarrow \exists y\, y \geq x, \qquad y \in \mathrm{Max} \cap Ac(\alpha) \cap bPP.$$

Proof: Let (u, v) be a call blocked in an accessible state x. If possible, use alternative (*ii*) of Lemma (16) to find a call (q, r), unblocked in x, neither of whose terminals is on an idle PP with u and v. Putting up (q, r) according to α results in a state in which (u, v) is blocked. Add calls in this way repeatedly until no such (q, r) can be found, i.e., alternative (*ii*) fails. Then alternative (*i*) holds: The state z we have reached consists of an idle PP with one of u, v on each of its distinct paths. It is obvious that there is one maximal y above z, reachable under α by having the right two calls request connection, and that y has a busy PP.

(18) *Theorem: Let α be a routing algorithm. Some blocking state is accessible under α iff some state with a busy* PP *is accessible under α; in symbols,*

$$Ac(\alpha) \cap B \neq \emptyset \quad \text{iff} \quad Ac(\alpha) \cap bPP \neq \emptyset.$$

Proof: Suppose $x \in Ac(\alpha) \cap bPP$. By (15) there exists $y \geq x$ such that y is maximal and $y \in Ac(\alpha)$. Since $y \geq x$ and bPP is upper closed, we see that y has a busy PP. By (9) there is a $z \in B$ such that $z \leq y$ and $z \in Ac(\alpha)$, since $Ac(\alpha)$ is closed below. Hence, $Ac(\alpha) \cap B \neq \emptyset$. The converse follows from (17).

We illustrate two aspects of Theorem (18) by reference to Figs. 4 and 5. According to the converse part of (18), if an algorithm α made the state in Fig. 4 reachable, then there must be a continuation (of the events that led to this state) leading to a blocking state. In Fig. 4 this happens if there is, first, a call between the two highest outer switches, and then, a call between the two second-highest outer switches. There are two ways of satisfying this extended assignment, and both lead to blocking. The direct part of (18) is exemplified in Fig. 5. If an algorithm α makes the state in Fig. 5 reachable, then according to (18) a state with a busy PP is also reachable. It can be verified that in Fig. 5 putting up any new call leads at once to a state with a busy PP.

(19) *Corollary: A routing algorithm α makes the network nonblocking in the wide sense if and only if no state with a busy* PP *is accessible under α.*

An example in which it actually happens that a natural routing rule makes all blocking states unreachable was shown in Fig. 1. The relation

B shown in Fig. 1 by arrows is to be interpreted as "better than." Any algorithm for routing which uses the "better" alternatives avoids all blocking and all busy PPs. Note that there are actually two such algorithms, as suggested by the "don't care" label.

REFERENCES

1. V. E. Beneš, *Mathematical Theory of Connecting Networks and Telephone Traffic*, New York: Academic Press, 1965, p. 62 and in references therein.
2. V. E. Beneš, "Reduction of Network States Under Symmetries," B.S.T.J., *57,* No. 1 (January 1978), pp. 111–49.
3. V. E. Beneš and R. P. Kurshan, "Wide-Sense Nonblocking Network Made of Square Switches," unpublished manuscript.
4. V. E. Beneš, "Programming and Control Problems Arising From Optimal Routing in Telephone Network," B.S.T.J., *45,* No. 9 (November 1966), pp. 1373–438.
5. V. E. Beneš, "Algebraic and Topological Properties of Connecting Networks," B.S.T.J., *41,* No. 4 (July 1962), pp. 1249–74.
6. V. E. Beneš, "Semilattice Characterization of Nonblocking Networks," B.S.T.J., *52,* No. 5 (May–June 1973), pp. 697–706.